No. 16

Proceedings of the

EASTERN JOINT COMPUTER CONFERENCE

December 1-3, 1959 Boston, Massachusetts

Sponsors:

THE INSTITUTE OF RADIO ENGINEERS
Professional Group on Electronic Computers

THE AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
Committee on Computing Devices

THE ASSOCIATION FOR COMPUTING MACHINERY

Printed in the United States of America

Price $3.00

ADDITIONAL COPIES

Additional copies may be purchased from the following sponsoring societies at $3.00 per copy. Checks should be made payable to any one of the following societies:

INSTITUTE OF RADIO ENGINEERS
1 East 79th Street, New York 21, N. Y.

AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
33 West 39th Street, New York 18, N. Y.

ASSOCIATION FOR COMPUTING MACHINERY
2 East 63rd Street, New York 21, N. Y.

COPIES OF PRIOR PUBLICATIONS

Copies of publications issued in connection with the prior Joint Computer Conferences listed below may also be purchased from the above societies.

NUMBER	CONFERENCE	LOCATION	DATE
1	Eastern	Philadelphia	Dec. 10–12, 1951
2	Eastern	New York City	Dec. 10–12, 1952
3	Western	Los Angeles	Feb. 4 – 6, 1953
4	Eastern	Washington	Dec. 8–10, 1953
5	Western	Los Angeles	Feb. 11–12, 1954
6	Eastern	Philadelphia	Dec. 8–10, 1954
7	Western	Los Angeles	Mar. 1 – 3, 1955
8	Eastern	Boston	Nov. 7 – 9, 1955
9	Western	San Francisco	Feb. 7 – 9, 1956
10	Eastern	New York City	Dec. 10–12, 1956
11	Western	Los Angeles	Feb. 26–28, 1957
12	Eastern	Washington	Dec. 9–13, 1957
13	Western	Los Angeles	May 6 – 8, 1958
14	Eastern	Philadelphia	Dec. 3 – 5, 1958
15	Western	San Francisco	Mar. 3 – 5, 1959

No. 16

PROCEEDINGS OF THE EASTERN JOINT COMPUTER CONFERENCE

PAPERS PRESENTED AT
THE JOINT IRE-AIEE-ACM COMPUTER CONFERENCE
BOSTON, MASSACHUSETTS, DECEMBER 1-3, 1959

Sponsors

THE INSTITUTE OF RADIO ENGINEERS
Professional Group on Electronic Computers

THE AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
Committee on Computing Devices

THE ASSOCIATION FOR COMPUTING MACHINERY

Published by
THE 1959 EASTERN JOINT COMPUTER CONFERENCE
FOR
THE NATIONAL JOINT COMPUTER COMMITTEE

NATIONAL JOINT COMPUTER COMMITTEE

TABLE OF CONTENTS

EASTERN JOINT COMPUTER CONFERENCE COMMITTEE

Foreword

For some time it has been customary to hear complaints about the limited value of large technical conferences. Despite this fact, I represent a group of people who have worked assiduously to arrange this affair. And over 2,300 people have expended considerable effort to attend. It is interesting to inquire seriously as to the reason for so much effort. There are, of course, a number of very cynical answers. However, I would like to offer a less cynical one.

Perhaps it is only a convenient rationalization, but I still find the computer field an exciting and stimulating domain. The excitement about the computer arises in much the same way as the excitement about atomic energy; one may almost *feel* the changes being produced in society. For an applied scientist or engineer, it is usually the applications which lend to a discipline an aura of excitement. Well, then, I believe that many of us are here because of a continued enthusiasm in the possibilities of the computer. The number and importance of the potential applications are still increasing more rapidly than the onset of general boredom.

In the tiny span of years from the first to the ninth EJCC, we have been witness to a wholesale change in the techniques of scientific computation, witness to a revolution in business data handling, and witness to the use of computers for real time control of weapons systems, industrial plants and space vehicles. Surely these events are exciting enough to partially justify our large conferences.

And yet I believe that the most important applications of the computer have not yet been realized. Certainly computer inroads in the business world and the industrial plant have only just begun. However, for me, the most exciting applications are those which threaten to affect all aspects of human progress. I would like to point toward two such potentially pervasive applications — two impending applications that excite me considerably.

The first is the application of the computer in studying and copying the characteristics of biological systems. This is a doubly potent use of a computer, involving useful feedback, because real gains in understanding biological systems might lead to better computer systems. The first steps in this direction have already been taken. Computers are being used for analysis of electroencephalograph data and will be used to study many other types of clinical data. Computers have been used to permit construction and study of models of neuron assemblages. A whole gamut of pattern recognition techniques is undergoing intensive investigation. People are trying to learn about learning. (Actually, even if we don't get very far, we will have the harmless fun of constructing more and better maze-solving programs and chess-playing programs while trying.)

The second application may be characterized as the library problem. I think that the proper way to measure the importance of this application is to think of it as a new way for people to tap the accumulated knowledge of the recent and distant past. The printing press was one such new way to tap the experience of the past, but now there are difficulties. The large number of printed books and journals, the existence of important scientific communities separated by language barriers and the inadequacies of our present retrieval techniques have seriously restricted our ability to connect pertinent information to pertinent researchers.

The pace of scientific progress might well take a large jump if, upon receiving a new project, a researcher might receive a graded synthesis of all human experience on that subject from the local library computer. Similarly, a lawyer, faced with a new case, would surely like to receive a relevance-ordered listing of all applicable court experience, and a doctor might be willing to trade clinical data and careful reporting in return for ordered estimates of diagnosis. In its present form, the technical journal itself may be facing its last few decades. The library computer concept is quite powerful, and it may some day be expedient for an author to send a new technical paper only to the library, without the continued expenditure of quite so much paper.

So, I don't think the excitement is dying out; I think it is increasing, and I expect that computer conferences will be of interest and value for some time to come.

FRANK E. HEART
Conference Chairman

LIST OF EXHIBITORS

Aeronutronic Division, Ford Motor Company
AMP, Inc.
Ampex Corporation
ANelex Corporation
Autonetics Division, North American Aviation, Inc.
Bel Air Industries, Inc.
Bendix Computer Division
Benson-Lehner Corporation
Bryant Computer Products Division
Burroughs Corporation, ElectroData Division
C-E-I-R, Inc.
C & K Components, Inc.
C. P. Clare & Company
Computer Control Company, Inc.
Di/An Controls, Inc.
Digital Equipment Corporation
Digitronics Corporation
Elco Corporation
Electro-Measurements, Inc.
Electronic Associates, Inc.
Engineered Electronics Company
Fairchild Publications, Inc.
Fairchild Semiconductor Corporation
Ferranti Electric, Inc.
General Ceramics Corporation
General Electric Company, Light Military Electronics Department
G P S Instrument Company
Harford Metal Products, Inc.
Harvey-Wells Electronics, Inc.
Instrument Specialties Company, Inc.
Intellectronics Laboratories, Ramo Wooldridge Division, Thompson Ramo Wooldridge, Inc.
International Business Machines Corporation
Laboratory for Electronics, Inc.
Librascope, Inc.
Micro Switch Division, Minneapolis-Honeywell Regulator Company
Minneapolis-Honeywell Regulator Company, DATAmatic Division
Minnesota Mining and Manufacturing Company
The National Cash Register Company
Packard Bell Computer Corporation
George A. Philbrick Researches, Inc.
Philco Corporation, Government and Industrial Division
Potter Instrument Company, Inc.
Radio Corporation of America
Reeves Soundcraft Corporation
Remington Rand Univac Division, Sperry Rand Corporation
Rese Engineering, Inc.
Royal McBee Corporation
Sprague Electric Company
Stromberg-Carlson — San Diego
Sylvania Electronic Systems Division, Sylvania Electric Products, Inc.
Tally Register Corporation
Telemeter Magnetics, Inc.
Teletype Corporation
F. D. Thompson Publications, Inc.
Union Switch & Signal Division, Westinghouse Air Brake Company
Wang Laboratories, Inc.
Washington Aluminum Co., Inc.
John Wiley & Sons, Inc.

Award for the Best Presentation of a Technical Paper

In recognition of the fact that technical programs are sometimes marred by careless or obtuse presentation of papers, the Eastern Joint Computer Conference Committee decided to emphasize the importance of a good oral presentation by making an award of $300 for the best presentation at the Conference of a paper describing significant work in the computer field.

Awarded to

DR. HAROLD K. SKRAMSTAD

National Bureau of Standards
Washington, D.C.

for his presentation of a paper entitled:

"A Combined Analog-Digital Differential Analyzer"

An analog-digital differential analyzer has been designed which combines the analog advantages of high speed and continuous representation of variables with the digital capability of high precision and dynamic range. It is based on representing dependent variables by two quantities, a digital number representing the more significant part and an electrical voltage representing the less significant part. As in the electronic analog computer, time is the independent variable.

The design of components required to build a computer of this combined type, such as integrators and multipliers, are given, and examples of how the solution of a few elementary differential equations would be carried out are presented.

Dr. Harold K. Skramstad was born in Tacoma, Washington, in 1908. He received a B.S. degree from the College of Puget Sound and a Ph.D. in physics from the University of Washington. He has been with the National Bureau of Standards since 1935 and is presently Assistant Chief for Systems, Data Processing Systems Division.

He worked in the field of aerodynamics until World War II, when he turned his efforts to guided missiles. He was a pioneer in this field, playing a key role in the development of the "BAT" missile. He was responsible for the development of one of the first flight simulator facilities.

He is an Associate Fellow of the Institute of Aeronautical Science and a Senior Member of the Institute of Radio Engineers. He also served as a member of the Air Force "Advisory Board on Simulation," and he was first chairman of the Eastern Simulation Council.

Computers of the Future

REX RICE†

Introduction

THIS PAPER considers the advances required in many related technologies to revolutionize the construction and use of digital data processing systems. In the following discussion we are particularly concerned with the radical change in fabrication technology and wish to analyze the effect that this change will have on our methods of computer design and specification.

Present Methods

The manufacturing techniques used in the electronic portion of today's digital data processing systems are illustrated in Fig. 1. The active devices are

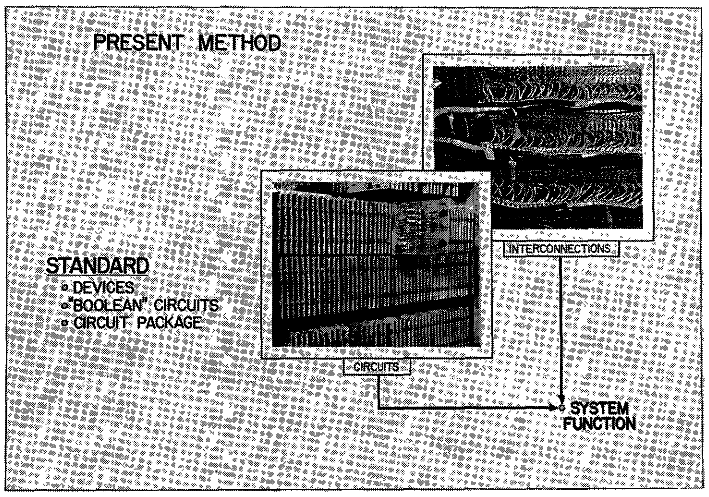


Fig. 1—Present method.

standardized in these systems. Circuit standardization is established at what may be defined as the Boolean function level. Circuits for AND, OR, Invert, Latch, Trigger, etc., are standardized individually. The pluggable packaging usually combines several circuits, either of the same type or in selected groups. A major system function such as a complete working storage register and all its controls, an arithmetic processing unit and its controls, etc., is obtained by assembling a group of circuit packages on a panel and interconnecting the circuit packages with individual wires. At the time the individual circuits and packages are designed and optimized, very little information is available regarding their specific employment in systems functions.

A digital "system function" may be defined as a combination of logical elements interconnected and timed to perform major operational sequences in a data processor. One of our future objectives is to create major digital system functions in one continuous, automated manufacturing sequence.

Future Methods

A possible future method for producing major system functions such as complete working storage registers, process units, memory arrays, etc., is illustrated in Fig. 2. We envision this manufacturing line as a set of printing presses through which a conveyor system passes. Substrate material is placed on the conveyor and proceeds through the line. At each stage one pattern of interconnections, insulation, or active material is printed on the substrate. As required, bake ovens, etc., may be strategically placed. Here, devices are standard by virtue of the materials used. These materials are applied by a standardized method to produce active elements, interconnections, insulation, etc., in batches. The plates, inserted in each press, are made in an automatic machine which develops the appropriate layout under equation control for major system segments.

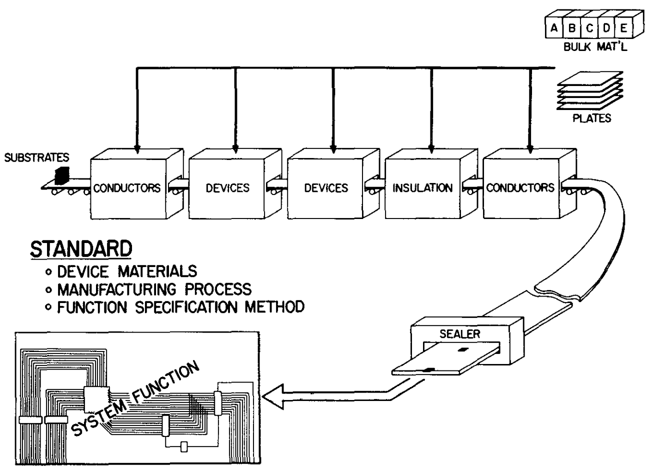


Fig. 2—A future method.

The figure illustrating future methods is only diagrammatic. The manufacturing method chosen will probably depend on the basic component technology and may be different for each type of component. Before complete automation is realized it will be necessary to manufacture active elements separately and to rely on automatic testing and insertion. The field will be dynamic and the illustration indicates a trend, not a specific technique.

†IBM Research Laboratory, Poughkeepsie, N. Y.

Illustrative Example of a System Function

A serial-by-digit, decimal adder is used to illustrate a system function as shown in Fig. 3. This represents a portion of an arithmetic processing unit. The digital code assumed is a decimal "one out of ten" representation, chosen because decimal matrix addition is well understood. Other examples or codes would have served equally well.

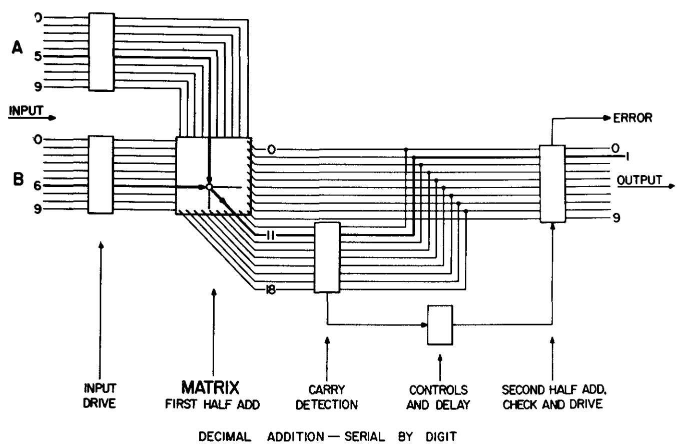


Fig. 3—Illustrative example of a system function.

In this function a pair of decimal digits enters a process unit at A and B and the added result is obtained at the output. A matrix, to be described in detail, performs the first half-addition. Other elements provide input drive, output carry detection, recombination, and the second half addition. It is also necessary to store the presence or absence of a carry, so that as succeeding pairs of digits are processed, the second half-addition circuit may be activated. Let it be assumed by way of example that A equals 5 and B equals 6, as emphasized with heavy marked lines. In the matrix the 5 on the vertical axis together with a 6 on the horizontal axis activates an AND circuit which places an output on the eleventh diagonal. After passing through the carry detection element, the eleventh diagonal is recombined with the output line 1. The carry condition is remembered for later use. Let us now consider circuits for the matrix in more detail.

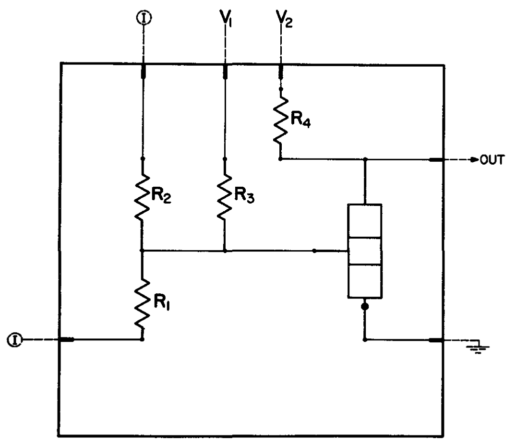


Fig. 4—Standard "and-inverter" circuit (TRL).

Matrix Utilizing Individual, Standardized Boolean Circuits

The circuit in Fig. 4 is a Boolean standardized two-way AND circuit with one transistor, four resistors, and various internal interconnections. Several outputs may be wired together to form an appropriate OR circuit. A two-way circuit is chosen, since for our purposes in the addition matrix a three- or four-way AND circuit has no advantage.

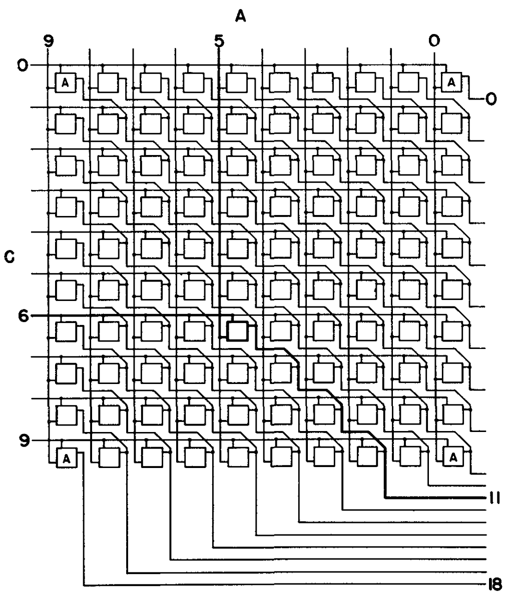


Fig. 5—Matrix utilizing standardized "Boolean" circuits.

A ten by ten matrix of these AND circuits is illustrated in Fig. 5. For clarity, the internal circuit connections and devices have been omitted. In the matrix, addition is accomplished by the coincidence of current on any pair of lines such as $A = 5$ and $B = 6$. When the AND circuit at this intersection is active, its output is placed on the eleventh diagonal. For packaging purposes the designer has the choice of packaging several AND circuits on a single pluggable unit. When the circuits were optimized, only the two-way AND logic together with the output loading conditions were known.

Let us now reexamine this same matrix from a system rather than a circuit viewpoint (Fig. 6). In this specific matrix element only one AND circuit in the $A = 5$ column and the $B = 6$ row is "on." This is a system consideration and was not known at the time the Boolean AND circuit was optimized. The vertical column $A = 5$ will now be considered as a single element.

System-Tailored Circuits

A circuit which is tailored to this system function

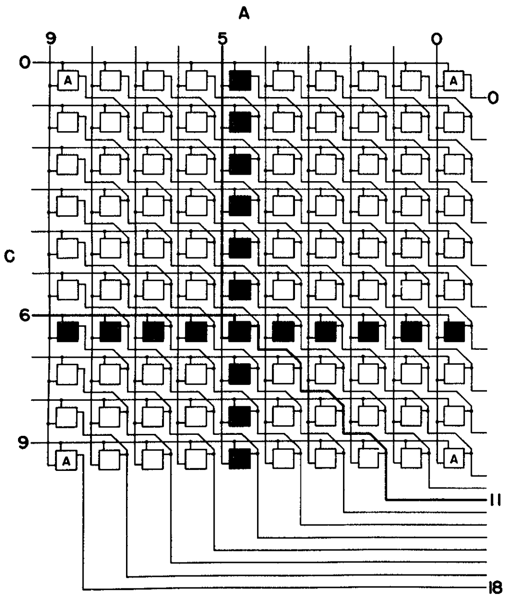


Fig. 6—Matrix utilizing standardized "Boolean" circuits.

is illustrated in Fig. 7. For convenience, transistors have been shown, although other devices such as relays, tubes, cryogenic devices, etc., could have been used. The input *A* supplies current to a common control which goes to all the bases of the ten transistors. Since only one line on the *B* input to the emitters is active at any instant, only one transistor will be conducting. Let us now examine the addition matrix utilizing this "system tailored" circuit.

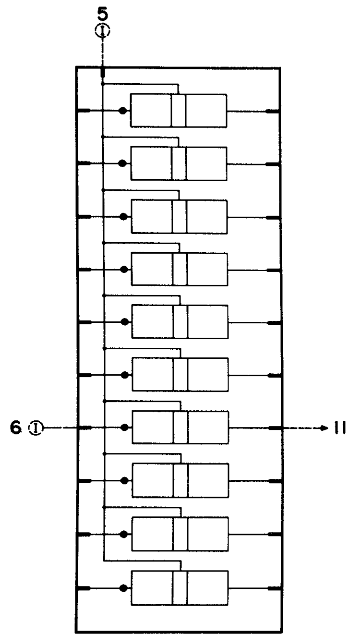


Fig. 7—System tailored circuit (CS).

Matrix Utilizing System-Function Circuits

The complete matrix is again shown in Fig. 8, this time utilizing ten of the system-function circuits. The "*A*" entries on the vertical axis go directly to the common control connections of the ten AND circuits. The "*B*" entries are connected to the emitters of the ten transistors in each of the ten circuits. The collectors are connected to the output lines, which are functionally equivalent to diagonals in the previous matrix. Note the identical configuration of the wiring to the inputs of all ten matrix columns. The outputs of each "system AND" circuit are connected in a pattern which drops down to the next output line for each successive group. Thus, to add 5 to the number entering *B* the sixth AND circuit is activated. The number 6 on the *B* entry is moved down five units on the output, giving a sum of 11. Although the number of transistors required in both matrix examples remains the same, the passive elements are eliminated and the packaging pattern for both interconnections and devices is drastically improved.

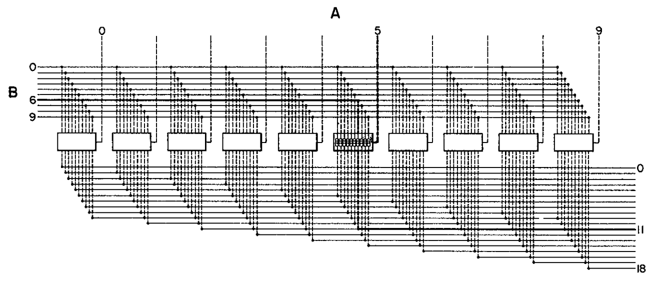


Fig. 8—Matrix utilizing "system function" circuits.

In the illustration the solid lines represent a layer of interconnections on the front of a printed substrate and the broken lines, a second layer on the rear. Connections through the substrate are indicated by dots. Inasmuch as ten system-function circuits are used, ten component packages consisting of active elements only may be mounted on a single substrate that contains the complete interconnection wiring.

A computer may be described as "a bunch of wires connected by active elements." This second method of matrix design underscores that definition. Three important features become apparent in this example. First, careful attention to system-function circuits will lead to logical layouts that are much easier to express algebraically for equation-controlled manufacturing. Second, the amount of packaging and interconnections, and the number of elements involved can be reduced over present methods. Third, new system-function device specifications will emerge.

System-Tailored Devices

The previous discussion presented an example in which circuits and system-function logic were combined using standard transistors. Present active devices are individual elements packaged separately, as shown in Fig. 9. The connections between the active and passive elements are generally made by individual wires, although more recent systems use printed wiring for circuit packages.

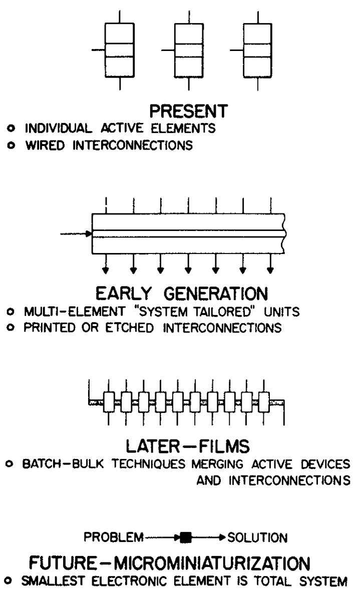

PRESENT
o INDIVIDUAL ACTIVE ELEMENTS
o WIRED INTERCONNECTIONS
EARLY GENERATION
o MULTI-ELEMENT "SYSTEM TAILORED" UNITS
o PRINTED OR ETCHED INTERCONNECTIONS
LATER—FILMS
o BATCH-BULK TECHNIQUES MERGING ACTIVE DEVICES AND INTERCONNECTIONS
PROBLEM → SOLUTION
FUTURE-MICROMINIATURIZATION
o SMALLEST ELECTRONIC ELEMENT IS TOTAL SYSTEM

Fig. 9—Devices "system tailored."

In an early generation, multi-element system-tailored devices will be available. In addition, a much greater proportion of the interconnections will be etched and printed. Multi-element miniaturized components have been made available in small quantities by American Bosch Arma, the Diamond Ordnance Fuze Laboratory, Hughes Aircraft, RCA, Texas Instruments, and others. Programs in molecular electronics to permit the use of plating and vacuum-deposition processes are also receiving attention. Much of this work is for military applications but will probably be available for commercial use in the near future.

The production of interconnections and active elements in one continuous manufacturing process will occur with the introduction of films, either thick or thin, into systems. At this time, semiautomatic methods of manufacture will be mandatory. Here it is obvious that separate considerations of system functions, circuits, and devices may no longer exist. Magnetic coupling is used to accomplish switching in thin film cryogenic systems and speeds are very high. One suspects that nature also provides a medium speed and cost arrangement if we are clever enough to detect it.

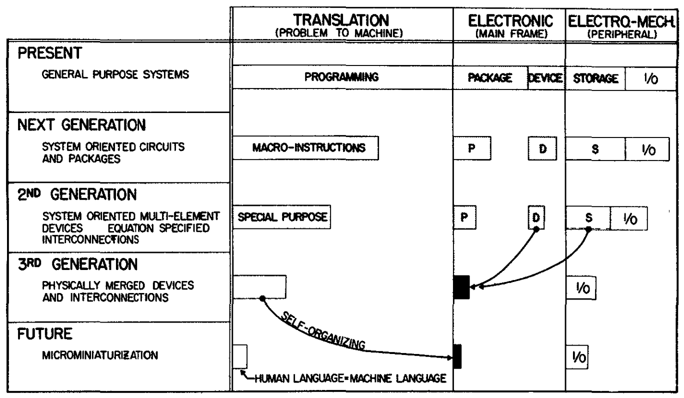
Further in the future we may anticipate true microminiaturized systems constructed from automatic, computer-controlled processes utilizing bulk materials. The late Professor Dudley Buck has defined a microminiature computer as: "A computer on a scale which could never be looked at in an optical microscope." In this technology, the cost of active elements will approximate the cost of interconnections. Logical designers may enjoy the luxury of utilizing thousands of active elements to perform logical functions of a complex nature.

One of our major objectives is to reach the future system illustrated here. Let us now consider some of the more important work to be done to make this possible.

Digital Data Processing

Approximate Relative Costs

The bar graph (Fig. 10, Line 1) shows the approximate relative costs of processing data in presently available commercial general-purpose digital systems. Problem preparation and programming costs are generally accepted as being approximately half of the total. The remaining costs may be divided into two major items: electronic main-frame costs and electromechanical peripheral-equipment costs. The percentages vary from system to system, but are essentially as follows: The cost of main-frame electronics varies between 15 and 25 percent of the total, and includes the main random access storage, the arithmetic and logic unit, and controls. In the main-frame, the switching devices cost approximately one-third and the packaging (which includes circuit cards, panels, interconnections, frames, display, covers, etc.), approximately two thirds. The cost of the electromechanical portion of a system may vary between 25 and 35 percent of the total and may be divided into two parts. The first is bulk storage involving mechanical motion. This part includes tapes, discs, drums, etc., and their attendant electronic equipment. The second part is the input-output equipment, including communication devices.



Present Generation

General-purpose systems predominate at the present time. This is probably due to the relatively high cost of research and development coupled with long design and manufacturing lead-times for initial production. Instructions usually include an operation, one or two addresses, and a few special control bits. The instruction code at the machine language level is relatively "micro" due to the general-purpose requirement and for other reasons not covered here.

System specification normally starts with a market analysis so that a potential product may be defined. Performance, storage volume, input-output equipment, etc., are established at this time. Available standard circuits and packages are considered during the specification of system logic. Outputs from the system design are block diagrams, or equations, or both. At this stage we do not know where each device or circuit will be placed, nor the length of interconnections.

In programming, present generation machines use autocoders to translate from problem language into machine language. The autocoders, in many instances, involve execution time and occupy storage space. This combination of autocoders and machine language is the result of the programmer's desire to have a machine language different from the one technology is able to economically provide.

Devices used in present systems, both active and passive, are individually manufactured by semiautomated methods. This allows individual testing, selection, and replacement in the event of malfunction.

The circuits are Boolean optimized and the minor packaging assemblages usually include several elementary functions. Recent trends as evidenced in machines like the Philco TRANSAC, are toward the inclusion of more Boolean-type circuits on each pluggable element. Interconnections are a mixture of printed cards and hand inserted wires and cables.

The major mechanical design of a system starts when logical specification and Boolean standardized circuits are available. With this information, the active and inactive elements may be located and packaged. For the first time, lead lengths become accurately known. The output from mechanical design is generally a complete set of blueprints which go to the manufacturing engineering groups.

In the peripheral equipment area the bulk storage usually involves magnetics and includes much mechanical equipment. Access to data in this type of storage is either serial-by-bit or serial-by-character. The input-output equipment is essentially mechanical, taking data from a keyboard to a buffer storage and, later, taking data from a buffer to a printer to produce hard copy.

Servicing is usually done by a combination of electrical tests and diagnostic programs. It involves locating the defective active or passive elements and substituting new pluggable cards.

The specification and design of present systems is thus essentially a serial process in which most major elements are individually standardized and then assembled to make a system. The design feedback loops, while many, have rather high impedance.

Next Generation

The next generation as illustrated by the bar in Fig. 10, Line 2, may be *characterized mainly by system-oriented design and manufacturing techniques.* Commercial machines will probably remain general-purpose in nature.

The bars illustrating approximate relative cost on this and succeeding generations does not necessarily indicate that the cost of an equivalent advanced machine will be reduced. The length of the bars represents the relative proportionate cost for each of the major elements in a system for a particular generation. Past experience has shown that as more powerful techniques become available we solve larger problems; therefore, we have an option of obtaining more computing for our millions or reduced costs for the same amount of processing. This is obviously a designer's choice and will be adjusted to suit requirements as he specifies a particular system.

A major change will occur in the specification of systems. Logic and circuits will be merged to produce new system function circuits utilizing standard devices. The physical location of components, the interconnection lengths and paths, and layout of the package will be specified as an integral part of logic. To attain these objectives a new "system-function algebra" is necessary. This algebra, which will begin with the logical Boolean expressions, must be enriched to include the active and passive device characteristics, the physical location of all components, the interconnection paths and lengths, and timing.

Programming in this generation will be done with more powerful macro-type instructions. Machine language instructions will approximate the level typified by coding systems such as FORTRAN. Relatively speaking, more hardware will be in the instruction controls with the objective of making programming easy and fast.

Improved single-function devices and some use of multifunction devices may be anticipated.

A major change in packaging as well as in logic-circuit specification will occur in this generation. Complete system functions will be packaged on one replaceable element. Interconnections will be etched, printed, evaporated, or batch produced by other automated techniques. Manufacturing equipment, methods, and mechanical design techniques must undergo the appropriate changes.

Service will be accomplished by locating and replacing malfunctioning major system functions. If the individual devices are expensive, they may be replaced at a testing and service center so that the system function may be returned to stock. If not, the whole unit may be discarded. Extensive built-in checking and automatic program diagnosis will be included. The logic of the machine will require more redundancy for checking and diagnostic purposes.

The next generation of systems thus involves major improvements in logical design and packaging. New devices or other research items are not necessarily required.

Second Generation

Two major changes characterize the second generation systems. (Fig. 10, Line 3). *First, system-tailored multi-element devices* will be used extensively. This will influence mechanical design, packaging, and manufacturing equipment. *Secondly, special-purpose machine systems to solve classes of problems* will be made on the same manufacturing line. The logical specification of these machines will be generated by computers utilizing system-function algebra. Extensions of the algebra will control the manufacturing setup. This combination will drastically reduce design and production lead times and cost of the product.

The availability of special-purpose systems will ease programming difficulties through the use of application-tailored languages to solve related classes of problems.

System-function design techniques and devices will be applied to bulk storage. For input-output, electronics will replace mechanical equipment wherever possible.

No on-line service will be performed since the machine will be able to select alternate logical paths in the event of a malfunction. At inspection periods, previously-flagged defective system elements will be removed and replaced.

Third Generation

The true revolution begins in the third generation. (Fig. 10, Line 4). Here, *device, package, and interconnections are inseparably merged.* Major system functions will be produced from bulk materials in computer-controlled continuous manufacturing processes. Techniques such as vacuum deposition, electron-beam writing, spraying, printing, etc., will be utilized, depending on device technology chosen relative to the speed and cost range desired. The use of three-dimensional connections will alter packaging concepts. Miniaturization for complete systems may now be realized. This miniaturization will allow dramatic increases in the number of active elements available for both logic and storage.

The availability of vast amounts of homogeneous storage with internal logical capabilities will drastically alter programming methods. In particular, built-in symbolic addressing will eliminate the inefficient and tedious housekeeping associated with present-day machines. Coupled with special-purpose instruction sets, this will allow machine language to approximate problem language.

The input-output equipment will now be reduced to that which is used to communicate with humans or from machine to machine, since bulk storage is now merged with the main frame.

Service will be simple because automatic error detection and correction by the machine will allow continuous operation. Defective elements will be replaced at the next service period.

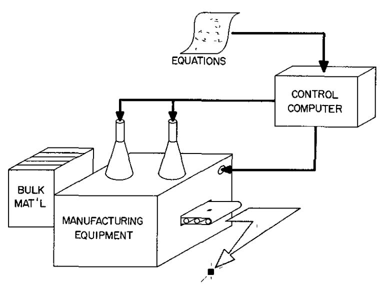
Future Generation

We may envision a few aspects of future generations now (Fig. 10, Line 5). True *microminiaturization* meeting Professor Buck's definition will be realized. Self-organizing systems will become possible due to microminaturization and better understanding of the logic involved. The use of self-organizing systems to find optimum solutions to problems will allow us to synthesize more economical, special-purpose systems for on-line use.

For programming, we may anticipate that machine language will approximate or equal human language if we have progressed properly to this point and if we use self-organizing systems appropriately. A major change in input-output techniques is required. Voice and pattern recognition, and vastly improved display and printing systems are needed.

In this generation, service will be accomplished by throwing the whole computer away.

In summary, to progress from the present day data processing capabilities to more desirable future systems, we require greatly increased logical capabilities, vast amounts of storage, improved input-output methods and more speed. All these elements tend to require microminiaturization, batch-bulk processing, automated logical synthesis, and equation-controlled manufacturing. Consequently, both speed and system cost require and benefit from this revolution.



FUTURE COMPUTER
"ELECTRONICS"

STANDARDIZED ON·
- BULK RAW MATERIALS
- MERGED DEVICES AND INTERCONNECTIONS
- SYSTEM FUNCTION ALGEBRA
- MANUFACTURING METHODS—COMPUTER CONTROLLED

OBTAINING·
- EFFICIENT SPECIAL PURPOSE SYSTEMS
- PRODUCED RAPIDLY AND ECONOMICALLY

RESULTING IN·
- MORE BRAINPOWER ON DEFINING PROBLEMS
- CHEAPER PROBLEM SOLUTION

Fig. 11—Future computer "electronics."

Conclusion

Future computers (Fig. 11) will be standardized as follows:

1. Interconnections and active devices will be made in a continuous process from bulk raw materials to finished product.
2. The device, circuit, and interconnection technology will merge.

3. System-function algebra will be used to specify all aspects of design.
4. Completely automated, computer-controlled manufacturing methods will be used.

From these techniques we will obtain efficient special-purpose digital data processing systems. They will be produced economically with short design and construction lead-times through complete automation. This will result in more brain power being devoted to discovering and defining new problems, and in their cheap, efficient solution.

Discussion

J. H. Felker: (AT&T) I would like to hear you complete the job of prophecy, Mr. Rice, and give us some idea of the timetable you envision for these first, second, third (and) fourth generation machines.

Mr. Rice: I have given that question considerable thought. We are working on the next generation right now in many research laboratories. The universities are probably ahead in some respects in their thinking on research in this area. It is not necessarily true that each generation requires the specific items at the same time as shown in the paper. If we develop microminiature computer devices ahead of new programming techniques, they may be utilized early. I suspect, and this is a personal observation, that the first models of microminiature computers are ten years off and the other items for the next generation are scattered from three to five years away in production. This is a guess on my part.

Mr. Felker: Thank you. With the three year period it takes to design and get production of the conventional computer, how can you anticipate anything as drastically different from what we do today as your microminiature computer in only ten years? Where are the people and the knowledge that will permit this in ten years?

Mr. Rice: I agree that the ten years is probably on the optimistic side. However, you will note that the methods of specification for what I call the "algebra" of these systems includes the device characteristics and the physical layout. This implies that much of our early work is in development of a new "system function algebra." Once this algebra is automated, the design of new systems will be done rapidly, and we will be less dependent on present day design techniques.

H. Richmond (System Development Corp.): What is meant by "machine language is approximately problem language"? What is done in this case if a new variable is needed and your hardware is built?

Mr. Rice: We have to recognize in our future designs that problem language is not static. In other words, FORTRAN, if I may use that example, has already proven that we need extensions. Therefore, I think the computer designers — and I happen to be one who believes this — must design control sections which admit that programming language is dynamic. We should be able to incorporate new instructions without going back and completely rewiring. There is much research work to be done on the type of control situation implied. I, for one, am very anxious and excited about working in this area.

J. Feitler (IBM): What about analog-computer logic with digital-computer hardware with many arithmetic elements (100 to 1000-plus arithmetic elements) using microminiature components at "3rd generation level"?

Mr. Rice: I am not certain that I fully understand the implication of analog-digital computer hardware. If you mean we are working on separate portions of the problem in parallel, using the accuracy obtained by digital techniques, I think there are existing machines showing this tendency. Assuming that we can assemble these systems to solve the classes of problems we have to solve, this is an interesting area for development. As to the 3rd generation I don't think I would hazard a guess.

L. B. Harris (GE): How do you propose to implement self-checking of system functions, that is, to pin-point the trouble?

Mr. Rice: Much work is being done on this subject in various research and development laboratories. I think we have to reanalyze where we want to spot errors. For example, in the talk a complete arithmetic process unit is shown as a single system function. I purposely chose the one-out-of-ten code in this example, because it is possible to put a single check device at the far end of the system. If more than one pulse arrives, there is trouble. If less than one pulse arrives, there is trouble. If only one pulse arrives, I would assume it is correct, because the logical paths do not cross. Much research remains to be done in this area, so I don't have a complete answer. I believe we should analyze how small or how large an element should be when we look for trouble. We should probably diagnose trouble in major elements rather than at the Boolean circuit level. We should also examine our need for a single code throughout a complete system. That is to say, do we need the same bit code in the processing element that we need in bulk storage. There are many ways of tackling the problem, and I think we will have to look to future generations for the complete answer.

C. H. Propster (GE): What reason do we have to think a self-organizing computer will ever be produced?

Mr. Rice: Perhaps you are in a better position to answer this question than I am. I believe that two things are necessary before self organizing systems are more than (if I may use the expression loosely) ideas: First, we have to really understand what we want to do in the system to make it self-organizing. This is the logical consideration. Secondly, it is fairly obvious it will take lots of components, so we have to develop the manufacturing techniques to produce large numbers of components economically. Whether or not we will get to the most blue-sky systems is hard to predict, and I will shy away from that. I think that manifestations of self-organizing systems are possible, and that they will be developed.

P. J. Scola (GE): On the throw-away computer, what will the input-output wiring look like? Will there *be* any input-output?

Mr. Rice: This is a very difficult question to answer, even in an hour and a half. At all stages in the future, we will need communications from humans to the machine. We hope that voice recognition will allow us to get from a human to the machine language. In the throw-away portion, I am specifically referring to the electronic elements of the computer: that which we now know as the main frame. In particular, the capacity of the bulk storage associated with the main frame is drastically increased. This will reduce the peripheral equipment such as tape, discs and so forth. So in effect we will be throwing that section of I/O away. The concept of throwing away is also hard for *me* to accept. However, I ask myself how are we going to repair microminiature devices; and I come up with the answer that we had better make them cheap enough so we can throw them away.

F. Panch: Would you care to speculate on what kind of computers might be in use twenty to forty years from now?

Mr. Rice: Frankly, I have trouble envisioning what I call future computers. I think that the major changes beyond these generations will be in new uses for computing systems. If we can make computer language approximate human language, or at least equal problem language, the challenge will be in what we do with the system and in making the systems cheaper so we can use them more frequently.

R. J. Brousseau (U of C): In saying that computer language should approach problem language in future computers, are you suggesting that the computer hardware should accomplish the functions now being borne by present automatic programs, such as mnemonic instructions, symbolic memory names?

Mr. Rice: The answer to that question, in terms of generalities, is yes. First we need better devices to go into memory so we may perform logic in the memory itself. At a time when this technique is sufficiently advanced, we may expect that instead of "addressing," we can tell the memory to find a particular field of data by specifying the "tag" inherent in the data. There are several other logical techniques which may be used for symbolic look-up. The extent to which designers can do this is dependent on the person specifying the problem. He must establish a set of rules that is fixed.

Negative-Resistance Elements as Digital Computer Components*

MORTON H. LEWIN†

Introduction

IN DETERMINING the maximum repetition rate of a given switching circuit, the response of the switching device and the effect of other circuit parameters (including stray elements) must be taken into account. Although the switching speed is ultimately limited by the device, in many cases one never reaches this theoretical maximum because circuit limitations play the dominant role. To solve this problem, one is forced to devise extremely simple circuits with few components in order to minimize the effect of stray reactance. The use of two-terminal negative-resistance elements allows one to do this.

Shockley and Mason[1] have proposed that the ultimate high-speed semiconductor amplifying device is a two-terminal negative-resistance element. They reason that, since the speed of semiconductor components is basically limited by the transit time of carriers, the physical dimensions of devices operating in the highest frequency ranges must be extremely small. In the limit, fabrication problems dictate two-terminal active elements, where only one dimension need be small.

This paper is first concerned with the general problem of using two-terminal negative-resistance devices as the only active switching elements in a digital system. Specific circuits are then discussed, using a particular voltage-controlled negative-resistance device as an example. Much of this treatment can be adapted to other negative-resistance elements.

Gain

A combinational switching circuit is defined as a circuit whose outputs depend only on the present inputs. This is to be distinguished from a sequential switching circuit in which the outputs depend not only on the present inputs but also on the past history of inputs. Thus, a combinational circuit, by definition, has no memory.

Consider a system of combinational circuits employing negative-resistance devices as the active switching elements. The requirement of no memory dictates either monostable operation of the negative-resistance elements or bistable operation with a built-in reset to eliminate storage. (The possibility of combinational circuits composed of sequential subcircuits is ignored on the grounds that such complicated circuits will reduce the maximum speed of the system.)

For the case of monostable operation, if one removes any one of the negative-resistance elements from the circuit, measures the static V-I characteristic seen looking into the rest of the circuit from its two terminals and then superimposes this on the negative-resistance characteristic, there is always only one stable intersection, for all input combinations. For an all-passive circuit, such as a conventional diode gate, one intersection (operating point) is assured. Assume that this measured characteristic can be approximated by a straight loadline, in the region of interest (A design to insure monostability is feasible only if this characteristic is "well-behaved" (*i.e.*, monotonic) in the region where it intersects the negative-resistance characteristic.) This leads to a simple situation which can be directly analyzed.

Typical voltage-controlled and current-controlled negative-resistance characteristics are shown in Fig. 1a. The two states for each device are most conveniently chosen as operation in the two positive-resistance regions on both sides of the negative-resistance region. Thus, for a voltage-controlled element, the state is defined by the voltage across the device, and for a current-controlled element by the current through it. Under the conditions described above, the circuits to be analyzed become those shown in Fig. 1b. The combination of R and the power source represents the Thévenin equivalent of the linearized measured characteristic.

Monostable operation can be achieved in two ways as indicated by the load-lines in Fig. 1a. In the first case, labeled "I", $R < R_{n\,\min}$ for the voltage-controlled element and $R > R_{n\,\max}$ for the current-controlled element, where $R_n = |\,dV/dI\,|$ in the negative resistance region. In the second case, labeled "II", R does not satisfy this inequality but the power supply values are chosen to result in one intersection. Thus, for case I monostable operation results regardless of the power supply parameters (assuming no reactance), while for case II the power supply values must be chosen to avoid bistable operation.

* This work was supported by the Bureau of Ships, U. S. Navy, under Contract NObsr 77523 with RCA. It is the basis of a dissertation to be submitted in partial fulfillment of the requirements for the Ph.D. degree at Princeton University, Princeton, N. J.

† RCA Laboratories, Princeton, N. J., and Department of Electrical Engineering, Princeton University, Princeton, N. J.

[1] W. Shockley and W. P. Mason, "Dissected Amplifiers Using Negative Resistance," *Journal of Applied Physics*, Vol. 25, No. 5, p. 677; May 1954.

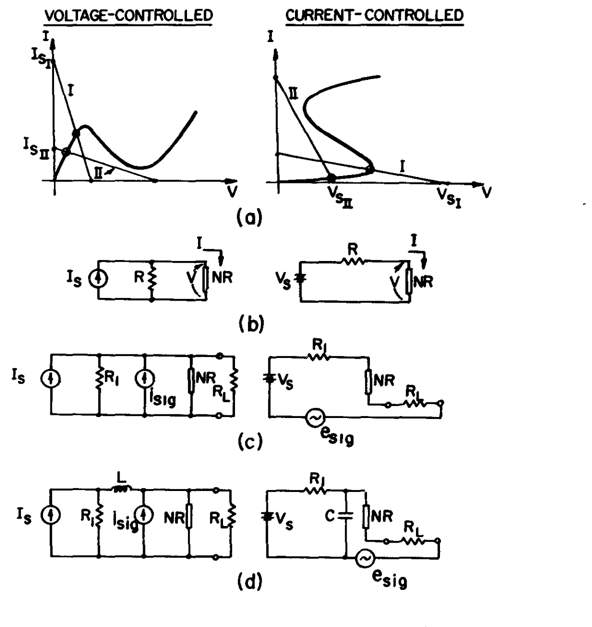


Fig. 1—(a) Load-lines for monostable operation. (b) Equivalent circuits. (c) Load and signal source included. (d) Addition of reactive element.

A DC-coupled system with no reactive elements will be assumed. The output terminals and equivalent load resistance R_L are shown in Fig. 1c. Input signal sources are also included. R_L represents the load furnished by other gate circuits in the net. R_1 represents the contribution to R of the internal parameters of the circuit under consideration. The series or parallel combination of R_1 and R_L, as appropriate, yields R. For the voltage controlled case, R_L can vary from ∞ to some minimum value and for the current-controlled case, from 0 to some maximum value. The fact that R_L varies as indicated is a direct result of the two-terminal nature of all components. For example, an examination of the possible configurations using voltage-controlled elements reveals that, in general, the output current from a stage in a given state depends on the states of the circuits being driven.

The values of R_1 and I_s or V_s must be chosen to assure monostability for all loads. Thus, they must be chosen such that only one intersection (of the type shown in Fig. 1a) occurs for $R_L = \infty$ in the voltage-controlled case and $R_L = 0$ in the current-controlled case. If these conditions are satisfied, monostable operation is assured for all R_L.

Recall that any reactance in the circuit is assumed negligibly small. For case I, looking into the circuit from the output terminals, the load resistor R_L sees a net positive resistance for all voltage-current conditions. Hence, there is no possibility that an increment of energy delivered to the load will be greater than that supplied by the signal source, for any value of R_L. For case II, assuming a rectangular signal pulse which raises the load-line sufficiently to cause the operating point to switch to the other positive-resistance region, a simple calculation[2] reveals that the input energy is at least as great as the output energy. Thus, the requirement of monostability, in the absence of adequate reactance, leads to a circuit which has no gain.

If one now allows the use of appropriate reactive elements (*i.e.*, capacitance in parallel with the current-controlled device and inductance in series with the voltage-controlled device), as shown in Fig. 1d, gain can be achieved. Note that the added reactance cannot simply be greater than zero but must be greater than a certain minimum established by stray elements and the properties of the negative-resistance device. (For an AC-coupled system, the reactive coupling elements must also be taken into account.) In this case the gain arises from the fact that energy stored in the reactive element is delivered to the load when the negative-resistance device is triggered by a small signal. Such circuits have been treated in the literature[3,4,5]. It is shown there that the recovery time associated with the reactance is a factor which limits the maximum repetition rate of the circuit.

Bistable-with-reset operation allows one to achieve gain without the use of reactive elements. Since the furnishing of a reset signal may be considered to be an additional function of the power supply, effectively a time-varying power source is now being considered. One possible arrangement is to let the power supply (current or voltage) deliver a continuous train of rectangular pulses, such that during each pulse (excitation) the negative-resistance device can go to either one of its two states, depending on input conditions. The "reset" is then the termination of the excitation pulse. It can be seen that such a power supply also serves as a master clock. If one now calculates[6] the transition and recovery times for such a system and compares this to the system with DC power supplies and reactive elements, it is evident that the former scheme has the higher maximum repetition rate.

Directionality

Another fundamental problem is concerned with making the system unilateral. For example, since the negative-resistance element is a two-terminal device, when one terminal is grounded, the other must act as both the input terminal and the output terminal. One must therefore provide some means to dictate the direction of flow of information in the system (*i.e.*, to make a circuit directional, so that a signal propa-

[2] See Appendix.

[3] B. G. Farley, "Dynamics of Transistor Negative Resistance Circuits," *Proc. IRE*, Vol. 40, pp. 1497-1508; Nov. 1952.

[4] A. E. Anderson, "Transistors in Switching Circuits," *Proc. IRE*, Vol. 40, pp. 1541-1558; Nov. 1952.

[5] A. W. Lo et al., "Transistor Electronics"; Prentice Hall, 1955.

[6] See Appendix.

gates *from* input *to* output). Some possible techniques for achieving directionality include use of passive elements such as Hall-effect couplers or gyrators, use of non-linear interstage coupling elements such as conventional diodes, synthesis of three-terminal circuit configurations with some unilateral properties, and separation of input and output functions in time using a time-varying power supply. Some of these techniques, as applied to circuits involving voltage-controlled elements, are discussed in more detail following the treatment of basic logic circuits.

Power Supply

Assuming the bistable-with-reset mode of operation, with the momentary removal of power supply excitation as the method of resetting, the waveform shown in Fig. 2a represents an acceptable source waveform. The sequence of operations performed by each stage is then as follows: having been reset, a given circuit is energized to an initial state. If the combination of inputs presented to it is favorable, it will switch to its other state. The state of the circuit is then detected by the next stages. Finally, the circuit is reset, energized again and ready to receive a new combination of inputs.

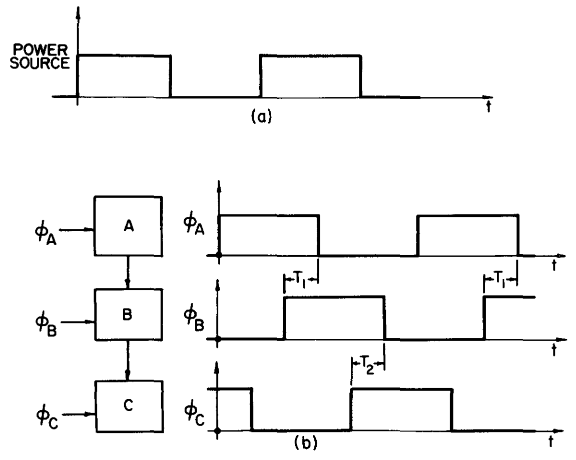


Fig. 2—(a) Power supply waveform.
(b) Three-phase power source.

If the entire system is powered from the same source, all circuits are reset simultaneously. The energizing pulses must then be wide enough to allow signals to propagate from the inputs of the system to its outputs, so that the repetition rate is limited by the longest signal propagation time expected. To increase the repetition rate, the system is broken up into small groups of gates such that each group is reset immediately after it has performed its function. A *sequence* of resets is then required in order that information will continue to propagate and will not be erased. These requirements can be satisfied by a multiphase power supply such as, for example, the three-phase waveform shown in Fig. 2b. Using this method, a given gate or group of gates is powered by one phase, drives other circuits powered by the next phase and is driven by still other circuits powered by the previous phase. The excitation pulses overlap in time such that information propagates between two stages during the period when both are energized simultaneously. Considering the block B in Fig. 2b, one can see that the beginning of its supply pulse, T_1, corresponds to an "input" region and the end of the pulse, T_2, to an "output" region. The three-phase arrangement shown is characterized by the fact that there is always a group of circuits in the de-energized condition at any given moment. As a result, in many cases spurious signals are prevented from propagating. The similarity between this scheme and the multiphase clock systems used in conventional machines is only superficial. Here the clock source is also the power supply.

Generalized Analysis

Load-curves

Consider a two-terminal "black-box" A whose static V, I characteristic is given by either $I_A = g_1(V_A)$ or $V_A = f_1(I_A)$. The box may simply hold a single negative-resistance element or may include a more complicated arrangement of elements whose composite two-terminal V, I characteristic is given by the above equations. f_1 (or g_1) may be any continuous function and is *not* single-valued in both V and I if there are any negative-resistance regions. Now consider a second two-terminal "black-box" B whose static V,I characteristic is given by either $I_B = g_2(V_B)$ or $V_B = f_2(I_B)$. This will correspond to the device which determines the load-curve. If $g_2(V_B) = V_B/R$ (*i.e.*, $f_2(I_B) = I_BR$), then the device is the resistor R, mentioned before, and the load-curve is a straight load-line. In general, however, both f_1 and f_2 are non-linear, negative-resistance characteristics. The two cases of interest are the following configurations:

(a) A constant-voltage source V_s across the series combination of elements A and B.

(b) A constant-current source I_s feeding the parallel combination of elements A and B.

The pertinent equations are:

Case (a)		*Case (b)*
$I_A = I_B$		$V_A = V_B$
$V_S = V_A + V_B$		$I_S = I_A + I_B$
$V_A = f_1(I_A)$	[*1*]	$I_A = g_1(V_A)$
$V_A = V_S - f_2(I_A)$	[*2*]	$I_A = I_S - g_2(V_A)$

The equilibrium points or quiescent operating points for a circuit are determined by the intersection points of the two curves [*1*] and [*2*]. In either case, curve [*1*] is the characteristic of A and curve [*2*] is the load-curve determined by the power supply and the characteristic of B. For case (a), the load-curve is the image of B's V, I characteristic reflected through the current axis, translated in the positive voltage direction a distance V_S. For case (b), the load curve is the image of B's V, I characteristic reflected through the voltage axis, translated in the positive current direction a distance I_S. The only stable operating points are those determined by the intersection of two positive-resistance regions.

Composite Characteristics

The determination of the composite V, I characteristic of two or more two-terminal elements, given their individual characteristics, is important in the analysis of negative-resistance circuits. To graphically obtain the composite characteristic from the individual curves, one follows these simple rules:

(1) For two elements in series, each point of the composite curve with coordinates V_1, I_1 is obtained by choosing any I_1 and letting $V_1 = V_{A1} + V_{B1}$, where V_{A1} is the voltage across element A at the current I_1 and V_{B1} is the voltage across element B at the current I_1. Thus, one adds the voltages across the individual elements at the same current.

(2) For two elements in parallel, each point of the composite curve with coordinates V_1, I_1 is obtained by choosing any V_1 and letting $I_1 = I_{A1} + I_{B1}$, where I_{A1} is the current through element A at the voltage V_1 and I_{B1} is the current through element B at the voltage V_1. Thus, one adds the currents through the individual elements at the same voltage.

Tunnel Diode

The remainder of this discussion is concerned with one particular voltage-controlled negative-resistance element. Similar or "dual" treatment can be given to current-controlled devices.

The device to be considered was first reported by Esaki[7] and has since been investigated by others[8]. Since the phenomenon responsible for the unique characteristics of the device is the tunneling phenomenon predicted by quantum mechanics, the device has been called the tunnel diode. It holds promise of being an extremely fast element. Units with time constants of a fraction of a millimicrosecond have been fabricated. Preliminary tests verify that the device is capable of very high speed operation.

For the purposes of this analysis, the V, I characteristic of the tunnel diode will be assumed. Descriptions of the physical operation of the device are given by Esaki[7] and Sommers[8].

The static voltage-current V, I characteristic for a typical germanium unit is shown in Fig. 3. Typical values for the critical points are indicated. The inverse slope R_0 of each positive resistance region is of the order of a few ohms.

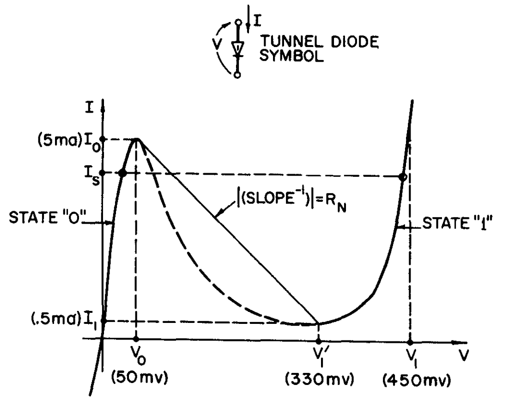


Fig. 3—Tunnel diode static characteristic. (Number indicated for typical Germanium unit.)

Since the tunnel diode is such a low impedance element, it is not practical to assume that a constant voltage source is available to supply power to many units. In view of the fact that the source impedance of any realizable voltage source will be of the order of that of its load, it is more practical to assume that the power to individual units is supplied from current sources. In cases where a voltage source is desired, an individual auxiliary device for each circuit is necessary to simulate it. (This is demonstrated later.) Therefore, in line with previous discussions, a three-phase square-wave current source as shown in Fig. 2b will be assumed.

Threshold Gate

Consider the circuit shown in Fig. 4. Assume the input terminals are connected to output terminals of other similar circuits. As long as the tunnel diode D is in the O (low voltage) state, the current into D, in addition to I_S, is approximately $M(V_1 - V_0)/R$, where M is the number of driver units which are in the *1* (high voltage) state and I_s, V_1 and V_o are defined in Fig. 3. D will switch to the *1* state only if $I_S + M(V_1 - V_0)/R > I_o$, the high threshold current at which the resistance becomes negative. Once it has switched to the *1* state, it will remain there even though the current into D is substantially less than I_S (corresponding to loading), as is evident from an examination of the characteristic. Thus, the circuit is capable of logical gain, since it can now furnish a number of output current increments $(V_1 - V_0)/R$ to the next stages. The output of the threshold gate, then, is *1* only if the total number of

[7] L. Esaki, "New Phenomenon in Narrow Germanium p-n Junctions," *Phys. Rev.* vol. 109, p. 603; Jan. 1958.

[8] H. S. Sommers, Jr., "Tunnel Diodes as High-Frequency Devices," *Proc. IRE*, p. 1201; July, 1959.

K. K. N. Chang, "Low-Noise Tunnel Diode Amplifier," *Proc. IRE*, p. 1268; July 1959.

Chang, Nelson, et al.,"Tunnel Diodes for Low Noise Amplification," *Proc. IRE WESCON*, Aug. 1959.

Aarons, Holonyak, et al., "Germanium and Silicon Tunnel Diodes-Design, Operation and Application," *Proc. IRE WESCON*, Aug. 1959.

1 inputs is greater than or equal to some integer T. I_S is adjusted to result in the correct logical function (*i.e.*, the correct T). For an OR gate, T is one; for an AND gate, T equals the number of inputs; to generate the CARRY output in a full adder, for example, the number of inputs is three and T equals two, etc. The circuit must be reset back to the operating point below the threshold in order to be able to perform its function again.

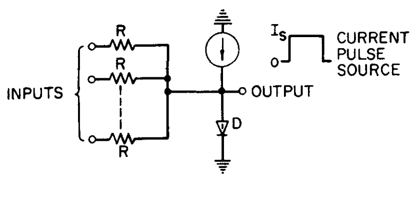


Fig. 4—Single-ended threshold gate.

It is evident that the merit of this circuit depends primarily on the uniformity of diode characteristics and the power supply tolerances involved. The maximum variations in I_0, I_S, V_1 and V_0 dictate the minimum current increments for reliable switching. Advances in fabrication techniques have already resulted in high yields of diodes matched well enough that a reliable logic system involving such circuits appears readily realizable.

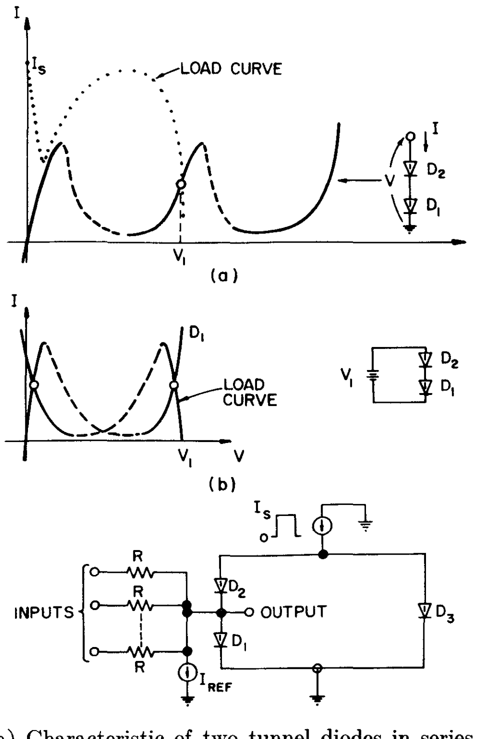


Fig. 5—(a) Characteristic of two tunnel diodes in series and load-curve formed by D_3 and I_S. (b) Bistable operation. (c) Balanced threshold gate.

The operation of the "single-ended" threshold gate, described above, relies on the accurate determination of the operating point on the negative-resistance characteristic. A balanced or symmetrical circuit offers advantages in many applications. Consider the series combination of two tunnel diodes. Their composite characteristic is shown by the solid curve in Fig. 5a. If a voltage V_1 is applied across the series combination, it is possible for the circuit to exist in either of two states (*i.e.*, one diode in the high voltage state and the other in the low-voltage state and *vice versa*). This is depicted in Fig. 5b where D_2 and V_1 determine the load-curve across the characteristic of D_1. If the voltage V_1 is applied as a *pulse* as is done in the proposed system, one can determine to which state the circuit goes by a small signal at the junction of D_1 and D_2[9]. This can be explained by noting that during the rise of the pulse, the current through D_1 and D_2 builds up to the point where both are very near the crest of the hill. The small current into the junction is sufficient to determine which diode breaks down. Thus if this current is positive, D_1 goes to the *1* state and if it is negative D_2 goes to the *1* state and and D_1 is forced to the *0* state.

The difficulty in obtaining a constant-voltage pulse source to drive a large number of such low impedance circuits has already been mentioned. However, the tunnel diode has another important property in that it can simulate a low impedance voltage source, of magnitude V_1, if the current through it is greater than I_o, the high threshold current. This property is utilized to arrive at the final form of the balanced circuit, shown in Fig. 5c. As is shown in Fig. 5a, the dotted load-curve formed by D_3 and I_s intersects the characteristic of the series combination of D_1 and D_2 at the appropriate point, if I_s is large enough. The circuit is now powered by the more realizable current source.

The logical functions OR, AND and THRESHOLD are achieved by requiring that the current into the junction be positive only when at least one, all or some of the inputs are *1*'s, depending on the function desired. This requires a reference current or bias as shown. The source of this reference current can be another tunnel diode again acting as a voltage reference.

From the above description one can see that the balanced circuit has several advantages over the single-ended scheme. First, the sensitivity of the circuit depends only on the matching of the two negative-resistance elements and not on the exact values of the critical points of the characteristic. Second, the sensitivity is virtually independent of reasonable power supply variations.

INVERTER

The composite characteristic of a tunnel diode D_1 in series with a resistance R_1 is shown by the solid curve in Fig. 6a. R_1 is chosen to be approximately R_N, the magnitude of the linear approximation to the negative-resistance (see Fig. 3). Suppose points a and b were the only stable points for the circuit (corre-

[9] This scheme was suggested by A. Lo.

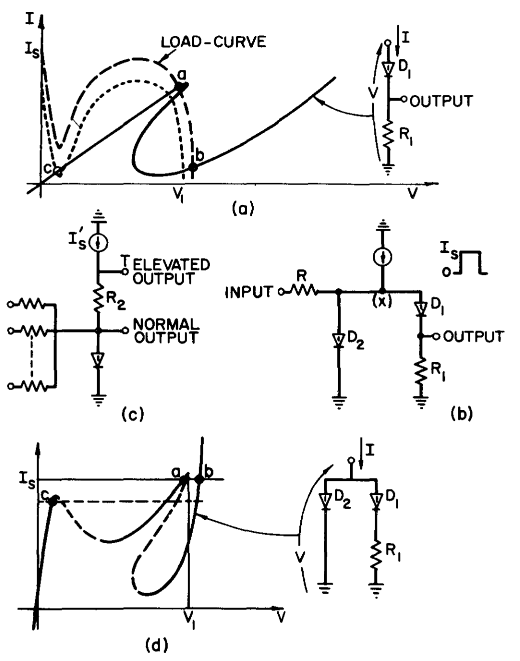


Fig. 6—(a) Characteristic of tunnel diode and resistor in series and load-curves formed by D_2 and I_S. (b) Inverter circuit. (c) Provision for obtaining elevated output. (d) Composite characteristic of D_1, D_2 and R_1.

sponding approximately to a voltage V_1 applied across the series combination). Taking the voltage across the resistor as the output voltage, we have that point *a* yield as *1* output (high voltage; high current through the resistor) and *b* yields a *0* output (low voltage; low current through the resistor). Thus, if the circuit is always at *a* with an *0* input and at *b* with a *1* input, it would realize the inversion function.

To make *a* and *b* the only stable operating points, one can again use another tunnel diode D_2 to simulate a voltage source. The inverter circuit is then as shown in Fig. 6b, and the intersections of the dashed load-curve determined by D_2 and I_s with the composite characteristic of D_1 and R_1 in series are shown in Fig. 6a. To clarify the operation further, one can plot the composite V,I characteristic of the entire configuration of D_1, D_2 and R_1. It is shown in Fig. 6d. The horizontal (constant-current) load-line formed by I_s is indicated.

Since the voltage at point *x* (Fig. 6b) is high for both operating points, one must include some provision for adding a constant to the normal output voltage levels of the driver tunnel diode, in order that a "1" driver output can furnish the current necessary to bring the inverter over the *a*-hill to *b*. This can be accomplished for a single-ended gate by the addition of a resistor R_2 to the circuit, as shown in Fig. 6c. The voltage of terminal *T* (during excitation) is greater than the normal output voltage by a constant amount I'_sR_2, assuming negligible loading at *T*. By adjusting R_2 so that the *0* output voltage is approximately V_1, the *1* output voltage is then approximately $2V_1$, and the required operation can be achieved.

Assuming the pulse excitation scheme described before, the operation of the inverter is now clear. Whenever the circuit is excited and the input is a *0*, the inverter moves to point *a*, stays there, and the output is a *1*. If the input is a *1*, there is sufficient current input to bring the inverter over the hill in the characteristic to point *b* and the output is a *0*. Note that for this latter case the output waveform will show a transient high voltage before reaching the low voltage *0* output. The next stage must therefore be powered by the next phase so that it is only interested in the voltage level at the *end* of the pulse (*i.e.*, the "output function" region). For that case such operation is satisfactory.

Note that the height of the *a*-hill in Fig. 6d depends on the value of R_1. By adjusting I_s to lie sufficiently below the crest of this hill and driving the circuit from a number of "elevated" outputs, one can obtain the logical function of a threshold gate whose output is inverted (*i.e.*, NOT, OR-NOT, AND-NOT, etc.).

Unilateralization

Unilateral operation can be defined as operation in which signals can propagate in one direction only. This is required to insure that spurious signals are not generated in the system. The most obvious way to insure unidirectional operation is to use normal diode rectifiers as coupling elements. Current between stages can then flow only in one direction. Other methods are possible in which the coupling between stages is resistive. For example, considering the inverter circuit driven by an elevated output, one can see that when the input to the inverter is *0*, there is essentially no current in the coupling resistor. When the input is *1*, there is a relatively high current in the coupling resistor. Thus, again, the current in the coupling resistor flows only in one direction. By reversing the positions of R_1 and D_1 (Fig. 6), so that the output is taken across the tunnel diode, one has a threshold gate with this unilateral property. Another unilateralization method is associated with the ability of the power supply to separate input and output functions in time. This scheme is effective for the balanced type of threshold gate. The circuit is receptive to an input signal only during a very short time (*i.e.*, the rise time of the power supply pulse), after which it "locks" into one state or the other.

Multilevel Circuits

Consider the inverter configuration (Fig. 6b) in which I_s is reduced so that a third stable operating point *c* exists. This is indicated by the dotted curves in Fig. 6 (a) and (d). Note that point *c* yields a *0* output. Assume that R_1 has been reduced sufficiently to make the height of the *a*-hill, in Fig. 6d, comparable with the height of the *c*-hill. Let the circuit have two inputs, driven from the normal outputs of other tunnel diodes. The circuit operates in the following fashion: If the two inputs are both *0*, *c* is a stable point and each time the circuit is excited the output is a *0*. If one of the inputs is a *1* while the other is a *0*, there is enough current input to make the circuit move over the first hill to point *a* where it is stable and the output is a *1*. When both inputs are *1*, there is sufficient current input to make the circuit move over *both* hills to point *b*, and the output is again *0*. Thus, the output is *1* only when the two inputs are different. This is the EXCLUSIVE-OR (modulo-2 sum) function.

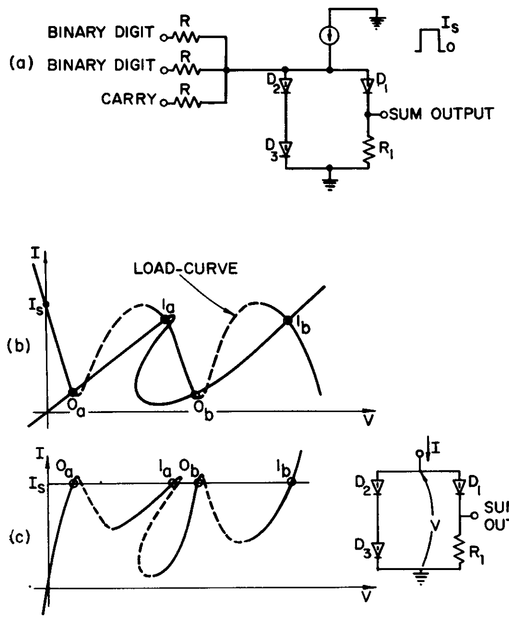


Fig. 7—(a) SUM output circuit for full adder. (b) Determination of operating points. (c) Composite characteristic.

One can also realize the SUM output for a full adder using a slightly different configuration. Consider the circuit shown inFig. 7a. The operating points can be found by plotting the load-curve, determined by I_s and the characteristic of the series combination of D_2 and D_3 (Fig. 5a), across the characteristic of D_1 in series with R_1 (Fig. 6a). This situation is depicted in Fig. 7b. I_s is chosen so that there are four stable intersections, labeled 0_a, 1_a, 0_b and 1_b. These correspond to *0* and *1* outputs as explained before. The composite V,I characteristic of the whole configuration of three tunnel diodes and the resistor is shown in Fig. 7c. Note the four intersections with the constant-current (horizontal) load-line I_s. They are also labeled appropriately.

There are three inputs corresponding to two binary digits and the CARRY from the previous digit. The circuit operates in the following manner: When all three inputs are *0*, each time the circuit is excited it moves to point 0_a and is stable there so that the output is a *0*. When one of the inputs is a *1* while the others are *0*, there is enough input current that the circuit moves over the first hill (Fig. 7c) to point *1a*, where it is stable, and the output is a *1*. When two inputs are *1*, the circuit moves over the first *two* hills to point 0_b and the output is again *0*. For three *1* inputs, the circuit moves over all *three* hills to point 1_b and the output is again *1*. Thus we have

Number of 1 inputs	SUM *output*
0	*0*
1	*1*
2	*0*
3	*1*

This fulfills the SUM function of a full adder. To realize the CARRY function, one simply uses a threshold gate, of the type described before, which has the same three inputs and which gives a *1* output when the number of *1* inputs is two or greater.

Storage

Since any negative-resistance element can exist in two stable states with the proper DC load-line, it is possible to use such a device to store information. The term "static storage" can be applied to this situation (DC load-line), because the voltage or current level of the negative-resistance device is fixed when it is storing a particular bit. This type of storage might be used in the memory of a digital computer.

Storage is also necessary in the logic section of a computer. Here another means of storage, known as "dynamic storage," is directly compatible with the three-phase pulse-overlap system. Dynamic circuit techniques are used in the SEAC and DYSEAC computers[10]. The method involves the circulation of information around a closed loop, so that a circulating pulse represents a *1* and no pulse circulating represents a *0*. In the original circuits using this technique, the pulse is introduced at one end of a delay line. At the other end it is amplified, reshaped and clocked and is then returned to the delay-line input. The delay-time is adjusted so that the pulse makes one trip around the loop in one clock period.

Consider the circuit shown in Fig. 8. All blocks under *A* are powered by phase *A*, all under *B* by phase *B*, etc. The block with the arrow represents a delay gate (one-input OR gate). This takes the place of the delay-line. Because of the phase relationship between the three power sources (see Fig. 2), it is possible to close the loop as shown. The circulation of a *1* or a *0* is thus made possible. The circuit has built-in amplification, reshaping and clocking. The particular circuit shown in Fig. 8 is a basic flip-flop, there *S* is the "set to *1*" input and *R* is the "reset to *0*" input.

[10] D. D. Elbourn and R. P. Witt, "Dynamic Circuit Techniques Used in Seac and Dyseac," *IRE Trans. on Electronic Computers*, pp. 2–9; March 1953.

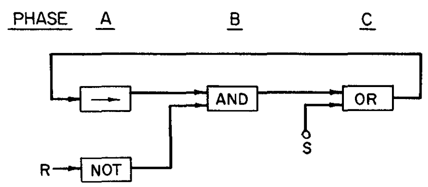


Fig. 8—Dynamic flip-flop.

The basic flip-flop can be included in more complicated storage circuits. Fig. 9a shows a binary counter and Fig. 10 shows one stage of a shift register. Other circuits involving dynamic storage techniques are possible.

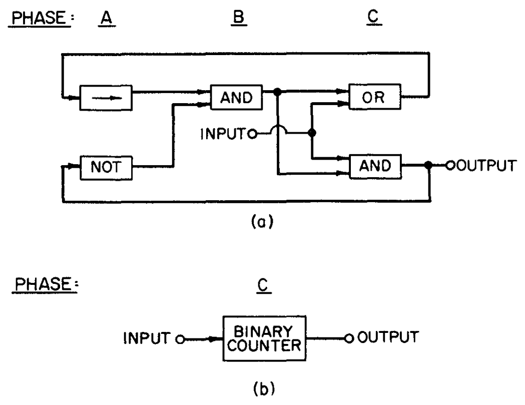


Fig. 9—Dynamic binary counter.

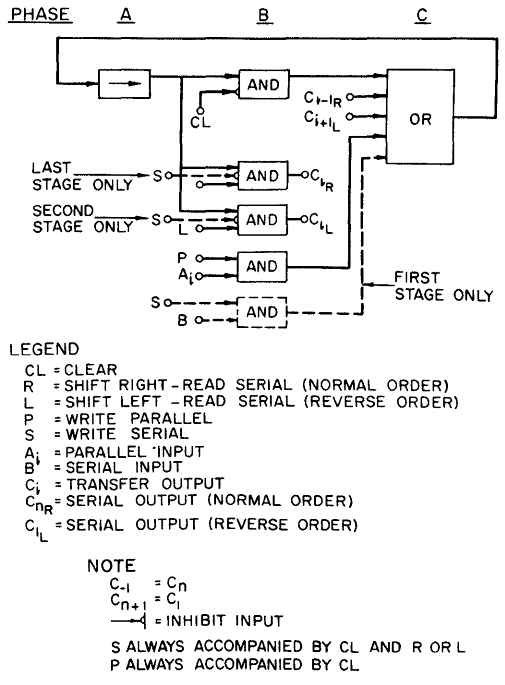


Fig. 10—Shift register stage.

Note that no time need be lost in obtaining an output from a dynamic circuit, even though the information stored is in the form of a circulating pulse. For example, the binary counter of Fig. 9a may be represented by a single block powered by the appropriate phase, as far as the input and output terminals are concerned. This is shown in Fig. 9b where the input comes from a circuit powered by phase B, the binary counter is considered powered by phase C and the output goes to a circuit powered by phase A.

Experimental Verification

In order to investigate the operation of tunnel diode logic circuits in a small sub-system, one cell of a simple experimental arithmetic unit was constructed and tested. A block diagram of the cell is shown in Fig. 11. All of the fundamental logic circuits, including dynamic storage, are evident. The cell contains a storage loop, a full adder and auxiliary read-in and read-out gates for shifting right and left, complementing the input from memory, and reading out to memory.

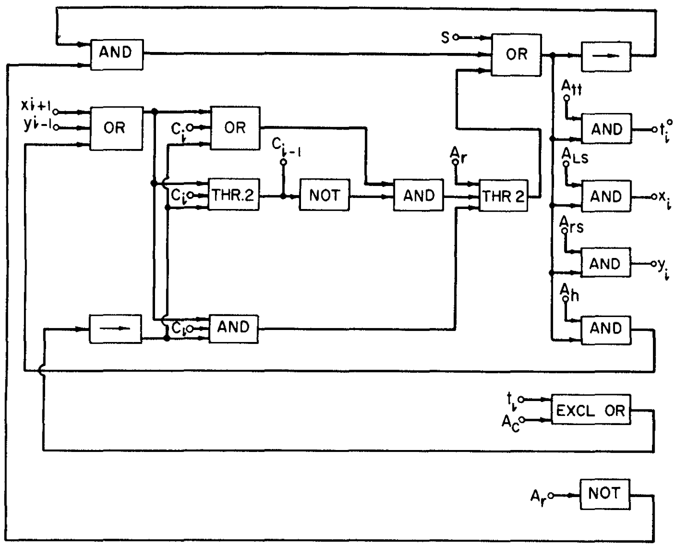


Fig. 11—Block diagram of experimental unit.

The schematic diagram for the unit is given in Fig. 12. Fig. 13 contains photos of the complete experimental circuit. The unit contains 27 tunnel diodes. Resistive coupling is used throughout. It is powered from a transistorized power supply which delivers a three-phase, 1-mc, 10-volt square-wave. This repetition rate was chosen to most easily demonstrate the fundamental principles involved. The inputs to the system are DC levels simulating the output voltages of the tunnel diode (*i.e.*, *0* = 50 mv, *1* = 450 mv), with the correct internal impedance.

Typical waveshapes, taken across one of the diodes in the storage loop, are shown in Fig. 14. (a) shows a circulating *1*, after the loop has been set and (b) shows a circulating *0*, after the loop has been reset. One can also make the bit stored in the loop alternate between *0* and *1* as shown in (c). This is accomplished by making $A_r = 1$, so that the storage loop is cleared and the SUM output of the full adder is gated into the loop, by letting $A_h = 1$, so that one of the inputs to the adder becomes the bit presently stored in the loop, and by allowing any one of the other inputs to

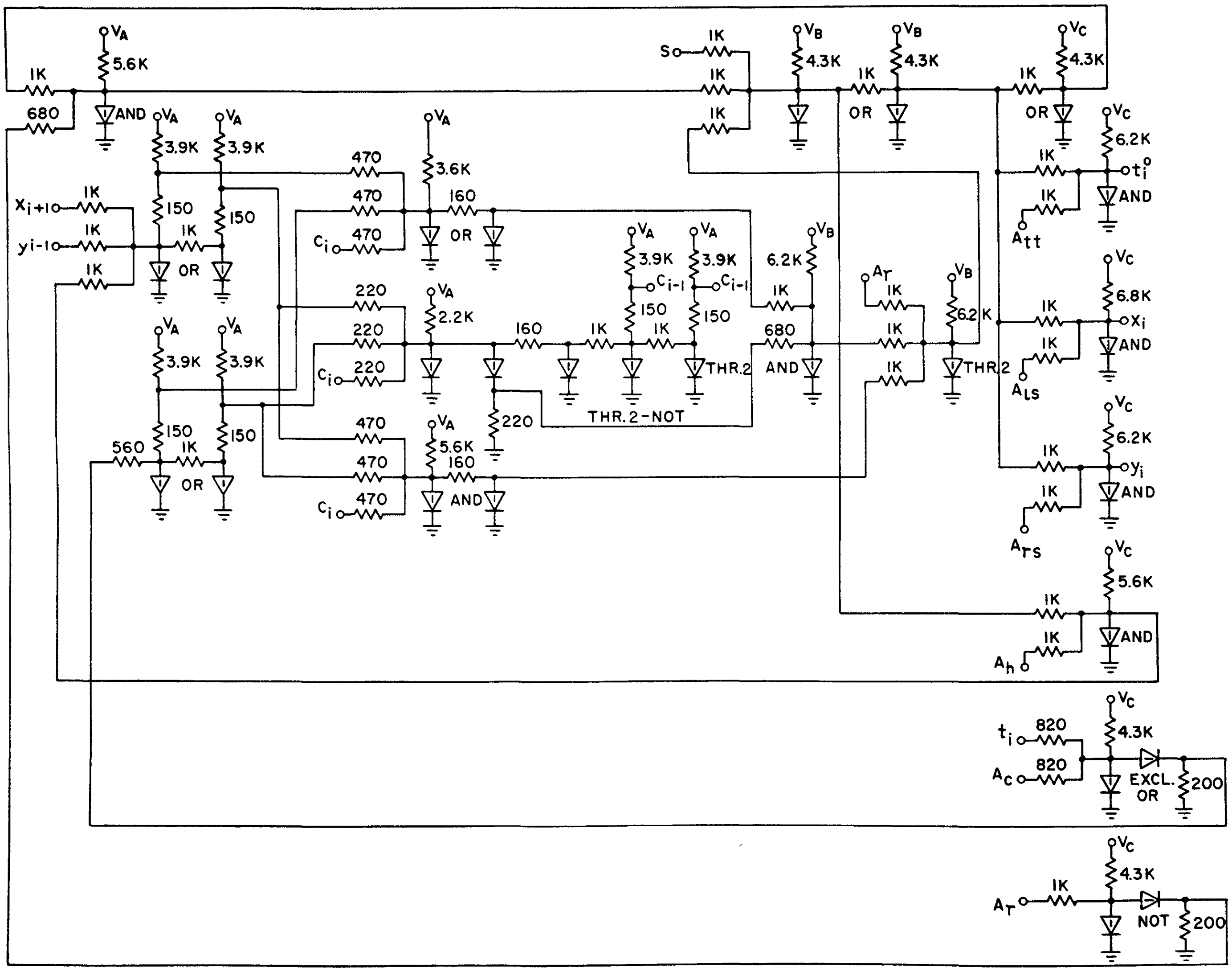
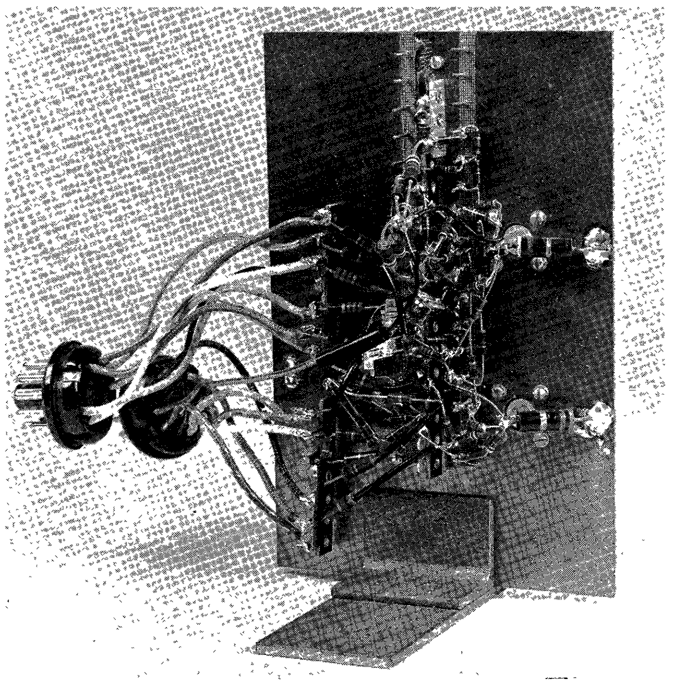


Fig. 12—Schematic diagram of experimental circuit.



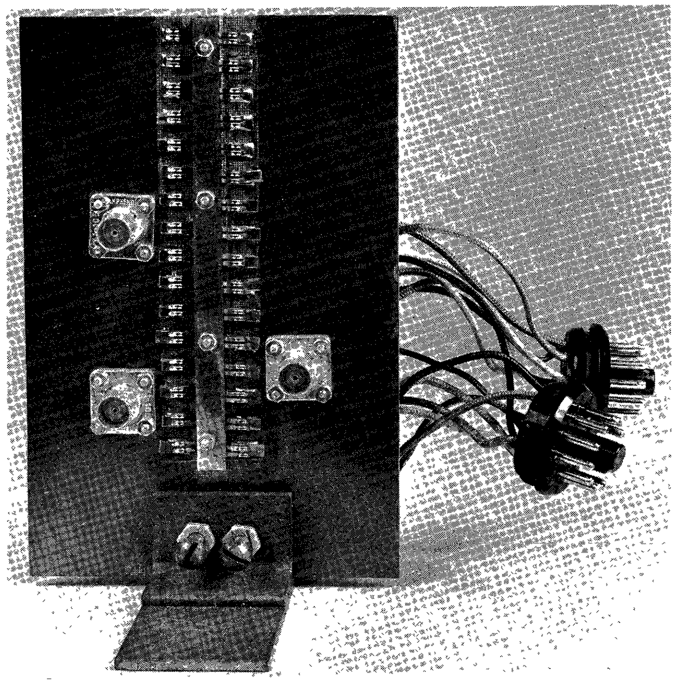


Fig. 13—Front and rear views of experimental cell. (Tunnel diodes are mounted under finger contacts.)

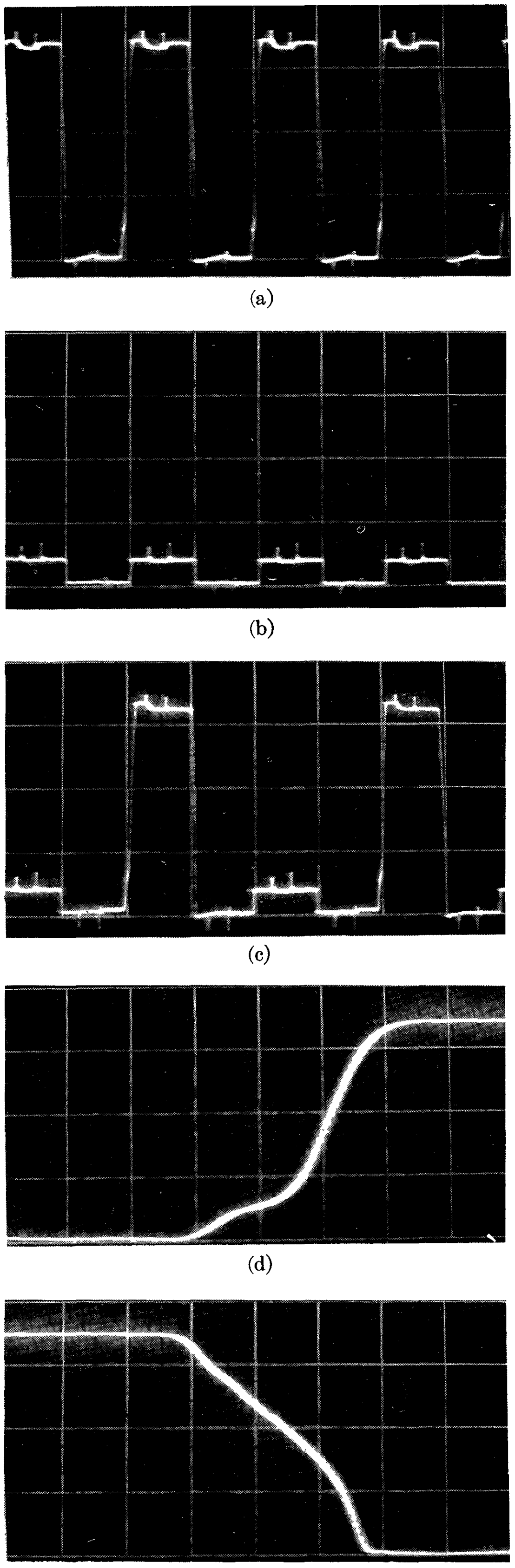


Fig. 14—Typical waveforms. (a) Circulating 1. (b) Circulating 0. (c) Alternating 1 and 0. (d) Rise time. (e) Fall time. Time goes from left to right. Vertical scales are 0.13 v/div with base-line at bottom. Horizontal scales are 0.5μs/div for (a), (b) and (c) and 20 mμs/div for (d) and (e).

the adder to equal *1* (*i.e.*, either C_i or A_c or $t_i = 1$), so that the sum output becomes the complement of the bit presently stored. Thus, each cycle the complement of the bit previously stored is read into the loop and the stored bit alternates as shown.

The peak currents (I_0) of the tunnel diodes used in the experimental cell range from 1.9 to 2.6 ma. The capacity of each diode is of the order of 100$\mu\mu$f. Peak-to-valley current ratios vary between 5 and 8. The currents from the power supply to each of the logic circuits were adjusted for proper operation. Observed switching times (see Fig. 14d and e) are of the order 50 mμs.

The experiment demonstrates a number of important facts concerning tunnel diode logic circuits. First, it demonstrates reliable operation of all fundamental logic circuits in a realistic system. These circuits include OR, AND, THRESHOLD, NOT and EXCLUSIVE-OR. Second, it demonstrates that such circuits can supply logical gain. For example the OR gate in the dynamic storage loop has a fan-in of 3 and a fan-out of 5. The circuit contains two tunnel diodes in cascade. Third, it demonstrates agreement between rough estimates of switching time, based on the time constant of the device (capacity time magnitudes of average negative resistance), and the actual switching time.

Acknowledgments

The author wishes to thank A. W. Lo, G. B. Herzog, H. S. Sommers, Jr., J. C. Miller and A. G. Samusenko of RCA Laboratories and Professors E. J. McCluskey, Jr. and W. H. Surber, Jr. of Princeton University for many interesting discussions and helpful suggestions.

This work is part of a larger effort which is summarized in a paper by Dr. Jan Rajchman.[11]

Appendix

I. DC power supplies, monostable operation, zero reactance, case II:

The voltage-controlled case will be considered. Dual treatment can be given to the current-controlled case. Referring to Fig. 1c, in order to insure monostable operation for all R_L, one must choose I_s and R_1 such that only one intersection occurs for $R_L = \infty$. Assuming R_L is very large, a particular limiting case is shown by the solid load-line in Fig. 15. For a given I_s, the negative reciprocal of the slope is $R_{1_{\max}}$ for monostable operation. Then, for any finite R_L, the load-line changes as shown by the dashed line *a*. For a square pulse of input current of magnitude Δi and width Δt, we have that the energy input $= \Delta v \, \Delta i \, \Delta t$, where these values are depicted graphically in Fig. 15.

[11] J. A. Rajchman, "Solid State Microwave High Speed Computers," *Proc. EJCC*, this issue.

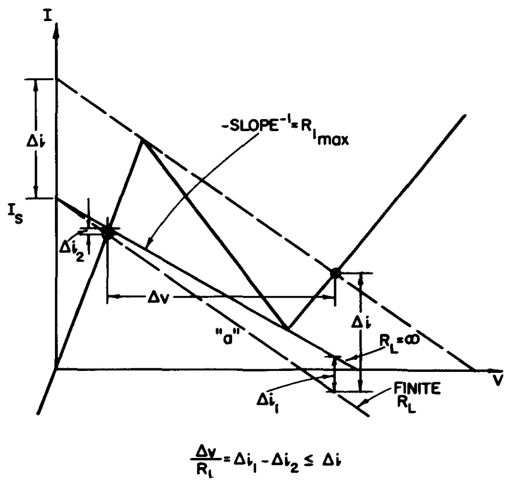

I
V
Δi
I_s
$-\text{SLOPE}^{-1} = R_{l\,max}$
Δi_2
Δv
"a"
Δi
$R_L = \infty$
Δi_1
FINITE R_L
$\frac{\Delta v}{R_L} = \Delta i_1 - \Delta i_2 \le \Delta i$

Fig. 15—Graphical comparison of Δi and $\Delta v/R_L$.

The energy output is $\Delta v^2 \Delta t/R_L$. Thus, we must compare $\Delta v/R_L$ to Δi to determine which increment of energy is greater. These are also found graphically in Fig. 15. An examination of the geometrical constructions involved shows that if the conditions stated above are satisfied, $\Delta v/R_L \le \Delta i$ for all R_L. Therefore, energy input $\ge$ energy output.

II. DC power supplies with reactance vs. pulse power supply:

Again, the voltage-controlled case will be considered. Comparison is being made between the two circuits shown in Fig. 16. C is the sum of the stray capacity plus the capacity inherent in the negative resistance device. The path of operation looking into the parallel combination of C and NR is assumed to be approximately the dashed path shown in Fig. 16.

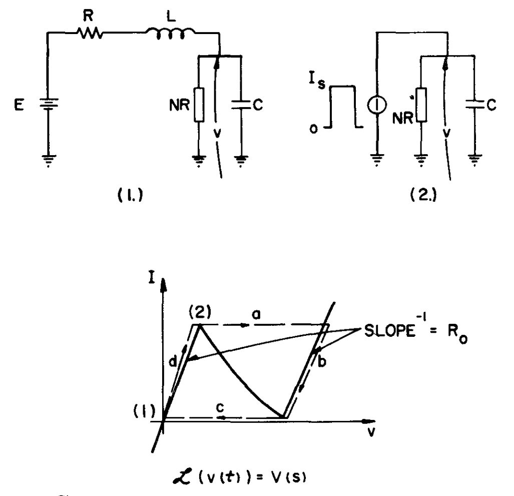


Fig. 16—Comparison of monostable and bistable-with-reset modes. Assumed path of operation.

This will be true, in the case of circuit (1), if L is sufficiently large. Under these conditions, one can assume that the transition times (paths a and c) are comparable for the two circuits. If L is not large enough, the transition time of circuit (1) can be considerably larger than that of circuit (2), and in addition, the path followed is no longer horizontal.[12]

We wish to compare the recovery times (paths b and d) of both circuits. Specifically, let us calculate the time to go from point *1* to point *2* (path d). For circuit (2) the rise time is approximately 2.2 R_oC, since the voltage follows a simple exponential with an R_oC time constant. For circuit (1), using Laplace transform techniques, the transform of v is given by

$$V(s) = \frac{E_oR_o}{s[R_oLCs^2 + (L + R_oRC)s + R_o + R]} \quad (1)$$

Assuming that

$$\frac{4}{LC\left(\frac{1}{R_oC} - \frac{R}{L}\right)^2} . < \sim 0.4 \,, \quad (2)$$

which is equivalent to assuming that no oscillations occur, one finds that the roots of the characteristic equation are approximately

$$\begin{aligned} s_1 &= -\frac{R}{L} - \frac{1}{L/R_o - RC} \\ s_2 &= -\frac{1}{R_oC} + \frac{1}{L/R_o - RC} . \end{aligned} \quad (3)$$

The approximate solution is therefore

$$V(s) = \frac{A}{s} + \left[\frac{As_2}{s_1 - s_2}\right]\frac{1}{s - s_1} + \left[\frac{As_1}{s_2 - s_1}\right]\frac{1}{s - s_2} \quad (4)$$

where

$$A = \frac{E_oR_o}{R_o + R} . \quad (5)$$

An examination of this solution shows that the time constant of the dominant exponential is always greater than R_oC for values of L and R consistant with the approximation (*2*).

Thus circuit (1) has the lower maximum repetition rate.

Discussion

J. H. Felker (*AT&T*): What speed were you getting on the pulses you showed and what were the valley and peak points of the negative resistance characteristics?

Mr. Lewin: The transistorized power supply delivered a three-phase 1-mc square wave with rise and fall times of about 15 to 20 milliseconds. As I mentioned before, the tunnel diode switching times (for the units used) were about 50 milliseconds, so that this was the rise or fall time of the pulses shown. The tunnel diodes used were early experimental ones with a nominal peak current at around

[12] W. J. Cunningham, "Introduction to Non-Linear Analysis", pp. 106–114; McGraw-Hill, 1958.

2 milliamps. The peaks varied almost plus or minus 20 per cent. The valley points also varied such that peak-to-valley ratios were anywhere from about 5 to 8. The diodes could be considered as relatively "low speed" diodes.

R. A. Vaenel (BTL): Which do you consider the speed-limiting parameters in tunnel diodes? Please distinguish between negative portion of characteristic and positive portions.

Mr. Lewin: There are two physical processes to consider — tunneling and injection. These can be treated separately. Throughout the reverse-bias region and well into the forward-bias region, tunneling is the primary process taking place. This is a majority-carrier phenomenon and is therefore very fast; so that, for example, switching from low voltage to high voltage ought to be quite good. For sufficient forward-bias, that is from around the valley region to greater forward-bias, injection takes over and the V,I characteristic follows that of a normal forward-biased junction. This region is characterized by parameters similar to those of any forward-biased junction. Since the primary phenomenon is now a minority-carrier effect, it should not be as fast. Note, however, that storage times for the highly forward-biased tunnel diode can be very short because of the extremely small lifetime of the stored minority carriers — due to the very high doping.

The switching times naturally also depend on the external circuit parameters. For example, in the circuits described here, we are really interested in switching time from low voltage to high voltage. The circuit is *forced* from high back to low by a strong reset signal.

Mr. Vaenel: Do you have available regular diodes to work in conjunction with tunnel diodes?

Mr. Lewin: I assume this means conventional diodes for use as coupling elements. This is the most obvious way of unilateralizing the system — to couple with normal rectifiers so that currents can only flow in one direction. The trouble one runs into here is that, for tunnel diodes with sufficiently high peak current value, the forward impedance of any conventional diode would be so high that, at the appropriate voltage drop (say 0.4 volts), the coupling diode could not conduct the desired current increment. In addition, if one were to use conventional diodes as the *only* coupling elements, a small change in this voltage drop would mean a relatively large change in current increment — due to the nonlinearity of the coupling-diode forward-characteristic. Since the system relies on fixed current increments, this might not be tolerated. It is most advisable to use conventional diodes in series with resistors as coupling elements. Then, the diodes establish directionality while the resistors establish the current increments. This type of coupling can only be used with tunnel diodes with sufficiently low-peak current value. For example, in the system described in this paper, since 2 milliamp tunnel diodes were used, one could easily have coupled with conventional diodes and series resistors. For much higher current tunnel diodes, this may no longer be possible.

Mr. Vaenel: What type of stabilizing means do you anticipate for high points?

Mr. Lewin: I assume that this refers to the uniformity of tunnel diode characteristics required. This is one of the main problems. As you would expect, we require characteristics that are reasonably well-matched. By well-matched, I mean diodes whose essential parameters are within ten per cent, and preferably five per cent, of nominal values. Now in addition we require something of this order or better on the power supply parameters. Of course, if we use a square-wave generator, this becomes a pretty tough job. However, this is not the only way to do it. All we require is bistable-with-reset operation. The power requirements for this type of operation can be furnished in many ways. We could have a well-controlled d-c supply for the "holding power" unit, to establish the bistable operation, and a second reset source to supply the reset signal.

L. Thayer (Food Machinery): Do you believe cost and reliability of tunnel diodes will make them a possible replacement for transistors in computers?

Mr. Lewin: I think that eventually tunnel diodes will be fabricated to quite accurate tolerances simply because, in the short time that people have been working with them, tremendous advances have been made toward realizing high yields of well-matched characteristics. These tunnel diodes will also be high-speed and, hopefully, low-cost units. Of course, you won't be able to simply pull out a transistor from a circuit and plug-in a tunnel diode. The transistor is a unilateral device. The tunnel diode is not. I believe, though, that in many applications, notably in the digital field, one can devise tunnel diode logic circuits to replace transistor logic circuits and obtain, for the same cost, a much higher speed.

G. E. Saltus (Bell Telephone Labs.): What was the total power dissipation in your experimental adder?

P. E. Stuckert (IBM): Can you give an estimate of three-phase drive power required by those circuits?

Mr. Lewin: The tunnel diodes in the system dissipate a very small amount of power. Assuming a single 2-milliamp unit always in the high voltage state when energized, we come up with a power dissipation in the diode of less than half a milliwatt. However, for the system described in this paper, a relatively large amount of power was dissipated in resistors establishing the current source. For example, the source used was a 10-volt-peak square wave converted to a current square wave of about 2-milliamp peak. This gives a value of approximately 10 milliwatts per stage dissipated in the current source resistor, and this is not necessarily excessive. It is, though, by far the major power dissipation in the system. Now a source of about 3 volts peak would have been sufficient for a reasonable current source. This reduces the power dissipated per stage to about 3 milliwatts. Of course, the power dissipated per stage is directly proportional to the peak current of the tunnel diodes used, so that this could rise appreciably for higher current tunnel diodes. Note that the method used is not the only way to get a square wave of current. For example, one could return the tunnel diodes through a small capacitor to a triangular wave voltage source. This could also furnish the current square wave but without the relatively high dissipation of real power.

N. F. Gianola (BTL): Over what ambient temperature range do your matched diodes maintain sufficiently similar characteristics for use in threshold circuits?

Mr. Lewin: This has not been fully determined yet. The part of the tunnel-diode characteristic that is appreciably temperature dependent is the injection or high voltage region. This is temperature sensitive in the same way that any conventional forward-biased diode is sensitive. For a germanium unit at a given current, a typical number for the rate of reduction of the voltage in the high state is about 1 mv per degree (centigrade) rise in temperature. Most of the tunneling region is essentially temperature independent, so that for some of the threshold circuits described, the threshold would not change with temperature.

R. Turner (Philco): Please describe your three-phase clock and method of obtaining it?

Mr. Felker: The other questions have to do with the practicality of generating high power that would be required in operating a very large and complex computer system using these tunnel diodes. Would you care to comment on the problem of supplying them having perhaps tens or thousands of tunnel diodes in them?

Mr. Lewin: First, to answer the question on the power supply used. The outputs were taken from three transistor binary counters, each driven by a 2-mc input. The clock for the power supply was a 2-mc astable multivibrator. One counter was driven directly from this clock source. Each of the other two counters was driven by a monostable circuit whose input was connected to the clock source. The pulse width of the monostable circuit was adjusted to shift or delay the output 1-mc square wave by the appropriate amount with respect to the output of the counter driven directly from the clock. This resulted in a three-phase square wave. It was simply a convenient method of making the experiment.

Now turning to future power supplies, this method may not be the best. As I mentioned before, one possibility for obtaining bistable-with-reset operation is to use a d-c source plus a reset source which delivers a train of negative pulses. These reset pulses are not critical with respect to width or amplitude. The magnitude of the pulses must only be greater than a certain minimum necessary for reset. If you will recall the tunnel diode V,I characteristic, you see that for a larger reset pulse the diode is forced into the reverse-bias region, and this is also a low voltage region so that such operation is perfectly satisfactory.

Naturally, at the higher speeds we will require higher speed diodes. At present, fabrication techniques are such that these can most readily be furnished in large numbers by keeping the area of the junction the same and just increasing the doping, resulting in a much higher current diode with approximately the same capacity as the "low speed" units. A large system would then require many amperes from the power source, and this becomes a very serious problem. Now hopefully, as the result of experiments now going on, we will be able to use much lower current tunnel diodes with reduced area, and thus smaller capacity to obtain the same speed. That is, by reducing the area of the junction and increasing the impurity concentrations one can increase the speed of the tunnel diode without changing its peak current value. Eventually, for example, we may have a one milliamp tunnel diode which could switch at the same speed of present 20 ma diodes. This would naturally appreciably reduce the magnitude of the problem of supplying power to many units.

H. Hellerman (IBM): What sensitivity to noise may be expected from these circuits, especially ground noise?

V. J. Sferrino (Lincoln Lab.): Do you envision many problems of noise triggering with the advent of larger, faster systems?

H. P. Peterson (Lincoln Lab.): Do you anticipate tunnel diodes with a negative resistance region at higher voltages, like 5–10 volts, to lessen noise problems?

Mr. Lewin: As far as the noise problem is concerned, I am really not familiar with the measurements on the tunnel diode noise figures. I do know that the primary noise source is shot noise associated with the junction, the shot noise power being proportional to the junction current. As you noticed on the scope pictures, the little spikes on the waveforms were due to coupling to the other phases of the power supply. From the pictures of the experimental set up, you can see that this was a very crude breadboard model and no care was taken to design for minimum crosstalk, etc. Much more careful design with a good ground system would be needed at higher speeds. Preliminary high speed tests in the millimicrosecond range indicate as you would expect, that a very good ground system is required for reliable operation. It is too early to more clearly define the design requirements for very high-speed operation at this time.

To answer the question concerning tunnel diodes with larger voltage swings — work is now going on with materials other than germanium. Since the high voltage region of the tunnel diode characteristic is the same as that of a normal forward-biased junction, the voltage of the high state depends on the material used. For germanium, this is at about 450 mv, for silicon about 700 mv, for gallium arsenide about 900 mv, etc. Thus, in the future, tunnel diodes should be available with various voltage swings. Since this forward voltage drop must be less than the width of the forbidden gap of the material, with present technology one cannot expect voltages as high as 5 to 10 volts.

G. A. Barnard (Ampex): What have you found to be the current high-speed switching rates of circuits as those you have shown? What speeds do you see within five years?

D. Baker (BTL): What is the expected maximum prf for resistive coupled diode logic circuits? Approximately what value of power supply regulation is required?

Mr. Lewin: As I mentioned before, the diodes that I used were 2-ma-peak-current units with 100-$\mu\mu$f capacity, and these switched in about 50 mμs. Experiments with 20-ma diodes with about the same capacity show switching times of about 4 or 5 mμs. One can now extrapolate to the kinds of diodes required for higher speed operation. Obviously, keeping the same capacity and going to higher currents is not the best way to achieve this. What we need are much lower capacity units, and these are currently being developed. I really do not have enough information to make any useful predictions about the future.

The power supply regulation problem has already been mentioned. Using a d-c-plus-reset scheme, we would require a d-c source with an output which did not vary by more than 5 per cent from the nominal value. A more accurately controlled source would of course be welcomed.

W. Lawrence (IBM): How does the tunnel diode's capacity vary from its low to high state?

Mr. Lewin: I believe that the depletion layer capacity variation is the same as that for any forward-biased abrupt junction. That is, the capacity increases with forward-voltage. Normally, a value of capacity is determined by measuring the maximum frequency of oscillations when the diode is biased in the negative-resistance region. This gives a value which, to a first approximation, may be attributed to the whole of the forward-bias region of interest.

P. Smith (General Transistor): Have you done any work on three-terminal negative-resistance devices, and, if so, what speeds of operation were considered?

N. F. Gianola (BTL): Can a third control electrode be added?

Mr. Lewin: You are probably aware of the fact that, at the recent Washington Electron Device Conference in November, a paper was presented concerning experiments on replacing the emitter junction of a transistor with a tunneling junction. I really cannot see how you would be able to combine the tunneling action, which requires high conductivity material (implying short lifetime of minority carriers), with transistor action, which requires long lifetime of minority carriers in the base region. But of course, I am not an expert on fabrication of semiconductor components. I do know that work is going on to consider the possibility of fabricating a three-terminal device utilizing the tunnel principle. I think such a device would be extremely useful, but I do not have any quick ideas on how to make it.

Deposited Magnetic Films as Logic Elements*

A FRANCK†, G. F. MARETTE† AND B. I. PARSEGYAN†

THE USE of thin magnetic films as storage elements is well known. Several papers on the subject have appeared in the literature, particularly in recent years[1]. Less emphasized, perhaps, is the use of magnetic films as logic elements. The authors' study in this area has revealed that film elements are both flexible and versatile as logic devices.

This paper describes two modes of film-core operation, namely reversible-rotation and saturable-transformer action, as they pertain to a particular circuit. Also described are certain principles of array logic. These principles involve writing multiple copies of a word in a film-core array. Then, by the proper arrangement and selection of sense lines linking parts of these copies, some desired result is obtained from the array. This approach makes it possible to perform in one or two clock periods operations that have previously required many clock periods. The application of magnetic films as logic elements is illustrated by a scale-factoring device whose function is to find the most significant digit in a binary word, shift that word to the left until the most significant digit is in a position immediately to the right of the position reserved for the sign bit, and record the number of places shifted in an auxiliary register. The methods and advantages of accomplishing these operations with deposited magnetic film-cores are given in detail in the paper.

In some of the subsystem designs investigated, where comparisons between film-element logic and its conventional counterparts were made, definite reductions in both the required number of semiconductor components and the operating time were observed. For instance, throughout all of the designs, the use of separate NOT elements was easily avoided by appropriate wiring and biasing of film cores. Use of separate OR elements may also be eliminated by appropriate wiring between film elements. This principle and the component savings it produces are illustrated by the encoder that is described in this paper (as part of the scale-factoring device). By interconnecting film elements to form functional logic arrays (such as the shift matrix described below), great gains in speed of entire sequences may often be realized. These logic advantages — together with such properties as small size, high reliability, low power requirements, relative insensitivity to environment, and low cost — make magnetic film elements very desirable as logic devices.

* This work was carried out under the sponsorship of the Bureau of Ships, U.S.N.

† Remington Rand Univac, St. Paul, Minn.

[1] A list of references on this subject appears in an article by A. J. Kolk and J. T. Doherty, "Thin Magnetic Films for Computer Applications," *Datamation*, vol. 5. pp. 8–12; September/October, 1959.

Logical Properties of Film Elements

This section introduces those logical properties of film elements that are used in the scale-factoring device. Specifically, these four ways of using the logical properties of film elements are described:

a. AND logic using the reversible-rotation mode of operation;
b. AND logic using the saturable-transformer mode of operation;
c. Inverter logic using the saturable transformer mode of operation;
d. Functional array logic.

AND *Logic (Reversible-Rotation Mode)*

Fig. 1 illustrates a method of obtaining AND logic using a film element in the reversible-rotation mode of operation. If inputs to such a film element are x and y, respectively, then the output is xy, as shown.

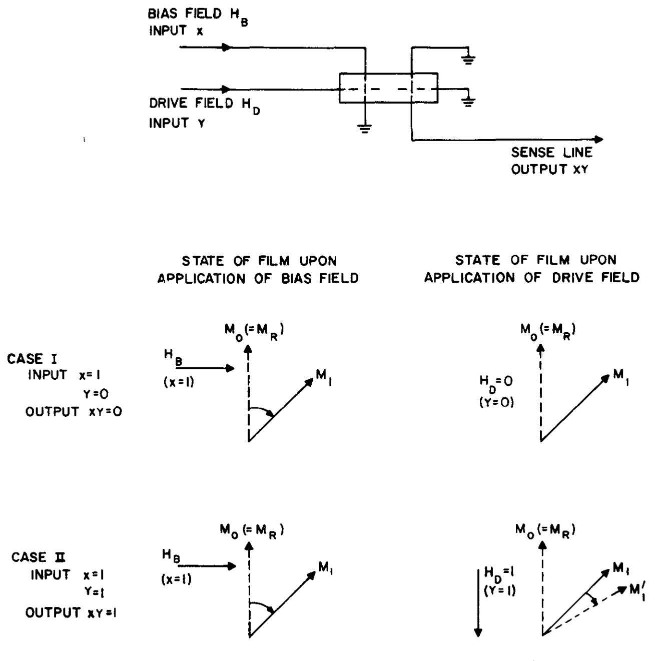


Fig. 1—AND logic (reversible-rotation mode).

The "0" state of the film core, represented by the magnetization vector M_O, is made to correspond to the remanent state of the film core, M_R. A bias field, H_B, transverse to M_O, corresponding to the logical input x, rotates the vector M_O through an angle to a position shown as M_1. Subsequent application of a drive field, H_D, corresponding to a logical input y, in a direction antiparallel to the vector M_O, further rotates the vector M_1, altering the state of the film core to M'_1. (In the reversible rotation mode of operation, application of bias field H_B and drive field H_D is time-sequenced so that the biasing precedes the driving.) Change of the magnetization of the film core from state M_1 to M'_1 induces a voltage on the sense line, corresponding to a logical output of "1". No output is obtained unless both the bias field H_B and the drive field H_D are present, as illustrated by the vector diagrams in Fig. 1. Logically, then, the output xy is "1" only if inputs x and y are both "1".

AND *Logic (Saturable-Transformer Mode)*

Fig. 2 illustrates a method of obtaining AND logic using the film core as a saturable transformer. In this mode of operation, the film core is initially biased to one of its remanent states of magnetization in the hard direction. Its state is then caused to change or not to change in a direction of high permeability, depending upon certain control conditions.

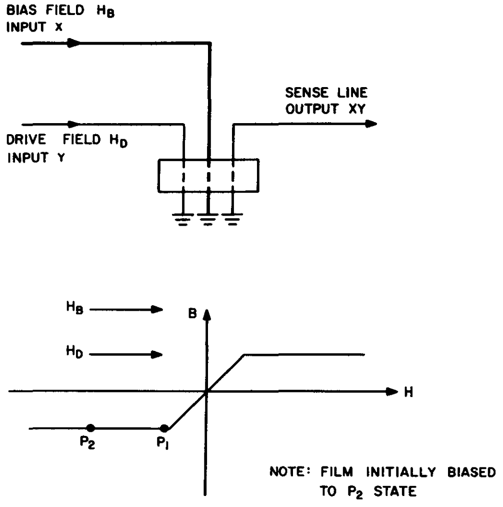


Fig. 2—AND logic (saturable-transformer mode)

Fig. 2 shows the film core as initially biased to the P_2 state. (Means of effecting this initial bias are not shown in the figure.) A bias field H_B, corresponding to logical input x, further biases the film core to state P_1. Application of a drive field H_D, corresponding to logical input y, then causes a change in the state of the film core. This change is in the steep region of the *B–H* diagram, so that an output voltage is induced in the sense line. This voltage corresponds to a logical output of x AND y. A "1" output is obtained only if the bias field H_B and the drive field H_D are both present.

Inverter Logic (Saturable-Transformer Mode)

Fig. 3 shows a simple method of obtaining logical inversion (*i.e.*, negation) using a film core in the saturable-transformer mode of operation. The film core shown in the figure is initially biased to the P_1 state. (Means of effecting this bias are not shown in the figure.) A bias field H_B, corresponding to logical input x, biases the film to the P_2 state. Application of a drive field H_D, in a direction opposite to the bias field, then merely biases the film core toward P_1. It is, however, not of sufficient strength to drive the film core into the steep portion of the *B–H* curve. Consequently no voltage is induced on the sense winding; this corresponds to a NOT x logical output. If, on the other hand, the bias field H_B were absent, meaning a NOT x input, the film core would remain in its original biased state at P_1. Application of a drive field H_D would then induce a voltage on the sense winding which would correspond to output x.

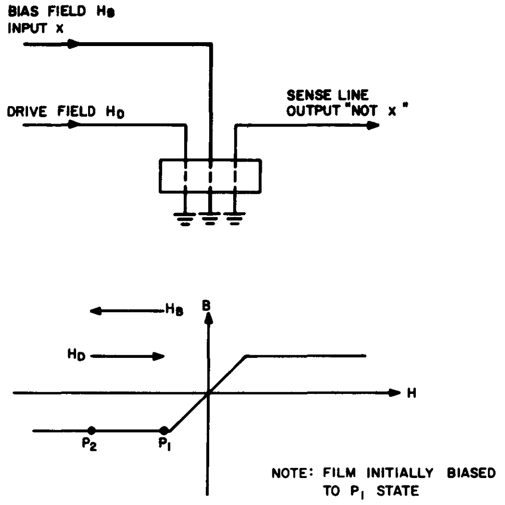


Fig. 3—Inverter logic (saturable-transformer mode).

Functional-Array Logic

A very powerful feature of magnetic film elements is their adaptability to a technique of logic described as *functional-array logic*. Use of this technique results in a great saving in time for many operations that may be sequential in nature. An example of a sequential operation is a shifting operation where the total time to shift a number is dependent upon the number of shifts required. The accomplishment of shifting by functional-array logic is explained in detail in this paper. In general functional-array logic may be thought of as an arrangement of information in an array based upon an input word or bit configuration for the purpose of accomplishing a specific logical operation in one step.

In the preceding sections on the saturable-transformer and reversible-rotation modes of operation, the logic of the individual films was presented. Because of their size, it is possible to assemble these film elements in compact arrays and to bias and drive many film elements simultaneously without appreciable time delays or power losses. One such arrangement is illustrated in Fig. 4, which is an

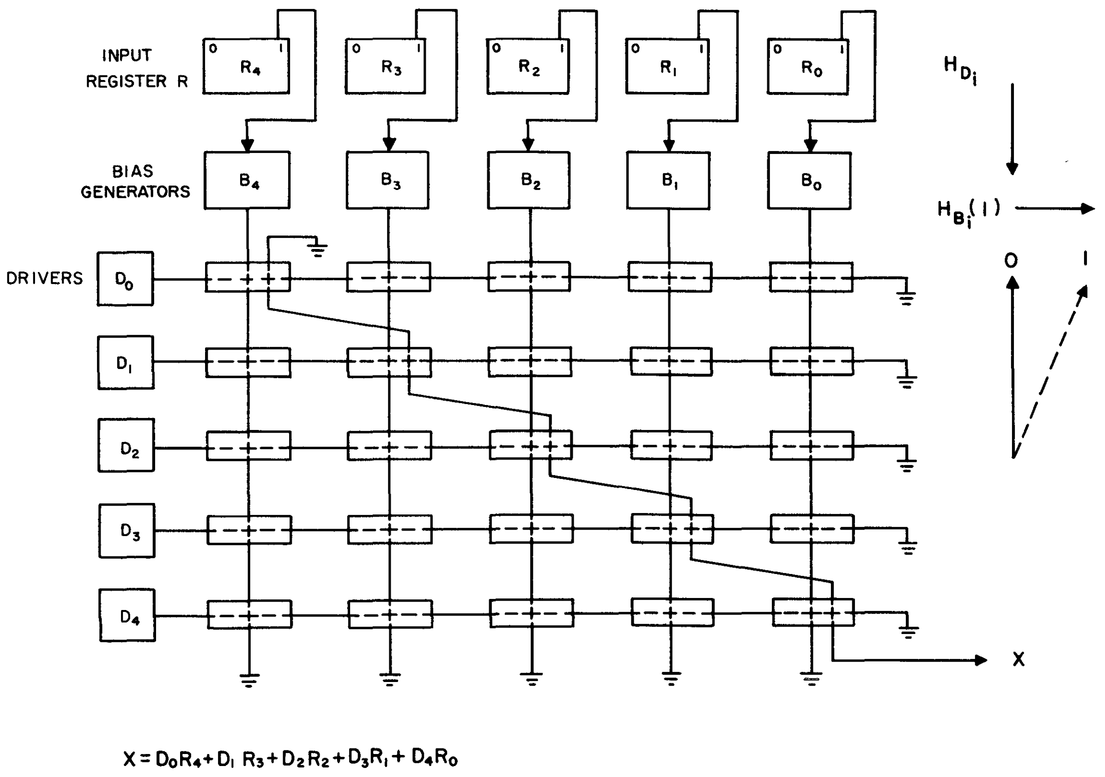


Fig. 4—(a) Film magnetization directions. (b) An example of functional-array logic.

example extracted from one of the arrays to be presented in a later section.

The function of this array is to sense for an information bit in a position within the input word. The input word is contained in the input register R, and each of the bias generators (B_0–B_4) supplies a bias field to the film element below it if the corresponding input-register stage (R_0–R_4, respectively) contains a "1". This bias field rotates the magnetic vector to the "1" position as indicated in Fig. 4(a). If a drive field H_D is applied to a film core thus rotated, an output is induced on a sense line linking that film core.

If driver D_0 is energized after the array is biased, and there is a "1" in input register-stage R_4, an output is obtained on the sense-line X. This simple arrangement could be used as a sign test. The other drivers, D_1–D_4, could also be initiated singly to determine whether input register stages R_3–R_0, respectively, contain binary "1"s. Another way of using the same functional array depends on initiating all drivers simultaneously so that a "0" output indicates that the input world is all "0"s. Various effects can be produced with functional arrays by varying the wiring of the sense, drive, and bias lines. Applications of two of these effects are discussed under "Circuit Descriptions."

Circuit Functions

The logical operation that will be described to illustrate the utilization of magnetic film elements is that of scale factoring of a data word. In certain number representations this operation is also referred to as "normalizing a number." In both scale factoring and normalizing, a binary word is examined to determine the location of its most significant information bit. The entire word is shifted until this bit is in the highest order non-sign position, and the amount of this shift is stored in an auxiliary register. If the word is given in a complement representation, such as one's or two's complement, the operation is referred to as the process of scale factoring. On the other hand, if the word is represented in the sign and magnitude form, the operation is referred to as normalizing.

For the purposes of this description the one's complement representation is used. It follows that the leftmost bit of a binary word is the sign bit; "1" for negative numbers and "0" for positive numbers. Therefore the most significant information bit is the leftmost "0" for negative numbers and the leftmost "1" for positive numbers.

Circuit Descriptions

The scale-factoring operation consists of three separate operations that are first treated separately in the description that follows and then integrated into one unit in the last portion. The three operations and the order in which they are presented are as follows:

a. Location of highest-order information bit;
b. Shifting;
c. Encoding the amount of shift.

Location of Highest-Order Information Bit

In conventional logic circuits, the process of determining the location of the highest significant information bit of a binary word is a time-consuming operation. The method normally used depends on shifting the binary word one position at a time in the direction of most significance. After each shifting operation, a check is made for a difference between the sign bit and the bit occupying the most significant position. If the bits are alike, the word is shifted again and the check repeated. The first time that the bits are found to be unlike, the word is in its proper position. The amount of shift that has been accomplished is then read from a counter that has been counting the number of shifts. The number of sequential steps and therefore the time for this operation can be large, especially where the word size is large and the highest order information bit appears in one of the lower order positions.

In the preliminary section on functional-array logic, it was shown that logical operations could be performed using functional arrays. An array of this type is illustrated in Fig. 5; its function is to determine the most significant information bit in the word "00010". The word is arranged in a 5 × 5 bit array such that one row contains a negative copy of the word and four rows contain positive copies of the word. Sense lines S_0–S_3 are arranged in such a manner that they couple one bit in the negative row and one bit in each column to the left of that position. The figure shows that one and only one sense line may link bits that are all in the "0" state and that this sense line has a direct relationship to the location of the most significant information bit. The sense lines to the left of this position will always link a bit in the "1" state because of the negative row, and the sense lines to the right will always link a bit in the "1" state because of the column immediately below the highest-order "0" in the negative row. In the example of Figure 5, sense line S_2 is the only sense line linking bits that are all in the "0" state. This condition dictates that a shift of two positions is required to properly scale the number "00010". If the number had been "001XX", it could be shown by similar means that sense line S_1 would be the only one linking bits all in the zero state, and the corresponding scaling shift would be one.

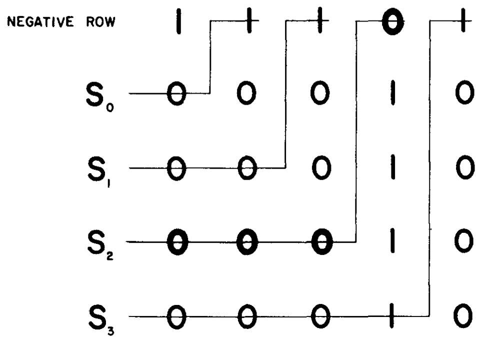


Fig. 5—Functional array for determining most significant information bit.

A method of implementing this logic with film elements is illustrated in Fig. 6. The number to be scaled is located in the input register, stages R_0–R_4. If any stage of the input register contains a binary "1", it will initiate one of the corresponding bias generators B_0–B_4. The function of the bias generators is to supply a field transverse to the remanent state of the films linked by its output line. This field is represented by vectors $H_B(1)$ in Figs. 6(a) and 6(b). After the films in the array have been appropriately biased by the bias generators B_0–B_4 and B_p, the action of which will be explained later, the film elements are driven by drive generator D. This driver links all films in the array and supplies a field that is antiparallel to the remanent state of the film cores, but not of sufficient magnitude to completely switch the film core with no other applied fields. This field is identified by vectors H_D in Figs. 6(a) and 6(b).

Referring to Fig. 6(a) and 6(b), the effects of the bias and drive fields are shown for the various rows of films in the array. A permanent bias field — represented by vector H_{BP} in Fig. 6(a) — is applied to all film elements in the first or negative row of films, where it is desired that a negative copy of the word be represented. This field has the effect of rotating the magnetic state vector away from the remanent direction of magnetization so that, without the application of another biasing field by one of the bias generators B_0–B_4, an output would be obtained on sense lines linking these film elements when drive field H_D is applied. If a field is applied by one of the bias generators, the magnetic state vector is rotated back into alignment with the remanent direction, so that no output is produced on sense lines linking these film elements when drive field H_D is applied. Therefore, in the first row, (1) no output is obtained on any sense line that links a biased film element if there is a "1" in the corresponding input register, and (2) an output is obtained if there is a "0" in the input register. Thus the action of the permanent bias generator B_p produces a negative copy of the input word in the first row. Examination of Fig. 6(b) for the remaining rows shows that the converse conditions apply; *i.e.*, a "0" output is obtained on a sense line linking a film element associated with a register containing a "0", and a "1" output is obtained on a sense line linking a film element associated with a register containing a "1".

The array of Fig. 6 is arranged in the manner described in the example of Fig. 5. A signal to indicate the amount of shift required is obtained by the use of inverters that terminate each sense line. Since only one sense line will have zero signal induced on it, only one inverter will have an output signal. Because a zero

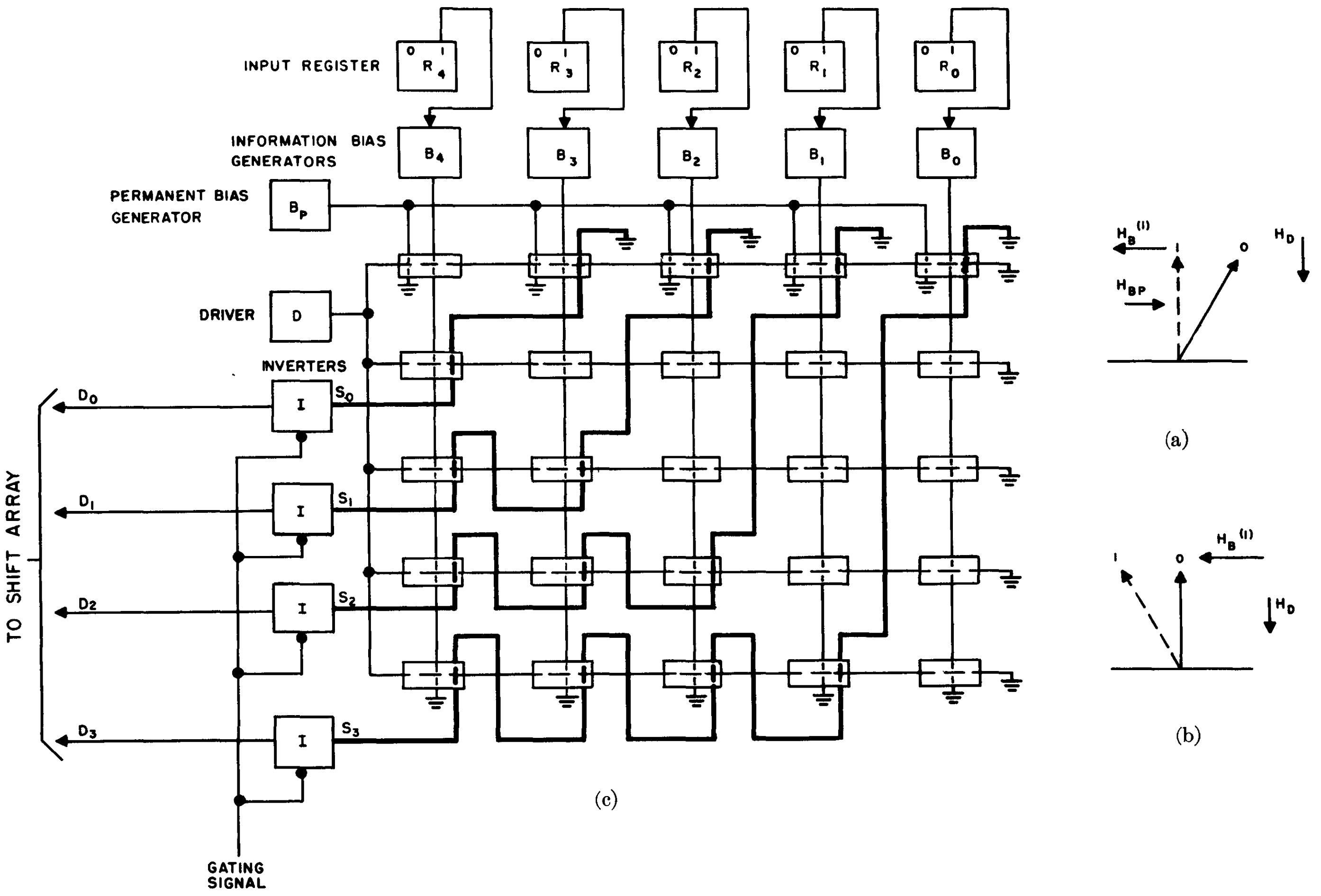


Fig. 6—(a) Film magnetization directions of Row 1 (negative row) for zero bias. (b) Film magnetization directions o. Rows 2-5 for zero bias. (c) Network for determining most significant information bit.

or null signal is used to the inverters, the inverters are necessarily gated as indicated in Fig. 6.

Shifting

For shifting operations, it is also desirable to be able to shift a word an arbitrary number of positions in a time not dependent upon the number of positions shifted. Here again functional arrays are readily applicable. An array for accomplishing the left-shifting of a word two positions is illustrated in Fig. 7. As in the example of Fig. 5, the word "00010" is used. Five copies of the word are represented in the array, and sense lines are diagonally drawn through the array as shown in Fig. 7.

Shifting in this array is accomplished by transferring a selected row of bits via the sense lines to the output register. In the example of Fig. 7, an open-ended left shift of two is obtained by selecting the third row and transferring the bits via sense lines S_2, S_3, and S_4 to the output register. Similarly, other shifts can be obtained from the same array by selecting other rows. If, for example, row 1 is selected, a shift of zero is obtained; if row 5 is selected, a shift of four is obtained.

The circuit of Fig. 8 illustrates a film-element array

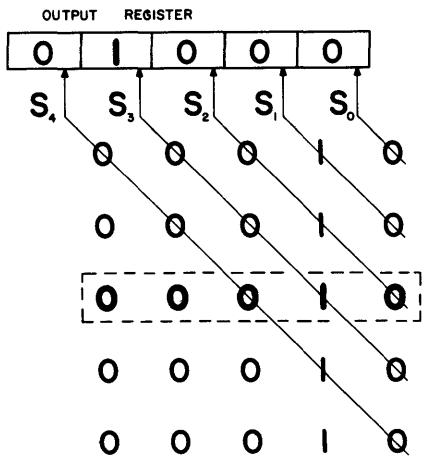


Fig. 7—Functional array for left-shifting a word.

for the execution of left shifts. The word to be shifted is originally in the input register R_0–R_4, and bias generators B_0B_4 are initiated if there is a "1" in the corresponding input register. The bias generators supply a field transverse to the remanent magnetization direction. This field rotates the magnetic state vector in each film element away from the remanent direction so that a drive pulse applied antiparallel to

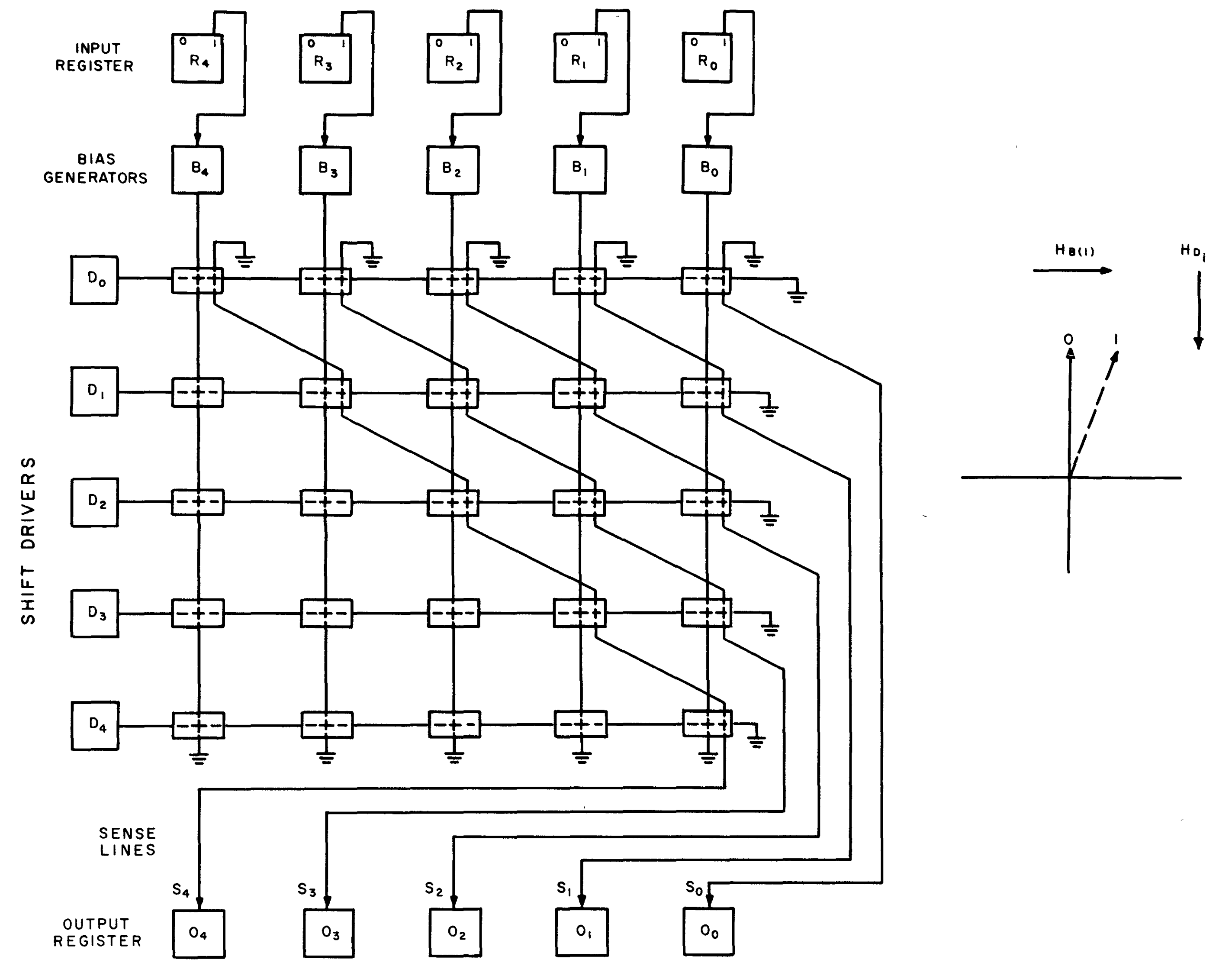


Fig. 8—(a) Film magnetization directions. (b) Network for left-shifting operation.

the remanent direction produces an output on a sense line linking that film element. The bias field is represented by vector H_B (1) in Fig. 8(a), and the drive field is represented by vector H_{D_i}. To implement a shift of from zero to four in this array, one of the shift drivers, D_0 to D_4, respectively, is initiated, and the corresponding row of films is supplied with a drive or interrogation pulse. The sense lines linking the film elements in the interrogated row will have an output signal only where the film element linked is initially biased away from the remanent state. These sense-line signals are coupled to the stages of the output register and the resulting word is in its shifted position.

Encoding

In a preceding paragraph the amount of shift was determined by locating the position of the most significant bit in a word. This shift count appeared as a unique signal on one of the lines D_0–D_3, as shown in Fig. 6. In many applications it is desirable to store this signal as a binary number. This requires a "one-to-many" translation. An encoder is a device for accomplishing this result.

Physically, a signal representing the number is applied to the encoder input. The output from the encoder then appears as one or more signals, corresponding to the respective "1" bits of the binary representation of the given number. For example, the number "13" would be encoded as "1101" with signals from the output of the encoder setting corresponding stages 3, 2 and 0 of a four-bit encoder register.

Fig. 9 shows a three-bit magnetic film encoder with its associated register. Inputs to the encoder are shown as D_1, D_2, D_3 and D_4, corresponding to shift counts of 1, 2, 3 and 4, respectively. (Note that only one of these inputs is active at any given time.) The output from the encoder appears in the scale-factor shift-count register, stages K_2, K_1, K_0. Film elements F_0, F_1 and F_2 act as AND gates operating in the saturable-transformer mode, as described previously.

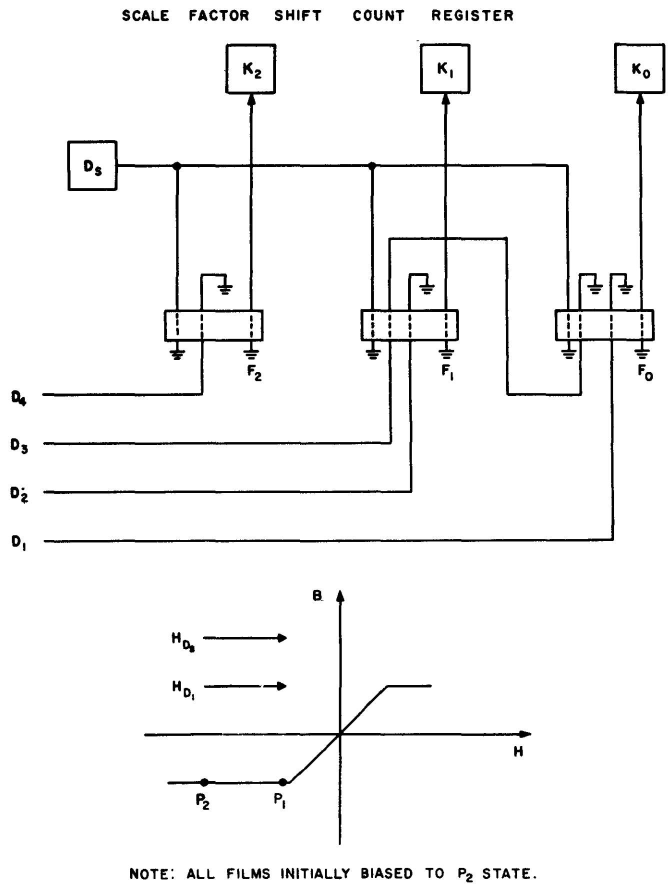

SCALE FACTOR SHIFT COUNT REGISTER
K2
K1
K0
Ds
F2
F1
F0
D4
D3
D2
D1
B
HDs
HD1
H
P2
P1
NOTE: ALL FILMS INITIALLY BIASED TO P2 STATE.

Fig. 9—Scale-factor shift-count encoder.

All three film cores are initially biased to the P_2 state. The input lines are so wired that lines D_1, D_2 and D_4 link film elements F_0, F_1 and F_2, respectively, while line D_3 links both film elements F_0 and F_1. A field H_{D_i}, corresponding to a D_i input, biases the film element (or elements) that it links to the P_1 state. The drive generator D_s subsequently supplies a drive field H_{D_s} to all the films. Any film element that is in the P_1 state therefore produces an output signal on its respective sense line. This output then sets the corresponding scale-factor shift-count register stage to "1".

In the example used to illustrate the scale-factoring operation, where the shift count was 2, a field H_{D_2}, corresponding to input D_2, biases the film element F_1 to the P_1 state. Subsequent application of drive field H_{D_s} then produces an output on the sense line of film element F_1, and thereby sets scale-factor shift-count register stage K_1 to "1". Film elements F_0 and F_2 do not have outputs because their states are unaltered, having remained at P_2. Consequently, the scale-factor shift-count register reads "010", which is the binary representation of 2.

Combined Circuit Operation

Circuit Operation

Fig. 10 is a composite drawing incorporating the circuit for determining the highest-order information bit with the circuit required for shifting. Since both of these arrays utilize the same mode of operation, namely, reversible-rotation, it is possible to combine them in the same array and use the same bias generators. Inspection of Figs. 6 and 8 reveals that certain film elements in each array are not used in the performance of the logic operation. These unused film elements are not included in the combined array of Fig. 10.

The operation of the circuit is divided into two major sequences: determination of the highest-order information bit, and the shifting operation, with the encoding being accomplished during the shifting operation. After the array has been biased, the first sequence is initiated by driver D, which supplies a drive field to the film elements in the highest-order information-bit-determination portion of the array. These film elements are linked by sense lines S_1–S_4. These sense lines are coupled to the shift driver inverters D_0–D_3. Since a zero output on one of the sense lines is the required signal to the shift driver inverter, these drivers must necessarily be gated. The output of drivers D_0–D_3 is used to drive the film elements in the shift array and encoding network. The operation of the shift and encoding circuits is as described above.

In the example of Fig. 5, a positive number is used for illustration. If the same approach is applied to a negative number, *i.e.*, the complement in the first row and the number itself in the remaining rows, there is not a unique method of determining the location of the highest-order information bit. If, however, the negative number itself is placed in the first row and its complement, or the positive copy, in the remaining rows, the previous rules apply. It follows that some form of gating between the input register stages and the bias generators is necessary. Similarly, since the information in the shift array is in its complement form for negative numbers, some form of gating between the shift array and the output register is necessary. The conditional complementer circuits shown within the dotted line enclosures of Fig. 10 accomplish these gating functions.

If the number originally in the input register is negative, R_4 is "1", and the R or negative generator drives the row of P_1 saturable-transformer film elements in both conditional-complementer networks. These P_1 film elements act as AND inverters (*i.e.*, Sheffer-stroke functions) in both networks and complement the information supplied to the bias generators for the array and again recomplement the information from the array for the proper output representation. If the number in the input register is positive, R_4 is "0", and the $\bar{R}$ or positive generator drives the P_2 row of film elements in the conditional-complementer networks. These film elements are AND gates and allow the information to be transferred

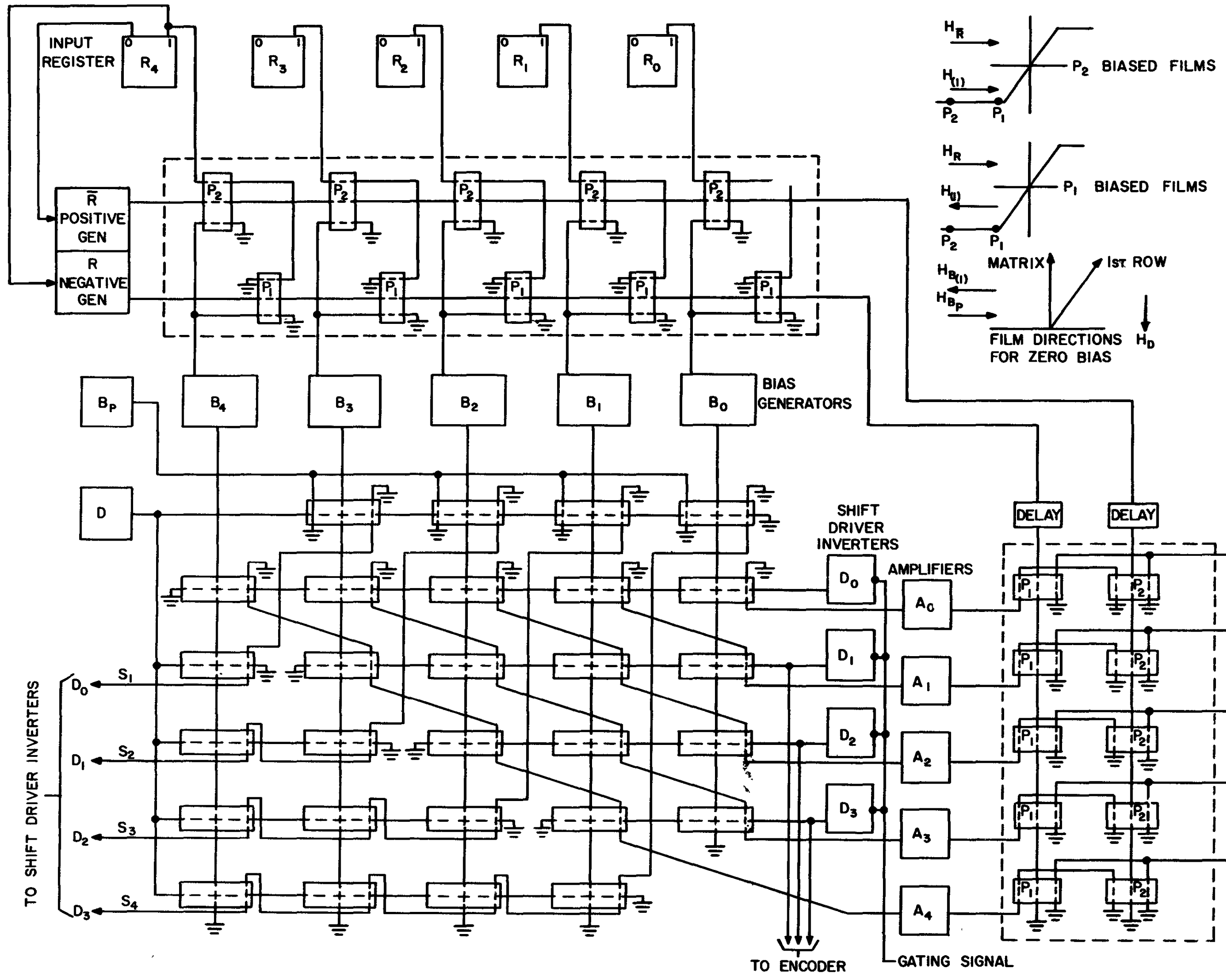


Fig. 10—Scale-factor network.

directly to the bias generators and output circuits.

The encoding network shown in Fig. 9 serves in conjunction with the scale-factoring network of Fig. 10. The outputs of shift driver inverters D_1–D_3, which drive the shift array, are also used as the inputs to the encoding network. In this manner the amount of the shift performed is recorded in the shift count register at the same time that the shifted number is entered into the output register.

Circuit Timing

A detailed timing sequence for the scale-factor operation is presented in Table I. Included in the table are approximate expressions that might be used to determine execution times on the basis of word length and other circuit parameters. The parameters used are defined as follows:

M = word size in bits
T = transistor rise time
f = film, drive, bias, and sense line transmission time
R = rise time of film element in saturable transformer mode

From Table I the approximate expression for the execution time of the scale-factor operation is

$$\text{max. time} = 4T + (4M + 5)f + 2R$$

Assuming a transistor rise time $T = 5$ mμsec, film-element transmission time $f = 0.12$ mμsec, film-element rise time $R = 1$ mμsec, and a word size $M = 36$ bits, the maximum shift time for the scale-factor operation would be 39.9 mμsec.

Circuit Components

The components required for the scale-factor operation, exclusive of the component requirements for the design of the input register and the encoder, are as follows:

TABLE I

Scale-Factor Timing Sequence

Initiate Scale Factor Operation		Locate Most Significant Bit	Store Shifted Word	Encoding
Test sign +	(Negligible) −			
Initiate $\bar{R}$ Generator (T)	Initiate R Generator (T)			
Transmission time (Mf)		Initiate input transfer		
		Conditional complementer Drive transmission time (2f) Film element rise time (R)		
		Initiate bias generators (T)		
		Bias transmission time (Mf + f)	Initiate scale-factor driver (T)	
			Drive- and sense-transmission time (Mf)	
			Initiate inverter driver (T)	
			Drive- and sense-transmission time (Mf)	Encoder bias transmission time (maximum) (Mf)
			Amplifier (T)	Initiate read driver (T)
			Conditional complementer Drive transmission time (2f) Film-element rise time (R)	Drive and sense transmission

1. Film elements

$$\begin{aligned} \text{Input} &= 2M \\ \text{Output} &= 2M \\ \text{Matrix} &= M^2 + M - 2 \\ \hline \text{Total} &= M^2 + 5M - 2 \end{aligned}$$

2. Transistors

$$\begin{aligned} \text{Bias Generators} &= M + 1 \\ \text{Inverter Drivers} &= M - 1 \\ \bar{R}\ \&\ R\ \text{Generators} &= 2 \\ \text{Amplifiers} &= M \\ \hline \text{Total} &= 3M + 2 \end{aligned}$$

Table II presents the film-element and transistor requirements for encoders of various sizes. For comparison purposes, the number of diodes that would be required for conventional encoders of equivalent size are shown. The input and output components are omitted for both types of encoders.

Since film elements, unlike diode elements, permit the use of more than one input per element, the film-element encoder uses very few film-elements in comparison with the number of diodes in a diode encoder. Furthermore, the only semiconductor devices required are one transistor for each output bit. The number of diodes required for the larger diode encoders would be greater because of the diode OR circuit input limitation. For these encoders the diodes would probably be arranged in a "tree" or "pyramid" configuration, which would result in an increased time requirement for the diode encoder. Without the "pyramid" arrangement, the times for the two encoders are approximately equal.

TABLE II

Encoder Component Requirements

Output Word Size (In Bits)	Film-Element Encoders		Diode Encoders
	Film Elements Required	Sensing Transistors Required	Diodes Required
2	2	2	4
3	3	3	12
4	8	4	32
5	20	5	80
6	48	6	192

Other Applications

The illustration given in this paper utilizes a five-bit word in the one's complement number representation. The method applied, however, is not restricted to this representation, this word size or the particular application which has been described. With minor modifications the device can be adapted to any complement, sign and magnitude, or binary-coded number representation. Devices to perform such operations as locating the least significant information digit and shifting the word accordingly, or locating a predetermined information digit within a certain field or portion of a word can also be readily designed.

Although this paper is concerned primarily with the application of film-element logic and the design of a specific logical device, the techniques described have a much wider range of applicability. The authors have investigated and designed a variety of logical devices such as decoders, counters, accumulators, and special-purpose devices.

Acknowledgment

For consultation on the physical properties of magnetic film elements, the authors are indebted to members of the Physics Department of Remington Rand Univac (St. Paul), especially Dr. A. V. Pohm[2] and Dr. R. M. Sanders.

Discussion

H. Aiken: I wonder if you would discuss a couple of points for me. One, your primary approach to the logics was with the aid of the matrix, so I am wondering what you have done to minimize the number of elements, making further use of Boolean for this purpose. And the second comment you can answer yes or no, have you any attempt so far to build whole circuits with this technique in one fell swoop to set up a standard?

Mr. Franck: As to the first question, no attempt was made to phrase the logic in such Boolean form as to use minimization techniques. In general, as can be noted for the device discussed, which uses the film elements quite efficiently, these techniques would probably offer little if any results in the way of reducing the number of components.

[2] Now on the staff of the Department of Electrical Engineering, Iowa State University, Ames, Iowa.

In respect to the second question, a design for a shifting matrix is actually working. For obvious reasons, I cannot say too much about it. In your sense of one fell swoop, one can say it essentially was so done, *i.e.*, a single evaporation placed films on a substrate, then printed-wiring techniques were used for the wiring arrangement.

P. D. Goodman (Clevite Transistor): What switching speed can be obtained with these devices? What current and voltage are required for switching? How large is each element?

Mr. Franck: I might point out I am not trained as an electrical engineer but as a mathematician and obtained this type of information from appropriate sources. I can give estimates. For full switching, the speed is 250 millimicroseconds, whereas for rotational switching, it is 3 to 30 millimicroseconds. Input voltages of 10 volts and currents of 200 milliamperes have been used in the design of the shift array. Typical sizes for the element range from 1 millimeter for circular elements to 1½ by 5 millimeters for rectangular elements. Output voltages of 4 millivolts per turn have been measured for the shift array.

G. A. Sellers (Bell Labs.): Please describe the physical characteristics of a "thin film": size, etc., and how they are fabricated?

Mr. Franck: I am not sure whether you mean actual dimensions. I think I have described this as essentially one millimeter. The thickness is 1 to 200 angstroms. Typical dimensions of films range from 1 to 4 millimeters. A few of 8 millimeter size have been used in experiments. The films have been deposited on thin cover-skip glass. Both 6-mil and 9-mil glass have been used. The methods of fabrication are described in an article in the *Physical Review* by C. D. Olson and A. V. Pohm.

R. Turner (Philco): What sort of switching speed is realized?

Mr. Franck: The speed for rotational switching is as fast as 3 millimicroseconds. For full switching, a quarter microsecond is typical.

J. Jacoby (BTL): How are the leads, drive and sensing leads, physically associated with films?

Mr. Franck: Printed wire techniques are used.

L. Mintzer (Honeywell-DATAmatic): Since these are passive elements, the number of sense amplifiers is not negligible. Approximately how many active elements in sense amplifiers?

Mr. Franck: In the shift array which has been designed, three transistors have been used on the output of a given sense line for amplification.

Solid-State Microwave High Speed Computers

JAN A. RAJCHMAN†

Introduction

THIS PAPER presents results of an effort aimed at developing the principles and technology required to speed the rate of computers up to the order of a thousand megacycles. The approach is based on the use of two types of two-terminal semiconductor devices: the variable-capacity diode and the tunnel diode, in combination with microwave techniques for the couplings within the computer.

Both devices provide amplification of binary signals by mechanisms depending on negative resistance. Their speed limitation is primarily due to the capacity of the junction and internal series resistance and can be two orders of magnitude higher than that of transistors which are limited by the travel time of minority carriers. The variable-capacity diode can be used for computer logic in parametric phase-locked oscillators according to concepts* described by Goto[1] and Von Neumann[2]. The negative resistance of the tunnel diode can provide amplification and gain directly. Both devices have only two terminals, *i.e.* a single port for the input and output, so that special methods are required to give direction to information flow. These methods and the means to perform the other necessary functions of storing and gating signals are described in the following sections.

Parametric Phase-Locked Sub-Harmonic Oscillator (PLO) Computers

Principle of Operation

Consider a tuned circuit composed of a fixed inductance and a capacity whose value depends on the voltage across it (*i.e.* junction diode). Let the tuned circuit be excited by a frequency $2f$ which is approximately equal to twice the resonant frequency of the circuit. (Fig.1) This excitation will tend to produce oscillations at frequency f in the circuit, and oscillations will actually be sustained if the excitation is sufficiently intense and the losses in the circuit are sufficiently small. This effect is a special "degenerate" case of a broad class of parametric excitation effects.

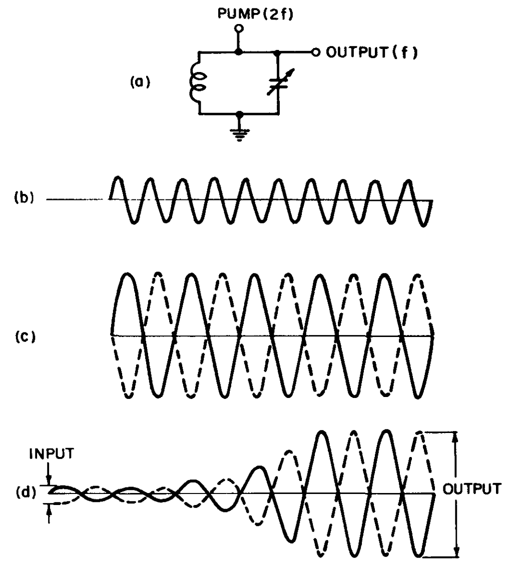


Fig. 1—Principle of parametric phase-locked oscillator.

The general theories of parametric oscillations, as well as the particular theory of this degenerate case have been reported by several authors[3,4,5,6,7].

The reason for the build-up of oscillations can readily be understood by a simple physical reasoning. Let us assume that every time at which the capacity has maximum charge, the value of the capacity is reduced, as would be the case if the plates were pulled apart. The work necessary to reduce the capacity increases the energy stored in the condenser. The value of the capacity is restored to its initial value at the instant when the charge in the condenser is zero. In this way a certain amount of energy is added to the circuits at every half-cycle of the oscillation. If this increase of energy is greater than the loss of energy in the half-cycle due to damping in the circuit, the amplitude of oscillations will grow. It is easy to see that this "pumping" of energy occurs at twice the frequency of oscillations and therefore can sustain an oscillation of either of two opposite phases. In the actual case of a variable capacity diode, the change of capacity is due to the voltage of the pump source

† R.C.A. Laboratories, Princeton, N. J.

* R. L. Wigington, "A New Concept in Computing," *Proc. IRE*, Vol. 47 — No. 4, pp. 516–523, April 1959. (An account of J. von Neumann ideas in footnote 2).

[1] Eichii Goto, "On the Application of Parametrically Excited Nonlinear Resonator," *Denki Tsushin Gakkai-shi*, Oct. 1955.

[2] J. von Neumann "Nonlinear Capacitance or Inductance Switching, Amplifying, and Memory Organs," *U. S. Patent 2,815,477*, Dec. 3, 1957, assigned to IBM.

[3] J. J. Stocker, "Nonlinear Vibrations in Mechanical and Electrical Systems," Interscience Publishers, Inc., New York, 1950.

[4] W. J. Cunningham, "Nonlinear Analysis," McGraw-Hill, New York.

[5] K. L. Kotzebue, "A Semiconductor-Diode Parametric Amplifier at Microwave Frequencies," *Stanford Electronic Laboratories Technical Report*, No. 49, Nov. 1958.

[6] A. Uhlir, Jr., "The Potential of Semiconductor Diodes in High Frequency Communications," *Proc. IRE*, Vol. 46, pp. 1099–1115. June 1958.

[7] J. M. Manley and R. E. Rowe, "Some General Properties of Nonlinear Elements — Part 1 — General Energy Relations," *Proc. IRE*, Vol. 44, p. 904–913, July 1956.

applied to it instead of the mechanical work necessary to pull the plates apart, but the effect is analogous.

The oscillations are sustained in either of two opposite phases which are locked to the phase of the pump and can be used to denote "zero" and "one" of a binary digit. The phase-locked-oscillator, PLO, constitutes thus a storage cell. The steady-state phase depends on the conditions under which oscillations start. If a small locking signal at the frequency f is present in the tank, oscillations will build up in the phase closest to the phase of the locking signal. The input locking signal is thus "amplified." (Fig. 1.)

Logic can be performed by arraying the PLO's in three or more groups, which are separately activated either by pump modulation or diode bias gating. Every PLO is loosely coupled to PLO's in other groups, the pattern of couplings determining the logic task to be performed. The groups are clocked in succession with some overlap, *i.e.*, a given clock is turned off after the next one is turned on. This sequence causes information to flow in a given direction despite the bilateral character of the PLO. A PLO will start at the phase determined by the phase of the majority of oscillating PLO's to which it is coupled. The majority decision can be exploited directly in many circuits or can be reduced to "and" or "or" decisions by the use of a reference signal on one input. For example, with two inputs, and a reference in phase zero, the output will be in phase π only when both inputs are in phase π. Negation is easily obtained by phase inversion. In a typical example of logic circuit, Fig. 2, each PLO may be connected to two inputs, two outputs and one reference, or to five other PLO's. Consequently, the input is at most one fifth of the output of preceding PLO's. Thus, a minimum "logic gain" of five is required. In a simple shift register, Fig. 3, minimum logic gain is two.

There is a certain increase of amplitude of oscillation at each cycle, which depends on the parametric pumping, *i.e.*, specific variation of capacity and power, and on the losses of the circuit which are made up of the useful loading and unavoidable circuit dissipations. To build up the amplitude by a factor corresponding to practical logic gains, about 5 cycles of oscillations or 10 pump cycles are required in typical PLO's. Therefore to obtain 1000 mc. information rates, *i.e.*, phase switching in about 3 x 10^{-10} sec., pump frequencies of about 30 KMC or higher are required.

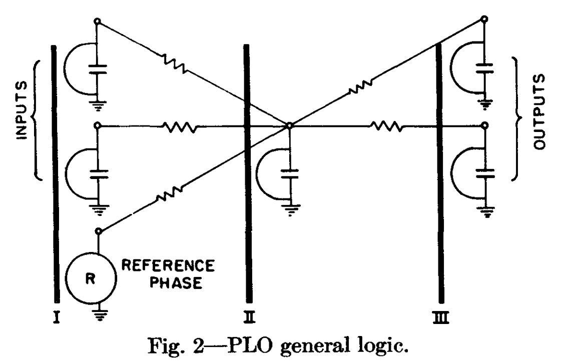
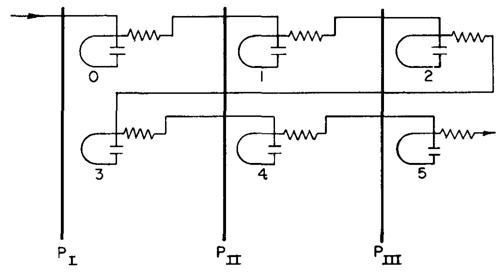


Fig. 2—PLO general logic.



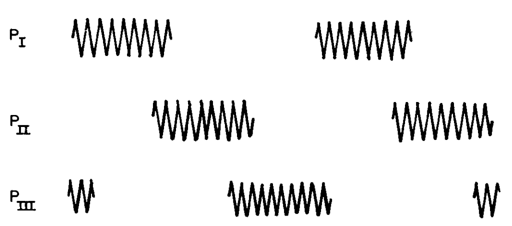


Fig. 3—Shift register with three-phase pump system.

Experimental Results

An experimental program ultimately aimed at PLO computers pumped at frequencies of about 30 KMC resulted in the following:

A. Microwave Sub-Harmonic Oscillators

Microwave circuits obtained by photographic engraving of copper-clad insulating boards, known as strip transmission lines, and point-contact diodes in conventional microwave cartridges, were used for PLO's pumped at 4 KMC. A typical early configuration (Figs. 4 and 5) included: a 2 KMC quarter-wave resonator with diode in shunt at one end, a 4 KMC resonator bar isolating pump and oscillating circuit, d-c return for optimum bias, and one or more loosely coupled inputs and outputs.[8,9,10,11,12] Output-vs-input power characteristics (Fig. 6) show broad operating range, efficiencies of a few percent, and required pump power levels of about 100 mw. Typical more recent configurations utilize a series connected-gold-bonded diode (Fig. 7) and multiple impedance-matched antennas for coupling inputs and outputs. The characteristics (Fig. 8) show uniform couplings to various antennas, and broad operating regions.

[8] W. R. Beam, D. J. Blattner and F. Sterzer, "Microwave Carrier Techniques for High Speed Digital Computing," (Symposium on Microwave Techniques for Computers, Washington, D. C., March 12, 1959) *Trans. IRE on Electronic Computers*, Sept. 1959.

[9] F. Sterzer and D. Blattner, "Fast Microwave Logic Circuits," *Proc. IRE National Convention*, March 1959; also *Proc. EJCC*, Dec. '3-5, 1958.

[10] F. Sterzer, "Microwave Parametric Sub-Harmonic Oscillatosr for Digital Computing," *Proc. IRE*, July 1959.

[11] F. Sterzer, "RF Circuits Using Sub-Harmonic Oscillators," *Proc. PGMIT National Symposium*, Harvard University, Cambridge, Mass., June 1959.

[12] F. Sterzer and W. R. Beam, "Parametric Sub-Harmonic Oscillators," *Digest of Technical Papers, Solid-State Circuits Conference*, Philadelphia, Pa., Feb. 1959.

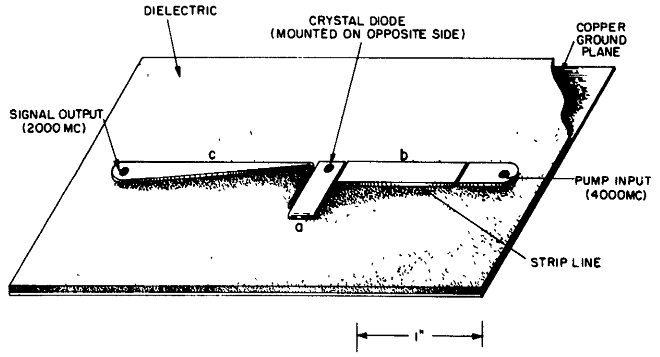


Fig. 4—Microwave subharmonic phase-locked oscillator.

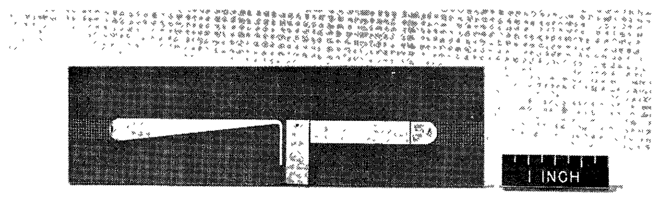


Fig. 5—Photograph of 4 KMC PLO.

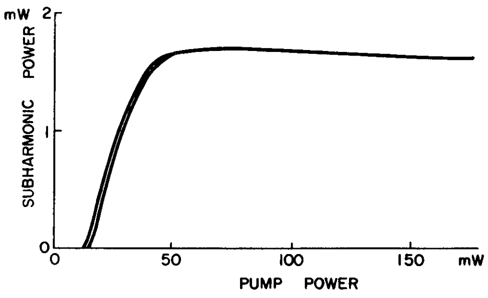


Fig. 6—Characteristic of PLO.

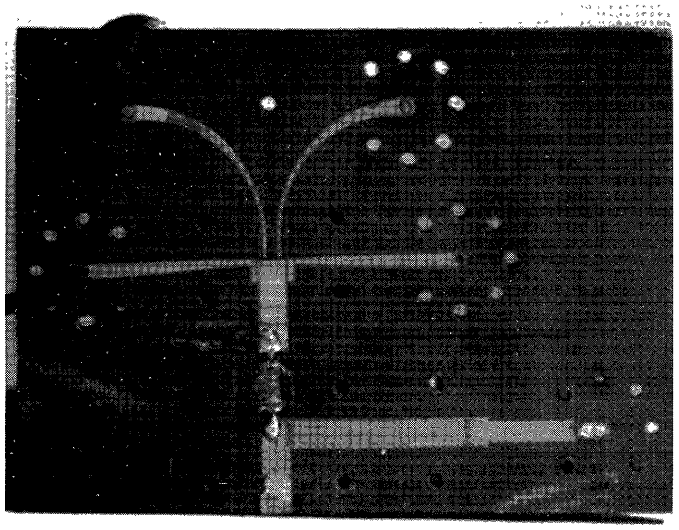


Fig. 7—Phase-locked subharmonic oscillator pumped at 4 KMC with four coupling antennas.

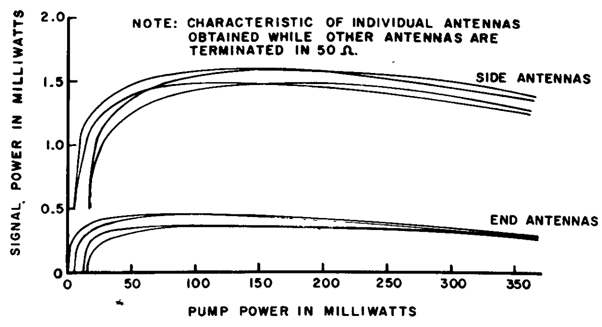
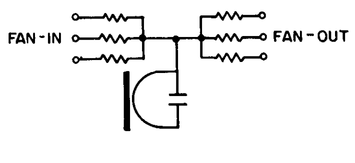


Fig. 8—Characteristic of four antenna PLO.



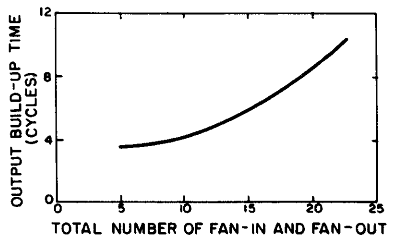


Fig. 9—Number of cycles of build-up as a function of logic gain.

Methods of switching phase, first investigated in lumped-parameter circuits pumped at 5 mc,[13] demonstrated great flexibility of "phase script" and yielded quantitative relations between logic gain and build-up cycles under a variety of conditions (Fig. 9). Qualitative confirmation of these results was obtained through more elaborate experiments with PLO's pumped at 4 KMC using mercury-wetted relay pulsers and travelling-wave oscilloscopes. Typical results: rise from noise level to saturation in 10 nanoseconds (nanosecond = 10^{-9} sec.) and decay in 1.5 nanoseconds when the diode was pulsed slightly into conduction.

B. Microwave Computer Techniques and PLO Logic

Microwave transmission line techniques provide methods for linearly combining signals to exploit direction of transmission. For example, a hybrid ring can be used to translate amplitude-modulated to phase-modulated signals and *vice-versa* through appropriate combination with a CW signal[9]. (Fig. 10) Another example is the use[11] of a hybrid ring fed by two PLO's energized by pumps $\pi/2$ out of phase, so as to obtain cancellation at one (input) terminal and reinforcement (output) at another. (Fig. 11). This provides undirectional information flow and thereby lowers the required logic gain and permits the use of two rather than three clocks. Experimentally two PLO's have been balanced so that only 5% of the power appeared in the input.

[13] L. S. Onyshkevych, W. F. Kosonocky and A. W. Lo, "Parametric Phase-Locked Oscillator — Characteristics and Applications to Digital Systems" (Symposium on Microwave Techniques for Computers, Washington D. C., March 12, 1959) *Trans. IRE on Electronic Computers*, Sept. 1959.

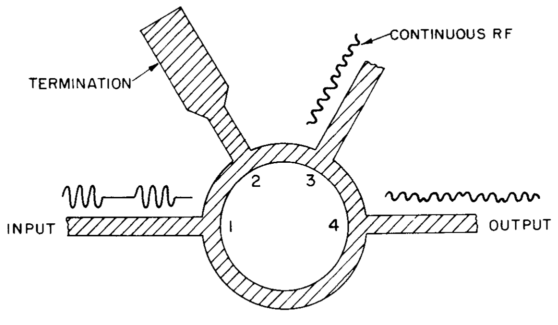


Fig. 10—Hybrid circuit for translating amplitude-to-phase scripts.

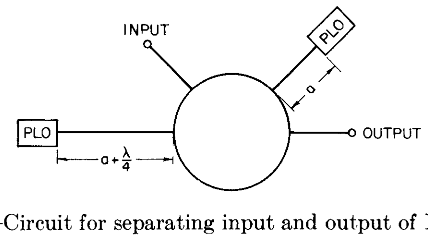


Fig. 11—Circuit for separating input and output of PLO's.

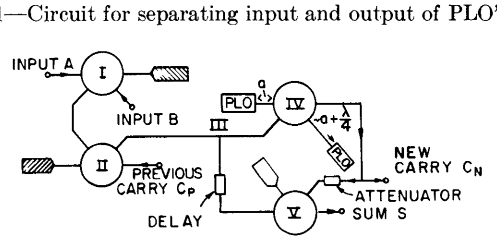


Fig. 12—Binary adder.

A full stage of a binary adder with two PLO's and four hybrid rings was made.[10,11] It operates through linear combinations of phase-script signals and PLO's acting as a majority decision elements and amplifiers (Fig. 12). Operation with pulses at a repetition rate of 100 mc was obtained. Other types of adders were made also.

A 125-mc binary scaler was made[8] using a PLO deliberately tuned at a frequency slightly different from half the pump frequency so that its phase changed for every momentary deactivation of the pump which lasted for a time sufficient to allow the natural oscillations to drift more than $\pi/2$ in phase.

In general, making of computer subsystems with PLO's can take advantage of well-developed microwave strip transmission techniques. Boards with 3 layers, with ground planes on both sides of transmission strips, permit compact subsystems without the deleterious radiation pick-ups of 2 layer boards.

C. PLO Random-Access Memory

The random-access memory made of PLO's would be particularly suitable in a machine with PLO logic. This possibility was investigated[13]. In a two-dimensional array of PLO's continuously activated by a pump, the access problem consists of (1) selectively establishing the desired phase in a selected PLO without disturbing the phase of any other and (2) of interrogating the phase of any selected PLO without ambiguity due to possible masking signals from all other PLO's.

The writing problem (1) is solved easily by forced switching. It is possible to choose the amplitude of locking signals such that each separately is too small but together the two signals are strong enough to change the phase of the PLO.

The reading problem (2) requires a more elaborate artifice. For example, a standby PLO in addition to the storing PLO can be used for each bit. Each standby PLO is loosely coupled to its associated storing PLO and is also loosely coupled to a read-out circuit. For read-out, the standby PLO, normally not activated, is selectively activated by the coincidence of two bursts of pump energization. It starts to oscillate at the phase of the associated storing PLO and thereby conveys the sought phase information to the read-out circuit. The signals of all other storing PLO's are effectively blocked from masking the read-out signals since their standby read-out PLO's are not activated.

Experimental memories have operated successfully at frequencies of less than 10 mc but complexities of technology with early designs of PLO's have made operation at microwave frequencies difficult. Recent improved designs would greatly facilitate the memory design.

D. Variable-Capacity Diodes for PLO's

Microwave-cartridge point-contact diodes of commercial type and laboratory units made by specially-developed techniques, as well as gold-bonded diodes mounted within the boards in quarter-wave series resonant circuits, permitted the experiments reported above but had serious drawbacks due to limitations in speed and wide variations from unit to unit. A program to develop junction types has resulted in a clear understanding of the limiting factors[14] and diodes of practical design with an order-of-magnitude better performance.

The simultaneous realization of low series resistance r_s within the diode and high variation of capacity with respect to voltage $(\frac{1}{C}\frac{dC}{dV}$, around a value C_0 arbitrarily taken at -1 volt) requires that the impurity concentration be not uniform but have a specially-designed profile. This was realized in solution-grown and out-diffused p-n germanium junctions. High cut-off frequency f_0 for reasonable impedance and low power requires that the capacity be low and the area of the junction be of the order of 10^{-6} sq. inches.

In addition to the constants of the semi-conductor proper, it is essential to minimize the capacity C_C, the series inductance L, and the resistance of the

[14] J. Hilibrand, C. W. Mueller, C. F. Stocker and R. D. Gold, "Semiconductor Parametric Diodes for Microwave Computers" (Symposium on Microwaves Techniques for Computers, Washington, D. C., March 12, 1959) *Trans. IRE on Electronic Computers*, Sept. 1959.

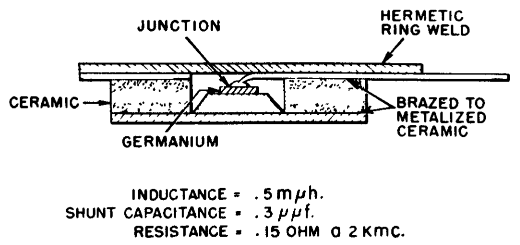

JUNCTION
HERMETIC RING WELD
CERAMIC
BRAZED TO METALIZED CERAMIC
GERMANIUM
INDUCTANCE = .5 mμh.
SHUNT CAPACITANCE = .3 μμf.
RESISTANCE = .15 OHM @ 2 KMC.

Fig. 13—Cross-section of variable capacity diode.

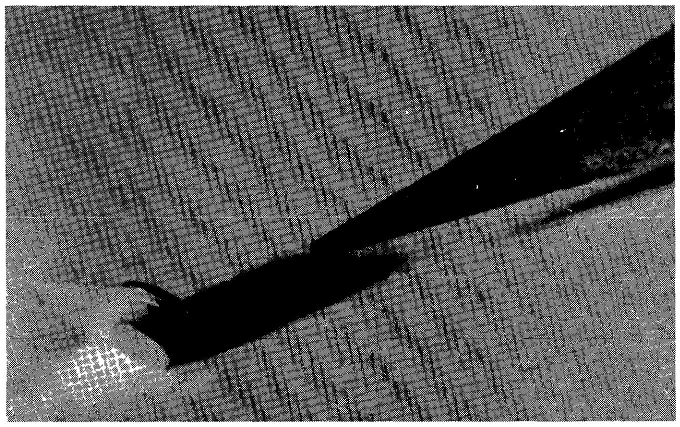


Fig. 14—Microencapsulation for variable capacity and tunnel diodes.

contact of the diode encapsulation. A special micro encapsulation was developed (Fig. 13, 14) in which the case capacitance is only .5 $\mu\mu$F and the lead inductance is only 300 $\mu\mu$H. It consists of a ceramic ring .085″ in diameter sealed hermetically between two metal plates, with a metal finger entering from one side. A thin wire contacts the dot on ther germanium wafer and is soldered to the finger. The diode is inserted in the printed-wiring boards, and the upper and lower tabs can be directly soldered to the printed lines. The resulting circuits are not only superior in performance, but simple to construct. Typical constants of a microencapsulated variable-capacity diode are $C_0 = 1$ $\mu\mu$F, $R_s = 1$ ohm, $fo = 150$ KMC $L = 300$ $\mu\mu$H, and $C_c = .6$ $\mu\mu$F. Capacity-voltage sensitivity, difficult to measure directly, is relatively high as judged by improved PLO performance.

These diodes have permitted the design of 10-kmc-pumped PLO's with which gains of 20 db and rise times of 2 to 3 nanoseconds, and efficiencies of 10% were realized. The corresponding information-switching rate would be about 300 mc as judged from the extrapolation of 100 mc rates of 4-KMC-pumped PLO's.

Tunnel-Diode Computers

Tunnel Diodes

Abrupt-junction diodes made of very highly-doped material exhibit a negative resistance at small forward bias. This effect was described by L. Esaki[15], who interpreted it as due to quantum tunneling. As the negative resistance makes amplification possible and the effect is inherently fast, it was realized that the tunnel diode is particularly suited for high-speed computers. The device was investigated in detail and a number of germanium units especially adapted for this use were made.[16]

The current-voltage characteristic (Fig. 15) exhibits in the forward direction a maximum, a drop corresponding to negative resistance ($-R$), a minimum, and a subsequent rise. The negative resistance is well understood by semiconductor theory, and can be thought of as due to a diminution in the number of electrons which can tunnel through a potential barrier as that barrier is lowered — a seemingly paradoxical fact resulting from the decrease in electronic states adjacent to the potential barrier.

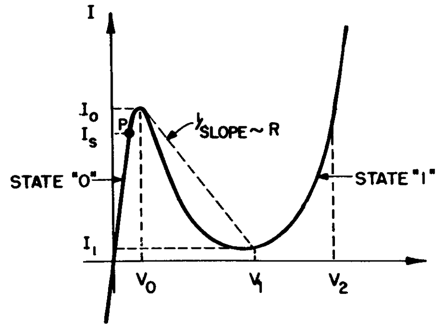


Fig. 15—Tunnel diode characteristics.

The gain-bandwidth product, as well as the upper frequency of oscillation realizable, were found[16] to be inversely proportional to RC, where C is the physical capacity of the diode junction. The time constant RC, independent of junction area, can be small despite the large value of C (typically 3μf/cm^2). This is because the resistance R can be made very small, as it is a negative-exponential function of the impurity concentration. High concentrations are obtainable by suitable doping techniques. Time constants as low as 5 x 10^{-11} seconds have been determined by measuring R and C separately. In another experiment, a tunnel diode was switched by means of a mercury-wetted relay and the resulting switching was observed on a sampling oscilloscope. Rise and decay times less than 10^{-9} seconds and a plateau of about 10^{-9} seconds were observed. Tunnel-diode oscillators of 1600 mc were made. Recently oscillations as high as 10,000 mc were reported [17].

It is necessary that the series resistance r and the inductance L of the diode and its mount be sufficiently small to make the time constants rC and L/R small compared to RC. Only very short lead-ins can be tolerated. Microencapsulations with wide, short, closely-spaced terminals have been designed and

[15] L. Esaki, *Physical Review*, 109, p. 60, 1958.

[16] H. S. Sommers, Jr.,"Tunnel Diodes as High Frequency Devices," *Proc. IRE*, p. 1201, July 1959

[17] R. N. Hall, "Tunnel Didodes," *Proc. 1959 Electron Devices Meeting*, Washington, D. C. October 29, 1959.

permit direct incorporation of diodes in low-impedance (typically 10 ohms) transmission lines. It turns out that the ceramic microencapsules originally designed for the variable-capacity diodes is particularly suitable for the tunnel diode. (Fig. 14.) These units have only about 300 $\mu\mu$H of series inductance.

Hundreds of tunnel diodes were fabricated on a laboratory scale[16]. These were germanium types with impurity concentrations of about 2.5 x $10^{19}/\text{cm}^3$. Various sizes were made with peak currents varying between 1 and 700 mc. The area of the junction in these diodes is about 5 x 10^{-5} cm^2 which entails a capacity of about 100 $\mu\mu$F.

Tunnel Diode Logic Circuits

Logic switching can be performed by tunnel diodes because their characteristics have sharp thresholds permitting gating, and negative resistance permitting signal amplification. Signals are baseband pulses, in contrast to the carrier-modulated signals of the PLO. Two distinct regions of the characteristic are used: a voltage range below the voltage of the current peak to denote "O" and a range above the voltage of the current valley to denote "1". (Fig. 15.) With germanium tunnel diodes the low "O" state is typically less than 50 mv and the high "1" state about 450 mv.

Gain is obtained through a triggering action. An operating point P, with current I_s and voltage V_s, is established near the maximum of the characteristic (I_o, V_o) through appropriate biasing of the power supply. The input signal adds a relatively small increment of current (or voltage) to go over the "hump" This causes the diode to switch to the high state "1". An output current as large as the difference between the maximum I_o and the minimum I_i can be obtained without losing the state "1". The ratio of output to input currents, *i.e.*, the gain, can thus be large if the operating point is very near the maximum. Practical nearness of biasing depends on the uniformity of diodes, which was found sufficient in experimental diode batches to permit logic gains of 4 to 6. These were observed in circuits pulsed at a rate of one megacycle. The excess of the input signal over the value required to reach the maximum has an appreciable effect on the speed of triggering, as it determines the rate at which the capacity of the diode is charged to reach the triggering point. Therefore some reduction of possible logic gain must be suffered to obtain high switching speeds.

Logic switching can be accomplished by properly interconnecting simple logic elements. Each such element, made of one or more tunnel diodes, serves as a bit-store, as an amplifier, and as a threshold gate. The gating is of the majority type in which simple majority, "and", and "or", decisions are obtained by proper choice of the threshold levels. Three main types of logic elements were investigated: (*1*) bistable, consisting of a tunnel diode in series with a resistance or resistance network (Fig. 16); (*2*) bistable, consisting of two tunnel diodes in series (Fig. 17); (*3*) monostable, using diodes in series with inductances (Fig. 18).

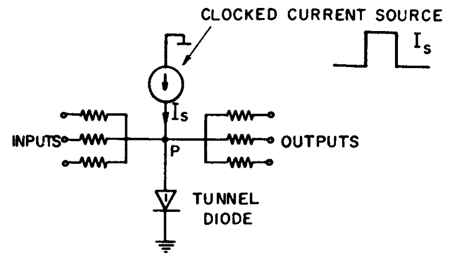


Fig. 16—Tunnel diode bistable logic element (single diode type).

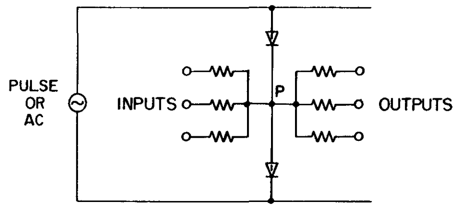


Fig. 17—Tunnel diode bistable logic element (symmetric type).

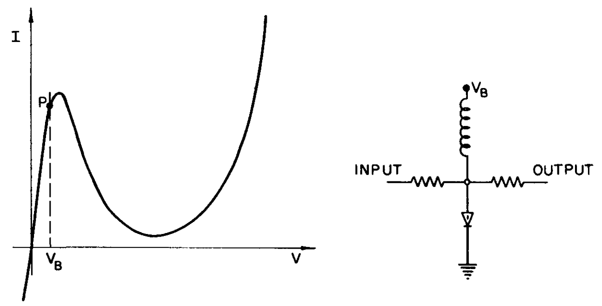


Fig. 18—Tunnel diode monostable logic element.

(*1*) In the first bistable logic element, the single port P is coupled resistively to a number of inputs and outputs as well as to a source of current I_s which can be pulsed (Fig. 16). In the absence of I_s the contributions of currents to the diode are so small that its voltage is small on the "O" part of the characteristic. The value of I_s is so chosen that when the activating pulse is applied, the total current will either be smaller of greater than the maximum I_o, depending on the sum of the inputs, and therefore the voltage will remain small or will be switched abruptly to the "1" part of the characteristic. This change of voltage influences in turn other logic elements to which it is coupled when these are activated. The logic elements are in three or more groups which are clocked by overlapping pulses, in a manner similar to the PLO clocking. An experimental unit[18] has been made to demonstrate this type of logic. It contains a storage loop, a full adder stage, means to shift right and left into registers, and other functions, including inversion. The unit contains 27 tunnel diodes with

[18] M. H. Lewin, "Negative-Resistance Elements as Digital-Computer Components" (included in this volume of *Proc. EJCC*).

peak currents of 2 ma. Observed switching times with these early, relatively slow diodes were about 50 nanoseconds. The unit was driven by a three-phase clock at 1 megacycle.

An alternative to the pulsed power supply for energizing the bistable elements is the use of a DC supply and resetting pulses. Any element of the network is set to state "1" directly according to the combined outputs of preceeding elements and is reset to state "0" by a clocking pulse. Setting of the elements of a group occurs immediately following resetting.

(*2*) In the second bistable type, the logic element is made of two tunnel diodes in series (Fig. 17). To these diodes is applied a voltage from an a.c. source, (pulsed or sine-wave) of an amplitude sufficient for one diode to be in the "1" state but insufficient for both to be in that state. The polarity of a relatively small input voltage applied to the mid-point is sufficient to determine which diode will trigger. This triggering will cause a greater voltage swing of the same polarity at that point and in effect will amplify the input signal. Logic networks can be obtained by arranging the logic elements in three or more groups and resistively coupling the midpoints of the elements of different groups. The pattern of connections determines the desired logic operations. The groups are energized in succession by overlapping voltage waves in a manner similar to the clocking of the PLO's. A particularly simple system is the use of a 3-phase sine-wave power supply. Negation is obtained by inverting transformers. Practical obtainable logic gains depend on matching of diodes in pairs rather than general uniformity of diodes as in the single-diode logic elements.

(*3*) Monostable logic elements are obtained with a tunnel diode in series with an inductance biased to a point *P* near the maximum of the characteristic (Fig. 18). A relatively small voltage can trigger the diode to the high state and thereby produce a relatively high voltage swing at the same point. Logic networks can be obtained by resistive couplings between logic elements in a manner similar to bistable elements. Resetting to the low state is produced automatically by the voltage induced in the inductance. Asynchronous operation can thus be obtained. In an experimental 7-stage delay chain, very uniform successive triggering was observed. Synchronous operation is possible also, by superimposing clocking pulses on the inputs; but it appears that higher speeds are realizable for a given logic gain in bistable circuits in that mode of operation.

All three types of circuits utilize logic elements in which the inputs and outputs are on the same single terminal. To insure separation of input and output functions, *i.e.*, direction of information flow, several methods are possible. In the above-described systems, separation in time was used by multiple-phase clocking, the inputs at any one logic elements being effective at a different time than the outputs. Directionality by electrical separation can be obtained through the use of unidirectional couplings. Unfortunately, adequate rectifying diodes of speed comparable to that of the tunnel diodes are not available at the present time. Directionality can be achieved also by making the level of energy of successive logic elements in a chain or the couplings between them progressively weaker so as to insure that the setting influence of the inputs dominates over the backflow influence of the outputs.

The multiple-phase clocking is a simple solution to the directionality problem and the key to the successful use of single-port two-terminal devices. An increase of speed can be realized if several logic steps instead of a single one are made at each clock pulse. This is possible by using a cascaded arrangement of logic elements driving each other asynchronously. In these circuits, means for directionality of information flow other than clocking must be provided.

From the above considerations, it is evident that simple logic circuits of complete generality can be made from tunnel diodes. The signals are direct pulses and require no carrier. Energization can be either by multiphase sine wave or pulsed ac or by a combination of dc and clocking pulses. Experimental circuits have demonstrated general system flexibility. The speed of the circuit can be very high. With early, relatively slow experimental diodes having 2 ma peaks, one-megacycle repetition rates were demonstrated; but switching times were short enough to permit 10 megacycle rates. With newer, faster 20 ma units, logic elements were switched in times permitting 100 megacycle rates. The speed capabilities of tunnel diodes are still being increased dramatically so that there is great promise for realizing logic circuits with 1000 megacycle rates.

Tunnel-Diode Memory

Random-access memories can be made using arrays of tunnel diodes and promise to be very fast. Each bit is stored by a current-driven tunnel diode having two stable voltages (Fig. 19). Row and column are resistively coupled to each diode (Fig. 20). Any selected diode can be set to one or the other state by voltage pulses, of appropriate polarity and amplitude, on the corresponding buses. The memory can be organized for coincident-bit addressing requiring two-to-one selection discrimination or for word addressing needing only three-to-one discrimination. The maxima and the minima of the characteristics provide the necessary thresholds and are sufficiently uniform to make possible either of these coincident write-in systems.

Read-out is obtained by driving the selected element or elements to the high state "1" and observing whether or not switching results. In a second following writing cycle the elements which have changed

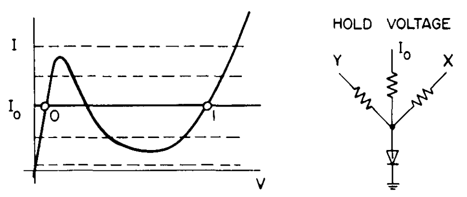


Fig. 19—Current coincident write-in.

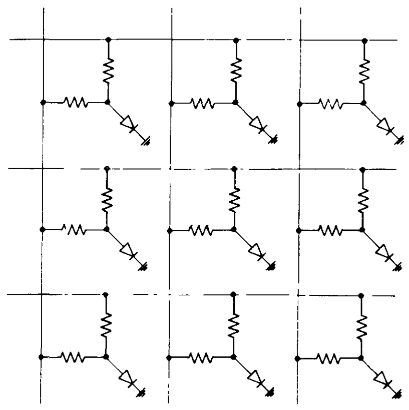


Fig. 20—Tunnel diode memory array.

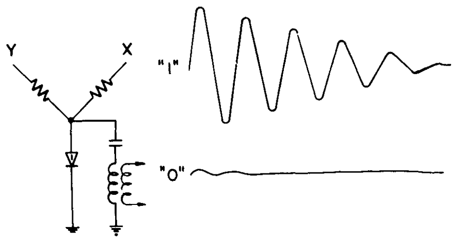


Fig. 21—Read out by sensing ringing.

state are restored by appropriate control of the writing circuits in a manner analogous to that used in conventional core memories. The read-out signal can be obtained by direct pick-up from a common circuit resistively coupled to all elements. Read-out can be obtained also through inductive or radiative pick-up of high frequency which can be generated by the selected diode in several ways. A resonant circuit, which may be a simple stub at microwave frequencies, is associated with each element. In one method, to read, a write "0" is applied to the selected element and thereby switches it or not. If there is switching the relatively large voltage excursion through the negative region of the characteristic shock excites the tuned circuit and the resulting natural-frequency oscillation is sensed on a circuit loosely coupled to all elements (Fig. 21). In another method, selective readout-addressing circuits impress signals at frequency f_1 on the selected row bus and f_2 on the selected column bus. In the high state the curvature of the voltage-current characteristic is about 4 times greater than in the low state, producing a corresponding ratio of amplitudes of the beat-frequency $f_1 - f_2$ to which the elemental circuits are tuned.

Experiments with small arrays and array skeletons have demonstrated (*1*) two-to-one coincident write-in with reasonable tolerances of operation despite the use of experimental diodes with relatively wide variations of characteristics. (*2*) adequate discrimination direct read-out pulse signals with word addressing. (*3*) inductive read-out signals of high discrimination with ringing frequencies as high as 250 mc and beat-frequency of about 1 kmc with bit-coincident addressing.

Drivers of tunnel-diode arrays must be able to supply pulses of relatively large power. The required current is large because all parallel-connected half-selected elements load the selected lines, and the required voltage is large because the voltage of the series current-regulating resistance must be several times greater than the voltage swing of the diode. Typically, hundreds of milliamperes and several volts must be provided. It is unlikely that transistors will be adequate to drive large arrays at high speed (although a line of an array was driven in less than 10^{-8} sec.). The best promise for solving the driver problem lies in the tunnel diode itself. Sufficient voltage can be obtained by connecting a number of tunnel diodes in series, and adequate current may be obtained by using sufficiently high-current units. Such arrangements have been operated and appear adequate.

Tunnel-diode random-access memories offer at the present time good promise and possibly the only promise of achieving the cycle-time of 10^{-8} seconds necessary in a memory associated with 1000 megacycle rate logic: the tunnel diodes themselves are or soon will be fast enough, and there does not seem to be any insurmountable system problem. Experiments to date have demonstrated the essential write-in and read-out steps, and have indicated a solution for the drivers. Furthermore, it appears that propagation delays along addressing lines can be kept low enough so as to be insignificant. This results chiefly from the small size of the diode and of the resulting array.

Conclusions

Computer logic by microwave-carrier techniques and the junction diode PLO has been demonstrated in elementary subsystems operating at one hundred megacycles, with single elements switching in times corresponding to 300 megacycles. Microencapsulated improved diodes promise to provide thousand megacycle rates with power supplied at 30 KMC or higher.

Computer logic by tunnel diodes, already demonstrated at low rates, promises to be possible at a thousand megacycles. Sufficiently uniform diodes and diodes capable of fractional nanesecond switching have been made. Random-access memories with cycle times of the order of 10 nanoseconds appear possible using arrays of tunnel diodes. Tunnel-diode computers operate with direct pulses and are powered by DC, or AC at signal frequency, or both.

Two-terminal semiconductor devices thus provide the manipulative elements required to gate, store, and amplify binary signals in nanoseconds. Furthermore, simplicity, small size, and power dissipation per element of tens of milliwatts permit packing of 10 or more elements per cubic inch. Therefore, reasonably comprehensive computers with several thousand logic elements and several tens of thousands of memory elements can be made in a volume less than two feet in diameter. Unavoidable delays due to signal propagation, of about 1/8 to 1/6 nanosecond per inch in normal transmission lines, are thus kept at about one nanosecond. This presents no serious difficulty in a "thousand megacycle" machine which can be assumed to have elementary logic functions executed in about one nanosecond and a memory cycle time of 10 nanoseconds.

We can, therefore, look forward to a new era of billion-bit-per-second information-handling machines which are likely to produce as large, if not larger, an impact on the information processing art as was produced a decade ago by the introduction of present million-bit-per-second machines.

Acknowledgments

The work reported in this paper is the result of the joint efforts of a number of scientists and engineers of the Radio Corporation of America in its Laboratories, Electronic Data Processing, Tubes and Semi-Conductor and Materials Divisions. The author has attempted to present a summary of their ideas and experimental results. Detailed papers by various authors have been published already and others are following. The work was supported by government contract NObsr-77523.

Discussion

H. Aiken: I wonder if you would answer a question or two for me. I was very much struck with your comment that you are now thinking about weighted logics. This is a subject that would have enormous implications all over the computer field. Would you say a few more words about this subject for us?

Mr. Rajchman: We have under this project a study as to what we could gain in a general way by weighted logic. This study at the moment is incomplete, so I can't draw conclusions, other than the fact that it isn't obvious whether all logical functions can be performed with the same amount of weighting and we are studying which can and cannot be produced that way. There is also the practical aspect of the problem that you don't want to have two weights different from one another, because you introduce the problem of accuracy of control. Since I have the microphone, I should have mentioned that I was merely a spokesman for a large project that RCA has, in which many divisions contributed; also that many of the results have been reported in detail already, and others will be reported in journals; also that the project was supported by the Navy Bureau of Ships.

D. R. Erb (Instron Eng.): How is the coincident-current memory reset? Is it always destructive or is it possible to have non-destructive read-out?

Mr. Rajchman: There is no problem. By coincident writing one can bring the elements from the low state to the high, or high to low state, depending on the linkage being positive or negative. One has to resort to the usual rewriting scheme customarily employed in memories.

E. C. Johnson (Bendix): Would you summarize the relative potentials of parametric diodes and tunnel diodes for high-speed logic?

Mr. Rajchman: I wish, and so do other people in our organization that we had a completely clear answer to this. The fact of the matter at the moment is that we have attained a higher logic-gain-speed product with the variable-capacity diode than the tunnel-diode approach. On the other hand, the tunnel-diode approach is simpler in actual organization; also, the rate at which we are making progress in the field is very much higher than the rate we are making in the other. So at the moment we just cannot answer the question fully. This is as close as I can put it.

S. Rogers (Convair): Would you comment on the problem caused by phase shifts in leads interconnecting microwave parametric logical elements?

Mr. Rajchman: This, of course, is an important design consideration, and circuits must be designed with the precise geometric location of the elements fully marked, and must be put at distances such as to correspond to the right fraction of the wave length or as artifices have been produced to make them shorter, as the case may be.

J. Ronnally (Philco): Has asynchronous logic any hope with new tunnel devices?

Mr. Rajchman: Yes, I believe this is possible. I didn't have time to go into this, but it is possible to have several sets of logic performed by tunnel diodes synchronously between clock pulses. In other words, the clock pulses could in effect be slower than the elementary logical calibration within the computer. I don't believe, I may be wrong, that a complete synchronization would be practical. I believe the composite system is more practical.

J. A. Githens (BTL): Please compare potentials of the two devices for high-speed logic.

Mr. Rajchman: I am afraid I can't say any more.

S. Cohen (Raytheon): How critical is the phase stability of the pump for the PLO?

Mr. Rajchman: It is, of course, critical; but I don't think there is any practical difficulty in maintaining the phase.

V. J. Sferrino (Lincoln Lab.): Is there a parallel effort to develop solid-state microwave power supplies?

Mr. Rajchman: I don't have any in that direction although we have made some experiments trying to use the same tunnel diode as a power supply for the phase oscillator. We have also made some circuits that are double, take two parts of power supply and then use semiconductors to double.

Mr. Sferrino: What degree of difficulty do you envision in the distribution of this power in a large system, regarding noise and time delay?

Mr. Rajchman: This, of course, is a very important aspect of the high-speed computer. One thing has to be designed very carefully. Light or electrical signals travel on the order of eight or seven inches per millimicrosecond and if we deal with this, delays are important. There are two broad ways to handle the problem. One, to make distributive devices such as the phase oscillator and take delay in your stride as you plan the design in the first place; the other approach is to make devices so very small as to make the delay negligible, hopefully. In the tunnel-diode, the second approach seems to be the more opportune one. The actual solving of this fully I think has to await the actual engineering of such a machine. I don't think one can solve this problem without stopping to make one, and one has to make the elemental functions themselves before one sees the problems in assembling many of them into units.

T. R. Finch (BTL): You and Mr. Lewin have mentioned low cost possibility of tunnel-diodes. What cost figures are you thinking of, and what precision of control of parameters such as peak and valley constants?

Mr. Rajchman: This is always an embarrassing question when one asks a research man who is optimistic by nature. The tunnel-diodes are manufactured by two large companies in this country and cost from $50 to $100 apiece, so that at the moment this isn't inexpensive. On the other hand, the fundamental element it is made from is very simple and the processing control required to maintain the properties is relatively simple, say to the making of a transistor. So I don't see fundamentally why the device couldn't be made very simply. Now I am not a production man, so I really can't say how to translate this into dollars and cents but I wish I could, especially the latter!

R. Turner (Philco): What do you conceive as a phase detector, say at 1000 mcs, to interpret phase-sensed outputs?

Mr. Rajchman: I think this can be done by simply comparing this with a continuous wave in amplitude modulation as shown in one of the slides. If you wish to obtain a signal or pulse from this, you can rectify it and feed it to other circuits. I will say that the general problem of matching high-speed cricuitry to low-speed input-output we have not looked into in great detail. Our aim was to find out whether or not it was possible to operate at these rates, no matter what the cost, and no matter what other problems were brought about.

R. A. Karend (*BTL*): Have you done any work in etching a tunnel-diode memory on a semiconductor wafer?

Mr. Rajchman: I believe the answer is yes. I have a slight amount of hedging, because we did use a number of techniques, including etching, I believe, to obtain the dots. Generally, the diodes are made by taking a fairly large crystal and producing the right impurities in the surface and then making smaller dots out of it by a number of techniques. The main difficulty is that after you obtain on a small curve of 1 mil or less it is very low resistance. So we have tried many techniques. I believe we have tried etching, but if you ask that question specially, I am not sure.

V. Vulcan (*General Transistor*): How do you prevent shunt loading of the tunnel-diode and consequent shift of its V-I characteristic in the direction of lower peak-to-valley current ratio?

Mr. Rajchman: If I understood the question correctly, the question is whether the shunting due to the capacity of the diode presents a certain shifting. The main penalty you have to pay for that is to have two waits. That is to say, you have to wait for the current to change, and then discharge the capacity. Unfortunately we see no other way than waiting. This is precisely the speed at which the device would work. This is the reason we are asking our semiconductor people to make the capacity less.

D. G. O'Connor (*GPE*): At the high frequencies used, what crosstalk problems have been encountered and what radiation loss?

Mr. Rajchman: In the case of the microwave solution and one or two of the others it is a serious problem. Most of the work that we have done recently is using printing with two ground plates, one on each side. We haven't made high frequencies yet so we don't know this for sure.

The Engineering Design of the Stretch Computer

ERICH BLOCH†

Introduction

THE STRETCH Computer[1] project was started in order to achieve two orders of magnitude of improvement in performance over the then existing 704. Although this computer, like the 704, is aimed at scientific problems such as reactor design, hydrodynamics problems, partial differential equations etc., its instruction set and organization are such that it can handle with ease data-processing problems normally associated with commercial applications, such as processing of alphanumeric fields, sorting, and decimal arithmetic.

In order to achieve the stated goal of performance, all factors that go into the computer design must contribute towards the performance goal; this includes the instruction set[2], the internal system organization, the data and instruction word length, and auxiliary features such as status-monitoring devices, the circuits, packaging, and component technology. No one of them by itself can give this hundred-fold increase in speed; only by the combining and interacting of these contributing factors can this performance be obtained.

This paper reviews the engineering design of the Stretch System with primary concentration on the central computer as the main contributor to performmance. In it, these new techniques, devices, and instructions have been pushed to the limit set by the present technology and, therefore, its analysis will convey best the problems encountered and the solutions employed.

The Stretch System

Early in the system design, it appeared evident that a six-fold improvement in memory performance and a ten-fold improvement in basic circuit speed over the 704 was the best one could achieve. To meet the proposed performance criteria, the system had to be organized in such a way that it took advantage of every possible overlap of systems function, multiplexing of the major portion of the system, processing of operations simultaneously, and anticipation of occurrences, wherever possible. The system had to be capable of making assumptions based on the probability that certain events might occur, and means had to be provided to retrace the steps when the assumption proved to be wrong.

This simultaneity and multiplexing of operations reflects itself in the Stretch System at all levels, from overall systems organization to the cycle of specific instructions. In the following description, this will be discussed in more detail.

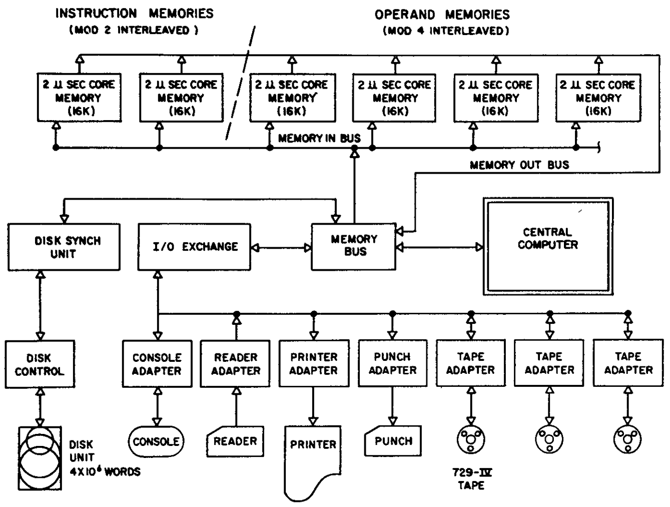


Fig. 1—The Stretch system.

If one considers the Stretch System (Fig. 1) from an overall point of view it becomes apparent that the major parts of the system can operate simultaneously:

a. The 2-μsec, 16,384-word core memories are self-contained, with their own clocks, addressing circuits, data registers and checking circuits. The memories themselves are interleaved so that the first two memories have their addresses distributed *modulo* 2 and the other four are interleaved *modulo* 4. The *modulo*-2-interleaved memories are used primarily for instruction storage; since, for high-performance instructions, halfword formats are used, the average rate of obtaining instructions is one per $\frac{1}{2}$ μsec. Similarly, a 0.5-μsec data-word rate is achieved by the use of four *modulo*-4 organized memories. The addressing of the memories and the transfer of information from and to the memories by a memory bus permits new addresses, information, or both to pass through the bus every 200 mμsec.

b. The simultaneously-operating Input/Output units are linked with the memories and the computer through the Exchange, which, after initial instruction by the computer, coordinates the

† Data Systems Division, IBM, Poughkeepsie, N. Y.

[1] S.W. Dunwell, "Design Objectives for the IBM Stretch Computer," *EJCC Proc.*, p. 20, Dec. 1956.

[2] W. Buchholz, "Selection of an Instruction Language," *WJCC Proc.*, p. 128, May 1958.

starting of the I/O equipment, the checking and error-correction of the information, the arrangement of the information into memory words, and the fetching and storing of the information from and to memory. All these functions are executed without the use of the computer, so it can in the meantime continue its data processing and computation.

c. The central computer processes and executes the stored program. Here, now, the simultaneity and multiplexing of functions has reached its ultimate.

Before discussing the computer organization, a few general features must be mentioned for completeness:

a. Word length: 64 bits plus eight bits for parity checks and error-correction codes.

b. Memory capacity and addressing: A possible 256,000 words can be randomly addressed. These storage positions are all in external memory, except for the 32 first addresses. These positions consist of the internal registers (accumulators, time clocks, index registers).

c. The instructions are single-address instructions with the exception of a number of special codes that imply the second address explicitly.

The instruction set (Fig. 2) is generalized and contains a full set for single- and double-precision floating-point arithmetic, and a full set for variable-field-length integer arithmetic (binary and decimal). It also has a generalized set for index modification and a branching set, as well as a set of I/O instructions. All told, 765 different types of instructions are used in the system.

COMPUTER VOCABULARY

INSTRUCTION CATEGORY	CLASS	MODIFIER	EXAMPLES	NUMBER OF INSTR.
VARIABLE FIELD LENGTH ARITHMETIC	BINARY DECIMAL	SIGNED UNSIGNED SAME SIGN NEGATIVE SIGN	ADD (TO MEMORY) LOAD/STORE MPY DIVIDE CUMULATIVE MPY	280
RADIX CONVERSION	BIN/DEC			32
LOGIC CONNECTS			16 LOGIC STATEMENT	48
FLOATING POINT ARITHMETIC	NORMALIZED UNNORMALIZED	SAME SIGN OPPOSITE SIGN NEGATIVE SIGN NOISY MODE	ADD (SINGLE & DOUBLE) LOAD/STORE MPY/(SINGLE &DOUBLE) DIV (WITH REMAINDER) INTERCHANGE DIVIDE CUMULATIVE MPY SQUARE ROOT	240
INDEXING ARITHMETIC	DIRECT IMMEDIATE PROGRESSIVE			43

BRANCHES
 	
STORE INST CTR
	UNCONDITIONAL
	INDEXING
	INDICATOR }
	BIT }
		IF $\left\{\begin{matrix}1\\0\end{matrix}\right\}$
		SET O
		LEAVE BIT
		INVERT BIT
				68

TRANSMIT/SWAP I/O INSTRUCTION				24
			TOTAL	735

Fig. 2—The instruction set.

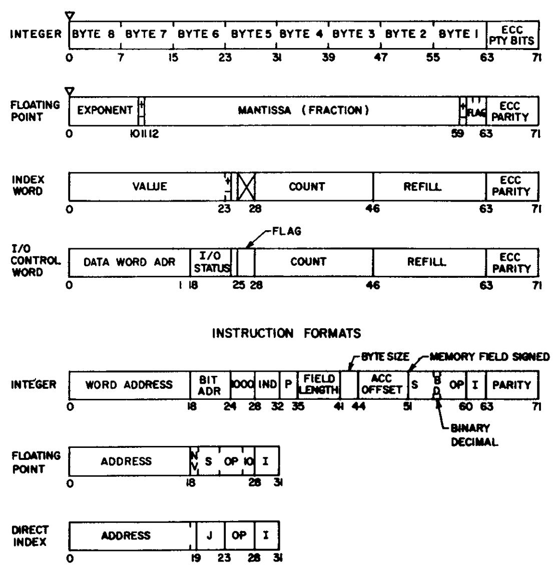


Fig. 3—Data word — and instruction word formats.

d. The instruction format (Fig. 3) makes use of both half and full words; half words accommodate indexing and floating-point instructions (for optimum performance these two sets of instructions use a rigid format), and full-word formats are used by the variable-field-length instructions. Notice that the latter specifies the operand field by the address of its left-most bit, the length of the field, and the byte* size, as well as the starting point (offset) of the implied operand (accumulator). Both halves of the word are independently indexable.

e. A general monitoring device used for important status triggers is called the Interrupt[3] System. This system monitors the flip-flops which reflect internal malfunctions, result significance (exponent range, mantissa zero, overflow, underflow), program errors (illegal instruction, protected memory area), and input/output conditions (unit not ready, etc.). The status of these flip-flops can cause a break in the normal progression of the stored program for fix-up purposes. Their status is automatically interrogated at all times.

The Stretch Computer

If one considers the internal organization of the majority of computers that have been produced during the last eight years (and the 704 is a case in point), the organization looks as shown in Fig. 4a. There is a sequential flow of instructions into the computer, and after due processing and execution, the next instruction is called from memory. Compare this with

* *Byte:* a generic term to denote the number of bits to be operated on as a unit by a variable-field-length instruction.

[3] F. P. Brooks, Jr., "A Program-Controlled Program Interruption System," *EJCC Proc.*, p. 128, Dec. 1957.

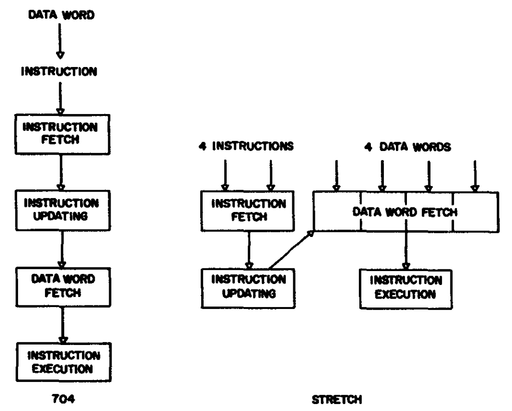


Fig. 4—Comparison of Stretch and 704 organization.

Fig. 4b, showing the organization of Stretch, where two instruction words and four operands can be fetched simultaneously. In addition, the execution of the instruction is done in parallel and simultaneously with the described fetching functions.

All the units of the computer are loosely coupled together, each one controlled by its own clock system, which in turn is synchronized by a master oscillator. This multiplexing of the units of the computer results in a large number of registers and adders, since time-sharing of the major computer organs is no longer possible. All in all, the computer has 3,000 register positions and about 450 adder positions.

Despite the multiplexing and simultaneous operation of successive instructions, the result appears as if sequential step-by-step internal operation were utilized. This has made the design of the interlocks quite complex.

Data Flow

The data flow through the computer is shown in Fig. 5 and is comparable to a pipeline which in a steady state (namely, once filled) has a large output rate no matter what its length. The same is true here; after start-up the execution of the instructions is fast and bears no relation at all to the stages it must progress through.

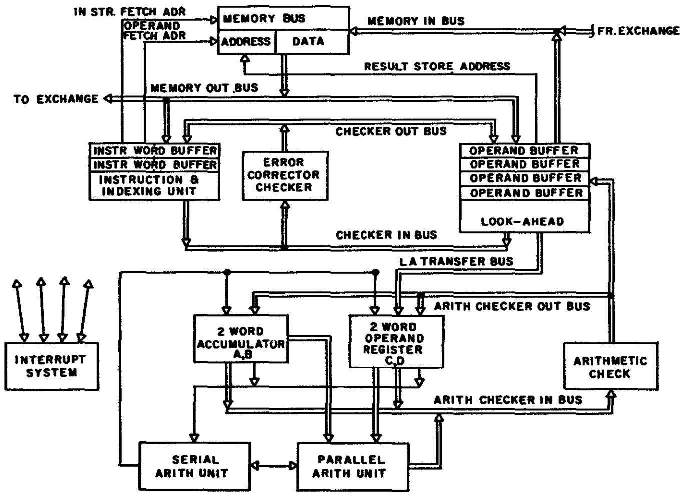


Fig. 5—Stretch Computer — units and dataflow.

The *Memory Bus* is the communication link between the memories on one side and the exchanges and the computer on the other. It monitors the requests for storage to, or fetches from, memory, and sets up a priority scheme. Since I/O units cannot hold up their requests, the exchange will get highest priority, followed by the computer. In the computer the instruction-fetch mechanism has priority over the operand-fetch mechanism. All told, the memory bus gets requests from and assigns priority to eight different channels.

Since memory can be accessed from multiple sources, and once accessed it is on its own to complete its cycle, a busy condition can exist. Here again, the memory bus tests for busy conditions and delays the requesting unit until memory is ready to be interrogated on data fetches. The return address is remembered and the requesting unit receives the information when it becomes available. To accomplish this, from the time information is requested the receiving data register is in a reserved status.

Requests for stores and fetches can be processed at a 200 mμsec rate and the time, if no busy or priority conditions exist, to return the word to the requesting unit is 1.6 μsec, a direct function of the memory read-out time.

The *Instruction Unit*[4] is a computer of its own. It has its own instruction set, its own small memory for index word storage, and its own arithmetic unit. During its operation as many as six instructions can be at various stages of execution.

The Instruction Unit fetches the instruction words from memory, it steps the instruction counter, and performs the indexing of instructions and the initiation of data fetches. After a preliminary decoding of the class of instruction, it recognizes its own instructions and executes indexing instructions. On branches, conditional or unconditional, the instruction unit executes these. In the case of conditional branches, it makes the assumption that the branch will not be successful.

This assumption and the availability of two full-word buffer registers keep the flow of instruction to the computer continuous. Therefore, the rate of instructions entering the instruction unit is for all practical purposes independent of the memory cycle.

Since, for high speed instructions, half-word formats are used, four of these at any one time can be in buffer storage. As soon as the instruction unit starts processing an instruction, it is removed from the buffer, thus making room for the next memory-word access (Fig. 6). Incidentally, half-word instructions and full-word instructions can be intermixed

[4] G. A. Blaauw, "Indexing and Control-Word Techniques," *IBM Journal*, July 1959.

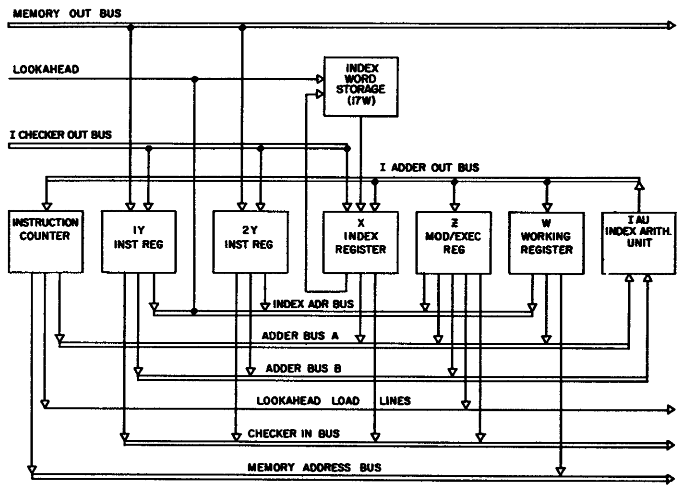


Fig. 6—Instruction unit.

within the same word, and therefore the latter can cross a word boundary. This permits maximum packing of instructions in memory and also serves as a facility for automatic program assemblers and compilers.

The adder path, index registers, and transfer bus to look-ahead complete the instruction unit system (Fig. 6). It should be noted that the index registers are part of the instruction-unit data path, therefore permitting fast access (no long transmission lines) to an index word. There are 16 index words available to the programmer. The index registers, consisting of multi-aperture cores, are operated in a non-destructive fashion, since in a representative program, the index word is used nine out of ten times without modifying it. This permits fast operation under these conditions, and additional time is only applied where modification is involved.

After processing through the instruction unit, the updated (indexed) instruction enters a level of the *Look-ahead* (Fig. 5). Besides the instruction, all necessary information, its associated instruction counter value, and certain tag information are also stored in the same level. The operand, already requested by the instruction unit, will enter this level directly and will be checked and error-corrected while awaiting transfer to the arithmetic units for execution.

An interlocked counter mechanism in the look-ahead keeps its four levels in step, preventing out-of-sequence execution of instructions, even if all information for a succeeding one is available, before the previous instruction has been started.

The pre-accessing of operands by the look-ahead and of instructions by the instruction unit leads sometimes to embarassing positions, for which a fix-up routine must be provided. Consider the program

(n)	STORE Accumulator m
$(n+1)$	LOAD R
$(n+2)$	ADD m

and assume instruction (n) is in look-ahead, waiting for execution. If $(n+2)$ now enters the look-ahead, a reference to m cannot be made, since the data stored in that position is subject to change by the STORE instruction. The look-ahead must recognize this and "forward" the result of instruction (n), when received, to the level where $(n+2)$ is stored.

Another example is the case where the instruction unit assumed that a conditional branch would not be executed. This instruction is stored in look-ahead and, when it is recognized that the branch was successful, all modifications of addressable registers made by the instruction unit in the meantime must be restored. Look-ahead in this case acts as a recovery memory for this information. A similar condition exists when interrupts occur due to arithmetic results. The look-ahead here again has the data stored pertaining to registers which were modified erroneously in the meantime. The restoring and recovery routines described break into the instruction unit processing, interrupting temporarily the flow of instruction and their indexing.

The arithmetic units described later are slaves to the look-ahead, receiving not only operands and instruction codes but also the start-execution signal. Conversely, the arithmetic units signal to the look-ahead the termination of an operation and, in the case of "To Memory" operations, place into the look-ahead the result word for transfer to the proper memory position.

Arithmetic Units

The design of the arithmetic units was established along lines similar to the design of look-ahead and the instruction unit. Every attempt was made to speed up the execution of arithmetic operations by multiplexing techniques and overlapping of the algorithm, where mathematically permissible.

The arithmetic units, consisting of the Serial Unit and the Parallel Unit, use the same arithmetic registers, namely a double-length accumulator (A,B) consisting of 128 bits and a double-length operand register (C,D) consisting of 128 bits. The reason for the use of the same arithmetic registers is the fact that at any time, a shift from floating-point to variable-field-length operation (or *vice versa*) can be made by the program. Therefore, the result obtained by a floating-point operation can serve as the starting operand for a variable-field-length operation. The chief reason for the double-length registers is the definition of maximum field length to be 64 bits. The field can start with any bit position, and therefore can cross the word boundary.

The executions of floating-point mantissa operations and variable-field-length binary multiply and divide operations are performed by the parallel unit, whereas the floating-point exponent operation and the variable-field-length binary and decimal add-type operations are executed by the serial unit. The square-root operation and the binary-to-decimal conversion algorithm are executed in unison by both units. Salient features of the two units will now be described.

The Serial Arithmetic Unit.[5] (Fig. 7) The serial arithmetic consists of a switch matrix which can extract 16 consecutive bits from *A,B* and *C,D*.These 16 bits then can be aligned in such a way that the low-order bit of a field as specified by the instruction is at the right end of the field. This wrap-around circuit then feeds into a carry-propagate adder or, in case of logical-connect instructions, into the logic unit. At the adder output, a true complement unit and a binary-to-decimal correction unit are used for subtract and decimal operations. The inverse process of extracting is used to insert the processed byte back into the register without disturbing any neighboring positions. Notice that in one clock cycle, the information is extracted, the arithmetic is performed and the result inserted back into the registers. In addition, the arithmetic information is checked by parity checks on the switch matrices and by duplication and comparison of the arithmetic procedure in a duplicate unit.

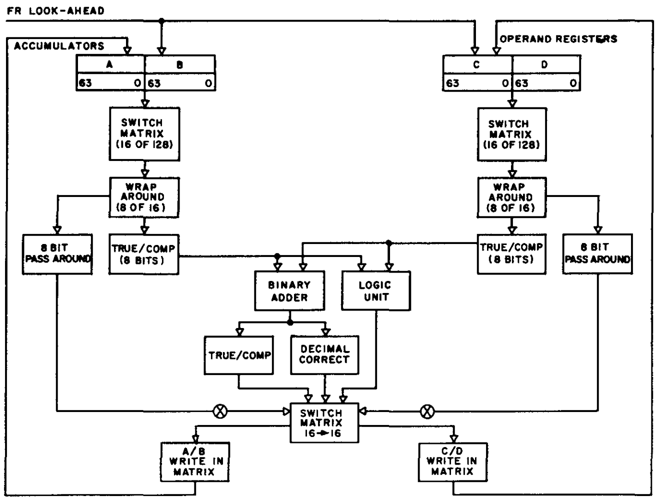


Fig. 7—Serial arithmetic unit.

Parallel Arithmetic Unit. The parallel arithmetic unit (Fig. 8) is designed to execute floating-point operations with a maximum of efficiency. Since both single- and double-precision arithmetic is performed, the shifter and adder exist in a double-length format of 96 bits. This insures almost the same performance for single- and double-precision arithmetic. The adder is of a carry-progapation type with look-ahead over 4 bits at a time to reduce the delay that normally results in a ripple-carry adder. This carry look-ahead

[5] F. P. Brooks, Jr. et al; "Processing Data in Bits and Pieces," *Trans. IRE on Electronic Computers,* June 1959.

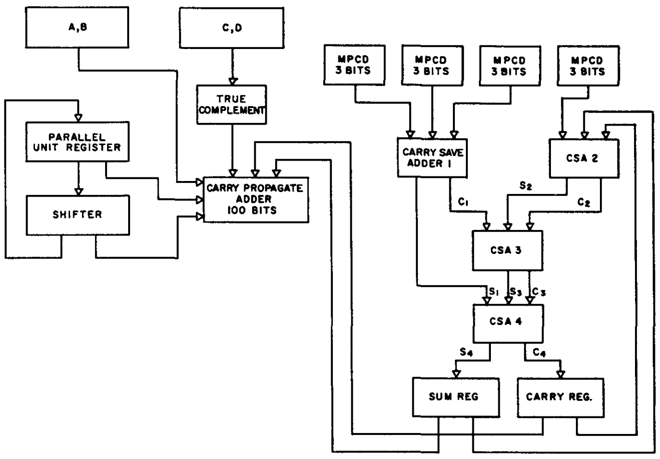


Fig. 8—Floating point arithmetic unit.

results in a delay time of 150 mμsec for 96-bit binary-number additions. All additions and subtractions are made in one's complement form with automatic end-around carry.

The shifter is capable of shifting up to 4 positions to the right and up to 6 positions to the left. This shifter arrangement takes care of the majority of shifting operations encountered under normal operation. Where higher-order shifts are required, a successive operation is set up between the parallel unit register and the shifter.

To expedite the execution of the multiply instruction, 12 bits of the multiplier are handled within one cycle. This is accomplished by breaking the 12 bits into groups of three bits each. The action is from right to left and consists of decoding each group of three bits. By observing the lowest-order bit of the next higher group, a decision is made as to what multiple of the multiplicand one must add to the partial product. Since only even multiples of the multiplicand are available, subtraction and addition of the multiples can result. The following example will elaborate this point: (*MCD* means multiplicand)

Groups				
n+4	*n+3*	*n+2*	*n+1*	*n*
		Multiplier, 12 bit group		
xx0	011	110	101	010
		Octal value		
	3	6	5	2
		If two additions of multiples were permitted		
	4×*MCD*	6×*MCD*	6×*MCD*	2×*MCD*
	−1×*MCD*		−1×*MCD*	
		Instead of subtracting 1×*MCD* in *n*+1, subtract 8×*MCD* in *n*.		
	4×*MCD*	6×*MCD*	6×*MCD*	2×*MCD*
		−8×*MCD*		−8×*MCD*
		Resulting decoding		
	4×*MCD*	−2×*MCD*	6×*MCD*	−6×*MCD*

The four multiple multiplicand groups and the partial product of the previous cycle are now fed into carry-save adders of the form,

$$\text{Sum } S = A \veebar B \veebar C$$
$$\text{Carry } C' = AB + AC + BC.$$

There are four of these adders, two in parallel followed by two more in series (Fig. 8). The output of Carry-Save Adder 4 then results in a double-rank partial product, the product sum and the product carry. For each cycle this is fed into Carry-Save Adder 2, and, during the last cycle, into the carry-propagate adder, for accumulation of the carries. Since no propagation of carries is required in the four cycles, where multiple multiplicands are added, this operation is fast and is the main contributor to the fast multiply-time of Stretch.

The divide scheme[6] has a similarity to the multiply scheme. Multiples of the divisor are used, namely, 3/2 × divisor, 3/4 × divisor and 1 × divisor. This, plus shifting over strings of ones and zeros, results in the generation of the required 48 quotient bits within thirteen machine cycles. Most machines using a nonrestoring divide method require 48 cycles for 48 quotient bits. The following example explains this technique. This scheme depends on the use of normalized divisors:

DIVIDEND	(DD)	= 101000000000000
DIVISOR	(DR)	= 1100011
2's COMP DR	($\overline{\text{DR}}$)	= 0011101
3/4 DR		= 100101001

(*a*) *Using skip over 1/0 only:*

	101000000000000	DIVIDEND
Step 1:	0011101	ADD $\overline{\text{DR}}$
	1101101	

Remainder negative, 1st quotient bit = 0; shift one position. Leading 1 indicates that next quotient bit must be 1; $Q_1Q_2 = 01$

	011010000	REMAINDER
Step 2:	1100011	ADD DR
	10010111	

Overflow: Remainder positive and $Q_3 = 1$, leading zero indicates $Q_4 = 0$

	1011100	REMAINDER
Step 3:	0011101	ADD DR
	1111001	

Negative remainder; $Q_5 = 0$; leading 1's indicate $Q_6Q_7Q_8 = 111$

Number of quotient bits per cycle:

Cycle 1:	01	= 2
Cycle 2:	10	= 2
Cycle 3:	0111	= 4

(*b*) *The same problem with both skip over 1/0 and 3/4 – 3/2 complement:*

	101000000000000	
Step 1:	0011101	
	11011010000	

Same as before, $Q_1Q_2 = 01$

Step 2:	100101001	Add 3/4 DR
	111111001	

This (by table look-up) indicates $Q_3Q_4Q_5Q_6Q_7Q_8 = 100111$

Quotient bits generated per cycle:

Cycle 1:	01	= 2
Cycle 2:	100111	= 6

In general, this method results in the generation of 3.7 quotient bits per subtraction. While the mantissa operations of multiply and divide are performed by the parallel unit, the serial arithmetic unit executes the exponent arithmetic. Here again is a case where overlap and simultaneity of operation is used to special advantage.

3. *Checking.* The operation of the computer is checked in its entirety and correction codes are employed where data transfers from memory and input-output units are involved. In particular, all information sent to memory has a correction code associated with it, which is checked for accuracy on its way from memory. If a single error is indicated, then correction is made and the error is recorded via a maintenance output device. Within the machine, all arithmetic operations are checked, either by parity, duplication, or a "casting out three" process. These checks are overlapped with the execution of the next instruction.

4. *Hardware Count.* Fig. 9 shows the percentage of transistors used in the various sections of the machine. It becomes obvious that the parallel unit and the instruction unit use the highest percentage of transistors. In case of the parallel unit this is due to the extensive circuits for multiply and to the additional hardware to achieve speed of up the divide scheme. In the instruction unit, the controls consume the majority of the transistors, because of the high multiplexed operation encountered.

5. *Performance.* The performance comparisons in Fig. 10 show the increase in speed achieved, especially in floating-point operations, over the 704. It should be noted that for a large number of problems this particular increase in all arithmetic speeds is almost proportional to the performance increase of the prob-

[6] J. E. Robertson, "A New Class of Digital Division Methods," *Trans. IRE on Electronic Computers*, vol. EC-7, pp. 218–222; Sept. 1958.

UNIT	# OF TRANSISTORS	% OF TOTAL	# OF FRAMES
MEMORY CONTROLS	10,500	6.0	2
INSTRUCTION UNIT			
DATA PATH	17,700	22.0	2
CONTROLS	19,500		3-1/2
LOOK-AHEAD			
DATA PATH	17,900	15.6	1
CONTROLS	8,600		1-1/2
ARITH. REGISTERS	10,000	5.9	1
SERIAL ARITH. UNIT			
DATA PATH	10,000	10.5	1-1/2
CONTROLS	8,700		1
FLOATING PT. UNIT			
DATA PATH	32,700	21.0	2-1/2
CONTROLS	3,000		1/2
CHECKING	24,500	14.5	1
INTERRUPT SYSTEM	6,000	3.5	1/2
TOTAL	169,100	100.0	18

DOUBLE CARDS	4,025
SINGLE CARDS	18,747
POWER	21 KW

Fig. 9—Component count.

	OPERATION	IBM 704	IBM 705	STRETCH
	1. FLOATING POINT			
	EXPONENT RANGE	$2^{\pm 128}$		$2^{\pm 2048}$
	MANTISSA BITS	27		48
	FLOATING ADD	84 USEC		1.0 USEC
	FLOATING MPY	204 USEC		1.8 USEC
	FLOATING DIV	216 USEC		7.0 USEC
	LOAD/STORE	24 USEC		.6 USEC
	2. BINARY VARIABLE FIELD LENGTH ARITH.			
	BIT RANGE			1 TO 64
16 BIT FIELD	ADD/LOAD/STORE			2.0 USEC
16 BIT FIELD	MPY			10.0 USEC
16 BIT FIELD	DIVIDE			15.0 USEC
	3. DECIMAL ARITHMETIC			
	DIGIT RANGE		1 → MEM CAPACITY	1 TO 21
FOR 5 DIGITS	ADD		119 USEC	3.5 USEC
FOR 5 DIGITS	MPY		799 USEC	40.0 USEC
FOR 5 DIGITS	DIVIDE		4828 USEC	65.0 USEC
FOR 5 DIGITS	LOAD / STORE		204 USEC	3.2 USEC
	4. MISCELLANEOUS			
	ERROR CORRECTION	NO	NO	YES
	CHECKING	NO	YES	YES
	WORD SIZE	36 BITS		64 BITS

Fig. 10—Comparison of Stretch and 705/704 operation times.

lem as a whole, since the instruction execution-times are overlapped to a great extent with the preparation and fetching of instructions. Simulation of Stretch programs on the 704 proved a performance of 100 × 704 speed in mesh-type calculations. Higher performance figures are achieved where double- or triple-precision calculations are required.

Circuits

Having reviewed the systems organization of Stretch, it is now of interest to discuss briefly the components, circuits, and packaging techniques used to implement the design.

The basic component used in Stretch is the high-speed drift transistor which exists in both an NPN and a PNP version. This transistor has a frequency cut-off of approximately 100 mc and for high-speed operation must be kept out of saturation at all times. This then explains why both the PNP and NPN version are used: mainly to avoid the problem of level translation, which would be required due to the potential difference of the base and the collector. This difference is 6 volts, an optimum point for this device.

Fig. 11 shows the basic circuit configuration. It consists of a current source, represented by the -30 volt supply and resistor R. The functional operation of the circuits consists of two possible paths represented by transistor A or C. Which path is chosen by the current depends on the condition existing on base A. If point A is positive with respect to ground by 0.4 volts, that particular transistor is cut off, making the emitter of transistor C positive with respect to the base and, therefore, making C conducting. The current supplied by the current source (6 ma) will then flow through transistor C to the load ϕ. Output ϕ, then, is positive by 0.4 volts with respect to the -6 volt reference. This indicates at ϕ the equivalent function impressed on A. At the same time, $\bar{\phi}$ is negative with respect to the -6 volt power supply by 0.4 volt, representing, therefore, the inverse of the function impressed on A. Conversely if A is negative with respect to the ground reference, transistor A is the conducting one, keeping emitter C negative with respect to its base. The current flows through transistor A, making $\bar{\phi}$ positive with respect to -6 and ϕ negative with respect to -6. Again, the output of ϕ reflects the function impressed on A, whereas $\bar{\phi}$ represents the inverse of the function.

If an additional transistor now is paralleled with A, it becomes obvious that only if both bases A and B are positive will output ϕ be positive and $\bar{\phi}$ nega-

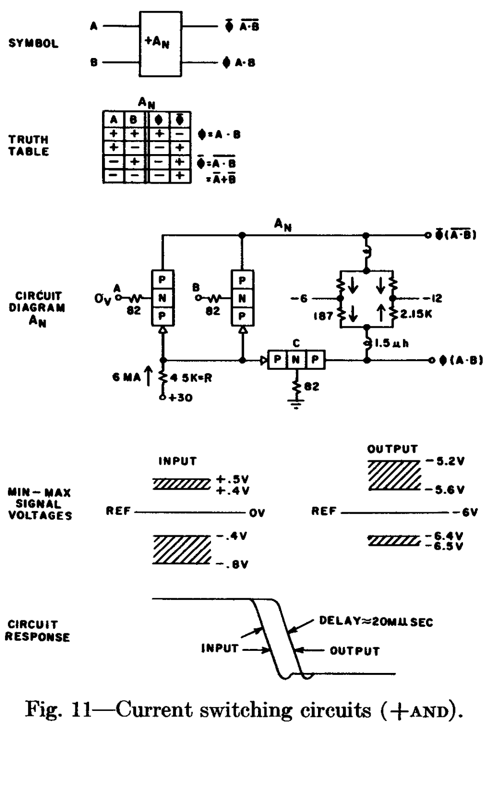


Fig. 11—Current switching circuits (+AND).

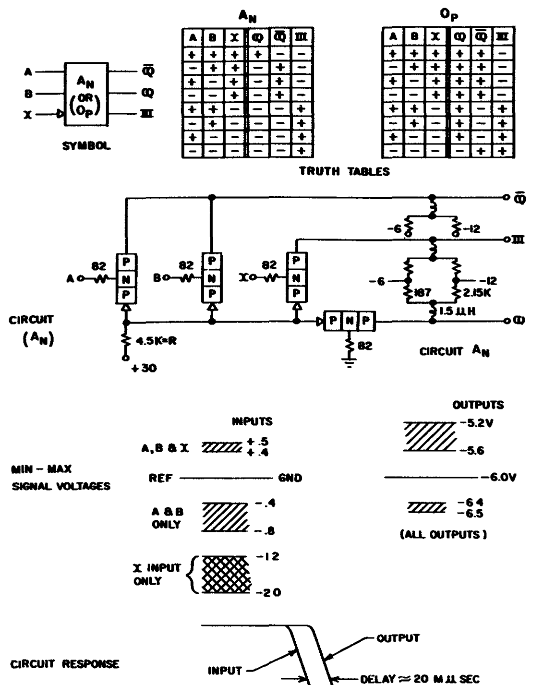
tive. If any or none of the bases A and B are positive, then ϕ will be negative and $\bar{\phi}$ will be positive. In other words, an AND function is obtained on output ϕ.

This principle, which is reflected in all the circuits, is essentially the principle of current switching or current steering.

Logical functions for the PNP circuits are, therefore, a +AND or −OR. Two outputs from each circuit block are available: the AND function and the inverse of the AND function.

A dual circuit exists for NPN transistors with input levels at −6 volts and output levels at ground. This circuit will give the +OR or −AND function.

A thorough investigation of the systems design showed that the circuits described so far are versatile enough to be used throughout the system. However, there are enough special cases (resulting from the many data buses and registers throughout the machine) that could use a distributor function or an overriding function. This caused the design of a circuit which permitted great savings in space and transistors by adding a third voltage level. Fig. 12 shows the PNP version of the third-level circuit.



A_N

A	B	X	ϕ	$\bar{\phi}$	III
+	+	+	+	−	−
−	+	+	−	+	−
+	−	+	−	+	−
−	−	+	−	+	−
+	+	−	−	−	+
−	+	−	−	−	+
+	−	−	−	−	+
−	−	−	−	−	+

O_P

A	B	X	ϕ	$\bar{\phi}$	III
+	+	+	+	+	−
−	+	+	+	+	−
+	−	+	+	+	−
−	−	+	+	+	−
+	+	−	+	−	+
−	+	−	+	−	+
+	−	−	+	−	+
−	−	−	−	+	+

Fig. 12—Third level circuit.

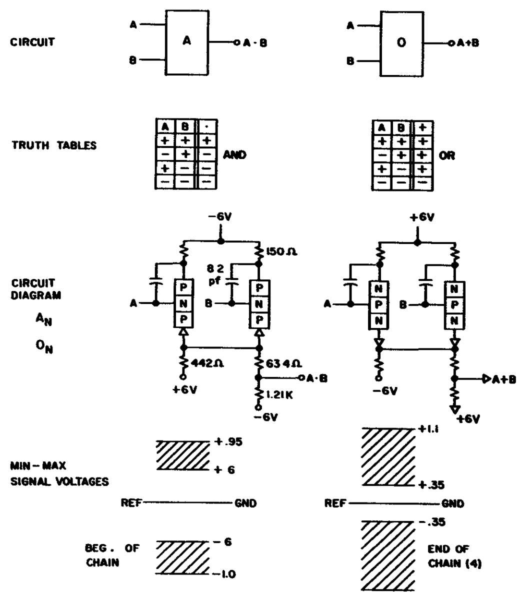
If transistor X were eliminated, then transistors A and B in conjunction with the reference transistor C would work normally as a current switching circuit, in this case a +AND circuit. If transistor X is added with the stipulation that the down level of X is more negative than the lowest possible level of A or B, it becomes apparent that when X is negative, the current will flow through that branch of the circuit in preference to branch ϕ or $\bar{\phi}$, regardless of inputs A and B. Therefore, the output of ϕ and $\bar{\phi}$ will be negative, provided input X is negative. Output *III* is the inverse of input X. If, however, X is positive, then the status of A and B will determine the function ϕ and $\bar{\phi}$ implicitly. This demonstrates the overriding function of input X.

Similarly, the NPN version (not shown) results in the OR function of ϕ if input X is negative and in a positive output at ϕ and $\bar{\phi}$, regardless of status A and B, if X is positive. Again minimum and maximum signal swings are shown in Fig. 12.

The speed of the circuits described so far depends on the number of inputs and the number of circuits driven from each load. The response of the circuit is anywhere between 12 and 25 mμsec per logical step with 18 to 20 mμsec average. The number of inputs allowable per circuit is eight. The number of driven circuits is three. Additional circuits are needed to drive more than three bases and where current switching circuits communicate over long lines, termination networks must be added to avoid reflections.

To improve the performance of the computer in certain critical places, emitter-follower logic is used as shown in Fig. 13. These circuits, having a gain less than one, after a number of stages require the use of current switching circuits as level setters and gain devices. Both AND and OR circuits are available for both a ground-level and a −6-level input. Change from a −6-level circuit to a ground-level circuit is obtained by applying the appropriate power supply levels. Due to the variations in inputs and driven loads, the circuits must be designed so that the load can vary over a wide range. This resulted in instability which had to be offset by the feedback capacitor C shown in the circuit.

All functions needed in the computer can be implemented by the use of the aforementioned circuits,



A	B	·
+	+	+
−	+	−
+	−	−
−	−	−

AND

A	B	+
+	+	+
−	+	+
+	−	+
−	−	−

OR

Fig. 13—Emitter follower circuit.

including flip-flop operation, which is obtained by tying a PNP current switch block and an NPN current switch block together with proper feedback.

Packaging

The circuits described in the last paragraph are packaged in two ways:

A circuit package using the smaller of the two printed circuit boards shown in Fig. 14, called a single card, contains AND or OR circuits. It should be mentioned that the printed wiring is one-sided and that besides the components and transistors, a rail is added which permits the shorting or addition of certain loads depending on the use of the circuits. This rail then has the effect of reducing the different types of circuit boards in the machine. Twenty-four different boards are used and of these, two types reflect approximately 70% of the total single card population.

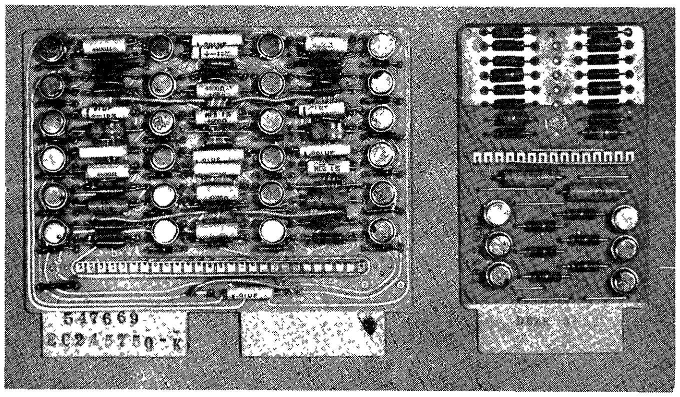


Fig. 14—The circuit package.

Due to the large number of registers, adders, and shifters used in the computer, it seems reasonable that functional packages could be employed economically, because of wide usage. This results in the high-density package also shown in Fig. 14, called a Double Card, which has 4 times the capacity of a single card and which has wiring on both sides of the board. Furthermore, components are double-stacked; and again, the rail is used to effect circuit variations due to different applications. Eighteen double card types are used in the system. Approximately 4,000 double cards are used, housing 60% of the transistors. The rest of the transistors are on approximately 18,000 single cards.

The cards, both single and double, are assembled in gates, and two gates are assembled into a frame. Fig. 15 shows the gate back-panel wiring, using wire-wraps; and Figs. 16 and 17 the frame construction, both in a closed and open version.

To achieve high performance, special emphasis must be placed on keeping noise to a low level. This required the use of a plane which overlies the whole back panel, against which the intercircuit wiring is laid. In addition, the power-supply distribution system must be of such a low impedance that extraneous

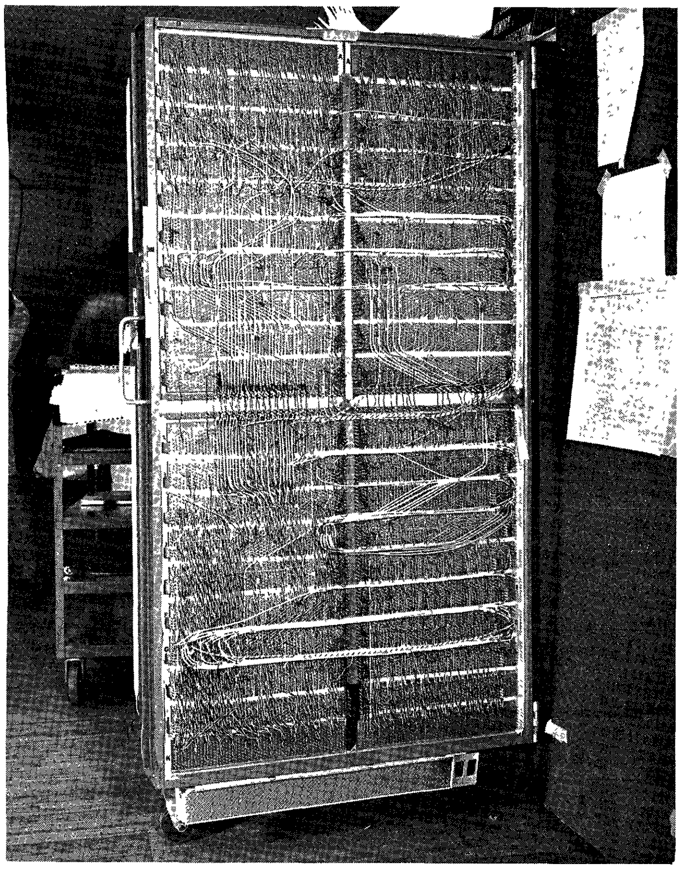


Fig. 15—The backpanel.

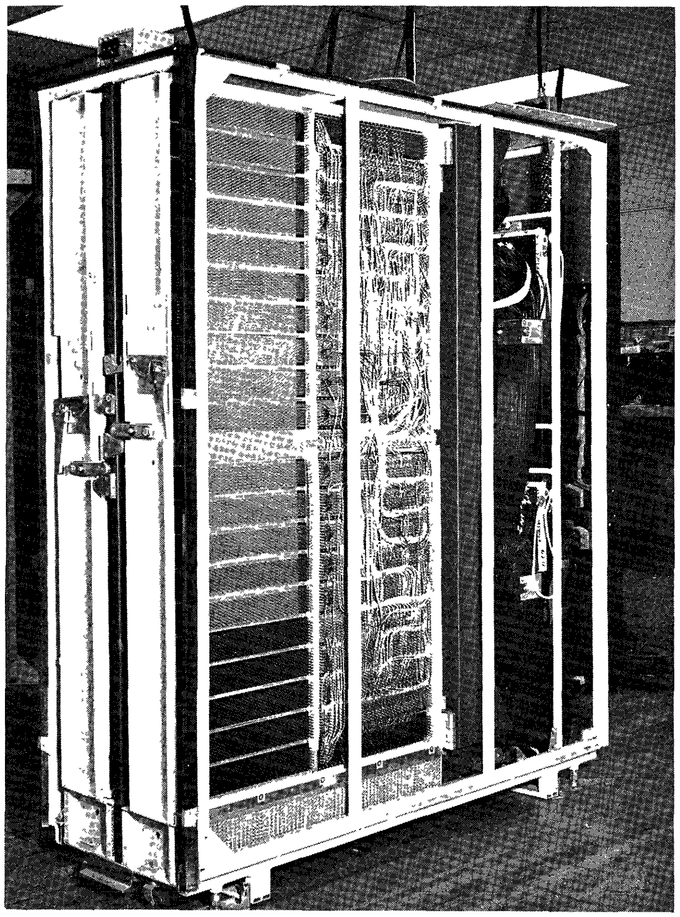


Fig. 16—The frame (closed).

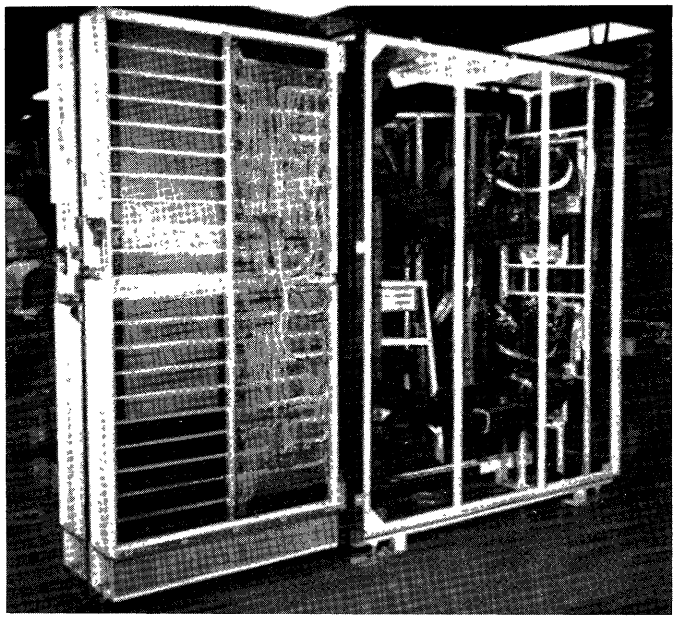


Fig. 17—The frame (extended).

noise cannot induce circuit malfunction. For this reason, a bus system, consisting of laminated copper sheets, is used to distribute the power to each row of card sockets. The wiring rules are such that single-conductor wire is used up to a maximum of 24″, twisted pair to a maximum of 36″, unterminated coax to a maximum of 60″, and terminated coax to a maximum of 100 feet. The whole back-panel construction and the application of single wire, twisted pair, or coax are calculated by a computer program to minimize the noise on each circuit node.

The two gates of a frame are a sliding pair with the power supply mounted on the sliding portion. All connecting wires between frames are coax and arrayed in layers which are formed into a drape.

Summary

The Stretch computer is an advanced scientific computer with variable facilities for floating-point, fixed-point, and variable-field-length arithmetic and data-handling facilities.

The performance goal of 100 × 704 speed is achieved by high-speed circuits, multiplexing, and simultaneous-operation technique of instruction and data-fetching, as well as overlap within the execution units. This massive overlap and multiplexing results in complicated recovery routines between the look-ahead and instruction units. These units are described in detail, as are the arithmetic units and significant algorithms used in the floating point arithmtic.

A flexible set of circuits using a current-switching technique with overriding-level facility is described, as well as the packaging of circuits on printed cards. The frame and gate concept is also shown. Performance figures and hardware count illustrate the size, complexity, and performance of the system.

Acknowledgments

The efforts and contributions of many people have gone into the engineering design of the Stretch computer. To mention all would be impossible. However, the following individuals and their groups were responsible for the units indicated; Mr. R. T. Blosk for the Instruction Unit, Mr. J. F. Dirac for the Look-ahead Units, Messrs. J. A. Hipp and O. L. MacSorley for the Arithmetic Units and Mr. L. O. Ulfsparre for the Memory Bus. The Systems Development was under the guidance of Messrs. S. W. Dunwell and R. E. Merwin.

Discussion

H. Aiken: You have told us a great deal on schemes used to speed up the computer. Now I wonder if you would spend a minute or two telling us what gains you have made in system logic, or what concessions you have had to make.

Mr. Bloch: The gains in system logic were in novel ways of performing high speed arithmetic, in the way multiplexing of operations was achieved, in the considerations necessary to interlock the individual units of the computer, and in designing complex interrupt and information-recovery networks.

C. W. Rosenthal (Bell Tel. Labs): With respect to your goal of increased speed over the 704, what portion do you attribute to faster devices and what portion to organization changes? Can you separate the effect of the individual organization changes?

Mr. Bloch: I think one order of magnitude of improvement is due to faster devices and faster circuits. The other order of magnitude of improvement is due to system organization, multiplexing and so forth. As to your second question, overlapping techniques and look-ahead contribute less than half to the performance; the remainder is due to new schemes in the execution units.

D. Hammel (RCA): What is the full time required to execute a short instruction such as an add instruction? Identify the various steps.

Mr. Bloch: This question is not so easy to answer. Because of the computer organization which is extensively overlapped, the only time that can be charged to the ADD operation is the execution time in the arithmetic unit. For a Floating Add, which I assume you have reference to, it amounts to the following: 30 per cent of the time is spent to find out what the relative pre-shift of mantissas is. About 40 per cent of the time is spent in shifting and performing the actual addition operation. The rest of the time, which is quite considerable, is spent in doing significance tests on the results, such as exponent ranges, zero operands, etc., and in checking and transfer of the information over a bus.

V. Enstein (Brooks Research): Can you mention the general characteristics of the transistors used and the achieved switching speeds?

Mr. Bloch: To answer the transistor question first: it is a drift device with a cutoff frequency of over a hundred megacycles and a forward drop of about two-tenths of a volt. The gain is 20 at end of life and the dissipation is 50 mw. Both PNP and NPN versions have the same characteristics. As far as the circuit speed is concerned, it varies from 12 to 25 millimicroseconds, depending on fan-in and fan-out. The third-level circuit shown is slightly slower than the normal current-switching circuits, due to larger level swings.

W. A. Cava (Philco): What programming procedures are necessary to produce a minimum number of interruptions in the normal sequence of operation?

Mr. Bloch: Some of the interrupt bits which trigger routines can be inhibited by the programmer. Also, the definition of the interrupt conditions is such that only extreme occurrences can bring them into play. Therefore the frequency of interrupts should be small in the majority of problems.

M. Lewin (RCA): What adjustments are required on the plug in cards from the time they are wired up until they are ready to be plugged in?

*Mr. Bloch:*The only adjustment you have to make to the cards is the clipping of the rail. This changes the configuration logically, and changes the circuit as far as load networks are concerned. This is the only change that has to be made.

D. Neumann (Lincoln Lab.): Why do you assume a branch will not take place? Use of programming loops usually has branches occurring more often than not.

Mr. Bloch: This is quite an arbitrary decision. It could have been done the other way. Once it is specified arbitrarily, the programmer is not better or worse off, whichever way it is defined.

D. H. Daggett (Convair): Would you please mention some of the considerations involved in selecting input-output equipment of sufficient speed to be compatible with the high processing speeds in Stretch?

Mr. Bloch: The system organization is set up in such a way that input-output equipment really does not interfere with the computation. The Exchange, which is an input-output computer, so to speak, takes care of this. Therefore the speed of the input-output devices is not such a consideration as it is in a machine where simultaneous operation is not possible. As far as input-output equipment on the STRETCH computer is concerned, there was no greât consideration for special input-output devices; rather, more effort was put into a novel system organization.

G. A. Sellers (Bell Labs.): Are the speeds quoted statistical averages — dependent on numbers — or absolute, — independent of numbers operated upon?

Mr. Bloch: Both. The multiply speed is worst-case. The floating-point-add speed depends on the number of pre- and post-shift cycles. The shifter is capable of shifting six bits at a time, and experience showed that within the six shifting cycles, 80 per cent of the numbers that are normally flowing through a computer can be handled.

T. R. Finch (BTL): At one time I believe you employed a ½-microsecond store, but today you showed only a block of 2-microsecond stores. Does this change result from improved system organization or necessary change due to fast store problems or what?

Mr. Bloch: I think from improved system organization. Let me mention, however, one item: I showed the 2-microsecond memories. Now the instruction unit has a memory of its own of about 16 words, used as index storage, and it runs at a speed which is comparable to the speed of the instruction unit itself. In this application it has been shown that for fast memories to be useful, they must be tightly interwoven with the computer networks.

F. H. Tendrik (Bell Tel. Labs): What is the logical use of the circuit with the "X" input?

Mr. Bloch: The circuit — third-level circuit — is an overriding function. Essentially what you can do is the following: The "X" input can be assumed to be an information bit and then normal inputs A and B might be mutually exclusive signals directing the information to one out of many registers. This is employed for shifters, read-out matrices, gating and distributing functions.

S. DeMaio (ITT Lab.): What is the access time of the memory?

Mr. Bloch: About 1.6 microseconds. This includes bus transfer test for busy and priority conditions, etc.

R. M. Horowitz (Lincoln Lab.): How much power is dissipated in STRETCH?

Mr. Bloch: The whole STRETCH system dissipates about 70 KW.

P. J. Scola (GE): Do you use marginal checking?

Mr. Bloch: Yes.

Mr. Scola: How effective is it in detecting marginal transistors and circuits?

Mr. Bloch: What you are doing in varying the voltages is checking gain, characteristics as well as frequency response of the circuits. By the way, each frame has its own built-in marginal-voltage supply.

G. E. Saltus (BTL): What is the approximate size of the central processor? What total power dissipation is associated with the central processor?

Mr. Bloch: It dissipates 21 KW and is about 30 feet long by 6 feet high by 5 feet deep.

W. Renwick (Plessey Co.): What is the present status of the STRETCH Project?

Mr. Bloch: Right now we are in the process of testing out the system units and tying them together.

A. Dowkont (Rand Corp.): When is the first delivery? What is the cost? What is the commercial availability?

Mr. Bloch: As you realize, STRETCH is designed under contract with the Atomic Energy Commission. The delivery is scheduled for May, 1960. As far as cost and commercial availability is concerned, I would rather not answer this question. As I pointed out before, right now it is strictly considered a one-shot affair under a development contract.

H. P. Peterson (Lincoln Lab.): Is there now a working, reliable, 2-microsecond 16K core memory?

Mr. Bloch: Yes, three are operating on Stretch, and two have been supplied to a customer the other day as part of the first 7090's. Many more are under assembly.

D. Dickman (Los Alamos Lab.): What is the basic cycle time of the computer?

Mr. Bloch: There is no such thing, since the individual units of the computer operate asynchronously. However, each unit has a clock which has a cycle anywhere between 200 and 300 millimicroseconds.

J. Katz (GE): Are you coding in machine language or are compilers or interpreters in use?

Mr. Bloch: We are writing essentially two compiler-type programs. One is written in STRETCH language; the other is written in 704 Fortran language.

F. Mazziotti (IBM): How many instructions per second can your machine perform in a typical scientific problem?

Mr. Bloch: Well, I don't think I am able to answer this question here. This depends obviously on what problem you are talking about and what are the housekeeping functions you are performing during the computation. I think if you look at the speeds shown before, you can interpret this for yourself.

L. Clapp (Sylvania): To what extent, if any, have you used computer techniques in the processing of your design and production data? If so, what computers were used for this program and how extensive was the effort?

Mr. Bloch: We used computers quite extensively to process logic pages, and also to compute the noise on each node of the back panel. The back panel layout and routing was done by computers. Computers used were both 704 and 705 systems.

G. A. Barnard (Ampex): Were you to continue to extend the techniques expounded here, would you comment on the widening gap between internal speeds and the load/unload speeds of input/output equipment? What about pressures to speed up the in/out equipments instead of merely using more of them?

Mr. Bloch: I don't think we are right now input-output limited, because of the philosophy the system operates under. Also, we have made great advances in higher-speed and high-storage-capacity disks.

Design of Univac®- LARC System: I

J. P. ECKERT,† J. C. CHU,† A. B. TONIK,† AND W. F. SCHMITT†

THIS TALK is a progress report and in many respects a final report on the Univac® — LARC system which has been developed by Remington Rand Univac. It is a companion not only to another talk on LARC design being given at this time but with an earlier talk given three years ago to the EJCC in New York.[1]

The talk three years ago described the objectives of a system still in the design stage. This talk describes progress since then. In order to prepare for this talk we have, of course, reviewed the one delivered three years ago. What we read was quite interesting. We found that in one sense of the word we have no progress to report. Fortunately in the larger sense of the word we have great progress to report. The area in which no progress, or more precisely no change, has been made deals with the design objectives of the LARC system. All of the goals set up for the LARC system three years ago have been very safely achieved. None of the speeds of any of the operations in the LARC system have been changed. Basic decimal-checked addition takes exactly one microsecond as described three years ago. Memory cycle times of both one microsecond and four microseconds have been adhered to.

By a conservative design approach and by choosing existing components, we expected LARC to be completed in 1958. Although we did use then existing components, modifications required to increase reliability on some of the components, along with the logical complexity of the system, delayed completion until 1959. If we had compromised our speed or reliability objectives, the delay would have been much shorter.

Now we come to the point where great and for computer engineering unusual progress can be reported. We have precisely met and proven the validity of our original goals on the world's fastest, most versatile computing and data processing system as described in our earlier published talk. Fig. 1 shows a typical LARC system as it would appear in an operating installation.

The original objective of the LARC system was to build a good system which pressed the limits of the art while still maintaining balance between its various elements. We feel a good system is one in which everything involved is accomplished in a manner which is not only dependable but consistent in the degree to which it presses the art at the time the design is frozen. The talk of three years ago established the goals. We have now accomplished them in the area of logic circuits, high-speed memories, input-output equipment, and most important of all in overall system organization. Further, we have established an automatic procedure for the prodigious record keeping required to carry out such a design. The companion paper will discuss this point further.

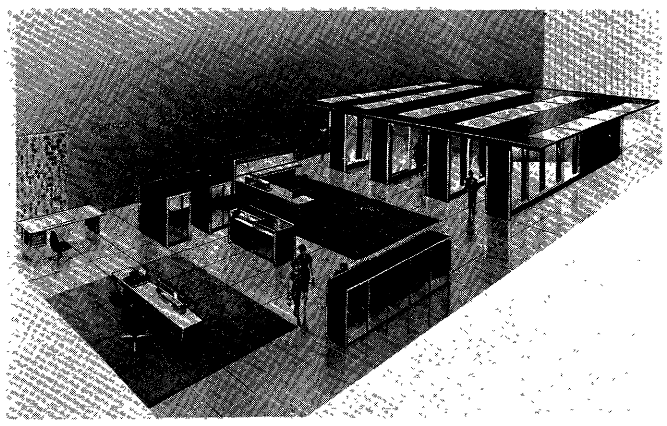


Fig. 1—A typical LARC system as it would appear in an installation.

By a balanced system, we mean a system designed to obtain the greatest work output for the costs involved. Since the problems faced by different users vary, considerable flexibility of the input-output equipment and the memory equipment of the computer system is required. In addition to providing more flexibility in these areas than any other existing system, the LARC system has the added flexibility of enabling either one or two computing units to be included as part of the system to allow increased speed.

COMPUTERS AND PROCESSOR

A basic LARC system contains a Computer and a Processor, each of which has most of the attributes of a general purpose Computer but perform different functions in the system. The primary function of the Processor is the flexible, parallel, and coordinated control of all input-output equipment and transfers between this equipment and the memory system. The Computer is designed to perform rapid arithmetic computation with a minimum of interference.

The two Computers in an expanded system (see Fig. 2) can be programmed and controlled to solve jointly a single problem; or each can solve inde-

† Remington-Rand Univac Div., Sperry-Rand Corp., Phila., Pa.

[1] J. P. Eckert, "Univac-Larc, The Next Step in Computer Design," *Proc., 1956 EJCC,* (A I E E Special Publication T-107) pp. 16–19.

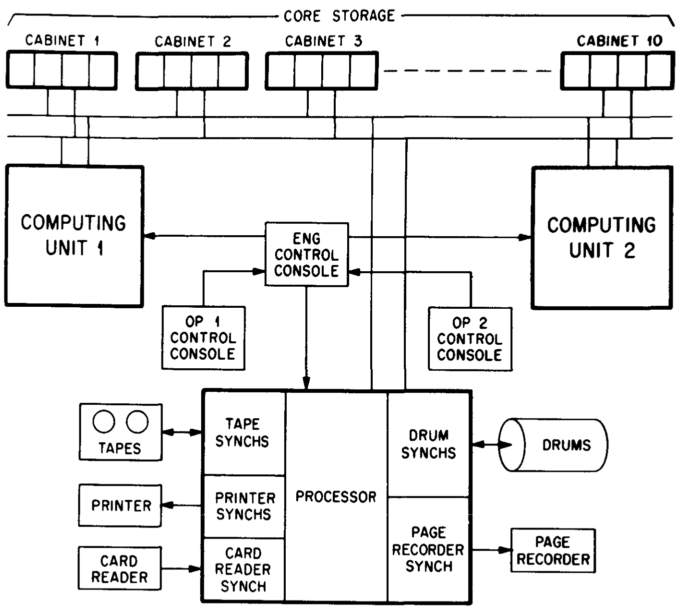


Fig. 2—Block diagram of a LARC system.

pendent problems. The Processor is designed to take care of the input-output and other on-line equipment needs of both Computers and to do any necessary editing of the input-output data. If input-output demands are not excessive, it may also be used to run various side routines such as sorting and merging. It receives compact instructions from the Computer and expands them by program to provide the control, interlocking, and timing required to operate simultaneously a large group of on-line equipment. This function requires only limited arithmetic ability.

To allow for simultaneous communication between the various units (a necessity with the high degree of parallel operation achieved) with a minimum of switching circuitry, a time-slot system of inter-unit communication is used. Time-slotting of the elements of a computer system is quite new and is one of the features which has given LARC such a great increase in speed and flexibility without a corresponding increase in cost.

In LARC we sought to minimize the total amount of electronic equipment by considering the logical design as a whole, rather than seeking to optimize the equipment required for each individual instruction. For example, there are many methods of performing multiplication which reduce the number of individual addition operations required without expanding equipment requirements. We examined a number of these multiplication methods and picked the arrangement which required the minimum amount of equipment, not just for multiplication but for all of the machine instructions, frequently making compromises on some specific instruction. A modified short-cut multiplication procedure was finally employed that requires only one addition per multiplying digit and allows 11-by-11 digit products to be formed in 8 microseconds. Floating-point multiplication of 9-by-9 digits also takes 8 microseconds, the extra time being used for normalizing. The division process is of fixed length, requiring 5 cycles or $2\frac{1}{2}$ microseconds per quotient digit. It requires 28 microseconds for a floating-point division or 32 microseconds for a fixed-point division.

The pulse code in LARC was carefully chosen to yield a ninety percent chance of a single digit superposition error being detected. Of course, since most such errors occur on several digits, the chance of not detecting such an error is remote. However, where a single digit must be transferred special checking circuits are employed to reduce this possibility; or some special circumstance is noted which makes the chance of detection far greater.

The names Processor and Computer assigned to the central units of the LARC are somewhat misleading. The entire LARC system may be considered a data processor. What we call the Processor in the LARC system contains a computer of its own in addition to all of the circuits necessary to synchronize the on-line equipment into the system. In designing the LARC system, we carefully avoided incorporating any feature that would increase costs unless it realized a much greater proportional increase in over-all speed and performance. On the basis of this criterion alone, the incorporation of a computer within the Processor is justified, although it does provide other important benefits by way of increasing flexibility and simplifying programming and communication within the system. If the LARC Computer were designed to perform the duties of the Processor, it would require additional instructions and other facilities. In such a system, all of the circuits devoted to synchronizing the on-line equipment would still be required. In a typical LARC system, about two-thirds of the Processor consist of circuits exclusively involved in synchronizing on-line equipment. The Processor computer represents the remaining one-third of the Processor or about one-sixth of the equipment and cost of the Computer and Processor combined. Considering also the cost of the on-line equipment used in the system, the incorporation of a computer in the Processor has contributed between one-seventh and one-eighth to the total cost and complexity of the LARC system. The LARC Processor, however, may relieve the Computer of half of the work load it would otherwise have to bear. We thus have achieved as much as eight percent increase in speed for every one percent increase in cost resulting from incorporating the computer in the processor.

Since the programs for the Computer and Processor are stored in separate memories, debugging can be done independently, so that the programming is usually made easier. Thus we have speeded up LARC without overstepping the state of the art either in circuits or in the demands put upon present day programming technology.

Storage

There are four levels of storage in the LARC system which differ in speed, capacity, and cost per character.

The first level of storage operates on a one-microsecond cycle. It consists of a number of registers that may be used interchangeably as accumulator registers for storing operands and results or as index (B) registers for storing constants used in addressing operations. Up to 99 12-digit one-microsecond registers may be included in each Computer unit.

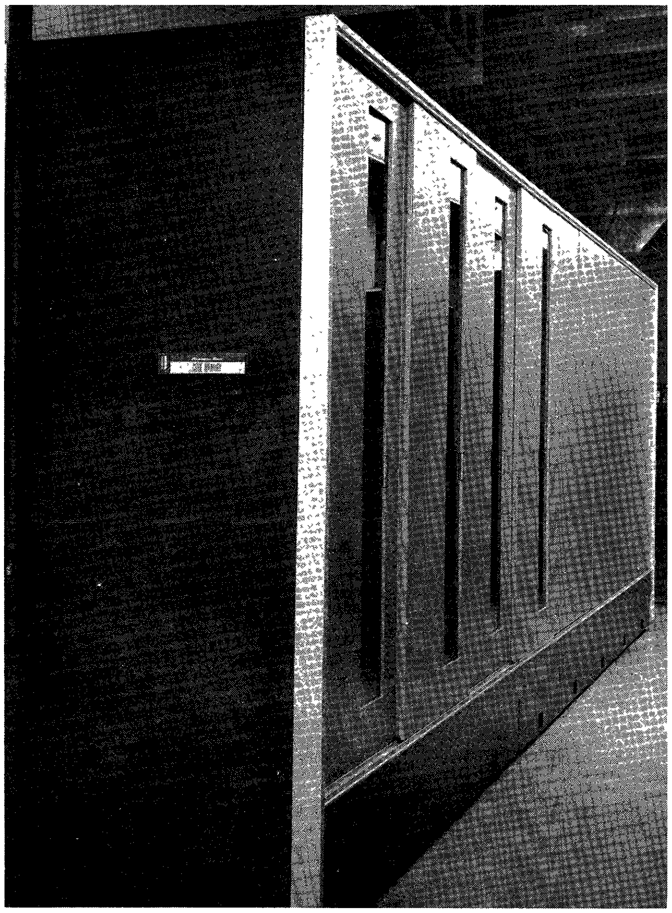


Fig. 3—A 10,000 word (600,000 bit) core memory unit.

The second level of storage is a ferrite-core storage which has a read-write cycle of four microseconds. Fig. 3 shows a 10,000-word (600,000 bit) core memory cabinet. It is accessible to both Computers and the Processor. It serves as both the main storage and buffer storage of the system and as a common communication link between the Computers and the Processor. To increase the rate at which reference may be made to the main storage, it is divided into 2500-word modules. Since the Computers and the Processor have access to the same storage, they can control each others' instruction sequences as desired. Nearly 100,000 12-digit words of ferrite-core storage may be included in the system (6,000,000 bits of core storage).

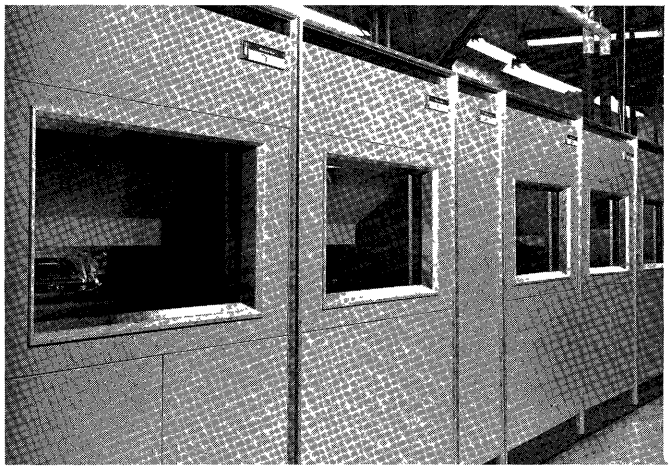


Fig. 4.

The third level of storage consists of movable-head magnetic drums. Fig. 4 shows a group of such units. Data can be transferred between the main storage and the drums over one of several simultaneous drum circuits at a data transfer rate of 500,000 decimal digits per second (2,500,000 bits per second). A maximum of 72,000,000 digits of drum storage (360,000,000 bits of drum storage) may be included in a system. The drum units are capable of transferring information twenty times as fast as the tape units, and match the high computing rates of the LARC system. In addition they provide for random access in a fraction of a second. Several drum synchronizers are employed so that computation and printing can proceed in parallel with the loading and unloading of drums to the tape units.

The moving-head drums themselves are designed very conservatively. They rotate at 884 RPM. They are recorded at 448 pulses to an inch and 29 channels to the inch. Several times this pulse rate and channel density have been achieved in the laboratory on these drums. The magnetic heads which move on a carriage along the top of the drums are moved with moderate accelerations. This is possible since the drums are used in pairs, first moving one head while another head is reading and then alternating so that the synchronizer is never idle. This arrangement allows a very conservative approach to the mechanical problems and a fool-proof head-moving mechanism of simple and rugged design driven by a servo motor. The drum assembly is mounted in an air-tight enclosure. Inside air is circulated through special filters to collect any dirt particles. The design allows the units to be opened for repair in a non-air-conditioned room. The magnetic heads fly on the hydrodynamically generated air film generated by the rotating surface of the drum. Fig. 5 shows a head flying on the surface of a drum. No filtered high pressure air supply which might direct contamination under the head is needed. A simple mechanism lowers the head on the drum without causing even momentary contact with the drum. An electrical circuit detects any contact between head and drum and instantly retracts the head if any contact occurs.

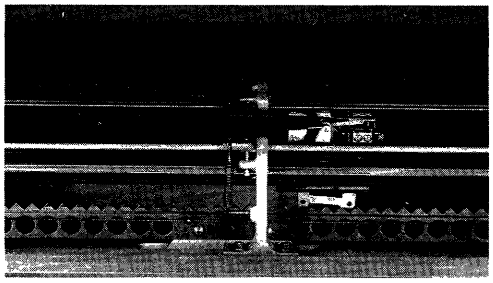


Fig. 5—A head flying on the surface of a drum.

Flying-head drum units have been life-tested for well over a year's continuous operation to ensure that their reliability is consistent with the reliability of the solid-state circuit equipment. The order of reliability achieved by these drums is believed to exceed considerably that obtainable on tape or disc equipment at this time.

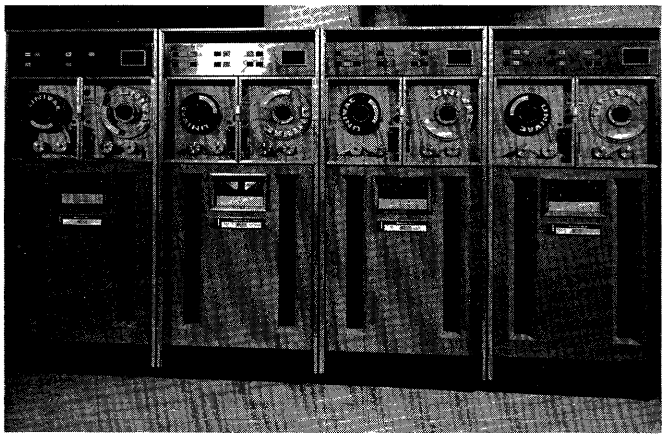


Fig. 6 A group of UNIVAC tape units.

The fourth level of storage consists of magnetic tape units which transfer data at 25,000 alphanumeric characters per second. Fig. 6 shows a group of Univac tape units. The tape units provide for input-output and long-term storage of data. LARC presently uses medium-speed tape units which not only have the advantage of being compatible with other Univac systems and off-line input-output equipment but have the advantage, due to moderate pulse densities and moderate tape velocities, of being quite reliable. Faster tape units as they become available may be used with the LARC. Tape speed is not usually a limitation, however, since the bulk of the input-output load has been taken off of the tape units in the present LARC system by the drum units.

On-Line Equipment

The on-line equipment includes, in addition to the tapes and drums used for input-output and intermediate storage:

1. An electronic page recorder, which employs a cathode-ray tube and microfilm, as shown in Fig. 7. It provides high-speed recording of output data. Sixty-four symbols are available for tabulating and curve plotting. Plots are complete with titles, scales, and grid patterns. While originally specified at 25,000 characters per second, this has been changed to 20,000 characters per second to allow more accuracy in the curve-plotting mode. Fig. 8 is a chart of the units for a typical and for an expanded LARC system.
2. Electro-mechanical line printers for multiple-copy printing of numerical characters at 720 lines per minute or printing of combined alphabetic and numeric characters at 600 lines per minute. The synchronizers will provide signals for printers operating up to 1200 lines per minute.
3. A card reader for introducing data into the system directly from punched cards.

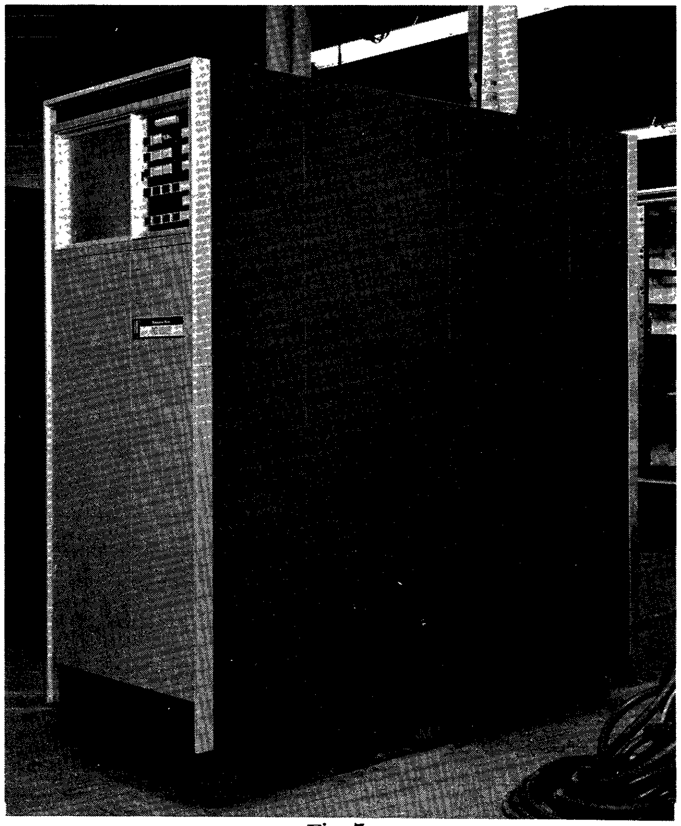


Fig. 7.

EQUIPMENT NAME	TYPICAL	EXPANDED
Magnetic Core Storage Units (2500 words each)	8	39
Computers	1	2
Multipurpose Fast Registers (per Computer)	26	99
Processor	1	1
Drum-read Synchronizers	2	3
Drum-write Synchronizers	1	2
Tape Read-Write Synchronizers	2	4
Electronic Page Recorder Synchronizer	0	1
High-Speed Printer Synchronizer	1	2
Card Reader Synchronizer	0	1
Console Printer Synchronizer	1	1
Tape Positioning Checker	1	1
Magnetic Drum Storage Units (250,000 words each)	6	24
Uniservo II Magnetic Tape Units	12	40
Electronic Page Recorders	0	2
High-Speed Printers	1	2
High-Speed Card Readers	0	1
Control Consoles	1	2
Numeric Keyboards (one per Console)	1	2
Alphanumeric Console Printers (one per Console)	1	2

Fig. 8—Modular units of a typical and completely expanded UNIVAC LARC system.

4. Console typewriter printers with an attached paper-tape reader and punch.

The original design called for a Processor to contain a minimum of five "synchronizers" and a maximum of eight synchronizers as required for simultaneous operation of the various pieces of on-line equipment. The capabilities of the LARC system turned out to be such that in order to provide more flexibility for customer requirements the minimum was raised to seven and the maximum to fourteen. The synchronizers in themselves have flexibility in the pulse rate they can accept and this, along with the concept of the Processor and the optional number of synchronizers that may be used, is one of the more important features in obtaining the high speed and flexibility of the LARC system.

LARC Consoles

Supervisory control of the LARC system is achieved through the use of the LARC Operator's Console and the LARC Engineer's Control Console. Fig. 9 shows an operating console to the left and an an engineering console to the right. The LARC Operator's Console includes a decimal display unit, an automatic typewriter, and an input keyboard. This console is designed for the utmost simplicity of operation and includes only the manual-intervention and start and stop buttons. Certain signals from the on-line apparatus are also displayed where operator attention is required.

The LARC Engineer's Control Console includes a replica of the Operator's Console and in addition, all necessary equipment for the monitoring and control of all of the LARC units. A special feature of

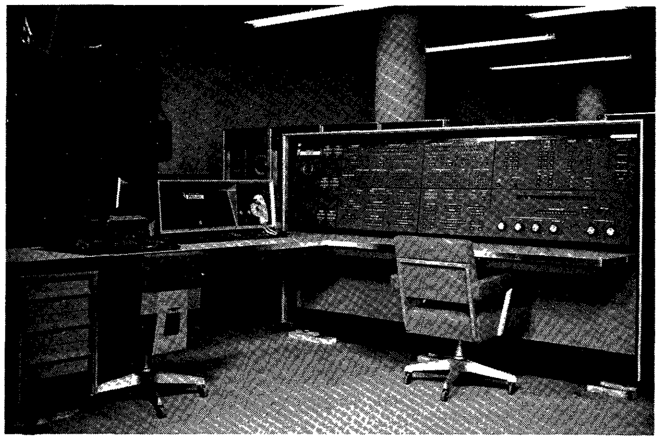


Fig. 9—An operating console to the left and an engineering console to the right.

the consoles is the ability to monitor the operations of the computing unit in a unique way. The digital display units and corresponding binary display units may be connected to any one of a number of key points of the computing unit. Synchronizing equipment allows these registers to display at a particular pulse time and program step selected by the engineer. Various modes of operation, including an error-display mode for trapping both permanent and intermittent faults, greatly reduce trouble shooting time. The console also includes all voltage-monitoring and alarm indicators for the various units, power control for the system, marginal-checking controls, and all necessary error and contingency indicators.

Complete Parallel Operation

The designers of LARC realized that even with automatic programming some programs must first be written by hand; and further, that programming must not be so difficult to understand that the people maintaining these machines cannot interpret the situation when trying to locate a fault. For these reasons, all of the parallel time-saving features built into LARC were done in a way that would require the minimum of planning on the part of the programmer and would be as similar as possible to his present programming techniques. Therefore, in our system design we have been consistent with the straightforward electronic and mechnical designs employed. Table I shows a list of the equipment in simultaneous operation in a maximum LARC system.

TABLE I

Complete Parallel Operation in LARC

(A list of equipment in simultaneous operation in an expanded LARC system)

1. *Input-output equipment*
   Read from 3 drum units
   Write on 2 drum units

Position drum head assemblies
Read from 2 tape units
Write on 2 tape units
Position a tape unit or check read a tape unit
Rewind all tape units as required
Print on 2 line printers
Record on electronic page recorder
Print on console printer
Read fast card reader
Operate decimal display unit

2. *Computers*
   Compute on data by 2 computing units
   Edit or compute on input-output data and control on-line equipment by input-output Processor
3. *Operation of instructions in computing unit*
   Different parts of five consecutive instructions are performed at once.
4. *Decimal arithmetic*
   60-bit parallel self-checking arithmetic circuits
5. *Automatic checking*
   During operation of instruction examine for possibility of tracing, programmer making a mistake, and computer making a mistake.

Without exception, we have obtained overall system speed without resorting to difficult programming tricks such as might require a knowledge of the detailed timing of the machine and the setting up of complicated interlaced patterns of data and instructions. There are many examples of the point we are trying to make here. The memory organization is one of the best examples. At the time the LARC was conceived we decided to have a very fast core memory (one-microsecond cycle time) in combination with a much larger somewhat slower core memory (four-microsecond cycle time).

Our first thinking on how to use these two memories was to take all information out of the slower memory and put it in the faster memory — instructions, operands, index numbers, etc. Computing would then be carried out almost entirely from this fast memory. When we were finished, a group of operation results from the fast memory would be returned to the slower memory. This idea led to all kinds of difficulties. First, information must be taken out of and put back into the smaller fast memory both rapidly and frequently in most of the problems studied. The large memory, being four times slower, would have to be separated into several parts so that by a time-interlacing system it could keep pace with the small memory. In turn the smaller, fast memory would also have to be broken into at least two or three parts so that parts could be loaded and unloaded while another part was in use. In addition, the fast memory could no longer be a hundred words or less and still be effective. It would usually have to be at least one thousand words. Even then, this complicated process would not work on many problems where the nature of the information coming in is such that one does not know for a few hundred words ahead where his next instructions and data are coming from. Since LARC is a very fast new machine and will be used in problem areas which have not been well investigated, we did not feel that the introduction of such restrictions on the programming would be practical. Less is gained by this arrangement than one might expect.

Instruction routines are frequently long enough between transfer instructions that they might just as well be taken from the slower memory, interlaced in the same way as would be required to transfer to the fast memory. Some small loss of time using only the slower memory can occur when a transfer order is encountered due to interruption of the interlace pattern. A similar argument will often hold for strings of data. Thus the faster memory, except for accumulator and index registers, does not help the speed situation much but would be a considerable program complication. In LARC, however, to facilitate matters still further rather than purely time-slotting all of the memories together to get a high speed flow of information, we have done this in such a way that the interrelationship is automatic and does not have to be programmed. Up to eight one-half-microsecond subdivisions of the four-microsecond memory cycle are provided for time-slot operation. The programmer does not have to plan how to intermix information in order to achieve the time-slotting necessary to match memory speeds to the Computer. Because of the time-slot system, the four-microsecond memory system looks almost like a one-half-microsecond memory in an overall system sense.

In this system it is possible for the Computer to ask for information which is not yet available. This is because the Computer receives orders ahead of their execution time in order that circuits have time to set up without delaying the operation. Since LARC is to achieve its speed without the programmer's being required to give attention to timing problems occasioned by parallel operation, special circuits accommodate the situation when inconsistent logical demands occur. These circuits very rarely delay computation in actual operation.

When two computing units are used in a LARC system, the common memory system allows either separate operation of the units or, alternately, any degree of interplay desired. A common memory system and certain common instructions provide the means of interrelating operation. To handle the priority problem involved in the parallel operation of many different units, the memory units give preference first to the input-output synchronizers, then the Processor, and finally the computing unit.

One kind of parallelism not commonly thought of as parallelism occurs with LARC's one-microsecond decimal self-checking adder. Clearly, we could convert input data into binary form and checking could be done by program. These extra operations take additional time as well as additional programming effort. In LARC's decimal-checking adder these operations are effectively combined in parallel with the arithmetic operation. A binary unchecked adder would have to be considerably faster to equal the speed of LARC's decimal-checking adder. The adder is normally used as part of a sequence in which an instruction is obtained and corrected as necessary by an index register, operands are obtained, exponents are sensed, numbers are shifted as necessary, the sum is obtained, and finally the sum and new exponents are put in an arithmetic register. All of these operations take place in a four-microsecond interval.

LARC does not expect the programmer to take care of such timing and interlocking problems. Parallelism has been obtained in a system that does not reduce flexibility or require advanced programming technology.

We feel that LARC is a very important step forward in system design in that all of its units are in balance both in regard to speed, reliability, and cost for the present state of the art. Sufficient flexibility has been designed so that as new auxiliary equipment is developed, it can be effectively added to the system. Univac I was a better-balanced system than any other computer of its era. This statement has never been challenged. The LARC system is similarly well balanced.

Discussion

See "Design of Univac-LARC System: II."

Design of Univac®- LARC System: II

H. LUKOFF†, L. M. SPANDORFER†, AND F. F. LEE†

THE engineering design of the LARC solid-state computer was a monumental challenge.

The initial job in the LARC development program consisted of determining the type of circuitry and logic that would meet the speed requirements, which called for a decimal self-checking adder which could add two 11-digit numbers in 1 microsecond and multiply them in 8 microseconds. The basic machine timing was determined by the multiplication instruction; it requires 11 additions of partial products, several cycles devoted to the generation of multiples of the multiplicand and a few more cycles for handling signs and transfer of results to a register. Sixteen operations have to be accomplished in 8 microseconds; thus the basic repetition rate is 2 megacycles or, in other terms, information must be capable of entering the arithmetic circuits every half microsecond.

We realized that the circuitry had to be faster than anything then devised in order to meet these speed specifications. A program was set up to consider all conceivable variations of solid-state circuits in order to select the best circuit. Our definition of "best circuit" was primarily based on a figure-of-merit formula given in Fig. 1. The formula is based on the following considerations: The more drives or output a circuit has, the more logically useful it is. Similarly, the more logical levels included within the circuitry, the better it is. In particular, an AND/OR circuit is logically more powerful than an OR circuit by itself.

$$\text{Figure of Merit} = \frac{\text{(Drives) (Logic Levels) (Repetition Rate)}}{\text{(Transit Time) (Cost)}}$$

4534-R1

Fig. 1.

On the other hand, the greater the transit time (electrical delay plus a portion of the rise time) the poorer the circuit. Also, the number of times the circuit can be used per second is an important factor that determines how many will be needed for a given task. A circuit having a long recovery time, such as a blocking oscillator, would be poor. Finally, the lower the cost of the components, the greater the figure of merit. Reliability is also reflected into the figure of merit through the cost factor. In general, the lower the cost, the fewer the components involved and the greater the reliability. As a first approximation, all factors in the equation are assumed equally weighted, but it should be remembered that low transit time per logical level was what was wanted most, but not at too high a price for the amount of equipment involved. If for some reason one factor is considered more important than another it can have a multiplier or exponent applied. On this basis the circuit shown in Fig. 2 was selected. If the figure of merit is normalized at 100 for the circuit selected, it may be compared to others, such as the DCTL with 2.8 or the transistor-transformer combination having a figure of merit of 4.1, or a circuit similar to Fig. 2 with a feedback diode to prevent saturation. This latter combination produced a figure of merit of 52.5. The circuit in Fig. 2 is recognized as a basic AND inverter or OR inverter circuit. The surface-barrier transistor was selected for use with this basic LARC circuit because it was the only high-speed transistor that was reliable and in mass production at the time of circuit design.

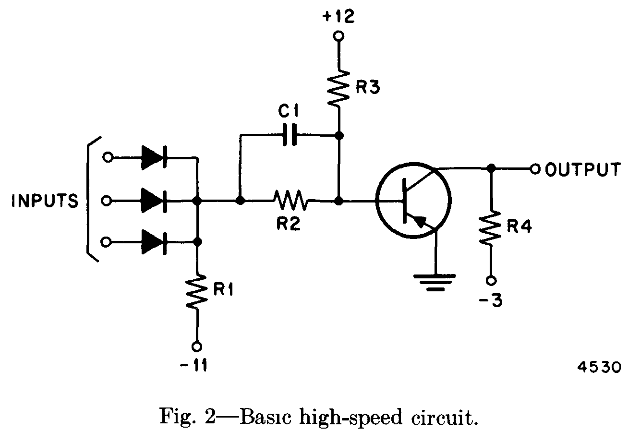


Fig. 2—Basic high-speed circuit.

Diodes are used to perform logical operations, and transistors perform the negating and amplifying function. The same diode network acts either as an AND circuit or an OR circuit, depending upon the polarity of the incoming signal. When used as an AND circuit, all inputs must be low ($-3v$) in order for the transistor to turn on. When used as an OR circuit, any input going high ($0v$) will act to turn the transistor off. The transistor is operated into saturation and at cutoff; therefore, the output swings from -3 to approximately 0 volts, and is directly coupled to the next circuit. Effective switching occurs during the first ½ volt of the transition. The resistor network R_1, R_2, and R_3 provides the proper transformation of DC levels to the base of the transistor. The capacitor, C_1, acts to supply extra current in initially turning the transistor either on or off. R_4 is used to provide a minimum load on the circuit, to limit the storage time of the transistor, and also to

†Remington-Rand Univac Div., Sperry-Rand Corp., Phila., Pa.

provide current for discharging the stray capacitance. The circuit is designed to accept a fan-in of as many as 13 inputs and is capable of providing 3 full drives output. However, with the use of mutual exclusion (load sharing) it may fan out to several additional places. The maximum output capacity allowed for this circuit is 80 $\mu\mu$f.

Now let us consider how the basic circuit is used to perform logic in LARC. The circuits are used in cascade, so that they form an AND/OR, AND/OR, etc., chain. Ideally, it would be desirable to continue in this fashion. However, pulse timing becomes inaccurate and therefore retiming must occur in certain places. This is accomplished by means of the pulse-former circuit shown in Fig. 3. This circuit permits a logical operation to be performed at its input. The first transistor stage operates a gating network which then selects either a positive or negative timing pulse (90 mμs. wide) to set or reset the following flip-flop combination. The flip-flop can only be set or reset at precisely-timed one-half microsecond intervals. A chain of logic is depicted in Fig. 4. The sequence is started when pulse-former 1 initiates a transition at one of the timed intervals. The wavefront progresses through the chain of AND/OR circuits where the necessary logical operations take place. The maximum transit time per circuit under worst-case conditions is forty millimicroseconds. Allowing for 9 levels of logic, the wavefront arrives at the timing gate of the second pulse-former 360 millimicroseconds after it leaves the first pulse-former. Since the timing pulse at the second pulse-former is scheduled to arrive in 500 millimicroseconds, the difference of 140 millimicroseconds is allowed for timing pulse jitter, pulse-former operation and safety factor. Under normal operating conditions the average propagation time of the signal has been found to be 20 millimicroseconds. Thus there is a very large factor of safety in the circuit timing. The circuit transit time corresponds to 25 megacycles. If less than 6.6 levels of logic are employed between pulse-formers, there is the danger of the signal's propagating too fast and arriving at the pulse-former before the previous clock pulse has disappeared. To prevent this, it is necessary to add sufficient delay elements to pad the chain to minimum delay levels. Padding delay has been required in fewer than 10 per cent of the logical chains.

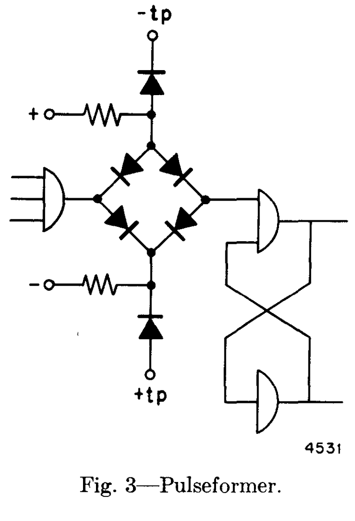


Fig. 3—Pulseformer.

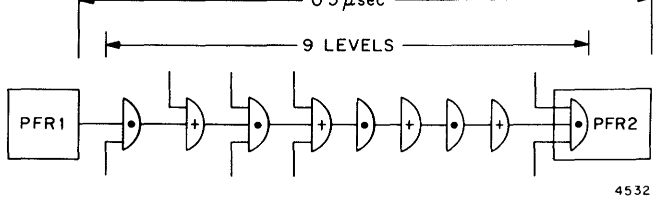


Fig. 4—Logical chain.

In addition to the basic logical circuit, the pulse-former, and the delay element already mentioned, there are several other high-speed circuits. One is a high-power amplifier capable of producing about 10 times as much current output as the basic circuit which can therefore drive 32 basic circuits. However, this circuit pays for its extra output by consuming 3 levels of delay. Another useful circuit is a scaled-down basic circuit to operate at ⅓ the power level. Thus, one of the basic circuits can drive 9 of the lower-level circuits. The lower-level circuit, however, consumes 1.6 delay levels. The last useful high-speed element is the high-power amplifier coupled with the pulse former to form a high-power pulse-former.

The fact that LARC circuitry is clocked allows the use of what is called the "pulse-envelope system," or more familarly, "non-return-to-zero" (NRZ) logic. If direct-coupled circuitry is used in asynchronous circuits with NRZ logic, two difficulties appear. Either the length of delay through all logical paths must be predicted and be reasonably constant, or a signal from a circuit must be derived to tell when the logic in a chain has been completed. If the former method is used in a parallel computer, there is no advantage over the clocked system.

In the arithmetic circuits, for example, many logical paths operate in parallel. Therefore, if a delay element is to signal when the last of the data has come through, it must have a delay equal to the clock-pulse spacing in a synchronous machine. On the other hand, if the signal itself is used to tell when an operation is completed, there is the problem of discriminating between a normal signal and the "spikes" (switching transients). This is because a gate with one input being turned on at the same time another input is being turned off will, for a finite signal rise-time, allow a spike to come through, as shown in Figs. 5a and 5b.

Now if a return-to-zero or regular pulse system is used there is always a dead space between signals. All signals return to zero during this dead interval and avoid the spike problem. However, the speed at

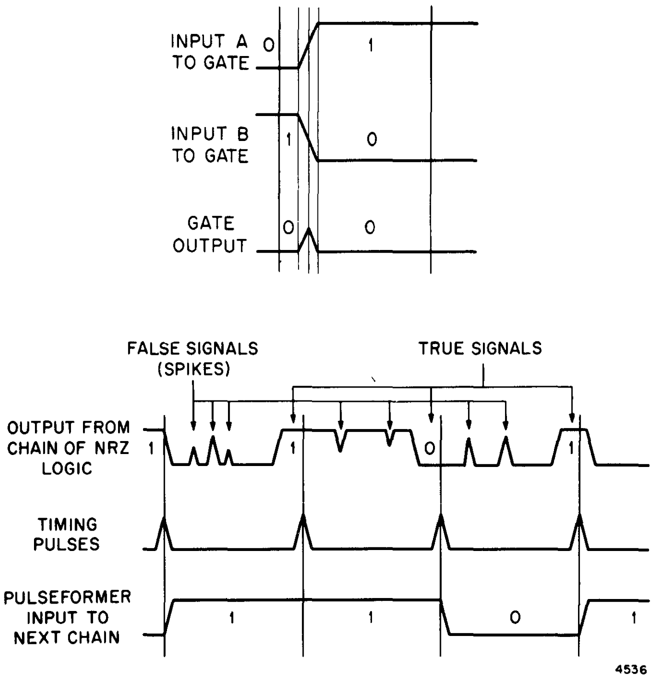


Fig. 5(a)—"Spike" formed due to overlap of gating signals. 5(b)—Discrimination against "spikes" occurring in a NRZ system using clocking methods.

which pulses can now be put through the circuit (for equal rise and fall time) will be halved because of twice the number of transistions. Since LARC transistor circuit delays are quite stable, less than a two-to-one timing margin is allowed in the clocked circuits. Thus the necessity of using return-to-zero signals would decrease the data rate by more than asynchronous operation would increase it.

Asynchronous operation not only requires more equipment and has no apparent speed advantage, but makes testing and maintenance more difficult, since no definite points to trigger an oscilloscope can be found: timing depends on actual circuit parameters and the actual signals present.

A large number of logical configurations involving adders, complementers, and other devices were examined. Methods of multiplication and division were chosen carefully, not just on the basis of speed, but the circuits were examined to find out how many drives an individual element need provide so that an overall minimization of the speed through a chain of these elements might be made. Since the product of current gain times the band width for a given transistor is normally constant, it is possible to estimate the delay through a network, given the number of drives required at each level.

With a simple fan-out network, this optimum occurs with three drives; however, there is no assurance that this is the optimum for an adder network. Although it was suspected that the same limitations were probably true in actual circuits, many circuits were carefully laid out and it was shown that not only were three drives adequate, but such an arrangement provided the optimum speed. Further, the current gain of the surface-barrier transistors is low and made more drives undesirable. With high-speed mesa transistors now becoming available, LARC speeds could have been achieved with circuits that allowed more drives. Thus fewer transistors would be required in the circuit. Studies show that perhaps 15 per cent to 20 per cent of the transistors might be eliminated in this way. Nevertheless, if the objective is to obtain optimum speed out of the newer mesa transistors, indirect-coupled circuitry, use of the present design based on three drives would still be applicable.

Since many, many thousands of high-speed circuits are employed in the machine, it is obvious that the packaging is an extremely important consideration; otherwise wire lengths become too great and exceed the 80 $\mu\mu$f circuit allowance. Also, crosstalk problems could become severe with longer lead lengths. A study and optimization program showed that it was necessary to compromise at a maximum of ten circuits per printed-circuit card. This was a compromise between achieving efficient packing densities and minimizing the number of different card types. Larger numbers of circuits per card provide higher packaging efficiencies because DC voltages and clock lines utilize a smaller total number of backboard contacts. A successful solution to the basic and conflicting problems of circuit speed and capacitance and crosstalk minimization was largely achieved by the development of a special connector, which provided a large number of connections with the smallest possible backboard area. Fig. 6 is a photograph of a typical LARC printed-circuit card. The connector which is affixed to the end of the card has 42 through connections in it, in addition to the guide pins which also carry through the ground connection. Connectors are held to the printed circuit board by means of a metal framework around the card, which also acts to eliminate any problems with card misalignment or warpage. The back half of the card nearest the connector contains five circuits. The other half contains five additional circuits. Test terminals are electrically connected to the outputs of each one of the ten circuits. The seven basic circuits are used with different configurations of input diodes. This results in a total of 30 different types of high-speed cards. Spare diodes and transistors are removed where not needed, for economic reasons. Spare cards used for maintenance

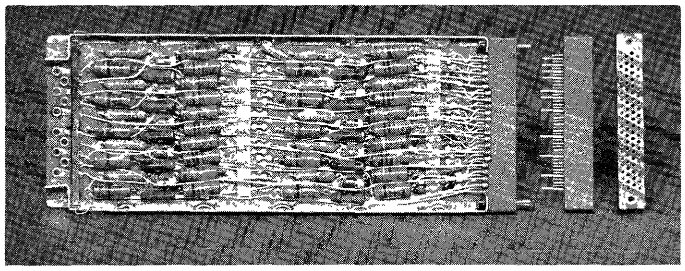


Fig. 6.

purposes have all diodes and transistors in place. Card dimensions are approximately 3½″ by 9″. This shape was chosen to obtain the necessary volume for housing the components. Female contacts are used on the card connector so that there is no possibility of damage which might occur if male contacts had been used. The wires observed on the package serve the function of providing connection between the floating female contacts and the printed circuit wiring. Lower stray capacitances are also achieved and input-output printed-circuit wire bottlenecking is reduced. The male connectors are packaged extremely close in the basic modules which go together to form the backboard. Fig. 7 is a photograph of a module backboard before wiring. The degree of packing efficiency is very high when it is realized that 88 per cent of the backboard wiring area is composed of connectors. As a consequence of this, and the close packing of contacts, there are over 6000 wire terminations per square foot. Reducing the backboard area in this way has allowed the use of wires sufficiently short so that no special line-driving elements are necessary. It was necessary to leave small spaces (30 mils) between connectors to permit a small amount of air flow for cooling purposes. All wiring on the backboard is done with taper pins. With this high a wiring density, as shown in Fig. 8, it becomes impractical to consider soldered or wire-wrapped connections. The connectors are color coded to make the job of terminal identification much easier for the wiremen. Each terminal on the connector is bifurcated and appears as two taper-pin holes so that it is possible to propagate the chains of wires without using auxiliary tie-points. The wiring is extremely dense and piles up to a depth of several inches over most of the backboard. Throughout the development phases of the program, accessibility to the backboard was a matter of gravest concern, since it was impossible to accurately predict or simulate actual backboard wiring buildup. We have since installed many thousands of wiring changes and have fully proven that the wiring technique is indeed practical. We have developed new technology and new tools for working with this new high wiring density. Fig. 9 is a photograph of special tools, which

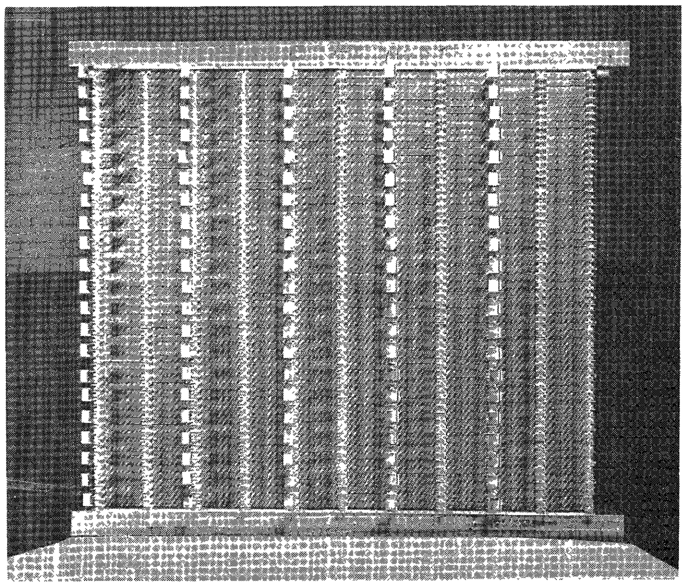


Fig. 7.

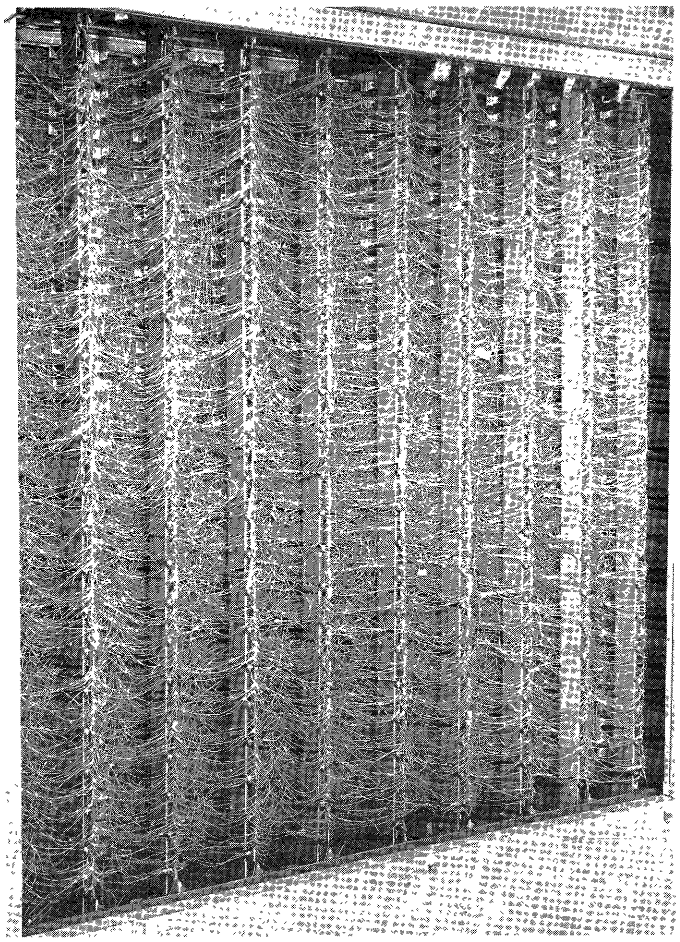


Fig. 8.

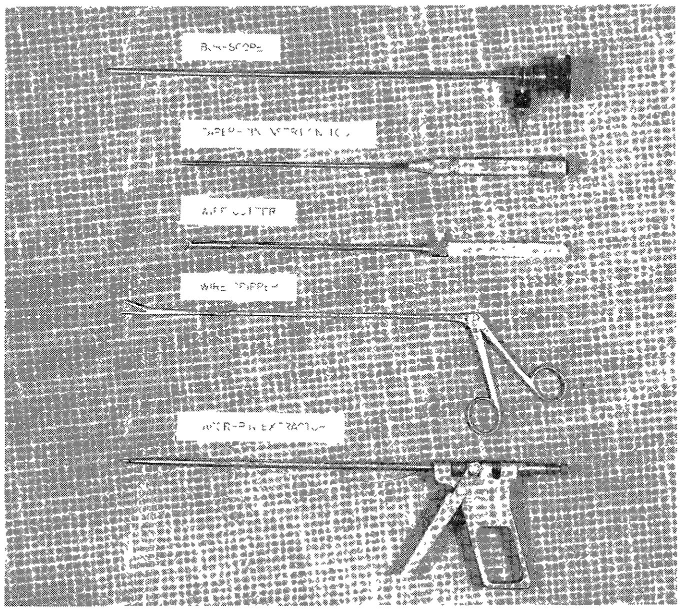


Fig. 9.

include long armed taper pin inserters and extractors, pin point light sources, and a "Borescope," originally developed for examining the inside of a gun barrel, but proven effective for penetrating the mass of backboard wire and giving the wiremen a close-up view of the connector.

In order to reduce the congestion on the backboard, very fine steel-core copper wire was used with a thin teflon insulation. The physical strength of size 30 steel-core wire is approximately equal to copper wire of twice the cross sectional area. Reduction in wire size not only helped to reduce congestion but also produced a necessary reduction in the stray wiring capacitance.

One of the more serious problems that could occur with many thousands of transistor circuits operating simultaneously is that of crosstalk and other noises being coupled into the transistor circuits. Crosstalk effects were calculated as much as was possible and further experimentally checked in the laboratory. To prevent crosstalk due to coupling between backboard wires, twisted pair is used for any wire lengths greater than about nine inches. Although tests indicate that twice this length could be used under worst tolerance conditions, approximately 23 per cent of the backboard wires are twisted pair. The twisted pair is composed of very fine steel-core teflon-insulated wires with a capacitance of 8 to 9 $\mu\mu$f per foot. A network of taper-pin ground straps is provided at each ½-inch interval over the entire backboard to form a ground plane. The twisted-pair ground wires terminate at these ground straps. All framework elements of the module carrying ground currents are gold-plated so that the contact resistance between abutting structural members is low and stable over a long period of time, despite atmospheric conditions. Common coupling on the DC voltage lines is minimized by the use of very wide strip transmission lines having extremely low impedances. Strip transmission lines are mounted in vertical columns behind the backboard. These lines have an impedance in the region of 30 milliohms (.03 ohms). Timing-pulse signals to pulse-formers are also available on similar strip transmission lines.

Usually, three DC voltages are distributed to the printed-circuit packages through the strip transmission lines. Up to 6 voltages are available for other miscellaneous card types.

The task of laying out the backboard wiring and positioning the various circuits in such a way as to avoid too much wiring load on any one circuit, as well as the problem of keeping an inventory of all circuit cards, would have been impractical had this not been accomplished by processing the data on a Univac® data-processing system. Thrity-five different categories of information, as well as complete production-wiring tables, were generated by the Univac system to supply necessary information for production, maintenance, manufacturing inventory, and engineering test of LARC. For example, printouts were obtained on wires sorted by lengths, potentially bad cases of stray capacity or crosstalk, spare diode and circuit positions, checks on certain types of logical errors, and general data vital for testing and maintenance. All logical revisions which were made during the test period were handled in this automated and systematic manner.

The automated backboard program guaranteed that wiring changes could be made without fear of overlooking any of the myriad of details involved in the change. Fig. 10 shows one page of a printout of the backboard wiring table.

Solid-state power supplies are used throughout the system. Each cabinet has its own set of power supplies and controls, so that lead lengths between card library and power supply can be kept to a minimum. The lead impedances between power supplies and the card library are kept low by the use of bus bars with electrolytic filter-capacitors distributed along the length of the bars. The supplies are all voltage-regulated, either by shunt transistor regulators or, in the case of the very high current supplies, transistor-driven magnetic amplifiers. The size, cost and time of response of the power supplies have been reduced by the use of 400-cycle 3-phase input power derived from a motor-alternator set. The motor-alternator also has the advantage of providing complete line isolation and will produce full putput even though power line "dropouts" occur for as long as 3 seconds. The power supply design has proven extremely reliable, with voltage regulation much better than the specified 2 per cent.

A major virtue of the circuitry, not found in many high-speed computer circuits, is its adaptability to a simple and effective marginal-checking system. Varying the collector-return voltage has proven to be an effective way of determining the beta margins of the circuit. Fig. 11 is an actual plot, showing how the

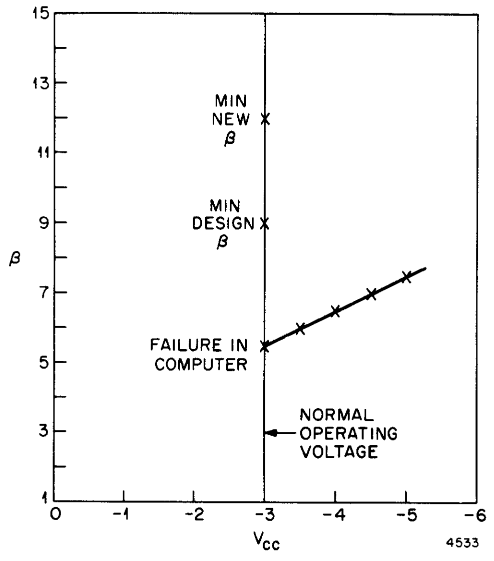
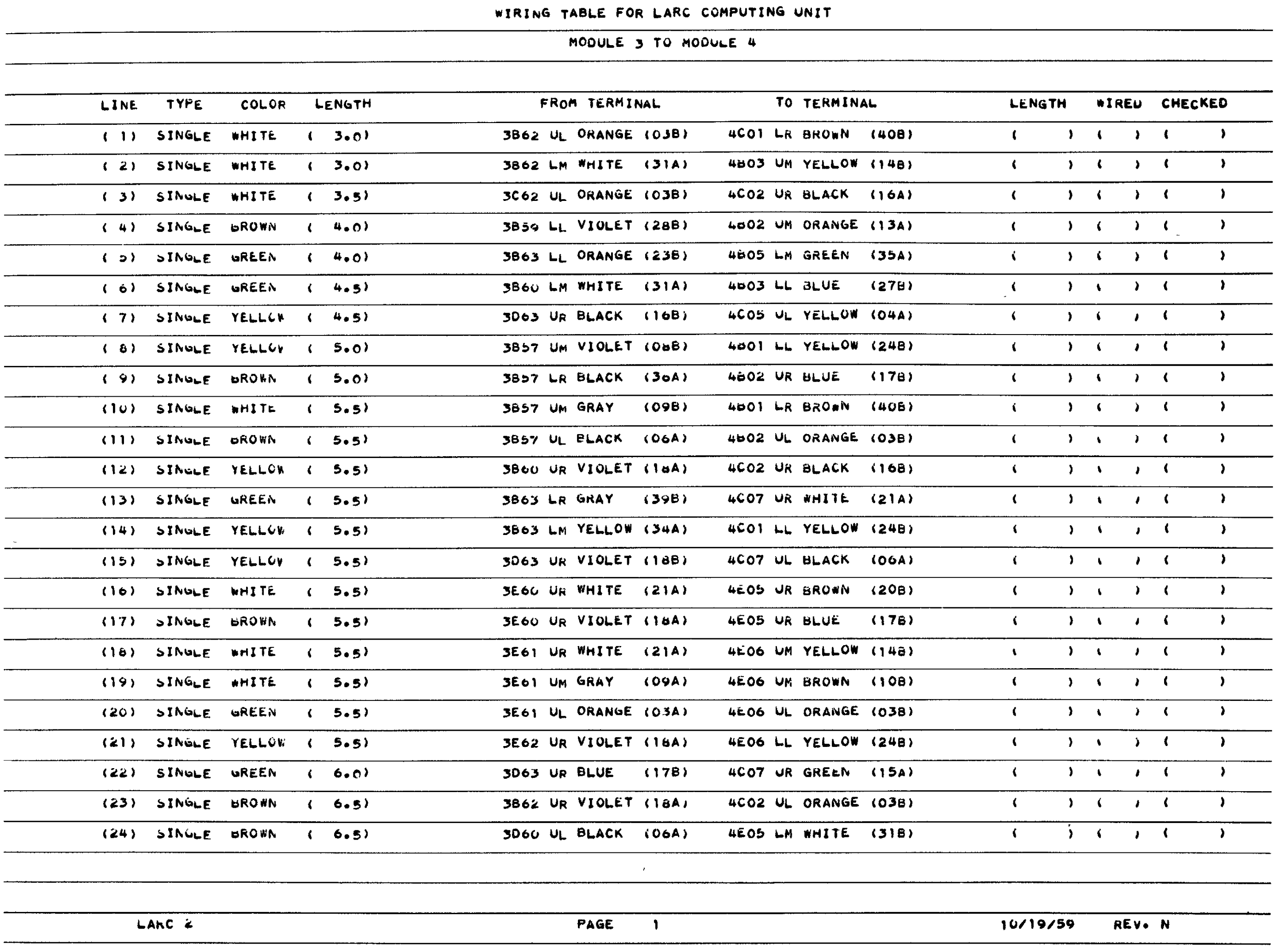


Fig. 11—Circuit failure as a function of V_{cc} and β.



WIRING TABLE FOR LARC COMPUTING UNIT

MODULE 3 TO MODULE 4

LINE	TYPE	COLOR	LENGTH	FROM TERMINAL	TO TERMINAL	LENGTH	WIRED	CHECKED
(1)	SINGLE	WHITE	(3.0)	3B62 UL ORANGE (03B)	4C01 LR BROWN (40B)	( )	( )	( )
(2)	SINGLE	WHITE	(3.0)	3B62 LM WHITE (31A)	4B03 UM YELLOW (14B)	( )	( )	( )
(3)	SINGLE	WHITE	(3.5)	3C62 UL ORANGE (03B)	4C02 UR BLACK (16A)	( )	( )	( )
(4)	SINGLE	BROWN	(4.0)	3B59 LL VIOLET (28B)	4B02 UM ORANGE (13A)	( )	( )	( )
(5)	SINGLE	GREEN	(4.0)	3B63 LL ORANGE (23B)	4B05 LM GREEN (35A)	( )	( )	( )
(6)	SINGLE	GREEN	(4.5)	3B60 LM WHITE (31A)	4B03 LL BLUE (27B)	( )	( )	( )
(7)	SINGLE	YELLOW	(4.5)	3D63 UR BLACK (16B)	4C05 UL YELLOW (04A)	( )	( )	( )
(8)	SINGLE	YELLOW	(5.0)	3B57 UM VIOLET (08B)	4B01 LL YELLOW (24B)	( )	( )	( )
(9)	SINGLE	BROWN	(5.0)	3B57 LR BLACK (36A)	4B02 UR BLUE (17B)	( )	( )	( )
(10)	SINGLE	WHITE	(5.5)	3B57 UM GRAY (09B)	4B01 LR BROWN (40B)	( )	( )	( )
(11)	SINGLE	BROWN	(5.5)	3B57 UL BLACK (06A)	4B02 UL ORANGE (03B)	( )	( )	( )
(12)	SINGLE	YELLOW	(5.5)	3B60 UR VIOLET (18A)	4C02 UR BLACK (16B)	( )	( )	( )
(13)	SINGLE	GREEN	(5.5)	3B63 LR GRAY (39B)	4C07 UR WHITE (21A)	( )	( )	( )
(14)	SINGLE	YELLOW	(5.5)	3B63 LM YELLOW (34A)	4C01 LL YELLOW (24B)	( )	( )	( )
(15)	SINGLE	YELLOW	(5.5)	3D63 UR VIOLET (18B)	4C07 UL BLACK (06A)	( )	( )	( )
(16)	SINGLE	WHITE	(5.5)	3E60 UR WHITE (21A)	4E05 UR BROWN (20B)	( )	( )	( )
(17)	SINGLE	BROWN	(5.5)	3E60 UR VIOLET (18A)	4E05 UR BLUE (17B)	( )	( )	( )
(18)	SINGLE	WHITE	(5.5)	3E61 UR WHITE (21A)	4E06 UM YELLOW (14B)	( )	( )	( )
(19)	SINGLE	WHITE	(5.5)	3E61 UM GRAY (09A)	4E06 UM BROWN (10B)	( )	( )	( )
(20)	SINGLE	GREEN	(5.5)	3E61 UL ORANGE (03A)	4E06 UL ORANGE (03B)	( )	( )	( )
(21)	SINGLE	YELLOW	(5.5)	3E62 UR VIOLET (18A)	4E06 LL YELLOW (24B)	( )	( )	( )
(22)	SINGLE	GREEN	(6.0)	3D63 UR BLUE (17B)	4C07 UR GREEN (15A)	( )	( )	( )
(23)	SINGLE	BROWN	(6.5)	3B62 UR VIOLET (18A)	4C02 UL ORANGE (03B)	( )	( )	( )
(24)	SINGLE	BROWN	(6.5)	3D60 UL BLACK (06A)	4E05 LM WHITE (31B)	( )	( )	( )

LARC 2 PAGE 1 10/19/59 REV. N

Fig. 10.

collector-return voltage may be varied to determine the beta margin, which is perhaps one of the most variable and critical of the transistor parameters. The extremely difficult problem of switching individual portions of the very low-impedance voltage-distribution system has been avoided by varying the voltage over the whole unit rather than in a given area. The location of a weak circuit is indicated logically, by error detecting circuits which are located at strategic points throughout the logic of the machine. Thus almost no equipment not already present for continuous checking is necessary to provide very effective marginal checking. An overall marginal check can be performed simply by flipping a switch at the engineer's console while an engineering-test routine is being run on the unit. A comprehensive voltage-monitoring system is used to detect the fact that a particular power-supply voltage is drifting out of tolerance. This fact causes visual and audible indications before a voltage drifts far enough out of tolerance to cause actual errors. The voltage monitor makes use of solid-state elements.

The Computing Unit and Processor are each supplied with a system of fast magnetic registers which are composed of fast-switching tape-wound cores having a read-regenerate or clear-write cycle of 1 microsecond. The fast-register core consists of 4 wraps of $\frac{1}{8}$ mil thick, $\frac{1}{32}$ wide, 4-79 molybdenum permalloy tape on a 50-mil diameter stainless-steel bobbin and has a read, a write, and an output winding. The register uses one core and diode per bit and is organized on a word-selected basis. As employed in the Computing Unit, information write-in or read-out is done in 60-bit parallel form. Fig. 12 is a photograph of a fast-register package. The cores and diodes are contained in the small rectangular boxes and are

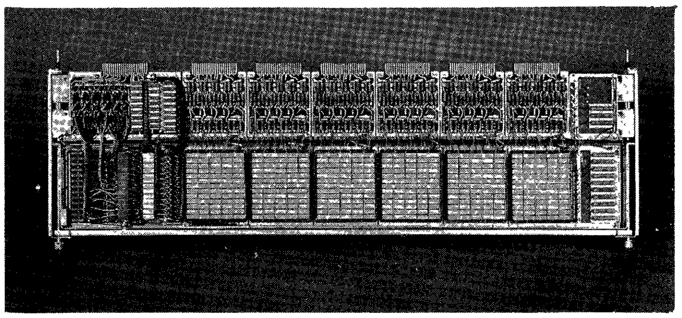


Fig. 12.

arranged in a 26 × 30 array. Two of these packages are used to form 26 registers of 60-bit storage. More of these packages may be plugged into the Computing Unit to make available a maximum of 99 registers that can be used interchangeably for indexing or arithmetic operations.

To carry out the logical operations in parallel fashion, large quantitites of circuits are required. To present some idea of magnitude, a few of the statistics will be listed: 8800 printed circuit cards are used in the basic system, with approximately 3700 in the Computing Unit, 3200 in the Processor and 1800 in the memory. This represents a total of 62,000 transistors in the entire typical system, with approximately 57,000 of them being surface-barrier transistors. 28,000 transistors are used in the Computing Unit alone. The number of logical diodes in the system is approximately 2.8 times the number of transistors. 75,000 wires are used on the backboard of the Computing Unit.

With these large quantities of components used in the system, it is quite obvious that reliability is a major problem and, therefore, a big part of the LARC development program was concerned with component reliability. Many of the basic component selections were made on the basis of reliability rather than cost. All components used in the system were subjected to extensive reliability tests. The resistors used in the circuitry, for example, are of the ½ watt size, even though circuit dissipation is in the milliwatt region. The ½ watt resistor available at the start of the project proved to be much superior to the smaller wattage sizes. A large engineering effort was required to achieve the high reliability obtained in the LARC printed-circuit card connector. The contact pressures are accurately designed to be in the 4- to 6-ounce range per contact. The use of electropolishing and 200 micro-inches of gold permits the connector to be withdrawn and inserted 400 times while still retaining gold both on the male and female contact area. This is an important point, in view of the low voltage and current levels encountered. All circuits are designed to work under worst-tolerance conditions assuming 4 per cent variation in DC supply voltages, a 3-per cent variation in the value of the resistors, and the end-of-life transistor beta. Three units of beta over the end-of-life beta required are specified for new transistors. For example, for a transistor driving 3 loads, an old-age beta of 9 is required; a transistor with a beta of 12 minimum is initially inserted in the circuit. Under nominal tolerance conditions, the circuit will function even if the beta drops to 6. The most pessimistic calculations of reliability have predicted a mean-error-free time of at least 11 hours, which is 3 hours beyond that called for by the specifications. Typical power requirements of the system are 15 KVA of 400-cycle power for the Computing Unit, 56 KVA for each memory unit, and 18 KVA for the Processor. The power factor is .5 because of the type of regulated power supplies employed. Most of this power is actually dissipated within the power supplies.

Each unit of the system has its own cooling equipment built into the base of the unit. A heat exchange and blower system circulate 75°F maximum-temperature air up the backboard and through the card library; the air then returns through the front side of the machine to the base area, thus forming a closed-circuit path. The circuits will operate over a much wider temperature range, but internal cooling is used to guarantee long life and reliability. Operation is reliable over a room temperature range of 32 to 110°F.

The LARC system is being readied for the customer acceptance test. All units of the system are functioning. The Computing Unit, in conjunction with the memory, has been running test routines for many months. Although the overall long-term reliability data on the system is not yet available, preliminary information is very satisfactory. Runs of about 12 hours have yielded one or no intermittent errors. Fortunately, the very few failures encountered thus far have been of the catastrophic variety, such as open or shorted components (probably due to abuse during testing) rather than a change in parameter value which leads to marginal operation and accompanying difficulty in isolation. It is extremely gratifying that no circuit redesign of any sort has been found necessary since the commencement of test. This has led to an unusually rapid testing of the overall system. The efforts put into very thorough engineering beforehand have paid off handsomely.

Many people were involved in making the LARC program a success, and it would be impossible to list them all by name. Recognition must be granted to all of them for their part in this major engineering achievement.

Discussion

S. Levine (*Teleregister*): What is the expected LARC reliability: *i.e.*, mean free time to failure, fault location time, mean repair time of main processor and sub-systems? What size of maintenance crew do you expect to require for a typical installation operating 24 hours a day, seven days a week?

Mr. Lukoff: Well, we have calculated the mean free error time using reliability figures provided to us by our component engineering groups and by Bell Telephone, RCA, and so on. The mean free error time is predicted as at least 11 hours which is three hours beyond that called for in the LARC contract specifications, which ask for an eight-hour mean free error time.

Mr. Eckert: The few runs we have made so far are better than this We have made a few runs of 12 hours during the course of system testing, and it has actually been better than this. We haven't run enough to know the ultimate mean free time to failure.

Mr. Lukoff: The repair time on a card would be, say, two minutes to replace a diode or transistor. We have automatic package-checking equipment that checks the package under dynamic operating conditions. It measures the transit time to within 3 millimicroseconds and pretty well down to the component that has to be replaced. This can be done very rapidly. We have a supply of spare parts the maintenance man is furnished with. He would, of course, replace the defective package in the machine with a good spare package and make the repair on the bench.

We are presently in the process of developing and refining the LARC maintenance procedures. However, we believe that maintenance will be easier than with any other machine we have built thus far because of the packaging techniques, and because of the large amount of checking which permits logical isolation of the circuit.

On the maintenance crew I would expect it to be of the same order of magnitude as it is for any other computer in the field.

J. Capobianco (Hughes): What type of logic diode was used? What are its transient specifications?

Mr. Lukoff: The diode is a point-contact diode selected for its high speed capabilities. Also, the forward drop was an important consideration in the direct coupled circuitry. I don't remember the exact specification for speed but it is a fast diode.

Mr. Eckert: It is about two or three times faster than the usual run-of-the-mill gold bonded diode. Actually we have two types of diodes, gold bonded and plated tungsten. We have four or five different suppliers who could make such a diode. It is a better diode than the one you go out and buy over the counter. It is one manufactured for well under a dollar.

V. Enstein (Brooks Research): What method of switching is used to switch between the various drums? What significance is attached to the two different chairs for the operator and the engineer?

Mr. Lukoff: Purely esthetic, in answer to the second question. In answer to the first, we use relay switching cabinets.

Mr. Eckert: We have plenty of time to switch because you are really stripping one drum while you are reeling off the other.

Mrs. J. Schot (D. Taylor Model Basin): What is the maximum reliable pulse density of the LARC tape units? In other words, is it possible to store two drum loads on one reel of tape?

Mr. Eckert: We are using standard Univac, which packs 250 to the inch. We have experimental tape which goes much higher. As I said, we will put either these tape units or competitive tape units on the machine to suit the customer.

G. Neuman (Magnavox): With the large number of transistors used, what percentage of rejects did you have in construction?

Mr. Eckert: About 10 percent, wasn't it?

Mr. Lukoff: We knew we would have about 6 percent rejection on transistors because we bought the complete line of transistors from the manufacturer, and we knew some would not meet our specifications.

Mr. Eckert: We buy transistors to normal specifications and buy everything on the line and then throw out about 6 percent due to low current. Again this is no problem.

D. Hammel (RCA): What are the possibilities of a third computer operating in the LARC system?

Mr. Eckert: That is easy. It is not designed for it. It doesn't have enough time slots.

P. J. Scola (GE): Do you use diagnostic programs? What percentage of the faults do they find? Average time to locate fault?

Mr. Lukoff: We can only partly answer by saying we definitely plan to use diagnostic programs but have not had the opportunity to fully explore their potential in the computer yet.

Mr. Eckert: I suspect you could ask Mr. Tonik after the conference session.

M. Sendrow (RCA): Is there any way of loading the drums from any external media? If so, what media and how?

Mr. Eckert: Yes, they can be loaded from tape units, by punch cards, card readers, through the core memory. You could, if you wanted, from keyboards, but it would take all your life.

J. Katz (GE): How many wiring mistakes in the backboard were traceable to malfunctions in the data processing program?

Mr. Lukoff: None.

Mr. Eckert: Most of the wiring errors we found in the backboard were either production errors or due to logical design errors. There were lots of errors, but there have been on every machine I have seen.

W. C. Mann (Westinghouse): About what percentage of the backboard connections are used?

Mr. Lukoff: It is very high. I don't remember the exact figure, but it is in the order of 75 percent.

Mr. Eckert: As far as correcting errors, these logical errors, that is the reason for the long handled tongs.

G. A. Sellers (Bell Labs.): Please repeat the characteristics of the high-speed microfilm printer. Is it going to be commercially available?

Mr. Eckert: Yes, it is commercially available. It prints at 20,000 characters a second. I think you can produce it on a polaroid, too, if you want a single frame and print points at this speed. It could print at a somewhat higher speed, but we didn't choose to. So it makes a good plotter.

F. F. Jenny (IBM): How do you differentiate between errors in the error detection system and the operational system?

Mr. Lukoff: The error detection circuits are not distinguished from the computer circuits proper. Nobody watches the watcher. A detected solid error is to be repaired regardless of source. We ignore the possibility of a double failure over a short time period and employ error generating routines periodically to check the operation of the check circuitry. In some cases error insert switches have been provided where input/output equipment is involved.

Mr. Eckert: The machine doesn't necessarily stop when it makes an error. It is rigged up so it can go into a routine and try something over with different conditions. After a certain number of times, it finally stops. When you have an intermittent error the frequency may become high enough that you want to service it.

T. Digan (IBM): How do you marginal check the 170,000 diodes?

Mr. Lukoff: The diodes are marginally checked along with the rest of the circuits. There is just one marginal check test of circuits, not able to find if the transistors are low or voltages are out of tolerance or high drop in the diode or the diodes become slow.

Mr. Eckert: You see, we are satisfied to find out which group of circuits the fault is located in. Then we find out within three diodes or two resistors which is bad. It is an overall clump test, so to speak.

Mr. Thomas (MH): What automatic programming systems are available or planned for LARC? Do they include special features to aid in the effective simultaneous use of the several computing units?

Mr. Lukoff: At the present time we are planning a complete assembly system and a compiler system for LARC to use the argol language. There will probably be more automatic program work done in the future although we can't say much about this at the moment.

T. Gilmer (ITT Federal): You have indicated an unusually short time in test for LARC. Can you give an indication of the elapsed time and the number of crews working?

Mr. Lukoff: The processor unit, which was the last one to go into test, has been in test for approximately five or six months, and it is just about completed. I think this is very, very short for the number of circuits involved and for the complexity of the machine.

Mr. Eckert: This is done on three shifts, and, of course, some shifts are missed now and then for holidays and other reasons.

M. Relis (Curtiss-Wright): What transistor is used in the gate-inverter?

Mr. Eckert: Philco surface barrier transistor, although it differs in that a slightly lower resistivity and a slightly different size of germanium is used. Other than that it is a regular SBT transistor.

G. E. Saltus (BTL): What maximum logical fan-out and fan-in are allowed in the basic circuit?

Mr. Lukoff: A maximum of 13 fan-in for basic circuits and a fan-out of 3 but more places are sometimes allowed because of load-sharing.

Mr. Eckert: You hook up to 10 in some cases, but it only drives 3 due to mutual-exclusion. There are some places where you could get more load by taking the dummy load transistor off. We do that only in the memory.

P. W. Core (IBM): Can both arithmetic computers perform operations in the times quoted simultaneously?

Mr. Eckert: Yes.

R. Adams (DATAmatic): Are there any further requirements for other than twisted pair for those over 9 inches? Specifically I am thinking of termination requirements or coaxial runs.

Mr. Lukoff: There are no terminations required within the main units, but there is still a maximum length; that is, we must not exceed the micro-microfared capacity units. You can't allow longer wirelength for a circuit to drive or else it will take more than its allotted delay level.

Mr. Eckert: Actually we computed — we had to compute — everything twisted above 15. We actually twisted everything over 9 inches, and it amounted to 30 percent of the wire.

M. Lavel: Are there any tubes used in the LARC?

Mr. Eckert: Yes, to get the fairly sizable clock powers. At the time we designed there weren't suitable transistors, but they probably exist now.

J. E. Veal (RCA): What is the figure of merit of the diode-core registers as compared to your figure of merit for other circuits in the memories?

Mr. Eckert: We didn't design the core register by a figure of merit. We considered several types, and this was the only type we knew how to build. We knew of others, but the development time was long, so they were thrown out.

Miss H. Bein (Philco): What do you mean by a LARC system with 2 computers? Would they be handling different programs and sharing one memory and data processor?

Mr. Eckert: They would share one memory bank, but don't forget a memory bank can have up to 39 functionally independent 2500-word units in it.

Miss R. Pitche (Northeastern Univ. student): Do you use a parity bit? If so, how?

Mr. Eckert: Well, there are extensive uses of parity bits. They are on the tape unit. The code is carefully chosen to cut down the errors. There are parities all over the place.

E. Morenoff (RADC): What factors limit the number of computing units to 2?

Mr. Eckert: The number of time slots and the needs we saw.

E. L. Lawler: Was the program for backboard wiring primarily for record keeping or was some attempt made to optimize the location of the packages and the interconnections between them?

Mr. Eckert: For both of those and also the calculation of loads and various other things. In the inventory class alone there were 35 different types of lists needed.

Arithmetic and Control Techniques in a Multiprogram Computer

N. LOURIE†, H. SCHRIMPF†, R. REACH†, AND W. KAHN†

IN THE DESIGN of a data processor for commercial applications, the designer is very often striving for better machine performance for little or no increase in cost. In the system design of the Honeywell 800 transistorized data processing system, several design concepts were utilized to help achieve this objective. One of these techniques involves the use of a small auxiliary memory to aid in the control of the high speed central processor. A second technique uses a new word organization that results in a faster and less costly arithmetic element.

In a modern transistorized computer, the speeds that are economically achievable in the central processor are very often much higher than necessary to keep up with peripheral devices. The concept of time sharing the central processor among several programs in order to utilize otherwise wasted time then becomes attractive. In order to achieve this time-sharing automatically without the use of cumbersome supervisory routines, at least one sequence counter per program is required. If a small coincident current memory running out of phase with the main memory were available, a relatively liberal number of programs could easily be run simultaneously by assigning these sequence counters to this control memory. Also, since additional memory locations become economical, it is now simple to assign each program two sequence counters for greater flexibility. These are known as the sequence and consequence counters. The Honeywell 800 has 8 pairs of sequence counters, thus allowing the simultaneous operation of eight independent programs.

TABLE 1

USE OF CONTROL MEMORY FOR SIMULTANEOUS PROGRAM ORATION

Program	Sequence Counter	Location	Cosequence Counter	Location
1	00050	2	00600	3
2	02090	34	03002	35
3	°	66	°	67
4	°	98	°	99
5	°	130	°	131
6	°	162	°	163
7	°	194	°	195
8	°	226	°	227

To illustrate how this is performed, Table 1 shows a possible state of the various counters. If, upon starting, the first order is specified from the sequence counter, 00050 will be read from location 2 of the control memory, and the contents of 00050 in the main memory will be read and performed as an order. The sequence counter will then be incremented by unity so that 00051 will be immediately reinserted into address 2 of the control memory as the location of the next order to be performed under control of the sequence counter in program 1. If the previous order in program 2 specified that the cosequence counter was to be used to obtain the next order, the contents of address 35 will then be read out of the control memory and 03002 will then be used as a main memory address to select the next order performed. Similarly the computer will then cyclicly perform one order from each program. Some orders that leave useful information in the central processor do not relinquish control to another program, so that occasionally, several orders from one program will occur before any orders from another program are performed. The multiply order is an example of an order that requires such treatment since there is still a low order product that may be required after the completion of the order. Because the control memory is running simultaneously but out of phase with the main memory, this multiple operation not only is extremely flexible, but is performed without loss of speed.

Each of the sequence and cosequence counters in the Honeywell 800 has associated with it in the control memory another register known as a history register. Whenever a sequence or a cosequence counter is modified because of a sequence change, the associated history register is changed so that it contains the address that the sequence or cosequence counter would have contained if there were no sequence change. With this feature available, the programmer can easily sequence change into a subroutine and then, at some later time, revert back to the main routine. Table 2 gives a numerical example of this use of a history register.

The same control memory can be extremely useful for control of information to and from peripheral devices and the main memory. Eight input registers and eight output registers have been reserved in the control memory for controlling the transfer of data between peripheral devices and main memory. Each of these control memory registers is uniquely associated with an input or output trunk. When an input trunk signals that it has a word available, the central

† DATAmatic Division, Minneapolis-Honeywell Regulator Company, Newton, Mass.

TABLE 2
USE OF SEQUENCE HISTORY REGISTER TO RELOCATE AFTER SUBROUTINE

	Sequence Counter	Sequence History Register
	00122	00000
	00123	00000
	01200	00124
Subroutine	01201	00124
located	°	°
from 01200	°	°
to 01280	°	°
	01280	00124
	00124	01281

processor is interrupted at the end of the next memory cycle. The buffer control register in the control memory associated with this input trunk is read, and the contents used to select a main memory address into which the word from the input trunk can be inserted. The content of the buffer control register is then incremented by unity and immediately placed back into the same control memory location. Thus, the next word from the same input trunk will be inserted into the next highest memory location. In the case of a reverse tape read order, the content of the control memory register is decremented by unity prior to insertion back into the control memory so that the information from tape will be in correct order regardless of the direction of tape motion. Similarly, words are delivered from the main memory to a peripheral device in the case of a write order.

If more than one trunk is on demand at the same time, a simple buffer traffic control system establishes priority, and the trunks are processed one at a time. After all input and output trunks are processed, the machine control then reverts back to normal order processing.

To increase the average data rate from tape and improve the utilization of tape space, it is desirable to place more than one item on a block of tape. When this is done, it would be desirable to be able to place each individual item in a different section of the memory. In order to accomplish this "distributed reading or writing", a set of address locations that will serve as the starting locations for each of the consecutive items after the first item, is inserted into the main memory. The starting location of this group of beginning item addresses is placed into a control memory address called a distributed item counter. There is one distributed item counter for each input trunk and each output trunk. As before, when a block of information is read into the memory, the initial item is placed into main memory locations as specified by the associated buffer control register. However, when a special bit configuration representing an end of item is sensed, the content of the main memory location as specified by the distributed item counter is read into the buffer control register, thus creating a new starting address for the next item. The distributed item counter is incremented by unity prior to reinsertion into the control memory, to prepare for the next item. When this change of item location occurs, one extra memory cycle is required for all the associated housekeeping. Distributed writing is performed in a similar manner.

A numerical example of the handling of a four-item block is shown below:

Read Buffer Control Register contains	01400
Distributed Read Item Counter contains	00100
Main Memory Address 0100 contains	01500
Main Memory Address 00101 contains	01600
Main Memory Address 00102 contains	01700

With these constants located as shown, the first item would be placed in consecutive memory address locations starting with 01400, the second item starting with location 01500, the third item starting with location 01600, and the fourth item starting with 01700.

Two locations in the control memory are reserved for each of the eight programs to serve as counters for such orders as multiply and multiple transfer orders between groups of memory locations.

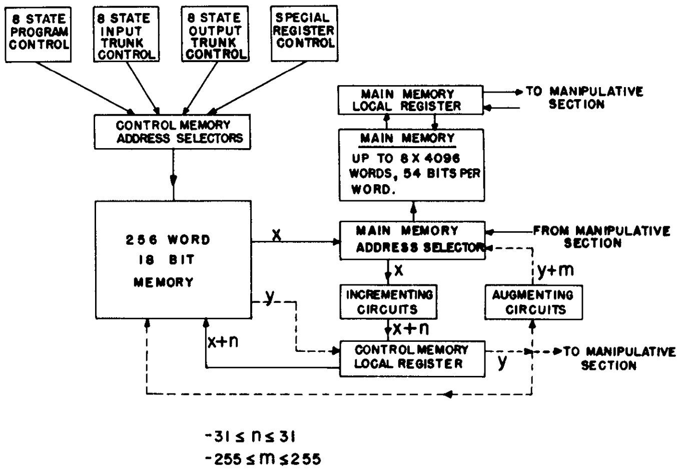

8 STATE PROGRAM CONTROL
8 STATE INPUT TRUNK CONTROL
8 STATE OUTPUT TRUNK CONTROL
SPECIAL REGISTER CONTROL
CONTROL MEMORY ADDRESS SELECTORS
MAIN MEMORY LOCAL REGISTER
TO MANIPULATIVE SECTION
MAIN MEMORY UP TO 8 X 4096 WORDS, 54 BITS PER WORD.
256 WORD 18 BIT MEMORY
x
y
MAIN MEMORY ADDRESS SELECTOR
FROM MANIPULATIVE SECTION
x
y+m
INCREMENTING CIRCUITS
AUGMENTING CIRCUITS
x+n
x+n
CONTROL MEMORY LOCAL REGISTER
y
TO MANIPULATIVE SECTION
-31 ≤ n ≤ 31
-255 ≤ m ≤ 255

Fig. 1—Control memory in a multiprogram computer.

The control memory also functions as an aid to indexing addresses. Referring to Fig. 1, the base address "y" is read out of one of 64 index registers, eight of which are available for each of the eight programs. An eight-bit augmenter "m" specified by the order is either added to or subtracted from this base address to form the indexed address for the main memory. The base address "y" is reinserted into the control memory unmodified. One extra memory cycle time may be required to perform an order if any of the addresses in that order are indexed.

Indirect addressing is another feature that can easily be accomplished by use of a control memory. In this mode of operation, a number "x" is read from a specified control memory location and used as an address in the main memory. The number "*x*" is incremented by a constant "*n*" prior to reinsertion in the control memory, so that next time this control memory content is used as a main memory address, a different main memory location will be addressed. Thus, with one order it is possible to operate on a whole series of main memory addresses without the necessity for order modification.

The control memory also serves to store a constant *U* that will give rise to an unprogrammed transfer of control if special situations such as end of tape, addition overflow, or read error occur. When one of these situations arises, the constant *U* is incremented by *n*, and an unprogrammed transfer of control to address $U + n$ occurs. The constant "*n*" is a function of the type of situation that calls for the unprogrammed transfer of control. Since there is an unprogrammed transfer register for each of the eight programs, eight independent *U* constants can be stored.

A mask index register is available for each program, such that any one of 64 mask constants can be called out of main memory by using one of the mask type orders and an incrementing constant.

A summary of the assignment of control memory locations is shown below:

Address	*Description*
0	AU-CU Control Counter No. 1
1	AU-CU Control Counter No. 2
2	Sequence Counter
3	Co-Sequence Counter
4	Sequence History Register
5	Co-Sequence History Register
6	Unprogrammed Transfer Register
7	Mask Index Register
8–15	Index Registers 0 through 7
16–27	General Purpose and Indirect Addressing Registers

These 28 locations are repeated 8 times so that each of the eight programs has a unique set of these registers.

In addition, there are eight each of the following registers which are associated with input-output. These registers are not uniquely associated with any program, but are available for convenient assignment to any program.

Address	*Description*
28	Read Address Counter
29	Distributed Read Address Counter
30	Write Address Counter
31	Distributed Write Address Counter

When the computer designer initially considers the specifications of the arithmetic unit of a digital computer, one of the prime considerations is the method of performing addition. A good design will be capable of meeting the speed specifications with a minimum of hardware. The format of the bits in the arithmetic unit is an important factor in fulfilling this objective. Various formats for a 48 bit word are discussed below.

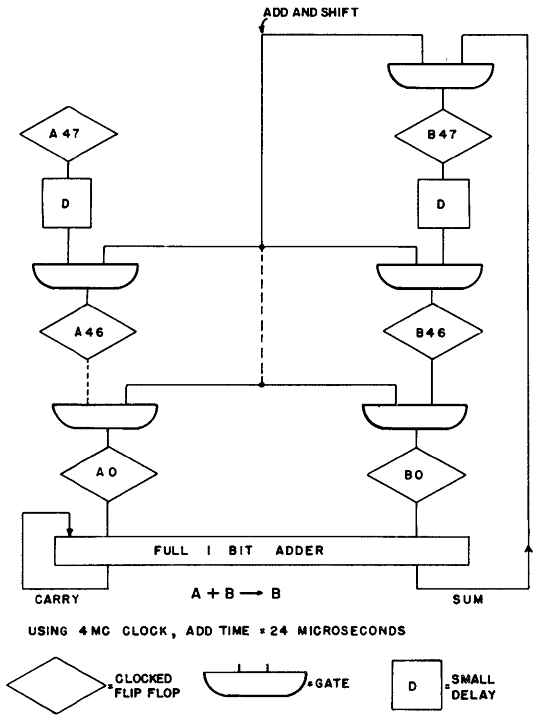


Fig. 2—48-bit serial accumulator.

Serial

A pictorial representation is shown in Fig. 2. In this arrangement, as well as all others to follow, for the sake of simplicity it is assumed that the A and B operand each reside in a 48 bit flip-flop register, each stage of which is capable of shifting. At the completion of the addition, the sum will be located in the B register. Since the addition is taking place only one bit at a time, a minimum of equipment is required. However, in order to achieve a reasonably fast add time, relatively high speed shifting flip-flops would be required. For instance, with 4 MC flip-flops, 24 microseconds would be required for a complete addition with end around carry.

Parallel-Serial

A pictorial representation of a 48-bit parallel-serial accumulator with 4 bits in parallel and 12 digits in serial is shown in Fig. 3. Other geometries could be used here, but this is a very important one, inasmuch as 4 bits in parallel can be used for a binary coded decimal digit. Since the adder is now a 4-bit adder instead of a 1-bit adder, more time will be required to propagate carries, thus resulting in a slower information shifting rate than in the serial adder. Using a 125 millimicrosecond carry propagation per stage and a 1.33 MC shifting rate, the add time will be 18 microseconds, again including end around carry propagation. The addition time is not too much faster than the example given in the serial adder, but the speed requirement of the flip-flops is reduced.

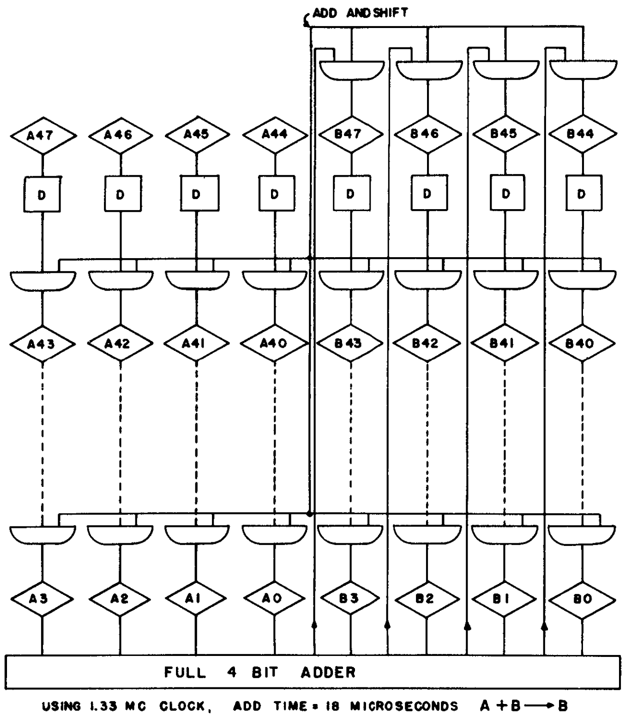


Fig. 3—48-bit parallel serial accumulator. 4 bits = 1 character in parallel, 12 characters in serial.

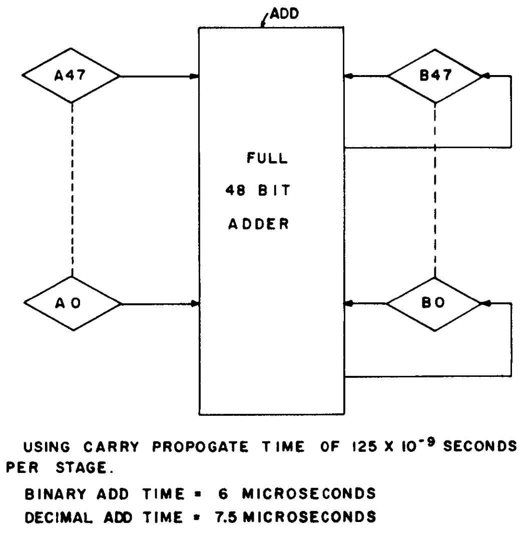

ADD
A47
B47
FULL
48 BIT
ADDER
A0
B0
USING CARRY PROPOGATE TIME OF 125 X 10⁻⁹ SECONDS PER STAGE.
BINARY ADD TIME = 6 MICROSECONDS
DECIMAL ADD TIME = 7.5 MICROSECONDS

Fig. 4—48-bit parallel accumulator.

Parallel

When the ultimate in addition speed is required, a complete parallel accumulator, as shown in Fig. 4, is often used. To achieve the fullest speed advantages, the carry propagation time should be completely asynchronous. With no "carry hopping" or "end of carry" sensing, an equivalent add time with the same circuits and assumptions above would be 6 microseconds. If decimal add were included, another 1.5 microseconds would be required. Assuming the speed-up techniques suggested previously are used, average add times on the order of 1 to 2 microseconds are feasible, but the increase in the number of logical statements is substantial. The number of logical statements required without these speed-up techniques is about 12 times as many as the parallel serial adder since the full add logic is required for each of the 48 stages.

Upon examining the requirements of the response time of any adder stage in the parallel accumulator, it is noted that although any stage is required to propagate a carry in a short time, once that stage has responded, it rests for the remainder of the carry time, resulting in a very inefficient use of the inherent speed available. If good speed-up techniques are used, then this inefficiency is greatly reduced. This observation then suggests that a parallel accumulator without speed-up techniques is an extremely wasteful device. It was this observation that led to the invention of the parallel-serial-parallel accumulator. The parallel-serial-parallel accumulator is an efficient extension of the parallel-serial accumulator which results in speeds comparable with that of a parallel accumulator with no speed-up techniques, but with approximately one-fourth the number of logical inputs to the logical expressions for the adder.

Parallel-Serial-Parallel

The parallel-serial-parallel arrangement described here consists of three parallel 16-bit parallel-serial registers, with the bits of a 4-bit character in parallel, 4 characters in serial. Each of the three 16-bit groupings is referred to as a major character. Major character 1 contains bits 0–15, major character 2 contains bits 16–31, and major character 3 contains bits 32–47. In 4 pulse times the sum within each major character is computed. Carries generated as a result of these additions are then propagated and added into the next major character in the next 4 pulse times. At the end of these 8 pulse times, the probability that the carries will be finished propagating and the answer will be correct is $1 - 3 \times 2^{-17} = 0.999977$. Carry propagation completion can be sensed by means of a three-leg buffer, and as much additional time as necessary (8 pulse periods maximum) allowed to complete the carry propagation.

Fig. 5 shows a 48-bit binary PSP accumulator capable of either addition or subtraction. The equations for this are shown below.

S	=	Subtract; $\bar{S}$ = Add
A_n	=	Addend
B_n	=	Initial augend and final result
C_n	=	Carry functions
P_n	=	Final sum functions
CC_n	=	Character carry from each adder
T_{4n}	=	Timing function every 4th clock time such that $T_{4n}CC_n$ is the carry from the highest

order minor character in each major character.

Equation for Major Character 1 Adder

$$C_0 = T_{4n}\,CC_3 + \overline{T}_{4n}\,CC_1$$

$$C_1 = C_0 B_0 + \bar{S} A_0 B_0 + \bar{S} A_0 C_0 + S \bar{A}_0 B_0 + S \bar{A}_0 C_0$$

$$C_2 = C_1 B_1 + \bar{S} A_1 B_1 + \bar{S} A_1 C_1 + S \bar{A}_1 B_1 + S \bar{A}_1 C_1$$

$$C_3 = C_2 B_2 + \bar{S} A_2 B_2 + \bar{S} A_2 C_2 + S \bar{A}_2 B_2 + S \bar{A}_2 C_2$$

$$CC_1 = C_3 B_3 + \bar{S} A_3 B_3 + \bar{S} A_3 C_3 + S \bar{A}_3 B_3 + S \bar{A}_3 C_3$$

$$B_{12} = P_0 = A_0 B_0 C_0 + A_0 \bar{B}_0 \bar{C}_0 + \bar{A}_0 B_0 \bar{C}_0 + \bar{A}_0 \bar{B}_0 C_0$$

$$B_{13} = P_1 = A_1 B_1 C_1 + A_1 \bar{B}_1 \bar{C}_1 + \bar{A}_1 B_1 \bar{C}_1 + \bar{A}_1 \bar{B}_1 C_1$$

$$B_{14} = P_2 = A_2 B_2 C_2 + A_2 \bar{B}_2 \bar{C}_2 + \bar{A}_2 B_2 \bar{C}_2 + \bar{A}_2 \bar{B}_2 C_2$$

$$B_{15} = P_3 = A_3 B_3 C_3 + A_3 \bar{B}_3 \bar{C}_3 + \bar{A}_3 B_3 \bar{C}_3 + \bar{A}_3 \bar{B}_3 C_3$$

B_i shifted into B_{i+4} where $O \leq i \leq 11$

Equations for the major character 2 adder are the same with subscripts on A_n, B_n, C_n increased by 16, CC_1 substituted for CC_3, and CC_2 substituted for CC_1.

Equations for the major character 3 adder are the same with subscripts on A_n, B_n, C_n increased by 32, CC_2 substituted for CC_3, and CC_3 substituted for CC_1.

The average add time with 125 millimicrosecond carry propagate time and 1.33 MC flip-flops is approximately 6 microseconds.

This accumulator has been organized in such a manner that decimal arithmetic using binary coded decimal representation can be easily incorporated, since each minor character is a binary coded decimal digit. To include decimal arithmetic two areas need to be changed. The first is involved with rectification of the binary sum where either the binary coded decimal sum is greater than nine, or a major character carry was generated. This can easily be done by inserting the logic between B_{13} and B_9, B_{14} and B_{10}, and B_{11} below.

D = Decimal $\quad\quad \overline{D}$ = Binary

$$B_8 = B_{12}$$

$$B_9 = \overline{D}\,B_{13} + \overline{CC_1}\,\overline{B}_{15}\,B_{13} + D\,B_{15}\,B_{14}\,\overline{B}_{13} + D\,CC_1\,\overline{B}_{13}$$

$$B_{10} = \overline{D}\,B_{14} + \overline{CC_1}\,\overline{B}_{15}\,B_{14} + \overline{CC_1}\,\mathrm{B}_{14}\,B_{13} + D\,CC_1\,\overline{B}_{15}\,\overline{B}_{13} + D\,CC_1\,B_{15}\,\overline{B}_{14}\,B_{13} + CC_1\,B_{14}\,\overline{B}_{13}$$

$$\mathrm{B}_{11} = \overline{D}\,B_{15} + \overline{CC_1}\,B_{15}\,\overline{B}_{14}\,\overline{B}_{13} + D\,CC_1\,\overline{B}_{15}\,\overline{B}_{14}\,B_{13} + CC_1\,B_{15}\,B_{14}\,B_{13}$$

The above can be verified with a simple truth chart.

The second area that needs change is the generation of inter-digit carries. This is accomplished by adding a few terms to C_0 to take care of those cases where the decimal sum or difference is between 10 and 15.

$$\mathrm{C}_0 = \mathrm{T}_{4n}\,CC_3 + \overline{T}_{4n}\,CC_1 + \overline{T}_{4n}\,D\,B_{15}\,B_{14} + \overline{T}_{4n}\,D\,B_{15}\,B_{13} + T_{4n}\,D\,B_{37}\,B_{36} + + T_{4n}\,D\,B_{37}\,B_{35}$$

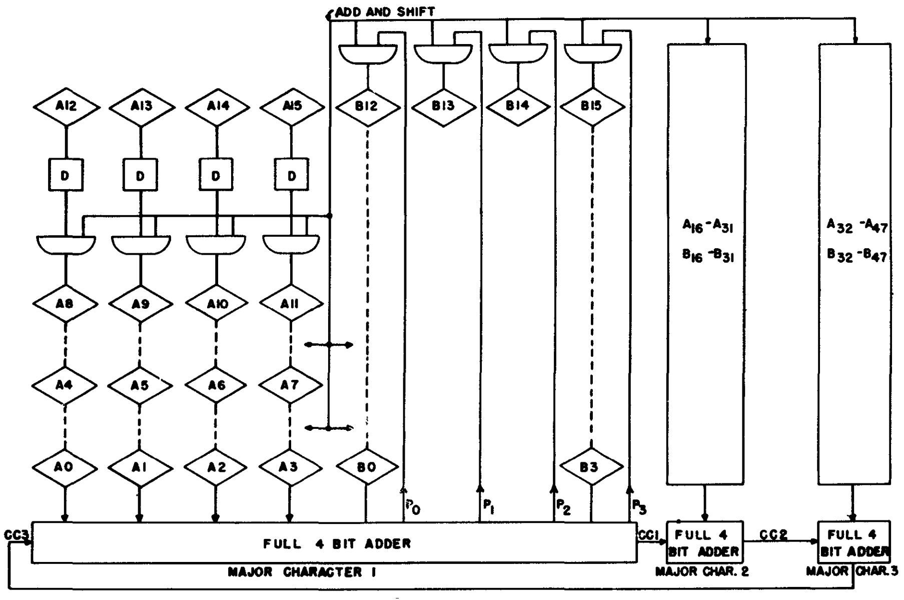


Fig. 5—48-bit binary parallel serial parallel accumulator.

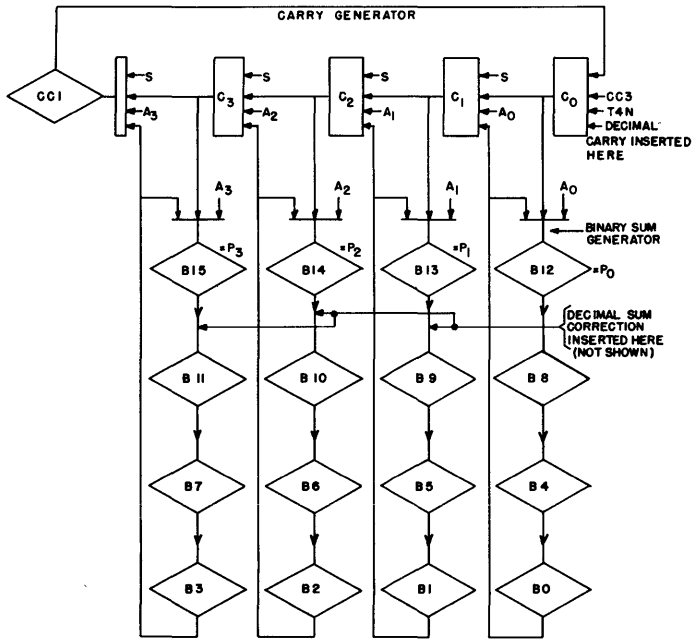


Fig. 6—Logical organization for major character 1 of parallel serial parallel accumulator.

The two corrections required the addition of only 230 diodes to the accumulator. No amplifiers were added. The probability that the answer will be correct after 8 pulse times is $1 - 3/2000 = 0.9985$. For these cases, up to 9 more pulse times may be required for the correct answer.

In addition to allowing a very economical method of arithmetic, the PSP format allows other machine simplifications over a parallel format. In particular, it is possible to transmit a 48-bit word to remote portions of the machine by means of time sharing 12 lines, resulting in a decreased number of cable drivers. It also allows the use of one flip-flop and 3 pulses of delay line to store 4 bits of information in those cases where the other 3 flip-flops are not required for manipulation reasons. A block diagram of type of storage is shown in Fig. 7.

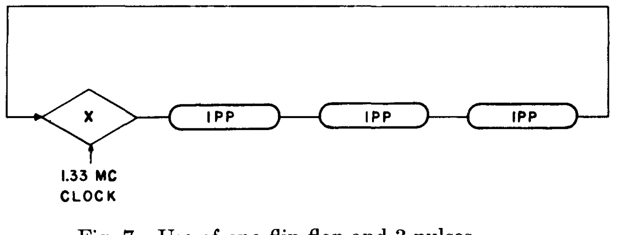


Fig. 7—Use of one flip-flop and 3 pulses of delay to store 4 bits.

Parallel-Serial-Parallel Arithmetic Example

Fig. 8 is an example illustrating two numbers being added together in a parallel-serial-parallel adder. The zeros immediately beneath the operands indicate the initial state of the carry circuits at the beginning of the addition. All of the numbers on a line with *tl* indicate the computation being performed during the first pulse period. The 11 indicates that the "4" and "7" in the low order position of major character 3 is added to form a 1 sum and a 1 carry, and, simultaneously, the "5" and "4" in the low order portion of major character 2 is added to form a "9" with 0 carry indicated by the 09, and, similarly, 1 and 6 are being added to form 07.

During the next pulse period, *t*2, (and through the same addition circuits which produced the "*tl*" addition) "2" plus "9" in major character 1, together with the previous carry, "0", forms the 11 on the *t*2 line. Similarly "1" plus "8" plus "0" carry forms 09 and "7" plus "1" carry forms 09.

This process is repeated until the *tl* line of word cycle 2. At this point, the carry from the high order position of major character 1 is added to the low order digit in the partial sum of major character 2. The carry is then propagated as shown until the 3699 is "corrected" to 3700. Similarly, the 9991 formed in major character 3 is corrected to 9992.

As seen in this example, the addition is completed after 8 pulse periods.

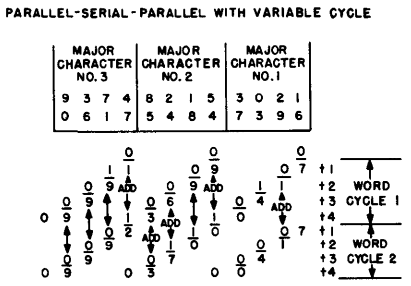


Fig. 8—Parallel-serial-parallel with variable cycle.

In summary, the parallel-serial-parallel format provides a fast arithmetic speed at a relatively economic cost of logical circuitry. The addition of the control memory to the system provides a wide range of flexibility in order to achieve an efficient usage of the computer.

The authors of this paper wish to extend credit to all those at the Datamatic Division of Minneapolis Honeywell Regulator Company, whose ideas contributed to the creation of this data processing machine.

Discussion

M. Rubinoff: A criterion often used to measure the efficiency of a computer design is computing per second per dollar. Can you comment on the efficiency of multiple operation for various types of problems?

Mr. Lourie: The multiple independent program operation feature costs remarkably little when integrated properly in the design of a data processor. The amount of work which can be accomplished per second — such work includes card reading, tape operations, etc., as well as actual computing — is, of course, increased tremendously in a multi-programmed design, since these various operations are being performed simultaneously rather than serially. This is especially so where the traffic control of the various programs is automatic so that maximum use is made of each memory cycle of the machine. I think, therefore, that it is obvious that the efficiency as measured by your criterion is exceptional in a machine of this nature.

J. Gosden (Leo Computers): What are the major uses of the co-sequence registers?

Mr. Lourie: The major uses of the co-sequence registers are to perform sub-routines without the use of "housekeeping" type orders.

M. S. Maxwell (US Naval Weapons Lab): What is the speed of the central computer memory and control memory?

Mr. Lourie: Six microseconds complete cycle time for both memories.

J. Daniels (ISI): Are there provisions to prevent one program from overwriting another in the main memory?

Mr. Lourie: An executive routine has been designed for use with all H-800 installations. This routine automatically solves all problems of interference with regard both to equipment and to memory space allocations. Automatic assignment of memory locations is arranged for by this routine on a non-conflicting basis.

Miss E. Berezin (Teleregister): How is the 256 bit control register loaded, particularly if one wished to start one program while others were in operation?

Mr. Lourie: The loading routine would use one of the unused eight sequence registers to load the control memory.

L. Clapp (Sylvania): Is there any possibility of losing information if several output/input trunks are working simultaneously?

Mr. Lourie: The machine speed is such that it can simultaneously handle eight input and eight output trunks, all of which occur to and from magnetic tape mechanisms. There would be 16 trunks acting simultaneously with approximately one-third of the computer time available for computation. The maximum input/output rate is such that no data would be lost.

J. H. Hughes (American Mutual Liability): What happens when you try to add, say, hexadecimal (15) plus (12)?

Mr. Lourie: If any generalized binary configuration is to be added to another binary configuration, the Binary Add order of the machine should be used and will lead to a correct answer. The normal Decimal Add orders are used only for operations involving the binary coded decimal code for digits 0–9.

B. Tasini (IBM): Can you comment on the debugging of problems on a multi-programmed computer?

Mr. Lourie: Debugging is not very much more difficult on a multi-program machine, since each program would be debugged independently, and then these programs would be run simultaneously.

P. Seaman (IBM): Is the program priority system and the I/O priority system fixed, or can it be specified by the programmer?

Mr. Lourie: There is no priority with regard to programs. Rather they are handled cyclicly in order. If one program requires the machine for a certain period of time, it can insert an order that will shut off all other programs and then, at some later time, turn these programs back on. The priority of peripheral equipment is fixed.

C. L. Foster (IBM): Is control memory addressable from main memory?

Mr. Lourie: Yes.

The Virtual Memory in the STRETCH Computer

JOHN COCKE AND HARWOOD G. KOLSKY†

EARLY in the planning of the STRETCH computer it was seen that by using the latest solid state components in sophisticated circuits it would be possible to increase the speed of floating point arithmetic by almost two orders of magnitude over that in existing computers. However, there seemed to be no possibility of developing on the same time-scale economically feasible large memories with more than a factor of ten or perhaps twenty increase in speed. As a result, the proposed system appeared to be in danger of being seriously memory-access limited.

Moreover, as the speed of the floating point operations increases, a larger and larger percentage of the computer's time is spent on "parasitic operations", *i.e.*, operations whose only function is program control and data selection. It was obvious that a radically new machine organization was necessary in order to capitalize upon the possibilities opened up by the high arithmetic speeds in the presence of relatively slow memories.

At this time, a number of persons were considering the possibility of a "look-ahead" device in which an independent indexing arithmetic unit would prepare the effective addresses of instructions and initiate memory references to a multiplicity of memory boxes. The data thus fetched would be held in high-speed buffer registers until needed by the arithmetic unit. This device would serve two desirable purposes: (1) some of the parasitic operations would be done in parallel and thus not delay the principal calculations, and (2) several memory boxes could be running simultaneously, giving the effect of higher memory speed.

Since our original work on the virtual memory and simulation in 1957–58, a large number of detailed changes have been made in the actual hardware design of STRETCH. These necessitated several modifications in the simulation program to estimate their effect on the overall system performance. In this report we are omitting many of these changes for expository reasons, since our purpose is to describe the virtual memory and timing simulation concepts, not to describe the STRETCH hardware exactly. The result is that the system described below embodies a more general system than that found in the simulator, which in turn is more general than that found in the actual computer.

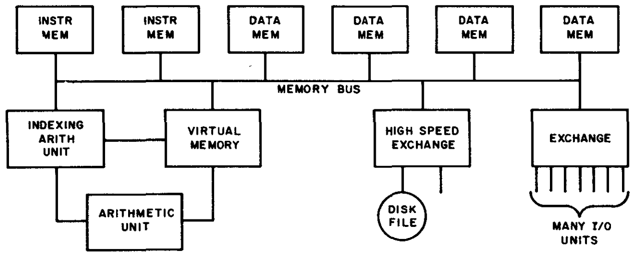


Fig. 1—Schematic of Stretch computer.

† International Business Machines Corporation, Poughkeepsie, New York.

General Description of the System

The major logically-independent blocks of the STRETCH computer are shown in Fig. 1. Each of the units pictured may be considered as operating asynchronously. That is, each does its tasks as fast as possible independently of the others. In theory, each box could have its own clocking circuits and still operate properly. In practice, for economy's sake they are all timed by the same master oscillator, but this does not destroy their *logical* independence.

The bus control unit serves as a routing agent between the memories and the various data processing units. If two or more units make a request simultaneously the control unit assigns priorities in the following order: (1) High-speed Exchange, (2) Basic Exchange, (3) Virtual Memory, and (4) Indexing Arithmetic Unit.

The Indexing Arithmetic Unit fetches instructions, performs all necessary indexing operations and sends the instructions to be executed to the Virtual Memory.

The Virtual Memory fetches and receives the data required by the instruction and holds this data until the arithmetic unit is ready for it. The Virtual Memory also performs all store operations. It holds the data generated by the arithmetic unit or indexing arithmetic unit until the memory to which the data must be sent is available. Thus the virtual memory acts not only as a "look-ahead" for instructions to be fed to the arithmetic unit, but also acts as a "look-behind" storage buffer.

The actual design of such a "look-ahead" device posed a number of logical problems, particularly in connection with conditional branches. However, a machine organization of this complexity requires a detailed timing analysis in order to determine the value of adding hardware in the form of the virtual memory. This is especially true since the sole function of the virtual memory is to increase machine speed, by increasing the efficiency of other devices. It was also felt that the timing analysis could not be made on the basis of a few trivial examples (e.g. matrix multiply). Machine performance obtained in this fashion can be extremely deceptive. Since a detailed timing analysis of a computer of this complexity is extremely tedious to carry out by hand, it became clear that if the job were to be done, it would be necessary to simulate the proposed machine on another computer. This prompted us to write the simulation program to be described later.

With the above general organization in mind, let us discuss some of the logical problems posed by such a system. The first problem is a result of the very concept which enables us to obtain such great benefits from the stored program computer — the ability to treat instructions as data. In a system such as we have proposed there is a large amount of simultaneous operation. For example, the indexing arithmetic unit may be busy preparing an instruction before previous instructions have been completed or even started by the arithmetic unit. One of these previous instructions may modify the instruction which is presently being indexed. The virtual memory must recognize this situation and allow the intervening instructions to be completed before doing the modified instruction.

A similar problem exists with respect to ordinary data. In order to operate several memories simultaneously, it is necessary to start obtaining data from these memories before the preceding operations have been completed. Yet, one of these operations may be a store into one of the data locations. The virtual memory must make provisions to insure that each instruction obtains the most up-to-date data as implied by the order of the program.

One of the novel features of the STRETCH computer is its elaborate interrupt system. Under this system, whenever some unexpected occurrence arises, the program will be interrupted and control will pass to a special routine which is designed to take care of the case in question, then return control to the original program. In this situation the virtual memory must have provisions to retain enough information so that when an interrupt occurs we can resume the computation exactly where we left off. It must be able to recognize which of the changes that have been made in advance are not desired and should be obliterated, and which are exact solutions that must be restored.

Another special case arises when a conditional branch on arithmetic results occurs. Here we will not know which of the two branches we should have taken until the preceding instruction is executed. In the case where the wrong path has been selected, the virtual memory must be prepared to drop the intermediate results which have been computed and pick up the correct branch in a way very similar to that of an interrupt.

Summing up all these logical problems, we may state that the fundamental rule for the virtual memory is that it must make the asynchronous and non-sequential computer give results identical to those which would be obtained by performing the program one instruction at a time in the order in which they are written.

Definitions

Operations

Operations are considered to be of three types:

(1) Bring or Fetch Type — All instructions requiring data to be transmitted from external memory to the virtual memory.

(2) Store Type — Instructions requiring the transmission of data from the virtual memory to external memory or index memory.

(*Note:* We consider all indexing instructions to be of the store type, although the store may be to either external memory or index memory.)

(3) Immediate Type — All operations not requiring data transmission.

Virtual Memory Quantities

(1) Virtual Memory — A number of virtual memory (or look-ahead) levels (numbered *0* to $N - 1$).

(2) Level of Virtual Memory — A collection of registers and control bits. The contents of the jth level are shown in Fig. 2.

(3) Instruction Address Register (I_j) — Contains the address of the instruction currently in the jth level.

(4) Operation Code Register (OP_j) — Contains the operation to be performed by the arithmetic unit.

(5) Store Bit (S_j) — a one-bit trigger which indicates the level, contains a store type instruction.

(6) Bring Bit (B_j) — A one-bit trigger which indicates the level, contains a fetch type instruction for which the data access has not been started.

(7) Forwarding Bit (F_j) — A one-bit trigger which indicates that the jth level must transmit data to another level.

(8) Forwarding Address (FA_j) — A register which contains the number of the level to which the data must be sent if F_j is set.

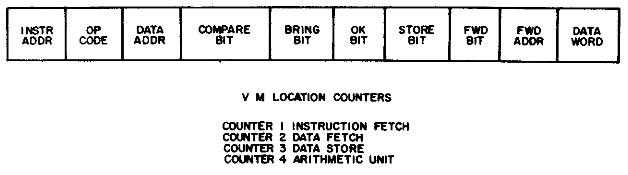


Fig. 2—Virtual memory — contents of one level.

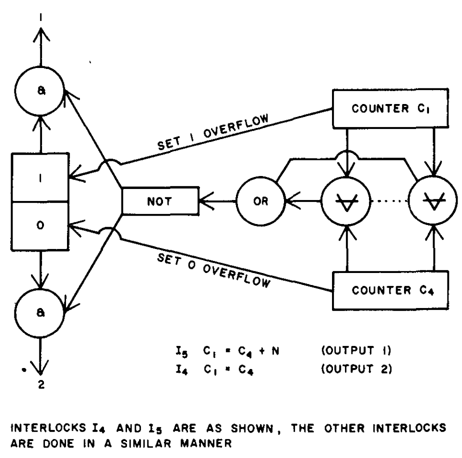

1
a
COUNTER C1
SET 1 OVERFLOW
1
NOT
OR
0
SET 0 OVERFLOW
COUNTER C4
a
I5 C1 = C4 + N (OUTPUT 1)
I4 C1 = C4 (OUTPUT 2)
2
INTERLOCKS I4 AND I5 ARE AS SHOWN, THE OTHER INTERLOCKS ARE DONE IN A SIMILAR MANNER

Fig. 3—Virtual memory interlocks.

(9) O. K. Bit (OK_j) — A trigger which when set indicates that the correct data for the instruction to be executed is present in the jth data field.

(10) Data Field (D_j) — A register which contains the operand data for the instruction.

(11) Data Address (DA_j) — The operand data address (already indexed by the IAU) for D_j.

(12) Compare Bit (C_j) — A trigger which if not set indicates the address in DA_j should not be included in any address comparisons being made.

Counters

The virtual memory is controlled by a set of counters which count mod(N), where N is the number of virtual memory levels.

(1) Counter one (C_1) — Indicates the level into which the next instruction may be placed.

(2) Counter two (C_2 — Indicates the level from which the next bring type instruction may be initiated.

(3) Counter three (C_3) — Indicates the level from which the next store type instruction may be initiated.

(4) Counter four (C_4) — Indicates the level from which the arithmetic unit will get its next operation and data.

Interlocks

The above counters must be interlocked in the following manner to assure proper sequential operation of the computer (see Fig. 3:)

(1) Interlock one (I_1): $C_1 = C_3 + N$ Prevents the IAU from placing the next operation into the level indicated by C_1 because an unexecuted store is still in the level.

(2) Interlock two (I_2): $C_1 = C_3$ Prevents a store from being initiated from the level indicated by C_3 because the store has already been done.

(3) Interlock three (I_3): $C_1 = C_2$ Similar to I_2, prevents a fetch from being initiated.

(4) Interlock four (I_4): $C_1 = C_4$ Prevents the arithmetic unit from executing an old instruction.

(5) Interlock five (I_5): $C_1 = C_4 + N$ Prevents the IAU from placing the next instruction into the level indicated by C_1 because the instruction there has not been executed yet.

Logic of the Virtual Memory

There are two basic precepts which must be kept in mind to understand the operation of the virtual memory:

(1) The OK bit (O_j) being set in the jth level indicates that the contents of D_j is the correct data called for by DA_j. All operations will be performed only under this condition, and logical decisions will be made in such a manner as to make sure this is the case.

(2) Addresses can be compared by the IAU with every DA_j address simultaneously. DA_j is not used for any level which does not have its C_j bit set. If a comparison exists between a new DA_j being placed in the virtual memory and an old DA_k, the compare bit C_k is turned off and the address of level j is placed in FA_k. This insures a unique meaning for the comparison. If this were not done, another instruction address DA_e might compare against *two* levels and thus cause an ambiguity.

Instruction Fetch Logic

Fig. 4 is a flow diagram of the IAU Instruction Fetch Procedure. The logic is as follows: If the IAU is ready to fetch another instruction, it compares the instruction address with all the DA_j's of virtual memory. If there is no comparison, the instruction fetch is initiated. If there is a comparison, the IAU

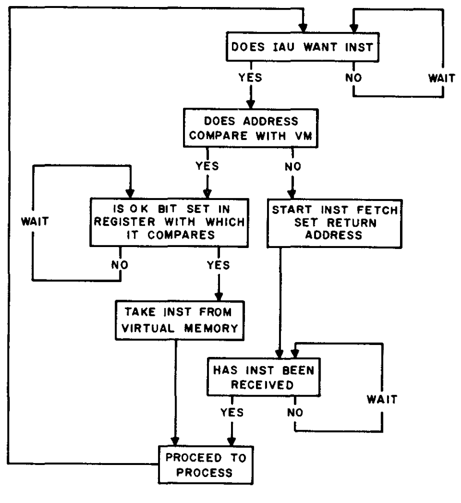

DOES IAU WANT INST
YES
NO
WAIT
DOES ADDRESS COMPARE WITH VM
YES
NO
WAIT
IS OK BIT SET IN REGISTER WITH WHICH IT COMPARES
START INST FETCH SET RETURN ADDRESS
NO
YES
TAKE INST FROM VIRTUAL MEMORY
HAS INST BEEN RECEIVED
YES
NO
WAIT
PROCEED TO PROCESS

Fig. 4—Instruction fetch procedure.

must take its instruction from the virtual memory provided the OK bit is set; otherwise, it must wait until the OK bit is set.

Note: This procedure prevents the logical difficulty mentioned earlier whicn would occur if the virtual memory contained a store order into the instruction presently being fetched.

For Example:		
	a	STORE Address *a* + *2*
	a + *1*	LOAD *M*, *i*
	a + *2*	ADD *N*, *i*
	a + *3*	----

The store to *a* + *2* must be done in sequence or the old value *N* would be used for the address instead of the quantity being set by *a*.

Indexing Logic

Fig. 5 shows the flow for instruction indexing. After determining that an instruction is ready to be indexed, the IAU tests whether or not the index value is available. If it is, the indexing operation is started; if not, the memory reference is started and the IAU waits until the data returns before proceeding. If the index-fetch has not been started, the IAU compares the index address against all the data addresses in virtual memory. If none compare, the index value is fetched normally. If one does compare, the index fetch is held up until the OK bit is set for the data. This value from the virtual memory is then used for indexing the instruction.

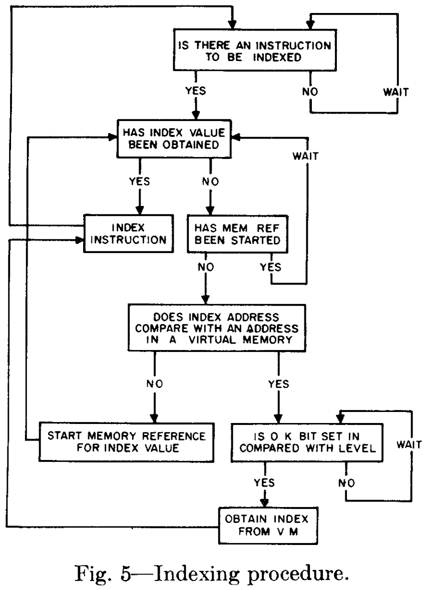

IS THERE AN INSTRUCTION TO BE INDEXED
YES
NO
WAIT
HAS INDEX VALUE BEEN OBTAINED
WAIT
YES
NO
INDEX INSTRUCTION
HAS MEM REF BEEN STARTED
NO
YES
DOES INDEX ADDRESS COMPARE WITH AN ADDRESS IN A VIRTUAL MEMORY
NO
YES
START MEMORY REFERENCE FOR INDEX VALUE
IS O K BIT SET IN COMPARED WITH LEVEL
WAIT
YES
NO
OBTAIN INDEX FROM V M

Fig. 5—Indexing procedure.

Logic of Putting Instructions in the Virtual Memory

(1) Figs. 6, 6A, 6B, 6C represent the logical flow for putting instructions into the virtual memory. If the indexing arithmetic unit has an instruction prepared for the virtual memory, it may transmit the instruction into the virtual memory if interlocks one and five do not forbid it. These interlocks prohibit a new instruction from destroying an old one which has not been executed as yet, whether an arithmetic operation (I_5) or an unexecuted store (I_1). The handling of the instructions varies depending on whether they are of the bring type, store type, or immediate type.

(2) The bring type, as described in Fig. 6A, proceeds as follows: If the effective data address of the instruction compares with the *DA* address in some level, the instruction, its op code, and effective data address are loaded into the level marked by C_1. The compare bit for level C_1 is set to one while the compare bit for the compared-with level is set to zero. If the OK bit in this compared-with level is set, meaning that the data located there is correct, the data is transmitted directly to the C_1 level and its OK bit is also set. If the OK bit is not set, we must tag the compared-with level by setting its forwarding bit and by putting the value of C_1 into its forwarding address; the bring bit for level C_1 is also set to *zero* since no further data fetch is required.

If the effective data address does not compare with any Virtual Memory level, the instruction is put directly into level C_1, its OK bit is set to *zero,* and its bring bit is set to *one,* indicating that a fetch must be started.

(3) Fig. 6B shows the store type procedure. If the effective address of the instruction does not compare with the *DA* address in some level, the instruction is placed into the level marked by C_1. The store bit is set to one indicating that a store will be required. The level's bring bit and forwarding bit are set to *zero*; its compare bit is set to *one.* If on the other hand the addresses do compare, the same procedure is followed; but in addition, the compare bit in the level compared-with is set to zero so that future comparisons will not use it.

The OK bit has not yet been set. It is set to *one* if the operation is an index store and set to *zero* if it is an ordinary store. For the ordinary store it is clear that the OK bit should be *zero* since the data must come from the arithmetic unit after the preceding instruction is executed.

As was mentioned in the definition previously we treat all indexing instructions as store type and place the new value of the indexed quantity into the virtual memory. This is done because the indexing arithmetic unit is going ahead of the normal order of instruction execution and an interruption may occur before this indexing instruction should have

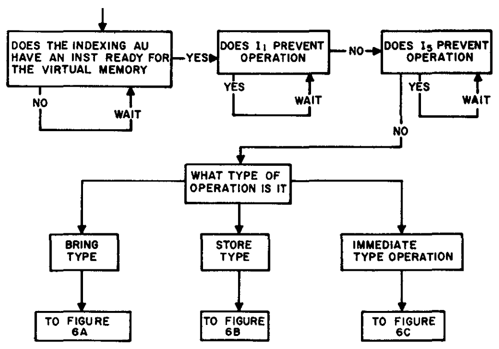

DOES THE INDEXING AU HAVE AN INST READY FOR THE VIRTUAL MEMORY
YES
NO
WAIT
DOES I1 PREVENT OPERATION
NO
YES
WAIT
DOES I3 PREVENT OPERATION
YES
WAIT
NO
WHAT TYPE OF OPERATION IS IT
BRING TYPE
STORE TYPE
IMMEDIATE TYPE OPERATION
TO FIGURE 6A
TO FIGURE 6B
TO FIGURE 6C

Fig. 6—Procedure for placing instructions into the virtual memory.

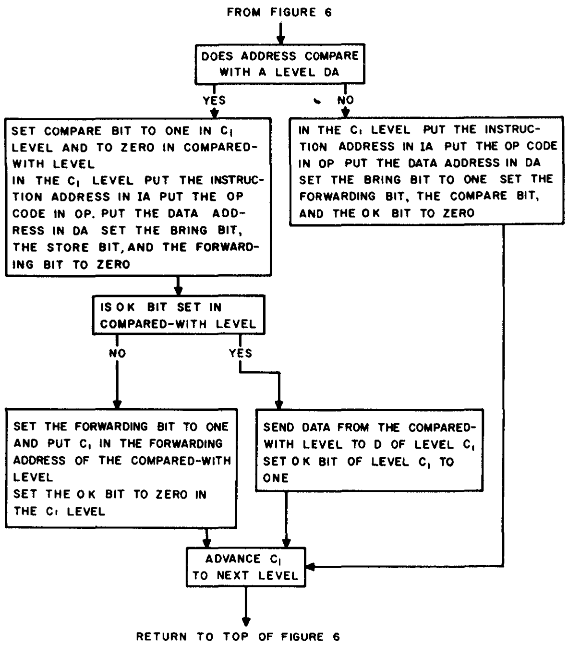

FROM FIGURE 6
DOES ADDRESS COMPARE WITH A LEVEL DA
YES
NO
SET COMPARE BIT TO ONE IN C1 LEVEL AND TO ZERO IN COMPARED-WITH LEVEL
IN THE C1 LEVEL PUT THE INSTRUCTION ADDRESS IN IA PUT THE OP CODE IN OP. PUT THE DATA ADDRESS IN DA SET THE BRING BIT, THE STORE BIT, AND THE FORWARDING BIT TO ZERO
IN THE C1 LEVEL PUT THE INSTRUCTION ADDRESS IN IA PUT THE OP CODE IN OP PUT THE DATA ADDRESS IN DA SET THE BRING BIT TO ONE SET THE FORWARDING BIT, THE COMPARE BIT, AND THE OK BIT TO ZERO
IS OK BIT SET IN COMPARED-WITH LEVEL
NO
YES
SET THE FORWARDING BIT TO ONE AND PUT C1 IN THE FORWARDING ADDRESS OF THE COMPARED-WITH LEVEL
SET THE OK BIT TO ZERO IN THE C1 LEVEL
SEND DATA FROM THE COMPARED-WITH LEVEL TO D OF LEVEL C1 SET OK BIT OF LEVEL C1 TO ONE
ADVANCE C1 TO NEXT LEVEL
RETURN TO TOP OF FIGURE 6

Fig. 6(a)—Logical conditions for bring type operations.

been done. In this case, the old value of the index is still in the index register. On the other hand the indexing arithmetic unit compares with the virtual memory and extracts the most recent value of the index for indexing succeeding instructions. The OK bit is set to *one* since the appropriate data is in the above level. Both the new and old index values must be carried along to give logically correct conditions in the case of an interrupt. A situation very similar to interrupt occurs in branches on arithmetic results where the indexing arithmetic unit "guesses" which branch will be taken and proceeds with fetching and processing the instructions on this branch, subject to being wiped out if the guess proves to be wrong. (See the discussion on "Wrong way Branches" below.)

(4) Immediate type instructions are the simplest type because they essentially carry their data with them. Fig. 6C shows the logic in this case.

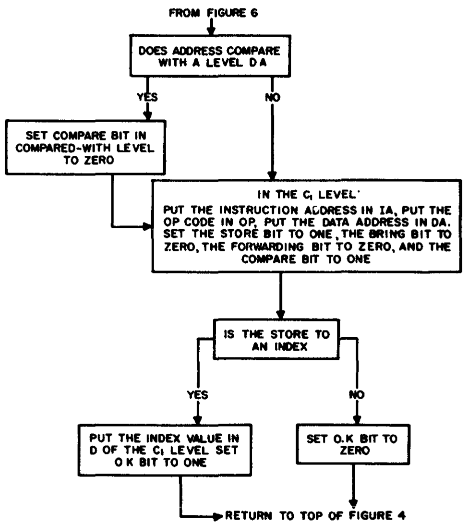

FROM FIGURE 6
DOES ADDRESS COMPARE WITH A LEVEL DA
YES
NO
SET COMPARE BIT IN COMPARED-WITH LEVEL TO ZERO
IN THE C1 LEVEL
PUT THE INSTRUCTION ADDRESS IN IA, PUT THE OP CODE IN OP, PUT THE DATA ADDRESS IN DA. SET THE STORE BIT TO ONE, THE BRING BIT TO ZERO, THE FORWARDING BIT TO ZERO, AND THE COMPARE BIT TO ONE
IS THE STORE TO AN INDEX
YES
NO
PUT THE INDEX VALUE IN D OF THE C1 LEVEL SET O K BIT TO ONE
SET O.K BIT TO ZERO
RETURN TO TOP OF FIGURE 4

Fig. 6(b)—Logical conditions for store type operations.

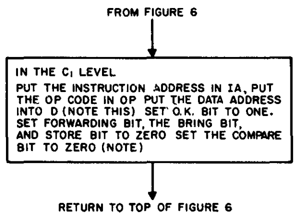

FROM FIGURE 6
IN THE C1 LEVEL
PUT THE INSTRUCTION ADDRESS IN IA, PUT THE OP CODE IN OP PUT THE DATA ADDRESS INTO D (NOTE THIS) SET O.K. BIT TO ONE. SET FORWARDING BIT, THE BRING BIT, AND STORE BIT TO ZERO SET THE COMPARE BIT TO ZERO (NOTE)
RETURN TO TOP OF FIGURE 6

Fig. 6(c)—Logical conditions for immediate type operations.

The instruction is placed in the virtual memory level marked by C_1. The address field of the instruction is placed in the data field of C_1. The OK bit is set to *one* indicating the data is present. The bring and store bits are both set to *zero*. The compare bit is set to *zero* since the DA address field has no meaning for immediate type ops. (The data address of the last instruction which occupied this level still remains in DA, so it has no relation to the present *D* field.)

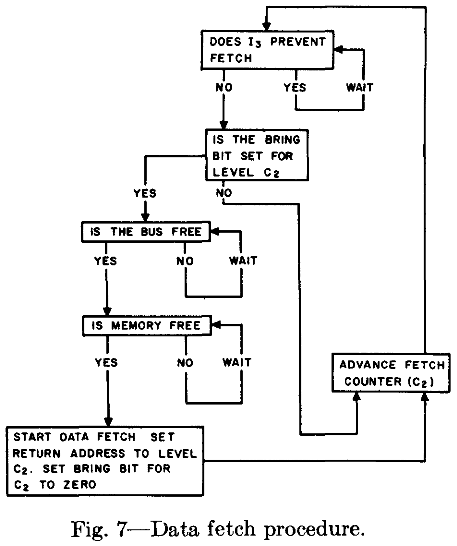

DOES I3 PREVENT FETCH
NO
YES
WAIT
IS THE BRING BIT SET FOR LEVEL C2
YES
NO
IS THE BUS FREE
YES
NO
WAIT
IS MEMORY FREE
YES
NO
WAIT
START DATA FETCH SET RETURN ADDRESS TO LEVEL C2. SET BRING BIT FOR C2 TO ZERO
ADVANCE FETCH COUNTER (C2)

Fig. 7—Data fetch procedure.

Logic of Data Fetching

See Fig. 7: When an instruction of the bring type has been placed in the virtual memory, the data required by the instruction in general will not be present (unless a comparison exists as was described above) and thus the data must be obtained from core storage. The fetch cannot be started if interlock I_3 holds, which means all the fetches corresponding to the instructions presently in the virtual memory have been started. If a fetch is possible, the bring bit at level C_2 indicates whether or not a fetch is necessary. If necessary the fetch may be started if the memory bus and memory unit corresponding to the data address are not already being used. When the fetch is started, the bring bit for level C_2 is set to zero. The counter C_2 is then stepped forward to the next level.

Logic of Data Storing

Fig. 8 shows the Data Store Logic, which is very similar to that for data fetching just described. The only significant difference is that the OK bit must be set before the operation can be started.

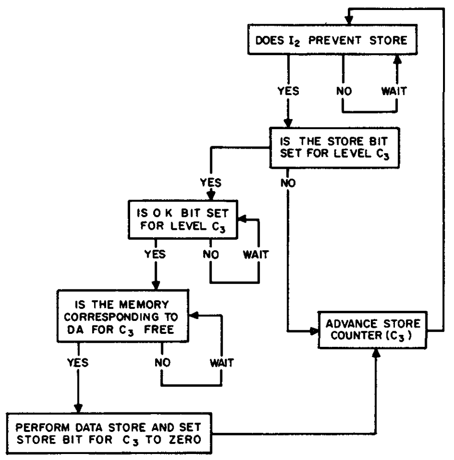

DOES I_2 PREVENT STORE
YES
NO
WAIT
IS THE STORE BIT SET FOR LEVEL C_3
YES
NO
IS O K BIT SET FOR LEVEL C_3
YES
NO
WAIT
IS THE MEMORY CORRESPONDING TO DA FOR C_3 FREE
YES
NO
WAIT
ADVANCE STORE COUNTER (C_3)
PERFORM DATA STORE AND SET STORE BIT FOR C_3 TO ZERO

Fig. 8—Data store procedure.

Logic for Placing Data into the Virtual Memory

In Fig. 9, we see the logical conditions which must be satisfied by the data returning from memory addressed to the virtual memory. The return address which was supplied when the fetch was started selects the level into which the data will be placed. The OK bit is then set to *one*, indicating that the proper data is in the level. The operation is complete at this point unless the forwarding bit is set. In this case, the data must be forwarded to the level designated by the forwarding address. This procedure continues from level to level as long as the data continues to arrive into a level whose forwarding bit is set. This procedure automatically supplies all operands present having identical data addresses with the proper data, without additional memory references.

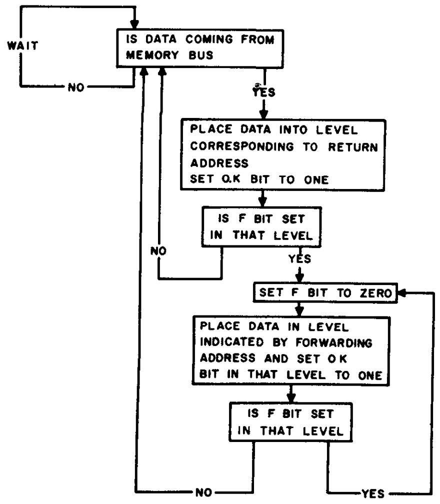

WAIT
IS DATA COMING FROM MEMORY BUS
NO
YES
PLACE DATA INTO LEVEL CORRESPONDING TO RETURN ADDRESS SET O.K BIT TO ONE
IS F BIT SET IN THAT LEVEL
NO
YES
SET F BIT TO ZERO
PLACE DATA IN LEVEL INDICATED BY FORWARDING ADDRESS AND SET O K BIT IN THAT LEVEL TO ONE
IS F BIT SET IN THAT LEVEL
NO
YES

Fig. 9—Procedure for placing data into virtual memory.

Logic of Removing Instructions from the Virtual Memory

In Fig. 10, we notice that as the arithmetic unit completes an instruction it checks to see if the next instruction in the virtual memory is ready to be executed (indicated by interlock I_4). Note that the operation may be an unconditional branch, a conditional branch, or an index type store, as well as a normal bring or store type instruction involving the accumulator. Fig. 10 shows only the cases which involve the universal accumulator. Instructions such as the unconditional branches are merely ignored at this point. They are carried along only to provide the data for recovery in the event an interrupt occurs. The execution of the conditional branches on arithmetic results are described in the next section.

If the next instruction marked by counter C_4 is ready, it is fed into the arithmetic unit. If it is a store

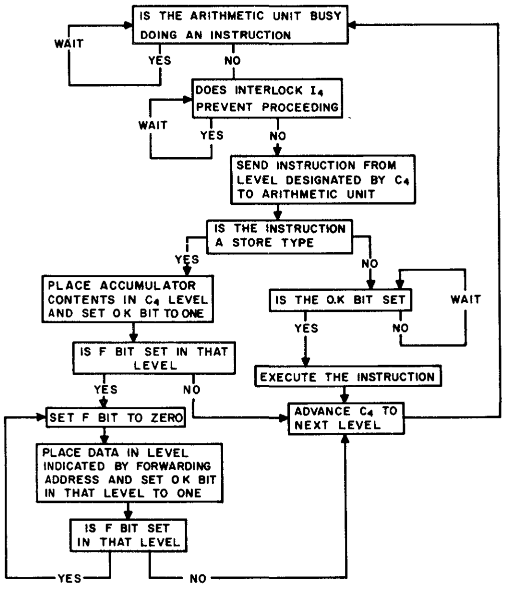

IS THE ARITHMETIC UNIT BUSY DOING AN INSTRUCTION
WAIT
YES
NO
DOES INTERLOCK I_4 PREVENT PROCEEDING
WAIT
YES
NO
SEND INSTRUCTION FROM LEVEL DESIGNATED BY C_4 TO ARITHMETIC UNIT
IS THE INSTRUCTION A STORE TYPE
YES
NO
PLACE ACCUMULATOR CONTENTS IN C_4 LEVEL AND SET O K BIT TO ONE
IS THE O.K BIT SET
WAIT
YES
NO
IS F BIT SET IN THAT LEVEL
YES
NO
EXECUTE THE INSTRUCTION
SET F BIT TO ZERO
ADVANCE C_4 TO NEXT LEVEL
PLACE DATA IN LEVEL INDICATED BY FORWARDING ADDRESS AND SET O K BIT IN THAT LEVEL TO ONE
IS F BIT SET IN THAT LEVEL
YES
NO

Fig. 10—Procedure for removing instructions from virtual memory.

type, the data is gated from the accumulator into the data field of level C_4, and the OK bit is set to one. If the forwarding bit of the level is set, a forwarding procedure in this case is *essential* for the proper logical operation of the computer, whereas in the bring case it is a time-saver only.

If the instruction is not a store type, the arithmetic unit must hold up until the OK bit for the level is set. When the OK bit is set, the instruction is gated into the arithmetic unit and executed.

Logic of Interrupt Procedure

If for any cause an interrupt (or trap) from a special condition occurs, the instruction which is being executed in the arithmetic unit is completed. However, the next instruction is not executed in spite of the fact all the data preparation for it may have been completed. The address in the *IA* (instruction address) field will serve as the value to reset the instruction counter if it is desired.

The Virtual Memory is initialized, *i.e.*, set to the starting conditions of an interrupt, with the exception that all store orders which have already received data from the accumulators must be executed first. If the interrupt is of such a nature that the normal flow of instructions is not resumed, the procedure of storing the modified values of the index registers in the Virtual Memory gives logically correct results, *i.e.*, the same as if the interrupt had occurred before the indexing took place.

Description of Timing Simulation Program

During the logical design of STRETCH it was necessary to prove the value of the virtual memory concept and to assist in the selection of optimum values of various system design parameters. Examples of such parameters are: The number of memory boxes, interlace and allocation of memory addresses, and numbers of virtual memory levels. Also of interest were trade-off factors for speeds of indexing arithmetic unit, memories, etc.

In November 1957 the Timing Simulator (SIM-2) described here was written for the IBM 704. This program attempted to answer such questions quantitatively by simulating the time-wise operation of STRETCH on typical test programs coded in STRETCH language.

The basic logic of the 704 program follows the principles just described in the preceding section for the virtual memory. It should be stressed that the simulator is a *timing* simulator and does not execute the instructions in an arithmetic sense. It traces the time-wise progress of the instructions through the components of the computer, observing all the interlocks and time delays necessary for correct representation of the behavior of the machine.

One of the fundamental concepts in the STRETCH design is that of *asynchronous* operation of the components. This means that there are a large number of logical steps being executed at any one time in the computer, each of them proceeding at its own rate. To simulate this flow of many parallel continuous operations, we have broken the continuous time variable into finite time steps. The basic time step is taken as 0.1 microsecond in the simulator.

By taking 0.1 microsecond as our quantum of time, we are automatically setting the scale of the smallest circuit entities which we will consider as being those which accomplish complete functions in 0.1 microsecond or few multiples thereof. Thus, by using this pholosophy, and considering many of the components of the computer as "black boxes", we greatly simplify the details which must be considered without introducing serious timing inaccuracies.

Our experiénce has indicated that more information was gained by making a large number of fast parameter studies using different configurations and programs than could have been obtained by a very slow, detailed simulation of a few runs with more precision per run. Even so, our time scale is *too* fine to make serious input-output application studies. These would require a simpler simulator having at least a factor of 10 coarser basic time interval.

Logic of the Simulator

In the asynchronous organization of STRETCH there can be many major components operating at any one time. To achieve this parallel effect in the simulator we essentially "hold time still" and scan the entire machine representation at each time step. Although every major block of the program is traversed at each time step, if there is no activity required in a given block, only a few tests need be made by the code.

If in this process it is determined that a given logical unit should do an operation, the time interval required for the operation is obtained from a table of constants. The speed of the various logical units can thus be changed parametrically by changing the values in the tables. A constant obtained from the tables is inserted into a memory location called the time counter for that unit. At each time step the program reduces this counter by one until it reaches zero. Thus, the fact that the counter is non-zero can be used to indicate that the particular logical unit is busy and not available to service other requests. When the counter is zero the unit can consider a new input.

In addition to the time counters many of the logical blocks contain other conditions or interlocks which affect the operation of the block. These conditions are stored in the program and tested before action is undertaken.

It is interesting to note that since the simulator simulates timing only, the sequence of instructions to be executed must be furnished as a "string" with all loops unwound. However, to make the computer behave as it actually would, the loops must be furnished with "wrong way" paths given for the cases where the computer would take such paths. Also one must furnish *more* than enough information along such paths since it is difficult to predict in advance how far the computer will get down the wrong path before it it called back.

Parameters are changed from one run to another by use of control cards. The control cards are set up in such a way that any number of parameters may be changed between runs. Results are given either as detailed timing charts or as summary listings for each problem. The usual procedure has been to print only summary results while making a series of parameter studies. The detailed timing charts as printed on the 704 for most problems would be about 50 feet long for each run. Since over 1000 cases have been run, it is clear that only a few cases could be printed in full detail. These are particularly useful in seeking the causes of conflicts which slow the computer.

Results of Parameter Studies

When the simulator program was completed, we undertook a series of studies in which the main parameters describing the STRETCH system were varied one or two at a time in order to get a measure for the importance of different effects. After this we began to specialize the studies towards answering specific questions in the STRETCH design.

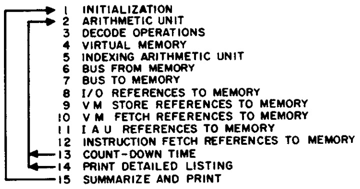

1 INITIALIZATION
2 ARITHMETIC UNIT
3 DECODE OPERATIONS
4 VIRTUAL MEMORY
5 INDEXING ARITHMETIC UNIT
6 BUS FROM MEMORY
7 BUS TO MEMORY
8 I/O REFERENCES TO MEMORY
9 V M STORE REFERENCES TO MEMORY
10 V M FETCH REFERENCES TO MEMORY
11 I A U REFERENCES TO MEMORY
12 INSTRUCTION FETCH REFERENCES TO MEMORY
13 COUNT-DOWN TIME
14 PRINT DETAILED LISTING
15 SUMMARIZE AND PRINT

Fig. 11—SIM — 2 simplified flow diagram.

The simplified flow diagram in Fig. 11, indicates the order in which the subroutines for the various logical units are executed at each time step. Using the types of techniques just described above, the logical subroutines simulate the action of the components of the computer such as the virtual memory, arithmetic unit, etc.

Some Results of the Simulation Studies

Fig. 12 shows examples of the type of output listings given by the simulator. Fig. 12 is a piece of a long timing chart with each line of printing representing 0.1 microsecond of time. The columns represent the various components of the computer. On the left and right are timing counts subdividing each microsecond. On the far right are conflict indicators (*C* on the charts) and waiting indicators, *W*, which indicate when interlocks prevent operations from proceeding.

The 2nd column, *II*, gives the number of the instruction being indexed. The 4th column, *AU*, gives the number of the instruction using the arithmetic unit. The next four columns represent the instructions using the memory buses. The columns labeled *X*- *F*-, and *M*- represent the index, fast, and main memories. A string of *X*'s in the columns represents the cycle time of the memory. The number indicates the instruction using the memory and the number of times which it is repeated gives the readout time of the memory. The columns *L*- indicate which instruction is located in the virtual memory levels. The other columns are for details in analysis and need not be considered here.

Five of the test problems used most frequently are described below. Other test problems were used for specific studies, but since the results were similar for all problems of a given type, we gradually discontinued using them. The following were originally selected as being typical of different classes of problems.

(1) *Mesh Problem* — Part of an hydrodynamics problem from Los Alamos. It contains a more or less "average" mixture of instructions for scientific problems: 85% floating point instructions, 14% index modification instructions, and 1% VFL. It is usually arithmetic unit limited.

(2) *Monte Carlo Branching Problem* — Part of an actual Monte Carlo neutron diffusion code. It represents a chain of logical decisions with very little arithmetic in between. It contains 47% floating point, 15% index modification instructions, and 36% branches of the indicator and unconditional types. It is largely instruction-access limited.

(3) *Reactor Problem* — The inner loop of a neutron diffusion problem. It consists of 90% floating point arithmetic (39% of which are multiplys) and 10% index modification instructions. It is almost entirely arithmetic unit limited.

(4) *Computer Test Problem* — The evaluation of a polynominal using computed indices. It has 71% floating point, 10% index modification, 6% VFL and 13% indicator branches. It is usually arithmetic unit limited, but not for all configurations.

(5) *Simultaneous Equations* — The inner loop of a matrix inversion routine 67% floating point and 33% index modification. Arithmetic and logic are about equally important. It is limited both by arithmetic and instruction-access speeds.

Speed vs. Number of Levels of Virtual Memory

Fig. 13 shows the effect on computer performance

	II	IS	AU	IF	IM	OF	OM	X1	X2	F1	F2	F3	F4	M1	M2	M3	M4	M5	M6	M7	M8	L1	L2	L3	L4	L5	L6	L7	L8	FD	MD	MC		
1	1																													1		2	1	W
2	1																													1		1	2	CW
3	1																													1			3	CW
4	1					1																								3		2	4	W
5	1					1																								3		1	5	W
6	1										1X																			3			6	W
7	1					3					1X																					2	7	W
8	1					3					1X																					1	8	W
9	1									3X	1X																						9	W
10	1			1						3X	X																					2	10	W
1	1			1						3X	X																					1	1	W
2	1									3X																							2	W
3	1			3						X																						2	3	W
4	1	1		3						X																						1	4	W
5	1	1																															5	W
6	1	2																														2	6	W
7	1	4																														1	7	W
8		1																															8	W
9	2	1																							1							2	9	
10	2	2																							1							1	10	
1	2	4	1					x																	1								1	W
2		1																							1							2	2	W
3	3	1																						2	1					5		1	3	
4	3	2																						2	1					5			4	
5	3	4	2			5		x																2	1							2	5	W
6		1				5																		2	1							1	6	W
7	4	1									5X												3	2	1								7	
8	4	2									5X												3	2	1							2	8	
9	4	2	3					x			5X												3	2	1							1	9	W
10	4	4									5X												3	2	1								10	W
1		1		5							X												3	2	1							2	1	W
2		1		5							X											4	3	2	1					7	4	1	2	
3		1																				4	3	2	1					7	4		3	W
4	5	1				7	4															4	3	2	1							2	4	W
5	5	2				7	4															4	3	2	1							1	5	W
6	5	2								7X				4X								4	3	2	1								6	W
7	5	4								7X				4X								4	3	2	1							2	7	W
8		1								7X				4X								4	3	2	1							1	8	W
9	6	1								7X				4X								4	3	2	5						5		9	W
10	6	2		7						X				4X								4	3	2	5						5	2	10	W
1	6	4		7						X				4X								4	3	2	5						5	1	1	W
2		1												4X								4	3	2	5						5		2	W
3	7	1					5							4X								4	3	6	5					9		2	3	W
4	7	2			4		5							X								4	3	6	5					9		1	4	W
5	7	2			4									X	5X							4	3	6	5					9			5	W
6	7	4				9								X	5X							4	3	6	5							2	6	
7		1	4			9								X	5X							4	3	6	5							1	7	
8	8	1	4								9X			X	5X							4	7	6	5						7		8	
9	8	2									9X			X	5X							4	7	6	5						7	2	9	W
10	8	2						x			9X			X	5X							4	7	6	5						7	1	10	W
1	8	2									9X			X	5X							4	7	6	5						7		1	W
2	8	4		9			7				X			X	5X							4	7	6	5							2	2	W
3		1		9	5		7				X			X	X							4	7	6	5							1	3	W
4		1			5									X	X		7X					8	7	6	5					11			4	CW
5	9	1												X	X		7X					8	7	6	5					11		2	5	C
6	9	2	5												X		7X					8	7	6	5					11		1	6	C
7	9	4	5												X		7X					8	7	6	5					11			7	C
8		1	5			11									X		7X					8	7	6	5							2	8	C
9	10	1	5			11									X		7X					8	7	6	9							1	9	C
10	10	2	5							11X					X		7X					8	7	6	9								10	C
1	10	2	5					x		11X					X		7X					8	7	6	9							2	1	C
2	10	2			7					11X					X		X					8	7	6	9							1	2	CW
3	10	4	6		7			x		11X					X		X					8	7	6	9								3	C
4		1		11						X					X		X					8	7	6	9							2	4	C
5		1	7	11						X							X					8	7	10	9					13		1	5	C
6		1	7														X					8	7	10	9					13			6	C

Fig. 12—Listing of simulator print-out.

of varying the number of levels of virtual memory. Curves for the Monte Carlo and Mesh Calculations with two sets of arithmetic and indexing arithmetic speeds are shown. The *AU* times given are averages for all operations.

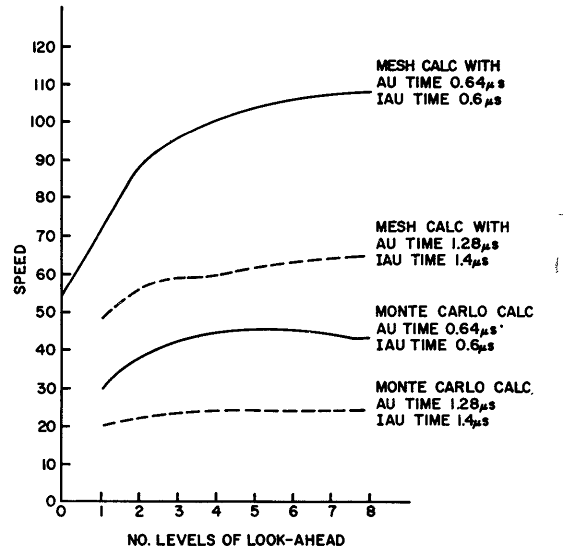


Fig. 13—Computer speed vs. no. of levels of look-ahead registers; 4 main mems. 2.0 μs; 2 fast mems. 0.6 μs for two sets of arith. speeds.

A number of interesting results are apparent from these curves:

(1) There is a tremendous gain to be had in going to the virtual memory organization. The point for "0 levels" means that the arithmetic unit is tied directly to the instruction preparation unit, although simple Indexing-Execution overlap is still possible.

(2) The gain in performance goes up very rapidly for the first two levels, then rises more slowly for the rest of the range.

(3) A large number of levels does the Monte Carlo problem less good than the Mesh problem because constant branching in the former spoils the flow of instructions. Notice that the curve for the Monte Carlo problem actually *decreases* slightly beyond six levels. This phenomenon is a result of memory conflicts caused by extraneous memory references started by the computer running ahead on the wrong-way paths of branches.

(4) The computer performance on a given problem is clearly less for slower arithmetic speeds. However, it is important to note that the *sensitivity* of the performance is also less for slower arithmetic speeds. The virtual memory improves the performance in either case, but it is not a substitute for a fast arithmetic unit.

Speed vs. Number of Main Memory Units

Fig. 14 shows how internal computer performance varies with the total number of memory units for a particular problem. The entire calculation is assumed to be contained in memory for all cases. The speed gain from overlapping memories is quite apparent from the graphs.

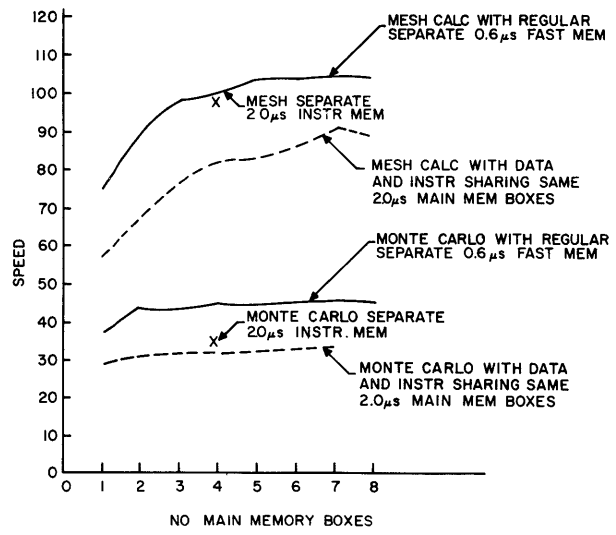


Fig. 14—Computer speed vs. number of main memory boxes: 4 level LA; 0.6 μs I AU time; 0.64 μs AU time.

The speed differential between having and not having instructions separated from data arises from delays in instruction fetches caused by the memory units being busy with data. The size of this effect varies from problem to problem, being less pronounced for problems which are arithmetic limited and more for logical problems.

The *X*'s on the graph show the effect of replacing the 0.6 μsec instruction memories by a pair of 2.0 μsec memories. The resulting performance change is small for the Mesh problem, which is arithmetic limited, but large for the instruction-fetch limited Monte Carlo problem.

Speed vs. Arithmetic Unit and Indexing Arithmetic Unit Times

Although everyone realizes the importance of arithmetic speed on overall computer performance, it was not until the simulator results became available that the true importance of the indexing arithmetic speeds was recognized. Figs. 15 and 16 show a two-parameter family of curves giving the computer speed as a function of the *AU* and *IAU* times.

Fig. 16, in which the arithmetic time is the abscissa, shows an interesting "saturation" effect where the computer performance is independent of *AU* speed below some critical value. Thus it makes no sense to strain *AU* speeds if the *IAU* is not improved to match. The curves in Fig. 15 show the same effect, *i.e.*, the *IAU* speed serves as a "ceiling" on performance beyond which the *AU* speed cannot pass.

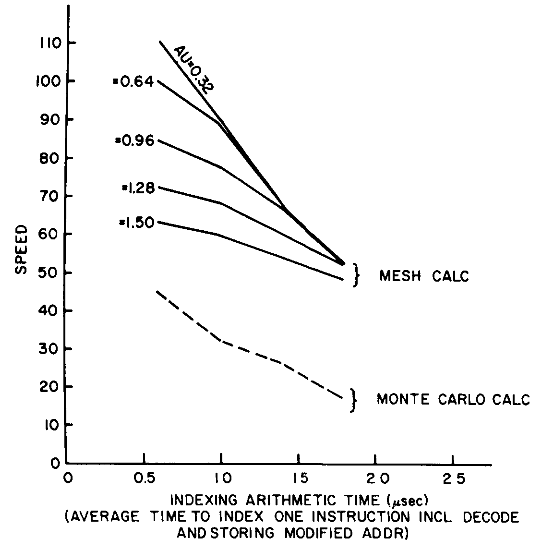


Fig. 15— Computer speed vs. indexing arith. times for various arithmetic unit times: 4 main mems. 2.0 μs; 2 fast mems. 0.6 μs; 4 levels of look-ahead.

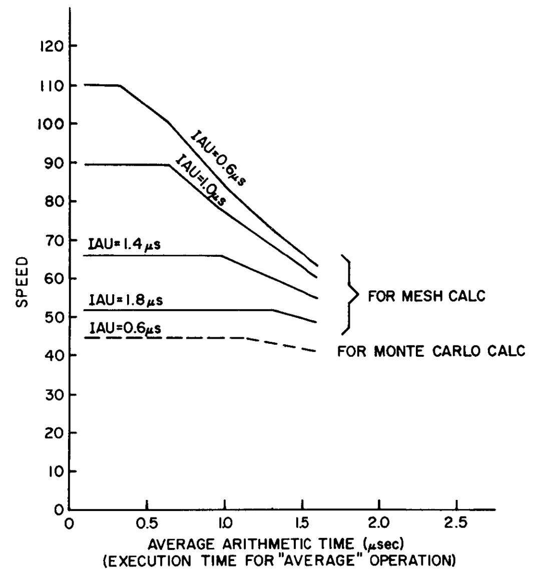


Fig. 16—Computer speed vs. arithmetic times for various indexing arithmetic unit times: 4 main mems. 2.0 μs; 2 fast mems. 0.6 μs; 4 levels of look-ahead.

Arithmetic Unit Efficiency

One fallacy which is frequently quoted is that the goal of improved computer organization is to increase the arithmetic unit efficiency. Actually there are two reasons why this is not the goal in itself. The first is that arithmetic efficiency depends strongly on the mixture of arithmetic and logic in a given problem so that a general purpose computer cannot hope to give equally high percentage utility to all. The second reason is that the simplest way to increase the arithmetic unit efficiency in any asynchronous case is to slow down the arithmetic unit.

The real goal in improved organization is maximum overall computer performance for minimum cost. One will tend to increase the arithmetic unit speed as long as its percent efficiency is reasonable for a variety of problems. One will stop this process when the overall performance gain no longer matches the increase in hardware and complexity. Thus the arithmetic unit efficiency is a by-product of this design process, not the prime variable.

Speed vs. Concurrent Input-Output Activity

Because of the relative time scales of *I/O* activity and the *CPU* processing speeds, the simulator cannot take account the availability or non-availability of data from *I/O* on the program being run. However, we can observe the effect on the computation of the *I/O* devices operating at different rates simultaneously with computing.

Using the STRETCH control word philosophy, it is possible to have a number of input-output units operating at the same time the Central Processing Unit is running. The Basic Exchange can reach a peak rate of 1 word every 10 microseconds. The high speed disk normally operates at 1 word every 4 microseconds. Since the mechanical devices take priority over the *CPU* in addressing memory, the computation slows down because of memory-busy conflicts.

Fig. 17 shows an example of how internal computing speed is slowed as the *I/O* word rates are varied continuously. At the theoretical "choke off," the *I/O* devices take all the memory cycles available and stop the calculation. Notice that this condition can never arise for any *I/O* rates presently attainable.

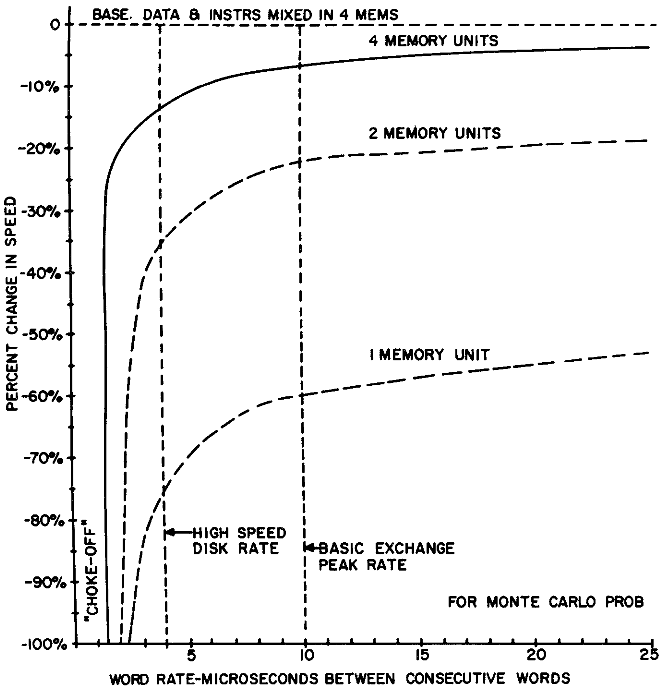


Fig. 17—Internal computing speed. Percentage reduction in speed caused by input-output devices referencing memory at different rates while the calculation is proceeding.

A STRETCH system with only 1 or 2 memory units has less performance than a larger one for three reasons: (1) The top speed of the system is reduced by the loss of memory overlap, (2) it has a larger I/O penalty when I/O is run concurrently with the computation, and (3) the smaller amount of data which can be held in the memory at one time increases the amount of I/O activity needed to do the job. Note, however, that increasing the memory size on a computer of conventional organization only improves the third area.

A Study of Branching on Arithmetic Results in Stretch

One penalty of the non-sequential preparation and execution of instructions used in STRETCH is that if there is a branch in the problem code it spoils the smooth flow of instructions to the indexing arithmetic unit. Any branch in a program will cause some delay, but the most serious ones are the branches on arithmetic results which cannot be detected by the indexing arithmetic unit in advance.

There are two fundamental ways in which branches on arithmetic unit results can be handled by the computer.

(1) The computer can stop the flow of instructions until the arithmetic unit has completed the preceding operation so that the result is known, then fetch the next correct instruction. This places a delay on every AU result branch whether taken or not.

(2) The computer can "guess" which way the branch is going to go before it is taken and proceed with fetching and preparing the instructions along one path with the understanding that if the guess was wrong, these instructions must be discarded and the correct path taken instead.

A detailed series of simulator runs were made to study this situation and to decide which way STRETCH should be designed. Some of the general observations were:

(1) The performance variation in a problem with considerable arithmetic data branching can vary by approximately $\pm$ 15% depending on the way in which the branches are handled.

(2) Holding-up on every branch seems to be less desirable than any of the guessing procedures. Some time is lost whenever a branch is executed rather than proceeding to the next instruction. Unless there is an unusual situation which there is a very large probability that the branch will always be taken, the least time will be lost if one assumes that the branch is *not* taken.

(3) The theoretically highest performance would be obtained if each branch had an extra "guess bit" which would permit the programmer to specify which way he estimates each branch will most likely go. However this would place a considerable extra burden on the programmer for the gains promised. (It also uses up many valuable OP codes.)

(4) It is realized that there is a "feedback" in such decisions because the way in which the machine guesses the branches will influence future programmers to write their codes to take advantage of the speed gain. The result is that the statistics of the future will be biased in favor of the system chosen for the machine, and thus "prove" that it was the right decision.

Acknowledgment

The general idea of "look-ahead" was under consideration by many people in IBM before the authors became involved. What is represented here is a realization of the detailed logic of look-ahead, similar enough to STRETCH for practical simulation purposes. The actual precise detail of the logic as it appears in the STRETCH computer represents, of course, the accomplishments of many individuals in the STRETCH project.

Discussion

M. Rubinoff: What happens to the look-ahead process if a sequence of branch instructions is programmed, such as in the binary selection of one of many subroutines? An example is the selection of the desired piece of a piece-wise function approximation.

Dr. Kolsky: If it is an unconditional branch then it takes a correct path.

Mr. Rubinoff: These are conditional?

Dr. Kolsky: The machine makes the assumption that the branch is *not* taken. If the path is not taken then the branch time is covered up.

M. S. Maxwell (US Naval Weapons Lab.): Discuss maintenance on diagnostic programs to insure proper operation of virtual memory.

Dr. Kolsky: The STRETCH machine has as one of its unusual features a part of the interrupt system capable of recording the status of the machine at the instant the interrupt occurs, so that one gets a "snapshot" of the machine as of that moment. This occurs so you do not have to go back and duplicate the error by running the program over and over again. I think you can see by the way the virtual memory operates that it would be very difficult to duplicate the error again. This feature, whereby a snapshot is made at the time of the error occurs, enables the engineers to go over the records and determine exactly what it was that caused the failure. Of course, the machine has a very elaborate checking mechanism as was described by Erich Bloch in his paper yesterday.

J. Anderson (Burroughs): Is the addressing of STRETCH's main memory sequential within a memory unit or sequential across several memory units?

Dr. Kolsky: The Los Alamos machine has six memories. Two are alternating and the other four are sequential across all four.

R. MacIntyre (Bausch & Lomb): Is the virtual memory addressable in case of a branch?

Dr. Kolsky: No, it is completely unavailable to the programmer. You can see that one would get into some rather tricky logical problems if it could be addressed. We discussed this at length and one gets into a terrible spider web of logical complications when one does that.

A Combined Analog-Digital Differential Analyzer

HAROLD K. SKRAMSTAD†

Introduction

THE ELECTRONIC analog computer, although very useful in solving many problems, and particularly useful in solving dynamic problems described by differential equations, suffers from limitations of accuracy and dynamic range. The digital differential analyzer, although capable of providing any required degree of accuracy or dynamic range, is slow in operation and subject to possible instability of solution due to quantization and the use of finite difference calculus in integration. By combining analog and digital techniques, it is possible to combine the analog advantages of high speed and continuous representation of variables with the digital capability of high precision and large dynamic range.

Dependent variables in such a combined system are represented by two quantities, a digital number, representing the more significant part, and an electrical voltage representing the less significant part. As in the electronic analog computer, the independent variable is always time. Let us consider what form some of the required computer components, such as integrators and multipliers, would take in such a combined system.

Integrator

Assume we wish to obtain the following:

$$y = y_0 + \frac{1}{T}\int_0^t x\,dt \tag{1}$$

where x and y are functions of the time, and T is the "time constant" of the integration. As in the digital differential analyzer, it is necessary that the problem be scaled so that the maximum value of all dependent variables will not exceed unity. Let each of the two dependent variables x and y consist of a digital part and an analog part, denoted by the subscripts D and A, respectively. Thus, we have:

$$x = x_D + x_A \tag{2}$$

$$y = y_D + y_A \tag{3}$$

$$y = y_{0D} + y_{0A} + \frac{1}{T}\int_0^t (x_D + x_A)dt \tag{4}$$

Let us assume time to be divided into discrete equal intervals of duration Δt, and that the digital parts of x and y can change only at times which are integral multiples of Δt. We may then write for the value of y at a time t somewhere in the nth interval:

$$y = y_{0D} + y_{0A} + \frac{1}{T}\left[\sum_{i=1}^{n-1}(x_D)_i\Delta t + (x_D)_n\{t-(n-1)\Delta t\} + \int_0^t x_A dt\right] \tag{5}$$

where $(x_D)_i$ is the value of x_D during the ith interval Δt. Fig. 1 shows a curve of x as an arbitrary function of t. The area under this curve from $t = 0$ to any arbitrary t would equal y^T in equation (5), assuming that the first two terms (y_{0D} and y_{0A}) on the right of equation (5) are *zero*. The first term in the bracketed expression, represented by area 1, is the integral of the digital part of x up to the time $(n - 1)\Delta t$. The second term in the bracket expression, represented by area 2, is the integral of the digital part of x between $(n - 1)\Delta t$ and t. The third term, represented by area 3, is the integral of the analog part of x from $t = 0$ to t.

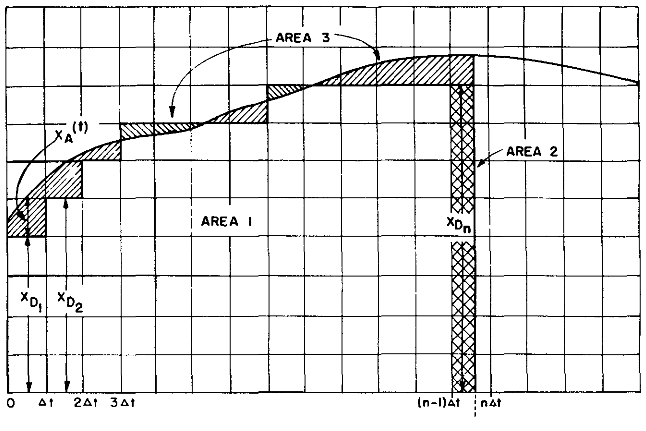


Fig. 1—Diagram of integration method.

Fig. 2 is a block diagram of an integrator unit. It contains an input digital register x_D, a digital register R, two digital-to-analog converters, a conventional analog integrator, a special resettable analog integrator, an analog summer, and a comparator unit. The register y_D shown on the far right of the figure is the input register of the next component to which this unit might be connected in solving a problem. E is the analog reference voltage supplied to the digital-to-analog converters, and α is the digital equivalent of the reference voltage E, chosen for any given problem so as to provide the desired compromise between speed of solution and precision, but subject to the limitation that $|dx/dt|_{\max}$ should not exceed $\alpha/\Delta t$. The number of digits required in the x_D and R registers will depend upon the minimum value of α for which provision is to be made; the minimum value of α will be one in the least significant digit of the x_D register.

† National Bureau of Standards, Washington, D. C.

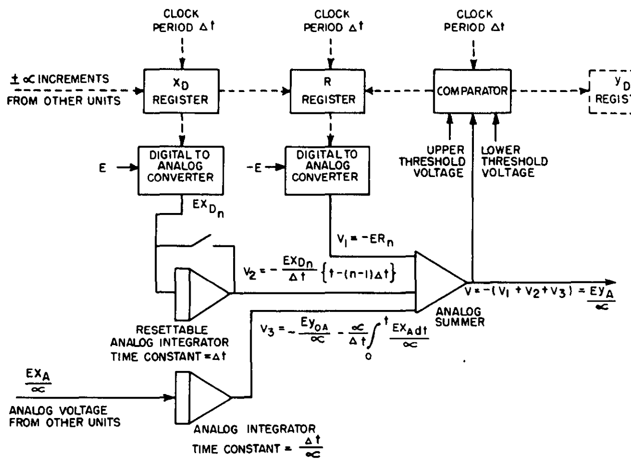

CLOCK PERIOD Δt
CLOCK PERIOD Δt
CLOCK PERIOD Δt
±α INCREMENTS FROM OTHER UNITS
x_D REGISTER
R REGISTER
COMPARATOR
y_D REGISTER
DIGITAL TO ANALOG CONVERTER
DIGITAL TO ANALOG CONVERTER
UPPER THRESHOLD VOLTAGE
LOWER THRESHOLD VOLTAGE
E
-E
Ex_{D_n}
$V_1 = -ER_n$
$V_2 = -\frac{Ex_{Dn}}{\Delta t}\{t-(n-1)\Delta t\}$
$V = -(V_1+V_2+V_3) = \frac{Ey_A}{\alpha}$
ANALOG SUMMER
RESETTABLE ANALOG INTEGRATOR TIME CONSTANT = Δt
$V_3 = -\frac{Ey_{0A}}{\alpha} - \frac{\alpha}{\Delta t}\int_0^t \frac{Ex_A dt}{\alpha}$
$\frac{Ex_A}{\alpha}$
ANALOG VOLTAGE FROM OTHER UNITS
ANALOG INTEGRATOR TIME CONSTANT = $\frac{\Delta t}{\alpha}$

Fig. 2—Integrator unit.

At the beginning of each Δt period, the values x_D and R are sampled and converted to analog voltages which are held constant during the period, unaffected by future changes in x_D or R which occur during the period. The value of x_D is then algebraically added to the R register. The voltage V_1, which represents that portion of the prior summation of $(x_D)_i \Delta t$ which is of analog magnitude is given during the *nth* interval Δt by:

$$V_1 = -ER_n \tag{6}$$

The voltage V_2, which provides integration of the current x_D value within the *nth* interval Δt, and which is reset to *zero* at the end of this interval, is given by:

$$V_2 = \frac{E(x_D)_n\{t - (n-1)\Delta t\}}{\Delta t} dt \tag{7}$$

The voltage V_3, which results from the purely analog integration of the continuously varying analog part of x, is given by:

$$V_3 = \frac{-Ey_{0A}}{\alpha} - \frac{\alpha}{\Delta t}\int_0^t \frac{Ex_A}{\alpha} dt \tag{8}$$

These three voltages are added in the analog summer to give voltage V. The analog part of the output of the integrator is equal to:

$$\frac{Ey_A}{\alpha} = V = -(V_1 + V_2 + V_3) \tag{9}$$

If, at any time during a period Δt, the voltage V at the output of the analog summer exceeds a predetermined upper threshold, this is sensed by the comparator and, during the next Δt interval immediately following the addition of x_D to R, unity is subtracted from the R register and the number α is added to the input register of the following unit (y_D in Fig. 2). Conversely, if the voltage V falls below a predetermined lower threshold, unity is added to the R register and the number α is subtracted from the input register of the following unit.

The time constant T of this integrator unit is equal to $\Delta t/\alpha$, as can be seen from the following. Assume that from time O up to a time t during the *nth* interval Δt, the comparator has caused N subtractions of unity from the R register, and the addition of N to the y_D register. The contents of the R register at this time is:

$$R = \sum_{i=1}^{n-1} (x_D)_i - N \tag{10}$$

and the value of y_D is given by

$$y_D = y_{OD} + N\alpha \tag{11}$$

Substituting equations (6), (7), (8) and (10) into (9) and solving for y_A, we obtain:

$$y_A = \alpha \sum_{i=1}^{n-1} (x_D)_i - N\alpha + \frac{\alpha}{\Delta t}(x_D)_n \{t - (n-1)\Delta t\} + y_{OA} + \frac{\alpha}{\Delta t}\int_o^t x_A dt \tag{12}$$

Adding equations (11) and (12), we have

$$y = y_D + y_A = y_{OD} + y_{OA} + \frac{\alpha}{\Delta t}\left[\sum_{i=1}^{n-1}(x_D)_i \Delta t + (x_D)_n \{t-(n-1)\Delta t\} + \int_o^t x_A dt\right] \tag{13}$$

Equation (13) is seen to be identical to equation (5) if $T = \Delta t/\alpha$.

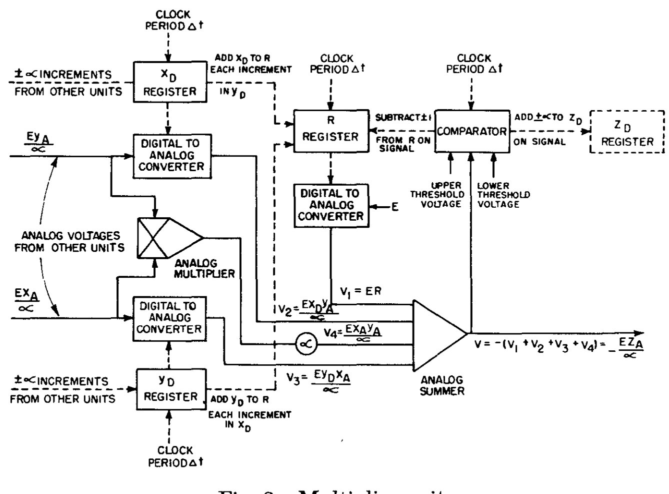

CLOCK PERIOD Δt
ADD x_D TO R EACH INCREMENT IN y_D
CLOCK PERIOD Δt
CLOCK PERIOD Δt
±α INCREMENTS FROM OTHER UNITS
x_D REGISTER
R REGISTER
SUBTRACT ±1 FROM R ON SIGNAL
COMPARATOR
ADD ±α TO z_D ON SIGNAL
z_D REGISTER
$\frac{Ey_A}{\alpha}$
DIGITAL TO ANALOG CONVERTER
DIGITAL TO ANALOG CONVERTER
E
UPPER THRESHOLD VOLTAGE
LOWER THRESHOLD VOLTAGE
ANALOG VOLTAGES FROM OTHER UNITS
ANALOG MULTIPLIER
$V_1 = ER$
$\frac{Ex_A}{\alpha}$
$V_2 = \frac{Ex_D y_A}{\alpha}$
DIGITAL TO ANALOG CONVERTER
α
$V_4 = \frac{Ex_A y_A}{\alpha}$
$V = -(V_1+V_2+V_3+V_4) = -\frac{Ez_A}{\alpha}$
±α INCREMENTS FROM OTHER UNITS
y_D REGISTER
ADD y_D TO R EACH INCREMENT IN x_D
$V_3 = \frac{Ey_D x_A}{\alpha}$
ANALOG SUMMER
CLOCK PERIOD Δt

Fig. 3—Multiplier unit.

Multiplier

Let us now investigate the form taken by a combined analog-digital multiplier. Suppose we wish to obtain the product $z = xy$. Assuming, as before, that each variable consists of a digital part and an analog part, we have:

$$z_D + z_A = x_D y_D + x_A y_D + x_D y_A + x_A y_A \tag{14}$$

where the subscripts D and A signify the digital and analog parts, respectively. Assume, as before, that time is divided into equal intervals of duration Δt, and that the digital parts x_D and y_D can change only at times which are integral multiples of Δt. Fig. 3 is a block diagram of a multiplier unit. It has three digital registers for x_D, y_D, and R, three digital-to-analog converters, an analog summer, an analog multiplier, and a comparator unit. As before, E is the analog reference voltage and α is the digital value of the reference voltage E, chosen for any given problem so as to provide the desired compromise between speed of solution and precision, subject to the condition that neither $|dx/dt|_{\max}$ nor $|dy/dt|_{\max}$ should exceed $\alpha/\Delta t$.

At the beginning of each period Δt, the values of x_D, y_D, and R are sampled and converted to voltages which are held constant during the period. If, during the period, x_D receives an increment (or decrement) α from another unit, y_D is added to (or subtracted from) R; and if y_D receives an increment (or decrement) α from another unit, x_D is added to (or subtracted from) R. If both x_D and y_D change during Δt, the additions to R must either be done serially, using the new x_D or y_D obtained after each addition to R, for the next addition to R, or some other system must be used to obtain a true digital product x_Dy_D. The quantity x_Dy_D can contain twice as many digits as x_D or y_D; the more significant part will be of digital magnitude, and appear in z_D, the input register of the following unit; and the less significant part will be of analog magnitude, and remain in the R register. The reference voltage E is applied to the digital-to-analog converter connected to register R, producing an output voltage $V_1 = ER$; the input voltage y_AE/α is applied to the converter connected to the register x_D producing an output voltage $V_2 = Ex_Dy_A/\alpha$, and the input voltage Ex_A/α is applied to the converter connected to register y_D producing an output voltage $V_3 = Ey_Dx_A/\alpha$. An analog multiplier is connected to the two analog inputs Ey_A/α and Ex_A/α. Its output, attenuated by α, produces a voltage $V_4 = Ex_Ay_A/\alpha$. An analog summer sums the voltages V_1, V_2, V_3, and V_4 to produce a voltage V equal to $-Ez_A/\alpha$.

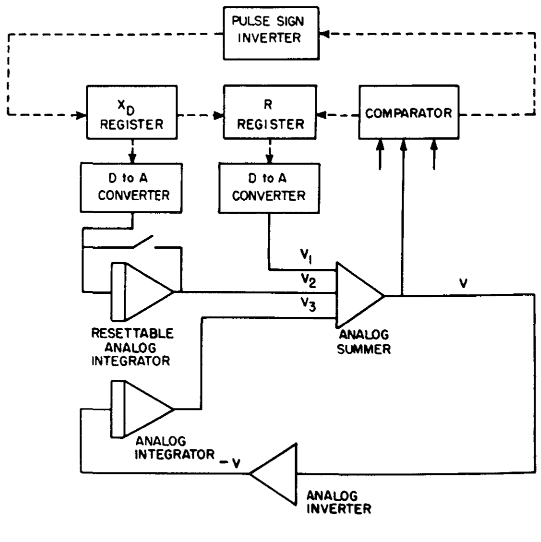


Fig. 4—Integrator used to solve: $\dot{x} = -x$.

During the next Δt after the voltage V exceeds (or falls below) predetermined threshold voltages, unity is subtracted from (or added to) the R register and the number α is added to (or subtracted from) the input register of the following unit (z_D in Fig. 3).

It should be noted that for small values of α the analog multiplier may be omitted, producing a maximum error of α. For values of α less than the resolution of the analog components, say .001 or less, this error is negligible.

If one of the factors to be multiplied is a constant, the equipment required is simplified, since only one digital register needs to be capable of accepting increments, and the R register receives additions from only one other register. If the factor is a purely digital quantity, one of the digital-to-analog converters and the analog multiplier may be omitted.

Summing

Summing may most easily be done by permitting each integrator or multiplier unit to accept digital increments and analog voltages from several units. For example, in the integrator of Fig. 2, if the $\pm\alpha$ increments from a number of other units are connected to its x_D register, and if the sum of the increments put out by these units is $N\alpha$ during any period, the increment in x_D would equal $N\alpha$. The analog outputs from the other units would each be connected to an input summing resistor in the analog integrator.

In the case of the multiplier unit, if the $\pm\alpha$ increments from a number of other units are connected to its x register, and if the sum of the increments put out by these units is $N\alpha$, y_D would be summed into the R register N times. The analog outputs from the other units would be connected to inputs of an analog summer whose output would form the analog input x_AE/α to the multiplier.

Solution of Simple Differential Equations

Examples of the operation of this proposed combined system can be seen from following in detail how some simple differential equations would be solved.

Let us consider first the differential equation

$$\dot{x} = -x \qquad (15)$$

Fig. 4 shows a block diagram of how a single integrator unit with output fed back into its input would solve this equation. The voltages V_1, V_2, V_3, and V are those defined in equations (6) to (9).

Differentiating equations (6) to (9), we obtain, since $\dot{V}_1 = 0$

$$\dot{V} = \frac{Ex_D}{\Delta t} - \dot{V}_3 \qquad (16)$$

From the interconnections of Fig. 4, the following expression must hold:

$$\dot{V}_3 = \frac{\alpha}{\Delta t} V \qquad (17)$$

V will then be given by the following differential equation:

$$\dot{V} + \frac{\alpha}{\Delta t} V = \frac{Ex_D}{\Delta t} \qquad (18)$$

Subject to the initial conditions that at $t = 0$, $x_D = x_{OD}$, and $-V = Ex_{OA}/\alpha$ the differential equation will have the following solution:

$$V = \frac{Ex_D}{\alpha} - \frac{E(x_{OD} + x_{OA})}{\alpha} e^{-\frac{\alpha}{\Delta t} t} : \qquad (19)$$

and x will be given by

$$x = x_D - \frac{\alpha V}{E} = (x_{OD} + x_{OA}) e^{-\frac{\alpha}{\Delta t} t} : \qquad (20)$$

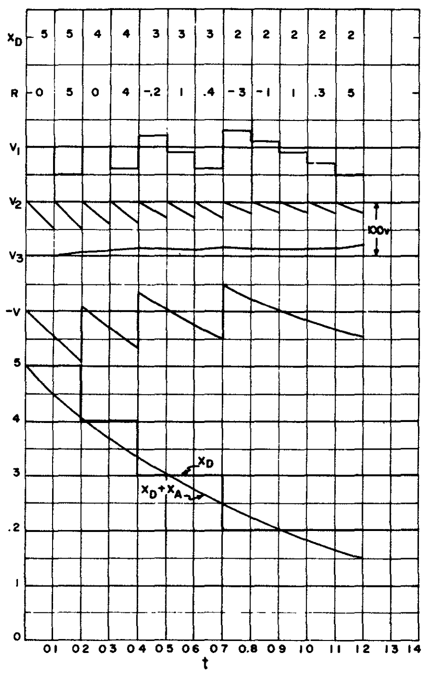


Fig. 5—Time history of solution of $\dot{x} = -x$.

Fig. 5 shows in detail the quantities that would appear in the x_D and R registers, the voltages V_1, V_2, V_3, and V as functions of the time, using the following parameters:

$E = 100$ volts $\quad \alpha = 0.1 \quad \Delta t = 0.1$ sec

$x_{OD} = 0.5 \quad x_{OA} = 0$

The threshold values on V for initiating decrements to the x_D register and subtracting one from the R register are plus and minus 50 volts. In this example, since $\alpha = 0.1$, the precision obtained in solving this equation should be 10 times that which would be obtained in solving this on a purely analog computer with analog components of equivalent precision to those of the combined system.

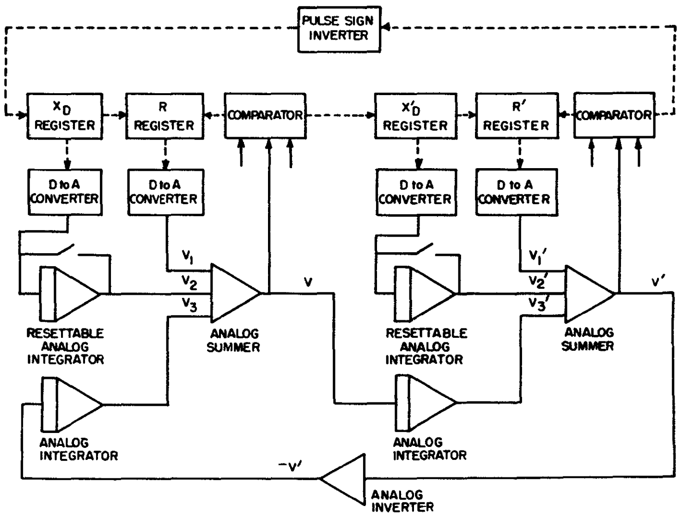


Fig. 6—Two integrators interconnected to solve: $\dot{x}' = x$, $\dot{x} = -x'$.

Another example of the operation of the proposed combined system is the solution of the following pair of simple differential equations:

$$\dot{x}' = x \qquad (21)$$

$$\dot{x} = -x' \qquad (22)$$

Fig. 6 shows a block diagram of how two integrator units would be interconnected to solve these equations. The voltages V_1, V_2, V_3, V are those defined by equations (6) to (9), and occur in the integrator containing x; the primed voltages are those which occur in the integrator containing x'. Differentiating equations (6) to (9), we have, since $\dot{V}_1 = \dot{V}'_1 = 0$:

$$\dot{V} = \frac{x_D E}{\Delta t} - \dot{V}_3 \qquad (23)$$

$$\dot{V}' = \frac{x_D E}{\Delta t} - \dot{V}'_3 \qquad (24)$$

From the interconnections of Fig. 6, the following expressions are seen to hold:

$$\dot{V}_3 = + \frac{\alpha}{\Delta t} V' \qquad (25)$$

$$\dot{V}'_3 = - \frac{\alpha}{\Delta t} V \qquad (26)$$

The quantities V and V' therefore will be given by the following differential equations:

$$\dot{V} = \frac{x_D E}{\Delta t} - \frac{\alpha}{\Delta t} V' \qquad (27)$$

$$\dot{V}' = \frac{x'_D E}{\Delta t} + \frac{\alpha}{\Delta t} V \qquad (28)$$

Subject to the condition that at $t = 0$, $x_D = x_{OD}$,

$x'_D = x'_{0D}$, $V = x'_{0A}E/\alpha$, $-V' = x_{0A}E/\alpha$; these differential equations will have the following solution:

$$V = -\frac{E}{\alpha}x'_D + \frac{E}{\alpha}(x'_{OD} + x'_{OA})\cos\frac{\alpha t}{\Delta t} + \frac{E}{\alpha}(x_{OD} + x_{OA})\sin\frac{\alpha t}{\Delta t} \quad (29)$$

$$V' = \frac{E}{\alpha}x_D - \frac{E}{\alpha}(x_{OD} + x_{OA})\cos\frac{\alpha t}{\Delta t} + \frac{E}{\alpha}(x'_{OD} + x'_{OA})\sin\frac{\alpha t}{\Delta t} \quad (30)$$

and the quantities x and x' will be given by the following expressions:

$$x = x_D - \frac{\alpha}{E}V' = (x_{OD} + x_{OA})\cos\frac{\alpha t}{\Delta t} - (x'_{OD} + x'_{OA})\sin\frac{\alpha t}{\Delta t} \quad (31)$$

$$x' = x'_D + \frac{\alpha}{E}V = (x'_{OD} + x'_{OA})\cos\frac{ot}{\Delta t} + (x_{OD} + x_{OA})\sin\frac{\alpha t}{\Delta t} \quad (32)$$

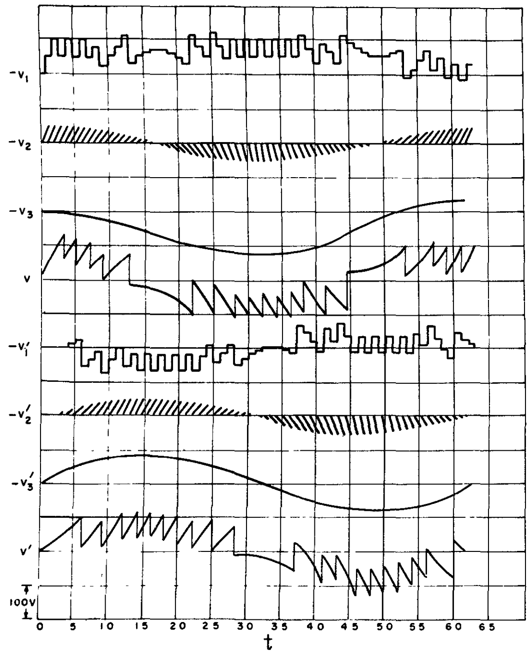


Fig. 7—Time history of solution of $\dot{x}' = x$, $\dot{x} = -x'$.

Fig. 7 shows in detail the voltages which would appear as a function of the time when solving the above equation, using the following values of the various parameters:

$E = 100$ volts $\alpha = 0.1$ $\Delta t = 0.1$ sec
$x_{OD} = 0.5$ $x_{OA} = 0$ $x'_{OD} = 0$ $x'_{OA} = 0$

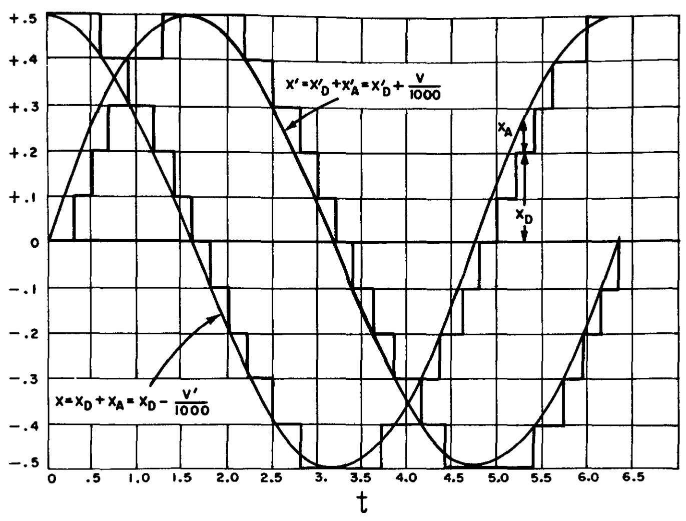


Fig. 8—Plot of x and x' as functions of time.

As in the previous example, the threshold values on V and V' for initiating positive or negative increments to the x'_D or the x_D register, are plus and minus 50 volts, respectively. Fig. 8 shows the results of combining the digital and analog portions. The stepped curves are the digital part only; the smooth curves are the sum of the digital and analog parts. As in the first example, the precision attained should be 10 times that which would be obtained solving these equations on an analog computer with analog components of precision equivalent to those used in the combined system.

Component Requirements

Now a few words on the characteristics needed in the hardware to realize the above components. For a maximum speed-precision product to be obtained, the value of Δt should be as small as possible consistent with hardware limitations. The smaller Δt is made, however, the greater the bandwidth required in the operational amplifiers, since the analog voltages must be capable of a full scale voltage excursion E during the time Δt. The digital-to-analog converters should be capable of holding their output values constant during each period of Δt and equal to its value at the beginning of the period, and then rapidly changing to its new value at the beginning of the next period. In the case of the integrator, the necessary additions of x_D to R, subtractions of ± 1 from R, and incrementing the input registers of following units and, in the case of the multiplier, the additions of x_D and y_D to R, subtractions of ± 1 from R, and incrementing the input registers of following units must all be completed within the period Δt. In the integrator unit, the resettable analog integrator might well consist of two analog integrators with switching between them so that each is used to integrate during alternate Δt periods while the other is being reset.

Discussion and Conclusions

There are a number of problems associated with this combined system that have not yet been investigated. One of these, which this computer has in common with both analog and digital differential analyzers, is that of proper scaling so as to prevent overflowing of the digital registers or saturation of the analog integrators. Another is the choice of the threshold voltages for the comparator units. It is possible that the best value for both upper and lower threshold voltages should be *zero* — requiring a positive or negative digital increment to be sent to the next unit each Δt, depending upon the sign of the voltage V. The problem of scaling and choice of thresholds are interrelated. and it should be possible to exercise some control over the overloading of analog integrators by proper choice of the threshold voltages.

The number of digits to be carried in the digital registers depends, of course, on the minimum value of α for which provision is to be made. In general, the R registers should contain one more binary digit than the other registers to prevent overflow under conditions where a large x_D of the same sign as R is added to R.

For any particular problem, the value of α to be chosen depends upon the particular compromise between precision and speed of solution desired. As a simple illustration, consider integration of the function $x = A \sin \omega t$, and assume Δt equals .001 second, $\alpha = .001$, and A is 1. Since the maximum time rate of change of this function $A\omega$ should not exceed $\alpha/\Delta t$, the highest frequency representable at full-scale amplitude would be $\omega = \alpha/A\Delta t = 1$ radian per second, and the precision (assuming an analog resolution of .001) would be one part in one million. If we chose $\alpha = .1$, the highest frequency representable at full scale amplitude would be 100 radians per second, and the precision would be one part in ten thousand.

The combined analog-digital differential analyzer of the type described shows promise of overcoming some of the limitations of present designs of differential analyzers which use purely analog or purely digital techniques. Also if analog components of sufficient bandwidth are available, the precision-speed product of this combined system should be greater than that possible with a parallel digital differential analyzer having equal length digital registers and equal iteration rate, by a factor dependent upon the resolution of the analog components — perhaps a factor of one thousand.

The greatest usefulness of the proposed system is believed to be on problems where the precision required is of the order of 10 to 100 times that obtainable by analog methods, yet requiring the real-time speed of analog methods. In this case, integrators and multipliers of the combined system would contain short digital registers and only moderate requirements would be put on the speed of switching circuits and the bandwidth of the analog components.

Work is under way at the National Bureau of Standards to construct breadboards of two integrator and two multiplier units, each capable of receiving inputs from two other units. The digital registers and digital-to-analog converters are being constructed from transistorized digital building block packages developed at NBS, and the analog components from commercially available wide band amplifiers. These units are planned to have 7 bit plus sign input registers, 8 bit plus sign "R" registers, an analog reference voltage of 10 volts, and operating with a Δt of one millisecond.

Acknowledgment

The author wishes to acknowledge the assistance of the Navy Bureau of Aeronautics in supporting the construction of the breadboard units for evaluating the proposed computer system.

Discussion

M. Rubinoff: I note there is no analog to digital conversion in the drawings. Is this true for the entire system?

Dr. Skramstad: This is true of the entire system. At no time do we have to convert an analog voltage to a digital quantity. Going from digital to analog is the easy way to go.

Mr. Rubinoff: I believe you said something about — I hope I am quoting you correctly — this system has 10 times the precision of an analog computer. What does this mean, 10 times as many digits? How does this compare with the DDA with greater precision?

Dr. Skramstad: The factor 10 applies only to the example shown; the actual factor depends upon the value chosen for α. In the example given, assume that 100 volts could be divided to a resolution of a part in a thousand, say, so that you might say that a tenth of a volt would represent, in a way, the least significant digit. Now in the examples I showed, the analog voltages would go through this 100 volts 10 times as the variable goes from zero up to unity. The effective resolution would thus be a part in 10,000.

D. Cohen (Airborne Inst. Lab.): Does the comparator have to be more accurate than in the pure analog computer to achieve the 10 times resolution?

Dr. Skramstad: The comparator does not have to be precise; it only detects that a theshold voltage has been exceeded sometime during the interval.

Mr. Labin (AERO, France): Would the cooperation of a digital block to the differential analyzer allow treatment of non-linear equations, at least approximately?

Dr. Skramstad: I think it would. I have not had time to go into what various non-linear components would be like in this system. I believe non-linear components could be devised that would work with the system.

N. Nesenoff (Republic Aviation): Could you indicate the percentage of equipment saved by the combined analog-digital system when compared to the equipment required by a purely digital system of equal accuracy?

Dr. Skramstad: This would be a hard one to put a figure on. I think the big advantage here is that you still have what you might call real time speed capability for real time simulation of physical systems. As you all know, analog computers are very useful in real time simulation of dynamic missile systems and aircraft systems, but often there are a number of variables that have to be carried to higher precision than the analog is capable. This system would allow you to carry those variables in the combined systems where additional precision is required.

Mr. Nesenoff: Did you ever build a computer of this type?

Dr. Skramstad: No, this is just an idea. We are in the process of building up a breadboard to try out the idea. We are planning a device carrying 7 bits plus sign in the X register, 8 bits plus sign in the R register, and voltage of 10 volts. I hope to be able to report how this works at a later date.

F. Verzuh (*MIT*): Comment on the solution of general integration: $W = I/K \int udv$ where v = time using your device. (*Chairman*: I think he is getting back to the fact that digital differential analysers will work with any variable whereas the analog computer insists that the independent variable in the integration is time.)

Dr. Skramstad: This is the limitation of this system, just as it is a limitation of the electronic analog computer. You have to devise the program in such a way as to associate time as the end variable.

Mr. Rubinoff: So it loses the advance that the digital has?

Dr. Skramstad: That is right. The multiplying device, however, is working only in dependent variables. For problems not involving integration, you are free of this limitation.

D. A. Bourne (*IBM*): Please summarize what you consider the most important advantages of a hybrid computer.

Dr. Skramstad: Well, I think I just mentioned that in the answer to another question. I think the greatest usefulness would be in simulation of physical systems where the precision of the analog computer is not quite great enough, and thus by the addition of short digital registers and digital components to an analog system, it would be able to handle these cases.

E. M. Ginsberg (*Burroughs Corp.*): Please comment on the accumulation and propagation of errors due to tolerances in the two systems and incurred by overlapping at different precision points.

Dr. Skramstad: Any errors due to drift in amplifiers in the analog portion can affect only the precision of the analog part, and these errors would be decreased in the overall solution by the factor Alpha.

Mr. Ginsberg: Please comment on any effect in slew time with this system as compared to a DDA.

Mr. Rubinoff: Slew time is the rate at which numbers can be changed in the differential analyzer, which limits the simple DDA.

Dr. Skramstad: I think some of the same techniques used in DDA would apply directly to this computer. I would think it would be a similar problem to that of the DDA.

K. Enslein (*Brooks Research*): How about the possibility of two different Deltas, one coarse and one fine?

Dr. Skramstad: This is something I haven't considered as yet. It might be worth looking into.

M. Tayyabkhan (*Union Carbide*): Would the use of floating point arithmetic in the digital registers solve some of the scaling problems of analog computers?

Dr. Skramstad: Well, now there is a possibility here. Again, this has not been investigated so that I cannot give a definite answer to that one.

The System Organization of MOBIDIC B

STANLEY K. CHAO†

MOBIDIC B is an all transistorized, militarized computer mounted in a standard Army trailer. It is a general-purpose, parallel, binary, synchronous, fixed point, and duplexed data processing system.

It contains two basic processors, identical in characteristics and internally tied to the same system transfer bus. Both processors share a common set of input-output devices and each processor is capable of operating on an independent program without interference. They are also capable of duplex operations, allowing either processor to monitor and exert control over the other.

In addition to the 8192-word high speed Core Memory in each processor, there exists a 50 million-bit Mass Memory. This memory is treated as an input-output device, addressable by in-out instructions. A Data Retrieval Unit is incorporated to facilitate data searching from the Magnetic Tape and the Mass Memory. The control console is also duplexed, containing two independent and identical panels, one for each processor.

Introduction

There are a number of large-scale, general-purpose, mobile digital computers being developed at Sylvania[1]. They are known as the MOBIDIC family of computers. MOBIDIC C and D are identical in internal design to MOBIDIC A[2]. They differ only in peripheral equipment.

MOBIDIC B is the second member in the MOBIDIC family. It will be used primarily in the field to meet the Army's data processing requirements, and is to be installed in a standard 30 foot Army trailer. MOBIDIC B contains all the instructions utilized in MOBIDIC A and also an additional 12 new instructions. To minimize equipment, some of the instructions have been made subroutines which are initiated automatically.

MOBIDIC B is a duplexed data processing system. It contains two identical central processors connected to a common system transfer bus. Each central processor has an individual real time system which provides direct communication with external FIELDATA equipment. Each of the two central processors is a general-purpose computer. They can use any of the input-output devices available and either processor can run separate programs without interference from the other. In addition, the two processors can operate together and communicate with each other through an in-out device such as a magnetic tape. Either processor can use the other's core memory through an input-output device. It is also possible for either processor to monitor and exert control on the other through the direct connections between them.

Special Requirements

There are 4 major requirements of a special nature imposed on this computer. The first is that MOBIDIC B is to be a duplexed data processing system. The second requirement is that it must be compatible with other MOBIDIC computers. Compatibility is sought not only through instruction type and word format, but also through the interchange ability of input-output devices and the physical element-cards and packages. Minimum equipment is another design criterion. This is achieved by having all full-length registers simulated in the core memory and also by mechanizing some of the infrequently used instructions through automatic subroutines. Only two full-length working registers are used within each processor. Data retrieval capability is the fourth of the special requirements. Data stored on either the magnetic tape or the mass memory must be retrieved at full device speed. This continuous and speedy retrieval ability necessitated the incorporation of a data retrieval unit together with the addition of some special instructions.

Instructions and Word Format

Instructions

There are 64 instructions in MOBIDIC B (Table 1). They are classified into 3 categories, namely, directly-mechanized instructions, subroutine instructions, and special instructions. There are a total of 40 directly mechanized instructions. In most cases, the execution time has a variation of 2 μs. The instruction LGM, for example, may sometimes be executed in 36 μs instead of the 38 μs shown in Table 1. The variation is due to the fact that the system bus is time-shared by both processors. The system bus may or may not be available to the processor which is executing the instruction that requires access to it at that particular instant. The times given in Table 1

† Sylvania Electric Products, Inc., Needham, Massachusetts.

[1]J. Terzian, "MOBIDIC Computer Series," THE SYLVANIA TECHNOLOGIST, Vol. XII, No. 3, July, 1959, pp. 58-64.

[2]J. Tersian, "System Organization of MOBIDIC A," presented at the 1957 WESCON Convention, August 20, 1957, San Francisco, California.

TABLE I

MOBIDIC B INSTRUCTION REPERTOIRE

Op Code	Abbr.	Time (μs)	Type			Instr. Description
			Direct	Sub	Special	
00	HLT	24	D			Halt
01	RPT	—		R		Repeat
02	LGM	38	D			Logical multiply
03	LGA	38	D			Logical add
04	LGN	36	D			Logical negation
05	SEN	28	D			Sense
06	SNS	28	D			Sense and set
07	SNR	28	D			Sense and reset
10	CLA	36	D			Clear and add
11	CAM	36	D			Clear and add magnitude
12	ADD	44	D			Add
13	ADM	44	D			Add magnitude
14	CLS	36	D			Clear and subtract
15	CSM	36	D			Clear and subtract magnitude
16	SUB	44	D			Subtract
17	SMB	44	D			Subtract magnitude
20	MLY	88–774	D			Multiply
21	MLR	88–786	D			Multiply and round
22	DVD	—		R		Divide
23	DVL	—		R		Divide long
24	ADB	—		R		Add beta
25	SBB	—		R		Subtract beta
26	BSPL	—			S	MOBIDIC B special
27	BSPL	—			S	MOBIDIC B special
30	SHL	30–66	D			Shift left
31	SLL	46–118	D			Shift left long
32	SHR	30–66	D			Shift right
33	SRL	—		R		Shift right long
34	CYS	30–66	D			Cycle short
35	CYL	—		R		Cycle long
36	SLA	30–66	D			Shift left and logical add
37	NRM	—		R		Normalize
40	TRU	28	D			Transfer unconditional
41	TRL	—		R		Transfer & load PCS
42	TRS	—		R		Transfer to PCS
43	TRX	—		R		Transfer on index
44	TRP	26	D			Transfer on positive
45	TRZ	26	D			Transfer on zero
46	TRN	26	D			Transfer on negative
47	TRC	—		R		Compare
50	STR	34	D			Store
51	LOD	36	D			Load
52	MOV	36	D			Move
53	LDX	—		R		Load index
54	RPA	—		R		Replace address
55	MSK	—		R		Mask
56	BSPL	—			S	MOBIDIC B special
57	TRY	38	D			Transfer on index B
60	BSPL	—			S	MOBIDIC B special
61	BSPL	—			S	MOBIDIC B special
62	BSPL	—			S	MOBIDIC B special
63	BSPL	—			S	MOBIDIC B special
64	BSPL	—			S	MOBIDIC B special
65	BSPL	—			S	MOBIDIC B special
66	SKP	30	D			Skip
67	BSP	30	D			Back space
70	RAN	30	D			Read alphanumeric
71	RRV	30	D			Read reverse
72	ROK	30	D			Read octal
73	SCH	30	D			Search
74	WAN	30	D			Write alphanumeric
75	WWA	30	D			Rewrite alphanumeric
76	WOK	30	D			Write octal
77	RWD	30	D			Rewind

are maximum.

There are 15 subroutine instructions. These instructions are performed through subroutine programs, which must be stored in memory and executed automatically. As such, all instructions used in the program to execute the sub-routine instruction must be of the mechanized type. Nine of the MOBIDIC B OP codes are unassigned. They are available to perform any special subroutine operation that may be required in a particular application. Whenever one of these "special" instructions is decoded, operation is automatically transferred to a unique location from which the program is re-routed to the desired subroutines.

According to their logical functions, the 64 instructions can be classified as follows: 16 arithmetic operations, 17 sequencing and indexing, 10 input-output; 12 editing and data handling; and 9 special purpose instructions.

Word-Format

The length of the MOBIDIC B word is 38 bits, the length used in other MOBIDIC computers. The word format is illustrated in Fig. 1. Numerical data is represented by a fixed point, magnitude and sign conventions. Magnitude is registered in bits 1 to 36. The binary point is understood to be placed between bits 36 and 37. Bit 37 is used for sign of a number stored and bit 38 is used as a parity check bit. Alphanumeric data is represented in the same manner as numeric data, except that the sign bit is eliminated and the 36 bits are grouped into 6 alphanumeric characters. Since MOBIDIC uses a weighted code, alphanumeric data can be sorted without conversion to binary form by direct use of the logical and arithmetic operations.

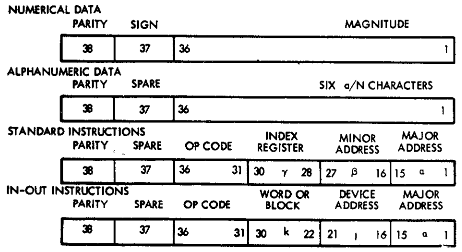


Fig. 1—MOBIDIC B word format.

A standard instruction word is divided into 6 parts:

(1) Major Address (α): Bits 1 to 12 specify a memory address while bits 13 to 15 specify which memory will be used. Since many of the internal MOBIDIC B registers are addressable, one of the eight configurations for bits 13 to 15 represents internal register addresses. The actual register addresses in these cases are specified by bits 1 to 5.

(2) Minor Address (β): Bits 16 through 27, the β bits, have several uses, depending on the particular instruction being performed. They may be loaded into or added to the contents of an index register. The β bits, either alone

or in combination with γ may also be used to specify a second address.

(3) Index Register (γ): Bits 28 to 30, γ bits, are used primarily for indexing. They specify which, if any, of the Index Registers are to be used with the instruction. For some instructions, γ is used as part of a second address.

(4) OP Code: Bits 31 to 36 designate the instruction to be performed.

(5) Spare: Bit 37 is a spare.

(6) Parity: Bit 38 is the parity bit.

The format for input-output instructions is identical to that of standard instructions except for the assignments made to bits 16 to 30, the j and k portions of an in-out instruction. The j bits, 16 to 21, are used to specify the particular input-out device addressed. Sixty-three input-output devices may be handled in this manner. The k bits, 22 to 30, are used to specify the number of words or blocks to be processed.

Overall System Description

The block diagram of the MOBIDIC B system is given in Fig. 2. In it are shown: two identical computers (processor 1 and processor 2), two in-out converters, a data retrieval unit, a mass memory unit, two real time systems, and a family of in-out devices. The two processors are fed by a common system clock to facilitate system synchronization. They are connected to a comon system transfer bus and also share the same in-out system. The processors can operate independently as separate computers or can communicate with each other through magnetic tape. It is possible for either of the two processors to give a set of instructions which will read from or write into the memory of the other processor. It is also possible for one processor to monitor the other through the signal lines connecting them directly. Furthermore it is possible for one processor to control and give commands to the other processor.

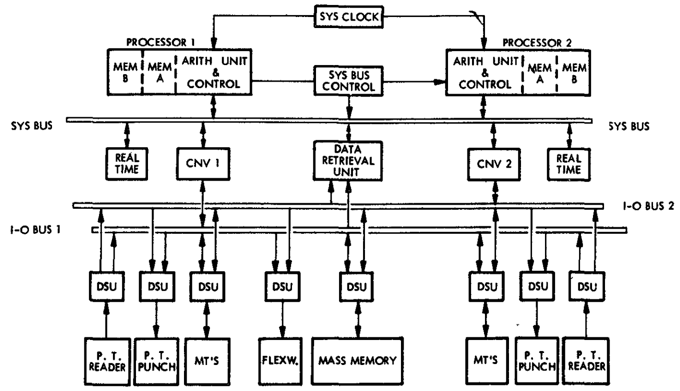


Fig. 2—MOBIDIC B system block diagram.

The converters serve as format control and synchronizing buffers. They are capable of handling all devices including the mass memory unit. The converters are assigned to either of the two processors on a first-come-first-serve basis. Since the two processors are connected to a common system bus, it is entirely possible that both may want to get on the system bus at the same instant. To avert this uncertainty, the system bus control circuit is used to continuously assign the system bus to the processors alternately. Each is given a 2 μs period to use it. Therefore, a waiting period of 2 μs may be required at an arbitrary time. The danger of both processors trying to get to the same converter is also avoided by the same control circuit.

In addition to the standard family of MOBIDIC in-out devices, the MOBIDIC B system includes a 50-million-bit memory which is also treated as an in-out device. Data transfer to and from the mass memory will be handled by the converter through the in-out bus. The data in the mass memory is arranged in blocks, separated by block start and block end marks. The block number on each track is addressable. Both the track and block addresses must be loaded into the mass memory control unit prior to giving the write or read instruction. These addresses are sent out from the processor through the converter, similar to the manner by which the data is transferred.

The data retrieval unit is designed to assist the open format search program. A retrieval program examines the specified fields to determine whether or not the search criteria are satisfied. Closed format search can be accomplished entirely by programming and does not require any auxiliary equipment.

The two real time systems enable the processors to communicate with other data processing equipment external to the MOBIDIC B system. Each has a fixed assignment to serve one definite processor and is not accessible to the other processor.

On the other hand, the balance of the entire input-output system does not have a fixed assignment. Operating on a first-come-first-serve basis, it is entirely possible for one processor to automatically monopolize the use of both converters, the data retrieval unit and a complement of devices. Without a converter the other processor cannot reach any device even if it is available. Under such a circumstance, the other processor would have no choice but to wait for a converter to become available. In some special cases, the other processor may have just received, through its own real time system, an urgent request which requires data processing through the service of a converter. Upon receiving such a request, it is possible for the other processor to shorten the in-out operation of the first processor, making the converters available.

Brief Description of Processor

Since minimum equipment is a major design requirement, most full length registers normally existing in other MOBIDIC Computers are stored as locations in the memory. These reserved memory locations are referred to as simulated registers. Simulated registers include the accumulator, the Q-register, the B-register, the program counter, the program counter store, and seven index registers. They can be addressed in exactly the same way as their counterparts in other MOBIDIC computers. As shown in Fig. 3, only two full-length physical registers are used in each MOBIDIC B processor, the memory register (MR) and the control Register (CR). These registers and other essential parts in the processor are described as follows:

Timer The timer, containing three counters, receives pulses from the system clock. The processor executes instructions by proceeding through a sequence of events. Certain events, such as memory operations, occur so frequently that a separate counter TM is used to control these operations. The execution of instructions requires several memory operations. Counter TI indicates which of the several memory operations is currently in progress. Finally, if an input-output access to the central processor is required, the instruction execution must be interrupted and a new sequence of events must start. A third counter TB controls these input-output processing operations.

Core Money Each processor of the MOBIDIC B system is provided with two 4096-word core memories with a read-write cycle of 8 μs. It can be readily expanded to 4 memories per processor when desired. It can be ultimately expanded to 7 memories, totaling 28,384 words.

Memory Register The MR is directly connected to the memory. It is a 38-bit register and is used as the memory in-out register. The MR is also used as an arithmetic register during execution of the instructions.

Control Register The CR serves primarily as a 37-bit arithmetic register corresponding to the accumulator in MOBIDIC. In addition, the first 15 bits of CR are also used as the memory address register during initial access to the high-speed memory.

Decoder Register This is a 6-bit register used to store the instruction while it is being executed. Its output interprets the instruction stored and energizes appropriate control lines to initiate execution of the instruction specified.

Control The control unit contains the logical circuits to control all of the detailed operations of the computer.

Special Address Control The special address control unit contains the decoders and control circuits to address the core memory locations which are reserved for special registers of the processor and the data retrieval unit. Among the registers specified by the special address control unit are most of the simulated registers. The contents of these registers can be transferred to the MR and then to the CR whenever it is requested.

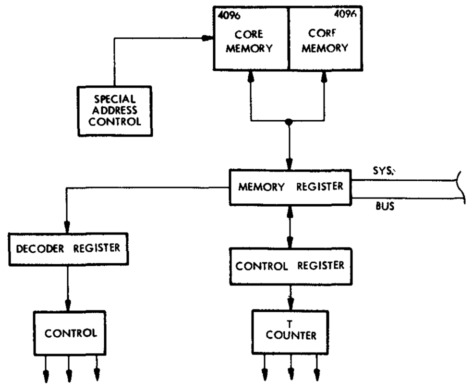


Fig. 3—MOBIDIC B processor.

T Counter The T Counter is a 7-stage counter used in both shifting and multiplying operations.

System Clock The system clock provides standard p and t pulses spaced one microsecond apart to the entire system. There is a separate clock for each processor. This makes the processors identical. One system clock may be used to control the entire system operation when the processors are working together, depending upon which processor is in full control of the program.

System Bus Control The system bus control circuits regulate and direct the flow of traffic between the two in-out converters, the data retrieval unit, and the two processors. These control circuits give either processor access to the system bus.

Input-Output System

A more detailed description of the various components in the In-Out system will now be given. Reference should be made to Fig. 2.

In-Out Converter There are two in-out converters in MOBIDIC B. They are used as buffers between the input-output devices and the central processors. They assemble data coming in from a device and put it into MOBIDIC word format before transferring it to the central processor. Conversely when data is to be sent out to the device from the central processor, the converters decompose the standard MOBIDIC word and reassemble it into the proper format for the particular device which is to receive it. In addition, converters also have the function of synchronizing the operation of the devices with that of the central processor. In this way, the information transfer between the converter and the device can take place quite independently from the operation of the central processor. Internal computation is only interrupted during access to the memory.

Data transfers between converters and processor memories are handled on a "busy-bit" basis over the system bus. As soon as a converter is selected the processor will transfer the entire in-out instruction to the converter. The processor subsequently goes on to execute the next instruction and exerts no further control over the converter. The converter, taking upon itself to execute the instruction it has just received, proceeds to communicate with the device addressed and sends out or receives data from the device. When the converter is ready to send in or to receive another word from the processor, it will signal the processor by raising a busy-bit. Detecting a busy-bit, the processor will interrupt its operation and take care of the converter.

Real-Time System There are two identical real time systems. Each system consists of an input register, an output register and an input address register. Each real time system is assigned to a processor. These real time systems can be used to provide the communication link between the two central processors of a single MOBIDIC B system, between two MOBIDIC B systems, between a MOBIDIC B and MOBIDIC A computer or with other FIELDATA computers and communication equipment. In all cases except the last, the real time output register in one real time system can be directly connected to the real time input register of the other real time system. In the FIELDATA application, a buffer unit may be required between the real time system and the external equipment. For example, a Kineplex is required when the teletype communication equipment is connected to MOBIDIC B.

Data Retrieval Unit The DRU is a special unit designed to assist the data retrieval program from the storage fiiles. It is connected to the system bus as well as the in-out bus. It will examine all the data being transferred from the magnetic tape (or mass memory) to the converter. After extracting the desired portion, the data is then sent to the core memory for further processing.

Mass Memory The mass memory is needed to provide an exceptionally large data storage capability. It consists of 8 magnetically coated discs, giving a total of 16 usable disc sides. There are 4,096 tracks on which the data can be stored. Storing is done in a serial-serial manner which can be continuous from one track to the next. Track switching is done automatically. There are two magnetically engraved clock tracks, one at 150 KC bit rate and the other at 225 KC bit rate. The maximum random access time of the mass memory is less than 0.25 seconds.

Magnetic Tapes A total of 8 magnetic tapes are currently provided in MOBIDIC B. They could be either the commercial FR-300 type or the militarized type. They have 8 channels which incorporate a parity error detection channel. The nominal tape speed is 150 inches per second (reversible) with approximate start and stop time of 1.5 ms. The nominal character rate is 45 KC.

Flexowriter The Flexowriter is a special electric typewriter that operates at a speed of 10 characters per second. It can be used on- or off-line or as an output device for producing hard copy or punched paper tape.

Paper Tape Reader There are two photoelectric paper tape readers, one being a 5-hole and the other an 8-hole reader. These input devices have a nominal reading rate of 270 characters per second.

Paper Tape Punch The two paper tape punches include a 5-hole punch and an 8-hole punch; both are directly adaptable to the reader. Both types prepare punched paper tape at a nominal character rate of 100 per second.

Duplexing Capabilities

The two MOBIDIC B processors are tied to a common system bus and share a common set of in-out equipment. It is beyond the scope of this paper to give a detailed description of the entire duplexing capabilities existing in MOBIDIC B. A separate technical paper is to be published treating this subject in greater detail. Briefly, there exists a set of control lines connecting the two processors directly. Through these lines the operating status of one processor can be monitored by the other processor. Through these control lines also, one processor can exert control and give command to the other processor in a limited manner. In particular, one processor can prevent the other from coming to a complete halt condition, thus keeping the other processor in a state of readiness to accept information which may be transferred into it from the first processor. Moreover, one processor can restart the other after that processor has completed a program. In addition, one processor can give an input-output instruction which is to be executed by the other processor. For example, one processor may give a write instruction to have information contained by the other processor written out onto a magnetic tape. That same controlling processor can subsequently give a read instruction to have the same information read from the magnetic tape back into its own memory for immediate use. Thus one processor can effectively make use of the data stored in the other processor's memory. Conversely, one processor could give a set of instructions which will result in the transformation of data from its memory into the other processor. For this type of operation the programs in the two processors must be coordinated. Some circuits are built into the MOBIDIC B system to direct such traffic bwtween the processors and avoid any uncertainty as to the direction of information flow between them.

Full utilization of the "built-in" duplexing capabilities of MOBIDIC B should provide a challenge to the imagination and foresight of programmers. Many programs could be written to take advantages of these duplexing facilities. For example, one processor could enter into a different program as a result of the decisions made by the other processor. Also, one processor could take over the other's task if it discovered that the other processor was either overloaded or incapacitated.

Marginal Check and Control Console

An automatic marginal checking system is incorporated in MOBIDIC B. The checking circuit is so designed that the bias voltage in each row of every rack in the computer is modified by a predetermined amount. While the bias voltage of the row is maintained at this changed value, a simple check program, which is stored in the computer, will be run through once. The result of this program can be observed, an error condition will be indicated by a pilot light on the console. All rows are automatically tested in succession in such a manner. It is possible to bypass some racks in the computer so that marginal voltages are not applied and checking is not performed on them. This is necessary to enable one processor and some associated input-output equipment to be in continuous operation while the other processor is undergoing test.

The control console for MOBIDIC is also duplexed. It consists of two independent and identical consoles assembled side by side. Each console is permanently connected to one processor and thus communicates only with its assigned processor. For ease of operation, each control console is divided into 6 horizontal areas. At the very top of the console is located the marginal check voltage control. Immediately below this area is the control for the power to the computer. The master clock selector is also located in this area. The display register is situated next in line, extending completely across the console. This is a full length indicating light register which is used to display the contents of any register or memory location selected by the operator. The area below holds the controls for the flip-flop and error detection. Directly beneath are the controls for the insertion of manual instructions. Initiation control for the computer such as start, halt, and single pulse, are laid out in the bottom row on the control console.

Inasmuch as the control consoles are assigned to their respective processors, operation of one console is entirely independent from the other. However, since the two consoles are located side by side, the operator can easily observe the status of both processors, allowing convenient control of both processors during duplex operations.

Concluding Remarks

Since there are only two physical working registers in each central processor, connections from the central processor to the in-out system are made only through the memory register. Traffic coming in and out of the memory as well as going to and from the in-out system frequently create logical problems. Resolution of these logical problems results in a slight reduction of computation speed.

The internal duplexing features introduced in MOBIDIC B represent a new approach in the design of a large scale, general purpose data processing system. There will undoubtedly be many areas where such an approach is highly desirable from the standpoint of reliability and economy.

Acknowledgment

Many people have contributed to this project. In particular, Messrs. W. S. Humphrey, Jr., and J. Terzian, who have been directing the system and logical design effort on this project have made numerous suggestions and contributed heavily towards the original concept for this entire system. Mr. G. Rocheleau was instrumental in the detail design of the central processor and the design of duplexing capability in MOBIDIC B.

Discussion

M. Rubinoff: Would you include simultaneous solution of the same problem in what you mean by duplex operation?

Mr. Chao: Yes I would. For instance, you can have two processors operating simultaneously, each doing a part of the entire problem, or you can have both processors operating on the same problem and duplicating the computation for reason of reliability.

Mr. Rubinoff: You can make comparisons?

Mr. Chao: Yes, you can make comparisons at the end, or any convenient intermediate point. Note that a simple yes or no type of comparison could be made through the mutual monitoring facilities built into the hardware. For more elaborate comparison, data would have to be transfered over through the input-output device.

C. W. Einolf (IBM): Would you discuss mass memory more thoroughly? Is it a single disc or many? Is it Sylvania designed or built by others?

Mr. Chao: The mass memory consists of eight large discs, 34½ inches in diameter. Both sides of the discs are used. There is a total of 4,096 tracks. The recording and reading is done in serial track by track. You can consider the entire storage to be a looped spiral. In other words the track sequence is continuous. If you go down to the 4,096th track, for the next increment you will get number one track again. Sylvania is collaborating with a contractor in the design of the mass memory; the contractor supplies the discs and we design the necessary electronic circuits.

J. Reitman (Teleregister): Can the MOBIDIC B system be extended to 3 or 4 processors? If so, is there a limit?

Mr. Chao: The logical problems involved in duplexing are not trivial. Extending the number of processors beyond two and tying them all to the same system bus would be undoubtedly a little tougher. There must be a limit somewhere. We have not yet investigated this limit.

G. Gaschnig (RCA): What is the speed of the core memory?

Mr. Chao: 8 microseconds read-write cycle.

W. Towles (Martin Co.): What is the clock frequency and what type of logic circuitry is used?

Mr. Chao: Clock frequency is 1 megacycle. The transistor inverter logic is used.

Mr. Towles: Do the core memories operate reliably at the high temperatures required of a military computer?

Mr. Chao: Yes, it has to. This entire system is designed to meet military specifications.

Mr. Rubinoff: Well, I think the question is which one?

Mr. Chao: Temperature compensation technique used in the memory circuit will be discussed in a paper tomorrow.

S. Levine (*Teleregister*): What error checking facilities are included and what is the expected undetected error rate of each processor?

Mr. Chao: The automatic error checking facility we have is the parity bit error indication. This gives you single error detection.

Mr. Rubinoff: Any odd number?

Mr. Chao: Yes.

Mr. Rubinoff: What is the suspected undetected rate?

Mr. Chao: We haven't looked into that.

Mr. Levine: Is interlocking automatic or programmed when both processors try to get to the same device?

Mr. Chao: They are automatically interlocked so that whichever comes first will get it.

R. E. Warner (*Ford Instrument*): If processor No. 2 determines that processor No. 1 is in error, how are you sure that processor No. 2 is itself not in error while processor No. 1 is actually correct?

Mr. Chao: I think we shall have to rely on the statistics. This is a double-error.

W. Buchholz (*IBM*): The numerical data word appears to have one more bit, the sign, than any other word. How is this sign bit transferred to and from magnetic tape? Does the tape control have to know whether a word is numerical or not?

Mr. Chao: The tape is controlled by the converter. In the case of numerical data, the sign bit is taken out and composed into a character. Therefore, for interpret sign mode of operation, you have 7 characters instead of 6 for each MOBIDIC word.

E. B. Cohen (*Auerbach Electronics*): How are the built-in subroutines constructed? How is return from these built-in subroutines effected?

Mr. Chao: The subroutine is written as program using the mechanized instructions. Whenever a subroutine instruction is encountered, the accumulator and the program counter are automatically stored by the logic circuit. They have to be saved because these two registers are used in the subroutine. Subsequently, the logic circuit will automatically transfer the program into the correct subroutine. Upon completion of the subroutine, all you have to do is to bring back these registers and return to the main program.

A Universal Computer Capable of Executing an Arbitrary Number of Sub-Programs Simultaneously*

JOHN HOLLAND†

Introduction

THIS PAPER describes a universal computer capable of simultaneously executing an arbitrary number of sub-programs, the number of such sub-programs varying as a function of time under program control or as directed by input to the computer. Three features of the computer are:

(1) The structure of the computer is a 2-dimensional modular (or iterative) network so that, if it were constructed, efficient use could be made of the high element density and "template" techniques now being considered in research on microminiature elements.

(2) Sub-programs can be spatially organized and can act simultaneously, thus facilitating the simulation or direct control of "highly-parallel" systems with many points or parts interacting simultaneously (e.g. magneto-hydrodynamic problems or pattern recognition).

(3) The computer's structure and behavior can, with simple generalizations, be formulated in a way that provides a formal basis for theoretical study of automata with changing structure (cf. the relation between Turing machines and computable numbers).

The computer presented here is one example of a broad class of universal computers which might be called universal iterative circuits. This class can be rigorously characterized and formally studied (the characterization will be published in another paper). The present formulation is intended as an abstract prototype which, if current component research is successful, could lead to a practical computer.

General Description

The computer can be considered to be composed of modules arranged in a 2-dimensional rectangular grid; the computer is homogeneous (or iterative) in the sense that each of the modules can be represented by the same fixed logical network. The modules are synchronously timed and time for the computer can be considered as occurring in discrete steps, $t=0, 1, 2, \ldots$

Basically each module consists of a binary storage register together with associated circuitry and some auxiliary registers (see Fig. 1). At each time-step a module may be either active or inactive. An active module, in effect, interprets the number in its storage register as an instruction and proceeds to execute it. There is no restriction (other than the size of the computer) on the number of active modules at any given time. Ordinarily if a module $M(i,j)$ at coordinates (i,j) is active at time-step t, then at time-step $t+1$, $M(i,j)$ returns to inactive status and its successor, one of the four neighbors $M(i+1,j)$, $M(i,j+1)$, $M(i-1,j)$, or $M(i,j-1)$, becomes active. (The exceptions to this rule occur when the instruction in the storage register of the active module specifies a different course of action as, for example, when the instruction is the equivalent of a transfer instruction).

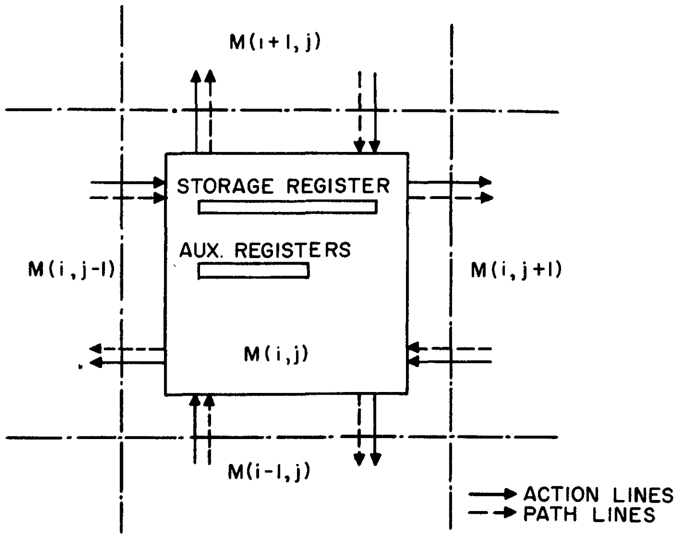


Fig. 1—A module.

The successor is specified by bits s_1, s_2 in $M(i,j)$'s storage register. If we define the line of successors of a given module as the module itself, its successor, the successor of the successor, etc., then a given sub-program in the computer will usually consist of the line of successors of some module. Since several modules can be active at the same time the computer can in fact execute several sub-programs at once. We have noted parenthetically that there are orders which control the course of action — there are also orders equivalent to store orders which can alter the number (and hence the instruction) in a storage

* Funds for this study were supplied to the Logic of Computers Group of the University of Michigan initially by the Air Force Office of Scientific Research under Contract No. AF18(603)-72 and later through a National Science Foundation grant (G4790).

† University of Michigan, Ann Arbor, Michigan.

register. Therefore the number of sub-programs being executed can be varied with time, and the variation can be controlled by one or more sub-programs.

The action of a module during each time-step can be divided into three successive phases:

(1) During phase one, the input phase, a module's storage register can be set to any number supplied by a source external to the computer.

(2) During phase two, an active module determines the location of the operand, the storage register upon which its instruction is to operate. This the module does by, in effect, opening a path (a sequence of gates) to the operand. Phase two is called the path-building phase.

(3) During phase three, the execution phase, the active module interprets and executes the operation coded in its storage register.

Path-Building

An active module determines the location of the storage register upon which its instruction is to operate by, in effect, opening a path to it. The path-building action depends upon two properties of modules:

First, by setting bit p in its storage register equal to 1, a module may be given special status which marks it as a point of origination for paths; the module is then called a P-module.

Secondly, each module has a neighbor, distinct from its successor, designated as its predecessor by bits q_1, q_2 in its storage register; the line of predecessors of a given module M_0 is then defined as the sequence of all modules $[M_0, M_1, \ldots, M_k, \ldots]$ such that, for each k, M_k is the predecessor of M_{k-1} and M_{k-1} is the successor of M_k (see Fig. 2). Note that the line of predecessors may in extreme cases by infinitely long or non-existent. The line of predecessors of an active module ordinarily serves to link it with a P-module (through a series of open gates). During the initial part of phase two the path specification bits $y_0, \ldots, y_n$ and d_1, d_2, in the storage register of an active module M_0, are gated down its line of predecessors to the nearest P-module (if any) along that line. The path specification bits are then used by the P-module to open a path to the operand (the storage register addressed by the active module).

Each path must originate at a P-module and only one path can originate at any given P-module. The path originating at a P-module is gated by means of a sequence of auxiliary registers called *-registers. Each module possesses 4 *-registers and if the module belongs to a path in direction (b_1, b_2) the appropriate *-register, $(b_1, b_2)^*$, is turned on. When $(b_1, b_2)^*$ is on it gates lines (to be described) from the module $M(i,j)$ to its neighbor $M(i+b_1, j+b_2)$ permitting certain signals coming into $M(i,j)$ to be passed on to $M(i+b_1, j+b_2)$ and vice-versa. Since each *-register

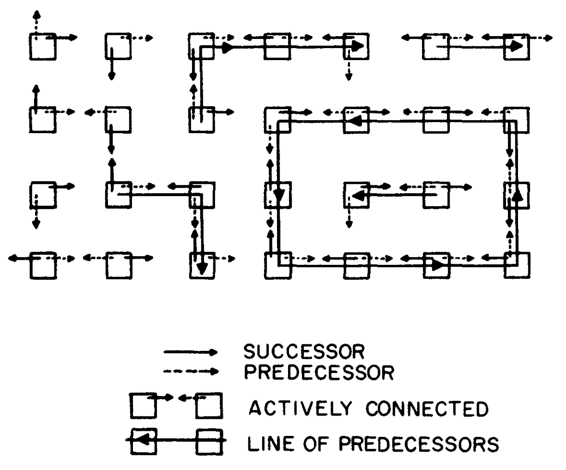


Fig. 2—Lines of predecessors.

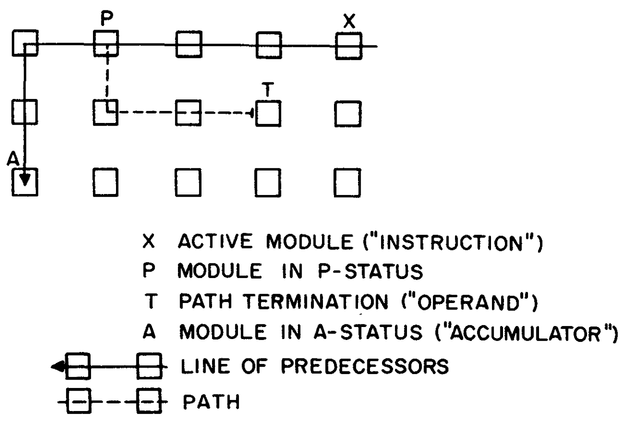

P
X
T
A
X ACTIVE MODULE ("INSTRUCTION")
P MODULE IN P-STATUS
T PATH TERMINATION ("OPERAND")
A MODULE IN A-STATUS ("ACCUMULATOR")
LINE OF PREDECESSORS
PATH

Fig. 3—Modules used in the execution of an instruction.

gates a separate set of lines, a module may (with certain exceptions) belong to as many as four paths. Once a *-register is turned on its stays on until turned off; thus a path, once marked, persists until erased.

The modules belonging to a given path can be separated into sub-sequences called segments. Each segment of the path is the result of the complete phase two action of a single active module. A segment consists of y modules extending parallel to one of the axes from some position (i,j) through positions $(i+b_1, j+b_2)$, $(i+2b_1, j+2b_2), \ldots,$ $(i+(y-1)b_1,$ $j+(y-1)b_2)$, where $b_1 = \pm 1$ or 0 and $b_2 = \pm(1-|b_1|)$; the module at $(i+yb_1, j+yb_2)$ will be called the termination of the segment (note that the termination of the segment is *not* a member of the segment.) The segments are ordered so that the first segment constructed has as its initial module the P-module. The k^{th} segment constructed as part of the path has as its initial module the termination of the $k-1^{th}$ segment. If the path consists of n segments, the termination of the n^{th} segment (the last segment constructed) will be called the path termination (see Fig. 3).

As noted, the path specification bits $y_n, \ldots, y_0$ and d_1, d_2 are gated down the line of predecessors from the storage register of the active module to the nearest P-module at the start of phase two. If $y_n = 0$, bits $y_{n-1}, \ldots, y_0$ and d_1,d_2 determine the length and direction, respectively, of the new segment. The total number of digits $y_{n-1}, \ldots, y_0$ equal to 1 gives the length of the segment — if j of the digits are equal to 1 then the segment will be j modules long. The digits d_1,d_2 turn on and set an auxiliary register, the direction register, in the initial module of the new segment. This gives the direction b_1,b_2 of the segment. The direction registers of the other modules belonging to the segment are all off, but each of the modules belonging to the segment (including the initial one) has its $(b_1,b_2)^*$ register turned on.

When $y_n = 1$, the final segment of the path originating at the P-module is erased. That is, the direction register in the initial module of the last segment is turned off and, as a consequence, all *-registers marking the last segment are turned off. If the path consists of a single segment or none at all the effect of $y_n = 1$ is to turn off the direction register in the P-module thereby making the P-module the termination of the path. That is, in this latter case, the path has no segments but it does have a termination — the P module itself (note that the *status* of the P-module is unchanged).

The following additional rules apply to paths:

(1) When a given module is the termination of several paths and direction register on-pulses arrive over more than one path at the same time, t, the result is no action — the direction register is not turned on and none of the paths is extended.

(2) Only one path can proceed through a module in which the direction register is on. Whenever the direction register of a given module M is turned on or given a new setting, any paths already running through that module will now have it as their termination. Furthermore, for each such path, the portion lying between M and the previous path termination is at once erased — the *-registers and direction registers marking that portion of the path are turned off.

(3) No P-module can belong to any part of a path other than its origin. If a path in the process of construction reaches a P-module then all construction ceases and the P-module becomes the termination of the path regardless of the value of digits $y_n, y_{n-1}, \ldots, y_0$. Further extension of the path will not be carried out unless the P-module's status is changed (its p bit set to zero).

Execution

Three modules play an important role during the execution phase of an active module: the active module itself holds the order code in bits i_1,i_2,i_3 of its storage register; the storage register of the nearest path termination contains the word to be operated on (the operand); finally there must be a module which serves as accumulator (see Fig. 3). In order to serve as an accumulator, the storage register of a module must first have bits (p,a) in it set to the value (0,1), giving the module special status — A-module status. (Note that this means a module in P-module status, $p = 1$, cannot be an A-module). If $M(i,j)$ is an active module then the first A-module along its line of predecessors serves as the accumulator. An A-module serves, in effect, to terminate a line of predecessors, since it can have no designated predecessor.

In the present formulation there are eight basic orders:

(1) The arithmetic operation ADD. Execution of ADD causes the number in the storage register at the nearest path termination (the operand) to be transferred to the nearest A-module and there added to whatever number is in the storage register of the A-module. (By using complements and iteration all the arithmetic operations, such as subtraction and multiplication, can be accomplished by means of this operation).

(2) The storage operation STORE. Execution of STORE causes the number in the storage register of the nearest A-module to be transferred to the storage register at the operand.

(3) The transfer operation TRANSFER ON MINUS. Execution of TRANSFER ON MINUS depends upon the number in the storage register in the nearest A-module. If $y_n = 0$ in this number then the active module, after completing phase two, becomes inactive and its successor becomes active. If $y_n = 1$ then the module at the nearest path termination, rather than the successor, becomes active.

(4) The index operation ITERATE SEGMENT. If $y_n = 0$ in the nearest A-module, execution of ITERATE SEGMENT (upon completion of phase two) reduces the number in the A-module by 1 and the active module remains active *without* causing its successor to become active. If $y_n = 1$, then execution of the order simply causes the successor to become active and the active module inactive at the completion of phase one. This operation provides a convenient means of building long paths in a given direction since, if N is the number in the nearest A-module, the path-building phase of the active module is iterated N times.

(5) SET REGISTERS causes the first 9 bits of the number in the nearest A-module to be used to set all 9 auxiliary registers at the nearest path termination, the j^{th} register being set on if the j^{th} bit is a one. It is important that the SET REGISTERS order can give the operand module active status by setting the appropriate auxiliary register. In this case the active module gives rise to two active modules on the next time-step, its successor and the operand module. By this means one sub-program can initiate activity in another.

(6) RECORD REGISTERS causes the state of the 9 auxiliary registers at the nearest path termination to be recorded in the first 9 bits of the nearest A-module (in the same order as used by the SET REGISTERS instruction).

(7) NO ORDER causes the execution phase to pass without the execution of an order.

(8) STOP causes the active module to become inactive without passing on the activity to its successor at the next time-step.

With the exception of the STOP, ITERATE SEGMENT, and TRANSFER orders, the active module becomes inactive and its successor becomes active at the conclusion of the execution of an order.

It is possible for a given active module to have *no* nearest P-module (or A-module) for any one of three reasons: (1) the module does not have a line of predecessors, (2) none of the modules along the line of predecessors is currently designated a P-module (or A-module), (3) there is no P-module along the line of predecessors *between* the active module and the nearest A-module. If there is no nearest P-module then there is neither path-building nor execution of instruction with respect to the active module (regardless of the content of its storage register). If there is no nearest A-module along the line of predecessors then the instruction of the active module is not executed although the path-building phase will be carried out (assuming a nearest P-module).

The following additional rules apply to active modules and their action with respect to P-modules and A-modules:

(1) If M_0 belongs to the line of predecessors of M_1, if the nearest P-module of M_0 is also the nearest P-module of M_1, and if M_0 and M_1 are both active, then the action of M_0 proceeds normally but M_1's action is as if it had *no* nearest P-module.

(2) If M_0 and M_1 are situated as in rule (1) except that they have the same nearest A-module, without sharing the same P-module, then the action of M_0 proceeds normally but M_1 acts as if it were executing a NO ORDER instruction.

(3) As mentioned earlier, a module can be given A-module status by setting the pair of bits (p,a) to the value (0,1). This turns on an auxiliary register in the module, the A-register. At the same time the bits of another auxiliary register pair, the $(D_1,D_2$-register, are set to match the bits s_1,s_2 in the module's storage register; i.e., when the A-register is on the (D_1,D_2)-register indicates the successor of the A-module.

Once a module is given A-module status it can be returned to normal status only in one of two restricted ways. The first way requires that a STORE order be executed by an active module which has the given A-module as its operand module (nearest path termination). Then, if bit a is 0 or bit p is 1 in the number being stored, the A-module reverts to normal status and the word in its storage register is that specified by the STORE instruction. Otherwise the A-module is unchanged, the STORE order not being executed. The other way of returning an A-module to normal status requires that the A-module receive external input during phase one. The above restrictions prevent the A-module from changing status when numbers are placed in its storage register during the normal course of its operation as an accumulator. During the time a module is an A-module the bits in its storage register are not interpreted in any way except as the digits of a binary number.

(4) A module in A-module status can become part of a path (or several paths) so long as it is not to be the initial module of a path segment. In this latter case the path-building action, which would make the A-module the initial module of a segment, is not carried out — the A-module remaining the termination of the path.

(5) A given module can be acted upon simultaneously by 2, 3, or even 4 STORE instructions if it is the termination of more than one path. Some provision must then be made to resolve conflicts when the numbers being stored are not identical. In the present formulation the conflict is resolved digit by digit: a 1 is stored at bit j in the storage register if and only if at least one of the incoming numbers has a 1 at position j.

(6) When a STORE instruction changes the word in the storage register of a module it is assumed that this change does not take place until the completion of phase three. Thus, for example, there is no conflict when the STORE instruction of an active module acts upon that module's own line of predecessors or, for that matter, upon the module itself.

(7) If an active module has an A- or P-module as successor then, at the next time-step, *the successor of the A- or P-module* becomes active, rather than the A- or P-module itself (unless, of course, the instruction just executed specifies otherwise).

INPUT

During phase one, the initial phase of each time-step, a module's storage register can be set to any arbitrarily chosen value and its auxiliary registers to any desired condition. The numbers and conditions thus supplied are the computer's input. Although the number in the storage register can be arbitrarily changed at the beginning of each time-step, it need not be; for many purposes the majority of modules will receive input only during the first few moments of time ("storing the program") or only at selected times $t_1,t_2,\ldots$ ("data input"). Of course, some modules may have a new number for input at each time-step; in this case the modules play a role similar to the inputs to a sequential circuit.

SUMMARY OF ORGANIZATION AND SYMBOLS

As noted in the general description of section 2, each module consists of a storage register plus some auxiliary registers. The earlier discussions indicate that the auxiliary registers required are:

(1) the E-register, a one-bit register which is on if and only if the given module is active;

(2) the A-register, a one-bit register which is on if and only if the given module is an A-module;

(3) the D-register, a one-bit register which is on if and only if the given module is the initial module of a path segment;

(4) the (D_1,D_2)-register, a register, with two bits of storage, which indicates the direction (b_1,b_2) of a segment if and only if the D-register is on and which indicates the direction of the module's successor if and only if the A-register is on.

(5) the $(b\ ,b_2)^*$-registers; each is a one-bit register which is on if and only if the given module is a member of a path segment with direction (b_1,b_2).

For formal purposes we can symbolize the state of a given register, X, at coordinates (i,j) and time t by the predicate $X(i,j,t)=1$ if the given register is on at time t.

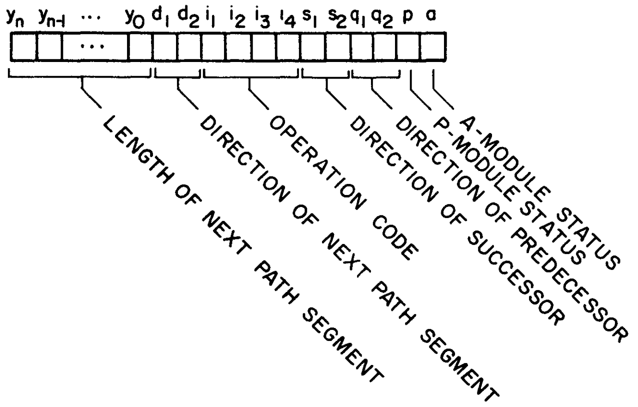

y_n y_{n-1} ··· y_0 d_1 d_2 i_1 i_2 i_3 i_4 s_1 s_2 q_1 q_2 p a
LENGTH OF NEXT PATH SEGMENT
DIRECTION OF NEXT PATH SEGMENT
OPERATION CODE
DIRECTION OF SUCCESSOR
DIRECTION OF PREDECESSOR
P-MODULE STATUS
A-MODULE STATUS

Fig. 4—Control of module by storage register.

The storage register of each module in the present formulation consists of $n+12$ bits (see Fig. 4) labelled in the following order:

bit number:	$n+12$	$n+11$	...	12	11	10	9	8	7	6	5	4	3	2	1
label:	y_n	y_{n-1}	...	y_0	d_1	d_2	i_1	i_2	i_3	s_1	s_2	q_1	q_2	p	a

The bits s_1,s_2 and q_1,q_2 designate the successor and predecessor, respectively, of the module. If bit p is 1 the module has P-module status. If the pair of bits (p,a) are set to the value (0,1) as the result of input or a STORE operation, the module has A-module status. During the path-building phase bits $y_n, \ldots, y_0$ and d_1,d_2 in an active module are interpreted as segment length and direction respectively. During the execution phase bits i_1,i_2,i_3 in an active module are interpreted as the operation to be performed. The word in the storage register of an A-module is treated strictly as a binary number with y_n being the sign bit and the other $n+11$ bits being arranged as indicated with y_{n-1} being the high order bit and a being the low order bit.

Comment

A universal machine in which the programs have a spatial organization has several properties over and above those usually associated with Turing machines and their concrete counterparts. For example, cycles in the program can actually be stored as cycles (of successors) in the rectangular grid (see Fig. 5). This, in effect, provides *each* cyclic sub-program with an instruction address counter which counts modulo the number of instructions in the sub-program (cf. an index register which can be set to cycle modulo any base number). Furthermore, each sub-program can be allotted a certain area in the grid and this allows the spatial arrangement of the sub-programs to match, for example, the structural organization of a process which is being simulated — each subprogram in this case directly simulating one of the components of the process.

Efficient programming of certain types of problem will require techniques similar to those required for asynchronous operation. That is, when several sub-programs are operating simultaneously, each sub-program will from time to time require results from other sub-programs, however these results will not in general be available at just the time desired. In problems like this, usually arising in the control or simulation of "highly-parallel" systems with many points or parts interacting simultaneously, the programmer will employ many of the techniques of the logical designer.

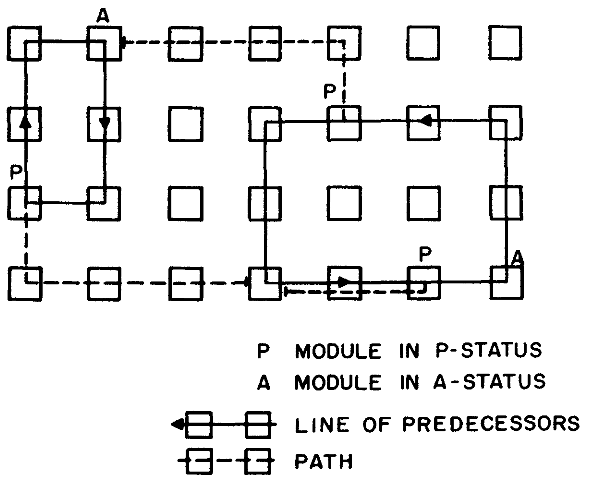


Fig. 5—Two interacting subroutines.

Problems such as the one just discussed emphasize the desirability of a computer formulation amenable to theoretical investigation. The present formulation is one example of a broad class of computers which can be rigorously characterized and investigated by abstract deductive techniques. Actually, the definition of this class of computers comes as part of an effort to provide a formal basis for the study of growing automata. By considering the rectangular grid to be infinite (or potentially infinite) in each of its dimensions (in analogy to the infinite tape of a Turing machine) many problems of automata theory can be expressed in a formal framework similar to that provided by Turing machines for problems of computability. Thus, for example, models of various processes can be stated as programs, or classes of programs, for the machine and investigated both directly and theoretically.

There are several variants of the formulation given here which yield computers which are either more flexible or have simpler modules. As a single instance, the path-building procedure could be altered to make branching paths possible; in this way the same sub-program could operate on several storage registers simultaneously.

A final word about concrete realization of such a computer: a partial rendering of the logical diagrams for a module in the described computer indicates that a module with a 40-bit storage register could be constructed with approximately 1000 basic elements. If this is actually the case and if switching is accomplished with micromodular densities, say 10^8 elements per cubic foot, then the basic portion of a computer with 100,000 modules should be realizable within a volume of a few cubic feet (exclusive of input-output equipment, power supply, etc.).

Discussion

M. Rubinoff: Would you expect this two-dimensional mondular structure to have important significance for character recognition and other aspects of "gestalt" or area vision?

Dr. Holland: I would hope so. This was part of the original intent in constructing the computer. The hope was that it could operate on all parts of the pattern at the same time. The computer is really a by-product of research on what we call growing automata.

P. Rosenblatt (*Teleregister*): Since any module may be in P status at some time, will not this "anywhere-to-anywhere" transmission result in extremely large cost of gating when the number of modules is large?

Dr. Holland: I think the point here is that any module may be in P status; however, gating is from module to module, and you open these gates in serial fashion as specified by the number in the storage register of the active module. Since each module will have a fixed number of gates controlling the path lines to its four neighbors, the number of such gates is directly proportional to the number of modules. Note that only one path can proceed from any one P module at any given time.

R. A. Kirsch (*Nat'l Bureau of Standards*): Why did you constrain the machine to the two-dimensional rather than the multi-dimensional?

Dr. Holland: This was for purposes of exposition. As I say, this is one example of a large class of machines and you can have any number of dimensions in general. If you get beyond three dimensions you are operating entirely in theory.

G. Richmond (*Cornell Aero Lab.*): Can paths be destroyed as well as created?

Dr. Holland: Yes, I didn't mention this in the talk but if you set the first bit in the storage register (labeled Yn) to one, in effect this says that, when the path-building phase of that module is carried out, the last segment of the path is to be erased. In other words, you pay no attention to the rest of the bits $Yn-i$ through yo; you simply erase the last segment of the path.

L. R. Bowyer (*Bell Labs.*): Is only one module active at a time? How does a path get formed through non-active modules?

Dr. Holland: No, any number of modules may be active at the same time. In fact, by means of interlocking conditions you can allow modules along the same line of successors to be active. You recall there was a square red line of successors in the last slide; it is even possible for more than one module to be active in that red square. The number of interlocking and override conditions is surprisingly small. There are only 12 of them. The activity of a module has no direct effect upon paths passing through it. If a module is active it makes no difference as far as these paths are concerned. The path lines are separate sets of lines that pass through modules. The opening of gates in the path lines are caused by active modules (via a P-module) but the path doesn't have to pass through an active module.

G. J. Moss (*Naval Ordnance Lab.*): Could you elaborate on the process of reading data into the computer?

Dr. Holland: This is a part I haven't paid a great deal of attention to. I would say this, that in general one would like to have some device that could put data into any storage register of this modular arrangement at any time. This seems a little difficult from a practical viewpoint so it seems you would have to settle for some sort of scanning technique to put data in at particular times. In order to make effective use of this computer in some communication system or pattern recognition device you would certainly put data in at many modules at many times; perhaps at each time-step during the input phase. All data has to go in during the input phase.

W. Buchholz (*IBM*): What happens if one module tries to create a path that would have to cross an already existing path?

Dr. Holland: As long as it does this at a place which is different from the termination of a segment this is all right. As I mentioned during the talk, each *segment* of the path proceeds in a given direction (parallel to one of the axes). Modules can have as many as four paths running through them one to each of the four neighbors. These lines are independent except at points where the paths turn. In other words, at the termination of a segment where you add on a new piece you may have a right angle turn. At this point you need a crossover matrix from one path line to another. At this point an interlock would prevent a second path going through. If you tried to build a second path at this point nothing would happen.

T. Gilmer (*ITT Federal*): What is the function of the auxiliary storage? How do you deactivate a net once started?

Dr. Holland: I didn't go through this but I can give you an example. I mentioned that if you have the a bit in the storage register equal to one then the module is in A-module status. This means that during an ADD operation you might change the content of the storage register in the A-module and change the a bit to a O. Then the module would lose its A-module status. This you don't want. So one of the auxiliary registers keeps the module in an A module status so that changes in the contents of its storage register make no difference. Other auxiliary registers perform functions that are similar.

L. Clapp (*Sylvania*): Would Mr. Holland care to comment on debugging, diagnostic and preventive maintenance techniques of this computer?

Dr. Holland: The programming here would involve two kinds of techniques. It would take quite a bit of the best of programming techniques together with techniques of logical design. You may have circumstances very similar to asynchronous operation, if the second program has to wait for the first program to produce the data. A programmer of this machine has to be a good programmer but also has to be a good designer, too. In some cases, e.g. the solution of two-dimensional differential equations, since you repeat the same program over and over across the module, the debugging might be simpler than in a single-sequence computer because you have no logical effort to put into a scanning program — the program structure would match the mesh structure.

The Multi-Sequence Computer as a Communications Tool

J. N. ACKLEY†

THIS is a report on the merging of two fields: communication switching and computers. Recent advances in the computer art make it possible to satisfy the ever increasing communication switching requirements brought on, in part, by computers themselves employed in centralized data processing systems. Present record communication systems have significant delays which are not primarily caused by the transmission times but by the time required for the operations in the communication message switching centers.

In the past, most communication has been from human to human. Communication systems have become more complex and automated to meet the ever increasing needs of commercial and military activities. We are faced with a revolution. Increasingly, communication will be between humans and machines and between machines and machines. This transition will place more stringent requirements on accuracy, reliability, and speed.

Present day electromechanical switching centers are limited in their speed of operation due to their electromechanical nature and due to limitations of the transmission means, namely, teletype. Little error detection and correction capability is presently found in these systems.

Centralized data processing requires the error free transmission of large volumes of data to a central point. Usually, the communication with machines or between machines involves little redundancy such as that found in plain English. The computer, although it can make validity checks on the data it receives, cannot fill in missing letters or words as a human can. Since the present electromechanical systems are special purpose devices, all messages routed through the system must adhere to a very rigid format. This leads to difficulty when trying to integrate data gathering devices and different types of computers with different codes and formats. Military command control systems, especially, require data inputs from varied sources.

Stored program techniques could solve many of these communications problems. A programmed switching center could provide the error checking and error correction procedures as required. It could be programmed to translate from one code to another, indeed, perhaps from one language to another. Various speeds and code structures could easily be accommodated.

† International Electric Corporation, Paramus, New Jersey.

In addition to improving the features presently found in some switching centers, stored program techniques could be used to implement features which are not practicable with electromechanical or special purpose switching systems. A programmed switching center could generate, receive and interpret service messages to and from other machines or human operators. It could test and monitor all of the communication links and reroute messages if necessary to avoid inoperative links. One of the biggest advantages of a programmed switching center is its ability to be reprogrammed to account for changes in operational procedures, routes and equipment.

One of the most important considerations in employing computer techniques to the communications switching problem is the large number of input and output channels required. Also, all channels must operate simultaneously and independently. Military practice requires that each message be forwarded as far as possible whenever the communication link is available. Thus, the communication switching center must accept a message on each communication line whenever the subscriber wishes to transmit.

For many years the bottleneck on efficient use of computers and on the application of computers to real-time systems has been the problem of integrating input-output devices. Most input-output schemes have involved a large amount of equipment external to the central computer to provide for buffering and control. Now that the versatility of the high speed random access core memory has been fully appreciated, an almost limitless number of schemes is possible. Indeed, a single computer may use several schemes to integrate various input-output devices.

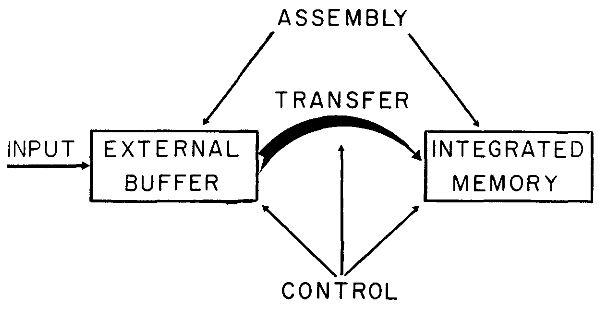


Fig. 1—Input-output classification.

In order to discuss the various schemes and compare their advantages and disadvantages, it is necessary to establish a classification system. There are four parameters, as shown in Fig. 1, which characterize an input-output scheme:

Assembly refers to the process of packaging or unpackaging information into definite size units. In order to characterize the assembly (or the disassembly) process, one must specify how and where the transformation takes place among bits, characters, words, records, and files.

Buffer refers to the unit external to the central computer which holds information until it can be transferred. The buffer may also be involved in the assembly process. It is characterized by its capacity, access time, and assembly features.

Transfer refers to the process of exchanging information with the integrated (addressable) memory of the central computer. The transfer is characterized by the number of bits transferred in parallel.

Control refers to the process which determines the sequence of operations of an input-output channel. In order to carry out the control function, control words must be supplied to a control device. Typical words that are usually involved in input-output are:

Selection code
Number of units of information to be transferred
Address in integrated memory at which the transfer is to begin
Address in external device at which the transfer is to begin
Location of next control word

Some or all of the control words are required for any input-output transfer. The device which uses these control words can be the central computer or a separate input-output control device which provides for the proper sequence of operations.

In designing a system, engineering compromises must be made. Many combinations and permutations of the above factors can be made. Each permutation will have certain advantages and disadvantages which must be weighed against the system application. For example, to take the extremes, a system can be designed like the IBM 709 system which has an external control device (the Data Synchronizer). This device has a register for buffering and additional registers for storing the control words. This system requires only a core memory cycle for a transfer and thus takes little time away from the central computer during input-output transfers. However, this scheme requires extensive hardware for the external control device.

On the other hand, a system may be designed to store all of the control words in integrated memory and the central computer could supply the control. Such a system would require a minimum of equipment external to the central computer for control of the input-output transfers. However, in a system application which requires a large volume of input-output, a large percentage of the capacity of the machine would be tied up in the input-output transfers.

A typical communications switching center may have 50 to 100 two-way communication lines. In order to have a computer perform the functions of a store and forward switching center, it must have corresponding numbers of input and output channels. In addition, it must have a sufficient processing capacity to determine distribution and optimum routing, priority, and to perform validity and parity checks if required. Not only must it provide for these large numbers of input and output channels, but all channels must be capable of operating simultaneously. Neither of the extreme systems described above is satisfactory for this type of operation. However, both of these schemes can be made practical for this type of application by multiplexing as shown in Table I.

TABLE I

IN-OUT SYSTEMS FOR COMMUNICATION SWITCHING

Assembly	Buffer	Transfer	Control
1. Bits to character external	Character	Character break-in	External-multiplexed interrupt or multisequence for special cases.
2. Bits to character external	Character	Character programmed	Central computer-multisequence

For example, the external control device can be multiplexed so that it may service a number of input and output channels simultaneously. This also requires that the central computer have a number of independent memory banks so that the transfer of the control words from storage to the external control device and back to integrated memory on each input or output transfer will not saturate the system.

By multiplexing the central computer, it can be used to handle a large number of communication lines. A central computer can be made to time share over a large number of channels by utilizing the multisequence configuration.[1] This configuration provides a number of program counters and a system for switching from one program counter to another, in response to external stimuli.

In order to select the proper scheme, a detailed study must be made of the particular communication switching center requirements. The external control scheme gives a higher capacity under certain conditions, but it requires more equipment. The study of

[1] W. A. Clark, "The Lincoln TX-2 Computer Development"; J. W. Forgie, "The Lincoln TX-2 Input-Output System"; *Proceedings of the W.J.C.C., 1957.*

the communication requirements must determine the ratio of the number of core memory cycles required for internal processing to the number of core memory cycles required for input and output transfers. If this ratio is sufficiently high, then the central computer control scheme is to be preferred, since relieving the central computer of the burden of input-output transfers would free only a relatively small percentage of the system capacity. On the other hand, if this ratio is low, then the external control scheme is to be preferred since a low ratio implies that the main function is that of input-out transfers.

In a system which requires utmost accuracy achieved by the use of redundancy checks, and which must automatically route and reroute messages to account for communications outages and supply other functions such as code translation; this ratio may be in the order of 2 or 3 to 1. Thus, only a 33 to 50 percent increase of capacity is the maximum that could be expected by utilizing an external control device and multiple access integrated memory — neglecting memory reference conflicts.

Another significant factor which affects total capacity of the system is the assembly process and the capacity of the external buffer. Since most communication is based upon characters of five, six, seven or eight bits, it is desirable to handle the assembly from bit to character externally.

Systems have been proposed utilizing only a single bit external buffer, with the assembly from bit to character to word to message in integrated memory. However, the capacity is severely reduced since now a transfer must take place on each and every bit of a message. Increasing the size of the external buffer on the other hand increases the capacity, but at the same time increases the amount of equipment. The maximum buffer size to be considered is that of the word length. Therefore a compromise must be reached between the amount of equipment in the external buffers and the capacity of the system. This compromise is influenced by the number of lines to be terminated, by the speed of the circuitry available to the programmed element, and by the nature of the transmission system.

Those who have been through a computer development program will immediately ask, "Is it necessary to design a special computer for the communication switching system?" or, more generally phrased, "Can a general purpose computer be utilized?" This question is not easily answered. The answer must be based on a thorough study of the communication system application. Surely, if the only job of the programmed element is that of communication switching, then there is no need for floating point or even multiply or divide instructions in the instruction vocabulary. Thus, simplifications can be made in the design of a programmed element which is to be used solely for communication. The specialized programmed element can be optimized for communications.

On the other hand, if the computer is to be used for both communications and computations, it may be desirable to design an external control device which can be adapted to a general purpose computer. However, the range of applicability of the general purpose computer seems rather limited. If the computational requirements are very great, it may be profitable to employ the special purpose programmed element for the communications switching center and also to act as an input-output processor for a much larger data processor. This philosophy is illustrated in the STRETCH and LARC computers where a simple input-output processor is used to relieve the complicated, high speed data processor of the simple routine tasks associated with input and output editing. The in-out processor can also be used to operate or schedule the operation of the larger data processing system to achieve a much higher utilization of the data processor than would be possible with a human operator.

In selecting a computer or designing a computer for communication system switching center application, consideration must be given to the peak and average traffic rates. Generally, the utilization factor of communication lines is approximately 0.1 to 0.2. This surprisingly low figure is justified because of the queueing problems which would ensue with a very high utilization of any communication link. Assuming a Poisson distribution of message arrivals, a utilization factor approaching unity would result in a queue approaching infinity. Therefore, to minimize the queueing problems and to assure rapid transmission of the message, the utilization factor is desirably kept approximately at 0.2. The message processing capacity of the switching center must also be designed to exceed the average traffic load in order to eliminate excessive queues of messages awaiting processing.

The use of a multisequence computer for a communication message switching center permits the termination of a number of lines which would saturate the computer with input transfers if all lines were operating at full capacity all of the time. The internal and output processes can be assigned a lower priority in the multi-sequence scheme and these operations suspended until the input peak passes. The probability of such an occurence is extremely small. Since it also is extremely rare that many of the communication lines would be simultaneously busy with input traffic, additional lines may be terminated with the probability that the switching center would lose an input character. In a completely automated system, this would be caught by the error detection and correction system and cause only a slight delay in transmission. The amount of excess message processing capacity required to reduce the processing queue to satisfactory proportions depends upon the delays permitted and upon the priority structure of the messages.

A programmed Traffic Control Center is currently being fabricated at ITT Laboratories. This Traffic Control Center, which utilizes the multisequence technique, was undertaken as a study project in 1957 at the Laboratories and was then proposed by this author as the solution to the communication message switching problem of the SAC Control System, Project 465L, in July 1958. The system design resulted in a computer with several distinguishing design characteristics which make it peculiarly efficient in handling communications. The central computer is multiplexed by use of the multisequence technique. Separate memory units are provided to store 256 program counters and 256 index registers. One index register is associated with each sequence; thus, in essence, 256 separate sequences may time-share the central computer. Each communication line is terminated by a simple character buffer and each buffer has associated with it a service request flip-flop. As data becomes available, the service request flip-flop is set and competes for time on the central processing unit. Each instruction has provision for a break-bit or a dismiss-bit, as shown in Fig. 2, which when set indicates that the present sequence may be interrupted in favor of a higher priority sequence or that the present sequence may be dismissed, in which case the service request flip-flop is cleared and the remaining sequence with the highest priority is activated.

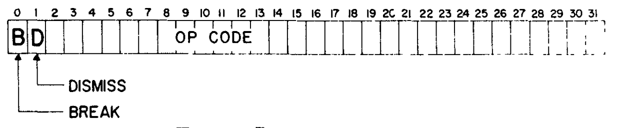


Fig. 2—Instruction word.

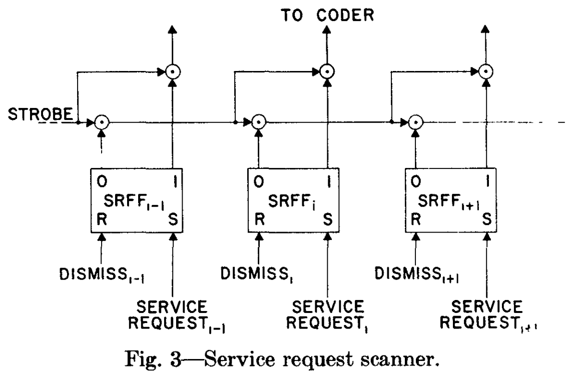


Fig. 3—Service request scanner.

To take advantage of statistical averaging of the inputs and outputs, an on-demand scanner (shown in Fig. 3) does not operate on a fixed cycle, but service is requested immediately if no other channels are busy at the same time. In order to incorporate devices of different speeds, and to provide an orderly procedure for servicing requests if more than one should arrive at a time, the service request scanner operates on a priority basis. A strobe signal proceeds from one service request flip-flop to the next until the first one which is set is encountered, at which point the strobe signal is routed to the coder.

Fig. 4 shows a typical input channel. When the strobe signal arrives at the coder it is converted into binary code. If the instruction that the central computer is executing contains a break bit, the output of the coder will be compared with the number in the sequence register. If they are not equal, a sequence of higher priority number has requested service. Notice that the number in the sequence register operates a switch which selects the particular external device to be used at any given time and routes control signals to the external device. The dismiss signal is issued whenever the present sequence has completed all operations necessary to answer the service request. This signal clears the service request flip-flop and prepares it for receiving the next request for service from the external device.

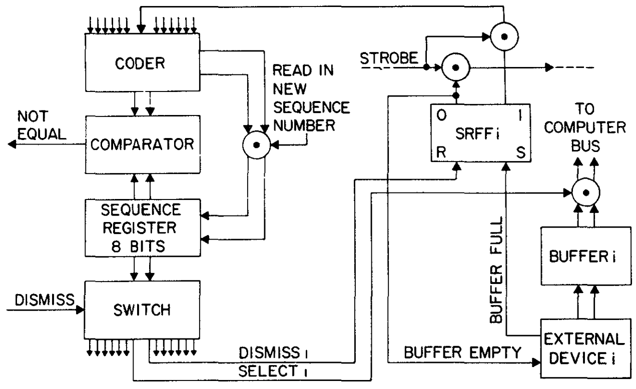


Fig. 4—Typical input channel.

Fig. 5 shows the registers involved in changing sequences. If the number from the coder is found not equal to the number in the sequence register and the current instruction has a break or dismiss bit, a change of sequence is called for. Since the control must return eventually to the old sequence whenever that particular channel requests service again, the program counter must be stored for future reference.

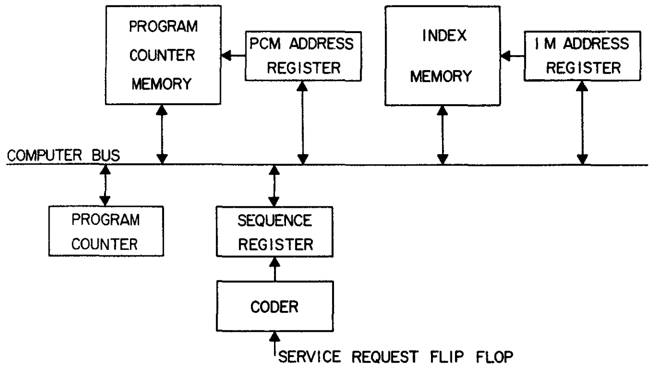


Fig. 5—Registers involved in changing sequences.

Therefore, the first operation is a transfer of the sequence number from the sequence register to the program counter memory address register. The program counter is then stored at the location so specified. After the old program counter has been stored, the new sequence number is read from the coder into the sequence register and then to the program counter memory address register. The stored program counter stored at this location is now placed in the program counter and the computer continues to take instructions specified by the program counter.

In order that each sequence may have its own index register, the contents of the sequence register are transferred into the index memory address register and the contents of the location so specified are used for the indexing operation. Since the program counter memory and the index memory are independent units and can operate concurrently with the main integrated memory of the computer, changes of sequence can take place without any loss of time, that is, an instruction with a break or dismiss in one sequence may be immediately followed by an instruction in some other sequence.

Note that if the contents of the accumulator were stored along with the contents of the program counter, we would truly have a multiplexed computer. However, this feature was not necessary for the communication switching center, but care must be taken during programming in placing break and dismiss bits only at those points in a sequence where the contents of the arithmetic unit are immaterial.

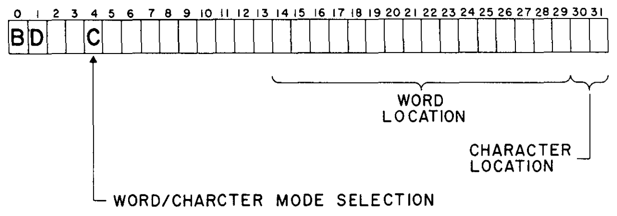


Fig. 6—Instruction word.

Since most communication and data processing systems have five, six, seven or eight bit characters, most instructions are capable of operating in two modes. In the word mode, the operation applies to the entire computer word of 32 bits. In a character mode a single 8-bit character is referenced. Since there are four eight-bit characters in a 32-bit word, an addressing scheme as shown in Fig. 6 was devised which permits convenient addressing of consecutive characters in memory. The two least significant bits of the address part of an instruction refer to a character position within a word. In the word mode, these bits are ignored. The instruction vocabulary contains a liberal quantity of logical instructions, but does not contain multiply or divide or any other numerical operations other than add and subtract.

Most instructions have a bit to indicate the repeat mode, as shown in Fig. 7. This permits most input-output transfers to be accomplished with a single instruction requiring two core memory cycles. As the characters are transferred in from the external buffer,

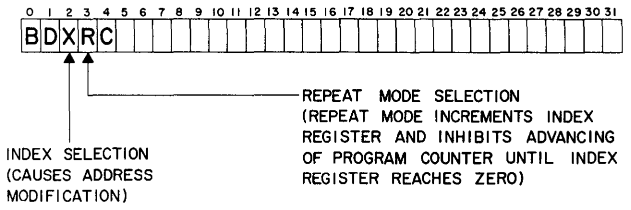

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
B D X R C
REPEAT MODE SELECTION (REPEAT MODE INCREMENTS INDEX REGISTER AND INHIBITS ADVANCING OF PROGRAM COUNTER UNTIL INDEX REGISTER REACHES ZERO)
INDEX SELECTION (CAUSES ADDRESS MODIFICATION)

Fig. 7—Instruction word.

a check on parity and a check for a control character are performed by common circuitry. If either condition prevails, it is indicated to the program by means of a skip. Otherwise the input-output transfers are handled by a single instruction in the repeat mode with the dismiss-bit set. This causes the character to be transferred, the index register to be incremented, and the present sequence dismissed in favor of some other sequence. The message processing sequence has the lowest priority and processes messages between transfers.

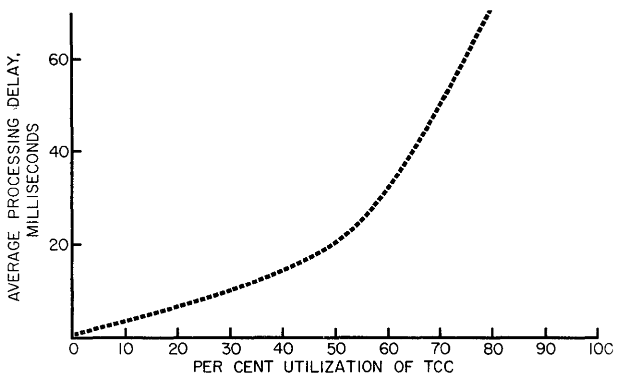


Fig. 8—Comparison of per cent utilization (of TCC) *vs.* average processing delay.

Fig. 8 shows the comparison of the percentage of utilization of the Traffic Control Center versus the average processing delays. This truly represents an advance in the communication message switching art. Present switching systems delay each message by many minutes. Now the delay is measured in milliseconds.

Most comprehensive communication systems require both circuit and message switching. The programmed switching center can effect digital circuit switching by associating, by program means, a particular input channel with a particular output channel. The delay under these conditions would be only a few microseconds and would be insignificant compared to the delays in long transmission lines. However, since the traffic is just coming in and going out without any processing, the capacity of the programmed element is used unnecessarily. In a communications system which requires a large amount of circuit switching, it will be preferable to terminate all communication lines in an electronic cross-bar switch, as shown in Fig. 9, which has many trunks to the programmed element and is under its control.

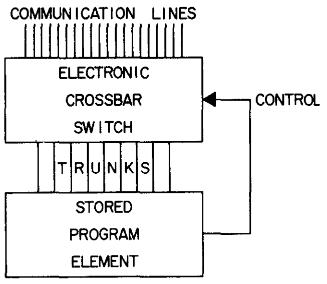


Fig. 9—The ultimate communications switching center.

This configuration would permit both message and circuit switching on an intermixed basis and permit changing from one type of service to another by simple indications to the stored program.

Discussion

W. G. Stevens (IBM): What is the maximum character rate on a single channel?

Mr. Ackley: The maximum character rate on a single channel is determined basically by the number of core cycles per second. In the present ITT design, which has an 8-microsecond memory cycle, the maximum speed is 62,500 characters per second. This is available over to the totality of inputs.

Mr. Stevens: With all channels operating and the computer transferring characters from input to output channels, but performing no editing, what is the maximum character rate per channel?

Mr. Ackley: The maximum average input rate is fifteen 250-character messages per second. That runs it up to 3,750 characters per second.

G. W. Bleisch (Bell Tel. Labs.): What provisions are made for multiple address messages?

Mr. Ackley: I hate to say it but this is programming detail. This has frustrated quite a few of the communicators using this system. The entire message processing operation is determined by stored program.

K. Enslein (Brooks Research): Have you given any thought to what an electronic crossbar would be composed of?

Mr. Ackley: That is outside my field of interest although there are several developments under way. I understand that Stromberg Carlson is working on one and ITTL has been also working on an electronic switching system known as Digicom.

D. A. Bourne (IBM): What error reduction scheme is used?

Mr. Ackley: In our system we have both parity per character and also parity characters at the end of each message providing a cross check. If an error is detected by these parity checks, a request for automatic retransmission is given by the Traffic Control Center which causes the transmitter to retransmit the message. As an additional feature, again this is a program detail, every message must be acknowledged. If a transmitter sends a message and it is not acknowledged in three seconds it automatically retransmits the message.

R. G. Turner (Philco): How is message priority established? How do you determine relative priority of input signals?

Mr. Ackley: The relative priority of input channels is not related to the priority of the messages. The priority of the input channels is to permit the integration of different speed devices and provide orderly procedure for handling a multiplicity of requests. As to the priority of the message, the message format has in it a provision for indicating the priority, and again the priority processing is a program detail. First thing after it receives a message, the program checks the priority, and establishes proper program connections to process it on a priority basis.

E. B. Cohen (Auerbach Electronics): How much message storage capacity does the system have?

Mr. Ackley: This is determined by the system applications. This ITTL system has 16,000 words of core, 130,000 words of drum and approximately 12 tapes for bulk storage. Usually messages come in and go out so fast there is little working storage required in the core memory and no additional storage is required for messages except for the journal which is on magnetic tape.

D. Dittberner (IBM): You mentioned a study program in 1957. Did this give rise to a piece of hardware and who sponsored the study?

Mr. Ackley: This was a study sponsored with ITT General Development Funds and it was just that, a study, not a development.

H. P. Peterson (Lincoln Lab.): Can any sequence get at the program counters and index registers of other sequences?

Mr. Ackley: Yes, all of the program counter core memory and the index core memory is addressable.

Mr. Peterson: What is the number of instructions in your machine?

Mr. Ackley: Instructions list runs, I think, 40 some instructions.

Realization of Boolean Polynomials Based on Incidence Matrices

S. OKADA†, Y. MORIWAKI‡, AND K. P. YOUNG

THIS PAPER describes a general algebraic method of finding minimum contact networks for any given Boolean polynomial. Solutions obtained by this method may in general be any kind of connection with any number of contacts for each variable. Furthermore, any practical requirements such as series-parallel cases usually found in most electronic devices, and single contact for specified variables, can all be considered in the calculation, if necessary. A routine algorithm on incidence matrices will automatically yield any ingenious connections. The node-branch and branch loop incidence matrices which were revealed by G. R. Kirchhoff[1] 1847, are adopted as unknown, especially those of modulo 2 which were elaborated by O. Veblen[2] in 1916 are mostly used. However, simultaneous use of modulo zero (or infinity), 2 and other integers was found useful for combinatorial consideration in the multi-contact case. General Galois fields are already used in switching theory by Moisil[3] and his Rumanian group.

General non-series-parallel synthesis of switching 2-terminals[4] by incidence matrices modulo 2 began in 1939 and has been developed further.[5,6] Especially R. Gould had established a systematic method for the general multicontact case. His significant improvements of solutions of the four variable problem[6,7] prove the usefulness of incidence matrices.

The main novelties of this article are as follows:

(1) A rigorous use of incidence matrices gives an essentially new approach to find all possible complicated connections. The method can also be used for finding a single solution or giving proofs of various minimalities.

(2) Topological enumeration of nodes, branches and degrees of freedom gives a criterion of realizability.

(3) Realization[8] of individual connections from each incidence matrix was reduced to a routine process by the new graphical or algebraic "*ambit-method*". It can also be used for network synthesis of other kinds. Fortunately an algebraic-topological review[9,10] of network equations from the geometric standpoint of H. Weyl[11] and G. Kron[12] clarifies the possibility because topological properties relating to connections of networks belong to affine geometry[10] which has no "metric"; that is, these cases are perfectly independent of any branch causality such as Ohm's law in d.c. or a.c.[10] Such *affine network theory* can be called the "*pre-Ohmic*" or "*Ohm-free*" *network theory* to which the *Boolean network* theory belongs.[13] This shows that incidence matrices can be used in the synthesis of networks with rectifiers, hysteresis or any other nonlinearity.

(4) Loops corresponding to ties are expressed by vectors, and cut-sets for barriers are expressed by covectors which mean pairs of initial and terminal hyperplanes in a branch-number-dimensional *affine* space. Hermann Weyl's "*method of orthogonal projection*"[11,12] is generalized to affine projection for vectors and to intersection with a subspace for covectors, and forms the general foundation of the whole topological network theory.[10]

(5) The invariant transformations of variables for given Boolean functions form a non-commutative group[15] which plays an important role in this method.

(6) Semi-ordered vector sets. All vectors of only zero and unity components (modulo 2) of A-dimensional space form an additive group of order 2^A. All loop vectors (modulo 2) of a network form its subgroup. However, its subset of vectors of a single loop passing the relay branch, or more shortly expressed as "*single relay-loop vectors*" does not generate a group because only a sum of an odd number of these vectors generates a relay-loop vector, and further, a sum can be either a single relay-loop or a multiple loop which consists of a single relay-loop and unseparated or separated loops of contact branches. Therefore, if the numbers of independent single relay-loop vectors and all dependent relay-loop vectors of this subset are respectively denoted with C and K, the total number W of the relay-loop vectors is given by

$$W = C + K = {}_CC_1 + {}_CC_3 + \ldots + {}_CC_n = 2^{C-1} \quad (2)$$

where

$$n = C - 1 \text{ if } C \text{ is even,}$$
$$n = C \quad \text{ if } C \text{ is odd.}$$

† Microwave Research Institute of Polytechnic Institute of Brooklyn, N. Y.

‡ Professor of University of Tokyo, Japan, now at Microwave Research Institute of Polytechnic Institute of Brooklyn, N. Y.

Dually, the total number V of cut-set covectors cutting the relay branch is given by

$$V = B + H = 2^{B-1} \tag{1}$$

where B and H are numbers of independent single relay-cut-set covectors and all dependent covectors.

An order relation of vectors and covectors will be defined, taking an example on the following three vectors U^κ, V^κ and W^κ.

$$\begin{aligned} U^\kappa &= [U^1\ U^2\ U^3\ U^4\ U^5\ U^6] \\ &= [1\ \ 1\ \ .\ \ 1\ \ 1\ \ 1], \\ V^\kappa &= [1\ \ .\ \ .\ \ 1\ \ .\ \ 1], \\ W^\kappa &= [.\ \ 1\ \ 1\ \ .\ \ 1\ \ 1], \end{aligned}$$

there holds

$$U^\kappa \geqq V^\kappa \text{ for all } \kappa \tag{i}$$

and an inequality holds

$$U^\kappa > V^\kappa \text{ at least for one } \kappa\ (\kappa = 2 \text{ and } 5). \tag{ii}$$

If any pair of vectors or covectors is in relationships i and ii, the vector U is called *"larger than"* V, or V is called *"smaller than"* U and is expressed as

$$\vec{U} > \vec{V} \quad \text{or} \quad U > V. \tag{iii}$$

The relation is called a *"vector order relation"* (covering, or inclusion). W has no order relation to U and V. In general, a vector set forms a *"partially ordered"* (*semi-ordered*) set.

If all possible K relay-loop vectors $C_k{}^\kappa$ are determined by odd number sums of C linearly independent relay-loop vectors $C_q{}^\kappa$, all W relay-loop vectors generally form a semi-ordered vector set. Because W vectors include all possible relay-loop vectors, if there exists a multiple loop U^κ in these vectors, its single loop part V^κ exists in the remaining vectors, and this multiple loop vector U^κ is larger than its single loop part V^κ. Thus a multiple relay-loop vector U^κ always possess a smaller relay-loop vector V^κ. This is an algebraic criterion that a relay-loop vector is multiple. Therefore, the necessity of singleness of given short circuit conditions $C_g{}^\kappa$ algebraically means the nonexistence of a smaller relay-loop vector in all K dependent relay-loop vector $C_k{}^\kappa$. Dually, a multiple relay-cut-set covector singleness of given open circuit conditions $B^f{}_\kappa$ algebraically means the nonexistence of a smaller relay-cut-set covector in all H dependent relay-cut-set covectors $B^h{}_\kappa$.

Statement of Problems

If the problem is given by short circuit conditions, one can start either from the standard sum (canonical form) S_0 or product R_0. Either is easily obtained from the other. Generally there exist certain transformations t_i of variables which keep the given sum S_0 invariant, and these transformations form a noncommutative group[15] concerning successive substitutions. Then, there is a set of transformation elements of the group called *"generators"* from which all other transformations can be generated. Also as is well known, only permutations of pairs of variables such as (xx'), (xy), (xy') or (wx) (zz') are sufficient to be considered as generators. From the total number of each literal in the standard sum and from its configuration, the group generators are determined from R_0 or S_0 by a routine process. If necessary, the multiplication table of the group can be easily made.

Change of Boolean Expressions by a General Process

From the standard sum S_0, the prime implicant S_1 and all other possible Boolean expressions S_i, including their dual product expressions $R_0, R_1, \ldots R_i$, can be algebraically obtained. Though a prime implicant happens to be a monotone function, that is, of purely unprimed literals or all primed literals, a minimum network is often obtainable from its non-monotone expression.

From the Boolean standpoint any S_i is equivalent to R_j for all values of i and j. However in topological design, the simultaneous consideration of S_i and R_j is more convenient, especially for topological enumerations. In this case for each S_i, if it has a corresponding network, the choice of R_j of the same network is desirable. For each S_i, the corresponding R_j is generally determined in this manner by examining whether the factors of R_j form a minimum necessary set of variables which should be zero in order that the S_i under consideration becomes zero. In Example 1,

$$R_i \rightleftarrows S_i \text{ for all } i.$$

However, in general the correspondence is not one to one.

A necessary condition of realization of a standard sum S_0 or G terms by single contacts is that all linearly dependent relay-loops are included in the original sum. If a set of generating loop-vectors, C in number, is realizable as a network, the above is also the sufficient condition and there holds

$$G = 2^{C-1} \tag{4}$$

Its proof is based on the exclusiveness of make and break contact literals in all terms of S_0, on the odd number in addition and on the non-existence of multiple loops. Dually for the standard product, there holds

$$F = 2^{B-1} \tag{3}$$

for a realizable case.

Topologization

A set of short circuit conditions of each of S_i is topologically a set of vectors $\overrightarrow{C_g}(i)$ in $A(i)$— dimensional affine space expressing single relay-loops and forms a branch-loop incidence matrix $C_g^{\kappa}(i)$ of $G(i)$ rows and $A(i)$ columns. Dually, a set of open circuits of each of R_j is topologically represented by a set of covectors $\overrightarrow{B^f}(j)$ in $A(j)$ — dimensional affine space expressing single relay-cut-sets and forms a cut-set matrix $B^f{}_{\kappa}(j)$ of F(j) rows and $A(j)$ columns.

Realizability by the Numbers of Nodes, Branches and Degree of Freedom

Each row of C_g^{κ} must be a single relay-loop. Then, the number $\alpha^0(i)$ of nodes must be equal to the *"topological length"* of this loop, that is, the number of $\alpha^1(i)$ of branches. Thus, the network must have at least α^0 nodes which is equal to the maximum number $E(i)$ of unities in a loop vector:

$$\alpha^0(i) \geqq E(i); \tag{6}$$

especially for the prime implicant S_1,

$$\alpha^0(i) \geqq E(1) \tag{8}$$

This is expressed also as

$$\alpha^0(i) - 1 \geqq D(i) = E(i) - 1 \tag{10}$$

where $D(i)$ is a *topological distance* of the terminals of the relay branch, that is, the maximum number of variables (contacts) in a tie.

The rank $C(i)$ of C_g^{κ} gives the degree of freedom $P^1(i)$:

$$P^1(i) = C(i) = \text{rank } (C^{\kappa}{}_g(i)) \,. \tag{12}$$

Euler's relation

$$P^1 = \alpha^1 - \alpha^0 + 1 \tag{14}$$

gives

$$C \leqq A - E + 1 \tag{16}$$

If this is not satisfied, there does not exist a circuit for this $C_g^{\kappa}(i)$.

$$N = A - E + 1 - C \tag{18}$$

gives the maximum permissible number of additional linearly independent *"pseudoties"*, (which mean topological single loops but not Boolean ties by) including make and break contacts of one relay or more in series. However, a relay-loop vector with unities in pair contact components is a single relay-loop pseudotie only when there are no smaller relay-loop vectors. It is a multiple loop vector if there is a smaller relay-loop vector. Dually, each row of $B^f{}_{\kappa}$ must be a single relay-cut-set covector. If the maximum number of contacts in each row of $B^f{}_{\kappa}$ (which can be regarded as a *topological thickness* of this barrier between two terminals of the relay) is denoted by $P(i)$, and that of branches (which may be called a *topological width* of the cut-set) is denoted by $Q(i)$, then there holds

$$P^1(i) \geqq P(i) = Q(i) - 1 \tag{9}$$

$$P^1(i) + 1 \geqq Q(i) \tag{5}$$

Especially, for a dual prime implicant R_1,

$$P^1(i) + 1 \geqq Q(1) \tag{7}$$

The proof is based on the singleness of cut-sets, and that P branches can form chords (links) of a cotree.

$$\alpha^0(i) - 1 = B(i) = \text{rank } (B^f{}_{\kappa}(i)) \,. \tag{11}$$

Euler's relation

$$\alpha^0 - 1 = \alpha^1 - P^1 \tag{13}$$

gives

$$B \leqq A - Q + 1 \tag{15}$$

If this is not satisfied, there is no circuit for this $B^f{}_{\kappa}(i)$.

$$M = A - Q + 1 - B \tag{17}$$

gives the maximum permissible number of additional linearly independent *"pseudocuts"*, which are topologically single cut-sets but not Boolean cuts, by including make and break contacts of one relay or more in parallel. However, a general cut-set covector including such pair-contacts is either a single relay-pseudocut or a *multiple cut-set*, of which the included single loop is not necessarily a pseudocut.

Base Vectors and Covectors by Semi-Diagonalization

The vector set C_g^{κ} defines an affine subspace of C-dimension. Its base vectors C_q^{κ} can be determined by transforming C_g^{κ} into such a form that each row has at least one unity which is the only one unity in its column. If all rows acquire such unities, such C_q^{κ} will be called a *"semi-diagonalized"* form. Further transformation of the first square part of C_q^{κ} to a unit matrix by adequate exchange of its rows and columns is dispensable. The semi-diagonalization is more easily done by a routine addition modulo 2 than by multiplication of an inverse of a square part. Then, all K linearly dependent vectors C_k^{κ} are obtained from the base vectors C_q^{κ} by all odd number sums. C_k^{κ} will be called *"subties"* in short. Dually, base covectors $B^a{}_{\kappa}$ and all H dependent cut-sets $B^h{}_{\kappa}$ can be determined. $B^h{}_{\kappa}$ will be called *"subcuts."*

Determination of Connections

If the given vector set C_g^{κ} exactly coincides with its base C_q^{κ} and all subties C_k^{κ}, that is, with W C_w^{κ},

$$G = C + K = W, \tag{20}$$

then the remaining part is to realize the base C_q^{κ} as a connection by its semi-diagonalization. If C_q^{κ} is not realizable and N of Eq. 18 is a positive integer, there is a possibility to realize it by the addition of pseudoties C_n^{κ} up to N. C_n^{κ} should not be smaller than any of $C_g{}^\kappa$, and all subties generated by $C_n{}^\kappa$ and the $C_q{}^\kappa$ should be either multiple loops or pseudoties. If it is still nonrealizable, other Boolean expressions of the same number of contacts should be calculated. If all of them are unrealizable, only an increase of contacts enables realization. If some of the subties $C_k{}^\kappa$ are pseudoties and the rest are all included in $C_g{}^\kappa$, the process is the same. Dually, if $B^f{}_\kappa$ coincides with $B^v{}_\kappa$, that is, $B^a{}_\kappa$ and $B^h{}_\kappa$ and

$$F = B + H = V \tag{19}$$

or $B^h{}_\kappa$ include some pseudocuts, then the realizability of $B^a{}_\kappa$ is examined and if M of Eq. 17 is positive and all subcuts by $B^m{}_\kappa$ and $B^a{}_\kappa$ are either multiple-cuts or pseudoties, the addition of $B^m{}_\kappa$ may change $B^a{}_\kappa$ to a realizable one.

The above case is the simplest. In general, not all of the subties $C_k{}^\kappa$ are included in the original $C_g{}^\kappa$, and the rest of the subties $C_k{}^\kappa$ form multiple loops and Boolean ties which are not included in $C_g{}^\kappa$. These can give a new algebraic definition of the familiar "sneak paths". Subties $C_k{}^\kappa$ may include a "smaller vector" than any vector of the given $C_g{}^\kappa$. Sneak paths and such order relations should be eliminated either by a change of i of S_i, or by an increase of contacts. Dually, the algebraically defined "sneak barriers" and smaller covectors in $B^k{}_\kappa$ should be eliminated either by a change of j of R_j or by an increase of contacts.

Example 1[16]

Decimal expression of short circuit combinations: 4, 2, 1, 0.

$$\begin{array}{llllcc}
 & & & & \text{primes} & \text{name} \\
1.\ S_0 = & x & y' & z' & 2 & 1 \\
 & \smile x' & y & z' & 2 & 2 \\
 & \smile x' & y' & z & 2 & 3 \\
 & \smile x' & y' & z' & 3 & 4 \\
 & 1.3 & 1.3 & 1.3 & &
\end{array}$$

$$\begin{array}{llllll}
\text{a} & \text{b} & \text{c} & \text{name} & & \\
 & & y'z' & 1' & R_0 = & (\text{x}\smile y'\smile z') \\
 & x'z' & & 2' & & (x'\smile y\smile z') \\
x'y' & & & 3' & & (x'\smile y'\smile z) \\
x'y' & x'z' & y'z' & 4' & & (x'\smile y'\smile z')
\end{array}$$

The combinations in R_0 are identical with those in S_0 in this example.

2. The generators of the invariant transformations are

$$t_1 = (xy) = \begin{pmatrix} xx' & yy' & zz' \\ yy' & xx' & zz' \end{pmatrix}, \quad t_2 = (xz)\ .$$

All other elements t_i are obtained from these two:

$$t_1t_1 = t_0\text{: unit element,}$$
$$(yz) = t_3 = t_1t_2t_1\ ,$$

$$t_1t_2 = \begin{pmatrix} xy \\ yx \end{pmatrix}\begin{pmatrix} xz \\ zx \end{pmatrix} = \begin{pmatrix} xyz \\ yzx \end{pmatrix} = t_4\ ,$$

$$t_2t_1 = \begin{pmatrix} xz \\ zx \end{pmatrix}\begin{pmatrix} xy \\ yx \end{pmatrix} = \begin{pmatrix} xyz \\ zxy \end{pmatrix} = t_5\ .$$

The group table is as follows.

TABLE I

THE GROUP TABLE

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	0	4	5	2	3
2	2	5	0	4	3	1
3	3	4	5	0	1	2
4	4	3	1	2	5	0
5	5	2	3	1	0	4

$$\begin{array}{lll}
3.\ F = & (a\smile 3)(b\smile 2)(c\smile 1)(a\smile b\smile c\smile 4) & \\
 = & abc & = S_1 \\
 & \smile ab1\smile ac2\smile bc3 & \smile S_2\smile S'_2\smile S''_2 \\
 & \smile a12\smile b13\smile c23 & \smile S_3\smile S'_3\smile S''_3 \\
 & \smile 1234 & \smile S_0\ .
\end{array}$$

All primed expressions are reduced to unprimed S_i by t_j:

$$t_1S'_2 = t_2S''_2 = S_2, \quad t_2S'_3 = t_3S''_3 = S_3$$

Therefore it is sufficient to consider only $S_i (i = 1, 2, 3, 0)$ in further calculation.

TABLE II

		D			D			D			D
	$S_1 = x' y'$	2		$S_2 = x' y'$	2		$S_3 = x' y'$	2		$S_0 = x\ y' z'$	3
	$\smile x' \quad z'$	2		$\smile x' \quad z'$	2		$\smile x\ y' z'$	3		$\smile x' y\ z'$	3
	$\smile \quad y' z'$	2		$\smile x\ y' z'$	3		$\smile x' y\ z'$	3		$\smile x' y' z$	3
										$\smile x' y' z'$	3
o 6	2 2 2	S_{10}	7	1 2 2 2	S_{20}	8	1 2 1 2 2	S_{30}	12	1 3 1 3 1 3	S_0
x 3	1 1 1	S_{11}									
x 4	2 1 1	S_{12}	x 4	1 1 1 1	S_{21}						
x 4	1 2 1	$t_1 \rightarrow S_{12}$									
x 4	1 1 2	$t_2 \rightarrow S_{12}$									
o 5	2 2 1	S_{13}	o 5	1 2 1 1	S_{22}	x 5	1 1 1 1 1	S_{31}			
o 5	2 1 2	$t_3 \rightarrow S_{13}$	x 5	1 1 2 1	S_{23}						
o 5	1 2 2	$t_2 \rightarrow S_{13}$	x 5	1 1 1 2	$t_3 \rightarrow S_{23}$						

o: realizable x: unrealizable

4. R_{11} and S_{11}

$$\begin{array}{llll}
 & & \text{Terms P} & \\
R_{11} = & (x' \smile y' \quad) & 2 & \\
 & (x' \quad \smile z') & 2 & \\
 & (\quad y' \smile z') & 2 & \\
 & 2 \quad 2 \quad 2 & &
\end{array}
\qquad
\begin{array}{r|ccccc}
 & x' & y' & z' & R_{11} & Q \\
 & 1 & 1 & . & 1 & 3 \\
B^f_\kappa = & 1 & . & 1 & 1 & 3 \\
 & . & 1 & 1 & 1 & 3 \\
\hline
B^h_\kappa = & . & . & . & 1 & \\
 & & & & & \text{(subcut)}
\end{array}$$

F = factors = 3, B = rank of B^f_κ = 3.
P = maximum no. of terms = 2, Q = max. no. of branches = $P + 1$ = 3.
nodes: $\alpha^0(R_{11}) = B + 1 = 4$.

If pseudocuts are added

$$\alpha^0(R_{11}) > B + 1 = 4.$$

degree of freedom $P^1(R_{11}) \geqq P = Q - 1 = 2 = 3 - 1$.
$\therefore$ Branches: $\alpha^1(R_{11}) = P^1 + \alpha^0 - 1 \geqq 2 + 4 - 1 = 5$.

$$\begin{array}{lll}
 & & \text{factors } D \\
S_{11} = & x' \ y' & 2 \\
 & \smile x' \quad z' & 2 \\
 & \smile \quad y' \ z' & 2 \\
\hline
 & 2 \quad 2 \quad 2 &
\end{array}
\qquad
\begin{array}{r|ccccc}
 & x' & y' & z' & S_1 & E \\
 & 1 & 1 & . & 1 & 3 \\
C_g^\kappa = & 1 & . & 1 & 1 & 3 \\
 & . & 1 & 1 & 1 & 3 \\
\hline
C_k^\kappa = & . & . & . & 1 &
\end{array}$$

D = max. no. of factors = 2,
E = max. no. of branches in loops = $D + 1$ = 3.
G = no. of terms = 3, C = rank of C_g^κ = 3.

$\alpha^0(S_{11}) - 1 \geqq D = E - 1 = 2$,
$\alpha^0(S_{11}) \geqq E = 3$
$P^1(S_{11}) \geqq C = 3$
$\alpha^1(S_{11}) = P^1 + \alpha^0 - 1 \geqq 3 + 3 - 1 = 5$.

Furthermore, though R_1 corresponds to S_1, their incidence matrices do not form null-factors:

$$B^f_\kappa C^\kappa_q = 1^f_q \neq 0, \quad C^\kappa_q = C_q^\kappa.$$

This shows that there is no network which simultaneously satisfies B^f_κ and C_g^κ. Also both the subcut B^h_κ and subtie C_κ^h have the order relation:

$$B^f > B^h, \quad C_g > C_k.$$

For S_{12}, the corresponding C_g^κ is given by

$$\begin{array}{r|ccccc}
 & 1x' & 2x' & y' & z' & S_{12} \\
C_g^\kappa = & 1 & a & 1 & . & 1 \\
 & . & 1 & . & 1 & 1 \\
 & . & . & 1 & 1 & 1 \\
\hline
C_k^\kappa = & 1 & a' & . & . & 1
\end{array}, \quad a' = a + 1.$$

The above C_g^κ is the most general 2 contact case of x' if a is regarded as an unknown (O or 1), as is easily proved.

$$\alpha^1 = 5, \quad \alpha^0 \geqq 4, \quad P^1 \leqq 2.$$

But $G = C = 3$.

Also the subtie C_4 forms a sneak path. Hence, there is no 4 contact realization in S_1. In S_2 obtained from the C_g^κ and C_k^κ of S_{21} one gets $\alpha^1 = 5$, $\alpha^0 \geqq 4$ $P^1 \leqq 2$s. But $G = C = 3$. Furthermore, the subtie C_4 forms a sneakpath. Hence, a single contact realization for S_2 does not exist. Thus, there is no 4 contact realization.

Cases of 5 Contacts

$$\begin{array}{r|cccccc}
 & 1x' & 2x' & 1y' & 2y' & z' & S_{13} \\
C_g^\kappa(S_{13}) = & 1 & b & 1 & d & . & 1 \\
 & a & 1 & . & . & 1 & 1 \\
 & . & . & c & 1 & 1 & 1 \\
\hline
C_k^\kappa = & a' & b' & c' & d' & . & 1
\end{array}$$

$\alpha^1 = 6$, $\alpha^0 \geqq 3$. $\therefore P^1 \leqq 4$. $G = C = 3$.

From R_i, $a^0 \geqq B + 1 = 4$.

$\therefore P^1 = a^1 - a^0 + 1 \leqq 6 - 4 + 1 = 3.$

$\therefore N = P^1 - C = 3 - 3 = 0.$

$\therefore a^0 = a^1 - P^1 + 1 = 6 - 3 + 1 = 4.$

Concerning the subtie C_4, if $a' = b' = c' = d' = 0$ or $a = b = c = d = 1$, then $C_4^\kappa = \ldots\ldots 1$ is smaller than C_g. Hence, this is unrealizable. The only other possibility is that C_4 expresses $x'y'$. Then

$$a' \smile b' = 1, \quad c' \smile d' = 1, \quad \text{or } ab = cd = 0$$

TABLE III

	a	b	c	d
Case 1	0	0	0	0
2	0	1	0	0
3	1	0	0	0
2′	0	0	0	1 → Case 2 by t_1,
3′	0	0	1	0 → Case 3 by t_1.
1′	0	1	0	1 → Case 1, e.g. $C_1 = C_4$ of case 1.
3″	0	1	1	0 → Case 3 by($1y'$, $2y'$) and t_1,
3‴	1	0	0	1 → Case 3 by ($1x'$, $2x'$).
4	1	0	1	0

Case 1

$$\begin{array}{rcccccccl}
 & 1x' & 2x' & 1y' & 2y' & z' & S_{13} & & \\
 & 1 & . & 1 & . & . & 1 & 1x' & \\
C_g^\kappa = & . & 1 & . & . & 1 & 1 & 2x' & \text{(Fig. 1)} \\
 & . & . & . & 1 & 1 & 1 & 2y' & \\
 & & & \dot{2} & & \dot{3} & \dot{1} & & \\
\hline
C_4^\kappa = & 1 & 1 & 1 & 1 & . & 1 & &
\end{array}$$

$C_4 > C_1.$

$\therefore$ C_4 is a multiple loop.

The connection is obtained from C_g^κ by the ambit method as shown in Fig. 1, because C_g^κ is already semidiagonalized.

Case 2

$$\begin{array}{rcccccc}
 & 1x' & 2x' & 1y' & 2y' & z' & S_{13} \\
 & 1 & 1 & 1 & . & . & 1 \\
C_g^\kappa = & . & 1 & . & . & 1 & 1 \\
 & . & . & . & 1 & 1 & 1 \\
C_k^\kappa = & 1 & . & 1 & 1 & . & 1
\end{array}$$

$$\begin{array}{rccccccl}
 & 1x' & 2x' & 1y' & 2y' & z' & S_{13} & \\
 & 1 & . & 1 & . & 1 & . & 1x' \\
C_g^\kappa = & . & 1 & . & . & 1 & 1 & 2x' \\
 & . & . & . & 1 & 1 & 1 & 2y' \\
 & & & \dot{3} & & \dot{2} & \dot{1} &
\end{array}$$

The circuit is obtained from semi-diagonalized C_g^κ, (see Fig. 2)

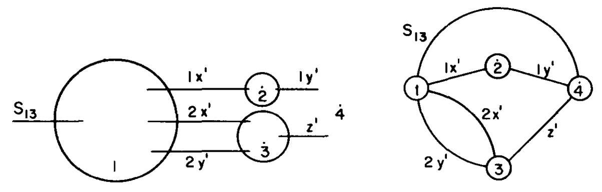


Fig. 1.

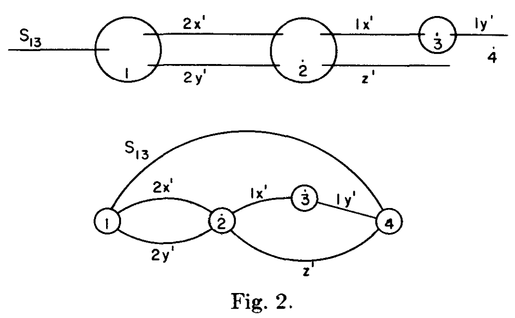


Fig. 2.

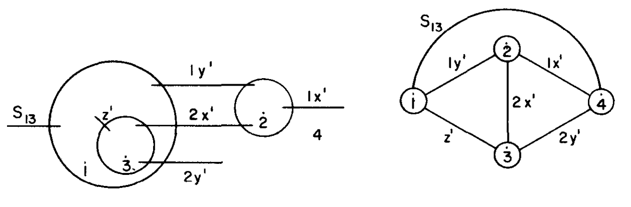


Fig. 3.

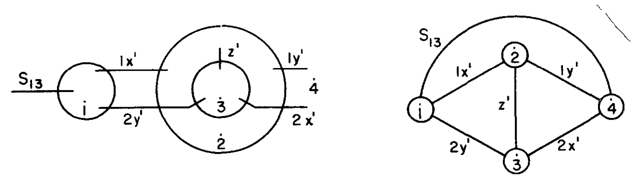


Fig. 4.

Case 3

$$\begin{array}{rccccccl}
 & 1x' & 2x' & 1y' & 2y' & z' & S_{13} & \\
 & 1 & . & 1 & . & . & 1 & 1y' \\
C_g^\kappa = & 1 & 1 & . & . & 1 & 1 & 2x' \\
 & . & . & . & 1 & 1 & 1 & 2y' \\
 & \dot{2} & & & & \dot{3} & \dot{1} & \\
C_k^\kappa = & . & 1 & 1 & 1 & . & 1 &
\end{array}$$

This gives a bridge of Fig. 3.

Case 4

$$\begin{array}{rccccccccccccl}
 & & & & & & & & & 1y' & & z' & S_{13} & \\
 & 1 & . & 1 & . & . & 1 & 1 & . & 1 & . & . & 1 & 1x' \\
C_g^\kappa = & 1 & 1 & . & . & 1 & 1 & . & 1 & 1 & . & 1 & . & 2x' \\
 & . & . & 1 & 1 & 1 & 1 & . & . & 1 & 1 & 1 & 1 & 2y' \\
 & & & & & & & & & \dot{2} & & \dot{3} & \dot{1} &
\end{array}$$

Add C_1 to C_2 in order to obtain semi-diagonalized C_g^κ.

$$C_k^\kappa = \; . \;\; 1 \;\; . \;\; 1 \;\; . \;\; 1 \;\; .$$

This yields another bridge of Fig. 4.

For S_{22}, the following case is the most general form of C_g^K based on t_3:

$$\begin{array}{cccccccc}
 & & x & 1x' & 2x' & y' & z' & S_{22} \\
 & & . & 1 & a & 1 & . & 1 \\
C_g^K & = & . & . & 1 & . & 1 & 1 \\
 & & 1 & . & . & 1 & 1 & 1 \\
C_k^K & = & 1 & 1 & a' & . & . & 1
\end{array}$$

$$a^1 = 6, \ a^0 \geqq 4, \quad \therefore P^1 \leqq 3. \quad G = C = 3.$$

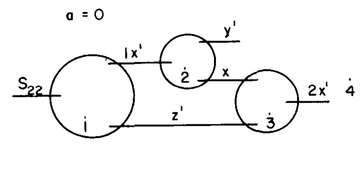
C_4^K forms a pseudotie and it cannot be a multiple loop.

$$N = P^1 - C = 3 - 3 = 0.$$

Hence, there is no room of adding pseudoties. This is semi-diagonalized to C_q^K by adding C_2 to C_3.

$$\begin{array}{ccccccccc}
 & & & & 2x' & y' & & S_{22} & \\
 & & . & 1 & a & 1 & . & 1 & 1x' \\
C_{q'}^K & = & . & . & 1 & . & 1 & 1 & z' \\
 & & 1 & . & 1 & 1 & . & . & x \\
 & & & & \dot{3} & \dot{2} & & \dot{1} &
\end{array}$$

This generates 2 kinds of bridges as shown in Figs. 5 and 6. For S_{23}, the subtie C_4 forms a sneak path xy' or x. Hence, it is not realizable.



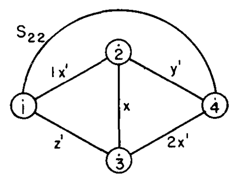
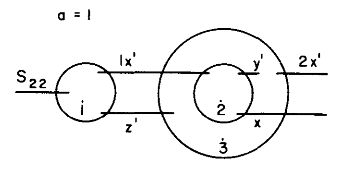


Fig 5.



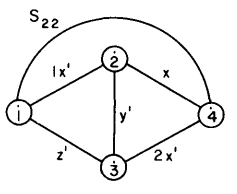


Fig. 6.

Application of t_i yields more networks and the total number is given by Table IV.[16,18]

TABLE IV

Fig. 1	2	3	4	5	6	Sum
i 1–5	1, 2=4, 3=5	1–5 1,	2=4, 3=5	1=4, 2=5, 3	1–5	
5	3	5	3	3	5	26

Example 2[17]

1. The decimal expression of short circuit combinations is

$$11, 6, 1, 0 .$$

That of open circuits is

$$13, 12, 11, 10, 8, 7, 6, 5, 3, 2, 1, 0 .$$

$$\begin{array}{llll}
S_0 = w\ x'\ y\ z & & S_1 = w\ x'\ y\ z & \\
\quad \smile w'\ x\ y\ z' & & \quad \smile w'\ x\ y\ z' & \\
\quad \smile w'\ x'\ y'\ z & & \quad \smile w'\ x'\ y' & \\
\quad \smile w'\ x'\ y'\ z' & & & \\
16 & 1\cdot3\ 1\cdot3\ 2\cdot2\ 2\cdot2 & 11 & 1\cdot2\ 1\cdot2\ 2\cdot1\ 1\cdot1 \\
R_0 = (w \smile x \smile y' \smile z) & & R_1 = (w \smile x \smile y') & \\
\quad \cdots & & \quad (w' \quad \smile y\ ) & \\
 & & \quad (w' \qquad \smile z) & \\
 & & \quad \quad (x' \smile y\ ) & \\
 & & \quad \quad (x' \qquad \smile z') & \\
48 & 5\cdot7\ 5\cdot7\ 6\cdot6\ 6\cdot6 & 11 & 1\cdot2\ 1\cdot2\ 2\cdot1\ 1\cdot1
\end{array}$$

2. The invariant transformation is only one:

$$t_1 = (wx)(zz') = \begin{pmatrix} w & w' & x & x' & y & y' & z & z' \\ x & x' & w & w' & y & y' & z' & z \end{pmatrix}$$

3. $F = S_1 \smile S_0$.

f for R_i is long, only the dual prime implicant R_1 is given above.

4. The incidence matrix for S_1 is

$$\begin{array}{cccccccccccccc}
 & & w & w' & x & x' & y & y' & z & z' & S_1 & E & \\
 & & 1 & . & . & 1 & 1 & . & 1 & . & 1 & 5 & w \\
C_g^K & = & . & 1 & 1 & . & 1 & . & . & 1 & 1 & 5 & x \\
 & & . & 1 & . & 1 & . & 1 & . & . & 1 & 4 & y' \\
C_k^K & = & 1 & . & 1 & . & . & 1 & 1 & 1 & 1 & & \\
 & & & \dot{2} & & \dot{3} & \dot{4} & & \dot{5} & \dot{6} & \dot{1} & &
\end{array}$$

The only subtie C_4 is a single loop pseudotie. Furthermore C_g^K itself forms C_q^K. Hence, if it is realizable, it forms an answer. This C_g^K is already semidiagonalized as seen in columns w, x, and y'. However, realization of ambits in the order of dotted number proves that this is not realizable as seen in Fig. 7. On the other hand, the standard sum S_0 is not realizable by single contacts as easily seen from its subties or for the reason explained in *3* (see eq. 4).

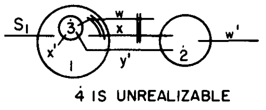


Fig. 7.

5. $E(S_1) = 5 . \quad a^0 \geqq 5 .$

$$P^1 = a^1 - a^0 + 1 \leqq 9 - 5 + 1 = 5.$$

$$G = C = 3 .$$

$$N = P^1 - C \leqq 5 - 3 = 2.$$

That is, an addition of pseudotie base vectors is possible, the maximum addition being 2.

$$Q(R_1) = 4 , \quad P^1 \geqq Q - 1 = 3 .$$

6. At first an addition of one pseudotie will be considered. In this case,

$$P^1 = G + 1 = 4 , \quad a^1 = 9 ,$$
$$a^0 = a^1 - P^1 + 1 = 9 - 4 + 1 = 6 .$$

The 8 components of the adding pseudotie form an unknown quantity:

$$C_g^{\kappa} = \begin{array}{ccccccccccc} 1 & . & . & 1 & 1 & . & 1 & . & 1 & & 1 \\ . & 1 & 1 & . & 1 & . & . & 1 & 1 & & 2 \\ . & 1 & . & 1 & . & 1 & . & . & 1 & & 3 \end{array}$$

$$C_n^{\kappa} = \begin{array}{ccccccccccc} a & b & c & d & e & f & g & h & 1 & & 4 \end{array}$$

$$C_k^{\kappa} = \begin{array}{cccccccccccccc} & & & & & & & & & & 1 & 2 & 3 & 4 \\ 1 & . & 1 & . & . & 1 & 1 & 1 & 1 & 5 & x & x & x & \\ a' & b' & c' & d' & e & f & g' & h' & 1 & 6 & x & x & & x \\ a' & b' & c & d & e' & f' & g' & h & 1 & 7 & x & & x & x \\ a & b & c' & d' & e' & f' & g & h' & 1 & 8 & & x & x & x \end{array} \quad 8 = 2^{C-1} = 2^3 .$$

$$t^1C_2 = C_1, \; t_1C_8 = C_7 .$$

(a) Pseudotie conditions of the subties C_k^{κ}.

TABLE V

Case										
1	.	.	.	.					1	
2	.	.			.	.			1	$t_1 \rightarrow 9$
3	.	.					1	.	1	$t_1 \rightarrow 10$
4	1	1	.	.	.	.			1	$\rightarrow 9$
5	1	1	.	.			.	1	1	$\rightarrow 10$
6	1	1	1	1	.	.			1	
7	1	1	1	1			.	.	1	
8	1	1	1	1	1	1			1	
9			.	.	.	.			1	
10			.	.			.	1	1	
11			1	1	.	.	.	.	1	$\rightarrow 13$
12			1	1	1	1	1	.	1	
13					.	.	.	.	1	
14	1	1			1	1	.	1	1	$t_1 \rightarrow 12$
15			.	.	1	1	.	1	1	$\rightarrow 10$
16	.	.			1	1	1	.	1	$t_1 \rightarrow 15$
17			1	1	1	1	1	.	1	$\rightarrow 12$

The blank part is perfectly arbitrary.

(b) The pseudotie condition of C_4^{κ} algebraically means that at least one pair of make and break contacts of one relay must both be unity. If this is considered in the above combinations of values of C_k^{κ}, all possible values are given as follows.

TABLE VI

Case No.											
1a		.	.	.	.	1	1	g	h	1	
	1	.	.			1	1	.		1	unr.
	2	.		.	.	1	1		1	1	unr.
	3	.	.	.		1	1	1	.	1	unr.
	4	.	.	.	.	1	1	1	1	1	unr.
1b		.	.	.	.	e	f	1	1	1	
	5	.	.	.	.		.	1	1	1	No. 1
	6		.	.	.		1	1	1	1	unr.
	7	.	.	.	.	1	.	1	1	1	No. 2
	8	.	.	.	.	1	1	1	1	1	= 4
6		1	1	1	1	.	.	g	h	1	
	9	1	1	1	1	.	.	.	.	1	No. 1 C_6
	10	1	1	1	1	.			1	1	x $C_6 < C_1$
	11	1	1	1	1	.		1	.	1	x $C_6 < C_2$
	12	1	1	1	1		.	1	1	1	x $C_6 < C_g$
7		1	1	1	1	e	f	.	.	1	
	13	1	1	1	1	.	.	.	.	1	= 9
	14	1	1	1	1		1	.	.	1	6 C_6
	15	1	1	1	1	1	.	.	.	1	No. 2 C_6
	16	1	1	1	1	1	1	.	.	1	4 C_6
8		1	1	1	1	1	1	g	h	1	
	17	1	1	1	1	1	1	.		1	= 16
	18	1	1	1	1	1	1		1	1	3 C_6
	19	1	1	1	1	1	1	1		1	2 C_6
	20	1	1	1	1	1	1	1	1	1	1 C_6
9a		1	1	.	.	.		g	h	1	
	21	1	1	.	.	.	.	.		1	3 C_7
	22	1	1	.	.	.	.	.	1	1	4 C_7
	23	1	1	.	.	.	.	1	.	1	1 C_7
	24	1	1	.	.	.	.	1	1	1	2 C_7
9b		a	b	.	.	.	.	1	1	1	
	25	.	.	.	.	.	.	1	1	1	= 5
	26	.	1	.	.		.	1	1	1	unr.
	27	1	.	.	.	.	.	1	1	1	No. 3
	28	1	1	.	.	.	.	1	1	1	= 24
10a		1	1			e	f	.	1	1	
	29	1	1	.	.	.		.	1	1	= 22
	30	1	1	.	.	.	1	.	1	1	No. 2 C_7
	31	1	1	.	.	1		.	1	1	6 C_7
	32	1	1			1	1	.	1	1	No. 1 C_7
10b		a	b		.	1	1	.	1	1	
	33				.	1	1		1	1	= 2
	34		1		.	1	1	.	1	1	No. 3 C_7
	35	1				1	1	.	1	1	26 C_7
	36	1	1			1	1	.	1	1	= 32
12		a	b	1	1	1	1	1		1	
	37	.		1	1	1	1	1	.	1	No. 1 C_8
	38		1	1	1	1	1	1		1	26 C_8
	39	1		1	1	1	1	1		1	No. 3 C_8
	40	1	1	1	1	1	1	1	.	1	= 19
13b		1	1	c	d	.	.	.	.	1	
	41	1	1	.		.	.	.	.	1	= 21
	42	1	1		1	.	.	.	.	1	No. 4 C_6
	43	1	1	1		.	.	.		1	unr.
	44	1	1	1	1	.	.	.	.	1	= 9

The ambit method can be applied to case 1a, 1b, . . . and for each no., its subties C_k^{κ} are listed in the following. Then, only 11 pseudoties are determined to be linearly independent. 4 pseudoties are realizable cases.

Subties C_k^{κ}:

TABLE VII

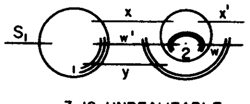
	Unrealizable Cases		Realizable Cases
1	4 . . 1 1 . 1	No. 1	4 . . . . . . 1 1 1
	6 1 1 1 1 1 1 1 1 1		6 1 1 1 1 1
	7 1 1 . . . 1 . 1		7 1 1 . . 1 1 1 1
	8 . . 1 1 . . . 1 1		8 . . 1 1 1 1 1 . 1
2	4 . . 1 1 . 1 1	No. 2	4 . . . . 1 . 1 1 1
	6 1 1 1 1 1 1 1 . 1		6 1 1 1 1 1 . . . 1
	7 1 1 . . . 1 1 1		7 1 1 . . . 1 . 1 1
	8 . . 1 1 1		8 . . 1 1 . 1 1 . 1
3	4 . . . 1 1 1 . 1	No. 3	4 1 . . . . 1 1 1
	6 1 1 1 1 1 1 . 1 1		6 . 1 1 1 1
	7 1 1 1		7 . 1 . . 1 1 . 1 1
	8 . . 1 1 . . 1 1 1		8 1 1 1 1 1 1 . 1
4	4 . . . . 1 1 1 1 1	No. 4	4 . . 1 . . . 1 1 1
	6 1 1 1 1 1 1 . . 1		6 1 1 . 1 1
	7 1 1 . . . . . 1 1		7 . . . 1 1 1 1 . 1
	8 . . 1 1 . . 1 . 1		8 1 1 1 . 1 1 . 1 1
6	4 . . . . . 1 1 1 1		
	6 1 1 1 1 . 1 . . 1		
	7 1 1 . . 1 . . 1 1		
	8 . . 1 1 1 . 1 . 1		
26	4 . 1 . . . . 1 1 1		
	6 1 . 1 1 . . . 1		
	7 1 . . . 1 1 . 1 1		
	8 . 1 1 1 1 1 1 . 1		
43	4 1 1 1 . . : . . 1		
	6 . . . 1 . . 1 1 1		
	7 . . 1 . 1 1 1 . 1		
	8 1 1 . 1 1 1 . 1 1		

Case 1a

$$C_q^{\kappa} = \begin{array}{ccccccccccccccccccc}
 &&&&&&&&& &&&x'&&y'&z&z'&S_1& \\
1&.&.&1&1&.&1&.&1& 1&.&.&1&.&1&g'&h&.&w \\
.&1&1&.&1&.&.&1&1& .&.&1&1&.&.&g&h'&1&x \\
.&1&.&1&.&1&.&.&1& .&1&.&1&.&1&.&.&1&w' \\
.&.&.&.&1&1&g&h&1& .&.&.&.&1&1&g&h&1&y \\
 &&&&&&&&& &&&\dot{2}&&\dot{3}&\dot{4}&\dot{5}&\dot{1}&
\end{array}$$

Add C_3 to C_2, then add C_4 to C_1 and C_2.

3 is unrealizable independent of g and h as seen from Fig. 8.



3 IS UNREALIZABLE

Fig. 8.

Case 1b

$$C_q^{\kappa} = \begin{array}{ccccccccccccccccccc}
 &&&&&&&&& &&&x'&y&y'&z&&S_1& \\
1&.&.&1&1&.&1&.&1& 1&.&.&1&1&.&1&.&1&w \\
.&1&1&.&1&.&.&1&1& .&.&1&1&e'&f'&1&.&1&x \\
.&1&.&1&.&1&.&.&1& .&1&.&1&.&1&.&.&1&w' \\
.&.&.&.&e&f&1&1&1& .&.&.&.&e&f&1&1&1&z' \\
 &&&&&&&&& &&&\dot{2}&\dot{4}&\dot{5}&\dot{3}&&\dot{1}&
\end{array}$$

This is semi-diagonalized by adding C_3 and C_4 to C_2.

See Figs. 9 and 10.

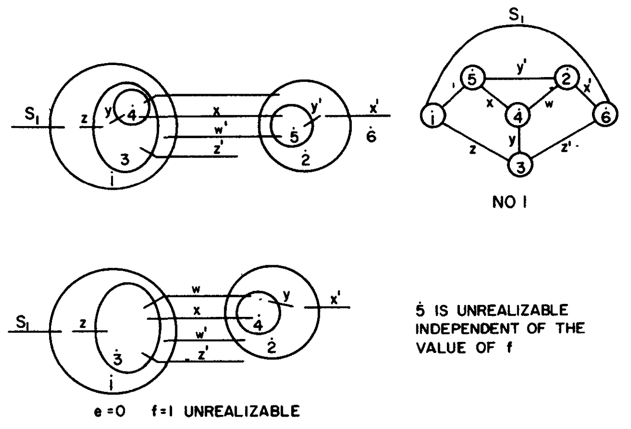


Fig. 9.

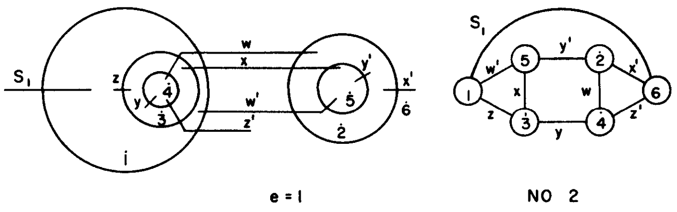


Fig. 10.

ef = . . realizable No. 1
  = . 1 unrealizable as seen from Fig. 9
  = 1 . realizable No. 2
  = 1 1 unrealizable as seen from Fig. 10

Case 6

$$C_k^{\kappa} = \begin{array}{cccccccccc}
4&1&1&1&1&.&.&g&h&1 \\
6&.&.&.&.&.&.&g'&h'&1 \\
7&.&.&1&1&1&1&g'&h&1 \\
8&1&1&.&.&1&1&g&h'&1
\end{array}
\qquad
\begin{array}{cccl}
gh =&.&.&\text{No. } 1 \\
&.&1&C_6 < C_1 \\
&1&.&C_6 < C_2 \\
&1&1&C_6 < C_g
\end{array}$$

No. 9 of Case 6 is identical to No. 1. C_6. All other 3 are smaller than one of C_g^{κ}. Hence, it is unrealizable.

Case 9b

	a	b
Only No. 26	.	1
No. 27	1	.

are unknown.

$$C_q^{\kappa} = \begin{array}{ccccccccccccccccccc}
 &&&&&&&&& w&w'&&&y&&z&&S'& \\
1&.&.&1&1&.&1&.&1& 1&1&.&1&1&.&1&.&1&x' \\
.&1&1&.&1&.&.&1&1& a&b'&1&.&1&.&1&.&.&x \\
.&1&.&1&.&1&.&.&1& 1&1&.&.&1&1&1&.&.&y' \\
a&b&.&.&.&.&1&1&1& a&b&.&.&.&.&1&1&1&z' \\
 &&&&&&&&& \dot{4}&\dot{5}&&&\dot{3}&&\dot{2}&&\dot{1}&
\end{array}$$

This is semi-diagonalized by adding C_1 to C_3 and C_4 to C_2.

The ambit 5 is unrealizable for No. 26 as seen from Fig. 11.

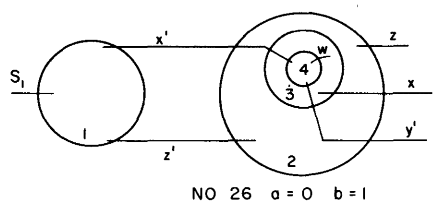


Fig. 11.

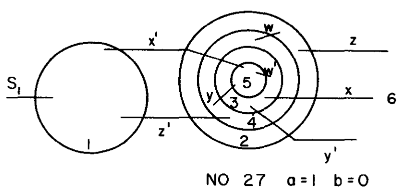
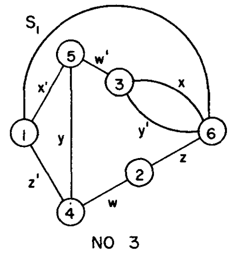


Fig. 12.



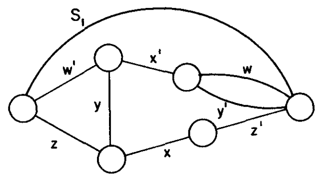


Fig. 13.

No. 27 is realizable and gives No. 3 of the solutions as seen in Fig. 12.

The first two connections Nos. 1 and 2 of Figs. 10 and 12 do not change by the transformation t_1. However, No. 3 changes to No. 4 of Fig. 13 by t_1. $C_4{}^\kappa$ of No. 4 is obtained from $C_4{}^\kappa$ of No. 3 by t_1.

Case 13b

No. 43

$$
C_q{}^\kappa = \begin{array}{cccccccccccccccccccc}
 & & & & & & & & & & w' & x & x' & y & & & & S & \\
1 & . & . & 1 & 1 & . & 1 & . & 1 & . & 1 & 1 & 1 & 1 & . & 1 & . & . & z \\
. & 1 & 1 & . & 1 & . & . & 1 & 1 & . & 1 & 1 & . & 1 & . & . & 1 & 1 & z' \\
. & 1 & . & 1 & . & 1 & . & . & 1 & . & 1 & . & 1 & . & 1 & . & . & 1 & y' \\
1 & 1 & 1 & . & . & . & . & . & 1 & 1 & 1 & 1 & . & . & . & . & . & 1 & w \\
 & & & & & & & & & & \dot{2} & \dot{3} & \dot{4} & \dot{5} & & & & \dot{1} &
\end{array}
$$

This is semi-diagonalized by adding C_4 to C_1.

In Fig. 14, the ambit $\dot{4}$ is unrealizable.

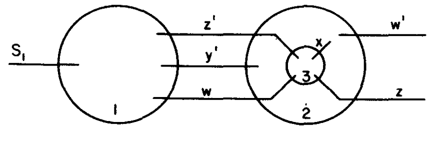


Fig. 14.

The algebraic ambit-realization is as follows.

Case 1b: $ef = \ . \ .$ (Fig. 9, No. 1)

$\dot{1}$: $(S\,w\,w'\,x\,z')\,S\,w\,w'\,x\,z'$

$\dot{2}$: $(S\,w\,w'\,x\,z')\,(w\,w'\,x\,x')\,x'\,S\,z'$
$= (S\,w\,w'\,x\,z')\,(w\,w'\,x\,x')\,S\,x'\,z'$

$\dot{3}$: $(S\,w'\,(w\,x\,z\,z')\,z)\,(w\,w'\,x\,x')\,S\,x'\,z'$
$= (S\,w'\,z\,(w\,x\,z\,z'))\,(w\,w'\,x\,x')\,S\,x'\,z'$

$\dot{4}$: $(S\,w'\,z\,((w\,x\,y)\,y\,z\,z'))\,(w\,w'\,x\,x')\,S\,x'\,z'$

$\dot{4}'$: $(S\,w'\,z\,(w\,x\,z\,z'))\,((w\,x\,y)\,w'\,x'\,y)\,S\,x'\,z'$

$\dot{5}$ (possible only from 4):

$(S\,w'\,z\,((w\,x\,y)\,y\,z\,z'))\,((w'x\,y')\,w\,x'\,y')\,S\,x'\,z'$
$(S\,w'\,z)\,(w\,x'\,y')\,(y\,z\,z')\,(w\,x\,y)\,(w'\,x\,y')\,(S\,x'\,z')\,.$

The coboundaries of the individual nodes are contained in the parenthesis.

Case 10a, No. 30, x, x', z and z' are semi-diagonalized by adding C_3 to C_1, C_4 to C_3. (Fig. 10, No. 2)

$\dot{1}$: $(S\,x'\,z')\,S\,x'\,z'$

$\dot{2}$: $(S\,x'\,z')\,S\,x'\,(z'\,w\,x\,z)\,w\,x\,z$
$= (S\,x'\,z')\,(w\,x\,z\,z')\,S\,w\,x\,x'\,z$

$\dot{3}$: $(S\,x'\,z')\,((w\,x\,z\,z')\,w\,x\,x'\,w')\,S\,z\,w'$

$\dot{4}$: $(S\,x'\,z')\,((w\,y\,z'\,(x\,y\,z))\,w\,w'\,x\,x')\,S\,w'\,z$
$= (S\,x'\,z')\,(((x\,y\,z)\,w\,y\,z')\,w\,w'\,x\,x')\,S\,w'\,z$

$\dot{5}$: $(S\,x'\,z')((((x\,y\,z)\,w\,y\,z')\,w\,x'\,y')\,w'\,x\,y')\,S\,w'\,z$
$(S\,x'\,z')\,(w\,y\,z')\,(w'\,x\,y')\,(x\,y\,z)\,(w\,x'\,y')\,(S\,w'\,z)\,.$

7. The addition of 2 pseudoties in S_1.

					1	1	.	.	1	1	.	1	.	1
			$C_q^K \rightarrow$		2	.	1	1	.	1	.	.	1	1
			C_k^K		3	.	1	.	1	.	1	.	.	1
			$\downarrow$		4	a	b	c	d	e	f	g	h	1
1	2	3	4	5	5	m	n	p	q	r	s	t	u	1
x	x	x			6	1	.	1	.	.	1	1	1	1
x	x		x		7	a'	b'	c'	d'	e	f	g'	h'	1
x		x	x		8	a'	b'	c	d	e'	f'	g'	h	1
	x	x	x		9	a	b	c'	d'	e'	f'	g	h'	1
x	x			x	10	m'	n'	p'	q'	r	s	t'	u'	1
x		x		x	11	m'	n'	p	q	r'	s'	t'	u	1
	x	x		x	12	m	n	p'	q'	r'	s'	t	u'	1
x			x	x	13	$1+a+m$	$b+n$	$c+p$	$1+d+q$	$1+e+r$	$f+s$	$1+g+t$	$h+u$	1
	x		x	x	14	$a+m$	$1+b+n$	$1+c+p$	$d+q$	$1+e+r$	$f+s$	$g+t$	$1+h+u$	1
		x	x	x	15	$a+m$	$1+b+n$	$c+p$	$1+d+q$	$e+r$	$1+f+s$	$g+t$	$h+u$	1
x	x	x	x	x	16	$1+a+m$	$b+n$	$1+c+p$	$d+q$	$e+r$	$1+f+s$	$1+g+t$	$1+h+u$	1

The base vectors C_q^K and the subties C_k^K must satisfy the following conditions:

(1) Vectors C_4 and C_5 are added and all subties C_k are either pseudoties or multiple loops.

(2) The above 12 vectors C_4, C_5 and C_k are not smaller than any of the 3 original short circuit relay-loop vectors C_1, C_2 and C_3. These conditions are necessary for realization.

The minimum number of branches in a multiple loop can be determined by the following reasoning. The minimum number of branches in C_q^K is 4 (in C_3). However, a pseudotie can have only 3 branches as in C_4 of No. 1 of Fig. 9. On the other hand, the minimum number of contact branches of a local loop consisting only of contact branches is 2 as xy' in No. 3 of Fig. 12. Therefore, in this example, the minimum number of total branches of a multiple loop is $4 + 2 = 6$.

Since the number of variables is 4, every multiple loop has at least one pair of make and break contacts of one relay, that is, all multiple loops will be included in the subties C_k^K if all vectors satisfying algebraic pseudotie conditions are considered.

In *6*, all possible vectors satisfying the algebraic pseudotie condition are already obtained for single vectors. Therefore, if any pair of these vectors further satisfies the algebraic pseudotie condition for C_{13}, C_{14}, C_{15} and C_{16}, and the resulting base vectors C_q^K are realizable, then a network of solution can be obtained. For this purpose, the following vectors were chosen, one from each of 11 sets of the subties of Table VII.

TABLE VIII

		w	w'	x	x'	y	y'	z	z'	S_1
1	No. 3–4	1	.	.	.	.	.	1	1	1
2	26–4	.	1	.	.	.	.	1	1	1
3	No. 4–4	.	.	1	.	.	.	1	1	1
4	43–6	.	.	.	1	.	.	1	1	1
5	No. 2–4	.	.	.	.	1	.	1	1	1
6	6–4	.	.	.	.	.	1	1	1	1
7	No. 1–4	.	.	.	.	.	.	1	1	1
8	2–7	1	1	.	.	.	.	1	1	1
9	3–8	.	.	1	1	.	.	1	1	1
10	4–4	.	.	.	.	1	1	1	1	1
11	1–6	1	1	1	1	1	1	1	1	1

Out of ${}_{11}C_2 = 55$ combinations, it is not so difficult to find all possible combinations for which the new 4 subties $C_{13} - C_{16}$ satisfy the algebraic pseudotie condition. C_1C_3, C_1C_6 and C_3C_6 are the only possible combinations, and these build pseudoties at wx, wy' and xy', respectively.

wx, 1, 3:

$$C_q^K = \begin{array}{cccccccccccccccccccc}
 & & & & & & & & & & & x & & y & & & z' & S_1 & \\
1 & . & . & 1 & 1 & . & 1 & . & 1 & 1 & . & 1 & . & . & . & . & . & . & w \\
. & 1 & 1 & . & 1 & . & . & 1 & 1 & . & 1 & 1 & . & 1 & . & . & 1 & 1 & w' \\
. & 1 & . & 1 & . & 1 & . & . & 1 & . & . & 1 & . & . & 1 & . & . & . & y' \\
1 & . & . & . & . & . & 1 & 1 & 1 & . & . & . & 1 & 1 & . & . & 1 & . & x' \\
. & . & 1 & . & . & . & 1 & 1 & 1 & . & . & 1 & . & . & . & 1 & 1 & 1 & z \\
 & & & & & & & & & & & . & & & . & & . & . &
\end{array}$$

Semi-diagonalization: 2 4 3 1

1. Add C_1 to C_4, C_2 to C_3.
2. Add C_4 to C_1 and C_3.
3. Add C_5 to C_1.

This is realized in Fig. 15 as No. 5.

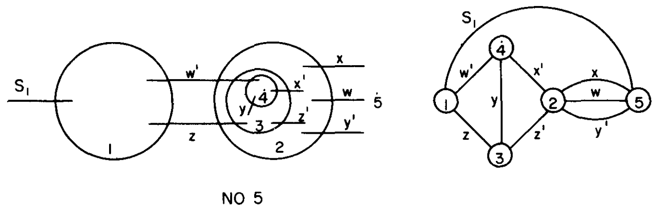


Fig. 15.

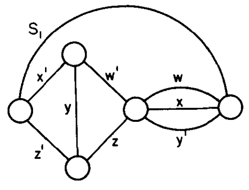


Fig. 16.

The other two: wy': 1, 6 and xy': 3, 6 generate only the networks of the same type. t_1 changes No. 5 to No. 6 of Fig. 16. Thus, these 6 networks are the only possible single contact realizations.

Acknowledgments

This research was sponsored by the Office of Scientific Research under Contract No. AF-18(600)-1505.

The authors extend their thanks to Dr. E. F. Moore, Dr. T. H. Crowley of B. T. L., N. J., and to Dr. C. F. Hobbs of C. R. C., Mass., for their kind information of recent publications.

References

[1] G. R. Kirchhoff, tr. by O'Toole: "On the Solution of the Equations Obtained from the Investigation of the Linear Distribution of Galvanic Currents," *Pogg. Ann. Phys. Chem.* 72(1847) 497–508 = Coll. W. 22–33 (briefly : 64(1845) 497–514 = Coll. W. 1–22) Trans, IRE PGCT-5.1 (March 1958) 4–7.

[2] O. Veblen, "Analysis Situs (1916) Cambridge Colloqium," (1922, 2. ed. 1931) *AMSCP.* 5, 2.

[3] Gr. C. Moisil, "Sur la théorie algébrique de certains circuits électriques, J. de Math. pur et appl." 9.36 *Fasc.* 4(1957) 313–24 and many others.

[4] S. Okada, "On a Theory of Relay Circuits," lecture note at Nippon Elec. Co. Tokyo (May 22, 1939) pp. 24 in Jap.

S. Okada, "Preliminaries on Network Theory," *J.I.E. Com Eng. Japan* 27.12 (Dec. 1943) 9–22 in Jap.

M. Hanzawa, "Theory of Relay Networks," (18) *Nichiden Geppo* 19.4 (April 1942) 10–7. in Jap.

T. Kojima, "Introduction of Automatic Exchange System," (1948) Kagaku-Shinkohsha, 239–241 in Jap.

S. Okada, "Topology Applied to Switching Circuits," Proc. Symp. Information Networks, PIB (April 1954) 267–90. This will be referred as O-tasc.

S. Okada, "A Topological Synthesis of Switching 2-Terminals," Res. Rep. R-756-59 (July 1959) MRI, PIB.

K. P. Young, "Analysis and Synthesis of 2-Terminal Contact Networks by Algebraic Topology and Combinatorial Analysis," Master's Thesis, E. E. (Sept. 1959) in prep. as Res. Rep. R-779-59, MRI, PIB. Also see R-790-59.

[5] L. Lund, "Koplings muligheter," *Norsk Mat. Tids.* 31(1949) 9

S. Seshu, "On Electrical Circuits and Switching Circuits," Trans. IRE PGCT-3 . 3 (Sept. 1956) 172–8.

J. P. Roth, "Combinatorial Topological Methods in the Synthesis of Switching Circuits," IBM Res. Rep. RC-11 (June 29, 1957) Poughkeepsie, N. Y.

J. P. Roth, "Algebraic Topological Methods for the Synthesis of Switching System I," Trans. AMS. 88. 2 (July 1958) 301–26.

J. P. Roth, "Combinatorial Topological Methods in the Synthesis of Switching Circuits," Proc. Symp. Switching (April 2, 1957) in printing.

J. P. Roth, "Algebraic Topological Methods for the Synthesis of Switching Systems, IV, Minimization of Singular Boolean Trees," Meeting of AMS, Detroit, Michigan (Nov. 27–8, 1958).

O. Wing, "The Path Matrix and the Realization of its Associated Graph," Doctor Thesis of Eng. Sc. (May 1959) Columbia Univ. N. Y.

O. Wing and W. H. Kim, "The Path Matrix and Switching Functions," *J. Frankl. I.* 268.4 (No. 1606) (Oct. 1959) 251–69.

L. Löfgren, "Irredundant and Redundant Boolean Branch Networks," Trans. IRE PGCT-6. Spec. Suppl. (May 1959) 158–75.

L. Löfgren, "Solution to the Realizability Problem for Irredundant Boolean Branch-Networks,"*J.Frankl. I.* 268.5 (No. 1607) (Nov. 1959) 352–77.

U. L. Vasilev, "Minimum Contact Networks for Boolean Functions of Four Variables," Doklady Acad. Nauk. USSR 127.2 (June 1959) 242–5.

[6] R. Gould, "The Application of Graph Theory to the Synthesis of Contact Networks," Ph. Doctor Thesis (May 1957) Harvard Univ. Cambridge. Proc. Symp. Switching (April 3, 1957) Harvard Univ. in printing.

R. Gould, "Graphs and Vector Spaces," *J. Math. Phys.* 37.3 (Oct. 1958) 193–214.

R. Gould, "A Note on Contact Networks for Switching Functions of Four Variables," Trans. IRE PGEC-7.2 (Sept. 1958) 196–9.

[7] Staff of Comp. Lab., "Synthesis of Electronic Computing and Control Circuits," (1951) Annals of Comp. Lab. 27, Harvard Univ. Cambridge.

R. A. Higonnet and R. A. Gréa, "Logical Design of Electric Circuits," (1958) McGraw Hill, N. Y. Orig. in French (1955) Berger. Lerault, Paris.

[8] B. J. Dasher, "Semantics and Kirchhoff's Current Law," Proc. IRE 47.6 (June 1959) 1158–9.

E. A. Guillemin, "How to Grow Your Own Trees from Given Cut-Set or Tie-Set Matrices," Trans. IRE PGCT -6 Spec. Suppl. (May 1959) 110–26.

R B. Ash and W. H. Kim, "On Realizability of a Circuit Matrix," Trans. IRE PGCT-6.2 (June 1959) 219–23.

W. T. Tutte, "A Homotopy Theorem for Matroids," Trans. AMS.88 (May 1958) 144–60, 161–74.

W. T. Tutte, "Matroids and Graphs," Trans. AMS. 90.3 (March 1959) 527–52.

L. Anslandei and H. M. Trent, "Incident Matrices and Linear Graphs," *J. Math. and Mech.* 8.5 (Sept. 1959) 827–35.

[9] S. Okada, "On the Fundamental Equations of the Networks," *Nippon Elec. Com. Eng. 14* (Dec. 1938) 504–8, Tokyo.

[10] S. Okada, "Algebraic and Topological Foundations of Network Synthesis"Proc. Symp. Mod. Network Synth. (April 13–5, 1955) PIB 283–322.

S. Okada, and R. Onodera, "Theory of Interlinked Electromagnetic Networks and Fields in the Tensor Geometry of Linear Space, et.," in K. Kondo, "Memoirs of the Unifying Study of the Basic Problems in Eng. Sc. by Means of Geometry," Vol. 1 (1955) Gakujutsu Bunken Fukyu-kai, in *Tokyo Kogyo Daigaku*, Oh-Okayama, Meguro-ku, Tokyo, 1–112.

S. Okada, "Network Topology in N-dimensional Geometry," lec. note at Columbia Univ. (Dec. 16, 1958) Mimeogr.

[11] H. Weyl, "Repartición de Corriente en una red conductora," *Rev. Mat. Hisp.-Amer.* 5(1923) (1) 153–64 (3): 241–9.

W. Cauer, "Synthesis of Linear Com. Networks," (2ed. 1958) McGraw Hill, N. Y. p. 90-3.

[12] G. Kron, "Non-Riemamsian Dynamics of Rotating Electrical Machinery," *J. Math. Phys.* 13.2 (May 1934) 103-94.
G. Kron, "Tensors for Circuits," (1959) Dover, N. Y. (This includes a complete list of his publications.)

[13] = 4: O-tasc. 267, 275, 279.

[14] e.g. H. Whitney, "Geometric Integration Theory," (1957) Princeton Math. S. 21.
J. A. Schouten, "Ricci-Calculus" (1954) Springer, Berlin.

[15] R. L. Ashenhurst, "A Method for Determining Functional Invariance, Theory of Switching," Rep. BL-2, (April 1953) Harvard Comp. Lab.
E. J. McCluskey, Jr., "Detection of Group Invariance or Total Symmetry of a Boolean Function," B.S.T.J. 35.6 (Nov 1956), 1445-53. Mon. 2720.

[16] = Reference 7, Hig. 191, IV_2.

[17] = Reference 7, Hig. 197, H21.

[18] See Reference 5, Vasilev.

DISCUSSION

H. G. Boehm (IBM): Can your method be programmed on a computer?

Mr. Okada: Yes, we are considering it, but as yet we have not made a program of it. If the number of variables increases, calculations by these computers seems to be the only possible method.

Mr. Boehm: How long would it take to prove Moore's table as minimal?

Mr. Okada: I don't know. Unfortunately we could not finish. The proof is not so difficult, but to exhaust all possible minimum solutions, which means determination of all possible connections would take a long time. We are now doing research on this particular program.

G. G. Murray (RCA): Existing minimization methods are limited to series-parallel nets. However, these methods are quite useful. The method described in the present paper presumably gives a true minimum but is it a practical procedure? How many Boolean variables can be handled?

Mr. Okada: As soon as the number of variables increases, the juxtaposition of all possible Boolean functions is not so easy, but with a computer will be possible.

S. Sharin (RCA): How can this method be used to advantage to design code translators? Can the logic element used — relays, diode circuitry, and so forth — be introduced as a factor in getting a direct solution?

Mr. Okada: As a general principle this principle is applicable for design of any switching circuit. As I mentioned at the beginning this method can also be used to design series-parallel connections.

K. Enslein (Brooks Research): How does this new method differ from that which resulted in the Harvard tables?

Mr. Okada: According to my knowledge, the late Dr. Roderick Gould gave the best results, especially in the improvement of Dr. Moore's table. But Dr. Gould's method was not mathematically rigorous, and in most cases he gave only one solution; but, as I showed, all possible solutions can be given by our method. Our ambit method of realizing connections from a cut-set matrix is the dual of Gould's method of realization from loop matrices, which needs three kinds of operations: parenthesis, brackets and braces. In our algebraic ambit method, one kind of operation is sufficient as shown at the end of Paragraph 6 of Example 2. Also, we gave the maximum permissible number of pseudocuts or pseudoties which can be added, as shown in Paragraph 5 of Example 2. Generally, we use tie-set matrices and cut-set matricies simultaneously. We are planning to publish a paper on the "ambit realization."

Applications of Boolean Matrices to the Analysis of Flow Diagrams

REESE T. PROSSER†

Introduction

Any SERIOUS attempt at automatic programming of large-scale digital computing machines must provide for some sort of analysis of program structure. Questions concerning order of operations, location and disposition of transfers, identification of subroutines, internal consistency, redundancy and equivalence, all involve a knowledge of the structure of the program under study, and must be handled effectively by any automatic programming system.

The structure of a program is usually determined by detailed specifications describing the program, and may usually be given a convenient geometric representation by means of flow diagrams. Ordinarily, neither of these forms is immediately adaptable for handling by machine, and for this purpose another representation of the same information must be found. Such a representation should certainly have these properties:

(1) It should be easy to construct and reproduce.
(2) It should be adaptable to handling by machine.
(3) It should contain all of the information provided by the topology of the flow diagram.

The Connectivity Matrix

A representation which has all these properties may be given by means of Boolean matrices. By a Boolean matrix we mean a matrix whose entries consist entirely of 0's and 1's. The representation is constructed as follows: Suppose we are given the structure of a program, say in the form of a flow diagram consisting of boxes, representing program operations, connected by directed line segments, representing the program flow. We are interested only in the structure, or connectivity, of this diagram, and not in the properties of the individual boxes. We make no restrictions at all on the connectivity and, in particular, branches and loops of all kinds are admissible. We begin by numbering the boxes of the diagram, say from 1 to n, in any convenient manner whatever. For later convenience we adjoin to the diagram a box numbered 0 as the initial, or input, position and a box numbered $n+1$ as the final, or output, position of the diagram. We next construct an $(n+2)\times(n+2)$ Boolean matrix, $A=(a_{ij})$, called the *connectivity matrix* associated with the diagram, by stipulating that $a_{ij}=1$ if the diagram contains a directed line segment leading directly from box i to box j, and $a_{ij}=0$ otherwise. Thus $a_{ij}=1$ if box i may be followed *immediately* by box j in the program, and 0 otherwise.

It is evident that this matrix is easy to construct and easy to handle. It is determined uniquely by the diagram, up to a permutation of the entries due to a renumbering of the boxes, and in turn it determines the diagram, in the sense that the diagram may be completely reconstructed from the matrix. Thus it meets all of our requirements.

This idea is certainly not new. Boolean matrices have been used extensively to study the connectivity and orientation of graphs [7], [12]; networks [4], [6]; organization and group dynamics problems [8]; and more generally, finite Markov processes [11]. Shannon [13] has pointed out that every flow diagram is essentially a finite Markov process, so that we have here a very special case of [11]. On the other hand it is worth emphasizing how well this idea adapts itself to program analysis. A similar attempt with a somewhat different viewpoint appears in [14].

Analysis

Certain elementary computations on the connectivity matrix yield detailed information on the program flow. To show how this comes about, we define a one-row matrix

$$e_i = (0, 0, \ldots, 1, \ldots, 0)$$

with 1 in the ith place and 0's elsewhere. Then, from the definition of A, we see that the matrix product e_iA is a one-row matrix which has 1 in the jth column if it is possible to proceed from box i to box j in one step, and 0 otherwise. By repeating this argument, we see that the *product* $e_iA^2=(e_iA)A$ is a one-row matrix whose jth column is 1 (or more) if it is possible to proceed from box i to box j in *exactly two steps*, and 0 otherwise. A similar interpretation may evidently be given to higher powers of A.

Now A^2 need not be a Boolean matrix. But it is clear that for our purpose we lose nothing if we replace all *non-zero* entries in A^2 with 1's. This amounts to multiplying A by A according to the following rule: *The Boolean product* $A \vee B$ *of the Boolean matrices* A *and* B *is that Boolean matrix whose i-j entry is*

$$\bigvee_k (a_{ik} \wedge b_{kj})$$

Here $\vee$ and $\wedge$ denote the Boolean operations of max

* The work reported in this paper was performed at Lincoln Laboratory, center for research operated by M.I.T. with the joint support of the U. S. Army, Navy and Air Force.

† Massachusetts Institute of Technology Lincoln Laboratory, Lexington, Mass.

and min, respectively. In the same spirit we define: *The Boolean sum $A \wedge B$ of the Boolean matrices A and B is that matrix whose i-j entry is $a_{ij} \vee b_{ij}$.* Thus Boolean sums and products of Boolean matrices are formed in the same way as ordinary matrix sums and products, except that $+$ is replaced by $\vee$ and $\times$ by $\wedge$.

Now the way is clear for induction. Let A be the connectivity matrix of a flow diagram, and define

$$A_m = A_{m-1} \wedge A = A \wedge A \wedge \ldots\ldots\ldots \wedge A \quad m \text{ times}$$
$$B_m = B_{m-1} \vee A_m = A_1 \vee A_2 \vee \ldots\ldots\ldots \vee A_m$$

Theorem 1

The i-j entry of A_m is 1 if it is possible to proceed from box i to box j in exactly m steps, and 0 otherwise. The i-j entry of B_m is 1 if it is possible to proceed from box i to box j in at most m steps, and 0 otherwise.

Proof: For $m = 1$, both statements reduce to definitions. Now suppose both statements hold for $m = r$; and consider the case $m = r + 1$. The i-j entry of A_{r+1} is just $\bigvee_k(c_{ik} \wedge a_{kj})$ where c_{ik} denotes the i-k entry of A_r. This is *zero*, unless for some k we have $c_{ik} = a_{kj} = 1$. But this means that it is possible to proceed from box i to box k in exactly r steps, and from box k to box j in exactly one step. Thus the i-j entry of A_{r+1} is 0 unless it is possible to proceed from box i to box j in exactly $r + 1$ steps. The second statement follows immediately from the first.

Theorem 2

The limit $\lim B_m$ as $m \to \infty$ exists as a Boolean matrix, which we denote by B. Moreover, we have $B = B_m$ for all $m \geq p$, where p is the length of the longest open path in the diagram.

Proof: Since the entries of B_m are monotone increasing with m, it is clear that $\lim B_m$ as $m \to \infty$ exists and forms a Boolean matrix. The second statement follows from the observation that if it is possible to proceed from box i to box j at all, it is possible to do so along an open path (*i.e.*, one containing no loops), and hence in less than $p + 1$ steps. Thus if the i-j entry of B_m is 1 for *any* m, it is 1 for $m = p$. This means that $B_m = B_p$ whenever $m \geq p$.

Theorem 3

The i-j entry of B is 1 if it is possible to proceed from box i to box j in any number of steps, and 0 otherwise.

Proof: This follows immediately from the proof of Theorem 2.

The matrix B is obviously computable by machine from the matrix A, and since only Boolean operations are involved, the time required for this computation is not prohibitive even for fairly large n. On the other hand, it follows from Theorem 3 that the matrix B contains detailed information about the consistency of the flow diagram. We cite some obvious examples:

(1) It is possible to get from the input to box i only if $b_{0i} = 1$. Thus if there are no spurious boxes, the top row of B must contain all 1's (except for b_{00}).

(2) It is possible to get from box i to the output only if $b_{i(n+1)} = 1$. Thus if there are no boxes without exits, the last column of B must contain all 1's (except for $b_{(n+1)\ (n+1)}$).

(3) It is possible to get from box i to box i only if $b_{ii} = 1$. Thus if there are no loops in the program, the main diagonal of B must contain all 0's. Boxes involved in loops are represented by 1's on this diagonal.

(4) After leaving box i, it is possible to go through box j only if $b_{ij} = 1$. Now if we alter box i then only those boxes following box i in the program will be affected. These boxes are represented by 1's in the ith *row* of B.

(5) If the matrix decomposes into relatively independent submatrices, then the program decomposes into relatively independent subprograms. Thus it may be possible to identify natural subprograms directly from the form of the matrix B.

Examples

The foregoing theory will be further illuminated by application to concrete problems. As a first example we choose a flow diagram containing an obvious inconsistency, and show how this inconsistency is

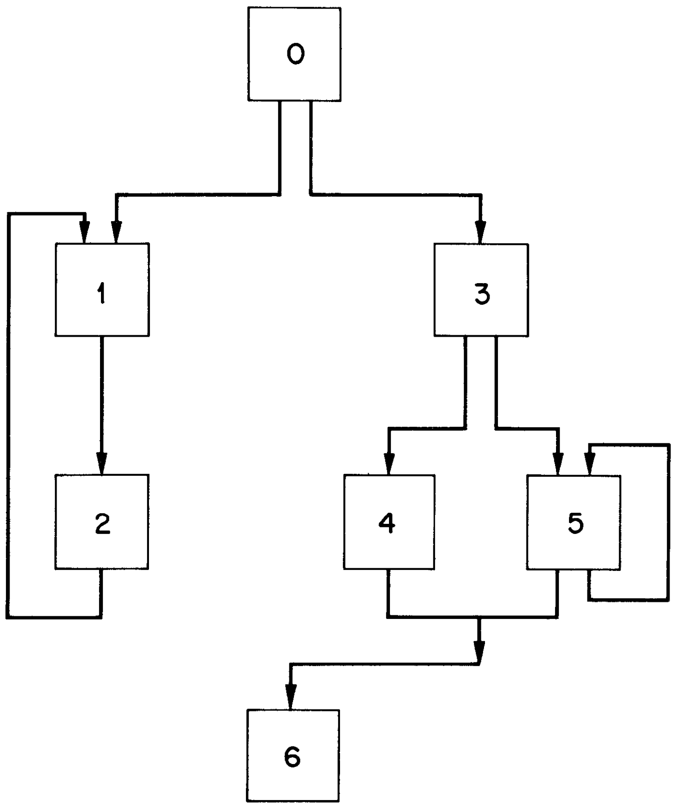


Fig. 1.

reflected in the matrix B. The diagram is shown in Fig. 1. Here the boxes are already numbered, including the input and output boxes. The connectivity matrix for this diagram is a 7×7 matrix, whose entries are

$$A = \begin{matrix} 010\ 100\ 0 \\ 001\ 000\ 0 \\ 010\ 000\ 0 \\ 000\ 011\ 0 \\ 000\ 000\ 1 \\ 000\ 001\ 1 \\ 000\ 000\ 0 \end{matrix}$$

Now $A_1 = B_1 = A$. Straightforward computation gives

$$A_2 = A \wedge A = \begin{matrix} 001\ 011\ 0 \\ 010\ 000\ 0 \\ 001\ 000\ 0 \\ 000\ 001\ 1 \\ 000\ 000\ 0 \\ 000\ 001\ 0 \\ 000\ 000\ 0 \end{matrix}$$

$$B_2 = B_1 \vee A_2 = \begin{matrix} 011\ 111\ 0 \\ 011\ 000\ 0 \\ 011\ 000\ 0 \\ 000\ 011\ 1 \\ 000\ 000\ 1 \\ 000\ 001\ 1 \\ 000\ 000\ 0 \end{matrix}$$

$$A_3 = A_2 \wedge A = \begin{matrix} 010\ 001\ 1 \\ 001\ 000\ 0 \\ 010\ 000\ 0 \\ 000\ 001\ 0 \\ 000\ 000\ 0 \\ 000\ 001\ 0 \\ 000\ 000\ 0 \end{matrix}$$

$$B_3 = B_2 \vee A_3 = \begin{matrix} 011\ 111\ 1 \\ 011\ 000\ 0 \\ 011\ 000\ 0 \\ 000\ 011\ 1 \\ 000\ 000\ 1 \\ 000\ 001\ 1 \\ 000\ 000\ 0 \end{matrix}$$

A glance at the diagram shows that all possible paths (without repetition) can be traversed in at most three steps, so that by Theorem 2, $B = B_3$. This can be checked by computing B_4, which is equal to B_3. From this matrix we verify immediately that all boxes are connected to the input (first row), but boxes 1 and 2 are not connected to the output (last column). Boxes 1, 2, and 5 are involved in loops (main diagonal). Moreover, if we delete the first row and last column of B, then the remainder can be decomposed into submatrices:

$$\begin{matrix} 11\ 000 \\ 11\ 000 \\ 00\ 011 \\ 00\ 000 \\ 00\ 001 \end{matrix} = \begin{matrix} M & 0 \\ 0 & N \end{matrix}$$

where $M = \begin{matrix} 11 \\ 11 \end{matrix}$ and $N = \begin{matrix} 011 \\ 000 \\ 001 \end{matrix}$ This implies that boxes 1 and 2 and boxes 3, 4 and 5 form two independent subprograms whose associated matrices are just M and N. (Of course, the simplicity of this decomposition is due to the particular scheme adopted for numbering the boxes.) This simple example serves to illustrate the scope of the method.

This same method has an obvious application to the problem of debugging programs already compiled. In this case the boxes are already numbered by the sequential description of the program. Moreover, it is not necessary to draw the corresponding flow diagram, since, except for transfers, each operation is followed by the next in sequence. As a second example we take a typical SAP writeup of an IBM 704 program, with no inconsistencies. (This program computes an array of 100 quantities c_{ij} according to the formula

$$c_{ij} = \begin{Bmatrix} A_i - B_j \text{ if } i > j \\ A_i + B_j \text{ if } i \leq j \end{Bmatrix}$$

SAP Program

```
 1. LXD   8
 2. SXD   4
 3. CLA  B1
 4. TXL   6
 5. CHS
 6. ADD  A1
 7. STO  C1
 8. TXI   9
 9. TXI  10
10. TNX   2
11. TXI  12
12. TNX   2
13. END
```

The associated connectivity matrix can be written down directly, and is simply

$$A = \begin{matrix} 010\ 000\ 000\ 000\ 0 \\ 001\ 000\ 000\ 000\ 0 \\ 000\ 100\ 000\ 000\ 0 \\ 000\ 010\ 000\ 000\ 0 \\ 000\ 001\ 000\ 000\ 0 \\ 000\ 000\ 100\ 000\ 0 \\ 000\ 000\ 010\ 000\ 0 \\ 000\ 000\ 001\ 000\ 0 \\ 000\ 000\ 000\ 100\ 0 \\ 010\ 000\ 000\ 010\ 0 \\ 000\ 000\ 000\ 001\ 0 \\ 010\ 000\ 000\ 000\ 1 \\ 000\ 000\ 000\ 000\ 0 \end{matrix}$$

(Note that, except for transfer instructions, 1's appear only on the super diagonal.)

The Precedence Matrix

A further analysis of the structure of a program can be made if information concerning the precedence relations in the program is available. If we know, for example, that the output of box i is required for the input of box j, then we know that the operation represented by box i must precede that represented by box j in the program sequence. Clearly this places additional requirements on the internal connectivity of the program.

The precedence relations may be incorporated into our analysis through the introduction of a second Boolean matrix C associated with the program, which we call the *precedence matrix*, (cf. [1, 9]). It is constructed as follows. We number the boxes of the diagram as in Section II, and stipulate that *the i-j entry c_{ij} of C is to be 1 if the output of box i (or any part of it) is required for the input of box j, and* 0 *otherwise.* Clearly this matrix contains the precedence relations in the same way that the matrix A contains the connectivity relations of the program, and will yield to a similar analysis. We observe here that the two matrices are closely related, though they need not be identical.

Proceeding as before, we define

$$\begin{aligned} C_m &= C_{m-1} \wedge C \\ D_m &= D_{m-1} \vee C_m \\ D &= \lim_{m \to \infty} D_m \end{aligned}$$

and observe that the results of that section may be translated immediately into the present situation. In particular, the i-j entry of the matrix D is 1 if and only if there is a chain of boxes in the diagram beginning with box i and ending with box j such that each box in the chain must precede the next. Obvious applications include the following:

(1) The precedence requirements are internally consistent only if the diagram contains no closed chain of boxes each of which must precede the next. This is the case only if no diaonal entry of D is 1. Thus we require that trace $D = 0$ for this consistency (cf. [1]).

(2) In general, box j depends on box i only if $d_{ij} = 1$. Thus if box i is altered, this will affect only those boxes whose entries in the ith row of D are 1.

(3) Occasionally it is desirable to reorder the sequence of operations in some part of the program. This is possible only if the precedence requirements are not violated by the reordering. Thus box i may be interchanged with box j in a chain of operations only if $d_{ij} = d_{ji} = 0$. Information of this kind is evidently useful in optimizing flow diagrams for time or storage requirements.

The Dominance Matrix

In studying problems involving the reordering of operations in a program, it is often useful to introduce a notion of *dominance* in the flow diagram, defined as follows: *We say box i dominates box j if every path (leading from input to output through the diagram) which passes through box j must also pass through box i.* Thus box i dominates box j if box j is subordinate to box i in the program. It may happen that two boxes dominate each other (in which case we say they are equivalent), or that neither dominates the other (in which case we say they are independent). The idea here, of course, is that reordering is possible only among boxes which are equivalent in this sense. Proceeding along these lines, we define a third Boolean matrix E, called the *dominance matrix*, by stipulating that *the i-j entry e_{ij} of E is 1 if box i dominates box i, and 0 otherwise.* It is clear that the dominance matrix is determined by the connectivity matrix, and can be produced from it by a suitable scanning procedure. Applications include:

(1) Box i and box j may be interchanged, precedence requirements permitting, only if they are equivalent. This is the case only if we have $e_{ij} = e_{ji} = 1$.

(2) In preparing a program for a machine which admits *parallel* operation, it is desirable to know which operations in the program may be performed simultaneously. Two operations may be performed simultaneously without further investigation only if they are equivalent and subject to no precedence requirements, *i.e.*, only if $d_{ij} = d_{ji} = 0$ and $e_{ij} = e_{ji} = 1$.

(3) It is sometimes useful to know when two programs are equivalent in some sense. Any effective definition of equivalence requires a detailed knowledge of what happens at branch points in the program (*i.e.*, the transfer conditions). An interesting analysis of this problem is summarized in [14], but does not seem readily adaptable to machine handling. By requiring a less effective definition of equivalence, we can give here an effective criterion for determining whether or not two programs are equivalent.

To be precise, let us agree that *two programs, containing the same operations subject to the same precedence requirements, are equivalent, if, for each path (leading from input to output) through the first, there is a corresponding path through the second passing through the same operations.* We do not require that the operations appear in the same sequence, or even that they appear the same number of times, in both paths. This definition, however, is sufficient for most purposes, at least for programs containing no loops; loops cannot be incorporated under so simple a scheme, and require special consideration.

In terms of flow diagrams, the equivalence criterion may be stated as follows. *Two diagrams, made up of the same boxes subject to the same precedence requirements, are equivalent only if their dominance matrices are identical.*

Remarks

The essential point of our discussion is that the entire analysis given here can be readily performed on any (large-scale) digital computer. The feasability of computing the derived matrices B, D, and E by machine is assured for programs which are not too large. A very crude estimate indicates that the time required to compute B from A on the IBM 704 is of the order of $10\ n^3$ cycles, where n is the number of boxes in the diagram. In practice, this time may be reduced considerably by combining into one box any subroutine whose behavior is known. Thus for example it is advantageous to replace any chain of boxes by a single box. Similarly, in analyzing program writeups it is sufficient to consider only transfer operations. For instance, a reduced form of the matrix A of our second example is:

$$A' = \begin{matrix} 010\ 000\ 0 \\ 001\ 000\ 0 \\ 010\ 100\ 0 \\ 000\ 010\ 0 \\ 010\ 001\ 0 \\ 000\ 000\ 0 \end{matrix}$$

where boxes 1 through 9 have been combined in a single box.

Finally we remark that it is a straightforward problem to construct a debugging routine which could be used to analyze any program writeup whose transfer instructions have constant addresses. Such a routine would scan the writeup, enumerate the transfer instructions, construct the connectivity and dominance matrices from them, compute the derived matrices and point out any errors detectable by these methods. Thus the whole analysis becomes completely automatic.

Various other applications of this analysis are suggested by the results. By utilizing the evident adaptability of these matrices to computer handling, it is possible to construct automatic program analysis schemes which would detect in proposed programs a large class of common errors, isolate and identify key subroutines and reorganize them in optimal equivalent programs. Such a scheme is currently under investigation here at Lincoln Laboratory, MIT.

Bibliography

[1] E. W. Barankin, "Precedence Matrices," *Univ. of Chicago Management Sciences Research Project*, Research Report no. 26; December, 1953.

[2] I. M. Copi, "Matrix development of the calculus of relations", *Jour. Symbolic Logic*, vol. 13, pp. 193–203; 1958.

[3] W. Feller, "An Introduction to Probability Theory and its Applications," John Wiley and Sons, New York, N. Y., p. 350; 1957.

[4] F. Hohn and L. Schissler, "Boolean matrices and the design of combinational relay switching circuits," *Bell System Tech. Jour.*, vol. 34, pp. 177–202; 1955.

[5] M. Kac and J. C. Ward, "A combinatorial solution of the 2-dimensional Ising model," *Phys. Rev.*, vol. 88, pp. 1332–1337; 1952.

[6] G. Kron, "Tensor Analysis of Networks," John Wiley and Sons, Inc., New York, N. Y.; 1939.

[7] S. Lefschetz, "Topology," *Colloq. Publications Amer. Math. Society*, New York, N. Y.; 1930.

[8] R. D. Luce and A. D. Perry, "A Method of matrix analysis of group structures," *Psychometrika*, vol. 14, pp. 95–116; 169–190; 1949.

[9] R. B. Marimont, "A new method of checking the consistency of precedence matrices," *Jour. Assoc. Comp. Mach.*, vol. 6, pp. 164–171; April, 1959.

[10] J. Riordan, "An Introduction to Combinatorial Analysis," John Wiley and Sons, Inc., New York, N.Y.; 1958.

[11] D. Rosenblatt, "On the graph and asymptotic forms of finite Boolean relation matrices and stochastic matrices," *Naval Res. Logist. Quart.*, vol. 4, pp. 151–167; 1957.

[12] H. Seifert and W. Threlfall, "Lehrbuch der Topologie," Chelsea, New York, N. Y.; 1947.

[13] C. Shannon and W. Weaver, "The Mathematical Theory of Communication," Univ. of Illinois, Urbana, Ill.; 1949.

[14] Y. I. Yanov, "On matrix schemes," *Dokl. Akad. Nauk. USSR*, vol. 113, pp. 39–42; 1957.

Discussion

E. Fredkin (*Bolt, Beranek and Newman*): In the case of a closed subroutine used by more than one calling sequence, how do you represent the fact that, while many routines enter and many exit, the subroutine box may return only to the calling routine?

Mr. Prosser: Problems of this kind, of course, are not handled at all by this formalism. Nothing has been said about how you make the decisions about where to go. I have deliberately avoided this. Thus, the whole theory is a black box theory. Now actually in some flow diagrams there are paths which you cannot follow at all, because the appropriate combination of logical requirements is never satisfied. This formalism will not tell you that. The best it can do is tell whether there is a path going from here to there, without telling whether or not the conditions for it are met.

Mr. Shapiro (*Nat'l Institute Health*): Given that your formalism does not account for the nature of decision-making elements, what is the definition of equivalence?

Mr. Prosser: Well, there is an interesting problem here. Let me say, first of all, there is a study by the Russian mathematician, Yanov, who has made a definition of equivalence which says, roughly speaking, that two programs are equivalent if they go through the same boxes in the same order. That is, for each path through one program, there is a path through the other one which does the same operations in the same order. In order to prove statements like that, you have to know something about how many times you go around loops, and I have no way of counting this in the present formalism.

So the definition of equivalence which I'm using here must be very weak. It would run something like this, and this is, in fact, the precise statement to which I was referring: two diagrams are equivalent if, for every path through the first one, there is a path through the second one which goes through the same boxes, not necessarily in the same order and not necessarily the same number of times. But they do the same things. You recognize that this definition is quite weak unless you know something more about the number of times you go around loops. Using this for a definition, then the statement is that two diagrams are equivalent only if their dominance matrices are identical.

Mr. Miller (MITRE): Do you have a way of automatically generating your first matrix?

Mr. Prosser: You can do this in some cases, but not from a diagram. You can do it, for example, from an SAP write-up or from certain other kinds of write-ups automatically, providing that certain restrictions are placed on the write-ups — that they not be too complicated. As far as flow diagrams go, there is an obvious problem here. If you work from a flow diagram, you have to try somehow or other to get the diagram into the machine. We don't really know how to do this effectively, but then we really haven't studied the question. What we've done in actually running this experimentally is to take a typical flow diagram and try to draw these matrices by hand. All you have to do is record the 1's, of course. I don't know how to do it automatically.

I would like to say, though, that this program which I refer to, which computes the derived matrix B, we intend to submit to SHARE, so it ought to be available to the computing world fairly soon.

C. E. Dorrell (IBM): Do you have a program to compute F, the dominance matrix, and, if so, what is its running time?

Mr. Prosser: This is in the process of being put together. It should take about the same running time.

H. D. Friedman (Technical Operations): Since B is a geometric series of powers of A, although in the Boolean sense, isn't there an analytic method for obtaining B?

Mr. Posser: The question can be rephrased this way. Let B_K be the *Kth* step of the B matrix. How far out do you have to go before you have reached the limit? As I have indicated already, there is a number such that, beyond that, the B_K's are already constant and are equal to the limiting matrix. How far out is it? Well, an upper bound is the length of the longest path through the diagram, which is always less than the number of boxes in the diagram. Actually, our routine doesn't compute B by the process which I showed on the slide. If you look at the matrix A plus A square, and raise this to high powers, it turns out that this process gives you B. Now for a 500 by 500 matrix, something like 2^9 powers is enough, so the maximum number of squarings required is something like nine. So there is an upper bound which is not too big. The thing which makes this feasible, of course, is that the matrices are all zero's and one's. It is much easier to do matrix operations with them than with the usual matrices with arbitrary entries.

SIMCOM—The Simulator Compiler*

THOMAS G. SANBORN†

IN MANY present-day activities involving the use of digital computers, the need often arises to run programs on a computer other than the one for which they are written. For example, the computer on which a program is intended to be run may exist only as a proposed design, or it may be in some stage of construction, or it may simply be at a remote location[1]. One solution to the problem posed by such a situation is to prepare a program for an available computer which, in effect, transforms the available computer into the unavailable computer. Such a transformation program is called a *computer simulation program,* since it gives one computer the ability to simulate another.

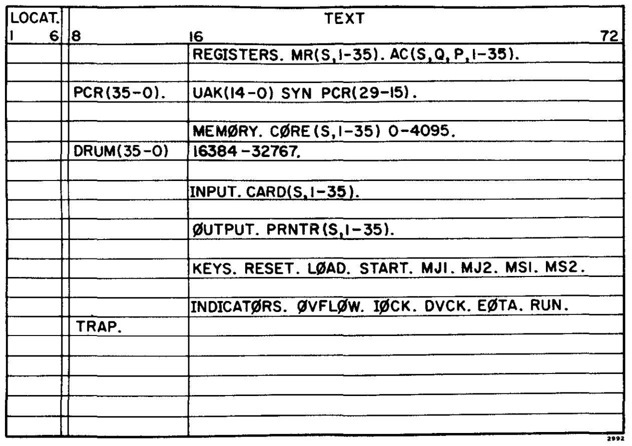
Because of the intricate logical relationships which prevail in computers, the preparation of a simulation program is time consuming and fraught with opportunity for error. Furthermore, changes in the specifications of the computer being simulated may necessitate a major overhaul of the simulation program. For these reasons a new programming language and its associated compiler, SIMCOM (standing for *Sim*ulator *Com*piler), are being developed to assist in the preparation and modification of simulation programs which are to be run on the IBM 709.

It must be clearly understood that SIMCOM is *not,* itself, a simulation program. It is a generating program which accepts statements written in a specialized simulation-oriented source language, and from these statements generates instructions in SCAT language similar to those a human programmer would write in preparing a simulation program.

The fundamental unit of SIMCOM coding is the *statement.* Each statement is either a definition of a component of the simulated computer or a description of some data manipulation or control function which occurs during the execution of instructions within the simulated computer. These two kinds of statements are known, respectively, as *definition statements,* and *procedural statements.* Related statements are grouped into *paragraphs.* SCAT coding, including SCAT-type remarks, may be intermixed with the paragraphs should the SIMCOM language prove inadequate for describing some involved procedure. The characters which may be used to write statements include the upper-case Roman letters, the decimal digits, and certain special characters. Combinations of alpha-numeric characters are called *symbols.* The three uses of symbols are: 1) to represent components of the simulated computer; 2) to identify locations within the simulation program; and 3) to denote integers. Every symbol, unless it represents an integer, must contain at least one alphabetic character, and no symbol may be identical to a word of the basic SIMCOM vocabulary.

A simulation program written in the SIMCOM language consists of three parts: the "Machine Definition;" the "Instruction Interpretation;" and the "Panel Operation." These sections describe, respectively, the static machine, the machine in operation, and the effect of operator intervention.

The Machine Definition is given in six paragraphs labeled "REGISTERS," "MEMORY," "INPUT," "OUTPUT," "KEYS," and "INDICATORS." Each definition statement describes a machine component or cell, giving its name, bit structure, and, if appropriate, its address or range of addresses. For coding convenience, a register may be defined as being synonymous with part or all of another register. Furthermore, registers can be defined which have no counterpart in the real computer being simulated. No distinction is made by SIMCOM between so-called primary and secondary storage.



LOCAT. 1 6		8	TEXT 16 72
			REGISTERS. MR(S,1-35). AC(S,Q,P,1-35).
		PCR(35-0).	UAK(14-0) SYN PCR(29-15).
			MEMØRY. CØRE(S,1-35) 0-4095.
		DRUM(35-0)	16384-32767.
			INPUT. CARD(S,1-35).
			ØUTPUT. PRNTR(S,1-35).
			KEYS. RESET. LØAD. START. MJ1. MJ2. MS1. MS2.
			INDICATØRS. ØVFLØW. IØCK. DVCK. EØTA. RUN.
		TRAP.	

Fig. 1—Simcom coding form showing typical definition paragraphs.

Fig. 1 shows examples of some typical definitions selected from several well-known computers. This figure also illustrates the basic requirements of the form on which SIMCOM coding is to be written. Some users may wish to use a form on which each

* Presently being developed under a purchase order from Thompson Ramo Wooldridge, Inc., in support of their contract to supply technical direction to the Automatic Data Processing Facility, U. S. Army Electronic Proving Ground, Fort Huachuca, Arizona.

† Space Technology Laboratories, Inc.

[1] For a description of the applications of SIMCOM anticipated by USAEPG see: A. B. Crawford, "Automatic Data Processing in the Tactical Field Army", Proceedings of the Western Joint Computer Conference, pp. 187–189; San Francisco, March 1959.

card column is marked since blank positions are frequently essential in the language. Note that the first line of each paragraph is indented to column 16 and that subsequent lines begin in column 8. The compiler uses this convention to aid in distinguishing between SIMCOM statements and SCAT instructions which may be included in the source code.

In all of the machine component definitions, the symbols used to identify the various devices are quite arbitrary, the only limitations being that they conform to the previously stated rules pertaining to symbols, and that they do not conflict with the basic vocabulary. The most common method of selecting symbols will undoubtedly be to adopt those used by the machine manufacturer in his manual since these are usually highly mnemonic.

The Instruction Interpretation and Panel Operation sections of the simulation program are written in terms of procedural statements. A procedural statement consists of a *primary operation* together with one or more operands called *expressions*, arranged to form a stylized sentence. Each primary operation is denoted by one or more words from the basic SIMCOM vocabulary. This vocabulary includes words for transferring, clearing, complementing, testing, comparing, and shifting arrays of bits in the various components of the simulated computer, plus words which control the logical flow of the simulation program and the compilation process itself.

The expressions upon which the primary operation act consist of symbols combined by means of secondary operations. These secondary operations include + (add), − (subtract), * (multiply) $ (indirect address), and certain words of the basic vocabulary which denote logical arithmetic and scaling. A symbol in an expression may have bit designators appended to it if only part of the component identified by the symbol is to participate in the operation.

Fig. 2 shows a few paragraphs of procedural statements. In the figure the expressions are underscored for emphasis but this would not be the normal practice on a coding form. Note that the first paragarph bears a location symbol. In many instances, however, cross references between paragraphs are implicit in their relationship to one another. Therefore, many paragraphs will need no location symbols attached to them.

The instruction interpretation section is the heart of the source program. It will normally contain statements which describe the procurement of instructions from the simulated computer's storage, followed by statements which describe the effects of each instruction. The various instruction interpretations can usually best be initiated by use of a "table look-up" statement. Depending on the complexities of the instructions and the associated timing, each instruction description may require as little as one simple statement or as many as several paragraphs including, perhaps, entries to subroutines and additional table look-ups. Fig. 2 illustrates a simple example of this technique.

LOCAT. 1 6	TEXT 8	16 72
LØØP		IC $ TØ IR. IC + 1 TØ IC. LØØK UP IR(1-5) IN
	ØPER TABLE.	CLØCK 4. EXECUTE LØØP.
ØPER		TABLE.
		0. IR(6-17) TØ IC. TURN RUN ØFF.
		1. IR(6-17) TØ IC.
		2. IF ØVFLØW IS ØN. TURN ØVFLØW ØFF.
	IR(6-17)	TØ IC.
		5. EXECUTE GET. EXECUTE SUBTRA.
		CLØSE.

Fig. 2—Typical procedural paragraphs showing instruction procurement and interpretation technique. "Expressions" underscored for emphasis.

A "Table," as understood by SIMCOM, is an ordered set of paragraphs of procedural statements, each paragraph being identified by an integer. The table look-up operation provides a means for selecting one of these paragraphs for execution, depending on the value of the argument expression. If a paragraph in a table does not terminate with an explicit transfer of control to some other point in the simulation program, then control returns to the statement following the "LOOK UP . . ." statement which invoked the paragraph. Thus each paragraph in a table is like a closed subroutine.

The panel operation section of the simulation program includes an interrogation of the status of each console key and a description, written in SIMCOM statements, of the behavior of the simulated computer if the key has been activated.

It is not uncommon for certain keys on computer consoles to be so constructed that they are turned off as soon as the function which they perform has been initiated. The programmer's statements must include this action, if appropriate. Furthermore, in some cases certain keys are inoperative unless other keys or indicators are in a particular status. The programmer must also provide this logic.

One of the most interesting features of the system is the subroutine library. Subroutines are stored in the library in the SIMCOM language, except that the symbols denoting the subroutine parameters are replaced by variable symbols of a special kind. At compilation time, as a subroutine is called from the library, its special variable symbols are replaced by the parameter symbols given in the library call statement, and its location symbols are replaced by arbitrary unique symbols. The subroutine is then inserted into the source code where the SIMCOM decoder and instruction generator processes it in the same manner as any other set of SIMCOM statements. This process is illustrated in Fig. 3.

```
THE SUBROUTINE IS ORIGINALLY CODED AS

                    SUBRØUTINE ADD, A, C.
            A + B TØ C. CLØSE.


THE LIBRARY MAINTENANCE ROUTINE WILL PLACE THE SUBROUTINE IN
            THE LIBRARY IN THE FORM

ADD                 V1 + B TØ V2. CLØSE.


AT A SUBSEQUENT COMPILATION, A STATEMENT OF THE FORM

                    LIBRARY ADD, P, Q.

WILL CAUSE THE SUBROUTINE TO BE INCORPORATED INTO THE PROGRAM AS

ADD                 P + B TØ Q. CLØSE.
```

Fig. 3—Sample subroutine showing generalized variable technique.

A given subroutine may be called from the library any number of times during one compilation and, depending on the parameters listed in the library call statement, each version may give rise to a different number of 709 instructions. Each version of a subroutine called from the library is a "closed" routine which can be executed from any point in the simulation program.

The output from SIMCOM is a translation into SCAT language of the source program. This includes a direct expansion of the procedural statements, plus certain pseudo operations for assigning storage and certain utility routines whose necessity is only implied by the source language. These include routines for loading the simulated computer, diagnostic output routines and, of fundamental importance, a routine which allocates the simulated computer's storage to the various 709 storage media. This storage management routine must partition oversize words, should such have been defined, into the 36-bit words of the 709, and shuttle simulated computer storage to and from 709 tape units if it exceeds the capacity of the 709 core storage. The endowment of the compiler with the ability to generate efficient storage management routines is the most challenging problem facing the creators of SIMCOM.

Fig. 4 is a schematic representation showing the allocation of the generated program to the various parts of the 709. The heavily outlined areas indicate the parts of the 709 used to represent the various registers and storage of the simulated computer. The remainder of the 709 contains the generated simulation program and its associated utility routines. The arrows indicate the communication paths between the various areas of the 709.

Because the SIMCOM output is in SCAT language, the compiler need not contain within itself an assembly program, nor does it have to be able to process the SCAT instructions included in the input code other than to recognize them as SCAT in-

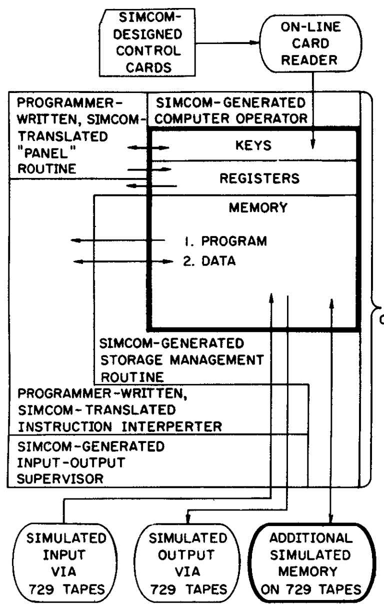

SIMCOM-DESIGNED CONTROL CARDS
ON-LINE CARD READER
PROGRAMMER-WRITTEN, SIMCOM-TRANSLATED "PANEL" ROUTINE
SIMCOM-GENERATED COMPUTER OPERATOR
KEYS
REGISTERS
MEMORY
1. PROGRAM
2. DATA
709 CORE
SIMCOM-GENERATED STORAGE MANAGEMENT ROUTINE
PROGRAMMER-WRITTEN, SIMCOM-TRANSLATED INSTRUCTION INTERPERTER
SIMCOM-GENERATED INPUT-OUTPUT SUPERVISOR
SIMULATED INPUT VIA 729 TAPES
SIMULATED OUTPUT VIA 729 TAPES
ADDITIONAL SIMULATED MEMORY ON 729 TAPES

Fig. 4—Allocation of elements of a Simcom-prepared simulation program to 709 core and tapes.

structions. Most important, however, is the fact that generation from SIMCOM statements to SCAT instructions can be done during a single pass through the source program. In addition to generating SCAT language instructions, the compiler transforms each line of SIMCOM coding into a SCAT-type remark (* in column 1) and inserts each paragraph into the generated code immediately ahead of its SCAT language expansion. Thus each paragraph serves as commentary to describe the function of the generated SCAT instructions which follow.

The SIMCOM language is such that apparently minor modifications to the input statements can completely alter the character of the generated program. For example, a change in the definition of a register of the simulated computer may cause SIMCOM to generate instructions to do multiple precision arithmetic where single precision arithmetic was formerly sufficient, or a change in word size in the simulated computer may cause SIMCOM to reorganize completely the simulated computer's storage in the 709 core. Thus changes which could be made to a machine-like language simulation program only by completely rewriting the program can be incorporated into a SIMCOM-written simulation program by a simple re-compilation.

The SIMCOM system will provide a means whereby users who are not necessarily professional programmers may prepare simulation programs for binary computers in a language not unlike that used by computer manufacturers in their manuals. There seems to be no escaping the fact that the user will need to be more than casually familiar with the computer to be simulated before he can write an adequate simulation program, even with SIMCOM.

Discussion

P. Armer: Is your work far enough along so you can comment on two things with respect to timing? One, let us say with a simulated machine not too different from the 709 that didn't have to do with double precision arithmetic. Could you give us some notions on the efficiency time lapse?

Mr. Sanborn: Since we have not actually constructed any simulation programs yet, I couldn't give you an answer from experience, but we are optimistic that the simulation programs generated by SIMCOM will be relatively efficient as compared to handwritten codes. I am not sure it would be too efficient for a machine similar to the 709, because it could not capitalize on the similarities of the machines.

Mr. Armer: In simulating a machine one would be interested in how fast the program would run. Have you anything to solve this problem?

Mr. Sanborn: There are language statements for keeping track of time; the clock statement we saw was one. One can, in writing down the description of the instructions, also include information as to the execution time — as a matter of fact, keep a running total of the elapsed time in the computer. This may become particularly important where you have independent devices running and may want to keep several clocks running in order to determine which unit is going to run next and keep them in proper synchronization.

G. L. Foster (IBM): Do you assume the machine being simulated has a fixed word length? Will SIMCOM handle variable word lengths as on the 705?

Mr. Sanborn: I think it would if the registers are defined in terms of smallest accessible storage. I question whether it would be practical, but it would be possible.

R. Cornish (IBM): Is this program universal only with respect to the particular family of computers?

Mr. Sanborn: No, I wouldn't say this. We tried to make it general for binary computers with word sizes not greater than 72 bits.

Mr. Cornish: Approximately how many man years were involved?

Mr. Sanborn: I haven't checked the figures recently but I imagine up to this time we have invested two man years.

D. J. Campbell (AGT): How much machine time would be used in a typical compilation?

Mr. Sanborn: I don't think I would want to predict this. I haven't been at all concerned on compiling time. I have been concerned about execution time of the generated program.

R. J. Scott (Dept. of Defense): Could SIMCOM simulate a machine with an automatic interrupt feature as is used on STRETCH?

Mr. Sanborn: I think if a machine has a feature like this, you must write statements in the language which quite frequently enters the panel operation section to test interrupt conditions.

I. Flores (Dunlop Assoc.): In debugging a program for a computer to be simulated, how will the 709 indicate program faults or other errors?

Mr. Sanborn: I am not sure that the 709 will indicate faults in the program of the simulated computer. There are diagnostic facilities in the generated program. For example, statements calling for certain registers to be printed out. This may be diagnostic or it may be results from the program running in the simulated computer.

R. W. Bemer (IBM): Does it simulate only binary machines?

Mr. Sanborn: Only binary machines in the sense that for any other sort of machine one would have to define a representation of the information in the computer in terms of binary digits.

Mr. Bemer: Does the syntax correspond to or use Gorn's microflow chart technique?

Mr. Sanborn: I don't know, because I am not familiar with this technique but I would guess not.

E. B. Shore (Pratt & Whitney): Can SIMCOM be used to compare various computers on a benchmark program? How about multi-sequence computers?

Mr. Sanborn: I am not sure what is meant by benchmark program.

Mr. Armer: Well, anyone can dream up a program and see how various machines do on this same program.

Mr. Sanborn: SIMCOM would provide the means for constructing the simulation program for trying out various computers. It has in itself no mechanism for evaluating these computers on the basis of running the benchmark program.

Mr. Armer: If I understood the question of the clocks, it would tell you how long it would take each machine to run the particular program.

Mr. Sanborn: Yes, the clocks would tell you running time but if you wanted to get more involved information about the relative merits of the computers this is beyond the scope of SIMCOM.

Unusual Techniques Employed in Heat Transfer Programs

D. J. CAMPBELL† AND D. B. VOLLENWEIDER†

A PROGRAM for the IBM 704 to solve general transient and steady state heat transfer problems is described. The computer program was developed by Evendale Computations Operation in cooperation with the Jet Engine Department, General Electric Company, Flight Propulsion Division. Jet Engine Department personnel who were instrumental in the formulation of the problem are Mr. William K. Koffel and Dr. J. M. Botje. Dr. James T. Anderson, of Michigan State University, acted as a consultant and played a central role in the development of the program. The method of solution permits the analysis of problems with arbitrary geometry and several combined modes of heat transfer. To facilitate the description and solution of extensive and complicated problems, many logical and computational techniques not usually applied to engineering calculations are incorporated in the program.

The program yields the temperature distribution at a maximum of 200 points in three-dimensions for homogeneous or composite bodies. Heat transfer by conduction, convection, surface flux, thermal radiation, both solid and gaseous, and internal heat generation is treated. One dimensional fluid flow in several channels can be incorporated. The thermal properties of the system may vary with temperature. Boundary conditions and heat transfer rates for convective exchanges as well as surface flux rates and mass flow rates may vary with time. Internal heat generation is given either as a function of time or of temperature.

THERMAL STORAGE — NON-SOURCE, SINK FLUID HEAT TRANSFER — CONDUCTION & CONVECTION — INTERNAL HEAT GENERATION

$$\frac{C\rho V(T_0 - T_0')}{\Delta\tau} = cW(T_f - T_0) + \sum_{n=1}^{6}\left[\frac{A_n(T_n - T_0)}{\frac{\Delta X_n}{K} + \frac{1}{h_c}}\right] + Q_0 V$$

SURFACE FLUX — SOLID RADIATION — GASEOUS RADIATION

$$+A_s Q_f + \sum_{i=1}^{3}\sum_{s=1}^{3}\left[\sigma A_i F_i (T_h^4 - T_0^4)\right] + \sum_{j=1}^{3}\left[\sigma A_j G_j(\epsilon_g T_g^4 - \alpha_g T_0^4)\right]$$

Fig. 1—The general heat balance equation. T_o indicates the temperature at the beginning of a time interval. Linearized versions of the radiation terms are actually programmed.

The method of solution for the temperature distribution is to divide the geometry into cells enclosing nodes and to write heat balance equations (in finite difference form) for each node. The general heat balance equation that is written for each node is given in Fig. 1. Note that T′ indicates temperatures at the beginning of a time interval. The radiation terms are linearized by writing them as a radiation coefficient times a first power temperature difference. The radiation coefficient is computed using the temperatures at the beginning of the time interval, while the linear temperature difference uses a given radiation source temperature. The implicit form of heat balance equation is used in preference to the explicit form, because the stability requirement for the explicit form places a limit on the time increment which is restrictive for large systems. The implicit form permits more freedom in the choice of time step. The accelerated Gauss-Seidel method is used to solve the system of heat balance equations for the n unknown temperatures. Gauss-Seidel is used in preference to other methods because the convergence criterion for the method is readily met for most applications and because there are a large number of zero elements in the coefficient matrix.

One of the most interesting features of the program is the method used for describing the geometry. Rectangular coordinate systems are not adequate to fully describe the variety and intricacy of shapes commonly encountered in applications of the program.

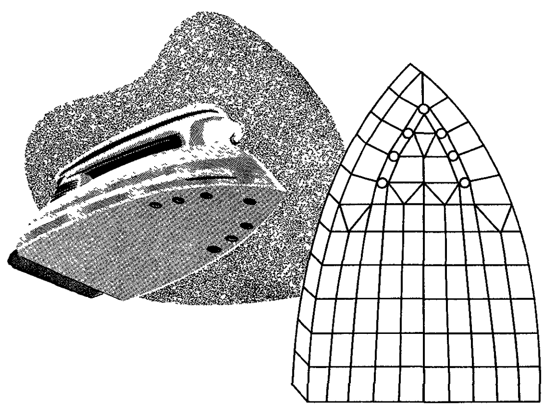


Fig. 2—As an example of the method of dividing an object into cells, the plate of a steam iron is shown. Note the irregular manner in which the cell division has been made.

Instead, the geometry is divided into cells of any shape or size, and the cells are given numbers to identify them. Fig. 2 shows how the plate of a steam iron might be divided into cells. At most six faces are defined for each cell, and the faces are labeled by numbers from one to six. Fig. 3 illustrates the numbering of faces of a cell. In order to write heat balance equations, it is necessary at each cell to give

† Flight Propulsion Laboratory Department, General Electric Company, Cincinnati, Ohio.

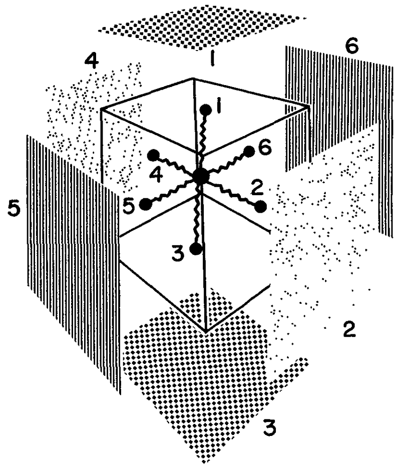


Fig. 3—Illustration of the convention for numbering the faces of a cell.

the adjacent cell, if any, for every face through which there is conduction. In the program, this is done by giving the number of the cell adjacent to each face. A convention with respect to adjoining cells makes it unnecessary to give the number of the face of the adjacent cell. The face common to two adjacent cells is, of course, assigned a number in each of the cells. As can be seen from Fig. 4, if a face is labeled "1" for one of the cells, then it must be labeled "3" for the other cell. Similarly, if the face is labeled "2" in one cell, it must be labeled "4" in the other, and if it is labeled face "5" in one cell, it must be labeled face "6" of the other.

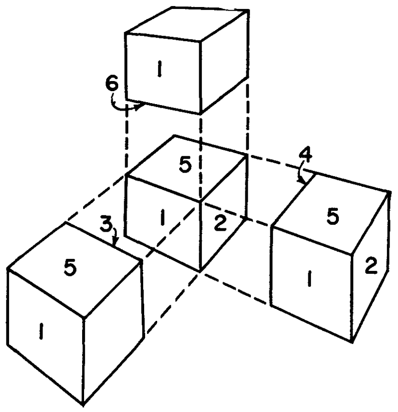


Fig. 4—Illustration of the face numbering convention for adjacent cells. The numbers assigned to a face of adjacent cells must be one of the pairs (1, 3), (2, 4) or (5, 6).

Although this method of describing geometry may seem awkward at first glance, it conveniently accommodates unusual patterns of cells and irregular surfaces. Fig. 5 shows some unusual shapes that may be accommodated with this method of describing geometry. It can be seen that cells need not have six faces and can be highly irregular in shape. Any face not needed for conduction exchanges may be used to accommodate special effects. For example, in Fig. 6, the two narrow cells exploded out of position are used to increase the number of faces for the cell that occupies the position in the center of the drawing. Since the cells are assumed to have zero volume and zero thickness, there are no thermal storage terms and no thermal gradients across them. Therefore, they will have no effect on the heat balance except to balance the cell in the center against all of the surrounding cells. These nodes increase the effective number of faces of the central node from six to eleven.

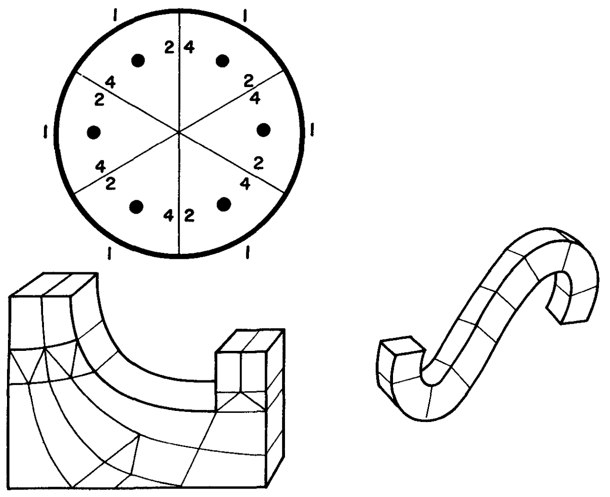


Fig. 5—Examples of irregular shapes that have been divided into cells.

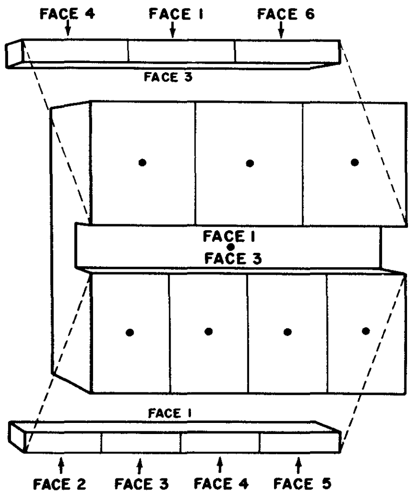
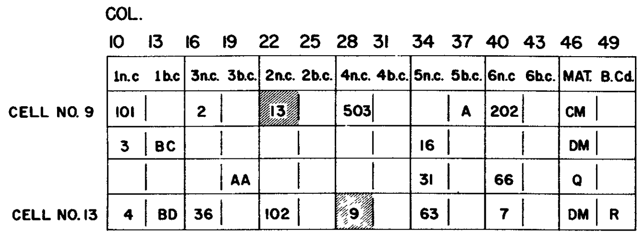


Fig. 6—Illustration of a method for increasing the effective number of faces of a cell. By including two 'zero volume' cells, the number of faces of the long, innermost cell is increased to eleven.

A symbol defining a table of thermal properties is given for each cell in the system. These properties are tabulated as functions of temperature. Given the geometry description and the material symbol just mentioned, sufficient information is available to assemble the conductive heat transfer equations. In describing the dimensions of a cell, a distinction is made between "regular" and "irregular" cells. Regular cells are rectangular parallelepipeds, or linear approximations thereof, and only height, width, and depth of each cell are required as input. For irregular cells, distances from each face to the node point, face areas, and the cell volume must be specified. Physical parameters and boundary conditions are given as functions of time or of temperature. For example, tables of adiabatic wall temperature and fluid film coefficient are given, and the collection of tables is assigned a symbol. The symbol defining the applicable tables of functions is entered for the proper face of the cell which has a convective heat exchange. An analogous method is used to indicate other modes of heat transfer. That is, the functions are defined and identified by a symbol, and the symbols are given at each appropriate cell. Consequently, there are two basic types of information for a cell. One gives the dimensions of the cell and other geometric constants such as configuration factors for thermal radiation. Secondly, symbols are given to define the surrounding geometry, to indicate the modes of heat transfer, and to define the tables of values of the various parameters.

From the symbols describing the cell configuration and corresponding properties, the heat balance equations may be assembled for each time step. However, an assembly directly from symbols would require a large number of sorting procedures on the basis of node numbers and table symbols. To avoid searching for information designated by symbols more than once, a simple device is utilized in the program. Each symbol is replaced by the machine address of the initial memory location of the information defined by the symbol at the beginning of the program, before any temperature calculations are begun. The searches that are undertaken automatically check for completeness. If a table or node has not been given, the search for the location of that table must fail. When such a failure occurs, a comment is printed giving the symbol of the missing table or node.



COL.	10	13	16	19	22	25	28	31	34	37	40	43	46	49
	1n.c	1b.c	3n.c.	3b.c.	2n.c.	2b.c.	4n.c.	4b.c.	5n.c.	5b.c.	6n.c	6b.c.	MAT.	B.Cd.
CELL NO. 9	101		2		▨ 13		503			A	202		CM	
	3	BC							16				DM	
				AA					31		66		Q	
CELL NO. 13	4	BD	36		102		▨ 9		63		7		DM	R

Fig. 7—An example of data describing the geometry of a problem. The connection between cells number 9 and 13 is illustrated to indicate the need for checking its consistency.

Further examination of the form of the input incicates that testing the cell connection information is highly desirable. The method of describing geometry requires that connection information for adjoining cells must be given for the designated face of each cell. This may be seen by examining Fig. 7, a facsimilie of the cell information input form. The entries necessary to define a connection between cells 9 and 13 are indicated. If the connection is not indicated for both cells, erroneous heat balance equations will be generated, and the equations may yield results that appear correct but are in error in the vicinity of the inconsistency. Moreover, the volume of data required for large problems encourages error in input. Consequently, the program checks the consistency of cell connections and makes many other tests for completeness of input data to recognize and identify errors before the equations are assembled. The machine time required for these completeness and consistency tests is negligible when compared with overall computing time.

It can be seen that the method for describing arbitrary cell configurations does necessitate considerable pre-processing, searching, and testing of input data. If the geometry were described by the usual mesh or lattice point scheme, input for the network could be organized by three-dimensional arrays and could be assigned to ordered blocks of machine storage. There would be a simple arithmetic relationship between nodes and the corresponding memory location of information pertaining to nodes. However, if it is necessary to describe and analyze arbitrary patterns of cells enclosing node points, a simple arithmetic formula for computing the address of the machine location of information for another node from the address of a given node cannot in general be established.

Having presented some of the interesting aspects of the program, we would like to describe possible extensions of the techniques and methods employed. In the present program, input data is stored in fixed sections of memory. Consequently, there are specified limits for the maximum number of entries in a table, the number of tables of a given variety, and the maximum number of cells in the division of the problem. These restrictions do not allow the available memory space to be used efficiently for many problems. In a subsequent program, similar input is stored in consecutive memory locations which are computed as the data are read into memory. Since sections of data are not assigned fixed storage locations, an economical means for finding specified quantities is necessary to accomplish the processing and testing of input data. Fig. 8 illustrates this memory packing system. Each table begins with two symbols that are shown shaded in the figure. The first symbol identifies the table and defines its type. The second symbol indicates the length of the table. The next table begins in the first unused memory location after the preceding table. Thus, for each table there is information which can be used to locate the next table. By allocating storage as the input data are read into the machine, considerable flexibility in the type of problem acceptable to the program is achieved. There are no pre-assigned limits on the maximum number of such quantities as nodes or physical parameters. Only the total amount of storage available for data limits the size and type of problem that can be handled by the program. An additional advantage to flexible input storage assignment is that the program can be run on a 704 with any memory capacity. In one program this idea was extended further by allowing tables of a particular type to have one of several different formats of varying length. In this case, the next table could begin at any one of several alternate locations. To determine the location of the next table, the symbols are defined in such a manner that the binary representation of these symbols are distinct from ordinary floating-point numbers. Thus, the table following a given table is found by searching each of several alternate locations for a word not in normal floating-point format, that is, for the symbol defining the new table.

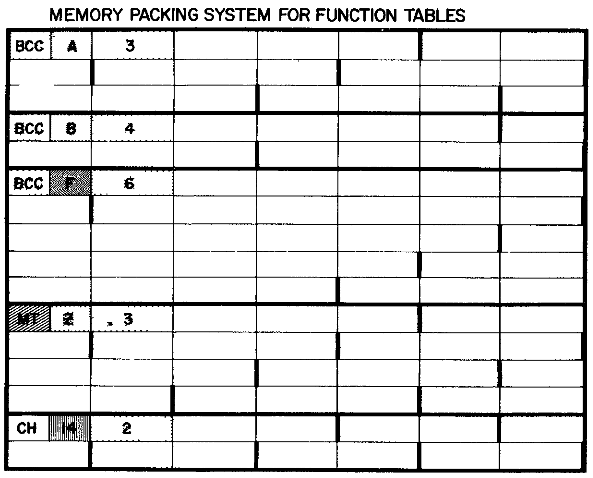


Fig. 8—Illustration of memory packing system. The shaded areas indicate symbols defining the type and length of a table. Each table begins immediately after the preceding table.

Another extension of the transient heat transfer program is to assign an order to the operation on cells. At present, the heat balance equations are assembled and solved in an arbitrary order. Certain applications of the program to large classes of less general problems suggest that methods be incorporated for computing quantities required as input to the present program. A body with two horizontal fluid passages is illustrated in Fig. 9. In some cases it is desirable to make calculations in the order indicated by the arrows, regardless of the order of the data within the computer. For example, the computation of weight flow and fluid film coefficients from given inlet and outlet pressure for one-dimensional fluid channels could be programmed. If the computer were programmed to operate on the cells in each channel in the direction of flow, calculation of such quantities as fluid film coefficient could be incorporated easily. A straight-forward method for constructing an order of operation on node information is to recognize the desired order from the symbols describing the problem and to store with information

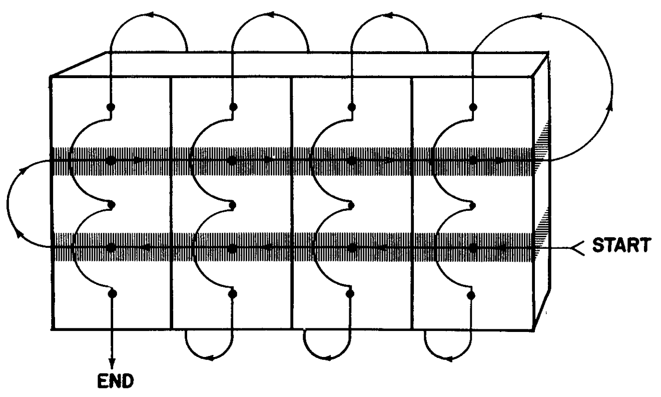


Fig. 9—Illustration of assigning an order to program operation on cells to accommodate specialized calculations or to increase the rate of convergence.

for one cell the address of the location of the next cell to be treated. In certain applications of the program, assigning an order to the operation on cells can facilitate the solution of the system of equations. If an order of rows of the matrix is known to yield faster solution, this order can be used in applying the Gauss-Seidel method. Moreover, ordering program operation on cells provides a simple means for utilizing the symmetry of many elements of the matrix. The conduction-convection terms between two connected cells are symmetrical, and considerable computing time is saved if these terms are computed once for each time step. One means for utilizing the symmetry of coefficients is to order the assembly of elements by increasing or decreasing machine address.

In conclusion, several interesting methods and computational techniques are incorporated in the transient heat transfer program to allow its application to a wide range of problems. The program has been used to analyze large problems incorporating unusual shapes and several combined modes of heat transfer. The machine time and consequent cost required for the solution of most problems is very reasonable. For example, typical problems with 150 nodes and 100 transient time steps are run in about ten minutes of 704 computer time. It is economical to use the heat transfer program to solve extensive problems and to obtain solutions that are considerably more adequate than those resulting from other methods. Consequently, extensions and further application of the ideas used in the present program are planned for the solution of other engineering problems.

Discussion

J. Ricketts (AC Spark Plug): Where can detailed information concerning the heat transfer program be requested?

Mr. Campbell: Information can be obtained by writing to me. However, I am afraid that you cannot obtain the program, because a large part of the information is company priority. I would be glad to discuss this with you or tell you as much as I can, but unfortunately the usual restrictions are hampering this exchange of ideas.

By the way, let me introduce Mrs. Vollenweider who is an expert on a different part of the programs and is here to answer the questions that I can't.

Mr. Ricketts: How long would a large problem take?

Mrs. Vollenweider: Large problems have been worked in approximately three minutes of computing time. One of the assets of the program is its short running time.

L. Hellerman (IBM): Breaking a solid into cells suggests G. Kron's "tearing". Could you comment on this?

Mr. Campbell: Neither of us know enough about this.

Mr. Hellerman: Has there been any thought of using your method of geometric specification in the control of machine tools?

Mrs. Vollenweider: Yes, and we are investigating this now.

H. T. Middleton (IBM): Will this program be available through SHARE?

Mrs. Vollenweider: Information on the solution methods we used is available. However, the actual program is not available except to other GE installations.

F. Engel (Westinghouse): What assumptions are made in arriving at the difference approximations for the conduction terms considering the irregular geometry permitted?

Mr. Campbell: The assumptions that are made in this case are, of course, the assumptions one usually makes for a finite different mesh. I believe it is clear that if you divide an object into cells in the same manner that a flux plot would break such an object into cells you have a better chance of getting reasonable answers. Our method does allow you to do this if you are so inclined.

The Automatic Transcription of Machine Shorthand*

GERARD SALTON†

Introduction

THE RECOGNITION of digitalized speech signals is an important data processing problem, still completely unsolved. In fact, speech is the basic input to many data processing problems, and before any processing can take place, the speech signals must first be transformed into digital form, recognized, and identified. Typical applications include: the processing of court testimony, the recording of data in medical practice, the handling of business correspondence, and the monitoring of radio broadcasts.

In spite of the fact that considerable research has been conducted within the past few years,[1,2,3,4,5,6] the problems which arise in the recognition of speech are not, as yet, close to solution. For this reason, it must be expected that speech cannot in the near future be made available automatically as an input for data processing purposes.

Since stenographers are trained to record spoken information at high speed, it may be possible to solve a substantial part of the problem of transforming spoken information into written form by using a stenographic transcript as machine input. Indeed, stenographic notes are written in standard form and rules exist for transforming the phonemic stenograms into the standard written language. Machine shorthand is the most useful form of input in this connection, since the problem of recognizing handwritten characters is eliminated. Moreover, in machine shorthand, the notes are furnished on a paper tape which may serve as a direct input to a computer handling the translation process. A study of methods for transcribing machine shorthand is thus of interest from a practical point of view to speed up the production of written output, and also because it provides opportunities for the analysis of both the written and the spoken language.

The Stenotype Machine

Most machine shorthand methods presently in use are based on a machine developed by a shorthand reporter, W. S. Ireland,[7] in the beginning of this century. The machines resemble miniature typewriters and are marketed under such names as "*Stenotype*" or "*Stenograph.*"

The keyboard developed by Ireland comprises 22 keys and a numeral bar. As each key is depressed, a character corresponding to that key is printed on a paper tape and a clutch driven roller advances the tape. The various keys can be depressed either singly or together; if depressed together, the corresponding characters all print on the same line of the tape. During actual operation an average of five to six characters are thus written across the width of the tape in a single *stroke*. Unlike an ordinary typewriter, the stenotype machine does not have a movable carriage; therefore, the characters always print in the same relative position on the tape when the corresponding keys are depressed on the keyboard. The paper tape folds into a pull-out tray at the rear of the device. The keys are unmarked and writing is done entirely by touch as in ordinary typewriting. A sample stenotype tape is shown in Fig. 1.

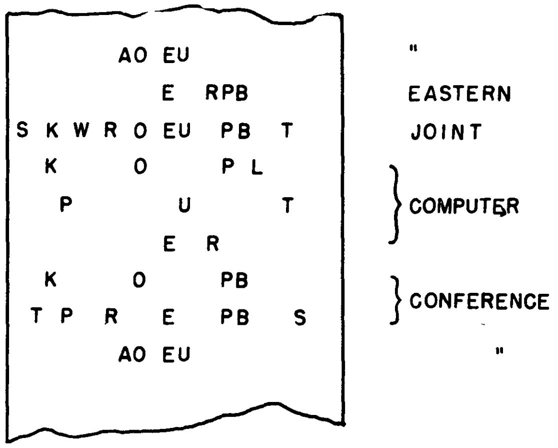


Fig. 1—Sample stenotype tape.

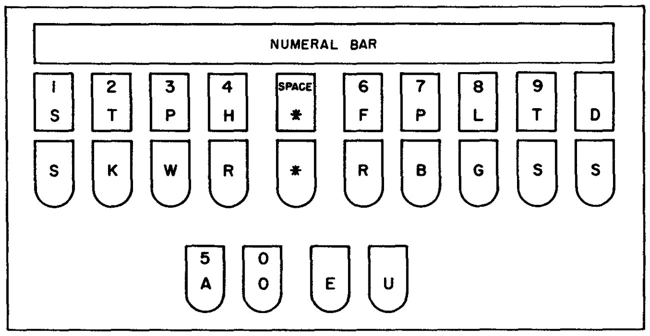


Fig. 2(a)—Stenotype keyboard.

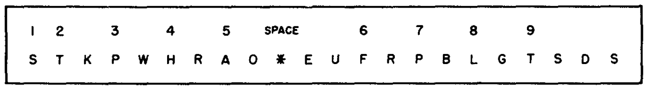


Fig. 2(b)—Printing order.

The standard stenotype keyboard is shown in Fig. 2(a) and the fixed printing order is shown in Fig. 2(b). If the numeral bar is not depressed, the lower

* This study was supported in part by the National Science Foundation.

† Computation Laboratory, Harvard University.

case character shown on each key is written; if the numeral bar is depressed, a decimal digit will print in some instances while the lower case character is printed unchanged in others. Thus for 12 of the 22 character positions across the width of the tape, the character appearing on the tape is unique, while for the remaining ten positions two possible characters can be printed in each position.

The 16 different letters of the alphabet which can be printed by means of the 22 keys are arranged in three groups: a set of seven consonants on the left-hand side of the keyboard, a set of four vowels and an asterisk in the middle, and a set of ten consonants on the right hand side of the keyboard. In order to keep the keyboard small and the fingering simple, ten letters of the alphabet are not provided at all and must therefore be represented by others letters or combinations of letters.

Basic Rules of Stenotypy

Before describing the problems which arise in the transcription of stenotypy, the basic rules of machine shortland will be briefly reviewed.[8,9] While there exist many variations of the basic theory,[10,11] resulting partly from the preferences of individual reporters and stenographers, and partly from the desire to cope with specialized subject matter and to increase reporting speed, all such variations are based on a common set of rules. It is this standard theory which is described here.

Consonant and Vowel System

In machine shorthand, consonants are generally written according to sound. Of the 24 consonant phonemes which are recognized in most English dialects,[12] 22 can occur initially in English words and 21 can occur finally. Direct representations are assigned on the stenotype keyboard for seven initial and for nine final consonant phonemes. Special letter combinations are used to represent those phonemes which are not available directly on the keyboard. The character combinations actually used are assigned for ease of fingering, rather than for economy in the number of keys which must be depressed.

The vowel system is more complicated in that vowels are written partly according to spelling and partly according to sound. A single vowel occurring in a word is generally written in accordance with the English spelling. Thus the phoneme /əh/ is reproduced as the letter *E* in HER and as the letter *U* in BURN. A vowel cluster is generally transcribed as a single vowel if only one vowel in the cluster is prominently sounded. Thus one writes BOT for "bought" or "boat," and BET for "beat" and "beet." Some vowel clusters do not follow the general rule: for example, the cluster "oo" as in "book" is transcribed *AO,* and the phoneme /ɔj/ as in "noise" is transcribed *OEU.*

Words are formed in stenotyping by merely juxtaposing the transcriptions of the individual phonemes. Thus,

"one" is written WOPB (WON),

"queer" is written KWER,

and

"tax" is written TABGS (TAKS).

The Elimination of Strokes

Since only one vowel cluster can be printed on any given line, it is in general not possible to write more than one syllable per stroke. Thus, the word "interviewed" is written

EUPB = TER = SRUD (IN = TER = VUD)

in three strokes, the equality sign denoting the separation of strokes.

Furthermore, because of the fixed order of the letters on the keyboard, some words require more than one stroke even though only one vowel cluster is present. Thus it is necessary to use two strokes for words such as

TH = WART (TH = WART),

PWRAPB = FP (BRAN = CH),

and

TKPWOL = F (GOL = F).

Theoretically, at least two strokes should thus be required on the average to render a given English word on a stenotype tape. In practice, however, the number of stenotype strokes is only slightly larger on the average than the number of English words.

The reduction from the expected number of strokes to the actual number of strokes is achieved by a set of rules designed to save strokes. Seven principal rules are used to eliminate strokes:

(1) Special keyboard letter combinations are assigned to some phoneme clusters such as /ŋk/ and /nʒ/ which would normally require more than one stroke.

(2) Certain high frequency consonant clusters ending in "t" cannot be written in one stroke because of the arrangement of the letters on the keyboard or because the required fingering would be too uncomfortable. For this reason final T is omitted in words ending in /st/, /kst/, and /kt/.

(3) Completely silent letters are treated as redundant and are therefore not written.

(4) An "unimportant" or "unaccented" vowel occurring in the middle of a word may be omitted if a stroke is thereby saved. Vowels which are not essential to the understanding of a word are therefore generally not written. Thus,

"eas*i*ly" is written ES = HREU (ES = LI), "b*e*lieve" is written PWHREF (BLEV),

and

"*com*mit" is written KPHEUT (KMIT).

(5) Special letter combinations are assigned to some high frequency affixes. In particular, special combinations are assigned to the prefix "ex" and to the suffixes "ing," "ity," "ment," "sion" and "tion," and "ction."

(6) Special abbreviations are used to represent about 300 high frequency words. These abbreviated forms can be used separately or as part of longer derived words. For example:

TA stands for "take,"

but

PHEUS = TA (MIS = TA) stands for "mistake."

Similarly,

TKEU (DI) stands for "difficulty,"

but

TKEUS (DIS) stands for "difficulties."

(7) Certain phrases consisting of more than one high-frequency word may be written in one stroke by juxtaposing the corresponding abbreviations. Thus in a single stroke one writes phrases such as

THAUFB → that you have been,
(THA U F B)

SWELS → as well as,
(S WEL S)

and

KWRUR → why you are
(KWR U R).

The Problems of Stenotype Translation

A number of problems must be considered if a stenotype input text is to produce usable English output. First of all, it is necessary to find procedures for mechanizing the input of stenotype texts. Second, methods must be sought for dealing with the large number of multiple English correspondents which are generated in the translation process. Finally, it is desirable to restitute correct English word forms for the mutilated forms which are derived. These problems will be dealt with in order.

The Input Problem

The paper tape prepared by the stenographer should clearly be used as a direct input medium during an automatic transcription process. Two main procedures are available for this purpose. The first one consists in modifying the stenotype machine in such a way that holes are punched into the tape between adjacent lines of printed information. The tape can then be treated in the same manner as ordinary punched paper tape; at the same time, the printed information can still be read as before. The second possible procedure consists in using the characters printed on the tape directly, by means of some character recognition process. This second method requires no equipment modifications in the recording device itself.

For the twelve character positions which are associated with a unique character it is sufficient to detect a hole immediately above the given character position on the tape. The presence of the hole will indicate the presence of the given character on the line printed immediately below. Alternatively, in the character recognition procedure it is sufficient to discriminate between the presence and absence of a character in the given character position. If a character is present, it will then necessarily be that character which is uniquely assigned to the given position on tape.

Each of the remaining ten character positions may contain either of two different characters. The character recognition process must therefore recognize two possible characters in each of these tape positions. This is a relatively simple system in comparison with many other recognition systems now operating, and no insurmountable technological difficulties should arise.

If a punched paper tape system is used, it is sufficient to detect the presence of decimal digits across the tape by using one extra hole position. If the extra position is punched, the characters on the line immediately below will be recognized as decimal digits in all positions where a possible conflict exists. If the extra hole position is left unpunched, only alphabetics will be recognized. The required decoding equipment is again extremely simple.

The Decoding of Abbreviations, Phrases, and Affixes

The number of abbreviated forms in general use is relatively small, of the order of several hundred. It is therefore possible to construct a table or dictionary of these forms and to obtain the English correspondents by a table look-up operation. In principle, the table look-up operation presents no difficulties. In practice, however, certain ambiguities may have to be resolved. Indeed, many abbreviations have a number of different correspondents in English and it is not always easy to discriminate between them. For example, TPOR is an abbreviation for both "inform" and "information"; similarly, W stands for both "were" and "with."

The difficulty is often compounded by the fact that many clusters may be abbreviated forms in some contexts, but not in others. Thus, the previously cited TPOR is transliterated "FOR" and may, therefore, correspond also to "four" and "for." Similarly, UL is the abbreviated form for "you will" since U stands for "you" and L stands for "will"; however, in some contexts it may be translated as merely "ul" as, for example, in TPEBG = *UL* = EU (effect*ually*).

Phrases are strings of abbreviated forms which may be written in one stroke. The dictionary of abbreviations can be used for the decoding of phrases, provided each phrase is properly partitioned before the table look-up operation. Consider as an example the phrase URBGT. If it is partitioned U/RB/G/T the table look-up operation will produce the correspondents

$$\text{U} \rightarrow \text{you}$$

$$\text{RB} \rightarrow \text{shall}$$

$$\text{G} \rightarrow \text{go, gone}$$

$$\text{T} \rightarrow \text{it, the}$$

and the most reasonable translation would be "you shall go the"; if, on the other hand, it is partitioned UR/BGT, the look-up operations produce

$$\text{UR} \rightarrow \text{you are, your}$$

$$\text{GBT} \rightarrow \text{account}$$

and the correct translation "your account" is then obtained.

While each phrase can generally be partitioned in many different ways, a preferred partitioning method can normally be found by storing certain phrases or parts of phrases in the dictionary, and by using the stored information during the translation process.

The number of affixes in general use is small. However, the decoding of affixes can present considerable problems. In particular, a letter cluster denoting an affix may generally also stand for a normal word ending, or it may represent an abbreviated form. Consider, for example, the letter G, denoting the suffix "ing." It may be variously translated as "g," "ing," "go" or "gone," or it may be a part of a longer letter cluster requiring still another translation.

Furthermore, a certain amount of conflict is inherent in the assignment of letter clusters to the various suffixes. Thus, the suffix "tion" is represented by "GS"; since G also stands for the suffix "ing," the ending GS can also be decoded as "ings."

In general, it is possible to distinguish three types of ambiguity resulting from the decoding process. First, the generation of improper English word forms as, for example, the translation of OFRGS by "offertion" instead of "offerings." This problem will be considered in the next section. The second is the generation of phrases which are made up of morphologically correct word forms, but are nevertheless syntactically improper. Such syntactically incorrect phrases would result, for example, from the replacement of "issue" by "I shall you," or of "a chief" by "achieve." The last is the generation of semantic ambiguities which cannot be resolved merely by syntactic analyses. Specifically, it is not possible to use English syntax to distinguish "sport" from "export," or "annex" from "annection."

Syntactic difficulties are generally easier to resolve than semantic problems. The former can to some extent be handled by building a dictionary or table including grammatical designations. Some sequences of grammatical forms will then be recognized as legitimate in English, while others will not. Semantic ambiguities, on the other hand, must be resolved by considering the subject matter being analyzed. This, in turn, might be attempted by generating and later identifying certain topic phrases or topic sentences by means of frequency analyses of the stenotype texts.[13,14] While frequency studies are helpful in analyzing semantic content, it is doubtful whether all semantic problems can be easily eliminated.

The Transliteration of Stenotype Clusters and the Determination of Word Boundaries

One of the required steps in the automatic transcription of stenotype texts is the substitution of English letters for the stenotype clusters. For most clusters, this step is mechanical, since the clusters correspond in general to a unique character in English. For example, PWEFP will be uniquely transformed into "BECH" since the transformations

$$\text{PW} \rightarrow \text{B},$$

$$\text{E} \rightarrow \text{E},$$

and

$$\text{FP} \rightarrow \text{CH}$$

are uniquely determined.

A few clusters, however, cannot be identified uniquely without use of the context. Thus SHR corresponds either to "sl" or to "shr" and both translations are admissible in some cases. For example, SHRED might correspond to "shred" or "sled." Ambiguous clusters of this type are fortunately rather rare.

The transliteration procedure does not restitute missing letters nor restore the correct English spelling. A first approach to this problem might be provided by the use of frequency tables for both stenotype and English digrams and higher order strings. These tables could be used to translate a given stenotype letter cluster into the corresponding English letter cluster with the highest frequency count. In most cases, the correct English correspondent would then be furnished.

In some cases, of course, the frequency tables would not avail, since the most probable correspondent is not always the correct correspondent. This is the case particularly for those words which require more than one stroke to be rendered on the stenotype tape. For example, words like

$$\text{PA} = \text{SHEPBS (PA} = \text{SHENS)} \rightarrow \text{patience}$$

or

$$\text{EUPB} = \text{TKUS} = \text{TREUL (IN} = \text{DUS} = \text{TRIL)} \rightarrow \text{industrial}$$

cannot properly be generated only by a frequency analysis of the individual strokes. Moreover, there are many stenotype strokes which may legitimately have more than one correspondent in English and in general, frequency tables cannot be used to make a choice among these correspondents. Consider, for example,

$$\text{HEUR standing for} \begin{cases} \text{hire} \\ \text{higher,} \end{cases}$$

$$\text{A} = \text{CUT standing for} \begin{cases} \text{a cut} \\ \text{acute,} \end{cases}$$

and

$$\text{PWAR standing for} \begin{cases} \text{bar} \\ \text{bear} \\ \text{bare.} \end{cases}$$

Only the last two words are legitimate homonyms in English. The other pairs are artificial homonyms which are created in stenotyping because the language is less redundant than English.

Thus, while digram and trigram frequency tables would prove helpful in the restoration of English word forms, accurate tables for longer strings of characters would almost certainly be required to solve the problem for a large number of strokes. A dictionary of high-frequency multistroke words could also be used to good advantage in the recognition of word boundaries. However, since it is often possible to split English words into stenotype strokes in several different ways, all possible forms of these multistroke words would have to be listed in the dictionary.

To summarize, the problems raised by the transcription of machine shorthand are identical to problems encountered in the automatic translation of languages with two important exceptions. In machine shorthand, it is not necessary to change the word order of the input text, or to alter a sentence by insertion or deletion of words; idiomatic expressions do not therefore cause any difficulties. On the other hand, in language translation it is not necessary to generate correct word forms by altering the spelling of words, or to recognize word boundaries. A comparison of the various translation problems is shown in Table I.

TABLE I

COMPARISON OF TRANSLATION PROBLEMS

Translation Problem	Automatic Translation	Stenotype Transcription
Automatic input	√	√
Dictionary look-up	√	√
Elimination of syntactic ambiguities	√	√
Elimination of semantic ambiguities	√	√
Generation of correct word forms	* —	√
Recognition of word boundaries	—	√
Restitution of correct word order	√	—
Insertion and deletion of words	√	—

* Limited to generation of correctly inflected forms.

THE PRODUCTION OF PSEUDO-ENGLISH

As a first step in the analysis of machine shorthand, computer routines were prepared for the production of pseudo-English, that is, comprehensible, although syntactically and semantically ambiguous English. In many applications, such as the monitoring of radio broadcasts or the use of machine shorthand in medical practice, pseudo-English might be an acceptable form of final output; in other applications, imperfect English would be undesirable. However, even then, pseudo-English is a useful intermediate output since it can be used to update the dictionaries, to refine the partitioning methods, and to derive rules for syntactic analyses. Since English is inherently redundant, understandable output sentences can be produced with comparatively little difficulty.

A variety of methods are used to produce pseudo-English. The stenotype strokes are first partitioned into "constituent" pieces using only the structure of each stroke as the criterion. Two short dictionaries are used to look up the correspondents of these constituents and to handle the transliteration process. The word pieces are later reassembled into complete English words or phrases.

No attempt is made to reduce ambiguity by making a choice among many possible English correspondents. Thus, if a given stenotype stroke can reasonably correspond to many different English letter clusters or words, all correspondents will be listed, and the reader is required to choose the most reasonable correspondent using the given context as a guide.

Similarly, the correct English spelling of words not found in the dictionaries is not always restored. Instead, the abbreviated word form, excluding silent letters and unaccented vowels, is used in the transcribed text. This reduces the aesthetic quality of the output although comprehension is not generally impaired.

Analysis of Stenotype Stroke Forms

Depending on the classification of a given stroke, one of five basic translation methods must be used to transform it into the English equivalent.

(1) To decode *abbreviations* the complete stenotype stroke must be looked-up in a dictionary of abbreviations; one or more correct English word forms will then be determined for each abbreviation. For example,

$$T \rightarrow \text{the, it.}$$

(2) *Phrases* must first be partitioned into two or more components, and each component must then be looked-up separately in a dictionary. The juxtaposed English correspondents will then represent the English phrase. For example

$$TPHAB \rightarrow \text{it may be,}$$

since

$$T \rightarrow \text{it}$$

$$PHA \rightarrow \text{may}$$

$$B \rightarrow \text{be.}$$

An analysis of stenotype phrases indicates that many phrases can be translated correctly if partitioned between the middle vowel string and the final consonant string. The above example would then be partitioned into the components TPHA and B, and the English correspondents "it may" and "be" would be found in the dictionary of abbreviations and phrases.

(3) *Derivatives of abbreviations* are first partitioned to isolate the affix. The correspondent of the complete stroke excluding the affix is then determined by dictionary look-up and the English correspondent of the affix is added. For example,

$$WREUG \rightarrow \text{write-ing,}$$

since

$$WREU \rightarrow \text{write}$$

$$G \rightarrow \text{ing.}$$

Here WREU is the abbreviation for "write" and G is the suffix.

(4) Stenotype strokes which are not abbreviations or phrases are rendered in English by transforming the stenotype letter clusters into corresponding English letters. The transliteration procedure uses a dictionary of stenotype letter clusters including many of the consonantal and vowel patterns used in English. In order to use the dictionary of letter clusters, it is necessary to partition each stenotype stroke into initial consonant string, middle vowel string, and final consonant string. For example,

$$KHRERBG \rightarrow \text{cl-e-rk,}$$

since

$$KHR \rightarrow \text{cl}$$

$$E \rightarrow \text{e}$$

$$RBG \rightarrow \text{rk.}$$

It should be noted that many stenotype clusters appear both in the dictionary of letter clusters and in the dictionary of abbreviations. However, the English correspondents are then necessarily quite different. As an example, it was seen that T is an abbreviation for "it" and "the"; in the dictionary of letter clusters, the correspondents are "t" and "th."

(5) Ordinary stenotype strokes which include a suffix but are not stenotype abbreviations are treated like the strokes of category (4), except that they are partitioned into four parts before look-up in the dictionary of letter clusters: initial consonant string, middle vowel string, final consonant string excepting the suffix, and suffix. As an example,

$$TKPWEUFG \rightarrow \text{giving,}$$

since

$$TKPW \rightarrow \text{g}$$
$$EU \rightarrow \text{i}$$
$$F \rightarrow \text{v}$$
$$G \rightarrow \text{ing,}$$

G being again the suffix corresponding to "ing.'

Since it is not clear *a priori* how a given stroke is to be classified, every stroke is submitted to the procedure required for all five categories; a correct translation will then necessarily be determined for each stroke. The strokes are first classified as belonging to one of ten *types*. The stroke type depends only on the combination of letter clusters which form the stroke. Specifically, the stroke type indicates whether the corresponding stroke terminates in one of the recognized suffixes and shows which combination of initial consonant, middle vowel, and final consonant strings constitutes the stroke. For example, stroke type O represents strokes made up of a vowel string only, while stroke type 2 represents strokes made up of an initial consonant string followed by a vowel string. Depending on the type, each stroke is then partitioned into constituent *parts*. Eight stroke parts including the complete stroke are recognized as shown in Table II. The English correspondents of the various parts are then looked-up in the dictionary of abbreviations and phrases, and in the dictionary of letter clusters. A third dictionary or table is required to handle suffixes.

After the table look-up operations, the English correspondents of the stroke parts must be reassembled to form complete words or phrases. For example, to form complete phrases (category (2) above) stroke

TABLE II

STENOTYPE STROKE PARTS

Code	Stroke Part
W	Complete stroke
C	Initial consonant string and middle vowel string
F_1F_2	Final consonant string
I	Initial consonant string
V	Vowel string
S	Complete stroke excluding suffix
A	Suffix
E	Final consonant string excluding suffix

parts C and F_1 must be reassembled; similarly, for ordinary transliterated words (category (4)) parts I, V and F_2 are reassembled. This will be explained further in the description of the machine program.

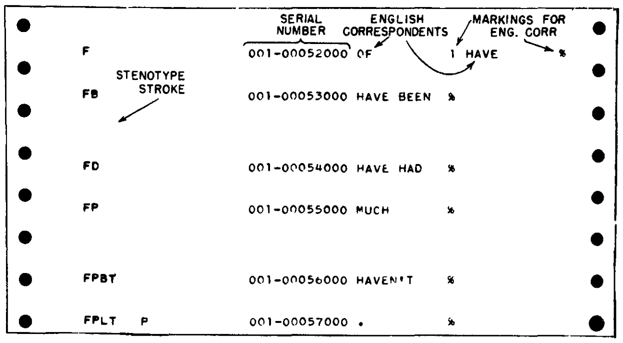
The Machine Program

The Univac I computer at the Harvard Computation Laboratory was used for the production of the pseudo-English output. In order to reduce the programming load, it was desirable to make use of existing machine programs. In particular, it was found that many programs originally written to handle various phases of Russian to English automatic translation could be used for the transcription of machine shorthand*. Some of these programs required small alterations; however, compared with the work required to write new programs, the effort expended in making the required changes was negligible.

In order to construct the two dictionaries, use was made of existing stenotype manuals. The present dictionary of abbreviations includes about 700 stenotype strokes arranged in alphabetical order, and approximately 2000 English correspondents. Punctuation marks are identified by the letter P following the stenotype cluster, and English correspondents by the digits 1, 2, 3, and so on; a percent (%) sign is used for the last correspondent. An excerpt from the dictionary of abbreviations is shown in Fig. 3.

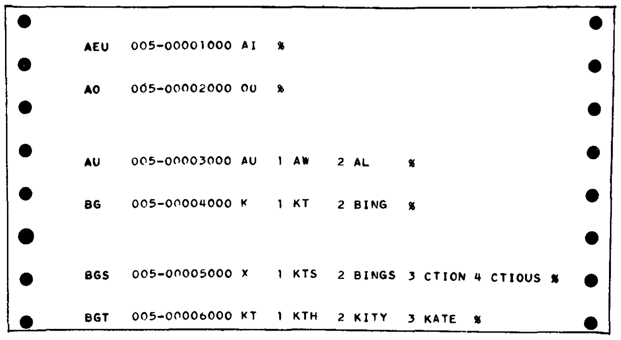
The dictionary of letter clusters includes only clusters whose English correspondents are different from the stenotype original. To construct this dictionary, lists of consonantal and vowel patterns used in English were consulted;[15] about 100 letter clusters arranged in alphabetical order are included. An excerpt from the dictionary of letter clusters is shown in Fig. 4.

* Several programs written originally by members of the Harvard Automatic Translation Project under sponsorship of the National Science Foundation were used; the writer is indebted to Prof. A. G Oettinger and to the other project members for having made these programs available.


SERIAL NUMBER
ENGLISH CORRESPONDENTS
MARKINGS FOR ENG. CORR
STENOTYPE STROKE

```
F        001-00052000 OF         1 HAVE     %
FB       001-00053000 HAVE BEEN  %
FD       001-00054000 HAVE HAD   %
FP       001-00055000 MUCH       %
FPBT     001-00056000 HAVEN'T    %
FPLT  P  001-00057000 .          %
```

Fig. 3—Excerpt from dictionary of abbreviations and phrases.



```
AEU   005-00001000 AI   %
AO    005-00002000 OU   %
AU    005-00003000 AU   1 AW   2 AL    %
BG    005-00004000 K    1 KT   2 BING  %
BGS   005-00005000 X    1 KTS  2 BINGS 3 CTION 4 CTIOUS %
BGT   005-00006000 KT   1 KTH  2 KITY  3 KATE  %
```

Fig. 4—Excerpt from dictionary of stenotype letter clusters.

The various steps used to produce pseudo-English are shown schematically in Fig. 5. Each box corresponds to one machine run. Only the partition pro-

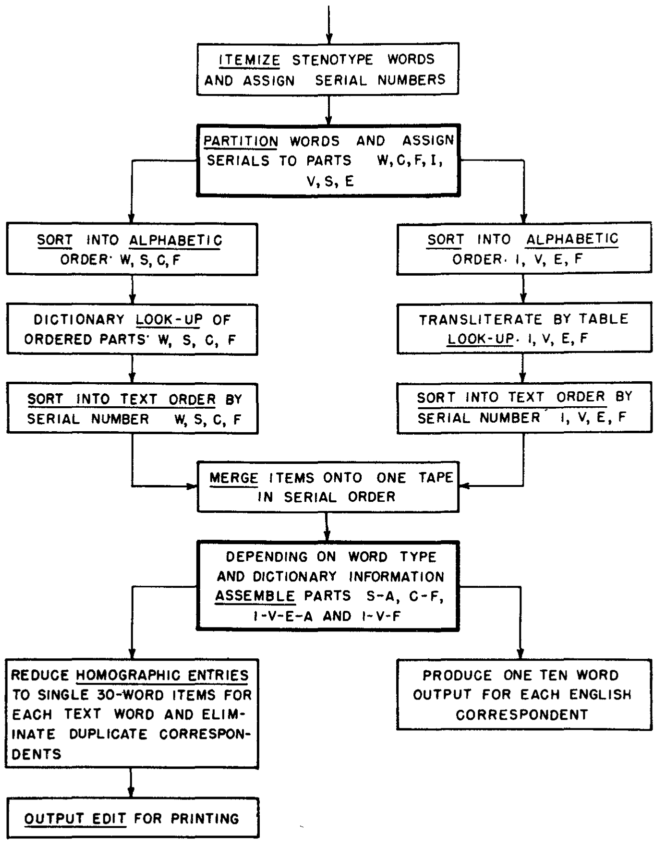
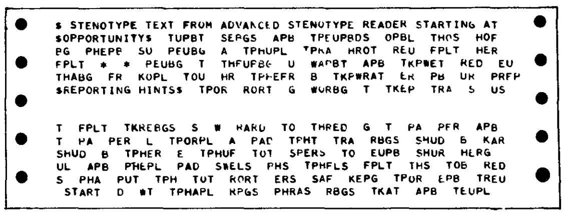

ITEMIZE STENOTYPE WORDS AND ASSIGN SERIAL NUMBERS
PARTITION WORDS AND ASSIGN SERIALS TO PARTS W, C, F, I, V, S, E
SORT INTO ALPHABETIC ORDER· W, S, C, F
SORT INTO ALPHABETIC ORDER· I, V, E, F
DICTIONARY LOOK-UP OF ORDERED PARTS· W, S, C, F
TRANSLITERATE BY TABLE LOOK-UP· I, V, E, F
SORT INTO TEXT ORDER BY SERIAL NUMBER W, S, C, F
SORT INTO TEXT ORDER BY SERIAL NUMBER I, V, E, F
MERGE ITEMS ONTO ONE TAPE IN SERIAL ORDER
DEPENDING ON WORD TYPE AND DICTIONARY INFORMATION ASSEMBLE PARTS S-A, C-F, I-V-E-A AND I-V-F
REDUCE HOMOGRAPHIC ENTRIES TO SINGLE 30-WORD ITEMS FOR EACH TEXT WORD AND ELIMINATE DUPLICATE CORRESPONDENTS
PRODUCE ONE TEN WORD OUTPUT FOR EACH ENGLISH CORRESPONDENT
OUTPUT EDIT FOR PRINTING

Fig. 5—Program for the production of pseudo-English.

gram and the assembly program shown by double lines were newly programmed.*

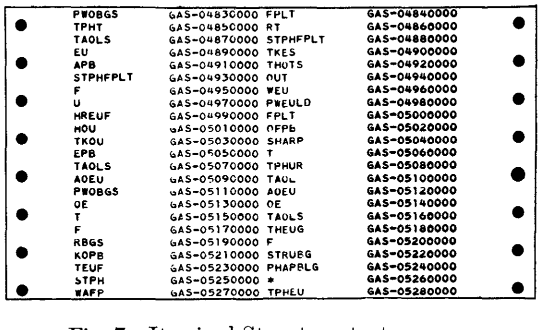
The stenotype text to be translated is first transferred onto magnetic tape. Since equipment for automatic input does not exist at present, the operation is manual. The text is typed like ordinary linear text with spaces between stenotype strokes. English comments are placed between dollar ($) signs to conform with the requirements of the existing programs.[16] No special punctuation marks are used in stenotyping except for a double asterisk to indicate a new paragraph. Instead, punctuation marks are denoted by special letter clusters which are included in the dictionary of abbreviations. An excerpt of a linear input text is reproduced in Fig. 6.



```
$ STENOTYPE TEXT FROM ADVANCED STENOTYPE READER STARTING AT
$OPPORTUNITY$  TUPBT  SEPGS  APB  TPEUPBDS  OPBL  THOS  HOF
PG  PHEPP  SU  PEUBG  A  TPHUPL  TPHA  HROT  REU  FPLT  HER
FPLT  *  *  PEUBG  T  THFUPBG  U  WAPBT  APB  TKPWET  RED  EU
THABG  FR  KOPL  TOU  HR  TPHEFR  B  TKPWRAT  ER  PB  UR  PRFP
$REPORTING HINTS$  TPOR  RORT  G  WORBG  T  TKEP  TRA  S  US

T  FPLT  TKREBGS  S  W  HARD  TO  THRED  G  T  PA  PFR  APB
T  PA  PER  L  TPORPL  A  PAD  TPHT  TRA  RBGS  SHUD  B  KAR
SHUD  B  TPHER  E  TPHUF  TOT  SPERS  TO  EUPB  SHUR  HERG
UL  APB  PHEPL  PAD  SWELS  PHS  TPHFLS  FPLT  THS  TOB  RED
S  PHA  PUT  TPH  TOT  RORT  ERS  SAF  KEPG  TPOR  EPB  TREU
 START  D  *T  TPHAPL  RPGS  PHRAS  RBGS  TKAT  APB  TEUPL
```

Fig. 6—Sample section of linear Stenotype text.



```
PWOBGS      GAS-04830000 FPLT        GAS-04840000
TPHT        GAS-04850000 RT          GAS-04860000
TAOLS       GAS-04870000 STPHFPLT    GAS-04880000
EU          GAS-04890000 TKES        GAS-04900000
APB         GAS-04910000 THOTS       GAS-04920000
STPHFPLT    GAS-04930000 OUT         GAS-04940000
F           GAS-04950000 WEU         GAS-04960000
U           GAS-04970000 PWEULD      GAS-04980000
HREUF       GAS-04990000 FPLT        GAS-05000000
HOU         GAS-05010000 OFPB        GAS-05020000
TKOU        GAS-05030000 SHARP       GAS-05040000
EPB         GAS-05050000 T           GAS-05060000
TAOLS       GAS-05070000 TPHUR       GAS-05080000
AOEU        GAS-05090000 TAOL        GAS-05100000
PWOBGS      GAS-05110000 AOEU        GAS-05120000
OE          GAS-05130000 OE          GAS-05140000
T           GAS-05150000 TAOLS       GAS-05160000
F           GAS-05170000 THEUG       GAS-05180000
RBGS        GAS-05190000 F           GAS-05200000
KOPB        GAS-05210000 STRUBG      GAS-05220000
TEUF        GAS-05230000 PHAPBLG     GAS-05240000
STPH        GAS-05250000 *           GAS-05260000
WAFP        GAS-05270000 TPHEU       GAS-05280000
```

Fig. 7—Itemized Stenotype text.

The *itemize* program[16] is used first to assign serial numbers to the individual text words. Each text word is transformed into a separate 5-word item. A part of a linear input text is shown in itemized form in Fig. 7. The four low-order digit positions of each serial number, filled with zeros, are later modified during the partitioning process.

The *partitioning* program uses the itemized text as input and generates a variable number of 5-word items for each input item. Each input stroke is first partitioned into initial consonant, middle vowel and final consonant strings, and a code representing the stroke type is assigned to the complete stroke. A test is also made for the presence of one of the ten recognized suffixes. If a suffix is found the proper suffix number is assigned. The various stroke parts are then recorded on one of two output tapes (the final consonant string F is actually recorded on both tapes). The dictionary of abbreviations and phrases will be searched during the dictionary look-up routine for the English correspondents of the stroke parts of tape 1, while the dictionary of letter clusters is used for the stroke parts of tape 2. For complete strokes (initial consonant — middle vowel — final consonant clusters) with affix, four stroke parts (W, C, F, S) are recorded on tape 1 and four stroke parts (I, V, F, E) are recorded on tape 2. If the stroke is truncated (*e.g.*, if the initial or the final consonant string is missing) or if no affix is present, a smaller number of stroke parts will be recorded out. All stroke parts are recorded with a modified serial number to show the stroke type and the stroke part represented by each individual item.

* The partition and assembly programs were coded by Mr. Rodney W. Thorpe, whose valuable assistance is gratefully acknowledged.

The two tapes are now processed separately through the same sequence of programs. The items are first prepared for the dictionary look-up by an alphabetic sort. The dictionary look-up routine will then expand each 5-word item into a 30-word item and add to each stenotype input stroke the English correspondents from the dictionary. The dictionary of abbreviations is used for the items of tape 1, while the dictionary of letter clusters is used for tape 2. For strokes not found in the dictionary, a dummy 30-word item is generated. The 30-word items are finally sorted back into text order, using the new serial number as a sorting key. The alphabetic sort and the dictionary look-up programs were taken over unchanged from the "Continuous Dictionary Run" used for Russian-English translation.[16]

The next program is a *two-way merge* program using the serial number as a key and tapes 1 and 2 as inputs. One merging pass is sufficient to transfer all 30-word items onto a single output tape. The merge program used is one of the existing Univac system routines.[17]

The assembly program is now used to assemble those 30-word items which do not correspond to a complete text word. Both the stroke type and the dictionary information obtained during dictionary look-up are used during the assembly phase to generate complete English correspondents. The stroke type indicates whether it is necessary to add a suffix to the English correspondents found in the dictionary and shows how many items are to be assembled to form complete English correspondents. For example, if the original stroke consisted of an initial consonant and a middle vowel string, the English correspondents of two items must generally be assembled; if a final consonant string was also present, three items are normally assembled.

Up to five different types of assemblies are made, For abbreviations, the English correspondents of the W item are taken intact; for phrases, the correspondents of the C and F_1 items are assembled; for derivatives of abbreviations, the affix A is attached to the correspondents of the S item; for ordinary strokes, the correspondents of the I, V, and F_2 items are attached; finally, for ordinary strokes which include a suffix, an affix A is attached to the correspondents of the assembled I, V, E items. These five types of assemblies are, however, not made for all strokes; the dictionary information obtained during the look-up operation is used to decide whether a given item is to be assembled or not.

Specifically, if the W item is contained in the dictionary, then the correspondents which would normally be generated by assembling the C-F, S-A, and I-V-E-A items are already contained in the W item, so that no need arises to assemble these other items. Similarly, if C and F_1 are not both in the dictionary, the C-F_1 item is not assembled since this item could not then be an English phrase. Finally, if S-A is to represent an abbreviation followed by a suffix, the S item must have been found in the dictionary; the S-A item is then assembled only under these conditions. The I-V-F_2 item (and I-V-E-A item if different from I-V-F_2) is always assembled, even if one or more of I, V, and F are not found in the dictionary. In that case the English letter cluster is assumed to be identical with the stenotype letter cluster which is therefore assembled as part of the English correspondent. Each assembled stenotype stroke is recorded out on tape as a separate 30-word item.

Many stenotype dictionary entries have more than one English correspondent. In order to avoid too much fragmentation, special provisions are made during the assembly to write such multiple correspondents in parentheses. The reader must then choose one correspondent form each set of correspondents shown in parentheses. Consider, for example, the following sets of English correspondents:

I item: TH,

V item: O,

F_2 item: S, Z, ST.

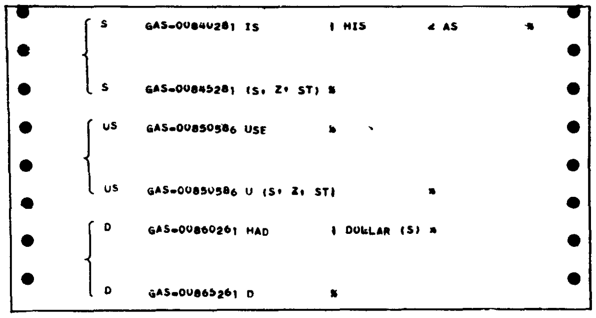
The English correspondent of the assembled I-V-F_2 item would be printed out

THO (S, Z, ST),

and the reader would normally choose the first correspondent between the parentheses to form THOS (those). A second output tape to be used for research purposes actually contains all correspondents "multiplied out."

At the end of the assembly each text word is represented by up to four 30-word items each including the stenotype original, the serial number assigned by the itemize routine, and the English correspondents. An output sample from the assembly routine is shown in Fig. 8.

It is now necessary to bring together all English correspondents belonging to a given stenotype stroke. This is done by reducing the assembled 30-word items corresponding to a given stroke to a single 30-word item including all distinct English correspondents.


```
S    GAS-00840281 IS         | HIS       < AS      *

S    GAS-00845281 (S, Z, ST) *

US   GAS-00850586 USE        *

US   GAS-00850586 U (S, Z, ST)         *

D    GAS-00860261 HAD        | DOLLAR (S) *

D    GAS-00865261 D          *
```
Fig. 8—Assembled Stenotype items.

The *homograph package* which was originally written to reduce all duplicate stem entries in the Russian dictionary to a single entry was adapted for this purpose.[18] As part of the "homograph" elimination, duplicate English correspondents which may have been generated during the assembly are condensed so that only distinct correspondents appear on the output tape.

The *output edit* program of the Continuous Dictionary Run is now used to edit the condensed output of the previous program. Each English correspondent is assigned a 24-character position (two Univac words), and up to five English correspondents of a given stenotype word are printed one underneath the other, thus making it easy to choose one correspondent for each stenotype word. An edited output text is read from left to right like ordinary English text. Multiple correspondents appearing one underneath the other are scanned, and the best correspondent is chosen.

The output edit program produces two outputs: the "interlinear translation" which includes the stenotype original with each set of English correspondents, and the "word-by-word translation" which omits the stenotype original. Excerpts of these two types of output are shown in Figs. 9 and 10, respectively.

For research purposes another edited output is prepared. This output is processed in such a way that a separate item is created for each English correspondent. A program originally prepared to aid in syntactic analysis and to correct the Harvard Automatic Dictionary Russian file is used.[19] An excerpt showing the listing of all English correspondents is shown in Fig. 11.

The several programs described above can be transformed into a single program by automatically reading into memory at the end of each step the program for the next step and by insuring that servo assignments are not conflicting. A flowchart for such a continuous run is shown in Fig. 12. Each box refers to one of the programs previously described. The programs marked CDR are part of the Continuous Dictionary Run, while the programs marked CST are on a Community Sequence Tape used for linguistic analysis. The broken line between the inverse sort and the alphabetic sort denotes that this set of pro-

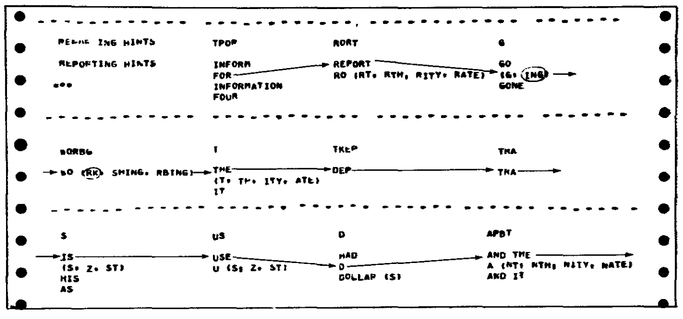


Fig. 9—Edited interlinear translation.

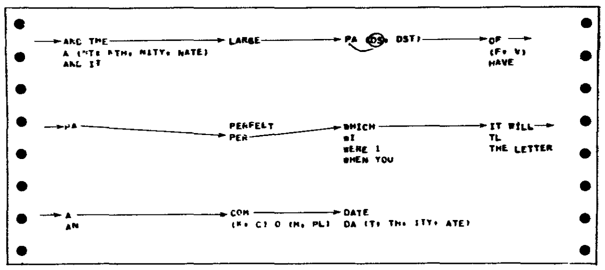
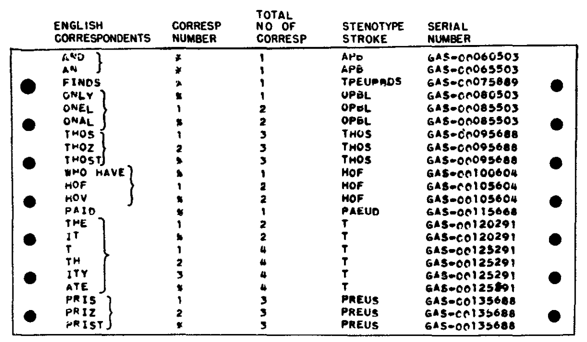


Fig. 10—Edited word-by-word translation.



ENGLISH CORRESPONDENTS	CORRESP NUMBER	TOTAL NO OF CORRESP	STENOTYPE STROKE	SERIAL NUMBER
AND	*	1	APD	GAS-00060503
AN	*	1	APB	GAS-00065503
FINDS	*	1	TPEUPRDS	GAS-00075889
ONLY	*	1	OPBL	GAS-00080503
ONEL	1	2	OPBL	GAS-00085503
ONAL	*	2	OPBL	GAS-00085503
THOS	1	3	THOS	GAS-00095688
THOZ	2	3	THOS	GAS-00095688
THOST	*	3	THOS	GAS-00095688
WHO HAVE	*	1	HOF	GAS-00100604
HOF	1	2	HOF	GAS-00105604
HOV	*	2	HOF	GAS-00105604
PAID	*	1	PAEUD	GAS-00115668
THE	1	2	T	GAS-00120291
IT	*	2	T	GAS-00120291
T	1	4	T	GAS-00125291
TH	2	4	T	GAS-00125291
ITY	3	4	T	GAS-00125291
ATE	*	4	T	GAS-00125291
PRIS	1	3	PREUS	GAS-00135688
PRIZ	2	3	PREUS	GAS-00135688
PRIST	*	3	PREUS	GAS-00135688

Fig. 11—Excerpt from complete list of English correspondents.

grams is used twice with different inputs and different dictionary tapes. Similarly, the broken line between the assembly program and the last box on the chart indicates the path for the second output of the assembly program.

Future Research

A number of methods for the improvement of the English output have already been proposed. These include, in particular, the use of digram and trigram frequency counts of both stenotyping and English to correct the spelling of the output, the inclusion of grammatical information in the dictionaries to remove syntactic ambiguities, and a frequency analysis of stenotype texts to be used in analyzing semantic content.

As a first step, however, the outputs produced by the present program can be used directly to update the dictionaries by reducing the number of possible English correspondents for each stenotype word. A large amount of the ambiguity now existing because of the multiplicity of correspondents might thus be

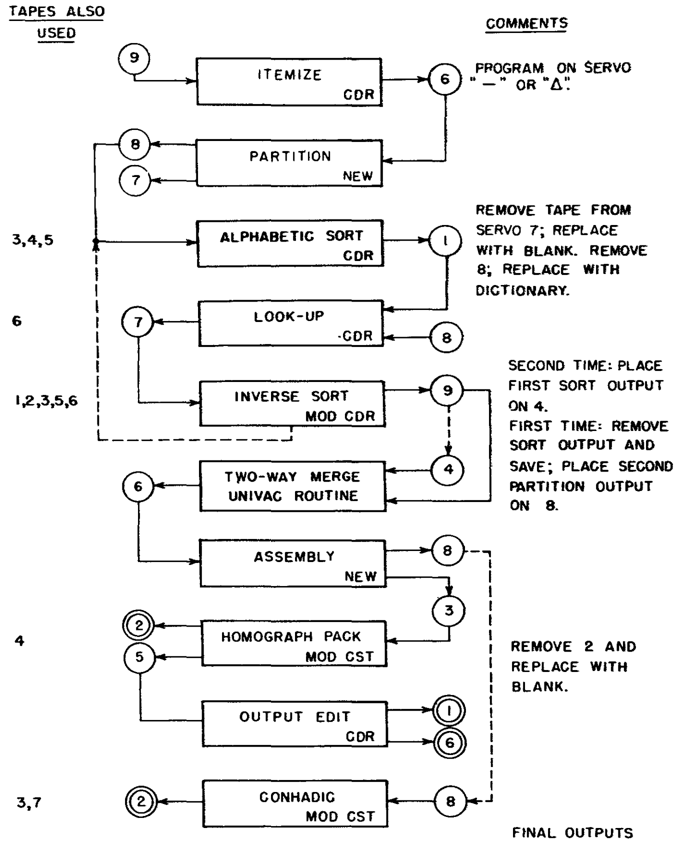

TAPES ALSO USED
COMMENTS
ITEMIZE CDR
PROGRAM ON SERVO "—" OR "Δ".
PARTITION NEW
3,4,5
ALPHABETIC SORT CDR
REMOVE TAPE FROM SERVO 7; REPLACE WITH BLANK. REMOVE 8; REPLACE WITH DICTIONARY.
6
LOOK-UP CDR
1,2,3,5,6
INVERSE SORT MOD CDR
SECOND TIME: PLACE FIRST SORT OUTPUT ON 4.
FIRST TIME: REMOVE SORT OUTPUT AND SAVE; PLACE SECOND PARTITION OUTPUT ON 8.
TWO-WAY MERGE UNIVAC ROUTINE
ASSEMBLY NEW
4
HOMOGRAPH PACK MOD CST
REMOVE 2 AND REPLACE WITH BLANK.
OUTPUT EDIT CDR
3,7
CONHADIC MOD CST
FINAL OUTPUTS
9
6
8
7
1
7
8
9
4
6
8
3
2
5
1
6
2
8

Fig. 12—Continuous computer run for the production of pseudo-English.

eliminated relatively easily. The more difficult problems of determining word boundaries, and analyzing syntactic and semantic content of the context of a given word can then be tackled at a later time.

A human post-editor first takes the English translation as given on one of the two prints furnished by the output edit routine.[20,21] The post-edited output can then be used for two purposes: to determine which of the several English correspondents listed in the dictionary for each stenotype word were actually used in obtaining the correct translation, and to determine which of the assembled items were chosen as correct correspondents.

The first type of information can be used to weed out from the dictionaries (especially the dictionary of letter clusters) those correspondents which are not believed to contribute to a correct English correspondent. Many stenotype strokes will be found to have invariably the same English correspondent, and no purpose is served by burdening the dictionaries with unnecessary information.

The second type of information can be used to refine the partitioning and assembly rules. It is believed, for example, that the I-V-E-A item, when produced, might replace the I-V-F item. At the present time both items are obtained. If this should be confirmed by the post-editing operation a large number of correspondents would be eliminated.

An automatic system for feeding back information obtained from the post-editor might be used to improve the translated output and also to correct the dictionary by deleting or adding information.[22]

References

[1] K. H., Davis, R. Biddulph, *and* S. Balashek, "Automatic Recognition of Spoken Digits," *Journal of the Acoustical Society of America*, Vol. 24 (1952), pp. 637–42.

[2] J. W. Forgie and G. W. Hughes, "A Real-Time Speech Input System for a Digital Computer," *Journal of the Acoustical Society of America*, Vol. 30 (1958), p. 668.

[3] D. B. Fry and P. Denes, "On Presenting the Output of a Mechanical Speech Recognizer," *Journal of the Acoustical Society of America*, Vol. 29 (1957) ,pp. 364–67.

[4] G. W. Hughes and M. Halle, "On the Recognition of Speech by Machine," Proceedings of the *International Conference on Information Processing*, Paris (June 1959), to be published.

[5] H. F. Olson and H. Belar, "A Phonetic Typewriter," *Journal of the Acoustical Society of America*, Vol. 28 (1956), pp. 1072–81.

[6] J. Wiren and H. L. Stubbs, "Electronic Binary Selection System for Phoneme Classification," *Journal of the Acoustical Society of America*, Vol. 28 (1956), pp. 1082–91.

[7] W. S. Ireland, "Stenotype Reporter," 3d ed., The Stenotype Company, Indianapolis (1914).

[8] La Salle Extension University, "The Theory of Stenotypy," Chicago (1949).

[9] Stenographic Machines Inc., "Keyboard and Theory for Machine Shorthand," Chicago (1957).

[10] B. H. Horne, "Stenotype-Stenograph Reporting — A Complete Course," rev. ed., New York (1952).

[11] Standard Theory Committee, "Recommended Standard Practices, Abbreviations and Phrases in Machine Shorthand," *Associated Stenotypists of America*, 3d ed., (1957).

[12] H. A. Gleason, *An Introduction to Descriptive Linguistics*, Henry Holt and Co. (1955).

[13] P. B. Baxendale, "Machine-made Index for Technical Literature — An Experiment," *IBM Journal of Research and Development*, Vol. 2, No. 4 (October 1958).

[14] H. P. Luhn, "The Automatic Creation of Literature Abstracts," *IBM Journal of Research and Development*, Vol. 2, No. 2 (April 1958).

[15] W. J. Plath, "The Relative Frequency of English Consonantal Patterns," A. B. Honors Thesis, Harvard University, (March 1957).

[16] P. E. Jones, Jr., "The Continuous Dictionary Run," *Mathematical Linguistics and Automatic Translation*, Report NSF-2, Harvard Computation Laboratory (March 1959).

[17] Richard Petonke, "2- 3- 4-Way General Merge," Univac System Routines, Vol. 4, Remington Rand Univac.

[18] William Bossert, "The Problem of Homographs in the Automatic Dictionary," Papers presented at the Seminar in Mathematical Linguistics, Vol. 4, Harvard University (Spring 1958).

[19] Orrin Frink, "Programs for Correcting the Harvard Automatic Dictionary and for Syntactic Study, (Conhadic, Checkhadic, Texthadic, and Freqhadic)," *Mathematical Linguistics and Automatic Translation*, Report NSF-2, Harvard Computation Laboratory (March 1959).

[20] V. E. Giuliano, "A Formula Finder for the Automatic Synthesis of Translation Algorithms," *Mathematical Linguistics and Automatic Translation*, Report No. NSF-2, Harvard Computation Laboratory (March 1959).

[21] I. G. Mattingly, "Logical Post-editing," *Mathematical Linguistics and Automatic Translation*, Report No. NSF-3, Harvard Computation Laboratory (August 1959).

[22] R. E. Jones, Jr., "A Feedback System for the Harvard Automatic Dictionary," *Mathematical Linguistics and Automatic Translation*, Report No. NSF-3, Harvard Computation Laboratory (August 1959).

Discussion

Mr. Galli (IBM): Has any work been done on the resolution of ambiguities arising from the lack of an "end of word" indication?

Dr. Salton: We have not done any such work. However, there exist machine shorthand systems, in particular a system known as Brevitype, where a slightly different keyboard is used, and an indication appears on the tape when strokes are connected together. If a system such as Brevitype is used, the problem of recognizing word boundaries would be eliminated. In stentoyping this is not the case, and I suggested on the last slide that we might be able to make frequency analyses of stenotype clusters; I intended to mean that we would go across stroke boundaries and thus pick up certain combinations of letters which might be written in separate strokes on the stenotype machine, but would belong to the same English word. We hope to do some work along this line.

E. H. Cabaniss (GE): Why not replace the paper tape with magnetic or punched paper tape?

Dr. Salton: I imagine that the gentleman wants to know whether we could replace the ordinary paper tape produced by the stenotype machine by punched paper tape. This can indeed be done. I have suggested in the written version of this paper two possibilities for solving the input problem. One of them requires no alteration at all in the stenotype machine and uses a character recognition scheme. The character recognition problem is very simple in this case because we have generally only one letter for each position across the tape, so that when a character is detected in a given position, it is generally a unique character. The other possibility would require certain modifications of the machine; in particular information could be punched between adjacent lines of printed information. In general, I would like to avoid changes in the present equipment.

Mr. Dittberner (IBM): You mention that vocal speech recognition does not appear to be feasible in the foreseeable future. Is this foreseeable future compatible with the rosy future promised in the kilomegacycle era? Do you really feel so pessimistic about even recognizing one individual's voice — such as the operations executive?

Dr. Salton: I can answer the second question. I think that the word boundary problems that arise are so severe that we cannot solve this very soon. If we could train a speaker to speak very slowly in disconnected pieces, in such a way that individual syllables are detected, the problem would be much easier; in a practical case, speech appears as one continuous flow, and the problem is very difficult. There are authorities in the field of speech analysis who may be more qualified to answer.

H. Freeman (Sperry Gyroscope): Your presentation was concerned primarily with ordinary English. What problems arise when very specialized technical terms are involved, such as stenotype records of scientific papers?

Dr. Salton: For special copy, reporters use special abbreviations. These special abbreviations must be included in the dictionary.

D. E. Bachman (IBM): What, if any, shorthand abbreviations do you use for plural forms of words and the past tense of words?

Dr. Salton: The past tense of verbs is written by adding a "D" in a second stroke. For plurals, we don't use any abbreviation, only an "S" added at the end.

D. A. Bourne (IBM): How difficult is it to learn the use of the machine as compared to Pitman shorthand, for example?

Dr. Salton: It is very simple. I got into this subject without knowing the stenotype system, and I learned all the letter combinations in practically no time. However, the difficult thing is to attain speed. A great deal of training is required to attain speed, and some reporters will undergo training for several years. Some reporters attain speeds of some 200 words or 220 words a minute. This takes a long time certainly.

K. Enslein (*Brooks Research*): Could one attach a tag to the stenotype word indicating its grammatical character?

Dr. Salton: We can do this in the dictionary, but I can't see how you can ask the reporter to add a grammatical indication on the tape directly. However, we could ask a reporter to indicate when strokes are connected together to form a single word. This would require one additional key on the keyboard and would slow down the reporter somewhat, but still appears feasible.

S. Pollack (*Rand*): What is the purpose of the serial number?

Dr. Salton: Before the strokes are looked up in the dictionary, they are sorted first into dictionary, that is alphabetical, order. Later on, we need to sort back into text order, using the serial number as a key. The serial numbers are assigned during itemization using Serial Number 1 for the first stroke, then 2, 3, 4, 5 to subsequent strokes, and so on.

I. Rotkin (*Diamond Ordnance Labs.*): In order to use syntax, you must be able to recognize the end of a sentence. How do you do this?

Dr. Salton: Punctuation marks are provided on the keyboard. FPLT, for example, represents a period; therefore, there is no difficulty.

E. H. Goldman: How many magnetic tape passes are required in the processing before output is ready for human editing?

Dr. Salton: There are quite a few passes involved in the dictionary look-up procedure. Logically there are only the five passes shown on the slide. However, because of machine limitations a lot of juggling may be needed.

L. B. Harris (*GE*): Why are we using a stenotypist rather than a tape recorder to record this meeting?

Dr. Salton: The fact is that the stenotypist has a great facility in inserting things, deleting things, in dealing with a situation where the speaker stops, or where the speaker makes an error, and so on. For this reason, the stenotype transcript is frequently more useful than a tape recorded transcript. When you report a conference, the participants frequently are particular about what appears in the proceedings. The stenotypist can handle this. Therefore, a stenotypist is often much more useful than a tape recorder.

Critical-Path Planning and Scheduling

JAMES E. KELLEY, JR.† AND MORGAN R. WALKER†

Introduction and Summary

AMONG the major problems facing technical management today are those involving the coordination of many diverse activities toward a common goal. In a large engineering project, for example, almost all the engineering and craft skills are involved as well as the functions represented by research, development, design, procurement, construction, vendors, fabricators and the customer. Management must devise plans which will tell with as much accuracy as possible how the efforts of the people representing these functions should be directed toward the project's completion. In order to devise such plans and implement them, management must be able to collect pertinent information to accomplish the following tasks:

(1) To form a basis for prediction and planning
(2) To evaluate alternative plans for accomplishing the objective
(3) To check progress against current plans and objectives, and
(4) To form a basis for obtaining the facts so that decisions can be made and the job can be done.

Many present project planning systems possess deficiencies resulting from techniques inadequate for dealing with complex projects. Generally, the several groups concerned with the work do their own detailed planning and scheduling — largely independent from one another. These separate efforts lead to lack of coordination. Further, it is traditional in project work that detailed schedules be developed from gross estimates of total requirements and achievements based on past experience. The main reason for this oversimplification stems from the inability of unaided human beings to cope with sheer complexity. In consequence, many undesirable effects may arise. Some important aspects of a project, which should be taken into account at the outset, may be ignored or unrecognized. As a result, much confusion may arise during the course of the project. When this happens, the management of the project is left to the coordinators and expediters. In such circumstances, management loses much of the control of a project and is never quite sure whether its objectives are being attained properly.

Reconizing the deficiencies in traditional project planning and scheduling procedures, the Integrated Engineering Control Group (I. E. C.) of E. I. duPont de Nemours & Co. proceeded to explore possible alternatives. It was felt that a high degree of coordination could be obtained if the planning and scheduling information of all project functions are combined into a single master plan — a plan that integrates all efforts toward a common objective. The plan should point directly to the difficult and significant activities — the problems of achieving the objective. For example, the plan should form the basis of a system for *management by exception.* That is, within the framework of the rules laid down, it should indicate the *exceptions*. Under such a system, management need act only when deviations from the plan occur.

The generation of such a coordinated master plan requires the consideration of much more detailed information at one time than heretofore contemplated in project work. In turn, a new approach to the whole problem of planning and scheduling large projects is required. In late 1956, I. E. C. initiated a survey of the prospects for applying electronic computers as an aid to coping with the complexities of managing engineering projects. The following were the questions of most pressing interest: To what extent can a computer-oriented system be used:

(1) To prepare a master schedule for a project?
(2) To revise schedules to meet changing conditions in the "most" economical way?
(3) To keep management and the operating departments advised of project progress and changes?

During the course of this survey outside help was solicited. As part of their customer service, Remington Rand UNIVAC assigned the first author to the job of providing some assistance. At the time the second author represented duPont in this effort. The result of our alliance is the subject of this essay.

We made a critical analysis of the traditional approach to planning and a study of the nature of engineering projects. It quickly became apparent that if a new approach were to be successful, some technique had to be used to describe the interrelationships among the many tasks that compose a project. Further, the technique would have to be very simple and rigorous in application, if humans were to cope with the complexity of a project.

One of the difficulties in the traditional approach is that planning and scheduling are carried on simultaneously. At one session, the planner and scheduler

† Mauchly Associates, Inc., Ambler, Pa.

consider — or attempt to consider — hundreds of details of technology, sequence, duration times, calendar deliveries and completions, and cost. With the planning and scheduling functions broken down in a step by step manner, fruitless mental juggling might be avoided and full advantage taken of the available information.

Accordingly, the first step in building a model of a project planning and scheduling system was to separate the functions of planning from scheduling. We defined planning as the act of stating what activities must occur in a project and in what order these activities must take place. Only technology and sequence were considered. Scheduling followed planning and is defined as the act of producing project timetables in consideration of the plan and costs.

The next step was to formulate an abstract model of an engineering project. The basic elements of a project are activities or jobs: determination of specs, blueprint preparation, pouring foundations, erecting steel, etc. These activities are represented graphically in the form of an arrow diagram which permits the user to study the technological relations among them.

Cost and execution times are associated with each activity in the project. These factors are combined with the technological relations to produce optimal direct cost schedules possessing varying completion dates. As a result, management comes into possession of a spectrum of possible schedules, each having an engineered sequence, a known elapsed time span, a known cost function, and a calendar fit. In the case of R & D projects, one obtains "most probable" schedules. From these schedules, management may select a schedule which maximizes return on investment or some other objective criterion.

The technique that has been developed for doing this planning and scheduling is called the Critical-Path Method. This name was selected because of the central position that critical activities in a project play in the method. The Critical-Path Method is a general interest from several aspects:

(1) It may be used to solve a class of "practical" business problems

(2) It requires the use of modern mathematics

(3) Large-scale computing equipment is required for its full implementation

(4) It has been programmed for three computers — UNIVAC I, 1103A and 1105 with a Census Bureau configuration

(5) It has been put into practice

In what follows we will attempt to amplify these points. We will describe various aspects of the mathematical model first. The mathematics involved will be treated rather superficially, a detailed development being reserved for a separate paper. The second part of this essay will cover the experience and results obtained from the use of the Critical-Path Method.

Part I: Analysis of a Project

1. *Project Structure*

Fundamental to the Critical-Path Method is the basic representation of a project. It is characteristic of all projects that all work must be performed in some well-defined order. For example, in construction work, forms must be built before concrete can be poured; in R & D work and product planning, specs must be determined before drawings can be made; in advertising, artwork must be made before layouts can be done, etc.

These relations of order can be shown graphically. Each job in the project is represented by an arrow which depicts (1) the existence of the job, and (2) the direction of time-flows from the tail to the head of the arrow). The arrows then are interconnected to show graphically the sequence in which the jobs in the project must be performed. The result is a topological representation of a project. Fig. 1 typifies the graphical form of a project.

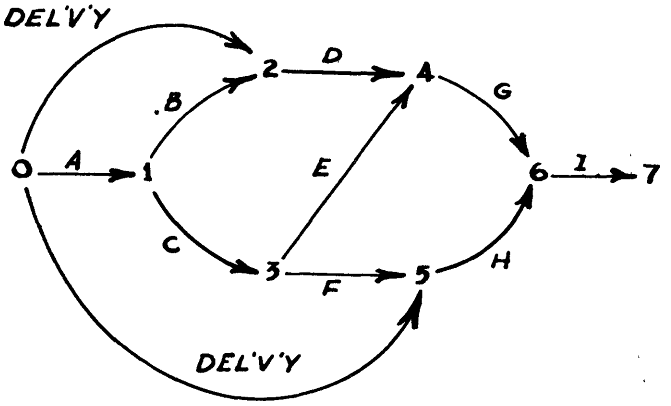


Fig. 1—Typical project diagram.

Several things should be noted. It is tacitly assumed that each job in a project is defined so that it is fully completed before any of its successors can begin. This is always possible to do. The junctions where arrows meet are called *events*. These are points in time when certain jobs are completed and others must begin. In particular there are two distinguished events, origin and terminus, respectively, with the property that origin precedes and terminus follows every event in the project.

Associated with each event, as a label, is a nonnegative integer. It is always possible to label events such that the event at the head of an arrow always has a larger label than the event at the tail. We assume that events are always labeled in this fashion. For a project, P, of $n + 1$ events, origin is given the label 0 and terminus is given the label n.

The event labels are used to designate jobs as follows: if an arrow connects event i to event j, then the associated job is called job (i, j).

During the course of constructing a project diagram, it is necessary to take into account a number of things pertaining to the definition of each job. Depending upon such factors as the purpose for making the project analysis, the nature of the project, and how much information is available, any given job may be defined in precise or very broad terms. Thus, a job may consist of simply typing a report, or it might encompass all the development work leading up to the report plus the typing. Someone concerned with planning the development work should be interested in including the typing as a job in the project while those concerned with integrating many small development projects would probably consider each such project as an individual job.

Further, in order to prepare for the scheduling aspects of project work, it is necessary to consider the environment of each job. For example, on the surface it may be entirely feasible to put 10 men on a certain job. However, there may only be enough working space for five men at a time. This condition must be included in the job's definition. Again, it may technically be possible to perform two jobs concurrently. However, one job may place a safety hazard on the other. In consequence, the first job must be forced to follow the second.

Finally, the initiation of some jobs may depend on the delivery of certain items — materials, plans, authorization of funds, etc. Delivery restraints are considered jobs, and they must be included in the project diagram. A similar situation occurs when certain jobs must be completed by a certain time. Completion conditions on certain jobs also may be handled, but in a more complicated fashion, by introducing arrows in the project diagram.

Project diagrams of large projects, although quite complicated, can be constructed in a rather simple fashion. A diagram is built up by sections. Within each section the task is accomplished one arrow at a time by asking and answering the following questions for each job:

(1) What immediately precedes this job?

(2) What immediately follows this job?

(3) What can be concurrent with this job?

By continually back-checking, the chance of making omissions is small. The individual sections then are connected to form the complete project diagram. In this way, projects involving up to 1600 jobs have been handled with relative ease.

From a scientific viewpoint, the idea of diagramming the technological relations among the jobs in a project is almost trivial. Such diagrams are used in many engineering and mathematical applications. However, diagramming is an innovation in project work which has given planners several benefits:

(1) It provides a disciplined basis for planning a project.

(2) It provides a clear picture of the scope of a project that can be easily read and understood.

(3) It provides a vehicle for evaluating alternative strategies and objectives.

(4) It tends to prevent the omission of jobs that naturally belong to the project.

(5) In showing the interconnections among the jobs it pinpoints the responsibilities of the various operating departments involved.

(6) It is an aid to refining the design of a project.

(7) It is an excellent vehicle for training project personnel.

2. Calendar Limits on Activities

Having a diagram of a project is only the first step in analyzing a project. Now the plan must be put on a timetable to obtain a schedule.

In order to schedule a project, it is necessary to assign elapsed time durations to each job. Depending on the nature of the project this data may be known deterministically or non-deterministically. Another way to say this is that the duration of each job is a random variable taken from an approximately known distribution. The duration of a job is deterministic when the variance of the distribution is small. Otherwise it is non-deterministic.

The Deterministic Case

On the basis of estimated elapsed times, we may compute approximations to the earliest and latest start and completion times for each job in a project. This information is important not only for putting a schedule on the calendar, but also for establishing rigorous limits to guide operating personnel. In effect, it tells those responsible for a job when to start worrying about a slippage and to report this fact to those responsible for the progress of the project. In turn, when this information is combined with a knowledge of the project's topological structure, higher management can determine when and how to revise the schedule and who will be affected by the change. This kind of information is not determined accurately by traditional methods. What this information provides is the basis for a system of management by exception.

Let us assume that the project, P, of $n + 1$ events, starts at relative time 0. Relative to this starting time each event in the project has an earliest time occurance. Denote the earliest time for event i by $t_i^{(0)}$ and the duration of job (i,j) by y_{ij}. We may then compute the values of $t_i^{(0)}$ inductively as follows:

$$(1)\quad \begin{cases} t_0^{(0)} = 0 \\ t_j^{(0)} = \max\,[y_{ij} + t_i^{(0)} \mid i < j,\ (i,j) \in P]\,,\ 1 \le j \le n. \end{cases}$$

Similarly, we may compute the latest time at which each event in the project may occur relative to a fixed project completion time. Denote the latest time for event i by $t_i^{(1)}$. If λ is the project completion time (where $\lambda \geqq t_n^{(0)}$) we obtain

$$(2)\quad \begin{cases} t_n^{(1)} = \lambda \\ t_i^{(1)} = \min\,[t_j^{(1)} - y_{ij} \mid i < j, (i,j) \in P], 0 \le i \le n-1. \end{cases}$$

Having the earliest and latest event times we may compute the following important quantities for each job, (i, j), in the project:

Earliest start time	$= t_i^{(0)}$
Earliest completion time	$= t_i^{(0)} - y_{ij}$
Latest start time	$= t_j^{(1)} - y_{ij}$
Latest completion time	$= t_j^{(0)}$
Maximum time available	$= t_j^{(1)} - t_i^{(0)}$

If the maximum time available for a job equals its duration the job is called *critical.* A delay in a critical job will cause a comparable delay in the project completion time. A project will contain critical jobs only when $\lambda = t_n^{(0)}$. If a project does contain critical jobs, then it also contains at least one contiguous path of critical jobs through the project diagram from origin to terminus. Such a path is called a *critical-path.*

If the maximum time available for a job exceeds its duration, the job is called a *floater.* Some floaters can be displaced in time or delayed to a certain extent without interfering with other jobs or the completion of the project. Others, if displaced, will start a chain reaction of displacements downstream in the project.

It is desirable to know, in advance, the character of any floater. There are several measures of float of interest in this connection.. The following measures are easily interpreted:

Total Float	$= t_j^{(1)} - t_i^{(0)} - y_{ij}$
Free Float	$= t_j^{(0)} - t_i^{(0)} - y_{ij}$
Independent Float	$= \max\,(0, t_j^{(0)} - t_i^{(1)} - y_{ij})$
Interfering Float	$= t_j^{(1)} - t_j^{(0)}.$

Non-Deterministic Schedules

Information analogous to that obtained in the deterministic case is certainly desirable for the non-deterministic case. It would be useful for scheduling applied research directed toward a well-defined objective.

However, in attempting to develop such information some difficulties are encountered which do not seem easily resolved. These difficulties are partly philosophical and partly mathematical. Involved is the problem of defining a "meaningful" measure for the criticalness of a job that can be computed in a "reasonable" fashion.

Although a complete analysis of this situation is not germane to the development of the Critical-Path Method, it is appropriate, however, to indicate some concepts basic to such an analysis. Thus, in the non-deterministic case we assume that the duration, y_{ij}, of activity (i, j) is a random variable with probability density $G_{ij}(y)$. As a consequence it is clear that the time at which an event occurs is also a random variable, t_j, with probability density $H_j(t)$. We assume that event 0 is certain to occur at time 0. Further on the assumption that it is started as soon as possible, we see that $t_i + y_{ij} = x_{ij}$, the completion time for job (i, j), is a random variable with probability density $S_{ij}(x)$:[1]

$$(3)\quad S_{ij}(x) = \begin{cases} G_{ij}(x), \text{ if } i = 0 \\ \displaystyle\int_{-\infty}^{\infty} H_i(u) G_{ij}(x - u)\, du,\ (i, j) \in P. \end{cases}$$

Assuming now that an event occurs at the time of the completion of the last activity preceding it we can easily compute the probability density, $H_i(t)$, of

$$t_i = \max\,[x_{ij} \mid (i, j) \in P,\ i < j]\,,$$

where x_{ij} is taken from $S_{ij}(x)$:

$$(4)\quad H_j(t) = \sum_{(i,j)\in P} S_{ij}(t) \prod_{\substack{(k,j)\in P \\ k \ne i}} \int_{-\infty}^{t} S_{kj}(u)du,\ \ 1 \le j \le n.$$

Several methods are available for approximating $S_{ij}(x)$ and $H_j(t)$. The one which suits our taste is to express $G_{ij}(y)$ in the form of a histogram with equal class intervals. The functions $S_{ij}(x)$ and $H_j(t)$ are then histograms also and are computed in the obvious way by replacing integrals by sums. It would seem that in practice one can afford to have fairly large class intervals so that the chore of computing is quite reasonable.

In computing $S_{ij}(x)$ and $H_i(t)$ above we assumed that job (i,j) was started at the time of the occurrance of t_i. For various reasons it may not be desirable to abide by this assumption. Indeed, it may be possible to delay the start of job (i, j) to a fair extent after the actual occurance of t_i without changing the character of $H_j(t)$. However, the assumption we have made does provide a probabilistic lower bound on the start time for job (i, j). By analogy with the deterministic case we may think of $H_i(t)$ as the probability density of the earliest start time for job (i, j). Similarly, $S_{ij}(x)$ in (3) then becomes the probability density of the earliest completion time for job (i, j). In this sense, (4) is the probabilistic analogue of (1).

It is desirable to be able to measure the criticalness of each job in the project. Intuitively one is tempted to use the probabilistic analogue of (2), running the project backward from some fixed or random comple-

[1]See M. G. Kendall, "The Advanced Theory of Statistics," Vol. 1, J. B. Lippincott Co., 1943, p. 247.

tion time as was done in the deterministic case. In this way one might hope to obtain information about the latest time at which events can occur, so that probabilistic measures of float might be obtained. It appears that this is a false hope since, among other things, such a procedure assumes that the project start time is a random variable and not a certain event. (The project start time can always be assumed certain, simply by making lead time for the project start on the day the calculations are made.)

To proceed further we must introduce the notion of "risk" in defining the criticalness of a job. On the basis of this definition one would hope to obtain probabilistic measures for float which would be useful for setting up a system for management by exception. We will not explore these possibilities further here.

3. The project cost function

In the deterministic case, the durations of jobs may sometimes be allowed to vary within certain limits. This variation may be attributed to a number of factors. The elapsed-time duration of a job may change as the number of men put on it changes, as the type of equipment or method used changes, as the work week changes from 5 to 6 to 7 days, etc. Thus, management has considerable freedom to choose the elapsed-time duration of a job, within certain limitations on available resources and the technology and environment of the job. Every set of job durations selected will lead to a different schedule and, in consequence, a different project duration. Conversely, there are generally many ways to select job durations so that the resulting schedules have the same shortest time duration.

Faced with making a choice, management must have some way of evaluating the merits of each possibility. In traditional planning and scheduling systems such a criterion is not too well defined. In the present context, however, there are several possibilities. The one we will focus our attention upon is *cost.*

Job Cost

When the cost (labor, equipment and materials) of a typical engineering job varies with elapsed-time duration it usually approximates the form of the curve of Fig. 2. This is what is usually called "direct" cost. Costs arising from administration, overhead, and distributives are not included.

Note that when the duration of job (i, j) equals D_{ij}, the cost is a minimum. On the surface, this is a desirable point at which to operate. Certainly management would seldom ever elect to require the job to take longer than the optimal method time. We call D_{ij} the *normal* duration for job (i, j). However, exogenous conditions may require that a job be expedited. This may be done in a variety of ways. But in any case there is a limit to how fast a job may be performed. This lower bound is denoted by d_{ij} in Fig. 2 and is called the *crash* duration for job (i, j).

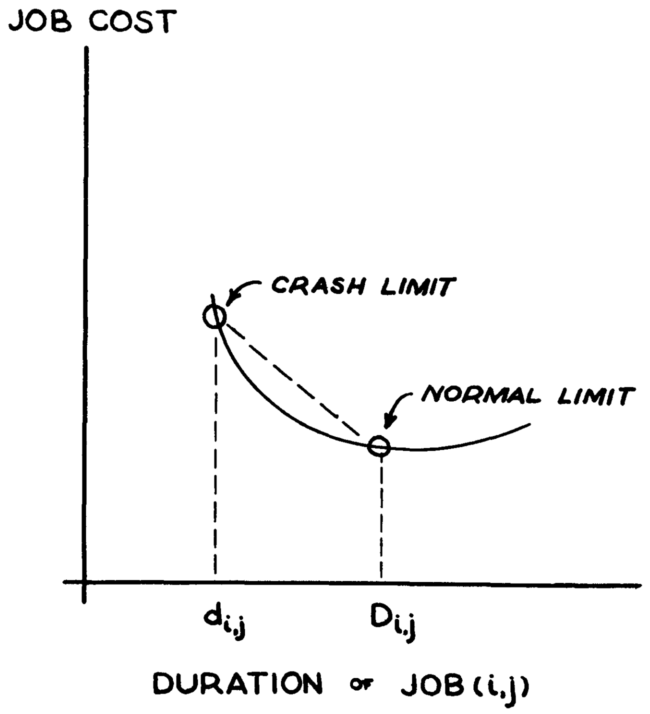


Fig. 2—Typical job cost curve.

It is thus reasonable to assume that the duration y_{ij} of job (i, j) satisfies

$$0 \leq d_{ij} \leq y_{ij} \leq D_{ij}. \tag{5}$$

The cost of job (i, j) is now approximated in a special way over the range defined by inequalities (5). The type of approximation used is dictated by the mathematical technique involved in what follows. Thus, we must assume that the approximate cost function is a piecewise linear, non-increasing and convex function of y_{ij}. Usually in practice insufficient data is available to make more than a linear approximation. There are exceptions, of course.

In the linear case we may write

$$\text{Cost of Job } (i, j) = a_{ij}y_{ij} + b_{ij}, \tag{6}$$

where $a_{ij} \leq 0$ and $b_{ij} \geq 0$. This is indicated by the dotted line in Fig. 2.

Minimum Project Costs

On the basis of job cost functions just developed we can determine the (direct) cost of any particular schedule satisfying inequalities (5) by simply summing the individual job costs. That is,

$$\text{Project (Direct) Cost} = \sum_{(i,j\epsilon P)} (a_{ij}y_{ij} + b_{ij}) \tag{7}$$

It is clear that there are generally many ways that job durations may be selected so that the earliest completion times of the resulting schedules are all equal. However, each schedule will yield a different value of (7), the project cost. Assuming that all conditions of the project are satisfied by these schedules, the one which costs the least invariably would be selected for implementation.

It is therefore desirable to have a means of selecting the least costly schedule for any given feasible earliest project completion time. Within the framework we have already constructed, such "optimal" schedules are obtained by solving the following linear program: *Minimize* (7) *subject to* (5) *and*

(8) $$y_{ij} \leq t_j - t_i, \quad (i, j) \epsilon P ,$$

and

(9) $$t_0 = 0 , \quad t_n = \lambda .$$

Inequalities (8) express the fact that the duration of a job cannot exceed the time available for performing it. Equations (9) require the project to start at relative time 0 and be completed by relative time λ. Because of the form of the individual job cost functions, within the limits of most interest, λ is also the earliest project completion time.

At this point it should be noted that the case where each job cost function is non-increasing, piecewise linear and convex is also reducible to a parametric linear program (see [7] and [8]). It does not add anything essential here to consider this more generalized form.

A convenient tool for generating schedules for various values of λ is the method of parametric linear programming with λ as the parameter. Intuitively, this technique works as follows. Initially, we let $y_{ij} = D_{ij}$ for every job in the project. This is called the all-normal solution. We then assume that each job is started as early as possible. As a result we can compute $t_i^{(0)}$ for all events. In particular, the earliest project completion time for this schedule is $\lambda = t_n^{(0)}$. By the nature of the job cost functions this schedule is also a minimum cost schedule for $\lambda = t_n^{(0)}$. We now force a reduction in the project completion time by expediting certain of the critical jobs — those jobs that control project completion time. Not all critical jobs are expedited, but only those that drive the project cost up at a minimum rate as the project completion time decreases. As the project completion is reduced, more and more jobs become critical and thus there is a change in which jobs are to be expedited. This process is repeated until no further reduction in project completion time is possible.

Mathematically speaking, the process utilizes a primal-dual algorithm (see [6]). The restricted dual problem is a network flow problem involving both positive upper and lower bound capacity restrictions. A form of the Ford-Fulkerson network flow algorithm [3] is used to solve it. The critical jobs that are expedited at each stage of the process correspond to a cut set in the graph of all critical jobs.

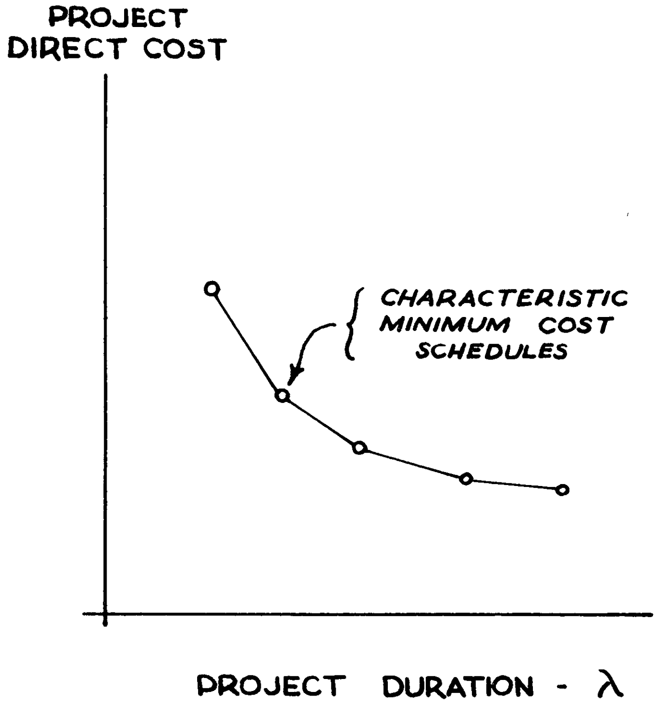


Fig. 3—Typical project cost curve.

This process produces a spectrum of schedules (characteristic solutions in the linear programming sense) each at minimum total (direct) cost for its particular duration. When the costs of these schedules are plotted versus their respective durations, we obtain a non-increasing, piecewise linear, convex function as depicted in Fig. 3. This function is called the project cost curve.

Uses of the Project Cost Curve

The project cost curve only reflects the direct costs (manpower, equipment and materials) involved in executing a project. However, other costs are involved which contribute to the total project cost, such as overhead and administrative costs and perhaps even penalties for not completing a project or some portion of it by a certain time. These external costs must be taken into account when management plans how the project should be implemented relative to overall objectives.

Relative to these external costs there are at least two types of considerations that management may make:

(1) The (direct) cost curve for the project may be compared with the indirect cost of overhead and administration to find a schedule which *minimizes* the *investment cost.*

(2) The investment cost curve may be compared with market losses, as when it is desired to meet the demands of a rising market in a competitive situation. The schedule selected in this case is one which *maximizes return on investment.*

4. Manpower Leveling

As developed in this paper, the Critical-Path Method is based primarily on the technological requirements of a project. Considerations of available manpower and equipment are conspicuous by their absence. All schedules computed by the technique are technologically feasible but not necessarily practical. For example, the equipment and manpower requirements for a particular schedule may exceed those available or may fluctuate violently with time. A means of handling these difficulties must therefore be sought — a method which "levels" these requirements.

Here we will outline the approach we have taken to this problem. We restrict the discussion to manpower, similar considerations being applicable to leveling equipment requirements.

The term "manpower leveling" does not necessarily mean that the same number of men should be used throughout the project. It usually means that no more men than are available should be used. Further, if this requirement is met, one should not use the maximum number of men available at one instant in time and very few the very next instant of time.

The difficult part of treating the manpower leveling problem from a mathematical point of view is the lack of any explicit criteria with which the "best" use of manpower can be obtained. Under critical examination, available levels of manpower and also changes in level are established arbitrarily. This situation exists to some degree regardless of the organization involved. Even in the construction industry, where the work is by nature temporary, the construction organization desires the reputation of being a consistent "project life" employer. The organization wants the employee to feel that once "hired on" he can be reasonably sure of several months' work at the very least. In plants and in technical and professional engineering fields the same situation exists but with more severity. The employee is more acutely aware of "security", and the employer much more keenly aware of the tangible costs of recruitment and layoff as well as the intangible costs of layoff to his overall reputation and well-being.

In most organizations idle crafts and engineers or the need for new hires are treated with overwhelming management scrutiny. This is an excellent attitude, but too often this consideration is short range and does not consider long range requirements.

The following approaches to this problem have been made:

Incorporating Manpower Sequences

It is possible to incorporate manpower availability in the project diagram. However, this approach can cause considerable difficulty in stating the diagram and may lead to erroneous results. Therefore, we recommend that this approach be dropped from consideration.

For example, assume there are three jobs — A, B, and C — that, from a technological viewpoint, can occur concurrently. However, each job requires the same crew. We might avoid the possibility that they occur simultaneously by requiring that A be followed by B, followed by C. It is also possible to state five other combinations — ACB, BCA, BAC, CAB, and CBA.

If we assume that this example occurs many times in a large arrow diagram, then there is not one, but a very large number of possible diagrams that can be drawn.

Now suppose a manpower sequence was not incorporated in the diagram and schedules were computed. It could be that the float times available for jobs A, B, and C are sufficient to perform the jobs in any of the six possible time sequences. However, by incorporating manpower sequences, we would never really know the true scheduling possibilities.

Examining Implied Requirements

Currently this method is performed manually and has been successfully used by applications personnel. It is possible to do much of the work involved by computer but, thus far, computer programs have not been prepared.

In preparing the work sheets for each activity, a statement is made of how many men per unit of time by craft are required for each duration. The planning and scheduling then proceeds in the manner prescribed by the Critical-Path Method. After a schedule is selected from all of the computed schedules, work on manpower leveling starts.

The first task is to tabulate the force required to execute the jobs along the critical path. Manpower commitments must be made to do these jobs at specific calendar dates. If manpower is not available, a longer duration schedule must be selected and the force requirements re-evaulated.

If adequate manpower is available to perform the critical jobs, then the total work force required by time units is tabulated. This is done by assuming every job starts at its earliest start date. The tabulation also is done, except assuming that every job starts at its latest start date.

Two total force curves result. These are then examined to be sure that they conform with some implicit statement of desired force. If not, the floaters are displaced to smooth the force curve. (In practice it has been found that one should displace the jobs with the least float first.)

During the tabulation and leveling processes, subtotals are kept by craft to ensure that, even though total force may be all right, craft restrictions also are met.

The smoothing (a purely heuristic process) is done until the desired force and craft curves are obtained, or until it is discovered that the schedule requires an unavailable force. In this case, the next longer schedule is selected, and the process is repeated until satisfactory results are obtained.

In one actual case, it was determined after attempts at smoothing that 27 mechanics were required when only 8 were available. Smoothing for this condition meant about a 20% lengthening of the critical path. Armed with this information, the planning and scheduling staff placed in management's hands a quantitative measure of the meaning of a manpower shortage so that, in advance, corrective action could be taken.

Solving for Best Fit

A procedure has been developed for computer programming that again is subjective in approach. One does not "solve" for the "best" force on the basis of some objective criteria. Rather, one states in advance what is "best" and then attempts to find the "best" fit.

The procedure is similar to examining the implied force requirements.

The total force curve desired, and craft breakdowns if required, constitute the input. Then a step-by-step procedure is followed to move the floaters so that the resultant force curve approximates the desired force curve. If the results are unsatisfactory, the procedure would be to begin again with a schedule of longer duration.

The detailed method is too long for presentation here. In its present form, it is too involved for manual use except on very small projects. The logical steps are not too difficult, but for even modest-size projects the amount of storage required for "keeping track of" program steps dictates a fairly large computer for economical processing.

5. An Accounting Basis for Project Work

From the very start of the development of the Critical-Path Method, it has been the practice to assign a cost account number or job work order number to every job in a project. With this data, a structure can be set up for accruing costs against the proper accounts as the project proceeds.

Because each job in a project has a cost curve associated with it, as duration times are computed, it is a simple matter to compute the estimated individual job cost for a schedule. This computation gives management and supervision the basis for project cost control. As actual costs are incurred they can be compared with estimated costs and analyzed for exceptions. Time and cost control are in herent in the system.

One of the difficult tasks on certain types of project work is closing the project to capital investment accounts. This frequently is not completed until long after the project ends. There are several reasons for the delay. One is that costs are sometimes not accrued so that they may easily be identified and/or apportioned to the proper facility. Another is the sheer magnitude of the accounting job. Under the Critical-Path system, it is possible to do this job as you go, keeping current with the project. Just as it is easy to close a project, it is easy to estimate in advance capital expenditures for labor, equipment and materials. This can mean many dollars in savings to project management in efficient capital usage.

Part II: Historical Development and Results

1. Early Developments

The fundamentals of the system outlined in Part I were developed during early 1957. Preliminary results were reported in [4] and [5]. By May 1957 the theory had advanced to the point where it was felt that the approach would be successful. At that time a cooperative effort to implement the method was undertaken by Remington Rand and duPont in order to determine the extent to which any further work was advisable. Remington Rand supplied the required programs for duPont's UNIVAC I located in Newark, Delaware. Engineers from duPont provided a small pilot problem with which to make the preliminary tests.

The results of this phase of the development were officially demonstrated in September, 1957. The demonstration showed that the technique held great promise. Accordingly, further tests of the system were authorized. These tests were set up to determine several things, among which were the following major points:

(1) To see if the data required were available and, if not, how difficult they would be to obtain

(2) To see if an impartial group of engineers could be trained to use the new method

(3) To see if the output from the new scheduling system was competitive in accuracy and utility with the traditional method

(4) To determine what kind of computing equipment is required for this type of application

(5) To see if the new system was economical.

2. Selecting a Team

By late December 1957 a team of six engineers was formed, and work on the test was under way. The team consisted of a field superintendent, a division engineer, and two area engineers, all with experience from construction, a process engineer from design, and an estimator. It is important to note that all these men had some experience in each of the other's specialty. For this reason they had very little difficulty in communicating with one another. Further, they averaged from 8 to 10 years' experience in the duPont organization. Knowing the organization helped expedite their work as a team by making it possible to avoid unnecessary red tape in acquiring the necessary data.

The objectives of the team were to collect the data required for the test project and then plan and schedule it, using the then available UNIVAC I system. In order to prepare the way, the team was given a 40-hour workshop course on the Critical-Path Method. This course covered the philosophy of the method, project diagramming, and interpretation of results. Some attempt was made to indicate how the computer determines minimum cost schedules, but purely for the sake of background. None of the mathematics involved was discussed. The team then spent about a week preparing and processing a small artificial project to test how well they absorbed the material of the course. It was subsequently discovered that as little as 12 hours of instruction are sufficient to transmit a working knowledge of project diagramming to operating personnel.

3. The First Live Test

The project selected for the first test was the construction of a new chemical plant facility capitalized at $10,000,000. We will refer to this project as Project A. In order to get the most out of the test, and because the method was essentially untried, it was decided that the team's scheduling would be carried out independently of the normal scheduling group. Further, the team's schedules would not be used in the administration of the project.

The plan of Project A was restricted in scope to include only the construction steps. More specifically, the project was analyzed starting just after Part II authorization — the point at which about 30% of the project design is complete and funds have been authorized to start construction. This approach was reasonable for the first test because the sequence of construction steps was more apparent than those of design and procurement. The latter were to be included in the analysis of some subsequent project.

As the team proceeded to prepare the plan for the project, the following kinds of data were collected and reviewed:

(1) Construction cost estimates
(2) File prints and specifications
(3) Scopes of work and correspondence
(4) Bids and quotations
(5) Material and equipment list and limiting equipment list with estimated deliveries
(6) Design schedule
(7) Craft and average wage rates and unit price data
(8) Details of pending contracts involving field labor
(9) Contemplated design changes with cost and time estimates

The whole project was then divided into major areas. The scope of work in each area was analyzed and broken down into individual work blocks or jobs. These jobs were diagrammed. The various area diagrams were combined to show all the job sequences involved in the project. The jobs varied in size from $50 to $50,000, depending on the available details and the requirements imposed by design and delivery restraints. All told, the project consisted of 393 jobs with an average cost of $4,000; 156 design and delivery restraints; and 297 "dummy" jobs to sequence work properly, identify temporal check points, and help to interpret results.

During the diagramming phase, normal and crash times and their costs were compiled for each job. In order to develop the normal time it was necessary to use the judgment and experience of the team members in determining the size crew that would normally be assigned to each type of work using generally accepted methods. The associated normal cost was obtained from construction cost estimates.

As only a 40-hour week was authorized for the project, the crash times were obtained by considering only the maximum reasonable increase in manpower for each job and its effect on elapsed time. Additional costs were found necessary because of the extra congestion and activity on a job as crew size increased. Therefore the crash cost was obtained by adding the extra labor costs to the normal cost with an allowance for labor congestion. A straight line was then fitted to this data to obtain the job cost function described by equation (6).

As the plan for Project A took shape, it became clear that we had grossly underestimated the ability of the team. They went into far more detail than expected. This first application made it impractical to continue with the existing computer programs. Fortunately, Remington Rand had previously agreed to reprogram the system for a much larger computer — 1103A. This programming was expedited to handle the test application.

4. Some Results of the Project A Test

By March of 1958, the first part of the Project A test was complete. At that time it was decided that most of the work on Project A that was being subcontracted would be done by duPont. This change in outlook, plus design changes, caused about a 40% change in the plan of the project. Authorization was given to modify the plan and recompute the schedules. The updating which took place during April, required only about 10% of the effort it took to set up the original plan and schedule. This demonstrated our ability to stay "on top" of a project during the course of its execution.

Several other indicative results accrued from the Project A computations. With only 30% design information, we predicted the total manpower force curve with high correlation. The normal scheduling group had it building up at a rate too fast for the facility to handle in the initial stages of the project. (The reason for this is that they were unable to take available working space into account.) It was not until the project was under way that the error was caught, and they started cutting back the force to correspond with actual needs.

Early in the planning stages the normal scheduling group determined critical deliveries. The team ignored this information and included *all* deliveries in the analysis. There were 156 items in total. From the computed results it was determined that there would be only seven critical deliveries, and of these, three were not included in the list prepared by the normal scheduling group.

As estimated by traditional means, the authorized duration of Project A was put at N months. The computer results indicated that two months could be gained at no additional cost. Further, for only a 1% increase in the variable direct cost of the project an additional two months improvement could be gained. The intuitive tendency is to dismiss these results as ridiculous. However, if the project manager were asked for a four-month improvement in the project duration and he had no knowledge of the project cost curve, he would first vigorously protest that he could not do it. If pressed, he would probably quote a cost penalty many multiples of the current estimate and then embark on an "across-the-board" crash program. As a point of fact, the reason for the large improvement in time at such a small cost penalty was because only a very few jobs were critical — about 10% — and only these needed expediting. The difference in time of two months from N to N–2 can be explained as the possible error of gross time estimates and/or the buffering used in them.

5. *The Second Test Case*

With the successful completion of the Project A test, additional projects were authorized. Now the planning was to be done much earlier in the project life and was to incorporate more of the functions of engineering-design and procurement. Project B, capitalized at $2,000,000, was selected for this purpose. By July 1958, this second life test was completed and was as successful as the first. Unfortunately, the recession last year shelved the project so that it could not be followed through to completion.

Experience gained up to this point indicated that even greater capacity than the 1103A provided was essential. In consequence, programs were prepared for the 1105.

6. *Applications to Maintenance Work*

In the meantime, it was felt desirable to describe a project of much shorter duration so that the system could be observed during the course of the whole project. In this way improvements in the system design could be expedited. An ideal application for this purpose is in the shutdown and overhaul operation on an industrial plant. The overall time span of a shutdown is several days, as opposed to the several year span encountered in projects such as Project A.

The problems of scheduling maintenance work in chemical plants are somewhat different from those of scheduling construction projects. From time to time units like the blending, distillation and service units must be overhauled in order to prevent a complete breakdown of the facility and to maintain fairly level production patterns. This is particularly difficult to do when the plant operates at near peak capacity, for then it is not possible to plan overhauls so that they occur out of phase with the product demand. In such cases it is desirable to maximize return on investment. Because the variable costs usually are small in comparison to the down-time production losses, maximizing return on investment is equivalent to making the shutdown as short as possible.

For purposes of testing the Critical-Path Method in this kind of environment, a plant shutdown and overhaul was selected at duPont's Louisville Works. At Louisville they produce an intermediate in the neoprene process. This is a self-detonating material, so during production little or no maintenance is possible. Thus, all maintenance must be done during down-time periods. There are many of these shutdowns a year for the various producing units.

Several methods and standards people from Louisville were trained in the technique, and put it to the test. One of the basic difficulties encountered was in defining the plan of a shutdown. It was felt, for example, that because one never knew precisely what would have to be done to a reactor until it was actually opened up, it would be almost impossible to plan the work in advance. The truth of the matter is that the majority of jobs that can occur on a shutdown must be done every time a shutdown occurs. Further, there is another category that occurs with 100% assurance for each particular shutdown — scheduled design and improvement work. Most of the remaining jobs that can occur, arise with 90% or better assurance on any particular shutdown. These jobs can be handled with relative ease.

The problem was how to handle the unanticipated work on a shutdown. This was accomplished in the following way:

It is possible in most operating production units to describe, in advance, typical shutdown situations.

Prioı to the start of a given shutdown, a pre-computed schedule most applicable to the current situation is abstracted from a library of typical schedules. This schedule is used for the shutdown. An analysis of these typical situations proved sufficient because it was possible to absorb unanticipated work in the slack provided by the floaters. This is not surprising since it has been observed that only 10% of the jobs in a shutdown are critical.

However, if more unanticipated work crops up than can be handled by the schedule initially selected, then a different schedule is selected from the library. Usually less than 12 typical schedules are required for the library.

Costs for these schedules were ignored since they would be insignificant with respect to production losses. However, normal and crash times were developed for various levels of labor performance. The approach here is to "crash" only those jobs whose improved labor performance would improve the entire shutdown performance. The important consideration was to select minimum time schedules. Information on elapsed times for jobs was not immediately available but had to be collected from foremen, works engineering staff members, etc.

By March 1959, this test was completed. This particular application is reported in [1]. By switching to the Critical-Path Method, Louisville has been able to cut the average shutdown time from an average of 125 hours to 93 hours, mainly from the better analysis provided. Expediting and improving labor performance on critical jobs will cut shutdown time to 78 hours — a total time reduction of 47 hours.

The Louisville test proved so successful that the technique is now being used as a regular part of their maintenance planning and scheduling procedure on this and other plant work. It is now being introduced to maintenance organizations throughout duPont. By itself, the Louisville application has paid for the whole development of the Critical-Path Method five times over by making available thousands of pounds of additional production capacity.

7. *Current Plans*

Improvements have been made continually to the system so that today it hardly resembles the September, 1957, system. Further improvements are anticipated as more and more projects are tackled. Current plans include planning and scheduling a multimillion dollar new plant construction project. This application involves about 1800 events and between 2200 and 2500 jobs. As these requirements outstrip the capacity of the present computer programs, some aggregation of jobs was required which reduced the size of 920 events and 1600 jobs. This project includes *all* design, procurement and construction steps, starting with Part I authorization. (Part I is the point at which funds are authorized to proceed with sufficient design to develop a firm construction cost estimate and request Part II authorization.)

Also included in current plans are a four-plant remodernization program, several shutdown and overhaul jobs, and applications in overall product planning.

8. *Computational Experience*

The Critical-Path Method has been programmed for the UNIVAC I, 1103A, and 1105 with a Census Bureau configuration. These programs were prepared so that either UNIVAC I or the 1100 series computers may be used independently or in conjunction with one another.

The limitations on the size problems that the available computer programs can handle are as follows: UNIVAC I — 739 jobs, 239 events; 1103A — 1023 jobs, 512 events; 1105 — 3000 jobs, 1000 events.

In actual practice input editing has been done on duPont's UNIVAC I in Newark, Delaware and computation and partial editing on 1100 series machines at Palo Alto, St. Paul, and Dayton. Final editing has then been done at Delaware. System compatibility with magnetic tapes has been very good. In one major updating run, input, output and program tapes were shipped by air freight between Palo Alto and Delaware.

Generally computer usage represents only a small portion of the time it takes to carry through an application. Experience thus far shows that, depending on the nature of the project and the information available, it may take from a day to six weeks to carry a project analysis through from start to finish. At this point it is difficult to generalize. Computer time has run from one to 12 hours, depending on the application and the number of runs required. (Seven runs were required to generate the library for the Louisville project.)

Input and output editing has run less than 10% of the cost curve computations. Indeed, the determination of the earliest and latest start and finish times, and total and free float for a project of 3000 jobs and 1000 events takes under 10 minutes on the 1100 series computers. This run includes input editing computation, and output editing. If a series of these runs are to be made on the output solutions from the cost curve computation, only from three to four minutes more are required for each additional solution.

Table 1 indicates typical cost curve computation times. Of the total number of characteristic solutions that this computation produces, no more than 12 ever have been output edited. The reason for this is that many of the characteristic solutions have very small differences in total project duration.

It has been found that fruitful use of parts of the Critical-Path Method do not require extensive computing facilities. The need for the hardware is dictated by economics and depends upon the scope of

TABLE I

TYPICAL RUN TIMES

Events	Jobs	Sol'n's	Minutes	
			UNIVAC I	1103A/1105
16	26	7	8	1
55	115	14	125	3
385	846	21	—	100
437	752	17	—	24
441	721	50	—	49
920	1600	40	—	210

Fig. 4—Typical run times.

the application and the amount of computation that is desired.

9. A Parallel Effort.

Early in 1958 the Special Projects Office of the Navy's Bureau of Ordnance set up a team to study the prospects for scientifically evaluating progress on large government projects. Among other things the Special Projects Office is charged with the overall management of the Polaris Missile Program which involves planning, evaluating progress and coordinating the efforts of about 3000 contractors and agencies. This includes research, development and testing activities for the materials and components in the submarine-launched missile, submarine and supporting services.

A team staffed by operations researchers from Booz, Allen & Hamilton, Lockheed Missile Systems Division and the Special Projects Office made an analysis of the situation. The results of their analysis represent a significant accomplishment in managing large projects although one may quibble with certain details. As implemented, their system essentially amounts to the following:

(1) A project diagram is constructed in a form similar to that treated earlier in this paper.

(2) Expected elapsed time durations are assigned to each job in the project. This data is collected by asking several persons involved in and responsible for each job to make estimates of the following three quantities:

a. The most optimistic duration of the job
b. The most likely duration, and
c. The most pessimistic duration.

(3) A probability density function is fitted to this data and approximations to the mean and variance are computed.

(4) Expected earliest and latest event times are computed, using expected elapsed times for jobs, by means of equations (1) and (2) of Part I. Simultaneously variances are combined to form a variance for the earliest and latest time for each event.

(5) Now, probabilistic measures are computed for each event, indicating the critical events in the project.

(6) Finally, the computed schedule is compared with the actual schedule, and the probabilities that actual events will occur as scheduled are computed.

This system is called PERT (Program Evaluation and Review Technique). The computations involved are done on the NORC Computer, Naval Proving Grounds, Dahlgren, Virginia. More information about PERT may be found in references [2], [11] and [12].

There are some aspects of the PERT system and philosophy to which exception might be taken. Using expected elapsed times for jobs in the computations instead of the complete probability density functions biases all the computed event times in the direction of the project start time. This defect can be remedied by using the calculation indicated by equation (4) of Part I. Further, it is difficult to judge, *a priori*, the value of the probability statements that come out of PERT: (1) because of the bias introduced; (2) because of the gross approximations that are made; (3) because latest event times are computed by running the project backward from some fixed completion time. If there is good correlation with experience then these objections are of no concern. At this moment we are in no position to report the actual state of affairs.

Finally, PERT is used to evaluate implemented schedules originally made by some other means, usually contract commitments made by contractors. To be of most value PERT, or for that matter the Critical-Path Method, should be used by the contractor in making the original contract schedule. In this way many of the unrealities of government project work would be sifted out at the start.

10. Extensions of the Critical-Path Method

The basic assumption that underlies the Critical-Path Method, as developed thus far, is that adequate resources are available to implement any computed schedule. (In some cases, this assumption can be avoided by inserting certain types of delivery and completion restraints in the project plan. However, in many cases this is an unrealistic assumption.)

Apparently there are two extremes that need to be considered:

(1) Available resources are invested in *one* project.

(2) Available resources are shared by *many* projects.

In the first case experience has shown that there is usually no difficulty in implementing any computed schedule. Any difficulty that does arise seems to be easily resolved. The Critical-Path Method applies very well in this case. It may be called *intra*-project scheduling.

In the second case, however, we run into difficulties in trying to share men and equipment among several projects which are running concurrently. We must now do inter-project scheduling.

The fundamental problem involved here is to find some way to define an objective for all projects which takes the many independent and combinatorial restraints involved into account: priorities, leveling manpower by crafts, shop capacity, material and equipment deliveries, etc. For any reasonable objective, it also is required to develop techniques for handling the problem. Preliminary study has indicated that this is a very difficult area of analysis and requires considerable research. However, it is felt that the Critical-Path Method as it stands can form a basis for systems and procedures and for the requisition of data for this extension of scheduling.

It would be of some interest to extend the method to the case where job durations and costs are random variables with known probability density functions. The mathematics involved appears to be fairly difficult. Due to the problems of obtaining data in this form, such an extension may be purely academic for several years to come.

11. Other Applications

The potential applications of the Critical-Path Method appear to be many and varied. Consider the underlying characteristics of a project — many series and parallel efforts directed toward a common goal. These characteristics are common to a large variety of human activities. As we have seen, the Critical-Path Method was designed to answer pertinent questions about just this kind of activity.

We have already treated applications of the technique to the construction and maintenance of chemical plant facilities. The obvious extension is to apply it to the construction and maintenance of highways, dams, irrigation systems, railroads, buildings, flood control and hydro-electric systems, etc. Perhaps one of the most fruitful future applications will be in the planning of retooling programs for high volume production plants such as automotive and appliance plants.

We have also seen how it can be used by the government to report and analyze subcontractor performance. Within the various departments of the government, there are a host of applications — strategic and tactical planning, military base construction, construction and overhaul of ships, missile countdown procedures, mobilization planning, civil defense, etc. Within AEC alone, there are applications to R & D, design and construction of facilities, shutdown, clean-up, and start-up of production units. Another example is in the production use of large equipment for the loading and unloading portion of the production cycle of batch processes. Because each of these operations is of a highly hazardous nature, demanding very close control and coordination of large numbers of men and/or complex equipment, they appear to be natural applications for the Critical-Path Method.

Common to both government and industry are applications that occur in the assembly, debugging, and full-scale testing of electronic systems.

References

[1] Astrachan, A., "Better Plans Come From Study of Anatomy Of An Engineering Job," *Business Week*, March 21, 1959, pp. 60-66.

[2] Fazar, Willard, "Progress Reporting in the Special Projects Office," *Navy Managemen. Review*, April 1959, pp. 9-15.

[3] Ford, L. R., Jr. *and* D. R. Fulkerson, "A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem," *Canadian Journal of Math.*, Vol. 9, 1957, pp. 210-218.

[4] Kelley, J. E., Jr., "Computers and Operations Research in Roadbuilding," *Operations Research, Computers and Management Decisions*, Symposium Proceedings, Case Institute of Technology, Jan. 31, Feb. 1, 2, 1957.

[5] ——————, "The Construction Scheduling Problem (A Progress Report)" UNIVAC Applications Research Center, Remington Rand UNIVAC, Philadelphia, April 25, 1957. (Ditto)

[6] ——————, "Parametric Programming and The Primal-Dual Algorithm," *Operations Research*, Vol. 7, No. 3, 1959, pp. 327-334

[7] ——————, "Critical-Path Planning and Scheduling: Mathematical Basis," in preparation.

[8] ——————, "Extension of the Construction Scheduling Problem: A Computational Algorithm," UNIVAC Applications Research Center, Remington Rand UNIVAC, Philadelphia, Nov. 18, 1958. (Ditto)

[9] ——————, *and* M. R. Walker, "Critical-Path Planning and Scheduling: An Introduction," Mauchly Associates, Inc., Ambler, Pa., 1959.

[10] Martino, R. L., "New Way to Analyze and Plan Operations and Projects Will Save You Time and Cash," *Oil/Gas World*, September, 1959, pp. 38-46.

[11] *PERT, Program Evaluation Research Task*, Phase I Summary Report, Special Projects Office, Bureau of Ordnance, Dept. of the Navy, Washington, July 1958.

[12] Malcolm, D. G., J. H. Roseboom, C. E. Clark *and* W. Frazar, "Application of a Technique for Research and Development Program Evaluation," *Operations Research*, Vol. 7, 1959, pp. 646-669.

Discussion

Mr. Carkagan (Western Electric): In Slide 10 you did not include negative costs due to earlier payoff resulting from earlier implementation. Do you in effect recognize these gains?

Mr. Walker: Yes, we do. That slide was an artistic rendition. It didn't illustrate the fact in this particular case.

T. J. Berry (Bell Tel. of Penn.): What techniques were employed to ascertain where you actually were in relation to the estimated schedule?

Mr. Walker: In the first slide it was not intended to use this particular schedule. What we did was in one case we placed a man on the field site. He had the traditional schedule and our schedule, and he observed what actually took place. In this manner we made a comparison.

More formal arrangements, in terms of a reporting system, have been developed.

B. Silverman (*Syracuse*): Is the assumption of linear variation of cost versus time for completion of job realistic? Have you tried other approximations?

Mr. Walker: The assumption is realistic from two standpoints: first, the cost of many jobs does vary linearly with time; second, if the cost curve is non-linear, it is, more often than not, difficult or impossible to determine. Thus, a linear approximation is reasonable. However, we have developed a method of using a piece-wise linear approximation to the cost curve. Ample accuracy has been obtained, thus far, by using linear approximations.

Mr. Kelley: For some jobs you don't have a continuous variation of cost with time. For instance, when pouring concrete you may elect to use ordinary concrete or a quick-setting concrete. In the first case you have to wait a relatively long time for the setting and curing to take place. The reverse is true in the second case. There is no intermediate curing time. To treat cases like this precisely involves solving a combinatorial problem of high order. We approximate the situation by assuming a continuous variation of cost with the job's elapsed time. The results are then rounded-off to their proper values.

E. I. Pina (*Boeing*): It is possible that, based on a normal point schedule, you would be late in delivery? Thus you may wish to shorten whole schedules. In doing so, a previously critical path may be replaced by another new critical path. How do you make sure this does take place?

Mr. Walker: Once you have a critical path and you wish to compress the schedule you are going to add more critical paths. One is not going to drop out entirely from the picture.

The Automatic Digital Computer as an Aid in Medical Diagnosis

C. B. CRUMB, JR.† and C. E. RUPE, M. D.‡

Introduction

THE IDEA to be presented here is that an automatic digital computer can provide great assistance to the medical diagnostician by rapid calculation, based upon symptoms and physical findings, of the relative probability that a certain disorder fits that set of findings. In a given diagnostic problem, the physician is practically always able to select the correct category of disorders which will include the disorder which constitutes the correct diagnosis. But it is also true that in many such categories the distinction between the different disorders included is quite difficult.

This paper proposes that the incidence of correct diagnosis in such difficult cases may be very greatly increased by the use of systematic statistical correlation technique. Basically, the idea is that for each category of disorders, the computer remembers a large number of cases, each with the correct diagnosis and the complete pattern of associated symptoms and physical findings. From these, it forms correlation constants between symptoms and disorders, and is able when confronted with a new set of symptoms to bring to bear all of the case history evidence in selection of the disorder most likely to be the correct diagnosis for the particular case.

The need for the development of this technique is demonstrated by three conditions which apply to present day diagnostic procedures. The most important of these is the fact that up to this time medical diagnosis has remained, to a marked degree, an art. Although some scientific methods are utilized, the necessary practical limitations of laboratory and clinical procedures, plus the great complexity of many diagnostic problems have thus far prevented the full development of a *science* of medical diagnosis. This plan of statistical diagnosis with calculating machine aid should enable the medical doctors to move much closer to the "science" objective, and achieve a markedly increased incidence of correct diagnosis.

The other two justifications hinge on the cost of present day medical services. The computer technique should greatly increase the speed of reaching a decision in diagnosis, and thus should very significantly reduce the total diagnosis cost. Furthermore, in many instances, it may be that a clear cut differentiation reported by the digital calculator would make unnecessary further expensive and time consuming clinical and laboratory procedures which might now be employed to verify an initial diagnosis. The second area of cost saving would be avoidance of expensive hospitalization which might be rendered unnecessary by quick and accurate diagnosis.

There is a by-product virtue in this technique which should not be overlooked. This is that in order to use the technique, it is necessary first to record all symptoms and manifestations concerning the particular disorder in a systematic and uniform fashion. This is fed into the machine for the calculation of appropriate probabilities. That same information also becomes a part of the machine memory and of the foundation for correlation constants. Thus, the record on a particular case is not only made fully and recorded permanently, but quickly added to the body of statistical data upon which future diagnoses of cases in that category will be based. This represents a very marked improvement over present day case history recording, filing and utilization.

Technique Proposed

Here we will use the term "symptom" in the broad sense to include all pertinent forms of evidence, — patient's reports of symptoms, doctor's examination, laboratory test results, etc.

Assume a table of disorders D_a, D_b, D_c, etc., and symptoms S_1, S_2, S_3, etc.; let's say we can derive a correlation constant C expressing the probability of association of each symptom with each disorder. Then, we will have a table like Table I.

TABLE I

FORM OF THE TABLE OF CORRELATION CONSTANTS

Disorders	D_a	D_b	D_c	D_d
Symptoms				
S_1	C_{1a}	C_{1b}	C_{1c}	C_{1d}
S_2	C_{2a}	C_{2b}	C_{2c}	C_{2d}
S_3	C_{3a}	C_{3b}	C_{3c}	C_{3d}
S_4	C_{4a}	C_{4b}	C_{4c}	C_{4d}
S_5	C_{5a}	C_{5b}	C_{5c}	C_{5d}
S_6	C_{6a}	C_{6b}	C_{6c}	C_{6d}
S_7	C_{7a}	C_{7b}	C_{7c}	C_{7d}
S_8	C_{8a}	C_{8b}	C_{8c}	C_{8d}

† Bendix Aviation Corporation, Seattle, Washington.
‡ Henry Ford Hospital, Detroit, Michigan.

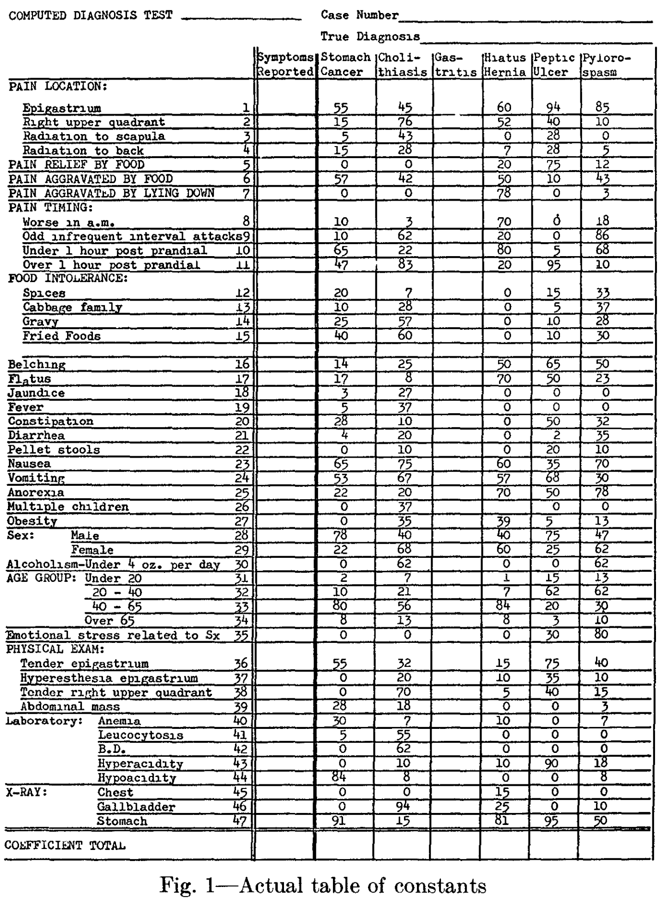
Now the simplest imaginable approach to the task is to note a set of observed and reported symptoms such as S_2, S_3, S_5, and S_8 applying to the case at hand; and for each disorder in the table obtain the total of the corresponding correlation constants.

TABLE II

Process of Obtaining Relative Probability Index Numbers

Disorders	D_a	D_b	D_c	D_d
Symptoms				
S_2	C_{2a}	C_{2b}	C_{2c}	C_{2d}
S_3	C_{3a}	C_{3b}	C_{3c}	C_{3d}
S_5	C_{5a}	C_{5b}	C_{5c}	C_{5d}
S_8	C_{8a}	C_{8b}	C_{8c}	C_{8d}
	N_a	N_b	N_c	N_d

This process is illustrated in Table II, which is Table I with the not-observed symptoms and corresponding constants deleted, and the totals indicated. The largest total N should be the correct diagnosis, or the one with highest correctness probability. We call the totals N the relative probability index numbers; they don't represent quantitative probability, but rather they rank the diseases in the group from "most likely" to "least likely" to be the true diagnosis for the particular case. Fig. 1 is the actual set of correlation constants used in our experimental work.



COMPUTED DIAGNOSIS TEST ______________ Case Number______________

True Diagnosis______________

		Symptoms Reported	Stomach Cancer	Choli-thiasis	Gas-tritis	Hiatus Hernia	Peptic Ulcer	Pyloro-spasm
PAIN LOCATION:								
Epigastrium	1		55	45		60	94	85
Right upper quadrant	2		15	76		52	40	10
Radiation to scapula	3		5	43		0	28	0
Radiation to back	4		15	28		7	28	5
PAIN RELIEF BY FOOD	5		0	0		20	75	12
PAIN AGGRAVATED BY FOOD	6		57	42		50	10	43
PAIN AGGRAVATED BY LYING DOWN	7		0	0		78	0	3
PAIN TIMING:								
Worse in a.m.	8		10	3		70	0	18
Odd infrequent interval attacks	9		10	62		20	0	86
Under 1 hour post prandial	10		65	22		80	5	68
Over 1 hour post prandial	11		47	83		20	95	10
FOOD INTOLERANCE:								
Spices	12		20	7		0	15	33
Cabbage family	13		10	28		0	5	37
Gravy	14		25	57		0	10	28
Fried Foods	15		40	60		0	10	30
Belching	16		14	25		50	65	50
Flatus	17		17	8		70	50	23
Jaundice	18		3	27		0	0	0
Fever	19		5	37		0	0	0
Constipation	20		28	10		0	50	32
Diarrhea	21		4	20		0	2	35
Pellet stools	22		0	10		0	20	10
Nausea	23		65	75		60	35	70
Vomiting	24		53	67		57	68	30
Anorexia	25		22	20		70	50	78
Multiple children	26		0	37			0	0
Obesity	27		0	35		39	5	13
Sex: Male	28		78	40		40	75	47
Female	29		22	68		60	25	62
Alcoholism-Under 4 oz. per day	30		0	62		0	0	62
AGE GROUP: Under 20	31		2	7		1	15	13
20 - 40	32		10	21		7	62	62
40 - 65	33		80	56		84	20	30
Over 65	34		8	13		8	3	10
Emotional stress related to Sx	35		0	0		0	30	80
PHYSICAL EXAM:								
Tender epigastrium	36		55	32		15	75	40
Hyperesthesia epigastrium	37		0	20		10	35	10
Tender right upper quadrant	38		0	70		5	40	15
Abdominal mass	39		28	18		0	0	3
Laboratory: Anemia	40		30	7		10	0	7
Leucocytosis	41		5	55		0	0	0
B.D.	42		0	62		0	0	0
Hyperacidity	43		0	10		10	90	18
Hypoacidity	44		84	8		0	0	8
X-RAY: Chest	45		0	0		15	0	0
Gallbladder	46		0	94		25	0	10
Stomach	47		91	15		81	95	50
COEFFICIENT TOTAL								

Fig. 1—Actual table of constants

In the very limited testing of the idea which has been done thus far, this straightforward simple summation scheme has been tried and yielded the encouraging results shown in Figs. 2 through 7. It is clear that some more sophisticated approach may have to be used for fullest effectiveness of the technique. For example, it appears that in some instances the correlation constant C must contain not only the occurrence correlation but also a factor representing the importance of the symptom in indicating the particular disorder, that is, whether or not it is a primary symptom having a clear cut relation to the disorder. It also appears that the intensity of certain symptoms should be noted in the preparation of the summation equations yielding the relative probability figures. For example, a mild abdominal pain might be considered a generally inconclusive evidence; while an intense abdominal pain localized in a certain area would be much more telling.

There may also be some more decidedly non-linear factors in this technique such as the existence of a symptom which positively eliminates one or more of the disorders in the appropriate category. Another possibility is that a certain combination of two or more symptoms may positively eliminate or positively identify a certain disorder within the appropriate category. A further example is the importance of history of symptoms in the preparation of the input to the computer. Thus the time rate of change of certain conditions occurring among the symptoms may be highly significant in the determination of the diagnosis.

In spite of all of the above possible changes, we feel strongly that the fundamental technique of summation of an appropriate set of correlation constants, modified or unmodified, will be very effective in indicating the most likely correct diagnosis.

Medical Foundations

The history of medicine shows a well established parallel of the strides in therapeutics and diagnostic methods. When the practice of medicine was confined largely to the relief of symptoms it often mattered little, except academically, whether or not the precise diagnosis was made so long as the resulting symptoms could be managed. In the current era, on the other hand, remarkable strides in specific treatments for diseases have greatly increased the need for improved precision in diagnosis.

Present day diagnosis is largely an intuitive and subjective process which is not easily described. Essentially it consists of the collection of data and interpretation leading to the synthesis of a diagnosis. The data collected by the patient's recounting of symptoms and the doctor's examination are interpreted in the light of the physican's training, experience, interest, and knowledgeof published statistics.

While the data collection process may be fairly uniform among various doctors, the interpretation in some areas of medicine is highly variable. These areas are the rare and unusual disorders.

Past efforts to attack that problem of variability in interpretation of symptoms have lead to the evolution of the practice called differential diagnosis. This is the technique of listing all the possible disease processes which might account for the patient's symptoms and physical findings. Each of these possibilities is individually related to the pattern of symptoms, and the diagnosis selected by an elimination process. The big weakness of this technique lies in the fact that the disease list *must* contain the correct diagnosis. If it is not there, the further study of the patient is not oriented toward the discovery of the proper diagnosis unless the various laboratory examinations happen to overlap the areas between the differential diagnosis and the correct diagnosis.

From this standpoint, then, the physician as a diagnostician, which he must be if he is to choose precise therapy, is the repository of data from his training, his reading of the current literature, and his day-to-day experience. This body of knowledge to be applied at the bedside in deriving a diagnosis from symptoms and signs suffers a few obvious disadvantages. First, the individual physician's practice may or may not have contained experience with the unknown disease at hand. Second, the individual physician's memory, not only of past experience but of past training and current efforts to keep abreast of his field, may not be adequate to the task. Finally and much less frequently, the individual physician's own interest in one area of disease may focus attention in this direction effectively raising blinders to his diagnostic acumen — this in contrast to blind spots in his diagnostic acumen that may always have been blank because of rarity of disease or absence of experience.

In choosing a group of diseases to test the proposition of the use of simple statistical procedures in making diagnoses, it was felt that a group of diseases diagnosable with certainty by some objective method was the first requirement. The second requirement for the test group of diseases was that they have enough overlapping symptomatology that diagnosis by history and physical examination presented some difficulties. The next problem was that of converting the subjective and intuitive information given by the patient to the physician to black, white, and reasonable shades of gray. Finally, and most difficult, is the construction of the statistical pattern of the disease to serve as a template with which to compare the individual patient's data. This particular difficulty serves to focus attention upon the need for the collection and collation of dependable statistics about the occurrence of major and minor manifestations of a given disease.

The diseases chosen are those involving the upper gastrointestinal tract and its appendages and all causing intermittent upper abdominal pain. The validity of the proposal was then ready for examination in the trial here reported. The construction of the statistical templates and the collection of the data from the individual patients for the trial pointed up additional problems such as the relative importance of individual symptoms and combinations of symptoms which bore greater weight in the consideration of one diagnosis versus others in the group. In this respect the attempts to objectify and "to mechanize" the subjective process will serve to further objectify diagnostic methods and procedures.

Steps in Development of the Technique

The following is considered to be a logical sequence of steps toward evolving a technique and testing the idea.

(1) *Test Area Selection*

This is selection of a group of diseases or disorders which are symptomatically similar and difficult to diagnose; and for which a sizable body of data in the form of actual case histories exists.

(2) *Test Data Compilation*

A statistical number of cases from the literature on the selected category must be reviewed. The symptom data must be extracted in sufficiently uniform format to permit derivation of correlation constants.

(3) *Derivation of Constants*

This is the "dog work" of actually counting up the incidence of each manifestation as associated with each disorder within the selected category.

(4) *Test Computations and Evaluation*

First trials of the method should be made using the cases upon which the table was based. This enables development of the technique, as described in the following steps, without introduction of new variables.

(5) *Development of the Correlation Technique*

Here the so called nonlinearities or irregularities in the preparation and use of the correlation constant table will be tried and introduced as they are found to be effective in improving the resolution of the technique.

(6) *Establishment of Data Format and Computer Programs*

This is the adaptation of the process to high-speed machine use. The two major tasks are selection of an appropriate form for input information, and programming.

(7) *Parallel Use With Conventional Diagnosis in New Cases*

With no change in technique, doctors will make diagnoses as they are now being routinely made, while in parallel with that the computed diagnosis technique would be utilized and the effectiveness compared.

Progress to Date

For the first trial of this idea, the authors selected a set of five abdominal disorders, — cholelithiasis (gall stones), cancer of the stomach, peptic ulcer, hiatus hernia, and pylorospasm. In the hope of making an inexpensive first trial, we elected to derive the initial set of correlation constants by asking several specialists in internal medicine to report their estimates of the various constants. Each was given a blank table with the five disorders, each heading a column; and 48 rows for the 48 symptoms of significance in this set of disorders. The correlation constants which they reported were averaged to obtain a trial set of constants. These were then tested with a few case histories. The results were correct; but the difference between the correct and the incorrect probability index numbers was very little. Furthermore, in some of the manifestations, the doctors reported rather markedly different estimates of the appropriate correlation; and this cast doubt on the validity of the particular trial. It was evident that the rather laborious formulation of a true statistical basis would have to be performed.

This has been done, utilizing a statistical number of case histories for each of the five disorders. With the resulting table of constants, Fig. 1, the technique is now being tested on the same case histories. This puts us in Step 4 of the development sequence described in the last section. The first results are shown in Figs. 2 through 7.

Fig. 2 shows the result for 23 cases of hiatus hernia and 22 of peptic ulcer, in all of which the actual disorder was determined by X-ray. Each case is represented by a vertical column of the five different symbols. The vertical position of a symbol indicates N, the relative probability index number. The horizontal position for each column has no significance except that within each group the cases were plotted arbitrarily in descending order of highest N for the case. This places at the left the cases in which many symptoms were noted, proceeding to those with fewer symptoms at the right.

If the method were perfectly effective the flagged symbols would all be at the top of the mass in both sets. This is fairly closely approached in the peptic ulcer set; but with hiatus hernia the results are less encouraging. It is interesting to note that in about one-sixth of all the cases, the total for hiatus hernia was closely approximated by that of cholelithiasis.

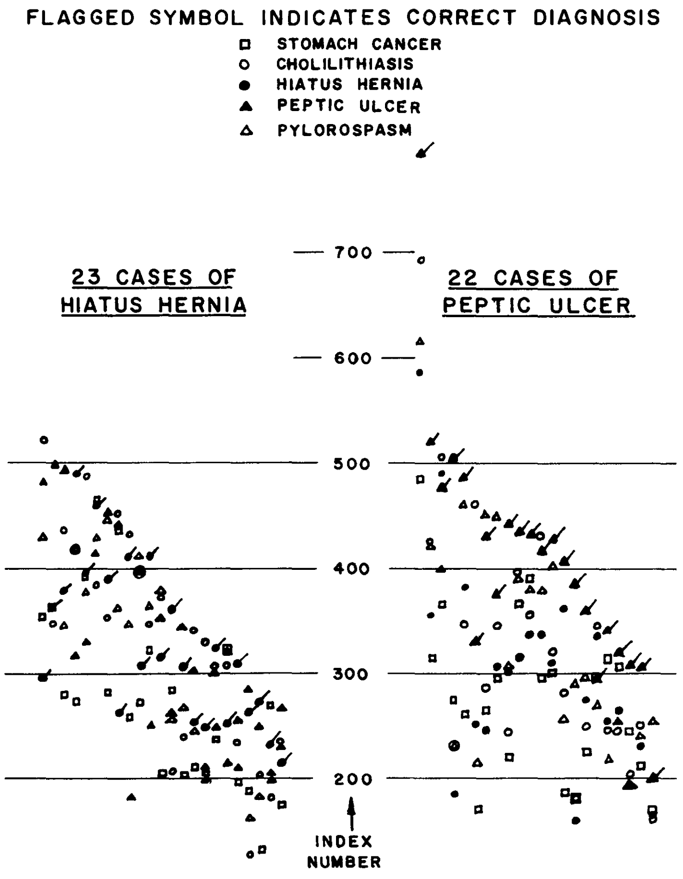


Fig. 2—Probability index numbers for 45 cases.

In all of these there was a history of gall bladder surgery or an established diagnosis of gall stones in addition to the hiatus hernia.

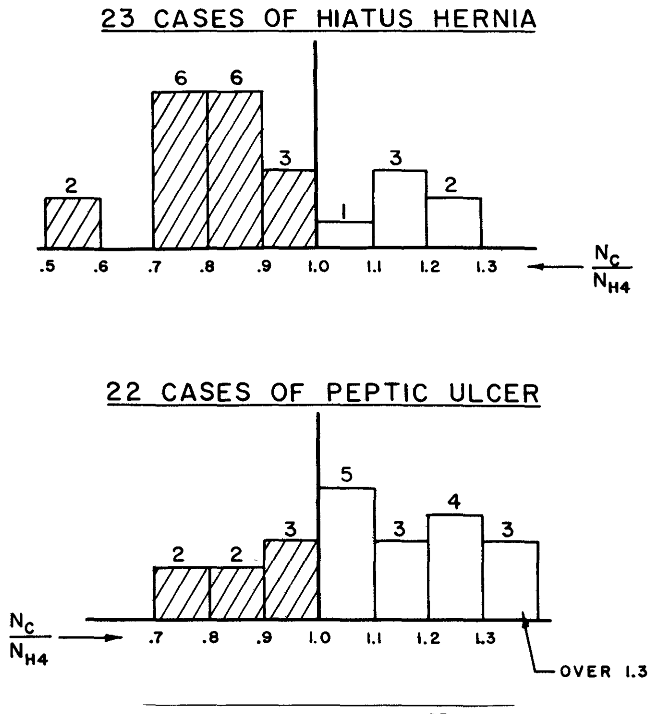


Fig. 3—Distribution of $\frac{N_C}{N_{H4}}$.

Fig. 3 is an effort to summarize the most significant data of Fig. 2. For each case the ratio of N for the correct diagnosis to the highest of the other four N values was computed; and the distribution of these ratio values is represented by Fig. 3. Where this ratio $N_C : N_{H4}$ is greater than one, the correct diagnosis has been defined by the technique. The cross-hatched bars represent groups of cases where the ratio value is less than one and the technique has failed.

Figs. 4, 5, and 6 are like Fig. 3; they show the test results for the other three diseases in our test group. They are made up of somewhat smaller numbers of cases, but the results are very good, only 4 failures in 35 cases.

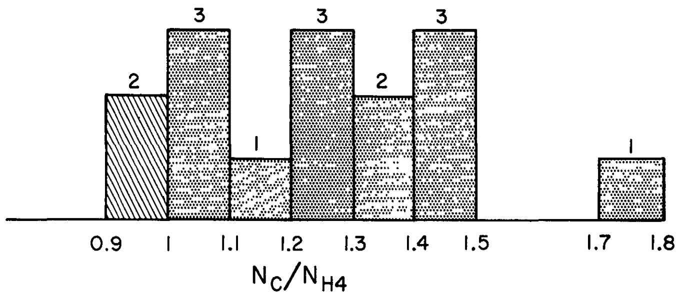


Fig. 4—Pylorospasm.

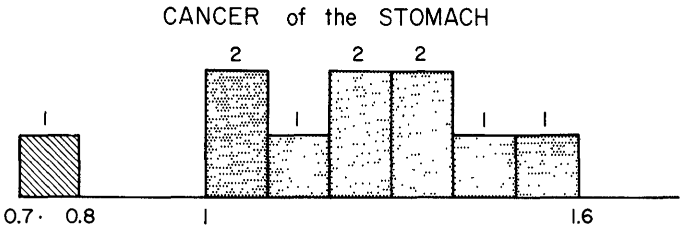


Fig. 5—Cancer of the stomach.

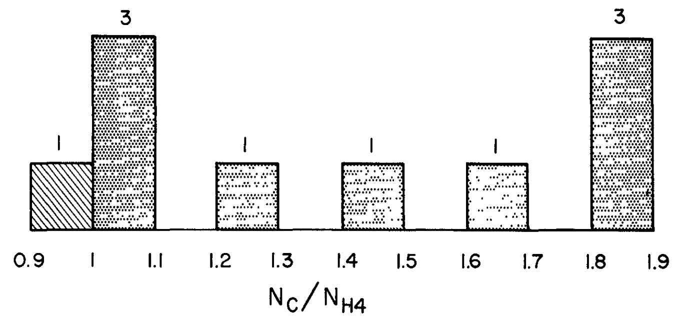


Fig. 6—Cholelithiasis.

Fig. 7 is a summary chart, like Figs. 3 through 6 but representing the whole test group of five diseases. It shows that in the test mass of 80 cases, there were 52 correct diagnoses and 28 failures for this initial trial technique. Nearly two-thirds of the failures were in cases of hiatus hernia, a disorder difficult to diagnose and often complicated by the coexistence or history of other abdominal troubles.

These figures, based on very few cases, still make it pretty clear that refinements of the method, suggested in the section on technique, will indeed be essential.

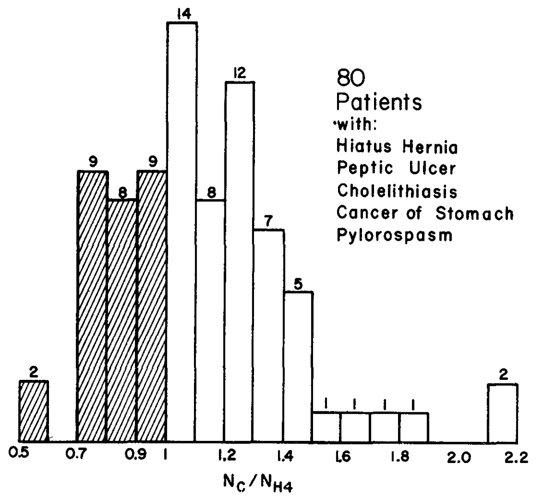


Fig. 7—Summary of test results.

Computer Considerations

The block diagram, Fig. 8, shows the organization of a digital calculator considered suitable to this problem. Assuming for the moment that the machine would be used for the analysis of individual cases, one at a time, then the determination of the diagnosis probability index numbers would involve the following steps:

(1) Place in the working storage of the computer the input information which consists of the disorder category, selected by the physician, and the manifestations noted and reported by him.

(2) According to the disorder category, search the bulk storage for the appropriate table of correlation constants and transfer it to the working storage.

(3) Modify any of the constants in the table according to any of the various nonlinear effects which were noted in the section on technique.

(4) Extract the appropriate correlation constants under the heading of each of the disorders in the category, and accumulate the sum for each of the disorders.

(5) Report the appropriate sum for each disorder.

The bulk storage requirement shown in the diagram might be satisfied by a magnetic tape unit, a large capacity magnetic drum, or a large capacity random access memory unit such as the I. B. M. Ramac. A typical table of correlation constants is expected to require perhaps one thousand decimal digits. Thus, a moderate capacity magnetic drum as working storage could easily accept one such table of correlation constants and a fairly detailed program.

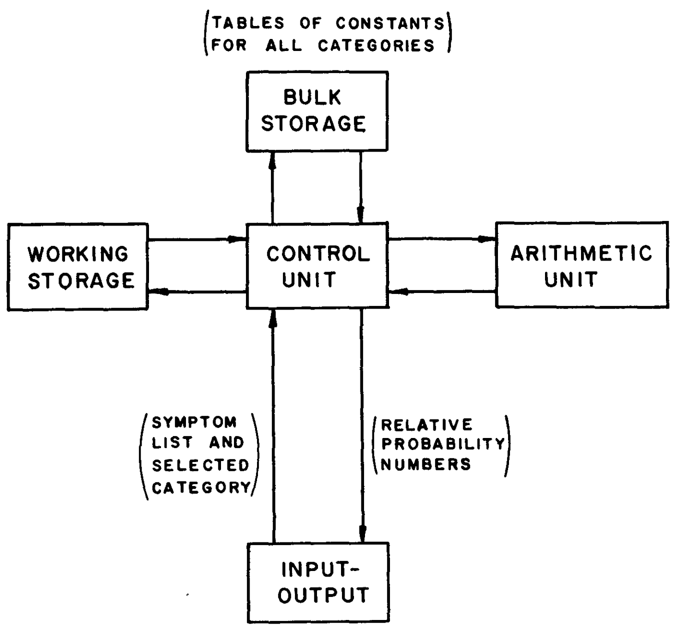

TABLES OF CONSTANTS FOR ALL CATEGORIES
BULK STORAGE
WORKING STORAGE
CONTROL UNIT
ARITHMETIC UNIT
SYMPTOM LIST AND SELECTED CATEGORY
RELATIVE PROBABILITY NUMBERS
INPUT-OUTPUT

Fig. 8—Machine organization.

If we look at the diagnosis operation as a whole, it becomes evident that the time requirement for manual preparation of the input information for the individual diagnosis is going to be of the order of several minutes. Thus, it would seem that allowance of two minutes for the total computer operation would not be excessive. On that scale, the actual arithmetic operations time becomes practically insignificant; and the most of the time can be allotted to the extraction of the appropriate correlation constants table and placing it in the working storage. If something like one minute and a half is allotted to this, then a magnetic tape servo should be sufficiently fast.

This line of thought leads to the conclusion that an inexpensive medium size general purpose machine with magnetic tape facility, such as the Bendix G-15, would be entirely appropriate.

It may turn out to be desirable to treat diagnosis problems in groups, by accumulation of a number of cases, prepared for machine input, and all to be handled in a continuous machine run. In this method, the machine time requirement per case may become much more important, and faster bulk storage, working storage, and arithmetic unit may be called for.

With such a small machine time requirement per case, the use of a single machine serving many hospitals, clinics, and independent doctors is a leading possibility. This scheme substitutes a communications requirement for the local machine requirement. By intelligent selection of coding and format, the information content of these messages will be kept low enough such that existing communications media such as teletype will be adequate. Thus it appears almost certain that the central computer service will be the most practical way of applying the technique.

You may note that the real problem in this project is not the computer, nor the program, but rather the development of statistical correlation procedures. That is true; and we admit that we haven't yet made use of an electronic calculator in the experimental work. From these facts one might conclude that high speed machine use is only incidental to the idea. Of course, that isn't so.

The statistical procedure proposed cannot be applied practically unless it can be performed rapidly. Thus in a sense we have a classic computer application, one in which the machine is not just a convenience nor a luxury, but rather an essential requirement. The job simply can't be done without it.

Conclusions

No doubt it is quite evident from the limited progress which we reported above, that we are hardly in a position to state many conclusions on the effectiveness of this scheme. There are two statements which we can make in which we feel very confident. One is that, qualitatively at least, the scheme is definitely effective although some doubt still remains about the capability of the idea, as it is now visualized, to make a sharp and unmistakable differentiation of the correct from the incorrect diagnosis.

Our second conclusion is in a sense a by-product one arising from the observations during the first preparation of correlation constants. This is that the opportunity for improving the doctor's basis for diagnosis is greater than we expected. Present day diagnosis ideally is based upon all the information to which the diagnostician has been exposed, — his total textbook and classroom content plus all the data from his daily practice. This ideal is not realized because of the limitations of the human memory, and the differences in constants reported by several internists suggested that the natural distillation of memory yields quite variable results in various men.

On the other hand, the digital computing machine can remember and utilize all the data it is given; and that fact is the foundation for our strong confidence in the need for and effectiveness of the proposed technique.

This is the real essence of our notion that here is another way that automatic computers can contribute to the good of mankind.

Bibliography

[1] Dr. K. Brodman, Dr. A. J. van Woerkom, Dr. A. J. Erdmann, Jr., and Dr. L. S. Goldstein, "Interpretation of Symptoms with a Data-Processing Machine," Reprinted from the *A. M. A. Archives of Internal Medicine*, May 1959, Vol. 103 pp. 776–782.

[2] R. S. Ledley and L. B. Lusted, "Reasoning Foundations of Medical Diagnosis," *Science*, Vol. 130, No. 3366, pp. 9–21, July 3, 1959.

[3] D. M. Cleary "The Role of Automation in Diagnostics," *Current Medical Digesi*, April 1959, pp. 57-63.

[4] "Dr. Automation," *Time*, Vol. 74, No. 6, p. 48, August 10, 1959, on work at System Development Corporation under C. J. Roach.

[5] M. Lipkin and J. D. Hardy, "Mechanical Correlation of Data in Differential Diagnosis of Hematological Diseases," *Journal of American Medical Association*, Vol. 166, No. 2, pp. 113-125, Jan. 11, 1958.

[6] C. L. Chiang, J. L. Hodges, Jr., and J. Yerushalny, "Statistical Problems in Medical Diagnosis," Third Berkeley Symposium on Mathematical Statistics and Probability, 1954-1955, Vol. 4, pp. 121-123, University of California Press, Berkeley 1956.

[7] "Dr. Robot," *Time*, Vol. 68, p. 44, August 20, 1956. Punched card machine diagnosis of diseases of cornea, by Dr. Francois Paycha

[8] Charbonnier, Cyffers, Schwartz, and Vessereau, "Discrimination entre ictères médicaux et chirurgicaux a partir des résultats de l'analyse électrophorétique des protéines du sérum," *Revue Internationale d'Hepatologie*, Vol. 5, No. 7, 1955.

[9] C. F. Paycha, "Memoire diagnostique," *Montpellier Med.*, Vol 47, p. 588, 1955.

Discussion

N. Shapiro (Nat'l. Institute of Health): One should distinguish between the use of computers as a research tool, say in developing diagnostic techniques, and its other use as a diagnostic tool. Please comment.

Mr. Crumb: I understand the distinction and also understand the probable origin of the question. I did not bring up the subject because I don't know enough about it. There are several related projects in some of which the computers are used as tools, and some are pure research. The particular project we are working on is toward use of the computer as a diagnostic tool; I think this would be called a non-research application. However, the aim at present is the technique itself; we have not utilized any large scale machine.

R. Tevonian (Western Electric): How does the likelihood of an incorrect diagnosis by the computer in each case compare with the likelihood of error by a doctor in the same case?

Mr. Crumb: In order to answer that I would have to know what the doctor's performance on diagnosis is. I don't know that. I don't know the status of statistics on incorrect diagnosis by doctors. When I talk to doctors considered objective, they admit that there is lots of room for improvement.

C. Kagan (Western Electric): Did you have anything to do with the three-year-old, excellent TV science fiction show "Dr. Robot," which presented the story of a medical diagnostic computer, incidentally a Bendix G15?

Mr. Crumb: No, I regret to say I had nothing to do with that. I am glad to see those things. There are lots of people working on the job, and I had no part in that.

T. E. Digan (IBM): Do plans exist for recording many case histories in punched card form to improve statistical data for computer usage?

Mr. Crumb: There are several projects dealing with machine handling of medical data per se for more than one application of the data. These are projects that have more general objectives than the one we are working on. This means the answer is "yes." There are in various places jobs going on to improve the character and availability of statistical medical data by machine handling.

J. Rothstein (Edgerton, Germeshausen & Grier): Have you considered that logical constraints — for example, the necessity or impossibility of associating some symptoms with certain diseases — might occasionally yield sharper discriminations than the lumping of probabilities which might blur them?

Mr. Crumb: Yes, we have. We have not tested any of these procedures, but we do have in preliminary form plans for things like that. We consider that they will be modifications, and we will test them to try to increase the computer technique effectiveness. We very fortunately have had comments and suggestions made by some very expert people to improve the discriminatory ability of the technique.

Dr. B. R. Hutcheson (Dept. Mental Health): Have you determined that these physicians can call the same symptom by the same name; for example, rate symptoms?

Mr. Crumb: Well, we have determined that making data uniform is very greatly aided by providing the framework for case history reporting which is uniform in detail and would have to be done for such a project as this. Fortunately there have been efforts in previous years without this kind of an objective but with the objective of attempting to uniformize reports of what they call case histories for medical records. It has been demonstrated that this can make the histories much more utilizable by machine methods.

T. J. Kobb (IBM): The Soviet doctors claim a great success in employing computers for medical diagnosis. Are you familiar with their techniques and results?

Mr. Crumb: No, all I know is what I saw in the newspapers. I have not observed any source of detailed information.

R. S. Ledley (Na't. Bureau of Standards): How do you take into account the important seasonal, social, local, epidemiological, and so forth effects on making a diagnosis?

Mr. Crumb: This technique does not yet do that at all. I recognize that other projects, including Dr. Ledley's, are aimed at actually accounting for those influences on the parameters used in preparation of the input to the machine. This method as yet does not do that.

F. Reilly (Veterans Administration): Do you foresee a national standard of correlation constants being established? If so, by whom?

Mr. Crumb: I would expect that would be the logical result if this technique were found to be the best one of the various techniques that are now being worked on.

M. Sendrow (RCA): How much of an increase in computing time do you estimate if the number of symptoms and diseases is increased greatly; for instance, to 200 symptoms and 2000 diseases?

Mr. Crumb: We visualize no such table of correlation constants, because we feel certain that diseases will be treated in logical groups which are much smaller than 2000. Even though there may be many more ailments to which humans are subject, the method will deal only with difficult diagnostic problems for which data actually exists to compute dependable correlation constants. In general, we expect the groups to be fairly small in number of disease per group and the number of symptoms to be typified by the slide I showed with 47.

E. H. Cabaniss (GE): Have you considered learning procedures for optimizing your *C*'s automatically? That is increase those that help and reduce those that hinder each diagnosis?

Mr. Crumb: We have condidered that, in the ideal application of the method, each case, when its true diagnosis was verified, would be immediately added to the data of the table of correlation constants on which its diagnosis was based; but this is an ideal which I think would be only approximated for some time after the introduction of the procedure.

An Advanced Magnetic Tape System for Data Processing

DR. RICHARD B. LAWRANCE†

Introduction and Tape Mechanism

IT IS a truism that for any but the smallest digital data processing systems major attention must be be given to the provision of an adequate magnetic tape transport, reading and writing system, and means for insuring the correctness of information all the way from the central processor to the magnetic tape and back again. This paper will describe some of the above mentioned features of the Honeywell 800[1].

Early in the layout and specification of a new system it is necessary to decide on the specifications for and approach to the magnetic tape mechanism and recording system. As regards the tape mechanism itself, our earlier experience (particularly with the DATAmatic 1000) had favorably inclined us toward the vacuum capstan approach. Our several years of experience with electrostatic clutching had led us ultimately to abandon the electrostatic approach for the DATAmatic 1000, and after re-evaluation, it was again excluded from consideration for the new system. As to the other two widely-used methods of achieving fast stop-start tape motion, we felt that the faster and more positive of these — namely the pinch-roller approach — should be the most seriously considered as an alternative to the use of vacuum capstans.

In this comparative evaluation and in the design effort which followed, we placed an overriding importance on providing the magnetic tape itself with a benign environment. This is in accord with our belief that in every stage of manual handling or manipulation by the mechanism all stresses in the tape (both during normal operation and under failure conditions) should be made zero by design or kept to demonstrably safe values.

The following tabulation compares inherent features of presently used pinch-roller mechanisms with the corresponding features of pneumatic mechanisms.

Pinch Roller

(1) In some designs (but not those in which the idler is continuously driven) the tape to be accelerated must bear not only the forces to accelerate its own mass but also the forces to give angular acceleration to the idler.

(2) The tape accelerating forces are applied in a concentrated area surrounding the line of tangency of two typically rather small cylinders (slightly spread by resiliency in the tape itself and at most one of the cylinders).

(3) In order to prevent non-simultaneous clutching across the width of the tape (with attendant tracking and skew problems) a very accurate pivoting or translatory motion is required; fast operation demands that this be designed for minimum inertia. Thickness variation across the tape is a possible source of skew.

(4) Compressive action of pinch-roller tends to emboss wear particles or other dirt into the oxide surface of the tape. (Not applicable to metal tape.)

(5) Powerful fast-pickup driving and braking mechanisms may be slow to release, placing safety restriction on minimum interval between drive and brake commands.

(6) For high performance, auxiliary air lubrication may be required.[2]

(7) Usual embodiment employs rollers of flanged-spool construction, with tape unsupported and not otherwise edge-guided over important portions of its path.

Vacuum Capstan

(1) Only the mass of the tape itself requires acceleration, thus minimizing the force transmitted to and by the tape.

(2) The tape accelerating forces are distributed over a typically fifteen-fold larger area, whose length may equal or exceed one-fourth of the capstan circumference. The capstan diameter may be conveniently large.

(3) Symmetrical engagement of the tape to capstan or brake is automatically achieved by symmetrical design of pneumatic passages. Engagement always commences along tape center line, minimizing skew. Transverse variation in tape thickness does not add to skew.

† DATAmatic Division, Minneapolis-Honeywell Regulator Co., Newton, Massachusetts.

[1] Together with the paper "Control & Arithmetic Techniques in a Multi-Programmed Computer." By: N. Lourie, H. Schrimpf, R. Reach, W. Kahn, presented at this conference, the present paper forms a partial technical description of this new data processing system.

[2] R. A. Skov, "Pulse Time Displacement in High-Density Magnetic Tape." *IBM Journal of Research and Development*, April 1958.

(4) Free from dirt embossing. No material body need touch the oxide surface of the tape (although usually the magnetic head is made to do so).

(5) No restriction on interval between successive commands. Moving parts of mechanism are offset from tape path, completely covered, and cannot touch tape. No danger to tape from tug-of-war.

(6) Air lubrication of tape is a built-in feature.

(7) Essentially complete edge guiding over entire path from supply reel to takeup reel is easily incorporated.

The considerations tabulated above led us to design for the Honeywell 800 a tape mechanism utilizing the vacuum capstan principle and embodying many of the techniques and principles used in the earlier DATAmatic 1000 three-inch tape mechanism.[3]

Brief Description of Tape Mechanism

Fig. 1 shows a photograph of the Type 804 magnetic tape mechanism. The unit stands approximately 5 feet 9 inches high and occupies a floor area slightly over 2 feet square. The tape is a nominal $\frac{3}{4}$-inch wide and is moved at a normal speed of 120 inches per second in either direction as desired. High speed rewind is provided, in one direction only, at 360 inches per second.

The cabinet shown includes the separate write-amplifier final stages for the ten recording channels as well as the final stage for the AC-excited separate erase gap; the three-stage transistor preamplifiers for each of the ten playback channels; and the solid-state switching equipment for placing the read-write head and circuits in the selected mode. Also included are the power supplies, loop position sensors and servo control for the DC-operated reel motors; the power supplies for the read-write circuits; vacuum and pressure sources for the capstan, brake, and suction loop chambers; beginning-of-tape and end-of-tape sensing means; storage; and other electronic packages facilitating testing and maintenance.

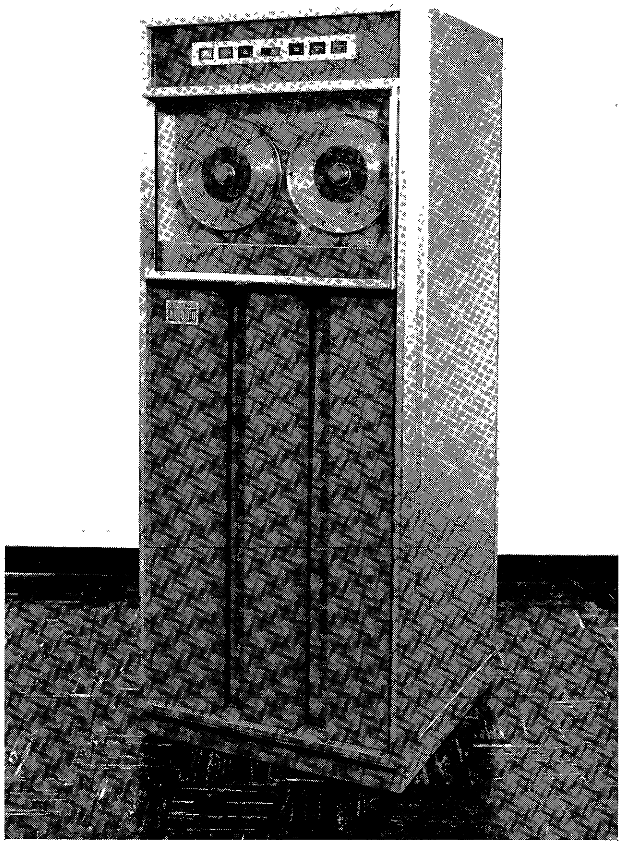


Fig. 1—Type 804 magnetic tape mechanisms.

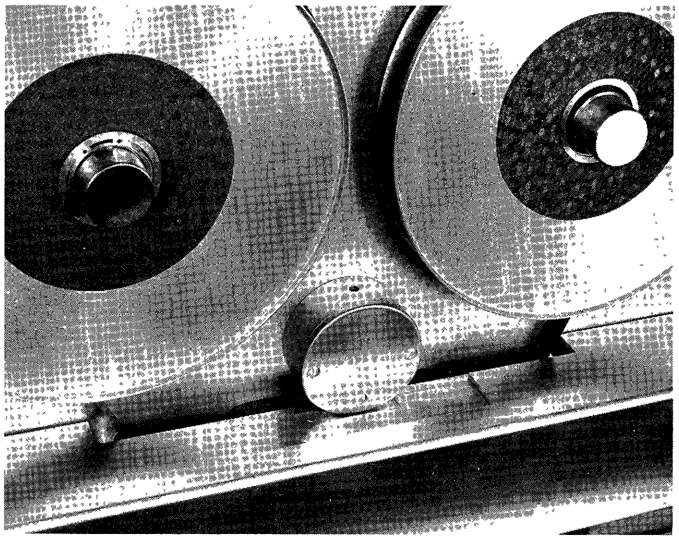


Fig. 2—Head area detail, in information transfer position.

Fig. 2 shows a closeup of the capstan area as it appears when tape is in position for information transfer. The centrally-located three-inch diameter billet contains the magnetic head assembly, and the oxide surface of the tape is uppermost. The tape lies horizontally, immediately over the two-piece horizontal vacuum brake, and thence executes a 90-degree downward turn at each vacuum capstan before dropping directly to the pneumatic loop chambers. Each capstan, when not actively engaged in driving tape, is provided with continuous air lubrication of approximately 2 psig, which effectively prevents all contact between the capstan and the tape. Unbroken edge guiding is present in the vicinity of the capstans, brake, and head, and indeed is present all the way from one reel to the other except in a space of $1\frac{3}{4}$-inches immediately next to each reel. Even in these regions back edge guiding is present, the deliberate absence of front edge guiding being in the interest of eliminating possible finger-catching accidents. It has

[3] R. B. Lawrance, R. E. Wilkins, R. A. Pendleton, "Apparatus for Magnetic Storage on Three-inch Wide Tapes." *Proceedings of the Eastern Joint Computer Conference*, 1956. Special publication T-92.

been our experience with magnetic tapes (as with other elongated flexible substrates) that continuous edge guiding is far more advantageous than guiding by periodically-spaced flanged spools, provided only that the tape be slit accurately enough. For nearly two years we have made complete and detailed observations of commercially produced magnetic tape with respect to width and the periodic curvature usually called snakiness. We can state that the snakiness can reasonably be reduced to complete insignificance while the width of slitting is held well within a total range of .002 inch.

We feel that these edge-guiding arrangements, together with accurate tape width control, yield considerable benefit in drastically reducing tracking and skew errors within the mechanism, as well as contributing to long tape life since the edges of the tape are nowhere subjected to localized sideways forces. Spring-loaded parts for exerting side-thrust on the tape are themselves subject to excessive wear, so their elimination enhances reliability.

Returning to Fig. 2 and comparing it with Fig. 1, we note that in normal operation, with the tape in the loop chambers and the head in position, the oxide surface is in rubbing or pressure contact with no parts of the mechanism except the magnetic head, bringing tape wear to a practical minimum. Fig. 3 shows the capstan and head area with the head eccentrically rotated, removing the head from contact with the tape oxide surface. Thus in high speed rewind (at whose beginning the head rotates away automatically) not even the head touches the oxide. Rotation of the head, automatically controlled, is also used during tape changing, at which time it enables the tape to slip easily over what is otherwise an unbroken edge guide.

In Fig. 3 the magnetic portion of the tape is all on the left hand supply reel and the leader of heavier-gauge clear Mylar® (permanently attached to the tape) is lying over the capstan and brake. This enables the nature of the exterior pneumatic passages of the capstan and brake to be seen. The two capstans are continuously rotated in opposite directions by individual 1200 rpm hysteresis synchronous motors, the capstan circumference being exactly 6 inches. The left and right portions of the brake (lying between the normal head location and the two capstans) are internally connected to a common working air passage which is supplied appropriately with medium suction, strong suction, or air at atmospheric pressure.

Fig. 3 also shows that no pressure pad is employed to keep the tape in contact with the magnetic head assembly. Wrapping contact between tape and head is adequately maintained by having the head press the tape down into a short and very shallow "V", the outer edges being defined by the rounded shoulders of the brake, closely adjacent. By means of this wrap, with its elimination of pressure pads, and by means of the unconventionally large radius of curvature of the magnetic head (both essentially the same in dimensions as in the DATAmatic 1000) we achieve good transient and running contact between tape and head, together with a gratifyingly low rate of head wear. Measurements carried on over more than a year's two-shift operation of a DATAmatic 1000 show for all channels of all magnetic heads a quite uniform and unexpectedly low rate of wear. The average yearly loss of material from the head under these conditions amounted to 0.0001 inch.

By implication, the tape wear produced by friction between head and tape is correspondingly small.

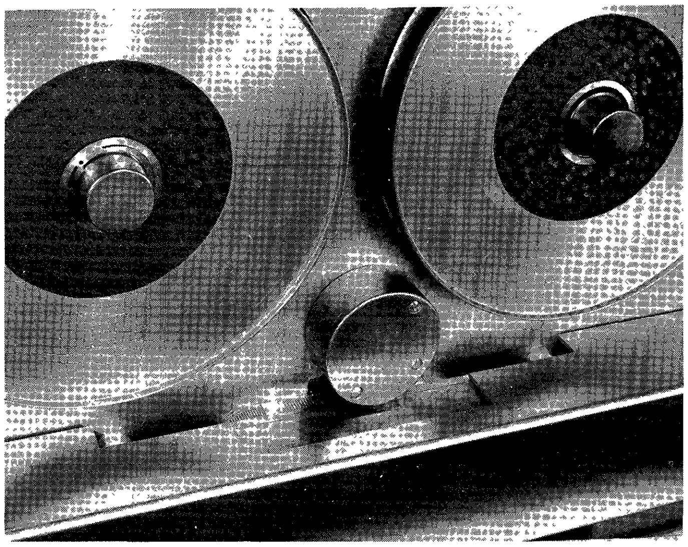


Fig. 3—Head area detail, with head pivoted for tape change.

Rewind and Tape Change

The central processor instructions to which the tape drive responds are Write (forward), Read Forward, Read Reverse, and Rewind. The Tape Change operation is initiated by manipulating a lever switch on the tape mechanism itself.

The position shown in Fig. 3 occurs at the termination of a tape change operation, which starts with a high speed rewind unless the tape is already rewound. Every rewind command is executed by the mechanism as a high speed rewind, and once received from the tape control unit the rewind is performed under local control until completed. During high speed rewind the tape speed is controlled by the left-hand vacuum capstan, whose motor speed is increased automatically to 3600 rpm. The tape remains in both vacuum loop chambers and accordingly receives the benefit of controlled tension and complete edge guiding. Upon rewinding past the designated beginning of tape, as sensed by a photoelectric arrangement described later, the mechanism shifts down from 360 ips to 120 ips. This latter speed endures for a fraction of a second and the tape is then stopped in normal fashion by pneumatic disengagement from the capstan and engagement to the brake. The head, which has been automatically moved out of contact with the tape during the rewind, now rotates back into contact with the tape, and the closing of a Microswitch® signals the computer that the rewind has been completed and that the tape mechanism is again ready for instructions. At this time the magnetic head is positioned part way down the clear leader and has access to the first magnetic information location by moving the tape in the forward direction (to the right).

If it is desired to change tape, a centrally located manual switch on the control panel is thrown to the tape change position. It is irrelevant whether the tape is already rewound or not, although some head rotating operations are bypassed if the rewind is continuous with the tape change. The tape proceeds to the left, along the clear leader and at 120 ips, until a single short centrally placed slot in the leader is sensed by an orifice and vacuum switch associated with the upper end of the right hand loop chamber. When sensed this causes the tape to stop with only two or three turns remaining to be manually unwound from the right-hand reel, and the appropriate partial shutdown of the mechanism is initiated so that the reel is ready for removal.

Time taken for a rewind operation can be characterized by the equations

$$t_{\text{rewind}} \leq \frac{t_{\text{wind}}}{3} + 2.6$$

$$\text{or } t_{\text{rewind}} \leq \frac{\text{distance in ft.}}{30} + 2.6$$

in which all times are given in seconds.

Tape Reels and Mounting

Data processing magnetic tape mechanisms do not as a rule use standard reels, and the present equipment is no exception. Since a partial vacuum (about one half atmosphere absolute) is provided within the equipment for use in the clutch, it is quite natural to use this vacuum for holding the reels onto the reel mounts. This technique has already proved highly satisfactory with the three-inch-wide 23-pound reels of the DATAmatic 1000. Advantageous features include the lack of metal-to-metal contact between reel and reel mount, the fact that the reel hub is not subjected to hoop stress, and the provision of a large flat reference surface on the reel mount, which insures wobble-free rotation and accurate positioning of the reel relative to the back reference surface.

Suction is similarly used for attaching the free end of the tape leader to the right hand reel whenever a tape is loaded on the machine, as well as for initially attaching the inner end of the tape to the left-hand supply reel. It is worth mentioning that vacuum attachment of the tape to the reel makes it unnecessary to perforate the reel flanges for finger access to the hub during loading. The unperforated flanges are helpful in protecting tape from dirt and mechanical damage while in storage or during handling. Hazard to the operator is also reduced materially.

The design of the reel and reel mount involved additional factors, however. It was desired to make use of a demountable ring, capable of being stored with the reel of tape and serving by its presence or absence to enable or inhibit the recording of information on the tape. (This is in addition to a manual switch on the operator's panel.) Various embodiments of this principle have been used for several years by other manufacturers, but we believe our version has some useful and novel advantages. One desirable feature present in our arrangement is that the physical presence or absence of the write-enable ring does not need to be inferred from the status of a concealed electrical switch (which requires that electrical power be applied and that the circuit be functioning with some means of indicating its status). We have placed the write-enable ring in plain view on the front of the reel. It thus becomes easy to remove or insert the ring while the reel is in place on the mechanism, without the necessity for first rewinding the tape in order to remove the reel.

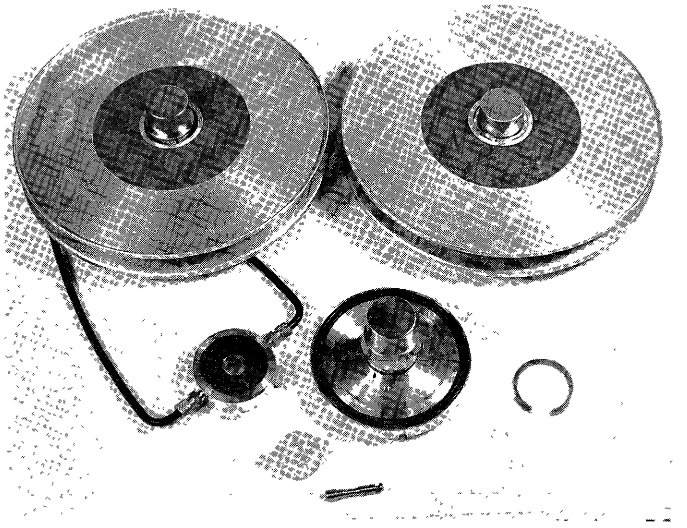


Fig. 4—Reels, reel mounts, and write-enable ring.

Fig. 4 shows a photograph of three reel mounts, one having a write-enable reel of tape mounted on it and another carrying a write-inhibit reel. The removable snap ring which converts a write-inhibit reel to a write-enable reel is shown beside the third reel mount. The principal working part of the reel mount subassembly is the central bell-shaped cylinder. Its axial motion controls three retractable nylon latches, spring-loaded radially outward, and also an internal piston, spring-loaded axially outward. To remove a tape reel from the mount the reel flanges are lightly grasped by the fingers, while the thumbs press the central cylinder so that it moves axially inward. The three nylon latches are thus moved radially inward to the point where the reel can slide over them and be removed. In putting a reel on the mechanism, the central cylindrical bore of the reel performs a similar operation in reverse — as the reel is moved inward it presses on the nylon latches, retracting them. A small fraction of an inch before the reel is fully seated the latches snap outward and thus hold the reel in place even with no power or no vacuum. When vacuum is applied (automatically, as part of the normal cycle-up procedure) the reel is drawn into intimate sealing engagement with the rubber driving rings and is fully positioned ready for operation.

The write-enable ring operates by capturing the outer rim of the central cylinder, as the reel is pressed on. By this means the central cylinder is moved inward about ¼-inch as the reel is seated home. The internal piston-and-cylinder arrangement is thereby vented to atmospheric pressure rather than being connected to the half-atmosphere suction reservoir. The electrical image of these two pressure states is created in a stationary vacuum-diaphragm-operated Microswitch® located at the rear of the main mounting plate and sampling the pressure in the reel mount cylinder via a carbon rotary seal.

The Vacuum Clutch

Figs. 2 and 3 showed portions of the vacuum capstan and brake, and their relationship to the magnetic head as mounted in the mechanism. Fig. 5 shows an exploded view of these components of the pneumatic clutch, viewed from the side rear. Of the components all but the capstan motor are mounted to the front of the heavy flat vertical plate which serves as structural support and back edge guide, and which is omitted from the photograph. The capstans, of which only one is shown in exploded position, are directly mounted to the shafts of their respective hysteresis synchronous motors. Precision bearings are used in the motors, and capstan runout and taper are held to tight tolerances in order to achieve good tape tracking.

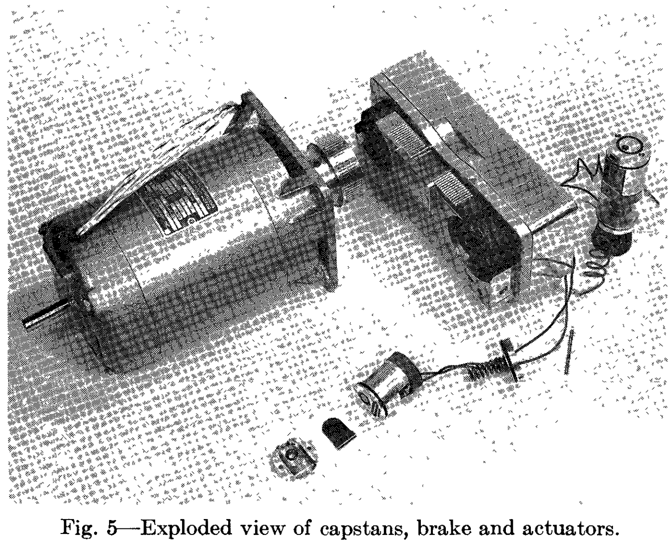
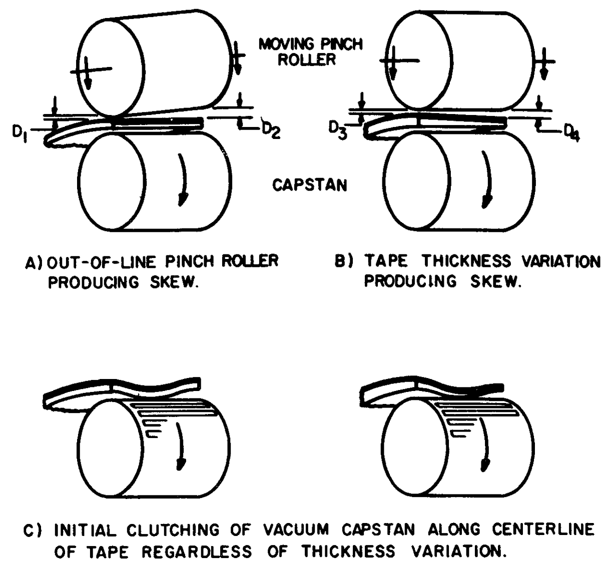


Fig. 5—Exploded view of capstans, brake and actuators.

A 90-degree segment of each capstan is connected via the working air passage to the electropneumatic valve, mounted nearby in the actuator housing body. (As shown in Fig. 5, this is bolted directly to the capstan housing body.) The fixed portion of each pneumatic commutator consists of a carbon composition cylinder which fits closely without rubbing inside the cuplike capstan. The portion of the working air passage within each carbon piece consists of a single slot centrally spanning the active arc of the capstan, and a drilled hole connecting to the actuator.


MOVING PINCH ROLLER
D1
D2
D3
D4
CAPSTAN
A) OUT-OF-LINE PINCH ROLLER PRODUCING SKEW.
B) TAPE THICKNESS VARIATION PRODUCING SKEW.
C) INITIAL CLUTCHING OF VACUUM CAPSTAN ALONG CENTERLINE OF TAPE REGARDLESS OF THICKNESS VARIATION.
Fig. 6—Skew considerations for pinch rollers and vacuum capstans.

The location of this slot along the center line of the tape track, together with the pneumatically symmetrical design of the capstan itself, leads to what we believe is an important advantage for the vacuum clutching technique. Fig. 6 shows a series of sketches representing the clutching action and the production of skew in pinch roller and vacuum clutches respectively. Part A shows (greatly exaggerated) the engagement of a tape to a capstan when the moving pinch roller is slightly out-of-line so that distances D1 and D2 are unequal. While we have no quantitative measurements available it is not too difficult to imagine that an inequality of perhaps 0.0001 inch will result in significant time difference in the engatement of the two tape edges, to the capstan. Parallelogram distortion of the tape would then produce skew. Similarly, as shown in Sketch B, it would appear to be possible for skew to be produced even if the moving pinch roller were to be perfectly aligned with the capstan. Any thickness taper across the width of the tape will produce the same effect as an out-of-line pinch roller. Again we have no quantitative data to support this conjecture but the point-to-point thickness tolerances to which tape backing is produced are large enough so that the possibilities of skew production from this cause should not be overlooked. The unsettling thing is that since tape is not customarily inspected for thickness uniformity it appears possible for portions of an otherwise perfect tape to produce random skew when used with a clutch of the pinch-roller type.

The situation is different with a vacuum capstan, however, as shown by experiment. A priori expectations (verified in detail by observations using a time-delayed stroboscopic flash) are that since the working air passages communicate to the underside of the tape symmetrically about the tape center line it should be the case that the center of the tape always engages first. Thereafter the region of engagement spreads symmetrically to the edges. Prior to the evacuation of the working air passage it and the underside of the tape have been supplied with air lubrication at slightly above atmospheric pressure; thus at the start of a clutching operation the underside of the tape is at rather definite and reproducible location with respect to the capstan surface. As shown in the fourth sketch of Fig. 6 it is thus to be expected that, to first order at least, any variation of tape thickness across the web will not significantly affect the symmetrical tape engagement.

Returning to Fig. 5, it can be seen that the valve actuators each contain a small but efficient electromagnet which is energized when its associated capstan is intended to drive tape. The highly effective eddy-current shielding of the aluminum actuator housing body prevents any external magnetic influence on the tape or in the head.

Fig. 7 shows a sketch of one of the electropneumatic valves, each of which is essentially a pneumatic SPDT switch. With no current in the coil the flat armature, resiliently pivoted near the right-hand end, will seal off the upper valve seat, being maintained in position by pivot bias and by air pressure differential. During this time the working air passage to the capstan is supplied with lubricating air from the pressure reservoir, at approximately 2 psig. When current is passed through the magnet winding the valve assumes the position of Fig. 7, with the armature magnetically drawn down to seal off the compressed air from the working air passage. The capstan and working air passage thus exhaust into the reservoir at half-atmosphere vacuum.

The armature is tapered slightly, as shown, to reduce inertia and speed up pull-in. Life tests on a group of similar armatures and magnets, driven at 120 operations per second for a period of over 20 months showed no measureable change in performance after 6.2 x 10^9 operations.

The transistor circuits which drive the actuators supply an initial high-current pulse for fast armature pull-in, dropping to a reduced holding current which lasts until the stop command is received. The ferromagnetic material of the magnet is an alloy with relatively low saturation flux density so that drop-out time is held to a minimum. The transistor circuits are interconnected in such a way that engagement of the tape to both capstans simultaneously is most unlikely, even under failure conditions; even if this should occur, however, the tape suffers no damage since the capstan motors will stall without the tensile elastic limit of the tape having been exceeded.

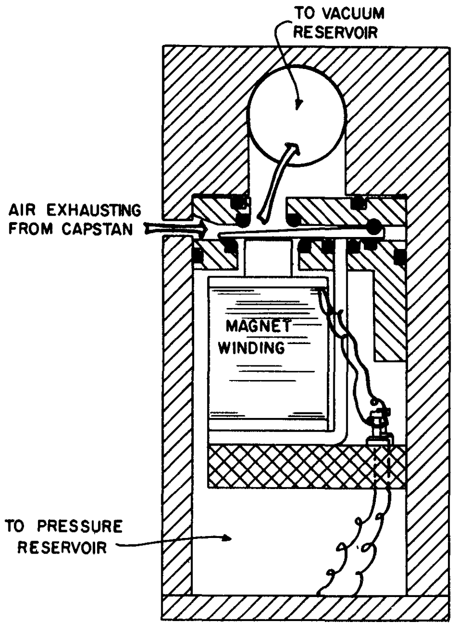


Fig. 7—Schematic of electro-pneumatic valve.

Typical curves of tape velocity versus time in starting and stopping are shown in Figs. 8 and 9. These curves were taken by recording on the tape a train of 64 pulses derived from a 5 kc keyed oscillator, turned on at the time of the stop or start command. Magnetic development with colloidal Fe_3O_4 and position measurement with a microscope and traveling micrometer table were then used to give an accurate history of tape position and velocity, relative to the read-write gap, versus time.

Fig. 8 shows that in response to a start command the tape commences to move at slightly less than one millisecond; at 2.7 milliseconds the tape has traveled 0.12 inch and is traveling at 120 inches per second. Speed fluctuations thereafter do not exceed approximately 3 or 4 percent, although the read system will tolerate many times this amount. Fig. 9 shows that in stopping, the initial deceleration occurs after about 1.2 milliseconds and that the total distance to come to rest is substantially less than 0.3 inches. As mentioned briefly earlier the 804 mechanism allows a start command to follow a stop command arbitrarily closely. The present curves indicate why the tape continues at full speed when the interval between commands does not exceed approximately

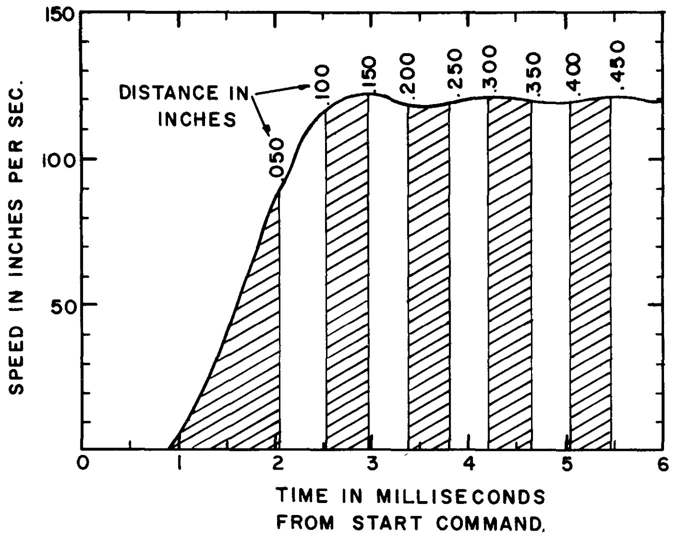


Fig. 8—Typical curve of velocity and distance *vs.* time during tape acceleration.

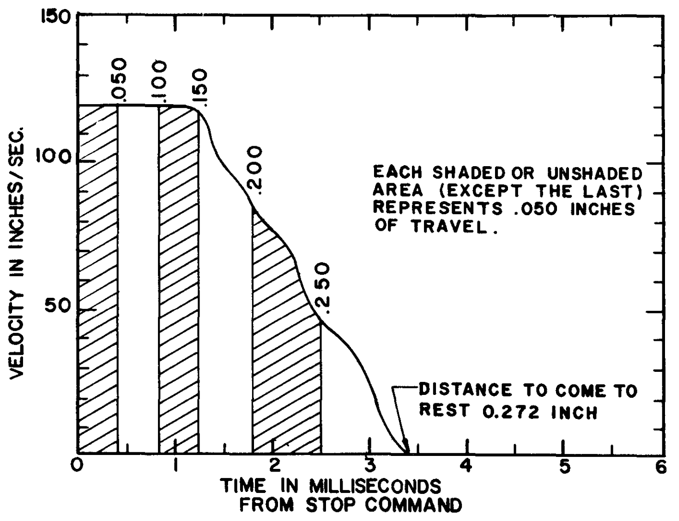


Fig. 9—Typical curve of velocity and distance *vs.* time during tape deceleration.

0.7 milliseconds; for longer intervals there is a smooth transition to the isolated-stop, isolated-start condition shown in the graphs.

Sensing of Beginning and End of Tape

In the interest of brevity we will not give a complete description of the circuit arrangements for keeping track of the position of the tape in the machine: that is, whether the magnetic head is positioned over the permanently-attached clear leader, initial information space, mid-tape information space, or one of the recognizable end-tape zones. With one exception, the task of remembering tape positions on all of the eight connected tape mechanisms is assigned to their common tape control unit. The exception is concerned with rewind operations, in which the controlling elements are completely local to the respective tape drives: a relay picks up at the start of the rewind and only releases upon sensing clear leader at the completion of the rewind.

The boundaries of all logically distinct tape regions are marked off by small windows at the front edge of the tape, created by removing oxide for a distance of 0.1 inch along the tape and .035 inches in from the edge. Since the nearest recording channel ends 0.041 inches from the edge there is no conflict between optical sensing and magnetic recording. It is possible to sense the passage of a window without interfering in any way with the execution of any write or read instruction which may be in process. Significant program advantages and time savings result from this feature.

The special illuminator contains a miniature long-life tungsten filament bulb and a one-piece optical element consisting of a lens, cylindrical barrel, and angular refracting surface. This illuminator is positioned at a fixed distance from the magnetic head near the upper end of the right-hand loop chamber, with the optical element extending at an angle through the loop chamber outer wall to a position nearly flush with the inner surface. By this means, since the angle of the refracting surface is nearly the angle for grazing refraction, a satisfactory intense light source is effectively positioned directly opposite the outer edge of the tape, yet without mechanical projection into the path of the tape.

Upon passage of one of the windows, light falls on a miniature silicon photodiode (part of the sub-assembly) which issues the window-recognition signal for interpretation and storage.

Read-Write System

In the Honeywell 800, as in nearly all other systems, the tapes are written in the forward direction only, *i.e.* with the tape moving to the right. Reading takes place in either direction as desired, and uses the same head gaps as for writing. Ten channels are used, of which eight are information channels, one is an Orthotronic parity channel, and the tenth is a clock. A separate full-tape-width erase gap, located a fraction of an inch upstream of the read-write gaps, applies AC erase to the tape at the time of recording. The read-write gaps are in-line across the tape and are spaced on 0.070 inch centers.

The AC erase serves the primary function of cleaning out the inter-record gaps and leaving the tape magnetically neutral, which facilitates record-entry recognition in bidirectional readback. NRZ1 recording (saturation-to-saturation, flux change denotes a "one") is used on the information and parity channels. The Honeywell 800 word contains 48 bits (not counting the parity bits which accompany the information on tape and in memory) so that a word occupies six frames on tape, a frame being defined as the time-simultaneous record of a bit in each information channel. The parity bit is also recorded simultaneously with the eight information bits. The frame interval is 21 microseconds, corresponding to a frequency of 47,619 frames per second and a bit density (at 120 inches per second) of 397 per inch.

The clock channel is similarly recorded from saturation to saturation, but undergoes one flux reversal per frame. The recording of the clock is not simultaneous with the recording of the other bits of the frame but is offset by one half of the frame interval. By this means the read circuit is made self-timing, highly tolerant of speed variation in the tape mechanism, and free from one-shot circuits with their jitter and delay tolerance accumulations.

As soon as a write instruction is received the erase head is excited and remains so, independent of tape motion, until receipt of the next instruction of a different type (read, rewind, tape change). At the beginning of a record to be written, with the tape in motion and the inter-record gap just traversed, write current is initiated in all ten channels in the same standard polarity. This results in half-strength magnetic poles of known polarity being written in all channels (automatically ignored in subsequent playback). Thereafter the clock begins its 21-microsecond beat and 10.5 microseconds after the first clock beat the first frame is recorded, with flux reversals in those channels where ones are to be written. Writing continues, at six frames per word, until all words of the record have been recorded. Before cessation of writing two orthotronic words (twelve frames) and an end-of-record word are appended, after which one more clock pulse is written and all write currents drop to zero.

The construction of the orthotronic words is on a per-channel basis, roughly as follows: half the first, thirteenth, twenty-fifth . . . bits are added modulo 2 and the first bit of the orthotronic word is the complement of their sum. Similarly the second orthotronic bit is formed from the second, fourteenth . . . bits of the record, etc. The result is a very powerful check having the following properties:

1. Garbled information confined to a single channel can be recreated regardless of the length of the difficulty.
2. Garbled information extending up to twelve bits in length can be reconstructed regardless of the number of channels affected.

In playback the ten channels are connected, by means of solid-state switching, to ten individual preamplifiers located at the tape mechanism and thence are passed via the tenfold read bus to the Type 803 Tape Control Unit. Further shaping culminates in peak detection of each signal and the production of a one-half microsecond pulse essentially coincident with each flux change in the channel. These pulses set nine individual high-speed flip-flops, which accumulate the bits of the frame; the next peak-detected clock pulse (half a period later) resets all flip-flops and sends the bits into buffer storage where they reside until a complete word is available for transmission to memory.

Fig. 10 shows the appearance of playback from a single channel, and has the typical NRZ1 waveform. Because of the conservative bit density satisfactory resolution is achieved with a comparatively large head gap, minimizing signal fluctuations due to the passage of lint or other debris between the head and the tape. Fig. 11 shows the effect of a recording dropout deliberately produced by blowing fibers of cotton lint into the region between the magnetic head and the tape being written. The amplitude decrease shown is typical and produces no error in reading, as shown by the associated peak detector waveform. The read system is designed to tolerate signal decrease to well below one-fourth of normal amplitude. It is well to mention, also, that the tape mechanism incorporates the conventional positive pressurization of the region occupied by reels, capstans, head, and loop chamber entrances, thus excluding airborne dust except during necessary tape changing.

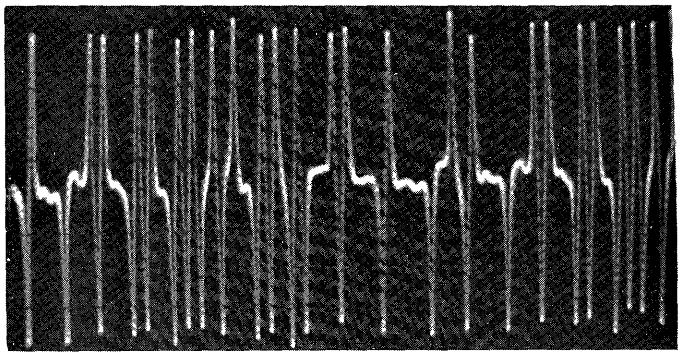


Fig. 10—Normal preamplifier output of read system.

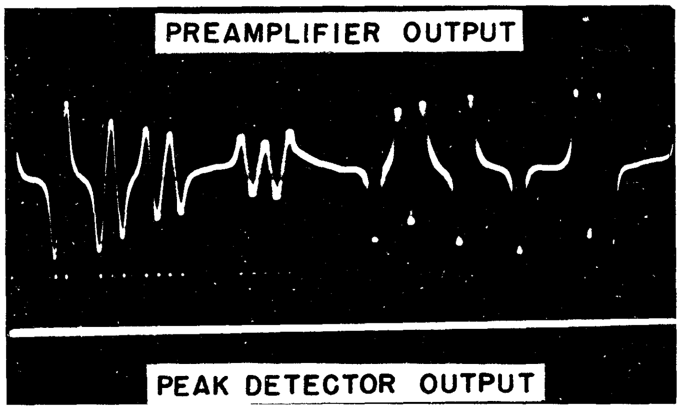


Fig. 11—Read system waveforms, including deliberate recorded dropout due to loose cotton lint.

We have not dealt at length with the internal checking of the Honeywell 800 but it is well to mention, in conclusion, two of these features associated with the magnetic tape system. Writing cannot occur (and its absence is made known) unless an enabling check shows that the erase head and the clock channel are both excited. The transmission of data to the tape drive is checked for transverse parity at each frame and for longitudinal parity on each channel of each record.

The net result of the features described in this paper is a strong, efficient, and trouble-free tape system. The approach deliberately taken has been to design high reliability into all electrical and mechanical components, effecting error detection and correction by means of the powerful capabilities of Orthotronic control.

Acknowledgments

The read-write circuit and system development has been done by a group directed by Dr. Way Dong Woo. I am sure that the many other contributors to the design of the tape drive itself will not object to my singling out Mr. Robert A. Pendleton as major contributor.

Bibliography

[1] N. Lourie, H. Schrimpf, R. Reach, W. Kahn, "Control & Arithmetic Techniques in a Multi-Programmed Computer." Eastern Joint Computer Conference, 1959.

[2] R. B. Lawrance, R. E. Wilkins, R. A. Pendleton, "Apparatus for Magnetic Storage on Three-Inch Wide Tapes." *Proceedings of the Eastern Joint Computer Conference, 1956.* Special Publication T-92.

[3] R. A. Skov, "Pulse Time Displacement in High-Density Magnetic Tape." *IBM Journal,* April 1958.

Discussion

W. N. Papian: I wonder if you could give us a recap of the key characteristics.

Dr. Lawrance: The number of channels is ten, of which eight are information, one is the Orthotronic parity channel, and one is the clock, recorded at the same time as the other nine channels. High-frequency AC erase occurs upstream of the read-write gap at the time of writing. Each channel is recorded at 397 bits per inch (nominal 400). The three-fourths-inch wide tape moves at 120 inches per second in either direction. Rewind speed is 360 inches per second, and every rewind command is executed as a high-speed rewind.

D. Allen (Ampex): What types of errors do you typically encounter and what is their frequency? Are they so numerous as to justify correction hardware? What causes them?

Dr. Lawrance: In the oral presentation I omitted to stress sufficiently one unique feature of our mechanism. The oxide surface of the tape is in contact only with the magnetic head and does not rub or abrade against any other surfaces in the entire mechanism. We followed the philosophy that the less abuse to the oxide, the better. Familiarity with less gentle tape mechanisms may condition one to think in terms of frequent tape errors. By contrast we have had excellent tape reliability both during our extensive DATAmatic 1000 experience and in our experience to date with the Honeywell 800. Consequently, it is not correct to imply or infer that Orthotronic control is required for correction of an inordinate number of tape errors. In engineering fact, what Orthotronic control does is to take a tape system which, by all present standards, is excellent and extend its performance to still higher standards.

With regard to errors and their causes, it is extremely difficult to obtain much data without taking a long time to do so. Typically, in a fully operating computer with eight tape mechanisms, several days elapse between individual malfunctions which could possibly be tape errors. Some evidence supports our belief that airborne dirt (notably clothing lint) momentarily passing between tape and head is responsible for a significant part of these residual errors. Fig. 11 shows results obtained undeı these circumstances; the printed paper mentions the pressurizing used to avoid introduction of airborne dirt.

G. A. Barnard (Ampex): How many 19-inch racks, or equivalent worth, of electronic, mechanical, power supply, compressed air, and vacuum equipments are needed for the total tape system?

Dr. Lawrance: The unit shown in Fig. 1 contains everything mentioned in the question. Specifically, all actuator power supplies, reel control power supplies, compressed air and vacuum sources, erase electronics, and read-write electronics associated with the head are inside the unit. By the latter is meant the write amplifier final stages (each of ten channels), the read amplifiers (each of ten channels) and the power supplies for these circuits.

The single Type 803 Tape Control Unit, which is associated with up to eight drives, handles the data to be written (including checking) and contains the peak detecting and buffer storage for the read system. This unit, through which the eight tapes communicate with the central processor, is free standing and, with its power supplies, is almost exactly four feet wide.

D. E. Killen (Oliver Shepherd Industries): What is the maximum skew in microseconds?

Dr. Lawrance: We do not have final figures for publication yet, but I would estimate not much more than two microseconds.

H. S. Davin (Sylvania): Is there any prospect for increasing recording density on the H-800 Tape Drive?

Dr. Lawrance: We have chosen a conservative figure of 400 bits per inch (each channel) which gives the very high information rate of 96,000 decimal digits per second. If a special application required higher bit rates, I feel the density could safely be increased somewhat.

J. Hunter (Broadview Research): Are you using sandwich tape?

Dr. Lawrance: No. It is possible to do so, but our tape life and cleanliness lead us to feel that sandwich tape is not necessary.

R. Pacel (Rem. Rand): What is the tape-to-head separation?

Dr. Lawrance: The oxide surface of the tape is in contact with the head.

J. M. Kolb (Lincoln Labs.): Do you use separate Read-Write windings on heads? What happens if Start order is given before Stop is completed?

Dr. Lawrance: There is only one winding per channel, used for both reading and writing. The full-tape-width erase is applied from another gap located upstream of the read-write gap line.

With regard to the second part of the question, if a start order is given before the stop is completed, the start is executed without requiring any artificial delay. There is no restriction on the interval between commands, and the last one received prevails.

G. F. Roe (Hughes Aircraft): Do you use the "MARK" method of NRZ recording?

Dr. Lawrance: Every "one" bit is a current (and flux) reversal, putting a pole on the tape. Every "zero" bit is no change in current direction and no pole on the tape.

E. Seif (Burroughs Corp.): How much time is required to stop and then start in the opposite direction; *i.e.,* time elapsed from normal speed in one direction to normal speed in opposite direction?"

Dr. Lawrance: In reversing direction, the stop command must first be given; *i.e.* the first-existing command to drive (say forward) must be released. As far as concerns the mechanism and the tape, it is safe and permissible to issue the reverse drive command within microseconds after the termination of the forward drive command. In use with the computer, however, it is presently planned to incorporate a delay (approximately two milliseconds) so that the turnaround point is far enough down the tape to guarantee perfect reading of the just-traversed information in the second direction. To answer the question, then, the time from full speed to full opposite speed is approximately five milliseconds.

A High Speed, Small Size Magnetic Drum Memory Unit for Subminiature Digital Computers

M. MAY†, G. P. MILLER†, R. A. HOWARD† AND G. A. SHIFRIN†

A MEMORY with dimensions compatible with microminiature assemblies is required for future computers to be used in missiles and aircraft. A drum memory is described which can fulfill this need. The bit rate of 546 kc makes possible a 20-bit serial word time of the order of 40 microseconds. For a computer with add and multiply times of 40 microseconds, the drum memory described is adequate. Moreover, the technique described can be extended to provide a 20-microsecond word time by doubling the rotational speed of the drum, and to 10 microseconds or less by reading out two or more bits in parallel. A memory capacity of 15,000 twenty-bit words is available in the 7.4 × 3.7 × 3.7 inch total unit size, which is adequate for the type of computations usually made in an aircraft or missile. The advantage of such a drum memory as compared with a ferrite core memory, for example, is in cost, size, and ability to perform over wide temperature ranges. The disadvantage of the lack of immediate access to any address can for the most part be overcome by suitable programming precautions.

The magnetic drum development was performed under contract for Wright Air Development Center of the USAF to determine whether recording densities of 500 to 1000 bits to the inch and more than 30 tracks to the inch could be achieved in a small unit which would meet the requirements of MIL-E-5400, Class 2.

General Description of Memory Unit

Size	3.7 × 3.7 × 7.4 inches over-all
Power	400 cps, 3 phase, about 30 watts
Weight	11.3 pounds
Motor	Mounted inside recording drum
Tracks	30 per inch, total of 122 tracks
Recording density	350 bits per inch using Manchester phase modulated recording
Clock frequency	546 kc
Total storage capacity	300,000 bits plus timing tracks and spare tracks
Speed	12,000 rpm approximately

† Thompson Ramo Wooldridge Inc., Los Angeles 45, California.

Fig. 2 shows a partly assembled unit. To achieve the high degree of stability required for high density recording over a wide temperature and vibration range, an especially rigid unit was constructed. The framework and most critical parts are made of stainless steel selected to have a coefficient of expansion to match that of the ball bearings. A cross section drawing of the rotating part of the unit is shown in Fig. 1.

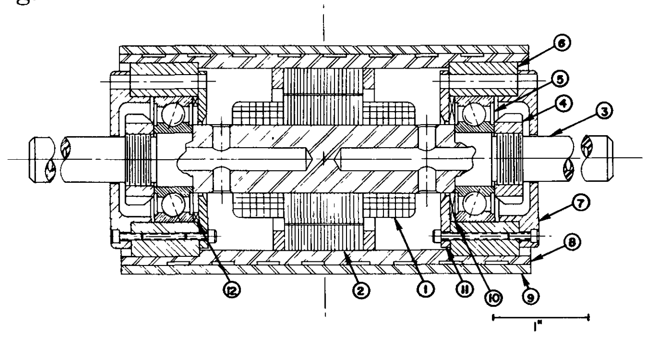


Fig. 1—Cross section of rotor assembly.

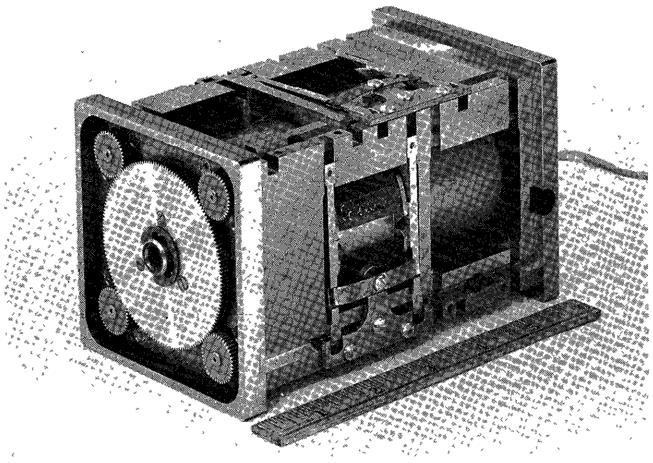


Fig. 2—Photograph of partly assembled magnetic drum.

The recording drum is made up of an internally mounted, 400-cps, 3-phase induction motor whose stator (1) is attached to a fixed shaft (3). The squirrel cage type rotor (2) is fixed inside a steel cylinder (8) which provides magnetic shielding and forms a mounting for a nonmagnetic stainless steel cylinder (9). This cylinder is plated with nickel-cobalt by an electroless method to form the recording surface. A shoe holding 27 read-record heads can be seen rest-ing on the recording surface in Fig. 2. This shoe is loaded with a 6–10 pound force against the recording surface when the drum attains full speed. Since the shoe and 27 heads weigh less than 1.5 ounces, accelerations of 10 g's have little effect on the head spacing (which is maintained by an air film between the shoe and the drum). The shoe is positioned radially by means of pivoted arms. The pivots are held in V-grooves to eliminate any possible play.

A gear wheel can be seen which turns cam shafts mounted down the length of the four corners of the framework. The cam shafts take the pressure off the shoes for starting or stopping. A very small motor (not shown) will be mounted to turn the gear wheel against a spring when the drum has attained full speed. Upon the removal of the driving power, the spring will turn the large gear wheel and take the load off the shoes.

The shoes are self-aligning and no adjustments other than spring pressure are required. The use of two independent arms loaded by a single cantilevered spring achieves this self-alignment.

Positions for four large-sized shoes are visible in Fig. 2. On the other faces of the frame similar mounting spaces for smaller shoes are provided. These shoes are intended to hold both read and record heads for circulating registers.

The electrical characteristics are summarized as follows:

Recording. Peak currents of 100 ma are required for recording. The current is built up linearly during half a bit time for the Manchester type recording. A silicon transistor push-pull circuit with 6 volts on the collectors is used for the recording amplifiers.

Reading. The read signal is about 10 mv peak-to-peak at 546 kc and about 30 mv at 273 kc. No noise is noticeable on the signal under test conditions. Using Manchester or variable phase type recording, no transients are apparent beyond one recorded bit before and after each word. Pattern sensitivity has been eliminated by the use of narrow pole piece heads described later. Typical read signals made up of single eight-bit words are shown in Fig. 3.

Selection of Magnetic Coating

For digital recording the head-to-drum spacing should be of the order of one-tenth or less of the length of the recorded bits to achieve customary margins of operations. For 350 or more bits per inch, a head-to-drum spacing of less than 300 microinches is indicated. For both temperature ranges of $-50°$ to $+125°$C and high shock and vibration, the small head-to-drum spacing required of 300 microinches cannot be maintained unless the head is made to bear on the recording surface. Gas lubrication is satisfactory for the maintenance of spacing in this range. For practical reasons it is desirable that a small particle of dirt (or accidental mistreatment during assembly or service) not do appreciable harm to the recording surface. This puts a requirement on the durability of the magnetic coating. For this reason magnetic plating is preferred to oxide films. There is an optimum plating thickness (which in practice turns out to be of the order of 100 microinches) for 350 bits per inch as is shown later. Since oxide coatings are usually ground after application, there would be an especially difficult problem in grinding them down to a uniform thickness of 100 microinches. Thus it became necessary to develop a suitable plating. Electroplated nickel-cobalt alloys have been tried and work perfectly well magnetically. They can be plated as thinly as desired and have been tested at thicknesses of 60 microinches and less. Mechanically this plating is not the best that can be obtained since it is not especially hard and has not been made to have both a high coercive force and adhesion strength comparable to the bulk material strength. This type of coating is magnetically satisfactory, but slight damage may put several 0.03 inch wide tracks out of operation due to local peeling of the coating. Nickel deposited by the Brenner electroless process forms a very hard coating which has excellent adhesion and hardness after suitable heat treatment. This coating markedly improves the wear resistance of almost any material that might be used to make the drum. A modification of the Brenner process to include cobalt produces an alloy which has good recording characteristics. This alloy is satisfactory magnetically without heat treatment but can be made harder with heat treatment.

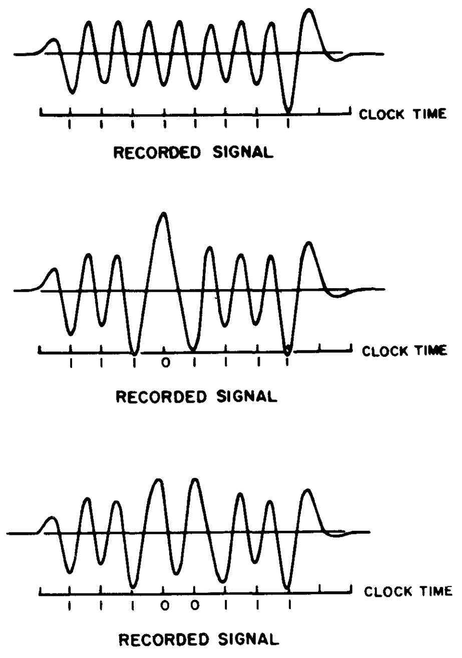


Fig. 3—Read signals with clock times indicated.

Determination of Magnetic Properties and Thickness

The signal read from the recording surface will be

$$E \text{ peak-to-peak} = 2\phi\omega \cdot n \times 10^{-8} \text{ volts} \qquad (1)$$

ϕ = maximum number of flux lines in the head

n = number of turns on head

ω = frequency in radians per second

This assumes that the readback signal is essentially sinusoidal. The parameter ϕ will be less than the flux lines remaining in the recorded dipoles after magnetization since not all the lines can be made to link the head. It will be proportioned to track width. It will be dependent on B_r and H_c for the magnetic coating.

For a thin magnet which is very wide, it can be shown that

$$H = H_0 - (2t/\pi L)(B\text{-}H) \qquad (2)$$

where

H_0 = applied field

H = effective magnetizing field

t = plating thickness

L = length of the recorded dipole

B = magnetic induction

A nickel-cobalt plating having a coercive force of 320 oersteds and a saturation induction of about 6000 gauss was selected. The ratio t/L can be varied so that a demagnetizing H just intersects the corner of the B-H loop for the material. Since L is fixed by the recording density, t is selected so that the residual induction is near the maximum induction, thus taking advantage of the squareness of the hysteresis loop of the nickel-cobalt alloy. A greater thickness would provide no greater residual flux because of demagnetization, but would require a greater recording magnetomotive force and would magnetize more slowly due to eddy current effects. Thus both magnetic plating material and its thickness can be optimized for the drum memory.

Fig. 4a shows an actual B-H loop of a nickel-cobalt plated film to show the effect of thickness on the residual induction due to demagnetization. A line is drawn of slope determined by t/L which intersects the B-H loop at the point of residual induction.

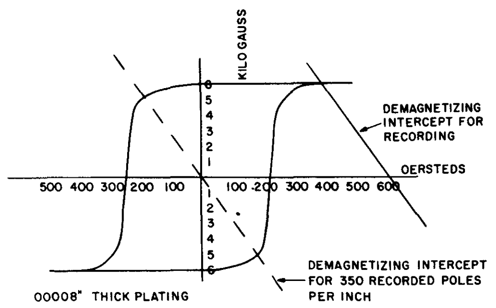


Fig. 4(a)—Actual B-H loop for electroplated nickel-cobalt.

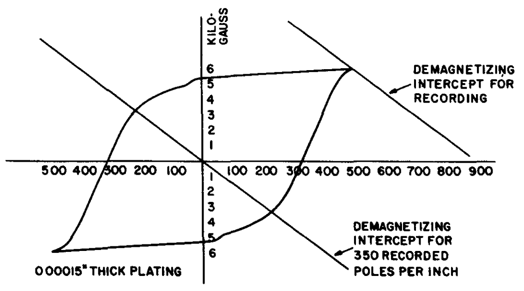


Fig. 4(b)—Actual B-H loop for heat treated electroless nickel-cobalt.

Fig. 4b shows a similar B-H loop for a heat-treated, nickel-cobalt alloy chemically deposited by the Brenner process. The squareness is not as good as that obtained by electroplating, but it is expected that this could be improved.

The B-H loops were taken on actual plated 2½ inch diameter by 4½ inch long stain steel cylinders before they were mounted on the drum assembly. (See Fig. 5 for photograph of the B-H tester.)

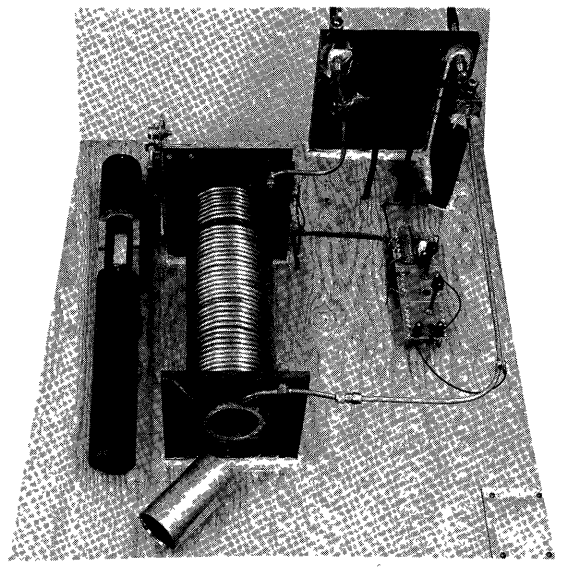


Fig. 5—B-H loop tester.

The B-H loops were taken by magnetizing the plating axially in a solenoid whereas recording takes place around the periphery of the drum. There was some doubt as to whether or not anisotropic effects would invalidate this measurement, and so several disks were plated and tested along various axes in the B-H tester. Very little change in B-H characteristics was noted as the direction of magnetization was changed. The disks were purposely ground so that the effect of grinding marks would be observed if they set up an easy direction of magnetization.

Design of Suitable Read-Record Heads

The design goals called for 350 bits per inch recording density and at least 30 tracks per inch. Reading resolution of 350 Manchester cells per inch requires coupling as much flux as is possible from a 0.0014-inch-long magnetic dipole into a magnetic structure around which are wound a number of turns of wire. Coupling much of the flux requires a head gap of the order of 0.0004 inch and head to recording surface spacing smaller than 0.0001 inch. However, a compromise can be made which will cause a loss of signal but not necessarily loss of operational margins. Recording densities of more than 1000 bits per inch have been obtained in systems using a single floating head assembly. However, this usually is accomplished with very closely spaced heads and wider tracks than 0.025 inch. In the interest of economy and development time a compromise which utilized many heads mounted in a single air-floated pad was adopted. To make the construction problem easier, a head-to-drum spacing of 200–300 microinches was adopted. This limits a practical digital recording system to the region of 500 recorded bits to the inch. In the present system a recording density of 350 bits per inch is used, but this does not represent the practical system limit. The floating pads holding about 27 heads are of the order of 1.3 inches by 1.25 inches. Economies in space and cost are achieved by this mass mounting method which at present requires the use of recording densities of 500 bits per inch and less. The problems of recording and reading will be discussed separately although it is highly desirable that a compromise head be used which can both record and read. Apart from economy it greatly relaxes mechanical tolerance problems.

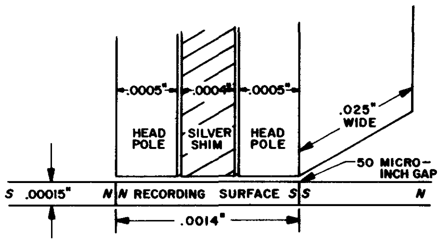


Fig. 6—Ideal geometry for recording 350 bits/inch.

Recording

Fig. 6 shows an idealized read-record head at its pole face. If the resistivity of the pole pieces were high so that eddy currents could be neglected, the amp turns required for recording and the read signal obtained per turn of the head winding could be quite closely calculated. Such a head is most difficult to make and the desirable spacing to the recording surface of 50 microinches or less is also most difficult to obtain in multiple head assemblies. The performance of the idealized head is of interest, however, for comparison with the compromise design which has been presently adopted but which clearly could be improved. To determine the recording amp turns required, let the *B-H* loop (Fig. 4b) be assumed to be the *B-H* loop for the recording surface. For the 0.00015-inch thick plating whose *B-H* loop is shown on Fig. 4b assuming

$L = 0.0014''$ (350 bits per inch Manchester recording)

It can be seen from Fig. 4b that 500 oersteds are required to saturate the magnetic plating at 6000 gauss. From Equation (2) we find that

$H_0 = 900$ oersteds approximately for 500 oersteds effective magnetizing force

Two parallel lines are shown on Fig. 4b, whose intersections with the *B-H* curve and *H* axis give the residual flux density and the recording force required. This gives a flux density after magnetization of 3200 gauss. If the curve of Fig. 4a were used, magnetizing force of 600 oersteds would give a remnant density of 5500 gauss. However, because the electroplated coating is thinner (80 microinches versus 150 microinches), the remnant flux would be only 90 percent of that obtained for the electroless plating.

The overriding consideration for selecting the electroless plating was its hardness and resistance to wear.

The remnant flux for a recorded dipole 0.0014 inch long, 0.025 inch wide and 0.00015 inch thick would be about 7.7×10^{-2} lines for a flux density of 3,200 gauss.

About 2.5-amp turns must be provided for magnetizing the plating. In the ideal head (Fig. 6) 14.5×10^{-2} lines must be maintained across two gaps in series to saturate the coating at 6,000 gauss. The gap dimensions are 50 microinches in extent, 0.025 inch long and 0.0005 inch wide. This infers an average flux density in the air gap of 1,800 gauss, the maintenance of which will take about 0.36-amp turns. The maintenance of flux in a very small continuous permalloy or ferrite circuit will take a negligible extra number of amp turns.

In practice, sufficient amp turns must be provided to generate a large number of fringing lines which form closed circuits around the side of the head and under and over the recording surface. If the ideal head as drawn in Fig. 6 were made, 3–4 amp turns would be sufficient for recording on the magnetic coating specified.

In practice, allowance has been made for the fact that the air gap may be 300 microinches instead of 50 microinches since this is much more readily achieved in a multiple assembly holding 27 heads. The best compromise for recording also includes making the silver shim gap larger than would appear ideal for small head-to-drum spacings since the flux density drops off rapidly in terms of the head gap dimension. A practical though not very efficient head would utilize 0.001 inch wide pole pieces with a 0.001 inch wide silver shim. (See Fig. 7.) Such a head records with 15-amp turns but gives a slightly greater read

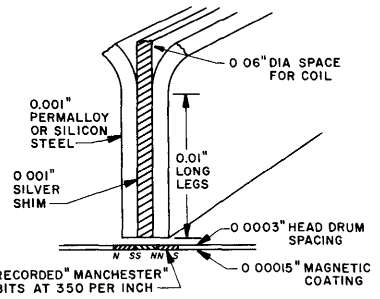

0 06" DIA SPACE FOR COIL
0.001" PERMALLOY OR SILICON STEEL
0.01" LONG LEGS
0 001" SILVER SHIM
0 0003" HEAD DRUM SPACING
N SS NN S
RECORDED "MANCHESTER" BITS AT 350 PER INCH
0 00015" MAGNETIC COATING

Fig. 7—Practical geometry for recording.

signal using 30-amp turns. Since these figures are large compared with the calculated 3-amp turns, it is clear that recording efficiency was sacrificed in order to make the head easier to fabricate and less sensitive to spacing than the ideal head. This inefficiency becomes important only if the recording circuitry becomes large or impractical. A head made to the dimensions shown on Fig. 7 has been driven at 546 kc with a silicon transistor circuit using 6 volts on the collectors and 100 to 200 ma peak current. Since this circuit is quite acceptable for a microminiature computer, recording efficiency can be sacrificed if this results in a net savings in manufacturing cost. The practical geometry of Fig. 7 clearly looks inefficient magnetically, but economy and ease of manufacture are in its favor. The 0.01 inch long legs are highly desirable for mechanical structure since a clamp holds the permalloy against the silver shim. The silver shim is wide for the size of the recorded dipole, but head spacing is far less critical than if the silver shim were closer to a more reasonable appearing dimension. Laminating the legs of the magnetic structure will improve the performance since penetration of the magnetic field at 546 kc is about 10 percent into either side of the material (assuming nonsaturation) for the half amplitude point. In practice, excess drive is used which causes the penetration to be greater than the 10 percent mentioned above. The penetration is greater because the permeability of the material is lowered as it becomes saturated, resulting in an increased speed of propagation in the saturated region. The final choice of magnetic head is likely to be a compromise between the schemes shown in Figs. 6 and 7. For practical reasons, the dimensions shown in Fig. 7 make a good starting point for the development of a useful system.

The magnetic head structure is made of 0.001-inch permalloy rather than ferrite which would be too hard to handle in sufficiently small sizes. Under less than ideal conditions for recording, there are very marked transients where recording starts and stops, since some recording on a minor hysteresis loop takes place under the full region of the magnetic head. As recording density is increased without scaling down the head gap and head-to-recording surface spacing, this problem becomes more marked. For a chosen minimum head-to-drum spacing, the useful recording density can be greatly increased if the magnetic structure of the recording head is reduced to the smallest dimensions possible so that its influence does not appreciably extend beyond the recorded dipole. Care must be taken in using legs of small cross sectional area because there is not a large excess of flux over the amount required to saturate the coating. Flux leakage may prevent recording altogether unless the over-all head structure is kept very small.

Design of Read Head

It was shown earlier that a 150-microinch thick recording surface with the *B-H* loop characteristic of Fig. 4b would have 7.7×10^{-2} flux lines at the center of a 0.025 inch wide recorded dipole. An ideal head would intercept these lines (and even increase the available flux by reducing the demagnetization). If the flux change were sinusoidal (any other wave form would give greater peak-to-peak volts) the read signal would be

$$E = \phi\omega \cos \omega t \times 10^{-8}$$

where ϕ is the total flux in the recorded magnetic dipole and $E = 2\phi\omega \times 10^{-8}$ peak-to-peak volts per turn of the reading head. At 546 kc, which is the maximum frequency used, a signal approaching 5.2 mv per turn could be expected from an idealized structure. With this ideal structure, it would be easy to determine that resolving signals at a much higher density would be possible and thus it would most likely be used at a density where it would give much less than the theoretical maximum signal. The magnetic head tested with the memory system described falls far short in obtaining the maximum obtainable signal at maximum density. In fact the presently used heads develop a signal in the range of 12 mv peak-to-peak at 546 kc as against a possible 780 mv calculated for a 150-turn head. Reference to Fig. 7 indicates that unlike the situation in Fig. 6 where more than half the flux would couple the head windings, only a small part of the flux will be useful in generating a read signal. Calculation of the exact magnetic flux coupling in this situation is most difficult, but a glance at a scale drawing makes the finding of 1/66 of the possible signal quite plausible. The fact that the low output is tolerated is a compromise between signal level, and economy and ease of manufacturing the heads. Since an excellent signal-to-noise ratio and margins in clock pulse timing are obtained in this situation, the compromise is quite tolerable.

At 273 kc, which represents the pattern 0 1 0 1 in Manchester recording, the read signal obtained is about 30 millivolts in comparison with a possible 390 millivolts if all the flux in the recorded poles interlinked the head winding. The loss of signal by a 13 to 1 ratio is explained by the presence of an air gap, which provides a substantial reluctance in series with the head structure, and also by the fact that the head

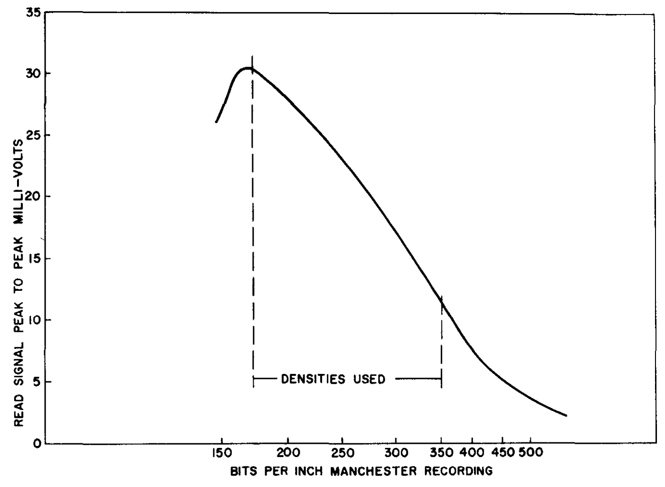


Fig. 8—Typical response for magnetic read-record head.

structure itself does not have a zero reluctance. Fig. 8 shows the response of the read head versus recording density and indicates that the head shown in Fig. 7 is being used beyond its optimum density.

Fig. 3 shows signals read by the head with clock times indicated. As can be seen, the signals can be interpreted with adequate reliability since there is no noise or mistiming in evidence.

There is, of course, much room for improvement of the magnetic head; however, each improvement increases the difficulty of making the head and the increased cost must be balanced against the economic benefits of the improvement.

Construction of the Magnetic Head

Fig. 9 shows the essential detail of the magnetic head. In assembling these heads the lower part is insulated and slipped into an aluminum tube. The tube is compressed forming a subassembly which can be tested. The subassembly heads are clamped into a holder (Fig. 10) and fixed in place with a suitable high temperature epoxy resin compound. Two such assemblies are made with the heads staggered so that with the assemblies mounted 15 to the inch a track density of 30 per inch is achieved. The assemblies are then mounted in the shoes.

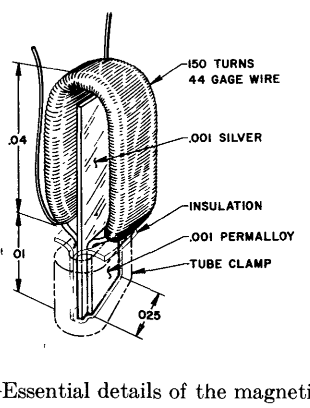


Fig. 9—Essential details of the magnetic head.

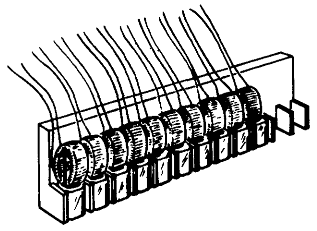


Fig. 10—Partly assembled magnetic heads.

Mechanical Design Details

Principle of Operation of Head Support Mechanism

A rotating drum moves a considerable volume of air in its close vicinity even though the drum surface is quite smooth by normal standards. This phenomenon is due to a boundary layer effect. That is, air molecules which are immediately in contact with the drum tend to adhere to that surface. Due to the viscosity of the air, the air molecules immediately about this initial layer are dragged along and as the distance from the drum surface increases, the velocity of the air molecules which are dragged along decreases. With this concept in mind, it is seen that if a stationary surface which is curved to match that of the drum is held near the rotating drum surface, the air will be dragged between the two surfaces. Since the air will also tend to adhere to the second surface, there will be a drag or friction force as shown in Fig. 11. If this stationary surface is inclined to the drum surface so that the space decreases in the direction of rotation, the air which is dragged in is squeezed into a progressively smaller space as is shown schematically in Fig. 12. This squeezing effect is of course a compression process, and pressure forces normal to the two surfaces develop. If this second surface is held in place by a spring force of proper magnitude, it will be held off the drum to a distance where the fluid

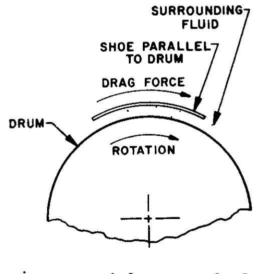


Fig. 11— Schematic view of drum with shoe in parallel position.

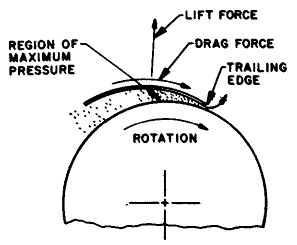


Fig. 12—Schematic view of drum with shoe at angle to develop wedge of lubricant.

pressure force equals the spring load force. When such a condition exists, the layer of fluid which separates the two surfaces is referred to as a hydrodynamic lubricating film and such surfaces which react in this manner are referred to as a self-acting bearing. In the example given, air is used as the lubricating fluid; however, any fluid, liquid, or gas which will adhere to the bearing surfaces without causing damage will perform in this manner.

The theoretical aspects of this phenomenon were first proposed by O. Reynolds about 75 years ago, and solutions of his equation for the incompressible lubricating films have been well accepted in the literature on bearing lubrication. In recent years considerable attention has been directed toward the case of the compressible or gas lubricating film for many promising advantages such as chemical stability, extremely low friction, the maintenance of close clearance between moving parts, and the use of the ambient gas as a lubricant. The technology of the analysis of the bearing using a compressible fluid as a lubricant as in the example (Fig. 12) is quite involved and beyond the scope or purpose of this paper. Work on this phase for use in the design of such bearings for use in memory drums is in progress; and for the technology the reader's attention is directed to the list of references.

Properties of Lubricating Film Supported Shoe

The lubricating film supported shoe possesses certain unique properties which make it a useful device for the support of a recording head. The most important properties will be described below. For the purpose here, let us denote the angle between the drum and shoe surfaces as the attack angle α, the edge farthest from the drum surface as leading edge, and the edge nearest the drum as the trailing edge. Drum rotation is in the direction from the leading edge towards the trailing edge.

The gas lubricating film is not only unique for its thinness but also its high spring rate.

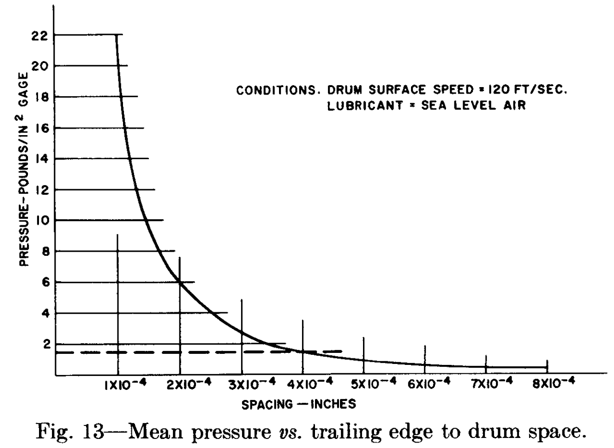


Fig. 13—Mean pressure *vs.* trailing edge to drum space.

Fig. 13 shows the typical relationship of the pressure force which can be developed under typical operating conditions. It will be noted that at the operating conditions shown in the figure, a mean pressure of 1.5 psi gage at a trailing edge spacing of 400 microinches is obtained. As the trailing edge spacing is decreased, the mean pressure increases at a rapid rate so that at a spacing of 200 microinches the mean pressure has increased virtually four fold or inversely with the square of the spacing. This characteristic is most desirable from electrical and mechanical points of view for recording drum applications. For any fixed design as the load is increased the shoe and hence recording head to drum spacing is decreased. This is, of course, helpful to the electrical performance as far as output signal is concerned. As a greater load is applied to the shoe, the ratio of the applied load to the weight or inertia of the shoe and its associated mechanism is increased. When this ratio is increased, the ability of the shoe to withstand accelerations and run-out-irregularities of the drum is also increased. In the case of the drum which is the subject of this paper, the effective area of the shoe is 1.6 square inches and its normal operating load is 10 pounds, which gives a mean pressure of about 6 psi. The shoe with recording heads in place has an effective weight of about 1.5 ounces, and so the load-to-weight ratio is slightly over 100. Since at this operating condition, slight changes in the spacing result in a considerable change in the lift force, there is available a large force to restore the proper head to drum spacing. Let us consider an example at the conditions cited above. In a broad sense, since the curve shown in Fig. 13 is one of a force *vs* displacement, the lubricating film may be regarded as a spring of variable rate. If the displacements are left small, the lubricating film may be approximated by a linear spring and the slope of the curve may be taken as the spring constant. For the conditions cited above, this slope or linear spring constant is about 100,000 pounds per inch for a show of the given effective area. The spring rate of the spring used to produce the load force would have to be added to this rate; however, since this spring would have a rate of about 100 pounds per inch, it is virtually insignificant in its effect on the natural frequency of the system of forces acting on the show. The spring rates of 100,000 pounds per inch acting on the effective mass of the shoe give a resonant frequency of more than 3000 cycles per second. Thus it follows that such a mechanism is quite capable of withstanding accelerations of 10 g's up to 2000 cps without seriously affecting the output electrical signal.

Another unique property of the floating shoe is its inherent stability. Fig. 14 shows a typical pressure distribution between the trailing and leading edge of the shoe. As the attack angle α is increased, the center of pressure shifts towards the trailing edge, and similarly as the angle α is decreased the center of pressure shifts to the leading edge. Let us fix a certain shoe

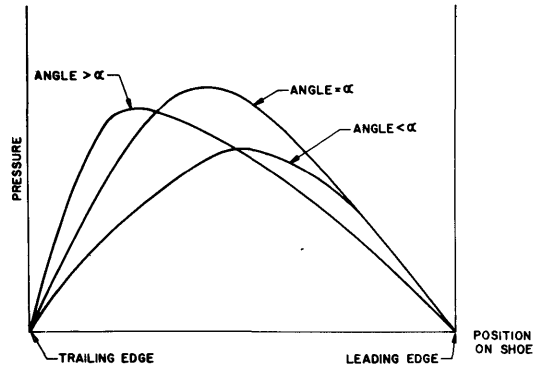


Fig. 14—Typical pressure distribution *vs.* change in attack angle α.

geometry and allow the shoe to pivot about an axis at the center of pressure and parallel to the drum surfaces. Now if the shoe is tipped so that the angle α is increased, the center of pressure moves toward the trailing edge. This action develops a turning moment on the shoe. The turning moment is in the direction required to return the shoe to the original position. Similarly, when the shoe is tipped to an angle less than the stable angle, a turning moment of opposite sign develops to return the shoe to the original stable position. From experience it has been found that the system has sufficient damping to make it stable. Thus it follows that the location of the pivot axis is not critical, for the shoe will tend to seek a value of the angle α so that the action line of the center of pressure will pass through the pivot axis.

Design Requirements

The design of a mechanism to make use of the lubricating film supported shoe or for keeping a recording head in proper location with respect to the drum recording surface requires careful attention to the precision requirements of the mechanism. The development of a design framework which requires a minimum of very precise parts which are amenable to precision manufacturing techniques is necessary to the successful execution of the task. It is not only necessary to have surfaces which are geometrically true, but it is also required that the proper geometric relationship between the various parts be accurately maintained. The most important of these relationships is the alignment between the shoe and the drum. It is essential that the center of curvature of the shoe be maintained parallel to the axis of rotation of the drum. The limits of accuracy required are dependent upon the particular design and the performance required. For the design the out-of-parallelism is kept to less than 3 parts in 10,000. The other important requirement is that the load on the shoe be uniformly distributed so that tipping does not occur. As will be shown later, the load on the shoe of the subject drum is applied at two points. The difference between these forces is kept to a value less than 7 percent. The tolerances given above are those used in the design of the drum with due allowance for possible manufacturing tolerance and also the expected deflections of the mechanical parts.

Figs. 2, 15, and 16 show the drum in various stages of assembly. It will be noted that the rotating portion of the drum is set into a very rigid frame, and access to the drum-recording surface is through appropriately located cutouts in this frame. A V-groove is machined into the sides of this frame so that it is accurately parallel to the axis of the drum. Guide slots for radius arms are machined at precise right angles to the V-groove. Each of the radius arms are provided with polished sapphire pivot pins which are cemented in place in an assembly fixture. The centerline distance between the pins is accurately maintained so that it is virtually the same for a given pair of arms associated with a given shoe. One pin of each arm operates in the V-groove of the frame, while the other pin operates in a V-groove in the shoe. The V-groove in the shoe is located in the line of action of the center of pressure, and it is made accurately parallel to the axis of the cylindrical surface of the shoe. To prevent smearing of the pole pieces of the recording heads, the curvature of the shoe is ground by means of a contoured abrasive wheel so that the lay of the grinder marks is parallel to the head gaps. Final finishing is done on a cylindrical lapping tool which has a diameter 0.1 percent greater than the drum. The load for the shoe is supplied by the spring which is adjusted by a single centrally located screw. By this means, equal forces are applied to each side of the shoe. The load forces the pins to seat in the V-grooves of the shoe and frame and precisely locate the shoe with respect to the drum so that the axis of the drum and shoe are parallel within the extremely close limits previously cited.

Special consideration must be given to start and stop conditions, for without sufficient drum speed the lubricating wedge or film cannot develop and a high-

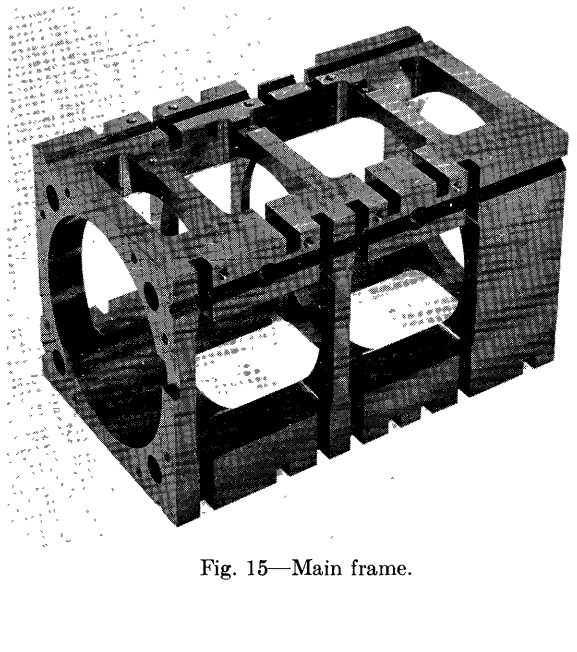


Fig. 15—Main frame.

friction condition will exist. To prevent this, it is necessary to unload the shoe and lift it slightly off the drum surface until sufficient speed for normal operation is attained. For stopping the drum, the procedure is reversed. There are basically two methods by which this may be accomplished. One method involves removing the spring load until operating speed is reached. The second method involves introducing lubricant under pressure through a very small hole in the shoe into the space between the shoe and drum. If sufficient lubricant (in the subject drum it is air) is supplied, the shoe will be lifted off the drum surface. After operating speed is reached, this supply of air may be shut off and normal operation resumed. This latter method requires the use of an air compressor, a fact which makes it somewhat unattractive for airborne use. The first method is used in this drum design. It will be noted that in Fig. 16 the radius arms extend from the side of the frame which has the V-groove to the opposite side.

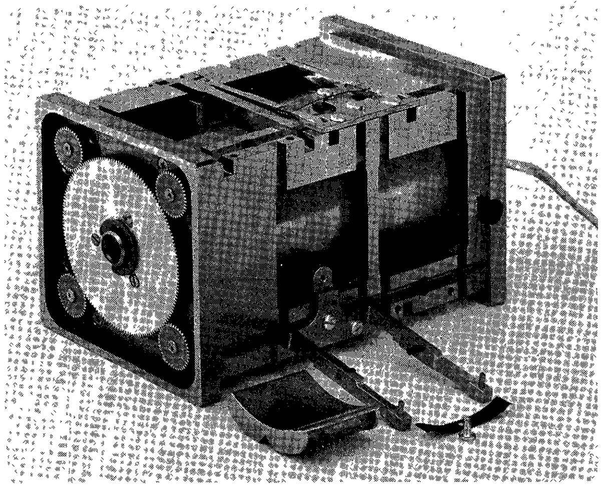


Fig. 16—Shoe and radius arms.

At this side of the frame, the ends of the arm can ride on a simple eccentric cam which is operated by the small gears. During normal operation, these ends of the arms are free of the cam. For offspeed operation the cam is rotated to a position where the ends of the arms are lifted. Since the mechanism is extremely rigid, a movement of less than one mil of the end of the arm is sufficient to transfer the spring load from the shoe to the cam. In this condition, the lubricating film between the drum and shoe must support the weight of the shoe. Since the weight of the shoe is very much less than the operating load, the resulting friction is negligible. If the magnetic coating is very durable, the slight contact between the shoe and drum under these conditions is not serious and may be eliminated completely by operating the drum with the axis in a vertical position. When the shoe is in this free condition, a state of instability may develop if the cam is inadvertently set to lift the ends of the arm too high. Should this condition develop, serious

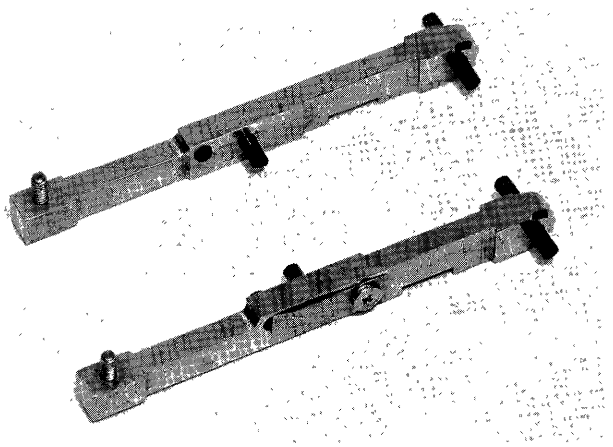


Fig. 17—Typical pair of radius arms.

damage to the drum and shoe surface will occur. To eliminate this possibility, one arm of each pair for a given shoe is provided with a spring-loaded pin as shown in Fig. 17. This pin is allowed to act upon the side of a shoe to cause a small amount of friction damping. Since the load at which the shoe is operated is much higher than the weight of the shoe, this damping friction does not affect the operation any noticeable amount.

The main frame, as almost all other parts of the drum assembly, is made of a precipitation-hardening stainless steel. For the sake of rigidity and precision, it is fabricated from one piece of stock and provided with generous ribs.

References

[1] W. A. Gross. "A Gas Film Lubrication Study" Part I, "Some Theoretical Analyses of Slider Bearings," *IBM Journal of Research and Development*, vol. 3, No. 3 (July 1959), pp. 237–255.

[2] W. A. Michael. "A Gas Film Lubrication Study" Part II, "Numerical Solution of the Reynolds Equation for Finite Slider Bearings," *IBM Journal of Research and Development*, vol. 3, No. 3 (July 1959), pp. 256–259.

[3] R. K. Brunner, J. M. Harker, K. E. Haughton, and A. G. Osterlund. "A Gas Film Lubrication Study" Part III, "Experimental Investigation of Pivoted Slider Bearings," *IBM Journal of Research and Development*, vol. 3, No. 3 (July 1959), pp. 260–274.

[4] D. D. Fuller. *Theory and Practice of Lubrication for Engineers*, John Wiley and Sons, Inc. New York, N. Y. (1956).

[5] A. Kingsbury. "Experiments with an Air-Lubricated Journal" *Journal of the American Society of Naval Engineers*, vol. 9 (1897), pp. 267–292.

[6] J. S. Ausman and M. Wildman. "How to Design Hydrodynamic Gas Bearings" *Product Engineering*, (25 November 1957) pp. 21–28; 103–106.

[7] A. Brenner, Grace E. Riddell. U. S. Patent No. 2532283, (5 December 1950).

[8] A. Brenner, Grace E. Riddell. "Deposition of Nickel and Cobalt by Chemical Reduction," Bureau of Standards Research Paper No. RP 1835, vol. 39 (November 1947).

Addendum

In the body of this paper, it has been stated that as recording density is increased without scaling down the head and drum coating geometry, recording on a minor hysteresis loop takes place under the full region of the head. Since present conventional memory drum design does not follow the practice of scaling down the entire recording geometry with increases of recording density, it would appear appropriate to describe here two series of experiments which yield data supporting the above statement.

In the first series of experiments, a recording geometry shown in Fig. 7 in the body of this paper was used. It was found that for a given set of operating conditions there was an optimum head drive current for maximum output signal. This phenomenon can be explained by the demagnetizing effect of the fringe flux upon the adjacent dipoles, which is increased in strength by the increase in drive.

Saturation of the head pole pieces and drum coating can not be the case because saturation would yield a limiting effect and not a maximum point of operation.

In the second series of experiments, a typical small ferrite head with pole pieces about 25 mils square was used which gave a head length of slightly more than 50 mils. The head to drum spacing and all other conditions were the same as in the first series of experiments described above. At a recording density of 109 bits per inch, the output signal was about 40 millivolts and there was little or no tendency to pattern sensitivity. When the recording frequency was increased to give 350 bits per inch, the output fell to 6 millivolts and the pattern was distorted to the extent that errors would be caused in reading it. It is believed that this pattern distortion and signal attenuation is due to an anticipation effect. That is to say that as each dipole starts under the leading edge of the pole piece, some of its flux passes into the pole piece and gives a read out signal. In these experiments, it was possible to correlate exactly the output signal wave form with the recorded bit length and the size of the pole piece.

In one case the drum was demagnetized and then a single 8 bit pattern was written on it. Upon reading back, the entire pattern was read three times: first as the leading edge of the head pole point passed over the dipoles, secondly with larger amplitude as the dipoles passed under the gap, and once again as they left the trailing edge of the pole piece.

In an actual computer application, if this anticipation effect occurs, the true output signal may be so distorted that serious errors may be introduced or the effectiveness of the system considerably reduced.

It appears that from a practical point of view for an adequate design compromise for freedom from pattern sensitivity and signal level output the head dimension measured in the direction of drum rotation should not exceed the length of the recorded dipole by an appreciable amount.

Discussion

W. N. Papian: I wonder if you could tell us what the status of the project is right now?

Mr. Howard: This was a research and development effort to determine the feasibility and the basic design requirements of a drum of this type. We have built several test models and the model shown, which is in the form factor suitable for use in a micro-miniaturized computer. At the present time, it is running and undergoing environmental tests. We are still working on it, and we believe it to be a practical device. If our hopes materialize we shall wrap a computer around it.

M. J. Haims (IBM): How was the head to drum spacing and angle of attack measured?

Mr. Howard: It is very difficult to measure. The curve shown is for calculated values and has been verified by laboratory measurements. We have verified the general shape of the curve by at least two methods. One method made use of electrical capacity between the drum and small pads set into the shoe. The other method measured the lift distance of the shoe off the drum surface by means of sensitive dial gages and special electrical capacity probes.

D. Killen (Oliver Shepherd Industries): What is the peripheral speed of the drum, and what is the head inductance?

Mr. Howard: The drum is 2.5 inches in diameter; the sychronous speed is 12,000; and actual speed, allowing for motor slippage, is about 11,600. This gives a peripheral speed of about 120 feet per second. The head inductance is 60 microhenries.

W. G. Dosse (MIT): What magnetic shielding do you have between motor windings and drum and readers, or is stray motor flux not a problem?

Mr. Howard: It would be a problem if you didn't take care of it. The motor is surrounded by a magnetic shield, and the magnetic recording material is plated on a separate sleeve which is shrunk on the motor drum assembly.

K. Enslein (Brooks Research): The storage density in your system is approximately 2,500 bits per cubic inch. Could you discuss the relative advantages of magnetic cores and your drum for the application at hand?

Mr. Howard: I am in no condition to do any mental computing up here. But, as for comparing, you can get a cheaper bit stored on a drum system than you can in a core. Of course, the disadvantage of the drum is the access time. By suitably programming a fixed program and using circulating registers or revolvers to bring the words up as required, you can cut this down. One of the main advantages is the possibility of extending this concept to a system with extremely large capacity. I don't think that cores could take the punishment that this thing will.

D. Roberts (Librascope): What air pressure range will the drum operate under? Is it pressurized?

Mr. Miller: A lot depends on the loading and hence spacing at which the shoe is going to be operated. We have operated the unit up to a pressure altitude of 30,000 ft. Since equipment like this should be operated in a dust free atmosphere, we do not feel it is a great disadvantage to the design if it is kept in a pressurized container. When it is finally installed, it will be in a pressurized box filled with some inert and dry gas such as nitrogen.

P. Smith (Gen'l. Transistor): What keeps the shoe from the drum before the drum gets up to speed.

Mr. Miller: The four small gears you saw in the slide are attached to ends of eccentric cams. During the start operation, the cams are rotated into a position in which the spring load is carried by the cam. There is also a spring loaded pin in the arms to help keep the shoe off the drum during the start operation. When the operating speed is achieved, the cams are rotated and the spring load is transferred to the shoe. During the first moment of starting there may be some tendency for the shoe to contact the drum; however, sufficient lift is developed at 500 rpm. To prevent damage during this early part of the starting cycle, we have used a very durable hard electroless plating of a nickel cobalt alloy.

J. Russell (University of Calif.): What is the peak to peak read signal at your bit density?

Mr. Howard: 12 millivolts.

G. G. C. Randa (IBM): What is resonant frequency of shoe and its spring mechanism?

Mr. Miller: If you will recall what the curve of pressure *vs.* spacing looks like, it is, of course, not linear and so does not really yield a resonant frequency as such. If we consider only small displacements and operate in the 200 microinch range, we find that the spring constant is about 100,000 pounds per inch. Since the shoe and arms have an effective weight of about one tenth of a pound, the resonant frequency would appear to be about 3000 cycles per second. With induced vibrations up to 500 cycles per second no malfunction of the shoe support mechanism seems to occur. However, at the higher frequencies, say at 2000 cycles per second, malfunction is impending and so some small amount of vibration isolation may be required in the final installation if the supporting structure does not attenuate these frequencies sufficiently.

P. Skelly (RCA Service): Are there any temperature controls used?

Mr. Howard: No, None at all.

Temperature Compensation for a Core Memory

A. H. ASHLEY†, E. U. COHLER† AND W. S. HUMPHREY, JR.†

FOR FIXED installation, it is often possible to control the temperature of ferrite core memories within narrow limits. However, in a small mobile computer designed to operate over world-wide conditions, this control is not feasible because of the added weight, volume, and cost encountered. A memory designed for such application has been temperature compensated by the use of temperature sensitive components in the current sources to the X-Y drivers and in the power supplies for the Z drivers. In addition, core derived strobing has provided peaking time compensation for the sense amplifiers as changes in transistor characteristics delay or advance drive current. This compensation permits operation of an 8192 word 38-bit transistorized memory running at an 8 microsecond cycle time in an ambient environment which may vary between −30°C and +55°C.

Introduction

Most computers use some form of temperature control to maintain the operating temperature of the ferrite cores within very close limits. This precaution is required because of the sensitivity of the ferrite material to ambient temperature variations. When the environmental temperature goes up, the coercive force will go gown and the material then loses some of its squareness, consequently becoming more disturb-sensitive as shown in Fig. 1. Therefore, if the

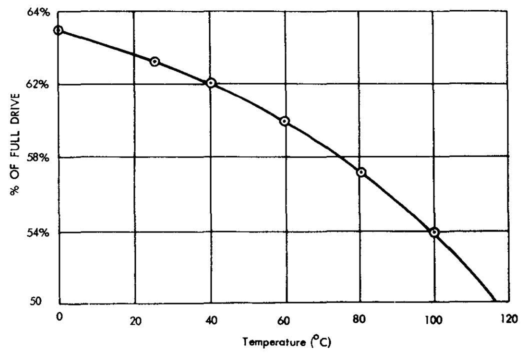


Fig. 1—Percentage of full drive which will disturb stored information *vs.* core temperature.

drive current is held constant while the temperature rises, the cores switch faster, giving greater amplitude to the output, and the cores are more sensitive to disturbance by the half-selecting drive pulses. The reverse effect is observed as environmental temperature is lowered. Below 0°C, the drive which has proven satisfactory at +25°C will produce a ONE about half as great as previously observed at normal room temperature. Under this condition, there would not be an output from a conventional sense amplifier. Moreover, a fixed strobe would miss the peak signal-to-noise time if the switching characteristics were changed by such an amount.

† Sylvania Electric Products Inc., Needham, Massachusetts.

Temperature Control

The general solution to the temperature problem has been to control the temperature within the memory enclosure within a few degrees Centrigrade. While at first this appears to be a simple solution to the problem, it has proven unsatisfactory over a large temperature range. To maintain the temperature at 95°C above the ambient (say at 65°C in a −30°C ambient) it will be required to install a rather large insulating oven complete with blowers and high wattage heaters and provisions for creating turbulence for proper mixing. When operating in conjunction with accurate thermostatic equipment, it will suffer from the inherent disadvantages of all mechanical components. The reliability from such components will result in degrees of magnitude lower than that of the memory or the accompanying solid-state circuitry. Moreover, the cost of a good air thermostat is considerably greater than that of the few electrical components required to do the job.

Temperature Compensating the Drive-Currents

Another alternative to control of the temperature of the memory cores is control of the drive-current amplitude. If the drive current is varied with temperature so that half selected cores are not disturbed but the fully selected cores are properly driven for full switching output, satisfactory operation is obtained. From the memory cores of the type used in Sylvania's MOBIDIC it was determined that drive-current compensation aimed solely at maintaining constant switching time resulted in a considerably lower output signal amplitude at the low end of the temperature range. Since the cores are less disturb-sensitive at lower temperatures, it is feasible to compensate for constant output-voltage-amplitude. The constant amplitude compensation below 20°C minimizes sense amplifier problems since no variation in strobe level is required. The overall compensation curve, shown in Fig. 2, results in a constant core output below 20°C and constant switching time above that temperature.

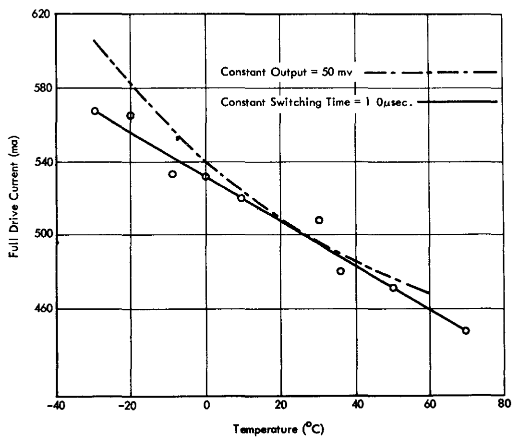


Fig. 2—Core current required *vs.* temperature.

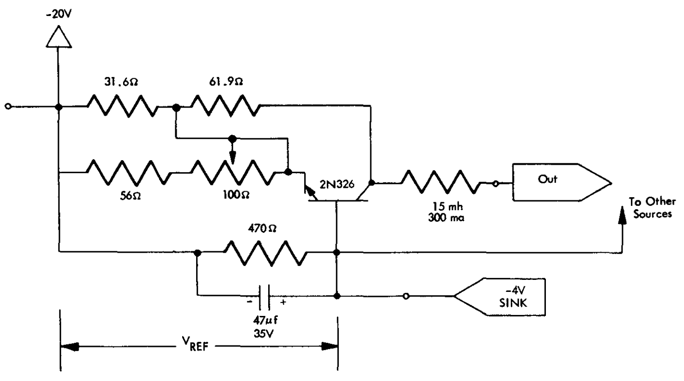


Fig. 3—Current source.

X-Y Drive-Current Sources

The drive currents for the X-Y coordinates originate from high impedance current sources, each source consisting of a power transistor connected in the common base configuration. The high impedance is required to maintain good current regulation under varying load conditions. The circuit for the current source is shown in Fig. 3. Current is supplied to the emitter of the current-source transistor by a source consisting of a reference voltage V_{REF} applied across a variable resistance network. The resistance is partially variable to compensate for initial differences in transistor parameters. The reference voltage V_{REF} is common to all current sources in the X-Y circuitry.

Choice of Compensation Technique

The output current may be varied with temperature by one of two methods. Either the external emitter resistors may have a positive temperature coefficient, or the Voltage reference may have a negative temperature coefficient. There are no positive temperature coefficient resistors available with sufficient power capability for the first method. Even if they were available they would not be very practcal to use because of drive current tolerances. The latter method is considerably better since it employs only one temperature-sensitive network per memory and uses readily available negative temperature coefficient elements. Moreover, the common compensation assures that all drivers vary equally, thus minimizing drive current tolerance problems.

Voltage Reference Design

Thermistors (negative temperature coefficient resistors) have a relatively low dissipation coefficient (watts/°C rise). It is, therefore, advisable to maintain a negligible dissipation within them in order to have their resistance remain a function of true ambient temperature without side effects from internal heating. Consequently, the thermistor net-

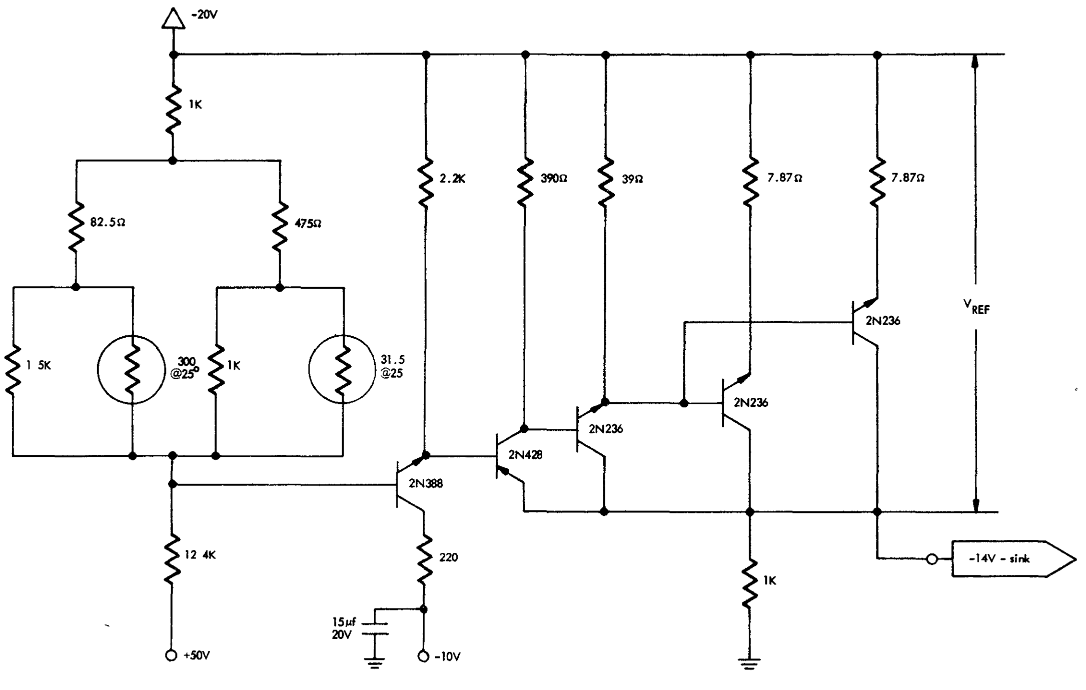


Fig. 4—Temperature compensated voltage reference (X-Y).

work is buffered by a power amplifier with a unity voltage gain and a high input impedance, allowing the use of current as low as 5 ma in the network.

The thermistor network and buffer amplifier are shown in Fig. 4. Notice that the V_{REF} is derived from the − 20 volt supply as indicated in Fig. 3. This means that variation in the − 20 volt supply will not affect the current source accuracy. Two thermistors are necessary to provide the proper compensation characteristics over the entire temperature range. A constant current of 5 ma through the 1k precision resistor provides a constant drop of five volts. The thermistor network with 5 ma through it will add a voltage drop of 1.0 volt at +25°C, 2.2 volts at − 30°C, and 0.52 volts at +55°C. The overall curve between temperature end points is nearly linear, (note that Fig. 2 is on an expanded scale) in great part due to the constant five volts superimposed on the temperature-sensitive voltage.

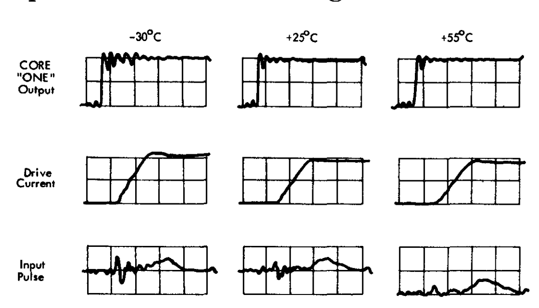


Fig. 5—Temperature compensated outputs.

Results Follow Theoretical Curve

The oscillographs in Fig. 5 were taken from an experimental system consisting of transistor core drivers and a memory core. The drive current varies through the desired pattern, although the compensation at this time was slightly less than that shown in Fig. 2. Even so, the ONE at − 30°C is within 10% of the ONE's at the other temperatures. A subsequent slight revision of the thermistor network was made to increase drive at the lower temperatures, resulting in a higher output at the low end without affecting the drive at other temperatures. The final compensation characteristic is as shown in Fig. 2.

Compensating the Z-Drive Current

The Z-drivers do not employ high impedance transistorized current sources such as those used for the X-Y drivers, because of less stringent current tolerances. The current for each Z winding is determined by the power supply voltage across a fixed resistance in series with the winding, as shown in Fig. 6. In order to vary the current with temperature, either the resistance or the total voltage across the resistance must be varied. The first method was impractical, because resistors with large positive temperature coefficients are not available. To vary the whole supply-voltage with temperature is not practical due to complications in the power supply design. To overcome these problems, one end of the current determining resistance R1 was connected to a fixed close-tolerance power supply (used elsewhere in the memory); and the emitter of the output transistor, (Q3, Fig. 6), was returned to a temperature sensitive supply V_{TEMP}. This supply was designed to vary from +0.5 volts at +55C to +5.0 volts at −30C. Because the maximum voltage swings up to 5 volts, considerably less power is involved in the temperature sensitive control than if the entire 20 volt supply were to vary from −20 volts to −25 volts, and the percentage variation is less critical.

Temperature Sensitive Emitter Supply

Because the thermistor network used in the X-Y coordinate has a quasilinear resistance-temperature characteristic, an identical network was used to derive

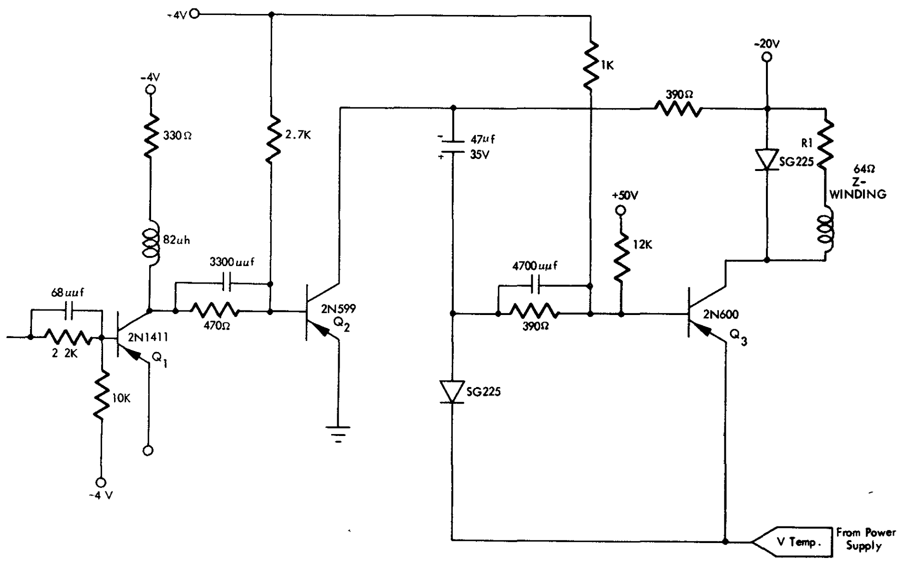


Fig. 6—Circuit for Z-driver.

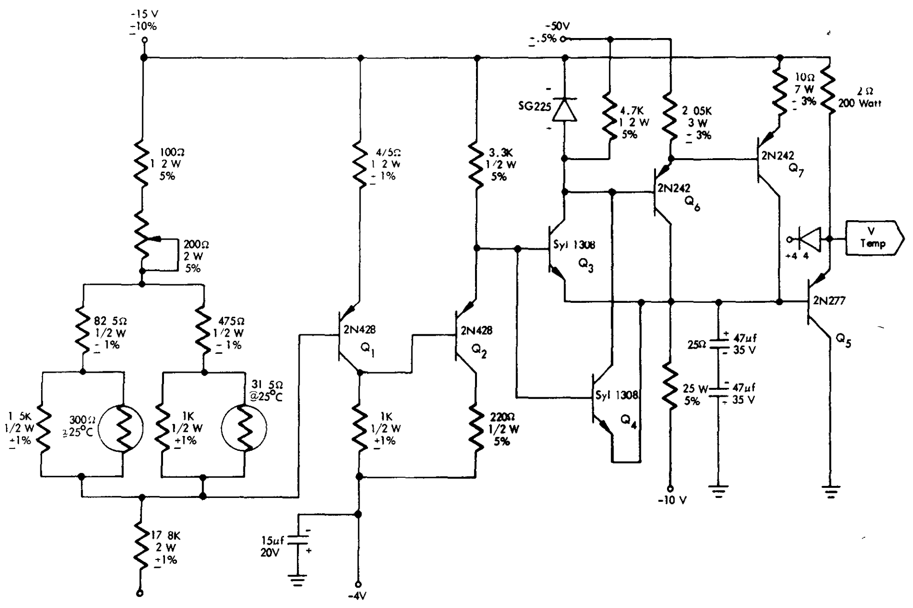


Fig. 7—Temperature controlled voltage source.

the emitter supply for the Z-drivers as shown in the circuit of Fig. 7. A stage of inversion with a voltage gain of 2 is interposed between the thermistor network and the power amplifier in order to provide the proper phase and amplitude to the variation. The output is clamped to ground on the low end by the transistor and to +5 volts (+4.4 plus diode drop) on the high end by a diode. This clamping insures against overvoltages on the Z-driver transistors. When none of the Z-drivers are in operation, 7.5 amps. are conducted to ground by the output transistor (Q5 of Fig. 7) of the V_{TEMP} circuit. When Z-drivers are being pulsed, the V_{TEMP} output conducts the difference between 7.5 amps and the average Z-driver drain.

Compensation with Core-Derived Strobe

The compensation of the drive currents still allows some variation in the peaking time of the core output, even for perfect amplitude compensation. Moreover, temperature affects the drive circuit delay. These effects can be observed in Fig. 5. The compensation for this variation is made completely and simply by the use of a core-derived strobe pulse. The time-discriminating-strobe is derived from a standard core receiving the same current as the selected cores in the memory. That core is essentially wired to receive a full read and full write from the *x* and *y* drivers selected to supply the rest of the memory. The output of the core is therefore a standard ONE produced at the same time as all other ONE's being read out. This output is then properly shaped and suitably delayed to supply a strobe pulse for the memory sense amplifiers. The block-schematic in Fig. 8 gives the outline of the method employed. Experimental results in a full memory show that variations in the sense amplifier output of 0.8 microseconds may occur and are compensated by the core-derived strobing even when ZERO's are larger than ONE's (under virgin-checkerboard test).

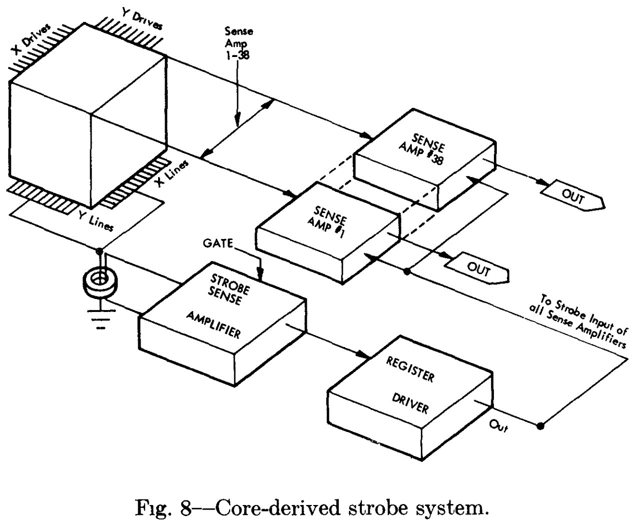


Fig. 8—Core-derived strobe system.

Conclusion

The operational limits of the memory were extended by the combination of core-derived strobing and temperature compensated drive currents, as shown in Fig. 9, which is a "shmoo" plot of temperature versus discrimination level limits of the sense amplifiers. The smaller area with cross-hatching shows the limit with core-derived strobe but without temperature compensation; the larger encompassing area shows extension of those limits by the temperature compensation. With neither core-derived strobing nor temperature compensation, the limits are reached at +10°C and +45°C.

Because the MOBIDIC computer in which this memory is being used is intended for battlefield operations, provision is made for retention of the information in the memory even after the computer

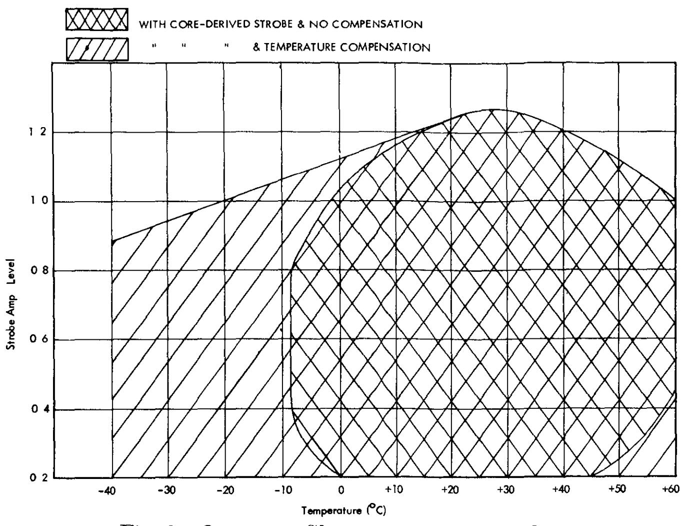


Fig. 9—Operation Shmoo for sense amplifiers.

is shut down. Conceivably the information could be read into the memory at one temperature and later read out at another. Tests performed on the cores showed no measurable difference between ONE's read out at a given temperature regardless of the core's temperature when the information was stored in it. Thus retention of memory is possible even if the machine is shut down and restarted in a new and widely different environment.

Discussion

W. N. Papian: I wonder if you could give us a very rough estimate of the percentage that has to be added as an increment to do this compensation?

Mr. Ashley: The biggest single item that has to be added is power supply capability because we have to increase this drive. We are using 32 current sources, each a quarter of an amp at 20 volts, and we have to have capability of power that the Z drivers might draw if the whole 38 drew at once. This increases by 20 percent the power requirements. There is an unregulated power that is required of 9 or 10 per cent regulation of 100 watts to effect this Z driver compensation, but beyond this the cost alone of the necessary additional parts is very slight.

W. Lawrence, Jr. (*IBM*): How do you regulate against temperature differences of memory locations receiving different interrogation rates?

Mr. Ashley: I assume that refers to whether or not the core is being self-heated by the application of the drive current. This particular memory is not working at a high enough power and speed to cause any appreciable heating, since the speed that is required is 8 microseconds cycle time. A total of 20 watts maximum occurs within the enclosure for which we have capable blowers to ensure that the air is circulated.

S. B. Yochelson (*Goodyear Aircraft*): Will you comment on the effects of local heating in the plane, such as might occur if a particular core is repeatedly interrogated?

Mr. Ashley: With the speed and power we have, we don't have that problem. We don't need to drive the cores for the real fast switching time that would require the high current. So the self-heating is rather minimized.

P. Barek (*Lincoln Lab.*): What is the variation in memory access and rewrite cycle time as function of temperature?

Mr. Ashley: The access time actually doesn't change much because we have a constant slope to the drive current. With increasing temperatures, although the circuit delay is more with a resultant later start for the current, the rise time between 10 per cent and 90 per cent of the current is faster. The two effects tend to cancel each other in access time. Except for the change in delay in the sense amplifiers and the external circuitry (registers, timing flip-flops, etc.) it is not much different. I would guess maybe 0.2 microsecond slower at the high temperature than at minus 30° C.

J. E. Veal (*RCA*): How much effect does low temperature have on core switching times?

Mr. Ashley: It has a great effect on it but the idea of compensation is to increase the drive current to overcome the effect.

G. N. West (*IBM*): What limitation do you impose on temperature variation between write and read on a given core to eliminate distrub on half select?

Mr. Ashley: To maintain the accurate write and read current? That is in line with what has to be done for the inhibit drivers. The inhibit current actually varies. Of course, the full select write current is kept the same as the read because it is derived from the same voltage reference. The inhibit current can vary as much as 8 to 10 per cent on either side of the nominal value of the read for that temperature with no harmful effect. Actually this is really conservative also, because I think it could be 10 or 12 per cent without any great harm.

Use of a Computer to Design Character Recognition Logic

R. J. EVEY†

The System

THE IBM 1210 Sorter/Reader recognizes characters printed in a specified location on paper with magnetic ink.[1] A schematic diagram of the machine system is given in Fig. 1. The characters first come to a writing head which induces a magnetic field in the special purpose ink with which the characters are written. Next this magnetic field is sensed by a multi-channel reading head. The utput of the reading head is a set of ten time-dependent voltage waves.

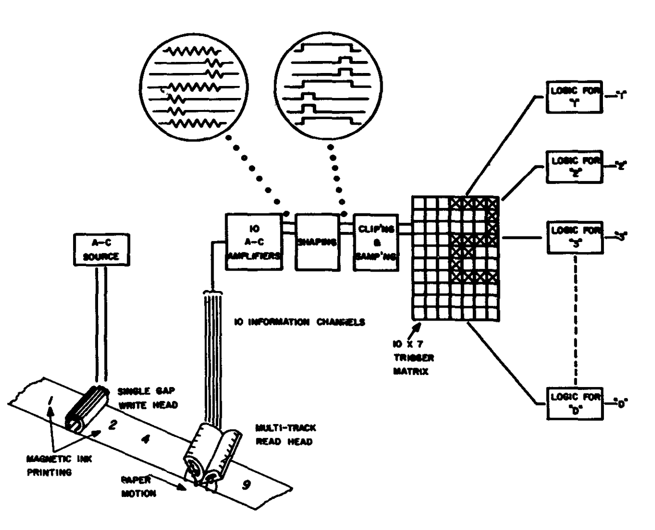


Fig. 1—System schematic.

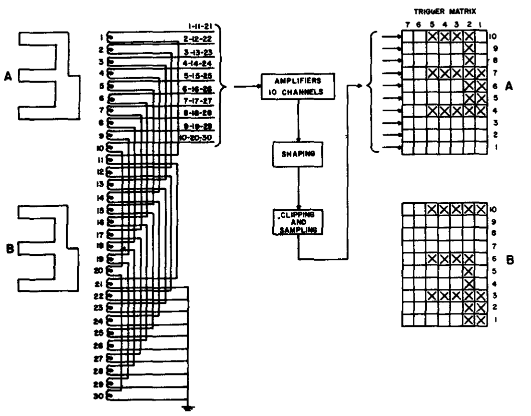


Fig. 2—Roll problem.

† International Business Machines Corporation, Poughkeepsie, New York.

[1] K. R. Eldredge, F. J. Kamphoefner, P. H. Wendt, "Automatic Input for Business Data Processing Systems", *Proceedings of Eastern Joint Computer Conference* (December 10–12, 1956), p. 69.

Actually (as Fig. 2 shows) there are thirty channels in the reading head. However, every tenth channel is "OR'ed" together (*e.g.*, 1-11-21, 2-12-22, etc.) so there are only ten outputs. These waveforms are time-sampled and changed into binary pulses by the quantizing circuits. The output of each quantizer is seven bits of binary information per character. The outputs of the ten quantizers (one per output channel of the reading head) are stored in a 10×7 trigger matrix.

The final section of the system is a set of 14 logical circuits (one for each character of the ABA alphabet[2]) made up of standard digital computer AND and OR components. These circuits are driven directly by the trigger matrix and operate in parallel. If a pattern in the trigger matrix satisfies any one of the logical circuits (called logics in the sequel), the corresponding character trigger is set. Recognition occurs if one and only one of these character triggers is set; otherwise the pattern is rejected.

It was mentioned previously that the thirty channels in the reading head are OR'ed together in groups of three. This means that the registration of the pattern in the matrix is unknown. So the system looks for recognition ten times per pattern; that is, it tries to recognize the pattern in the position in which it first appears in the matrix. Then the whole pattern is moved up one row at a time, with any bits in the top row being brought down into the bottom row. Thus each pattern really presents ten different patterns to the logics. Only after a pattern has "rolled" through all ten positions are the fourteen character-triggers examined for recognition or rejection.

This paper deals only with the design of the fourteen logics in this final part of the machine. It will attempt to make clear the problems which we tried to solve in this design and the methods we used to develop these circuits.

The Problem

The total number of different patterns possible in a 70-bit matrix is 2^{70}; and the total number of logics that can be designed for this input is $2^{2^{70}}$. The size of these numbers requires that some simplification be found to make the logical design tractable. Much of the required simplification lies in the two-dimensional

[2] Bank Management Commission: American Bankers Association, "The Common Machine Language for Mechanized Check Hand.ing", *Bank Management Publication 147, Automation of Bank Operating Procedure,* 12 East 36 Street, N. Y. (April, 1959).

correlation of bits in the matrix; that is, most of the logically possible patterns do not look anything like a possible character pattern. We found that a basic set of about 20 to 30 different patterns are obtained 90% of the time a given character is scanned. Almost all of the rest of the time a pattern is obtained which differs in one, two, or three bit positions from one of the patterns in the basic set. If these noisy bit positions are treated as don't-care positions, logical combinations of the common logical characteristics of the patterns in the basic set can be formed which will recognize virtually all the patterns obtained from scanning a character. Noisy bit positions for a given character account for over half the matrix, but this is not serious because four bits actually overdetermine the entire set of 14 characters.

The problem is thus reduced to that of finding the stable combinations of bits for a given character. At this point, however, we must consider the problem of registration — a problem which is present in all character recognition systems. Some are designed from the point of view that this is the major problem of character sensing and must be eliminated entirely; that is, an attempt is made to design the system so that once the first character is found there is no further problem occasioned by registration. In the 1210 system, however, even after the character has been scanned by the reading heads, the registration of the pattern in the matrix is unknown. A solution to the horizontal problem is the requirement that the leading edge of the character be located in the right-hand column of the matrix. The E13B font, with its strong leading edges, is designed for this.

The problem of vertical registration reduces to the "roll" problem, and the main problem here is that of cross-recognition. A degraded two, for example, may "roll" around to make a pretty good five (it should be noted that in the 1210 system this situation would result in a reject rather than a substitution, because both the "two" and "five" character triggers would be set). Part of the solution to this problem lies in the fact that the normal pattern is only eight rows high. Therefore, a condition which required at least one blank row at the top or at the bottom of the matrix was made a part of each logic. Once the pattern has been restricted so that it can move only a few rows vertically in the matrix and cannot roll completely around, the problem of design of the logics has been reduced to the required degree.

Solution

Theory

We assumed that the set of patterns to be recognized could be *approximated* by the union of two other sets of patterns which we could construct. The first of these would be the set of all admissible patterns assuming ideal printing and machine operation; that is, if the edges of characters were not ragged, there were no voids and no splatter, the magnetic field induced was uniform over the whole character, etc.

This set was generated by a program which we called the Theoretical Shape Program (TSP). Details of its operation can be found in Appendix 1. Letus say briefly here that the input to the program was a coding of each ABA character into binary bits. Each bit represented one square mil of ink. Hence, E13B characters, which are nominally 117 mils high and 91 mills wide, were entered into the 704 in the form of about 500 36-bit binary words (allowing for some blank border). This "micro-matrix" was then "scanned" by a program which simulated the operation of the reading head and quantizers. The output was a set of 10×7 "macromatrices" (*i.e.*, simply a set of patterns for each character) which were written directly onto 704 tape. The program assumed that registration, variations of magnetic density from character to character, timing across the character, fringing of the magnetic field, printing tolerances, etc. (see Appendix 1 for complete list of parameters), cannot be held firm. Hence, these "theoretical variables" were varied in the program and used to generate a set of different patterns for each character. This set was called the theoretical shapes.

We resorted to experiment to get a feel for the less systematic problems (such as voids). A hardware model of the scanning and quantizing part of the system was constructed and tied into an IBM 519 Reproducing Punch. This "print tester" scanned single characters from checks run at 1210 S/R speed and punched the resulting pattern into an IBM card. A small sample (about 10,000 checks per character) of printing chosen to cover the range of ABA printing specifications[3] was scanned and punched into one card per pattern. The resulting patterns (called "real life" shapes) were transferred from cards to tape and used to indicate the types of "noise" which might be expected to degrade the theoretical shapes.

We now had two sets of patterns (each stored on its own IBM 704 tape). Each of these was now reduced to a set which was composed of only the unique shapes of the original. These patterns were now examined by a second 704 program called the Logic Processing Program (LPP — see Appendix 2 for complete details). This program accepted, as input, logics (*i.e.*, logic statements) punched into cards in a "Boolean" notation. It interpreted each logic and stored it in core memory; then one pattern at a time was read from tape and tested against the logic. If a pattern which represented a two, say, were being tested against a logic which was supposed to recognize two's (self-test), and if the pattern was not

[3] Bank Management Commission, op. cit.

recognized by the "two" logic, but met a preset number of conditions (see Appendix 2), the pattern was printed. If it met the logic, that fact was simply noted in summary tables printed at the end of a run. If a two were being tested against a logic which was supposed to recognize, say, fives (cross-test), the criteria for printing the pattern or entering the tables were nearly the reverse of those for self-testing.

Method of Designing Logics

With these tools at hand, the following method was used to design the logics. A simple trial logic consisting of single black (1) or white (0) bits was tried against the set of theoretical shapes for that character (*i.e.*, a self-test was run against theoretical shapes). After several trials it is possible to determine a set of 10 to 15 positions consisting of single black bits inside the character outline and single white bits close to the character outline. It must be emphasized that it is always a *set* of "sure bits" which is found. For different criteria a different set will be found. For example, a program was written which determined the maximal set of sure bits for each theoretical character. However, in some cases, a more desirable set of sure bits would be one which distinguished sharply a given character from that character (or characters) which looked the most like it. These "sure bits" were then used as a trial logic for running a cross-test against the rest of the theoretical shapes. The result of this run would be a reduced set of "sure-bits", which were useful in telling this character apart from the other theoretical characters. Then these "useful sure-bits" were used as conditions for a trial logic for the given character.

First, this trial logic would be self-tested against corresponding real-life shapes. Samples of real-life shapes would not be recognized because of voids, ink-splatter, skew, etc. By examining the tabulations and patterns printed by LPP, the designer would attempt to modify the single-bit trial logic by OR'ing a more complex condition to the sure-bits which gave trouble. This new logic would again be real-life self-tested. After a number of trials, a logic would be obtained which would recognize all of the real-life shapes the designer felt were realistic. Then the logic was real-life cross-tested and modified using a similar procedure. Here, however, the criterion for final acceptance was that *no* character should be misrecognized by the logic (this was due, of course, to the system's more stringent requirements on substitutions than rejections). A flow-diagram of the above procedure is shown in Fig. 3.

Several modifications of each logic would usually have to be made at each step in the process before the logic would be considered satisfactory. Sometimes it was necessary to start from the very beginning with a search for a new set of useful sure-bits. In all cases a complete, transmissible record of the design of each

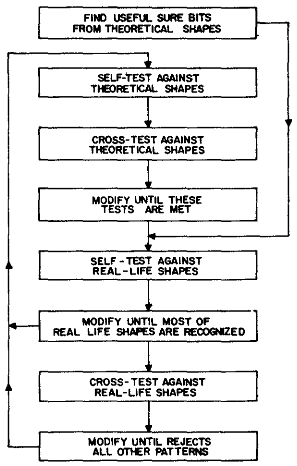

FIND USEFUL SURE BITS FROM THEORETICAL SHAPES
SELF-TEST AGAINST THEORETICAL SHAPES
CROSS-TEST AGAINST THEORETICAL SHAPES
MODIFY UNTIL THESE TESTS ARE MET
SELF-TEST AGAINST REAL-LIFE SHAPES
MODIFY UNTIL MOST OF REAL LIFE SHAPES ARE RECOGNIZED
CROSS-TEST AGAINST REAL-LIFE SHAPES
MODIFY UNTIL REJECTS ALL OTHER PATTERNS

Fig. 3—Statement writing procedure.

trial logic, the results obtained in testing it against the trial shapes, and the reasons for modification existed in the summaries kept by LPP.

Conclusion

Only two other methods of designing logics of this type are known to the author. One of these consists of building hardware which allows the engineer to shift wires in the model quickly (somewhat like IBM plugboards for EAM equipment). In this way logics can be wired directly into the machine and paper can be fed through an actual model of the system. This method has the advantage that the engineer knows the logics are trying to recognize patterns which are produced under field conditions. It has the great disadvantage that there are no records of patterns successfully recognized by the logic. When a change is made in logic wiring and a retest is run, the engineer has no way of knowing whether the same patterns as before are being presented to the logic. Hence, he has no assurance that he is really comparing the new logic against the old. The new logic may work better; but it may be because it is seeing more easily recognizable patterns. This method of designing logics has been tried at IBM and has not been as successful as the subject method in either time, cost of logics, or reliability. However, using the procedure described in this paper, a set of statements for each of the fourteen characters was developed with an expenditure of six man-months for the 704 programs (which are of an exceedingly general nature and have been used in whole or part for other applications) and two man-months for the design of the logics. Further, it was found that two different designers working independently on the same statement tended to produce logics that were equivalent in cost, performance, and the bit positions used (see Fig. 4). The best proof of the method, however, lies in the fact that the initial set of statements developed through its use have been wired into models of the

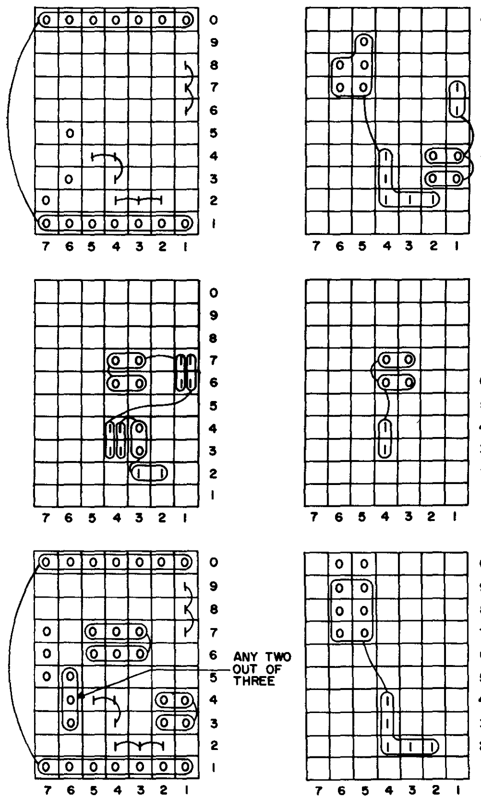


Fig. 4—Two "2" logics.

1210 S/R and have remained there unchanged after more than a year of rigorous testing.

Another method known to the author is that of devising an automatic procedure to design these logics. Most exhaustive procedures can be ruled out due to the astronomical number of possible logics, but useful procedures have been developed by limiting the complexity of the conditions used in the statements.[4] However, possibly because of this limitation, statements so produced have never been as successful in practice as those designed by people.

There was a time, nevertheless, when we felt that a definite short-coming of this method was that it was not automatic. In the many areas in which there is an attempt being made to utilize computers for the solution of complex decision problems (*e.g.*, theorem-proving, language translation, network analysis and synthesis, etc.), the goal is complete automation. However, this was not our goal. We needed a reliable set of logics and we were able to utilize the computer to advantage in completing this task. It processed a trial logic against a controlled set of input patterns. It ran tests and tabulated the results of this processing. Under a variety of sense-switch controls it displayed specific items of interest to the designer. Finally, it kept accurate records of this continuing iterative process of logic design, so that previous work could be re-examined. In this way the human beings in the process were freed from monotonous tasks and could devote their experience and creative judgment to the actual task of designing logics that recognize characters.

[4] P. H. Howard, "A Computer Method of Generating Recognition Logics for Printed Characters," *IBM Technical Note*, TN 00.10070.357 (May 5, 1959)

Acknowledgments

It would be impossible to mention all the people who have helped with this project. But at the risk of omitting equally worthy contributors, I would like to single out D. R. Andrews, G. E. Bartholomew, P. C. Murley, and L. O. Nippe, who wrote most of the programs; A. J. Atrubin and P. H. Howard, whose constructive criticism and helpful suggestions contributed both to the writing of the programs and to the design of the logics; E. C. Greanias and A. Hamburgen, who made significant contributions in the earlier stages of this project; and especially Dr. J. P. Lazarus, without whom neither the project nor this paper could have been completed.

Appendix 1 — Details of TSP

Input to the program consisted of two sets of IBM cards. The first set consisted of a coding of the character into one-mil squares. This was accomplished in this fashion:

A detail drawing of the character was blown up to 50 times life-size. A grid marked off in squares, which represented one square-mil to the same scale, was then laid over the character. Each coordinate on this grid was marked. Hence, a person could quickly see the coordinate where each row started into black and where it left. One card was then punched per row — with first the coordinate when black was started, when it was left, when it started again (if it did) and so on. Since each ABA character is 117 mils high nominally, this would result in 117 cards per character. A further coding was incorporated, however, in that where the edges of the character are not curving, one card may be the same as a preceding card. Hence, there is no need to repeat the next card; simply punch into the first card the number of times it is to be repeated. Fig. 5 gives the listing of the cards required to code the character 2. These cards were read by the program (actually they were put on tape and read from there) and interpreted into bits where there was black indicated in the character and blanks where the character was white.

The second set of cards (an example may be seen in Fig. 6) contained a complete set of the parameters which could be varied in the program. These parameters (and the card fields in which they were punched) were:

(1) The dimensions of the macromatrix (*i.e.*, the output matrix or trigger matrix of the S/R).

	IDENTIFICATION	ROW	NO. OF REPEATS	BLACK STARTS	BLACK STOPS	BLACK STARTS	ETC.
CARD COLS.	1,2	3-5	6-8	9-11	12-14	15-17	ETC.
	D2	117	001	005	048		
	D2	116	001	003	050		
	D2	114	002	002	051		
	D2	109	005	001	052		
	D2	107	002	001	051		
	D2	106	001	001	050		
	D2	105	001	001	048		
	D2	104	001	001	017		
	D2	103	001	001	015		
	D2	101	002	001	014		
	D2	070	031	001	013		
	D2	068	002	001	014		
	D2	067	001	001	015		
	D2	066	001	001	017		
	D2	065	001	001	048		
	D2	064	001	001	050		
	D2	062	002	001	051		
	D2	057	005	001	052		
	D2	055	002	002	052		
	D2	054	001	003	052		
	D2	053	001	005	052		
	D2	052	001	036	052		
	D2	051	001	038	052		
	D2	049	002	039	052		
	D2	018	031	040	052		
	D2	016	002	039	052		
	D2	015	001	038	052		
	D2	014	001	036	052		
	D2	013	001	005	052		
	D2	012	001	003	052		
	D2	010	002	002	052		
	D2	005	005	001	052		
	D2	003	002	002	051		
	D2	002	001	003	050		
	D2	001	001	005	048		

END OF DATA FOR GIVEN CHARACTER.

Fig. 5—Coding for Character 2 for TSP.

(2) The font (this would be varied by changing the first set of input cards).

(3) The width of a reading channel to the nearest mil.

(4) The width of the dead space between the channels to the nearest mil.

(5) The horizontal sampling interval in mils.

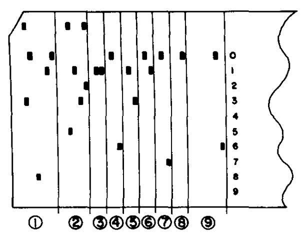
(6) The clipping level of the quantizing circuits (*i.e.*, the height of the voltage waveform they would have to see to call it above the noise level.)

(7) The integration time of the quantizing circuits.

(8) The initial registration of the character (that is, whether its leading edge were sensed too soon due to magnetic fringing or other effects, right on time, or late due to missing or low-density ink).

(9) Printing tolerance.

The program would first read a set of character coding input cards, interpret them, and position the coded character in storage in such a way that it simulated a character with its bottom edge on the bottom edge of a reading channel. Then a parameter card would be read and the character "scanned" in accordance with the parameters punched therein. The result of this "scan" would be a pattern (or macromatrix) which was written on tape immediately. Then the character would be "moved" (or "rolled")



FIELD	COLS
①	1-6
②	7-12
③	14-15
④	17-18
⑤	20-21
⑥	23-24
⑦	26-27
⑧	29-30
⑨	35-37

Fig. 6—Parameter card for TSP.

up one mil in its relation to the channel and land (dead space) and again scanned in accordance with the same set of parameters. This process would continue until the character had rolled up to the position in which its bottom edge just rested on the bottom edge of the next higher channel. At this point it is obvious that we would begin to see the same set of patterns all over again. So another parameter card would be read and this process repeated for that card. This would continue until all the parameter cards for a given character were read, at which point a new set of character coding cards for the next character would be read and the whole process repeated. This process is illustrated in the simplified flow-chart of the program shown in Fig. 7.

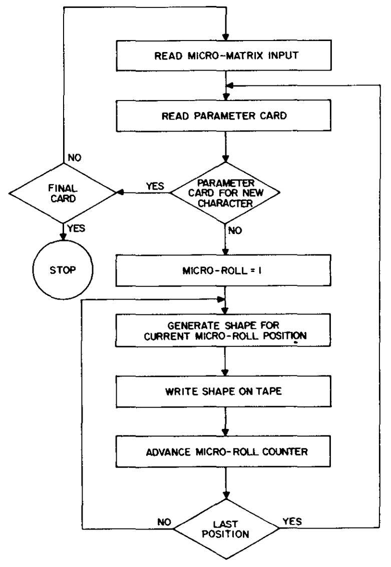


Fig. 7—TSP flow chart.

There was one parameter which does not appear in a parameter card. That is the system of quantizing used. This was varied by reprogramming. That part of the program was made into a closed subroutine and reprogrammed whenever the engineers changed their quantizing circuits. About five different types of quantizers were tried and they had so little in common we felt this was better than attempting a general program. It should be mentioned here that after the program was used a couple of times, it was so successful in simulating the scanning that the engineers would try a new idea for quantizing here before they would try it in hardware.

Appendix 2 — Details of LPP

The input to LPP consisted of two parts also. First, of course, was a set of cards into which was punched the logic to be tested. These were punched in this manner:

The character for which the logic was written was punched in column 1, (A, B, C, D being used for the four special symbols of the ABA alphabet). The number of conditions was punched in columns 2 and 3. A condition is a multistaged logical AND'ing and OR'ing of trigger matrix bits which, when AND'ed with other conditions, forms the logic for the given character. No assumption of minimal form is made, so that the same logic may be decomposed in different ways into conditions. For example, if A and B are two conditions, the total logic consists of $A \cdot B$ and 02 is the number of conditions. AB may be taken as a condition and the total logic then has one condition. Suppose $A = C + D$, then there are two conditions, $(C + D) \cdot B$; or the logic can be written $BC + BD$, which is only one condition. Hence reference is most easily made to a logic picture to see what was constituted as a condition. Fig. 8, which shows a simple logic and what would be punched into the logic card, may make this clear.

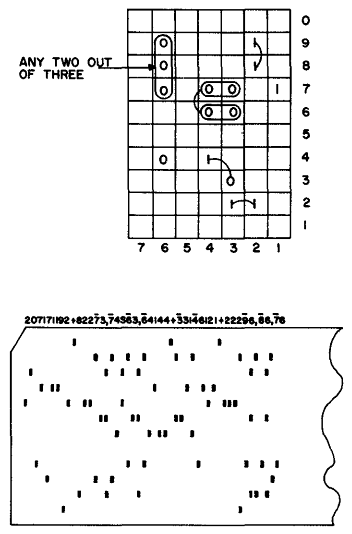


Fig. 8—Simple logic.

Starting in column 4, a cycle of symbol-row-column started and kept up until column 72 or until all the logic was punched. If the logic had to extend over to a second card the same sequence was used; that is, character, total number of conditions, symbol, row, column, etc., starting where one left off on the preceding card. The symbols used were numerals 1 to 9, "+" for OR, a "," comma for AND, and the letter "S", which also symbolized a logical OR but had a larger scope than the plus sign. The numerals indicated that a new condition was starting and told how many of the subconditions following it were to be satisfied (2 out of three for example). A subcondition is one bit specified by the row and column location. If the bit were to be a blank, a negative sign (X over-punch) was punched over the row.

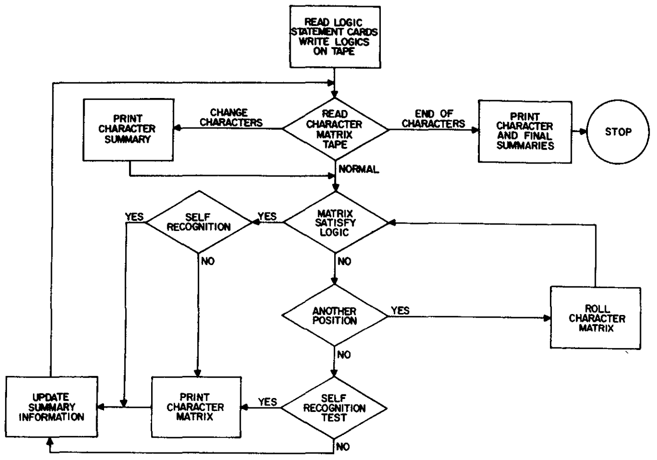


Fig. 9—Logic processing program.

```
FREQ. TABLE   2 LOGIC    CHAR. 2          490 NCR      MFC 486       CNML 2

                                           A
                 10   54   55   55   44    9
                 12`  86   93   96   99    8
                  0    9   10   39  100    7
                 .10  56   58   64  100    6
                 13   92   98   97   97    5
                 13   98    8    4    4    4
                 13   93   69   60   48    3
                 12   91  100  100  100    2
                                           1
  8     7     6    5    4    3    2    1
POS / FREQ.    1 /  486,
```

Fig. 10—Frequency table (FT).

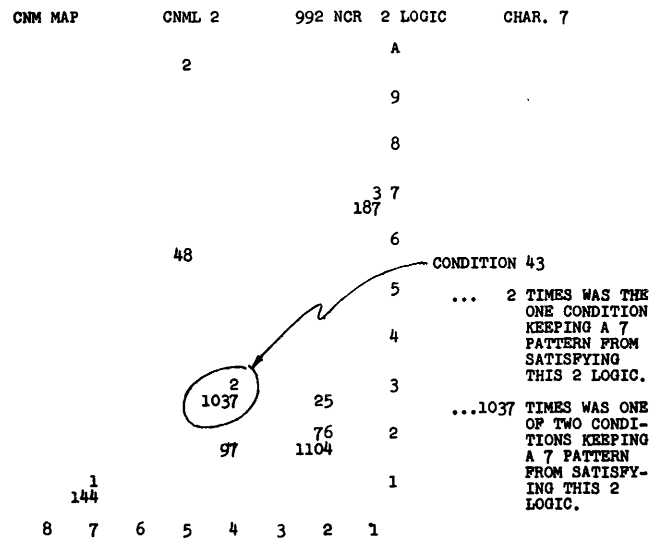
These cards were read by the program, interpreted, and stored in memory. (See Fig. 9 for a flow-chart of LPP.) Then the program read one character pattern (the second element of input) from tape. This pattern was tested against the logic. As we have said, if the character were being tested against its own logic and met all the conditions this was noted in a final summary table. Actually more was done with it. The whole pattern was added, a bit at a time, into a *frequency table* (Fig. 10). That is, this table kept track of how many times the characters had bits in each matrix position when considered in the roll position in which they were recognized by the logic. Now, if the pattern was not recognized, it was rolled through all ten roll positions, and the program kept track of the roll position in which it missed the fewest number of conditions (or the first position in case of a tie).

```
                                                     A
                                                     9
                                   1     1  1  1     8
2 LOGIC                                        1     7
CHAR. 2                                        1     6
CNMLXX                                1  1  1        5
1 CNM /       22                      1              4
POS    1                           1                 3
     27B515051528                     1  1  1        2
*             040                                    1
                         8  7  6   5  4  3  2  1
```

Fig. 11—Printed pattern (PRAT).

```
BPFT 1       2 LOGIC    CHAR. 2       1000 NCR     MFC      3       CNML 2
                                        A
                  100   66   66   66    9
             33    33   66   33  100    8
                                 100    7
                   33        66  100    6
                   33   66   66   66    5
                   33                   4
                   66   33   33         3
             33    33   66   33         2
                                        1
 8    7    6    5    4    3    2    1
```

Fig. 12—Best position frequency table (BPFT).



```
CNM MAP       CNML 2        992 NCR  2 LOGIC      CHAR. 7
                                      A
               2
                                      9
                                      8
                                  3   7
                                187
                                      6
              48                         CONDITION 43
                                      5       ...  2 TIMES WAS THE
                                                     ONE CONDITION
                                      4              KEEPING A 7
                                                     PATTERN FROM
                                                     SATISFYING
                 2                    3              THIS 2 LOGIC.
               1037          25
                             76       2       ...1037 TIMES WAS ONE
                 97        1104                      OF TWO CONDI-
        1                             1              TIONS KEEPING
       144                                           A 7 PATTERN
                                                     FROM SATISFY-
  8     7     6    5    4    3    2    1             ING THIS 2
                                                     LOGIC.
```

Fig. 13—Condition-not-met map (CNMM).

```
CNM TABLE          2 LOGIC   CARD TOTAL  6482
CHAR. 0   1    2    3    4    5    6    7     8    9    A    B    C    D
CNM

0             708                       1
1         1    5    5        17        48          1
2        60        209   7  186    5  265     3  118
CNML 2
```

Fig. 14—Summary.

Then the whole pattern would be printed out (these printouts were called *printed patterns*, or PRAT's — Fig. 11). Further, the pattern was added into a table called a *best position frequency table* (Fig. 12). Further, a table was made up of the conditions which were missed. These were called *condition-not-met-maps* (Fig. 13) and told the conditions which kept patterns from recognizing. If the pattern came within one condition of recognition, the count was printed on one line; but if it were two or more conditions from recognition, the count was printed on a different line.

As has been said previously, if a pattern were being tested against a logic for a different character all the above tables were entered but the criteria for entrance were simply reversed. Further, entrance was made in the tables for every roll position. In this way the CNMM (condition-not-met-maps) told what conditions were actually keeping characters from being recognized. All of these tables were printed at the end of each character run. Only a final *summary* table was printed at the end of the complete run (Fig. 14). This told for each character how many patterns came within 0 (*i.e.*, complete), 1, or 2 conditions of being met.

Complete control of entrance into each of the summary tables and printing of the summaries was maintained by using a combination of control cards and sense switches. The control cards specified whether or not a certain summary was to be kept and, if so, gave a limit of conditions. If a pattern missed recognition by more than this number of conditions, the summary table for that character was not entered. Then, as the program ran, the logic designer could choose to see certain tables (or even change the course of the program) by a selection of sense-switch settings. In this way the program displayed only that data the designer thought would be helpful at any given time.

A Self-Organizing Binary System*

RICHARD L. MATTSON†

Introduction

Any stimulus to a system such as described in this paper can be coded into a binary representation. A character on a piece of paper can be represented in binary form by dividing the paper into many small squares and assigning a 1 to a square if the character covers that square, and a 0 to the square if the character does not cover it.

A voltage waveform can be coded into a binary representation by quantizing the waveform in both time and amplitude, and assigning a binary code to each amplitude. In general, any data that are measurable can be coded into a binary representation with no loss of information. One form of an interesting logical system would be one which transformed this block of binary data into a single binary decision. For example, such a system would take the binary representation of a character and transform it into a binary decision (either it is an "A" or it is not an "A"). Combinations of such devices could then be used to make decisions of a higher order than binary.

The conventional method of designing such a system is to list in a table all possible combinations of the binary inputs, and to assign the most probable output value (decision) to each input combination. This table of combinations with output values is then reduced to a Boolean function which represents the method by which the decision is to be determined. This procedure will always yield the best possible technique of making decisions.

However, this synthesis procedure has two major disadvantages. First, the Boolean function is not easily obtained when the number of binary variables at the input is large. Second, even if the Boolean function were determined, any change in the decision process would require a repetition of the difficult synthesis procedure in order to arrive at a new Boolean function.

It is the purpose of this paper to define a model for a self-organizing logical system which determines, by an iterative trial-and-error procedure, the proper Boolean function for a process. This self-organizing logical system is composed of two separate systems, one a network of adjustable logical devices, the other a system for determining what logical function should be assigned to each device in the network. Such a system is depicted schematically in Fig. 1.

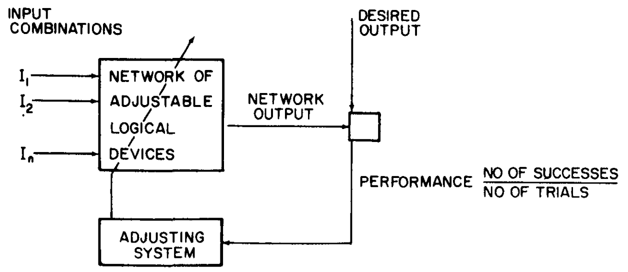

INPUT COMBINATIONS
I_1
I_2
I_n
NETWORK OF ADJUSTABLE LOGICAL DEVICES
NETWORK OUTPUT
DESIRED OUTPUT
PERFORMANCE $\frac{\text{NO OF SUCCESSES}}{\text{NO OF TRIALS}}$
ADJUSTING SYSTEM

Fig. 1—Self-organizing logical system.

The adjusting system makes a sequence of adjustments in the network, each adjustment being determined by the performance of the network for the previous adjustments, so that with the final adjustment the network realizes the desired Boolean function.

An Adjustable Logical Device

A logical device that can realize any one of many different logical functions by adjusting its internal parameter values is shown schematically in Fig. 2.

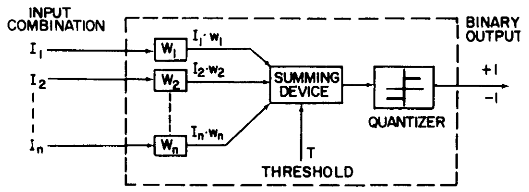


Fig. 2—The adjustable logical device.

The inputs to this device are binary, $+1$ and -1. The weights, w_i, are continuous real variables that can assume either positive or negative values. The threshold, T, is also a continuous real variable and can be either positive or negative. The output of this device is $+1$ if

$$T + \sum_{i=}^{n} I_i w_i > 0 \qquad (1)$$

and -1 if

$$T + \sum_{i=}^{n} I_i w_i < 0 \qquad (2)$$

The dividing condition between the output being $+1$ or -1 is given by

$$T + \sum_{i=}^{n} I_i w_i = 0 \qquad (3)$$

* Most of the work described in this report, including computer simulation studies, was done at the Digital Systems and Components Laboratory and the Computation Center, Massachusetts Institute of Technology, under Navy BuShips Contract NObsr 72716. The work was completed at Lockheed Missiles and Space Division, Logical Design, as a part of the Lockheed General Research Program.

† Missile Systems Division, Lockheed Aircraft Corporation, Sunnyvale, Calif.

A geometric representation of the logical properties of this device can be made by using the equation of the dividing condition (Equation 3), and a binary n-cube representation of the input space of the device. For example, with three inputs to the model, the dividing condition is

$$T + I_1w_1 + I_2w_2 + I_3w_3 = 0$$

By plotting this plane in an I_1, I_2, I_3 space, and indicating various input combinations by points in the space, the logical operation of the device can be interpreted geometrically. That is, all points on one side of the plane are mapped into a $+1$ by the device, and all points on the other side are mapped into a -1. A plot of the input space is shown in Fig. 3. In an n-dimensional input space the dividing condition specifies a hyperplane which separates the n-dimensional input space into two parts. It can easily be shown that the w_i values control the slope of the hyperplane in the space, and the threshold, T, controls the position of the plane in the space.

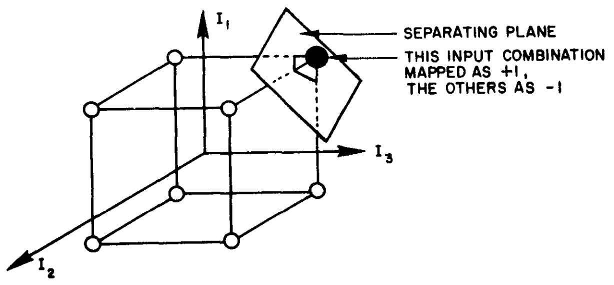


Fig. 3—Three-variable input space.

By changing the w_i values and T, different mapping functions can be obtained. For example, with this device one can obtain 14 different mapping functions for two-input variables, 104 for three-input variables, and approximately 1900 for four-input variables. These different mapping functions are termed "linearly separated" truth functions[2]. The idea of passing a hyperplane through an n-dimensional input space is important because it allows one to visualize types of functions that can be generated by the logical device. It is also useful for determining the types of processes that the model would be best suited to classify. For example, since the hyperplane divides the input space into two regions, each of which is a region of points unit-distant apart, the logical device would be useful in classifying processes in which one group of unit-distant points is separated from another group of unit-distant points. An example of such a process is a single-error correcting code where a legal n-bit word, and all combinations unit-distant from this word, are to be mapped as the legal word. Another possible use would be for character recognition where a legal character is to be separated from all other characters. For this application each character and its variations must be representable as a group of closely connected points in the input space, and each of these groups must be separable from all the other groups. When this condition is met, a single logical device would be an effective character classifier.

Another useful capability of the device is that of assigning output values to input combinations that have never been given to the device. For example, if a small number of input combinations have been given to the logical device and a hyperplane is placed in the input space so as to map these combinations correctly, then every point in the input space is automatically assigned an output mapping by this plane. This allows a mapping criterion to be established without being exposed to all possible input combinations. In the case of character recognition, where the number of possible variations of a legal character is so large that it becomes impossible to list all of them, a hyperplane could be situated in the input space on the basis of a partial listing of variations, and the mapping of the other variations would automatically be determined.

By interconnecting devices of this type, each one realizing a different linearly separated function, and feeding their outputs into an AND or OR device, any logical function can be obtained. This is easily shown since a single device of this type can realize the AND or OR function of n-variables and any Boolean function can be written as a sum of products or product of sums.

A network of adjustable logical devices can be constructed by using as a basic element the device shown in Fig. 2. By adjusting various weights and thresholds in the network any logical function can be realized. The remaining portion of this paper describes methods of determining what sequence of network adjustments should be made to obtain convergence to a desired Boolean function.

Analysis of the Performance of a Single Logical Device

Consider the input space of an n-variable logical device. There will be 2^n distinct points in this input space, each one corresponding to a particular input combination of the n-variables. Let these combinations be denoted as $C_1, C_2, \ldots, C_k \ldots, C_{2^n}$. Each of these input combinations has a probability, $P(C_k)$, of occuring at the input of the logical device. For each input combination the desired output has a probability of being $+1$, $P(D_k = +1 \mid C_k)$, and a probability of being -1, $P(D_k = -1 \mid C_k)$. If the input combination C_k is mapped as a $+1$ by the device, then the probability that the output of the device will agree with the desired output is

$$P_k(A) = P(D_k = +1 \mid C_k) \tag{4}$$

If C_k is mapped as a -1 by the ,device, the probability of agreement between the desired output and the output of the logical device is

$$P_k(A) = P(D_k = -1 \mid C_k) \quad (5)$$

Since C_k must be mapped either as a $+1$ or a -1, then one of the two above equations is valid for the probability of agreement for a fixed logical device, and the other is not. Define a variable x_k so that $x_k = 1$ when C_k is mapped as a $+1$ and is *zero* otherwise, and a variable x_k which is $+1$ when C_k is mapped as a -1 and is *zero* otherwise. The probability of agreement for input C_k can then be written as

$$P_k(A) = x_kP(D_k = +1 \mid C_k) + \bar{x}_kP(D_k = -1 \mid C_k) \quad (6)$$

The total probability of agreement between the output of the logical device and the desired output can be obtained by summing the individual performances of all the input combinations.

$$P(A) = \sum_{k=1}^{2n} [x_kP(D_k = +1 \mid C_k) + \bar{x}_kP(D_k = -1 \mid C_k)] \quad (7)$$

The performance of the logical device is measured by the number of agreements divided by the number of trials. The expected value of this measurement is given by $P(A)$, and thus $P(A)$ is the expected performance of the logical device.

Earlier it was pointed out that the weights in the logical device control the slope of the hyperplane in the input space of the model, while the threshold controls the location of the plane in that space. Thus, with the threshold set at an extremely negative value the hyperplane will lie outside the binary n-cube in that space, and all points in the input space will be mapped as -1. As T is increased in value the hyperplane will pass through the binary n-cube, changing the mapping of the points in space to $+1$ until T is extremely positive and all points are mapped as $a + 1$. With T extremely negative all points in the input space are mapped as -1 and Equation 7 becomes

$$P(A) = \sum_{k=1}^{2n} P(D_k = -1 \mid C_k) \quad (8)$$

If the index k is arranged so that the sequence $k = 1, 2, 3, \ldots, 2^n$ corresponds to the first, second, $\ldots$, 2^nth point in the input space to have its mapping changed to $+1$ as T is increased, then the performance will have the sequence of values.

$$P_0(A) = \sum_{k=1}^{2n} P(D_k = -1 \mid C_k) \quad \text{for } T \text{ extremely negative}$$

$$P_1(A) = P(D_1 = +1 \mid C_1) + \sum_{k=2}^{2n} P(D_k = -1 \mid C_k)$$

increasing T $\downarrow$

$$P_2(A) = \sum_{k=1}^{2} P(D_k = +1 \mid C_k) + \sum_{k=3}^{2n} P(D_k = -1 \mid C_k)$$

$$\vdots$$

$$P_{2n}(A) = \sum_{k=1}^{2n} P(D_k = +1 \mid C_k) \text{ for } T \text{ extremely positive}$$

as T is increased from negative to positive. In addition, if $P_0(A) < P_1(A)$, then $P(D_1 = -1 \mid C_1) < P(D_1 = +1 \mid C_1)$ and C_1 should be mapped as $a + 1$. By comparing $P_1(A)$ with $P_2(A)$ etc., it can be shown that if the performance continuously increases as T is increased, points in the input space are continuing to be mapped correctly as $+1$. Whenever the performance decreases for increased T, that point should not be mapped as $+1$. From this result it is possible to prove that if $P(A)$ increases monotonically to a maximum (peak) value and decreases monotonically thereafter, then the mapping function corresponding to T set at the peak is the best possible mapping function for that particular input space. This is easily shown since, for T set at the peak, the performance is

$$P(A) = \sum_{K=1}^{i} P(D_k = +1 \mid C_k) + \sum_{K=i+1}^{2n} P(D_k = -1 \mid C_k)$$

and for each $1 \leq k \leq i$

$$P(D_k = +1 \mid C_k) > P(D_k = -1 \mid C_k)$$

and for each $i + 1 \leq k \leq 2_n$

$$P(D_k = -1 \mid C_k) > P(D_k = +1 \mid C_k)$$

By using similar arguments it can be shown that if there exist m groups of $+1$ points in the input space that are separated from each other by a group of -1 points, there will be at least m peaks on the performance curve. A sample performance curve vs. threshold for $m = 1$ is shown in Fig. 4.

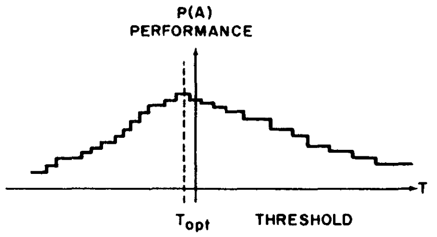


Fig. 4—Performance curves with one peak.

Equation 7 gave the performance of the logical device in terms of the performance for each input combination. The index k was associated with the sequence of points that change their mapping function as T is increased. When a given weight, w_i, is changed, the hyperplane will rotate and another sequence of points will change their mapping function. Thus, the same results that applied for the threshold can also be applied for each weight of the device, and a curve of the performance as a function of w_i can be plotted. Again it can be shown that as the performance increases a group of points in the input space is being correctly mapped, and as soon as the performance decreases a point is being incorrectly mapped by the change in parameter of the device.

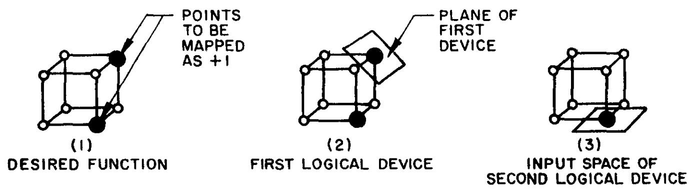

POINTS TO BE MAPPED AS +1
PLANE OF FIRST DEVICE
(1) DESIRED FUNCTION
(2) FIRST LOGICAL DEVICE
(3) INPUT SPACE OF SECOND LOGICAL DEVICE

Fig. 5—Synthesis with two logical devices.

The addition of other logical devices can be accomplished by letting each one adapt to a group of separated points in the input space. For example, consider the input space shown below. Here, since there are separated groups in the input space, a single logical device cannot realize the desired function. However, a single logical device can realize the function shown in Fig. 5-(2). If the performance curve of this logical device is plotted against its threshold it will have two peaks. One peak will correspond to the function shown in Fig. 5-(2) and this logical device should be adjusted to realize that function. Every time the first logical device has an output of $+1$ the desired output of the second logical device should be -1. This makes the input space of the second logical device be as shown in Fig. 5-(3). Clearly this is a linearly separated function and can be realized by the second logical device. By feeding the outputs of these two devices into an OR function, the desired function is obtained. It can be shown that the most difficult logical function to mechanize in this manner is the alternate symmetric or even-odd functions. In this case it requires 2^{n-1} logical devices all feeding into an OR gate to realize the function.

An Adjustment Procedure

Having related the performance of the device to the change in parameter values, an adjustment procedure can be devised for the adjusting system. Briefly, the procedure is to start with a single logical device and make a guess at initial values for weights in the device. It has been demonstrated experimentally and justified mathematically that good results are obtained if each weight is initially set to the value of the cross-correlation between the particular input and the desired output.[1] This quantity is computed from the equation

$$w_k = \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} I_i D_j P(I_i, D_j) = P(I_k = D) - P(I_k = D)$$

For certain processes this initial setting is not the best one to make; however, it is good for the types of processes considered in the demonstrations. Then vary the threshold from extremely negative to less negative values and measure the performance for each value of T. If the performance increases, increase T and measure the performance again. If the performance decreases adjust the threshold back to the peak. Then increase the weights one at a time until a peak is obtained for each weight. The threshold and weights should be re-adjusted until any change in any parameter causes a decrease in performance. At this point the hyperplane has a group of points on one side that are correctly mapped, and all points closest to the other side of the plane require the opposite mapping. Then T should be increased to an extremely positive value and a measure of the performance recorded for each value of T. If the performance monotonically decreases for increasing T then the single logical device with T set at the peak realizes the desired truth function for the process. If the performance curve has another peak as T is increased, then a single logical device cannot realize the desired function, and other logical devices must be added.

Because the adjusting system obtains only a measure of the $P(A)$ function, the measured performance may deviate from the theoretical performance. When this happens the performance curve will not look like the one shown in Fig. 4, but rather will have many small peaks and valleys as shown in Fig. 6. On this measured performance curve a slight decrease in the measured performance may be due to sample-size effects of measuring. Therefore confidence limits must be established by the adjusting system to determine if the performance has actually decreased, or if the decrease is due to measurement errors. Thus, statements about the performance of the logical system must be modified to include confidence limits. As an example, with 95 percent confidence the performance curve has only one peak and therefore with 95 percent confidence the logical system is realizing the best possible function for this process.

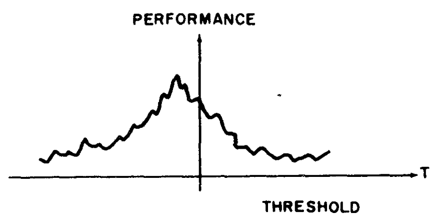


Fig. 6—Measured performance curve.

The process of determining a peak on the performance curve can be speeded up by assuming a slowly changing curve and by selectively sampling the curve. Consider the curve shown in Fig. 7. This curve is first sampled at $n + 1$ equally-spaced values of T. Because the curve increases monotonically to a peak and then decreases, one peak must be between the first and third sampling. The curve is then sampled n times in this region. The peak must now be between the second and sixth samples so the curve is sampled n times in this region, and this process is repeated until the peak has been determined. This process can be shown to converge geometrically to the peak.

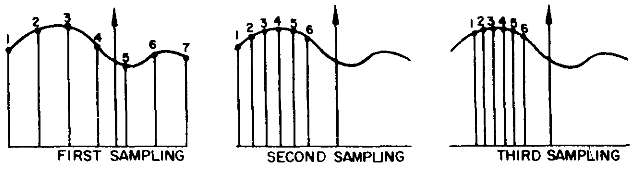


Fig. 7—Successive sampling of the performance curve.

Pattern Recognition Demonstration [3,4]

The devices and the adjusting system were simulated on the IBM 704 computer and four demonstrations of the self-organizing logical system were made. Three of these were character-recognition problems and one was waveform recognition. In the character-recognition problems the network of logical devices was to be adjusted until it realized an optimum truth function for distinguishing one group of characters from another group in the presence of noise. The initial settings of the weights was the cross-correlation and in each demonstration this was a setting which gave optimum performance. In each case the noise-free characters were arranged in a fixed $m \times n$ array of squares just large enough to contain the characters. Noise was introduced into the characters by selecting each of the squares in the array, one at a time; the representation of each square was changed with a probability, p, and was left unchanged with a probability, $1 - p$.

Demonstration 1: Recognize 5 and S; 25 Binary Variables

In this problem the system was required to distinguish a 5 and a shifted 5 from an S and a shifted S in the presence of noise. The noise-free and noisy characters are shown in Fig. 8.

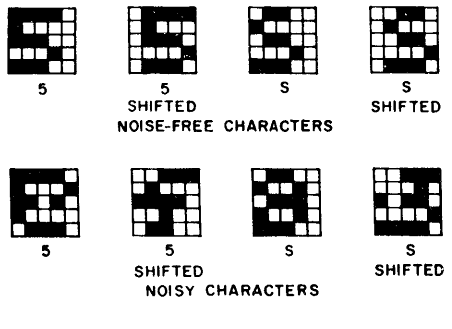


Fig. 8—Characters in Demonstration 1.

With 1.5 minutes of computer time, 2192 independent samples of noisy characters, and a single logical device the 5's were distinguished from the S's with 98 percent recognition. Because of the noise this was the best possible theoretical performance for this process. The performance curves were sampled three times to determine the best setting of the threshold. These curves are shown in Fig. 9. Since the adjusting system used a confidence limit of 99.5 percent, it is 99.5 percent certain that the self-organizing system was realizing the best Boolean function for this process.

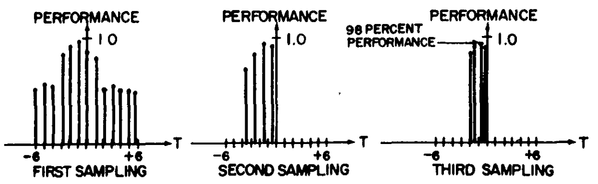


Fig. 9—Sampling of performance curve, Demonstration 1.

Demonstration 2: Recognize 5, S and 8; 49 Variables

In this problem the system was required to distinguish a 5 from an S and an 8 arranged in a 7×7 array. Noise was introduced in the same fashion as in Demonstration 1. The noise-free and noisy characters are shown in Fig. 10.

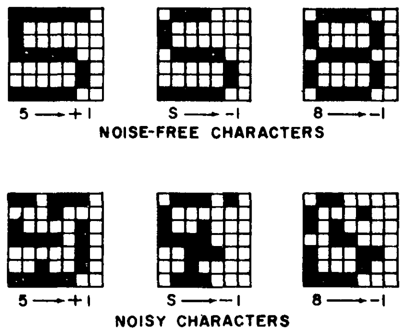


Fig. 10—Characters in Demonstration 2.

It required 1.6 minutes of computer time, 1337 samples of noisy characters, and one 49-input logical device to realize an optimum truth function for this recognition problem. The sampled performance curves for this process are shown in Fig. 11, and again there was 99.5 percent certainty of realizing the best possible Boolean function.

Demonstration 3: Recognize OF, AT, TO, and IN; 50 Variables

In this problem the system was required to distinguish the word OF from the words AT, TO, and IN displayed in a 10×5 array. Noise was introduced in the same manner as before. The noise-free and noisy characters are shown in Fig. 12.

It required 1.1 minutes, 718 samples of noisy characters, and a single 50-input logical device to realize

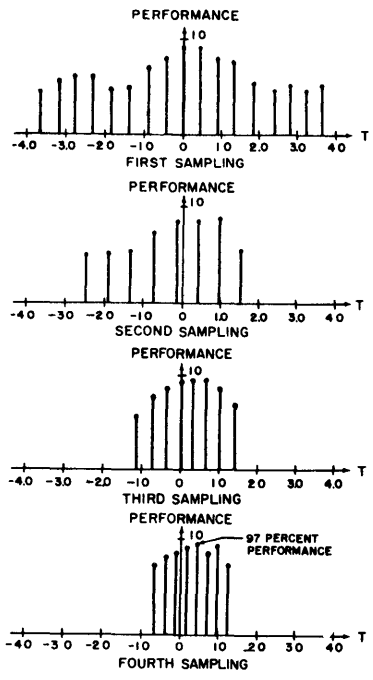


Fig. 11—Sampling of performance curve, Demonstration 2.

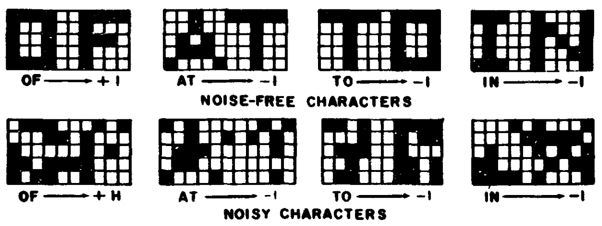


Fig. 12—Characters in Demonstration 3.

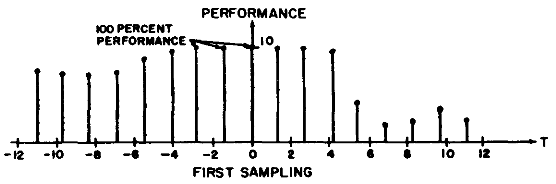


Fig. 13—Sampling of performance curve, Demonstration 3.

the optimum function (100 percent recognition) for this problem. The single sampled performance curve is shown in Fig. 13.

Demonstration 4: Recognition of Waveforms

The fourth demonstration problem for the self-organizing logical system was to distinguish one waveform from a similar waveform in the presence of noise. The two waveforms were quantized into 36 discrete instants of time and the amplitude at each instant was coded into a 6-bit binary code. Thus, the binary representation of a waveform required the combination of 216 binary bits. Noise was introduced so that the greatest amount of noise appeared in the least significant bits of the amplitude code, and the least amount of noise appeared in the most significant bits of the amplitude code. Examples of noise-free and noisy waveforms are shown in Fig. 14.

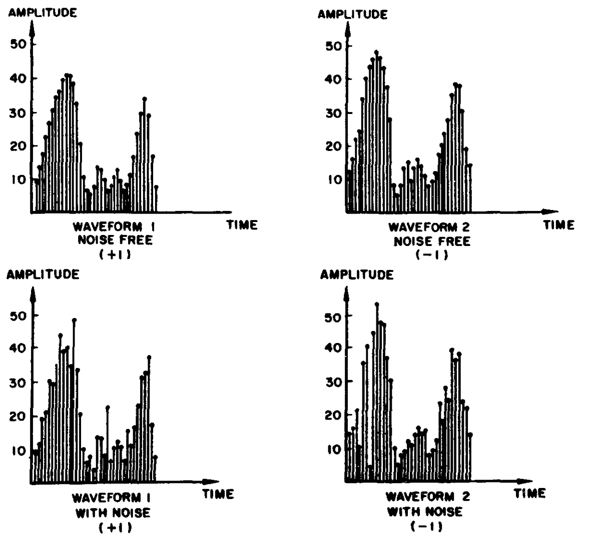


Fig. 14—Waveforms, Demonstration 4.

The self-organizing logical system required 0.9 minutes, 987 samples of noisy waveforms, and a single 216-input logical device to distinguish the waveforms with 100 percent recognition. A sampling of the performance curve is shown in Fig. 15.

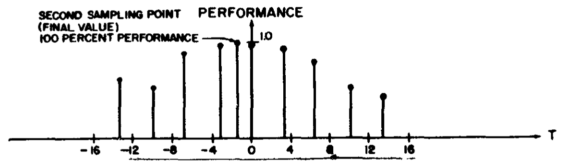


Fig. 15—Sampling of performance curve, Demonstration 4.

Conclusions

From the results obtained in the simulation studies, it appears that this technique of self-organization allows solution of many practical pattern-recognition problems. These problems could take a variety of forms, from visual recognition to speech recognition, and more complicated data reduction problems such as binary predicting and filtering in the presence of noise. The time and amount of data required for the system to "adapt" to a given arbitrary task seem to be within practical limits. It is not known what the equipment requirements would be for a general self-organizing system, and a variety of more diffiicult pattern-recognition problems are needed to determine the versatility of a "self-organizing" logical system requiring many logical devices.

References

[1] R. L. Mattson, *The Design and Analysis of an Adaptive System for Statitiscal Classification*, S. M. Thesis, M.I.T., Course VI, June 1959

[2] Applied Mathematics and Statistical Laboratory, *Unate Truth Functions*, by Robert McNaughton, Technical Report No. 4, Stanford University, Oct. 21, 1957.

[3] Computer Components and Systems Laboratory, *An Adaptive Classifier*, by R. L. Mattson, Quarterly Progress Report No. 6, Cambridge, M.I.T., March 1959.

[4] Computer Components and Systems Laboratory, *An Adaptive Classifier*, by R. L. Mattson, Quarterly Progress Report No. 7, Cambridge, M.I.T., June 1959.

Alpha-Numeric Character Recognition Using Local Operations

J. S. BOMBA†

1.0 Introduction

THIS PAPER describes a demonstration of the recognition of thirty-four alpha-numeric characters. The IBM 704 EDPM was used as a tool to study the method which led to this demonstration. The Generalized Scanner[1] was used as an input transducer for this study.

The method of character recognition which was used here is to extract from the character its "essential" features and then recognize it from these features. For this study such features as horizontal, vertical, and slant straight lines, and intersections of lines have been used.

These features have been extracted by means of local operations. A local operation is a transformation which produces a small section of a new pattern, or field, from data in a corresponding small section of the original pattern (Fig. 1). An entire pattern is transformed by considering *all* of the small sections, where each section is considered independently. Here, the pattern area was divided into a sixty by ninety array of bits (*i.e.* either black or white spots) in order to quantize the visual impression of a character. A typical local operation section would consist of fifteen bits.

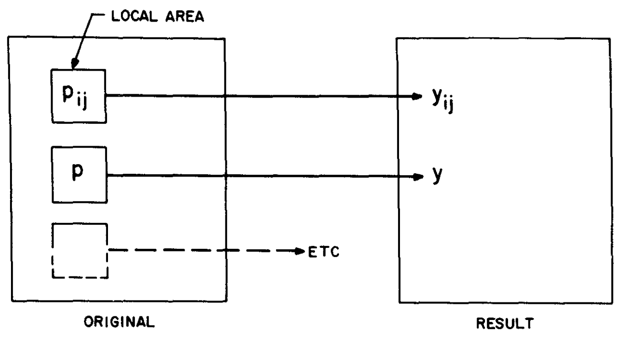


Fig. 1—Illustrated definition of a local operation. A "local area" is a configuration of spots not necessarily square as drawn which is examined by the program. The position p_{ij} of the local area is defined as the position of one particular spot in the configuration; e.g., point p in figure 8 and y_{ij} is the spot which is made into a "1" if the spots found in the local area indicate the presence of a specific feature. Both p_{ij} and y_{ij} have the same coordinates (i, j). Since p_{ij} scans all points in the original patterns, the resulting pattern must contain the same number of points as does the original.

In order to extract a specified feature, the whole pattern is processed and the resulting pattern then is blank unless the desired feature was present in the original pattern.

† Bell Telephone Laboratories, Incorporated, Murray Hill, New Jersey.

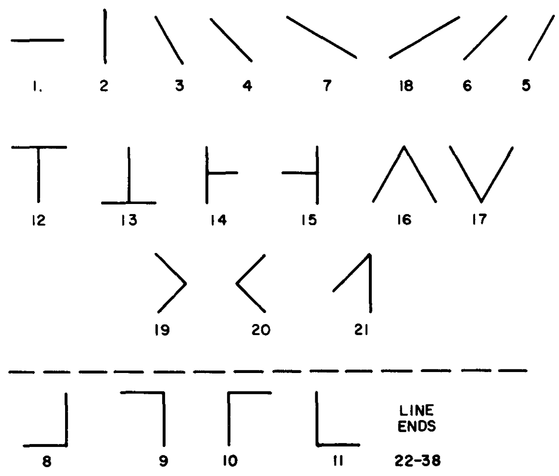


Fig. 2—Features extracted by Feastract. Only features above the dashed line were used for recognition. The numbers are to identify the features for Fig. 12.

The procedure for recognition is as follows. *First,* the characters are processed by a program which reduces the noise in the field by the method of local averaging[2]. In this method, the value of the majority of spots in a three by three rectangular local area dictates the value of the center spot in the new pattern. This process is more effective if it is repeated twice. *Second,* the line width is standardized. Since the matrix granularity is such that a typical line trace as written is always greater than four matrix elements thick, a line-width standardizing operation can extract the middle four matrix locations. *Third,* the features as shown in Fig. 2 are extracted. These are:

(a) straight lines which are horizontal and vertical, and slant straight lines which are at ± 45 degrees, ± 30 degrees, and ± 60 degrees from vertical:
(b) all four orientations of *T*- and *L*-intersections, and
(c) selected orientations of *V*-intersections.

The local areas (as defined in Fig. 1) which are used to extract these features, have the same shape as the desired feature. There are seventeen different features for which the original pattern is examined. In effect, a complete pass is made through the pattern for each feature and whenever a local area in the pattern matches the interrogation local area, a mark is made in a corresponding output matrix. This is done for each of the seventeen different features, thus generat-

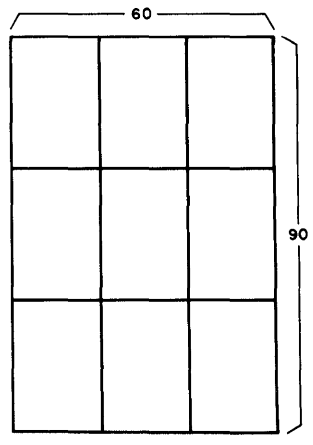


Fig. 3—Division of pattern field. The field consists of a 60 x 90 array of bits.

ing seventeen "new pattern fields." *Fourth,* these fields are divided into nine equal rectangular areas (shown in Fig. 3). The prevalence of spots in certain areas is used to indicate the general position of the feature (*i.e.* top or bottom) and its significance (*i.e.* whether a straight line is long or short). *Fifth,* the recognition is done from the detection of the presence of features. In most cases, it was only necessary to ask whether a given feature was extracted. However, with the limited variety of feature types which were used in evaluating the method, it was necessary to ask further questions about a few features such as horizontal and vertical straight lines. The identification was done with combinational logic. The logic was extended to allow some variation in character styles.

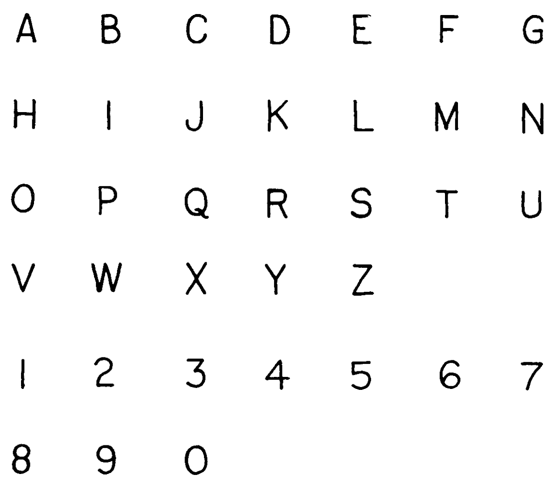


Fig. 4—Typical style of hand printed block capital alphanumerics.

Hand printed, block capital letters and numbers of the style shown in Fig. 4 can be recognized by this method. These are the letters "A" through "Z" and the numberics "1" through "0"; a "one" and "I", and "zero" and "0", are not separately distinguished. The characters may vary from full to one-half size vertically and horizontally. The line widths may vary horizontally and vertically, from a minimum average of four matrix elements to a maximum average of ten matrix elements. The characters must roughly be centered when they are less than two-thirds full size, and must be reasonably free from tilt.

2.0 Input Methods

The visual impressions have been transferred into the 704 by a manual and a machine method, both of which are described below (see 2.2 and 2.3).

The visual image is first quantized by dividing the pattern field into square elements and giving each element a value of "0" or "1" depending on whether the field is light or dark at that point. In this study the character lines were dark on a light background, and these character lines are represented by the bit locations which are ones.

The image, which then consists of "0"s and "1"s, is stored in the computer in this binary matrix form.

Each 60 element row of the matrix is split in half and each half is stored in the least 30 significant bits of a 704 core word. The rows are in 90 consecutive pairs of storage locations.

2.01 Matrix Size

A 60×90, 5400-element matrix was chosen because:

(1) It is large enough so that quantizing errors should be negligible.

(2) It is large enough so that the edges of the pattern need not be covered by the local operation if the character is more or less centered (thus, it was not necessary to program the special cases which occur when the local area is only partially present as would happen at the field edge. As you will see later in the discussion of feature extraction (see Fig. 10), since the largest local area used extends 7 spots from the local area center, the effective size of the matrix is thus 46×76.)

(3) Its width will just fit the printer which we use with the 704, (119 available type wheels are used to print the pattern with every other one printing a blank. The blanks are printed to limit the distortion of the field which the printer introduces because (a) in the original quantization the matrix elements are square whereas (b) on the printer the elements are rectangles whose height to width ratio is 6 to 5 when the blanks are printed and 6 to 10 when they are not).

2.2 Manual Procedure

In order to test the programs and to try sample characters before the Generalized Scanner[1] was finished, test patterns and characters were made up on IBM cards.

For test characters, first the true size characters were drawn on translucent paper. They were then made into viewgraphs which were enlarged by a slide projector. The enlarged image was focused on a 60 × 90 rectilinear grid 1.50 × 2.25 feet. Thus, each ¼″ × ¼″ element could be marked if more than half its area were black. IBM cards were then punched from this grid. One card per row was used with "ones" punched to correspond to black elements and nothing punched for white elements. Special program test patterns were made by marking the large grid.

2.3 Machine Methods

With the advent of the Generalized Scanner it became possible to write the characters on opaque paper and transcribe the scan results onto magnetic tape which can be fed to the 704. This machine method makes it possible to quickly and easily process a large number of characters.

3.0 Character Preparation by Local Operations

3.1 Noise Reduction

In order to allow as inputs to the feature extraction program, characters which have missing spots within the line of the character, extra spots in the field outside the line of the character, and fluctuations along the edges of the character lines, the character can be processed by a program which performs local averaging[2]. This program uses a local area, Fig. 5, which is 3 × 3 rectangle. The criteria which are applied for this local area are: (1) if there are five or more "1"s in the nine possible spots and if the center spot was a zero, then the center spot is changed to a one. (2) If there are five or more "zeroes" and if the center spot was a one then the center spot is changed to a zero. (3) In all other cases, the center spot is retained as it was in the original pattern. This process works most satisfactorily if it is repeated two or three times. That is, the original pattern is processed, then the result of the first processing is used as an input to the second process, etc. An example of the results of reducing the noise by this method can be seen by comparing a typical original pattern, Fig. 6, and the resulting pattern (processed twice), Fig. 7.

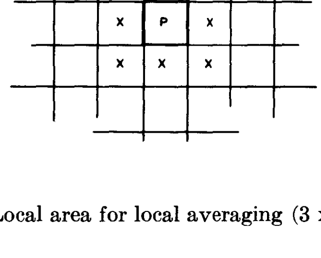


Fig. 5—Local area for local averaging (3 x 3 array).

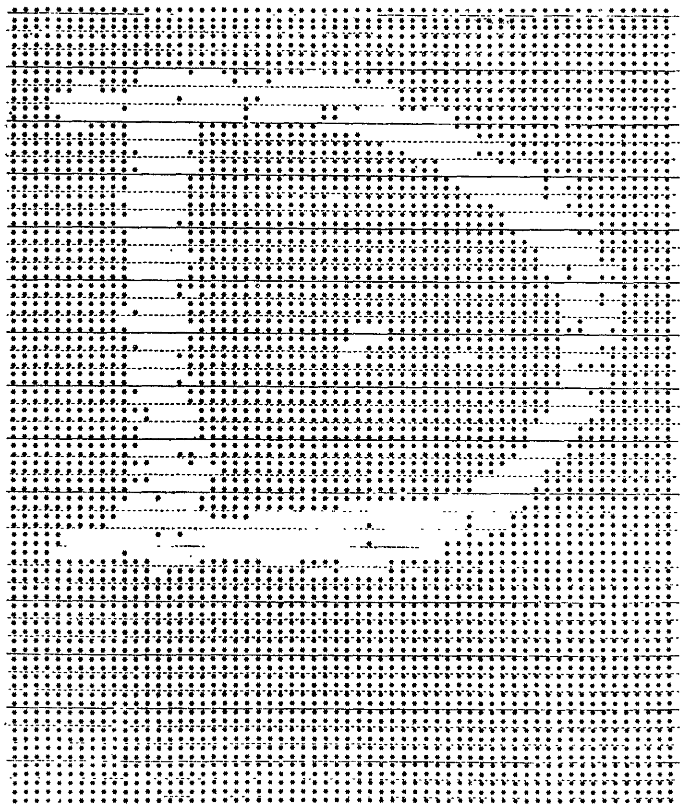


Fig. 6—Unprocessed character "D".

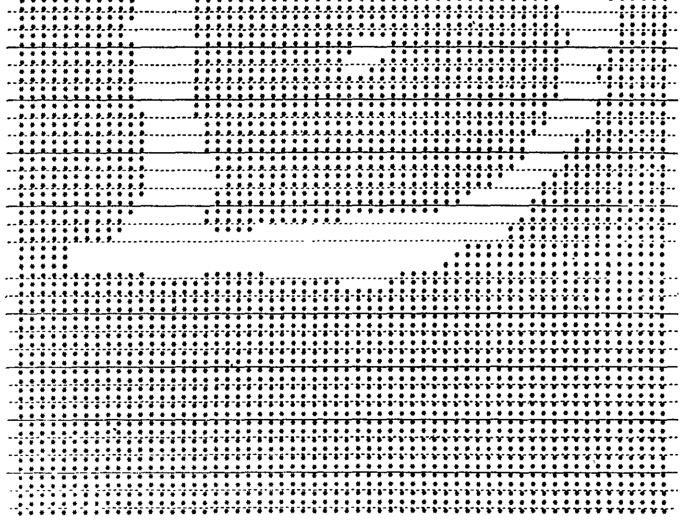


Fig. 7—Character "D" after local averaging twice.

Other criteria for the 3 × 3 local area and a 5 × 5 local area with a variety of criteria were tested. The 3 × 3 local area and the criteria of the previous paragraph were found to be the most satisfactory of the possibilities which were studied.

3.2 Line Width Standardization

The feature extraction program works best when the characters have a uniform line width. Therefore, a program was written to process characters with line widths which vary from 4 to 10 spots and produce characters with standard line widths. This program will change the character lines to a uniform average width of approximately 4 spots. The local area which is used by this program is shown in Fig. 8.

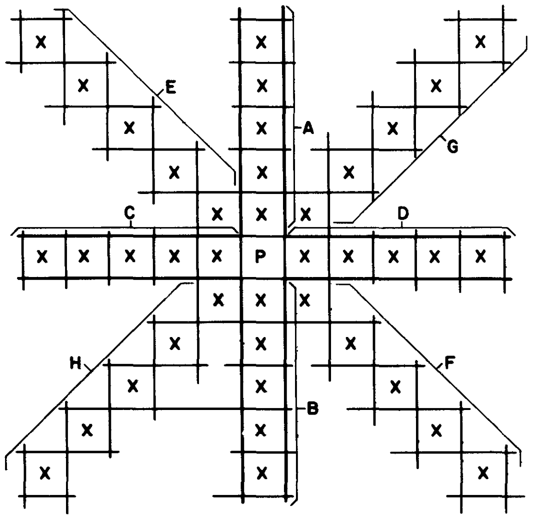


Fig. 8—Local area for line width standardization-thinning (11 x 11 array).

The criteria for this operation are that (1) a spot will be a one only if the original spot were a one and the difference between areas A and B and between areas C and D in the local area was less than 4,,or (2) the difference between the number of spots in the areas E and F and the areas G and H are less than 4. An example of "thinning" the previously "denoised" characters is shown in Fig. 9.

This program might be improved if we made the actual radial distances in the local area all equal. (It is obvious, Fig. 8, that five elements at 45 degrees extend further from the center than do five elements in a rectilinear direction).

4.0 Feature Extraction

The crux of this method of character recognition are the local operations which are performed by a program which is called Feastract. This program extracts the features which are shown in Fig. 2. Because this program was designed to study the method, it actually extracts more features than are used by the recognition procedure. The most useful features are the long straight lines: horizontal, vertical, and slanted.

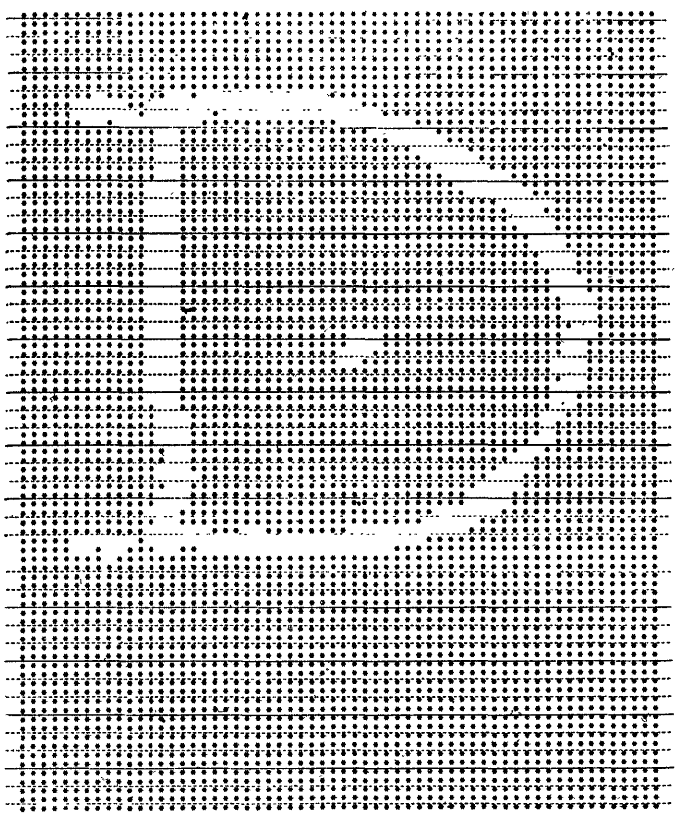


Fig. 9—Character "D" after local averaging twice *and* thinning.

Fig. 2 indicates which of the features are used and which are not.

4.1 Local Area

The local area which is used for the feature extraction is shown in Fig. 10. This local area is actually a combination of a number of local areas, each of which might be used for a different local operation to

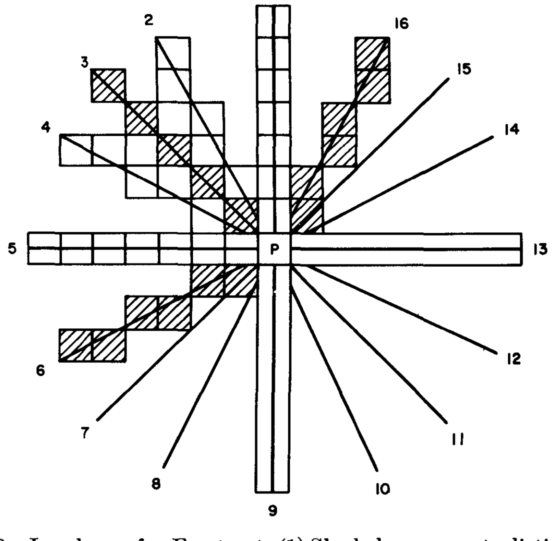


Fig. 10—Local area for Feastract. (1) Shaded areas are to distinquish radii. (2) Matrix elements are only drawn in completely for one quadrant. (3) The local area for a single feature would consist of selected radii. Thus, for a horizontal line, radii 5 and 13 plus *p* would be the local area.

extract a different feature. However, for purposes of study, it was convenient to arrange a local area in the manner shown, a radial pattern which enabled me to vary both the criterion for extraction and the shapes of the features which are found to be most suitable.

It should be noted here that this local area, as applied in this program, is used for local operations; however, if the number of points at which the local area is used is limited to fewer than all points in the overall pattern, then this local area falls into the class of feature extraction methods which can be called radial or polar scan methods[3]. There is one significant difference, however, between the polar scans and this scan. That is, each radius in my local area is required to contain "a" spots (where "a" = the number of spots in the entire radius, or nearly all spots to allow for some noise) whereas in the polar scans each radius is only considered from the viewpoint of crossing a line. That is if any radius contains any spot, it is considered to have crossed the line for the polar scans.

4.2 Effective Operation of Feastract Program

In order to clearly present the feature extraction process, the details of the Feastract program will be omitted. Thus, we will discuss the feature extraction process.

Let us consider a specific feature as an example, namely an L-intersection. The local area for an L-intersection consists of radii 1 and 13 and the spot "p" as shown in Fig. 10. The spot "p" with its associated spots is moved over the pattern field in a scanning manner. Whenever all spots in this local area are found to be "ones", a one is written in a buffer image for the L-feature at the same coordinates which "p" has. Since the original pattern is not changed during the above process, the entire operation could, of course, be done in parallel.

The same process can now be done for each feature which we wish to extract.

The result is a group of buffer images, one for each feature. Fig. 11 illustrates schematically some buffer images which would be produced from the original pattern. Each buffer image contains one feature.

After all of the spots in the original pattern have been examined and the results of the feature extraction process stored in various pattern buffer images, the program proceeds to examine each of these buffer images, and it determines whether or not there were any spots corresponding to the given feature in each image. If a spot is found in a buffer image, the program determines in which of nine areas the spot lies, as shown in Fig. 3. A series of counters are used to remember how many spots lie in each area of the field for the feature under question at the time. When the buffer image for a given feature has been completely scanned and the number of spots in each area of the field have been recorded in the counters, then the result is transferred to the recognition program and also printed out. The basic program was designed so that it would handle up to 50 features.

Note that this feature extraction program does not consider the feature "closed loops" (as encountered in a well-formed B). Although closed loops can be extracted with local operations[4], they are not a necessary feature for recognition of the symbols which were used here.

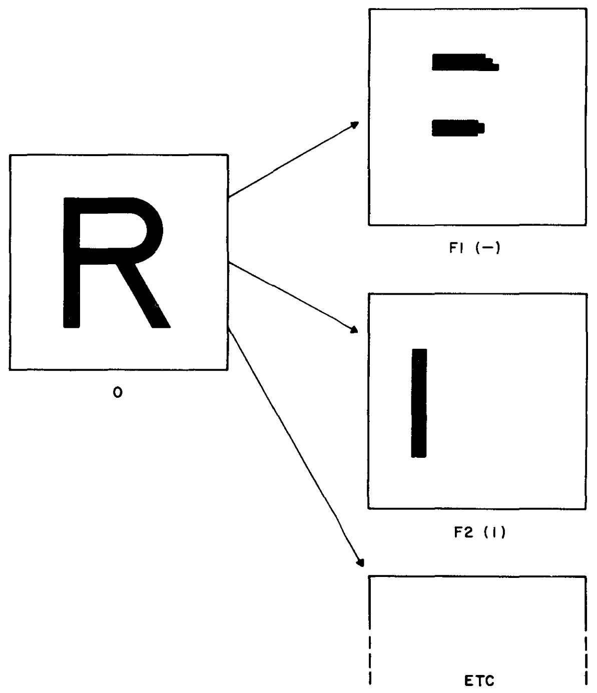


Fig. 11—Original pattern (0) and buffer images (F1, F2, etc.) which result from feature extraction process.

4.3 Output from Feastract

Typical information which is transferred from Feastract to the recognition program is shown in the following list for one feature. This list consists of — the identifying feature number and — the number of spots for the given feature which were found in each area of the field (Fig. 3).

	Register Contents
Feature Number	2
Top Right	27
Middle Right	78
Bottom Right	24
Top Middle	0
Middle Middle	0
Bottom Middle	0
Top Left	30
Middle Left	59
Bottom Left	16

These numbers resulted from the extraction of vertical lines (Feature Number = 2) for an H.

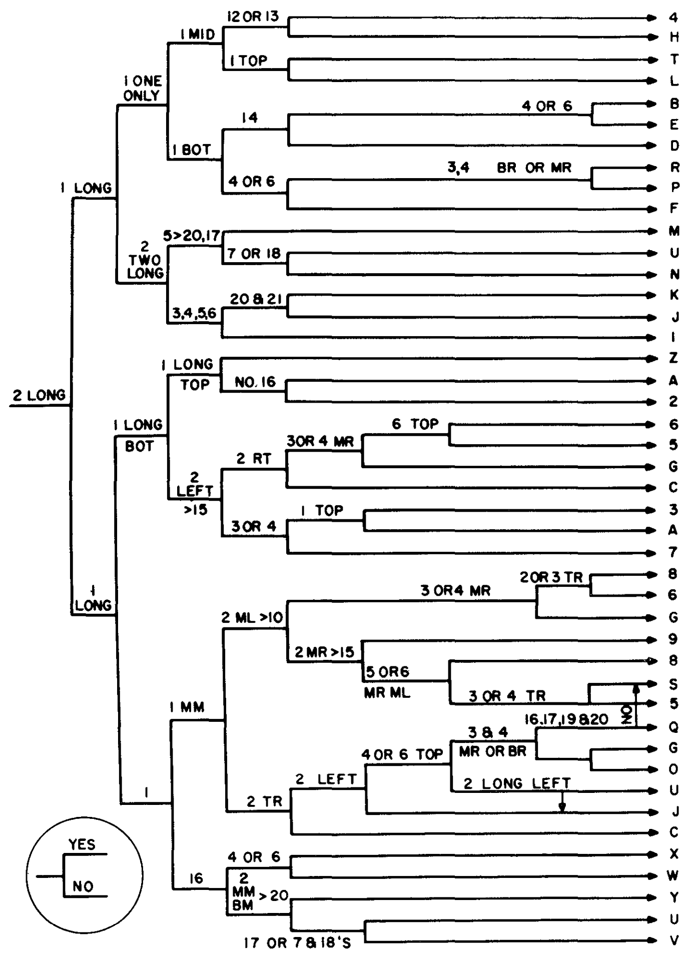


Fig. 12—Recognition tree — logic diagram. The feature numbers at each branch point are identified in Fig. 2. At each branch point the upper branch is followed if the feature is present.

5.0 Recognition from Features

Recognition is accomplished by combinational logic with the features as the input variables. Since there is a two-way choice corresponding to the presence or absence of each variable, the logic takes the form of a logical tree (Fig. 12). However, before the actual recognition takes place, certain preprocessing can be done on the features and their locations.

5.1 Preprocessing

A series of subroutines were written to take the information from Feastract and to partially interpret this information to answer such questions as, "how long is a horizontal line"? The detailed specification of said question would be, "does a horizontal line cross three areas in a horizontal row in the field"?

We can now consider more details of the answer to the above question. A general statement of the criteria for answering the question is: "Horizontal lines are considered to be long, if all three areas of the field in a horizontal row each contain more than two marks which were identified from horizontal feature sections." In other words, first, all those areas with only two marks indicated in the new or transformed pattern are eliminated. Second, a test is made to see if there are horizontal features in the three top areas. The same thing is repeated for the three middle areas and then for the three bottom areas. The results are stored in registers called TOP, MIDDL and BOTOM. Third, it is then possible to say that if the TOP register has the number three in it, there is a long horizontal line at the top. When there is a number one in it, then there is a short horizontal segment in the top. Thus, the number and length of horizontal lines can be determined for characters which have the restrictions as indicated in Section 1.0.

The same procedure which was used for horizontal lines can be applied to vertical lines. In this case, the information is stored in registers called LEFT, MIDLE and RIGHT.

There is also a subroutine to determine if there is, for instance, only one horizontal line in the field. The same procedure could be applied to slant lines. However, for this particular recognition tree, it was not found to be necessary to apply this type of preprocessing to other than horizontal and vertical lines.

5.2 Recognition

As indicated in Section 5.0, the recognition proceeds as a series of binary choices. The majority of tests which were used to make these choices are the following two:

(1) The presence of a feature is determined by testing the computer memory location of the feature number for non-zero. (See Section 4.3).

(2) The presence of a feature in one of the nine field areas is likewise determined, except that, in some cases, a minimum number of spots is required in the test. The other tests which were used are discussed in Section 5.1.

When a terminal branch of the recognition tree is reached, the identified character is printed.

It is interesting to note in Fig. 12 that a great many more characters result from all of the tree branches than just the 34 which we initially started with.

We thus have evidence from this tested recognition tree that, even with the restricted input patterns which we have used, redundancy has to be included in the recognition logic. Some subtle variations in characters have only become apparent when they were encountered in testing this recognition logic.

6.0 Extensions of This Character Recognition Approach

This character recognition approach, which I have described, has been used to actually recognize characters in order to show that it works. Here, in this feasibility demonstration, only a few of the features which can be extracted with local operations have been used. It seems clear that if the local area of Feastract were used for other features which it can extract (ends, curved segments), many of the restrictions which have been here (Section 1) imposed on the characters could be removed. Along this same line, if other local operations, such as concavities as described by Unger,[4] were included, the field would not have to be divided and thus larger variations in the characters could be admitted.

Unfortunately, there does not seem to be any good orderly way to specify the most desirable features or to specify the local area or local operation to extract a given feature. Fortunately we can use the 704 to test possibilities without the long process of building hardware. Nevertheless this is still a trial and error process.

Also, the logic for the recognition for this demonstration allows only limited variations in the characters. Although to the human being, characters of a wide variety appear to be the same, very often these characters are actually quite different from their feature viewpoint. Seriphs, slanted characters, and decorations tend to make the character into a different symbol. Thus, in an identification procedure which utilizes features, whether they be the normal features as specified by a human being or some special parameters which are not normally recognized by human beings, these various forms must be treated as separate symbols. The identification procedure could be lengthened to allow other forms of characters to be recognized.

Finally, unless self-organizing systems rapidly become a lot more sophisticated than they are now, any machine which we build to recognize large alphabets of characters of unspecified style will probably use a combination of several recognition methods, along with feedback to make more and more complicated tests until there is a high enough probability that there is a correct identification or a rejection.

7.0 Conclusions

It has been explained how features may be detected to recognize certain isolated alpha-numeric characters by using the local operations mentioned above.

Three sample alphabets were tested on an IBM 704, and the characters in these alphabets were separately recognized. With this limited amount of test data, it was felt that any attempt to specify allowable signal-to-noise ratio, character distortion (such as tilt) or a recognition error-rate would not be meaningful.

Others[4,5] have shown the power of using other local operations for this same purpose. Only a few of the many possible local operations have been investigated, and it seems likely that this approach will be useful not only for character detection but also in other areas of pattern recognition.

8.0 References

[1] W. H. Highleyman and L. A. Kamentsky "A Generalized Scanner for Pattern and Character Recognition Studies", PROC. W. J. C. C., pp. 291-294, March 1959.

[2] G. P. Dineen "Programming Pattern Recognition", PROC. W. J. C. C., pp. 94-100, 1955.

[3] T. L. Dimond "Devices for Reading Handwritten Characters", PROC. E. J. C. C., pp. 232-237, 1957.

[4] S. H. Unger "Pattern Detection and Recognition", PROC. I. R. E., vol. 47, pp. 1737-1752, October 1959.

[5] R. L. Grimsdale, et al, "A System for the Automatic Recognition of Patterns", PROC. I. E. E., vol. 106, Part B., No. 26, pp. 210-221, March 1959.

Pattern Recognition and Reading by Machine

W. W. BLEDSOE† AND I. BROWNING†

Introduction

MANY EFFORTS have been made to discriminate, categorize, and quantitate patterns, and to reduce them into a usable machine language. The results have ordinarily been methods or devices with a high degree of specificity. For example, some devices require a special type font; others can read only one type font; still others require magnetic ink.

We have an interest in decision-making circuits with the following qualities: (1) measurable high reliability in decision making, (2) either a high or a low reliability input, and (3) possibly low reliability components. The high specificity of the devices and methods mentioned above was felt to be a drawback for our purposes. All of these approaches prove upon inspection to center upon analysis of the specific characteristics of patterns into parts, followed by a synthesis of the whole from the parts. In these studies, pattern recognition of the whole, that is, Gestalt recognition, was chosen as a more fruitful avenue of approach and as a satisfactory problem for the initial phases of the over-all study.

In addition, we chose to concentrate upon the recognition of alphanumeric patterns, rather than upon other pattern types, for the following reasons:

(1) *Convenience.* Results can be handled easily since it is possible to use conventional print-out equipment. Furthermore, we could exploit our own familiarity with letters and words.

(2) *Background.* Research on alphanumeric pattern recognition has been vigorously pursued, and we were therefore able to make use of the relatively large literature on the subject.

(3) *Usefulness.* Success in our efforts would make available a technique which society needs and can use immediately, even though such a result would be only a by-product of our over-all study.

Because typewritten numbers were recognized without error in the cases considered, the investigation quickly shifted to hand-blocked print and finally handwritten script characters as displaying greater complexity and increasing individual variability. In this way ,the decision making powers of the system were more fully challenged.

Since a numerical output is the inherent mode of expression of a digital computer, our work was aimed at developing a numerical score for each pattern examined. The basic method employed to obtain these scores and to use them to identify each pattern uniquely will be described in the following section. Then various expansions and variations of the method will be covered. Finally, a method of extending identification by contextual relationships will be described briefly.

It may be mentioned at this point that this system is highly general — that is:

(1) It handles all kinds of patterns with equal facility.

(2) Because it does not depend upon absolute pattern-matching, it can identify a pattern which is not exactly like, but only similar to, a pattern it has previously learned.

(3) It does not depend significantly upon the location of a pattern on the photomosaic for identification.

(4) It is only partially dependent upon the orientation and magnitude of a pattern for identification.

It would also be well to mention the two major disadvantages of the system:

(1) When the learned patterns are quite variable, the memory can be saturated, especially in certain cases.

(2) A very coarse mosaic, especially if it has inconstant photocell performance, produces images of small letters which do not contain enough information for recognition. See, for example, the sixth character, an *e*, in Fig. 2b. The large letters, however, do not present this problem.

However, both of these disadvantages can be at least partially overcome; the first, by various techniques to be described later; the second, by using a mosaic with more photocells.

Basic Method

Of prime importance in this method is the way in which pattern discrimination is provided. The best way to describe the process is by example.

We start with a 10 × 15 photocell mosaic (this size being chosen because of immediate availability), the elements of which are related to one another as 75 randomly chosen, exclusive pairs. Fig. 1a shows the mosaic and two such randomly chosen pairs ($1_1 1_2$ and $2_1 2_2$). Images, letters for example, projected on the

† Sandia Corporation, Albuquerque, New Mexico

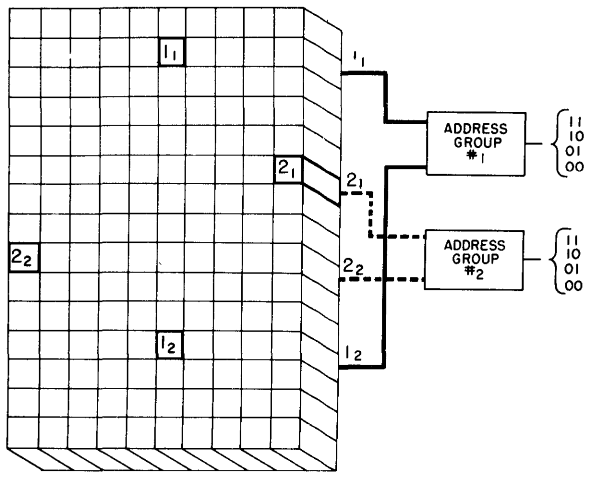


Fig. 1(a)—The photomosiac and two of the randomly chosen photocell pairs. The four digital groups to the right are the four possible states of each photocell pair.

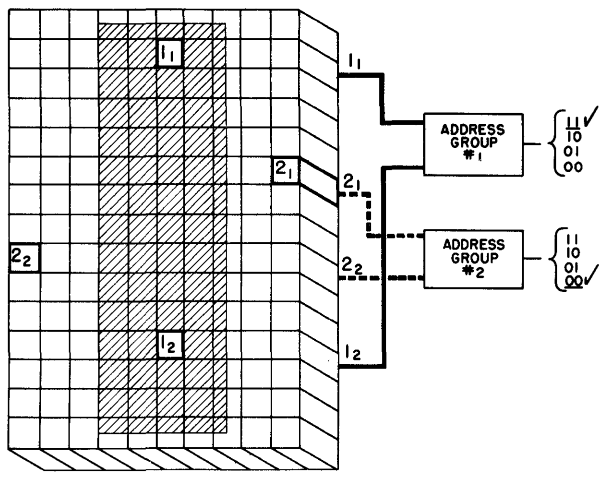


Fig. 1(b)—The system learning the letter I in a central position. Only two of the 75 pairs are shown.

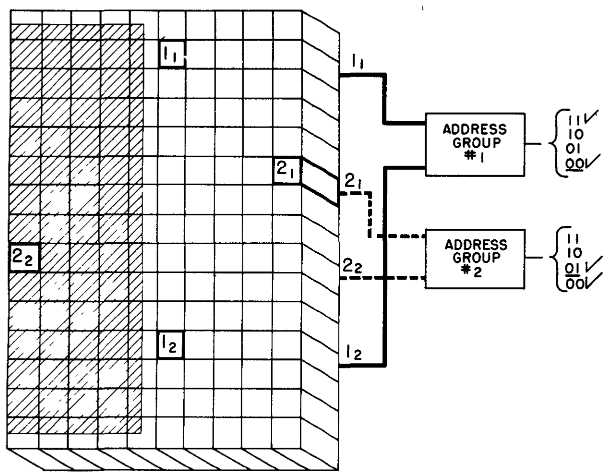


Fig. 1(c)—The system learning the letter I in another position. Note that the memory experience shown in the previous figure remains.

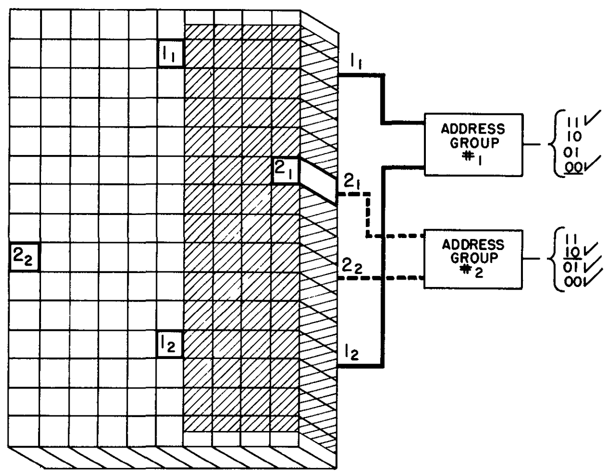


Fig. 1(d)—The system learning the letter I in a third position. The check marks to the right show all possible combinations of these two photocell pairs for the letter I.

mosaic will produce characteristic patterns, examples of which are shown in Figs. 2a and 2b as they appear on IBM cards. For computer convenience, the light values of an image on the mosaic are rendered in a binary system which treats dark as 1 and light as 0. When an image is on the mosaic, each pair of photocells (the members of which are ordered for this purpose) will represent the light values of the image as a two-bit number. Each pair of photocells has therefore four possible states — 00, 01, 10, and 11.

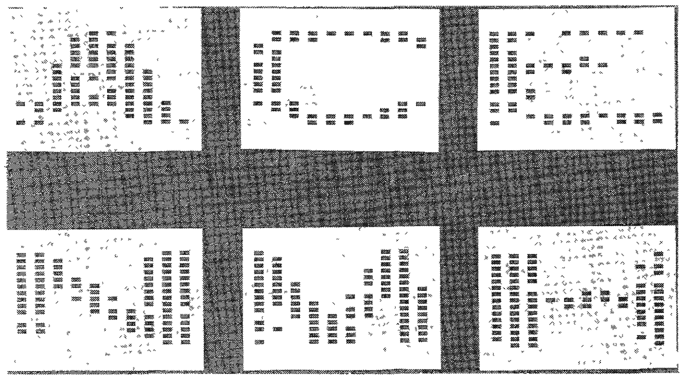


Fig. 2(a)—Hand-block print as it appears on IBM cards. (Top—A, C, E; Bottom—N, M, H.)

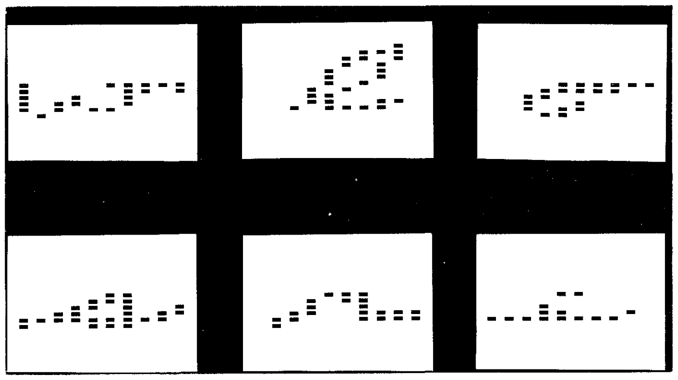


Fig. 2(b)—Handwritten script characters as they appear on IBM cards. (Top — w, l, o; Bottom — s, r, e.)

In the memory matrix of the computer, a 36-bit computer word is assigned to each state of each pair, giving four words for each photocell pair or 300 computer words for the 75 pairs. Furthermore, each bit position in the 36-bit computer words is assigned a pattern nomenclature. The sequence used in our experiment was:

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15...	35	36
Nomenclature	•	1	2	3	4	5	6	7	8	9	a	b	c	d	e...	y	z

This nomenclature sequence will hereafter be referred to as an "alphabet."

In order to demonstrate how patterns are "learned," we will use as an example the letter *I*. First, a letter *I* is projected on the photocell mosaic (Fig. 1b). Its image on the mosaic produces one of the pair states (00, 10, 01, 11) for each of the pairs, depending upon the amount of light falling on the pair. Since all 75 pairs are involved, the resulting 75 states address 75 words in the memory matrix. For each word addressed, a binary 1 is entered in the nineteenth position, the position corresponding to the letter being learned, *I*. Obviously, if the letter *A* were being learned, a binary 1 would be entered in the eleventh or *A* position, and so forth. The process described constitutes the learning of a single letter *I*, but whole series of letter I's, differing in shape or position or both, can be learned. For example, Figs. 1b, 1c, and 1d show the same *I* being learned in different positions, while Fig. 3 shows a case in which two *G*'s have been learned.

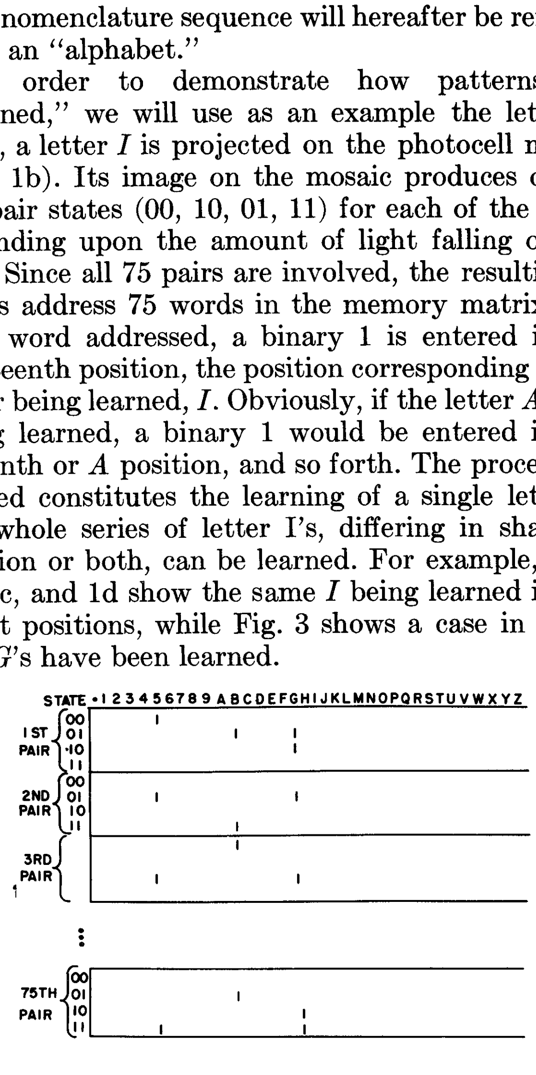

STATE • 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1ST PAIR
00
01
10
11
2ND PAIR
00
01
10
11
3RD PAIR
75TH PAIR
00
01
10
11

Fig. 3—The memory matrix with the characters B, G, and 5 learned. Note that two G's have been learned.

Since not all the letter *I*'s will be in the same position as the first, *some* different computer words will be addressed. That is to say, there is a degree of individual character variability. However, no letter *I* or combination of *I*'s will normally address the same 75 computer words as, say, a letter *A* would. This is a key point: *the very shape of a character, such as the letter I, forbids certain states for certain pairs. The existence of these forbidden states lies at the heart of our method, for without them the logic would saturate.* In sum, different patterns have different forbidden states and consequently score differently.

Now, suppose that we have taught the logic several alphabets, proceeding for each character as for the letter *I* above. We can then identify a specific unlearned character, an *A* for example. A letter *A* is "read" by imaging it on the photomosaic. Its image will address the 75 computer words in the memory matrix to correspond to the active states of the 75 pairs. Identification of the specific pattern in question is made by comparing the unknown image with the previously learned characters. In practice this is done in the following way:

(1) The binary 1's in position one (the position corresponding to •) are added up for all of the 75 computer words addressed by the unknown pattern. The score obtained shows the similarity of the unknown pattern to the • pattern.

(2) The same process is repeated for the other 35 positions, with the result that 36 numerical scores are obtained.

(3) These scores are compared by the computer, and the highest score wins. That is, the unknown pattern is identified with the character occupying the position scoring highest. If there is a tie for highest score, the computer arbitrarily selects one of the highest scores as the winner. Note that the highest score possible is 75.

Fig. 4 shows an example of scoring for hand-block *A* and *T*. Fig. 5 shows scoring for much more highly variable patterns, namely, handwritten *a* and *t*.

It will be noted that if an image corresponding *exactly* to the unknown image had been learned before by the matrix, a score of 75 would be made at that position. Again, if by learning several similar patterns (*A*'s, for example), all of the pair states now being addressed had been learned, a second 75 would be made. However, in most cases, an unlearned character will not make a perfect score. The degree of simi-

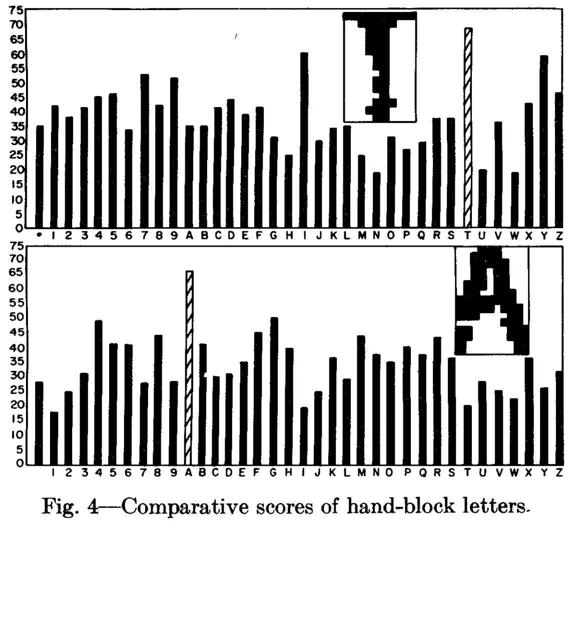


Fig. 4—Comparative scores of hand-block letters.

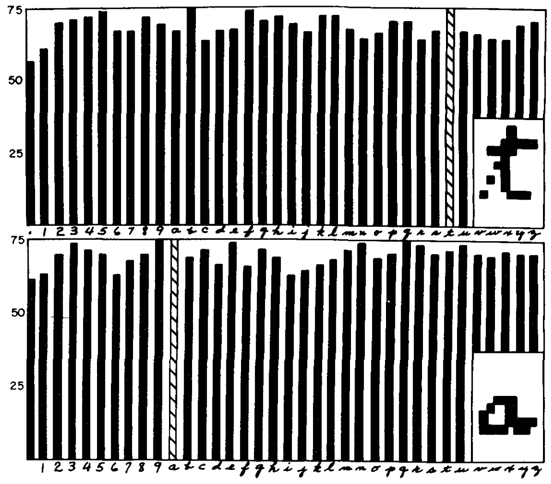


Fig. 5—Comparative scores of handwritten letters.

larity is measured by comparing the magnitude of the various scores with a perfect score of 75. Discrimination is defined as the difference between the score of the correct character and the next highest score. It can be seen that what actually happens in this process is that the images, both those learned and those being read, are transformed into a new space (the memory matrix) and are there compared for identification.

Logic Expansion and Manipulation

Our studies and experiments moved outwards from the basic method to include a variety of modifications and variations. An attempt is made below to evaluate each variation in terms of its final effect. It should be noted that the combination of two or more of the methods to be described results in substantial increases in correct readings.

Different Photocell Groupings

In the examples cited, the photocells were grouped as exclusive pairs. However, it is obviously possible to use n-tuples in which n has any value from 1 to 150. Let us begin by comparing the system employing photocell pairing ($n = 2$) with a system in which $n = 1$. In the latter case, each individual photocell addresses only two computer words, since its possible states are 0 and 1. The difference in the behavior of the systems is striking. If we re-examine Figs. 1b, 1c, and 1d, we note that in learning several images of the letter *I*, with $n = 1$, every single photocell would exhibit the values 1 and 0: this is so because the position of the letter *I* changes. In other words, unless the image on the mosaic is held within narrow limits, the memory loses most of its discrimination value with $n = 1$.

We can say then, that position is very critical in the case of $n = 1$, and that it has less importance for $n = 2$. A direct consequence of this difference is

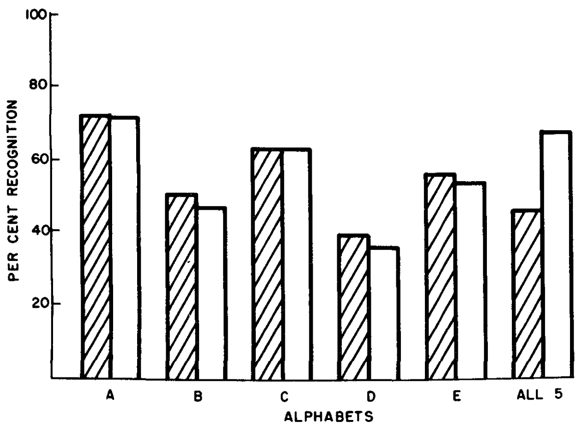


Fig. 6—Comparisons of the percent recognized for hand-block print read with different n-tuplings: $n = 1$ (hatched bars) and $n = 2$ (solid bars). Note that when all five alphabets are learned together, the percent for $n=2$ improves. In other words, for $n = 2$, the ability to read improves with additional learning in the memory matrix.

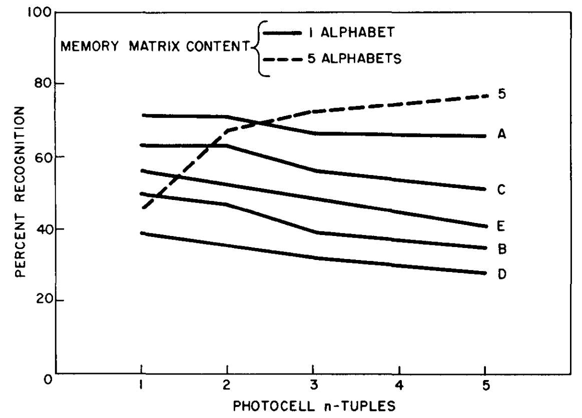


Fig. 7—Comparison of percentage recognition of hand-block print with different n-tuples. Five alphabets (labelled A, B, C, D, and E) are considered singly, and then together.

found when the matrix is taught more than one position or more than one example of a pattern. The scores will improve if $n = 2$; for $n = 1$, they will not improve and will probably deteriorate. Figs. 6 and 7 illustrate this characteristic with respect to five alphabets learned separately and then in combination. Marked improvement in the reading of this message, which was written in hand-block print, was achieved when $n = 2$ rather than $n = 1$. For the five alphabets learned separately, the average percent of recognition with $n = 1$ was 56.12 percent; for $n = 2$, 54.01 percent. But for the same five alphabets learned together, the percentages are 46.42 for $n = 1$, and 67.63 for $n = 2$. (See also Figs. 8a and 8b.)

Remembering that n can equal any number from 1 to 150, we can ask what effect is produced when higher n-tupling is used. The problem of pattern recognition with a multichannelled system, such as the one simulated for discussion here, has traditionally been approached from one of the two extremes, that is, $n = 1$ or $n = 150$. Consider the formula

$$S^n \times \frac{N}{n} \times C = L,$$

where

S = the number of operational states of the photocell. In the case being considered $S = 2$, for the possible photocell states are 0 and 1.

n = the parameter for n-tupling.

N = the number of photocells.

C = the number of categories of patterns learned and read (36 in the previous examples).

L = the number of storage sites in the memory matrix.

The factors held arbitrarily constant in our experiment were $n = 2$, $N = 150$, and $C = 36$. The traditional cases, as mentioned before, have involved $n = 1$ and $n = N = 150$. But the former has been shown to deteriorate or at least not to improve appreciably with learning. The latter, on the other hand, requires a prohibitively large memory matrix (36×2^{150}, using the same values as above), although its reading ability would be perfect if enough learning experience could be provided.

Let us summarize concerning these two extreme conditions. If $n = 1$, there are no forbidden combinations and therefore the memory will saturate with the learning of successive characters which vary in size, area, shape, or position. Such a logic has, consequently, an extremely limited use. If $n = 150$, saturation is impossible. But, even apart from the impossibility of having 2^{150} computer addresses available, images being read successfully would be restricted to exactly those that had been learned before. This logic, then, has even more severe limitations.

Our method avoids these several disadvantages by concerning itself with intermediate values of n, values which provide the learning advantages of a large exponential matrix but which retain a memory matrix more comparable in size to the photomosaic matrix. For example, with $n = 2$, the formula for the logic used gives:

$$2^2 \times \frac{150}{2} \times 36 = 10{,}800$$

The number of bits in the memory matrix for the simplest case of a system not position sensitive, under these conditions, is therefore 10,800.

Let us introduce another quantity, M, which will be the number of photocell n-tuples utilized in a given experiment. While M will normally be given by N/n, larger M values can be obtained by non-exclusive n-tupling of the photocells. We will have more to say about the non-exclusive cases later.

In any event, it is obvious from the formula that a larger memory matrix can be utilized if any of the variables are increased. During the course of our experiments, we used the following values:

$$n = 1, 2, 3, 5, 8$$
$$M = 30, 50, 75, 150, 128, 256, 512, 1024$$
$$C = 10 \text{ and } 36$$

The experimental data suggest that a greater amount of logic produces better discrimination. The primary effect of varying n is that as n increases, the percent of recognition increases with increased learning (Figs. 7, 8a, and 8b). However, a balance must be preserved among the various parameters in order to utilize to best advantage a given amount of logic and to minimize computing time.

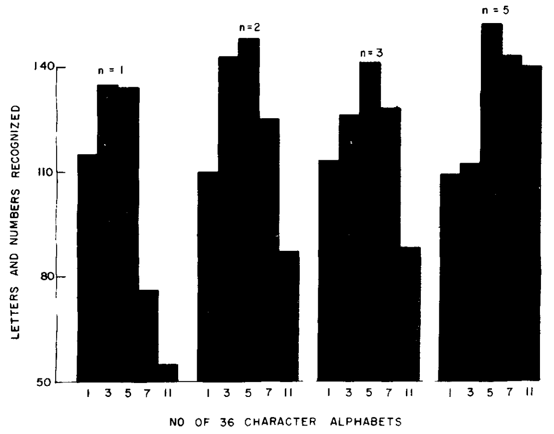


Fig. 8(a)—Scores made on handwritten script letters, showing that for larger values of n, larger amounts of learning are useful.

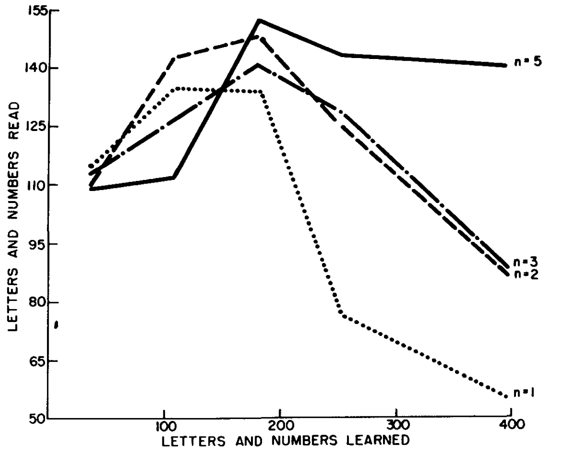


Fig. 8(b)—Material of Fig. 8(a) presented in different form.

Non-exclusive n-tupling

Some experiments were made in which non-exclusive n-tupling was used for the photocells. The number of n-tuples (M) used could in these cases have any value. Tables I and II show that non-exclusive pairing resulted in some improvement in the percent of characters recognized. But this improvement was at the expense of more storage space and longer computing time. We feel that a larger gain in percent recognized can be realized, for the same amount of storage and same length of computing time, by increasing the number of photocells (N) and continuing to use exclusive n-tuples. In other words, we see no real advantages in non-exclusive grouping.

TABLE I

Progress in Reading Hand Block Print

n-Tupling		Alphabets Learned	Manipulation	Percent Read
Exclusive	Non-exclusive			
1		1	None	39–72
5		1	None	28–66
1		5	None	46
3		3	None	78
2,3,5		2	Probability	77–84
2,3,5		3	Distribution	80–84
	2,3,5	1	None	80–85
2,3,5		4	Rotating Origin	88–92
	3	1	Rotating Origin	96
2,3,5		1	Context	94–100
2,3,5		1	Context-Positioning	98–100

TABLE II

Progress in Reading Handwriting

n-Tupling		Alphabets Learned	Manipulation	Percent Read
Exclusive	Non-exclusive			
1		1	None	26.14
1		3	None	30.68
2		1	None	25.00
2		5	None	33.64
5		5	None	34.55
5		3	Distribution	43.84
	5	5	None	24.55
	5	5	Positioning	53.15
	5	11	None	50.00
	5	11	Positioning	58.56
	5	11	Rotating Origin	60.00
5		11	Context-Positioning	94.32

Positioning

A procedure for pre-positioning characters for learning and reading by rotating an origin was attempted and found to be profitable in special cases. This rotating-origin technique is useful for digits and for print, but will not work with handwritten script. That is to say, if a character or pattern is separate and distinct, it can have an origin rotated with respect to some reference. Handwriting (as contrasted with the separate handwritten characters which we used) has continuity, and there is no obvious origin from which to start. Some method for separating handwriting into its components would be required before the origin of such components could be rotated profitably.

For each character an origin is arbitrarily defined. The character is then successively repositioned about this origin in the following sequence of x, y values: 0,0; 1,0; 1,1; 0,1; -1,0; -1,-1; 0,-1; 1,-1; 2,0; etc. Scores are obtained for each value, and the maximum score made by a character in any of the positions is chosen as the identifying score. This program involved a considerable amount of computer time, and is of interest mainly in connection with the possibility of simulating conditions for "servoing" the "eyeball." Such a feedback system appears feasible, since effective score criteria were found.

In a variation of the positioning program, the characters were all relocated by the computer to the upper left hand corner of the rectangle. This positioning, combined with the rotating-origin program just described, gives the maximum probability of reclaiming position-dependent data. This combination provides the largest increases in effectiveness for the $n = 1$ cases, those cases which we have seen are most sensitive to position. Typical increases in percent recognized for hand-block print with these techniques are:

Original	*Positioning*	*Rotating origin*
80	84	89
72	88	90

Distribution Processing

A method of processing the data obtained from the pattern scores was tried which was based on the entire scoring pattern rather than upon the maximum score only. The principle involved becomes clear at once if Fig. 5 is re-examined. Note that the sets of scores with respect to the previously learned letters are quite different for *a* and *t*. These different values are apparently consistent in their differences. For example, *t* scores high for *b*, while *a* scores low for *b*, and so forth.

The procedure is first to teach the memory matrix several alphabets as a primary experience. Scores made by one or more additional alphabets, constituting a secondary experience, are then averaged to give a score distribution typical of each character. An unknown pattern is compared with the memory matrix in the usual way to obtain its distribution of scores. This distribution is then compared with the typical distributions and the one most similar to it is chosen. For convenience, all of the scores were normalized, so that the sum of the scores in each distribution was one. Comparisons between two distributions were made in these experiments by summing the absolute values of the differences of the corresponding scores. It might well prove useful to employ a correlation technique in which a sum is taken of the products of corresponding scores, but this has not yet been tried.

As an example of results, in one case in which handwritten script characters were being read ($n = 5$, 3 alphabets learned), we found:

Undistributed	32.3% recognized
Distributed	45% recognized

A final approach in this effort was to introduce ten arbitrary shapes for the primary experience (Fig. 9).

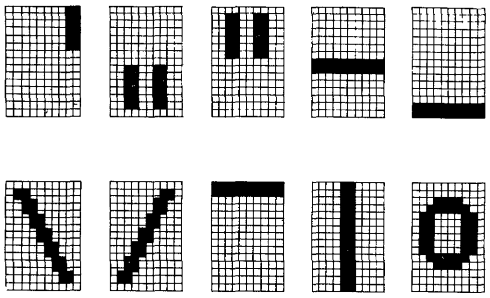


Fig. 9—Arbitrary shapes which were taught to the system as a basic distribution pattern for the subsequent reading of alphanumeric handwritten characters. Each shape was learned in the position shown and also in several positions resulting from lateral displacement.

After these were taught to the memory matrix, three alphabets were compared with the matrix to obtain a ten-component distribution analogous to the 36-component bar graph of Figs. 4 and 5. The three ten-component distributions were averaged. New alphabets could then be read by the distribution-comparison program. For handwritten script the results of this program were:

Undistributed	32.3% recognized
Ten-Component Distribution	51% recognized

This program was novel in that it involved two steps of disorder; that is, two arbitrary operations — random pairing and comparison with arbitrary configurations — were performed on patterns before attempting to read order out of them. It is also important to note that by using only 10 shapes instead of 36, a considerable saving in computer time is realized.

Probability

The method of reading characters described previously utilizes a memory matrix which is taught by a given set of experience patterns. Another method was tried in which the contents of several such memory matrices were *averaged* to obtain a "probability" matrix which was then used as the memory matrix in the reading phase. The memory matrices used in the averaging can be taught by different sets of experience patterns. An interesting (but not very successful) special case is one in which each of the matrices being averaged is taught only one alphabet of experience patterns.

In the few cases tried with this method, the percent of handwritten characters recognized was increased as follows:

Original	28% recognized
Probability Matrix Used	52% recognized

Certain variations of this "probability" method will undoubtedly yield some increase in percent recognition.

Discrimination Criteria

The scores obtained for each pattern read by any of the described methods lend themselves readily to the establishment of discrimination criteria. That is, if the standard of minimum margin is not met for a given image, a secondary program can be evoked which utilizes one of the higher (and probably slower) logic treatments for higher resolution and/or discrimination. Such a program would give the computer a second, and "more careful look" at a pattern which was not clearly recognized on the first trial.

Randomness

Since the elements of the photomosaic are related to each other by randomly chosen n-tuples, it was decided to test the sensitivity of reading ability to changes in the particular organization used. The random (actually pseudo-random) n-tuples were generated by the following program. First a random permutation, $k(1)$, $k(2)$, $\ldots$, $k(150)$ of the numbers 1, 2, 3, $\ldots$, 150 was generated. Then the elements of the mosaic EE_1, E_2, $\ldots$, E_{150}, were related in this manner:

$$(E_{k(1)}\,,\,E_{k(2)}\,,\,\ldots\,,\,E_{k(n)}\,,\,(E_{k(n+1)}\,,\,\ldots\,,\,E_{k(2n)}\,,\,\ldots\,,\,(E_{k(150-n)})\,,\,\ldots\,,\,E_{k(150)}).$$

The test was made by using five different randomly chosen permutations to read the same set of patterns. The results are shown in Fig. 10. Although admittedly the sample was rather limited, indications are that the percent recognized is fairly insensitive to the variation, especially when the precent recognized is high.

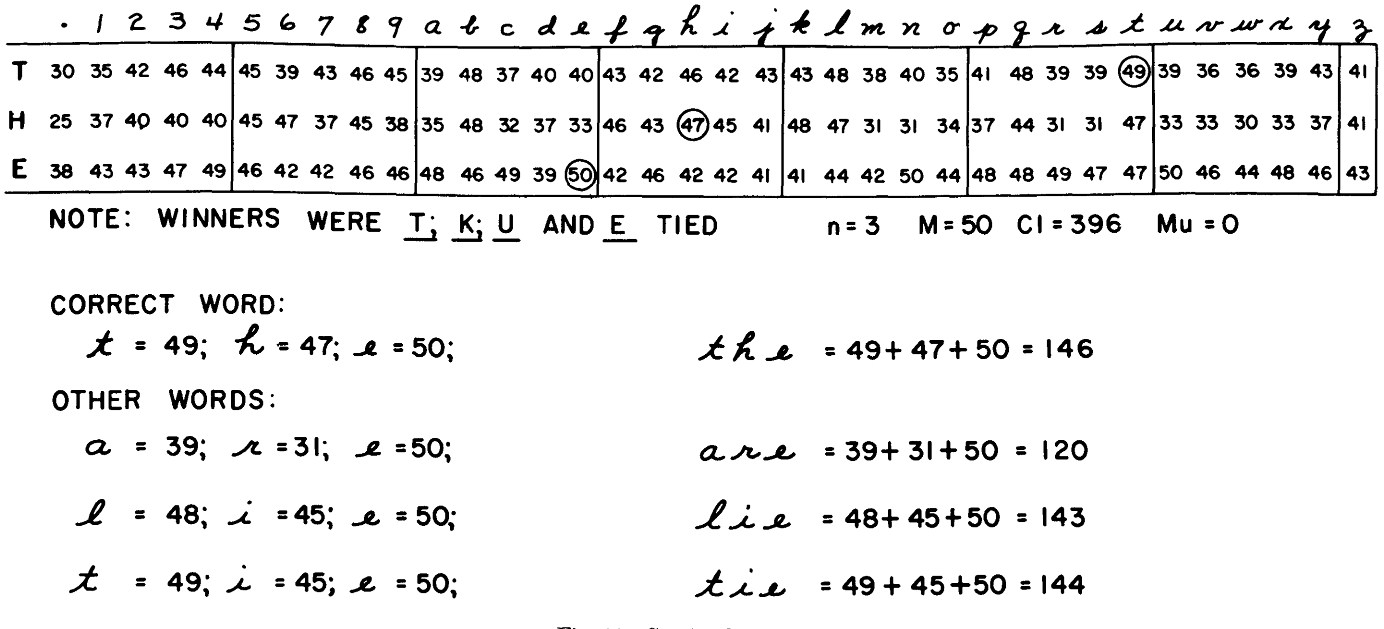
	CHOICE NUMBER						
n	I	II	III	IV	V	MEAN	σ
HANDWRITING							
2	31%	27%	32%	35%	35%	32%	3.0
5	36%	33%	33%	37%	36%	35%	1 7
HAND BLOCK PRINT:							
2	78.5%	78.2%	772%	778%	80.1%	78.4%	1.0

Fig. 10—Percentages of recognition for five different choices of random n-tupling.

Context

Another method to extend the basic technique deserves special attention, for it produced the highest percentage of correct readings. It is identification of letters by word context, and it operates as follows:

1. Establish the length of an unknown word by counting the number of characters between spaces.



	.	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
T	30	35	42	46	44	45	39	43	46	45	39	48	37	40	40	43	42	46	42	43	43	48	38	40	35	41	48	39	39	49	39	36	36	39	43	41
H	25	37	40	40	40	45	47	37	45	38	35	48	32	37	33	46	43	47	45	41	48	47	31	31	34	37	44	31	31	47	33	33	30	33	37	41
E	38	43	43	47	49	46	42	42	46	46	48	46	49	39	50	42	46	42	42	41	41	44	42	50	44	48	48	49	47	47	50	46	44	48	46	43

NOTE: WINNERS WERE T; K; U AND E TIED   n = 3   M = 50   CI = 396   Mu = 0

CORRECT WORD:

t = 49; h = 47; e = 50;   the = 49 + 47 + 50 = 146

OTHER WORDS:

a = 39; r = 31; e = 50;   are = 39 + 31 + 50 = 120

l = 48; i = 45; e = 50;   lie = 48 + 45 + 50 = 143

t = 49; i = 45; e = 50;   tie = 49 + 45 + 50 = 144

Fig. 11—Scoring by context.

2. Establish, by the techniques previously described, all the scores for all of the letters constituting the unknown word.
3. Using a vocabulary of words of the length in question, add in their proper order the letter scores of each word in the vocabulary to obtain a total score for each word.
4. The highest score wins.

Fig. 11 illustrates the whole process. Words can be read by context (see Fig. 12), or, since each word has a score, it would be possible to establish a similar program to deal with phrase context.

The word *the* in the message in Fig. 11 won against 100 other three-letter words even though it was badly misread letter by letter. The results shown

MESSAGE

THE COMPUTATION IS DONE BY THE USUAL MACHINE

FOR n = 2 (11 ALPHABETS)

LETTERS

TKU GXMPYTYTTEN LU DEYT FY TTU UUEET MNQHTUU

CONTEXT

THE COMPUTATION IT DONE BY THE GREAT MACHINE

FOR n = 5 (11 ALPHABETS)

LETTERS

TKE GVMSUTUTIVN 2U DVUM BY TKU USMAD MNCHTUE

CONTEXT

THE COMPUTATION IS DONE BY THE USUAL MACHINE

Fig. 12—Handwritten letters read by context.
Letters and words incorrectly identified are underscored.

in the table were obtained using a vocabulary of 677 most commonly used short words. Obviously a larger vocabulary would result in decreased recognition.

Tables I and II summarize the results obtained for reading hand-block print and handwritten script by the basic method and by the various modifications described.

Discussion

The problem posed by this investigation was: Can a general program be utilized to attenuate the information contained in a higher-order matrix pattern, while at the same time retaining enough of the essence of the information to categorize the pattern. Our results clearly indicate that this is possible. And although such a program will be useful at once for purposes of character recognition as is required in general reading machines, it has a much broader import.

A general program of this sort — as opposed to such specific logical programs as pattern matching or analytical character differentiation — will be useful as a basic tool in our investigation of decision-making circuits. This method could be expanded into such areas as phrase context, the automatic reading of books as a service to language translation programs, etc. It should perhaps be re-emphasized that the program identified typewritten numbers without error in the cases considered. Handwritten script was purposely introduced to challenge the program by offering it patterns of high variability.

Acknowledgments

The authors wish to express their sincere gratitude to Miss Tomasa Santos for her extensive work in programming these concepts and running them on the computer, to Dr. L. S. Lockingen for his large part in interpreting the data, and to Mr. S. D. Stearns for his generous assistance and the use of his equipment.

Discussion of Problems in Pattern Recognition

W. W. BLEDSOE, J. S. BOMBA, I. BROWNING, R. J. EVEY, R. A. KIRSCH, R. L. MATTSON, M. MINSKY, U. NEISSER, AND O. G. SELFRIDGE

Various problems encountered in pattern recognition were examined by an eight-man panel during the Thursday afternoon session. The panel included O. G. Selfridge, M.I.T. Lincoln Laboratory, Session Chairman; R. A. Kirsch, National Bureau of Standards; M. Minsky, Massachusetts Institute of Technology; U. Neisser, Brandeis University; and the authors of the four preceding papers — R. J. Evey, I.B.M. Corporation, "Use of a Computer to Design Character Recognition Logic;" R. L. Mattson, Lockheed Missiles and Space Division, "A Self-Organizing Binary System;" J. S. Bomba, Bell Telephone Laboratories, "Alpha-Numeric Character Recognition Using Local Operations;" and W. W. Bledsoe and I. Browning, Sandia Corporation, "Pattern Recognition and Reading by Machine." Following comments by Messrs. Kirsch, Minsky and Neisser, the panelists answered a number of questions from members of the audience.

R. A. Kirsch

I WOULD like first to make a few technical points. The first comment relates to the connection between the Bledsoe-Browning paper and the Mattson paper and the relationship between these two papers and some of the other work in this field that you may know about. If we have an input retina in a pattern recognition device which has n cells, then a pattern may be defined as a Boolean function of the n variables. Each of the various representations of a pattern is another one of the terms in the disjunctive normal form of the Boolean function which characterizes the pattern. This is substantially the way in which Mattson looks at the notion of a pattern. Mattson's criterion for similarity between two patterns is the usual metric of the symmetric difference of the two patterns or, what is the same, the area of the modulo-two sum of the two patterns. This measure has, of course, useful mathematical properties and leads to an interesting and perhaps fruitful analysis, but the important relevant question is whether this particular criterion for similarity corresponds to any very important class of pattern similarities. The evidence that Mattson offers shows that at least some of the types of similarity that we would like to attribute to sets of patterns are, in fact, reflected in his measure of similarity.

The economy in terms of number of devices required for a Mattson type of self-organizing pattern recognition system is determined by the extent to which the pattern classes represent linearly separable Boolean functions. Interestingly enough in the examples that Mattson gives, the functions appear to consist of single clusters of points in Boolean n-space. However, we know that there are important pattern classes which are not so economically realized by a Mattson type of device. In fact, Mattson recognizes that the alternate symmetric function is a very difficult one to implement in terms of number of devices required. The alternate symmetric function happens not to correspond to an important type of pattern, but certain other functions do; for example, a function whose disjunctive form contains all terms having exactly half of the variables complemented and half of them uncomplemented. This corresponds to the pattern class which has 50% black points and 50% white points. This, although not as difficult to implement with a Mattson device as the alternate symmetric function, nevertheless is reasonably difficult and perhaps corresponds to an important pattern class.

Bledsoe and Browning give some detailed discussion of the effects on their recognition program of changing the size of n for their n-tuples. It is interesting to note that the case of n equals 150, which they reject on the basis of the size of memory required to implement it, is the case that corresponds to the situation chosen by Mattson for his recognition scheme. Mattson gets around the large memory requirement by the use of his similarity criterion which, as I pointed out before, is somewhat debatable in terms of its usefulness. Bledsoe and Browning quite correctly point out that for a case of $n = 1$ and patterns subject to all possible translations the memory will saturate. Their discovery that for $n = 2$ saturation does not occur can be related to the fact that characters subject only to translation have a constant autocorrelation function, and a set of patterns with a constant autocorrelation function do not cause saturation for $n = 2$. If we allow the patterns also to change linearly in size, or also to rotate, the autocorrelation function changes and hence $n = 2$ saturation becomes possible. Again we are confronted, as we were in the Mattson paper, with an experimental question: "Does the autocorrelation function of a character tell us more about that character for purposes of recognition than does the character itself?"

It seems likely that we can invent important pattern classes that will confound the autocorrelation function method of recognition, which is implicit in the Bledsoe-Browning approach, just as we can do similarly with the Mattson approach. This should not be taken as an implied criticism of either of these approaches. Both of them have implicit in them, and, in fact, all pattern recognition schemes have implicit in them, certain built-in heuristic criteria which make them useful for certain types of pattern recognition problems. Since no pattern recognition scheme is truly universal, and we need not expect that any scheme will be truly universal, the fact that a heuristic will work in certain situations and will not work in others need not necessarily be a shortcoming. The significant question is how important that heuristic will be for the problems that are to be solved. It is for this reason that reference such as has been made, not in these papers but in other papers in the field, to general purpose pattern recognition devices seems at least misleading and probably just not true.

I would like also to make a few remarks of a less technical nature. The fact that there are some strong similarities between at least two of the papers in this session, and that these similarities are superficially not obvious, leads one to feel that what is needed in the pattern recognition field is at least some type of unified terminology and perhaps even a unifying theory which will make comparison of results simpler. It is not superficially obvious, although I nevertheless believe that it is true, that the two types of devices that we have seen here can accomplish the same type of recognition and no more nor less, for example, than that of the Perceptron device of Rosenblatt. If this is true, it is certainly a fact made more obscure by Rosenblatt's talk of neurons, Mattson's talk of Boolean functions, and Bledsoe and Browning's talk of *n*-tuples in a photocell mosaic. This lack of a unifying terminology may perhaps explain why so many of the workers in the pattern recognition field fail to give credit to their predecessors, who very often have contributed useful ideas.

Another possible explanation is the generally mistaken notion that proprietary interests in pattern recognition research must be protected because of important commercial applications; for example, to the problem of character recognition machine development. We have seen here in this session the example of Mr. Evey of how a successful character recognition machine may be developed without the necessity of being concerned with fundamental pattern recognition problems. Conversely we have seen in such papers as the Bledsoe and Browning paper, the Mattson paper, and, perhaps to an extent, in the Bomba paper that pattern recognition research can continue without necessarily contributing very directly to the development of character recognition machines. Confusing pattern recognition research with specific machine development does not accrue to the benefit of either of the two types of programs.

There is one final point that I would like to make. Pattern recognition research has been, I think, to some extent held back by a lack of proper data in several of the research efforts. About two years ago at the Eastern Joint Computer Conference in 1957, when I gave a paper on picture processing, I offered to the workers in this field the data that we at the National Bureau of Standards had generated with our picture digitalizing device. It was disappointing then and still is to see workers in the pattern recognition field preparing data by hand and attempting to investigate a problem which is so largely experimental in nature with data that is quite inadequate. What one needs for successful prosecution of pattern recognition research is large quantities of information in machine form and perhaps automatically generated information. I would like again to offer our service for helping make this data available to people in the field, and I would invite any of my colleagues who have such data to do the same. Where such data is easily generated and can be made available to active workers in the field, the general status of research in pattern recognition can be considerably advanced.

M. Minsky

I would like to make some remarks on how the character-recognition systems presented today may be related to more general pattern-recognition problems.

What are patterns? It is hard to define precisely this intuitive concept but I think we will agree first that the things we call patterns are *classes* of signals or figures, not single figures. They are classes of figures which are grouped together because, for some purpose or other, they can be treated alike. Now in the character-recognition problem it is perfectly clear in what sense the figures so grouped are to be treated alike. In the case of *printed* characters the patterns have a structure derived from the manner in which the figures are produced. In each case one starts with an ideal "prototype." To read a text requires that one decide, for each image on the paper, which of the prototypes was responsible. Each of the proposed systems computes the values of a set of functions of the image. Once these values are computed the system faces a statistical inference problem: on the basis of the evidence (the set of computed values) what is the prototype most likely responsible?

The figures are related to the prototypes by various kinds of noise and distortions. Simplest, perhaps, are the "additive" noises, in which pigment is added to or subtracted from the true image in a manner independent of the image. For these distortions one may do well with a simple correlation or area-matching analysis. If the noise is not extreme, or if the ensemble of patterns is small, it may be possible to confine the analysis to just a few of the points, or pairs of points, etc. With more complicated figure-dependent distortions (for example, smearing) more complicated functions may be necessary.

In the case of more global distortions involving, for example, variations in size and position, the simple area-matching tests will give poor results.

There is a question in my mind about the extent to which some of the methods discussed here can handle, or be easily extended to handle, that kind of problem. In the Bledsoe-Browning system the figures are constrained so that rather little translation is admitted. The method requires discovery of *simple* combinations which separate patterns, and these will be rather hard to find as the admitted distortions become more extreme. And when we look beyond prototype-derived patterns to, for example, patterns defined by connectivity, I think we will find that one needs to go to essentially more complicated functions; one cannot get by with larger numbers of simple ones.

The method of Mattson is useful, it seems to me, only where the area-matching kind of comparison is appropriate. The technique of Evey works on translated figures because the figure is actually "rolled" or scanned in Pitts and McCulloch fashion through the vertical translates after being normalized with respect to the horizontal translates. This seems to give excellent results for the chosen problem, but it really is difficult to make a case for the scanning method when faced with more complex classes of distortions.

The Bomba system has, I feel, rich possibilities for extension to harder problems. For it is based on the use of complicated but rather meaningful properties of the figures. The "meaningful" properties of figures can be used to define and recognize patterns based on ideas more abstract than that of the distortion of prototypes. (I certainly do not mean to imply that even the printed-character problem is trivial. Consider, for instance, the examples illustrated in Mary Stevens' Bureau of Standards report.)

In each model there is an inference stage in which one combines the values of a number of rather independent functions of the figures. One must require that the functions be at least partly insensitive to the differences between instances of the same pattern — otherwise the system will be overwhelmed by the need for a special treatment of each case. In the case of systems which generate test functions at random, through net connections or the like, we can expect that most of the functions defined will be useless or nearly so and a learning procedure will be needed to separate out and weight the relatively good ones. When the patterns in question are very complex, the chance of finding any significant functions at all in this way becomes prohibitively poor, and such systems certainly cannot "handle all kinds of patterns with equal facility" except when one is restricted to patterns of equal simplicity.

The papers suggest different ways to combine the results of the different tests or functions. Evey's system sets up fourteen "character triggers" with different tests and requires a stringent matching for one and only one of them — this harsh requirement is, of course, perfectly appropriate to the problem he is working with (namely, handling money). In the Bledsoe-Browning system one compares scores based on summations of contributions from many tests. They describe at least two ways of combining many fragments of evidence, none particularly decisive alone, to obtain conclusions which are conclusive and accurate. (A rather general discussion of "parallel" methods of combining evidence from many tests can be found in a paper of Selfridge, whose "Pandemonium" computer concept embraces many of the models being explored these days.)

In the Bomba system, with its decision tree structure, we find a rather different way of combining evidence. As Dr. Neisser will note, a program which has been run at our laboratory uses test functions which are not very remote from those of Bomba, but which are combined in a parallel decision rather than a sequential tree method. This program, due to W. Doyle, is of interest here, I think, in that it can be regarded as using tests related to the kind in Bomba's program, a decision method not far from that of Bledsoe and Browning, and with, I believe, comparable success. One can use the decision tree structure only when one is very sure that the decisions at the top of the tree are very reliable — of course, in that case one can avoid all the computations in the discarded portions of the tree. Otherwise one has to obtain in some way the kind of redundancy supplied by the parallel methods.

That is all I have to say, in general. I find I can't refrain from remarking (with all due apology to Mr. Evey, who is not responsible) that the choice of type-face "E13" in the ABA magnetic-ink system seems unfortunate to me. I don't find the characters very legible. Speaking as a human being I will grant that we do seem to have here a solution to the problem of character-reading-by-machine, but we are left with something of a problem in the way of character-reading-by-people!

U. Neisser

I will try to do a couple of things. One is to distinguish the two basic types of program in this field; the other is to emphasize the things they have in common which are worth considering in future work. The types I have in mind are those mentioned by Minsky: decision trees and parallel processing.

It does seem that where you cumulate separate decisions you are more apt to make mistakes. Bomba's program was the only one of these four involving a tree; you could see it most clearly in the diagram of decision logic that he showed. Other tree programs have been reported in the literature, however. I think all of them will have great difficulty in learning from experience. If the program makes a wrong decision at any fork it will necessarily be wrong at the end, with no obvious way of finding out what should be changed in order to improve performance. In Bomba's program, for example, the entire decision process is spelled out in advance by the programmer; the machine will never improve on his original descriptions of the characters.

In fact, all the tree programs have used quite well-printed characters as inputs. If the characters were made a bit sloppily, or slanted, the program could no longer distinguish one from the other. I would worry about that if your objective is to recognize hand-printed characters: *really* hand-printed, as when you ask somebody to print their name and address. If you look in detail at what people do under these conditions it is miserable.

In parallel programs the basic idea is to take a lot of intermediate functions of the stimulus pattern, and then make a decision on the basis of some more or less sophisticated maximum procedure. It seems to me that there are two important differences among parallel programs. The first is the degree to which the features used are shape-dependent rather than position-dependent. A program like Evey's is extremely position-dependent. His particular sure-bits are either on or off, and that's it; the shape of the input character is irrelevant except as it happens to fill particular positions. To some extent this amount of position-dependence characterizes the typical "neural-net" programs as well. In some of these, however, there is a good deal of preprocessing done; the character is centered and sized before the neural net takes over. One of the exciting things about the Bledsoe-Browning paper is that in considering pairs of bits, and larger n-tuples, their program is partly shape-dependent. The vertical *I* in the 7×10 matrix never excites bits on the left and the right side simultaneously because of its *shape*, not because of its original position.

The other important aspect of parallel programs is the sophistication of the decision process. The simplest procedure is to compute a lot of functions like the outputs of the n-tuples in Bledsoe-Browning and take the one with the highest score. They have done much more than that. In one of their procedures they look at the entire distribution of outputs rather than only at the largest; in another, as we have heard, they successfully introduced context into the decision process. There is a Lincoln Laboratory program for character recognition which is also parallel. It uses a large number of shape-dependent operations: features like Bomba's, directions of curvature, and things of that sort. It computes the outputs of all of these tests in parallel, and uses a very simple decision procedure. Even with this, it performs quite well.

My worry about Bledsoe and Browning's program stems from the small size of their matrix. It's hard to know how their system would perform if they introduced anyone's handwriting besides Ibn Browning's! Indeed, one of the impressive things about the character recognition problem is that you never know if something will work until you try it. Theory doesn't help; until a program is running you cannot know whether it will saturate or run out of storage or what have you, when confronted with sloppy characters. This may be a sign that we have a problem genuinely related to the simulation of higher mental processes.

Let me turn now to some of the similarities among the programs. One is the emphasis that several of these papers (as well as others in the literature) have shown in recognizing hand-printed characters rather than machine print. Let us all take warning from Mr. Evey, who showed how difficult even the IBM-font is if you have high standards of accuracy.

A second point is that most of these programs use some amount of pre-processing before they get into the real work. Bomba thins out lines and cleans up stray points. Unger's program, reported elsewhere, does similar things; so does the Lincoln Lab program. Evey moves the characters to one side of the matrix; Bledsoe-Browning have experimented with shifting them to one corner. Roberts' perceptron-type program, at M.I.T., uses centering and scaling. In other words, these programs involve two levels of processing. I think this will become increasingly important as we get to the more complex patterns that we would like to recognize successfully. Suppose, for example, that you are dealing with triadic patterns and you would like the computer to select that part which is maximally different from the other two. Such a task can't be accomplished with present types of programs; a different sort of analysis is necessary. You have to do several stages of processing, and I think that is what we are getting into.

One last point of similarity is a deficiency the programs have in common. None of them has yet looked at the problem of segmenting words into letters, either in cursive writing or in print. Yet you cannot actually read until you have some idea where one letter ends and the next begins. Even Bledsoe and Browning, as I understand, computed one letter at a time in their work with context. We will have to tackle segmentation pretty soon if character recognition is to be realistic; if it is to compare with the way people perform. It is quite clear, if you take handwritten characters (at least in my handwriting) that it is impossible to identify a letter directly. What you actually recognize is the word itself. So far, no paper that I have seen has faced this problem.

Discussion

Mr. Selfridge: While the question cards are being collected, I will give you a couple of biographies available in this field. The first is by Otis Minot, U. S. Navy, San Diego, California. This is PM 364, "Automatic Devices for the Recognition of Two-Dimensional Patterns." The other is by Mary E. Stevens of the National Bureau of Standards. The number is 5463, and it is called, "A Survey of Character Recognition."

J. K. Moore (Smith, Kline & French): How long did it take the 704 to recognize the sentence?

Mr. Browning: We did not attempt to recognize sentences per se; rather, we scored letters, then, with the context program, we scored words. At the beginning we were recognizing on the average one letter every 3/5 of a second. This is because, in order to understand the discrimination process, we had an elaborate print-out of the relevant data. We believe that this time could be reduced to perhaps a few hundredths of a second per letter. For contextual recognition of words, the time requirement is increased and will depend on the size of the dictionary.

D. Baumann (MIT): How does your device locate the character at the right side of trigger matrix before its recognition?

Mr. Evey: The character actually comes into the matrix from the left hand side and is moved across the matrix — we can think of it as this — and the machine decides that it has the character in the matrix when it sees any two bits in the first column.

E. B. Cohen (Auerbach Electronics): Have you considered other kinds of character distortion besides poor printing? Can the 1210 sorter read tilted or displaced characters?

C. E. Dorrell (IBM) What work is being done or what is the value of observing the profile density or edgeview of the characters?

Mr. Selfridge: If you take a character and project it to one side, you get a one-dimensional distribution with many fewer bits, which is presumably a good thing in processing. A number of people have considered this — Gerald Dinneen and I, for example, some years ago. We never did anything about it. It didn't look that advantageous. I think that the kind of program arising and being considered today might find projection an acceptable technique. I don't know of a program carrying through techniques like that today.

G. A. Barnard (Ampex): Please expand on the contextual positioning aspect of your method.

J. J. Murphy (Sylvania): Were the *n*-tuples ever other than randomly associated?

Mr. Bledsoe: Let me take the second question first. Since we had no preconceived indea of what patterns we wanted to recognize, we felt that randomly associated *n*-tuples would be a logical starting point. In a couple of test cases we found the non-random associations did not discriminate as well. We feel that, for a particular family of patterns to be read, some particular non-random association would be optimum.

Mr. Selfridge: Another general point was brought up about segmentation. This is going to be a real problem on this. Very few people hand print. The only ones I know are people who go to Radcliffe; they do tend to hand print. This is as in speech recognition; in identifying words you are aware that segmentation is the real primary problem.

M. Jacoby (Remington Rand): What is the spacing of the individual heads on the reading element?

R. P. Niquette (Ramo-Wooldridge): What kind of character rates are to be expected from the system you described? Failure rates?

Mr. Evey: The area of the check that has to be read is about a half inch across the bottom of the check. Actually in the IBM system there are 30 tracks which cover this half inch and you get it down to 10 tracks by tying every tenth track together, so you have three tracks together. So you actually have a positioning problem in the matrix. This is mentioned in the paper. I didn't want to take time to discuss this in the talk. So you have 30 tracks covering this half inch and each track ends up about 17.5 thousandths wide. There is a dead space in between each track of about six thousandths. The failure rate is actually part of the problem here because the specifications laid down by the ABA, where they wanted machines used by banks, were something less than one-tenth of one per cent reject and about a tenth or so of that for substitution. That is, the actual rates as originally spelled out were one reject in every 2,500 checks, which figures out to four-tenths of one per cent and about a tenth of that for substitutions, which figures out to about one substitution in a million characters.

Mr. Neisser: Won't serifs confuse your technique? How will you handle them?

Mr. Selfridge: Is the trêe decision technique absolutely essential to your problem?

Mr. Bomba: No, the tree decision technique isn't essential to the program. The basic idea that I presented was to extract figures. Now, whether I use a tree method or some sort of statistical method with features as the input variables, is unimportant. The use of the tree here did enable me to show that I could recognize all of the characters in the alphabet. This particular type of logic which I chose in order to illustrate the feature extraction process was arbitrary.

With regard to serifs and other types of distortion, in some cases, particularly typewritten characters, the addition of a serif will be quite significant in changing the type of the character from a feature viewpoint. Actually the character might vary considerably, and thus the recognition procedure might well call a character such as a "T with serifs on it" by another name until the final recognition is done. So serifs do make a difference. However, small distortions, small serifs, would tend to be ignored by the feature extraction process. Seriphs cannot be ignored as they are often as large as the features which distinguish two characters. For example, on a pica typewriter font, the serifs on the T are as large as the feature at the lower right of the G, which distinguishes it from the C.

R. Marcus (MIT): Doesn't your system essentially consider each point in the matrix independently; that is, no consideration of correlation between different points?

O. N. Minot (USN Electronics Lab.): Could you give an example of a problem which leads to the sort of double peaks which you mention as requiring additional units in the adjusting equipment?

Mr. Mattson: The answer to the first question. This device considered individual combinations of bits but not single bits by themselves. It is the combination of certain key bits that enables the machine to recognize characters and other patterns.

The function which results in the double peak for the process is considered a function where, say, the 1111 point wanted to be mapped as a 1 and the 0000 point wanted to be mapped as a 0. It is impossible to have both of these points on one side of the plane and the others on the other side, so this function would yield two peaks.

E. B. Cohen (Auerbach): Can your approach learn to distinguish the handwriting of different individuals?

T. T. Rocchi (BTL): How do you explain that percent recognition increased, other things being similar, when the number of alphabets learned was increased?

Mr. Browning: In answer to the first question, it seems impossible that wth the present definition — which we had with 150 photocells — that we could recognize the handwriting of different individuals. After all, that number of photocells is approximately one-tenth of the number that a gnat has. If the number were greatly increased, we might well be able to distinguish the handwriting of different individuals.

In answer to the second question. I would explain the percentage recognition increase as follows. If the memory matrix has learned only one alphabet, any attempt to recognize an unknown character will be essentially pattern-matching with a complete position sensitivity. If the memory-matrix has learned a number of alphabets, the probability is increased that an unknown character will match a similar pattern previously learned in this position. The novel feature of our type of memory-matrix access is that the learning of a few subsequent patterns does not destroy its memory of previously learned patterns.

A Computer Analytic Method for Solving Differential Equations

LEO HELLERMAN†

INTRODUCTION

IN RECENT years, numeric analysis has been claiming an increasing share of overall mathematical research activity. The reason for this is apparently the need to have answers — numeric answers — to problems of modern technology, along with the development of the stored-program digital computer for carrying through the computations of numeric methods. But this emphasis on numbers is also an indication of the attitude of problem solvers: to use the computer, use numeric methods. And yet these methods are not always adequate. It may be more important to know how x depends on other variables than to know that $x = 3$.

The inadequacy of numeric techniques for the solution of differential equations is highlighted by the following engineering problem: In the design of a transistor switching circuit for a high-speed computer, we wish to know the output-current level at a particular time after the start of an input pulse. This information is contained in the solution of a nonlinear differential equation, which can be solved very nicely by numeric methods on a computer in, say, ten minutes. In evaluating the reliability of this circuit with respect to component deviation and drift, we want to know the statistical distribution of outputs. A simple method for finding this is synthetic sampling, or monte carlo[1]; but at ten minutes per solution, this may not be practical. However, monte carlo is known to be practical in estimating distributions associated with analytic expressions. This suggests that we first obtain the analytic solution of the differential equation, and then apply monte carlo to the solution. The methods are compared schematically in Fig. 1.

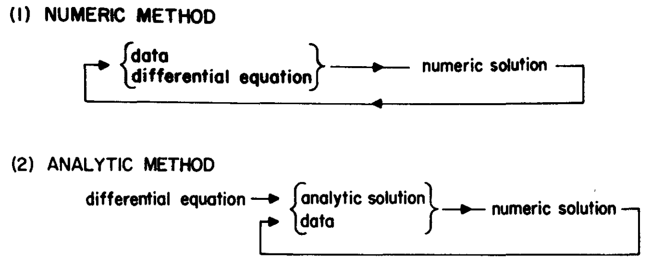


Fig. 1—Comparison of numeric and analytic methods of solving a differential equation many times.

† Product Development Laboratory, Data Systems Div. IBM, Poughkeepsie, N. Y.

[1] L. Hellerman and M. P. Racite, "Reliability Techniques for Electronic Circuit Design" *Trans. I.R.E. PGRQC*, Sept. 1958, pp. 9-16.

In the numeric method ((*1*) in Fig. 1), we must solve each case anew, starting with the data and differential equation. In the analytic method ((*2*) in Fig. 1), enough information is contained in the solution, so that we need solve the differential equation only once, and evaluate the solution for each case. Since a major portion of machine time is taken up with solving differential equations, there may be problems in which (*1*) is not practical and (*2*) is, provided (*2*) can be carried through by the computer.

The purpose of this paper is to call attention to a basic principle of analytic technique on a stored-program digital computer, and to illustrate this principle by a computer algorithm, and "address calculus," for finding solutions of ordinary differential equations by analysis. We also describe the implementation of this algorithm in an IBM 704 program. We see no reason why the same technique might not be applied to a host of other mathematical problems.

THE PRINCIPLE AND GENERAL APPROACH

The principle of numeric computation in a stored-program digital computer is well known: numbers are represented by the contents of storage cells, and computation is accomplished by arithmetic manipulation on these contents. Functions are represented by a finite table of numeric values. The principle of analytic computation may be stated thus: algebraic symbols are represented by the *locations* of storage cells, and analysis is accomplished by manipulating addresses. Functions are represented by machine programs. An algorithm for the analytic solution of a problem is an assignment of the correspondence of algebraic symbols with addresses, and a description of the way the addresses are manipulated.

In the description of the following programs, in referring to the address of some location corresponding to some symbol S, we will say "address S." Address and symbol are equivalent, and we may use the symbol to designate the address. On the other hand "address *of* S" refers to some other location and address, say T, which has the address S as part of its contents. Thus the address T may be the address of S.

Our approach to the analytic solution of ordinary differential equations will be to develop the Taylor series expansion of the solution. If the differential equation is

$$y^{(k)}(x) = f(x; y^{(0)}(x), \dots, y^{(k-1)}(x)) \quad (1)$$

then the formal solution is

$$y(x) = y(0) + y^{(1)}(0)x + \frac{y^{(2)}(0)\,x^2}{2\,!} + \dots \qquad (2)$$

where the $y^{(j)}(0)$ for $j = 0, \dots, (k-1)$ are assigned initial values, and for $j = k, k+1, \dots$ are determined from f and the derivatives of f.

Thus the heart of the problem is to develop analytic differentiation on a computer. In this connection we mention the work of H. Kahrimanian[2], and the LISP Programming System[3]. However, our approach is a bit different from both of these, being a close parallel to differentiation "by hand". A recently reviewed Soviet paper[4] appears to contain material quite similar to the work described here. The SHARE routine PE PARD[5] for differentiation and partial differentiation of rational functions is a prototype of our present program. Recall that the function to be differentiated is, in the computer, a stored program. PARD examines this program as a college sophomore examines a function to be differentiated, and when it finds it to be the sum of two parts, it applies the rule: the derivative of a sum is the sum of the derivatives. Or, if it finds a product, it uses $D(uv) = uDv + vDu$, and similarly for other differentiable combinations. Eventually, the derivative of a function is expressed in this way in terms of the derivatives of constants and the independent variable, and the differentiation process is complete. The problem in doing this on a computer is doing it in a uniform and orderly way, so that the method may be applied to arbitrary differentiable functions, and so that the results of the differentiation of each term can be combined in the end to one expression (program) for the derivative.

The Dendrite Nature of Functions

In this section, we examine a stored-program aspect of functions. Some notions will be defined which will facilitate the description of the differentiation algorithm.

A *binary operation* is an operation on two quantities. Addition, subtraction, multiplication, division and exponentiation are binary. *Unary operators* operate on a single quantity. Exp, log, sin, and cos are unary. Let the symbol $a * b$ have this meaning: $*$ is a binary or unary operation. If $*$ is binary it operates on a and b; if unary it operates on a, and b is ignored. Thus $a * b$ may be, for example, $a + b$, $a \div b$, a^b, or log a, or sin a. We will say a mathematical expression is a *finite dendrite* if it is composed by a finite number of binary and unary operations from a set of starting terms. We call a starting term an *elementary term*, or an *end:* it is not composed from other terms.

For example, consider the dendrite $y = (x + a)b - \sin x$. Its branching nature is shown in Fig. 2.

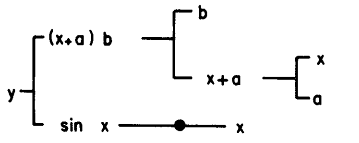


Fig. 2—The dendrite $y = (x + a)\,b - \sin x$.

The elementary terms are a, b, and x. The dendritic terms are, besides y itself, $(x + a)b$, $x + a$, and $\sin x$.

Note that the dendritic picture of y may serve as a flow chart for a program for its computation. First x is added to a, and the result is multiplied with b. Then x is operated on by some sine routine, and the result of this is combined by subtraction with $(x + a)b$ to give y. Thus a stored program for evaluating a function is essentially a sequence of binary and unary operations, starting with operations on elementary terms. That is, a program is a dendrite.

Blocks of *1*'s and *2*'s are a convenient notation for the branches of a dendrite. If $\alpha_1 \dots \alpha_n$ is such a block representing some dendritic term, (α_i = *1* or *2*), then, from the definition, $\alpha_1 \dots \alpha_n = \alpha_1 \dots \alpha_n 1 * \alpha_1 \dots \alpha_n 2$. The branch designations of the above example are shown in Fig. 3.

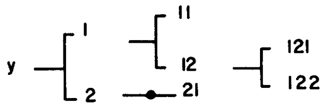


Fig. 3—Branch designations for the dendrite $y = (x + a)\,b - \sin x$.

A set of branches of the form

$$\alpha_1,\ \alpha_1\alpha_2,\ \dots,\ \alpha_1\alpha_2 \dots \alpha_n\,,$$

where the last branch is an elementary term, will be called a *chain*. All the chains of the example are

1, 11
1, 12, 121
1, 12, 122
2, 21

A finite dendrite has a finite number of chains.

The Differentiation Algorithm

Let us suppose we are given a y-program, that is, a stored program for computing y. We wish to extend

[2] H. G. Kahrimanian, "Analytical Differentiation by a Digital Computer," M. A. Thesis, Temple University, May 1953.

[3] J. McCarthy, "Recursive Functions of Symbolic Expressions and Their Computation by Machine" *Quarterly Progress Report* No. 53, Research Laboratory of Electronics, M.I.T., April 15, 1959.

[4] L. V. Kantorovich, "On Carrying Out Numerical and Analytic Calculations on Machines with Programmed Control," *Izv. Akad. Nauk Armyan. SSR.*, Ser. Fiz.-Mat., Nauk 10 (1957), No. 2, 3-16. See review No. 4360 by J. W. Carr, III, in *Mathematical Review*, June 1959.

[5] M. R. Dispensa and L. Hellerman, "Differentiation and Partial Differentiation of Rational Functions" *PE PARD*, SHARE distributed Program D2-445, March 18, 1958.

this to a program for its derivative, a *Dy*-program, by adding additional instructions. The block of storage for the *Dy*-program will include the block for the *y*-program. Since *y* is a dendrite, $y = \mathit{1} * \mathit{2}$ and

$$Dy = A_1 D(\mathit{1}) + A_2 D(\mathit{2}) \qquad (3)$$

where A_1 and A_2 are functions determined by the operation * and the branches *1* and *2*. That is, $A_\alpha = A_\alpha(*, \mathit{1}, \mathit{2})$ where $\alpha = \mathit{1}, \mathit{2}$. These functions are specified in Table I. It may happen that an A_α is the number 0, or 1, or some function which is known to exist in the *y*-program. This is the case for $y = u + v$ and $y = \exp u$, and in these cases it is unnecessary to place any instructions in the block reserved for the *Dy*-program. If an A_α is not of this type, say $A_\alpha = -u \div v^2$, then we do construct the program for this function and place it in the first available locations in the *Dy*-program block. Whether A_α is constructed or not, we save the addresses A_1 and A_2, in locations $L(\mathit{1})$ and $L(\mathit{2})$. Since we can only differentiate one term at a time, we also save the instruction to find $D(\mathit{2})$, in a location $I(\mathit{2})$.

TABLE I

A_1 AND A_2 IN $D(u * v) = A_1 Du + A_2 Dv$

$u * v$	A_1	A_2	I
$u + v$	1	1	v
$u - v$	1	-1	v
$u \cdot v$	v	u	v
$u \div v$	$1 \div v$	$-u \div v^2$	v
u^v, v constant	vu^{v-1}	0	v
$\exp u$	$\exp u$	0	0
$\ln u$	$1 \div u$	0	0
$\sin u$	$\cos u$	0	0
$\cos u$	$-\sin u$	0	0

We may now go on to find $D(\mathit{1})$. Suppose *1* is dendritic and $\mathit{1} = \mathit{11} * \mathit{12}$. Then

$$D(\mathit{1}) = A_{11} D(\mathit{11}) + A_{12} D(\mathit{12})$$

The functions A_{11} and A_{12} are constructed as A_1 and A_2, again by Table I, the addresses A_{11} and A_{12} are stored in $L(\mathit{11})$ and $L(\mathit{12})$, and after storing the instruction to find $D(\mathit{12})$ in $I(\mathit{12})$ we continue to find $D(\mathit{11})$.

In general, if $\alpha_1 \ldots \alpha_n$ is dendritic, then

$$D(\alpha_1 \ldots \alpha_n) = A_{\alpha_1 \ldots \alpha_n 1} D(\alpha_1 \ldots \alpha_n \mathit{1}) + A_{\alpha_1 \ldots \alpha_n 2} D(\alpha_1 \ldots \alpha_n \mathit{2}) \qquad (4)$$

The coefficients are constructed if necessary, and the addresses $A_{\alpha_1 \ldots \alpha_{n+1}}$ stored in $L(\alpha_1 \ldots \alpha_{n+1})$; the instruction to find $D(\alpha_1 \ldots \alpha_n \mathit{2})$ is saved in $I(\alpha_1 \ldots \alpha_n \mathit{2})$; then we go on to $D(\alpha_1 \ldots \alpha_n \mathit{1})$.

Eventually, since *y* is finite, we come to an elementary term, *11 . . . 11*. This will be a constant, the independent variable, or an initial condition $y^{(j)}(0)$, $j = 0, \ldots, (k-1)$. Thus $D(\mathit{11} \ldots \mathit{11}) = A_{11 \ldots 111}$ is known, and we store this in $L(\mathit{11} \ldots \mathit{111})$.

At this point we have traversed one chain of the dendrite *y*. We may now examine the *I*-cells for some deferred differentiation instruction, and proceed with the differentiation of this new term, until another end is reached. Continuing in this way, all chains will be completed, for there are a finite number.

It is clear from (3) and (4) that *Dy* is simply the sum of all products of *A*'s, where the subscripts of the factors of each product range over a complete chain. If $A_{\alpha_1 \ldots \alpha_j} = C_{ij}$, where *i* stands for the *i*-th chain, we may write

$$Dy = \sum_{i=1}^{s} \prod_{j=1}^{k(i)} C_{i1}\, C_{i2} \ldots C_{ij} \qquad (5)$$

where *i* ranges over the set of chains of *y*, *s* in number, and where $C_{ik(i)}$ is the derivative of the end of the *i*-th chain.

Thus the *Dy*-program is completed by construction of a program for evaluating (5). This can be done because the addresses $A_{\alpha_1 \ldots \alpha_n}$ are at hand in the locations $L(\alpha_1 \ldots \alpha_n)$.

The algorithm as it stands requires excessive storage. To differentiate any function composed with *n* operations, we should allocate 2^n locations for storing the addresses $A_{\alpha_1 \ldots \alpha_n}$, for there are as many of these as *n*-blocks of *1*'s and *2*'s. But consider the situation when the first chain has been completed.

At that point we know

$$\begin{aligned} Dy = {} & A_1 A_{11} \ldots A_{11 \ldots 11} A_{11 \ldots 111} \\ & + A_2 D(\mathit{2}) \\ & + A_1 A_{12} D(\mathit{12}) \\ & + \ldots \\ & + A_1 A_{11} \ldots A_{11 \ldots 1} A_{11 \ldots 12} D(\mathit{11} \ldots \mathit{12}) \qquad (6) \end{aligned}$$

The addresses of the *A*'s and *D*'s are consecutive cells in three blocks of storage, called the A_1-block, A_2-block, and *I*-block. The storage arrangement is shown schematically in Fig. 4. The lines in this figure indicate the formation of the products in (6).

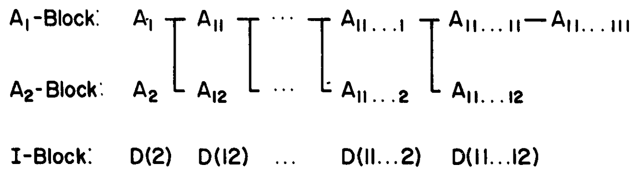

A1-Block: A1 A11 ... A11...1 A11...11—A11...111
A2-Block: A2 A12 ... A11...2 A11...12
I-Block: D(2) D(12) ... D(11...2) D(11...12)

Fig. 4—Contents of A_1-Block, A_2-Block, and *I*-Block of storage, at completion of a chain.

Instead of continuing by completing another chain, construct the program for the product of terms of the first chain just completed, and place this program in the *Dy*-block. Then the addresses $A_{11 \ldots 11}$ and $A_{11 \ldots 111}$ are no longer needed. We may replace $A_{11 \ldots 11}$ in $L(\mathit{11} \ldots \mathit{11})$ of the A_1-block by $A_{11 \ldots 12}$

and proceed to find $D(11 \ldots 12)$. The storage arrangement will now be as shown in Fig. 5.

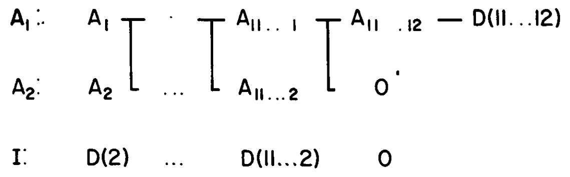


Fig. 5—Contents of storage blocks after down-dating of Fig. 4 arrangement.

The differentiation $D(11 \ldots 12)$ may generate further terms in the A_1, A_2, and I-blocks. Whenever an end is reached, we form the product for the completed chain, replace unnecessary A_1 terms by their corresponding A_2 terms, and then continue with differentiation of the next term, indicated by the last non-zero address in the I-block. This process of replacing terms no longer needed by terms of the next chain to be completed will be called *down-dating*. *Up-dating* refers to the process of adding new addresses to the A_1, A_2 and I-blocks, upon examination of each operation, as prescribed in Table I. These new addresses are stored in the appropriate blocks, and nowhere else. Since two cells are required for the storage of the A's from each operation, it is now only necessary to allocate $2n$ storage cells for saving the A's of any n-operation function.

An IBM 704 Program

A main feature of our IBM 704 implementation of the above algorithm is the pseudo-code used to specify functions. The function $A = B + C$ in 704 instructions would be

```
CLA  B
ADD  C
STO  A
```

But we do not actually need B in the accumulator, for we do not really intend to add numbers. The program is used only to recognize that B and C are composed by the binary operation addition, to form A. Thus a more compact code is possible, and desirable if the information we need is to be easily available. The code we use for $A = B + C$ is

$$A\colon \text{PZE } B, , C$$

That is, location A contains the addresses B and C in the address and decrement portion of the word, and the prefix PZE is used to indicate that these are composed with the binary operation of addition. The code for other operations is shown in Table II.

Table II also shows the detailed 704 version of Table I. When a function $K = u * v$ is differentiated, the construction of A_1 and A_2 and the updating of certain blocks of storage is specified by this table. After $K = u * v$ is differentiated, the next step is to find Du. In the flow chart of Fig. 6 "new K" refers to u.

TABLE II

UP-DATING OF Dy-PROGRAM, A_1-BLOCK, A_2-BLOCK, AND I-BLOCK

Function $K =$	Code at Location K	Dy-Program New Terms for Construction of A_1 and A_2	A_1	A_2	I
$u + v$	PZE u, , v	None	1	1	v
$u - v$	PON u, , v	None	1	-1	v
$u \cdot v$	PTW u, , v	None	v	u	v
$u \div v$	PTH u, , v	$L+0$: MON v, , $MFL1$‡ $L+1$: PTW K, , L	L	$-(L+1)$	v
u^v	MON u, , v	$L+0$: PON v, , $FL1$‡ $L+1$: MON u, , L $L+2$: PTW v, , $L+1$	$L+2$	0	0
exp u	MZE u, , 1	None	K	0	0
ln u	MZE u, , 2	L: MON u, , $MFL1$‡	L	0	0
sin u	MZE u, , 8	L: MZE u, , 16	L	0	0
cos u	MZE u, , 16	L: MZE u, , 8	$-L$	0	0

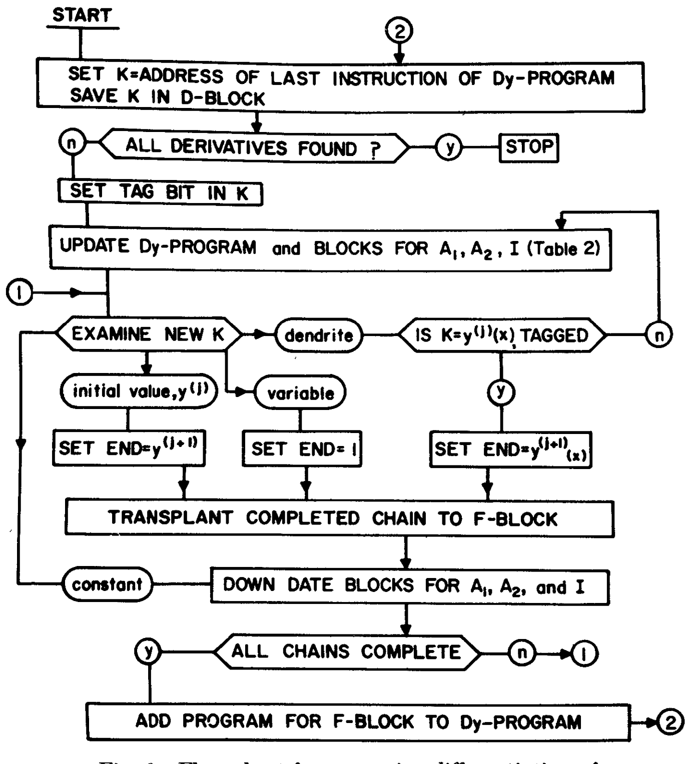
‡*MFL1* is the address of -1; *FL1* is the address of 1.

The 704 flow chart is clarified by a description of the roles played by certain blocks of storage.

(*a*) *Constants block.* All constants are given addresses of storage locations in this block.

(*b*) *Initial conditions block.* This contains the locations $y^{(j)}(0)$, $j \leqq k - 1$, as consecutive storage cells. When an end $y^{(j)}(0)$, $j < k - 1$, is recognized, its derivative is the address $y^{(j+1)}(0)$.

(*c*) *Variable of differentiation.* This is a single storage cell.



START
2
SET K=ADDRESS OF LAST INSTRUCTION OF Dy-PROGRAM SAVE K IN D-BLOCK
n
ALL DERIVATIVES FOUND ?
y
STOP
SET TAG BIT IN K
UPDATE Dy-PROGRAM and BLOCKS FOR A₁, A₂, I (Table 2)
1
EXAMINE NEW K
dendrite
IS K=y⁽ʲ⁾(x) TAGGED
n
initial value, y⁽ʲ⁾
variable
y
SET END=y⁽ʲ⁺¹⁾
SET END=1
SET END=y⁽ʲ⁺¹⁾(x)
TRANSPLANT COMPLETED CHAIN TO F-BLOCK
constant
DOWN DATE BLOCKS FOR A₁, A₂, and I
y
ALL CHAINS COMPLETE
n
1
ADD PROGRAM FOR F-BLOCK TO Dy-PROGRAM
2

Fig. 6—Flow chart for successive differentiation of $y^{(k)}(x) = f(x; y^{(0)}(x), \ldots, y^{(k-1)}(x))$

(*d*) *Function program block.* This contains the sequence of pseudo-instructions defining the function to be differentiated. The last pseudo-instruction is in $y^{(k)}(0)$. New terms for the construction of A_1 and A_2, as shown in Table II, are placed in the first available locations following $y^{(k)}(0)$, as needed. The program of pseudo instructions for formula (5) is also stored here, when all its terms have been constructed.

(*e*) *Derivative block, D.* The derivative of the initial condition $y^{(k-1)}(0)$ is $y^{(k)}(0)$, which is not an initial condition but an address in the function program block. The D-block cells contain addresses of $y^{(j)}(0)$, $j \geqq k$, stored in order, so that these may be treated in a manner similar to initial conditions.

(*f*) *A_1-block, A_2-block, and Instruction block I.* The roles played by these blocks are as described in connection with Figs. 4 and 5. In up-dating we add new terms as prescribed by Table II. In down-dating we eliminate terms that are no longer needed.

(*g*) *Factor block, F.* This saves all completed nontrivial (no zero factors) A_1 chains. In transplanting an A_1 chain into the F-block, all ones and minus ones are boiled down to a single sign for the entire product. All ones are omitted from the F-block, unless the particular product contains nothing but a single one.

In the flow chart of the 704 program, K stands for the address of some pseudo-order of the y-program currently under examination. The program starts with examination of the last K, $y^{(k)}(0)$. A tag bit in K will indicate that $Dy^{(k)}$ has been found, and is in the D-block, so that it need not be found over again when constructing higher derivatives.

The series construction, which involves multiplying each derivative by the appropriate power of $x - x_0$ and dividing by factorials, is straightforward, and is not shown in the flow chart.

The Taylor series solution of the differential equation which is finally obtained is in the form of a sequence of pseudo-instructions. It is always possible to convert these and print them on paper using familiar mathematical symbols, but we do not do this and will hardly ever want to. If a differential equation is sufficiently complicated to warrant using the program, the chances are that any significant information in the expression for the solution will be hidden in its complexity. If w is some complicated function of x, y, and z, $w = f(x, y, z)$, and we want to find out how w depends on x, it will do no good to inspect the expression f. Instead, we picture w versus x by evaluating f for a range of x values. Similarly, if we wish to study w versus y, we evaluate f with a range of y values. The point is, we need find the program for f only once. We may then evaluate it numerically as many times as we wish, illuminating the dependence on any desired variables.

To evaluate the solution obtained, it is necessary to convert the pseudo-code program to a regular

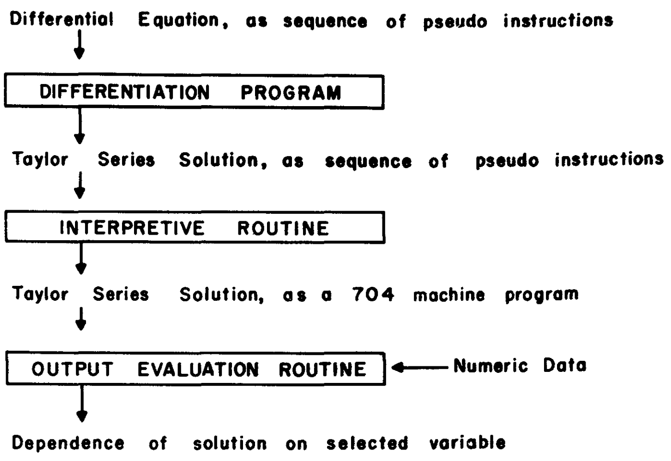

Differential Equation, as sequence of pseudo instructions
DIFFERENTIATION PROGRAM
Taylor Series Solution, as sequence of pseudo instructions
INTERPRETIVE ROUTINE
Taylor Series Solution, as a 704 machine program
OUTPUT EVALUATION ROUTINE
Numeric Data
Dependence of solution on selected variable

Fig. 7—Data flow for solution of differential equation.

machine-language code, and to supply numeric data. This is done by interpretive and output routines. The flow of information is shown in Fig. 7.

An example of the solution of a differential equation, showing the kind of information that can be obtained from these solutions, is shown in Figs. 8, 9, and 10.

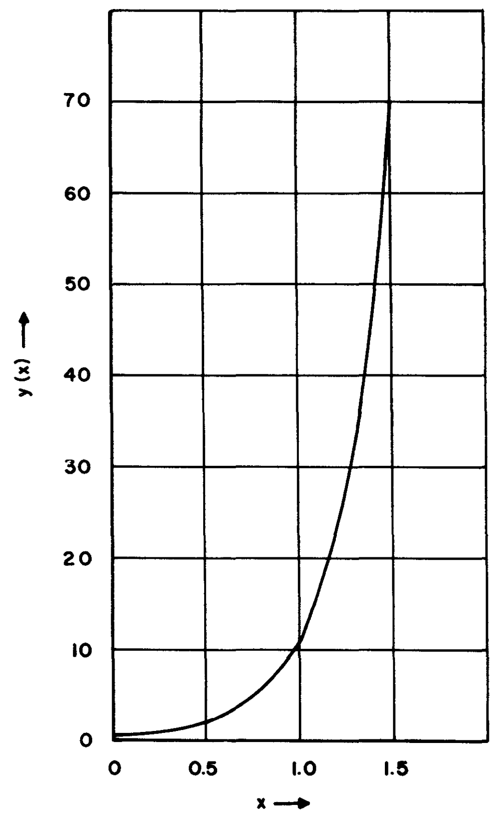


Fig. 8—Eight terms of series solution of $\frac{dy}{dx} = y^a + x$, $y(0) = 1$, $a = 2$.

Conclusion

We have described an analytic method for finding a series solution of ordinary differential equations on

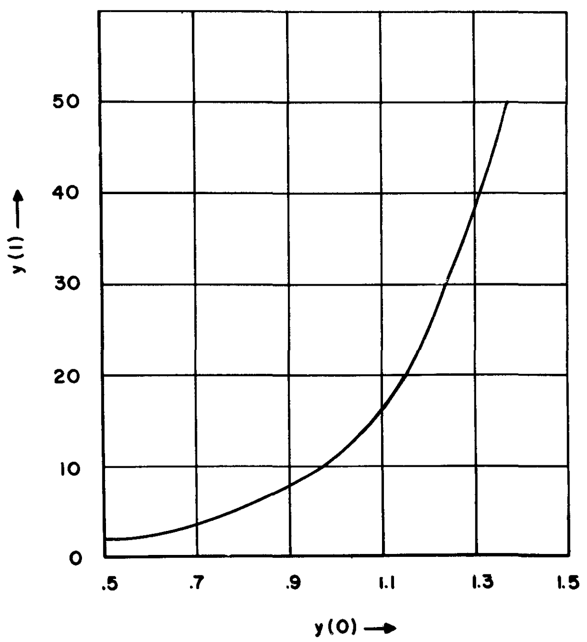


Fig. 9—Eight terms of series solution of $\frac{dy}{dx} = y^a + x, x = 1, a = 2.$

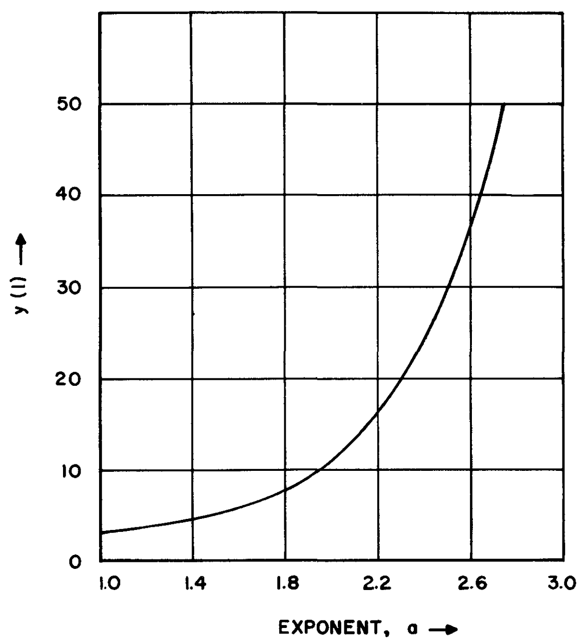


Fig. 10—Eight terms of series solution of $\frac{dy}{dx} = y^a + x, y(0) = 1, x = 1.$

a stored-program digital computer. Note, however, that the present IBM 704 program for implementing this method has room for improvement. Indeed, in the present program little attempt is made to simplify the generated derivative expression. This is a severe waste of storage capacity, and unduly limits the number of series terms that can be found. Further, the unsimplified expression, containing redundant and irrelevant terms, increases the machine time for evaluating a solution. For this reason, we cannot now obtain a significant estimate of the merit of the analytic method in comparison to conventional numeric techniques.

The method should be useful in illuminating local properties of solutions. It also appears to lend itself to extending solutions by analytic continuation, but this is a problem that has not yet been attacked.

Another needed improvement, if we are to handle the differential equations of electrical engineering practice, is the capability of handling simultaneous equations. The obvious modification to do this is to provide a separate function program and D-block for each differential equation of the system.

Acknowledgment

The research reported in this paper has been sponsored by the Electronics Research Directorate of the Air Force Cambridge Research Center, Air Research and Development Command, under contract AF 19(604)-4152.

The idea of using the computer to develop the series solution of a differential equation occurred in conversation with Ramon Alonzo.

The author also wishes to express his gratitude to Albert G. Engelhardt for substantial help in the planning and development of his work.

Normalized Floating-Point Arithmetic with an Index of Significance

HERBERT L. GRAY† AND CHARLES HARRISON, JR.†

General Remarks on Error and Significance

IT HAS BEEN frequently pointed out that the task of determining an error-bound for the results of a problem is usually a long difficult calculation, which is avoided as much as possible by the programmer. The introduction of floating-point arithmetic in modern computers and the ever-growing use of compilers makes the task of error analysis even more difficult and its computation even less probable. Clearly, a machine method is needed to automatically calculate a bound for the propagated and generated error, given the initial error in the input and the residual error due to approximating functions.

Two methods for doing this herein are called the "normalized significance" and the "unnormalized significance" methods. The "normalized significance" method always keeps the floating-point number normalized and provides an index of significance. The "unnormalized significance" method does not normalize floating-point numbers and uses the count of digits remaining after leading zeros as an indication of their significance.

Before discussing these methods, we should like to define what is meant by significance. If one says that α digits of a number are significant, one means that the error in the number is less than $B^{-(\alpha+n)}$ where B is the base and n is the number of leading zeros. In the normalized system, n would be zero. In the unnormalized system, n would be the total number of digits minus α, as there are no insignificant digits.

Thus, in the normalized method, as many digits as possible of a number are retained, and its index determines the number of digits which are significant; in the unnormalized method, only the digits considered significant are retained.

Since the error in any one step of a calculation is not usually a factor of the base, B, and α in either system only allows adjustment to within a factor of B, any set of arithmetic rules can have the desired property only "on the average". The design of such rules must rest upon general assumptions concerning the statistics of computation; these rules may, therefore, not be valid in special situations.

Normalized and Unnormalized Methods Contrasted

In the normalized method to be used with Argonne's arithmetic unit FLIP, each number is represented with base 2 by a triplet (x_f, x_p, x_i) where x_f is the fractional part of the number with $\frac{1}{2} \leq |x_f| < 1$, x_p is the associated power, and x_i is the index of significance. Thus $X = x_f \cdot 2^{x_p}$, to x_i significant figures. In addition and subtraction, the result has an index equal to the smaller of the two indices of the operands. Whenever a fraction is down-scaled, its index and power are increased; whenever a fraction is up-scaled its index and power are reduced. In multiplication and division, the index of the result is the smaller of the two indices; and even if the result needs to be scaled to bring it into the normal range, the index is not changed.

The "unnormalized method" is very similar to the one to be used by the University of Chicago in their new computer.[1] Each number is represented by a couplet (x_f, x_p), where $0 \leq |x_f| < 1$ and the number of digits of x_f minus the number of leading zeros equals x_i of the normalized scheme. Insignificant digits are shifted off the register. In addition and subtraction, no scaling of the result is carried out, unless, of course, its fraction is greater than or equal to unity. In multiplication and division, the resultant fraction is scaled so as to have the same number of leading zeros as the operand with the fewer number of significant figures.

In his error analysis of floating point procedures in the *Communications of the Association for Computing Machinery*[2], John W. Carr III points out that a normalized floating point procedure will always give a better result than an unnormalized procedure. While the normalized method will create less error, the loss of possible significant digits due to the shorter register-length remaining when an index of significance is used may outweigh this gain.

The logic for addition and subtraction, due to the normalization which a result may require and the handling of the index, are more complicated in the normalized method. However, division and multiplication are much more complicated in the unnormalized system because of the shifting required to obtain the correct number of leading zeros. Since the number of additions and subtractions in a calculation is usually greater than the number of multiplications and divisions, the unnormalized system might be a little faster.

* Work performed under the auspices of the U. S. Atomic Energy Commission.

† Argonne National Laboratory, Lemont, Ill.

[1] R. L. Ashenhurst, and N. Metropolis, "Unnormalized Floating Point Arithmetic," *Journal of the ACM*, July, 1959, pp. 415–429.

[2] Carr, John W. III, Error Analysis in Floating Point Arithmetic, *Communications of the ACM*, May, 1959, pp. 10–15.

Typical Examples

Following are some examples. Those numbered with (*a*) use the Argonne method, while those numbered with (*b*) use the second method. The length of the fractions is adjusted so that the register in each scheme contains the same number of digits.

(*1a*) (0.5782, 6, 3) + (0.1485, 4, 3): [Argonne scheme]

0.57820, 6, 3	Arithmetic register holds 1 more digit than storage.
+0.00148, 6, 5	Power and index adjusted.
0.5797, 6, 3	Result rounded, smaller index is used.

(*1b*) (0.00578, 8) + (0.00149, 6): [Other scheme]

0.005780, 8	Register again holds one extra digit.
+0.000014, 8	Power adjusted.
0.00579, 8	

(*2a*) (0.1397, 5, 4) − (0.9143, 4, 2):

0.13970, 5, 4	
−0.09143, 5, 3	Power and index adjusted.
0.04827, 5, 3	Smaller index used.
0.4827, 4, 2	Number normalized.

(*2b*) (0.01397, 6) − (0.00091, 7):

0.001397, 7	Power adjusted.
−0.000910, 7	
0.00049, 7	Number rounded to 2 significant digits.

(*3a*) (0.9143, 4, 2) + (0.1875, 4, 4):

0.9143, 4, 2	No power adjustment needed.
0.1875, 4, 4	
1.1018, 4, 2	Fraction exceeds one.
0.1102, 5, 3	Normalization and roundoff.

(*3b*) (0.00091, 7) + (0.01875, 5):

0.000910, 7	
0.000187, 7	
0.00110, 7	Rounded to 3 figures.

(*4a*) (0.5782, 6, 3) × (0.1485, 4, 4):
(0.5782, 6, 3) × (0.1485, 4, 4) = .08586270, 10, 3
[Double-register product]
= .8586, 9, 3
[Normalized and rounded]

Index not changed. The digit which was shifted into register was assumed to be good (optimistic approach).

(*4b*) (0.00578, 8) × (0.01485, 5):
(0.00578, 8) × (0.01485, 5) = 0000858330, 13
= .00858, 11

[Rule: result always contains same number of significant digits as the least significant operand.]

(*5a*) (0.5782, 6, 3) × (0.2485, 4, 4):
(0.5782, 6, 3) × (0.2485, 4, 4) = .14368270, 10, 3
= .1437, 10, 3
[Rounded]

(*5b*) (0.00578, 8) × (0.02485, 5):
(0.00578, 8) × (0.02485, 5) = .0001436330, 13
= .00144, 12
[Adjusted and rounded]

(*6a*) (0.5782, 6, 3) ÷ (0.2485, 4, 4)
(0.5782, 6, 3) ÷ (0.2485, 4, 4)
= 2.327605 . . . , 2, 3
= 0.2328, 3, 3
[Normalized and rounded]

Index does not change after normalization. Extra digit assumed to be no good (pessimistic approach).

(*6b*) (0.00578, 8) ÷ (0.02485, 5)
(0.00578, 8) ÷ (0.02485, 5) = 0.2325955 . . . , 3
= 0.00233, 5
[Adjusted and rounded]

Same rule holds for division as well as multiplication.

(*7a*) (0.2485, 4, 4) ÷ (0.5782, 6, 3)
(0.2485, 4, 4) ÷ (0.5782, 6, 3)
= 0.429782 . . . , −2, 3
= 0.4298, −2, 3
[Rounded]

(*7b*) (0.02485, 5) ÷ (0.00578, 8)
(0.02485, 5) ÷ (0.00578, 8) = 4.29930 . . . , −3
= 0.00430, 0
[Adjusted and rounded]

Remarks on Special Cases

In the test cases we have run so far using the Argonne scheme the number of multiplications far exceeds the number of divisions; this gives rise to optimistic indices, though none of the indices have been off by the equivalent of more than one decimal place. In both systems, a special number exists which does not obey the above rules. This number is zero and it follows the following rules in both systems.

$$X \pm 0 = X$$
$$X \times 0 = 0$$
$$0 \div X = 0$$
$X \div 0$ is not attempted.

If during the normalization process in the Argonne scheme, the index of significance is about to become less than zero, then the shifting is stopped so that the index remains at zero. Thus, at the end of this operation, the fractional part of the number may be less than one-half. This number, known as a relative zero, obeys the same rules as any normalized number during addition, subtraction and multiplication; but division by this number, as with true zero, is not attempted. Whenever the normalization is not completed, the Argonne machine will jump to a fixed location in the memory and let the program decide what is to be done about the number.

Relative zero may also be introduced as an input number. In this case, the fraction is unnormalized, but the index need not be zero. This number is used in special operations such as finding the integral part of another number, as follows:

```
  0.4597, 2, 4
 +0.0000, 4, 4
 ─────────────
  0.0045, 4, 4   Adjust power (index at max.)
  0.4500, 2, 2   Normalization.
```

This type of relative zero is also used in operations which convert from floating to fixed point. A relative zero with an index not equal to zero can never be generated by the floating-point unit.

The only other special number which may arise in the second scheme is a number with zero significant figures and thus a zero fraction. All operations with this number except division follow the normal rules.

If a number in the Argonne scheme does not have any associated error, it is said to be "totally represented". Such numbers are given a special index and operated on in a slightly different way. As long as a calculation uses only totally represented numbers, the index of the result does not change even if normalization occurs. If, however, during a calculation any non-zero digits of a number are shifted off, or if one of the operands is not a totally represented number, then computation of the index of the result reverts to its normal form. For example:

```
  (.4583, 2, TR)    (Totally represented)
 -(.4329, 2, TR)
 ───────────────
   .0254, 2, TR
   .2540, 1, TR     Normalization.

  (.4583; 2, TR)
 +(.1497, 1, TR)
 ───────────────
   .4583, 2, TR
   .01497, 2, TR    Adjust power
 ───────────────
   :4733, 2, 4      Results rounded to 4 figures.

   .4583, 2, TR
  -.4329, 2, 4      Not a totally represented number.
 ───────────────
   .0254, 2, 4      Index reverts to its normal form.
   .2540, 1, 3
```

Such a scheme helps to simplify floating-point-integer arithmetic.

Tests of Argonne Method in Representative Problems

To test the accuracy of the index in the floating-point scheme, we simulated FLIP with a GEORGE program. We combined this program with some GEORGE subroutines and compared the results obtained in this manner with those computed by other means.

Characteristic Polynomial of a Matrix

First, let us consider the routine involving the method of Leverrier for determining the coefficients of the characteristic polynomial of a real-valued square matrix.

Let λ_i = roots of the characteristic polynomial of our $n \times n$ matrix $\mathbf{A}$.

Define $S_k = \text{trace}\ (\mathbf{A}^k) \quad k = 1, 2, \ldots, n$

$$S_k = \sum_{i=1}^{n} a_{ii}^{(k)}$$

The characteristic equation is

$$\lambda^n + C_1\lambda^{n-1} + C_2\lambda^{n-2} + \ldots + C_{n-1}\lambda + C_n = 0$$

where

$$\begin{aligned} -C_1 &= S_1 \\ -2C_2 &= S_1C_1 + S_2 \\ -3C_3 &= S_1C_2 + S_2C_1 + S_3 \\ &\ldots\ldots\ldots\ldots \\ -nC_n &= S_1C_{n-1} + S_2C_{n-2} + \ldots + S_{n-1}C_1 + S_n \end{aligned}$$

The method requires the computation of $\mathbf{A}^2, \mathbf{A}^3, \ldots, \mathbf{A}^n$ and the corresponding $S_1, S_2, S_3, \ldots, S_n$. There are $n^3 (n - 1)$ multiplications and $n^3 (n - 1) + n (n - 1)$ additions involved in these calculations. Obviously, inaccuracies can be caused if n is very large or the elements of $\mathbf{A}$ are too large.

We ran tests with our combined program using four different matrices. In the first two cases tested, the $\mathbf{A}$'s were of the tenth order and of the forms:

$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ 2 & 2 & \ldots & 2 \\ . & . & \ldots & . \\ 10 & 10 & \ldots & 10 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 2 & 2 & \ldots & 2 \\ 4 & 4 & \ldots & 4 \\ . & . & \ldots & . \\ 20 & 20 & \ldots & 20 \end{pmatrix}$$

In both cases $C_1 = -S_1$ and $C_2 = C_3 = \ldots = C_{10} = 0$. We obtained exact results and the index indicated this for each coefficient.

Next we tried a twelfth-order matrix of the form

$$\begin{matrix}
4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4
\end{matrix}$$

Once again, we obtained exact results with the corresponding correct indices. It was with this case that we noticed the vast difference a change in method made in the final results. We computed the coefficients from the equation

$$C_k = (S_1C_{k-1} + S_2C_{k-2} + \ldots + S_{k-1}C_1 + S_k) \div (-k)$$

If, however, we use the slightly different equation

$$C_k = -\frac{1}{k}(S_1C_{k-1} + S_2C_{k-2} + \ldots + S_{k-1}C_1 + S_k) \, ,$$

we get different results. In fact, C_{12} is completely in error and the index associated with it is zero. This zero index means the number is worthless as a result.

The initial error was introduced in the computation of $1/k$, since non-terminating decimals like 1/3, 1/6, etc. cannot be exactly represented. The use of these inexact numbers produced inexact coefficients. The errors in both of these were propagated in the calculations of subsequent coefficients. As a result, C_{12} was completely in error.

The next matrix we used was a fifteenth order matrix having all its elements equal to 1. Even though the elements are not large, the size of n leads to some very large intermediate results. For example, $S_{15} = 15^{15} = 437{,}893{,}890{,}380{,}859{,}375$. The coefficients, however, were more reasonable. In fact $C_1 = -S_1$ and $C_2 = C_3 = \ldots = C_{15} = 0$. In this test we also found complete agreement with the indexed results and the known values for the coefficients.

Inversion of a Hilbert Matrix

Next we applied our program to Jordan's method for the inversion of Hilbert matrices of various orders. Hilbert matrices are of the form

$$\begin{pmatrix}
1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\
\frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\
\frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\
\cdot & \cdot & \cdot & \cdots & \cdots \\
\frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1}
\end{pmatrix}.$$

Hilbert matrices are almost singular and as the size of the order increases, the value of the determinant rapidly approaches zero. Several matrices were inverted including a 12×12 which had a determinant approximately equal to 10^{-75}. We compared the results we obtained with those computed by an inversion formula.[3]

The elements in our inverted matrices compared favorably with those in the table of inverses for all the orders tested. We were able to get 10- or 11-digit agreement in the elements of an inverted seventh-order Hilbert matrix and 6-or 7-digit agreement for a 10×10. The indices associated with these inverse elements were either correct or one-digit optimistic. This is about what we expected since we employ the optimistic approach in our floating-point scheme.

It may be interesting to note that Jordan's method on a floating-point machine produced inverses above the 6th order that were completely in error. Also, our GEORGE interpretive floating-point subroutine was useless beyond a 7th-order Hilbert matrix. The primary difference in these cases is the number of digits the machine can hold. However, the other two schemes give no indication of the approximate accuracy of the results.

Evaluation of a Polynomial

A third test was performed with a subroutine which evaluates a polynomial, with real or complex coefficients, having real or complex roots. We inserted known roots for certain 4th and 8th degree polynomials with real coefficients and noted the computed value of the polynomial.

When the roots were real and exact, the computed value of the polynomial was exactly zero. The same thing was true when the roots were complex but with exact real and imaginary parts. However, when the

[3] I. Savage and E. Lukacs, "Tables of Inverses of Finite Segments of the Hilbert Matrix" in *Contributions to the Solution of Systems of Linear Equations and the Determination of Eigenvalues*, NBS Applied Mathematics Series 39, p. 105–108, 1954.

roots were complex and at most one of the parts was exact we get an evaluation of the polynomial which is not exactly zero and has a zero index. Here, the zero index meant the digits in the number were meaningless but the zeros preceding the first of these digits were correct.

Numerical Integration

Finally, we applied our program to a trapezoidal integration subroutine. We evaluated some simple integrals between exact limits. If the results could be represented exactly, we obtained exact results. When the results could not be exactly represented we obtained numbers whose error was less than 5×10^{-18}. For these cases, the index on the partial sums was sometimes less than the maximum, but continued summations caused it to increase up to the maximum allowed for non-totally-represented numbers.

Conclusions

The tests we ran were considered fairly representative of the calculations involved in many problems. The fact that the index of significance was never more than one digit from its true value lends merit to the Argonne scheme. If some of the multiplications could have been replaced by divisions without increasing the errors, the index would have been even closer to the true value.

Determination of Optimum Production Tolerances by Analog Simulation

R. B. McGHEE† and A. LEVINE†

Introduction

THE ELECTRONIC analog computer has been widely utilized in recent years to obtain optimum parameter values for various types of control systems.[1,2] The procedure usually followed is to determine a control-system configuration by simplified analysis and then to program a computer in a realistic way, including all significant nonlinearities and appropriate random input disturbances. Next, a performance criterion of some sort is selected for the control system under study, and system parameters (gains, time constants, etc.) are varied until an optimum system is obtained. At this point, simulation activity usually ceases and is not resumed again unless the actual control system either cannot be constructed to obtain the desired parameter values or else fails to attain the level of performance indicated by the earlier simulation.

While this approach has produced many successful systems, it is the contention of the authors that it is seriously deficient in one respect: the essentially statistical nature of an actual system resulting from a manufacturing process is completely ignored. The representation of a control system as a differential equation with known coefficients is not realistic. An accurate simulation must effectively consider an ensemble of control systems, (*i.e.*, a random differential equation) characterized by parameters with common statistical properties. These statistical properties ordinarily take the form of production tolerances. Such tolerances must be considered concurrently with the search for optimum parameters if the simulator is to provide a reliable evaluation of the system performance.

The addition of random parameter values to a control-system simulation not only allows a more realistic determination of performance, but it also permits a more systematic and organized assignment of production tolerances. Conventionally, while the mean values for parameters are selected primarily on the basis of performance, the tolerances are selected primarily on the basis of cost. These two indices are somewhat at odds. For example, if cost were no object, but the desire to maximize performance dominated, then the ideal choice for each tolerance would be zero. On the other hand, if one desired simply to minimize cost per unit, infinite tolerances (one hundred percent acceptance of production) would be the proper policy. In the practical situation, a compromise between performance and cost must be effected. This compromise is too often accomplished on the sometimes dubious basis of intuition and "engineering judgment." It is unfortunate that such methods are used at a decision level so vital to the ultimate performance and cost of a product; more so, since the trade-off can be made more objective through analog simulation. By using a simulator to obtain performance as a function of the tolerance assignment, one can choose tolerances so as to maximize performance for a fixed cost or minimize cost for a fixed performance. In statistical terms, one can use the simulator to obtain optimum values for both the mean (nominal value) and standard deviation (production tolerances) for each important parameter.

A first step in this direction has been taken by R. C. Davis.[3] He postulated a linear dependence of a system performance index on parameter variances and obtained the necessary degradation coefficients by regression analysis of an analog computer experiment in which parameters were varied randomly. Although Davis did not explicitly mention cost, the coefficients he derived were utilized in juggling tolerances in such a way as to ease manufacturing difficulties without degrading performance. In principle, it would seem that these coefficients could be used to derive the original tolerances in an optimum manner. While Davis' method is probably the only sensible one if one is restricted to real-time simulation, it will be demonstrated with a simple practical example, that there are systems for which such a linearization can lead to wholly erroneous and misleading conclusions. The following paragraphs describe an analog simulation technique leading to an optimum tolerance-assignment which avoids the difficulties associated with an *a priori* assumption concerning the form of the functional dependence of performance on tolerances.

† Hughes Aircraft Co., Culver City, Calif.

[1] C. L. Johnson, *Analog Computer Techniques*, pp. 45–64, McGraw-Hill, New York, 1956.

[2] F. E. Nixon, *Principles of Automatic Controls*, pp. 287 to 305, Prentice-Hall, Englewood Cliffs, N. J., 1953.

[3] R. C. Davis, "A Statistical Method for Analyzing the Effects of Missile Guidance System Tolerance on Hit Probability," *Proc. Third Exploratory Conference on Missile Model Design for Reliability Prediction, White Sands Missile Range*, pp. 85–96, April 1959.

Proposed Tolerance Optimization Technique

Determination of the Performance Function

In order to arrive at an optimum set of tolerances, it is necessary first to determine the dependence of the control-system performance index, $P(\mathbf{Q})$, on the tolerance vector $\mathbf{Q}$. It is proposed that $P(\mathbf{Q})$ be determined by simulation according to the following procedure:

(1) Optimum nominal parameters are determined in the usual way.

(2) A tolerance vector, $\mathbf{Q}$, whose components are the standard deviations assigned to each parameter, is selected. This vector, together with the mean values determined in Step 1, is used to adjust filters operating on a stationary noise source so as to obtain voltages having the same mean and variance as the parameters under investigation.

(3) These voltages are fed to integrators as initial conditions while the computer is in *reset*. Each integrator in turn provides the input to a multiplier representing the value of the random coefficient in the simulation. When the computer is placed in *operate*, the integrators have no input. Thus, the value of the coefficient is fixed during the solution of the differential equation, but random when the computer is in *reset*. This effectively simulates a new production sample each time the computer is placed in operate.

(4) A large number of runs is made with a fixed tolerance-vector. An average performance-index is obtained for the whole set of runs (*e.g.*, probability of hit for a missile system).

(5) The components of the tolerance vector are varied in a systematic way, with a large number of computer runs being made for each new set of tolerances.

(6) The results are plotted with the aid of regression analysis using a digital computer. An arbitrarily high order of polynomial fit may be used if sufficient data is taken on the analog computer.

The enumerated steps lead to a representation of a system performance-function, $P(\mathbf{Q})$, without an *a priori* assumption regarding the form of the functional dependence of the system performance on parameter values.

It should be noted that step 5 may easily involve hundreds of thousands of runs. For this reason, it appears that the proposed simulation technique is economically feasible only when high-speed analog computers such as the GPS (General Purpose Simulator Instruments Company) or Philbrick Repetitive Computer are employed.[4] These computers are capable of providing solutions at rates up to 50 cps and can compute average performance indices simultaneously with solution of the differential equations of the control system.

Determination of the Cost Function

Once $P(\mathbf{Q})$ is obtained, it is necessary to determine the dependence of the unit cost of the system, $C(\mathbf{Q})$, on the tolerance assignment. This is a difficult problem which has long concerned manufacturing industries. It does not appear that analog simulation can aid in the determination of $C(\mathbf{Q})$. Nevertheless, $C(\mathbf{Q})$ is required in order to optimize $\mathbf{Q}$, so it is therefore necessary to discuss the estimation of costs sufficiently to indicate the difficulties to be expected and the general nature of the cost function.

Among the factors to be taken into account in the determination of $C(\mathbf{Q})$ are the raw cost per unit, salvage value, maintenance costs resulting from tolerances too wide or too narrow, lifetime, etc. Initial estimation of $C(\mathbf{Q})$ based on such considerations can probably best be accomplished by reference to manufacturing experience with previous products similar to the one under consideration. As production proceeds, the original estimate may be refined on the basis of actual costs. In addition to the experience accumulated by various manufacturing organizations, considerable literature exists on the subject of cost estimation.[5,6]

While the general problem is much too extensive to treat here, one can observe from experience that $C(\mathbf{Q})$ is generally hyperbolic in nature. That is, reduction of any element of $\mathbf{Q} = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ to zero will result in infinite cost while minimum cost is achieved by making all tolerances infinite.

Optimization of the Tolerance Vector

After $P(\mathbf{Q})$ has been determined, the optimization of $\mathbf{Q}$ may be considered. The first step required is to select a criterion for optimization. An obvious criterion is to optimize performance for a fixed cost. That is, on a surface of constant cost in the $\mathbf{Q}$-space, one seeks the point of maximum performance. On the other hand, it might be more desirable to minimize cost for a fixed performance. More complicated optimization criteria are possible by specifying the cost one is willing to accept as a function of performance, etc.

As more data becomes available, the estimate of the cost function will change, thereby shifting the

[4] Johnson, op. cit., pp. 222–232.

[5] D. H. Evans, "Optimum Tolerance Assignment to Yield Minimum Manufacturing Cost," *Bell System Technical Journal*, Vol. 37, pp. 461–484, March 1958.

[6] E. W. Pike and T. R. Silverberg, "Assigning Tolerances for Maximum Economy," *Machine Design*, Vol. 25, p. 139, Sept. 1953.

optimum. However, *the original performance function, $P(\mathbf{Q})$, remains valid; so the apportionment of tolerances may be altered to a new optimum without further simulation.*

Example

Description of Simulation

In order to establish the feasibility of the proposed technique, a simulation of a hypothetical radar-homing missile was undertaken. Besides random parameters, this simulation included random radar tracking noise and a random missile-heading error at launch. The performance index selected was probability of hit, P_h as measured against an idealized strip target. To limit the amount of data to be taken, only two parameters were varied randomly. These parameters were the navigation gain, K, and the principal missile filtering-time constant, τ. The missile which was simulated derived steering information from the angular rate of the line of sight from missile to target, so K is given by

$$K = \frac{\text{steady-state missile turning-rate}}{\text{line-of-sight angular rate}} \tag{1}$$

Fig. 1 is a functional block diagram of the missile simulated.

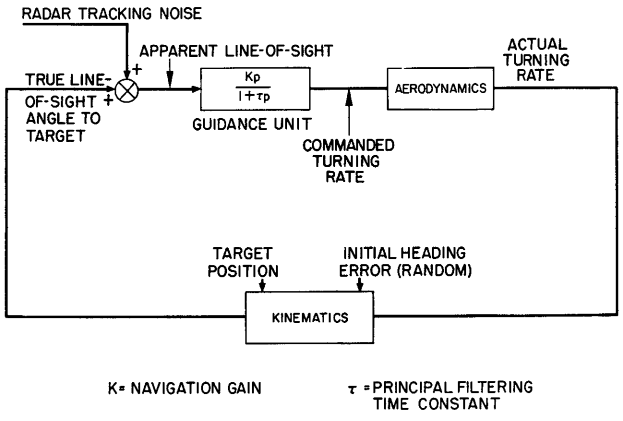


Fig. 1—Radar homing missile navigating in a plane.

To measure P_h with sufficient precision, it was necessary to simulate five hundred attacks for each value of the tolerance vector $\mathbf{Q} = (\sigma_K, \sigma_T)$. By considering four values for each tolerance, a total of sixteen cases was generated, resulting in a grand total of eight thousand simulated attacks. Since a high-speed computer was not available at the time, this simulation was performed on a real-time basis using automatic sequencing equipment. The total running time was about one week. Thus, this example shows quite clearly that a problem of realistic complexity will require high-speed simulation.

In this simulation, it was assumed that the population statistics for both K and τ were Gaussian with mean values K_0 and τ_0 and standard deviation σ_K and σ_T respectively. The sample values were therefore obtained by low-pass filtering of noise generators, using integrators which did not reset. Fig. 2 shows the computer program used to randomly vary the navigation gain, K, from run to run.

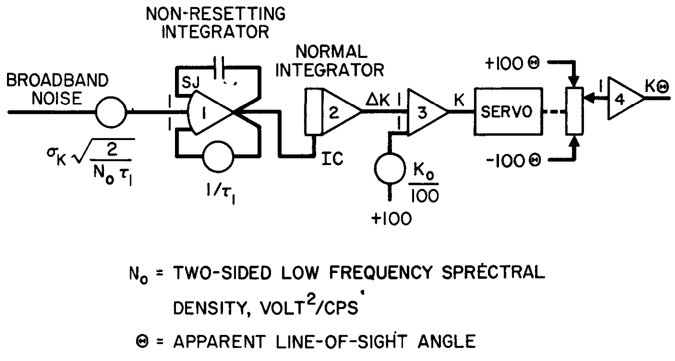

NON-RESETTING INTEGRATOR
NORMAL INTEGRATOR
BROADBAND NOISE
$\sigma_K\sqrt{\frac{2}{N_0\tau_1}}$
SJ
1
1/τ1
2
IC
ΔK
3
K
K_0/100
+100
+100 Θ
SERVO
-100 Θ
4
KΘ
N_0 = TWO-SIDED LOW FREQUENCY SPRĔCTRAL DENSITY, VOLT²/CPS
Θ = APPARENT LINE-OF-SIGHT ANGLE

Fig. 2—Random variation of navigation gain.

σ_T \ σ_K	0%	10%	20%	30%
0%	0.34	0.30	0.31	0.29
10%	0.30	0.31	0.29	0.29
20%	0 30	0.26	0.30	0.30
30%	0.24	0.22	0.28	0.27

Fig. 3—Measured probability of hit as a function of parameter standard deviations.

Fig. 3 is a table of the results obtained in the simulation. This table presents the measured P_h as a function of the percentage standard deviation assigned to the randomized parameters, K and τ. The raw data of this table was subjected to quadratic regression analysis on σ_K and σ_T using a digital computer; *i.e.*, the data points were fitted in a least-square sense by

$$P_h = P_0 + a_1\sigma_K + a_2\sigma_T + a_{11}\sigma_K{}^2 + a_{12}\sigma_K\sigma_T + a_{22}\sigma_T{}^2 \tag{2}$$

The results are plotted in Fig. 4. Fig. 5 is a contour map of Fig. 4 with hypothetical cost curves included. Examples of minimum-cost and maximum-performance optima are indicated.

Significance of the Experimental Results

The first and most obvious observation that can be made from Fig. 4 is that linearization of the dependence of P_h on σ_K and σ_T is impossible. Linear dependence would result in equally spaced, parallel,

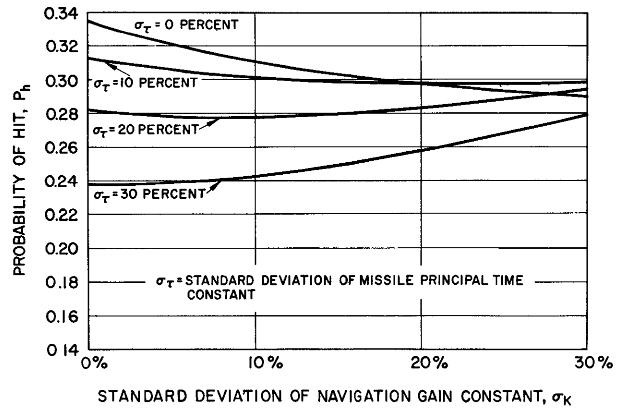


Fig. 4—Dependence of probability of hit on tolerance assignment for a homing missile.

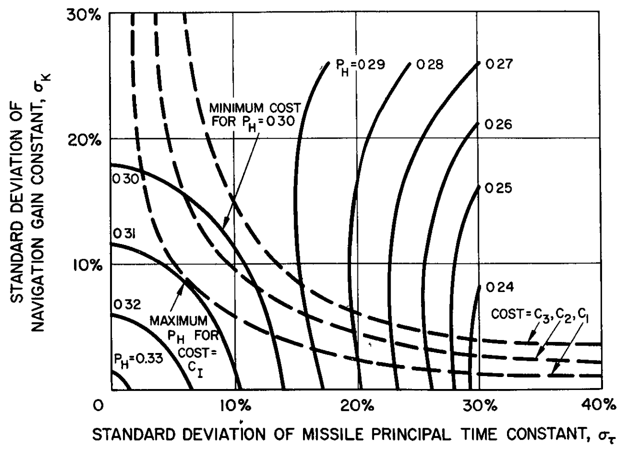


Fig. 5—Optimization of tolerances for minimum cost or maximum performance.

straight lines on Fig. 4. Replotting with σ_K^2 as the abscissa would show that linearization of the dependence of P_h on σ_K^2 and σ_T^2 is also impossible. Further examination of the figure reveals the remarkable fact that the performance of this system can actually be degraded by tightening tolerances. For example, when $\sigma_T = 30$ percent, the probability of hit is decreased from .28 to .24 as the tolerance on σ_K is reduced from 30 percent to 0 percent. This is quite at odds with the widespread opinion that when tolerances are relaxed in one part of a system, they must be tightened elsewhere to maintain the system performance. In the present example, this apparently anamolous behavior results from a cancellation of effects when K and τ both vary widely from their nominal values.

A more general and far reaching conclusion arising from this example is concerned with the form of the mathematical model selected to represent the dependence of the system-performance index on parameter *values*. In the more conventional approach to the study of tolerances, one begins by assuming a specific model for this dependence.[7] For example, on the basis of experience gained in the optimization of nominal values, the system designer might decide that an adequate model for the relationship between P_h and the values of the parameters, K and τ, is given by the quadratic expression:

$$P_h = \gamma_0 + \gamma_1(K - K_0) + \gamma_2(\tau - \tau_0) + \gamma_{11}(K - K_0)^2 + \gamma_{12}(\tau - \tau_0)(K - K_0) + \gamma_{22}(\tau - \tau_0)^2 \quad (3)$$

Now since the values of K and τ were independent in this experiment (and very often in practice) the average or expected value of P_h measured over the ensemble of missiles is given by

$$P_h^{ave} = E(P_h) = \gamma_0 + \gamma_{11}E(K - K_0)^2 + \gamma_{22}E(\tau - \tau_0)^2 \quad (4)$$

where the symbol E denotes the expected value or theoretical average resulting from repeated estimation of P_h. The other terms vanish because they have zero expected value. Now, by definition, the ensemble averages in the above expression are just the variances; *i.e.*,

$$E(K - K_0)^2 = \sigma_K^2 = \text{variance of } K \quad (5)$$

$$E(\tau - \tau_0)^2 = \sigma_T^2 = \text{variance of } \tau \quad (6)$$

so $$P_h^{ave} = \gamma_0 + \gamma_{11}\sigma_K^2 + \gamma_{22}\sigma^{k2} \quad (7)$$

This is the analytic origin of the common assumption that performance depends linearly on the variances. If one accepts this assumption, then all that is required is to perform an experiment, selecting many values of the variables and recording the performance index along with the values of the variables. Regression analysis is then accomplished on equation (3) yielding the desired coefficients, γ_0, γ_{11}, and γ_{22}.

While the above approach will succeed if the assumed model (3) is in fact correct, it has already been shown that the resulting conclusion of linear dependence of performance on parameter variances cannot possibly describe the behavior of the example at hand. The reason is that the effect of the terms σ_1, σ_2, and $\sigma_1\sigma_2$ are absent from (7) whereas a regression analysis of the experimental results based on (2) shows that $\sigma_1\sigma_2$ is in fact the most significant term. This term represents interaction effects which are of prime importance in this system. The actual financial cost of ignoring this term could be considerable since utilization of (7) would lead both to unnecessarily tight tolerances on K and to degraded performance when σ_T is large.

The process of deriving the effects of tolerances from a model for performance-dependence on parameter values can be turned around. That is, if one knows the coefficients in an equation like (2),

$$P_h = a_0 + a_1\sigma_K + a_2\sigma_T + a_{11}\sigma_K^2 + a_{12}\sigma_K\sigma_T + a_{22}\sigma_T^2 \quad (2)$$

then the effect of parameter values can be inferred (though not uniquely). For example, if it were true

[7] Evans, op. cit.

that

$$P_h = a_0 + a'_1 \mid K - K_0 \mid + a'_2 \mid \tau - \tau_0 \mid + a_{11}(K - K_0)^2 + a'_{12} \mid K - K_0 \mid \mid \tau - \tau_0 \mid + a_{22}(\tau - \tau_0)^2 \quad (8)$$

then (2) would follow and the coefficients in (8) could be obtained from a regression on tolerances such as was performed in the present example.

To summarize, in the light of the preceding discussion, it is maintained that one should in general reverse the usual approach to tolerance analysis by:

1. Performing a simulation with randomized coefficients as described in the section above on Tolerance Optimization.
2. Derive the mathematical model for the system from a regression analysis of the dependence of system performance on the tolerance vector.

In this way, new insight can be obtained into the relationship between performance and parameter values, and serious errors in tolerance assignment can be avoided.

Design of Experiments

The Need for Experimental Design

The experiment just described involved only two parameters. In a more realistic problem, it is likely that a much larger number of system parameters would be considered in the assignment of optimum tolerances. Utilization of the method presented in this paper leads to an alarming increase in the total number of runs as the number of parameters increases unless the experiment is carefully designed. The problem of determining just how to take data so as to obtain the most information with a minimum of effort falls within the realm of statistical design of experiment. This field is so extensive that it is neither possible nor desirable to consider it in detail in this paper. Fortunately, there are several excellent texts on the subject which treat a variety of types of experiment.[8]

While a general discussion of design of experiment is not appropriate, there are two aspects of this field which are particularily useful in experiments of the type discussed in this paper. In the first place, it is necessary to be able to estimate the total amount of data required to define $P(\mathbf{Q})$ before beginning the experiment. Secondly, in the event that this number is too large, it is very desirable to have a means for reducing this total by a judicious choice of data points in the tolerance-vector space. Both of these problems are considered below in connection with the present experiment.

Estimation of the Amount of Data

In the missile simulated, the measured values of the performance index, P_h, possess a binomial distribution, because they result from repeated trials of the same experiment. One can therefore estimate the variability of successive measurements of P_h.[9] It turns out that if P_h is the true or theoretical probability of hit and $\hat{P}_h$ is the measured ratio of hits to total firings, then

$$P_h = \hat{P}_h \pm 1/\sqrt{n} \quad (9)$$

n = total number of simulated firings

with probability .95 or better. Consequently, since the data in Fig. 3 was obtained by making five hundred simulated firings per data point,

$$P_h = \hat{P}_h \pm 1/\sqrt{500} \cong \hat{P}_h \pm .045 \quad (10)$$

That is, we can have at least ninety-five percent confidence that the true P_h lies within $\pm .045$ of the values listed in Fig. 3.[10] Utilization of quadratic regression (2) considerably improves the confidence we may have in the final result of Fig. 3.

If one were to decide that a precision of $\pm .045$ with ninety-five percent confidence was required of every point, and furthermore, that m values of each of k parameters must be considered, then for this example the total number of runs would be given by

$$n = 500\, m^k \quad (11)$$

This number may be so large as to make a "brute force" determination of $P(\mathbf{Q})$ impractical, even with a high speed computer. In such an event, some means such as the experimental design described below must be found for reducing this total.

Factorial Design

In the example treated in this paper, the data points were taken at uniformly spaced points in a σ_K, σ_T space (Fig. 3). Fortunately, it is not necessary to take data at all of these points to determine a quadratic relationship between $P(\mathbf{Q})$ and $\mathbf{Q}$. One type of design of experiment which allows fewer data points to be considered is called "fractional factorial design".[11] This design method involves a sequential determination of the points at which observations should be made, and is particularly well suited to the determination of $P(\mathbf{Q})$. The total number of computer runs may be drastically reduced, amounting possibly to several orders of magnitude in the total effort required to define $P(\mathbf{Q})$. While the complete

[8] W. G. Cochran and G. M. Cox, *Experimental Designs*, John Wiley and Sons, New York, 1950.

[9] P. G. Hoel, *Introduction to Mathematical Statistics*, John Wiley and Sons, New York, Chapter 10, 1947.

[10] Methods also exist for the estimation of confidence intervals in the event that the variables do not possess a binomial distribution. *Ibid.*, Chapter 10.

[11] J. S. Hunter, "Designing and Interpreting Tests," *Control Engineering*, pp. 137–141, Sept. 1959.

σ_T \ σ_k	0%	30%
0%	0 34	0 29
30%	0 24	0.27

Fig. 6—Measured probability of hit as a function of parameter standard deviations.

theory is somewhat involved, Fig. 3 can be used to illustrate the basic idea of fractional factorial design.

Referring to the experimental example, if one knew the relative importance of the terms σ_K, σ_T, and $\sigma_K\sigma_T$ in determining $P(\mathbf{Q})$, then a more intelligent selection of sampling points could be made. Now an *estimate* of the relative effects can be obtained by evaluting $P(\mathbf{Q})$ only at the vertices of the table of Fig. 3. Fig. 6 is an abbreviated table, obtained by using only the four corners of the other table. The relative effect of each of the three terms above is obtained as follows:

(1) Estimated effect of σ_K = average of P_h over column (2) − average P_h over column (1) = $(.29 + .27)/2 - (.34 + .24)/ = -.01$

(2) Estimated effect of σ_T = average of P_h over row (2) − average P_h over row (1) = $(.24 + .27)/2 - (.34 - .29)/2 = -.06$

(3) Estimated effect of $\sigma_K\sigma_T$ = average difference of row (2) − average difference of row (1) = $(.27 - .24)/2 - (.29 - .34)/2 = .04$

Thus it can be seen that the "interaction effect" of $\sigma_K\sigma_T$ is greater than the "main effect" due to σ_K and nearly as great as the "main effect" due to σ_T. While this result is obtained from only four data points, it is (qualitatively) quite in accord with the final result obtained by regression analysis (Fig. 4). The significance of this test is simply that the $\sigma_K\sigma_T$ term cannot be dropped from the regression equation (2). There is, however, a possibility of dropping the linear term σ_K. If this were done, then less data would be required to estimate the remaining regression coefficients. A more sophisticated approach to fractional factorial design must consider confidence intervals and other effects to determine whether or not a given term can be dropped from (2).

When a large number of parameter tolerances are involved in $\mathbf{Q}$, the dropping of even one term from the regression equation may provide a considerable saving in the simulation effort required to obtain $P(\mathbf{Q})$. Even if no terms can be dropped from the regression equation, uniformly-spaced data points do not provide the best grid for measurement of $P(\mathbf{Q})$. Further development of fractional factorial design provides a solution to the problem of determining the optimum location for data points in the parameter-tolerance space.

Special Simulation Considerations

Correlation Between Successive Parameter Values

In order to simplify the analysis of data, it is desirable that successive values of parameters selected from noise generators be uncorrelated. For noise sources shaped by first-order filtering, the correlation of output samples separated by time T is given by

$$\rho(T) = e^{-|T|/\tau_1} \qquad (12)$$

where τ_1 is the filter time constant.[12] Consequently,

Multiplexing of Noise Generators

In a problem of realistic complexity, a sizable number of uncorrelated noise voltages are required to provide parameter values. These voltages can be obtained from a single noise generator by frequency-multiplexing techniques. Rice has shown[13] that an ensemble of samples drawn from a Gaussian noise generator can be described by a Fourier series with uncorrelated random coefficients. Consequently, if such a signal is passed through a bank of bandpass filters, the correlation between filter outputs can be made as small as desired by reducing the spectral overlap of the filters sufficiently. In this process, the correlation between successive samples from a single filter must also be considered. This factor sets a lower limit on the bandwidth of the individual filters.

High-Speed Data Processing

When solution rates are as high as the 50-cps capability of repetitive computers, data reduction must be at least partially accomplished by the computer. One such computer (GPS) has a probability-distribution analyser which automatically provides probability of hit. This means that there is no need to process individual solutions off the computer. It appears that such an instrument or a related one is essential in simulations where hundreds of thousands of runs are required.

Conclusions

The role of simulation in the preliminary design of control systems can be expanded to include selection of optimum manufacturing tolerances. If a high-speed computer is available, one may abandon linear models for the effect of varying tolerances and thereby obtain a realistic method for performing the crucial trade-off between cost and performance. A lesser effort is apt to lead to false conclusions concerning system performance.

[12] J. L. Lawson and G. E. Uhlenbeck, *Threshold Signals*, p. 42, McGraw-Hill, New York, 1950.

[13] S. O. Rice, "Mathematical Analysis of Random Noise," *Selected Papers on Noise and Stochastic Processes*, N. Wax, ed., pp. 157–161, Dover Publications, New York, 1954.

The Crossed-Film Cryotron and Its Application to Digital Computer Circuits

V. L. NEWHOUSE,† J. W. BREMER† AND H. H. EDWARDS†

INTRODUCTION

THE NAME *cryotron* was applied by the late D. A. Buck to the superconductive relay which he described.[1] Buck's cryotron consisted of a wire of tantalum surrounded by a coil of niobium and was operated at the boiling point of liquid helium at atmospheric pressure. At this temperature the tantalum "gate" wire was only just superconducting. By passing a sufficient current through the niobium control coil, a magnetic field was created which was sufficient to transform the tantalum gate to its resistive state.

It was found by two of the authors that it is possible to produce the cryotron in a geometry suitable for deposition on a flat surface.[2] This will be referred to as the crossed-film cryotron (CFC). The CFC was first presented in public at the Electron Device Research Conference at Ithaca in June 1959. Similar devices were described at the same meeting by M. L. Cohen, J. L. Miles, A. E. Slade and C. R. Smallman of the A. D. Little Company, and by A. E. Brenneman and R. de Lano of the IBM Research Center.

This paper describes a crossed-film cryotron deposited on an insulated superconductor. This CFC has a time constant of less than one microsecond and is approximately one hundred times faster than the original vacuum-deposited cryotron.[2] The d-c dissipation is less than 5 microwatts and the active area of each element is approximately 5×10^{-4} square centimeters. These cryotrons and all their interconnecting circuitry can be vacuum deposited at one and the same time in a few simple steps.

The cryotrons can be applied to both switching and storage. Some experimental storage and shift-register circuits are described, which demonstrate a circuit property unique to superconductors. A shift-register circuit is shown which is deposited in an area corresponding to 20,000 active elements per square foot.

Calculations are presented which show that with this component density, a computer or memory containing more than one million elements can be accommodated in a one-cubic-foot liquid helium container using presently available refrigeration methods.

† General Electric Company, Schenectady, N. Y.

[1] D. A. Buck, "The Cryotron — A Superconductive Computer Component," *Proc. I.R.E.*, Vol. 44, pp. 482–493, 1956.

[2] V. L. Newhouse and J. W. Bremer, "High-Speed Superconductive Switching Element Suitable for Two-Dimensional Fabrication," *J.A.P.*, Vol. 30, p. 1458, Sept. 1959.

SUPERCONDUCTIVE FILMS

The devices to be described are made up of tin, lead and insulator films only. Of these, only the tin films change their state during operation. We can, therefore, confine our attention mainly to tin films. At temperatures below the so-called critical temperature T_c, tin and lead become superconducting. For lead, $T_c = 7.2°K$. For the tin films used, $T_c \approx 3.75°K$. The devices described are operated at approximately 3.6°K. At this temperature, the tin films can readily be switched from the superconducting to the normal (resistive) state, but the lead films remain superconducting throughout.

Just as in the case of bulk materials, it is possible to restore a superconductive film to the normal state by the application of a magnetic field greater than the so-called critical field H_c. The variation of H_c for bulk tin is shown in the insert of Fig. 1.

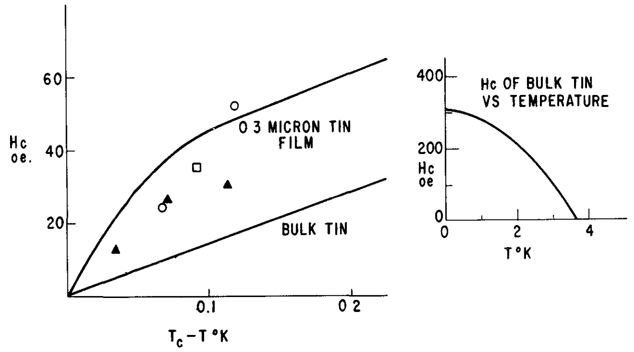


Fig. 1—Critical field of gate film as a function of temperature. Solid line: measured directly. Points: calculated from cryotron characteristics and grid widths, using Eq. 4. Grid widths: ▲ 16 microns (unshielded), □ 65 microns (shielded), ○ 40 microns (shielded).

The main portion of Fig. 1 compares H_c of bulk tin and H_c of a 0.3-micron-thick tin film. The film curve was determined experimentally, with a uniform magnetic field applied parallel to the film surface. The data points shown in the figure are values of H_c calculated from the electrical characteristics of crossed-film cryotrons. These are discussed in connection with Eq. (4) below.

Fig. 1 shows that H_c of the film is higher than for the bulk material. It can be established on the basis of thermodynamics[3] that if the film thickness is of the same order of magnitude as, or less than, the penetra-

[3] D. Shoenberg, *Superconductiviiy*, pp. 171–174, 2nd edition, Cambridge Univ. Press, 1952.

tion depth, H_c varies inversely with film thickness. The penetration depth is roughly equal to the thickness of the surface layer in which the current flows in a bulk superconductor. For tin, the penetration depth at absolute zero is approximately 5×10^{-6} cm, but at temperatures close to T_c, it is larger. At 3.6°K, a typical operating temperature for a crossed-film cryotron, the penetration depth is about 0.1 micron.

The variation of resistance with current for a 0.3-micron tin film is shown in Fig. 2. By applying the current in short pulses, it is possible to obtain the so-called isothermal transition shown in the broken line. This curve is connected with the actual superconducting behavior of the film, and is reasonably independent of other film characteristics, such as resistivity, and of the substrate properties. If a *slowly*-rising current is passed through a film, Joule heating causes thermal "propagation" of resistive areas in the film.[4] This behavior is shown in the solid curve of Fig. 2, and is strongly dependent on substrate thermal conductivity and on film resistivity.

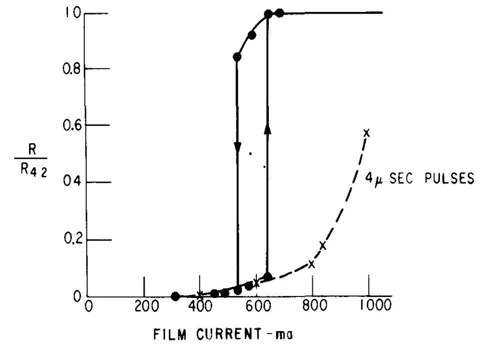


Fig. 2—D-c and pulse current-induced transitions for 0.3 micron thick, 4.05 mm wide tin film on sapphire substrate. ▲T = 0.08°K.

The current at which resistance first appears is known as the critical current I_c. For thin films $I_c = i_c W(T_c - T)$ where W is the film width and T the bath temperature, provided that $T_c - T \ll T_c$.[5] i_c increases as film thickness increases and appears to depend somewhat on heat treatment and film substrate. It has been found that i_c is more than doubled if the film in question is deposited on top of an insulated lead "shield" plane.

The explanation of why a tin film which lies adjacent to a lead shield plane has a higher critical current than a similar tin film deposited on glass is believed to be as follows: It can be shown[6] that a film in the shape of a cylinder will carry twice as much current as the same film unwrapped into a flat plate. When current passes through a tin film adjacent to a superconducting shield, surface currents are induced in the shield to prevent flux from penetrating into it. It can be shown that these surface currents double the field between the film and the shield, and produce an approximately zero field on the opposite side of the film. This field configuration is the same as would occur if the tin film were in the shape of a cylinder. It is to be expected, therefore, that the critical current for a shielded flat film is increased from the value for the unshielded flat film to that for the cylinder.

The mathematical problems of calculating the surface currents induced in a superconducting surface due to the presence of an external current-carrying conductor are similar to the problems of calculating the surface charge produced in a perfect conductor due to an external charge. It is found that some of the results of the "method of images" of electrostatics can be carried over to superconductors if an electronic dipole is replaced by a magnetic dipole, and a line of charge by a line of current. For a current-carrying wire above a superconducting surface, for instance, it can be shown that the net field outside the surface is equal to the field of the original current plus that of an equal and opposite shielding current which is the same distance behind the superconducting surface as the real current is in front. This effect increases the field between the current and the surface, but reduces it everywhere else. (It is assumed that the maximum net field is less than the critical field of the superconducting shield plane.) It can be seen therefore that if it is desired to reduce the effective inductance of a wire or length of film, it is simply necessary to place a superconducting plane with a high critical field in close proximity.

The Crossed-Film Cryotron

The basic structure of a crossed-film cryotron (CFC) is shown in Fig. 3. If a sufficiently large current is passed through the "grid" film, the resulting magnetic field produces a resistive channel across the much wider tin "gate" film. The grid remains super-

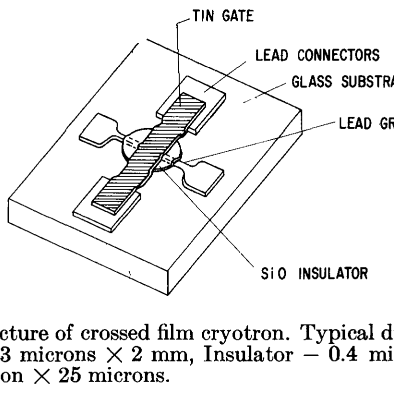


Fig. 3—Structure of crossed film cryotron. Typical dimensions: gate film — 0.3 microns × 2 mm, Insulator — 0.4 microns, grid film — 1 micron × 25 microns.

[4] J. W. Bremer and V. L. Newhouse, "Thermal Propagation Effect in Thin Superconducting Films," *Phys. Rev. Letters*, Vol. 1, p. 282, 1958.

[5] J. W. Bremer and V. L. Newhouse, "On Current Transitions in Superconductive Tin Films," *Phys. Rev.*, to be published.

[6] V. L. Ginzburg, "Critical Currents in Superconducting Films," *Soviet Physics "Doklady,"* Vol. 3, p. 102, 1959.

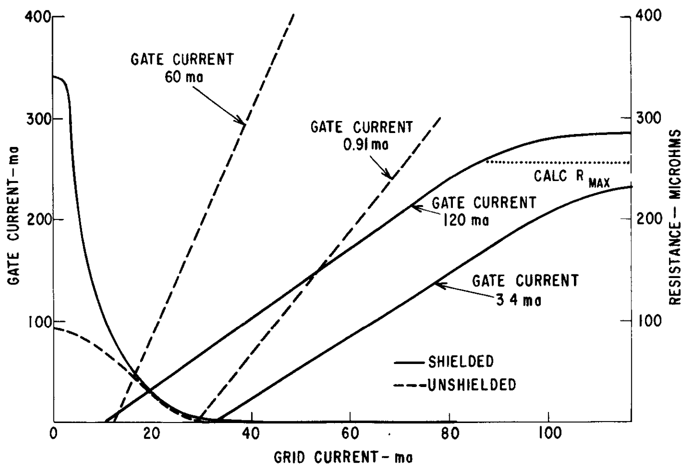


Fig. 4—Comparison of electrical characteristics of unshielded and shielded CFC. Shield insulation thickness = 4 microns. Grid width — 30 microns, gate width — 2 mm, $T_c - T = 0.07°K$.

conducting at all times. The characteristics of the unshielded cryotron are shown in the broken curves of Fig. 4. The curve intersecting the left-hand ordinate at 91 ma. shows values of gate and grid current at which resistance just begins to appear. Its intersection with the ordinate defines I_c for the gate film used. It can be seen that the presence of gate current "helps" the grid current to make the gate resistive.

The gain of the CFC can be defined as the ratio of the maximum current I_c which the gate can carry and remain superconductive, to the minimum grid current I_G required to make the gate resistive at low gate currents, *i.e.*,

$$g = I_c/I_G \qquad (1)$$

For the unshielded cryotron I_c = 91 ma and I_G = 31 ma at the temperature shown. The other broken-line curves shown in Fig. 4 refer to the right-hand ordinate and represent the variation of resistance with grid current at constant gate current. The slope of these curves can be accounted for, to within a factor of 2, in terms of the known critical field of the film and the film resistivity.

If the CFC shown in Fig. 3 is covered with an insulating layer followed by a film of lead, the inductances of the gate and grid will be reduced as explained above, and the d-c characteristics will be somewhat changed as shown in the solid lines of Fig. 4. It can be seen that I_c for the shielded tin film is more than three times that for the unshielded film. I_G is also somewhat larger. However, the gain of the shielded cryotron is larger than that of the unshielded one.

It is noteworthy that the curves of resistance as a function of grid current for the shielded CFC approach a saturation value of resistance. This is because the field due to a grid above a shield plane falls off very rapidly beyond the edge of the grid; because in this region the fields, due to the grid current and the shield current, tend to cancel one another. It is to be expected, therefore, that the portion of the gate film which can be made resistive by grid current action is that portion lying under the grid. The maximum calculated resistance of the shielded CFC is shown dotted and is seen to be in fair agreement with experiment.

We will now show that the gain of the CFC is proportional to the ratio of the gate to the grid widths. As mentioned above, for thin tin films, I_c is proportional to the film width W, *i.e.*,

$$I_c = i_c W . \qquad (2)$$

Here i_c is a constant of the material roughly proportional to $T_c - T$ as long as this is small.

In a current-carrying superconducting film (at least when the film thickness is large compared to the penetration depth of the current), the current density is not uniform over the film surface, but is more concentrated near the edges. For a current-carrying film close to a superconducting shield plane, however, the current is distributed more uniformly.

The field H between a film of width w carrying a current I which is assumed distributed uniformly, and a shield plane is

$$H \approx 0.4\pi I/w . \qquad (3)$$

This formula will apply to the field in the space between a grid and its shield plane, which contains the gate film. If the gate film has a critical field H_c at which it becomes resistive, then the grid current I_G at which the gate just becomes resistive will, using Eq. (3), be

$$I_G \approx H_c w/0.4\pi . \qquad (4)$$

The points shown in Fig. 1 are values of H_c calculated from experimental values of I_G for representative cryotrons of different grid widths. It is clear that Eq. (4) is at least approximately correct. From (1), (2) and (4) we finally obtain the gain as

$$g = 0.4\pi \frac{i_c}{H_c} \frac{W}{w} . \qquad (5)$$

It is seen that the gain is proportional to W/w. i_c/H_c is a property of the material and decreases with gate film thickness. i_c rises linearly with $\Delta T = T_c - T$ close to the critical temperature. However, as Fig. 1 shows, the curve of H_c vs. ΔT has a knee at $\Delta T \leq 0.08°K$. It is to be expected, therefore, that g should rise strongly as the temperature is decreased below T_c until T goes below the knee of the H_c-vs.-ΔT curve. This appears to be the case. The cryotrons described here are operated at $\Delta T = 0.07°K$, *i.e.*, just above the "knee" and are not, therefore, operating at their maximum gain.

The speed of a cryotron is, of course, dependent on the mode of operation. In the circuits described below, the cryotron gate is in parallel with the load, which consists of the grid crossing a similar cryotron. The time constant τ at which current will be diverted from a cryotron gate to the grid of the load cryotron is L/R where R is the resistance of the driving cryotron. L is the sum of the grid inductance of the driven cryotron, the gate inductance of the driving cryotron, and the inductance of the connecting circuits. Out of all these terms, only the grid inductance is important.

The inductance of a grid of width w spaced d cm from the superconducting shield plane can be shown to be $4\pi \times 10^{-9}\, d/w$ henries/cm. The driven cryotron has a width W, hence its grid has a length W and

$$L = 4\pi \frac{Wd}{w} \times 10^{-9} \text{ henries.} \tag{6}$$

As discussed in connection with Fig. 4, the maximum portion of a shielded cryotron which becomes resistive is that part of the gate film covered by the grid. Hence, the resistance R of the driving cryotron of width W, energized by a grid of width w, is

$$R = \rho \frac{w}{Wt} \text{ ohms} \tag{7}$$

where t is the gate film thickness and ρ the effective bulk resistivity. (For very pure films, ρ is itself a function of t, but for the relatively impure films used here, this dependence can be neglected.)

From (6) and (7), we find that the effective time constant τ of one cryotron driving another is

$$\begin{aligned}\tau &= L/R \\ &= 4\pi \frac{td}{\rho}\left(\frac{W}{w}\right)^2 \times 10^{-9} \text{ sec.}\end{aligned} \tag{8}$$

Substituting from (5) for W/w in (8) we find the time constant in terms of the gain:

$$\tau = 4\pi \frac{td}{\rho}\left(\frac{H_c}{0.4\pi i_c}\right)^2 g^2 \times 10^{-9} \text{ sec.} \tag{9}$$

There are two points of interest in (9). First, $\tau \propto t(H_c/0.4\pi i_c)^2$. This shows that there is an optimum value of the gate thickness t, because as we attempt to reduce τ by reducing t, $H_c/0.4\pi i_c$ increases. For solid wires, $H_c/0.4\pi i_c \to 1$, but for the 0.3-micron tin films presently used, $H_c/0.4\pi i_c$ is between 20 and 50. The second point of interest is that τ is not a function of the grid and gate widths. Hence, a reduction in cryotron area will not increase speed.

Present values for the material constants in (9) are

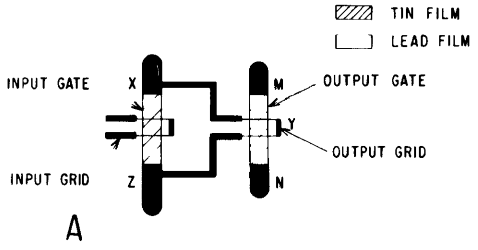
$$t \approx 0.3 \text{ microns}, \qquad \rho \approx 6\text{–}12 \times 10^{-7} \text{ ohm-cm},$$
$$d \approx 1.0 \text{ micron}, \qquad \frac{H_c}{0.4\pi i_c} = 20\text{–}50.$$

A practical value for the gain is 2. Substituting these values into (8), we obtain a theoretical range of $\tau = 5 - 65 \times 10^{-8}$ sec. A typical cryotron, described below, has an experimental time constant of 38×10^{-8} sec. at the temperature of operation.

A Simple Storage Circuit

We will now describe a simple storage circuit which makes use of a principle unique to superconducting networks. The principle will be illustrated with the circuit shown in Fig. 5-A.

In one mode of operation of this circuit, a current is applied between X and Z. Most of this flows through the path XZ rather than XYZ, because the former has much lower inductance. The equivalent circuit is shown in Fig. 5-B.



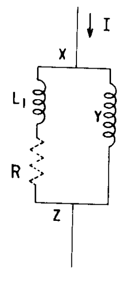


Fig. 5—(a) Cryotron storage cell. (b) Equivalent circuit.

If current is now passed through the input grid, XZ becomes resistive and I is diverted through the path XYZ. It is now possible to switch off the current through the input grid so that XZ becomes superconducting again. Since L_2 is still superconducting, it is to be expected that I will remain diverted through L_2, even though L_1 has become superconducting again. This does, in fact, happen experimentally. The current in XYZ can conveniently be determined with a d-c measurement by measuring the resistance of MN.

If, after I has been diverted to L_2 and after L_1 has become superconducting again, I is switched off, a circulating current will remain in loop XYZ. Its magnitude can be calculated as follows:

Assume that a current $+I$ has been injected into node X and completely diverted to L_2, through L_1's having been temporarily been made resistive. If now current $-I$ is injected into node X, it will divide itself between XZ and XYZ in the inverse ratio of their inductances. The current along XZ will be

$$I_{XZ} = -I \frac{L_2}{L_1 + L_2},$$

and the net current along XYZ will be

$$I_{XYZ} = I - I \frac{L_1}{L_1 + L_2} = I \frac{L_2}{L_1 + L_2}.$$

Therefore, $-I_{XZ} = I_{XYZ}$. Hence, the circulating current

$$I_{circ} = \frac{L_2}{L_1 + L_2} I. \qquad (10)$$

Eq. (10) has been confirmed experimentally, and currents have been stored for several hours.

The results which have been described in connection with the above storage circuit can be generalized. If a current is injected into a network of superconductive elements, it will distribute itself among them in inverse ratio to their inductances. By making one or more of these inductors resistive for various lengths of time, current will be diverted from the resistive elements to the superconducting ones. As soon as the resistive elements are made superconducting again, however, the current distribution stops changing, as long as the external injected current remains constant.

It is clear that in a cryotron computer, analog and digital storage will be simpler than in a transistor computer, where positive feedback circuits, or magnetic cores which cannot conveniently supply a continuous output, are required.

The circuit of Fig. 5 provides a convenient way of measuring the effective speed of the cryotrons used in it. The deflection of current from leg XZ to leg XYZ takes place with a time constant

$$\tau = (L_1 + L_2)/R \qquad (11)$$

(R is the resistance produced in XZ by the input grid current). By applying individual pulses of known length to the input grid, we can measure the current changes in the output grid and obtain an experimental value for τ.

The result of applying 0.05-microsecond current pulses to the input grid of one of the storage loops shown in Fig. 7 is shown in Fig. 6. The experimental curves have a time constant of approximately 0.38 microseconds. The value given by Eq. (11), using calculated values for $L_1 + L_2$ and experimental values for R, is 0.33 microseconds. The curve of rising current is obtained when 150 ma. is injected into the cryotron loop and gradually diverted to the output leg. The curve of decreasing current corresponds to a stored current (with the external current switched off) being destroyed by pulses on the input grid.

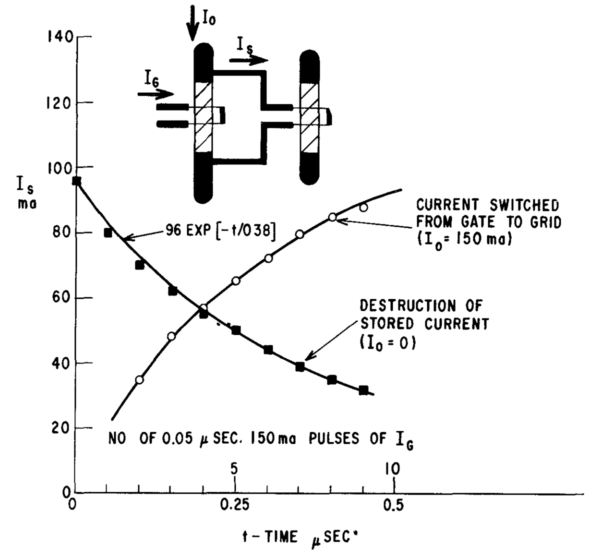


Fig. 6—Change of stored current due to 0.05 μsec pulses on input grid of storage cell of Fig. 7. Curves are fitted to the data points corresponding to a time constant of 0.38 μsec.

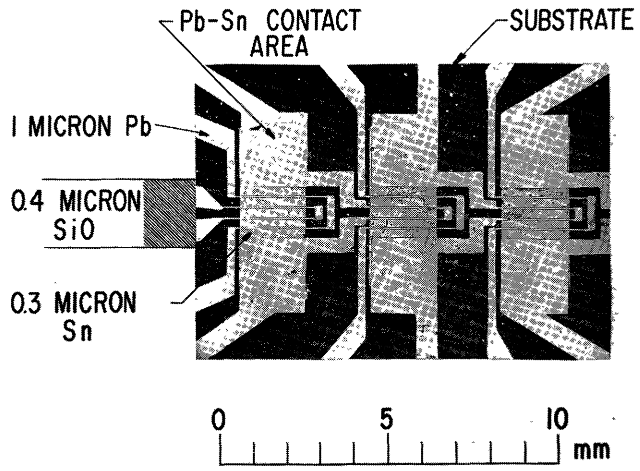


Fig. 7—Portion of experimental shift register. Note the insulator film separating the grids from the underlying gate films.

A Shift Register

The memory circuit described has been applied to a shift register. A short portion of such a register is shown in Fig. 7. The register is of the three-stages-per-bit type and requires two advance and two reset windings. A diagram and a set of calculated waveforms are shown in Fig. 8.

Information travels from left to right. To inject a "1" into the register, the input winding is pulsed while advance current I_1 is on. This diverts I_1 from the first cryotron to its output grid and when I_1 goes off, a circulating current C_1 remains in the first cell. I_2 is now injected into the second storage cell. Due to the existence of C_1, I_2 will be diverted to the output grid of the second storage cell. It is necessary at this time to destroy C_1. This is done by passing current through the reset winding R_2. C_1 has to be destroyed before I_2 is switched off since, otherwise, C_2, the circulating current in the second storage cell

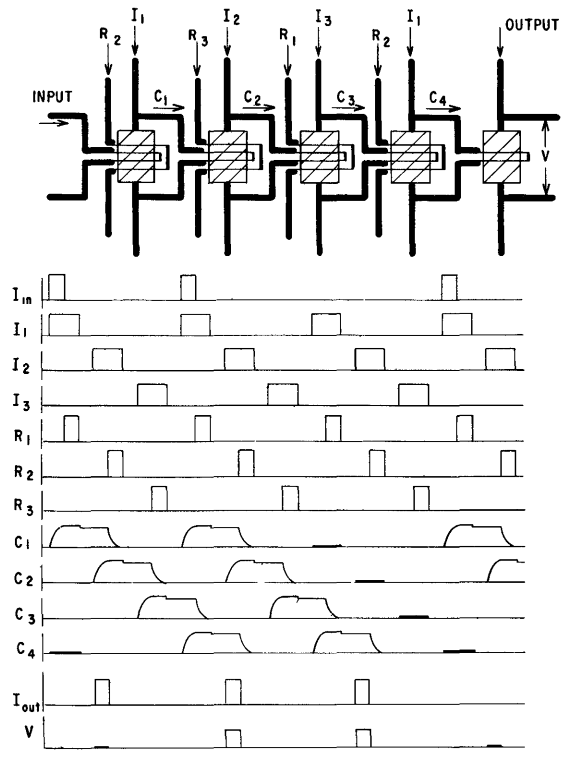


Fig. 8—Calculated waveforms for shift register.

caused by the effect of C_1 on I_2, would be destroyed. After C_1 has been destroyed and I_2 switched off, the injected "1" is represented by the circulating current C_2 in cell 2. In similar fashion, C_3 is created and C_2 is destroyed. Only now can a new "1" be injected into the first storage cell. The grid of the last storage cell crosses an output cryotron whose resistance is an indication of the presence or absence of the circulating current C_4. As described earlier, the experimental and calculated time constant of the cryotron in this circuit module is 0.38 microseconds. This time constant could probably be reduced to approximately 0.1 microsecond by working at a lower temperature and changing the cryotron dimensions to lower the gain. The register has been used to transfer information. However, it has not yet been operated at high repetition rates.

A summary of cryotron characteristics is shown in the table:

Time	0.4 μsec
Size	6 mm^2
Dissipation (d-c)	$= 5 \times 10^{-6}$ watts

The lower limit in size is presently set by difficulties in getting good contact between the CFC gate and its connecting circuitry.

Using our present elements and putting one per square cm (this would leave plenty of area for interconnections), it should be possible to accommodate 1000 circuits on a 1 ft^2 plate. Stacking two of these plates per cm would give a 50,000 element computer in 1 cu ft. Assuming each element to be on for 10 per cent of the time gives the total dissipation as $5 \times 10^4 \times 5 \times 10^{-7} = 0.025$ w. Approximately 0.25w of heat enters the liquid helium due to radiation and conduction through the container. It has been calculated that 100 input and output leads, 10 of which carry 200 ma. dc, would contribute less than 0.2w in conduction and Joule heating. (It was assumed that the upper ends of these leads would be in thermal contact with a liquid nitrogen chamber.) The total heat inflow is therefore just under 0.5w for the system under discussion. Helium refrigerators with 1-w capacity for a 1-cu.-ft. volume are presently being designed.

What role will cryotrons play in future computers? They are, of course, small. They appear to constitute the cheapest method of assembling a large number of circuits in a small place, and they are the only active circuit component which can be deposited in a few steps at the same time as the interconnections.

Because CFC's are, in principle, as cheap as magnetic cores, they make possible radically improved memory structures where each storage element can have logic associated with it. Memories, in fact, appear as an attractive first application because their structure is repetitive and because they have many less logic levels than even the simplest computer.

Computers are presently built from plug-in circuit packages where each plug-in unit represents a few logical elements. CFC's make it possible to deposit the equivalent of a present-day rack on a small plate in a few hours. The problems of testing and fault correction in such a complex multi-level logic module are real, but they are not greater for cryotrons than for any other component which would allow similar packing densities. If these problems can be solved economically, and we feel sure that they can be, the next ten years may see the development of cryotron ground-based computers, as well as air-borne computers and memory sustems.

Acknowledgments

The authors would like to thank D. MacKellar for his valuable contributions to device construction methods, and for his assistance with the experimental work.