Proceedings of the

T-114

EASTERN JOINT COMPUTER CONFERENCE

December 3–5, 1958 Philadelphia, Pa.

THEME: MODERN COMPUTERS:
OBJECTIVES, DESIGNS, APPLICATIONS

Sponsors:

AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
Committee on Computing Devices

THE ASSOCIATION FOR COMPUTING MACHINERY

THE INSTITUTE OF RADIO ENGINEERS
Professional Group on Electronic Computers

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

Printed in the United States of America July 1959 Price $4.00

ADDITIONAL COPIES

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

AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS

33 West 39 Street, New York 18, N. Y.

INSTITUTE OF RADIO ENGINEERS

1 East 79 Street, New York 21, N. Y.

ASSOCIATION FOR COMPUTING MACHINERY

2 East 63 Street, New York 21, N. Y.

Proceedings of the

EASTERN JOINT COMPUTER CONFERENCE

Papers and Discussions Presented
at the Joint Computer Conference
Philadelphia, Pa., December 3–5, 1958

Theme: MODERN COMPUTERS: Objectives, Designs, Applications

Sponsors
AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
Committee on Computing Devices
THE ASSOCIATION OF COMPUTING MACHINERY
THE INSTITUTE OF RADIO ENGINEERS
Professional Group on Electronic Computers

Published by
American Institute of Electrical Engineers
33 West 39th Street, New York 18, N. Y.
for the
National Joint Computer Committee

COMMITTEE FOR THE 1958 EASTERN JOINT COMPUTER CONFERENCE

Conference Chairman

JOHN M. BROOMALL	Burroughs Corporation, Paoli, Pa.

LOCAL ARRANGEMENTS COMMITTEE

Chairman

PETER E. RAFFA	Technitrol Engineering Company, Philadelphia, Pa.

Secretary

ALICE KIRSCH	Technitrol Engineering Company, Philadelphia, Pa.

Business Manager

BRUCE R. McFADDEN	Remington Rand Univac, Philadelphia, Pa.

Registration

WILLIAM E. BRADLEY	Philco Corporation, Philadelphia, Pa.

Publicity

NORMAN N. ALPERIN	Radio Corporation of America, Moorestown, N. J.

Exhibits Liaison

JOSEPH REE	Technitrol Engineering Company, Philadelphia, Pa.

Hotel

LESTER SPANDORFER	Remington Rand Univac, Philadelphia, Pa.

Printing

JAMES A. LEES	Technitrol Engineering Company, Philadelphia, Pa.

TECHNICAL PROGRAM COMMITTEE

Chairman

F. M. VERZUH	Massachusetts Institute of Technology, Cambridge, Mass.
T. H. BONN	Remington Rand Univac, Philadelphia, Pa.
WESLEY A. CLARK	Lincoln Laboratory, Lexington, Mass.
N. P. EDWARDS	International Business Machines Corporation, White Plains, N. Y.
E. L. HARDER	Westinghouse Electric Corporation, East Pittsburgh, Pa.
W. ORCHARD-HAYS	Corporation for Economic and Industrial Research, Arlington, Va.
REX RICE	International Business Machines Corporation, Poughkeepsie, N. Y.
F. H. TENDICK, JR.	Bell Telephone Laboratories, Murray Hill, N. J.

PUBLICATIONS COMMITTEE

Chairman

ARNOLD A. COHEN	Remington Rand Univac, St. Paul, Minn.

Vice-Chairman

ROBERT M. KALB	Remington Rand Univac, St. Paul, Minn.

Secretary

MARY L. MILLIGAN	Remington Rand Univac, St. Paul, Minn.
LeROY T. ANDERSON	University of Minnesota, Minneapolis, Minn.
GEORGE A. HARDENBERGH	Remington Rand Univac, St. Paul, Minn.
KENNETH H. MULHOLLAND	Remington Rand Univac, St. Paul, Minn.

NATIONAL JOINT COMPUTER COMMITTEE

1958

M. M. ASTRAHAN, Chairman	International Business Machines Corporation
H. H. GOODE, Vice-Chairman	Bendix Systems Division
MARGARET R. FOX, Secretary-Treasurer	National Bureau of Standards

AIEE Representatives

J. G. BRAINERD	University of Pennsylvania
FRED KALBACH	Burroughs Corporation
R. A. IMM	International Business Machines Corporation
G. S. GLINSKI	University of Ottawa
MORRIS RUBINOFF, Ex-Officio	Philco Corporation
R. S. GARDNER, Headquarters	American Institute of Electrical Engineers

IRE Representatives

R. D. ELBOURN	National Bureau of Standards
H. H. GOODE	Bendix Systems Division
L. C. NOFREY	General Electric Company
N. H. TAYLOR	ITEK Corporation
WERNER BUCHHOLZ, Ex-Officio	International Business Machines Corporation
L. G. CUMMING, Headquarters	Institute of Radio Engineers

ACM Representatives

SIDNEY FERNBACH	University of California Radiation Laboratory
F. M. VERZUH	Massachusetts Institute of Technology
PAUL ARMER	The Rand Corporation
A. J. PERLIS	Carnegie Institute of Technology
J. W. CARR III, Ex-Officio	University of North Carolina
J. MOSHMAN, Headquarters	Corporation for Economic and Industrial Research

Contents

New Frontiers

J. W. FORRESTER

NEW FRONTIERS imply departures from the past. In the next two decades the data processing industry can be expected to face challenges very different from those of the last 15 years. These new frontiers will require a new way of thinking, different kinds of leaders than are now being developed, and administrative organizational forms lying in the opposite direction from the present path of corporate evolution.

For some who have recently entered the field, it may be hard to realize that only 10 years ago there were no general-purpose electronic digital computers. Fifteen years ago there were no designs for such machines. It is a young industry populated by young people. Their thinking has been conditioned by the kind of challenges and goals that exist in the infancy of a new endeavor. Maturity of the industry will bring a new environment.

If the future is to be so different from the past, perhaps it would be well to start by reviewing, briefly, the past 15 years of the computer industry to see how it has arrived at its present position.

Past Frontiers

During the period from 1945 to 1950 the challenge lay in design and construction. Only 7 or 8 years ago was it demonstrated that an internally programmed digital computer was practical. Only that recently was electronic reliability first proved high enough that a computer with access to its own instructions could avoid chaos from erroneous manipulation of its own control orders.

About 1950, a new frontier was attacked to explore methods of programing digital computers and the application of computers to science and engineering. During the present 5-year span following 1955, the frontier has been in computer applications in commerce for the processing of information previously handled by clerical organizations.

Frontiers imply pioneers, heroes, and areas where men seek to establish reputations for new discoveries. Over the last 15 years, what have been these glamour areas? They have, for the most part, fallen within three categories:

1. Physical research and the development of components.
2. The construction of operable electronic machines rather independent of their end use.
3. The application of existing electronic machines to tasks already being done by other methods.

These represent a compartmentalized and, by now, a well-worn set of frontiers. They still, however, describe at least 90% of the papers in the program of this conference. Continuing in the present pattern will inherently preclude success in the large data processing system opportunities of the future.

Our frontiers and our pioneers have been associated with pieces and bits of data processing. The result has been so rich that major successes were possible without well-balanced projects. A new component or modest attention to reliability carried a computer to new heights of success. Applications were so numerous that any computer could be used to advantage. It has been a field in which limited goals still brought major advances. Technological research progress has been rapid enough to insure success without pioneers in design, or manufacturing methods, or sales. Most important, the field has succeeded without needing many leaders who could simultaneously perceive a new frontier a decade away and combine this foresight with the ability to blend research, design, education, and new manufacturing methods into successful commercial enterprises.

The economic foundation of the industry has received little attention. The computer industry is dedicated to processing information. But information has no value by itself. Information is valuable only as it produces more effective industrial operation. There is a need here for the "systems approach" that far transcends the systems discussed at professional meetings. The phrases, "electronic systems," "digital systems," and "system design" appear repeatedly in the program for this meeting. These phrases refer almost exclusively to the system of components making up a machine. They ignore the economic, organizational, and management system complexes which the machines are presumably intended to serve.

Product Life Cycle

Now the general life cycle of any product and the different phases of market development will be discussed. Fig. 1, illustrates some typical curves. Time is the horizontal axis. Two curves are shown, one for annual sales, and the other for percentage profit margin. The general shapes and their relationship one to the other are typical of almost any product. It might be automobiles, hula hoops, or airplanes. Depending on the product, the horizontal time scale may represent 90 days or 90 years.

The important points are the phases in market development and the general shapes of the curves. The early stages show the "Product Introduction." During this period, sales are low, profits may be negative, and initial market acceptance is being tested. In the second phase, labeled "Market Development," is the rapid growth in demand. During market development, demand often outruns production capacity and profit margins rise to a peak ahead of the peak in sales volume. The industry then enters a "Market Maturity" phase of steady sales. In the fourth stage is "Sales Decline" as the product becomes obsolete and is superceded by some new innovation.

Such curves can be considered as applying to a specific catalogue item like some particular computer design; or to a class of items as, for example, vacuum-tube computers; or to a whole product line and marketing philosophy as, for example, the digital computer, looked upon as a device isolated from responsibility for its economic mission and its contribution to the larger system of which it is a part.

During product introduction and much market development, one may find an exponential growth in sales. Here, the first part of the sales curve is exponential, and the exponential curve is continued by the dashed line. Note that exponential growth (for example, sales doubling every 3 or 4 years) can persist over many years while a product is finding its place in the economy. There comes a time, however, when continuation of exponential growth would suddenly carry the industry to an impossibly high fraction of the total gross national product. Therefore, in a very brief period the actual growth of the industry can break away sharply from its earlier exponential pattern.

Where might the computer industry be today in such a life cycle? In Fig. 2, it is suggested that the computer industry is somewhere in the shaded

J. W. FORRESTER is with Massachusetts Institute of Technology, Cambridge, Mass.

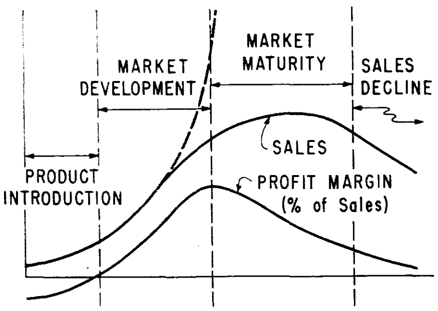


Fig. 1. Typical product-life cycle

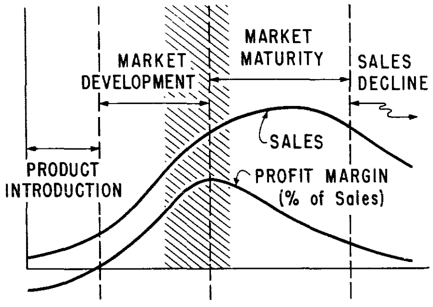


Fig. 2. Computer industry cycle

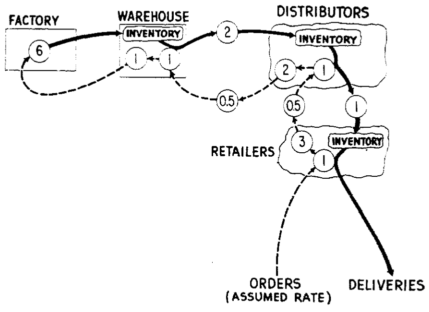


Fig. 3. Production-distribution system

portion of the diagram. For the computer industry as a whole, the exponential growth period is largely in the past. This is not to be confused with the growth patterns of individual computers, or companies, or systems. If they rise to the challenges of the new frontiers, individual enterprises can exploit a series of these cycles occurring one after the other so that a succession of market development and high profitability periods can succeed one another.

Why do I feel that the exponential growth period is ending? Perhaps two reasons will indicate the answer. Already the total sales of the data-processing industry are sufficiently high that they cannot double more than once or twice more before they will have reached their probable share of the country's industrial output. Second, the handwriting on the wall can be seen in the way in which orders for new computers dropped off during the recent recession. The computer industry is essentially a capital equipment industry like the manufacture of machine tools. It will be a highly cyclical peak-and-valley type of industry as soon as it reaches market maturity.

Recall the frontiers in data processing in the light of this diagram. Prior to, and during product introduction, the frontier is research and development. The challenge is to make a product which will function. During market development the technological advances continue at full speed. At the same time, emphasis develops on the easy and obvious applications. These are exploited and sales increase rapidly.

A whole new picture evolves in the market maturity phase. The cream has been skimmed off the research and technological opportunities. The easy and obvious applications have been exploited. The high profit margins accruing to the first entries into the field have attracted many competent competitors. Managements have projected the exponential growth curves further than justified, leading to excess manufacturing departments and to research and development organizations geared to the objectives of the exponential growth phase, not to the market maturity phase.

The Future

Two types of frontiers lie ahead. One is in the market maturity phase of the machines and applications known today. Here the challenge is that of commercial success in an intensely competitive, cyclical industry. The second frontier lies in the product introduction and the market development phases of new sytems applications of digital devices. Here the successful digital equipment company of the future must take more responsibility for the usefulness of the product and the manner in which the customer employs it. Integrated data-processing systems will become more complex. Fewer and fewer customers will feel justified in individually pioneering the applications. The emphasis will eventually lie not in equipment alone but in a complete service where the equipment is incidental to the over-all effectiveness of the management system it supports.

The new frontiers will be more demanding than those of the past. The emphasis will shift away from research laboratory results and new devices. The future will belong to those who are not merely good at research, or development, or applications, but to those fewer men who can take the leadership simultaneously in research, development, manufacturing, sales, and applications to build from these separate pieces a co-ordinated and competitive digital data service. The competitive advantage will lie with the company that provides the greatest utility to the customer by providing not only equipment, but leadership in how to use it.

Look at a few specific data-processing frontiers. One challenge is to reduce the cost of computers. Over the last 10 years the rate of technological advance at the research level has been tremendous with increases in speed, storage capacity, and component reliability. This rapid rate of research advancement has hung as a cloud over the exploitation of the results. Obsolescense has been so rapid that none of the developments were fully utilized before a new technology was opened. As a result, until now the entire industry has been exploratory and experimental in nature. The so-called "production computers" are really prototypes. Their numbers, even when they reach a few hundred, are still the numbers associated with pilot production runs. Costs are high. Reliability is low, compared to what is achievable. Design and marketing programs are unbalanced with emphasis on hardware, inadequate attention to computer programing, and almost nothing on the theory and practice of the end application. There is a physical science underlying design but little or no science underlying usage. These are the characteristics of the product introduction and market development stages of the life cycle.

Now in the midst of a maturing market, cost will become increasingly important. In the past, improvement has come from increasing speed and capacity faster than increases in cost. There has been almost no reduction in cost for the same performance. As an example of a future challenge in the area of cost, consider digital computers for engineering computation. Why not look forward to the performance now possible for one-fifth the present cost? Accomplishing this would permit and require larger numbers of machines than presently limited production runs. Instead of five servicemen permanently assigned to a large computer, it would be economically mandatory to give continuous trouble-free operation with perhaps a one-hour service call per week. Component and manufacturing methods would be ruthlessly standardized in the interests of reliability and low cost, rather than allowing machines to contain a different circuit as the creative mark and contribution of each participating engineer.

If a reduction factor of five in cost does not capture the imagination, what about a factor of 10? Even that may be possible. Often times, the big strides are easier to achieve than small improvements. The big step requires a new approach; one does not waste time trying to wring the last few per cent of improvement out of an old practice. Electronic computers themselves are an example, 100,000-machine operations per second did not come from improving punched-card mechanisms and the speeds of electromechanical relays.

A second frontier area lies in production process-control. This will come first in the chemical and the petrochemical industries and wherever continuous processing of materials is the accepted manufacturing method.

The goals reach far beyond cost reduction. Cost reduction reaches peak importance in the market maturity phase of a product life cycle. If the earlier diagrams are recalled, the market maturity and the sales decline periods are the ones where price competition enters most strongly. However, by the time these periods have been reached in many products, the managements of the companies have become conservative, profit margins are already falling, and management is under pressure for immediate improvement and is not likely to become experimenters in new manufacturing control processes which may be years away from profitable operation. The glamour, the innovation, and the high profit margins in the chemical industries are associated with new products and new markets.

How can digital computation contribute? Getting a new chemical product on the market ahead of competition is important. There are general-purpose computers, why not the general-purpose chemical plant which, through centralized control and building-block chemical processes, can turn out early production runs of new chemicals for market testing?

Or, why not look to a reduction factor of five in the capital cost of chemical plants? There is reason to believe that dramatically more efficient chemical processes are possible with co-ordinated multivariable, nonlinear control than the processes in use today. Little or nothing is known about such processes which are today uncontrollable. One cannot even experiment with them.

Advanced process control systems will require years to evolve. In the meantime, how is the evolution to be accomplished? Today, the approach is to propose a special equipment design. It takes 3 years to bring the equipment to operating condition, and by then the original idea is obsolete. A whole new concept is necessary; a concept which recognizes the critical importance of the learning phase. The knowledge, not the equipment or its efficiency, is the first objective. The industry should experiment with standard computers tied together with special transitional and terminal equipment.

For this special equipment, one needs the ability to go from an information flow diagram to reliable operating hardware in 30 days. The cost should be no greater than with production equipment today. I am sure this is possible. It means automatic design and it means automatic production. Several companies are planning the use of computers for the automatic design of parts of chemical plants, aircraft, and for the design of electrical devices and components. Why should not the computer industry be taking the lead in applying its own products? With special equipment available on 30-days' notice, the experiments on new applications would proceed unimpeded by long waits. Efficiently designed final devices would follow after the applications had been defined and market tested.

A third frontier is in fully integrated data-processing systems. The industry has been using the words "integrated data processing" but today's applications are embryonic. The industry has only started to consider the implications of co-ordinated handling of orders, inventory, production schedules, machine tool loading, parts list explosion, routing of assemblies through the production process, filling of orders, billing, point-of-sale data entry, and the analysis of sales and production data for management control purposes. It seems that too many people are trying to enter integrated data processing by designing specific special machines. They fail to observe that the hurdle here is also one of education and learning. The computer manufacturer does not know how the equipment will be used or how terminal equipment should be designed. The potential customer does not understand his application and does not know what equipment to ask for. It is a market in the product introduction stage where the product and its potentialities have not yet been defined.

The integrated data system should be aimed, not merely at reduced clerical costs, but toward enhanced total management effectiveness. Among machine designers there is inadequate knowledge of the relationship of information quality to corporate success. Automatic data-processing often obtains, more rapidly and accurately, information which need never exist. At the same time, tremendously useful information often exists and is readily available in a company's data system but lies unused.

In the past, the computer industry sold machines. Now, it sells machines with computer programs. In the future, it must sell machines with programs and the management control systems of which they are a part. The successful pioneer on this frontier must understand research, engineering, production, sales, and, at the same time, have a deep insight into management applications of better information sources.

As my own frontier for the next 5 or 10 years, I have chosen to explore this relationship between information and corporate success. It will be discussed briefly.

At the Massachusetts Institute of Technology School of Industrial Management, the new program is called Industrial Dynamics. It is a quantitative study of the forces which cause industrial growth and fluctuation. It is aimed at understanding the interactions between the flows of material, money, manpower, capital equipment, and information. It is a study of how the information and decision-making network gives the corporation its characteristic growth and behavior tendencies. It allows a quantitative approach to the value and the purpose of information, how information should be used in the decision-making process, and the importance of organizational form in controlling company destiny. It is obvious that there is a close relationship between all this and the field of digital computers. Nevertheless, the digital computer is to industrial dynamics what the slide rule is to engineering. It is a necessary tool, but only a tool.

Industrial dynamics is now timely because the necessary foundation exists from the results of military research over the last 20 years. We expect to build on the theory of information-feedback systems, on the use of simulation to explore complex nonlinear situations, and on the understanding gained in the last decade of how tactical military decisions can be automatized.

Perhaps this will be more tangible with a few illustrations. Fig. 3 shows a simple organizational form of a factory, a factory warehouse, distributors, re-

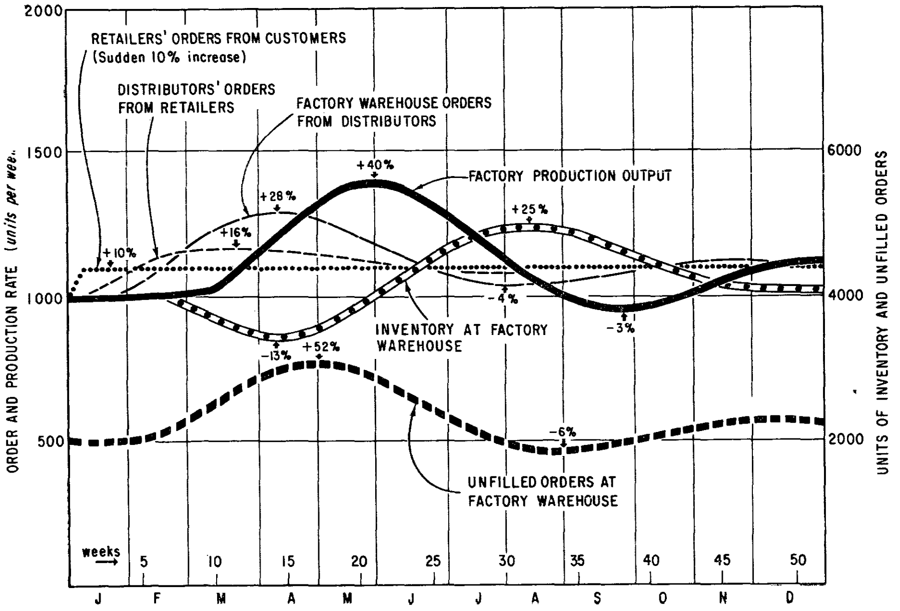


Fig. 4. Response of system to 10-per cent increase in sales

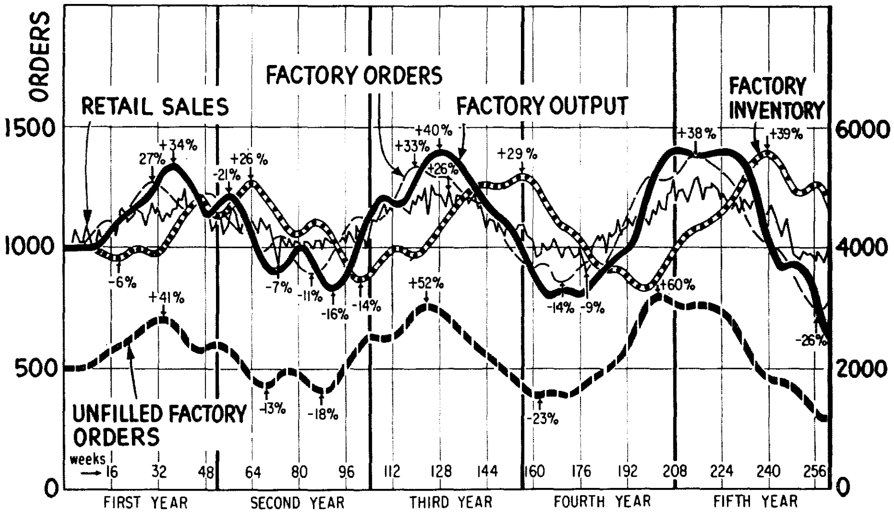


Fig. 5. Composite system

tailers, and customers. Some plausible assumptions about this system regarding time delays in the flow of information and materials are made. Included are some ordinary policies about the ordering of materials and the maintenance of inventories. This results in a nonlinear system of about 40, first-order difference equations. These describe the system mathematically. One can then begin to study the dynamic behavior of the production-distribution system. The reaction of the system to outside disturbances can be determined first.

Fig. 4 shows what happens to such a system if a sudden 10% increase in retail sales were to occur. Because of the necessity for the filling of supply pipelines and the normal practices of inventory accumulation, factory production may, after a substantial delay, rise by four times the retail increase. The rise, however, is temporary and as inventory accumulation and the pipeline filling runs its course, production may dip even below its intial value before stabilizing approximately a year later at a level corresponding to retail sales.

By changing the formulation of the system, one can see the effect of inaccurate data and delays in the transmission of information. After costs and money flow are added, improved information can be equated directly to the increase it can produce in profits.

The results get even more interesting as greater reality is added. One could represent some common corporate behavior policies regarding advertising and a segment of the consumer market dealing with the deferrability of consumer purchases. This might require 17 more equations added to the previous 40. The results emerge as in Fig. 5. Here a constant underlying demand is modulated by advertising and is superimposed on a random noise component to give retail sales. The system has tendencies toward the self generation of a 2-year production cycle. Such behavior is observed in several industries. Superimposed on the 2-year pattern is a higher frequency pattern created by the interaction of the random input and the inventory and pipeline effects seen in the previous illustration. A full discussion of the subject of industrial dynamics with more examples of results is available.[1]

An analysis of this kind can deal with information sources and the lateness and distortion in information; with corporate policy and organizational forms; with supply conditions in material, labor, and capital equipment; and with the behavior of the consumer market. The research is in its infancy but is already showing great promise. One graduate thesis last year by Abraham Katz dealt with the quantitative, underlying foundation of research and development management. This was done in the context of a military special products division of a company. It showed the possibility of interrelating such factors as the technical nature of the project, its degree of innovation, the managerial competence of the company, the resources of the company, the competence of the military contracting office, the government budget cycle, and the availability and training periods of new technical staff. Graduate thesis topics underway for this year include the quantitative analysis of automobile designing, inter-relating the long lead time between design and production, the market acceptance of the product and the evolution of styling trends. Another concerns the forces affecting the rate of growth of a new company in a new technological frontier.

The future of industrial dynamics and the future of computers are closely coupled. From the computer industry will come machines and a knowledge of what better information will cost. From industrial dynamics will come an estimate of what this information is worth. Only then can a balance be reached and a sound decision on when, where, and how to utilize the spectacular technical developments of the last decade.

The computer is a tool, but one which needs a theory and framework for its application. It is hoped that industrial dynamics will provide that framework.

We are attempting to develop a set of underlying principles which to the field of management will bear somewhat the relationship that physics does to engineering. If so, industrial dynamics can have a far greater impact on American industry during the next 20 years than has resulted so far from the appearance of digital computers alone.

The Pioneers

Three of the many new challenges in data processing—dramatically reduced cost, process control, and integrated data systems were mentioned earlier. The latter two present the same problems.

In fact, it would be possible to attack both of these areas by the same bold strokes of first taking the leadership in really understanding the application and second being able to provide one-of-a-kind equipment at low cost.

What are the prospects for today's computer companies in these new frontier areas The future for most organizations is not bright. If this surprises some people, look at older industries. The electric razor did not come from the safety razor companies. The airplane was not designed by automobile companies. To a considerable extent rockets, ballistic missiles, and space vehicles have come from outside of the aircraft industry. Atomic bombs, proximity fuses, and radar did not come from within the military research laboratories. In fact, electronic computers did not get their start 15 years ago from the calculating machine companies or the punched card machine companies. The handwriting is on the wall. New companies, never before associated with data processing, are already beginning to make their names in the new frontier areas. Unless the older existing organizations in the field can meet the challenge, they will find themselves in the market maturity and the sales decline areas of the first illustration. They will be left as a part of the cyclically fluctuating capital equipment industry. Newcomers will take over the new frontiers.

Why should this be true? Why should technological progress leave behind those organizations with a history of great success? The answer seems inherent in the typical evolutionary pattern of industrial organizational growth. This evolution starts with a pioneer. A leader, with a vision of the future he wishes to build, enters a new field. He is the man at the new frontier. He builds a successful commercial enterprise. This man grows with his business. In its early days he personally controls all facets of the enterprise from research, through development, design, manufacturing, sales, field service, and ultimate application. He guides products through this cycle several times, thus solidifying his competence and his self-assurance. He develops the habit of being successful.

However, as the enterprise grows, functional decentralization sets in. Manufacturing is broken off as a separate operation. Sales, advertising, research, development, design, all begin to fit into separate boxes. Then, staff functions emerge which further dilute responsibilities and authority. These things are done in the name of efficiency, a short-range efficiency lasting at most a few decades.

The organization takes a form that repels and suppresses the kind of spirit that built the company. The pioneer builds a company, but the company does not recreate its share of pioneers. More men are needed, each with expert ability in every aspect of the field from research to end application. Many men are competent to acquire this breadth of skills by age 30 or 35. By contrast, customary forms of corporate organization tend to prevent this full development even within a human lifetime.

Leadership and the vision of the future is replaced by the organization chart and the hope that a proper "textbook" organization will in itself create new ideas. An organization chart has never been a pioneer. Only on rare occasions can an organization play the role of a hero. The functionally decentralized organization operates against innovation and clear visions of the future.

Now the question of what creates leadership appears. What produces emotional attachment to achieving a goal and devotion to a cause beyond the ordinary call of duty? In a society like ours it is not money. It is the feeling of accomplishment. It is the urge to do something new, to do a job well, and to be recognized for it. A talk given by Professor Douglas McGregor is available concerning this point.[2] In an advanced economy such as ours, personal physical needs are satisfied well beyond minimum levels. In the upper quarter of our scientific, engineering, and managerial population, additional money and security are goals only as a substitute for unachievable but more important goals of self-assurance and the feeling of important accomplishment. In an environment of maximum achievement, of leadership, and of challenge, top men can be held at salaries half what they soon become worth elsewhere.

In the functionally subdivided organization what rewards are available to the individual? At the research laboratory there is the "Nobel prize effect." Men work for the recognition and plaudits of their colleagues, whether shown by the Nobel prize at the top or, further down, by having an article in a professional journal or by being heard on the program of a technical society meeting. Work toward this form of personal recognition will only indirectly and by happenstance result in computers at one-fifth the present price, or integrated data-processing systems; or pierce the frontier of chemical process dynamics and control.

In the sales department, personal reward is financial. The salesman is helpless who perceives a new frontier; he has neither the competence nor the managerial position to bring a major new product to an unknown market. Likewise, no one else in the functional subdivisions of development, engineering, manufacturing, or field service has sufficient breadth of responsibility and authority to attack the new frontier.

At the higher levels of management where the authority and responsibility are adequate, self-confidence to deal boldly with frontiers may be lacking. Furthermore, the day-by-day pressures of running the company too often tend to concentrate management attention on short-range problems or on the future plans for present products. Either one leads the company rather naturally into the market maturity phase of its products or does not provide effective entry into new frontier areas.

The primary challenge in the computer industry is therefore the same as the challenge to any new industry. It is the challenge to the existing organizations to develop the pioneers who can attack the new frontiers.

I am quite sure that this can be done. It will require, however, a reversal of many present-day trends. It will require more emphasis on individual leadership and responsibility. It means re-establishing, within the present economy and the present-day corporation, the spirit of individual capitalistic entrepreneurship, which characterized our society during the rapid growth period of our major industrial enterprises. It means placing even the new engineer or management college graduate in a position of broad responsibility relative to small projects so he can develop competence in concept planning, research, design, budgeting, effective use of resources, and accepting responsibility for his own decisions and the final outcome of his endeavor. It means a move away from the illusion of efficiency through specialization by functions. Instead, specialists in successful pioneering are demanded. It means a determined effort to grow "empire builders" because our new frontiers are indeed the size of empires.

References

1. Industrial Dynamics—A Major Breakthrough for Decision Makers, **J. W. Forrester.** *Harvard Business Review*, Cambridge, Mass., vol. 36, no. 4, Jul.–Aug. 1958, pp. 37–66.

2. The Human Side of Enterprise, **D. M. McGregor.** *Adventure in Thought and Action—The Proceedings of the Fifth Anniversary Convocation of the School of Industrial Management*, Massachusetts Institute of Technology, Cambridge, Mass., Apr. 1957.

Discussion

R. A. Spon (International Business Machines Corporation): Are there any examples of companies who have overcome the market decline problem?

Professor Forrester: Yes. Companies that are alert to the danger make it a policy to live in a succession of profitable peaks on a series of product life cycles.

This can be done as a considered, designed policy if one understands the forces at work. Such companies are in the minority, but enough have persisted in looking ahead to new products and discarding the old ones to indicate that the system can be successful.

B. C. Heyel (ADP Services, Inc.): The big trouble is that today's enterprise demands such huge investments that the old-type pioneers cannot get started. What can be done about this?

Professor Forrester: There is ample capital looking for good men with good ideas. I am not speaking of the old-type pioneer who would follow exactly the pattern of the past; that is no longer pioneering.

There are many examples of new enterprises which grow from the ability of one or two men to manage well and to attract and use money available for new investment. For example, two men that I know have started a new company in the digital equipment business; have shown a net profit for their first full year of operation; are exceeding a half million dollars a year of commercial, nongovernment business before the end of their second year; and they are effectively selling to the largest companies in the field and also successfully competing against them.

It only takes sound judgment, skill, imagination, integrity, a clear interpretation of the signs and trends of the future, and confidence. Investment money seeks out such people.

Data Processing in Banking and Other Services

B. W. TAUNTON

IN GENERAL, the managers of the more progressive industrial firms early recognized the advantages that might be gained through the use of digital and analog computers to supervise and control manufacturing and production operations. No doubt this is due, in part at least, to those pressures which require the enterprising firm to seek out and try every means of improving its products and the efficiency of its production, in order to maintain a competitive advantage. Perhaps it is also due in part to the familiarity of such concerns with machines.

But the use of computers for the processing of data, so far, has been relatively limited. Generally, management has been reluctant to depart from the more orthodox accounting and reporting procedures with which they have been familiar, in favor of a new system, which, at first blush, appears to be costly and perhaps, more important, involves so many unknowns. In the author's opinion, it is the latter uncertainties of the successful installation and operation of such equipment even more than the expense that is involved that has discouraged these managements from taking such a step. The manufacturers of the equipment, in many instances, may be partially responsible for this situation, since they have contributed to the uncertainties, through lack of knowledge of how to use effectively the equipment which they are making and, in many instances, they have been responsible for undue delays in the perfection of design, production, and delivery of equipment which they are endeavoring to market in advance of actual construction.

During the 10-year period between 1947 and 1957, professional and technical workers have increased at the rate of 60.6% and clerical workers at the rate of 22.8%. During the same period factory workers have increased only 4.4%, and the laboring forces only 4.1%, the latter in spite of the fact that over-all production of consumer and other goods has grown by leaps and bounds since the second World War. Unless one learns from industry the advatages to be gained by the use of more advanced and automatic techniques to increase effectively the production of clerical workers, the point where the economic burden of doing clerical work is impossible to bear, even if a sufficient number of workers is found to do it at all, will be reached soon.

It would seem that the use of electronic data-processing equipment will go a long way toward solving accounting and other data processing problems, and it is believed that this will be equally true in the case of the smaller firm, as well as the larger one, since such equipment may be obtained in large or small capacities and consequently at relative cost to meet the needs of concerns of various sizes. Furthermore, the author believes that, with proper study and planning, a computer system can be selected, installed, and put upon an operating basis on schedule, if the schedule is a realistic one. As a practical illustration, a discussion follows of the experiences leading to the selection and installation of a large-scale data processing system at the First National Bank of Boston.

As early as 1950, the Methods Department began to take cognizance of the existence and development of computer systems which might be used in data processing operations. In 1954 it became quite evident that the developments in this field were such as to warrant the assignment of at least one member of this department to its study on a full time basis. Thereafter research activities became much more intensified.

It was decided first to attempt to find some practical means of determining the answer to two basic questions. First, could electronic data-processing equipment be used in any one or more of the bank's applications effectively? Second, if the answer to the first question was "yes," what kind of equipment should be used?

The first of these two problems, that of whether or not to use electronics at all is considered. Of the more than 90 different types of services which are offered to cus-

B. W. Taunton is with the First National Bank of Boston, Boston, Mass.

tomers, there were several requiring the manipulation of large files of data, such as check handling for approximately 108,000 customers, personal trust accounting, installment loan accounting for approximately 65,000 borrowers, commercial loans, mortage loans, and factoring accounts involving records for some 78,000 debtors accounts, corporate trust accounting, involving 765,000 accounts, payroll, some 83,000 savings accounts, personal money orders, and others. Some had already been mechanized to a considerable extent through extensive use of such modern techniques as those involving punched-card equipment. Others had changed little over the years and encompassed a tremendous volume of paper work. Of course, one specific application could have been selected, perhaps one of those least mechanized, and the introduction of the use of an electronic system in that area could have been attempted. This approach, however, seemed to be too limited in nature. It is frequently heard that the stored program computer is referred to as general-purpose equipment. It was reasoned, therefore, that if this be true it might be anticipated that the equipment could be used in various applications, if it lived up to its label. So, a different approach was tried.

The larger applications were examined with a view to selecting two of them that might represent the extremes in requirements for data processing, that is, the upper and lower limits of data processing requirements that any particular system could be called upon to handle. As a result of these studies, two applications were selected, the deposit accounting function and the corporate trust operation. Here were two that were diametrically opposed to each other. To round off figures, more than 220,000 items were processed on the average day against 100,000 checking accounts. Forty-eight % of these accounts were active in a given day. A fixed record length might readily be assigned to each master record which would accommodate the largest as well as the smallest account. A substantial number of mathematical operations were required to process the accounts. Little alphabetical information was involved, it being presently limited to the name and address, which must be referred to ordinarily only once a month in order to prepare statements for customers. More than 900 inquiries a day had to be processed against the file, most of which required an immediate answer. While at first blush, it would seem that the files should best be kept in alphabetical order, it developed that there was no requirement to do so. The file could be operated efficiently with a numbering system.

On the other hand, the corporate trust function involved a tremendous file with relatively few items to be posted, actually 68/100ths of 1% would be affected daily. The file must be kept in alphabetical sequence. It was largely alphabetical in nature, and very little in the way of mathematical work was required. The record length necessary was variable, running from as few as 167 characters of information in one account to more than 47,000 characters of information in another. Inquiries that must be answered by reference to the file were relatively few and far between, and in most instances could be scheduled.

All other operations seemed to fall between these two, insofar as data processing requirements were concerned. Notably, they were the two largest applications and, at the same time, the two that required the greatest amount of clerical effort. In contrast, some other larger applications have been operated since as early as 1931 on punched-card tabulating equipment.

With a complete set of statistics available relative to current operations in these two areas, attention was turned to the electronic data-processing equipment available and an attempt was made to evaluate whether or not any of it might meet the requirements. To program both applications in their entirety would be an exhaustive job. The bank, therefore, tried to analyze the problem and to reach some common basis for comparing different types of equipment. It was concluded that file maintenance in accounting procedures was the most difficult and the most repetitive and time-consuming operation. File maintenance means the preservation of a record of each account and updating it with the daily transactions, in order that it might be available at all times for the preparation of the multitude of reports that are required to operate a business. It was, therefore, determined to limit initially the studies to this phase of the data processing problems.

Next, it was determined that, within each of the three areas in which computers are classified, that is, small, medium, and large, the internal speeds of the computers then available were not significantly different. Far more important were the speeds with which these devices could process files, that is, their ability to read and write large volumes of data in order to locate the account, bring in the transaction, and write out an updated account. Hence, the relative merits of various computer systems in terms of the time which it would require to read master files and transactions, and to write the new master files for the two applications selected for that purpose were measured.

In each instance, the records were established in such a fashion as to make the most efficient use of each of the systems tested, and to take into consideration the limitations and advantages peculiar to each such system. The medium-sized computer field was chosen and one of the better known computers in this area was selected because it was felt that if one medium-sized computer was found that might reasonably perform the work, the other competitive models would be examined in turn.

The results of these studies indicated processing of the deposit accounting application and 1/10th of the corporate trust file in 30 hours of machine time. Thus, it became obvious that two such systems would be needed, operating on at least two shifts in each 24-hour period, with at least four operating staffs to operate them in order to handle only two applications. The cost of these systems, without consideration of the personnel involved, more than exceeded the cost of a large-scale system. No time would be available to add other applications without adding additional systems. The obvious conclusion was that the medium-sized computer system could not fulfill the particular data processing requirements.

Next, attention was turned to a large-scale computer. A typical system in this area was capable of performing the work as it was being presently measured, in terms of reading and writing capacity, within a single shift. As a matter of fact, the bank found that the computer might handle the deposit accounting operation in less than 3 hours each day and 1/10th of the corporate trust file in approximately 1.8 hours each day. If any computer system could handle the specific data processing problems, it must be in this category. With this in mind, each of the large-scale computer systems then available was studied and one of them (the one which was selected) was found capable of handling the deposit accounting operation in something less than 1/2 hour, and 100%, as distinguished from 10% of the corporate trust file, in a little over one hour. It should be emphasized that these, of course, are not total processing times, but only a measure used to make comparisons, since it was the area in which the major differences in equipment existed.

The results of these studies were then, for the first time, made available to the equipment manufacturers who were requested to review them and to modify them, if they felt the calculations were substantially in error.

In June of 1956, a report was forwarded to senior management outlining the studies to date and reporting conclusions to the effect that, 1. electronic data processing equipment was available, which could do the job, and, 2. it appeared that the system, which was eventually selected, would be capable of handling a substantial number of applications on a one-shift basis, and permitted the addition of other applications, as time went on. It was pointed out that since the time required to do the processing on this system was substantially less than that of comparable models, at comparable prices, the machine selected, as first choice, would be the more economical of those available.

Until this time, little thought had been given to the economic feasibility of acquiring a computer system of any type. The department had merely determined the answer to two questions: Could such equipment be used effectively and, if so, what specific system might best do it?

Here the cost department took over and began to draw some comparisons between the cost of operating the recommended system and existing costs. This subject will be treated only briefly. Basically, the position was taken that installation and conversion costs should be amortized over a period of years. It appeared that during the first 2 to 3 years of operation, increased costs should be anticipated but that, thereafter, they might be reduced, particularly when additional equipment of a peripheral nature could be acquired which would facilitate the transfer of data encoded on paper checks to machine language to be processed within the system. No attempt was made to reduce to figures such intangibles as continuing increases in salaries with a consequent increase in conversion costs postponed to a later date, or the hidden costs to be found in the less efficient, but more orthodox, procedures which are followed today. Nevertheless, it is believed such increases will serve as a substantial cushion to absorb ultimately any miscalculations that may have been made.

When the department submitted what proved to be the last feasibility report to management, together with a recommendation that such equipment be acquired, a suggestion was included that the studies be reviewed by independent consultants. Subsequently, management retained a well-known management consultant firm to review the feasibility studies, and an equally well-known computer expert to examine the design and specifications of the computer, which then had not been constructed, and advise the bank upon the probability of the manufacturer being able to deliver one which would satisfactorily meet performance tests within the time limits suggested. With very minor exceptions, the consultants approved the findings and recommendations and, in turn, recommended that the bank proceed. Late in October 1956, a contract was executed for a system to be installed early in 1958.

Immediately, plans were implemented which had been developed for the installation and operation of the equipment. The day the contract was signed, a communication was addressed individually to each of the members of the staff. This communication briefly outlined the decision to acquire such equipment and contained reassurances that its use would not adversely affect anyone's position in the bank. Prior to this various articles in the house organ were published, periodically, with respect to the studies of electronic equipment. Therefore, the notice probably came as no surprise to the employees. Simultaneously with the distribution of the notice, a meeting of the officers of the bank was called. At this meeting, initial plans for the use of the equipment were outlined. Divisional and department heads were advised that the bank would shortly seek qualified personnel to enlist as programmers, and that their co-operation in arranging for the release of any individuals, so selected, was requested.

On October 29, 1956, the first of several courses which were to follow was inaugurated, designed to train personnel and programmers in the use of the equipment. Each of these courses was so arranged that the first 2 days provided an over-all discussion of the system; the first 2 weeks, a more detailed explanation of how it operated and how it might be used; and the remaining 4 weeks, actual training in the details of programming. Many of the senior officers and divisional heads attended the first 2 days of these courses, and department managers and supervisors attended the 2 weeks' courses.

In all, 23 people were selected for the programming staff. The group included staff members, supervisors or junior officers within the areas initially to be programmed, selected for their experience in these areas. In addition, the group included three persons selected for their knowledge of, and familiarity with, the installation and operation of tabulating equipment, and four who were on our Methods Department staff. Following the completion of the 6 weeks of basic training provided by the manufacturer of the equipment, these people then attended classes of instruction intended to initiate them in a method of systems analysis which would be consistent in each of the areas to be programmed, and which, in the bank's judgment, would meet the needs of an electronic installation of the type that was involved.

On January 2, 1957, the programmers were divided into three groups, one to cover deposit accounting, one to cover corporate trust work, and one to cover loan operations (the latter includes commercial loans, mortage loans, instalment loans, and factored accounts). Each group consisted of one person formerly associated with the Methods Department, one person experienced in the use of tabulating equipment, and two or more who had been drawn from the operating departments.

The first activity of each group, the most time-consuming of the entire operation, was to analyze in detail the current operations in the area to which they had been assigned. The analysis consisted of two basic factors. First, they were required to flow chart, in detail, the flow of data from the time it entered the bank to the time it was ultimately disposed of in the accounting procedures. The type of flow charting used was considerably different from that which might normally be expected. Only five basic symbols were used, and emphasis was placed entirely upon the form of, and the movement of data, rather than who handled it, or through what departments it passed. Very little attention was actually given to departmental lines in this analysis. The second factor in the study was an analysis of the forms used to convey these data through accounting procedures. A copy of each piece of paper used, whether a preprinted form or a piece of scratch paper, completed with a typical entry or entries, was attached to an analysis sheet which gathered those statistical data which are so essential to properly program a computer. The estimates indicate that, to date, approximately 65% of the time devoted to the development of the program was spent in defining the problem, 25% in solving it, and 10% in translating it into machine language.

By the fall of 1957, plans had been developed for the preparation of the site in which the machine was to be located, and toward the latter part of the year site preparations actually began. The system, containing all of the units which were ordered, normally required approximately 4,500 to 5,000 square feet of floor space including the field engineers' quarters and the central power supply. However, in order to provide room for expansion of the system to accommodate such things maintain an alphabetical file in this way in this operation, why do you not follow the same procedure with respect to all other files?" The answer is that, in this particular type of file, a substantial proportion of the daily transactions consists of opening and closing of accounts and there is no practical way of controlling input to the system so that account numbers are reflected in the data. Every transaction must be looked up, and a number assigned to it if a numbering system is to be used. On the other hand, in an operation such as the handling of checking accounts, there are relatively few accounts opened or closed from day to day, and practically all of the transactions affecting it can be precoded with an account number, so that it is readily available when the check or deposit is presented to the bank. The low activity in the file also has some bearing, since the amount of time required for the computer to calculate the key would be quite significant if approximately 10% or more of the accounts were to be affected on a daily basis.

In addition to the applications which are presently operating on the computer system, basic programs for the loan operations are rapidly being completed, and they will be added to the electronic data processing during the coming calendar year. Thereafter, attention will be turned to such other services as personal trust, payroll, savings accounting, expense distribution, and others. At the present time, it appears that the bank will be able to handle all of these applications on the computer which has been installed and that, as time goes on, there will be need only to expand the peripheral equipment, such as the printers, to take care of these additions as well as the increases in volume that is expected.

The author has taken some time to outline the bank's experiences in approaching the use of electronics for data processing, and installing and operating a computer system. It is hoped that this outline will be accepted as being in the nature of a progress report, and that it may be of some help to potential users of electronic equipment, both large and small, as well as to those interested in the manufacture of such equipment. It is felt that many who have not already done so, are capable of accomplishing the same task and undoubtedly with better results. Perhaps this progress will encourage some to take another critical look at this tool for business. By way of summarization, the following points are emphasized:

1. The use of electronics should be of paramount interest to senior management, who must determine the result that it wishes to accomplish and be willing to support rather drastic changes in organization if need be. Select one or two individuals in whom management has implicit confidence and who have an over-all knowledge of the business and an over-all interest in its success to study the potentials to be found in the use of such equipment, and to direct the installation and operation if a computer should be ultimately selected. While much can be said in favor of committees they frequently result in extensive and expensive periods of research and little in the way of decision.

2. Make personal evaluations. Do not depend upon the manufacturers. It has been said that electronic engineers understand the mechanics of the computer but rarely the mechanics of the company. It is much easier to teach the machine to someone who understands the business, and who is progressive and willing to accept new ideas than it is to teach one who knows the machine all of the intricacies of your business.

3. Consider all your accounting problems, not just the one or two that may be foremost in your mind because they are the most critical from one viewpoint or another. A computer should not be considered as another bookkeeping or tabulating machine to be superimposed upon one or more existing operations, nor ordinarily should it be left to individual departments to decide how or why a computer should be used.

4. If the use of a computer is indicated, management has a right to anticipate higher quality work, as well as increased quantity. Question the adequacy and efficiency of present methods, but in doing so, ascertain whether or not you are talking to the man who designed them. And remember that quality and quantity can be materially dissipated by the insistence that information shall be provided exactly as it always has been.

5. If it is decided that electronics is a tool that can be applied to the specific business, select the hardware that will not only perform best now, but that will perform, at least, equally as well, in so far as possible to judge, 5 years from now. A system that is limited to fulfilling only present requirements and permitting no expansion or changes in methods may well lead to substantial and expensive changes later; changes which can wipe out all of the advantages that might be gained as a result of the initial installation.

In conclusion, emphasis should be placed on the fact that the electronic computer is a tool which, when put to proper use, will serve banks and other service businesses, as well as industrial organizations, in solving a wide variety of data processing problems. Studies which have been conducted have convinced this bank that this tool can be useful to a wide range of firms of various sizes. Appropriately, electronic data-processing systems are available in a variety of sizes, and they are priced accordingly. Businessmen should not arbitrarily assume that their organizations are too small to make it economically feasible for them to use these tools. To the contrary, avoid such ill-considered conclusions and examine carefully the possibility of whether or not such equipment will be of aid along the road to success in business.

Discussion

R. D'Antonio (International Business Machines Corporation): For commercial applications, how sophisticated should the error detection system be? What is the maximum time allowable between failure and detection?

Mr. Taunton: I do not think that you really need too sophisticated an error detection system. I think in business we need to know immediately that an error has occurred. In using a computer on an application, you are processing a tremendous amount of data at pretty high speed. If we are getting off the track because the computer is not following through, we should know it so that we can stop the process. We want to know whether we are doing the job, or whether the maintenance engineer should repair the machine.

As to maximum time allowable between failure and detection, it should be kept pretty low. I would be more inclined to measure it in minutes rather than in milliseconds. I do think manufacturers have introduced too much sophistication into some of our data machines and made them perhaps a little more expensive than they really need to be, in order to have a good sales point in promoting a piece of equipment to uninitiated customers. I have seen one or two cases and recent announcements where the checking circuitry being offered to the customer is rather expensive and not at all economically justifiable.

The Role of Computers in Air Defense

W. H. TETLEY

IN GENERAL, the mission of the Air Defense Systems Integration Division is to insure a properly time-phased and effective Air Defense Mission System. (The definition of mission system will become clear later on in this paper.) With such an aim in view, the basic commodity with which this organization must deal is systems engineering, for it is necessary to combine the various functions of weapon systems, environment systems, and support systems into a co-ordinated over-all master system.

In essence, this master air defense system is a giant servomechanism of really spectacular proportions. Although its reflex action is achieved by information feedback, it still suffers the many vagaries of the simple feedback amplifier. Consider then, the weird prospect of a servomechanism, spread out over an area comparable to the whole North American continent, going suddenly into oscillation. This, of course, is most unlikely; it is still statistically possible, however, if the programs associated with the computing links of this servo chain possess certain subtle incompatibilities. Such an instability could be exploited by an intelligent aggressor by exciting the system at its resonant frequency with cyclic probing techniques.

It is the thesis of this paper, that the role of the computer in air defense is to function as a vital element in a vast computer system under centralized control. In order for the whole to have an optimum response characteristic, all included computer functions must be entirely compatible. Although in air defense one deals primarily with systems essentially servo in character, it is conceivable that in the face of certain inevitable odds, this idea may be abandoned in order to seek more sophisticated solutions along the lines of Descartes' "reasoning machine" or Shannon's automated chess-player.

In what follows, then, rather broad terms will portray what many in systems engineering believe lies ahead for the computer in air defense. Naturally, one cannot discuss in an unclassified paper the actual response characteristics of present and programmed systems. The systems engineer can, however, knowing the trends of the basic parameters which determine this response, postulate on the structure of the theoretical system required to solve the intercept equation.

The initial parameter governing this calculation is, of course, the potential enemy threat, which in the case of air defense will be some optimized combination of aggressive aero-space intruders. The term aero-space intruders connotes both aerodynamic, ballistic, and orbital vehicles. The second parameter, the defensive weapons system, is likewise an optimized combination of aero-space vehicles. The so-called Mission System, then, must insure that the latter combination can successfully vie with the former. Under the necessary wraps of security, its existence must be viewed only qualitatively here, and defer the actual assigning of magnitudes to the military planner and his budgeteer.

Discussion

In discussing air defense on this theoretical basis, the author has chosen to consider two levels of operational management, the continental or national level which ostensibly would be concerned with an intrusion of the aero-space anywhere over the North American continent, and the regional level which, for purposes of this discussion, represents a manageable area containing at least one basic mission system.

Theoretically then, it is the role of a regional control center to manage the mission system within its confines and to co-ordinate its activities with adjacent regions. And by the same token, the role of a national command center would be to manage all regions and to co-ordinate its activities with adjacent national command centers. It shall be assumed, for the purpose of this discussion, that adjacent command centers would exist for national level control of offensive mission systems as well as national level control of the domestic and "home front" civil defense programs. Precedence for this idea lies in the present Command Center of the Office of Defense Mobilization (ODM) and the Strategic Air Command (SAC) control center. Of course, since this paper deals only in theoretical terms, other national level command functions are not ruled out.

This paper will consider all computer functions which contribute to the notion of a national air defense effort. These computers would perform generally within the following areas of consideration:

1. The local computer functions located within the confines of individual regions and associated directly with individual weapons and their support systems.
2. The regional computer functions which, by the use of local or slave computers, perform the so-called air defense mission.
3. The central computers at national level would manage and co-ordinate all regional air defense activities.
4. Numerous support computer functions outside of air defense regions but which devote considerable frame time to the support of the air defense mission.

A distinction will be made here between computer functions, computer systems, and the actual computers themselves. Computer functions are definable functions closely associated with the ten basic air defense steps which will be enumerated later. Computer systems are networks consisting of the central computer function and its remote or slave functions; collectively these systems in different degrees of complexity, would perform at both regional and national levels. And finally, the term "actual computers" refers essentially to the hardware and the associated programs which perform compatibly within the computer system.

Definitions of some of the more important terms follow:

System. There is no quick and easy definition of the system. It does, however, possess at least the following properties:

1. It is an ensemble of specific functions.
2. It is a complete entity and definable within a boundary.
3. Coupling exists between these functions.
4. It has a definable input.
5. It produces a definable output or product.
6. Satisfactory operation of the individual functions does not necessarily insure satisfactory operation as an ensemble.
7. It is susceptible to a generalized form of the "Second Law."
8. In addition, it quite often possesses the property of being a servomechanism and of involving stratagems.

Mission System. The smallest ensemble of air defense functions capable of performing the air defense mission. It is interesting to compare the mission system with the generalized system previously mentioned:

1. It is an ensemble of ten basic functions.

COL. W. H. TETLEY is with the Air Defense System, Integration Division, L. G. Hanscom Field, Bedford, Mass.

2. It is definable within a region.

3. Information, economic, and logistic coupling exists between functions in a complex fashion.

4. Its input consists of both information and equivocation.

5. Its output constitutes information both for weapons guidance and for collateral action by cognate agencies.

6. It has pronounced Gestalt property in that the system characteristic is not directly ascertainable from the performance of individual functions.

7. It is susceptible to spontaneous and to deliberately generated equivocation.

8. It is an information servomechanism.

9. It plays a two-person zero-sum game.

Air Defense Mission. The sequence of ten basic events bridges the gap between intrusion and interception. These ten functions are: intrusion, detection, identification, surveillance, assessment, decision, commitment, tracking, guidance, and interception.

Intrusion. The appearance of a vehicle within the aero-space volume of primary interest to a regional mission system.

Detection. The gaining of enough information to indicate that an intruder has entered the aero-space.

Identification. The gaining of sufficient additional information to indicate that the intruder is either hostile or friendly.

Surveillance. The gaining of still more information concerning a hostile intruder, such that it is possible to decide upon a future course of action and, if so indicated, program an interception.

Assessment. The act of determining the severity of the intrusion relative to the status of possible counter-weapons within the regional inventory.

Decision. The act of deciding whether or not to initiate an interception, or to take other appropriate action.

Commitment. The selection and launching of the optimum weapon configuration and the calculation of its trajectory.

Weapon Tracking. The gaining of information concerning the status of the weapon after its commitment, with a view to comparing this information with the information gained by surveillance of the intruder.

Guidance. That information which is supplied to the weapon as a result of initial trajectory calculations and updated by comparing the tracking and surveillance information.

Interception. The successful completion of the intercept problem within the real time period allotted.

Consider in a little more detail, the steps following an intrusion:

Detection

Intruders are detected in the aero-space by the long-range radar warning function. Equipment for this purpose is located on land in the Far North, at sea on the Texas Towers, and aloft in the aero-space itself on vehicular platforms. It can be expected that at any time that the information gained by these various facilities will be seriously deteriorated by equivocation, especially at a time when the information is needed most. Equivocation means the introduction of all spurious inputs which in the Information Theory Analogy would characterize the system as a noisy channel. This equivocation will be generated by man as well as being of natural origin. This deterioration of information can be expected to become increasingly more troublesome in systems of the future and to impose a most serious limitation on the precision and success of long-range detection processes. The Theory of Information indicates that, given sufficient time and the appropriate coding techniques, it is possible to recover considerable information from cluttered channels. Thus, the system (the mission system) must buy the precision that it needs with that other very scarce commodity, time, with some hope of its eventual recovery elsewhere in the system. Due to the stringent restrictions placed on the system by the real-time solution, there will be little or no time available for the recovery of information from the highly cluttered channels likely to be encountered.

Here, then, is the first challenge to the computer engineer, to design data recovery computers which will be both efficient and rapid, and to design into those subsequent computers which use this information, sufficient speed to compensate for that lost in the early purging processes.

Identification

Assuming that an intruder has been successfully detected, it is vital that the vehicle be positively identified as either hostile or friendly. The process of identification is an immensely complicated and highly classified one. For that reason, it will not be discussed in detail here. It can be said, however, that if normal identification processes fail to identify the intruder, knowledge as to the status of all friendly vehicles in the aero-space must be immediately brought into play to correlate the position of the intruder with the positions of all known vehicles in the traffic inventory. Again, this is immensely time-consuming, and if it is necessary to go through the process of normal identification and simultaneously through the process of correlation with known vehicle positions, adequate storage and versatility must be engineered into the identification function. Any time lost here must also be recovered by subsequent computer functions. This, of course, becomes increasingly more critical as the number of remaining functions decreases, since the time recovery must be prorated over fewer and fewer operations. And, as will be shown, the subsequent operations involve calculations which in themselves need all the possible time that can be alloted to them.

Suffice it to say, however, that the process of identification is not only time-consuming but subject to much the same equivocation as is the detection process. Here again, the computer engineer faces the challenge of designing an identification program which is error-proof, versatile, and fast.

Surveillance

An intruder having been positively identified as a hostile vehicle, becomes a target, and must be placed under radar surveillance. The surveillance apparatus is much more precise than that used in the detection process, as it must acquire the position and velocity information needed to both plan and eventually execute an interception. Again this process is subject to the pressing problem of equivocation; and as this information must be highly precise, the time lost in recovering good data from the cluttered channel might be considerable. Inasmuch as the surveillance information has a vital role in determining the choice of weapon as well as its trajectory or profile, extreme care must be exercised in obtaining the very best information possible during this phase.

Assessment

In addition to completely assessing the nature of the hostile intruder, it is necessary to have timely access to the status of all weapons in the regional inventory in order to make an immediate assessment of one's own weapon diversity. This information results in the choosing of an optimum weapon configuration with which to retaliate. When one realizes that a potential intruder can be travelling at velocities many times the speed of sound, it is evident that this complicated inventory procedure must be done with extreme rapidity. As can be seen, this function is essentially one of extracting data on land-based weapons from computer storage and combining them with continually up-dated information on the status of vehicle-based weapons. The object here is to ascertain the best of the feasible weapons, together with its preferred trajectory, and to consider its physical location relative to the intruder at key times in the immediate future.

Decision

Assuming that there exists within the inventory of weapons, one which represents an optimum solution to the intercept problem, a man-made decision must then be made, whether or not to commit this weapon against this particular intrusion. This presumes, and rightly so, that there may be very good reasons for withholding a particular weapon at any given time.

Commitment

The decision to complete the interception would naturally call for the commitment of the best weapon configuration which fell out of the assessment process. This function must recheck to make certain that Weapon *X* is currently the optimum one, and if so, proceed with the complicated task of getting it on its way. This, of course, involves many operations associated with preflight and prelaunch preparations. Actually, the central computer could delegate this responsibility to local computers at the launching point and in the weapon proper. The choice of weapon might be an air-to-air missile from a long-range launching platform, an intermediate-range ground-based missile, or in the extreme case, a short-range local defense type of missile. This type of weapon diversity follows the classic principle of "defense in depth." In any event, these weapons have local computers which are slaves of and are coordinated under the direction of the regional computer.

The high cost of equivocation should be mentioned for it is well known that redundancy is often the easy way out and is thus the price one must pay to overcome cluttered channels. A very costly solution involving redundancy would be to commit more weapons than would normally be necessary to effect the interception.

Weapon Tracking and Guidance

In addition to maintaining surveillance over the intruder, it is necessary also to keep track of the weapon in order to generate and check the required guidance information and thereby complete the servo loop. This guidance function varies with the choice of weapon, and while some weapons perform this task themselves, others are controlled indirectly by the regional computer, or in some instances by a combination of both.

Here again there is ample opportunity for equivocation to creep into the system and deteriorate the performance, for noisy feedback loops introduce a host of new problems.

Naturally, the interception is the end product of this entire chain of events, and results only after a satisfactory solution has been achieved by the ensemble of computer functions acting in unison.

This has been a résumé of the more common computer functions which must take place within the regional or mission system. Now consider the examination of those functions which are outside of the actual system itself but which support or otherwise affect it.

Centralized Control

First, there is the national level function which links together all regional computers and co-ordinates their actions by means of centralized control. This computer is able to exchange views with the SAC operational computer and with that of ODM, and thus co-ordinate air defense on the governmental level. Also, there is the problem of regional assessment and regional commitment; this parallels the problem of weapon assessment and commitment at the mission level. There might be ample justification to inhibit a particular region and to allow its adjacent region to make the interception, especially if it involves less cost and better chances of success. This would be a decision made at the national level command center, based on calculations from its control computer. In other words, it might be advantageous to optimize decision-making at the two levels in the event of repeated or continuous intrusion and where the principle of "economy of force" must be invoked.

Diversity Data Link

A system of computers is no better than the information channels which link the slave units to the central control computer. Considerable flexibility might be achieved,· and much more reliable data transmission effected, if diversity data links with appropriate programs were utilized. It might be advantageous to initiate the data transmission program at one of the ten basic steps, say identification. If the identification should turn out to be hostile, the data link program is placed into effect and the remaining configurations follow in sequence. Such a program would call for a specific data link configuration for each step. In the ideal case these would be so programmed that an optimum mode (assuming several degrees of freedom obtained) would accompany each step in the air defense function.

Program Up-Dating

The physical task of up-dating the programs for such a bewildering system of computers as indicated would be insurmountable without some form of gaming and up-dating monitor. This would be a computer function which would conceivably cycle the system through all statistically possible intrusion patterns in an effort to find flaws in the central and local programs. Actually, there must be complete compatibility between all programs involved. This gaming computer would then, by Monte Carlo techniques, along the lines of the University of Michigan and the University of California in Los Angeles traffic models, shake out hidden incompatibilities by causing the system to respond to every statistical combination of intrusion, weapons inventory, and equivocation level.

Logistical Support

The logistical support effort of this hemispheric defense complex is of staggering proportions. This function can be performed, however, by standard logistics computers of the type now in use by the Air Materiel Command. Here is fruitful ground for a program-correcting computer which profits by the misjudgments made in the normal logistics process and which constantly up-dates its program to this end.

Conclusions

The very real problems which will plague the air defense systems designer of the next decade will be summarized and this paper will close with a brief glance at the economic outlook.

Weight and Volume. A major problem area involves the enormous weight and volume of high-capacity computers. The sheer mass of present installations, even the solid state versions, seriously hamper their use in weapon systems applications where weight and volume are critical. This is now a problem of manufacturing technique. Cryogenic bistable elements seem to have proven themselves highly suited for rapid switching and occupy an extremely small volume. This is, of course, an answer to one of the many prayers of the weapons systems engineer.

It now appears possible that the techniques of handling liquid helium will eventually make this device practical for vehicle-borne slave computers.

System Compatibility. When computers are integrated into a system, there immediately arises the problem of their compatibility. In some instances, slave functions may be quite loosely coupled to the master control center. Logical devices operating in the submicrosecond range must take into account the fact that transit time of pulses, even at the speed of light, becomes exceedingly significant at these high switching speeds. The netting of several slave computers in the millimicrosecond range with their central master is impossible under present data link standards. Added to this is the Doppler effect incurred when some of the slave functions move about the region at a speed several times sonic velocity. The need for absolute compatibility could rule out completely recent ideas for the use of burst communication techniques between master and slave stations.

Radar Accuracy. Serious limitations might very well be placed on the theoretical accuracy with which both the position of a target and its velocity can be determined simultaneously. It appears to be an extension of Heizenberg's uncertainty principle, which places an upper limit on the accuracy with which both position and velocity may be obtained simultaneously.

Information Capacity. Another area of interest occurs when one considers the gigantic task of performing a running inventory of our aero-space for the exact position of all friendly nonintruding vehicles, yet such a feat may well be necessary. This problem must be considered along with the problem of equivocation, which subject this paper has touched on most gingerly, as the more quantitative aspects of this subject cannot be discussed in a paper of this sort. Shannon and others have pointed out that as long as the information rate is kept below the capacity of the system, techniques exist whereby nearly errorless information retrieval is theoretically feasible. The price, other than high redundancy, which must be payed will result in intolerable time delays, or enormous increases in information capacity. This, coupled with the gargantuan task of keeping track of all vehicles in the aerospace, appears to be almost beyond the bounds of practical solution.

Equivocation and Redundancy. It appears from here that the most pressing problem facing the air defense system designer, and consequently the designer of air defense computers, it that of processing information in the presence of almost over-riding equivocation. Unfortunately, where redundancy is needed to buy off this equivocation it tends to manifest itself in the costly function of weapon commitment. Here, in the limiting case (that of completely over-riding equivocation), one can envision the illogical solution of commiting every available weapon simultaneously. Even solutions approaching this to a small degree can become logistically nonfeasible; thus, everyone in air defense is faced with this most formidable challenge. Perhaps there is another approach to this problem, along the lines of the gaming computer which was mentioned earlier. A system controlled by such a computer might, in the presence of completely over-riding equivocation, be able to draw judgments, through the process of "statistical inference," from the meager information available. This presupposes, then, that such a computer could retain in storage all conceivable intrusion patterns, together with their appropriate countermoves, and has the facility to select from meager inputs the most probable of all statistically possible solutions. Here in this ultimate form one departs from the cybernetic character of the problem, only to become involved in a sophisticated game of chess against a player who is a recognized master of that game.

Epilogue

It seems reasonable that a group of engineers with backgrounds in computers would also have a primary interest in the economics involved in the use of computers for air defense. Therefore, the author will conclude on this note, for there is considerable correlation between the professional challenge involved and the fraction of the military budget directed toward computer functions.

It appears at this time that the total investment in ground-based computer functions will not be less than 3.3 billion dollars by the end of the 1960 decade. The investment in vehicular-borne computer functions will start to become an important factor during this period and will undoubtedly exceed 78 million dollars. In addition, leased facilities for the various support functions involved could very well amount to 11 million dollars annually. These figures include only those computer functions which are considered closely associated with the Air Defense Mission. There is good reason for the computer engineer to look upon both the material and professional roles implied herein with unusual seriousness.

Microprogramming

M. V. WILKES

BEFORE going into the subject matter of this paper, the author will report, briefly, the state of the digital computer art in Britain. Just before leaving England for Philadelphia, the author attended an electronic computer exhibition which opened in London. This was the first exhibition of its kind to be held in Europe. It was a genuine computer exhibition, not only an exhibition of computer components, and there were 11 stored-program digital computers being demonstrated. The exhibition was at Olympia, a large exhibition building in the west of London, where many of the big exhibitions are held. The computers were not occupying the largest hall at Olympia, but, nevertheless, it is a source of pride to those who have been associated with the British computer industry that it should be possible to stage an exhibition on such a scale.

Anyone who promises to give a lecture or an address and is asked for a title in advance is likely to give a vague one, in order that he need not decide what to say until immediately before he is due to speak. The author's title perhaps comes into this category since "microprogramming" is a term which has been used in so many different senses that no one is now quite sure what it means. Generally, however, it is used in some context which has to do with the design of the control circuits of a digital computer,[1–4] and that is the sense in which it will be used here.

M. V. WILKES is with Cambridge University, Cambridge, England.

The author feels some responsibility for the term since he was, he believes, the first to use it in this way when he gave a lecture at a conference held at Manchester University in July 1951.[5,6]

Up to that time it appeared that the sequencing units of digital computers had generally been designed on an *ad hoc* basis. The design engineer would cover a sequence of large sheets of paper with block diagrams of gates and flip-flops until he found an arrangement which produced the waveforms he required, and one that appeared to be reasonably economical. The author's aim was to suggest a mode of design which would not only be more systematic, but would lead to a sequencing unit with the same underlying structure whatever the order code of the machine.

The need for a sequencing unit arises because the basic operations which a digital computer is required to perform, in response to orders written by the programmer, are more complex than the basic operations which take place inside the machine itself. A machine (it is convenient to think in terms of a parallel machine) contains a set of arithmetic registers together with an adder, and also a set of control registers (a register which holds the address of the next order to be executed, index registers, etc.) which also have an adder associated with them. It is the sequencing unit which brings the machine to life by supplying the pulses which cause the suboperations required in the execution of an order to take place. These consist simply of transfers of numbers from one register to another, either direct or via an adder. An addition, for example, requires two such suboperations. The first is the simple transfer of the number in the accumulator to a shifting register and the second is its transfer back to the accumulator via the adder. Similar suboperations take place on the control side of the machine when, for example, the number in an index register is being added to an address.

There is an analogy between the way in which the sequencing unit causes the various suboperations comprising an order to be performed and the way in which the machine as a whole executes the program of orders. This analogy becomes clearer when it is pointed out that some of the suboperations may be conditional; for example, in a multiplication one must add and shift if the multiplier digit under examination is a 1, and shift without adding if it is a 0. It is this analogy which gives rise to the term "microprogramming." The microprogram is the list of suboperations or micro-operations required to execute all the orders in the order code.

It will be seen that the sequencing unit must contain storage for the microprogram. In a machine which has been designed on an *ad hoc* basis, this storage will be provided by the wiring of the control unit. In the scheme which was suggested by the author in 1951, it was provided in a diode matrix, or rather in two diode matrices. The first matrix determines the transfers which take place during each micro-operation and the second matrix determines which micro-operation in the microprogram is to be executed next. The analogy is thus with a 1+1 address order code such as is often used in a machine provided with a magnetic drum as its main store. If the choice of the next micro-operation can be made conditional, the analogy with ordinary programming is very close; there is, however, no reason why the choice of the transfers which take place during the micro-operation should not also be made conditional.

It should be noted that conditional micro-operations are used not only in arithmetic contexts such as that mentioned previously in connection with a multiplication order, but also for switching from one part of the microprogram to another; for example, in a machine with both fixed and floating point orders, the same basic division sequence may be used for both modes, switching to and from it being arranged by means of conditional micro-operations.

In a machine designed along the principles described, the engineering problems to be tackled in the design of the sequencing unit do not depend on the details of the order code which are determined by the disposition of diodes in the matrices. Two advantages follow from this. One is that the order code need not be decided on until a late stage in the construction of the machine, and will therefore be more up-to-date than it might otherwise be. The other advantage is that people with programming experience can be called in to help with the design. Apart from division of labor, this has the advantage that the resulting order code is more likely to be free from those little twists and exceptions which may seem unimportant to an engineer but which are very irritating to a programmer.

Now in operation in Cambridge is a machine with a sequencing unit designed along these lines, but making use of a matrix of ferrite cores instead of diode matrices.[7] One matrix only is used and each core corresponds to a micro-operation in the microprogram. One set of wires determines the order in which the various cores are switched and corresponds to the second diode matrix previously mentioned. The other set of wires control gates throughout the machine and determine what transfers take place during each micro-operation. Conditional action is secured by arranging that at certain intersections there are two cores. At any given time one of the cores is biased off and the matrix behaves as though the other core only were present. Which core is biased off is determined by the value of the binary digit being sensed. Some of the intersections have four cores and provide four-way conditional action.

The control matrix of the machine, which is known as EDSAC 2, contains 1,024 cores of diameter of 8 millimeters, and arranged geometrically at the intersections of a 32×32 grid. The wiring, is such that at some electrical intersections there are single cores, at others two or four cores, and at some no cores at all. By this device it was possible to construct the matrix in two dimensions rather than in three, and to secure uniform loading of the tubes driving it.

The author has written mainly about his own work in this subject and that of his colleagues. He is well aware that others have been thinking along similar lines, and that various machines have been built with sequencing units designed along principles similar to those indicated. A reaction of many engineers when confronted with these ideas is to suggest that a sequencing unit should be composed of a series of delay lines, one for each order in the order code. When a particular order is to be executed, a pulse starts along the delay line corresponding to that order, and branches off to operate gates in the machine at appropriate points on its passage. Sequencing units of this type have been built, the first of which the author became aware being in the Memory Test Computer at the Lincoln Laboratory of the Massachusetts Institute of Technology. In its simplest form this scheme requires that entirely separate sequencing equipment should be provided for each order, even though there may be some overlap of function. Conditionally controlled cross-connections can be provided between the various chains in order to obviate this objection. It may be found, however, that the complexity of the cross-connections is such that it is better to go back to the idea of using a matrix. This is certainly true if magnetic cores are used; many engineers would now agree that magnetic cores pay off as switching elements only if they are used in matrices.

A distinctive approach to the rational design of the sequencing unit of a digital computer is that cultivated on the West Coast. This makes use of Boolean techniques, and instead of the design engineer calling in some one with programming experience to help with the design of the sequencing unit, he calls in a Boolean analyst. The relationship between this approach to the subject and the one described earlier will perhaps become clearer as time goes on.

So far the discussion has been about ways of designing the sequencing unit of an otherwise conventional machine. It is an advantage of these methods that changes may be made in the order code of the machine at a late stage in its design, or even after it has been built. It is not intended, however, that such changes should in general be made after the machine has been accepted for service. Nevertheless, many people have become fascinated by the idea of a machine in which the order code would not be fixed, but could be chosen at will by the programmer. In such a machine the microprogram would not be wired permanently into a matrix but would be held in an erasable store. The programmer would set up the order code by transferring an appropriate number of words from the main store of the machine to the erasable microprogram store. Each programmer would thus be at liberty to use his own private order code, and indeed could change codes as often as he liked in the course of the same problem.

This idea is certainly a fascinating one, and it is what is meant by many people when they use the term "microprogramming."

However, the view is expressed in the lecture the author gave at Manchester in 1951 that probably there was no real requirement for such a machine, and that it can be said that events have confirmed this view. There has been no lack of engineers who would have been glad to try their hands at designing such a machine, or of programmers who would have liked to have one to play with. As far as the author knows, however, none of them have succeeded in persuading a sponsor to put up the necessary money. Perhaps this is just as well because the many problems involved in running a computing laboratory are bad enough as it is, without the additional license which would be created by a system of private order codes.

References

1. Mikroprogramm-Steuerwerk, **H. Billing, W. Hopmann.** *Electronische Rundschau,* Berlin-Borsigwalde, Germany, vol. 9, 1955, pp. 349–53.

2. A Note on Microprogramming, **H. T. Glantz.** *Journal of the Association for Computing Machinery,* New York, N. Y., vol. 3, Apr. 1956, pp. 78–84.

3. Micro-Programming, **R. J. Mercer.** *Ibid.,* vol. 4, Apr. 1957, pp. 157–71.

4. Logically Micro-Programmed Computers, **J. V. Blankenbaker.** *Transactions,* Professional Group on Electronic Computers, Institute of Radio Engineers, New York, N. Y., vol. EC-7, June 1958, pp. 103–09.

5. The Best Way to Design an Automatic Calculating Machine, **M. V. Wilkes.** *Manchester University Computer Inaugural Conference,* Manchester, England, July 1951, pp. 16–18.

6. Micro-Programming and the Design of the Control Circuits in an Electronic Digital Computer, **M. V. Wilkes, J. B. Stringer.** *Proceedings,* Cambridge Philosophical Society, Cambridge, England, vol. 49, pt. 2, Apr. 1953, pp. 230–38.

7. The Design of the Control Unit of an Electronic Digital Computer, **M. V. Wilkes, W. Renwick, D. J. Wheeler.** *Proceedings of the Institution of Electrical Engineers,* London, England, vol. 105, pt. B, Mar. 1958, pp. 121–28.

The Athena Computer, A Reliability Report

L. W. REID G. A. RAYMOND

THIS PRESENTATION provides actual performance data, which makes feasible digital computer applications that require equipment to operate 24 hours a day for a period of 7 days without error or failure, with no preventive maintenance required, and involve real time problems in which a human life is involved.

These data have been derived from the performance figures of Remington Rand Univac's Athena Guidance Computer, Fig. 1. This computer is a large-scale general-purpose, transistorized digital computer. It is part of the Radio Inertial Guidance System for the Air Force ICBM Titan. This guidance system has been developed in co-operation with Bell Telephone Laboratories, Whippany, N.J. Bell Telephone Laboratories has designed the associated radar equipment.

The guidance system consists of a ground based computer, which communicates with the missile through the radar. Simply stated, the system continually determines the missile's present position and relates it to a desired position, computes and furnishes to the missile necessary steering orders to perform corrective action so that the assigned destination is reached.

Operational Summary

In the contract issued by the United States Air Force to Remington Rand Univac, the Air Force placed specific requirements on the manufacturer as to the desired level of reliability which was to be attained. This requirement is most easily explained by saying that no more than six out of every 1,000 missile firings shall fail to hit the target due to errors or failures on the part of the computer. This converts to a mean time to failure requirement of 55.4 hours.

All information presented is based on the following definition of a failure: A failure is any computer malfunction which affects guidance. This definition includes any and all computer errors or malfunctions, which occur at any time during the operation of the computer, whether it be in performing actual guidance or in test or maintenance routines.

The sections of the computer, included in the failure reports, are those sections active during guidance or used to confidence check the equipement prior to guidance. The failures are reported against a component population in each computer system of approximately 7,000 transistors, 21,000 diodes, and 24,000 resistors. Equipement concerned with monitoring guidance operation such as magnetic tape, typewriter, and tape punch is excluded.

The information presented is based on operational experience obtained on three Athena Guidance Computers, one of which is installed at the Air Force Missile Test Center, Cape Canaveral, Fla. This system is fully integrated with the Bell Telephone Laboratories' radar system and is operated on sched-

L. W. Reid and G. A. Raymond are with Remington Rand Univac, St. Paul, Minn.

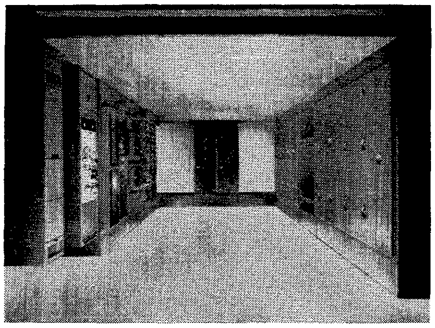


Fig. 1. The Athena computer

uled basis. The information included covers operation of these three equipments up to October 31, 1958. It covers a total operating time of 6,361 hours, 27 failures, a composite mean time to failure of 235 hours with a reliability ratio of 2/1,000. This compares with the contractural reliability ratio of 6/1000. The 27 failures occurred in 350,000,000 component hours. The detailed reliability figures for each of the computers are included in Table I.

If the failures are plotted on a cumulative basis, Fig. 2, it will be found that the failure rates of each of the three equipments appear to be relatively consistent and in smoothing the curves, a failure rate of about one failure per month is produced.

Maintenance Philosophy

In addition to being responsible for the development and manufacture of this equipment, Remington Rand Univac has the responsibility of recommending maintenance policies and manning requirements (the number of personnel required to maintain and operate the equipment) to the Air Force. In the process of developing the required maintenance philosophies and procedures, the proficiency of maintenance and operating crews has been measured as it is related to their ability to find failures and malfunctions. The initial result of this test showed that the trainees attending courses in computer maintenance demonstrated a trouble-shooting proficiency equal to or better than personnel with more than one year's experience in the field. One of the first reactions to this result was to suggest that the equipment, as designed, was not maintainable, but further analysis showed that in achieving reliability, new problems had been created. The basic principle, concerned here, is that the ability of an individual to trouble-shoot is directly proportional to the number of troubles he has opportunity to deal with. In dealing with equipment which demonstrates failure rates of about one per month, personnel who were concerned with the operational equipment were having very little opportunity to practice their trade.

For this reason a regularly scheduled program of hunting simulated troubles is now conducted at each operational site using specially produced highly reliable defective chassis. These chassis are identical in appearance to all other chassis in the computer but can be made to simulate various types of failure. A substantial improvement in proficiency has followed from this procedure.

In order to determine the number of personnel required to maintain this equipment, a sample of 4,871 hours of operation was taken and compared with the amount of time spent in looking for troubles that do not exist, that is to say, confidence checking the equipment. This is shown to be approximately 576 hours. Of this time 115 hours were spent in correcting malfunctions that did exist or roughly 2% of the total operation time. These figures would appear to substantiate the effect noted in the proficiency check, that there has been very little maintenance performed and, hence, very little practice.

Table I. Reliability Record

	Serial 1	Serial 2	Serial 3
Running Time-Hours	3,154	2,110	1,097
Meantime to Failure-Hours	175	422	274
Reliability	2/1000	1/1000	1/1000
Failure	18	5	4

Based on the knowledge that there is very little maintenance to be performed, the number of maintenance personnel required is recommended to the user. Since the equipment must be ready for use 24 hours a day, 7 days a week, current planning calls for one maintenance man on duty at all times. The lack of activity, loss of proficiency, and outright boredom of this man are problems or real concern. It is hoped that additional maintenance experience will give the required confidence so that one man can maintain a relatively large number of equipments on an on-call basis.

The point to be understood is that in achieving reliable equipment, it must be recognized that other problems have been created which, though more desirable types of problems; must be dealt with and compensated for so as to preserve system reliability.

It is believed that the results presented here constitute a genuine achievement in design, manufacture and operation of electronic equipment in general and digital computers in particular. Because of the inherent reliability of this equip-

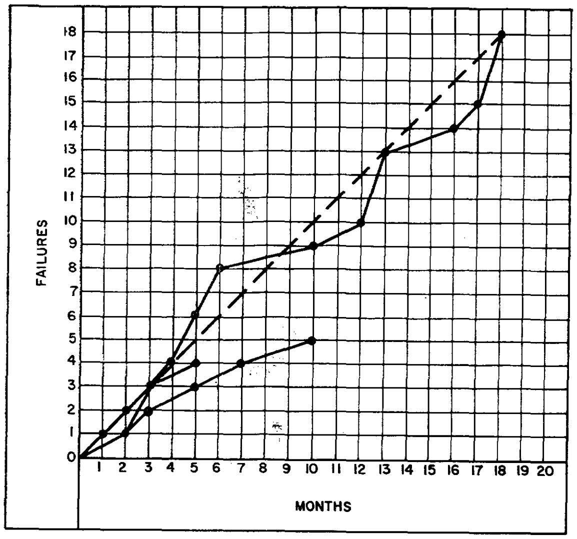
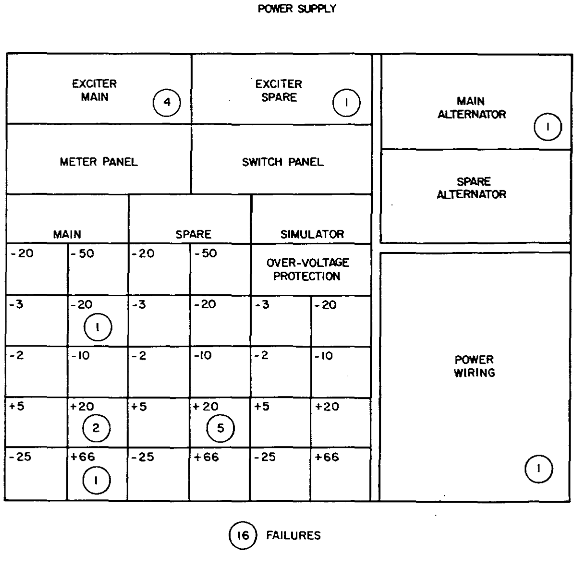


Fig. 2 (left). Cumulative failures versus times for 3 machines

Fig. 3 (right). Showing distribution of failures within power supply



ment, it is possible to install, operate, and maintain this equipment with a minimum of personnel and with a minimum usage of spare parts.

Discussion of Failures

In the missile business, success never seems to be quite as spectacular as failure. As a matter of fact, few things are quite as spectacular as a missile enveloped in a tremendous burst of flame and smoke. Fortunately, failure of a computer is only occasionally accompanied by flame and smoke.

This discussion concerns a reliable computer. It seems therefore, that success rather than failure should be discussed, but it is still necessary to look at modes of failure to grasp the real significance of what has been accomplished. As has been mentioned, 27 failures which could have affected missile guidance have occurred in 6,361-total machine hours. These are failures that could have affected guidance. Probably many of them would not have. It must be remembered that these include all failures, even those occurring during maintenance periods.

Table II shows the distribution of failures by cabinet totalled for three machines. It is obvious that investigation should begin in the power supply where 16 failures have occurred.

Table II. Distribution of Failures

Cabinet		Number of Failures
11000	Peripheral	2
12000	Peripheral	0
13000	Drum memory	2
14000	Drum and control	2
30000	Console	0
21000	Control	1
22000	Arithmetic	3
23000	Core memory	0
24000	Input-output	0
40000	Power Supply	16
70000	Simulator	1
Total Failures		27

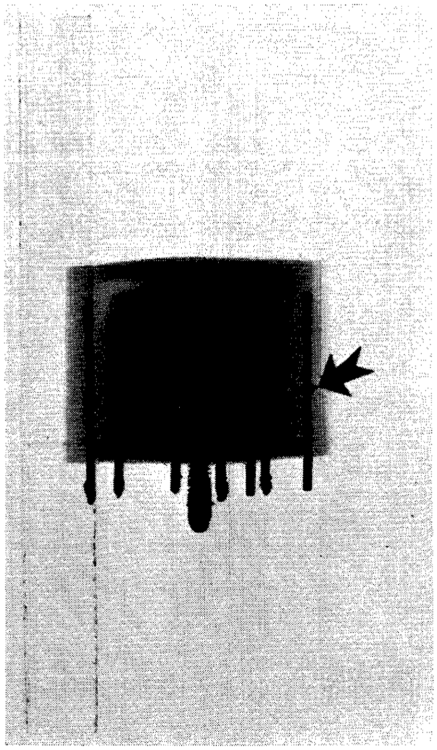


Fig. 4. X-ray photograph of failed pulse transformer

In this power supply there are 28 separate supplies arranged in the manner shown in Fig. 3. Ten are required to operate the computer. Ten are spare. Eight operate the simulator unit which is used to provide a real-time bit-by-bit check of computer operation just prior to guidance. Distribution of failures within the power supply is also shown. The two +20-volt supplies are most heavily loaded and show the highest failure incidence. The exciters are regulated rectifiers furnishing field current for the motor alternators.

The power supply is closely regulated using magnetic amplifier regulation of both the motor alternator voltage and the individual supplies. For a power supply the unit is extremely complex and quite crowded. Overcurrent limiting, overvoltage protection, and reverse voltage protection are integral parts of every circuit. There features were applied to enhance the system reliability by protecting the computer from damage due to power supply failures. But it is evident that in adding this complexity power supply reliability has been sacrificed. However, a number of design changes have been made and the failure rate now has decreased.

Ten of the power supply failures were overheated transformers, three were power diodes, one was a circuit breaker contact, one was a short circuit in external wiring, and the last was a failure of a field winding in the motor alternator.

The remaining 11 failures have occurred over a period exceeding 6,000 hours in electronic equipment containing 21,000 diodes, 7,000 transistors, 24,000 resistors, several thousand other parts, and 75 to 100 thousand soldered connections. Four of the failures occurred in wiring: one broken wire, two wires pinched by a cable clamp and grounded, and one wire scrap short circuiting two connector pins. This tiny scrap of wire caused a failure only when the humidity inside the computer was exceedingly high.

The last seven failures involve electronic chassis. Two chassis suspected as intermittent were removed. The failures have not recurred in the computers, but neither have the causes of failure been identified. These chassis have been investigated since removal in an attempt to reconstruct or locate the failure, but so far without success. Five chassis displayed definite failure of one or more components.

The sum total of all these electronic component failures in the guidance loop are: an intermittent pulse transformer with a cold solder joint, three surface barrier transistors, and two diodes which, being in series, account for only one failure. These six computer components failed during a period equalling 350 million component hours: over 50 million component hours between failure.

Fig. 4 is an X ray of the defective pulse transformer. The indicated connection was not soldered properly. A void existed in the potting around the pin so that movement of the pin could cause an open circuit.

The diodes that failed were destroyed by a short circuit which subjected them to 20 volts in the forward direction. This failure has been duplicated in the laboratory and microscopic examination verified the analysis.

Two of the surface barrier transistors that failed appear to have been destroyed by transient pulses or some external effect. One failed as a result of solution cavities accelerated by a defect in the crystal. This is a characteristic mode of failure in the surface barrier transistor. Above 85 degrees Centigrade dissolution

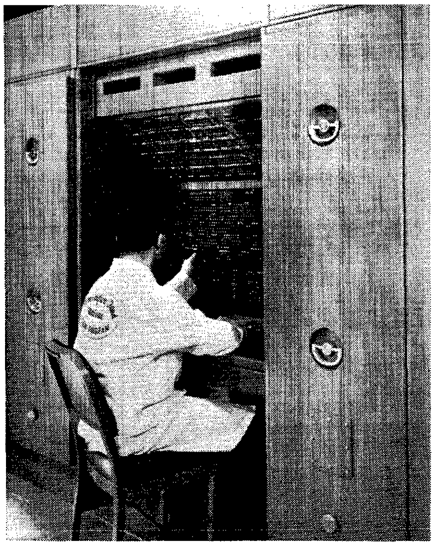


Fig. 5. Maintenance console

of germanium by indium proceeds at a rapid rate causing cavities to appear and grow until they extend completely through the base. Sufficient power applied to this short circuit may cause fusing of the internal leads and result in an open circuit.

Design Philosophy

There are differences between the design approach for a computer and the design approach for a reliable computer. It is well to consider these differences. At the start of any design program, two factors are considered; the "Requirements" and the "Tools at Hand." Now, how must the design approach be modified to achieve a reliable computer?

First, consider the requirements. The computer must have a certain speed, a memory, some type of input-output, a maximum size, and a certain reliability. Now, if as usually happens, this new requirement is ignored, either by the customer in his requirements or by the engineers in their design, the result is a conventional design and the usual reliability.

The first problem, then, is to help each individual involved in the design program see this new requirement, realize that is a design requirement and develop his design around it. Reliability must be come the most important requirement. It must be the most important requirement for each phase of the design and for each section of the computer. It cannot be left to be added at some convenient time late in the program.

The first step in the design program must be reliability education to get the reliability concept across to design engineers. Once this idea has been absorbed, the designer will apply the tools of reliability to the designs. The tools of reliability are methods used to reduce the probability of failure. Each phase of the design contributes to failure in many ways. Designing for reliability requires the determination of all modes of failure and the application of methods to eliminate these failures.

Logical Design

Examples of this approach are found in the logical design of the Athena computer. The computer program is fixed and stored on a drum. It cannot be modified, even intentionally, during guidance. The writing circuits are disabled so that nothing can disturb the program. A momentary failure or transient error cannot cause a permanent change in guidance.

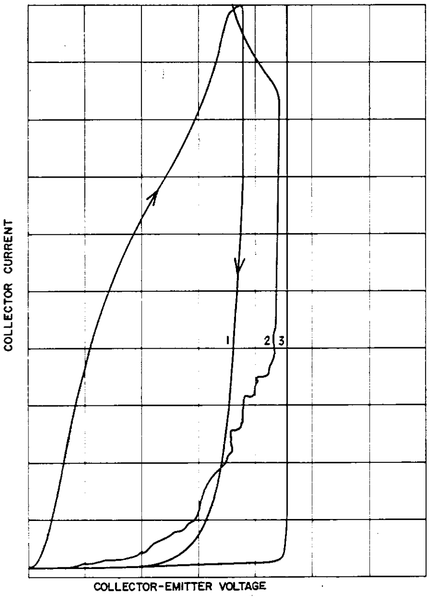


Fig. 6. Static characteristics for three transistors of the same type from the same lot showing wide variation in characteristics

The logical design recognizes the limitations of the building block. Operation is parallel rather than serial to reduce the circuit speed. Additional hardware is traded for slower, more reliable circuitry. Design rules for interconnecting the building blocks provide ample tolerance for variations in delay as components age. Parity checks are accomplished during program loading, and a "verify mode" is provided for verifying the fact that complete and correct data actually has been stored on the drum. A simulator using paper tape produces a bit by bit check of the output of the computer for a predetermined set of input data.

Special modes of operation, selectable from the console, aid in checking. A special exercise ties the computer input to the output and verifies correct operation. Step-by-step or slow-speed operation is possible.

Built in checks on the timing of synchronizing pulse and for overflow detection detect malfunctions as well as programming errors.

Both the one and zero side of every flip-flop are indicated on the maintenance console, Fig. 5. During trouble shooting the exact condition in every register is apparent at a glance.

Turning the test-guidance switch to guidance disables all other controls on the maintenance console so that operator error cannot cause a failure during guidance.

Component Design

Component design is equally important. Chassis connectors are designed for reliable contact at low potentials. High contact pressure (1 $^1/_2$ pounds), four contact points, gold plating, and simple design all contribute to reliability.

All components for the computer have been selected for their reliability, with electrical characteristics considered a secondary factor. Large samples of each type have been subjected to severe tests to failure. These tests to failure are important, for the modes of failure and the degradation characteristics of components are added tools needed to convert a design into a reliable design.

Static characteristic curves of semiconductors often reveal significant information for predicting early failures. Where reliability is important, the recording of characteristics of each component is one of the best ways of sorting out the poor units from a lot. Many things instantly apparent from the characteristic curves are completely missed in point measurements, Fig. 6. The Athena circuit designs were based on realistic design standards which provided for normal component variation. Results of many component testing programs were reflected in the design of the circuitry so effectively that not one computer failure has been chargeable to component degradation. Every com-

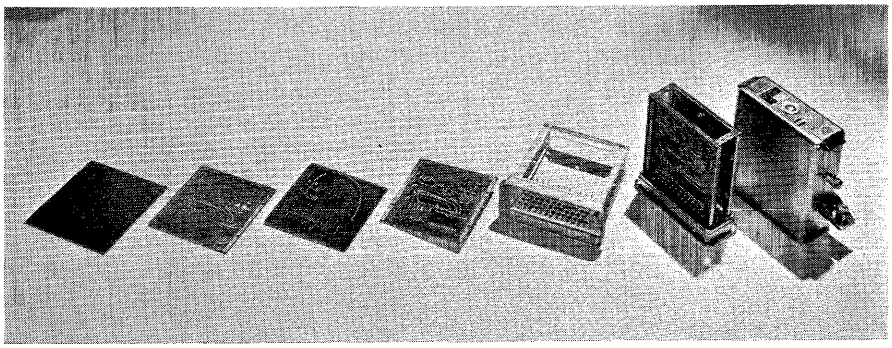


Fig. 7. Stages in the assembly of a sealed chassis unit

ponent failure has been of a catastrophic type.

Process Design

The final design step, the manufacturing process design, is most important and most often neglected. Careful, thorough process design to insure a consistent, high-quality product further improves the reliability of the best designs.

A clean, comfortable environment, tools, fixtures, jigs, and automated processes all have helped to produce this reliable equipment. Careful design considered the small details of wire preparation to be as important as any other part of the process. Soldering techniques have been devised to emphasize and treat each connection as an individual operation.

Conclusions

This paper has summarized significant aspects of the operation of a highly reliable computer. A different approach is required in maintaining this equipment because of the low failure rate just as a different approach was required in designing the equipment to achieve the low failure rate. The entire program has required careful attention to minor details and has emphasized the importance of every operation, every component and, especially each person involved. Where reliability is a design requirement, success or failure hinges on the attitude of the people doing the work.

Discussion

Question: What bit rate did you use in computing mean time failure for transistors, diodes, and solder joints?

Mr. Raymond: We did not use a bit rate in computing the mean time failure.

I believe you are referring to the bit rate for component failures used at Bell Telephone Laboratories. The mean time to failure, 55.4 hours, is derived from the requirement of not over six failures per thousand missile firings rather than from the component failure rate. We have not made use of failure rates in computing mean time to failure since existing failure rate data is not based on sufficiently well-controlled experiments to give us adequate confidence in the results.

Question: Would you state the number of failures which could have affected guidance?

Mr. Raymond: Any of the twenty-seven failures in 6,000 hours could have affected guidance. It is possible that some of them would not have affected guidance. Some of the power supply failures might have been in the spare equipment rather than in the operating equipment and thus might not have affected guidance, but all 27 could have.

W. L. Anderson (General Kinetics, Inc.): Are any reliability data available which includes magnetic tape units?

Mr. Reid: There were some circuit design problems that have been straightened out and we now have very, very few failures. There have been perhaps two or three failures in the entire period of operation of these three machines.

Arnold Jorgensen (ElectroDATA Div., Burroughs Corporation): Are silicon or germanium transistors used? What is the junction temperature? How much did you derate beta from the minimum beta given on the data sheets.

Mr. Reid: Germanium transistors are used with the junction temperatures as low as practicable. The computer is air-cooled to about 60 degrees Fahrenheit. The transistors are derated to such a degree that I believe the junction temperature is very close to this figure.

The minimum beta on the surface-barrier transistor procurement specification is 12. The circuits were designed to tolerate a beta of 8.

J. Lawler (Radio Corporation of America): Why are 1 and 0 sides of flip flops indicated at the console?

Mr. Raymond: This is merely redundancy. The indication is by means of neon lights. One light must be illuminated and the other not, at all times. Trouble exists if both lights are on or off at the same time. With only one light, an additional test is necessary to eliminate the possibility of a defective indicator causing an erroneous indication.

J. Lawler (Radio Corporation of America): What is the background of the maintenance men on your system, and how long did it take to train them?

Mr. Reid: The background of the maintenance personnel on this system consisted of either formal engineering education or training and experience received as members of the armed forces. We have enjoyed particularly good success with personnel whose military experience lay in the area of maintenance of relatively complex systems.

The training period required approximately 5 months.

E. Strandberg (Radio Corporation of America): What is the scope of the training for the maintenance personnel? Did you check each component individually before assembly?

Mr. Reid: The scope of the training includes the basic circuiting of the equipment, programming, and a detailed understanding of the logic and command structure.

Each component was checked before assembly.

P. J. Scalga (General Electric Company): Do you use marginal test techniques?

Mr. Reid: The equipment has provisions for variation of all supply voltages to simulate marginal conditions. To date, however, this has proved to be of limited value in maintenance practice in that the circuits are designed for wide variation in component characteristics and the components have been carefully tested and selected so as to reduce the variation of their characteristics with time.

Mr. Davidson (Nuclear Development Corporation of America): You state the number of failures (27 in 6,000 hours) which could have affected guidance. How many total failures were there which might have affected computation in some other computer? How many total failures (raw data)?

Mr. Raymond: There have been no failures affecting computation other than those listed. There have been failures in the tape reader, tape punch, magnetic tape units, and typewriter which would have affected the use of the computer for data processing but not for real-time control operations. There have been perhaps three or four of these for every guidance affecting failure.

The Philosophy of Automatic Error Correction

R. M. BLOCH

AUTOMATIC computing and data-processing equipment is depended upon today to perform a substantial portion of all data-processing requirements of government and industry in this country. It is natural, then, that the issue of reliability and the related issues of error detection and error correction have assumed a position of utmost importance from the viewpoint of users of such equipment. The manufacturers of data-processing equipment, recognizing the important role of their systems in the nation today, have likewise turned their attention to these areas and are bringing their most potent forces to bear on these vital issues.

Many important technological advances which are being incorporated in present-day machines are substantially reducing the frequency of error commission. The increasing speed with which these machines operate, and the corresponding increase in work load, have made it mandatory that not only the rate of error commission be reduced, but that efficient means be found for the treatment of these errors when they do occur. It is in this latter area where the principle of automatic error correction is now coming into the forefront of consideration.

Error Detection

In any technique for automatic error correction, the prerequisite of automatic error detection must exist. In this regard the manufacturers of data-processing equipment have developed many types of automatic detection techniques during the course of the past decade. Certain of these techniques are based upon maneuvers in the area of machine programming. This mode of attack is particularly suited to processing involving a great deal of mathematical computation, where various types of check formulae can be brought to bear to detect inconsistencies in the results obtained. The correction technique generally employed here is that of rerunning a portion or all of the program in which the error or errors are suspected to have occurred.

A second technique of detection involves that of a redundancy in equipment, specifically arranged so as to yield immediate detection unless two or more units have malfunctioned at the same time and in the same fashion. Although this method has been utilized, its applicability will probably be limited in advanced equipments. This is primarily due to the costs involved in the additional equipment, as well as certain difficulties in the subsequent application of modern error correction techniques.

The detection technique which has been put into practice most frequently involves that of redundancy of information. The redundancy has ranged from complete duplication of information to simple parity checks of long trains of information. Nearly all electronic data-processing installations in use today apply this concept of redundancy in one form or another. The power of these redundancy checks has usually been evaluated on a theoretical mathematical basis. That is to say, it has generally been assumed that the frequency distribution of error patterns is based upon independent and equal probabilities of single digit errors. This premise was not unreasonable in the absence of evidence to the contrary. However, sufficient operating data on various equipments are now being assembled, which indicate that this assumption is unjustified. These data indicate that electronic equipment is, in fact, subject to certain repetitive error patterns, and that the probabilities of single digit errors are neither equal nor independent. When one considers the fact that a complex digital system is comprised of discrete electronic elements, then it becomes increasingly apparent how difficult it is to predict in advance the nature of the error patterns. Transistors, resistors, diodes, and the other basic elements of an electronic system have inherently different reliability characteristics. This in itself would indicate that certain error patterns may be much more likely than others. To make matters more complicated, different systems, indeed, different sections of a given system, make use of these components in varying configurations. To illustrate this problem, Fig. 1 shows a system wherein four parallel trunks are emanating from a memory array. These trunks are then shunted into a single operational network. It is not important for the purpose of this discussion to detail the precise operations being performed within this large network. It is only essential to understand that there are elements within this network which can fail in such a way so as to affect the information on several of the output lines. Furthermore, if it is assumed that each of the entering trunks contains a continual flow of information digits, it is clear that the characteristics of the circuit failure with respect to time must also have an effect on the error content of the output information. If the component were to fail permanently, one would expect the error pattern with respect to time to be perpetuated. If, on the other hand, the failure is of a transient nature and of very short duration, one would expect a corresponding error pattern of short duration to result in the exiting information.

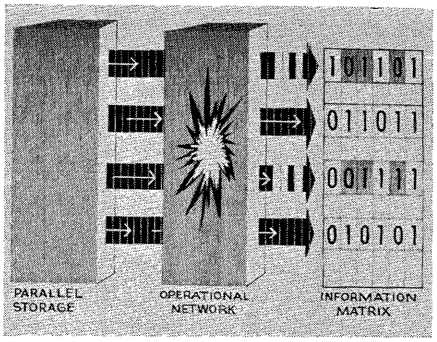


Fig. 1. Parallel information flow through a single operational network

Suppose the system design is now changed so that the four trunks emanating from memory are treated in independent operational networks, as shown in Fig. 2. A failure in one of the four networks will now presumably have no effect upon the proper operation of any of the three other networks. In particular, an error in the functioning of Network 3, as shown, will surely affect the output of this network; whereas the other three networks will be yielding correct information. Although the error patterns of a given trunk as a function of time may conceivably be similar in the two systems, it is obvious that the error pattern in Fig. 2 will generally be confined to a single trunk, whereas the pattern in Fig. 1 may frequently be distributed across several information channels. The implications of these two elementary diagrams upon the theory of error detection, and as will be seen later, upon error correction, are clear. Regardless of the error frequency, System 2 will yield a heavily biased error distribution

R. M. BLOCH is with the Datamatic Division of Minneapolis-Honeywell Regulator Company, Newton Highlands, Mass.

when viewed across the entire matrix of information.

Now if the operational network in Fig. 1 fails for an extremely short time duration, it is quite probable that a single column of the information matrix will be in error; several elements of this column, however, are likely to fail during this interval. If it can be demonstrated statistically that by far the greatest proportion of errors is of this nature, then a detection system which is capable of ferreting out a single error in a row of the matrix would be ideal. To the contrary, a detection system which will intercept a single error in a column might be very impotent indeed. If the transient failure in this network were of long duration, then single error detection in either dimension is ineffective. In the independent network array of Fig. 2, however, a somewhat different situation obtains. A transient of length equal to a digit time will likely affect but one element of the matrix and may be intercepted by any single error detection scheme. Failure of longer duration will lead to a multiplicity of errors in a single row; here, a single-error detection system in the columnwise direction will be extremely powerful, and single error detection along a row would be useless.

Empirical Concept of Design

From considerations such as these, a new approach is being taken by data-processing equipment manufacturers with regard to the design of automatic error detection and error correction networks. Heretofore, detection techniques were specified by systems personnel based upon theoretical formulations. These specifications were then transmitted to the circuit design area where the detection system became a physical reality. In this arrangement, the circuitry designers used their best reasoning and experience to assist the systems staff, but the specifications were still born of abstraction.

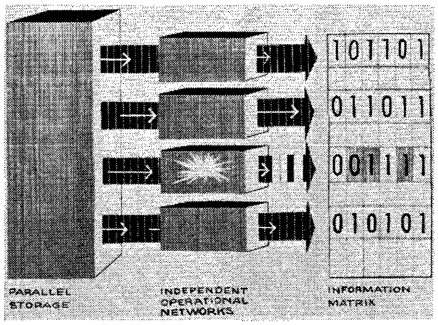


Fig. 2. Parallel information flow through independent operational networks

The new concept, which is coming into practice as of today, in effect states that the equipment itself is best able to define its own idiosyncrasies in respect to error behavior. The principle is the same as that which has been responsible for many outstanding advances in the field of science, namely, the use of empirical evidence to substantiate or refute a theory which has been propounded. Through laboratory observation of actual data-processing networks, it is possible to build up a sizable array of statistics on the error behavior of these very networks. These behavior statistics then enable the system designer to formulate a very powerful detection technique which has been tailored to the particular error pattern of the equipment. In actual practice, an interchange of information between the circuit and system groups not only may affect the detection technique used, but also may have its effect upon the original circuit design as well. Indeed, it is possible that certain changes may be made in the circuit design to yield an extremely favorable bias of error distribution for purposes of interception by a given detection system. Such a change in design is then reconfirmed by further empirical laboratory work, and thus the circle is eventually closed to yield a proven and powerful detection arrangement.

The final design that comes forth under this new concept may indeed defy the intuitive judgment of both the systems and the circuitry groups, but its validity cannot be denied; this will be borne out when the machine is put into operation in the field.

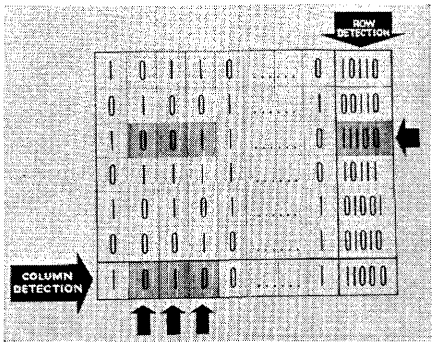
Error Correction

There are two broad classes of error correction. The first or "reversion" class consists essentially of re-executing the operation which is in error immediately upon detection. This class of correction includes all forms of program rerunning which involves reverting to some segment of the program prior to the error in question. In general, this class requires that the information in its correct form be retrieved from some previous point in the operation. It is presumed that this information in said correct form is available within some storage network of the equipment.

The second class of error correction, "deductive" correction, involves a powerful detection scheme which is capable of isolating the actual elements of information which are in error, correcting these elements, and proceeding with the program in progress. All error correcting codes have these characteristics. It is not necessary to revert to a previous point in the program, or to have access to the storage of the information in the correct form. It is literally possible to recreate the correct information by use of additional digits which have been transmitted along with the information.

The reversion class of correction has been used for some time. It is especially successful in those sections of the equipment where the retrieval of the original information is not difficult. For example, in arithmetic operational networks, once an error has been detected, say in an addition process, it is often possible to have the machine automatically retrieve the operands from memory and proceed once again with the addition process. Rereading or rewriting of magnetic tapes, once an error has been detected, are also reversion forms of correction which are being employed in various equipments today.

The second or deductive class of correction is much more difficult to deal with in practice, although it has certain outstanding advantages. Its primary importance rests in the fact that in certain areas of data-processing and computing systems it is extremely impracticable to retrieve the information in its original or correct form, and in some cases impossible to do so without a complete cessation of machine operation. It is for this reason that a great deal of attention recently has been directed to redundancy codes which can perform the function of automatic error correction. Inherent difficulties accompany the use of such codes, and these difficulties have thus far precluded the possibility of their use in any practical way in data-processing equipment. These codes generally have limitations upon the number of errors which they may detect, and if these codes are expanded so as to enable correction of a multiplicity of errors, then the redundancy required may become excessive, and the total information capac-



1	0	1	1	0	......	0	10110
0	1	0	0	1	......	1	00110
1	0	0	1	1	......	0	11100
0	1	1	1	1	......	0	10111
1	0	1	0	1	......	1	01001
0	0	0	1	0	......	1	01010
1	0	1	0	0	......	1	11000

Fig. 3. Typical information array showing principle of two-dimensional detection

ity of the system is thereby severely reduced. If, on the other hand, the number of error digits exceeds the limit, then actual erroneous correction may take place; this characteristic further complicates the usefulness of such codes.

Orthotronic Control

A system of automatic error correction has recently been developed and is being put into actual use today. This system, Orthotronic Control, utilizes a strong two-dimensional detection technique capable of isolating and correcting nearly all errors committed in the system which it monitors. The new concept of design of error monitoring systems, which was discussed earlier, was put into practice in the development of Orthotronic Control. To describe this correction technique, let it be assumed that information is transmitted along parallel channels from memory, enters an intermediate buffer storage stage, and is then recorded upon magnetic tape in a multiplicity of channels. At some later time this tape is read and the information again is transmitted through parallel channels to an input buffer storage stage and thence to the main memory. The correction system is intended to retrieve automatically any information which has been lost in the course of the full transmission cycle from the time information leaves the memory to the time that it is returned to the memory; the transcription to and from magnetic tape will have occurred in the interim period. Great strides have been made in the reliability of the input-output trunk systems of data processors; yet is it still generally true that this segment of any processing equipment is probably more prone to error than all other segments combined. It is, then, in this input-output trunk network where Orthotronic error correction comes into effective operation. Several channels of information are recorded on magnetic tape, as shown in Fig. 3. Each of these channels has been independently controlled by separate sections of the output buffering system, as well as independent writing circuits. The correction system adds an additional channel, referred to as an Ortho-channel. This channel consists of a simple parity count of the corresponding digits in all of the information channels on a column-by-column basis. This Ortho-channel is automatically generated as information leaves the main memory of the system enroute to the output buffer section. The information channels, as well as the Ortho-channel, are simultaneously recorded on magnetic tape. A series of check digits which are associated with each information channel, and which were actually stored with the information in the main memory, are also recorded at this time. These check digits may be referred to as row-detection digits in contrast to the column detection digits comprising the Ortho-channel.

Experimental evidence has shown that in a system such as the aforementioned, where parallel and independent transmission is in effect, errors will tend to be localized in a single channel as a result of any given equipment malfunction. In view of this behavior pattern, only a single binary digit may be expected to be in error in any given column. However, since the error disturbance may well affect a great number of binary digits within a single row, it is necessary that a much more powerful detection scheme be used in this dimension. Here again empirical evidence performs a most useful function in determining the principle upon which the row-detection digits are to be constructed. Assume, for example, that groups of 100 binary digits are to be monitored in each row, and this comprises a block of information. If, now, it can be determined that nearly all errors are of such a nature as to convert 1's to 0's but leave the 0's undisturbed, then a set of check digits which is nothing more than a binary sum of the 1's in this particular channel will prove to be a very powerful detection arrangement. If, on the other hand, erroneous inversion of a 0 to a 1 is as likely as the opposite transformation, then a binary count technique is not as effective, and some other technique may be better employed. For example, it is possible to use different weights in successive columnar positions over any arbitrary span to form an arithmetic or logical sum of the weights corresponding to those columns in which the digit 1 exists. Generally, the greater the number of digits used in the check sum, the greater the power of the detection system. It should be noted here, of course, that the use of a simple parity bit is completely ineffective in the light of the error pattern being discussed; any even number of errors within the channel would not be detected at all.

It is possible to arrange a weighting system which is distinctly biased so as to intercept the most common error patterns to be anticipated. Thus, suppose it is found that the predominant error patterns on magnetic tape involve a relatively small number of binary digits within a moderate span in a single channel. A weighting scheme has been devised for this pattern which intercepts all possible single bit, dual bit, and triple bit errors, and furthermore intercepts all but two-tenths of one percent of all other possible errors within a span of nearly 1,000 digits. This detection capability is valid whether the digit errors are due to drop-out of information, pick-up of false information, or any combination of the two whatsoever. When such a detection scheme is used, it can be confidently expected that a negligible number of errors will escape the detection network.

Once having established, then, what may be termed a strong row check, isolation of the actual erroneous digits is a rather simple matter. The row-detection digits are used to sense the channel in which the error has occurred, and the column-detection digits will indicate precisely those columns in which a digit is in error. It is then possible to have the machine automatically invert the erroneous digits and thus reconstruct the information, bringing it into its correct form. It is a characteristic of this Orthotronic system that whenever a rare error pattern occurs which is not capable of correction, the system is able to discern this fact; and fallacious correction will not occur. For example, if on occasion an error pattern were to occur extending across two or more channels within a given block of information, then it would not be possible in the system shown in Fig. 3 for correction to take place. However, in this case the two row-detection digit arrays will reflect an error, and this will be used to signal the uncorrectable condition. Another characteristic of the system rests in its ability to determine when errors have occurred in the check digits themselves. Under these circumstances the detection digits are corrected and the information is left untouched. The Orthotronic system may take various forms. Through the use of dual columnar-detection channels, for example, it is possible to detect and correct all error patterns which extend across any two adjacent channels. Again, the proper form for this Orthotronic system is dictated by the error patterns of the particular input-output system under consideration.

Future Role of Automatic Correction

The true value or merit rating of an equipment is measured by the net quantity of accurate processing which it produces within a given time period, as well as by the costs incurred in performing this processing. A certain premium will be paid for circuit designs and manufacturing techniques, especially conceived to attain an unusual level of reliability; this is particularly true in the case of certain critical military applications. There is, however, a point beyond which further equipment improvement becomes prohibitively expensive, especially in systems placed in commerical usage. It is at this point where the advantages of automatic correction are most clearly demonstrated. With a relatively small amount of additional equipment cost, the operational merit rating of the equipment can be raised to a degree, the equivalent of which is impossible of achievement by any other means. Appropriate automatic correction techniques, when integrated with equipment of fine reliability, can be expected to usher in a remarkable new era, one in which exceptionally long periods of machine operation can be anticipated with little or no human intervention, and with unprecedented confidence in the accuracy of the final output information.

The System Approach to Reliability

H. D. ROSS

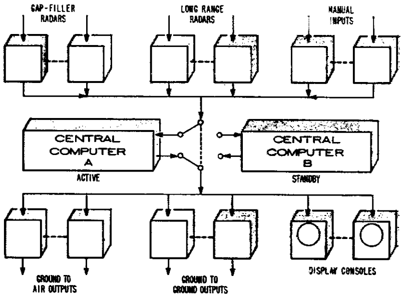


Fig. 1. Simplified AN/FSQ-7 block diagram

THE TITLE of this paper is intended to indicate concern not so much with the individual techniques of reliability, even though a number of these will be mentioned, but rather with the total impact of these various techniques on the reliability of a data processing subsystem, especially techniques above the circuit-component level. Hence, this paper will discuss the approach to reliability taken in the design of one system which is now reaching the operational stage, and to indicate what sort of results that approach is giving. Then, some more recent developments will be reviewed for their implications to a system now being developed. The type of application being considered is one requiring continuous 24-hour-per-day operation in real time, such as a continental air defense or air traffic control system. Many of the same problems are found in other combat operations center or command control systems.

The means used to gain system reliability in the *AN/FSQ-7*, which is a very-large data processing machine used in the continental air defense system, will be reviewed. This machine employs a state-of-the-art which could be considered as 1953-54 and was felt then to be quite an audacious undertaking, in view of its dependence on over 50,000 vacuum tubes per machine, components unreliable by today's standards. In addition to the customary measures of extensive component testing and tube development, use of pluggable electronic packages, built-in test equipment, and specially trained maintenance personnel, several techniques were used which were less common. Circuit optimization was carried to great lengths which caused the evolution of the so-called "schmoo diagram" technique. A substantial part of the machine consists of special equipment for computer-programmed automatic marginal checking. A large amount of effort went into the preparation of an elaborate set of diagnostic programs, through which the routine preventive maintenance was made largely automatic. Some 140 programs are in use here. Finally, about one-fourth of the machine consists of a second central computer, arranged so that either of the two computers can receive data from a common set of channelized input elements, and feed outputs to a common set of display consoles and output elements. In such a system, two computer programs are required; one being the active air defense program and the other a special program for the standby computer which interleaves certain portions of the air defense program with programs that test the condition of the standby computer and carry out marginal checking on it. The whole is arranged for very rapid interchange of the roles of the two computers when the performance of the active computer falls below standard, or when scheduled maintenance or modification is required.

Fig. 1 shows how the system is arranged for duplex operation. At the top are shown multiple channels for each of several representative types of inputs; at the bottom, multiple channels for various types of output data and consoles for data presentation to human operators. Switching is shown to connect the input, output, and display channels to either of two central computers.

Fig. 2 shows a possible schedule of use of the computers and the relative amounts of time spent in active and standby operation, and in routine maintenance in which one computer may be shut down. It is easily seen that the active program alternates between the two computers with each computer typically dividing its day between 12 hours of active program running and 8 hours of standby program, (4 hours are used for marginal checking of the computer and 4 hours for scheduled maintenance during which that computer is unavailable for other uses).

In the schedule shown, the active program is performed 24 hours per day, during 16 of which the standby computer is available for backup. If both the active and standby programs are being run, the system is in duplex operation; if one of the computers is undergoing maintenance, the system is said to be in simplex operation. The schedule shown is not necessarily the one actually used; in practice, demands for computer time for other purposes eat into time available for running the standby program. Doing this tends, of course, to reduce the effectiveness of duplex operation.

Now consider how these measures that are used to increase reliability have worked out in practice using data on the earlier of the *15 AN/FSQ-7*'s which have been installed. It should be pointed out here that the air defense system is in a state of shakedown and evolution, and this results in frequent modifications to the computers as the tactical operation is improved and new weapons are tied in.

Table I shows the estimated dependability of an *AN/FSQ-7* system and the performance of the individual computers upon which the system figures were based.

It is quite a complex matter to charac-

H. D. Ross is with International Business Machines Corporation, Kingston, N. Y.

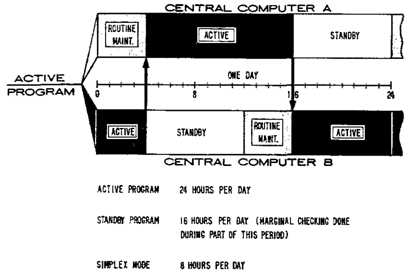

CENTRAL COMPUTER A
ROUTINE MAINT.
ACTIVE
STANDBY
ACTIVE PROGRAM
ONE DAY
0
8
16
24
ACTIVE
STANDBY
ROUTINE MAINT.
ACTIVE
CENTRAL COMPUTER B
ACTIVE PROGRAM 24 HOURS PER DAY
STANDBY PROGRAM 16 HOURS PER DAY (MARGINAL CHECKING DONE DURING PART OF THIS PERIOD)
SIMPLEX MODE 8 HOURS PER DAY

Fig. 2. Simplified AN/FSQ-7 operating schedule

terize adequately the dependability of a *Q-7* because of its duplex nature; it requires 18 different parameters to do this completely. Shown in the lower part of the figure are the four most important of these parameters, for one of the two computers in a *Q-7* and, above, the resulting duplex system performance. The first two simplex figures are based on performance of several machines after final installation and a substantial period of use for system shakedown and program checkout, and exclude malfunctions in certain equipment which is not used in air defense operation. The third point indicates that there were 4 hours of scheduled maintenance per day for each computer in a system. The fourth point refers to the fact that not all of the standby operating time is expected to be assigned to backing up the active computer as mentioned before. Here it is assumed that half the standby time would be devoted to other uses. DCS stands for "Direction Center Standby" proram.

Based on the simplex computer figures shown, the duplex system is expected to have an availability of greater than 99.5 percent. The remaining 0.5 percent stems, for example, from active computer malfunctions, occasionally occurring during scheduled maintenance of the standby computer and would be cut in half if full use were made of standby operation.

These performance figures are based on malfunctions which would cause a failure of air defense. In the computers themselves, most failures are of this sort; in channel equipment, the availability of new data greatly reduces the importance of occasional errors; even solid failures can be tolerated if spare channels or consoles are available.

Fig. 3 shows the mean-time-to-failure history for one of the first systems, based on the average of the mean-times-to-failure of the two computers in the system. It will be noted that over an 18-month period, the mean-time-to-failure rose from 7 hours to something over 20. It should be emphasized that this is not a duplex mean-time-to-failure figure since the two computers in a system were used separately during the period that these data were accumulated.

Fig. 4 shows the behavior of the mean-time-to-restore parameter over the same period of time for the same system. The average time-to-restore levels off at about one-half hour.

Fig. 5 shows the availability, in percent of total time; i.e. 24 hours per day, over a 9-month period immediately preceding the military operational date. During this period, each of the two computers was used separately rather than in the duplex mode described earlier, since the maximum amount of computer time was desired for program debugging, tying in other equipment, etc. It will be noted that in the last few months of operation covered by this chart, availability of approximately 80 percent or close to 20 hours per day was obtained. The grey area indicates the portion of time spent in marginal checking, which can be interleaved with useful operation via the standby program. Here again it must be emphasized that the figures are averages for the two computers in a *Q-7* operating individually. Due to the particular schedule of individual computer use which was shown earlier, care must be used in properly computing the resulting duplex system availability.

Fig. 6 shows the mean-time-to-failure performance of successive machines at the time of acceptance at the site after installation there. Later machines are compared with the first one as reference, and the increase in performance in later machines is apparent.

Table I. Dependability Estimates for the AN/FSQ-7 (8) Duplex Computer System at an Average Site

Duplex Computer System		
1. Availability For Air Defense	99.5%	
2. Average Down Time Per Year	44	Hours
Simplex Computer Section		
1. Mean Time To Failure In Air Defense Operations	25	Hours
2. Mean Time To Restore To Operating Condition	0.5	Hours
3. Scheduled Maintenance Per Day	4	Hours
4. DCS Time Per Day	4	Hours

To consider in more detail the distribution of emphasis on reliability in the *AN/FSQ-7*, dollar-cost is probably a reasonable indicator and this measure was used in preparing the next figure. Considering the cost of an *FSQ-7* system as 100%, and including in this the design and development cost as well as effort devoted to diagnostic programming and training maintenance personnel, the chart shown in Fig. 7 is obtained. The following points are of interest:

1. About 38% of the cost of a system derives from considerations of reliability apart from the intrinsic or built-in reliability of a basic system. This is divided about equally between a standby computer and everything else.

2. Providing the second or standby computer has increased the cost of a system about 28% but has increased system availability from 17 hours to nearly 24 hours per day, an increase of 40%. A further advantage is that during the critical stages of the installation of a site, with heavy demands on computer time for equipment tie-in and program debugging, etc., this increase in cost gives a 100% increase in availability, since the two computers can usually be utilized independently.

3. In spite of a very sizable effort in developing maintenance programs, less

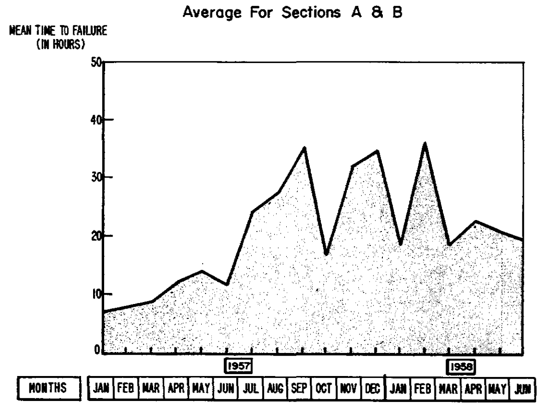
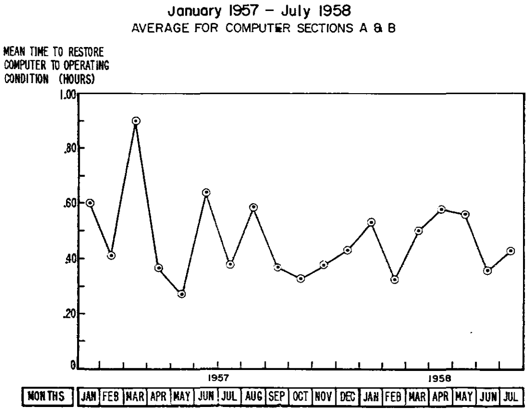


Fig. 3(left). Mean time to failure DC-1

Fig. 4 (right). Average maintainability parameter, DC-1



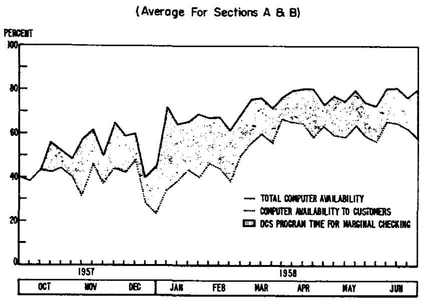


Fig 5. DC-1 simplex computer availability

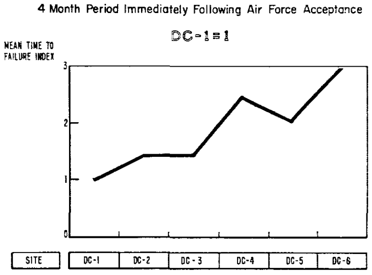


Fig. 6. Dependability comparison for production sites

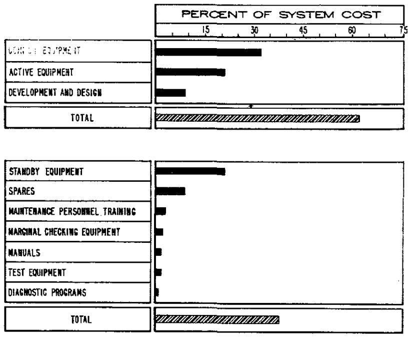


Fig. 7. Per cent of system cost

than 1% of the cost of a system is for these programs.

4. Only about 2% of the cost is due to the very extensive marginal checking system.

5. The cost of other techniques for enhancing reliability such as parity and other built-in checking, is a minute fraction of the total.

Now, having reviewed the reliability performance of the system and degree of emphasis given during design to various means for increasing its reliability, what about the resulting picture?

First, apparently the cost of the marginal checking system is about optimum. The ability to marginal check is a little sparse in common equipment areas but is adequate to catch about 80–90% of all malfunctions to be found by marginal checking. Of course, there are areas like faulty drum heads, short circuits, and other basically mechanical failures which cannot be detected by marginal checking.

Second, the amount of effort spent on diagnostic programs was about right, but with reservations. It is not that adequate programs have always been available but that progress here is a matter of increasing sophistication and cleverness, not weight of effort alone. Recent developments here have been so promising that it must be concluded that the value of such programs was probably underestimated during the original design. In view of the results now being obtained, diagnostic programs deserve increased emphasis in the future.

Third, the degree of automatism built into site test equipment seems excessive, in the light of experience to date. This is mainly because the frequency of testing and repair of the principal electronic assemblies has proven to be lower than initially estimated.

Fourth, in retrospect, initial spares were overestimated; experience so far indicates that stocks can be reduced.

Fifth, from a maintenance standpoint, manuals which conform to military specifications are of relatively little value. The problem here is one of changing the specifications since they were not written with maintenance of digital computers in mind.

Sixth, an insignificant price was paid for error-detection circuitry in the machine (parity checks, mainly). It would have been helpful to have considerably increased this, perhaps by 5 times, to give diagnostic programs more information from which to analyze and localize failures.

This paper will conclude by looking ahead to the design of future systems for 24-hour-per-day real-time use, in the light of the information presented here.

First, duplexing is hard to beat in those applications requiring 99+% availability over a long period of time. Duplexing has also proven very useful because of system evolution and consequent shutdowns for retrofiting. In military systems, the evolving nature of operational requirements will probably insure activity of this kind for a substantial part of the life of the system.

Second, programming techniques are highly effective for trouble diagnosis and should be developed further.

Third, greater use should be made of error-detection equipment together with program techniques that provide recovery from malfunctions. For example, a major advance along these lines has already been proven out in the semiautomatic ground environment (SAGE) system with gratifying results: These techniques:

1. Provide automatic recovery from internal errors 80–90% of the time, assuring correct results with a minimum of lost operational time.

2. Build up a learning table to permit immediate correction of some errors the second time they occur.

3. Nearly eliminate solid errors in parallel transfers as a cause of reduced efficiency.

4. Provide logging of all errors exactly as they occur to aid maintenance.

These advances have proven particularly valuable recently in the use of an *AN/FSQ-7* for controlling flight tests of the Bomarc missile, where any malfunction could delay the count-down or spoil the test.

Fourth, while device development proceeds rapidly and should greatly improve intrinsic reliability of computing equipment, it seems reasonable to expect that the increasing complexity of such equipment, the desire for speed, and the increasing number of basic circuit modules in such a system will require constant attention to techniques for high system reliability in the face of inevitable malfunctions. Experience gained from operation of the *AN/FSQ-7* is of considerable value in pointing the direction future development should take.

Impulse Switching of Ferrites

R. E. McMAHON

MEMORY designs at present reflect either the applications, size, or environment of the memory. Many desired memory characteristics oppose each other in the determination of the memory design and there has been no general solution to the basic problems associated with magnetic storage. Lincoln Laboratory's concern primarily has been to improve the memory speed, and hopefully obtain improved efficiency and design freedom. It has only recently become clear that these three factors, speed, efficiency, and design flexibility, are closely related and in fact lead to generalized memory designs.

Efforts at memory improvements have been concentrated in two areas; the magnetic elements, which is called the elemental approach, and the nonelemental sections or memory proper, called memory methods.

Recent memory methods include techniques such as Load Sharing,[1] Set a Line,[2] Anticoincident Current, and Linear Selection schemes.

The elemental approach includes the work on thin films, Twistors[3] and multiaperture devices.[4]

Memory Improvements

In order to obtain improved speed capabilities, linear selection memory designs have been utilized. There are also other important advantages of linear selection that have accelerated its use. However, it was clear that memory methods alone would not provide the speed improvements desired. An elemental approach that has been named Impulse Switching has provided added memory speeds that are an order of magnitude improved over previous methods. To evaluate the technique, ferrite cores are currently being used as the magnetic element; however, impulse switching is a mode of operation not restricted to ferrites.

Basically, impulse switching utilizes current drives that are controlled in both amplitude and width. By adjusting these parameters it is possible to control how much of the core switches, and in effect, there is a partial switching of the core. In the 50–30-mil size core, for example, there is 10 mils of material available for complete switching, but switching may be restricted to as little as 1 mil of material.

This technique might be described by saying that impulse currents electrically, rather than physically reduce the size of the core to improve the speed.

Impulse Switching Characteristics

The general characteristics obtained by impulse switching will best describe its capabilities and application. The switching constant (S_w) for impulse driven cores shows an effective decrease relative to the S_w obtained with normal drive currents. Fig. 1 shows the S_w curve for both cases.

There is, in practice, a family of S_w curves representing different degrees of partial switching. The switching time t_s is defined as

$$t_s = \frac{S_w}{H_m - H_o}$$

where

H_m = applied field
H_o = threshold field for irreversible domain wall motion and is approximately equal to H_c, the coercive force.

A reduction of S_w as shown in Fig. 1 provides a proportional decrease in the switching time.

There is also an apparent change in coercive force H_c when impulse currents are applied. As the drive current widths are decreased, the current magnitude necessary to initiate switching increases proportionately. This improves the speed still further, since the switching time is inversely proportional to the applied field.

The pulse characteristics are obtained by applying a write current variable in magnitude and width. An additional current called an exciter (opposite to the normal inhibit) is applied coincidently with the write current. The exciter current is set below the coercive field and is slightly wider than the write current. The coincidence of a write and exciter current cause partial switching during this interval. However, the absence of the exciter of course does not allow switching. The write current is first set at a particular value while the width is reduced to small values causing partial switching. There is a decrease in switching time for both the read and write core outputs and a decrease in core output voltage. A new value of write current is chosen and again the current width is reduced to obtain switching time and output voltage data. Curves of this sort are shown in Figs. 2 and 3. Fig. 4 shows the signal-to-noise ratio for these data. Additional data are taken with the read current set at a new value and again varying the write current in magnitude and width. Figs. 2, 3, and 4 are taken for a read current of 1.0 ampere. Additional curves for read currents of 0.5 ampere, 1.5 amperes, and 2 amperes will change switching time, output voltage, and signal-to-noise ratios considerably. In Fig. 4, the signal-to-noise ratio is obtained by applying a program of partial write and exciter currents in a manner similar to that used in normal core testing. It is observed that in many cases improved signal-to-noise ratios (U_{vl}, d_{vz}) are possible for the impulse mode.

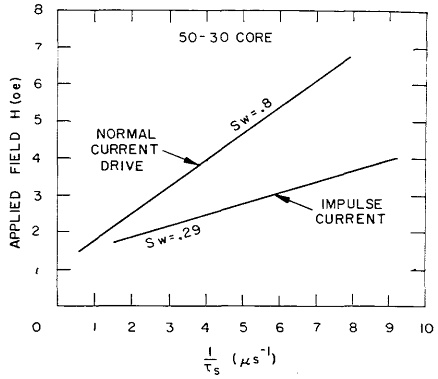


Fig. 1. Applied field versus switching time

In Fig. 3 the reduction of read output to zero for a write current of 0.6 ampere at a width of 0.1 microsecond (μsec) is an example of the apparent increase in coercive field for decreasing current width. For a write current of 0.6 ampere no switching will occur below 0.1-μsec width. At this width, however, the curves indicate switching will occur if the current is raised to 0.8 ampere. The regions below 70-mμsec (millimicroseconds) widths in Figs. 2, 3, and 4 are difficult to measure because of current rise time limitations. It is also difficult to show a composite picture of these curves, but the following generalizations can be made.

1. Variable core voltage outputs are possible at almost any switching time.
2. Switching times as low as 20–50 mμsec are possible with good signal-to-noise ratio.
3. The write interval switching time is approximately equal to the read switching time.

R. E. McMahon was with Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass., and is now with Transistor Applications, Inc., Boston, Mass.

The work reported here was performed at Lincoln Laboratory, a technical center operated by Massachusetts Institute of Technology with the joint support of the United States Army, Navy, and Air Force.

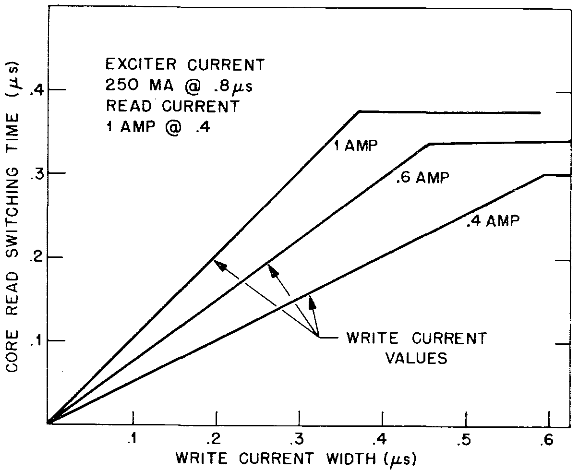
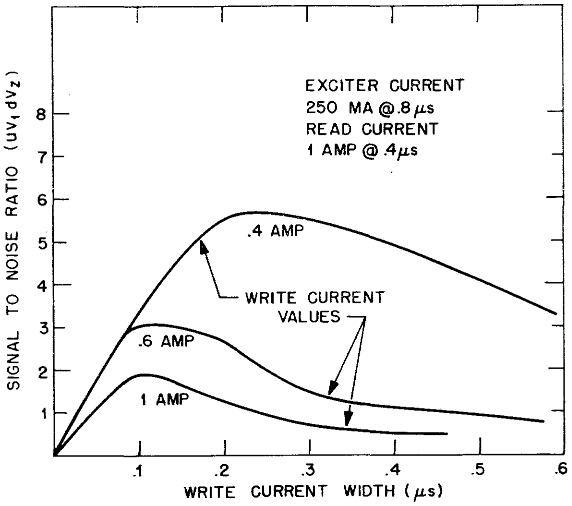


Fig. 2 (left). Read switching time versus write width

Fig. 4 (right). Signal-to-noise ratio versus write current width



General Advantages

These three facts represent the advantages of impulse switching, and in fact, demonstrate that memory speed, efficiency, and design flexibility are controllable for the impulse switching mode. Since the voltage output level of the core is adjustable at any speed setting, the input power required for a particular switching speed and the input power required for a particular voltage output are effectively separated. If, for example, a memory design requires a read switching time of 0.1 µsec, and an output voltage level of 15 mv (millivolts) is satisfactory, then by choosing a proper current and width relation, this may be obtained. The total power required is less than would be the case for obtaining this speed by simple overdrive methods. In overdriving methods, an unnecessary output voltage level of 500 mv would result indicating waste power.

The write interval switching time in impulse switching is approximately equal to the read switching time. The degree of partial switching is controlled during the write time and the function of the read current is simply to read out completely. The fast switching speed during the write time represents an improvement in power efficiency as well as speed. Previously, memory speeds were improved by overdriving during the read time while the write operation being of a coincident nature, allowed no overdrive. The read switching time could be an order of magnitude faster than the write time indicating the large power applied during the read time is wasted if the complete memory cycle time is considered. The average memory power of impulse switching is also low even at cycle times of 0.5 µsec because partial switching results in lower duty cycles. Calculations of peak and average power indicates that impulse switching compared with other elemental or memory methods (at comparable signal levels) requires much less power.

The switching times in the region of 20–50 mµsec, although almost unusable with present memory circuitry, represents possible memory cycle times of 0.1 to 0.2 µsec. Switching times below these values for ferrites represent the crossover point between wall motion and rotation.[5] Nondestructive read out conditions exist in the speed region below 20 mµsec and is worthy of further considerations.

The previous discussion should also suggest added design flexibility of the memory circuitry. Previously, driver limitations imposed design restrictions on the memory size. Using impulse switching, the memory element operation is variable and may be used to eliminate driver design problems. The variable output voltage level can also be adjusted to provide sense amplifier design freedom or overcome system noise level difficulties.

Related Problems

A program has been initiated to provide a better understanding of impulse switching and in particular, the mode of operation and the mechanism that are involved. Recent experiments indicate that the model of partial switching assumed is valid. The impulse switching process has been experimentally observed to initiate at the core center and expand outward. The physical amount of core material involved in the switching process is directly controlled by the current amplitude and width. Experimental observations indicate that irreversible domain wall motion is the major contribution to this mode of operation. The rate of wall motion and consequently the distance moved is proportional to the applied field and the duration of the field. Calculations of the average distance moved by domain walls agrees with experimental evidence of the amount of material switched.[6]

Reversible wall motion occurs if either the current amplitude or width is below a critical value. The applied energy must be sufficient to move the walls beyond certain potential levels or reversible motion results. This accounts for the nonswitching conditions described earlier at current amplitudes and widths below critical values. In addition at very small current widths (below 20 mµsec), no switching will occur until the current amplitude is increased orders of magnitude above the expected or calculated value. There is no wall motion possible below a critical current width. The mode of operation ob-

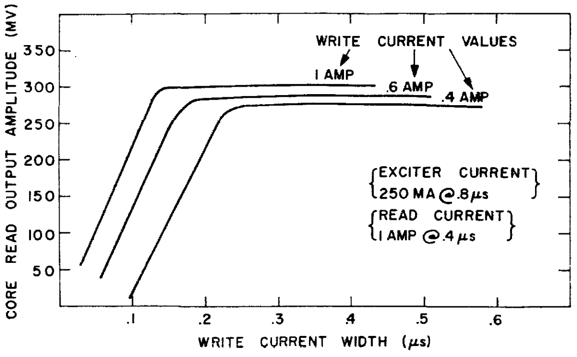


Fig. 3 (left). Core read output voltage versus write width tained for current widths below 20 mμsec and very large current amplitude is one of rotation. The switching control and modes of operation that exists are providing an excellent study method for magnetic phenomenon.

The current width control and its effect on margins has so far not been a problem. There is considerable variation possible in the current width at a particular speed setting, without a decrease in margins.

The heat problem presumed to exist at these speeds has been found to be greatly reduced in the impulse mode of operation. The power associated with the core may be expressed as,

$$P = K\surd fV_{\text{out}}I_m t_s$$

where

$\surd$ = volume of material switched
f = frequency of operation
V_{out} = core output voltage
I_m = applied current
t_s = switching time

For impulse switching, the volume of material and the switching time are very small compared to their values for normal high speed methods. For example, the power for an S_1 50–30 core at one megacycle using standard overdrive methods is 40 milliwatts (mw), while at the same speed, utilizing impulse switching, the power is 0.3 mw even with an output voltage of 50 mv. This low power level results in very small temperature increases.

The memory circuitry required for the impulse switching has presented some problems. To obtain maximum operation, current rise times of 10–40 mμsec are necessary at current levels of 0.5 to 1.0 ampere. At present the circuit of Fig. 5 using p-n-p-n elements is used to provide this current. The circuit is restricted to repetition rates of about 30 kc, but is in use in a sequential 1,000-word 80-bit 0.5 μsec memory under construction. The circuit of Fig. 6 will operate above one megacycle and will be used for random access memory applications.

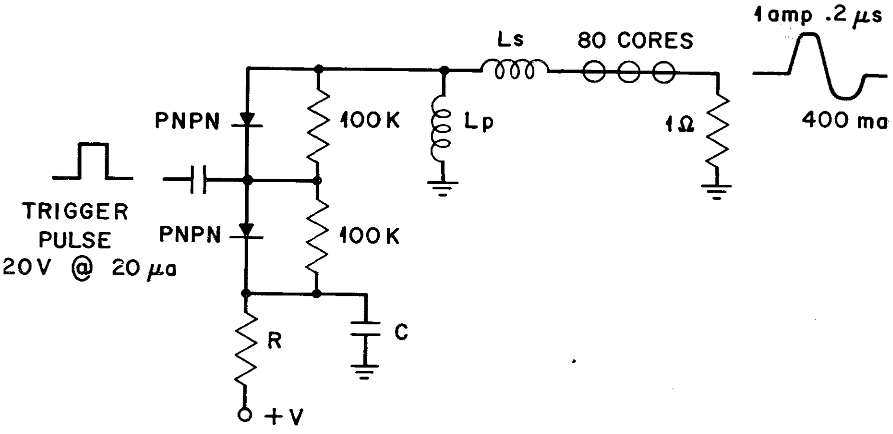


Fig. 5. P-n-p-n sequential driver

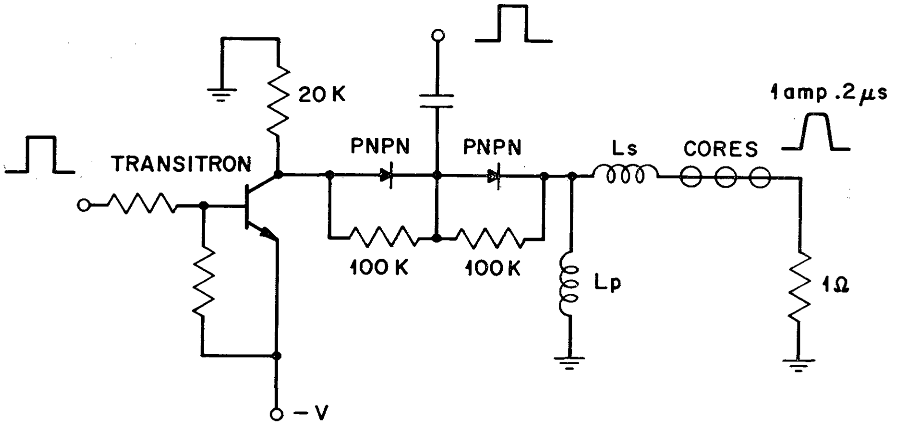


Fig. 6. Random access driver

References

1. A Load Sharing Matrix Switch, **G. Constantine, Jr.** *I.B.M. Journal of Research and Development*, New York, N. Y., Jul. 1958, pp. 204–11.

2. Ferrite Apertured Plate for Random Access Memory, **J. A. Rajchman.** *Proceedings*, Institute of Radio Engineers, New York, N. Y., vol. 45, no. 3, Mar. 1957.

3. The Twistor, **A. H. Bobeck.** *The Bell System Technical Journal*, New York, N. Y., Nov. 1957.

4. A High-Speed Logic System Using Magnetic Elements and Connecting Wire Only, **H. D. Crane.** *Stanford Research Institute Report*, Stanford, Calif.

5. The Utilization of Domain Wall Viscosity in Data-Handling Devices, **Vernon L. Newhouse.** 1957 Western Computer Conference.

6. Nucleation of Domains of Reverse Magnetization and Switching Characteristics of Magnetic Materials, **J. B. Goodenough, N. Menyuk.** *Engineering Note E-532*, Digital Computer Laboratory, Massachusetts Institute of Technology, Lexington, Mass.

Discussion

Lloyd Lambert (Aeronutronic Systems Inc.): If you write continuously, do you tend to walk up the loop?

Mr. McMahon: Continuous partial write currents will fully switch the core; however in a linear selection memory only one write current is possible before the next read out, so that this is not a problem. In addition the improved signal-to-noise ratios indicate that this one write current and any number of successive exciter currents will not disturb the core.

The use of impulse switching in coincident current applications has not been attempted although preliminary work indicates it could be done. Core matrix selection switches have been designed using impulse switching with excellent results.

G. M. Hyde (Lincoln Laboratories): To your knowledge, is anyone working on a program for a computer to determine the optimum value of switching time and amplitude?

Mr. McMahon: No, but we intuitively know the values of currents and widths. We hope to do some work along these lines later.

W. Lawrence, Jr. (International Business Machines Corporation): How critical are pulse widths at 0.1-microsecond switching time?

Mr. McMahon: The width variation is of course the control which allows us to obtain this design flexibility. It can also contribute to a reduction of margins if it varies considerably. We have found that all the circuits so far designed to produce impulse currents display very small variations in current width and no unfavorable reduction in margins exist.

High-Speed High-Capacity Photographic Memory

C. A. LOVELL

THIS PAPER concerns an information store which represents a considerable achievement in the development of photographic storage. Some suggestions will be made for the use of such a store in digital computers. Credit for the achievement belongs to those who developed the store.

Stored program control of digital computers is about 15 years old. Its advantages are so numerous and important that all general-purpose computers today use it to the limits of the capabilities of the storage systems available. There is a strong movement toward greater use of fixed programs of many types to reduce human effort in programming computers. Every computation laboratory has a much-used library of such programs. When in actual use, these programs are stored in high-speed erasable memory because no other type of memory in existence has the required access speed. This has created demands for enormous memory capacities and the demands continue to grow. Although the erasable memory costs are usually a large fraction of the total cost of a computer, the memory capacity is frequently found to be grossly inadequate.

The erasable property is not essential for fixed program storage. High speed readout, high capacity and lower costs are the requirements for a satisfactory store. The subject of this paper is a word-organized permanent information store having these properties. This store is expected to be suitable for program storage and other digital computer uses.

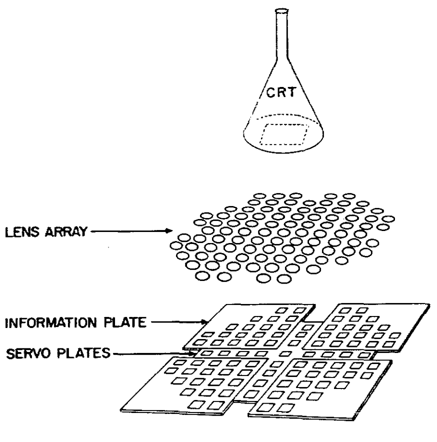


Fig. 1. Flying spot store circuit organization

The Flying Spot Store

The Bell Telephone Laboratories has developed a photographic information store[1] which, with presently known techniques, can be extended to have the characteristics shown in Table I. It is a 10-megabit word-organized permanent information store with random access time of 5 microseconds per word, which will cost on the order of 1/2 cent per bit. The present state of the development of the store will be described here so that the reader can see how much of the predicted characteristics have been achieved to date and how much is promised through extrapolation of the techniques. The stores which will be described as the "present form" of the flying spot store has a larger capacity and is somewhat slower than the store which was designed for control of telephone switching systems. Capabilities of the components are such, however, that the present form of the store can be constructed from them. It was chosen for discussion in the belief that it is better for digital computer use than the smaller store designed for control of telephone switching systems.

Circuit Schematic

Fig. 1 shows a diagram of the store. Except for minor detail, this diagram represents the present and predicted store equally well. The storage medium is a set of photographic plates each of which has a number of separate storage areas which will be called channels. A cathode-ray tube is used as a light source for writing and reading the stored information. A number of lenses, equal to the number of channels, are used respectively to focus the spot of light on the storage areas of the photographic plates. Random-access deflection circuitry to position the cathode ray beam is provided. There is a photodetector to read information stored in each channel. A light intensity stabilizer, input or address registers, and output information registers, not shown on the schematic, complete the list of major functional components of the store.

Table I. Characteristics of Flying Spot Store

Bit Capacity	131,000 76-bit-word or 10^7 bits
Random Access Readout Time	5 μs/word
Writing Time for 10^7 Bits	45 minutes
Time to Fill Memory from a Library of Plates	1 to 2 minutes
Shop Costs	Of the order of 1/2 cent per bit capacity

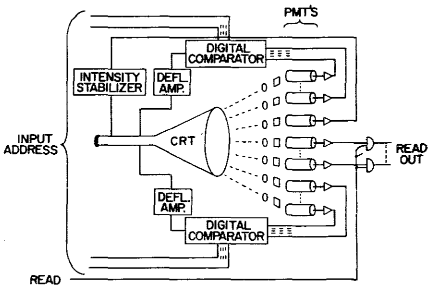


Fig. 2. Flying spot store physical organization

Physical Organization of the Store

Fig. 2 shows the physical arrangements of the cathode-ray tube, lens array, and photographic plate. There are four information plates, each containing 19 separate storage areas which provide a total of 76 storage areas or information channels. The lens for each channel focuses the used part of the cathode-ray tube face on the storage area and sets up a one–to–one reciprocal correspondence between differential elements of these surfaces. The capacity of the store is the product of the number of bits per channel and the number of channels. A word contains one bit from each channel. The address of the word is the binary address in rectangular co-ordinates of the spot of light on the cathode ray face which illuminates the word. Photodetectors used in reading are not shown on the slide. The array of channels seen in the form of a cross in Fig. 3 is used as a part of the means for getting accurate control of the beam deflection.

Servo Control of Beam Deflection

The resolving power of the photographic film is greatly in excess of that of ordinary cathode-ray tube and deflection circuitry. Since the optical system merely maps continuously the cathode-ray tube face on the photographic plates the channel capacity is limited by the resolution obtainable in the cathode-ray tube, lens system and deflection circuitry. The most significant advances made in the development of the flying spot store are

C. A. Lovell is with Bell Telephone Laboratories Whippany, N. J.

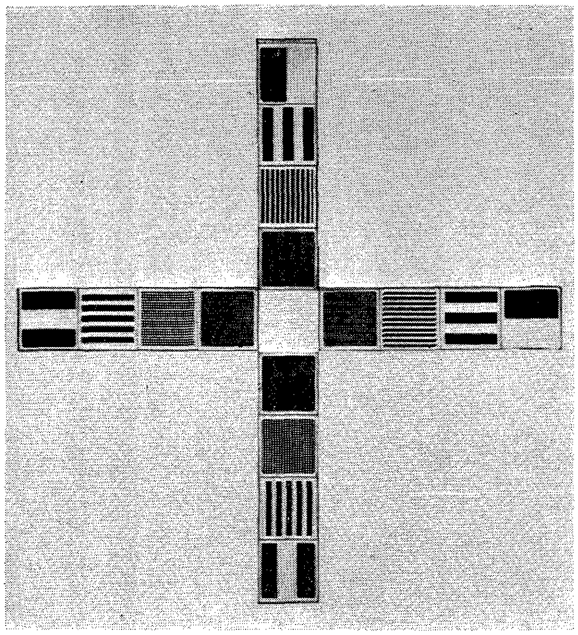


Fig. 3 (left). Servo beam encoder plates

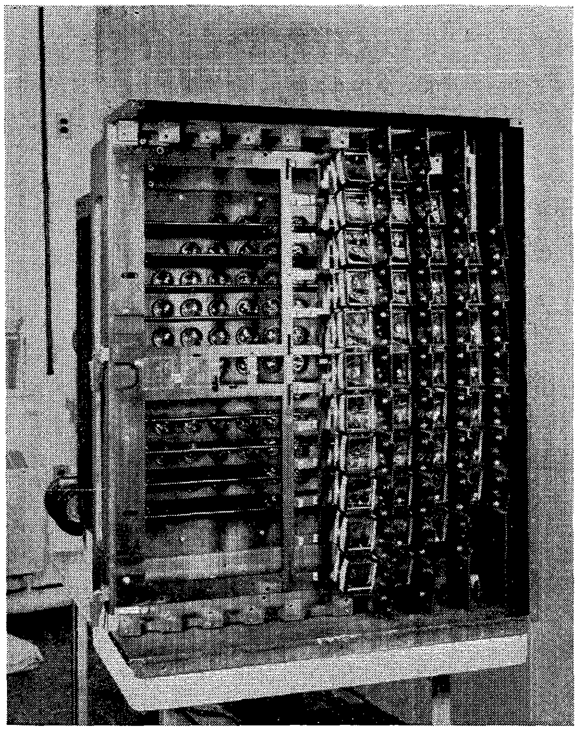


Fig. 4 (right). Optical assembly of flying spot store

in the development of high-speed high-resolution cathode-ray tubes and deflection means. The beam deflection is controlled by a servomechanism, the construction and operation of which will now be described.

A number of channels contain code plates from which can be read at all times the horizontal and vertical positions of the light spot on the cathode-ray tube. There is a servo channel for each binary bit required to define the beam position. Thus 8 channels are required for 256 positions in one co-ordinate and 16 channels will define $(256)^2$ positions on the cathode-ray tube face. In operation the input registers are set to indicate the address of the desired cathode-ray beam position. This information and that of the beam position, read from the servo plates, are compared by digital logic circuitry which gives as an output an error signal suitable for driving the beam to the desired address, where the error signal vanishes and the beam comes to rest. Such a servo exists with a resolution of 256 positions per co-ordinate and an effective bandwidth of about a half megacycle. The servo differs from the one described in the referenced Bell System Technical Journal article in two respects. It has 8 servo channels per co-ordinate instead of one in the beam position encoder. There is a moderately accurate digital-to-analog converter superposed on the servo to speed the action of the beam while moving between widely separated spots. These improvements result in better resolution and faster random access to stored information. The 10-megabit store will require a 9-bit address per co-ordinate.

Light Intensity Stabilizer

There are 17 channels in the servo plate but 16 are sufficient to define $(256)^2$ beam positions. The deflection servo acts on digitally quantized error signals while the beam is more than one cell from the desired address. However, when the beam nears the balance point, the error signal becomes a linear function of the error giving a more conventional analog servo. This servo is in balance when an edge of a code plate intercepts half the light from the cathode-ray tube. Absolute uniformity of light over the cathode-ray tube face is difficult to achieve with a given beam current and more difficult to maintain under nonuniform use conditions. A central channel is furnished to measure the light intensity in every beam position. The intensity is compared to a standard to produce a signal which is fed back to modulate the beam and hold the light intensity constant. In this way it is possible to recognize the half-light balance point of the analog servo. The light reference channel occupies the 17th position in the plate and is of course transparent over its entire area.

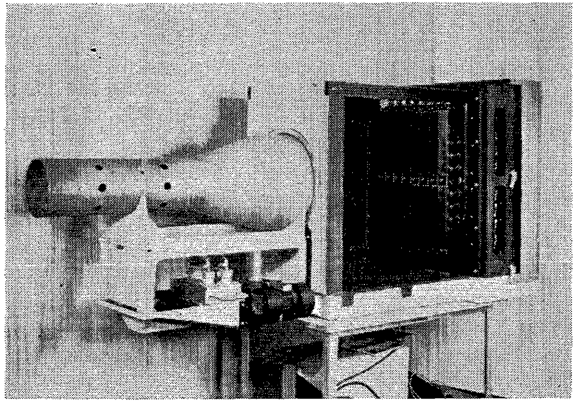


Fig. 5 (left). Side-front 3/4 view of flying spot store optics

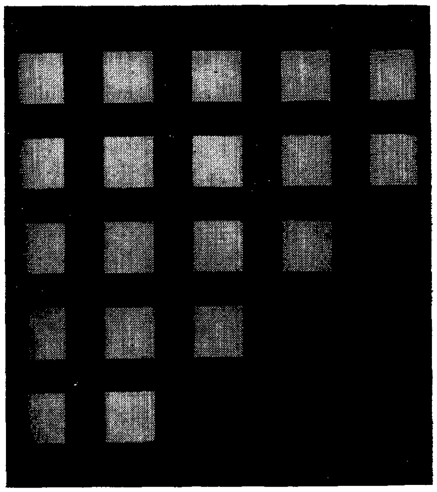


Fig. 6 (right). A 19-channel information plate

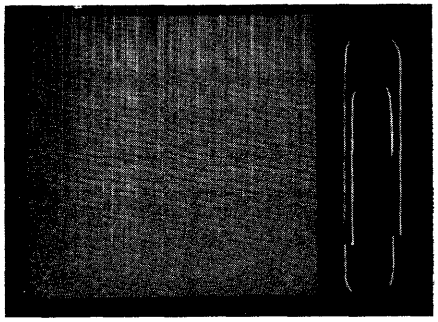


Fig. 7. An information channel

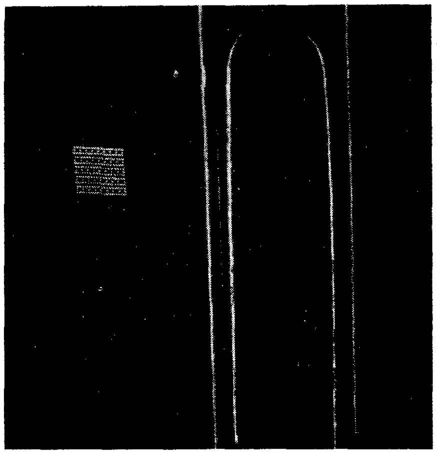


Fig. 8. Enlarged section of a test plate

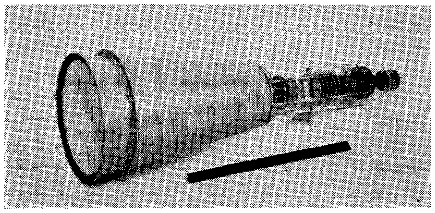


Fig. 9. Cathode-ray tube for flying spot store

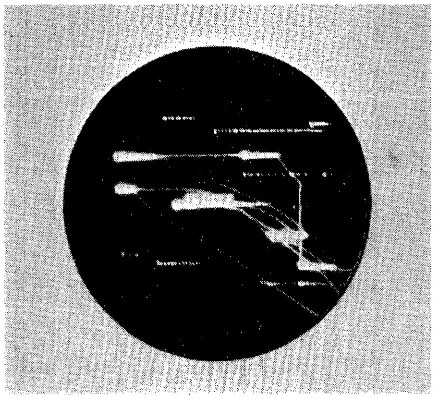


Fig. 10. Monitor scope in parallel with flying spot store tube

The Writing Process

The store is used to write information in its own storage channels. Servo deflection control is used in writing as well as reading. In the writing operation all channels except one are closed by shutters and the information is written into that channel. Since the beam must be left on for servo operation the writing is done by scanning the entire raster, stopping for a timed interval on each of the cells into which ones are to be written. The beam passes over the cells which are to contain zeros so rapidly that it does not expose the cell. The time required to expose a one varies with film speed and can be of the order of 250 microseconds (μsec). Since the information stored in a channel contains a bit from each word, some data processing is required to assemble the information in a form suitable for writing by channels. This is done automatically by a digital computer and stored on magnetic tape for use by a plate exposure unit.

Physical Aspects of Present Store

Fig. 4 shows the optical assembly of the store as seen from the photodetector positions. In the foreground are mounted condensing lenses which are placed between the information plates and the photodetectors. The condenser lenses have been removed from a part of the assembly so that the lenses which focus the light spot on the information plates may be seen. The information plates are not shown in this photograph. They will be mounted on the frame which is seen behind the condenser lenses. The horizontal bars are shutters used in writing. Fig. 5 shows another view of the assembly.

Fig. 6 shows one of the four 19-channel information plates.

Fig. 7 shows a photograph of a 256 × 256-bit channel with a one written in every cell.

Fig. 8 shows an enlarged section of an information plate with a pattern of spots recorded for test purposes. A one is represented by a transparent spot in an opaque background. This is achieved by a reversal which changes the negative resulting from the writing operation to a positive during development and fixing processes. This reversal gives a large improvement in signal-to-noise ratio.

Fig. 9 is a special cathode-ray tube developed in the Laboratories for use in the store. This tube has sufficient resolution over the phosphor face to read and write reliably channels of $(256)^2$ bits each. It has a small high-intensity light spot and a fast phosphor. The light spot is less than 25 mills in diameter over a $6^1/_4$ inch × $6^1/_4$ inch raster. No deflection de-focus corrections are required.

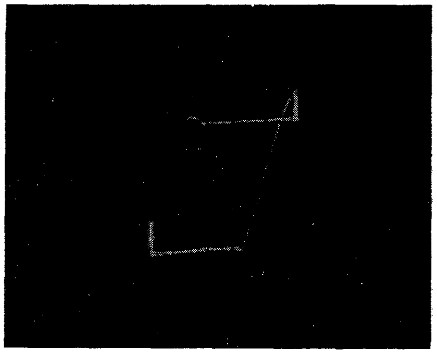


Fig. 11. Close-up of monitor scope showing beam traces

Table II. Comparison of Flying Spot Sto re Characteristics

	Present Store	Predicted Extension
Bit Capacity.......	$(256)^2$ 76-bit-words	131,000 76-bit-words
Random Access Read............	4 μs per word	5 μs per word
Writing Time......	30 min. per 5×10^6 bits	45 min. per 10^7 bits
Time to Fill Memory from Library............	1 to 2 minutes	1 to 2 minutes
Shop Costs.........	Order of 1 cent per bit	Order of 1/2 cent per bit

Fig. 10 shows a monitor scope with plates connected in parallel with those of a flying-spot store tube. The pattern of spots shown on the face represents part of the main program for controlling a telephone switching system. This program is repetitive so that a static pattern is shown in the photograph. Some spots are used more often than others as may be seen from variations in exposure in the photograph. Fig. 11 shows a close-up of the face of the monitor scope. Traces of the paths of the beam in going from spot to spot seen on both slides emphasize the random nature of the access to information stored on the photographic plates.

This completes a brief description of the store. Comparisons of its characteristics and those of the predicted extension are shown in Table II. It is seen that the extension simply assumes that the information per channel can be doubled and that the larger capacity store will be a little slower. Cost per store would not be increased appreciably.

There are under consideration a number of variations of the flying spot store which promise considerable increases in capacity, more convenient writing processes, and other improvements. Some of these advantages will be realized but characteristics of the resulting stores cannot be described at this time. Today these possibilities will be ignored and attention will be centered on the 10-megabit store which has been described.

Digital Computer Applications

Comparison with Existing Stores

The flying spot store differs materially from any store now used in digital computers. It is a fixed information store with a capacity about one third that of an average reel of magnetic tape. However, the fast random access on readout is comparable with that of fast erasable memories. While the write time is considerably slower, this fact is relatively unimportant for fixed information. Such a store should have a great many uses in digital computers. What are some of these uses?

Machine Language Translation

The store described is proposed for use in machine language translations. This subject has received considerable attention during the last 12 years. In 1956, Wall[2] described some of the engineering aspects of the problem. He estimated that 40 megabits of permanent information storage is required and that a machine must translate about 10 words per second to compete, costwise, with human translators. He postulated a photographic store with cathode-ray tube scanning for reading stored information and a repetition rate of 100 kc. Four photographic stores such as the flying spot store described will provide the required capacity.

The search routine is much more efficient than that assumed by Wall. If the storage is alphabetized and every word is found through searching the memory an average of 384 beam positions would be required per word. The time required is 5 μsec per beam position. Hence the average search time per word is less than 2 milliseconds. Thus between 10 and 500 words per second could be translated depending on the number of multiple searches required on the average. In estimating the storage capacity required, Wall assumed it would be necessary to store more information than is logically necessary in order to speed up the search operations. The speed of the store described is such that a better balance might be achieved by reducing the information stored to approximates more closely than logically required. In this case a number of stores less than four may be sufficient. It is apparent that the store described is a practicable permanent store for machine language translation.

Solution of Logical Problems

The flying spot store was developed to solve problems in this class[3] and its usefulness in this field is established. It is improbable that stored program control of a large telephone switching system is the only problem in the class worth solving by this technique. The author suggests as an application the problem of air traffic control in the neighborhood of a large airport such as may be expected to exist 10 or 20 years from now. The problem of control of all the traffic lights in a large city under the changing conditions which exist from hour to hour, day to day, and variations of longer periods appears to be another suitable application. Acute problems, created by special events, can be handled effectively by stored programs in both proposed applications.

These examples of logical problems are characterized by having a large number of more or less acceptable solutions. They are also problems in real time involving large number of variables about which the programmer has certain statistical information. His problem is to write a program which gives solutions that are optimum in some sense. The method of attack on these problems is indicative of the flexibility and power of stored program control. There must be an instruction for every combination of the input variables which requires action. Each particular combination of inputs is caused to generate an address suitable as an input to the flying spot store. At this address is stored either the instruction specifying the action to be taken or a part of the instruction and the address of the next part, and so on until the entire instruction is given. When more than one event requires action at the same time, a main program specifies an order for acting on them sequentially. The real-time aspect of the problem is to achieve a speed such that the maximum delay is inconsequential or tolerable. It is readily seen that an instruction of almost any length and degree of complication can be given as a result of a single input address, so long as sufficient information capacity and speed are available. More will be said about this.

Stored Program Control of Computation

Consider an application of more immediate interest; i.e., use of programs stored in a flying spot store to control computations. An application of this sort which is easy to describe is the storage of the library of programs, routines and subroutines used in computation laboratories to reduce human programming effort. Every computation laboratory has such a library stored usually in a form suitable for their input units so that the desired programs may be read into the erasable memory as needed. The availability of a low-cost fast-access permanent store as an internal computer component for this purpose offers advantages which may be taken in the following ways. The temporary memory can be reserved almost entirely for variable data relating to the problem, thus making it possible to do much larger computations in one step. The low cost per bit of capacity makes it feasible to store permanently an extensive library of routines of all sorts. The routines most frequently used can be assembled on one set of plates, while others can be made available merely by changing plates. Among the routines stored might be rather extensive diagnostic computer test routines which should lead to lower maintenance costs and less down time for the computer.

The entire library of programs which have been assembled at the Murray Hill Laboratory for use with an International Business Machines Corporation *704* computer would occupy only about 15 per cent of the capacity of one set of plates of the flying spot store described.

Many computation laboratories have such extensive libraries that they may exceed the capacity of one set of plates. The 10-megabit capacity provided in this store certainly makes feasible a much more extensive development in fixed computer programs than has been possible heretofore.

So far nothing has been said about possible changes in program structure which might be advantageous if a flying spot store is available. In a manner analogous to the use of a stored program to solve logical problems, a routine of any length or degree of complexity can be started by sending an 18-bit address to such a store. The resulting 76-bit readout makes it possible to use the computer as a 2- or 3-address computer or to operate on variable word length instructions. The nature and speed of access to the information stored allows the use of routines with many branches, re-entrant points and other interrelations. More and more of the programming work is being done by computer manufacturers. With the types of store described the manufacturers could not only furnish efficient and convenient programs with their computers, they could write the programs permanently on information plates ready for insertion in the store. The programs written for such memories could be problem oriented and each customer supplied with a number of programs best suited to his particular class of problems. This in effect makes a general purpose computer into a number of special purpose computers for ease and convenience in solving each customer's particular problems.

Summary

A photographic store has been described and an extension proposed for use in digital computers. Specific applications suggested include: 1. a difficult translation problem; 2. solutions of a class of logical problems; 3. storage of programs of several sorts to reduce human efforts in programming or gain other advantages in digital computer operations.

In closing, the question is asked, "If you had a store such as that described, how would you use it?"

References

1. Fundamental Concepts in the Design of the Flying Spot Store, **C. W. Hoover, R. E. Staehler, R. W. Ketchledge.** *Bell System Technical Journal,* New York, N. Y., Sept. 1958, p. 1161.

2. Some of the Engineering Aspects of the Machine Translation of Languages, **Robert E. Wall, Jr.** *AIEE Transactions,* vol. 75, pt. I, Nov. 1956, pp. 580–85.

3. An Experimental Switching System Using Electronic Techniques, **Amos E. Joel, Jr.** *Bell System Technical Journal,* New York, N. Y., Sept. 1958, p. 1091.

Discussion

G. R. Tiannaca (Rome Air Development Center): Did you have a problem of phosphor burning on the cathode-ray tube? If so, how was this problem overcome?

Mr. Lovell: There was a phosphor burning problem which was solved; in part by moving the tube physically and continuously so that the entire phosphor surface is used more or less uniformly; in part by changing the phosphor and its method of deposition.

C. A. R. Kagan (Western Electric Company): To what extent would you consider this photomemory suitable, technically as well as economically, in the performance of arithmetic computation using logic table look up in memory in lieu of chains of toggles or other forms of serial or parallel adders?

Mr. Lovell: I have not thought much about the use of such a memory in arithmetic computation in the manner described. It certainly could be used in this manner. However, for such simple and well defined operations I would guess that the usual forms of logic circuitry would furnish strong competition. The table look up method might win in cases where such a store is available and its capacity not fully used otherwise. A small amount of consideration has been given to this problem in the Laboratories. I am not familiar with the results and conclusions.

An Experimental Modulation-Demodulation Scheme for High-Speed Data Transmission

E. HOPNER

THE primary motivation of the group in this experimental development was to find a suitably economic system in terms of reliability and equipment simplicity for transmitting binary data from point to point. To this end solutions, primarily with respect to private telephone lines were considered. At the heart of this problem is the right choice of modulation and demodulation schemes, so that before describing the system, one should survey the limitations of various approaches which were faced in the light of boundary conditions of the present telephone network.

On-Off Modulation or Double-Sideband Amplitude Modulation Systems

The on-off modulation schemes are historically the first modulation schemes used for binary information transmission. The two binary states are characterized by the presence or absence of an information carrier frequency (subcarrier). This subcarrier is usually located in the middle of the available frequency band.

The advantage of such schemes is their simplicity. They are, however, sensitive to sudden amplitude variations of the line and are relatively vulnerable to noise. Since both sidebands are transmitted, better frequency spectrum utilization seems possible.

Frequency Modulation Systems

In low-speed data transmission applications (approximately up to 200 bits per second) the Frequency Modulation (FM) systems can tolerate about 10 db (decibels) more white noise and maintain the same systems performance as Amplitude Modulation (AM) double-sideband systems, in respect to speed or bandwidth utilization. The immunity to level variation is one of the crucial factors which paved the way for FM into telegraph carrier systems.[1–3]

The same arguments do not apply entirely for high-speed data transmission, where other factors have to be considered. In low-speed applications, with many carrier cycles per bit of information, the impulse noise does not represent a problem, since the bit is so much longer in duration than the impulse noise disturbances. At higher speeds where there is often only one cycle of the FM subcarrier per bit, the problem of vulnerability to impulse noise and to noise in general becomes severe. The FM capture effect loses effectiveness at higher data transmission speeds.

The FM systems remain insensitive to amplitude variations, even at higher speeds, and that makes them attractive, as long as the noise does not become a paramount problem. Since FM systems require both sidebands, they are like double-sideband AM systems in that they are not efficient in bandwidth utilization.

Vestigial Sideband On-Off Schemes

The vestigial sideband on-off modulation schemes operate with greater effi-

E. Hopner is with International Business Machines Corporation, San Jose, Calif.

The work described in this paper is a result of a joint effort between H. G. Markey and the writer. The equipment was assembled and tested by O. F. Meyer, whose patience during many hours of testing is greatly appreciated.

Dr. E. S. Kuh of the University of California gave substantial support in network synthesis problems. Pulse shaping networks and phase equalization of experimental lines, which made the tests possible, were his contributions.

The contribution of Dr. N. M. Abramson of Stanford University is in theoretical considerations of error probability and interbit interference.

T. C. Kelly and C. M. Melas of the International Business Machines Corporation Research Laboratory of San Jose were helpful in materializing the phase correction and pulse shaping networks.

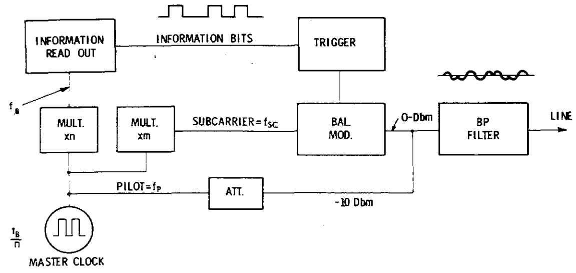


Fig. 1 (left). Transmitter

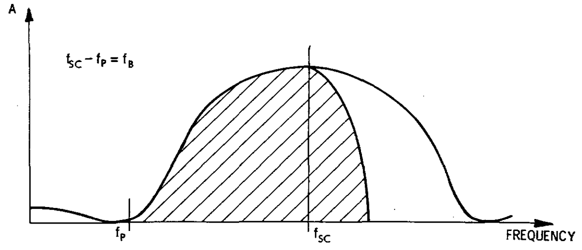


Fig. 2 (right). Signal spectrum

ciency than double-sideband systems. Only a vestige of the upper sideband is usually transmitted, resulting in higher speeds. Its sensitivity to amplitude variations and noise impose stringent and costly requirements on the telephone network in order to achieve the required reliability.

Phase Modulation and Synchronous Detection

The modulation and detection scheme most suited for telephone line data transmission appears to be phase modulation and synchronous detection (sometimes called homodyne detection).[4] Great immunity from noise can be achieved along with insensitivity to level variations. If vestigial sideband transmission is combined with synchronous detection techniques, high speed and reliability can be achieved with simple means. Quadrature component rejection makes the synchronous detection schemes the least sensitive to phase distortion of all the systems yet discussed.

These qualities make the phase modulation schemes very attractive for high-speed data transmission over telephone lines, but some difficult problems needed to be solved before high speed and reliability could be achieved with simple means. The main obstacle to the utilization of phase modulation schemes is the continuous shift of the frequency spectrum commonly found on telephone networks. Because the information-bearing entity of the received signal is its phase, the problem of synchronous detection is obviously difficult in an asynchronous telephone carrier system.

Another difficulty to be mentioned is the phase ambiguity of phase modulation systems, which does not exist in on-off or FM schemes, where the binary value is unambiguously given by the presence or absence of the subcarrier or by a predetermined frequency. The phase ambiguity of the synchronous detector does not represent a serious problem, since it can be eliminated by simple starting logic.

There are two basic philosophies for deriving the detection signal (carrier or subcarrier) in phase modulation systems.

The first is a phase detection system which uses the delayed preceding bit as a phase reference.[5] This type of system is usually simple, if applied on a single-channel basis. The drawback of such schemes is that single errors of the received signal can produce double errors in the system. This makes error detection and error correction more difficult.

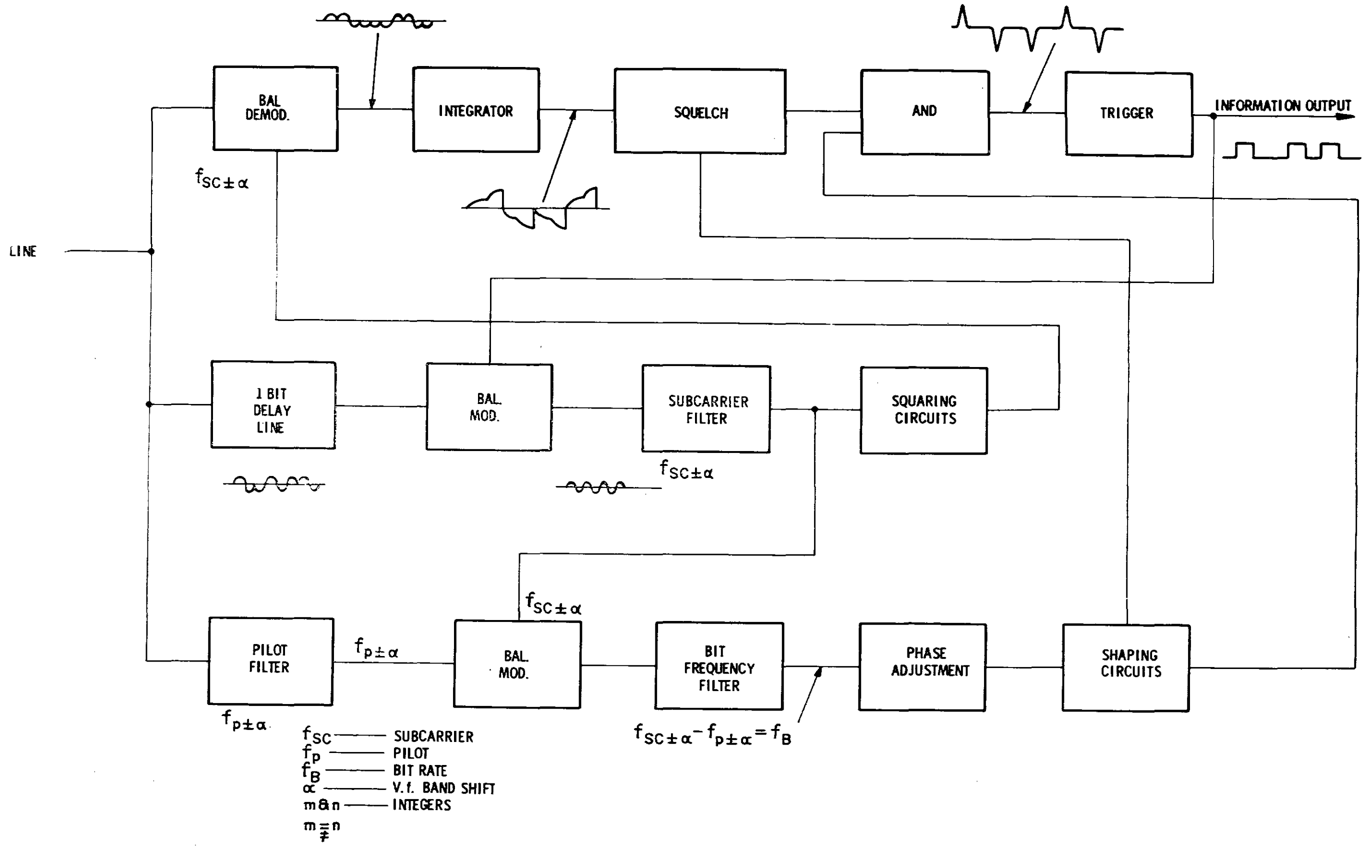


Fig. 3. Receiver

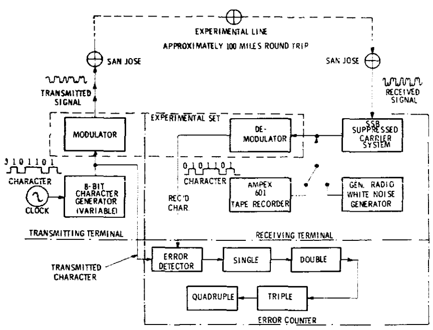


Fig. 4. Experimental setup

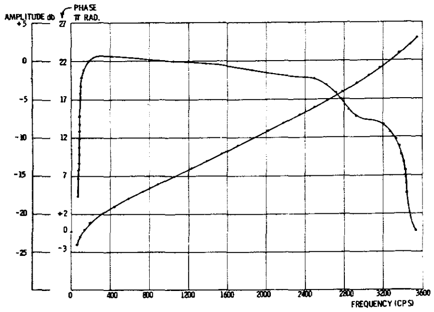


Fig. 5. Frequency and phase characteristics of the L carrier test loop

The second type of detection system involves phase control of the subcarrier oscillator. Such schemes as those used for microwave television links, for example, are usually quite complex.[6] Relatively simple control schemes are successfully applied in conjunction with double-sideband suppressed carrier sytems.[7–9]

An experimental phase modulation system has been built in the International Business Machines Corporation (IBM) Research Laboratories in San Jose that attempted to overcome some basic limitations in high-speed binary data transmission over the present telephone network, which includes microwave radio relay links.

The experimental system is designed to work on existing telephone lines with no special noise reduction treatment. Information was transmitted serially rather than in parallel, to achieve simplicity of equipment and to keep down its cost.

The Experimental System

The basic principles used in the laboratory experimental system were: 1. bit synchronous subcarrier modulation; 2. phase modulation and detection; and 3. clock derivation based upon the difference frequency principle.

As the bit rate approaches the speed of the subcarrier frequency, modulation jitter is introduced in the system, which is one of the speed-limiting factors in the available data transmission systems. Therefore, it is highly advantageous for the bits of information and the subcarrier frequency to be in rigid synchronism. Bit-synchronous subcarrier modulation thus eliminates one of the technological speed barriers outside of the channel itself, with its physical speed limitations of bandwidth, frequency shift, level changes, delay distortion, and impulse noise.

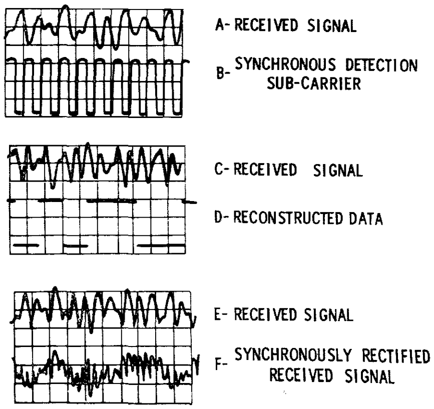


Fig. 6. Received signal (a, c, e) at 1,000 bits per second. The synchronous detection subcarrier (b) was derived from the signal and is in phase with it. The synchronously rectified signal is integrated on a bit-by-bit basis (see Fig. 7a, c, and e) and sampled to generate the reconstructed received binary information (d)

Then, in order to get the greatest possible rejection of the interference using relatively simple circuitry, a signal pulse is sent if either a mark or space bit is transmitted. Basically, the information is sent in 180-degree reversals of the bit synchronous subcarrier. In order to cause an error, the disturbance has to override the signal, or, in other words, it has to be of the same order of magnitude. In addition, the synchronous detector will more or less reject, depending on its phase, any disturbance out of phase with the expected bit of information. Maximum interference rejection is achieved when the disturbance is ± 90 degrees out of phase with the received signal. In order to create an error the disturbance has to be greater than the signal and 180 degrees out of phase with the information bit. The proposed system is basically a phase modulation and detection system with practically achieved ruggedness and disturbance rejection, as was predicted theoretically.

Level variations are basically no problem in the system since the detector has a positive or negative value at the bit-sampling time. A level change affects only the magnitude of the sample and not its polarity, which carries the information.

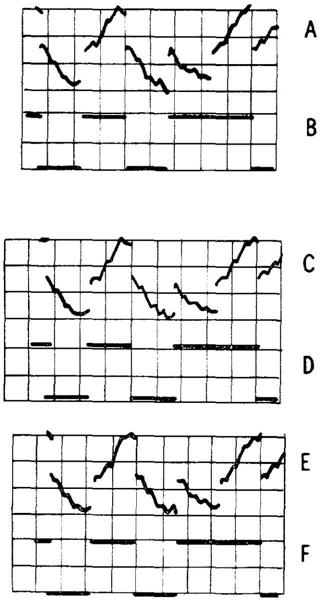


Fig. 7. Integrator output (a, c, and e) and reconstructed data (b, d, and f) at 1,000 bits per second

The advantages of phase modulation and detection systems in interference rejection have already been discussed, but the difficulties of phase detection with an asynchronous carrier arise when thé bit duration approaches that of a single cycle of the subcarrier and the bit is transmitted through a medium which shifts the original spectrum as in the case of single-sideband suppressed-carrier telephone systems.

Even though synchronously transmitted, the phase of the received information changes continuously at the rate of the frequency spectrum shift. It has been a challenging task to build phase detection systems under these circumstances.

Another problem of great importance which is created by shifted carrier operation is the problem of bit synchronism, or the problem of transmission of timing information over asynchronous carrier lines. At the transmitting end, the information-carrying subcarrier is in synchronism with the information bit. In a synchronous carrier system one can derive the bit timing from the subcarrier frequency. In shifted carrier operation this is not the case, since the bit synchronism of the subcarrier is lost.

The experimental system derives the bit-time information or the bit rate from the difference between the information-carrying subcarrier and the noninterfering pilot frequency which is 10 db below the signal. Regardless of the spectrum shift, the difference frequency and its phase stay

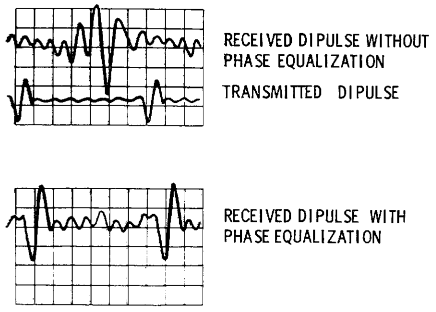


Fig. 8. Transmitted dipulse at 2,400 bits per second (b) received over the test circuit without (a) and with phase equalization (c)

constant and are used to clock the received information.

Along with binary information transmission, a start signal is customary in any data transmission system. A special character can be reserved for that purpose, if the information is coded on a character basis. The problem becomes more complex if the bits of information are transmitted more or less at random. A separate level for the start bit is a solution of the problem which results in increased vulnerability to noise of the information bits and is therefore not considered as satisfactory. A practical approach which does not affect the reliability of the system is one in which the start bit could consist of a sudden reversal of pilot and signal levels for a duration of 5 bits, for example. The level reversal could be followed by 2 bits of information of one polarity and one bit of the opposite. The last bit could be considered as the start bit. This start signal transmission approach is not an original one. It could be an integral part of a system, which, as a whole, represents a new approach to the problem of high-speed data transmission over available telephone lines.

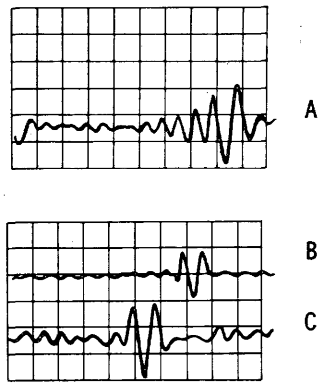


Fig. 9. Transmitted bit at 1,600 bits per second (b) received over the test circuit without (a) and with phase equalization (c)

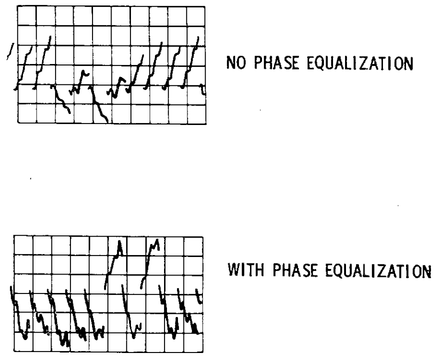


Fig. 10. Integrator output at 1,600 bits per second showing considerable interbit interference without phase equalization (a) and very little interference after the line has been phase equalized (b)

Transmitter

The master clock (see Fig. 1) which provides the clocking signal for the infortion read-out is also used to derive the subcarrier and pilot frequencies. The information-carrying subcarrier and the bits of information are thus in phase synchronism. The pilot and the master clock are of the same frequency, whereas the bit rate and the subcarrier may or may not be the same, depending on the transmission speeds and the frequency distribution of the signal spectrum. For example, the experimental system has a master clock frequency of 800 cycles per second (cps), a bit rate of 1,600 bits per second, and a subcarrier of 2,400 cps. Therefore, the bit rate is twice the clock frequency, and the subcarrier is three times the clock frequency. The pilot and the information signal spectrum are not interfering with each other since the bit spectrum has no energy at the pilot freuency (see Fig. 2) and the synchronous detector rejects the pilot frequency. The pilot frequency is transmitted considerably below the signal level, -10 dbm (power in decibels referred to 1 milliwatt). This is done in order to allow maximum channel loading for the information signal itself.

The frequency multiplication is achieved by conventional analog techniques in deriving the second and third harmonics from the clock signal.

The bit-synchronous modulation occurs at the balanced modulator. Depending on the information bit, the phase of the subcarrier is switched to either 0 degrees or 180 degrees for the duration of the bit.

The band pass filter eliminates the portion of the signal spectrum which cannot be adequately transmitted by the transmission channel. This filter is designed with special care in order to achieve acceptable phase linearity throughout the

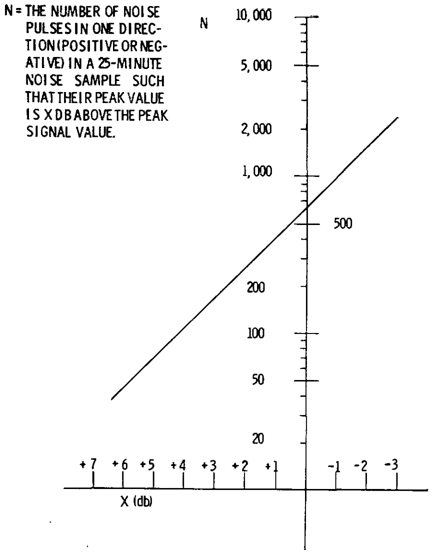


Fig. 11. Number of noise pulses, N, occurring on a telephone line during the 25-minute tape-recorded sample

cut-off region. Thus a good match between the signal and the channel is achieved. Consequently, delay distortion introduced by the transmission channel is minimized, resulting in less interference between bits at the receiving end. Fig. 3 illustrates the receiver.

Receiver

To understand the synchronous detector, assume for the moment that one has at his disposal the shifted subcarrier frequency $f_{sc} + \alpha$. It is the balanced demodulator which compares the phase of the incoming bit to the phase of the shifted subcarrier during the bit time. The received bit is either in phase, with, or in phase opposition to, the shifted subcarrier, depending on its binary value. As a consequence of the synchronous demodulator action, its output consists, during the bit time, of essentially positive or negative pulsations. The integrator integrates the demodulator output during every bit period. At the end of the bit time, the integrator output is squelched, producing a pulse of the bit polarity. The driving signal for the output trigger is generated in the following "and" circuit. The output trigger finally reconstructs the transmitted bits of information.

In order to make the synchronous detector work, two additional pieces of information have to be derived from the received signal: 1. The instantaneous phase reference for the incoming modulated bit of information, which is the shifted subcarrier; and 2. The bit-sampling time,, which is derived from the bit-rate frequency.

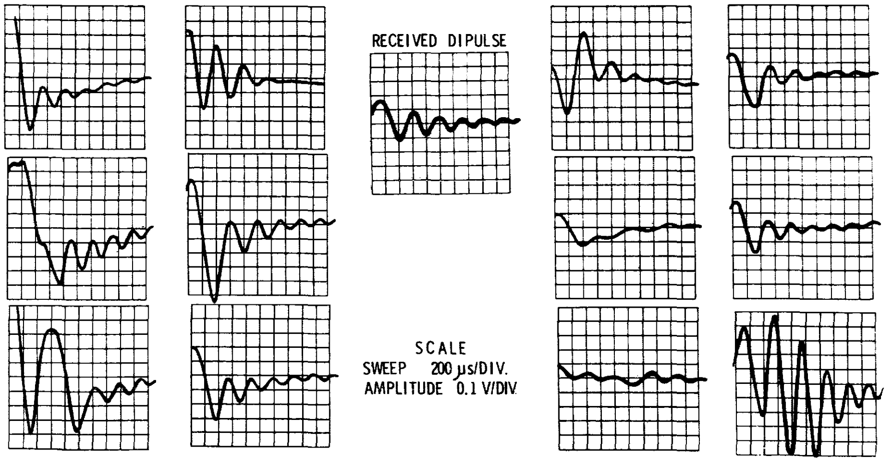


Fig. 12. A received signal without noise (center). All other waveforms are impulse noise without signal; all waveforms are shown at the same scale

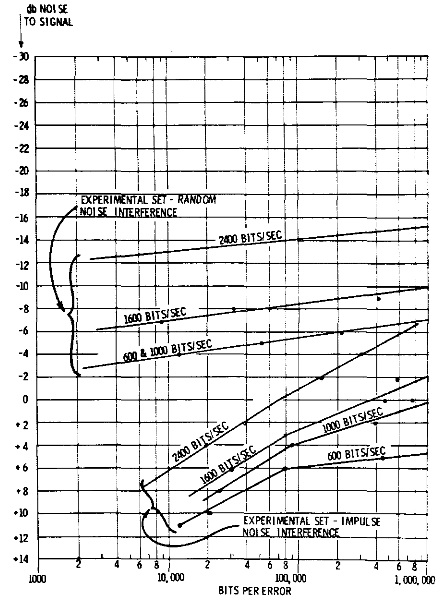


Fig. 13. Experimental data set. Error rates as functions of impulse and random noise levels

These essential detection signals are derived as follows: The received signal consists essentially of 180 degree reversals of the shifted subcarrier, which is therefore suppressed with respect to its steady state condition. In order to reconstruct the suppressed carrier from the information train, a feedback loop is used. The received signal is delayed for one bit time in order to give the synchronous detector a chance to decide if, during the bit time in consideration, the subcarrier should be left alone or shifted 180 degrees. This information is derived from the output signal. The received signal is then left unchanged or reversed in phase, depending upon the received bit of information.

This operation takes place in a balanced modulator, the output of which contains the shifted subcarrier frequency. The subcarrier filter supplies an essentially noise-free subcarrier signal to the system. The described feedback loop is a very stable one, since the loss of eight consecutive bits would not put the system out of synchronism. The reconstructed shifted carrier is clipped and shaped and fed into the synchronous detector and is also used for the derivation of the bit timing signal.

The bit-rate frequency or clock is derived from the shifted subcarrier and the pilot frequency. Since the frequencies of both signals are shifted by the same number of cycles per second in a suppressed carrier system, their difference stays constant and is used for clocking purposes. The pilot is filtered from the received signal by the pilot filter. The difference frequency $f_{sc+\alpha} - f_{p+\alpha} = f_B$ is generated in a balanced modulator. After filtering, the phase of the clock signal is adjusted for delay characteristics of the line so that the squelching and sampling of the integrator output occurs at the end of the bit time. The shaping circuits form the proper pulse shapes for squelching and for information readout.

Experimental Setup

In building the experimental system the emphasis was placed on basic principles rather than on miniaturization or transistorization. The system as described in this paper is a tube version and consists of 16 tubes, comprising the transmitter, receiver, and clock. Later on, the system was partly transistorized because some functions of the system could be performed better and simpler with transistor circuits. A completely transistorized version of the laboratory system consists of 15 transistors, without the clocking portion of it. It is estimated that a complete system including the clock could be built with 25 transistors. The transistor count is given to give the reader a feeling of the simplicity of the modulation and demodulation equipment.

The experimental system was designed to operate at 600, 1,000, 1,600, and 2,400 bits per second in order to evaluate the influence of impulse noise on error rates at different speeds.

Fig. 4 shows an experimental setup using a 100-mile private line circuit having frequency and phase characteristics as shown in Fig. 5.

Satisfactory results were obtained at 600 and 1,000 bits per second with no equalization of the line. Figs. 6 and 7 show the satisfactory operation of the system at 1,000 bits per second without phase equalization. For 1,600 and 2,400 bits per second, phase equalization was necessary for satisfactory operation.

Figs. 8 and 9 show the effect of phase equalization on a single bit at 1,600 and 2,400 bits per second respectively. The influence of phase equalization on system performance by reducing interbit interference is shown in the graphs of Fig. 10.

A 25-minute tape recording was made from a line which is considered to be very noisy for data transmission purposes. The noise peak distribution (impulse noise on the tape) is shown in Fig. 11. Fig. 12 illustrates the seriousness of the problem of binary data transmission with the presence of this noise, as the noise pulses were about the duration of the bit, and sometimes as high as 20 db above the signal.

Fig. 13 presents the results of tests performed under white and impulse noise conditions. Zero db signal-to-noise ratio for white noise means a flat signal-to-noise ratio in the 3 KC band, whereas zero db signal-to-noise ratio for impulse noise represents normal operating conditions under the circumstance prevailing during the recording of the noise tape, which represents a very noisy line condition.

Test Results

In operation under simulated impulse noise conditions with the noise tape, error rates increase drastically as the duration of the signal diminishes and approaches that of impulse noise spikes. At 600 bits per second, the expected error rates are about one in one billion bits; this increases to one in 70,000 at 2,400 bits per second.

At 1,000 bits per second, one error in one million may be expected on very noisy lines. Experience to date suggests that one may expect error rates to be reduced by two orders of magnitude on average lines, as opposed to the very noisy conditions on our tape recording.

Under white noise conditions (which are important for radio circuits), at 1,000 bits per second, with a signal-to-noise ratio of 7 db, the expected bit error rate is one in one million. Experiments indicate that satisfactory operation at 1,000 bits per second may be expected without phase equalization and at 1,600 bits per second with phase equalized lines and no special noise elimination treatment.

At 2,400 bits per second with the present experimental system, satisfactory service on private lines can only be established on lines with a noise peak distribution 6 db lower than shown in Fig. 5. Also, at the present time, such services could only be established on synchronous carrier lines and radio circuits.

The long-distance carrier network by itself does not appear to represent a serious problem for high-speed data transmission with respect to phase distortion. Excessive phase distortion for higher speed services is usually introduced, with loaded cables serving as a link between customer premises and the toll exchanges. In many cases phase correction does not represent an insurmountable problem and can be achieved with simple means and without excessive testing procedures.

References

1. A Frequency Modulation on Telegraph Terminal, **F. H. Cusack, A. E. Michon.** *AIEE Transactions*, pt. I, vol. 66, 1947.

2. A Nationwide FM Telegraph Network, **F. B. Bramhall, L. A. Smith.** *Ibid.*, pt. I, vol. 70, 1951.

3. Transmission of Business Machine Data Over Standard Telegraph Channels, **F. B. Bramhall.** *Ibid.*, September, 1956.

4. Kineplex, A Bandwidth-Efficient Binary Transmission System, **R. E. Mosier, R. G. Clabaugh.** *AIEE Transactions*, pt. I, vol. 76, Jan. 1958 section, pp. 723–28; Discussion by E. D. Sunde, p. 728.

5. A Digital Data Transmission System Using Phase Modulation and Correlation Detection, **F. A. Losee.** Presented at Southwestern Institute of Radio Engineers Conference, San Antonio, Tex., Apr. 1958.

6. The L3 Coaxial System. Television Terminals, **T. W. Rieke, R. S. Graham.** *Bell System Technical Journal*, New York, N. Y., July 1953, pp. 915–42.

7. Phase-Shift Radio Teletype, **J. P. Costas.** *Proceedings*, Institute of Radio Engineers, New York, N. Y., Jan. 1957, pp. 16–20.

8. Synchronous Communications, **J. P. Costas.** *Ibid.*, Dec. 1956, pp. 1713–1717.

9. Synchronous Detection of AM Signals, **J. P. Costas.** *Proceedings*, National Electronics Conference, Chicago, Ill., Oct. 1951, pp. 121–29.

Bibliography

On-Off Modulation or Double-Sideband AM Systems

1. Transmission of Digital Information Over Telephone Circuits, **A. W. Horton, Jr., H. E. Vaughn.** *Bell System Technical Journal*, New York, N. Y., vol. 34, May 1955, pp. 511–28.

2. A Data Transmission Machine, **C. R. Doty, L. A. Tate.** *AIEE Transactions*, pt. I, vol. 75, Nov. 1956, pp. 600–03.

3. Performance Characteristics of Various Carrier Telegraph Methods, **T. A. Jones, K. W. Pfleger.** *Bell System Technical Journal*, New York N. Y., vol. 25, 1956, pp. 483–531.

4. A Terminal for Data Transmission Over Telephone Circuits, **E. B. Ferrell.** *Proceedings*, Western Joint Computer Conference, AIEE Special Publication T-85, Feb. 1956, pp. 31–33.

Frequency Modulation Systems

5. Carrier Systems for Data Transmission, **F. B. Bramhall.** *Western Union Technical Review*, New York, N. Y., Apr. 1957.

6. An FM Digital Subset for Digital Data Transmission Over Telephone Lines, **L. A. Weber.** *AIEE Transactions*, pt. I, vol. 77, Jan. 1959, pp. 867–72.

7. Synchronized Clocks for Data Transmission, **J. O. Edson, M. A. Flavin, A. D. Perry.** *Ibid.*, pp. 832–36.

Vestigial Sideband On-Off Schemes

8. Some Results on the Transmission of Pulses Over Telephone Lines, **J. V. Harrington, P. Rosen, D. A. Spaeth.** *Symposium on Information Networks*, Microwave Research Institute, New York, N. Y., Apr. 1954.

9. Transmission Aspects of Data Transmission Service Using Private Line Voice Telephone Channels, **P. Mertz, D. Mitchell.** *Bell System Technical Journal*, New York, N. Y., Nov. 1957.

Phase Modulation and Synchronous Detection

10. Certain Topics in Telegraph Transmission Theory, **H. Nyquist.** *AIEE Transactions*, vol. 47, no. 2, Apr. 1928, pp. 617–44.

The Impending Revolution in Computer Technology

R. RICE

WEBSTER gives one definition of a revolution as follows: "A total or radical change." The following discussions are concerned with a radical change in the technology which provides components for use in digital data processing systems. At the onset of a revolution it is not clear what the outcome may be. Technological revolutions, as opposed to political revolutions, however, tend to have a common characteristic of inducing major changes regardless of whether or not the revolution is completely successful. The papers presented in this session consider various aspects of what is being anticipated as a technological revolution.

Disjointed papers dealing with individual aspects of the new technology, such as components, circuits, logical design, and programming have been presented in several conferences. But the full impact of the drastic reduction in logical device size and cost is somewhat difficult to grasp if these subjects are considered separately. This session is presented to unite the various aspects of the total picture. The papers are co-ordinated and each author will discuss anticipated changes in the particular field with which he is primarily concerned.

The authors are presenting their own views. In all fairness to them, it should be said that at this point in time, and with as little experience as engineers have had in this new technology, it is extremely difficult to predict exactly the future.

History

To set the stage properly for the revolution, it will be appropriate to review briefly what has been done, what is being done, and what is anticipated will cause a future radical change. In what follows, one is concerned primarily with those properties of computer elements which bear most directly on the subject matter of this session. Everyone is aware that other properties are involved in choosing a particular technology for use in specific systems.

Relay Logic

Fig. 1 illustrates a relay as a basic logical device. Logic using relays essentially consists of wire properly utilized to make contacts, interconnections, and coils. The active elements, the relays, are illustrated in the left of the picture and the necessary interconnections are shown

R. Rice is with International Business Machines Corporation, Poughkeepsie, N. Y.

This is the introductory paper in a panel session on "The Impending Revolution in Computer Technology."

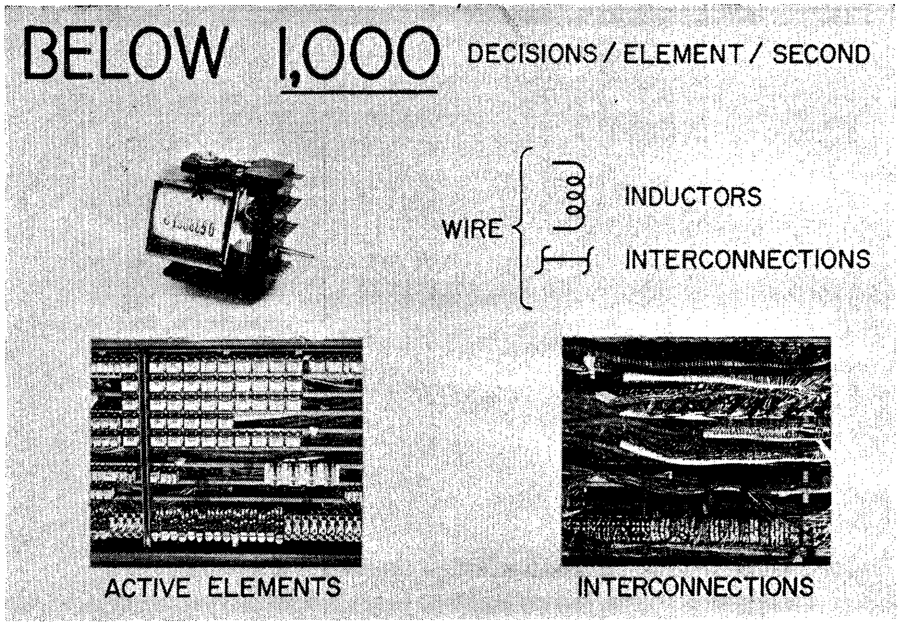


Fig. 1. Relay logic

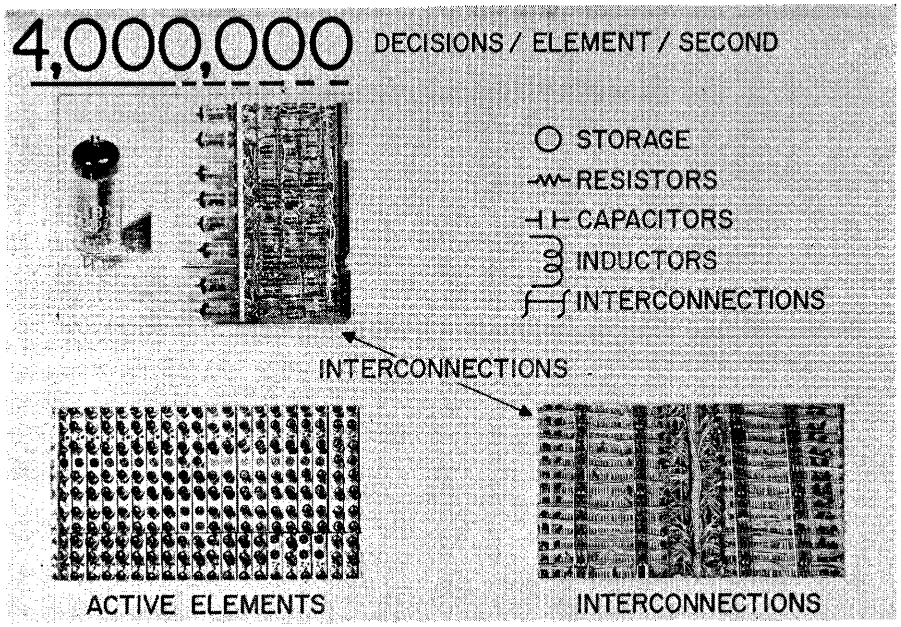


Fig. 2. Tube logic

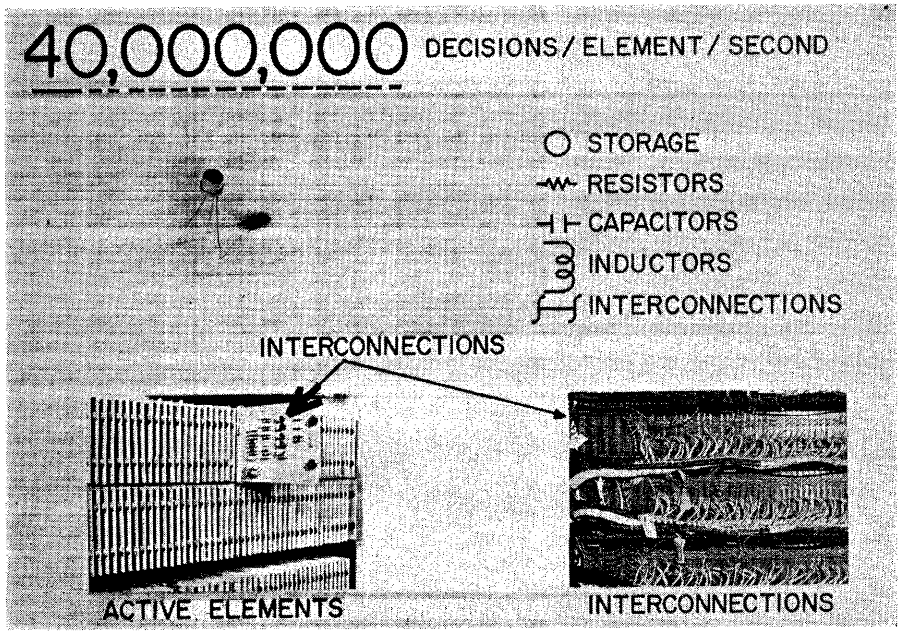


Fig. 3. Transistor logic

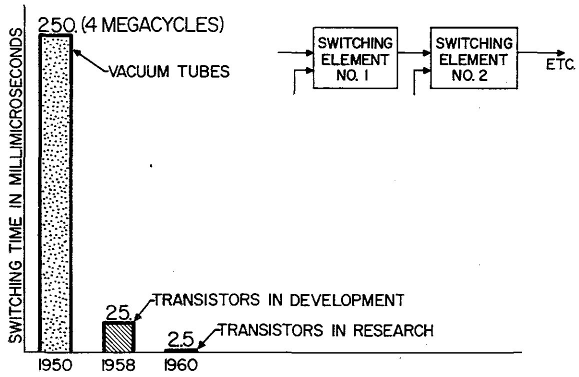


Fig. 4 (upper left). Switching time

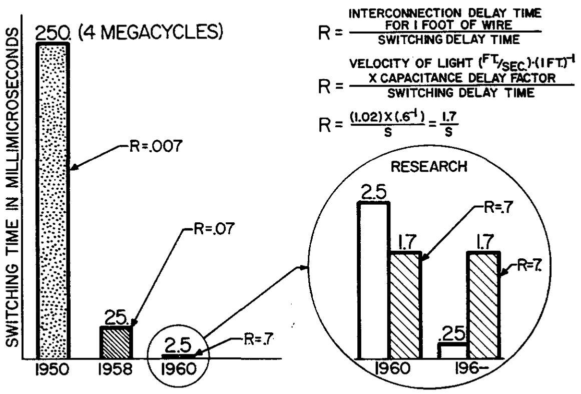


Fig. 5 (upper right). Interconnection problem

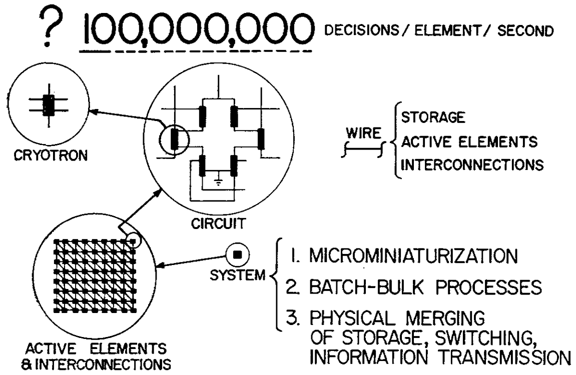

? 100,000,000 DECISIONS / ELEMENT / SECOND
CRYOTRON
CIRCUIT
WIRE
STORAGE
ACTIVE ELEMENTS
INTERCONNECTIONS
SYSTEM
1. MICROMINIATURIZATION
2. BATCH-BULK PROCESSES
3. PHYSICAL MERGING OF STORAGE, SWITCHING, INFORMATION TRANSMISSION
ACTIVE ELEMENTS & INTERCONNECTIONS

Fig. 6 (left). A revolutionary technology

in the back panel picture to the right.

In systems utilizing relay logic, speeds below 1,000 logical decisions per element per second are about the best that can be obtained.

Tube Logic

Fig. 2 illustrates tube logic. With the introduction of filament-type electron tubes, a radical increase in speed was achieved. Speeds above 4,000,000 logical decisions per element per second are possible. Note that the volume occupied and the interconnections involved show only minor improvements over relay systems. Note also that additional types of "passive" components are included in the circuits used. These passive elements increase the total number of interconnections required to perform equivalent logical operations.

Transistor Logic

Improvements in the speed achieved and in the space occupied are obtained with the use of transistors. This type of logic is shown in Fig. 3. Speeds above 40,000,000 logical decisions per element per second are reasonable. As in the case where electron tubes are used, additional passive elements and their interconnections are introduced. No significant reduction in lead lengths for interconnections has occurred despite the introduction of printed- and etched-circuit wiring techniques at about the same time. Advantages of these latter techniques are in areas other than those under consideration here.

Parallel with the change in components from relays to tubes and then to transistors, there has been an increasing demand for more complex logic, more storage, and the greatest possible speed economically obtainable. This is a result of applying machine systems to larger and more complicated problems.

The physical volume occupied by the electronic logic in machine systems has been increasing. This is a result of several factors. First, increased speed has been obtained by using faster circuits which, generally speaking, require more components. Second, speed increases have been obtained by paralleling of operations. This also requires more components. Third, as previously mentioned, larger systems to solve more complex problems increases physical volume. As a result, the average interconnection length has increased.

Switching Time

The bar graph, Fig. 4, illustrates the relative switching times for tube and transistor circuits. Relay switching times are completely off scale and are omitted. The ordinate represents switching time in millimicroseconds, and the abscissa, chronological time, in years, of approximate developments dates. The fastest tube and diode machine with which the author is familiar has a 4-megacycle clock rate producing circuit switching times of 250 millimicroseconds. Transistor machines presently being developed in universities and elsewhere are approximately an order of magnitude faster and switch in about 25 millimicroseconds. Devices which are now in research promise switching speeds greater by another order of magnitude. This yields switching times of 2.5 millimicroseconds.

Present Technology Limits

Fig. 5 includes the relative switching times as previously given in Fig. 4. The portion of the chart representing current research efforts is magnified in the insert. The ratio R of the propagation time of a signal down an interconnection of length one foot to the circuit switching time is introduced. A signal propagating with the velocity of light requires about one millimicrosecond to travel one foot. Propagation in a terminated cable is approximately 1.7 millimicroseconds per foot.

In tube systems the ratio R is 0.007, and may be ignored. For transistor circuits in development R is 0.07, and consequently lead length must be given some consideration. For the switching speed range of transistors currently in research, lead length becomes critical. The enlarged portion of the chart shows the relative importance of delay in interconnections to delay in switching circuits. If one is to achieve another order of magnitude in speed, a drastic reduction in lead length must be obtained.

As previously mentioned, the requirements for more complex logic and greater speed tend to increase the total volume occupied and consequently increases the average interconnection length. One may ask the following question. If volume requirements are greater and the speed desired is greater, how may these demands be met when one is faced with the interconnection problem? An adequate answer to this question is, perhaps, what determines a revolution in computer technology.

A Revolutionary Technology

Fig. 6 illustrates and characterizes what is believed will be a revolutionary technology. Cryogenic elements have been chosen for purposes of presentation, but several different types of devices under consideration by research groups would have served equally well. The small squares in the illustration represent logical elements. For easy visualization all the elements shown in Fig. 6 have been magnified. The expanded picture of one element shows as an example, a trigger circuit with its interconnections. The overall dimensions of this trigger are less than one eighth inch by one eighth inch.

Reasoning from this example, the impending revolution in computer technology will be based on:

1. Microminiaturization. Extremely small, active, passive, and interconnecting elements allowing dense packing to meet logical complexity and speed requirements.

2. Batch-bulk processes. As characterized by Professor Buck, it is anticipated that systems will be manufactured by producing interconnected batches of circuits from bulk raw materials in automatically controlled continuous processes.

3. Physical merging of storage, switching and information transmission. Only the merger of logical and systems design with microminiaturization and batch-bulk processes will produce the full impact of the revolution.

This revolution will have far reaching implications in all phases of the computer field, starting with research and proceeding through system design, manufacturing, including the user. It is the purpose of this meeting to bring attention to the impending changes and to provoke discussion.

Computer Design from the Programmer's Viewpoint

W. F. BAUER

MR. RICE HAS given a stimulating introduction to the subject of the impending revolution in computer technology. This paper will continue this introduction by calling attention to the spectrum of activity in computer design. On the one extreme of this design spectrum is the user and on the other extreme is the engineer-designer. Interpolated between these two positions are the following: problem formulator, programmer, system specifier, system designer, logical designer, and circuit designer. Mr. Rice has introduced the subject from the designer end of the spectrum; this paper will make some further remarks of introduction along the lines of the user's viewpoint and then further develop ideas of what the user expects in the way of computer design.

From the standpoint of the user the principal implication of the impending revolution in computer technology is that information processing will cost less; the advances in computer fabrication and in computer system design will mean that a given amount of information processing will cost less or, alternatively, for a given amount of money more information processing can be purchased. This in turn implies a greater sophistry in computer application. Since a major consideration in the application of computers is programming, attention is focused on that activity. Improved computer technology will imply less cost for programming because of the increases in automatic programming sophistication made possible by the machines of more advanced design. In the United States today, a condition of insufficient manpower to program computers efficiently is rapidly being approached; if adequate strides of progress in programming are not made, more and more people with less qualification will be brought into programming, and the costs will rise rapidly as a consequence. The author believes that the limiting factor in computers today is not the switching times of logical circuits nor the access times of memory units. Rather, the limiting factor is the lack of capability, on the part of machine and on the part of the user, for advanced programming and advanced application.

Attempts have been made to pin down the idea of the productivity of the programmer and to determine how this productivity has changed through the years since 1950. One such "programmer productivity index" would be the ratio of programming cost to computer cost. Another such index would be the ratio of the size of computation staffs to the speed of the computer. Any such definition shows a very great growth in programmer productivity since 1950 and, more specifically, shows a growth factor of at least five, and very possibly as much as 25, depending on the definition used and the assumptions accepted. This increase in programmer productivity is due mainly to the computer design improvements which, from the programmer's standpoint, make possible ease of coding, especially through automatic programming techniques. The importance of the computer design from the programmer's point of view is obviously great in the case of the general purpose computer or data processor. It is only somewhat less true in the case of the stored program computer designed for more specific application.

For present purposes some of the factors of modern computer design will be discussed and what may be considered to be the "ultimate" in computer design from the standpoint of the user will be described.

Design Features

There has been much discussion over the past years on trends in instruction repertoires of stored program computers. Here again the subject can be discussed in terms of a spectrum of design possibilities. On the one extreme is the micro-instruction which is not powerful but is universal in character; it provides a small unit building block for the computer program. On the other extreme is the macro-instruction which is more powerful but also more restricted in application.

As instruction repertoires become more "problem-oriented" they become less universal in character and more special purpose. Computer instructions more problem-oriented in nature can be synthesized through automatic programming—or, commands in the user's terminology are translated to the conventional machine instruction. To illustrate how the more problem-oriented machine instruc-

W. F. Bauer is with Ramo-Wooldridge Division of Thompson Ramo Wooldridge Inc., Los Angeles, Calif.

This paper is part of a panel session on "The Impending Revolution in Computer Technology."

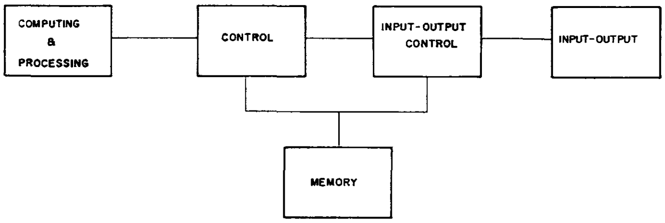


Fig. 1. 1103A-704-705 system schematic

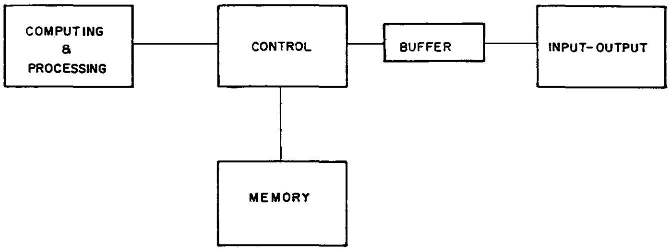


Fig. 2. IBM-709 system schematic

tion repertoire results in its more limited use, consider the very successful FORTRAN algebraic formula translation system device. In nearly all International Business Machines Corporation (IBM) *704* computer installations in the west, less than 30% of the problems use FORTRAN, and very frequently the use is around 10%.

History shows that, in general, special-purpose machines are a poor investment. Consider, for example, the lack of success of computers for the solution of linear algebraic equations. In computer development there has been an obvious trend to universality. Compare, for example, the 16-instruction code of National Bureau of Standards Eastern Automatic Computer (SEAC) with the very large instruction repertoires of the Livermore Atomic Research Computer (LARC) and (STRETCH) computers. However, included in this trend is the march toward including the more powerful instruction, the instruction which is more problem-oriented. To emphasize this, one can recall that in 1951 people who wanted built-in floating point were somewhat hard to find. At that time, many users, and probably more designers, thought that floating-point arithmetic instructions should be programmed rather than built in.

The real solution to the problem of design of instruction repertoires for computers seems to be incisiveness with universality, the set of micro-instructions heavily sprinkled with problem-oriented instructions.

The past few years have seen many important innovations in computer design. Some of these are as follows: the B-box, automatic overflow and underflow handling, the real-time clock available for program interrogation, and the automatic interruption of the computer by asynchronous devices. The last innovation mentioned, "the interrupt feature," bears extra mention and emphasis. The complexity of system design implies that events asynchronous to the computer operation (e.g., card reading, memory interrogation, input from externally timed sources) must be handled by the automatic interruption of the main computing stream; employing the interrupt idea is like asking a staff member to perform a task and then return for another assignment when the first task has been completed. The savings in computer time and programmer time in the employment of the interrupt idea is significant. Happily, this technique, first realized as a standard feature in the Univac Scientific *1103A*, is being included in such new computers as the IBM *7070* and the STRETCH computer.

Another new area arising is that referred to as the associative computer memory or its cousin the indirect address. The associative memory is that first introduced, developed, and used by Newell, Shaw, and Simon[1,2] in their work with a synthesized logic theory machine. Without attempting to describe the technical details of this technique, let it simply be said that it allows the programmer to set up strings of quantities within the computer memory, thereby allowing him to make better use of the computer memory and relieving him of the necessity to specifically outline computer memory requirements on an *a priori* basis. The indirect address technique is developed to a considerable extent on the IBM *709* computer and almost to an ultimate degree on the Gamma *60* computer.[3] This technique essentially allows the programmer the same advantages of the associative memory and, in particular, allows him to deal with names of quantities rather than the quantities themselves.

An idea which may appear important in computer technology from the user's standpoint is the symbolic memory. With this memory, the memory cell itself contains in symbolic form its "address" or, in better terminology, its "name." The programmer, instead of referring to a specific address, refers to a symbolic name in order to summon a quantity from the memory. The assignment of names to specific hardware memory cells is performed by the machine at the beginning of the computation, and a one-to-one correspondence between the symbolic names given and the specific memory cells is effected by the computer hardware. For those who would argue strongly that this is not desirable or feasible, consider the fact that computer programming by means of absolute machine addresses is nearly extinct in the United States today. When a usage becomes this universal, consideration should be given to its inclusion as a hardware item, if this is at all possible.

An interesting phenomenon in computer design is the increasing memory size. Although the memory size of the large-scale computer has increased roughly as the square root of the increased speed, it appears that the advent of concurrency of input-output with computing will change this trend. It also appears that, since large transfers between main memory and auxiliary memory can now be handled so efficiently and economically, there is a limit to, or an optimum for, the size of a high-speed memory. It appears that computer memories in the future will grow with increasing computer speeds but at a rate slower than the square root rate just mentioned.

One of the requirements for larger high-speed memory sizes lies in automatic programming. In the system for the IBM *709*, for example, which allows automatic operation of the computer between and during runs on the computer, a computer control program of 4,000 words in length is kept in the magnetic core at all times. It remains there for the use of the programmer for the handling of untoward situations and for performing the transition from one computer run to the next.

One of the principal faults of the modern computer is that it has characteristics of an arithmetic manipulator and does not have sufficient character along the lines of information handling. The modern large-scale computer is used as much as 40% of the time in program debugging. In this activity the computer is acting almost completely as an information handling device—

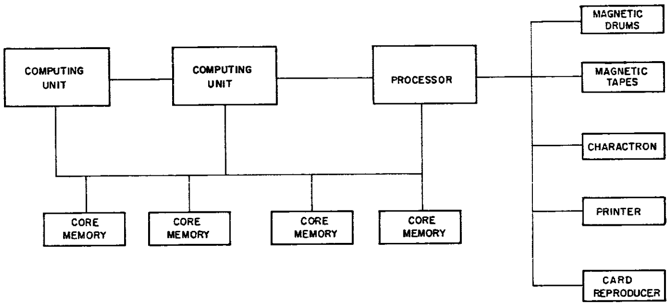


Fig. 3. Univac LARC system

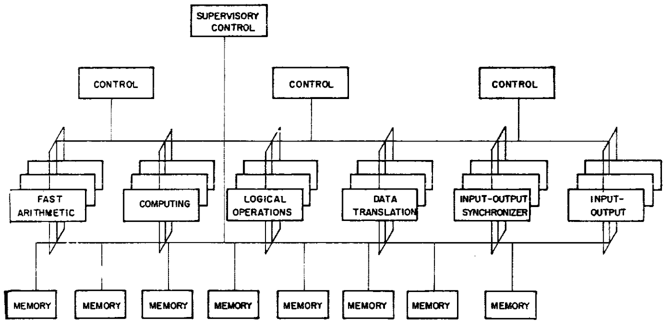


Fig. 4. "Ultradatic" system schematic

as a data processor. If one adds to this percentage the percentage of time the computer performs production operation on problems which are essentially of an information handling or data processing character, one finds that the so-called "scientific computer" in a scientific environment is performing data processing and information handling upwards of 70% of the time. Yet neither instruction repertoire nor system design factors seem to reflect this figure. A striking example is this: A great percentage of time the information processing performed is that of translating alphanumeric input to numerical input, that is, the mapping of the six-bit characters to series of, say, octal digits. Yet almost none of the modern computers allow the programmer ease and facility in the handling of six-bit characters; the translation must always be performed in an indirect way, through synthesis, with extract and shift instructions. The author believes it can be safely stated that no computer in existence today has a system design which reflects the fact that automatic programming is an important and ubiquitous labor saving device.

System Design

System design is the design of the computer with respect to the interrelation of its respective large components such as memory, input-output, control unit, etc. With the advent of the IBM *709* and the Univac *1105* computers, and, more specifically, such computers as the LARC, STRETCH, the Lincoln *TX-2*,[4] and the Gamma *60*, there is a significant revolution in system design taking place. This revolution certainly ranks in importance with the revolution taking place in the design and fabrication of logical elements.

Consider a historical view of design. Four phases of system design can be distinguished: The first of these can be called the "laboratory computer," characterized by such computers as SEAC, National Bureau of Standards Western Automatic Computer (SWAC), and Whirlwind. These computers were strictly sequential in their operation. Input and output interrupted the main computation. Frequently, a megacycle bit rate in the main computation was replaced by a 25 bit-per-second rate for input and output.

The second system design phase could be referred to as the "buffered computer" phase, characterized by the Univac, the IBM *701*, IBM *704*, IBM *705*, and the Univac *1103* computer. With these computers, input-output required only the time necessary for the transfer of information at electronic speeds. With the advent of this phase, input-output was taken out of the domain of mechanical speeds and placed in the domain of electronic speeds. It should be noted that within the computers of this phase there is a wide range of buffering. Fig. 1 shows the simplified block diagram of this computer.

The third phase of system design can be referred to as the "automatic computer" represented by the IBM *709*, the *TX-2* and the Univac Scientific *1105*. Here is seen the first concurrent operation of simple parallelism. With these computers, input-output proceeds independent of the central processing. Fig. 2 shows the simplified schematic of the IBM *709* computer system and, incidentally, shows how the memory is time-shared by the central control unit and the input-output control unit.

The phase which is now being entered could be referred to as the "parallel system" computer phase. This phase is represented by the LARC and the STRETCH computers just appearing. With these computers, there is concurrent operation of large components of the computer. Fig. 3 shows the block diagram of the Univac-LARC computer.

The advent of the "parallel system" computer will have important changes in computer programming. In the parallel machine, in the LARC or the Gamma *60*, for example, where large components of the machine can be time-shared by problems, it may be possible to use the machine to perform automatic programming processing as a fill-in operation during the normal course of the work of the computer. In this case, the use of large amounts of machine time will not inhibit computer automatic programming developments. On the other side of the coin is the fact that exploitation of these machines with parallel operation by automatic programming techniques will be most difficult. (Witness the difficulty of the B-boxes in the development of FORTRAN for the IBM *704*.) For a thorough introduction to the subject of programming of these new style computers, the reader is referred to a recent paper by S. Gill[5].

A failure of the system design of the present computers lies in the fact that the duty cycle of the various components is not sufficiently high. With most computers, for example, the circuits for high-speed multiplication are utilized only a small fraction of the time. In a sense the analog computer system design comes far closer to the ideal. In the analog computer installation, for example, the computer can be spread out and "sized" to the problem. If the average analog computer problem utilizes only a small fraction of the total number of amplifiers and nonlinear units, a high duty cycle can be maintained on all of the equipment. Taking the lead, then, from the analog computer design, it appears that concurrent operations of large parts of the computer should be possible, and that in some way the job for the computer must be made small with respect to the computer. This idea is exploited further in the design of the conjectural computer "Ultradatic" described later.

One of the goals in computer design is

to make the computer act in some sense more like a human. The reader has probably heard of "self-organizing" systems and will undoubtedly hear more of these in the future. As part of the self-organizing aspect, the psychologist or physiologist could refer to the "degree of introspection" of a complicated system in judging how "nearly human" the system is. Certainly the human stops his main stream of activities frequently to reflect on what he is doing from the over-all point of view. This introspection increase can be seen in computer design. In a sense, the interrupt feature mentioned, provides a degree of introspection since one part of the computer system automatically informs another part when it has performed or when it is ready to perform the asynchronous operation. The large-scale computer of the future, by the same token, will periodically examine its own operations and automatically take certain steps to change its function. Ultimately it will change its very nature of character and thereby take on a self-organizing aspect.

"Ultradatic," A Conjectural Computer

A computer system will be described, which is in some sense "ultimate." This is done in the hope that it will stimulate thinking and comment rather than with the thought of foretelling the future. The idea for "Ultradatic" was motivated by the thinking described, concerning the advantages of the analog computer system which can be sized to fit the problem, and by the related idea that it may be possible to make computation more economical by making problems small relative to the computer. It is fostered by the idea that the large computer is more economical to use today than the small computer, or a number of small computers, as long as the total workload is sufficient to keep the large computer busy, or as long as unnecessary expense is not incurred by idle time. The idea further stems from the fact that with increasing frequency one sees computers being operated from remote locations by means of communication over voice channels, or at least extensive input or output of data is being fed to computers from remote locations.

The central idea here is that each large metropolitan area would have one or more of these super computers. The computers would handle a number of problems concurrently. Organizations would have input-output equipment installed on their own premises and would buy time on the computer much the same way that the average household buys power and water from utility companies. In fact, the charges for the services would depend on the type of services rendered. If the problem required extensive use of the higher priced circuits, for example, high-speed multiplying, the rental rate for the time used would be higher than in other cases. The user would be charged for that time used and only that time used, and the accounting procedure would reflect the customers' detailed use.

Fig. 4 shows the schematic of the Ultradatic system. On the figure there can be seen two levels of control, the supervisory control and the normal control unit, and a number of processing units. There is a fast arithmetic unit to perform ultra high-speed arithmetic, and there is a computing unit to provide the lower speed arithmetic with, however, greater possibility for flexibility. There is a logical operations unit which specializes in activities such as sorting and collating. The data-translation processing unit specializes in translation and editing, and the input-output scheduling unit performs detailed control over input and output.

The reader will notice great similarity in Ultradatic to the Gamma *60* computer which was first generally known in this country in the spring of 1958. (The Ultradatic idea was first described in talks given by the author to the San Diego and Rio Grande chapters of the Association for Computing Machinery in late 1957 and early 1958.)

In this large computer system the supervisory control units schedule the multiple problems, monitor the usage of the control units, and initiate the work on the problems by the control units. Each problem has a priority connected with it, and the supervisory control monitors and changes the priorities according to the speed with which problems are being performed. In more advanced configurations the supervisory control would perform some kind of introspection and, perhaps, would effect the change of character of some of the units to other kinds of units depending on the requirements of the over-all problem situation.

The control units interpret macro-instructions, effect the transfer of subroutine variables and parameters to and from the processing units, distribute work among like processing units, and initiate the action of the processing units.

Each processing unit is a small computer in its own right. Each has registers for variables and parameters, each has a limited number of instructions which it can perform peculiar to its particular function, and each processing unit records the time spent on the various problems with which the computer is dealing.

A problem number and a priority number accompany each command sent from the control units to the data processing units. The problem number is essentially an account number used for accounting purposes and the priority number describes the order in which commands are to be performed by the processing unit in case waiting lines develop. Because branch A of a problem may be completed before branch B and the results of branch B are required to continue with branch A, a scheme based on the sequence of operations within a particular problem would be required for interruption or delay of operations.

At first thought one might conclude that the scheduling problem on Ultradatic would be an impossible one; that is, it would be impossible to schedule the various components so that a problem can be done in a short length of time or in a reasonable length of time. It is believed that scheduling a problem would not arise with such a computer. On the basis of the customer's desires a certain initial priority number would be assigned to a problem, and a rough estimate would be made by the computer itself with regard to its workload and as to how much time would be required to perform the problem in the problem multiplexed environment. As the time grew nearer for the completion of the problem, the computer would upgrade the priority if it appeared unlikely that under the current operation the schedule would be met. Thus, the computer by means of the supervisory control would monitor itself automatically and would automatically change the priorities of problems.

A very important characteristic of the Ultradatic computer is that it is modular in construction, and large components can be added or subtracted to make up a configuration which is currently in demand at the particular installation. The processing units, shown in multiples of three in Fig. 4, would be available in any combination and in any practical numbers. If, for example, the records showed that the fast arithmetic unit had a continual backlog, a new fast arithmetic unit would be added to the system configuration. Or, if one of the logical operations units was found to be idle a good fraction of the time, one of these units could be removed from the system. Thus the computer can grow and change to meet the changing requirements of the installation and the high duty cycle will be maintained at all times.

The memory units of Ultradatic warrant certain attention and discussion. This ultimate computer would have a symbolic memory; that is, the symbolic address of the cell assigned arbitrarily by the programmer would be stored within the memory cell, thus allowing the programmer always to refer to that cell in symbolic address notation. Because of the multiple branches of the programs proceeding at times and at speeds unpredictable, a different technique must evolve for the use of the memory in this computer. With present computers in rocket trajectory calculations, for example, only one computer cell is reserved for the height of the rocket. In this computer system a cell would be reserved for the height of the rocket at each quantized level of height. This would be necessary since the height at 9,000 feet might be required in one branch of the computations while the previously computed height, 8,500 feet, was being used simultaneously in a completely different branch of the computations. The symbolic address notation, now considered part of the address hardware, would include indices or subscripts to differentiate the various values of a variable. When the programmer realizes that the height at 8,500 feet is no longer necessary anywhere in the computations, he returns the cell to the available unused memory pool. There the memory cell can be assigned to another problem or to another value in the same problem. Here the associative memory idea would come into play; a string of unused memory cells would be established, with each cell of the unused memory storing the address of still another unused memory cell, so that the string could be added to or subtracted from with convenience as required.

There are certain everyday problems which would be performed by Ultradatic. For example, once or twice per day, the computer would examine all of the programs in its backlog and would make estimates of a gross character as to the length of running time. Once or twice per day the computer would carry out a detailed accounting and compute bills for its multitude of users. Of course, the computer would spend a considerable amount of its time in compiling and translating.

Existence of computers like Ultradatic would not obviate the necessity for small computers, either general or special purpose. However, almost every installation with a small computer today has at least one problem which should be placed on a higher speed machine. Usually the big problem is run on the small machine inefficiently and uneconomically because of the inconveniences of renting large scale computer time.

With respect to special-purpose computers, most of the ideas of Ultradatic are still applicable if, of course, attention is restricted to those of stored program design. Consider the idea of growth or of the inevitable changing character of the application. A computer truly modular in system design can be increased in power to handle the control job for the new polymerization unit or the new cracking tower. Or, as a new formulation of the control process evolves, the computer can be changed to efficiently handle the new computations. Also, ideas of system design to make programming easier will become more important with special-purpose machines. When real-time control computers appear in abundance certainly one of the limiting factors in application will be the preparation of computer programs.

Although Ultradatic will probably never appear at the computer market place, some of its design ideas may bear fruit. If not, service to computer technology advancement is still provided in unerringly pointing the direction not to follow.

References

1. Empirical Explorations of the Logic Theory Machine: A Case Study in Heuristic, **A. Newell, J. C. Shaw, H. A. Simon.** *Proceedings*, Western Joint Computer Conference, Feb. 1957, pp. 218–39.

2. The Logic Theory Machine, **A. Newell, H. A. Simon.** *Transactions*, Professional Group on Information Theory, Institute of Radio Engineers, New York, N. Y., vol. IT-2, no. 3, Sept. 1956, pp. 61–79.

3. France's Gamma 60, **P. Dreyfus.** *Datamation*, May–June 1958.

4. The Lincoln TX-2 Computer Development, **W. A. Clark.** *Proceedings*, Western Joint Computer Conference, Feb. 1957.

5. Parallel Programming, **S. Gill.** *The Computer Journal*, British Computer Society, 1957.

6. Modern Large-Scale Computer System Design, **W. F. Bauer.** *Computers and Automation*, Jan. 1957.

Discussion

I. O. Naughton (Westgate Laboratory): What do you see as an upper bound (a) to the ability of a computer to modify its program based on partial results? (b) to the comprehensiveness of the instructions that can be provided for a programmer?

Dr. Bauer: The upper bound is the limit of humans communicating with a device to tell it how to modify its program or logical structure or to tell it how to tell itself how to modify its own program or logical structure.

Concerning the comprehensiveness of instructions, there is no upper bound as far as the programmer is concerned as long as the instructions are grouped and presented in such a way so that subsets of them can be used. However, there is an upper bound from the viewpoint of the computer seller or the computer buyer. As the comprehensiveness goes up, the costs, of course, go up also, and the point of diminishing returns is soon met. Tomorrow's technology will allow comprehensiveness through or by means of self-organization.

Jerry Mendelson (National Cash Register Company): Ten years ago we built machines with 10-microsecond adders which we embedded in control configurations which reduced the potential addition (computation in general) rate from 100,000 per second to 10,000 per second. We then provided the control system with a command list in which about one command in ten did useful work, the other nine being required to get ready for this work. Result: 1,000 useful commands per second.

For 10 years we have been smug and complacent about having produced man's most inefficient machine, (1% efficiency), and all we have done is shove this inefficient design up the frequency spectrum in order to obtain faster effective output, 1 microsecond adders replacing 10 microsecond adders. We appear to be heading into the millimicrosecond domain with no great effort being applied to changing the internal design characteristics which drag us down by a factor of 100. It's about time we paid some attention to this area, is it not?

Dr. Bauer: It certainly is time. It seems that too much time has been given to the more local problem of circuit speed in comparison to the time given to the global problem of the effect of an executed instruction or an executed set of instructions on solving the problem to which the computer is being applied.

Robert Norman (Sperry Gyroscope Company): A centralized computer facility such as you envisage, with problem solutions available effectively at people's telephones, runs right into a problem in human relations.

Much business, personal and corporate, involves privacy. The availability of large volumes of data on the private and corporate affairs of the inhabitants of a community constitutes a threat to their privacy. It seems this would result in a degeneration of the centralized computer facility to one only used for the solution of an unimportant portion of the total volume of computation in the community.

Dr. Bauer: This would undoubtedly be a problem with a system like Ultradatic. However, difficulties are not insurmountable. For example, the computer itself would monitor in detail all communications between it and a customer to insure that no information was given to unauthorized destinations. Furthermore, proprietary information could be secured by encoding techniques similar to those used in military communications today. Even today we tend to use the telephone as if it were a completely secured and private instrument, and it seems that our "shared" computer of the future would be at least that private.

E. L. Harder (Westinghouse Electric Corporation): What is the present unit cost per computation (you define it), and what reduction do you anticipate as a result of this revolution?

Dr. Bauer: This is a very tough question to answer. Everybody can define the cost per operation of the digital computer itself. However, it is very difficult to define the amount of programming that is done in the application, assign some meaningful cost to it, and assign some meaningful effect to the amount of programming that is done.

As a matter of fact I believe that the emphasis on computer design from the standpoint of speed, size, cost, etc., and the relative lack of emphasis on computer design from the programmer's standpoint stems exactly from the fact that it is easy to put a dollar amount on the computer costs but very difficult to put dollar amounts on programming costs. Much attention is being focused on making the computer economical, but nobody is worried about how much money is being spent on programming.

The "programmer productivity" will probably again increase by many factors during the next 8 years. Weighing against this is the higher salaries which are being paid to programmers in reflection of their contribution to the general technology. Because of these facts I would guess that the cost of computation, all things being considered, certainly will decrease but at a slower rate during the 8 years than it has during the past 8 years.

New Logical and Systems Concepts

R. K. RICHARDS

BEFORE making any attempt to outline the course of future development in computer systems technology, it is well to review briefly the present state of the art and how it arrived at that state. The system designs of the first digital computers such as the Harvard Mark I, the Electronic Numerical Integrator and Calculator (Eniac), and others were largely influenced by ideas presented 100 years earlier by Babbage. In general, the system was comprised of a set of decimal registers, each capable of transmitting numbers in parallel fashion to and from other registers. When a register received a number from another source it was capable of adding that number to any number already contained in that register. Computations were achieved by transmitting numbers back and forth among the various registers with, of course, suitable refinements to obtain the desired results.

Although the concept of long sequences of arithmetic operations with storage of intermediate results was an integral part of the earlier systems design, the stored program concept as it is known today was absent. Further, it is quite likely that even if the stored program concept had been developed at the time of construction of the first computers, it would not have been used. The reason is that the registers used for storing numbers were (and still are) bulky and expensive things, either in the electromechanical or electronic versions. It did not seem feasible to think in terms of more than about 100 such registers in a machine, and this amount of storage is not sufficient to utilize the stored program concept in an effective manner.

The advances in computer systems development occasioned by the introduction of the stored program were achieved only because there was a simultaneous development of relatively low-cost large-capacity digital storage components. Magnetic drums, mercury delay lines, and various forms of electrostatic storage were among the first storage media to be used for this purpose although magnetic cores now play the leading storage role.

No similar progress in reducing the cost of logical components was realized however. Even today, the cost of an elementary "and" function, for example, is generally considered to be many times as costly as a bit of storage in a large-capacity storage array. As a result of this situation, the path of systems development has been in the direction of using the minimum possible number of arithmetic and control elements. The result has been the appearance of the one-operation-at-time general-purpose computer with that one operation being as simple as tolerable to the user of the computer. The binary number system appeared at this point in computer development. Originally the sole purpose (and still the major purpose) of binary numbers was to reduce the number of logical components required to make a computer. The considerable burden placed on the programmer as a result of binary numbers was considered of secondary importance.

With the passage of time the decimal system has been appearing in a larger percentage of new computer designs, and the operations capable of execution by the newer computers are more complex and powerful. However, each step of progress in this direction causes an agonizing increase in the cost of the control portion of the computer and is, therefore, taken only after it is well substantiated that there will be more than an offsetting gain to be obtained through factors such as an increase in computing speed or a decrease in programming effort.

The Course for the Future

Other authors have already discussed the possibility of obtaining low cost logical elements as well as low-cost storage elements through the use of film techniques. In view of the considerable advances in computer systems design that were made possible through the introduction of low cost storage elements, it might be expected that corresponding advances might be made through the introduction of low cost logical elements. It is the purpose of this presentation to examine the factors involved in an effort to determine what these advances might be.

It has been observed that whenever an improvement is made that might be described as being of revolutionary proportions, it has been as a result of solving a problem in one field through the application of a body of knowledge from another field. For example, no great improvements in illumination came as a result of studying flames, tallows, or other aspects of candles. Instead, a knowledge of electricity was used in the development of the electric lamp to replace the candle. Similarly, in the field of transportation only limited gains were realized by improving the care and feeding of horses or by developing better breeds of horses. Instead, the revolutionary advances came from a study of mechanical engines.

In the case of electronic digital computers, if anyone were to propose an ultimately successful concept that re-

R. K. RICHARDS is a Consulting Engineer at Wappinger Falls, N. Y.

This paper is part of a panel session on "The Impending Revolution in Computer Technology."

sulted in a device or machine that is as different from presently known computers as electric lamps are different from candles or as automobiles are different from horses, that person would become famous indeed. This author does not feel on the verge of becoming famous. On the other hand, it does appear that the computer art itself presently contains several widely divergent bodies of knowledge which need only be integrated in a much more cohesive manner to achieve outstanding progress.

At the present time, component development groups are feverishly trying to improve the speed of computers by increasing the operating speed of the components. Logical design groups are trying to simplify the logical design to make computers easier to service, for example. Programmers are working on improved automatic programs to make computers less complicated to use. The quality control engineers are desperately attempting to find and eliminate those factors which seem to prevent all items from being as reliable as the best specimen, and so on, with each individual group working on that particular phase of the problem which falls most naturally in their field of endeavor.

Undoubtedly, the computers that can be obtained through this kind of a piecemeal attack on the over-all problem will eventually show much improvement in comparison with the computers available today. Nevertheless, this approach is like having everyone working on problems analogous to improving the feeding and breeding of horses. It appears that better results can be obtained by backing off and looking at the entire picture.

The first thing to do in looking at the entire picture is to recognize the several fields of technology which are now or can in the future play major roles in computer development. These fields include the following:

1. Physics.
2. Components.
3. Circuits.
4. Logical design.
5. System design.
6. Programming.
7. Manufacturing.

In addition to these seven fields, other items such as maintenance and the economics and financing of computers would be given important consideration in any really comprehensive study of the problem, but it will not be possible to carry the subject that far in the space alloted for this presentation. Also, of course, a thorough knowledge of the problem or range of problems the computer is to solve must be at hand to achieve the objective of a revolutionary advance in the art.

The remainder of this presentation will be concerned with how these major bodies of knowledge can be combined to achieve more effective results than can be obtained by having separate groups of specialists working only on their individual part of the problem. Since the natural sequence of steps involved in making a computer generally proceed from consideration of physics to manufacturing in order listed, this presentation will be made in showing how successive steps might be better integrated with some concluding remarks on achievements that may be realized by tying all aspects of the subject together.

Physics and Components

The need to combine a knowledge of physics and the requirements of computer components has already been demonstrated. Components such as vacuum tubes, transistors, diodes, resistors, capacitors, etc., were invented and developed largely with applications other than computers in mind although, of course, there are many special characteristics of these devices which have been given particular attention as a result of computer needs. On the other hand, the magnetic storage core is a good example where a knowledge of magnetism and magnetic materials was combined with a knowledge of computer storage requirements to produce a new component which offered totally new characteristics and outstanding advantages in comparison with all previously known components.

More recently, computer engineers have ranged quite widely into the field of physics in search of physical phenomena which would be of use to them in the development of new components. The phenomena related to the evaporation of thin magnetic films is one example where important computer applications may be found. In this case the apparent path of progress is in the use of magnetic cores for logical functions with the cores being manufactured by an evaporation process rather than by one of the relatively expensive present methods such as winding a thin strip of magnetic material on a bobbin and subsequently using a core winder to place the turns of the windings one at a time on the toroidal core. Unfortunately, the improvement is largely limited to size, manufacturing, and perhaps power considerations with the basic problems of using cores in logical circuits being made more severe if anything.

A much more striking application of physical principles was introduced by D. A. Buck[1] wherein the phenomenon of the destruction of superconductivity by the application of a magnetic field was used to devise the component now widely known as the cryotron. This component, when suitably improved through further application of physical principles and suitably developed as indicated by the needs of computer logical design, system design, programming, and manufacturing, offers extremely fertile ground for development of computers with very advanced concepts.

The original cryotron was made by winding a wire of one material around a central straight wire of another material. While such a structure offers the possibility of very small size and other advantages in addition to the low-power advantage inherent in the cryotron, only a relatively slow speed of operation was achieved. A further study of superconductor phenomena has revealed that the original geometry is not necessary for producing the magnetic field. Instead, it is sufficient to form the cryotron merely by evaporating two films of superconductive material, one on top of the other with a suitable film of insulation between them.[2] Current gain can be achieved by using suitable widths of film strips, and because of the very low inductance of the structure very high speed operation can be obtained.

Components and Circuits

Many examples could be cited where, with conventional components and circuits, there has been a need for the component and circuit engineers to combine their respective bodies of knowledge. In the case of cryotrons, especially the film cryotrons just mentioned, the situation is unique in that the components and the circuits can be blended together into one unified assembly. Inasmuch as the cryotron structure is comprised of nothing more than two evaporated films of suitably superconductive material, these same films can be extended to form the interconnections among an array of cryotrons. The principal problem is to find suitable patterns of conducting and insulating films so that the number of cross-overs can be minimized and so that the number of masks used for evaporating the patterns can be minimized.

While this problem is simple in principle, it can become quite complex.

Engineers who have had occasion to work out patterns of circuits on printed wiring boards will have a good appreciation of the nature of the situation and may have some ideas about the types of interconnecting arrays that would be most suitable for film interconnections. It has even been suggested that mathematicians with a background in pure mathematics but with a specialization in the field of topology should be able to provide necessary assistance in this phase of the subject. However, the real point to be made is, in this instance, that a thorough knowledge of topology may be necessary but not sufficient. The topologist could perhaps find the best pattern of interconnections for a given circuit, but at the beginning of the design problem the circuits themselves are variables so a knowledge of both subjects must be blended together to find the most desirable combination of circuit and interconnecting array.

Circuits and Logical Design

Numerous papers describing different sets of circuits have appeared in the technical literature. Also, numerous papers have appeared which discuss logical design, that is, logical equations or Boolean algebra. In the circuit papers Boolean algebra is sometimes used to express some of the simpler combinations of logical functions, and in the logical design papers there is often a reduction to circuits of relatively elementary examples of the logical concepts under study. However, very few papers have appeared which report successful attempts to employ Boolean algebra (or any of its variations or extensions) to the problem of designing a computer in its entirety.[3–5] The success that has been obtained in this direction has apparently been only on relatively simple binary computers of the serial synchronous variety. Even with the introduction of the cryotron, which was an outstanding example of combining a knowledge of physics with a knowledge of computer component requirements, there was no apparent attempt to utilize any form of formal logic to develop the interconnecting circuits among an array of cryotrons to perform logical functions. It appears to be the case that the circuits initially presented were obtained by a "mental gyration" process rather than by any orderly procedure that can be described by a logical notation. Although some of these original circuits certainly display a high degree of ingenuity, it is found that even after the circuits are known it is extremely difficult, if not impossible, to devise an algebra which describes the circuits in a useful way. Therefore, as might be expected, it turns out that if more attention is given to an algebra at the start of the circuit design problem, it becomes possible to conserve much mental effort in finding circuits which perform the desired functions. Further, in some cases the circuits found through the use of an algebra are more straightforward and consume fewer cryotrons than those obtained through ingenuity alone. This subject is to be the topic of a paper to be presented at a conference next February.[6]

In other words, there is a close relationship between circuit design and logical design; an intense study of one subject with only a passing attention to the other subject yields a minimum of beneficial results. At the 1958 Western Joint Computer Conference a panel discussion[7] was held where the relative merits of logical equations and block diagrams were discussed at some length. Although several immediate problems in the use of logical equations can be pointed out, it can hardly be denied that a thorough utilization of logical equations is an essential intermediate objective in reaching the final objective of mechanizing the computer design and manufacturing processes. To achieve the intermediate objective, a knowledge of formal logic must be closely knit with a knowledge of circuit characteristics.

Logical Design and System Design

The term "logical design" usually implies the process of reducing a digital system of specified characteristics to an array of "and" circuits, "or" circuits, etc., in such a manner that the requirements of the system are met. Items in the province of the system designer usually include such things as the gross block diagram presentation of the storage unit, arithmetic registers, index registers, input-output buffers, etc. The system designer is also concerned with such items as storage capacity, the instruction list, and the relative speeds of the storage units, arithmetic units, and input-output units.

Again, a study of the literature reveals a minimum of attention to the relationships between logical design and system design. One example of where beneficial results might be obtained by combining the two subjects is in the serial-parallel question. Logical designers seem to have progressed much farther with serial machines because of the relatively simple step-by-step carry generation process. This situation is in contrast to the carry problem in parallel machines where each carry signal occurs at an undetermined point in time and must be represented as a function of all bits in the two numbers being added. On the other hand, system designers frequently specify parallel operation because of the increased speed which can be obtained. It is conceivable that if the computer design problem were to be considered in its entirety, it would be found, for example, that some compromise or arrangement not foreseeable to either group working independently would prove to be optimum. The advantages derived from the ability to use a notation might outweight the disadvantages caused by an increase, if any, in the number of components required or by an increased, if any, in the complexity of the system design.

System Design and Programming

Ever since the first appearance of electronic digital computers, the programmers of these machines have been intensely concerned with reducing both the tedium and the complexity of their job. In general, the line of attack taken by the programmers is the preparation of "automatic programs." For the most part, the programmers have been required to work with machines designed by systems designers, logical designers, and others who had low cost and, therefore, simplicity of the machine as a major objective. Usually, the programmers have not been able to exert any great amount of influence, except in relatively minor details, on the design of the machine.

Upon studying a fairly large sampling of automatic programs[8] it is found that a valuable clue to the course of future systems development is immediately apparent. At the risk of a slight oversimplification it can be said that the major purpose of most automatic programs has been to develop a list of powerful multifunction instructions by using as materials the list of limited function instructions built into the machine. There is no reason whatsoever why a computer could not be built wherein the machine language is made identical to the language the programmers want to use as evidenced by the automatic program. Such a machine would contain a substantially larger number of logical elements in the control portion than is found in present machines, but with low-cost film-type components the increase can be afforded. Not only would the need for an automatic program be eliminated but also a much higher speed computer would result. The higher speed would be obtained because with suitable built-in instructions many functions can be performed directly which with automatic programming would consume several so-called "paper work" instructions which do not achieve any computing but which are necessary to tie the automatic program together.

In a sense, the first step in the direction of recognizing automatic programs has already been taken. Many of the earlier automatic programs had as their major objective (sole objective in some cases) the converting a fixed point computer into a floating point computer. Some of the more recent of the commercially available computers have the floating point feature built into the instruction list so the need for this form of automatic program is eliminated. Conceptually, it is a short step to building in the instructions needed for computing sine, logarithm, and other functions and for handling multiple addresses and other elaborations found in current automatic programs.

The major point intended here is not just the simple one that the built-in instructions could be more complex and powerful; rather it is that the starting point in the system design of a computer should be in what the programmer would like to have, which in turn would be determined by the applications which the computer is to fulfill. This starting point is quite different from the more conventional starting point of what might be called a streamlined general-purpose computer that is capable of doing anything but which was designed with simplicity of the computer rather than efficiency of the entire system, including programming, as the prime consideration.

It may be observed that the concept presented here is almost diametrically opposite to the concept of microprogramming which has been the subject of several recent papers.[9,10] In microprogramming the idea is to provide means in the computer for controlling practically every individual logical function that is performed. Even the operations of add and multiply, which are normally considered basic, are assembled by using a set of several microinstructions. Certainly, most programmers do not wish to be bothered with this amount of detail except perhaps in certain problems of number theory where highly specialized nonarithmetic aspects of the problem force an attention to individual logical operations. For the bulk of the work to be performed by computers, it appears that the proper direction to go in improving over-all system performance is to find standardized sequences of operations analogous to the simulated instructions of automatic programs rather than to subdivide the existing order code in a manner which requires more detail work on the part of the programmer.

The Relationship Between Manufacturing and the Other Aspects of Computer Development

When some of the more "advanced" ideas with regard to future computer development are presented, one occasionally receives the impression the path of progress is somewhat as follows. In the past the development engineers succeeded in the task of making large digital computers work but left to the production engineers the impossible task of making them at low cost, whereas in the future the development engineers will solve cost problems but will leave production engineers with the impossible task of making them work. Of course, no such thought is ever intended, but it is true that in future computer systems development a more intimate attention should be given to production problems.

It has already been suggested that the course of computer progress in the matter of components is likely to be in radically new devices realized through a wider ranging in the study of physical phenomena. If it is true that the tubes and transistors known today are to be replaced eventually by film-type components such as film cores or film cryotrons, it seems reasonable to conclude that little value would be obtained by studying the body of knowledge which is related to the manufacturing of the former components. Instead, a new set of production problems can be expected, but a net saving in cost can be reasonably hoped for even if the new problems are quite severe. The reason for the cost saving is in the very simple structure of the new components.

However, there is much more to manufacturing than the mere fabrication of components and then assembling them into large arrays as needed for digital computers. The preparation of the detailed block diagrams, wiring tables, or other records needed to get started with the assembly of a computer is an important aspect of the manufacturing problem. It happens that some of the techniques currently being developed to determine and record the paths of wires in the computer offer much promise of being highly useful in not only determining and recording paths but also in the actual laying of interconnecting wires in an assembly of film-type components.

When a computer is represented in its entirety by logical equations, it becomes possible to prepare a wiring tabulation automatically by means of a suitable computer program, and this has been done in at least one instance.[4] (A wiring tabulation is a list of interconnecting wires where each wire is designated by means of a coded representation of the terminals or pins to which each of its two ends is connected.) With film-type components it is not sufficient to know the end points of each interconnecting wire. The exact path of each wire must also be specified if, as contemplated, the wires are to be fabricated by depositing suitable strips of conductive material on the same surface on which the film-type components are mounted. Logical equations alone provide no information about the path.

On the other hand, many computer designers are accustomed to working with block diagrams rather than logical equations. These engineers have also been hard at work finding methods of using digital computers to perform their task. The various advantages and disadvantages of detailed block diagrams are largely irrelevant here. It is sufficient to say that a major part of the task of using block diagrams is in the preparation of the diagrams themselves. One company has succeeded in preparing block diagrams by means of a computer. Given certain physical layouts of logical functions (flip-flops, "and" circuits, inverters, etc.) and interconnections among these functions, the computer finds suitable paths on a piece of paper for drawing the lines among the blocks and then actually draws these lines on its output printer to form the block diagram.

While neither the logical equation approach nor the block diagram approach is in itself sufficiently powerful to be used for the automatic manufacture of computers by film techniques, the two approaches together offer lucrative possibilities. The general plan of attack would be to use logical equations in the system and logical design aspects of computer development. By means of a computer already built, the logical interconnections of all wires would then be determined in complete detail. The logical design would then be reduced to a physical layout of functions, again by using a computer if possible. With film components, especially cryotrons, the layout of functions would correspond directly to the layout of components because no supporting components such as resistors, capacitors, or diodes are needed. Only the cryotrons and the interconnections among them are needed to perform all necessary logical functions. After deciding upon the positioning of the components, a computer would be used a third time to determine the interconnections in block diagram fashion. Here the purpose of the block diagram would not be so much for visual reference in assembly and maintenance as it would be to form the film patterns needed to make the actual interconnections.

As further refinement in the manufacturing process, the conventional printed form of the block diagram would be used only as a secondary document and may not even be prepared at all. Instead of a printer, the output device on the computer may be a punching or cutting mechanism suitable for preparing the masks which are in turn used to form the proper film patterns for the cryotrons and the interconnections.

Conclusions

Although further improvements in size, speed, and other properties of computers can be expected, much more revolutionary advances in the computer art are possible through a more complete blending of computer technology from physics of materials to automatic programming. It is conceivable that the computer of the future will be as follows. It will have a system design wherein the language of the automatic program will be executed directly and without the inefficiencies caused by having a machine language as an intermediate medium. The entire logical design of the computer will be accomplished by means of logical equations with computers used extensively in the design but with minimization of components having less importance with the inexpensive film components to be used than is the situation at present where the cost of logical elements is quite high. The logical design will be reduced to circuits by means of computers. Also, the step of producing a physical layout pattern of the components and circuits will be accomplished by means of computers. Then the circuit layouts will be automatically translated into masks for fabricating large numbers of film components and interconnections in one simultaneous process.

References

1. The Cryotron—A Superconductive Computer Component, **D. A. Buck.** *Proceedings*, Institute of Radio Engineers, New York, N. Y., vol. 44, Apr. 1956, pp. 482–93.

2. Superconducting Circuits, **D. R. Young.** *IBM Research Report RC-54*, Poughkeepsie, N. Y., May 1958, pp. 36–38.

3. Logical Design of a Simple General Purpose Computer, **S. P. Frankel.** *Transactions*, Professional Group on Electronic Computers, Institute of Radio Engineers, New York, N. Y., vol. EC-6, Mar. 1957, pp. 5–14.

4. *Panel Discussion by H. Engel* on Logical Design Methods Session, *Proceedings*, 1958 Western Joint Computer Conference, AIEE Special Publication T-107, pp. 182–86.

5. Logical Design Techniques for CG-24, **G. P. Dineen, I. L. Lebow.** See table of contents, this publication.

6. Proposed New Cryotron Geometry and Circuits, **R. K. Richards.** *Digest of Technical Papers*, 1959 Solid State Circuits Conference, Institute of Radio Engineers, pp. 30–31.

7. Panel Discussion on Logical Design Methods Session, *Proceedings*, 1958 Western Joint Computer Conference, AIEE Special Publication T-107, pp. 186–88.

8. *Communications*, Association for Computing Machinery, New York, N. Y., Aug. 1958, p. 8.

9. Micro-Programming, **R. J. Mercer.** *Journal of the Association for Computing Machinery*, New York, N. Y., vol. 4, Apr. 1957, pp. 157–71.

10. Logically Micro-Programmed Computers, **J. V. Blankenbaker.** *Transactions*, Professional Group on Electronic Computers, Institute of Radio Engineers, vol. EC-7, Jun. 1958, pp. 103–09.

Discussion

E. L. Harder (Westinghouse Electric Corporation): What is the fastest complete memory cycle of any machine today, experimental or otherwise?

Dr. Richards: A paper by R. E. McMahon, "Impulse Switching of Magnetic Elements," presented at this conference, quotes about the fastest memory cycle (0.5 microsecond) that I know about. I believe it is possible with cryogenic elements to exceed this speed. However, for various reasons, specific figures have not been announced as yet.

An Approach to Microminiature Printed Systems

D. A. BUCK K. R. SHOULDERS

THE DAY is rapidly drawing near when digital computers will no longer be made by assembling thousands of individually manufactured parts into plug-in assemblies and then completing their interconnection with back-panel wiring. An alternative to this method is one in which an entire computer or a large part of a computer is made in a single process. Vacuum deposition of electrodes onto blocks of pure silicon or germanium and the subsequent diffusion of the electrode material into the block to form junctions is a most promising method. The successful development of this method would allow large numbers of transistors and all of their interconnecting wiring to be made in one operation. Vacuum deposition of magnetic materials and conductors to form coincident-current magnetic-core memory planes is a second promising method that will allow an entire memory to be made in one operation. The vacuum deposition of superconductive switching and memory circuits is a third method that will make possible the printing of an entire computer. The authors feel sure that the most significant milestone in computer component technology will be the announcement by one or more firms, in perhaps 2 years, that all of the technical problems of building a printed system have been solved, and that one of their engineers with his vacuum system can make a digital computer in an hour.

All three methods mentioned, as well as others not mentioned, involve vacuum deposition through a mask. A cleaned glass substrate or a semiconducting surface is placed in a vacuum system, and the air pumped out until the residual pressure is below 10^{-6} atmosphere. A piece of metal near to the substrate is then heated and atoms of that metal evaporate. Some condense onto the substrate, forming a thin film. Between the source of atoms and the substrate a mask is placed to in-

D. A. Buck (deceased) was with Massachusetts Institute of Technology, Cambridge, Mass.

K. R. Shoulders is with Stanford Research Institute, Menlo Park, Calif.

This paper is part of a panel discussion on "The Impending Revolution in Computer Technology."

The work reported here was supported in part by the U. S. Navy (Bureau of Ships) and in part by the U. S. Army (Signal Corps), the U. S. Air Force (Office of Scientific Research and Development Command), and the U. S. Navy (Office of Naval Research).

tercept all of the atoms except those which pass through holes in the mask and condense onto the substrate. The pattern of holes in the mask therefore controls the pattern of deposited atoms in the film. By changing sources and switching masks, one can sequentially deposit conducting and insulating layers to form a circuit. The width of conductors in the circuit is no less than that of the holes in the mask. In practice, the minimum width is 0.001 inch.

The approach to microminiature printed systems that will be described has a line width of 0.1 micron as a goal (1,000 Angstrom units = 0.1 micron = 1/250 of 0.001 inch). If the goal is reached, it will then be feasible to make single-intersection cryotrons or other switching components which fit into a 1-micron square. Allowing 92% of the available area on a substrate for waste space and interconnections, the component density in a finished circuit could be 50 million per square inch per layer. Layers of cryotron circuitry can be isolated from one another by superconductive shielding films deposited between layers of circuitry. Holes in the shielding film would allow magnetic coupling between layers of circuitry. It is conceivable that more than 10,000 layers per inch could be formed, giving a volume density of 5×10^{11} components per cubic inch.

One can hardly justify work aimed at the manufacture of machines like present-day computers on such a small scale. If a present-day computer could be reduced to the size of a cigar box by one of the techniques mentioned in the first two paragraphs, there is little point in then reducing it to the size of a postage stamp. The computer would already be dwarfed by its own terminal equipment and, in the case of cryotron circuitry, its Dewar vessel. Man has ambitious plans for information-handling machines, however, and one can easily visualize a time, a decade from now, when truly vast numbers of components will be needed in a single machine. At present, there is a need for a large number of components to test models of self-organizing systems.[1–5] For an excellent review of neurophysiology see reference 6. Simulations of such systems on digital computers are relatively slow, and the number of neurons in a model of a neural network is presently limited to about 1,000. The slowness of a high-speed computer is a consequence of the step-by-step process by which it carries out a program and the small neural network size is a consequence of the limited amount of high-speed storage. A system of high-speed components that carries out a self-organization study in parallel, with many events occurring simultaneously as in an animal nervous system, would provide a much faster test of some of the rather ambitious models of "machines that think they can learn."[7] By the time the techniques subsequently outlined reach reality, the models of neural networks will undoubtedly have changed. To provide components for such large-scale tests of neural network models is, at any rate, one of the present justifications for attempting to build components on such a small scale and in such large numbers. In addition, however, there are many fine physical experiments that can be performed on a 0.1-micron scale as soon as the building process is developed.

Of many possible methods for making conductors on a 0.1-micron scale, the one chosen involves the selective removal of a thin film. In this respect the method resembles conventional etched-wiring processes. It differs from them in that it is carried out in a vacuum system, and in that a beam of electrons or ions replaces light as a means to control the deposition of a "resist."

Formation of the Thin Film

A selective removal process obviates the need for the selective deposition now being used in vacuum deposition through masks, described. The thin film which will ultimately be cut up into many small areas can be deposited by a technique that yields a film of the desired thickness, uniformity, crystal size, adhesion to the substrate, and other physical properties. Vapor plating,[8] the pyrolytic or chemical reduction of a compound which contains the material to be deposited, is a suitable method for forming many films. It is carried out by first forming a volatile compound of the desired material, and then passing that compound over a heated substrate at which the compound decomposes to form the film. Scrap tantalum, for example, can be heated in the presence of chlorine to form the yellow powdery pentachloride of tantalum. The pentachloride is then heated gently in one part of a vacuum chamber. Elsewhere in the chamber, a substrate is heated to 2,000 degrees Centigrade (C). The substrate soon becomes covered with a film. A small amount of dry hydrogen will allow the plating to occur at a much lower temperature (above 600 degrees C). Thin films of lead, as a second example, have often been formed by the pyrolytic decomposition of tetraethyl lead. Vacuum evaporation (without a mask) from a heated crucible, from a molten tip by electron bombardment, or from a water-cooled copper hearth by electron bombardment are also suitable means by which the thin film can be formed.

The Resist-Forming Process

Specimen contamination due to the free-radical polymerization of hydrocarbon and siloxane vapors under the influence of electron and ion beams has long been a serious source of difficulty to electron microscopists.[9–13] By purposely enhancing the effect, however, one can selectively deposit a "resist" onto an evaporated metal film that will protect areas of the film during the subsequent etching process. For example, tetraethoxysilane vapor, admitted to a high-vacuum system at a pressure of 10^{-4} millimeter of Mercury (mm Hg), allows a siliceous resist to be formed in one minute by an electron beam of approximately one milliampere per square meter in current density. The action of the electrons is to generate free radicals in the monolayer of siloxane which forms over the inside of the vacuum enclosure. A free-radical polymerization process then proceeds through its propagation phase, cross-linking the monolayer into a solid plastic which is effectively removed from equilibrium with its own vapor pressure. A second monolayer then forms and is likewise cross-linked. The nature of the termination phase of the polymerization reaction is not known, and the average molecular weight is also not known. Ennos[11] has studied carbonaceous and siliceous deposits up to 1,700 Angstrom units in thickness, formed in 100 minutes by an electron beam of 10 milliampere per square centimeter density, and he has followed the deposition rate as a function of temperature and beam density. Poole[12] showed the existence of free radicals by removal of a lead mirror, a standard test for free radical hydrocarbons. The rate of formation of the resist appears to be independent of beam voltage. The resist forms nicely in an electron microscope with 100-KV electrons, and it was possible to insulate the anode of a diode with 20-volt electrons. Carr[14] has used the resist formation process to trace trajectories for very low energy electrons.

The Etching Process

While still in the vacuum system, the areas of the thin film on which a resist has not been formed can be removed by vapor etching. Molybdenum films, for example, can be removed at a rate of about

1,000 Angstrom units per minute by heating the film to 300 degrees C and admitting chlorine at 10^{-4} mm Hg. Each metal has a different etchant and etching temperature. Ideally, the etching process will form a volatile compound of the metal so that not only will the film be etched, but the resulting compound will leave the substrate and be quickly pumped away. Some metals have a tenacious oxide which can be removed by a brief exposure to carbon tetrachloride vapor while hot. After removal of the oxide layer, a halide etch can then often be used. A silicaceous resist can withstand a higher etching temperature than a carbonaceous resist.

Insulation Between Layers

A silicaceous resist, when heated, becomes a high-silica glass, and can serve as a layer of electrical insulation between conducting films. The quality of the resist as an electrical insulator is not known. Tentatively, the plan is to deposit insulation selectively wherever a crossover is to be formed, as at a single-intersection cryotron, with the same process that forms the resist.

Removal of Resist

Hydrogen fluoride can be used to remove all exposed polysiloxane material (the resist) without appreciably affecting metallic surfaces. A convenient source of hydrogen fluoride (HF) in the vacuum system is a small amount of a metal acid fluoride which decomposes when heated, liberating HF. The resist must be removed whenever a metallic film on one layer must join a metallic film on another layer. After removing all of the resist with HF, it can be selectively replaced by the electron beam to form the insulation between films where needed.

Resolution

Electron microprobes 200 Angstrom units in diameter can be formed with a current density that will rapidly deposit a resist.[15,16] In fact, the resist formation is the undoing of many microprobe attempts. Since the free-radical polymerization process is nearly independent of electron voltage, one must be particularly careful that secondary electrons do not cause the resist to form in a large area surrounding the point at which the beam intercepts the substrate. Electrons accelerated by 10KV have a range of 3,000 to 5,000 Angstrom units in materials of density 5 to 10, and 1,500 Angstrom units for heavy metals such as gold and wolfram.[14] The range varies roughly as the square of the voltage, so that electrons accelerated by 2KV have a range of less than 200 Angstrom units. Fortunately, the region in which most of the electrons are stopped is well beneath the surface of substrate, so that the range of an electron probably gives a pessimistic estimate of the broadening effect of scattered electrons on the spot size.

The ultimate resolution of the etching process is not known. In one attempt to measure the resolution using polystyrene spheres of known sizes to form a shadow in the resist, a sharpness of shadow was observed in the etched film which suggested a 70-Angstrom unit resolution had been achieved.

The resolution which has been chosen as a target for component construction is 300 Angstrom units. Making conductors and insulators 0.1 micron in width should not be difficult if this spot size is reached. The goal of placing a component, such as a single-intersection cryotron, in a 1-square-micron area, then, consists of "drawing" that component with a 30-by-30 pattern of dots.

Formation of the Electron Image

A pattern of electrons can be formed in many ways. In an image tube, for example, a photocathode sensitive to X rays, light, or infrared radiation generates a pattern of electrons that can then be accelerated to strike a phosphor, and thereby produce an intensified image of the pattern which falls on the photocathode. The "snooperscope" is an example of an image tube. One could start in the drafting room with a set of drawings of the circuit to be formed, reduce that set of drawings photographically, and then generate a pattern of electrons for each stage of resist deposition by placing the appropriate photographic negative between the photocathode and a light source. The pattern of electrons can be reduced electron-optically with either electrostatic or electromagnetic electron optics.

A second technique for generating a pattern of electrons is to form a mask by photoengraving, exactly as if the mask were to be used to control the deposition of evaporated atoms, but then to allow the mask to intercept all of the electrons except those which pass through holes in the mask. This technique is used to form the letters in the charactron tube.[18] When using a mask, and when using electrostatic electron optics inside of the system containing the hydrocarbon or siloxane vapor, all parts on which a resist is not desired must be heated to 250 degrees C. At this temperature, the monolayer does not form, and the resist builds up at an exceedingly slow rate.

A scanning system for depositing the resist is one method by which the building process can be controlled electrically, possibly from another information-handling machine. When a large number of components are to be produced, the drafting of circuitry will become a burdensome task for all but strictly repetitive circuitry. Ultimately, therefore, the electron image must be formed by completely electronic means. Scanning is much slower than the simultaneous deposition of a reduced pattern, so a need exists for a means to accomplish the high-speed conversion of data from a computer into an electron image.

Connections to a Microminiature Circuit

The process by which narrow conductors are formed can be used to make conductors which become wider toward the edge of the substrate, eventually reaching a width of 25 microns. A 0.001-inch wire can then be soldered to the widened area, possibly by the pyrolytic decomposition of tetraethyl lead onto heated areas that correspond to holes in a reflective soldering mask. The power-supply connections and a limited number of information channels can thereby be made. Large numbers of input channels, such as a retinal field, pose a difficult problem. An optical input and output system involving photocathodes and electroluminescent "spots" make by the same 0.1-micron building technique is the best solution to date.

A retinal field would be focussed, using light optics, onto a pattern of 1,000 × 1,000 tiny photocathodes, which are formed at one place on the substrate. At the output region of the substrate, a pattern of 1,000 × 1,000 tiny electroluminescent spots might be constructed which will generate light to form an output image. If one takes a present-day light panel, and puts it under a microscope, he will find that all of the light comes from a large number of extremely tiny spots. The nature of the light-forming process is not completely understood. Field emission from the sharp corners of crystallites may be the cause of the localization of light. In order to bring the hypothetical output device to reality, then, one will have not only to learn how to construct the spot on a 0.1-micron scale, but will also have to learn what it is that must be constructed.

Latent Images

The resist deposited by an electron beam is a latent image of the circuit to be etched. Many adjacent latent images can be formed, and the etching then done in a single step. The postponement of etching until many resist images have been deposited will almost certainly be necessary if a large number of components is to be assembled. In any one photographic field, the number of resolution units is fixed, usually by aberrations in the lenses. Electron lenses, particularly, have a large spherical aberration, and therefore the electron trajectories must be restricted to nearly paraxial paths by apertures. A circuit of approximately 1,000 components is perhaps the largest that can be handled at one time by the electron optical system, and therefore a number of latent images must be formed. Registration problems can be eased by wasting some area near the junction of two latent images. For example, a connection might be made between a horizontal conductor at the edge of one field and a vertical conductor at the edge of the other; considerable leeway in positioning can thereby be provided. In the case of cryotrons, one conductor can be a control element and the other a gate element, so that a cryotron that is half in one field and half in the other would provide the interconnection.[19]

Fortunately, the same electron optical system which reduces the image can be used to magnify the scattered electrons and give an electron-microscopic view of the construction process. A person can thereby serve as a part of the alignment feedback loop, or a system of photocells can replace him. At any rate, for small systems of components, one can watch the entire process. Time permitting, the person could even look first at the area on which the next image is to be formed to see if any bad imperfections exist in that area. Corrective means might then be taken to avoid regions having holes or bad spots.

Document Storage

If the 0.1-micron building scale is reached, a 10,000-line per-millimeter photographic process will be available for the storage of documents such as texts, photographs, or maps. Electron optical reading has the advantage of high contrast. A second, and much more important advantage of electron optics over light optics is the very large depth of field and depth of focus of electron optics. In an electron microscope, the depth of focus is more than one foot. The large depths of field and focus result from the small numerical aperture that must be used in an electron optical system because the lenses have such large spherical aberrations. The specific energy density in an electron beam is exceeding high, however, so in spite of the small aperture, a high enough current density is usually available to produce chemical changes. A discussion of the beam current density versus demagnification is given by Cosslett and Haine.[20] A reduction of 2,500 to 1 appears feasible with the electron-optical process described, but such a reduction would be all but impossible in a light optical system because of the demands placed on the mechanical design by the extremely small depth of field. The electron-optical system, of course, can also allow reduction to a scale far below the wavelength of light.

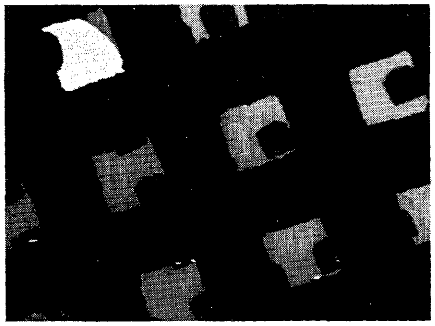


Fig. 1. Electron micrograph of a molybdenum film etched into 0.001-inch squares

A scanning read-out system for the latent resist image would also be possible, so that etching of the metallic film would be unnecessary. The difference in secondary emission characteristics of the resist and the metallic film would provide an output signal. Such a system might be useful in telemetering space-probe photographs back to earth.

Summary and a Demonstration

One possible process for producing etched wiring on a 0.1-micron scale has been proposed. Fig. 1 shows an electron micrograph of a simple test of the process carried out by C. Crawford and A. Baker at Massachusetts Institute of Technology. The large dark lines are the wires of a 200-mesh electron microscope specimen screen. Corresponding points on the screen are separated by 0.005 inch. On this mesh was deposited a Parlodion film, and on the Parlodion film was deposited a silicon monoxide film (the gray areas). The light area is a hole that occurred in the Parlodion film. The silicon monoxide film formed the working surface for the demonstration. A molybdenum film of about 800 Angstrom units was then deposited onto the silicon monoxide film in a vacuum system. The vacuum system was opened and a 400-mesh copper specimen screen (400 wires wires to the inch) was placed up against the molybdenum film. The vacuum system was again pumped down, and after admitting a small amount of tetraethoxysilane at 10^{-4} mm Hg, a 1200-volt electron flood beam 1 milliampere per square centimeter in current density was turned on for one minute. Then the vacuum system was opened, the 400-mesh screen removed, and the vacuum system closed again and pumped down. The substrate was then heated to 300 degrees C and chlorine was admitted to the vacuum system at 10^{-4} mm Hg. After one minute, the screen was placed in the electron microscope, and the molybdenum film was seen to be cut up into little squares about 0.001 inch on a side. Each square corresponds to a hole in the 400-mesh screen. Magnified pictures showed the edges of of the film which were formed around pieces of stray material. The resolution is estimated to be finer than one micron.

The chemical reactions involved in the process appear to work. Many details must be explored and many techniques developed before the entire process becomes a practical reality and systems of information-handling components can be printed.

References

1. Simulation of Self-Organizing Systems by Digital Computer, **W. A. Clark, B. G. Farley.** *Transactions,* Professional Group on Information Theory, Institute of Radio Engineers, New York, N. Y., vol. PGIT-4, Sept. 1954, pp. 76–84.

2. Generalization of Pattern Recognition in a Self-Organizing System, **W. A. Clark, B. G. Farley.** *Proceedings,* Western Joint Computer Conference, Institute of Radio Engineers, New York, N. Y., Mar. 1955, pp. 86–91.

3. The Perceptron, A Theory of Statistical Separability in Cognitive Systems, **F. Rosenblatt.** *Cornell Aeronautical Laboratory Report No. VG-1196-G-1,* Ithaca, N. Y., 1958.

4. Intelligent Behavior in Problem-Solving Machines, **H. L. Gelernter, N. Rochester.** *IBM Journal of Research,* New York, N. Y., vol. 2, 1958, p. 336.

5. The Computer and the Brain (book), **J. von Neumann.** Yale University Press, New Haven, Conn., 1958.

6. Organization of the Cerebral Cortex (book), **D. A. Sholl.** John Wiley & Sons, Inc., New York, N. Y., 1956.

7. Machines that Think They Can Learn, **P. M. Lewis,** (unpublished).

8. Vapor Plating (book), **C. F. Powell, I. E. Campbell, B. W. Gonser.** John Wiley & Sons, nc., New York, N. Y., 1955.

9. On the Investigation of Specimen Contamination in the Electron Microscope, **J. Hillier.** *Journal of Applied Physics,* American Institute of Physics, New York, N. Y., vol. 19, 1948, p. 226.

10. The Origin of Specimen Contamination in the Electron Microscope, **A. E. Ennos.** *British Journal of Applied Physics*, Institute of Physics, London, England, vol. 4, 1953, p. 101.

11. The Sources of Electron-Induced Contamination in Kinetic Vacuum Systems, **A. E. Ennos.** *Ibid.*, vol. 5, 1954, p. 27.

12. Electrode Contamination in Electron Optical Systems, **K. M. Poole.** *Proceedings*, Physical Society, London, England, Sec. B, vol. 66, 1943, p. 542.

13. Stehende Lichtwellen nach O. Wiener, electronenmikroskopisch sichtbar gemacht, **G. Mollenstedt, R. Speidel, W. Koch.** *Zeitschrift fur Physik*, Berlin Charlottenburg, Germany, vol. 149, 1957, p. 377.

14. A New Method for Recording Electrons, **P. H. Carr.** *Review of Scientific Instruments*, American Institute of Physics, New York, N. Y., vol. 1, 1930, p. 711.

15. Microanalysis by Means of Electrons, **J. Hillier, R. F. Baker.** *Journal of Applied Physics, Ibid.*, vol. 15, 1944, p. 663.

16. Single Crystal Electron Diffraction by Microcrystalline Materials, **N. Davidson, J. Hillier.** *Ibid.*, vol. 18, 1947, p. 449.

17. Comparison of the Practical Limits of X-Ray and Electron Microscopy, **V. E. Cosslett.** *Proceedings*, International Conference on Electron Microscopy, London, England, 1954, pp. 311–17.

18. The Type C19K Charactron Tube and Its Application to Aircraft Surveillance Systems, **J. T. McNaney.** *Convention Record*, Institute of Radio Engineers, New York, N. Y., pt. 5, Mar. 1955, pp. 31–36.

19. Design of a Cryotron Computer, **F. Herzfeld.** *B. S. Thesis* in Electrical Engineering, Massachusetts Institute of Technology, 1958.

20. The Tungsten Point Cathode as an Electron Source, **V. E. Cosslett, M. E. Haine.** *Proceedings*, International Conference on Electron Microscopy, London, England, 1954, pp. 639–44.

Organization and Retrieval of Records Generated in a Large-Scale Engineering Project

G. A. BARNARD III L. FEIN

THIS PAPER describes the organization of a file and retrieval system developed for use on a large-scale engineering project, the development of the Electronic Recording Machine Accounting (ERMA) Mark I. ERMA, which was built by Stanford Research Institute for the Bank of America, is a large-scale computer and data-processing system designed to process bank checks automatically.

Although the authors have had no experience in the field of documentation as such, a useful and potentially efficient paper system was produced. The work on the file system was motivated by a respect and appreciation for the importance of a good retrieval system, not only in terms of the cost and efficiency of a research project, but also for its success as well.

The principles developed and utilized in creating this sytem should be useful to technical and research management and to others concerned with projects that can be structuralized. Some research projects, in which only the broad outlines of the problem are initially understood, will not meet the specification of prior structuralization. However, many engineering and research projects do meet this specification.

General Background

Crash programs are a significant characteristic of American industrial, commerical, and governmental operations. When such programs are hurriedly organized and executed, there is always the danger of inefficiencies arising from an inadequate information and record system.

The following is a partial list of situations and requirements that such an adequate system would serve to improve:

1. Training and orienting new personnel: Too often newly assigned people are thrown into the middle of a project without training or orientation or they are given "some kind of literature" to read for a week or so and then are expected to be familiar with the project. A simple information system designed to provide for the trainee's needs for material and information already available will increase the efficiency of this orientation program.

2. Indispensable personnel: Frequently, vital information is carried in people's heads and, because it is not recorded, is often inconvenient and difficult to retrieve. The loss of such people due to illness or resignation may be catastrophic to the success of a project. An adequate information system produces a climate that discourages information hoarding.

3. Duplication of effort: It is wasteful, to have people working on one project duplicating the work of others on related projects, though this is sometimes unavoidable because of security or other reasons. But there is no legitimate excuse for duplication of effort within a project because of an inefficient information control system.

4. Action based on less information than is actually available: Often basic information has been generated within a project, but for some reason (usually because it has not been properly recorded and filed) it is not available to those requiring it. Under such circumstances, a duplicate program may be initiated to obtain the desired information lest decisions be made on the basis of insufficient information.

5. Patent preparation: Innumerable man-hours are expended by patent attorneys in interviewing an inventor to obtain relevant information for purposes of disclosure and patent. Proper documentation and filing of patentable ideas would reduce this lost time.

6. Manufacturing information: Many "one-shot" projects include designers whose design paper is the "back of an envelope." One need scarcely emphasize that this is inadequate as a basis for translation to production specifications. Obviously, even this insufficient information is never filed.

7. Dissemination of information: Various types of information distribution systems exist (functioning with varying degrees of success). An adequate record system, explicitly designed to co-ordinate with a good distribution system, will keep appropriate personnel promptly informed of current project developments.

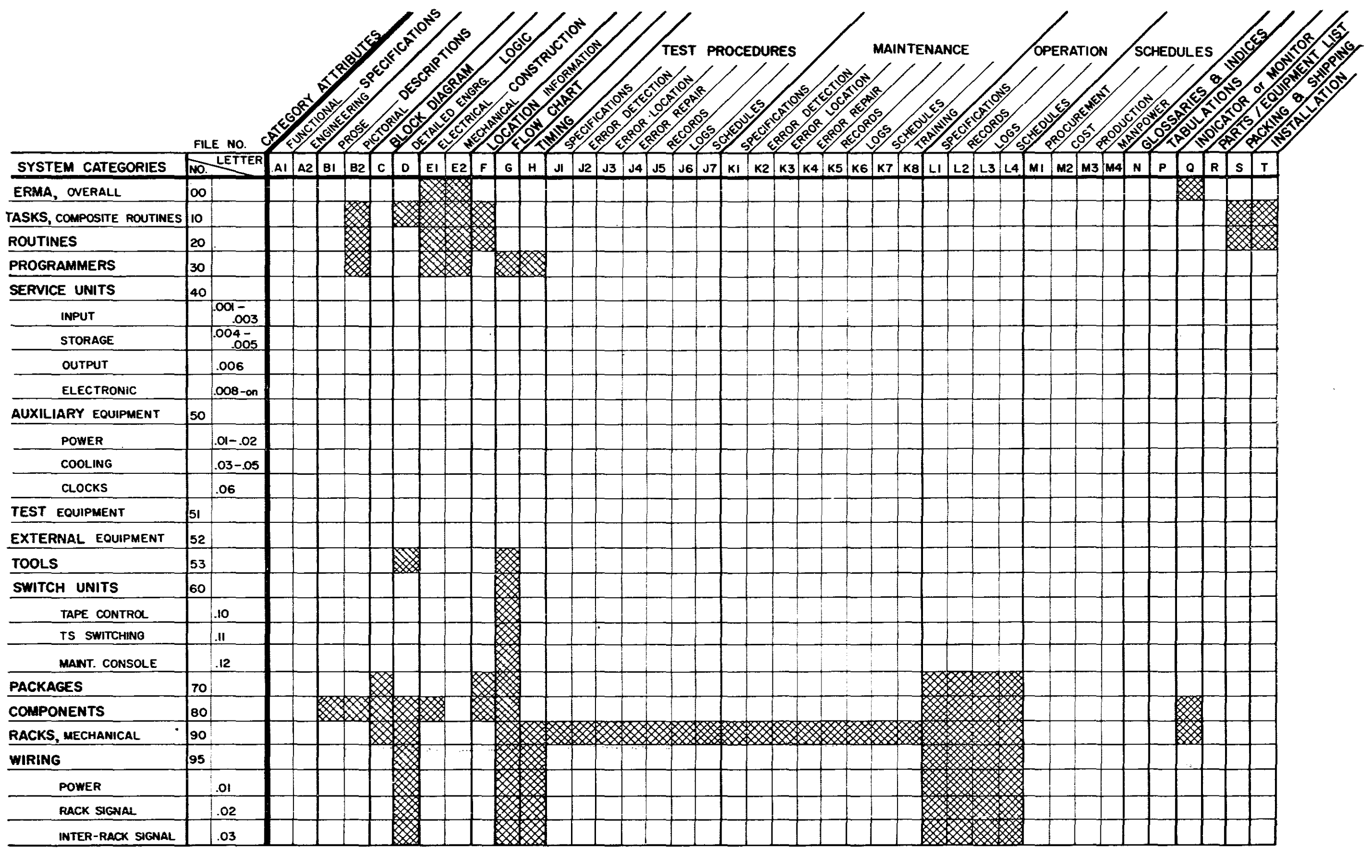
Performance Specifications

In attempting to avoid these potentially dangerous and inefficient situations, and, of course, in trying to keep an adequate record of the ERMA project, a set of performance requirements for a filing and retrieval system was developed. (During the development of the file system, these performance requirements were not actually recorded in detail as they are in this paper. Nor was the strategy actually detailed as it is here presented. With the benefit of hindsight, the authors recommend to readers intending to develop a similar file system, that performance requirements, strategy, and tactics be formalized and recorded in full detail during the course of the development.) The performance requirements

G. A. Barnard III, now with Ampex Corporation, Redwood City, Calif. was formerly with Stanford Research Institute, Menlo Park, Calif.

L. Fein is a Consulting Engineer at Palo Alto, Calif.

The work upon which this paper is based was performed at Stanford Research Institute, Menlo Park, Calif.



CATEGORY ATTRIBUTES	FILE NO.		SPECIFICATIONS		DESCRIPTIONS		BLOCK DIAGRAM	DETAILED ENGRG. LOGIC	CONSTRUCTION		LOCATION INFORMATION	FLOW CHART	TIMING	TEST PROCEDURES							MAINTENANCE								OPERATION				SCHEDULES				GLOSSARIES & INDICES	TABULATIONS	INDICATOR or MONITOR	PARTS / EQUIPMENT LIST	PACKING & SHIPPING	INSTALLATION
			FUNCTIONAL	ENGINEERING	PROSE	PICTORIAL			ELECTRICAL	MECHANICAL				SPECIFICATIONS	ERROR DETECTION	ERROR LOCATION	ERROR REPAIR	RECORDS	LOGS	SCHEDULES	SPECIFICATIONS	ERROR DETECTION	ERROR LOCATION	ERROR REPAIR	RECORDS	LOGS	SCHEDULES	TRAINING	SPECIFICATIONS	RECORDS	LOGS	SCHEDULES	PROCUREMENT	COST	PRODUCTION	MANPOWER						
SYSTEM CATEGORIES	NO.	LETTER	.A1	A2	B1	B2	C	D	E1	E2	F	G	H	J1	J2	J3	J4	J5	J6	J7	K1	K2	K3	K4	K5	K6	K7	K8	L1	L2	L3	L4	M1	M2	M3	M4	N	P	Q	R	S	T
ERMA, OVERALL	00								▨	▨																													▨			
TASKS, COMPOSITE ROUTINES	10					▨		▨	▨	▨	▨																														▨	▨
ROUTINES	20					▨			▨	▨	▨																														▨	▨
PROGRAMMERS	30					▨			▨	▨		▨	▨																													
SERVICE UNITS	40																																									
INPUT		.001 - .003																																								
STORAGE		.004 - .005																																								
OUTPUT		.006																																								
ELECTRONIC		.008-on																																								
AUXILIARY EQUIPMENT	50																																									
POWER		.01-.02																																								
COOLING		.03-.05																																								
CLOCKS		.06																																								
TEST EQUIPMENT	51																																									
EXTERNAL EQUIPMENT	52																																									
TOOLS	53							▨				▨																														
SWITCH UNITS	60											▨																														
TAPE CONTROL		.10										▨																														
TS SWITCHING		.11										▨																														
MAINT. CONSOLE		.12										▨																														
PACKAGES	70						▨				▨	▨																		▨	▨	▨	▨											
COMPONENTS	80				▨	▨	▨	▨	▨		▨	▨																		▨	▨	▨	▨							▨			
RACKS, MECHANICAL	90						▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨							▨			
WIRING	95							▨				▨	▨																	▨	▨	▨	▨											
POWER		.01						▨				▨	▨																	▨	▨	▨	▨											
RACK SIGNAL		.02						▨				▨	▨																	▨	▨	▨	▨											
INTER-RACK SIGNAL		.03						▨				▨	▨																	▨	▨	▨	▨											

Fig. 1. The file system framework

given next constituted a set of objectives which the actual system was designed to attain.

1. Any single question or statement about ERMA made by any interested observer should be keyed to a unique code number. The code number in turn should correspond to the position of the designated information in the file system.

2. The system should be closed. All major categories and category parameters characterizing the system should be identified and, at the outset, should be exhaustive. Increased knowledge about the project should be incorporable within the originally designed file system by adding subcategories and subparameters.

3. Authors or their representatives should be responsible for assigning file code numbers to all papers generated or received by them. Thus, a key should be available and it should be easy to use.

4. Every conceivable kind of paper generated within the project or received from the outside, but relevant to the project, should have a unique place in the system, the system should thus be integrated and coordinated. The file should be a technical, as well as an administrative, file.

5. The whole system should be easy to use.

The Strategy

Before deciding on the details of the filing system, a general strategy was devised to be used as a guide for the design of the actual system. This was not formalized, nor were alternative strategies identified and evaluated so that an optimum choice could be made. Nevertheless, the following is the adopted strategy:

Table I. A Representitive List of Categories

40.067	New tape classification gates
40.068	New tape write counter and controls
40.069	Nodule timing unit .3
40.070	Nodule timing unit .4
40.071	Non-an flex word transfer unit
40.072	:(Unassigned category)
40.073	Old tape read counter and control
40.074	Overflow senser
40.075	Parity checkers
40.075.01	Parity checker #1
40.075.02	Parity checker #2
40.075.03	Parity checker #3
40.075.04	Parity checker #4
40.075.05	Parity checker #5
40.075.06	Parity checker #6
40.075.07	Parity checker #7
40.075.08	Parity checker #8
40.075.09	Parity checker #9
40.075.10	Parity checker #10
40.076	Pre-routine switching unit timer
40.077	Programmer B clock gating service unit
40.078	Printer bit and column counter
40.079	Printer code wheel statisizors and serializers
40.080	Printer controls, miscellaneous
40.080.01	Code wheel data/complement
40.080.02	Format relay timer
40.080.03	Gate bar senser
40.080.04	Set and reset

1. A multico-ordinate system with corresponding codes should be used.

2. An explanatory index (key) with simple explanations was assumed to be the best way to insure selection of appropriate file numbers for new material to be incorporated in the file.

3. To facilitate use, the physical construction of the record storage system should be modeled upon the structure of the multico-ordinate file system.

4. The entires on one co-ordinate of the file system should be the elements of the various levels (categories) of the hierarchical structure chosen to represent the ERMA Mark I system.

5. The entires on the second co-ordinate of the file system should be attributes characterizing the elements (categories) on each level of the hierarchy. These attributes should be generic to every category.

6. The entries on the third co-ordinate of the file system should be a listing by type of those activities relevant to the ERMA Mark I project.

The Special Aspects of the ERMA Mark I Project as They Affected the File and Retrieval System

ERMA Mark I was intended to be an engineering model, later to be product-designed and built in quantity. The engineering model was to be shipped to,

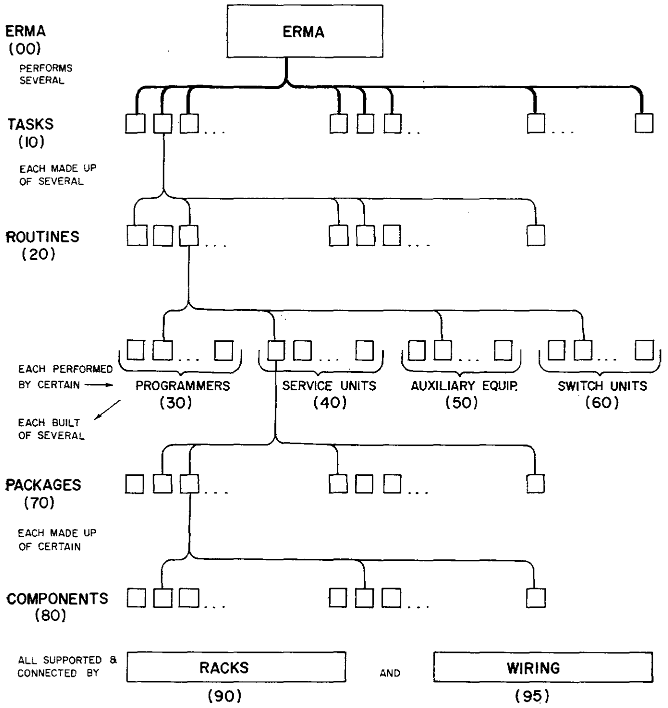
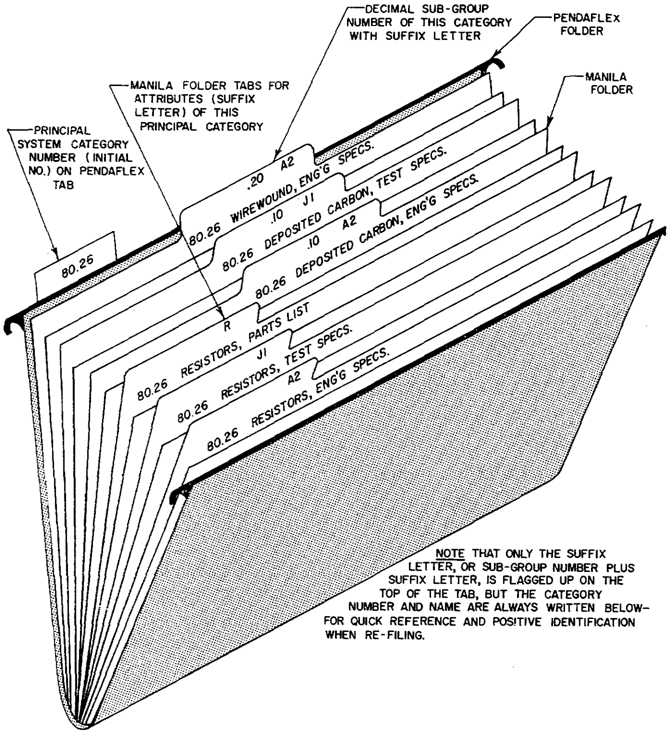


Fig. 2. The ERMA Mark I hierarchical structure


DECIMAL SUB-GROUP NUMBER OF THIS CATEGORY WITH SUFFIX LETTER
PENDAFLEX FOLDER
MANILA FOLDER TABS FOR ATTRIBUTES (SUFFIX LETTER) OF THIS PRINCIPAL CATEGORY
MANILA FOLDER
PRINCIPAL SYSTEM CATEGORY NUMBER (INITIAL NO.) ON PENDAFLEX TAB
80.26
.20 A2
80.26 WIREWOUND, ENG'G SPECS.
.10 J1
80.26 DEPOSITED CARBON, TEST SPECS.
.10 A2
80.26 DEPOSITED CARBON, ENG'G SPECS.
R
80.26 RESISTORS, PARTS LIST
J1
80.26 RESISTORS, TEST SPECS.
A2
80.26 RESISTORS, ENG'G SPECS.

NOTE THAT ONLY THE SUFFIX LETTER, OR SUB-GROUP NUMBER PLUS SUFFIX LETTER, IS FLAGGED UP ON THE TOP OF THE TAB, BUT THE CATEGORY NUMBER AND NAME ARE ALWAYS WRITTEN BELOW—FOR QUICK REFERENCE AND POSITIVE IDENTIFICATION WHEN RE-FILING.

Fig. 3. Records storage and arrangement

installed, operated, and maintained at a pilot installation. The data-processor of the system was a wired-program digital computer with storage drums, magnetic tapes, keyboard entry machines, and a statement printer.

The ERMA project generated recorded information concerning, not only design, engineering, and production, but also shipping, installation, maintenance, and related activities. Of course, potential manufacturers, patent people, public relations people, and accountants, as well as designers, shop people, etc., made demands on the system for information of special interest to each.

The following is a partial list of the various kinds of records generated in a project such as the Mark I ERMA project:

1. Correspondence
2. Logical design diagrams, equations, and corresponding prose descriptions
3. Circuit diagrams, schematics, and/or corresponding prose descriptions
4. Specifications
5. Mechanical drawings
6. Wiring lists and diagrams
7. Instructions
8. Standards
9. Statistics
10. Photographs
11. Schedules
12. Logs

The following is a partial list which shows the various types of individuals who were interested in using the recorded data:

1. Administrators and supervisors
2. Logical designers
3. Circuit designers
4. Wiremen
5. Technicians
6. Sponsors
7. Public relations people
8. Patent attorneys
9. Potential manufacturers
10. Trainees
11. Accountants
12. Maintenance people
13. Spare parts people
14. Technical writers

The Adopted File System

The file system originally considered was three-dimensional. The entries on two dimensions were the same as those in the system actually adopted (Fig. 1). The third dimension comprised entries consisting of the various user types enumerated.

In order to implement the 3-dimensional system, each type of user would have required separate records corresponding to the 2-dimensional entries in Fig. 1. This procedure was considered to be too costly and, reluctantly the plan was abandoned.

Co-ordinate 1

A technical file system, such as this is most efficient when it reflects the logical organizational structure of the equipment or system. The structure in turn should reflect the particular character of the equipment or system.

ERMA Mark I is a wired-program data processor with appropriate terminal equipment. The system and its functional organization may be described with the aid of the hierarchical structure illustrated in Fig. 2. (The following paragraph should be read in conjunction with Fig. 2.)

ERMA Mark I performs certain tasks. These tasks are performed by execution of some combination of a number of standard routines. A standard routine is executed with the aid of one of four programmers, several of the standard service units, auxiliary equipment, and switch units. These four kinds of equipments consist of some combination of standard packages. Each package consists of some combination of standard components. The packages and components reside in racks with the wiring constituting the connecting links.

Each italicised heading is designated as system categories with a 2-digit code number ending in 0. Thus, ERMA is represented by 00, tasks by 10, routines by 20, etc. (Fig. 1). Actually, column 1 entitled System Categories contains over 800 entries, that is 800 subcategories each related to one of the main categories. Fig. 3 contains the identification and the key for a representative portion of the entire list of categories and subcategories A numeric code is used throughout.

Co-ordinate 2

For this co-ordinate, names of attributes were sought which would be significant for all categories, where possible, and which would be sufficiently exhaustive to allow the filing of all generated and received documents. Such a list of category attributes is characteristic of the nature of a project or field of study. These attributes are listed on the horizontal co-ordinate. Success in selecting universally applicable attributes is measured by the number of shaded intersections in Fig. 1. These shaded intersections correspond to attributes which do not apply to a particular system category. Attribute codes are alphanumeric.

All file numbers are composed of two parts: the first is the number designating the category or subcategory; the second is the letter code for the attribute; the two are connected by a hyphen. Thus 40.075-A1 is the file number for functional specifications of parity checkers, while 40.075.08-H is the file number for timing diagrams of parity checker no. 8 (Fig. 3).

Originally, code numbers for the system categories had two decimal digits. There were 11 numbers assigned to these categories; the first ten were 00, 10, 20, 30, 40, 50, 60, 70, 80, and 90. 95 was used for the eleventh. Entries 51, 52, and 53 were added later. Expansions of each category code were made decimally. A decimal point was placed after the category-code where subcategory codes were listed numerically to correspond to an alphabetical list of the subcategory names. Occasionally subsubcategories were coded, also decimally. Thus service units called Parity Checkers are grouped under the subcategory code 40.075 with the ten parity checkers differentiated from one another by the use of subsubcategory codes 40.075.01 through 40.075.10 (Fig. 3).

Because of their size, it is often inconvenient to file drawings with other records. Drawing transparencies create the additional problem of preventing creases during handling. Nevertheless, the category key was maintained for drawings, except that a prefix and a suffix were added to the key. The two notations added were: 1. a letter, A, B, C, D, or R preceding the file number indicates the standard paper sizes so that the drawings can be sorted by size before being filed within each size group; 2. a lower case identification letter following the number (and separated from it by a hyphen) identifies the drawing from others that may be on the same subject. Thus the drawing number D 40.103.01-G-c is the third different flow chart drawn of the Temporary Storage sorter, Main Unit, and it is on "D" size paper. Revisions are not indicated by the file number.

In a large-scale computer project, various kinds of lists or tabulations are required and produced. In the ERMA project these invariably consisted of listings of one category or subcategory with respect to another. Examples are:

1. A listing of the quantities of each standard package in ERMA was filed under 00-P(70), where the parenthesis contained the number corresponding to the type of item being tabulated. This was done only in the case of tabulations, that is, only after the attribute letter P. It is evident that without following parentheses the file number 00-P represents many different types of lists and tables about ERMA.

2. A listing of the number and type of resistors used in the register package would bear the file number 70-P(80.26). Thus, the first two digits of the file number represent the item of principal interest about which a descriptive list has been made; the item type being listed is represented by the number in parentheses.

Fig. 3 is self-explanatory. It shows the arrangement of the records in the file folders.

This file system is "closed" in the sense that it is bounded by the decimally integral coding of categories from 00 to 100 and by an exahustive listing of generic attributes and categories. However, it is expandable in the sense that, when necessary, new entries are permitted within the subcategories and parameters without restorting to modification of the original file structure. For example, categories test equipment-51, external equipment-52, and tools-53, were added later in the only reasonable location; in the 50 series, auxiliary equipment. This method of adding new categories to the file can be likened to the use of the familiar accordian file in which a suitable pocket can be stretched to fit new material, but the structure of the containing folds of the file and its major pockets remain unchanged. This is in contrast to a file of rigid construction in which the only recourse for addition is to append material or to make a new container.

Technical Management of Research and Development Projects

A filing system such as this is a most important tool for technical management. Recognition of this fact is obviously a prerequisite for the success of the system. As is the case with the introduction of any new method or procedure into any organization, management must give it acceptance, understanding, and support.

Supplementary advantages of this system are as follows:

1. It provides an integrated, coordinated, and controlled record of project activity.

2. It reveals the paper and hence the activity which needs recording and follow-up. It literally reveals "the holes" in the project.

3. It compels structuralization of the project concepts. In order for system categories to be specified, structure specification is essential.

The authors believe that the principles of organization of this system are applicable to many technical projects.

Discussion

J. C. Leifer (AMF): How many people were required to maintain your ERMA files? How many man-hours were required to set up the filing system? Did the system work as well as was anticipated at the beginning of the project?

Dr. Fein: Before answering the questions, I would like to indicate the background against which this work was done.

The file system was not set up at the beginning of the project. It was set up after the project had already been in progress for a couple of years. A file system of a sort existed. It had to be redone.

How many people were involved in maintaining the ERMA file? After it was set up, it took a couple of people. These were actually part-time people.

How many man-hours were required to set up a filing system? In this case, it was a problem of redoing what already existed. Certain portions of the file had to be reconstituted. Five girls worked for about 6 weeks to redo this.

Does the system work as well as it was anticipated at the beginning of the project? The system was in operation for about 8 months before the bank decided that they wanted to get a manufacturer. During this 8 months, generally, the reports we got were good.

H. Y. Juliusberger (International Business Machines Corporation): How was the problem of cross-referencing between categories handled? (e.g., May related information be available in a different file than the one looked into?)

Dr. Fein: The problem of cross-referencing was handled in this way:

A given document was filed by looking into the file system developed and, in some cases, cross-referencing might have required the same document in three, four, or five different places.

This sort of thing would not have to happen if the file system had been set up at the beginning of the project, and if all the users had understood it. They would then create paper in such a way, and they indeed would be responsible for filing it in the first place, so that a single piece of paper would belong in one and only one cubby hole. Information and paper generated referring to many separate classifications should not exist, providing the user co-operates.

A. S. Rettig (Radio Corporation of America): How do you propose to protect any of the documents filed so that, if removed, they may be retrieved?

Dr. Fein: This is a problem of control of documents that most libraries have.

The way it was done here was to have a custodian of the files who actually checked out the information removed, similar to that which a librarian uses when you check out a book in a local public library. The effectiveness of the control procedure depends upon the co-operation of the user.

File Problems Associated with The National Menu Study

P. M. THOMPSON

THE ORGANIZATION and structure of an information file cannot properly be discussed by itself. The design of a file only makes sense when it is considered as part of the entire problem of information retrieval and maintenance of the file. In any particular instance consideration must be given to the expected frequency of use of the file, and to the particular machine logic that will be employed both in retrieval and in maintenance. A properly designed information file will reflect an optimization of some sort, weighting among other things speed, accuracy, and cost. The filing and retrieval activity, of course, is usually only a part of a total information processing system. A balanced over-all design of the entire machine system must, in turn, be achieved which will include not only all aspects of the information processing at the electronic center but in addition, all available criteria by which one can discover the proper relationships between the electronic data-processing system and the basic objectives of the entire business enterprise.

The Menu Study information file is unique from several points of view. It is not part of a routine data-processing system, but is of a one-shot nature. This file of information resulted from a very great field study of American eating habits. It is being subjected to many specific analyses by a number of food companies seeking information in very much the same way that they would were they to conduct a series of (relatively) small field surveys. It was the aim of the menu study to put enough data into one large file so that "field studies" could be made at the machine rather than in the field. There are many advantages to this approach once sufficient financial backing is available to make it possible. Relatively short questions can be formulated and answered by machine quickly and inexpensively. Research questions characteristically lead sequentially from one to another. With this large study, the data on tape are sufficient to answer large quantities of question sequences for a great variety of food interests. Thus it is seen that many and varied questions will be put to these data.

The National Menu Study

The National Menu Study was undertaken by the Market Research Corporation of America with underwriting co-operation of several large food processing companies. A representative sample of 4,000 families recorded the details of well over one million menu item servings. About 2,000 menu items are defined in a 5-level "dish" code structure. The ingredients of all home cooked dishes are coded and included in the file. In addition to classification codes for the individual dishes and ingredients, more than 50 other measurements were taken. They include other classification data related to the serving of the dish, and also classification data describing the meal and day of the serving and of the family serving. Examples of the measurements taken are:

Family: Geographic location, age, sex, and diet status of individual members, family income level, education of head, etc.

Day: Weather, who shopped for food, number of hours the housewife was away from home, etc.

Meal: Type of meal, where eaten, time, family members and guests present at meal, etc.

Dish: Family members and guests eating the dish, who prepared it, new or left-over, fundamental dish or additive, etc.

It is particularly important to note that all information is coded into discrete cells. A family with an income falling between $6,000 and $6,999 would, for example, be coded 6 under "income." A family member 13 to 17 years old would be coded 3 under "age." Thus, nowhere in the file is the actual value of a measurement recorded, only coded representations with each code number covering a cell. In general, the entire range of each variable is described in 10 or fewer cells. The greatest exceptions are the first and second levels of dish codes where 64 cells are available for each level.

Typical Analyses

In a typical analysis of these data, only two kinds of basic operations occur: 1. A selection of an element of data is made, and 2. some counting function is applied to the element selected. For example, a single question to ask would be: How many families served lobster, how many servings were there, how many people were present when lobster was served, and how many people ate lobster? In this example the selection operation involved selecting out of the data all servings of lobster. Under the first level of dish coding, code number 23 covers fish and shell fish. Under first level 23, the second level code number, 57, covers lobster. So the selection function is simply the logical expression (first level 23 and second level 57). Whenever this logical product is equal to one, the dish is lobster. If the product is zero, the dish is not lobster.

The counting function is applied only when the select function is satisfied (i.e., F = 1). In addition to the counts indicated, various distributions may be indicated. For example, one may wish to know the number of families who serve frozen beef dinners (a convenience item) on days the housewife is away from home 6 or more hours, and the distribution of those families over, say, income. The select function in this case would be: First level dish code 62 (commercial pies and prepared dinners) and second level code 31 (beef dinners) and fifth level code

P. M. THOMPSON is with the Market Research Corporation of America, Chicago, Ill.

3 (frozen) and hours housewife away code 6 (6 to 8 hours away) or code 7 (8 or more hours). The counting function would increase the family count by 1 and the proper cell count in the income distribution by one the first time the select function is satisfied for a family.

Another question might be: How many servings of frankfurters were made, and when frankfurters were served how many times were each of the following served at the same meal: sauerkraut, sweet pickles, beans; and how many people ate them? The selection would be made on frankfurters. Whenever a frankfurter serving would be encountered another count would be made to "servings." Having discovered a meal containing frankfurters, a secondary selection must be applied to the remaining dishes of the frankfurter meal to see if the secondary dishes, sauerkraut, sweet pickles, or beans were served. When a secondary select function is satisfied, the appropriate counting functions are then applied to the secondary dish.

In the following example an additional family classification variable is established based upon frequency of usage of cold wheat cereals. This classification is obtained by selecting on cold wheat cereal servings, counting the number for each family, and adding to the family classification data a new code number representing its usage rate. A cold cereal company may be interested in distributing the population of each cold cereal usage cell over its geographic regions, and then obtaining the usage of cold rice cereals for the families in each region within each wheat cereal usage cell. The machine process for this question would involve a primary selection on cold wheat cereals, counting, classification, and distribution, then secondary selection on cold rice cereals followed again by counting.

As a final example, a company may be interested in finding those people who do not eat its product when they are served, and then to discover that they do eat. Perhaps it may be of interest to distribute the noneaters according to diet status to see if there is any difference by diet in what is eaten when the company's products are rejected.

The foregoing are a few examples of kinds of questions that may be asked. While the variety of questions is almost endless, they are all made up of component select and count functions. A large percentage of the questions are found to be composed of a reasonable number of component functions put together in different combinations and relationships.

Data Organization

All of the data have been reduced to binary form and packed as tightly as possible on tapes for use with an International Business Machines Corporation *704* or *709*. Since families are individually and randomly selected for the sample, there are no functional relationships between families. Thus a select function need never encompass the data of more than one family at a time. One record of information has been created for each family and these records are brought into memory one at a time. All analyses are completed for one family at a time with no loss of generality.

For a given family, the data fall into a true hierarchial structure. Ingredients are inferior to dishes which are inferior to meals which are inferior to days which, in turn, are inferior to a given family. Advantage is taken of this structure to allow penetration into lower levels of data only when a select function is satisfied at higher levels. This saves much time in looking at data. For instance, if a question is asked for steak consumption by low income-level families, the dish level would not be examined for those families that did not qualify on income.

The family record is a variable length record since the number of ingredients, dishes, and meals vary from one family to the next. There was even a little variation in the number of days each family reported. The resulting records vary in length from several hundred *704* words to about 2,500.

The family data record is composed as follows: The first-12 words contain family classification data. The next group of words contains day classification data with two words required for each day the family reported. The next group of words contains meal classification data with two words required for each meal. Next are the dish words with two words per dish, and finally the ingredient words where each word contains three ingredients.

In order for the machine code to penetrate the family record, the following address information is contained in the record. The number of day words is contained at a known location in the family classification words at the beginning of the record. Since the first day word is the 13th word of the record, the data for any given day can be retrieved with a single address computation. In each pair of "day words" the location and number of meal words for that day are given. Thus, any given meal can be retrieved with two levels of address computation. Similarly, each meal group contains the location and number of dish words for that meal, and each dish group contains the number and location of the ingredients words for that dish plus the number of ingredients in the dish. Thus, with a maximum of four levels of address computation, any given element of data can be retrieved. If it is desired to simply examine all meal information, the location and number of all meal words are contained in the family words. The same is true for the dish words. The location and number of dish applied to an index register loop will examine all of the dish words directly.

Requirements for the Machine Code

The Menu Data is being subjected to a large number of fairly restrictive questions. Complex analyses are composed of sequential questions with successive questions reflecting knowledge gained in previous questions. An automatic machine compiler is required to prepare the individual machine codes for the various questions. To program individual questions would be too slow and costly. The code must work with variable length records. It must search only through pertinent data in the hierarchial levels of the family record. The compiler must allow matching of enough questions to fully utilize the machine memory during a pass.

There are two working codes in memory at object time. One code performs the selecting functions and the other the counting functions. The select code is simply a code that will evaluate any logical function that may be encountered. Whenever the value of the select function is unity, the select code transfers control to the count code. The count code is essentially a collection of individual routines for the various count functions.

If secondary selection is required (within primary selection), the count code, which was given control by the primary select function, passes control back to the select code after it completes the necessary counting for the primary selection. It passes along the parameters to define the secondary selection. Meanwhile the primary select parameters are preserved along with the location in the data record where the current primary selection occurred. When the secondary selection and its associated counting is completed, control will be passed back to continue the primary selection. This cyclical process is continued until the family record has been completely examined by the primary select function. The process is then repeated for successive questions until all questions in a batch have been completed. At this time a new family record is read and the entire process repeated.

The select code will consider as a unit of operation the following:

1. It will examine a logical product, or a logical sum, of as many terms as required with the extension that any number of specific codes can be tested for any variable of measurement on a logical or basis. That is to say, if one term in the logical product or sum involves say, income, several income cells can be treated on an or basis (i.e., income cell 5 or 6 or 7).

2. It will evaluate this sum or product for one combination of data involving a specific day, meal, and dish for the family who is in memory. The data are examined in descending hierarchical order and as soon as a logical product is determined to be zero or a sum to be unity the work is stopped, for the final value is by then determined.

More complex select functions are built up with successive levels of parentheses of sums or products. The inner sums or products are evaluated first (found to be zero or unity) whereupon they are then treated as single terms at the next higher level.

Parameters and other operating information are supplied to both the select and the count codes by means of tables. Each table may be considered to be a calling sequence though the codes are not employed in the usual subroutine sense. A table will supply to the select code the form of the logical expression, the specific code values that make up its terms, and the masks necessary for unpacking. In addition it will give the location of the associated counting code tables that are to be used. The counting tables will provide information as to the counting functions applicable and will give the location of any secondary select tables that may be required. They in turn give the location of secondary counting tables. Thus a chain is formed. Control will be passed as far down the chain as required and will return the same chain to the primary select table which supplies the master control for advancing through the family record.

The Compiler

The compiler has a relatively simple task. Information is presented to the compiler in a language centered about Menu Study nomenclature. The compiler has access to a large reference table where it can obtain specific address information, masks, etc. The compiler translates the questions from the menu language into the necessary sets of "calling sequence" tables. Output data space is an integral part of the count code tables. Printout heading information can be associated with the output data space for identification or results. The amount of table space required per question can quickly be determined and thus the number of questions that can be handled per batch established. The space requirements for the family record and for the select and count codes are known. Allocation of the remaining space is simply a matter of putting in as many tables as the space will hold.

All of the counting tables are located together in one section of memory. With the assistance of incremental address information available in the counting tables, a printing routine can work its way through the locations containing output data and associated headings.

Discussion

E. Herscher (Philco Corporation): When searching for the dishes that were served with steak, how does the system determine the dishes, in the same meal, that it passed before reaching steak?

Mr. Thompson: The steak in primary selection, and one set of index registers keeps track of where we are in the record on this selection. When control is passed through the accounting code back to the selecting code, another set of index registers takes over the secondary selectors, and the boundaries are set up by the mode of association, and it is completely independent. When this selection is completed, control is passed to the primary register and the original index registers pick up where they left off.

Data Processing and Information Handling

R. H. GREGORY M. TRUST

IT IS generally recognized that, at the present stage of development, businesses must process data to produce reports useful for management guidance in making decisions. Such reports are, however, often more dependent upon what data are available and the mechanics of processing than upon managerial needs for facts and abilities to use them. This arrangement might be called the "push" or "supply" approach to data processing.

An alternate arrangement to producing reports required for managerial purposes might be called the "pull" or "demand" approach. Reports required for managerial action are explicitly specified and a system, including both processing and data origination, is devised to produce the desired results. These two arrangements for getting useful reports from raw facts are extreme cases and many intermediate schemes exist.

Data and Information

In dealing with managerial reporting it is useful to draw a distinction between all of the facts available, "data," and those used for decision making, "information."

Data

Data can be defined as any facts that are a matter of direct observation, are known or available, and may be expressed as numbers, words, charts, or tables. Raw data arising from business transactions can be processed in the "push" or "supply" fashion, as described, to yield files and reports that might be called "processed data." Raw and processed data are interesting but may not be useful for management decision making.

Information

Information can be used to mean data, either raw or processed, that are new, accurate, and timely. A manager obtains information from a report if he learns something he did not know before, if the facts are accurate enough for the situation involved, and if the report is obtained in time for him to take action. Reports are

R. H. Gregory and M. Trust were formerly with the Massachusetts Institute of Technology, Cambridge, Mass. R. H. Gregory is now with Sylvania Electric Products Company, and M. Trust is now a Consulting Engineer, International Business Machines Corporation (IBM). *704* time used in conjunction with this paper was furnished by the Massachusetts Institute of Technology (MIT) Computation Center.

useful for their information content; not because they contain data.

A report that merely confirms what a person already knows does not provide him with any information, for newness or novelty is basic to the definition of information. Information content of reports depends on the reader's knowledge about the situation described. The more he knows, the less information it contains for him, although the report may contain information for someone else.

For business usage, "information" must meet two more criteria. In order to be useful, the reader must understand the language in which the facts are presented and the meaning of the facts themselves. Understanding implies compatibility of format, semantics, and code between report and user. Otherwise, he gets misinformation or none, even though correct information is available.

Managerial Use of Information

Finally, a manager needs to be able to use information in making decisions. Most managers make decisions for only a limited area of an organization and do not have complete freedom within that limited area. Their freedom to take desired action is restricted by their ability to identify problem areas and make decisions, the carry-over effect of prior decisions, the difficulty of tracing decisions through to ultimate results, whether operations are nearly static or highly dynamic, and ability of the system to respond to decisions made at short intervals.

Information and Decision Rules

The design of an efficient data-information system must consider the features listed, if information is to be used for decision-making purposes. Otherwise, managers are supplied with too little, too much, or the wrong kind of information. An intolerable burden of processing data may be put on report users to obtain information required for decision making.

The extraction of information from data poses many conceptual and operational problems. A fundamental assumption is that operating managers, not people in charge of data processing, will set organization goals as reflected in policy statements, budgets, and standards. Nothing said here is intended to imply any changes in the decision-making processes per se but only in the way that information is extracted from raw and processed data.

Explicit decision rules facilitate the production of information. At one extreme, management may devise completely explicit decision-making rules. Information required for the decision-making process can be produced and all raw and processed data discarded except to the extent required in developing information in a later cycle.

At the other extreme, no decision rule is formulated in advance. Lack of rules leads to open-ended accumulation and processing of data. If management is unwilling or unable to formulate explicit rules, then people responsible for data processing will develop patterns to anticipate information requirements that may arise. Unformulated decision rules give rise to retention of excessive quantities of raw and partially processed data to answer unexpected questions.

Most business requirements for information lie between the two extremes of completely formulated and unformulated decision rules. Reports are general purpose in nature and have a low information content. After a workable content and format are developed, the nature of reports changes slowly in relation to information requirements.

Information Content of Reports

Producing reports with increased information requires that reports be tailored to the needs of individual recipients and that content, format, and length vary as the information content of individual items changes. High information content for items may be discovered in initial stages so that such items can be handled appropriately in subsequent processing. A complex procedural problem arises because the information content of an item may increase or decrease at subsequent stages of processing. For example, county-by-county sales may just meet forecasts within suitable margins and be dropped from local reports as having no information content. In aggregate, nation-wide sales may fall below the forecast because of the bias in individual items. The fact that sales are below forecast will not be learned until they are totaled. In such a case, it is necessary to repeat some processing to get supporting detail to permit analysis of the off-standard total. The information content of items also changes over time. Events that are foreseeable (e.g., seasonal factors or the succeeding events in a chain of events) can be provided for in the program to modify the processing pattern.

Multiple-pass processing may be used to appraise operating results and identify situations with high information content and then to prepare suitable supplementary reports. Initial processing might be used to prepare full-length reports for reference purposes. Selective reports with high information content can be prepared later. The possibility arises that some information will be required that does not appear on the selective report. The cost of omitting supplementary information and having report users refer to reference reports should be balanced against the cost of extracting all information from voluminous reports. A 2-level reporting scheme is probably more efficient than a single-level one.

The idea of screening the output of a data-processing system to increase information content of reports depends on the development of quantitative relations that effectively perform the selection process. This is mandatory if the selection process is to be handled by machine methods.

Factors for Selection

Some quantitative factors for selecting problems deserving managerial action include whether actual and budgeted amounts are consequential, the absolute and relative differences of actual and expected amounts, and the degree of completion. The amount of managerial attention that an item selected by means of the criteria mentioned will get depends, among other factors, on the inpact of the variation upon final results or profits, managerial ability to change the impact of the variation upon profit, and the backlog of problems demanding managerial attention.

The factors used for selecting items and for determining the amount of managerial attention deserved are inter-related and, for some purposes, may be considered together. For example, a committee may pass on all capital outlays exceeding a specified amount of money. If a backlog of unconsidered proposals develops, the specified amount may be increased and many proposals handled by decision rules that are applied mechanically. By assigning numerical values to the factors for selecting items for managerial attention, it is possible to develop a quantity useful for both selecting items and ranking them in order of importance as well. Such a quantity, called "index of importance," mentioned in the following example, is discussed later in detail.

Example of Quantitative Selection

To reduce to more specific terms the idea of using quantitative factors to screen information from data, it is useful to consider an industrial situation in which monthly data can be obtained about a number of controllable items. For each

Test 2		Test 1 $A_i > M$ and $E_i > M'$ Outcome			
$\frac{A_i}{E_{it}} > p$ and $\frac{E_i}{E_{it}} > p'$		$A_i > M$ $E_i > M'$ Amounts exceed minima	$A_i < M$ $E_i < M'$ Amounts below minima	$A_i < M$ $E_i > M'$ Spending behind minimum	$A_i > M$ $E_i < M'$ Spending ahead of minimum
$\frac{A_i}{E_{it}} > p; \frac{E_i}{E_{it}} > p'$	Enough relative progress to warrant consideration	Consider further	Reject from further consideration now	Consider	Consider
$\frac{A_i}{E_{it}} < p; \frac{E_i}{E_{it}} < p'$	Too early in schedule to consider	Reject from further consideration now			
$\frac{A_i}{E_{it}} < p; \frac{E_i}{E_{it}} > p'$	Spending lagging behind estimates	Consider further	Reject now	Consider	Consider
$\frac{A_i}{E_{it}} > p; \frac{E_i}{E_{it}} < p'$	Spending ahead of estimates	Consider further	Reject now	Consider	Consider

Fig. 1. Decision rules for further consideration of an item

item the following data are available: Actual expenditure to date, A_i; estimated expenditure to date, E_i; and estimated total expenditure at completion (of project, fiscal year, etc.) which may be reestimated in any period, E_{it}.

Minimal Screening Amount

The first question to answer is whether a sufficient amount of money, either actual or estimated, is involved in an item to warrant bringing it to management attention. Tests are made, $A_i > M$ and $E_i > M'$, where M and M' (perhaps different) are minimal amounts below which an item is considered too small to deserve examination. (In making comparisons, the outcome "equal to" is treated as "greater than.")

These two tests may be considered jointly, with four possible outcomes. The outcome, $A_i > M$ and $E_i > M'$, indicates that both minimal screening amounts are exceeded and the item may, depending on the outcome of subsequent tests, deserve inclusion in management reports. The outcome, $A_i < M$ and $E_i < M'$, indicates the item is not yet important and might be rejected without further testing as having no information content and not deserving reporting. A third outcome, $A_i < M$ and $E_i > M'$, indicates actual expenditures are lagging behind a scheduled amount that now exceeds the minimal amount. The fourth possibility, $A_i > M$ and $E_i < M'$, indicates that an excessive amount of money was spent in relation to the original estimate. For each outcome except the second, further tests should be made to find whether the item warrants reporting to management.

For use in constructing the index of importance, a numerical value, MS (for minimal screening), can be assigned to each of the four outcomes for the minimal screening test. An arbitrary, large negative value can be assigned to the second outcome to offset the other factors used in computing the index of importance. A small value, say 1, can be assigned to the other outcomes to avoid affecting the index of importance which will depend on other factors. In short, the fact that an item exceeds minimal screening amounts is a necessary but not sufficient reason for including an item on reports to management.

Similar comments apply to each factor where a yes-no answer is obtained from a comparison. Calculations (subtractions, divisions, etc.) on the other hand, yield quantitative results that can be used directly in constructing the index of importance.

Absolute Difference

The absolute difference between actual and estimated amounts of expenditures is taken instead of merely the arithmetical difference for, at this point, the variation is more important than whether it is high or low. Also, the absolute difference avoids a minus sign (when estimate exceeds actual) that would interfere with later manipulation.

Comparison of absolute difference with a third minimal screening amount M' selects items that have consequential differences between actual and estimate, $|A_i - E_i| > M''$, and rejects those with trivial differences, $|A_i - E_i| < M''$.

For use in the index of importance, the outcome of the absolute difference test, called AD, is assigned a small value for the first outcome and a large negative value for the second outcome.

Relative Difference

Assuming that an item has information content in terms of minimal screening and absolute difference, it is necessary to obtain some measure of the difference between actual and estimated expenditure to date. The absolute difference, $|A_i - E_i|$, can be divided by the average of actual and estimated expenditures, $(A_i + E_i) \div 2$, to avoid the bias inherent in using either one alone. The resulting ratio, D_a, can be compared to a specified difference D_w between actual and estimated expenditures to date. If $D_a > D_w$, then the item ranks as an exception pending further analysis. Otherwise, the item is dropped as having no information content.

The specified relative difference, RD, can be set with the inherent variability of the item in mind with wide limits for highly variable items and narrow limits for nearer static items. Instead of merely comparing D_i with D_w to select consequential relative differences, more discrimination is obtainable by computing the ratio of D_a to D_w and comparing it with a constant, K, which is 1 or more, that can be varied to change the test for negative difference.

Relative Progress

The idea of relative progress, RP, can be used to supplement the comparison of actual and estimated expenditures with fixed minimum amounts, as described earlier. The objective is to eliminate from further consideration those items that were below some minimum degree of completion, without having to go through the process of computing actual and allowable percentage differences. Assuming that $A_i > M$ and $E_i > M'$, the following tests might be made:

$$\frac{A_i}{E_{it}} > p \text{ and } \frac{E_i}{E_{it}} > p'$$

where p and p' are two minimum ratios, or percents, if desired.

One outcome, $A_i/E_{it} > p$ and $E_i/E_{it} > p'$, indicates the item is far enough along to warrant consideration. The other outcomes have implications similar to those for the minimal screening amount test. Appropriate values can be assigned to this test for use in constructing the index of importance.

In fact, the minimal screening and relative progress tests combined have 16 possible outcomes. Selection rules can be based on the outcome of either test taken singly or both tests taken together. One possible set of decision rules is shown in Fig. 1. Other sets can be devised to select those items with desired patterns of progress or lack of progress as measured

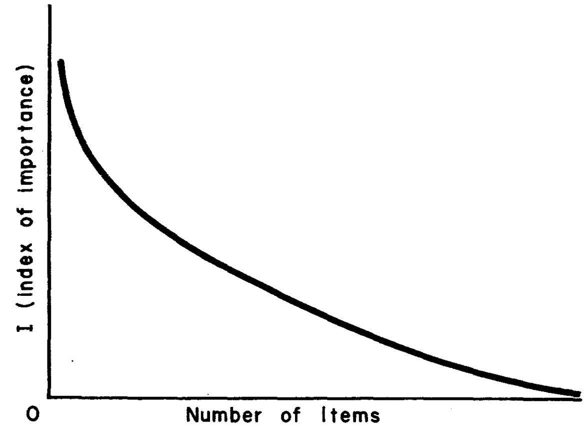


Fig. 2 (left). Suggested relation between the index of importance and the number of items

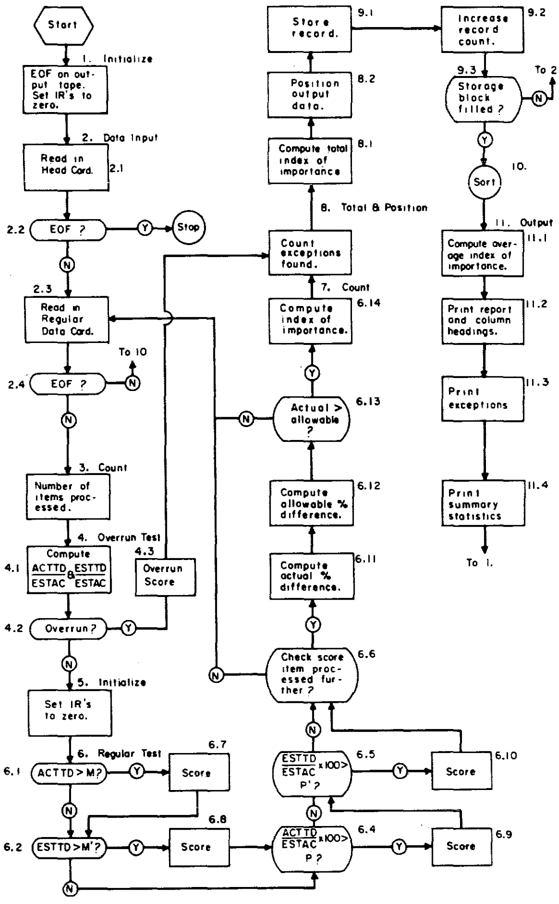


Fig. 3 (right). Flow chart for IBM 704 program

by expenditures. The selection rules would probably change over time and with external conditions.

If the minimal screening and relative progress tests are combined into one test, then one indicator, *CT* (for combined tests), can be used, with appropriate values, instead of *MS* and *RP* in computing the index of importance.

Degree of Completion

The criteria values M, M', M'' and D_w, previously discussed, were assumed to be related to the item involved but not to changeover time. In many cases it is desirable to provide for changing one or more criteria, especially D_w, as the item progresses toward a completion or review date. For example, some companies control manufacturing contracts very tightly during early stages to avoid having them go awry at the beginning. Other companies follow an entirely different pattern and tighten control over research and development contracts as completion approaches.

The average allowable difference D_w can be reduced linearly to get D_w' by the following expression, where K is a constant:

$$D_w'=\left[K-\frac{A_i+E_i}{2(E_{it})}\right]D_w$$

Special Tests

Other tests may be used to isolate items that have special significance. It might, for example, be vital to locate overrun situations; $A_i<E_{it}$. Therefore, if the minimal screening test shows $A_i>M$ and $E_i>M'$, the next test might be $A_i>E_{it}$. If $A_i>E_{it}$, then this item could be treated as an extraordinary case, information-wise, by means of a special routine for processing.

Index of Importance

An important, although commonly overlooked consideration in management reporting is to measure the relative importance of reported items, or exceptions, in order to select those most deserving attention. People usually rank items by means of qualitative descriptions. By combining some quantitative measures with judgment, a person can specify that one exception should be considered critical while another treated as routine. Further, exceptional items might be ranked in order of importance. One approach to the problem is to devise quantitative methods that reliably duplicate the capabilities of people when processing data. As has been suggested, one method is to develop, for each item, an index of importance whose numerical value is a measure of the item's significance to management.

Construction of Index

The factors to consider in developing an index of importance, as previously described are:

1. Whether the actual and expected amounts exceed minimal screening amounts $A_i>M$ and $E_i>M'$ as reflected in the value for *MS*.

2. The amount of money involved in the item, measured by the absolute difference between actual and expected amounts $|A_i-E_i|>M''$ as reflected in the value for *AD*.

3. The relative difference between actual and expected values as a percentage $2\ |A_i-E_i|/,A_i+E_i$ called *RD*.

4. Relative progress of actual versus expected to date and expected to date versus expected at completion $A_i/E_i>p$ and $E_i/E_{it}>p'$, called *RP*.

5. Degree of completion where the limits for variation are to be tightened or loosened as the item progresses toward completion $D_w'=[K-A_i+E_i/2(E_{it})]D_w$.

6. Any special facts related to the item that increase its importance K'.

The total expression for the index, I, may be the sum of these various terms:

$$I=MS+AD+RD+RP+D_w'+K'$$

Additional terms can be added to the index to cover additional tests as desired.

The index of importance is homogeneous for it consists of arbitrary numbers and ratios. Three of the factors (*MS*, *AD*, and *RP*) have either trivial values or arbitrarily large negative values. The third and fifth factors are ratios and the last factor (K') is an arbitrary number. Values can be selected or weights used to reflect the importance of individual factors.

Use of Index

Computing an index of importance for each item permits reporting the items in order of importance and refining the reporting system to vary automatically the number of items that are reported for management action. Reporting items in order of importance merely requires arranging them in descending sequence as reflected by the index.

Refinement to vary the number of items reported for management action assumes, first, that there is some finite limit to the number and size of problem situations which comprise the management load (L) that can be handled in a reporting period before newer reports become available. Reporting more than L units' worth of items leads to unused reports. Further, restricting the reporting system to the work load that management can handle presumes that the problem content of individual or inter-related items can be measured during processing operations and used to modify the program for report preparation.

A useful correlation probably exists between the number of items and the index of importance. A suggested relationship is shown in Fig. 2. Those items ranging from the most important downward that constitute a suitable work load can be reported. In order to use the idea of reporting those items that make up a suitable work load, extensive study of the range of I, frequency distribution of I, ability of managers, the decision-making process, and the way that the organization responds to decisions would be required.

Series and Parallel Test Information Screening

In some cases the implementation of mechanized management reporting schemes may require that many yes-no-type decisions be made and efficiently recorded in machine-usable form. Yes-no-type decisions or tests may be performed either in series or parallel.

In series testing, the failure of an item to pass any one test rejects it from further consideration. For data-processing work, series testing has significant economies in terms of speed. On average, processing time would be cut almost in half for those items that are rejected, since a rejected item would undergo only about half the total number of possible tests before it could be discarded. Those items that passed all tests would require similar amounts of processing whether a series or parallel scheme was used.

On the other hand, series testing has limited flexibility for it can be applied only where a decision does not depend on the result of more than one test. In management control systems it is often impossible to design completely independent tests. A decision is more likely to depend on the outcome of several tests, each of which may be of a yes-no nature. A parallel test pattern is useful in such a case by recording the results of all tests but making no disposition until all test results are examined. Processing time increases for parallel testing because of the need to perform all tests for each item, develop a composite score of outcome of tests, and examine composite score before making a decision.

By way of contrast, with a series test pattern the need to keep and examine test scores is eliminated since the result of prior tests is not pertinent to the current test. If testing is done in a mechanized fashion, the method of scoring for test outcome depends on the capabilities of the equipment used.

Computer Programming

In order to test in a more practical fashion some of the ideas advanced here, an exception processing program was prepared and run on the IBM *704* at the Massachusetts Institute of Technology Computation Center.

Program Description

The main purpose of the program was to test the effectiveness of quantitative rules similar to those described for selecting items for managerial consideration. The computer was also used for calculations and logical operations involved in preparing a report useful to management. A flow diagram showing the sequence of operations is given in Fig. 3.

Input Data

The data used to test the computer program under clinical conditions were obtained from a division of a company engaged in government research and development on electronic equipment. This division was interesting from a controls standpoint for it had only recently installed a budgetary control system. Planned for manual operations, much of the operation was quickly converted to punched cards to reduce labor and other costs. A small electronic calculator (Univac *120*) was the most sophisticated piece of equipment. The system produced a fixed-length report with all items to be controlled by the division shown in every reporting period.

Data obtained covered actual and estimated man-months of engineering effort expended up to December, 1957. No data were available for progress on each item so that it was necessary here to assume that progress was correlated with engineering effort. Each item (subaccount) was identified by a 9-digit number. A total of 104 subaccounts were used covering 12 departments that contributed engineering effort to the tasks.

Criteria Values

The criteria used to process the data were 3 man-months for M and M' and 10% for p, p' and D_w'. Examination of the distribution of estimated expenditures E_{it} at completion showed that about 15% of the items were less than 3 man-months. The criterion of 3, however, seemed reasonable from the viewpoint of total cost. A man-month was estimated to equal $1,500, giving a minimum screening value of $4,500. The percentage criteria were arbitrarily based on conservative judgment.

The specific expression used to compute the index of importance was $I=10/D_w+10E_i/E_{it}+|A_i-E_i|+|D_w-D_a|$. With the type of data being processed the relation was intended to yield an index value between zero and 50.

In case of overrun ($A_i>E_{it}$), the relationship was modified as follows: $I'=10/D_w+|A_i-E_{it}|+100$. The basic value of 100 for I' was increased as a function of the overrun and deviation permitted in the item.

Processing

The processing plan emphasized parallel-type testing as described. An overrun test was made early in the program to find whether $A_i>E_{it}$. In case of overrun, the item was always reported as an exception, but a different routine was used to calculate the index of importance. If no overrun occurred for an item, the program performed the tests and recorded the results as a test score, as described earlier.

After completing all tests for an item, a table lookup-type operation was performed which matched the test score against a predetermined set of rules. Based on the evaluation of the score, the program rejected the item (as being unimportant for managerial consideration) or continued processing to compute actual and allowable relative differences.

If $D_a>D_w'$, the index of importance was computed and the item reported as an exception. The particular expressions used to evaluate the actual and allowable relative differences were

$$D_a=\frac{2|A_i-E_i|}{A_i+E_i}(100), \text{ and}$$

$$D_w'=\left(1.5-\frac{E_i}{E_{it}}\right)D_w$$

Table I. Sample Exceptions Report Prepared on the M.I.T. 704 Computer

Dept. No. 9230
Report For Dec. 1957

Index of Importance	Identifying Number	Actual to Date	Estimate to Date	Estimate at Completion	Actual % Difference	Allowable % Difference	Test Score	Remark Code
17.4	255270930	7	16	22	81.8	15.6	15	0
16.3	255273630	20	25	58	23.0	21.4	15	0
8.9	255251734	0	3	3	187.1	10.0	10	0

Note: All expenditures in man-months.
Average allowable % difference = 20%.
Summary statistics.
Number of items processed = 11.
Number of exceptions found = 3.
Average index of importance = 14.2.

The value of D_w represented the average allowable percentage variation to be permitted for each item and would, presumably, reflect the significance of the item to the company. D_w', as defined, would start at 1.5 D_w and decrease linearly, as the contract progressed toward completion to 0.5 D_w.

Each item that was an exception gave rise to a 9-word record (in core storage) containing the following:

1. Index of importance.
2. Identifying number.
3. Actual to date.
4. Estimate to date.
5. Estimate at completion.
6. Actual percentage difference.
7. Allowable percentage difference.
8. Test score.
9. Remark code.

In case of overrun, the actual percentage difference, allowable percentage difference, and test score were given the value 0 and the remark code was assigned the value 1. The test score (with regular exceptions) contained the decimal integer representing results of tests to which the item was subjected. The remark code was not used in these test runs.

After processing and storing the exceptions for a department, the program was directed to a sort routine to arrange the exceptions into descending order using the index of importance as the key.

Results

Computer runs were made on the IBM *704* for twelve departments with the average allowable percentage difference varied from 20% to 5% in steps of 5%. A typical department report is shown in Table I.

Computer Time

Total running time for the program with data for 12 departments was 1.8 minutes of which 0.5 minute was required for reading in the binary program deck.

Input data, which were converted to tape before being read in, covered 416 subaccounts which consisted of the 104 original subaccounts run at four different values for the allowable percentage differences. In this number 291 exceptions were found in an average processing time of 0.19 seconds. At a cost of $500 an hour for computer time, processing costs were about $26 per thousand items handled, even with the high rate of exceptions encountered here because contracts had been rather loosely controlled.

Sensitivity of Parameters

An interesting point discovered during processing was that the number of exceptions changed little as the allowable relative difference, D_w', was changed. Increasing the allowable value from 5 to 20% cut the number of reported exceptions by 18%.

Allowable Percentage Difference	Number of Exceptions Reported
5	79
10	76
15	71
20	65

The implication is that managers may have considerable flexibility in establishing criteria values. Variations of plus or minus 5%, for example, may be acceptable in many applications and seemed acceptable here. Relative insensitivity to change in allowable differences reflected the fact that out-of-control items were terribly out of control. In actual application sensitivity tests would be required to determine the amount of flexibility in setting criteria.

General Effectiveness

The ability of the computer program to select and rank exceptions was evaluated essentially in qualitative terms. Reports prepared via the computer were examined by people to find how well programmed selection coincided with human judgment.

Personnel at the company supplying the data studied the computer output and indicated that there was basically no disagreement with the selection process. In a few instances the program missed items that, under a manual system, would have been selected. For the most part, personnel had special information about these situations that was not included as part of the computer input. It was agreed that a more sophisticated program could incorporate the additional facts and improve the selection process. The idea of developing an index of importance was well received. Some disagreement was expressed in the ranking of particular items; but generally this involved merely interchanging two items. In no instance was it felt that items should be radically shifted.

Industrial reaction, based on a limited sample, was favorable toward both the idea of applying large-scale equipment to this type of exception processing and the particular approach taken in this clinical case study. Several discussions with people concerned with management control and reporting schemes revealed that the problem of increasing the effectiveness of reports is becoming increasingly important to them.

For any large industrial operation, an enormous amount of diligent planning and co-ordination among various company functions is essential for widespread application. Input data must be prepared reliably and forwarded on schedule to the data-processing center. Processing must be carefully scheduled. Provisions are required for introducing changes and expanding the system to meet new requirements.

Summary

A distinction between data and information is imperative for efficient system design and operation. "Data" refer to facts that describe situations but do not necessarily convey anything useful for management decision making. "Information," on the other hand, is data useful for managerial purposes to the extent that it is unexpected, accurate, timely, and relates to situations where management has freedom to make and carry out decisions.

In general, the objective in designing a reporting system is to produce an output with high information content. This means that, at some stage in processing, those items requiring management attention are selected from the available mass of raw and processed data.

The quantitative factors used for screening depend on the particular environment involved. By using numerical weights for each factor, an index of importance can be developed for each item to be controlled. Such an index makes it possible to report items in the order in which they should receive attention. Moreover, the possibility exists of using the index to vary the number of items reported in keeping with management's ability to deal with the problems covered by reports.

Mechanized screening and ranking offers the advantage of economically reducing the amount of material that management must handle. Location of off-standard situations can be done within the processing system which would leave managers free to concentrate on decision making.

Based on experimental work done in connection with this paper, it appears that mechanized selection of information from data is practicable. From the standpoint of cost, speed, and effectiveness, results indicated that significantly improved management reporting can be obtained through the use of programs to select information from data as part of processing operations.

Discussion

Brian O'Brien, Jr. (Itek Corporation): Have you done any work on the relation between the amount of information a manager can handle and the method of presentation (graphical, tabular, etc.)?

Mr. Trust: Only in a very superficial sense We questioned the company as to what sort of graphical presentation they might want of these particular data. They indicated to us that they felt it would be useful if we could somehow, along with these reports, give them a type of bar chart in which the length of the bar indicated perhaps the significance of the item. We began to think about it, but as far as this project is concerned, it did not actually generate any reports of this type.

PILOT, The NBS Multicomputer System

A. L. LEINER W. A. NOTZ
J. L. SMITH A. WEINBERGER

AT THE National Bureau of Standards (NBS), a new large-scale digital system has been designed for carrying out a wide range of experimental investigations that are of special importance to the Government. The system can be utilized for investigating new or stringent applications of these general types: 1. data-processing applications, in which the system can be used for performing accounting and information-retrieval operations for management purposes; 2. mathematical applications, in which the system can be used for performing mathematical calculations for scientific purposes, including scientific data-reduction; 3. control applications, in which the system can be used for performing real-time control and simulation operations, in conjunction with analog computer facilities or in conjunction with other instrument installations, remotely located if necessary; and 4. network applications, in which the system can be used in conjunction with other digital computer facilities, forming an interconnected communication network in which all the machines can work together collaboratively on large-scale problems that are beyond the reach of any single machine.

Because the system was designed for such varied uses (ranging from automatic search and interpretation of Patent Office records to real-time scheduling and control of commercial aircraft traffic), the system is characterized by a variety of features not ordinarily associated with a single installation, namely: a high computation rate, highly flexible control facilities for communicating with the outside world, and a wide repertoire of internal processing formats. The system contains three independently programmed computers, each of which is specially adapted for performing certain classes of operations that frequently occur in large-scale data-processing applications. These computers intercommunicate in a way that permits all three of them to work together concurrently on a common problem. The system thus provides a working model of an integrated multicomputer network.

System Organization

Exclusive of data-storage and peripheral equipment, the central processing and control units of the over-all system contain approximately 7,000 vacuum tubes and 165,000 solid-state diodes. The basic component for these units is a modified version of the one megacycle package used in the NBS DYSEAC, which in turn was evolved from the hardware used in NBS Electronic Automatic Computer (SEAC). As a result of a more effective logical design and faster memory, however, the new NBS system will run more than 100 times faster than SEAC on programs involving only fixed-point operations; for programs involving floating-point manipulations, the advantage exceeds 1,000. The arithmetic speed of the new system derives in large part from connecting a novel type of parallel adder to a diode-capacitor memory capable of providing one random access per microsecond.

Table I. Arithmetic Operation Times

(including 4 random access times to fast memory)

	Total Time (Microseconds)	
Operation	Average	Minimum–Maximum
Fixed-Point Addition, Subtraction, Comparison	7.5	6–9
Fixed-Point Multiplication	31	22–40
Fixed-Point Division	73	72–74
Floating-Point Addition, Subtraction*	20	19–21
Floating-Point Multiplication	37	28–46

* For shift of 4 bits.

The system contains seven major blocks, which are indicated in Fig. 1, namely: 1. the primary computer, in the lower center of the figure, 2. the primary storage, upper center; 3. the secondary computer and the secondary storage, right; 4. the input-output control, upper left; 5. the external storage units, upper far left; 6. the external input-output units such as readers, printers, and displays, lower far left; and 7. lower left, the external control containing the special features that facilitate communication with people and devices in the world outside the system which is

A. L. Leiner, W. A. Notz, J. L. Smith, and A. Weinberger are with the National Bureau of Standards, Washington, D. C.

The authors wish to acknowledge the valuable contributions of their colleagues H. Loberman and W. Youden, who helped to develop the logical design and programming procedures for this system.

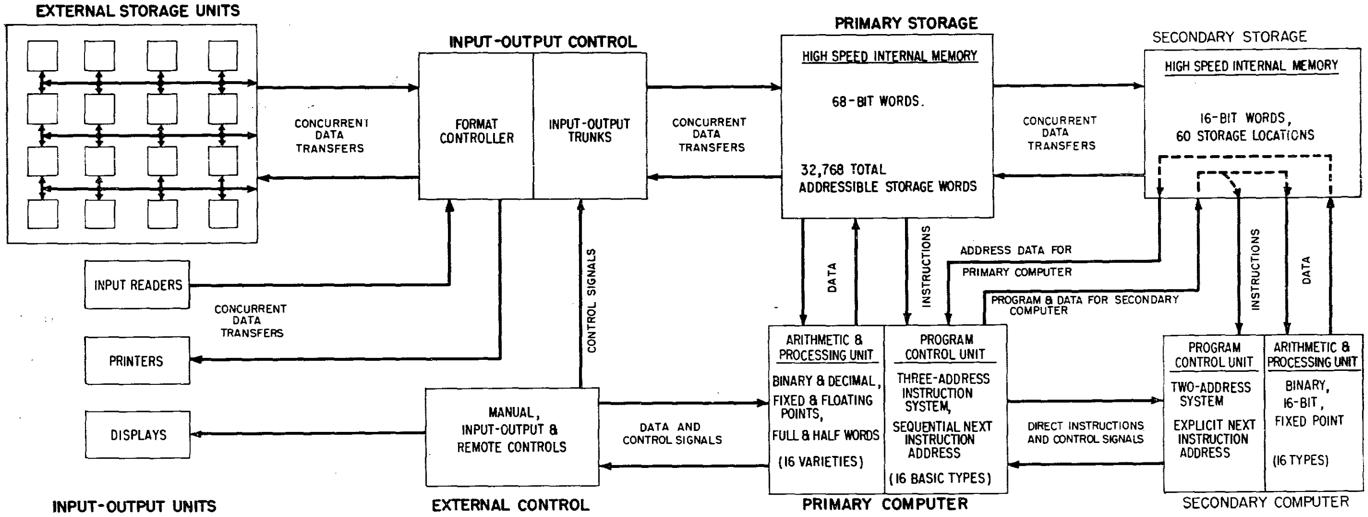

EXTERNAL STORAGE UNITS
CONCURRENT DATA TRANSFERS
INPUT-OUTPUT CONTROL
FORMAT CONTROLLER
INPUT-OUTPUT TRUNKS
CONCURRENT DATA TRANSFERS
PRIMARY STORAGE
HIGH SPEED INTERNAL MEMORY
68-BIT WORDS.
32,768 TOTAL ADDRESSIBLE STORAGE WORDS
CONCURRENT DATA TRANSFERS
SECONDARY STORAGE
HIGH SPEED INTERNAL MEMORY
16-BIT WORDS, 60 STORAGE LOCATIONS
INPUT READERS
CONCURRENT DATA TRANSFERS
PRINTERS
DISPLAYS
CONTROL SIGNALS
DATA
INSTRUCTIONS
ADDRESS DATA FOR PRIMARY COMPUTER
PROGRAM & DATA FOR SECONDARY COMPUTER
INSTRUCTIONS
DATA
MANUAL, INPUT-OUTPUT & REMOTE CONTROLS
DATA AND CONTROL SIGNALS
ARITHMETIC & PROCESSING UNIT
BINARY & DECIMAL, FIXED & FLOATING POINTS, FULL & HALF WORDS (16 VARIETIES)
PROGRAM CONTROL UNIT
THREE-ADDRESS INSTRUCTION SYSTEM, SEQUENTIAL NEXT INSTRUCTION ADDRESS (16 BASIC TYPES)
DIRECT INSTRUCTIONS AND CONTROL SIGNALS
PROGRAM CONTROL UNIT
TWO-ADDRESS SYSTEM
EXPLICIT NEXT INSTRUCTION ADDRESS
ARITHMETIC & PROCESSING UNIT
BINARY, 16-BIT, FIXED POINT (16 TYPES)
INPUT-OUTPUT UNITS
EXTERNAL CONTROL
PRIMARY COMPUTER
SECONDARY COMPUTER

Fig. 1. Over-all block diagram for PILOT

remotely located if necessary. Interchanges of information between the system and the outside world can take place at any time, on a completely impromptu basis, at the instigation of either the system or the external world, or both acting jointly.

The primary computer, a high-speed general-purpose computer, contains both an arithmetic unit and a program control unit of considerable versatility. This computer can carry out a variety of high precision arithmetic and logical processing operations, in either binary or decimal code and in a wide variety of word lengths and formats. Its partner computer, the secondary computer, specializes in short-word operations, usually manipulations on address numbers or other "red-tape" information, which it supplies automatically as needed to the primary program. The third computer of the system, called the format controller (see input-output control in Fig. 1), is specially designed for carrying out editing, inspecting, and format-modifying operations on data that are flowing in or out of the internal memory via the peripheral external units of the system. All three computers, and all the external units of the system, share access privileges to the common high-speed internal memory, which is linked to the input-output and external storage units via independent trunks for effecting data-transfers. Transfers of data can take place between the external units, the memory units, and the computers concurrently without interrupting the progress of the computational program. Because of the flexibility of the format controller, incoming data can be accepted from a wide variety of external devices and in a wide variety of formats.

Functions of the Major Units

The specific functions of the major units can be described briefly as follows:

Primary Computer

Arithmetic and Processing Unit

Using a 64-bit number word with algebraic sign, this unit carries out 7 different types of arithmetical operations, 5 types of choice (branch) operations, and 2 types of logical pattern-processing operations. See Table II. Arithmetical operations can be performed in any of 16 possible formats. For example, arithmetic can be performed using either a pure binary or a binary-coded decimal number code, and in both fixed-point and floating-point notation. Fixed-point operations can also be carried out in a special half-word format in which two independently addressable half-words are stored in a single full-word storage location. These two half-words can be processed either separately, as independent words, or concurrently in duplex format. In duplex format, the respective lefthand and righthand halves of each double operand are processed simultaneously in a single instruction time, and the two independent half-word results are written back in the corresponding halves of the full-length result location.

Program Control Unit

The program control unit interprets and regulates the sequencing of instructions in the program. It operates with a 68-bit binary-coded 3-address in

Table II. Types of Internal Operations

Primary Computer		Secondary Computer	
Name	Abbreviation	Name	Abbreviation
Arithmetic Operations			
Add	AD	Clear add	ca
Augment	AG	Hold add	ha
Subtract	SB	Store positive	sp
Multiply	MP	Transfer	tr
Divide	DV	Increase	in
Square-root	SQ	Decrease	de
Shift	SH		
Nonnumerical Processing Operations			
Transplant Segment with Shift	TL	Logical Multiply	lm
Generate Boolean Functions	GB		
Choice Operations			
Compare, Algebraic	CA	Compare, Zero	cz
Compare, Modulus	CM	Compare, Righthand Bit	cr
Compare, Equality	CE	Compare, Lefthand Bit	cl
Check Scale	CS	Compare, Negative	cn
Compare Boolean Functions	CB		
Control Operations			
Transfer Between Storage Units	TS	Check Primary and Proceed	cp
Regulate Secondary Computer	RS	Check Primary and Wait	cw
		Regulate Primary Computer	rp
		Replace Primary Instruction	ri
		Secondary Take Input from Primary	si

Table III. Contents of Primary Instruction Word

Digits numbered 1 through 68																
68-65	64-61	60-57	56-53	52-49	48-45	44-41	40-37	36-33	32-29	28-25	24-21	20-17	16 13	12-9	8-5	4-1
Tags	Address alpha				Address beta				Address gamma				Next Instn.	Code for Operation		Mon. Break Point
000±	a- Digits				b- Digits				c- Digits				d- Digits	Parameter	Basic Type	e- Digits

struction word. See Table III. Each instruction word contains three 16-bit codes which specify the addresses of each of two operands, alpha and beta, and usually the address of the result of the operation, gamma, in the main memory. The memory location of the next instruction word is specified by a 16-bit address number contained in one of 16 possible base registers; a 4-bit code in the instruction word (*d*-digits) specifies which one of the base registers contains the desired word. Whenever a register is so used as a next-instruction address source, its contents are automatically increased by unity. Choice instructions, used for program branching, from time to time may cause a new alternative address number to be inserted in any one of the base registers. This register is then used as the source of the address number of the next instruction.

Addresses alpha, beta, and gamma written in the instruction word are subject to automatic modification if desired by writing a 1-digit in a specified bit position. Such addresses are called relative addresses. Each of the three addresses (α, β, and γ) in each instruction word contains a 4-bit code group, called the *a*-, *b*-, and *c*-digits respectively, in which any base register identification number (0 through 15) may be written. When this is done, the address number to which the computer actually refers is equal to the sum (modulo 2^{16}) of the address number stored in the designated base register plus an address-modification constant, indicated in the remaining 12 bits of the 16-bit address segment of the instruction word.

Primary Storage Units

Fast Access Memory

Because of budget limitations, the initial installation of the system will contain only a relatively small section of internal memory of the diode-capacitor type. This diode-capacitor memory, originally developed at NBS in 1953, is very fast; i.e., capable of providing one random access per microsecond, but it has the disadvantage of relatively high cost per word of storage. This type of memory is available in modules of 256 words subdivided as follows:

Numerical information	64 bits
Algebraic signs and tags	4 bits
Parity check digits	4 bits
Total word length	72 bits

The over-all system is designed to accommodate up to 32,768 internally-accessible full-words, which may be held in storage units with access times ranging from 1 microsecond (μsec) to 32 μsec. Thus the minimum fast access memory can be backed up with a much larger and slower magnetic-core memory.

Inter-memory Transfer Trunk

Provision is made for transferring blocks of information between the various internal storage units in the system, concurrently with computation. The size of the block transferred may range from a single word to the entire contents of the memory, and the addresses between which the information is transferred are specified by a single programmed inter-memory transfer instruction. Automatic interlocks are provided to insure that all future references which the program may make to any memory positions involved in the inter-memory transfer operation are automatically made after the data have been shifted to the new locations.

Secondary Computer

Arithmetic and Processing Unit

The secondary computer is a high-speed independently programmable general-purpose computer that operates in conjunction with the primary computer and can perform 16 distinct types of operations using 16-bit words. These operations include 6 arithmetic-processing operations, 4 choice operations, 1 non-numerical processing operation, and 5 operations that transfer digital information or control-signals between the primary and the secondary computers. See Table II. Operation times for the secondary computer average about 2 μsec.

Both computers operate concurrently and can transfer information back and forth between each other. One of the principal functions of the secondary computer is to carry out so-called "red-tape" operations, such as: 1. counting iterations, 2. systematically modifying the addresses of the operands and instructions referred to by the primary program, 3. monitoring the primary program, and 4. various special tasks. Through the use of special subroutines for the secondary computer, both computers acting co-operatively can be made to carry out a wide variety of complex operations without unduly complicating the writing of the primary computer programs. Examples of such operations are: 1. special types of sorting, 2. logarithmic search, 3. routines involving cross-referencing, or items selected according to an attached code, 4. error analyses, and 5. operations involving small numerical fields.

Secondary Storage Unit

Associated with the secondary computer is the secondary storage unit which consists of 60 storage locations containing 16-bit words. Sixteen of these locations can be used as base registers by the primary computer and may be selected by the primary computer according to the *a*-, *b*-, *c*-, and *d*-digits in the primary instruction word. The contents of the registers selected by the primary computer in this way are automatically added to the address numbers specified in the primary computer instruction word. The secondary storage unit is also capable of being addressed directly by the primary computer. The fifteen 4-word blocks of the secondary storage are identified by 15 special primary address numbers. Other addressible registers associated with the secondary storage hold the address numbers of current and next instruction words in the primary program.

Program Control Unit

The secondary computer program operates with a 2-address instruction system, the addresses referring to words in the secondary storage unit, including the base registers. See Table IV. From time to time the primary instruction program may order the insertion of a new instruction into the secondary instruction register or may order the transfer of data in either direction between the primary storage units and the secondary storage unit. The secondary computer program may also cause data to be transferred into the secondary storage unit from the primary instruction register and can also cause information to be transferred into the primary instruction register from a location in the main memory.

Using these facilities, the secondary computer can inspect each instruction word in the primary program as it is selected from the primary store and, acting upon specifications written into the secondary program, can cause the primary instruction either to be executed as written or to be replaced by a new instruction word from a memory location determined by the secondary. Other types of discrimination can be effected by the secondary that depend upon the result of a primary operation, such as an overflow, jump, etc. These features facilitate the use of interpretive programming methods.

Input-Output Control

Concurrent Input-Output Trunks

The concurrent input-output trunks have the function of controlling the transfer of information in either direction between the internal memory and the external storage units. All input-output transfers are initiated by a single internally programmed instruction, and are carried out by the trunk units with the aid of automatic interlocks similar to those used in the inter-memory transfer trunk for preventing interference with the progress of the computing program. The size of the block of data that is transferred may range from a single word to the entire contents of the memory and may be directed to any addresses. Using two such trunks, it is possible to maintain two continuous streams of data simultaneously flowing between the internal memory and any two external storage units without interrupting the progress of the computations.

Format Controller

Data that are passing in and out of the internal storage system via the input-output trunks are subject to further concurrent processing by the format controller. The format controller is an independent internally-programmed data-processing unit specially designed for carrying out general-purpose editing, inspecting, and format-modifying operations on incoming or outgoing data. Programs for the format controller are stored on removable plugboards, and the primary computer program is able to direct the format controller to select whichever particular format program may be appropriate from among the small library of format programs contained on the boards currently attached to the machine. Among the typical kinds of programs that the format controller can carry out are: 1. searching of magnetic tapes for words bearing identifying addresses or other coded labels specified by the internal program, with selective input or output of data at these selected tape locations, 2. insertion of incoming data for the internal storage units of the system into address locations specified by the incoming data itself, 3. conversion and rearrangement of data that are stored on external units in formats not compatible with the formats used in the internal units; e.g., binary-decimal character conversion, adjustment of word-length modules, etc.

Table IV. Contents of Secondary Instruction Word

Digits numbered 1 through 16		
16....13	12....7	6....1
Operation code (0–15)	Address "g"	Address "h"

External Storage

External storage in the initial installation of the system will consist mainly of magnetic tape units. Because of the flexibility of the format controller, it will be possible to supplement these tape units later with a wide variety of other types of external units without making any significant changes in the existing equipment.

Input-Output Units

The system is designed to operate with a wide variety of input-output devices, both digital and analog.

Input readers and printers

Flexowriter units and paper-tape readers and punches will be available in the initial installation. Punched card input readers and high-speed printers, along with their auxiliary controls, may be attached to the format controller in the manner indicated in the preceding paragraph.

Displays

Two types of displays are provided for: 1. pilot-light display of data and control information in the various registers and flip-flops throughout the system, in order to aid the rapid diagnosis of equipment malfunctions of programming faults, and 2. picture-tube display of real-time data stored in the internal memory of the system. This kinematic diagram type of display is very important when performing dynamic simulation operations which require visual presentation of the simulated data in real-time to the human operators.

External Control

Manual-monitor control

The term "manual-monitor" was coined at NBS several years ago to describe certain types of control operations that are initiated either manually by the machine operator or by the machine itself under conditions which are specified by means of external switch settings. The former is referred to as a manual operation and the latter is called a monitor operation because the machine must monitor its internal program to determine precisely when the operation should be performed. The type of operation to be performed as well as the conditions under which it is to be performed are specified by means of external switch settings.

This feature provides for convenient communication between the data-processor and the operator, and allows the operator to monitor the progress of the program automatically, to insert new data and instructions, and to withdraw intermediate results conveniently, without need for advance preparation of special programs. This is particularly useful in debugging programs and in checking equipment malfunctions.

Monitor operations are performed by the machine whenever the conditions specified by the external switch settings occur in the course of the program; e.g., every time the program refers to a new instruction, any time the program refers to an instruction to which a special monitor breakpoint symbol (*e*-digits) is attached, any time an arithmetic overflow occurs, etc. By pairing a particular type of manual-monitor operation with a selected set of conditions. a variety of special composite operations can be performed.

Remote Controls

Manual-monitor operations can be specified and initiated by external devices as well as by human operators. Since all of the external switch settings control only d-c voltages, the external devices can even be remote from the machine itself, and from a distance, via ordinary electrical transmission lines, they can exercise supervisory control over the internal program of the machine. This makes it possible to harness together two or more remotely located data-processing machines, and have them work together co-operatively on a common task. Each member of such an interconnected network of separate data processors is free at any time to initiate and dispatch special control orders to any of its partners in the system. As a consequence, the supervisory control over the common task may be shared among the various members of the system, and may be passed back and forth from one machine to the other as the need arises.

Summary

PILOT, the new NBS system, possesses both powerful external control capabilities and versatile internal processing capabilities. It contains three independently operating computers. The primary and secondary computers each utilize only 16 basic types of instructions, thus providing a simple code structure; but because so many variations of the formats are possible, a wide variety of computing, data-processing, and information-retrieval operations can be performed with these instructions. The secondary computer is specially adapted for performing so-called "red-tape" operations, and both the secondary and the primary computers, acting co-operatively, can carry out special complex sorting or search operations. The third computer in the system, called the format controller, is specially adapted for performing editing, inspecting, and format modifying operations. The system is equipped to transfer information concurrently along several input-output trunks, though only two are planned for the near future. Using two such trunks, it is possible to maintain two continuous streams of data simultaneously flowing between any two external units and the internal memory, without interrupting the data-processing program. The system can operate with a wide variety of input-output devices, both digital and analog, either proximate or remotely located. The external control capabilities of the system enable the machine to supervise this wide family of external devices and, on an unscheduled basis, to interrupt or redirect its overall program automatically, in order to assist or manage them.

Data Handling by Control Word Techniques

G. A. BLAAUW

EARLY computing machines executed programs which could be specified by pluggable wiring, paper tape, or cards. These programs remained unchanged during their execution. The invention of the stored program made it possible to treat a program as data. This invention was a basic step forward in the development of computing machines. It then was possible for any instruction of a program to be modified by the program itself. In practice this general facility was used mainly for the modification of addresses. Subsequently, it became apparent that programmed address computation, though sufficient in theory, was cumbersome in practice. Too much computing time and program space were required to perform these auxiliary operations. As a remedy, an address register, also called index register or B-line, was provided, whose contents could be added to the operand address. In recent machines, several index registers, up to one hundred, have been made available.

In the design of the International Business Machines Corporation (IBM)-Los Alamos computer,[1] an attempt has been made to achieve great flexibility and generality in machine functions. The indexing functions and the associated instruction set have consequently been reviewed. The built-in functions which were developed as a result of this review are the subject of this discussion.

Discussion

Index Functions

Index functions may be divided into three groups: address modification, index arithmetic, and initialization-termination. The first group provides the justification for the existence of index quantities. The second group performs the task of changing the index quantities, often termed "housekeeping." The third group concerns tests for end conditions and set-up procedures.

Address modification is the addition of the contents of an index register, the index value, to the address part of an instruction, the operand address, in order to address memory with the sum, the effective address. Address modification is used in general to address the elements of an array. An array may be one dimensional, such as a vector, or multidimensional, such as a matrix; moreover, the elements of the array may be single-valued or multiple-valued. Multiple-valued elements are common in technical computations and in file maintenance operations. In the latter case, the records of a file correspond to the elements of an array. Each record in turn contains many different values.

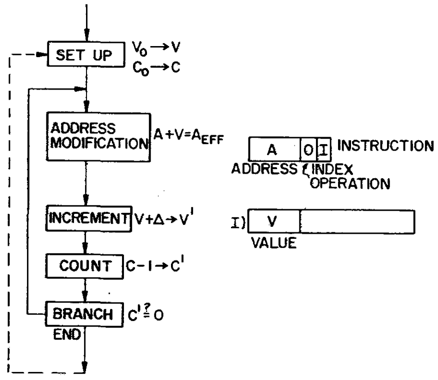


Fig. 1. Indexing loop

As a computation proceeds, successive elements of an array are addressed. Since the variable part of an address is the index value, the address computation may be accomplished by index arithmetic. Often, a simple recurrent process is used in which a new index value is obtained by the addition of an increment to the old index value.

Index arithmetic is continued until the last element of the array is addressed. Subsequently, the index value must be reset to the initial starting value for that array or for the array next to be addressed. This process is called termination and initialization. To determine when the last array element is reached, a test

G. A. Blaauw is with International Business Machines Corporation, Poughkeepsie, N. Y.

The work reported here represents the combined effort of a number of people at International Business Machines Corporation Product Development Laboratory, Poughkeepsie, N. Y.

is performed each time the index value is altered. Some of the forms of this test are: limit comparison, length subtraction, and counting. In limit comparison, the current index value is compared with a given constant, the limit. In length subtraction, a given variable, the length, is reduced by the value of the increment and tested for zero. In counting, a given variable, the count, is reduced by one and tested for zero. The three methods of test are almost equivalent. Limit comparison and length subtraction are independent of the number of increments applied. Counting, on the other hand, permits the test for completion to be independent of the size of the increments and makes it possible to use zero increments.

Instead of using a separate value such as a limit, length, or count, the index value itself can be used to determine the end of the process. In that case, the index value serves as a length, and a limit of zero is implied. The IBM *704* and *709* follow this indexing approach. A greater degree of freedom in specifying index values and tests is, however, very desirable. Therefore, an independent index value and test for termination are preferred. In the IBM-Los Alamos computer, counting has been chosen as a means for determining the end of an index value sequence.

A summary of the index functions which have been described is shown in Table I. The quantities which occur in the indexing procedure for a simple array are listed in the second column. The operations which make use of these quantities are listed in the third column.

Of the quantities listed, the index value is in the index register. This leaves four quantities which must reside somewhere. Earlier approaches have relied on storing these quantities in general-purpose memory locations. One of the four operations listed, address modification, was performed as a built-in machine operation. The other three operations normally have been performed by standard arithmetic instructions. A typical indexing loop which illustrates this earlier approach is shown in Fig. 1. The formats of an instruction and an index are shown on the right half of this figure. The instruction has three fields: an operand address field, *A*, the operation code field, *O*, and the index address field, *I*. The index has one field: the index value field, *V*. The steps which must be taken in the indexing loop are shown in the diagram on the left half of the figure. The dashed line on the diagram indicates that the program will be used several times. Each time the program is entered, the correct index value and count must be set up. In the following discussion, the advantage of storing more quantities in the index register and providing more built-in operations will be considered.

Control Word Functions

Index incrementing could be performed by a series of three single-address instructions which add the increment to the index value and return the result to the index register. Actually, only the increment and the subject index need specification. A single instruction, therefore, can be used to specify the entire operation. Such an "increment" operation can make use of the index adder which is normally provided for address modification. The main arithmetic process for data is then separated from the housekeeping process. Data registers need not be altered. Because of these advantages, an "increment" operation is normally provided when index registers are available. In the instruction format which is used in the IBM-Los Alamos computer, the operand address specifies the address of the increment.[2] The operand address can itself be indexed and thus provides indexable index arithmetic.

In index arithmetic, several increments may be used to change an index value. It therefore is desirable to address the increment and the index value independently. It also is possible to use several tests in the termination of the process. Most frequently, however, a single test is used. It is, therefore, profitable to associate the count used in the test with the index value to which the process applies. Since the test normally occurs when the index value is changed, it is logically consistent to specify incrementing and counting in one index arithmetic instruction: "increment and count." This instruction is available in addition to "increment." It becomes equivalent to "count" when the increment is zero.

The association of index value and count can be obtained in various ways. A solution, economical in time and space, is to place index value and count as separate fields in one word. Such a word will be referred to as a control word. The instruction "increment and count" adds the addressed increment to the index value, reduces the count by one and provides a signal when the count becomes zero.

The choice of counting as a test for termination and the use of an implied address for the count does not preclude other termination tests. In particular, a "compare index value" instruction is made available to allow limit tests and an "add to count" instruction can be used for length subtraction. These instructions make the instruction set more complete but are less efficient than "increment and count."

It was shown in Table I that an indexing operation requires specification of: index value, increment, count, next initial index value, and next initial count. All these values except the last two have been specified so far, either in instructions or in the control word. The last two values can now be specified by introducing a refill field in the control word. The refill address refers to a second control word, whose value and count field specify the

Table I. Summary Index Arithmetic

	Quantity	Operation
Index Use	Index Value (V)	Address Modification
Index Change	Increment (Δ)	Incrementing
Index Test	Count (C)	Counting and Zero Test
Index Reset	Next Initial { Index Value (V_0) Count (C_0)	Replace { Index Value Count

Table II. Record Deletion

Location	Control Word	Location	Control Word	Location	Control Word
b	C-c	b	C-c		
c	D-d	c	D-d		
d	H-h	d	E-e		
h	E-e			h	H-h
e	F-f	e	F-f		
f	G-g	f	G-g		
g	I-i	g	I-i		
i	J-j	i	J-j		
j	K-k	j	K-k		
Before		After			

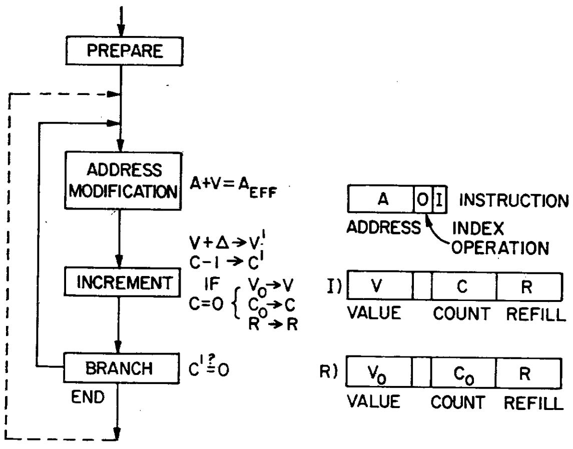


Fig. 2. Indexing loop with control word

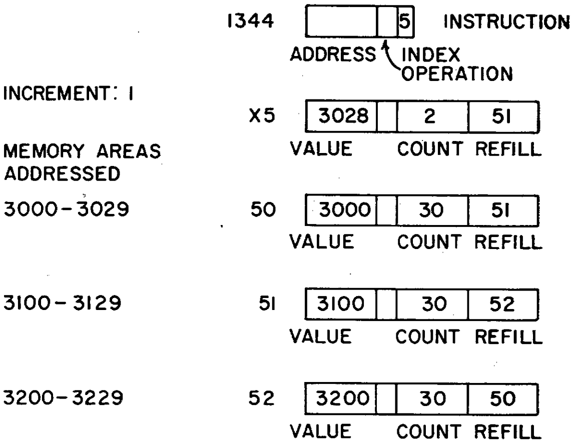


Fig. 3. Chained control words

next initial setting. In fact, the second control word is the next initial control word. The refill field then serves the general purpose of linking a control word with the next control word to be used.

The operations which use the five values previously mentioned were listed in Table I as: address modification, incrementing, counting and zero test, replacing of index value and count. All these operations except for the last have been specified as machine functions. The last operation can be restated as: replace the index word by the word at its refill address location. The operation can be part of an "increment, count and refill" instruction. This combination of operations is only meaningful when the refill operation is conditional. An obvious condition is that the count should reach zero. In addition, the instruction repertoire includes other instructions, such as "refill unconditionally."

In Fig. 2, the use of control words is illustrated for the indexing loop previously shown in Fig. 1. The current control word and the next initial control word are shown on the right half of the figure. The left half of the figure shows the simplified indexing loop. The "set-up" step is no longer required since the current control word is refilled with the initial values as the indexing loop is left. The "count" step has become part of the "increment" step. The step called "prepare" indicates that the program should be started with the proper initial contents for both control words.

The refill field of the current control word and the refill field of the initial control word are the same in Figure 2. Because of this, successive refills will all make use of the same initial control word. The use of different refill fields for successive control words is shown in the numerical example of Fig. 3. The instruction in the example specifies index 5, which has memory address X5. The control word in location X5 has a count of 2. When an "increment count and refill" instruction is given, the count becomes 1. The next time the instruction is given, the count becomes zero and a refill operation takes place from location 51. The new refill field now is 52. When, after 30 subsequent "increment count and refill" operations, the next refill occurs, the control word in location 52 will be used. The control words in locations 50, 51, and 52 refer to each other; they are chained. A series of control words which are linked to each other by their refill fields will be called a chain of control words.

Data Transmission

When an increment of one is used, as shown in the case of Fig. 3, the count becomes the equivalent of a length and represents the number of adjacent words in the addressed memory area. Furthermore, when the operand address is zero, such that the index value is used as the effective address, the initial index value addresses the first word of the memory area. A memory area can, therefore, be specified in position and length by the value field and count field of a control word. This makes it convenient to specify the memory areas involved in data transmission by means of control words and gives the control word the characteristic of a "shorthand" notation for a memory area.

Data transmission may be between two memory areas or between input units or output units and memory. The data which are transmitted in one operation will be called a "record." A control word may be used both for indexing and data transmission. This generality makes it possible to associate a control word with a record and use it to identify the record throughout an entire program, including reading, processing, and writing.

The use of control words in transmission instructions is particularly convenient when data can be moved directly between input-output units and general purpose memory. This ability is incorporated in recent computers and avoids special purpose areas to buffer records. An input-output instruction then specifies the input-output device used, the operations to be performed and the address of a control word. Data can move directly between the device and the memory area specified by the control word.

The use of control words for a simul-

Table III. Record Insertion

Location	Control Word	Location	Control Word	Location	Control Word
b	C-c			b	C-c
c	D-d			c	D-d
d	E-e			d	E-e
e	F-f			e	F-f
f	G-g			f	G-g
g	I-i			g	H-h
		h	H-h	h	I-i
i	J-j			i	J-j
j	K-k			j	K-k
Before				After	

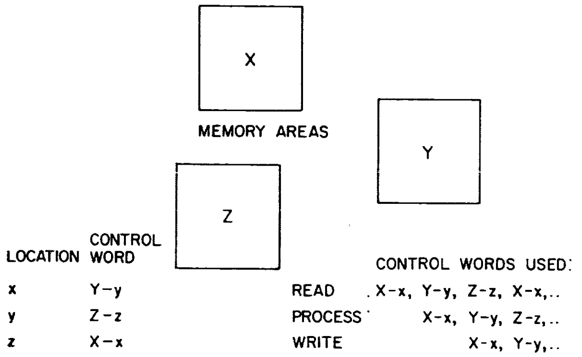


Fig. 4. Read–process–write chain

taneous read-process-write cycle is illustrated in Fig. 4. Here *X-x* describes a control word which by its value and count fields, defines memory area *X* and which has the address *x* in its refill field. Location *x* contains the next control word in the chain *Y-y*, defining record *Y*. Control word *Z-z* is placed at location *y*. Because control word *X-x* is stored at location *z*, a "ring" of three memory areas, *X*, *Y*, and *Z* is set up in which *X* is followed by *Y*, *Y* by *Z*, and *Z* again by *X*. The use of letters for memory areas and control word locations points out that the actual locations may be anywhere in memory. They need not be in numerical order. Both record areas and control words may be scattered throughout memory. In this notation capital letters are used for record areas and lower case letters for control word locations. Corresponding alphabetics are used in one control word and denote a record area and the control word of the next area in sequence.

The example of Fig. 4 has practical application in a read-process write cycle. While a record is read into area *Z*, as controlled by control word *Z-z*, processing proceeds with control word *Y-y* using data in area *Y*, and data from area *X* are written under control of control word *X-x*. At the conclusion of each of these operations, the appropriate control word is refilled and the areas are thereby cyclically permuted in function.

Instead of a single control word, a chain of *n* control words could be used in reading while a second chain of *n* control words is used in processing, and a third chain of *n* control words is used in writing. To further elaborate the example, assume that processing consists of a file maintenance operation upon the *n* records. In that case, it may be desirable to delete, insert, or sort records or groups of records. The use of control words in this type of data ordering is described next.

Data Ordering

A common procedure in data ordering operations is to move records from one memory area to another. With control words it is possible to replace the transmission of a record containing many data words by the transmission of a single control word which specifies that record.

Assume that the records *A...Z* are scattered throughout memory. The associated control words *A-a...Z-z* establish the correct order as indicated by the alphabetic sequence. It is desired to delete record *H* from this sequence and set its memory area aside. The control word *H-h* of this record is part of the chain *C-c* ...*K-k* shown in the left hand part of Table II. By interchanging the contents of locations *d* and *h*, a new order is established as shown in the right hand part of Table II, and *H* is no longer part of the sequence. A second interchange between *d* and *h* would re-insert *H*. This illustrates the complementary nature of insertion and deletion.

If it would be desired to insert *H* in the sequence...*G*, *I*, *J*,...between *G* and *I*, the second interchange would be between *g* and *h*. Table III illustrates this case.

Because the sequence...*G*, *I*, *J*...is part of sequence *A*...*Z*, the example is equivalent to a sorting operation. The sequence...*G*, *I*, *J*...may equally well be part of an independent sequence as is the case in file maintenance.

The interchange of two control words is performed conveniently by a "swap" instruction. This instruction interchanges the contents of two memory words. The insertion or deletion of a record involves only the "swap" of its control word with that of its successor. The insertion and deletion of a group of records is equally simple. Consider again the file *A*...*Z*. It is required to delete the group *P*...*R* from the file shown on the left in Table IV. By giving a "swap" instruction for locations *c* and *r*, the new order becomes as shown on the right in Table IV.

One "swap" instruction deletes the group of records just as one "swap" instruction in the previous example deleted a single record. The only differences are the addresses of the instruction. The records *P*...*R* form a ring in sequence. In the previous example, also, the deleted record *H* could be considered to form a ring in sequence since its control word was stored at its own refill location. The insertion of the records *P* ...*R* can be performed by again swapping the contents of locations *c* and *r*.

Table IV. Group Deletion

Location	Control Word	Location	Control Word	Location	Control Word
b	C-c	b	C-c		
c	P-p	c	D-d		
p	Q-q			p	Q-q
q	R-r			q	R-r
r	D-d			r	P-p
d	E-e	d	E-e		
e	F-f	e	F-f		
f	G-g	f	G-g		
g	H-h	g	H-h		
Before			After		

Table V. Merge

Location	Control Word	Location	Control Word	Location	Control Word
a	E-e	b	C-c	b	C-c
e	H-h	c	D-d	c	D-d
h	I-i	d	F-f	d	E-e
i	J-j	f	G-g	e	F-f
j	N-n	g	K-k	f	G-g
n		k		g	H-h
				h	I-i
				i	J-j
				j	K-k
				k	
		Before		After	

Swap a and d Leave F-f in a
Swap a and e Leave H-h in a
Swap a and g Leave K-k in a
Swap a and j Leave N-n in a

As a last example of data ordering, the merging of two chains of control words is shown in Table V. The "swap" operations which are required are listed in the table, which the reader may wish to verify.

Subroutine Control

Another application of control words is in subroutine control. In the preceding discussion, the control word specified a memory area which normally would contain data. However, the memory area might also contain instructions. A record can then be thought of as a subroutine. An illustration might be the use of exception subroutines which are stored on tape, drum or disk, and are called in when the exception arises. The control word is used in the "read" instruction and can subsequently be used for address modification in the branch instruction which transfers control to the subroutine and in the instruction which stores the instruction counter contents. The subroutine, therefore, can be inserted conveniently in a main sequence of instructions.

Conclusions

The preceding discussion has shown the application of control words in record handling. Both indexing and data transmission techniques make it desirable to have an index value, count, and refill facility. The three fields in the control word and the associated machine functions satisfy these requirements. The control words provide substantial saving in program space and increase in machine speed. They simplify programming of "housekeeping" operations.

Control words do not introduce entirely new functions, since their operation can be simulated on any stored program computer. Also, the introduction of count and refill is only a second order improvement as compared to the first order improvement of address modification through indexing. The simplicity of control word operation is, however, in sufficient contrast to the complexity of simulating its operation that several methods of record control are feasible which otherwise would have been impractical.

The indexing instructions have been described for the IBM-Los Alamos computer. Though elements of this instruction set are found in other machines, the effectiveness of control word techniques depends to a major extent upon the combination of all features which have been described. It is believed that control word techniques represent a significant step forward in the data handling ability of computers.

References

1. Design Objectives for the IBM Stretch Computer, S. W. Dunwell. *Proceedings*, 1956 Eastern Joint Computer Conference, AIEE Special Publication T-92, pp. 20–22.

2. The Selection of an Instruction Language, W. Buchholz. *Proceedings*, 1958 Western Joint Computer Conference, AIEE Special Publication T-107, pp. 128–30.

An Electronic Directory for Sorting Mail

A. W. HOLT

Synopsis: The primary concern of this paper is the design of a digital machine for use as a Post Office directory. Sorting must be fine enough so that the letters come out arranged in the order in which the carrier walks his route. Using straightforward memory techniques, 20 million bits would be required with an average access time of 20 milliseconds for only a medium sized city such as Washington, D. C.

The paper includes a general discussion of the information processing problem together with a brief description of some of the techniques used for physically handling the letters. The outstanding virtues and faults of several novel memory systems are also presented for comparison with the magnetic drum.

FROM THE point of view of this paper, the sorting problem of every Post Office has two parts: 1. the problem of physically handling the letters as they are divided from a few common sources into many smaller groups, and 2. the information processing problem of deciding which letters should go into what group. The physical handling problem has been studied by several laboratories, including the Rabinow Engineering Company, and this effort is resulting in the rapid development of conveyor belt systems suitable for sorting letter mail.[1–3] The problem of information processing has also been intensively studied, but the hardware is considerably less advanced.

This paper primarily concerns the application of magnetic drum memories to the information processing problem. The application of other types of memories to the problem will be discussed only as comparative examples.

The mechanization of mail sorting is described using Washington, D. C., as an example. All mail is either incoming or outgoing. Incoming mail is defined as that type of mail which is to be distributed within the city regardless of whether the mail originated in the city or out of the city. Outgoing mail is that mail which leaves the city, regardless of its source. The discussion in this paper centers around the incoming mail, which is in general a great deal more difficult to sort than outgoing mail.

Accurate statistics on the Post Office System in the United States are being gathered by the researchers who are staffing a Post Office project at the National Bureau of Standards. Some rough statistics are interesting: Washington handles approximately four million letters a day; there are about 200,000 addresses in Washington, and there are approximately 700 carriers. Chicago handles peak loads of 37 million letters per day.

The objective of sorting in the Post Office is to route to the appropriate carrier all of the letters which he will deliver. This is accomplished, information-wise, by having the sorters memorize the scheme of distribution, but this is a formidable memory problem for human beings to undertake. One of the more important end objects of the automation program for the Post Office is to relieve human beings of this memory problem. Additionally, each carrier must sort his mail to the order in which he walks his route. This operation is also a target for mechanization.

General Machine Sorting Methods

There are, at present, two important approaches in the field of automatic sorting of letter mail. In the first, and simpler form known as a "keysort" system, mechanical equipment is used to replace only the physical handling operations performed by human sorters. A human operator sits in front of a cabinet known as a keysort station where letters

A. W. Holt is with the Rabinow Engineering Company, Inc., Washington, D. C.

This study was performed under contracts with the National Bureau of Standards and the United States Post Office.

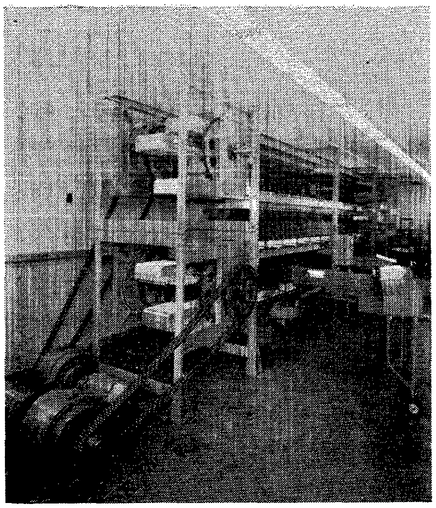


Fig. 1. Over-all view of the production prototype machine

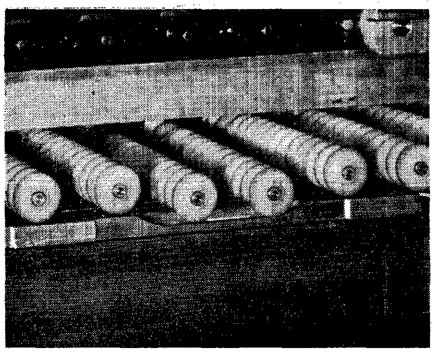


Fig. 2. Close-up view of code wheels

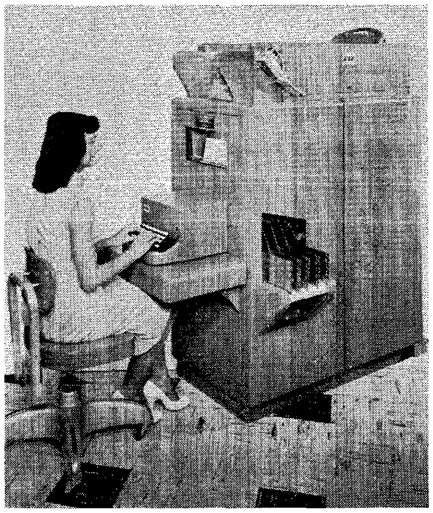


Fig. 3. Code printer

are automatically placed in front of him in position for reading, one at a time. The human operator reads the address of each letter, and directs a large sorting distributor to deliver the letter to its proper destination bin, by pushing the appropriate combination of keys in a keyboard that is located on the cabinet at which he is seated. A keysort system effectively increases the reach and speed of a human sorter's arms, but is severely limited in that the sorter must have memorized the entire sorting scheme for the addresses to which he is sorting. Both the use of the keyboard and the scheme of the city are difficult to learn, and it requires many months for an operator to become proficient.

Machines of the keysort type were introduced in Europe about 20 years ago. Two of these European machines have been installed in the United States recently for experimental studies. They are the Dutch "Transorma," in Silver Spring, Md., and the Belgian "Bell" machine in Washington, D. C.

The second, and more sophisticated, system of automatic sorting does everything that is done in the keysort system and, in addition, replaces the eyes and memory of the human operator with a reading head and an automatic electronic directory (sometimes called a memory or translator). It is called a "codesort" system. For the codesort system, a loading station picks one letter at a time off a stack, moves it past a reading head and inserts it into a conveyor pocket of the distributor. Between the time that each letter passes the reading head and is inserted into the distributor, the reading head reads the address and feeds it through electronic circuitry to the directory, which "looks up" the appropriate output destination bin and directs the distributor to deliver the letter to this bin.

In the Rabinow Sorting Distributor (shown in Fig. 1), the sorting directions are set into a group of 12 code wheels by arranging them into a specific binary combination. Fig. 2 shows these wheels. Such a group of wheels accompanies each letter and forms an "escort" memory. An escort memory of this type should be carefully distinguished from a generically different type of memory which the Rabinow Company calls the "hex" type. A hex-type memory is one where the main conveyor is duplicated in miniature; the two conveyors travel in exact synchronism, with the miniature machine carrying information instead of physical objects. The term "hex" comes from voodoo logic, where the "hex" is put on a real person by sticking pins in a doll-sized reproduction of the person.

The distributor is designed to operate in either a keysort system or a codesort system, or a combination of both, since it does not matter what loads it or what gives it direction. The Rabinow Distributor is modular in construction, and so can be contracted or expanded.

Complex arrangements of cross conveyor belts and multiple loading points can be used to increase the statistical efficiency of sorting, and the number of possible sorting arrangements using the modular distributor with transfer devices is very large. No one at this time can be completely sure exactly which sorting scheme is the best for any particular Post Office; the equipment has been designed to give, therefore, great flexibility in arrangement.

The reading head for the codesort system does not read original addresses on the mail because present reading machine techniques are not good enough to read all the different fonts and handwritings that make up the addresses on mail. Therefore, the codesort system requires multiple code printing stations at which the addresses are retyped by operators, with a machine-readable font and mechanized in a manner similar to that of the keysort station. A prototype code printing station is shown in Fig. 3. The operators of the code printers, however, do not memorize the Post Office sorting system. They merely retype the address in an abbreviated fashion on an essentially standard typewriter keyboard. The address is actually imprinted on the letter in a binary-coded abbreviation of the original address.

Two techniques of coding the information on the envelopes are in experimental use at the present time. One is a system of printing dots on the envelopes. These dots are read by an optical reader. These dots may be of ordinary black ink, or they may be of fluorescent or phosphorescent ink. Each vertical column contains 2 sprocket dots, 1 parity dot, and 6 information dots. The column represents one alphameric character. Any number of characters from 1 to 20 is allowed. Postcards cannot be coded in this manner because of the lack of free space.

The alternate method of recording information on the envelopes is to spray a strip of magnetic material on the front of the envelope (or the address side of a postcard) and record serially on this with a magnetic head. The required area is greatly decreased in comparison with the optical method. It is possible, furthermore, to spray the magnetic coating so thinly that it does not obliterate writing which may be underneath. Several millivolts of signal are obtained despite the thinness of coating. Because letters contain objects such as coins, pencils, etc.

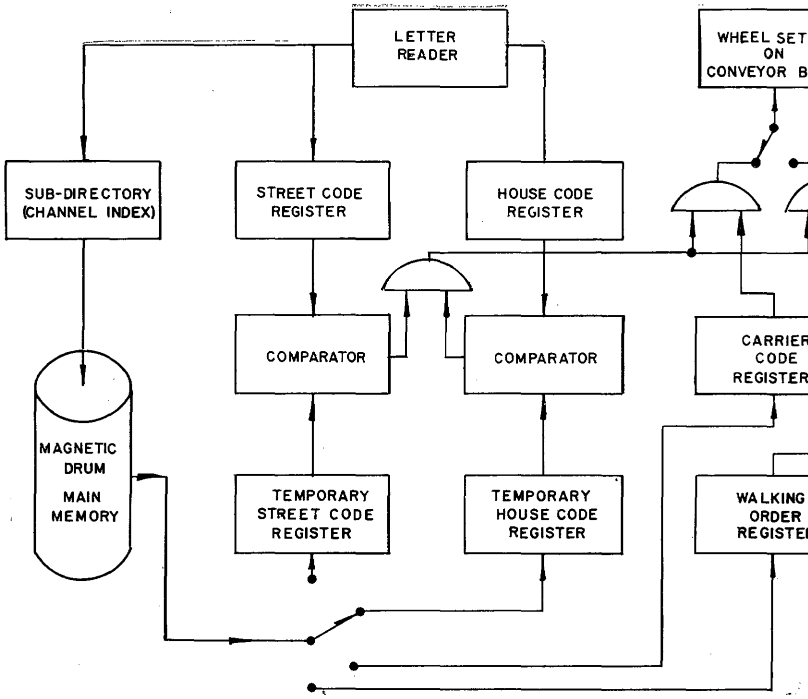


Fig. 4. General block diagram for serial magnetic drum directories

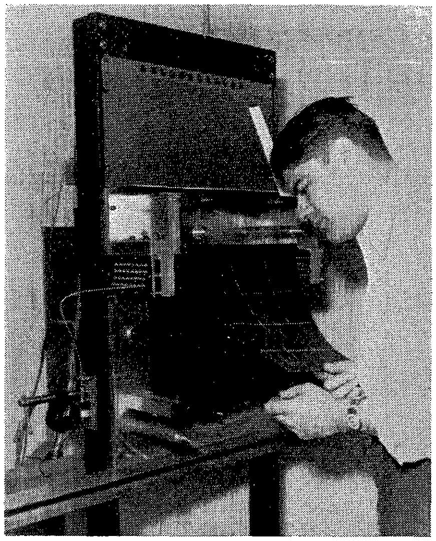


Fig. 5. Optical plate memory during building stage

they cannot be pressed into contact with a magnetic head by a roller. The solution to this problem is to suck the side of the envelope into the magnetic head by using vacuum.

It now appears that the magnetic technique will be the preferred method of recording on envelopes.

The type of information which is recorded on the envelope varies according to whether the letter is incoming or outgoing. Normally, if incoming and outgoing letters are separated at the code printing station, the full second and third lines of a normal address do not need to be coded on the letter. Thus, for incoming letters, only the second line of address needs to be carried on the envelope. Outgoing letters have a slightly different situation. If a letter is going to a city in which there are codesort machines, it may be desirable for both the second and third lines of address to be coded on to the envelope at the initiating post office. Coding the second line of address for an outgoing letter, however, takes an appreciably longer time, and the present emphasis in the Post Office system is on getting the outgoing mail out of the initiating post office as rapidly as possible. Because of this uncertainty, all the systems are being designed so that outgoing mail can be coded either with second and third lines of address or just the third line of address.

Machine Sorting to Walking Order

Machinery which sorts only to the carrier can easily be justified by itself. Sorting to the carrier is only part of the job, however, for the carrier himself must rearrange his mail in the order in which he walks his delivery route. If this very fine sort can be mechanized, the potential savings are much larger. A million dollars worth of machinery could be justified if each of Washington's 700 carriers were saved just one half hour per day. The detailed information about the walking order is so voluminous that it is not, in general, collected into a single directory, and this information for an entire Post Office becomes an impossible memory problem for one man. Machine memories of this capacity do exist, however, and this paper will present what is considered to be a reasonably economical method for mechanizing the complete sorting of mail down to the carriers' walking order.

The most direct method for sorting to the walking order would be to have one receptacle (bin) for each address in the city. For a city of 200,000 addresses, such as Washington, this number of automatically loaded bins is prohibitive because of the space and cost. It is, however, possible to obtain this fine a sort by passing this mail twice through a 1,000 bin distributor. There are several possible ways to organize this, but the most useful one for the purpose is a system which is called the "generalized walking order" method.

In this generalized walking order method, the first sort is made on the walking order and the second sort is made on the carrier number. This is equivalent to the well known technique of sorting International Business Machines Corporation (IBM) cards wherein the first sort is made on the least significant digit, the second sort on the next least significant digit, and so forth; the cards are finally in numerical order in such a system.

To illustrate, suppose that the memory has enough capacity to hold all of the walking order information. Entered in it, for instance, is the information that 200 First Street, N. W., is the 41st address (walking order no. 41) that carrier no. 112 delivers; that 201 First Street, N. W., is the 42nd address (walking order no. 42) delivered by carrier no. 112; etc. This kind of information is supplied about every address in the city. During the first pass, the letters are sorted on the walking order number. Thus, the first receptacle contains only those letters which are to be delivered to the first address on every carrier's route. In receptacle no. 2, there will be found only addresses which are the second stop on every carriers' route.

When all the mail has been run through the first pass, thus being sorted to a generalized walking order, all mail is removed from the conveyor belt, stacked, and run through the machine for a second pass. Mail belonging to walking order no. 1 is run through first and sorted to the appropriate carrier. Then, mail belonging to walking order no. 2 is sorted to the carrier. These letters fall on top of letters from walking order no. 1. When all mail has been run through this second pass, each carrier will have his own stack, arranged in the order in which he walks his route.

General Requirements for Directory

Processing Rate

The previous section of this paper has been devoted to a brief discussion of manual and automatic sorting methods. This section now discusses some of the general specifications that a suitable directory will probably be required to meet.

The necessary access time is related to the number of conveyor belts serviced and the average rate of letters per conveyor belts. Present conveyor belts seem to be limited to about ten letters per second. The number of conveyor belts serviced is related to the size of the Post Office as well as the directory capabilities.

A rough estimate for the number of conveyor belts necessary for a given Post Office is obtained by assuming that all the letters handled by the post office per day must be run completely through the sorting process in an 8-hour period. Thus, if one machine can completely sort ten letters per second, there will be about 250,000 letters per 8 hours; about 15 machines would be required to handle Washington's four million-letters-per-day volume. Chicago would require about 60 such conveyors.

A minimum directory rate of ten letters per second is therefore specified, with faster access systems being admissible.

Output Bits

The number of output bits from the directory should be somewhat greater than the number of bits required to dedescribe one out of the many receptacles on the conveyor belt.

The European machines, Transorma, the Bell machine, and the French machine, operate with about 300 output receptacles. This figure of 300 is dictated partly by the fact that these machines are directed by human memories rather than machine memories and are thus limited in the number of output points into which sorting can be accomplished. Nine bits output can handle 300 bins.

The Rabinow Distributor, using machine directories, will have upwards of 1,000 output receptacles, and provision is made for increasing this number to as high as 8,192 in the present designs. Thirteen bits will describe 8,192 bins.

Canadian designs (also using machine directories) are based on a small number of output receptacles whose contents are then resorted. The number of output receptacles in these machines is of the order of 30 to 50, which can be described with 6 bits.

Without discussing the pros and cons of whether it is better to sort mail by passing it many times through a small machine or few times through a big machine, suffice it to say that 14 or 15 bits should be more than adequate to allow complete flexibility in the organization of distributors. If the directory is to specify both the walking order number and the carrier numbers, there should be two 15 bit outputs available.

Input Bits

If both incoming and outgoing mail are sent through the system without presorting, both the second and third lines of an address on a normal envelope will need to be carried into the directory. One hundred bits will probably suffice to define this information using a moderately complex system of abbreviations. It is, however, expected that the mail will be pre-sorted in most cities into incoming, outgoing, and airmail groups. Approximately 64 bits suffice to process an incoming letter (abbreviated second line) and about 34 bits will describe an outgoing letter (third line of address). The following tentative requirements are the results of a thorough study of the Postal System by the staff of a project at the National Bureau of Standards.

Slot 1 (Outgo code)	Six characters total, Outgoing only 34 bits City: 4 characters of 6 bits State: 2 characters of 5 bits
Slot 2 (House code)	Six characters total, Incoming only 32 bits 2 characters of 4 bits 4 characters of 6 bits

Slot 3 (Street code)	Six characters total, Incoming only 32 bits
	4 characters of 6 bits for street;
	1 character of 5 bits for type;
	1 character of 3 bits for direction

Memory Capacity

The actual required capacity of the memory is a number whose discussion will make up much of the remainder of this paper. The required size of the memory is a function of the number of possible addresses which must be handled, and it is also an important function of the type of memory used. A frightening outside limit to the size of the memory could be given by dealing with New York City in the most naive possible way. For incoming mail, New York City has approximately one million addresses. Each address should be specified by 64 bits, and should be associated with a 15-bit walking order number and a 15-bit carrier number. The total number of bits for each entry will therefore be near 100. One hundred bits times one million is 100 million bits for the incoming memory. Any part of this must be available in 1/10 of a second. Taking Washington as a more average example, the incoming section of the memory would be 100 bits per entry times 200,000 addresses, or 20 million bits. The outgoing memory requirements are comparatively insignificant.

What methods can be used to reduce the necessary capacity? If the incoming mail in Washington was divided by presorting into 20 equal zones, then 20 memories of about one million bits each will be sufficient instead of one incoming memory of 20 million bits. This illustrates the rather obvious point that the memory size is dependent upon the "field of view" required. Instead of having the city divided into zones, which requires a complicated pre-sort, the mail can be sorted first to streets. The individual memories then only have to know what the carrier routes are along one street, or along only a few streets. This system is, in fact, the one used at Christmas time, and it allows the use of relatively untrained help. Another way of reducing the field of view is to first sort to the block numbers, regardless of the street. In general, any method which reduced the field of view requires multiple sorting of the letters. The Canadian system, which has only 30 to 50 output receptacles, works on this principle. Many small, fixed memories can be used, or a single, small, fast-reloading memory can be used.

This type of memory simplification using multiple sorting of the letters results in a great deal of extra handling of the mail by humans or by machines which may damage it. Manual handling will certainly delay the sorting.

Another general way to reduce the required memory size is to reduce the amount of information entered against each address. This can be done only because there has already been a great deal of effort by the Post Offices in organizing the addresses in towns into related groups. The most obvious first step in this direction is to take advantage of the "break" system which is used to describe the Post Office scheme to the present manual sorters. Basically, the break system sets up only the limits for a group of houses which are served by the same carrier. As an example, in looking at the first three columns of Table I, the first entry is interpreted as meaning that all the houses numbered 200 through 399 on First Street,

N. W., have the mail delivered by carrier no. 112; all those houses having numbers 400 through 599 are served by carrier no. 120.

This information can be entered into a machine memory in a number of forms. The particular form would be dictated by the specialized type of memory used. In columns 4, 5, and 6 of Table I under the heading, "Full parallel storage using break system," a method of entering this break information into a magnetic drum system has been illustrated. This is represented in strict parallel, using 32 parallel binary channels for the streeet code, 32 parallel binary channels for the house code and 15 parallel binary channels for the carrier number. The operation of such a memory would be as follows: The address on the letter abbreviated in the same form as the information on the drum, enters into a set of registers which is in continuous comparison with the parallel information on the magnetic drum. If, for example, the address on the letter was 252 First Street, N.W., the output signal designating carrier no. 112 would be delivered when the drum passed through the 400 entry in the street number section. It accomplished this by recognizing that the last entry on the drum had a street number less than 253 and that the next entry on the drum had a value (400) which was greater than the house number of the letter being sorted. It must have, in addition, recognized that the street name associated with this entry on the magnetic drum was identical with the street name on the letter.

The break system is plagued by what the Post Office calls "exceptions." These are single addressees whose mail is delivered by a different carrier than the rest of the addressees in the block. Usually these addressees are large mail users whose mail is delivered by a truck. Depending upon the type of memory system used, these exceptions range in nuisance value from mere annoyances up to real problems. It must be realized, however, that they do account for a large volume of the mail, and a rather larger share of sorting expenses can be written off to these particular numbers than can be justified for a single private house. In the examples given under the full parallel storage, an exception occurs at 901 Massachusetts Avenue. This is handled with no difficulty in the full parallel magnetic drum system by entering the exception ahead of the general break and requiring an exact comparison.

The break system, which has been discussed in the last two paragraphs, is a general method for reducing the amount of information which must be stored per address. More sophisticated methods of reducing the amount of necessary stored information will now be discussed. A study of the full parallel storage method reveals that the street designation is repeated a great many times. It should be clear that the street designations do not need to be continuously repeated for each break that occurs along a continuous street. Storage methods can be devised which will take advantage of this organization. This is called "Street Code Shift." Further reduction can be accomplished by observing that the breaks can be listed in always ascending order of block number. In order to take advantage of this high rate of redundancy, a technique called "differential" storage can be used. The term "additive" storage is also used sometimes. This technique will be described in detail in later sections of this report. Basically, the idea is simply to keep track of the house numbers in a separate house code register and merely to enter differences from one break to the next break. This technique can also be used to advantage in describing the output carrier code since the carrier numbers along any one street tend to be rather closely related. On the average, each break can be described in about 50 bits.

The break system tells only that the input address belongs to a particular part of a particular carrier's route; it does not yield the walking order number. This walking order number can be obtained by listing the house numbers which fall within the break. By listing these numbers in ascending walking order or by programming the computer to count in one out of several patterns, the number of bits needed to yield the walking order can be reduced to an average of about 7 bits per house number.

To summarize, then, if the techniques described are carried to their reasonable limits, each break will require about 50 bits, and each house number will require about 7 bits. Washington, with 11,000 breaks and 200,000 house numbers, will require about two million bits total.

Miscellaneous Requirements

The next important question to consider about the memory is how often it will have to be changed. Present estimates seem to be that approximately 10% of the entries are changed per year. This is being done on a continuous basis with the present manual sorting system by means of issued memoranda. This figure is low enough so that memories of the type normally considered nonerasable can be considered (such as wired matrices or various plug-board arrangements). The question of whether or not a memory should be electrically changeable does not revolve around the problem of making changes in the Post Office scheme so much as it does around the question of whether a particular memory should be put to multiple use, each use requiring the complete erasing of old data and entering of new data. This is, again, related to the question of whether a single sort is better than a multiple sort.

The cost of the memory must be related to the cost of the conveyor belt which it services. A round number estimate for the cost of one complete 1,000-receptacle conveyor belt, utilizing automatic letter insertion, is $300,000. A reasonably attractive price for the directory to go with one such conveyor belt would be $50,000. It has been shown in earlier parts of this paper that the memory capacity is going to vary almost directly with the number of addresses in a city. The price of the one conveyor belt does not vary in proportion to the number of output receptacles but increases at a slower rate, due to the fact that a good percentage of the automatic conveyor cost is spent in the automatic inserters, wheel setters, and other "overhead" equipment. The number of output receptacles on the conveyor belt can be increased merely by lengthening the system with relatively low cost units. It seems obvious, therefore, that the memory for the conveyor belts used in Chicago will cost a larger percentage of the total machine price than the memory used for a small town.

The reliability of such equipment is an important factor and should be considered along with the basic design. Like many other installations of modern machinery in various business, industrial, and government locations, this type of electronic machinery will be completely foreign to the present personnel. Although specialized personnel will doubtlessly be acquired, trained, and retained to perform service, the management is certain to look with jaundiced eye upon delays due to malfunctioning of the equipment which are above and beyond the delays normally encountered in doing the job in the old way. To some extent, the requirements on continued serviceability of equipment will be more severe during the initial phases of installation and operation than in later years after the equipment has well shown its worth and general serviceability.

Rather than putting letters into the wrong receptacle, the machine should

Table I. Drum Sorting to Carrier

Post Office Scheme			Full Parallel Storage Using Break System			Serial Methods		
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Street (N. W.)	House Number	Carrier Number	Street Code (32 Parallel Binary Channels)	House Code (32 Parallel Binary Channels)	Carrier Code (15 Parallel Binary Channels)	Full Serial, 1 Binary Channel	Serial, Street Code Shift, Differential House Code	Serial, Street Code Shift, Differential House Code, Hundreds-Units Shift in House Number, Differential Carrier Number
1st	200–300	112	1SNW	399	112	*	*	*
						1	1	1
						S	S	S
						N	N	N
						W	W	W
						@	@	@
						3	3	3
						9	9	9
						9	9	9
						¢	¢	¢
						1	1	1
						1	1	1
						2	2	2
	400–500	120	1SNW	599	120	*	@	@
						1	2	2
						S	0	
						N	0	¢
						W	¢	8
						@	1	
							2	
						5	0	
						9		
						9		
						¢		
						1		
						2		
						0		
	600–800	109	1SNW	899	109	*		
						1	@	@
						S	3	3
						N	0	¢
						W	0	—
						@		11
						8	¢	
						9		
						9		
						¢	1	
						1	0	
						0		
						9	9	
Nine similar entries stopping at Block 2,400, then skips to 5,000 for 3 blocks, then skips to 6,500								
	6,500–6,600	1,228	1SNW	6,699	1,228	1	@	@
						S	5	5
						N	8	9
						W	0	¢
						@	0	1
						6	¢	1
						9	1	1
						9	2	9
						¢	2	
						1	8	
						2		
						2		
						8		
Mass. Ave.	Unit–100	109	Mass A	199	109	*	*	*
						M	M	M
						A	A	A
						S	S	S
						S	S	S
						A	A	A
						N	N	N
						W	W	W
						@	@	@
							1	1
						0	9	9
							9	9
						0	¢	1
						0	1	¢
								—
						¢	0	1
						1	9	0
						0		1
						9		0

Table I. (cont'd.)

Post Office Scheme			Full Parallel Storage Using Break System			Serial Methods		
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Street (N. W.)	House Number	Carrier Number	Street Code (32 Parallel Binary Channels)	House Code (32 Parallel Binary Channels)	Carrier Code (15 Parallel Binary Channels)	Full Serial, 1 Binary Channel	Serial, Street Code Shift, Differential House Code	Serial, Street Code Shift, Differential House Code Hundreds-Units Shift in House Number, Differential Carrier Number
	200–300	111	Mass A	399	111	*	@	@
						M	2	2
							0	
						A	0	¢
						S		2
						S	¢	
						A	1	
						N	1	
						W	1	
						@		
						3		
						9		
						9		
						¢		
						1		
						1		
						1		
	400–600	174	Mass A	699	174	*	@	@
			NW			M	3	2
							0	¢
						A	0	
						S		1
						S	¢	6
						A	1	3
						N	7	
						W	4	
						@		
						6		
						9		
						9		
						¢		
						1		
						7		
						4		
Exception:	901	Firm	Mass A	901	3,000X	*	#	#
A. F. of L.		(3,000)	NW			M	9	3
Bldg.						A	0	0
						S	1	2
						S	¢	¢
							3	
						A	0	2
						N	0	8
						W	0	2
						@		6
						9		
						0		
						1		
						¢		
						3		
						0		
						0		
						0		
	700–1000	121	Mass A	1,099	121	*	@	@
			NW			M	9	
						A	8	9
						S	¢	8
						S		¢
						A	1	
						N	2	2
						W	1	8
						@		7
						1		9
						0		
						9		
						9		
						¢		
						1		
						2		
						1		

Note:
In column 7: * = street code coming.
@ = house code coming.
¢ = carrier code coming.
In column 8: * = street code coming.
@ = differential house code coming.
= exceptional house code coming.
Column 9 uses same symbols as column 8, plus:
¢ = add following number to carrier register.
¢̲ = subtract following number from carrier register.

be biased in favor of rejecting letters into the category which will be sorted by hand. If the first machine systems succeed in handling only 90% of the mail, the effort has been well worth while. One element acting as a check on the machine is that the carriers themselves can pick up mistakes rather readily; a man, finding a letter addressed to a point not on his route, can easily flip this letter into the hand sort category. This does not apply, however, to outgoing mail, since a long time may elapse before the error is detected. It seems reasonably possible to include more automatic checking features on outgoing mail than on incoming mail, since the total memory capacity required is considerably smaller for outgoing mail. Even for incoming mail, however, it would seem advisable to at least check that some kind of information was coming out of the computer.

In summary, this section discussed the general requirements for a memory suitable to direct letters to appropriate output receptacles in a post office letter conveyor system. The following summarizes the main directory requirements for sorting to the walking order.

1. Input information required to describe an incoming letter is 64 bits.

2. Input information required to describe an outgoing letter, is 34 bits.

3. Number of output bits is 15 for walking order number, 15 for carrier number.

4. Minimum average rate of memory look-up is 10 letters per second.

5. Memory capacity is about 50 bits per break, plus approximately seven bits per house address. Required capacity is strongly dependent on the street layout of the city.

Discussion of Drum Methods

Drum Sorting to Carrier

Thus far this paper has discussed the sorting techniques and the basic directory problems associated with the automatic sorting of mail. This section will go into more detail about the utilization of magnetic drums as a directory memory. Fig. 4 will serve as a general block diagram of a machine. It can be assumed that the drum will be loaded by a punched tape reader.

The general plan of attack is to locate the proper break first, (this gives the carrier number as a by-product) and then to generate the walking order number.

In Table I the first three columns show some more-or-less typical parts of the Post Office scheme for Washington, D. C. Columns 4, 5, and 6, under the heading "Full Parallel Storage Using Break System," show the same information organized suitably so that it can be entered into a magnetic drum directory. This particular system would require 79 parallel binary channels on the drum, and would require an amplifier for each channel. It will be seen that this parallel storage is not very efficient.

Table II. Memory Requirements for Drum Sorting to Carrier, Method of Col. 9, Table I

Type	Characters	Bits	Total Bits in Entry	No. of Times Entry is Repeated	No. of Bits per Entry Type
Street Entry (Slot 3)	1 order	4	36	One entry for each street in city	36(S)
	6 information	32			
House Entry (Derived from Slot 2)	1 order	4	16	One for each street	16(S)
	3 information (average)	12			
	1 order	4	8	One for each break (after the first) along the same street	8(B-S)
	1 information (average)	4			
Exceptional House Entry	1 order	4	16	(X)	16(X)
	3 information (average)	12			
Carrier # or Output Bin #	1 order	4	16	(S)	16(S)
	3 information (average)	12			
	1 order	4	8	(B-S+X)	8(B-S+X)
	1 information (average)	4			

S = No. of Streets
B = No. of Breaks
X = No. of Exceptions
# bits = 60(S) + 16(B) + 24(X)
For Washington, D. C. S = 3,600
B = 7,000
X = 4,000
# bits = 424,000

The operation of a system using full parallel storage is quite straightforward. The binary bits which represent the address of the letter to be sorted are stored in a 64 bit register and are continuously compared against the 64 parallel outputs of the drum which describe the street name and house number. Whenever the street number coming off the drum changes from less than the house number of the letter to greater than the house number, the signals from the carrier number channels will be read out of the directory as the applicable output. (The two street names must have also matched perfectly.)

The maximum number of bits required is 79 times the number of breaks plus exceptions. In Washington this amounts to approximately 800,000 bits. Much of the drum space is never used, however, because the average length of an address is less than 64 bits.

A considerable saving in reading head amplifiers and drum space can be made by using a strict serial system. The "Full Serial" system will be taken up first. The "Walking Order Register" does not apply. The coding for a full serial system is illustrated in Table I under the column, "Full Serial." In this system one out of many channels would have to be selected by a switching matrix. This selection can probably be based on the street name. Certain symbols must be used in a serial system to designate meanings of the numbers which follow this symbol. These symbols are given at the end of Table I. These numbers are recorded in the appropriate register and the end comparison occurs in a manner similar to that described for the full parallel storage.

The number of bits actually used in this full serial system is slightly more than the number used in the full parallel system, because of the commands. The total required drum space is somewhat less, however, because advantage can be taken of those addresses which require less than the maximum number of bits.

One of the problems inherent in any serial system is the selection of the proper drum channel. It is assumed that all the breaks on a particular street will be entered one right after the other along the same channel, though this is not necessarily required in the full serial system. A number of streets can probably be accommodated on any one channel. The street name on the letter to be sorted can be looked up in a subdirectory which will give as its output the number of the drum channel upon which all the breaks of this street are located. A general solution for this subdirectory would be to write down every street code, using 32 bits each, together with the appropriate channel number of the main memory.

A cheaper solution is to use an arithmetic code for the alphabetic characters which are used in describing the street name. With an arithmetic code, only the last street name appearing on each drum channel need be written in the subdirectory, and the proper channel number for a particular street can be obtained by arithmetic comparison. For a 320-channel drum, the total subdirectory storage is (32 plus 9)×320. This is only about 13,000 bits.

If it turns out to be important to end one channel and start the next channel with the same street (usually only for long streets), this can be taken care of by making the entry on the channel selector directory include not only the street name, but also as much of the house number as is necessary to describe the dichotomy.

The following is a detailed discussion of two major techniques which help greatly to reduce the volume of information to be stored. One of these is called the "shift" technique, and the other is called the "differential storage" technique. The differential storage technique is also sometimes called derivative or additive storage. These techniques can both be understood by comparing columns 7 and 8 of Table I. Examination of the full serial system shows that each of the break entries repeat much of the information that is already contained in the previous break. The street name, for instance, is repeated each time. In column 8, however, the street name shift technique is used. The first time (on any revolution of the drum) that a particular street name appears, this street name is preceded by an asterisk (*) which instructs the machine to store this street name in a register and keep it. The contents of this street name register are changed when the next asterisk appears, and the street name following that asterisk is substituted for the previous information.

The differential storage technique is related to the shift technique in that the first house number entry along each street is stored exactly as written down on the drum. Subsequent numbers are written on the drum, however, not as full numbers, but as the difference between the previous break number and the present break number. Every number which is preceded by the symbol (@) is, therefore, added to the accumulated total which is in the temporary house code register.

Table III. Two Methods for Drum Sorting to Walking Order

Post Office Scheme Extended				Information on Drum	
(1)	(2)	(3)	(4)	(5)	(6)
Street Code (N. W.)	House Code	Carrier Code	Cumulative Carrier Walking Order	One Entry per Address System	Limit System
1st				*	*
				1	1
				S	S
				N	N
				W	W
				@	@
				2	2
				¢	¢
		112		1	1
				1	1
				2	2
				$	$
				4	4
				0	0
				%	&
	200		40	0	0
				0	4
				0	
	201		41	1	
				0	
	202		42	2	
				0	
	203		43	3	
				0	
	204		44	4	%
				6	6
	268		45	8	8
				@	@
				1	1
				%	& even
				0	0
	300		46	0	2
				0	
	302		47	2	& odd
				0	0
	301		48	1	3
				0	
	303		49	3	
New break	400			@	@
				1	1
				¢	¢
		120			
				8	8
				$	$
			15	1	1
				5	5

Meaning of Symbols

*	Street code coming
@	Differential house code coming
¢	Plus differential carrier code coming
¢	Minus differential carrier code coming
$	Starting number coming for walking order
%	Paired digits of house code coming
&	Count by ones from 00 up to the limit given by the following two digits
&even	Count by twos from 00 up to limit
&odd	Count by twos from 01 up to limit

Follow through the activity in column 8. A letter is addressed to 652 First Street, N. W. When the computer senses the asterisk symbol, it clears its temporary street name register and writes in 1SNW, which is the abbreviation for First Street, N. W. It notes that this street name matches with the name on the letter to be sorted. It also at this time clears the temporary house number register. When the symbol "@" is sensed, 399 is added to the contents of the temporary house number register. The result is 399. Machine comparison shows that 399 is less than the desired street number 652. It therefore ignores the carrier symbol and takes no further action until the next break entry. After recognizing the next "@" it adds 200 to the previous contents of the temporary house number register. The additive result, 599, is stored back in the temporary house number register. The machine notes that 599 is still less than 652 and so ignores the carrier number on the drum. Three hundred more is added to the temporary house number register by the next entry, making the accumulated contents of the temporary house number register equal to 899. The machine now notes that 899 is a larger number than 652 and therefore picks up the carrier number information ¢109. This carrier number information is sent to the conveyor belt and used to set the correct wheels on the distributor.

Many systems are possible for handling exceptions. They differ only in detail. The exceptional case in column 8, illustrated by the American Federation of Labor (A.F. of L.) Building at 901 Massachusetts Avenue, is entered before the general break and uses a special command symbol "#." Exact match is required, not just a change in sign of the comparator output.

In column 9, the general technique for eliminating redundant information is carried still further. A few of the unnecessary zeros in the differential house numbers can be eliminated by using a hundreds-units shift. Except for the first entry along a street, the symbol "@" now means that the following number is to be multiplied by 100 and added into the temporary house number register. Exceptions are handled in the same way as in column 8. In column 9, advantage has been taken of the fact that many of the carrier numbers lie close to each other, and the differential storage is used for the carrier numbers as well as the house numbers. Since an arithmetic code is used for the alphabet anyhow, and since the street names will be arranged on the drum in alphabetic order, a differential storage might be used for the street names as well. This, in general, does not save much memory space, however, because the street names are not written down very often when the street name shift is used. Differential storage for the street names is not illustrated in any one of the columns. The commands must occasionally be used to either add or subtract values to the various registers. When this is required, the additional symbols plus and minus are used following the command symbol and this denotes the sign of the command.

By the use of these techniques the number of bits required to sort (to carrier) the incoming mail of Washington, D. C. can be reduced from about one million to about 500,000.

Table II summarizes the memory requirements for drum sorting to carrier; using the method of column 9, Table I.

Drum Sorting to Walking Order of Carrier

Thus far this section has discussed some of the detailed methods of entering information into a directory which is to be used only for sorting to the carrier. The following paragraphs describe a detailed method which makes possible a reasonably economical mechanization of sorting to the carriers' walking order. The potential savings to the Post Office are very large, here. As has been discussed before, this is a considerable extension of the information processing requirements since, essentially, every address in the city must be noted. This could, as a limit, mean that ten times as much information must be stored to sort to walking order as must be stored to sort to the carrier.

Table IV. Memory Requirements for Drum Sorting to Walking Order

Code: B = Number of Breaks; A = Number of Addresses

One entry per address system:
(Column 5, Table II)
Bits = 16 (no. of $ commands) = 16 (no. of breaks)
plus 4 (no. of % commands) = 4 (no. of blocks)
plus 8 (no. of @ commands) = 8 (no. of blocks)
plus 8 (no. of paired digits = 8 (no. of addresses)
Estimate: Number of Blocks is twice the number of Breaks = 2B
No. of bits 8A + 40B
For Washington
A = 200,000
B = 7,000 (excluding exceptions)
No. of bits for Washington = 1,880,000
(To this number must be added the number of bits for sorting to carrier)
Limit system:
(Column 6, Table II)
Bits = 16 (no. of $ commands) = 16B
plus 12 (no. of & commands) = 12 (no. of sequential runs)
plus 12 (no. of % commands) = 12 (no. of outhouses)
plus 8 (no. of @ commands) = 8 (no. of blocks)
Estimates:
No. of sequential runs = 2B
No. of outhouses = 1B
No. of blocks = 2B
Bits = 16B + 24B + 12B + 16B = 68B
For Washington, B = 7,000
Bits = 476,000
(To this number must be added the number of bits for sorting to carrier)

Luckily, however, techniques can be utilized which greatly reduce the required volume of storage.

Columns 1, 2, and 3 in Table III, labeled "Post Office Scheme Extended," show an imaginary break with the carriers' walking order written out. A normal walking order has generally some regular pattern, such as taking all the house numbers in numerical order with a break, or such as taking first all the odd numbered houses and then all the even numbered houses. There are, however, a few houses which fall out of any pattern, and these are designated as "outhouses." The carrier walking order shown has been chosen to give a reasonably difficult example of walking order.

Column 1 of Table III gives the street code, column 2 the house code, and column 3 the carrier code. Column 4 shows where these houses are located with respect to the rest of the carrier's route, for it is assumed that this break is somewhere along into the middle of his route. Thus, house number 200 will be the 40th house which he delivers, 201 will be the 41st, 202 will be the 42nd, etc.

Two different serial methods of sorting to the walking order are shown. Column 5 illustrates the "One Entry per Address" system, and column 6 shows the "Limit" system. The column 5 system is best adapted when the walking order shows very little pattern, and the "Limit System" is best when there are long runs of related addresses.

The "One Entry per Address" system will be discussed first. Column 5 lists the information that would be listed in a magnetic drum against this particular illustrated break. All of the information necessary to sort to the carrier must be contained in the entry as well as the information relating to the walking order. All the systems previously discussed for sorting to carrier are applicable for this part of the job; here the system used is shown in column 9, Table I, for designating the particular break which we are illustrating. Thus, the characters after the asterisk describe First Street, N. W., the numbers after the "@" denote which block is being dealt with, and the number after the "¢" symbol gives the carrier number.

A new register is now introduced into the machinery. This will be called the walking order register. A new symbol, the number sign "#," means that the number following this symbol is to be cleared into the walking order register. In this case the number 40 is cleared into the register. Following this is the command, "%," which means that the following numbers should be taken in pairs and cleared into the tens and units column of the temporary house number register. Note that these pairs of numbers are the last two digits of the house numbers in this break. They are listed in the order in which they are delivered; thus 00 is the first house to be delivered, 01 is the next house to be delivered, 03 is the next house to be delivered, etc. Each time a pair of these numbers passes by the reading head on the drum, the contents of the walking order register are increased by unity.

Comparison of the house number being sorted is being made continuously against the house numbers which are coming off the drum. When the two match exactly, the number contained in the walking order register is read out and used to direct the letter to its proper receptacle.

Note that the pairs of numbers that follow the symbol "%" are to be cleared into the units and tens positions of the temporary house number register. The differential system could be used in this instance, but the clearing system results in more flexibility at not much extra cost.

A suitable organization of distributors to accomplish the sorting to walking order was discussed earlier. The results of that discussion were that the best system is to have the mail sorted first to a generalized walking order and then sorted to the carrier. This is illustrated again. The first time the mail is passed through the distributor, the contents of the walking order register determine the conveyor receptacle to which a letter is distributed. For instance, during the first pass, letters addressed to 300, 301, and 302 will be delivered respectively to receptacle number 46, 48, and 47. During the second pass, the contents of the carrier code register will be read out as the significant output information, and the contents of the walking order register will be ignored. This machine change can be accomplished with a simple toggle switch. During the second pass of the exemplified mail, the letter addressed to 300 will be sorted first and will be dropped in receptacle 112, which corresponds to carrier no. 112. The next one of these three particular letters to be sorted will be the letter addressed to 302, and this will be dropped also in receptacle 112 on top of the letter addressed to 300. The next one of these letters to be sorted will be the letter addressed to 301. This will also be dropped into the 112 receptacle, and it will lie on top of the other two letters. It will be seen that these letters are now arranged in the proper walking order for carrier 112.

The Limit system shown in column 6 of Table III is a much more compact notation than the One Entry per Break system of column 5. The Limit system relies for its efficiency on having most of the addresses falling in patterns. The four most useful patterns are:

1. Start from 00 and count by twos up to limit Q_1
2. Start from 01 and count by twos up to limit Q_2
3. Start from Q_3 and count down by twos until limit 00 is reached.
4. Start from Q_4 and count down by twos until 01 is reached.

Outhouses (houses out of pattern) are preceded by the "%" command.

The memory requirements of both walking order systems are summarized in Table IV.

A final decision as to what particular system to use must await detailed examination of the carriers' routes, but at present it seems that some form of Limit system will be satisfactory.

Alternate Memory Systems

General

This paper has thus far discussed the application of magnetic drums to the large scale sorting problem of the Post Office. It will now recapitulate the particular advantages of the magnetic drum for Post Office sorting and then go on to discuss several other techniques which have also been seriously considered for this application.

Perhaps the two most important advantages of magnetic drums are the fact that they combine an extremely low cost per bit together with an access time which falls exactly in the range which is best suited for Post Office directory work. For clocked drums using noncontacting heads, the price for an entire drum system (including access circuitry and reading heads) is about 2¢ to 5¢ per bit. Nonclocked magnetic drum systems in which the heads are in contact with the surface of the drum can be obtained for a price of as low as 2/10 of 1¢ per bit. An example of the latter technique is the high density drum developed by the Laboratory for Electronics (L.F.E.) and installed in the Diana computer. The rotation speeds of the non-L.F.E. and contacting drums vary from 20,000 rpm to several hundred rpm. The speed of the present L.F.E. high-density drum is 180 rpm. The company is expecting to be able to raise this to about 500 rpm.

Another advantage of the magnetic drum is that it can be written into and changed at high speeds by electrical means. This is not an absolute "must" but it gives an advantage to any directory using such storage in that techniques requiring quick reloading can be used; furthermore, particularly during the experimental stage of the automation program, complete sorting techniques can be comprehensively revised or completely changed with relatively little difficulty. In other words, electrically changeable memories, particularly when used in conjunction with computer-like control equipment, lend a flexibility to the users' operation that no volume of permanent storage can equal.

The fact that the memory stays correctly written even after a power failure can certainly be construed to be an advantage, although a memory which does not possess this property is not necessarily ruled out of consideration.

The prime disadvantage of magnetic drums is that they have mechanically moving parts. Adjustments and physical repairs must, therefore be sometimes made, no matter how few and far between. Another disadvantage sometimes cited against magnetic drums is that they require a great many expensive heads, reading amplifiers, and complex timing equipment. It is true that many reading heads are required, but if a completely serial system is used employing a mechanical switching matrix, one rewrite circuit and one timing channel, the complexity and expense is reduced to a low value.

There are, of course, many other types of storage which are used in and around digital computers. Most of these are not well suited to the Post Office problem either because of their high cost per bit or because of their long access time. The coincident current magnetic core memory, for example, is a fine, reliable, high-speed technique, but the cost at present is approximately 50¢ a bit. Magnetic tapes are an example of a digital technique which has too long an access time. One digital technique which is just becoming a commercial reality is the magnetostrictive delay line. The price of this type of storage, including the recirculation amplifiers, is approximately 10¢ a bit at present. The fact that a power failure can erase this memory does not rule out such a technique in the author's opinion, because delay-line memories can be loaded quickly and accurately by tape inputs. The magnetostrictive technique is, in fact, very attractive, and if the price can be brought down by another factor of 5, would become competitive with the magnetic drum.

Optical storage on a rotating disk or drum has been seriously considered by at least one laboratory for the Post Office application. Its main advantage is that it should be very cheap. Logically, it is best operated in a serial mode similar to that of the magnetic drum; however, the cost of the memory in such a directory is not the major expense, being overshadowed by logical circuitry and input-output. Because of this, and because the magnetic drum allows electronic writing and changes, the laboratory has chosen to apply magnetic storage rather than optical drum storage to the large scale processing problem.

Instead of viewing the directory as a computer, one can view it as a switching network. There are, in the computer art, a great many types of gating techniques which are well understood and which are completely electrical in their operation. Outstanding among these techniques are the methods of diode gating and magnetic core gating. The diode gating technique is particularly attractive in those areas where direct-coupled high-impedance circuits are required. The magnetic technique is particularly attractive for those applications requiring enormous numbers of gates, where a-c coupling is permissible, and where long life and low cost are particularly essential. Both the diode and the magnetic gating techniques can be applied to the Post Office problem, but because of the basically longer life and lower cost of the magnetic technique, this laboratory has chosen to investigate the magnetic method rather than the diode method.

Directories based on the use of various types of gating principles characteristically have the memorized information written into the memory in a fairly permanent form which can not be changed electrically. In the diode-gated or magnetic-gated directories the memorized information is in the form of wires which are more or less permanently connected in characteristic paths. The diodes and magnetic cores act only as switches which select the desired path.

The "Billboard Magnetic Core Directory" is the latest of a long series of various systems using magnetic cores which have been studied at the Rabinow Engineering Company. The basic element in all the systems is a single square loop magnetic core, (preferably a ferrite core because of its cheapness) which will have a number of wires going through it. Output is obtained from this core only when every one of the inhibiting currents is removed and when an a-c drive current is applied. "And" gating can be accomplished by controlling the inhibitory wires, and "or" gating can be accomplished by adding more a-c drive wires.

It is proposed that physically, this type of directory be constructed on a large flat surface area, such as the wall of a room. Thus the name "billboard." The entire scheme of the city will appear on the wall, spelled out clearly in Arabic numerals and English alphabet. Each number or letter will be fixed to a tiny printed circuit board (called a "key") which can be independently slid into place. The key has a front surface, perpendicular to the printed circuit surface on which the symbol appears. Only 36 different types of these keys are needed. The insertion or "writing in" of information into this directory is accomplished by plugging in the alpha-numeric keys which spell out the break in the Post Office scheme.

The billboard magnetic core memory has the following advantages:

1. No moving parts.
2. Extremely high speed, allowing it to service many distributors.
3. Readily observable status board.
4. Changes are made simply and directly by nonspecialized personnel.
5. Changes can be made while most of directory is in operation.
6. Uses only passive components of high stability.

Some disadvantages of this technique follow:

1. Not electrically changeable, and so cannot be used with distribution systems requiring quick reloading.
2. Requires large display area.
3. Initial cost fairly high.

Optical Plate Memory

Historically, the "optical plate memory" was developed by the Rabinow Engineering Company originally as an inexpensive directory for sorting to the carrier only. Although its capacity is too small to be practical for sorting to the walking order, the principles should have considerable application to various types of sorting and translation problems.

From the application point of view, the optical plate memory is a fixed function table, translating from 64 bits input to 12 bits output. The translation time is 50 milliseconds.

The main component of this type of directory is a box having a large number of thin, metal plates which are stacked together. Each one of these plates has two binary positions, either left or right. Plates move only 1/16 of an inch. On one side of the box is a large lamp and on the other side of the box are a number of photocells. The sorting scheme is inserted into the memory by punching holes into the plates in many optical paths. Output information is obtained from the photocells. See Fig. 5.

The input information, which is the abbreviated address on the letter, is presented in binary form to solenoids which push the plates right or left. If there is a maximum of 64 bits of information on the letter, there are 64 plates in the memory stack. If there are 12 output bits required, then there will be 12 output photocells. The manner in which the sorting scheme is inserted is as follows: 64 bits of input information corresponding to a particular address are fed to the plate solenoids. Optical paths are then manually punched through the plates in such a way that the particular desired photocells are illuminated. The next time this particular arrangement of plates occurs, due to a letter going through the system which has that address, the same photocells will be illuminated as had optical paths punched through to them during the write-in operation. In the coding of a different address, completely different optical paths are punched. In general, no two codings ever use any part of the same optical path. The memory capacity of this type of directory is given by the area per plate divided by the area per hole times the number of plates. This resulting number should be divided by two because it takes two hole positions to define one binary number.

The first model constructed of this type of memory has approximately 40,000 holes per plate. Each plate is approximately one foot square, and there are 64 plates. This makes an approximate capacity of 20,000 bits per plate, and the total capacity is about 1,300,000 bits. The memory can be operated at better than 10 cycles per second in its present form, since all of the 64 plates are completely moved and settled down in 50 milliseconds. The holes are photoetched through the metal insuring perfect hole alignment, and then covered with paint before assembly. The writing-in operation is performed by pushing a long steel needle through the desired optical path; the paint is brittle and "pops out" cleanly.

A way of describing the logical behavior of the optical plate memory is to say that 20,000 "and" gates are available, each one with 64 binary inputs. These 20,000 "and" gates are grouped equally into twelve "or" gates, each with 1,666 inputs. This means that the equipment is a great deal more powerful logically than a memory which simply stores 1,300,000 binary digits.

The cost of this large switching matrix is less than 1/10 of a cent per bit. Furthermore, no equipment other than a steel needle is necessary to write information into the directory.

Changes are made by putting a dot of paint over one hole in the unwanted optical path and then punching a new optical path.

The main disadvantage of the optical plate memory is that it has moving parts and therefore requires at least occasional servicing. For some applications the mechanical rather than electronic writing will be a disadvantage.

The primary point to remember about this device is that extremely sophisticated answers can be obtained very cheaply, without using any logical circuitry whatsoever.

References

1. Coding Problems Related to the Electronic Mail Handling System, **M. Levy, A. Barszcewski.** *1957 IRE National Convention Record*, Institute of Radio Engineers, New York, N. Y., pt. 6, p. 157.

2. The Mechanization of Letter Mail Sorting, **I. Rotkin.** *Proceedings*, 1957 Eastern Joint Computer Conference, AIEE Special Publication T-92, p. 54.

3. Automatic Machine for Sorting Letter Mail. *National Bureau of Standards Technical News Bulletin*, Washington, D. C., vol. 42, no. 8, Aug. 1958.

The Logical Design of CG24

G. P. DINNEEN I. L. LEBOW I. S. REED

THIS PAPER discusses some features of the logical design of the CG*24* computer with emphasis on the techniques used in the logical design rather than on the particular characteristics of the computer itself.

Often the phrase logical design of a digital system is used to describe the detailed configuration of logical circuits, either schematically in block diagram form or algebraically in the form of logical equations. Here the term, logical design, is used to describe the system operation in terms of the flow of information or the transfer of information from one register to another. Such a description has two significant advantages. First, it provides a rather concise picture by which one may characterize the over-all structure of a system without the confusion introduced by detailed considerations and second, while relatively independent of detailed circuit configurations, it can be translated easily into equipment by a circuit engineer.

CG*24*

CG*24*[1,2] is a general-purpose real-time digital computer. It is general purpose in that it is a stored program machine with a fairly extensive list of single address instructions and conventional terminal equipment. In addition, it has a data input from a radar receiver and a data output to the radar antenna. Hence it may be used to process radar data in real time and also to direct the position of the radar antenna. The computer is entirely solid state with ferrite core storage and transistor and crystal diode logical and memory drive circuits. Its principal characteristics are shown in Table I. From this table it is readily observed that it is in the class of current commercial scientific machines of the International Business Machines Corporation (IBM) *704* type. It has, however, achieved its operational speed with slower arithmetic circuits than most other machines of this class. This was accomplished by introducing a good deal of parallel operation. Nevertheless, the over-all power requirement of 4.5 kw, about 2 kw of which is for the cathode-ray tube display units, is modest and the physical size is small as seen in Fig. 1.

The computer has been installed and has been operating at a Massachusetts Institute of Technology (M.I.T) Lincoln Laboratory field station since May 1958.

Registers and Transfers

A digital machine is composed of a number of registers with certain allowed interactions between them. Basically, a register R is a set of n bistable elements R_i, $i=0, 1, \ldots, n-1$. Such a register may contain 2^n different numbers. Let (R) designate the contents of register R. If S is another n bit register in the machine, then the simplest interaction between registers R and S is the transfer of the number in register R to register S or in symbols $(R)\rightarrow S$, which signifies that after the interaction (transfer) has occurred, the contents of register S are identical to those of register R before the interaction. Register R is unaffected by the transfer.

The class of registers in the machine is enlarged by considering as registers parts of registers, functions of registers, and functions of several registers. Thus, designate part of register R as its address part, $Ad[R]$, and consider this as a register. The complement of a register $\bar{R}$ is considered to be a register. The algebraic sum of two registers R and S, $(R)+(S)$, is considered to be a register. Thus the class of generalized registers of a machine contains not only the registers themselves but also all desired functions of these types.

The class of transfers is enlarged by admitting conditional as well as unconditional transfers. Thus if R, S and T are registers and λ is a two valued function, then

$$\lambda(R)+\lambda'(S)\rightarrow T$$

is a conditional transfer stating that if the function λ is true then the contents of R are transferred to T while if λ is false the contents of S are transferred to T.

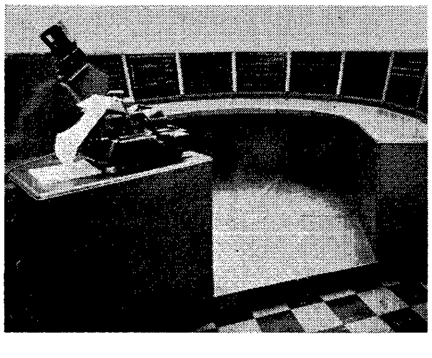


Fig. 1. CG24

Table I. CG24 Computer Characteristics

General
Application: General-purpose plus real-time control.
Timing: Synchronous.
Operation: Sequential, subject to self-modification.

Numerical System
Internal number system: 27-bit binary words, including two parity bits, with provision for double accuracy computation.
Single-address instructions.
Fixed-point arithmetic system, with a programed floating point subroutine.

Arithmetic Unit
Addition time: 24 μsec (including memory access)
Multiplication time: 84 μsec (including memory access)
Division time: 84 μsec (including memory access)
Square root time: 300 μsec (including memory access)
Pulse repetition frequency: 330 kilocycles per second.

Storage System
8,192 words, coincident current magnetic cores, 12-μs cycle time.

Terminal Equipment
Flexowriter providing 2-way communication between operator and machine.
Cathode-ray tube displays: Alpha-numeric display of the contents of up to 192 memory registers.
High-speed photoelectric tape reader.
Three input registers for transfer of real-time data to the internal memory.
Two servosystems for controlling two independent shaft positions.

Description of a Computer by Generalized Transfers

The operation of a digital system may be completely described by sequences of generalized transfers of the type defined in the previous section. For the description of a general purpose computer one must have sequences of generalized transfers which provide the following:

1. An algorithm for the performance of each operation or instruction in the repertory of the machine,
2. A method of executing a sequence of such instructions; i.e., a program, and
3. A method of operating a control unit to implement steps 1 and 2.

To design a digital computer to meet a certain set of operational requirements, one must implement the necessary generalized transfers for these three functions subject to the constraints imposed by the available hardware.

The first two of these requirements define the arithmetic and data processing capabilities of the computer. The third

G. P. DINNEEN, I. L. LEBOW and I. S. REED are with the Massachusetts Institute of Technology (M.I.T.) Lincoln Laboratory, Lexington, Mass.

The work reported here was performed at Lincoln Laboratory, a technical center operated by M.I.T. with the joint support of the United States Army, Navy, and Air Force.

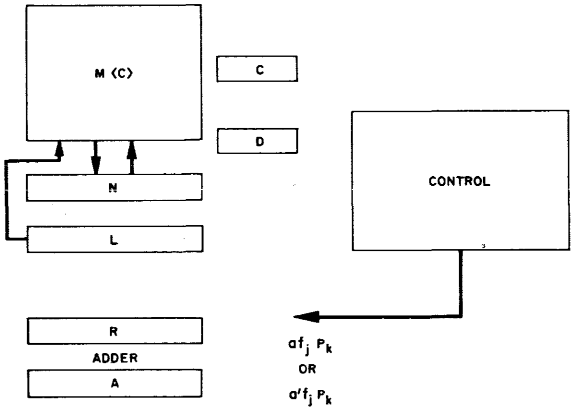


Fig. 2 (left). Simplified block diagram

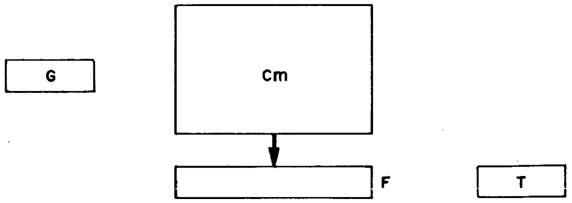


Fig. 3 (right). Control block diagram

requirement; i.e. control, is solely to implement the other requirements. Look first at the control and how it affects the rest of the computer. The control unit of any machine (regardless of physical implementation) may be thought of as having a finite number of configurations or states. Each of these states defines a certain set of generalized transfers to be performed in the machine. A machine is made to perform a given instruction by constraining the control unit to cycle through a definite sequence of states which specify uniquely the sequences of generalized transfers necessary for the execution of that instruction. More precisely, let f_i be a Boolean function which is true if and only if the control is in its ith state. The relationship

$|f_i|:(A)\rightarrow B,\ (C)\rightarrow D$

signifies that the contents of register A at the beginning of the time interval during which f_i is true are to appear in register B by the end of that interval. Similarly the contents of register C are transferred to register D during the same interval. It is easily seen that in this way a sequence of control states $\{f_i\}$ causes a series of generalized transfers to occur which can provide the means of executing any instruction or a sequence of such instructions. Furthermore, the necessary transfers for implementation of the control function itself may be described in this way.

Note even at this point that this method of description need not presupppose detailed knowledge of the circuits to be employed in the construction of a system. The sequences of states $\{f_j\}$ may be set up either synchronously or asynchronously. The indicated transfers $(A)\rightarrow B$ and $(C)\rightarrow D$ may occur at the same time or at different times; they may be performed serially or in parallel. All that is demanded is that they be concluded by the end of the indicated time interval.

Application to CG*24*

The basic notions of the previous sections were used in the design of CG*24*. The hardware upon which the design was to be based consisted of:

1. A coincident current magnetic core memory driven by transistors with a 12-microsecond (μsec) cycle time, this time being divided roughly into a 3-μsec read interval, a 3-μsec write interval and two 3-μsec post-write-disturb intervals (the post-write-disturb intervals were later eliminated but the cycle time remained at 12 μsec).
2. Transistor and diode logical circuitry capable of operation under load conditions at clock rates up to about 0.5 megacycle (mc).

The operational requirements indicated that the following computer characteristics were acceptable:

1. Addition time of 24 μsec (including memory access).
2. 8,000 words of core memory for storage of instructions and constants.
3. Single address structure for instructions.
4. Fixed point arithmetic with a 25-bit word.
5. 5 index registers.

On the basis of the circuit characteristics and operational requirements the following timing was established:

1. The entire machine (except perhaps for some terminal equipment) was to be synchronous with the core memory. The 12-μsec memory cycle interval was divided into four 3-μsec pulse intervals labelled P_1, P_2, P_3, P_4.
2. No flip-flop was to be required to change its state at a rate faster than 333 kc (once every 3 μsec).
3. The full 25-bit add time (exclusive of memory access) was to be less than 12 μsec.
4. The control unit was to change its state once every memory cycle. Hence the control state f_i was subdivided into the four substates, f_iP_j, $j=1, 2, 3, 4$, which would determine the machine transfers.

Having established the basic properties of the computer together with its elementary timing, the design techniques may now be applied to the actual machine. First the functions of a few registers of the machine will be detailed. Let M designate the memory of the computer and let C designate the memory address register. The symbol $M<C>$ means the particular memory register determined by the address in the C register. Let N and L designate respectively the memory output and input buffer registers. Let R and A designate two arithmetic registers, the operand storage register and the accumulator respectively. Addition is performed between numbers in R and A. Finally let D represent the program counter. For the present, the operation of the control unit will not be specified. Assume that it changes state (from f_i to f_j. etc.) in accordance with the previous description. For convenience, the control state will be designated by an additional function. Let a be a Boolean function which when true indicates that the current memory cycle is one in which a new instruction is being obtained from memory. Then the control substates will be designated by af_iP_j or $a'f_iP_j$. The machine as so far defined is shown in block diagram form in Fig. 2.

Some Typical Instructions

To describe some of the computer operation, consider first the instruction "add X" which signifies that the contents of register X in memory are to be added to the contents of A, the result being stored in A. The first memory cycle of operation, designated by af_2, is the one in which the instruction is obtained from memory. The following transfers occur during this cycle:

$|af_2P_1|: (M<C>)\rightarrow N,\ (C)\rightarrow D$

$|af_2P_2|: (N)\rightarrow M<C>,\ (I[N])\rightarrow$control unit

$|af_2P_3|: (D)+1\rightarrow D$

$|af_2P_4|: (Ad[N])\rightarrow C$

Assume at the start that the control unit is in state af_2 and that the C register contains the address at which the instruction to be interpreted is stored. The P_1 and P_2 intervals coincide with the memory read and write periods. During P_1 the contents of the selected memory register are transferred to the output buffer N and at P_2 this same word is rewritten in memory. This, of course, is necessitated by the destructive character of the read process. During P_2 the instruction code section of the word is transferred to the control unit. During P_1, the current address in C is stored in D and is counted up by 1 during P_3. This augmented address is the address of the next instruction in the absence of a jump instruction. Finally during P_4 the address part of the word just obtained (the address X) is transferred to C in preparation for the next cycle.

Assume that the control changes to state $a'f_1$ (the control state representing addition) which sets up the following sequence of transfers.

$|a'f_1P_1|: (M<C>)\rightarrow N$

$|a'f_1P_2|: (N)\rightarrow M<C>, (N)\rightarrow R$

$|a'f_1P_3|:$

$|a'f_1P_4|: (A)+(R)\rightarrow A, (D)\rightarrow C$

The memory operation as before occurs during P_1 and P_2 and the word so obtained is transferred to R during P_2. During P_4 the indicated sum is taken and stored in A. (The addition time of the computer is actually 3 μsec.) During this final interval the address of the next instruction stored in D is transferred to C and the computer is prepared to interpret the next instruction.

As another example, consider the instruction "store in X" which requires that the contents of the accumulator A be stored in register X. The first cycle of operation during which the instruction is obtained is designated by af_2 as for addition.

The second cycle is as follows:

$|a'f_{24}P_1|: (M<C>)\rightarrow N, (A)\rightarrow L$

$|a'f_{24}P_2|: (L)\rightarrow M<C>$

$|a'f_{24}P_3|:$

$|a'f_{24}P_4|: (D)\rightarrow C$

During P_1 the word in A is transferred to L and is stored in memory during P_2. As previously, the address for the next instruction in D is transferred to C during P_4.

As a final example consider the instruction "Transfer negative to X" which indicates that if A contains a negative number, the next instruction is to be taken from register X, otherwise from the next register in sequence. Again the instruction read-in cycle is the same as before. It is followed by:

$|a'f_{27}P_1|:$

$|a'f_{27}P_2|:$

$|a'f_{27}P_3|:$

$|a'f_{27}P_4|: A_0(C)+A_0'(D)\rightarrow C.$

The conditional transfer indicated at P_4 states that if A_0 is true (sign of the accumulator is negative) C remains unchanged whereas if A_0 is not true (positive sign), the address in D is transferred into C.

The necessary transfers for three instructions have been demonstrated. All the remaining instructions are designed in the same systematic way. What remains is to indicate how the control unit operates so as to require the machine to perform the indicated sequences of transfers.

The Control Unit

The control of CG*24* may be thought of as a memory unit. It contains a set of registers called *Cm*, an address register G and an output register F. There is also an auxiliary register T. The control is shown in block diagram form in Fig. 3. The states of the F register determine the control states af_i; one bit in F is designated a. During P_2 of every memory cycle an address is inserted in G. The control word selected by G, $Cm<G>$, is stored in F during P_4 of each cycle thereby setting up the proper control state af_i or $a'f_i$ for the succeeding cycle of operation. In CG*24*, *Cm* is a fixed diode memory. Logically speaking it could have been an active memory like a core memory. Each word in *Cm* represents a unique control state af_i or $a'f_i$. It may be divided into two parts, one which initiates the transfers in the machine such as those described in the previous section, and a second address part designated $G[F]$ which refers to operation of control itself.

Each instruction in the computer is designated by a unique sequence of control states, hence by a unique set of control words in *Cm*. The address part of a control word usually contains the address in *Cm* of the next control word. Actually the operation of the control memory is determined by the transfer to G which was mentioned previously. This is

$|f_iP_2|: a(I[N])+a'\{\lambda(G[F])+\lambda'(G)\}\rightarrow G.$

When a is true, the current cycle is one in which an instruction is obtained from memory. In this case the instruction code part of N is transferred to G. This transfer was implied in the previous section when the instruction read-in cycle was described. During all other cycles a is not true and either the address part of F is transferred to G or G remains the same depending upon a Boolean function λ to be described presently. Following this transfer into G results in the transfer,

$|f_iP_4|: (Cm<G>)\rightarrow F$

which sets up the proper state for the next cycle. This state is either that describing the beginning of an instruction $((I[N])\rightarrow G)$, the repetition of the previous cycle $((G)\rightarrow G)$ or the advancing to the next state of the current instruction $((G[F])\rightarrow G)$.

All that remains is to describe the advancing function λ. This function is true if the number in the T register is 0, 1, 2, or 3 and is false otherwise. Thus to "stick" the control in one state a number larger than 3 is inserted in T and to advance to a new state T must be counted down to 3 or lower.

The advantage in this kind of control is its inherent versatility. To change the nature of the machine it is necessary only to change the control memory which in turn represents different sequences of states and hence transfers in the machine. In CG*24* such changes can be made at a very slow rate by physically replacing the control memory with a different memory. If the memory were active such changes could be programmed.

Conclusions

The ideas presented here demonstrate a method of designing a digital computer by specifying control states which in turn specify sequences of generalized transfers. All this is done with some rather general ideas about the actual circuit configurations. To translate this description into a real computer is a relatively simple step logically. It involves setting up a system of gated transfers where the gating is derived from the F register and a counter whose state designates the P_i interval.

The versatility of the control memory was noted in the previous section. Indeed this does provide a method of setting up arbitrary sequences of states to execute instructions. This versatility is not put to use particularly in CG*24* where the control memory is limited to 64 states and where each of the generalized transfers is wired in. Any new instruction for CG*24* must be made up of existing transfers, states for which already exist in the control memory.

However, the notation using transfers between registers and the implementation of the control memory does permit the design of a much more versatile machine. The next step in designing such a machine is to provide a method of building in all possible transfers between existing generalized registers.[3] Suppose the set of generalized registers is designated by $\hat{M}$ and suppose that a control word in F contains three addresses, $(Ad_1[F])$, $(Ad_2[F])$, and $(Ad_3[F])$ each of which refers to one of the generalized registers in $\hat{M}$. Then the general transfer is stated as follows:

$$|f_i|: \mu(\hat{M}<Ad_1[F]>)+ \mu'(\hat{M}<Ad_2[F]>)\rightarrow \hat{M}<Ad_3[F]>$$

The function μ is a Boolean function designated in the machine. If this function is true the generalized register determined by the contents of $Ad_1[F]$ are transferred to the generalized register determined by the contents of $Ad_3[F]$; otherwise the contents of the register determined by $Ad_2[F]$ are transferred.

When viewed in this way, all the registers and derived registers are connected together through a large selection switch (or several such switches for simultaneous transfers) which is actuated by the state of control. Changing the nature of the machine is then accomplished by changing the contents of the control memory which essentially sets up a sequence of selection switch positions to perform the desired instructions.

References

1. Symbolic Design of Digital Computers, I. S. Reed. *M.I.T., Lincoln Laboratory Technical Memorandum, No. 23*, Lexington, Mass., Jan. 1953 not generally available.

2. Logical Design of CG24, G. P. Dinneen, J. A. Dumanian, I. L. Lebow, I. S. Reed, P. B. Sebring. *M.I.T. Lincoln Laboratory Technical Report, No. 139*, Lexington, Mass., Apr. 1956, not generally available.

3. Symbolic Design Techniques Applied to a Generalized Computer, I. S. Reed. *M.I.T. Lincoln Laboratory, Technical Report, No. 141*, Lexington, Mass., Jan. 1957, not generally available.

Discussion

D. P. Boone (Astronautic Company): You will notice that the feeder is tied to the radar. How do you accomplish the feeder from the radar?

Dr. Lebow: The way it is set up right now is the following:

There are three registers which take the data from the radar, and associated with these registers is another bit which tells the computer that some data has come in.

Then upon sensing this bit, the data are transferred directly into the computer memory, without going through the arithmetic unit.

W. J. Seiple (Federal Laboratories): In discussing the conversion of the data, how do you get them into the computer so that it is tagged to that correlation?

Dr. Lebow: One of the registers I spoke about is a register that contains the actual real time, the time at which the return was received, and this represents part of the data that are introduced into the machine.

K. L. Deane (Variomatic): Does this include some analog to digital conversion?

Dr. Lebow: Yes, for the data part of the word, not for the time part, of course.

Question: What consideration, or what are the considered optimizations of these controls?

Dr. Lebow: This kind of optimization was considered by the people who actually deal with circuit design. I, perhaps, did not say enough about that at the beginning of the talk when I explained what we meant by logical design. I think our use of the term is a little different than what most people mean by the term.

You can see from the talk that the term logical design does not get down to the detailed logical configuration.

Obviously, a lot of work is necessary in going from the transfer level to the actual circuit details, and this is a function of hardware that the machine will be built of.

This was optimized in some sense by the people who actually built this machine.

Design Criteria for Autosynchronous Circuits

J. C. SIMS, JR. H. J. GRAY

Synopsis: The circuits and organization of present computers are such that possible operating speeds are lower than the capabilities of the components. The speed limitations of such synchronous computers will be described, and design criteria for higher speed operation set forth. Examples will be given for a logic and circuit organization which results in both faster operation and improved performance to cost ratios. In particular, circuits which are free of transient logical malfunctions, sometimes called "spikes," will be developed and a typical autosynchronous system will be shown.

COMPUTING circuitry is generally organized to transfer digital information from a first storage through a logical net into a second storage register. As the information passes through storage, the wave shapes are standardized and the relative and absolute timing of the signals are restored. This situation is generalized in Fig. 1. Here information stored in two registers is read out on arrival of a timing pulse, CP_1, passes through a logical net, and is received by an output register. Upon receipt, it can be read out again to the same or to a further network in response to a timing pulse, CP_2. In synchronous machines, the signals CP_1 and CP_2 are clock pulses, and in the common single-phase systems are the same signal.

In order to establish a point of departure for the present discussion, a brief analysis will be given of synchronous systems of this form. The storage registers are usually flip-flop or shift registers, depending on whether the system is parallel or serial. The logical net is made up of combinations of "and," "or," and "not" elements, each element usually consisting of an amplifier and a group of logical diodes or resistors. The computing device is thus constructed of four basic devices, a store or flip-flop, an "and" circuit, and "or" circuit, and a "not" circuit. In some systems, a stroke element is used to function as an "and," "or," and, "not" device. Stroke elements are typified by the "Larc 1 c" circuit and the "nor" circuit.[1,2]

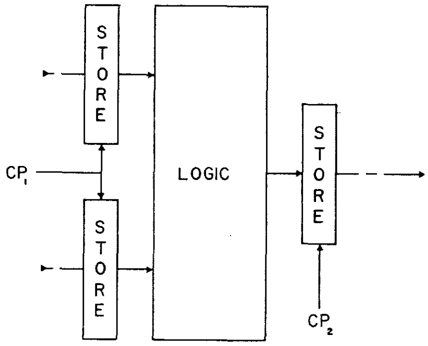
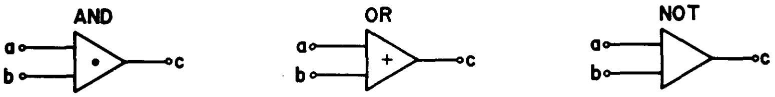


Fig. 1. General block diagram

J. C. Sims, Jr. is with Sylvania Electric Products Inc., Waltham, Mass.

H. J. Gray is with the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pa.



a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

a·b=c

a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

a	b	c
0	0	0
0	1	1
1	0	0
1	1	0

	INPUT	AND	OR	NOT
GENERAL CASE	00→ab→00	0→c→0	0→c→0	0→c→0
CASE NO. 1	00→00→00	0→0→0	0→0→0	0→0→0
CASE NO. 2	00→01→00	0→0→0	0→1→0	0→1→0
CASE NO. 3	00→10→00	0→0→0	0→1→0	0→0→0
CASE NO. 4	00→$\binom{01}{10}$→11→$\binom{01}{10}$→00	0→0→1→0→0	0→1→1→1→0	0→$\binom{1}{0}$→0→$\binom{1}{0}$→0

Fig. 2. Truth tables for "and," "or," "not" logic

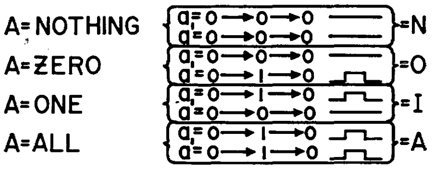


Fig. 3. Two-line notation

Conventional Logical Elements

Regardless of how the logical "and," "or," "not" elements may be constructed, they must follow the truth tables of Fig. 2. In analyzing the behavior of any logical element, it is necessary to consider the way in which the input signals can arrive and correspondingly the way in which the output signals, according to the truth table, will be developed. Consider these conditions for a return-to-zero signal notation. A signal is represented by unity, "or" by +, and "and" by "times."

The arrival and departure of a pulse at input *b* is indicated by the sequence of input states 00→01→00, etc. In the general case, two inputs (*a*) and (*b*) can change independently from zero to information and back to zero. There are four cases for binary signals, taken two at a time, and it will be noted that the first three do not introduce any ambiguity. In the fourth case, however, either input (*a*) or input (*b*) can arrive first and can leave first. The element thus passes through some intermediate input state during the set-up and drop-out periods.

The "and" and "or" functions are symmetrical, and accordingly, the fourth case ambiguity only serves to widen or narrow the output signal. Observe that the "and" function output occurs only during input signal overlap, and that misalignment narrows the output. Conversely, the "or" function widens the output.

The fourth case of the "not" function, however, exhibits a hazard or spike condition. Here coincidence of (*a*) and (*b*) in the truth table calls for zero output, but the set-up and drop-out can pass through intermediate input states calling for a "one" output. These spurious outputs, or "spikes," can operate as information on succeeding stages to cause errors.[3,4]

The most common method for eliminating spikes is through clock pulses. The outputs of the "not" gate can be sampled with a narrow probe occurring safely between the set-up and drop-out spikes. To allow this sampling to be done, the signals must be synchronized with the sampling pulses. A system so constructed uses a central clock and is called a synchronous machine.

One pays serious penalties in speed for synchronous operation. In a typical synchronous system, upwards of half the time will be expended in retiming and in time tolerances. For example, consider a recent computing system operating at 2 megacycles (mc), or one pulse every 500 microseconds (μsec). The system uses eight levels of logic between storage, and each level has a maximum signal delay of 40 μsec. The delay of an average element may, of course, be only 20 μsec, and, indeed, may be as little as 12 μsec. But in designing the circuits and assigning the repetition rate, one must assume the worst end-of-life case. Thus, allowing 40 μsec per level for eight levels, the result is a delay of 320 μsec.

The signals are propagated into the logical network from flip-flop storage. The flip flops, on the average, will read out with a delay of 20 μsec, but in the worst case, the delay may be 40 μsec. With a similar tolerance for reading into flip-flop storage, and allowing ±40-μsec clock jitter, the repetition time between clock pulses becomes 480 μsec. If, however, the clock and its jitter are eliminated, and average (rather than maximum) times are used, the delay becomes 200 μsec for a 5-mc repetition rate.

Spike-Free Logical Elements

One cannot realize this speed improvement unless two things are done. First a logical clock must be provided in place of the fixed time clock, and secondly, one must avoid the spike problem. Some work has been done in this direction, notably by Pomerene,[3] Meagher,[4] Mealy,[5] and Huffman.[6] Progress to date has been largely confined to the development of logical clocks. The spike problem has been resolved by careful control of network delays and inhibit pulse widths. It will be noted by referring to the "not" table of Fig. 2 that if the inhibit input signal (*a*) is wider than signal (*b*) and completely overlaps it, the last line of the transition table can be made to read: 0→0→0→0→0. This design method, however, is not easily accom-

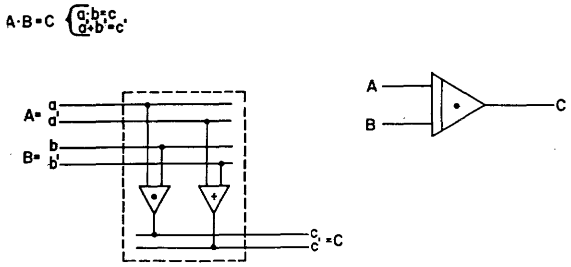


Fig. 4 (left). Two-line buffer

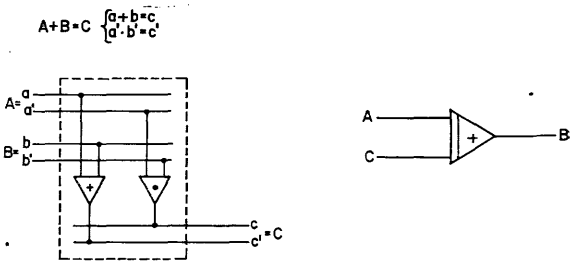


Fig. 5 (right). Two-line gate

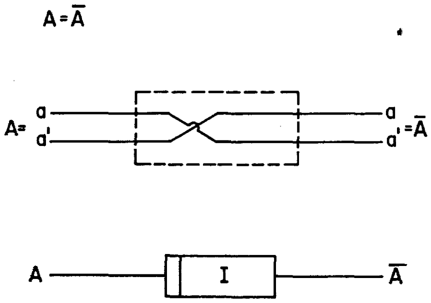


Fig. 6. Two-line negator

plished, particularly in large systems. In systems using this spike correction method, a situation called "races" or "dynamic hazard" exists which can be difficult to resolve and whose solution reduces the permissible circuit speed.

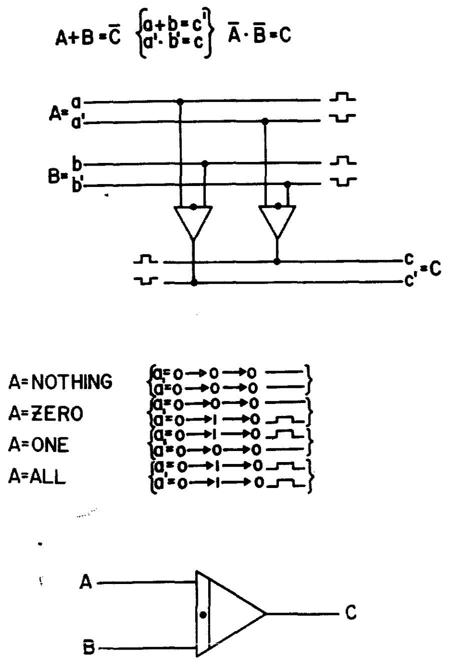
A more satisfactory solution to the spike problem would be to have a logical element which did not produce spikes. In Figs. 4, 5, and 6 are shown the well-known "2-line" notation, which imposes symmetry even in the inhibit case and results in spike-free operation. Two-line notation is shown in Fig. 3, there being four states for binary signals taken in pairs. The zero—zero case is defined as nothing, zero—one as 0, and one—zero as 1 and 1—1 as all or excluded. The signal forms and transitions are shown for these cases.

The 2-line "or" circuit of Fig. 4 consists of two logical elements, an "and" and an "or" circuit. Inputs A and B each have two lines, $a-a'$ and $b-b'$. Signals on (a) and (b) pass through the "or" circuit to output (c') while inputs (a') and (b') operate through the "and" circuit to output (c). It performs the function $A+B=C$ as $a+b=c'$ and $a'.b'=c'$. The proposed symbol for the 2-line "or" element is shown.

A 2-line "and" gate is shown in Fig. 5, together with the 2-line "and" symbol. The "and" circuit is similar to the "or" circuit, except that the connections are reversed to perform $A.B=C$ as $a.b=c$ and $a'+b'=c'$. Negation is accomplished by inversion simply by crossing over the a and a' lines as shown in Fig. 6.

A 2-line stroke element or buffer-complementer is shown in Fig. 7. It is constructed with two single-line stroke elements, the signals on the lines a and b being turn-on pulses and those on lines a' and b' being turn-off pulses. Also shown in Fig. 7 is a truth table for the 16 states of the element. The permissible states are 1 through 3, 5 through 7, and 9 through 11. The other states are excluded by definition since the 1—1 "all" case is not used, and, indeed, serves as a check, as will be described later.

It is a requirement of 2-line circuits that the signals return to "nothing" (line 1) between information. Input signals arrive on a or a' and on b or b'. Line 6 states the "and" operation, and it will be noted that the setup and drop-out conditions can pass only through lines 2 and 5, both of which give intermediate outputs of nothing. When operating as an "or" element, however, a "nothing" input does not inhibit, as can be noted in lines 3 and 9.



	A	B	C	
CASE	a a'	b b'	c c'	
1	0 0	0 0	0 0	NOTHING
2	0 1	0 0	0 0	UNDETERMINED
3	1 0	0 0	0 1	OR
4	1 1	0 0	0 1	EXCLUDE
5	0 0	0 1	0 0	UNDETERMINED
6	0 1	0 1	1 0	AND
7	1 0	0 1	0 1	OR
8	1 1	0 1	1 1	EXCLUDE
9	0 0	1 0	0 1	OR
10	0 1	1 0	0 1	OR
11	1 0	1 0	0 1	OR
12	1 1	1 0	0 1	EXCLUDE
13	0 0	1 1	0 1	EXCLUDE
14	0 1	1 1	1 1	EXCLUDE
15	1 0	1 1	0 1	EXCLUDE
16	1 1	1 1	1 1	EXCLUDE

Fig. 7. Two-line stroke element (buffer complimenter)

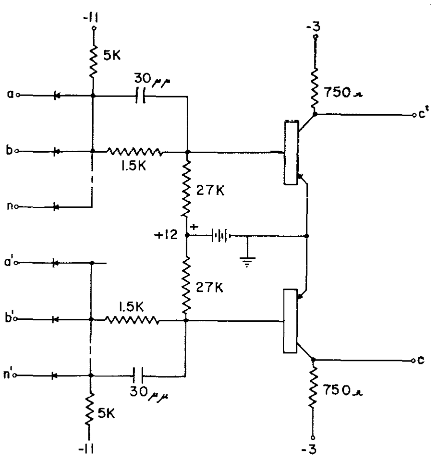


Fig. 8. Two-line stroke circuit

It is evident that the "not" gate is identical with the "and" gate; all 16 input states are shown, and inhibit operation is obtained by gating B against A. If $\bar{A}$ is zero and B is zero, the output is one. But if $\bar{A}$ is one, the output is zero, independent of the B input. In all permissible operations, the element is spike-free.

The circuit of a stroke element is shown in Fig. 8. It consists of two SB-100 transistors operating in the grounded emitter connection. Each amplifier is a complementer with a cluster of "or" diodes on its input. The fact that the system is spike-free allows the circuit time constants to be optimized for the signal bandpass without compromise for spikes. If spikes were present, it would be necessary to choose a value for the speed-up condensers such that the circuit delay for spikes would be the same as the delay for signals. Obviously, if this is not done, spikes may fall in the clock sampling period. But if spikes are not present, it is sufficient to optimize for signals alone, and an improvement in power gain or bandwidth of 25% is realized.

Two more elements are needed to construct a time-independent or autosynchronous system. These elements are: A 2-line flip flop and a 2-line checker. The flip flop is shown in Fig. 9 and consists, in fact, of two single-line flip flops since two bits must be stored. The circuit provides direct and complement outputs and for a clear to "nothing." The 2-line checker of Fig. 10 tests the lines for signals of "nothing," "something" (one or zero), and "error" (the excluded case 1—1).

It should be emphasized that all of these elements operate in "return-to-nothing" notation, each line operating in "return-to-zero." In this mode of op-

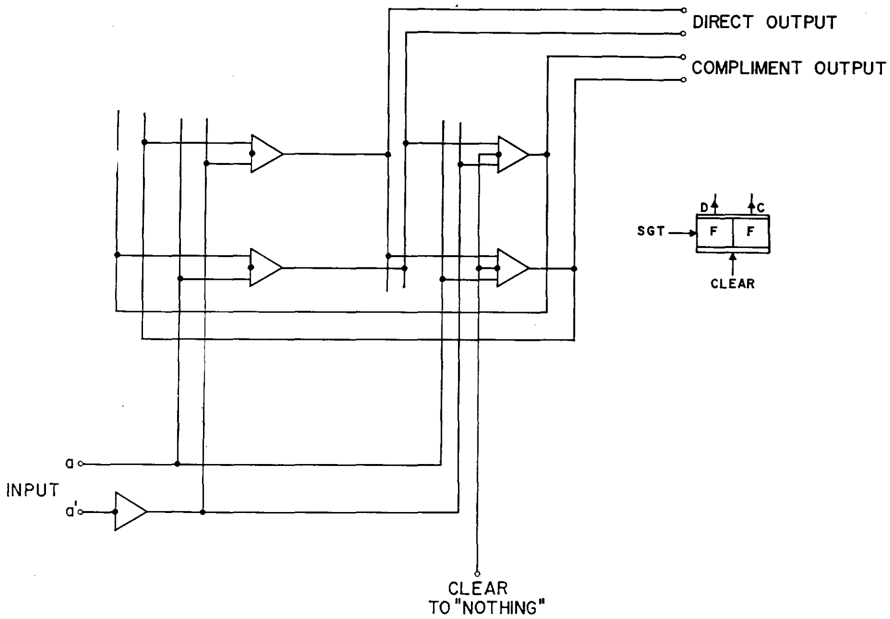


Fig. 9. Two-line flip-flop

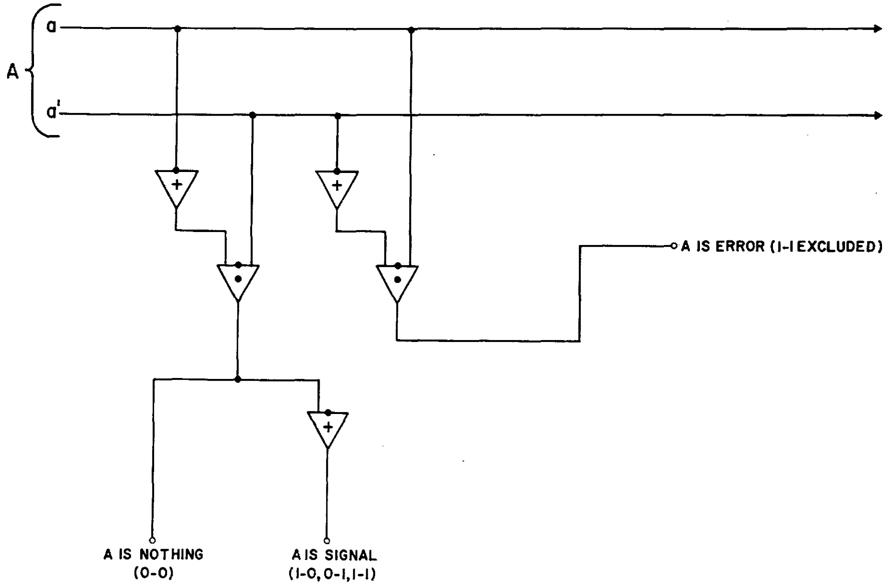


Fig. 10. Two-line checker

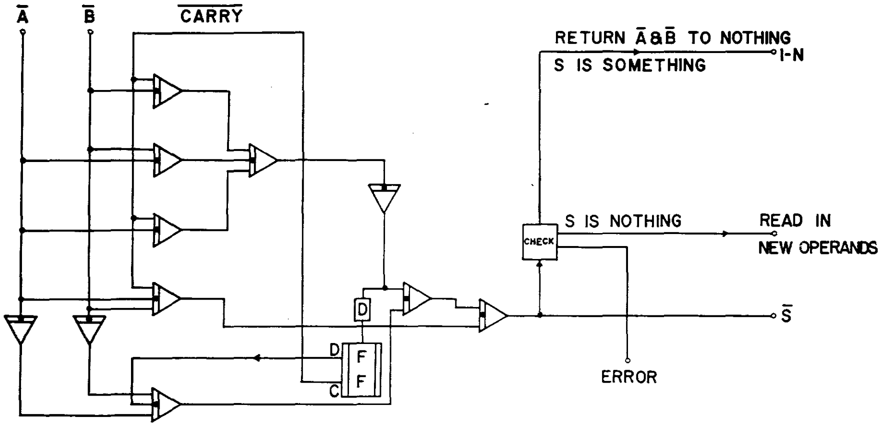


Fig. 11. Two-line adder

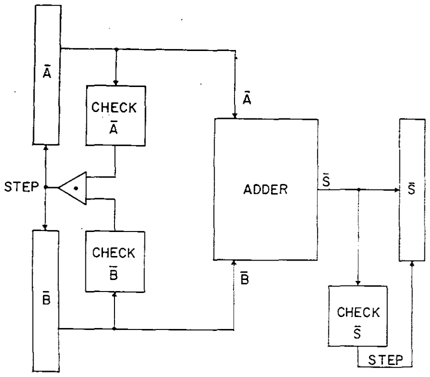


Fig. 12. Autosynchronous equivalent of Fig. 1

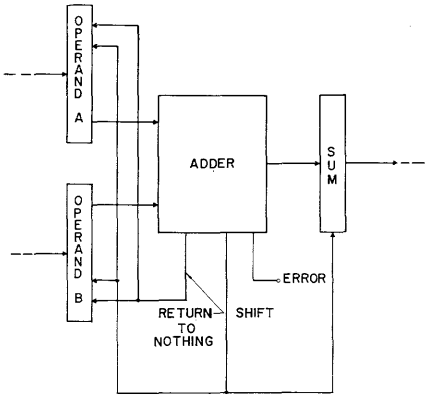


Fig. 13. Autosynchronous adder

eration they are spike-free. There are no known elements that are spike free in "non-return-to-zero."

Spike-Free Logical Networks

Using the elements first described, one can construct a serial adder, and this adder will take the classical 5-level form shown in Fig. 11. It will be noted that essentially no change in logical form is required, and the new 2-line elements can be substituted directly for single-line elements in existing logical structures. Of course, it is necessary to provide a logical clock to replace the conventional oscillator, and the 2-line checker executes this function.

In the serial adder, the first operand bits $\bar{A}$ and $\bar{B}$ are propagated into the network to initiate the process. When a sum output appears, the checker sends a pulse to the operand registers, returning the inputs to "nothing." This 0—0 condition then propagates through the net, and on appearing at the checker, sends a shift pulse to the registers to read in the next operand bits, and shifts the sum register for receipt of the result. This sequence proceeds in a self-synchronized manner until the addition is complete.

The speed is determined only by the circuit delays. This process, illustrated in Fig. 12, is the autosynchronous equivalent of the general system of Fig. 1.

At this point it would be in order to review critically the new organization, comparing it against the equivalent synchronous system. The first thing to note is that the amount of hardware has been doubled. Every logical element is push-pull and every storage holds twice as many bits. Has an improvement in performance which justifies the increase been achieved?

If this adder were a conventional single-line synchronous device, it would operate at 2.8 mc in non-return-to-zero. One would have to allow 200 μsec for the network delay and another 150 μsec for storage delay and clock jitter. Using autosynchronous elements optimized for signals, there is an average delay of around 15 μsec per level. The total network delay around the loop will be 135 μsec for signals and 120 μsec for return to "nothing" for a total cycle time of 255 μsec or 4 mc. The 2-line system is therefore a factor of $1^{2}/_{3}$ faster.

The 2-line system has other important advantages. First, it is inherently self-checking, and redundant logic and parity checks can be eliminated. The clock and distribution system is eliminated. One practical feature of considerable importance is that the power supply loading is constant and line transients are minimized. The autosynchronous system, which is nearly twice as fast, requires less than twice as much hardware since, in a typical computer, the checking circuits, redundant adder, and clock involve about one third of the equipment. There is a cost-to-performance ratio for the two systems that is about equal.

It will be noted that the discussion has been limited to serial circuits and logic. This is no accident. The advantages of 2-line autosynchronous logic cannot be fully realized in a parallel system. When a logical network has many parallel related paths, the slowest path provides the determining delay. The larger the number of paths, the greater the probability that one path will have a delay approaching the limiting worst case. It is conceivable that a parallel asynchronous system may be no faster than its synchronous equivalent. Indeed, it can be said that parallelism is never a pay-off from a performance-to-cost ratio standpoint and is only justifiable from the point of view of timeliness.

It is, therefore, ironic that most of the work to date on asynchronous system has been done in connection with large parallel logic computers. The effort, of course, has been to attain the highest possible speed of the equipment. For example, the serial adder described previously performs a full 50-bit addition including carry propagation in $12^{1}/_{2}$ μsec and with only 32 transistors. A full parallel asynchronous adder with its carry net and controls may be 10 to 20 times faster, but involves over 100 times as much equipment.

There is a philosophical lesson to be learned here. Prior asynchronous devices have been structurally parallel but logically serial; i.e., the machines have handled data words in a parallel manner, but instructions have been handled serially. It is felt that autosynchronous equipment should be structurally serial but logically parallel.

In the system described in Fig. 12, the checker is on the output line of the logical network. It therefore operates so that only valid results can be generated. Where speed is not essential, this is the preferable procedure. If higher speed is required, the checker can monitor the input lines to propagate signals and a further checker can monitor the output lines both for validity and to step the output storage. This has the disadvantage that information stored in the network may be lost in the event of an error. This system is shown in Fig. 13.

Since the checker has three levels of logic for "information" and two levels for "nothing," and assuming one delay for readout from storage, the readout period will be $6 \times 15 = 90$ μsec per bit or 11 mc. The delay in the adder no longer determines the bit rate, and the logic can have as many levels as desired provided the wave shapes can be preserved. Since the signals are, by definition, "return-to-zero," a-c coupled pulse forming amplifiers can be used and the logical chains can be indefinitely long. In such a network it will be necessary to include adjustable compensating delays so that all paths through the network have the same transit time. It should be pointed out, however, that this is different from the old "race" or "dynamic hazard" problem since changes in circuit time constants will not produce spikes or wrong answers, but will only cause the equipment to stall. It follows that the adder of Fig. 13 will perform the 50-bit addition in $4^{1}/_{2}$ μsec, which begins to compare with a parallel synchronous adder.

One final point should be made. The synchronization problems in clocked machines are such that they determine the limiting speeds. As the frequency of operation rises, wiring delays, both in the network and the clock lines, become comparable to the pulse rate. Synchronous operation much above 5 mc becomes difficult, even with very fast low delay amplifiers. The autosynchronous approach not only raises possible operating speeds, but reduces the amount of equipment required.

Conclusions

The autosynchronous logic and circuits that have been described are capable of higher performance than their synchronous counterparts. The new logic is self-checking and inherently more reliable, but the technique can be applied to existing logical networks by substitution of 2-line elements for single-line elements, the clock being replaced by information checking or test elements. The advantages of the autosynchronous approach become greater as new short delay amplifiers become available and as clock tolerances and wiring delays in conventional networks become the principal considerations. It is hoped that the autosynchronous concept can contribute to improved reliability and speed in future computing devices.

References

1. The Transistor NOR Circuit, **W. D. Rowe.** *AIEE Conference Paper 57-195.*

2. A New Method of Designating Low Level, High Speed Semi-Conductor Logic Circuits, **W. B. Cagle, W. H. Chen.** *Westcon Convention Record* (Circuit Theory), Institute of Radio Engineers, New York, N. Y., Aug. 1957.

3. On the Length of The Longest Carry in Binary Addition, **Herman H. Goldstine, James H. Pomerene.** *Ordnance Computer Research Report*, vol. II, no. 4, Institute of Advanced Studies, Ballistic Research Laboratories, Aberdeen Proving Ground, Md., Oct. 1955, pp. 23–26.

4. Method for Synthesizing Sequential Circuits, **George H. Mealy.** *Bell System Telephone Journal*, New York, N. Y., vol. 34, Sept. 1957, pp. 1045–79.

5. The Design of Hazard–Free Switching Networks, **D. A. Huffman.** Journal, Association for Computing Machinery, Baltimore, Md., vol. 4, no. 1, Jan. 1957.

Discussion

Question (Bell Telephone Laboratories): As far as I know, you gentlemen have invented the word "autosynchronous." I just want to be sure that I know what you mean by it.

Do you mean by autosynchronous, any system in which clock pulses are generated by checks of some sort on the order of computing circuits, such that the clock does not run at a steady rate, or do you mean specifically what you have described in the paper—the not returning to zero features?

Mr. Sims: I think your first definition is closest to what we have had in mind. That is, that an autosynchronous device is one in which there are essentially regenerative impulses which, by sensing information flowing in the network, can use this sensing to propagate new information.

Question: There was something I did not understand. I did not quite understand about your difference between parallel operation and the serial operation as far as—well, let us say you spoke about using, possibly adding, one bit at a time.

This could speed up multiplication, but what about addition? Would it not increase the delay or the length of time for addition?

Mr. Sims: The question was what I had intended in discussing the use of sequential logic and parallel operations in present machines.

I said this was the way in which most computers had been built up to this time, and I said that for autosynchronous or asynchronous equipment, it looked as if one should use a lot of independent serial arithmetic units, rather than a single parallel one, because you would get more work done for your money this way.

Question: Wouldn't this increase the amount of equipment rather than decrease the amount of equipment for one parallel?

Mr. Sims: No. I think this follows from the fact that a parallel adder has perhaps one hundred times as much equipment as a serial adder, and yet is only perhaps ten times as fast.

Therefore, when you put in, say, ten serial adders which can make ten serial additions and keep up with my synchronous (ten times as fast) parallel adder, I can do this for one-tenth of the price.

Question: Have you considered the problem of locating faults by diagnostic checks?

Mr. Sims: I think as far as locating troubles are concerned, it is to some extent simplified since the propagation of information depends on the propagation of prior information, and since each bit is checked by a little checking arrangement on the output.

Then, a fault of any kind causes the equipment to stall right at the commission of the error, and I would think that suitable indicator lights would tell you where the fault occurred.

Analysis of TRL Circuit Propagation Delay

W. J. DUNNET E. P. AUGER A. C. SCOTT

Synopsis: A program to design transistor-resistor logic (TRL) circuits and compile TRL propagation delay tables on a digital computer is presently underway at the Sylvania Data Processing Laboratory. This paper points to the need for such a program and describes transistor and TRL circuit studies that have resulted in the basic relationships being programmed.

The criterion of TRL circuit performance is taken to be signal propagation delay. This delay can be predicted by applying the transistor large-signal transient expressions of Ebers and Moll.

The accuracy and usefulness of these expressions are increased by measuring pertinent transistor parameters under large signal transient conditions and by considering the practical case where propagated signals have finite, rather than step, rise, and fall times.

COMPLETE knowledge and control of TRL building block performance is necessary to predict the performance of a large data processing system constructed using TRL logic. (The TRL building block is frequently referred to as the NOR circuit.) Control of the circuit performance permits confident application of the circuit thousands of times over, in both controlled and uncontrolled environments.

One of the most important limiting characteristics of a TRL circuit is its propagation delay. Propagation delay is defined as the time required, after application or removal of an input signal, for the TRL transistor output level to begin to change. The magnitude of propagation delay depends upon:

1. The transient performance of the transistor.
2. The way the TRL stage is wired into the system and the number of inputs being energized (circuit environment).
3. The state of the transistor (conducting or nonconducting) prior to the application or removal of an input signal.

Under the section headed Propagation Delay, transistor input capacity transistor rise, decay and storage times and their relationship to propagation delay are discussed. A procedure is outlined for calculating the propagation delay for a given transistor and circuit configuration.

Later, TRL circuit environment is considered and general expressions relating turn-on and turn-off circuit betas to circuit parameters (supply voltages, input and collector resistors) and transistor steady state voltages, V_{BE} and V_{CE} (saturated), are developed.

Finally, relationships developed in the paper that permit prediction of TRL propagation delay are summarized.

Introduction to Transistor-Resistor Logic

The TRL circuit consists of a resistor "or" gate followed by an inverting transistor amplifier. Fig. 1 shows two TRL circuits arranged in cascade. In this figure, the transistor parameters and circuit values are chosen so that the collector voltage of the off transistor Tr−1, is of sufficient magnitude to keep Tr−2 in saturation. If the collector voltage of an "off" transistor is defined as a "1" and the collector voltage of an "on" transistor is defined as a "0," then the logical performance of the TRL circuit can be described as follows. Referring again to Fig. 1, if neither inputs 1, nor 2, nor 3, are present, the collector voltage of Tr−1 will be at "1" level. If any one or more inputs are present, the collector voltage will be at "0" level. Any binary-digital logical function can be performed by combinations of the basic TRL circuit. For example, Fig. 2 shows a TRL flip-flop and Fig. 3 a TRL adder logic.

Compared to other popular types of logic, for example current mode[1] and direct-coupled transistor logic (DCTL)[2], TRL is more economical and reliable due primarily to its lower transistor count and simplicity. The one disadvantage of TRL is that it does not fully exploit the speed capabilities of the transistor.

As previously mentioned, the transistor parameters and circuit values of a TRL circuit are chosen so that the collector voltage of an "off" transistor is sufficient to keep a driven transistor in saturation. However, it is not meant to imply that a TRL circuit is designed on the basis of steady state conditions alone. Generally speaking, excessive propagation delay occurs well before a TRL circuit fails to meet steady state requirements. As a result, TRL circuit design is governed primarily by propagation delay considerations.

W. J. DUNNET, E. P. AUGER, and A. C. SCOTT are with Sylvania Electronic Systems, Division of Sylvania Electric Products Inc., Needham, Mass.

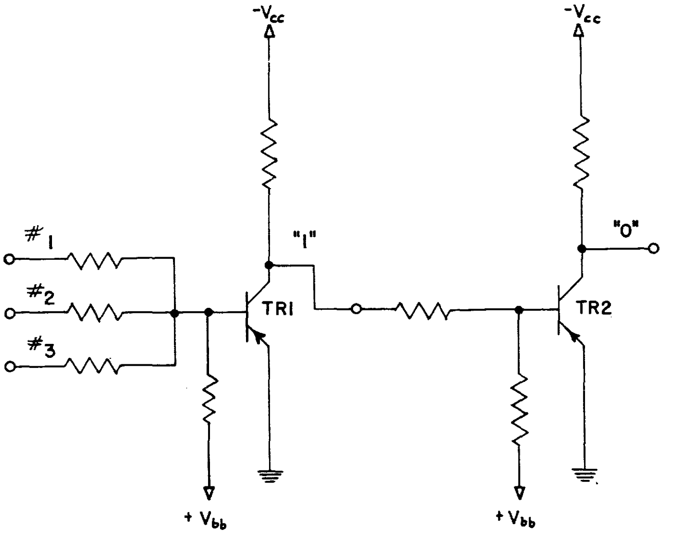


Fig. 1. Two TRL circuits arranged in cascade

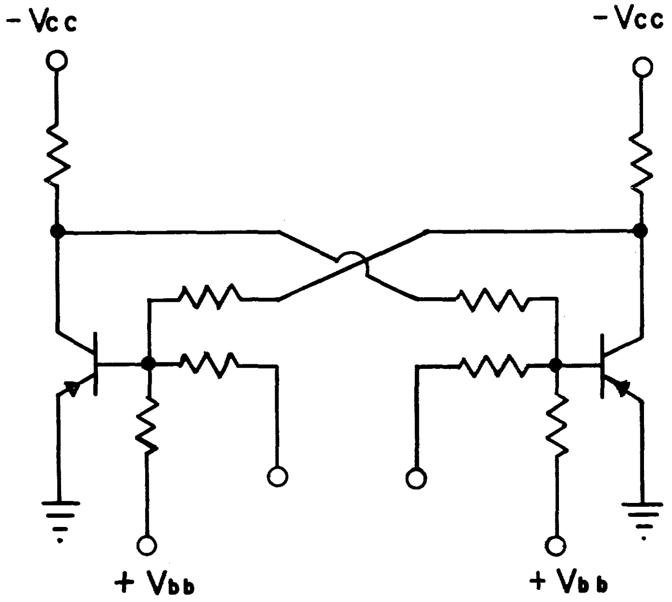


Fig 2. TRL flip-flop

Propagation Delay

Causes of Propagation Delay

There are four principal causes of propagation delay in a TRL circuit. Three of these which have been discussed by Moll[3] are the rise or turn on time, T_0 the decay or turn off time, T_2, and the storage time, T_1. A fourth cause of delay is the capacity from base to ground of an off biased transistor. This capacitance carries a small positive charge which must be drained away before the transistor can begin to conduct.

Moll has shown that rise, decay, and storage times for step changes in base current are determined by the following relations (see nomenclature for definitions of terms used in these and following equations).

$$T_0=\frac{1}{(1-\alpha_N)\omega_N}\ ln\ \frac{I_{B1}}{I_{B1}-0.9\,\dfrac{(1-\alpha_N)}{\alpha_N}\,I_C} \qquad (1)$$

$$T_2=\frac{1}{(1-\alpha_N)\omega_N}\ ln\ \frac{I_{C1}-\dfrac{\alpha_N}{1-\alpha_N}\,I_{B2}}{\dfrac{I_{C1}}{10}-\dfrac{\alpha_N}{1-\alpha_N}\,I_{B2}} \qquad (2)$$

$$T_1=\frac{\omega_N+\omega_I}{\omega_N\omega_I(1-\alpha_N\alpha_I)}\ ln\ \frac{I_{B1}-I_{B2}}{I_{C1}\,\dfrac{1-\alpha_N}{\alpha_N}-I_{B2}} \qquad (3)$$

Referring to equations 1 and 2, it can be seen that these expressions consist of two parts. (In deriving equations 1 and 2, Moll assumes that $\omega_N R_L C_C << 1$. The more practical case, where $\omega_N R_L C_C$ is of the order of magnitude of unity is discussed later in this paper. In this paper only p-n-p-type transistors will be considered. The positive sense of base, collector and emitter currents are as shown in Fig. 4.) One part, an intrinsic factor, is a measure of the transistor switching characteristic. The second term is a natural logarithm, essentially established by the transistor circuit environment, which is described in detail in a later section.

The intrinsic factor contains two transistor parameters, α_N and ω_N. The parameter α_N is the ratio of I_C and I_E, when $V_{CB}=0$, and ω_N is defined by Ebers and Moll as the alpha cut-off frequency of the transistor in the normal direction. The radio-frequency (r-f) alpha cut-off frequency varies considerably with both collector current and voltage. Therefore, a small-signal a-c measurement at any one particular bias point does not fully describe the frequency response of the transistor during a large transient. It can be seen from equation 1 that the rise time, when the transistor is driven from cut-off to the edge of saturation ($I_C=\beta I_{B1}$), is equal to the product .of the intrinsic factor and a constant, 2.3.

$$t_r=\frac{2.3}{(1-\alpha_N)\omega_N}\quad \text{(No Enhancement)} \qquad (4)$$

In order to avoid confusing the ω_N in equation 4 with the conventional r-f measured value, ω_N has been redefined as ω_0 the characteristic cut-off frequency.

$$\omega_0=\frac{2.3}{(1-\alpha_N)t_r} \qquad (5)$$

The characteristic frequency, ω_0, can be obtained by measuring β and t_r and applying to equation 5.

Referring to equation 3, Moll's storage time equation, it can be seen that this expression also consists of two parts. One part, an intrinsic factor, is a measure of the transistor storage characteristic. The second term is a natural logarithm, essentially established by the transistor circuit environment. As previously mentioned, the circuit environment is described in detail in a later section.

The intrinsic storage factor contains four transistor parameters α_N, α_I, ω_N, and ω_I. The first, α_N, is the ratio of I_C and I_E when $V_{CB}=0$. The second, α_I, is the ratio of I_E and I_C when $V_{EB}=0$. Finally, ω_N and ω_I are defined by Ebers and Moll as the alpha cut-off frequency of the transistor in the normal and inverted directions, respectively. Here again conventional r-f small signal values of ω_N and ω_I measured at any one particular bias point do not fully describe the frequency response of the transistor. Conventional r-f measurements are made with the collector junction reverse biased, however, during storage time the collector junction is forward biased.

It can be seen from equation 3 that the storage time, when $I_{B1}=I_{C1}$ and $I_{B2}=0$ is equal to the product of the intrinsic storage factor and the natural logarithm of β_N.

$$t_s=\frac{ln\beta_N}{(1-\alpha_N\alpha_I)\,\dfrac{(\omega_N\omega_I)}{\omega_N+\omega_I}}$$

$$(I_{C_1}=I_{B_1},\ I_{B_2}=0) \qquad (6)$$

In order to avoid confusing the ω's in equation 6 with the conventional r-f measured values,

$$\frac{(1-\alpha_N\alpha_I)\omega_N\omega_I}{\omega_N+\omega_I}$$

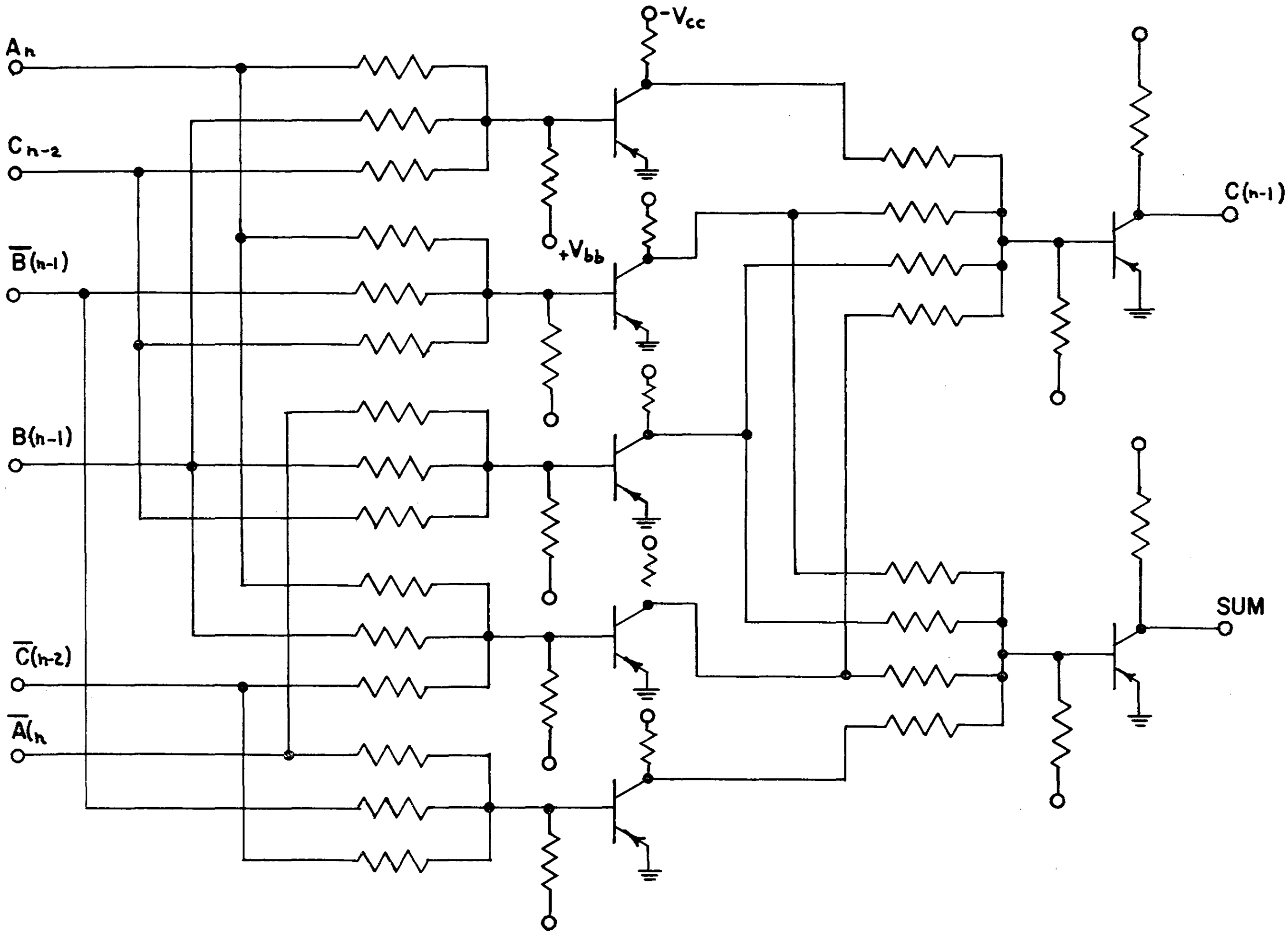


Fig. 3. TRL adder logic

has been redefined as ω_B, where ω_B is called the characteristic cut-off frequency in saturation.

$$\omega_B = \frac{ln\beta}{t_s} \tag{7}$$

The time, t_s, will be called the characteristic storage time of the transistor. Fig. 5 shows a plot of ω_B as a function of I_C. The characteristic cut-off frequency in saturation was obtained by measuring t_s and β_N and employing equation 7.

A comparison of measured and calculated values of rise time versus I_{B_1}, storage time versus I_{B_1} and I_{B_2}, and fall time versus I_{B_2} are shown in Figs. 6–8. The calculated values were obtained from equations 1, 2, and 3 by employing the characteristic frequencies ω_0 and ω_B as discussed. These plots indicate that considerable accuracy can be obtained in calculating the transient performance of a transistor.

The fourth delay effect to be considered is caused by the capacity from base to ground, C_1, of the off biased transistor in Fig. 9. This capacitance carries a small positive voltage when a stage is cut off which must be discharged before collector current begins to flow. Since both the collector and emitter diodes of a cut-off transistor are back biased and the load impedance is fairly small, C_1 will include both the collector and emitter transition capacities of the transistor in question as well as the wiring capacity. Transition capacity of a diode varies inversely with the 1/3 to 1/2 power of the voltage[4] but since the collector voltage stays constant until the base voltage falls to zero, the collector capacity will have an essentially constant value. The emitter capacity, however, will vary as the base voltage changes. The total capacitance should thus be considered voltage sensitive, but a fair account of its effect is obtained if it is measured under the approximately desired operation conditions.[5]

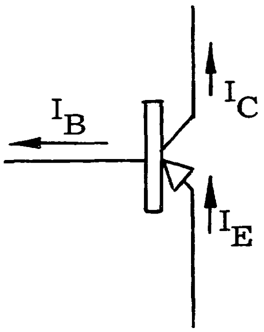


Fig. 4. Positive sense of base, collector, and emitter currents

In the design of medium and high-speed TRL circuits, the effect of collector capacity, in particular the R_LC_C product, cannot be neglected. Easley[6] has shown that when the product $\omega_0 R_L C_C$ is of the order of magnitude of unity the intrinsic factor in equations 1 and 2 becomes

$$\frac{1}{1-\alpha_N}\left(\frac{1}{\omega_0}+R_LC_C\right)$$

It follows then that equation 5 can be written as shown in equation 8.

$$t_r = \frac{2.3}{(1-\alpha_N)}\left(\frac{1}{\omega_0}+R_LC_C\right) = \frac{2.3}{\omega_0'} \tag{8}$$

where

$$\frac{1}{\omega_0'} = \frac{1}{(1-\alpha_N)}\left(\frac{1}{\omega_0}+R_LC_C\right)$$

Fig. 10 shows a plot of ω_0' versus I_C for a *2N428* transistor with a C_C of 8.1 micromicrofarad ($\mu\mu$f). This plot was obtained by measuring t_r and using equation 8.

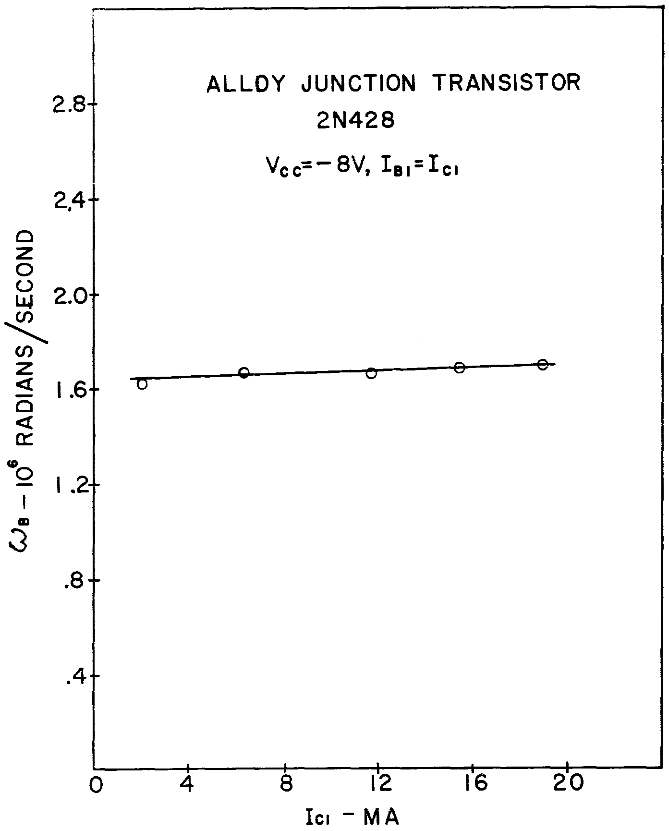


Fig. 5. Characteristic cut-off frequency in saturation versus collector current

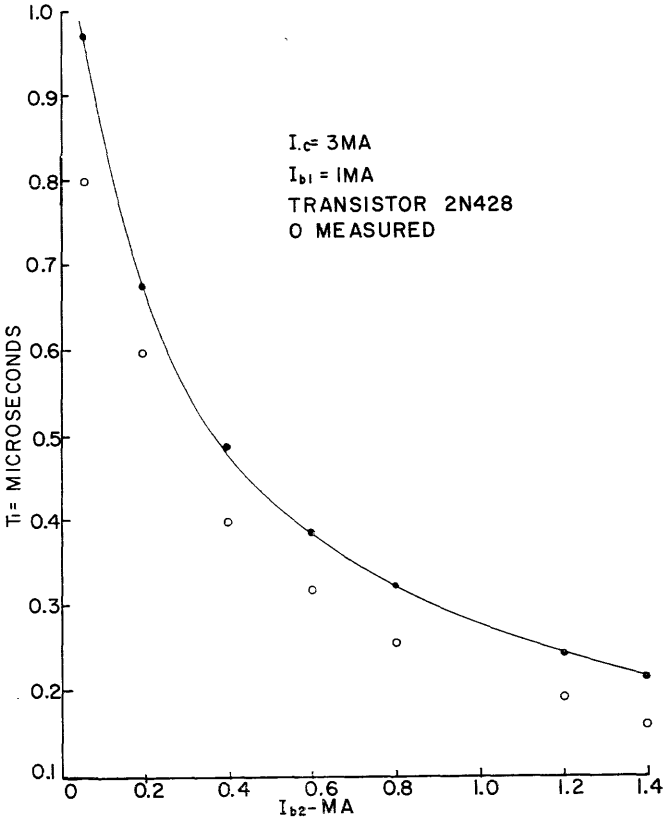


Fig. 7. Storage time versus turn-off base current

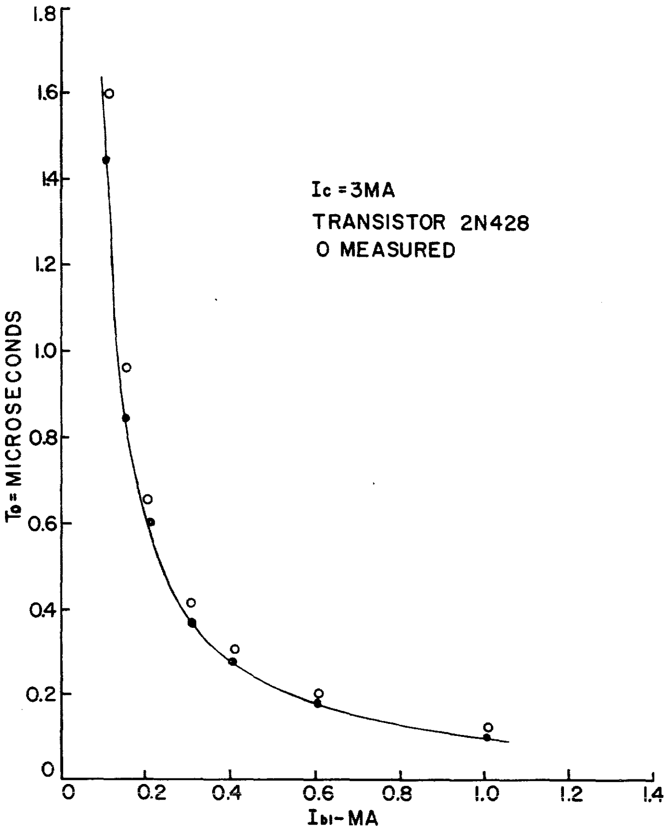


Fig. 6. Rise time versus turn-on base current

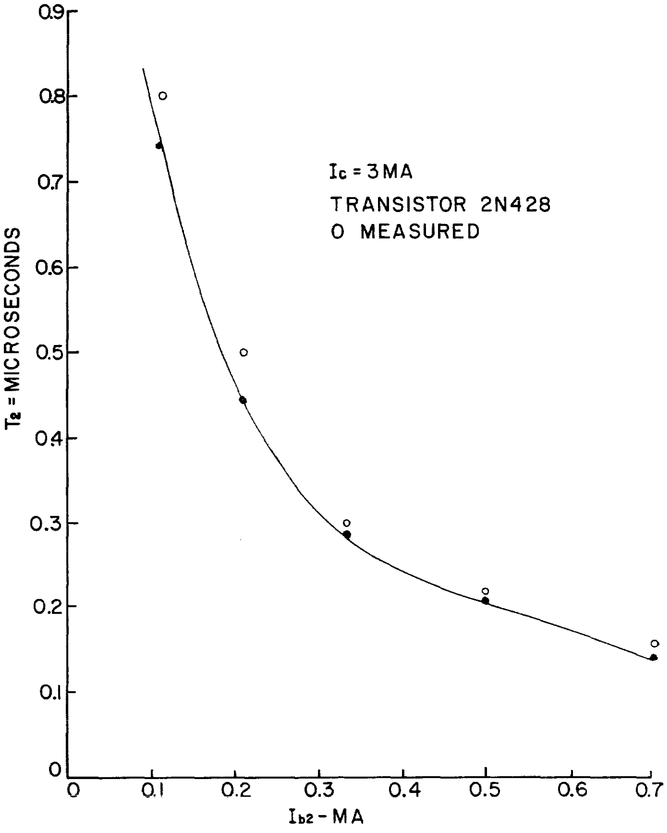


Fig. 8. Decay time versus turn-off base current

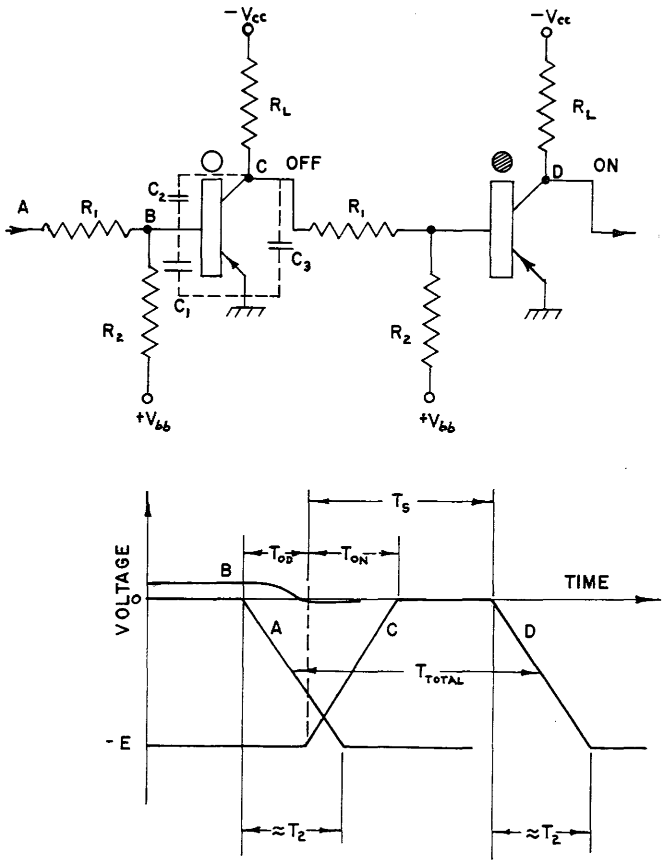


Fig. 9. Circuit analyzed for delay

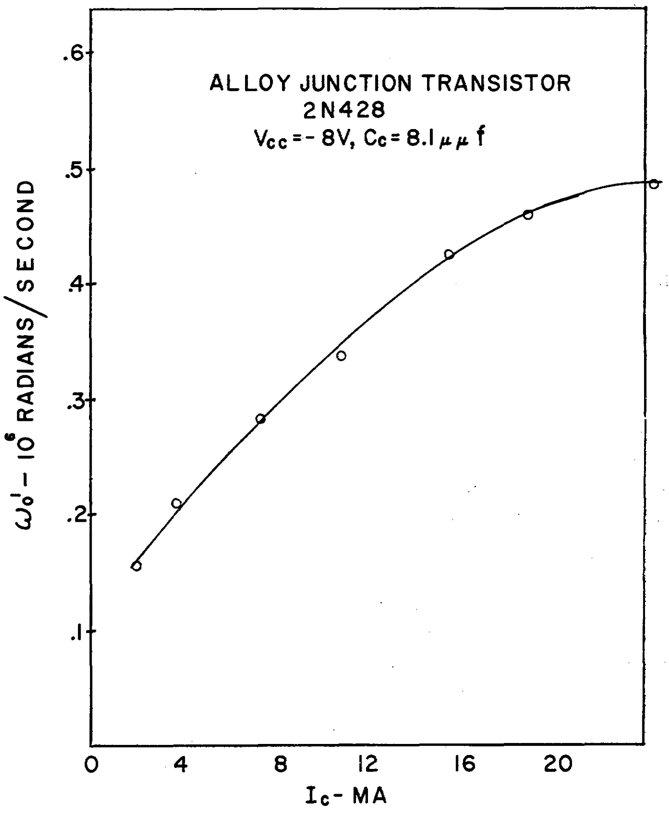


Fig. 10. ω_0' versus collector current

$R_1 = 6.2\text{K}$

$R_2 = 27\text{K}$ (TEKTRONIX 53/54C)

$R_L = 1.5\text{K}$

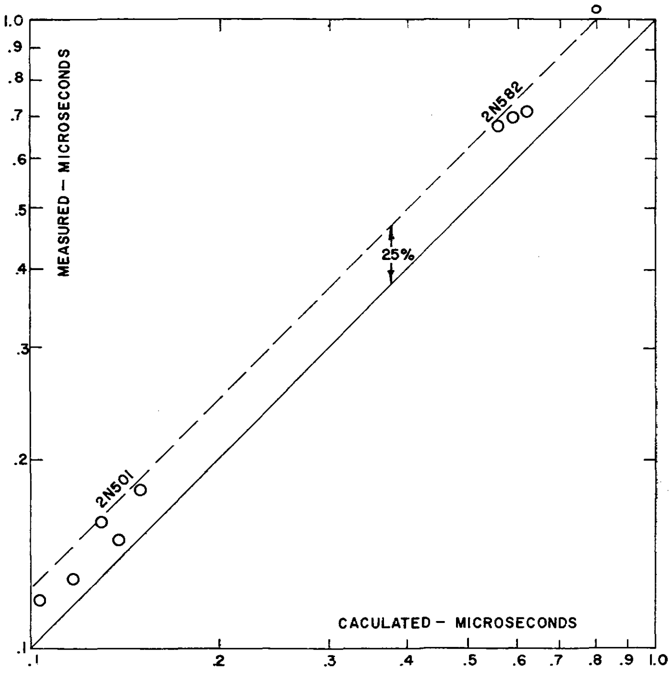
$V_{BB} = V_{cc} = 8$ volts

a. RCA 2N582
0.5 μsec/cm

b. Philco 2N501
0.04 μsec/cm

Fig. 11. Waveforms of circuit in Fig. 9

Fig. 12 (right). Comparison of calculated and measured values of total 2-stage delay time



When using fast transistors such as the microalloy diffused-base *2N501*, or the mesa *2N509* ($C_C \cong 2$ $\mu\mu$f) circuit parasitic capacities should be considered. In particular, because it is multiplied by β_N, parasitic capacity from collector to base, C_2 should be minimized.

If the product of the load resistance and the parasitic capacity from collector to ground, R_LC_3, is small compared with the turn on and turn off transients, its effect will be to introduce a delay R_LC_3.

Derivation of Propagation Delay

Consider the problem of calculating the time for a change of level to pass through the two stages of Fig. 9. The operation of the circuit is as follows: Point A is brought from ground to a negative value in a time, T_2 which is the turn off time of the previous stage calculated from equation 2.

The base of the first stage, point B, is brought from its small positive value to ground in the on delay time, T_{0D}, which is determined by T_2 and the capacitance C_1. It is shown in the appendix that T_{0D} is determined by

$$\left[e^{-T_{0D}/RC_1}+\frac{T_{0D}}{RC_1}-1\right]=\frac{V_1T_2}{(V_1-V_2)RC_1}$$

for $T_{0D} \leqslant T_2$

$$\frac{T_{0D}}{RC_1} = ln\left[\left(\frac{e^{T_2/RC_1}-1}{T_2/RC_1}\right)\left(\frac{V_2-V_1}{V_2}\right)\right]$$

$$\text{for } T_{0D} \geqslant T_2 \qquad (9)$$

where R is the equivalent source resistance to the left of C_1, V_1, is the voltage to which C_1 is initially charged, and V_2 is the voltage to which C_1 would finally be charged if the base did not begin to conduct. Expressions for R, V_1, and V_2, for a TRL stage with more than a single input resistor and for various driving conditions are derived in the appendix.

When the base voltage falls to zero, the base emitter diode becomes forward biased and C_1 is unimportant. The transistor is then turned on with a wave form which has attained T_{0D}/T_2 of its maximum value and rises to full value in time $T_2 - T_{0D}$. This is to say that the transistor is turned on with what is left of the input transient after time T_{0D} has passed. It is shown in the appendix that this on rise time, T_{0N} is given by

$$T_{0N} = T_0 + \frac{1}{\omega_0'} \ln\left[\frac{T_{0D}}{T_2} + \frac{e^{\omega_0'(T_2-T_{0D})}}{\omega_0' T_2} - 1\right] \qquad (10)$$

where T_0 is the on time calculated from equation 1, assuming the maximum base current is applied as a step.

The second transistor is then turned off by a transient which exists over a time T_{0N}. It is shown in the appendix that this results in a storage time

$$T_S = T_1 + \frac{1}{\omega_B} ln\left[\frac{e^{\omega_B T_{0N}}-1}{\omega_B T_{0N}}\right] \qquad (11)$$

where T_1 is the storage time calculated from equation 3 assuming the turn-off transient occurs instantaneously.

The total delay time between the change of level at point A and at point D is

$$T_M = T_{0D} + T_S + 2R_L C_3 \qquad (12)$$

Fig. 11 shows waveforms taken from the circuit of Fig. 9.

Under certain conditions, the calculation of storage time, T_S, previously described may be considerably simplified. For example if $\omega_0'(T_2 - T_{0D}) \ll 1$

$$T_{0N} \approx T_0 + \frac{(T_2 - T_{0D})^2}{2T_2} \qquad (13)$$

and for $(T_2 - T_{0D})^2/2T_2 \ll 1$

$$T_{0N} \approx T_0 \qquad (14)$$

And if furthermore $\omega_B T_{0N} \ll 1$

$$T_S \approx T_1 + \frac{1}{2} T_0 \qquad (15)$$

That is to say in many cases the storage time, T_S, may be approximated by the storage time plus one half the rise time calculated from Moll's equations for step inputs.

Equations 9, 10, and 11, were tested in a 5-stage TRL circuit for which $V_{CC} = V_{BB} = 8$ volts, $R_1 = 6.2K$, $R_2 = 27K$, and $R_L = 1.5K$. Two transistor types were used: The Radio Corporation of America (RCA)*2N582* and the Philco *2N501*. The frequencies ω_0' and ω_B were determined by measuring rise and storage times for a step input and using equations 7 and 8. C_1 was determined by measuring the on delay time for a step of input current.

Measured and calculated values of total delay time are compared in Fig. 12. Measured values are typically 10 to 25% greater than calculated values. For the *2N582* transistors this error was definitely shown to be due to an excess of storage time over that calculated from equation 11. This effect may be due to the non-planar geometry of the transistors. For example, Simmons[7] has pointed out that minority carriers in the fringe areas of the base region have a longer storage time than carriers in the narrower center. This means that for fast turn off the fringe carriers will contribute to the fall time while for slow turn off they will contribute to storage time.

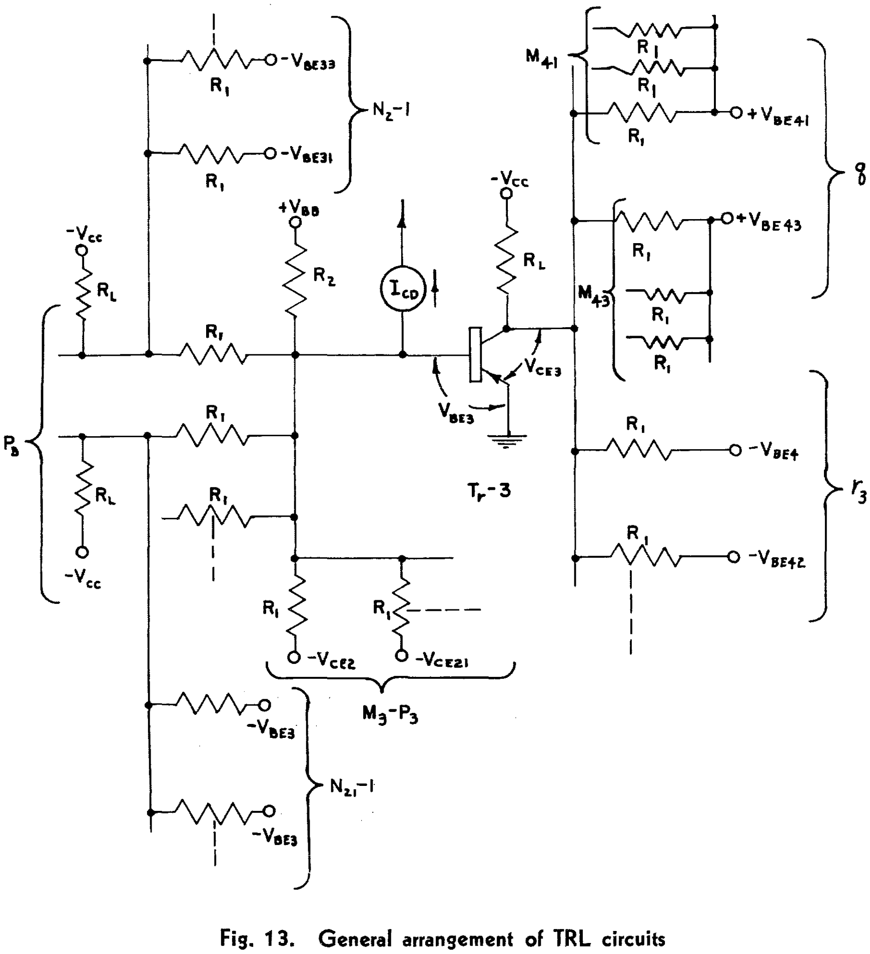


Fig. 13. General arrangement of TRL circuits

Transient and Steady-State Analysis of the TRL Circuit

In the preceding section, it was shown that the transient performance of the transistor is controlled by two factors. One, an intrinsic factor, is a measure of the inherent switching properties of the transistor. The second, a natural logarithm, describes the effects of the circuit environment on the transient time of the transistor. It can be seen by rearranging equations 1, 2, and 3, and letting

$$\beta_{A1} = \frac{I_C}{I_{B1}}, \quad \beta_{A2} = \frac{I_{C1}}{I_{B1}}, \quad \beta_D = \frac{I_{C1}}{I_{B2}}$$

that the magnitude of the natural logarithm term, in general, is established by the ratios of transistor beta, β_N, and circuit beta's, β_A and β_D.

$$T_0 = \frac{1}{\omega_0'} \, ln \, \frac{\beta_N/\beta_{A1}}{\beta_N/\beta_{A1} - 0.9} \qquad (16)$$

$$T_1 = \frac{1}{\omega_B} \, ln \, \frac{\beta_N/\beta_{A2} - \beta_N/\beta_D}{1 - \beta_N/\beta_D} \qquad (17)$$

$$T_2 = \frac{1}{\omega_0'} \, ln \, \frac{1 - \beta_N/\beta_D}{0.10 - \beta_N/\beta_D} \qquad (18)$$

For minimum rise time, it is desirable to

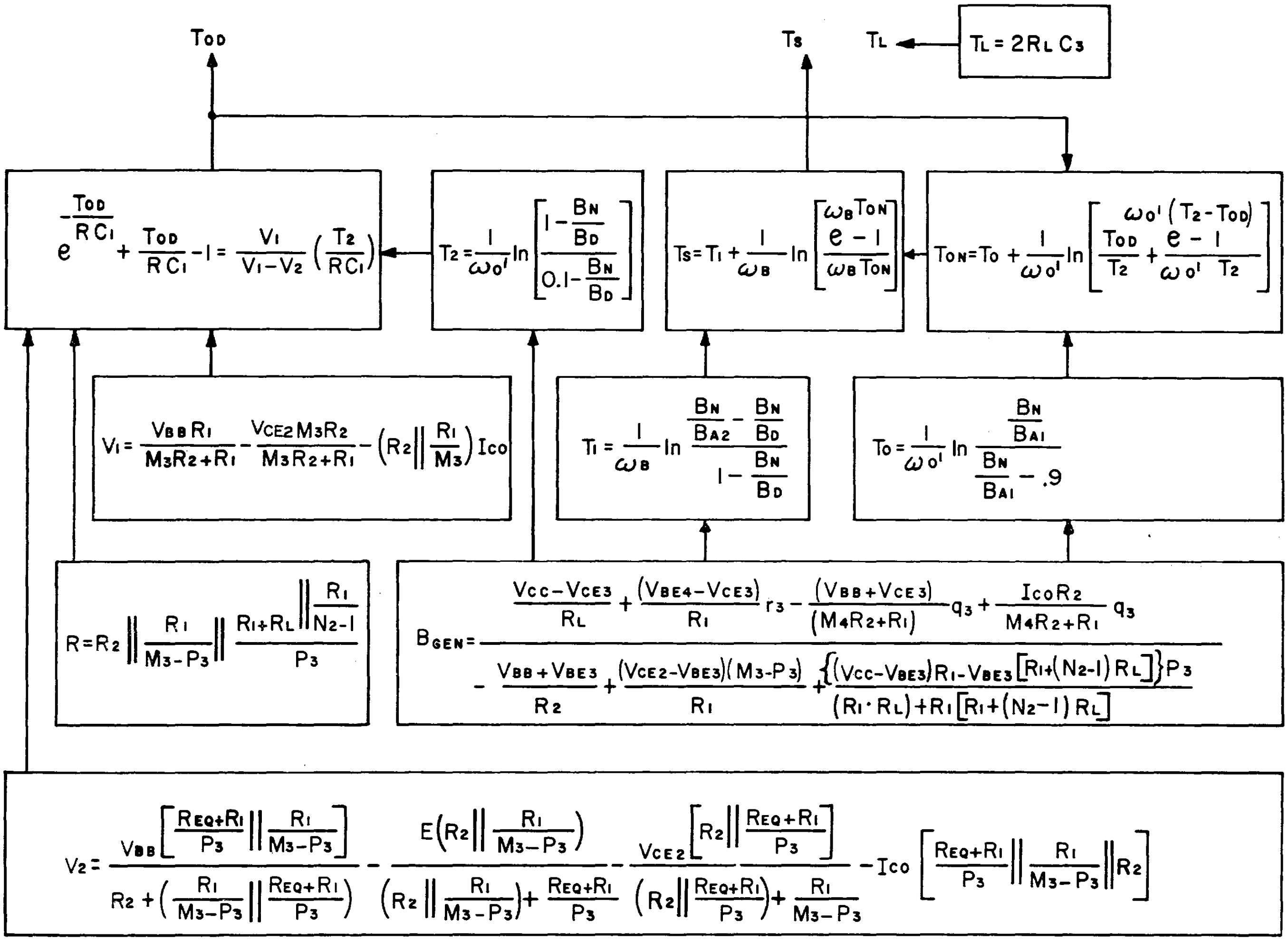

T_{OD}
T_S
T_L
$T_L = 2R_L C_3$
$e^{-\frac{T_{OD}}{RC_1}} + \frac{T_{OD}}{RC_1} - 1 = \frac{V_1}{V_1 - V_2}\left(\frac{T_2}{RC_1}\right)$
$T_2 = \frac{1}{\omega_0'} \ln \left[\frac{1 - \frac{B_N}{B_D}}{0.1 - \frac{B_N}{B_D}}\right]$
$T_S = T_1 + \frac{1}{\omega_B} \ln \left[\frac{e^{\omega_B T_{ON}} - 1}{\omega_B T_{ON}}\right]$
$T_{ON} = T_0 + \frac{1}{\omega_0'} \ln \left[\frac{T_{OD}}{T_2} + \frac{e^{\omega_0'(T_2 - T_{OD})} - 1}{\omega_0' T_2}\right]$
$V_1 = \frac{V_{BB} R_1}{M_3 R_2 + R_1} - \frac{V_{CE2} M_3 R_2}{M_3 R_2 + R_1} - \left(R_2 \| \frac{R_1}{M_3}\right) I_{CO}$
$T_1 = \frac{1}{\omega_B} \ln \frac{\frac{B_N}{B_{A2}} - \frac{B_N}{B_D}}{1 - \frac{B_N}{B_D}}$
$T_0 = \frac{1}{\omega_0'} \ln \frac{\frac{B_N}{B_{A1}}}{\frac{B_N}{B_{A1}} - .9}$
$R = R_2 \| \frac{R_1}{M_3 - P_3} \| \frac{R_1 + R_L \| \frac{R_1}{N_2 - 1}}{P_3}$
$B_{GEN} = \frac{\frac{V_{CC} - V_{CE3}}{R_L} + \frac{(V_{BE4} - V_{CE3})}{R_1} r_3 - \frac{(V_{BB} + V_{CE3})}{(M_4 R_2 + R_1)} q_3 + \frac{I_{CO} R_2}{M_4 R_2 + R_1} q_3}{-\frac{V_{BB} + V_{BE3}}{R_2} + \frac{(V_{CE2} - V_{BE3})(M_3 - P_3)}{R_1} + \frac{\{(V_{CC} - V_{BE3}) R_1 - V_{BE3}[R_1 + (N_2 - 1) R_L]\} P_3}{(R_1 \cdot R_L) + R_1 [R_1 + (N_2 - 1) R_L]}}$
$V_2 = \frac{V_{BB}\left[\frac{R_{EQ} + R_1}{P_3} \| \frac{R_1}{M_3 - P_3}\right]}{R_2 + \left(\frac{R_1}{M_3 - P_3} \| \frac{R_{EQ} + R_1}{P_3}\right)} - \frac{E\left(R_2 \| \frac{R_1}{M_3 - P_3}\right)}{\left(R_2 \| \frac{R_1}{M_3 - P_3}\right) + \frac{R_{EQ} + R_1}{P_3}} - \frac{V_{CE2}\left[R_2 \| \frac{R_{EQ} + R_1}{P_3}\right]}{\left(R_2 \| \frac{R_{EQ} + R_1}{P_3}\right) + \frac{R_1}{M_3 - P_3}} - I_{CO}\left[\frac{R_{EQ} + R_1}{P_3} \| \frac{R_1}{M_3 - P_3} \| R_2\right]$

Fig. 14. Propagation delay time

have a minimum value of β_{A_1}, however, minimum storage time corresponds to a maximum value of β_{A_2}. Minimum values of storage and decay times are obtained with minimum values of β_D.

This section will be concerned with expressing the circuit betas β_{A_1}, β_{A_2}, and β_D in terms of the circuit variables. Fig. 13 shows a general arrangement of TRL circuits from which a general expression for circuit beta can be obtained. Transistor Tr—3, under consideration, has a total of M_3 input resistors. P_3 of these input resistors are connected to TRL circuits in a one state. Each of the P_3 transistors is loaded by N minus one other TRL circuit. The collector of Tr—3 is itself loaded by N_3 input resistors; q_3 of these input resistors may be connected to the positive bases of "off" transistors; r_3 may be connected to the negative bases of "on" transistors. During a particular transient, the arrangement of TRL circuits shown in Fig. 13 is only valid if the coefficients P_3 and q_3 are properly defined; i.e., during storage and decay transients $P_3 = 0$; during rise time transient $q_3 = 0$.

A general expression for circuit beta, β_{GEN}, can be obtained by solving the circuit equations of Fig. 13 for base and collector currents and by making the following assumptions:

1. $V_{BE3} = V_{BE31} = V_{BE32} = V_{BE33} \ldots$
2. $V_{BE41} = V_{BE43} = V_{BE45} \ldots$
3. $V_{BE4} = V_{BE42} = V_{BE44} \ldots$
4. $V_{CE2} = V_{CE21} = V_{CE22} \ldots$
5. $N_2 = N_{21} = N_{22} = N_{23} \ldots$
6. $M_4 = M_{41} = M_{43} \ldots$

$$\beta_{\text{GEN}} = \frac{\frac{V_{CC} - V_{CE3}}{R_L} + \frac{(V_{BE4} - V_{CE3})}{R_1} r_3 - \frac{V_{BB} + V_{CE3} - I_{CO} R_2}{M_4 R_2 + R_1} q_3}{-\frac{V_{BB} + V_{BE3}}{R_2} + \frac{(V_{CE2} - V_{BE3})(M_3 - P_3)}{R_1} + \frac{\{(V_{CC} - V_{BE3}) R_1 - V_{BE3}[R_1 + (N_2 - 1) R_L]\} P_3}{(R_1 \cdot R_L) + R_1 [R_1 + (N_2 - 1) R_L]}} \qquad (19)$$

The turn on circuit beta, β_{A_2}, for storage time calculations is given directly by equation 19; i.e., $\beta_{\text{GEN}} = \beta_{A2}$. During a turn on transient, $q_3 = 0$, and the turn on beta is as shown in equation 20.

$$\beta_{A_1} = \frac{\frac{V_{CC} - V_{CE3}}{R_L} + \frac{(V_{BE4} - V_{CE3})}{R_1} r_3}{-\frac{V_{BB} + V_{BE_3}}{R_2} + \frac{(V_{CE2} - V_{BE3})(M_3 - P_3)}{R_1} + \frac{\{(V_{CC} - V_{BE3}) R_1 - V_{BE3} \times [R_1 + (N_2 - 1) R_L]\} P_3}{(R_1 \cdot R_L) + R_1 [R_1 + (N_2 - 1) R_L]}} \qquad (20)$$

Turn-off beta, B$_D$, can be obtained by letting $P_3 = 0$.

$$\beta_D = \frac{\frac{V_{CC} - V_{CE3}}{R_L} + \frac{(V_{BE4} - V_{CE3})}{R_1} r_3 - \frac{V_{BB} + V_{CE3} - I_{CO} R_2}{M_4 R_2 + R_1} q_3}{-\frac{V_{BB} + V_{BE3}}{R_2} + \frac{(V_{CE2} - V_{BE3})(M_3 - P_3)}{R_1}} \qquad (21)$$

For a particular transistor, a TRL cir-

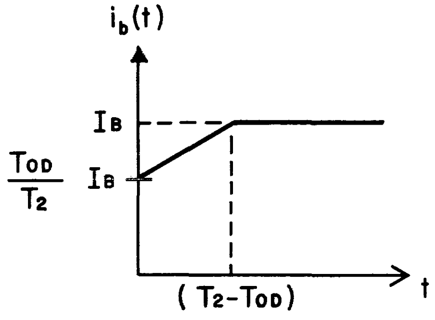


Fig. 15. TRL circuit turn-on base current

cuit has its maximum rise time when I_{B1} is a minimum and β_{A1} is a maximum. It can be seen from Fig. 13 that the arrangement of TRL circuits resulting in a minimum I_{B1} occurs when a transistor is being turned on by only one "off" TRL circuit ($P_3=1$) while the other (M_3-1) inputs are returned to TRL circuits that are on. These M_3-1 inputs shunt the base and reduce the turn-on base current. The collector of the single TRL circuit that is supplying the turn-on current must be loaded by N_2-1 other TRL circuits. Large values of M_3, and N_{21}, N_{22}, ... mean smaller values of I_{B1} and increased transistor rise time.

Maximum transistor storage time, for a particular transistor, occurs when β_{A2} is a minimum and β_D is a maximum; i.e., when the turn-on base current, I_{B1}, is a maximum and turn-off base current, I_{B2}, is a minimum. It again can be seen from Fig. 13 that the arrangement of TRL circuits resulting in a maximum I_{B1} occurs when a transistor is being turned on by a maximum of "off" TRL circuits ($P_3=M_3$). In this case, the collectors of the "off" transistors should be loaded by a single TRL circuit, i.e., N_{21}, N_{22}, ... $=1$.

Minimum I_{B2} occurs when $P_3=0$ and $M_3=1$ in Fig. 13. That $M_3=1$ corresponds to minimum I_{B2} can be explained as follows. During storage time, the base retains essentially the same voltage and polarity as when turn-on base current existed. This voltage causes current in the external circuit that contributes to I_{B2}. This base self-discharge current is a minimum when the external base circuit impedance is a maximum; i.e. $M_3=1$.

It should be noted that the combination of a maximum value of turn-on base current and a minimum value of turn-off current cannot be obtained with a single arrangement of TRL circuits. Maximum I_{B1} occurs when M_3 is a maximum while minimum I_{B2} occurs when $M_3=1$. However, M_3 must have the same value during both turn-on and turn-off. It can be shown that maximum storage time occurs when M_3 is a maximum.

Maximum transistor decay time occurs when β_D is a maximum, that is, when I_{B2} is a minimum. During decay time, I_{B2} again contains a component of current due to base self-discharge. Minimum I_{B2} corresponds to minimum base discharge current. In Fig. 13, when $P_3=0$ and $M_3=1$, I_{B2} has a minimum value.

The two steady-state transistor voltages V_{CE}, the saturated collector to emitter voltage, and V_{BE}, the base to emitter voltage, appearing in equation 19 have been defined by Ebers and Moll[8] as shown in equations 22 and 23.

$$V_{CE} = -\frac{KT}{Q} ln \left[\frac{\alpha_N - \frac{I_{C1}}{I_{B1}}(1-\alpha_N)}{\left|\frac{I_{C1}}{I_{B1}}(1-\alpha_I)+1\right| \frac{I_{C0}}{I_{E0}}} \right] + I_{C1} r_{CV} + (I_{C1}+I_{B1}) r_{eV} \quad (22)$$

$$V_{BE} = I_{B1} r_{BIII} + \frac{KT}{Q} ln \left[1 + \frac{I_{B1}+I_{C1}(1-\alpha_I)}{I_{E0}} \right] + (I_{C1}+I_{B1}) r_{eV} \quad (23)$$

Equation 22 consists of three parts, the first of these represents the emitter and collector junction voltage, the second and third terms represent the voltage drop across the collector and emitter contacts, respectively. The first term increases with the ratio of collector to base currents whereas the second ànd third terms are essentially a function of collector current.

Equation 23 defines the base-to-emitter voltage of the transistor. This expression consists of three voltages, the emitter junction voltage, the voltage drop across the base and emitter contacts, and the voltage developed across the body resistance of the base.

It is possible now to predict the transient performance of the TRL circuit by computing the circuit betas, β_D and β_A, from equations 19, 20 and 21 and applying these computed values to the transient equations 16, 17, and 18. This results in accurate expressions for T_0, T_1, and T_2 in terms of resistance values, transistor parameters and supply voltages. The usefulness of these expressions is further increased by accounting for tolerances, temperature effects, and end-of-life values of the various parameters.

Summary

The propagation delay associated with TRL circuit has been defined in terms of transistor and circuit parameters. It has been shown that transistor rise, decay,

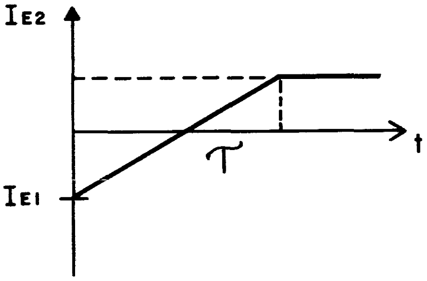


Fig. 16. TRL circuit turn-off emitter current

and storage times can be more accurately predicted by:

1. Measuring transistor parameters under large signal transient conditions rather than small signal r-f measurements at a particular bias point.
2. Considering transistor circuit turn-on and turn-off currents as ramp functions rather than step functions.

The significance of transistor input, collector capacities, and circuit parasitic capacities has been shown.

Fig. 14 summarizes relationships developed in the text, that permit accurate prediction of propagation delay. A computer program based on these relationships is being prepared. The goals of this program are to:

1. Achieve optimum TRL circuit design.
2. Compile propagation delay tables useful to the logical designer.

Appendix

Turn-on Delay Time

If the circuit driving the effective base capacity has an equivalent source reisistance R and the equivalent source voltage changes from V_1 to V_2, the problem is to find the time for the capacitor voltage to reach zero. If the open-circuit driving voltage is considered to vary linearly between V_1 and V_2 and to have the Laplace transform:

$$E_S(S) = (V_1 - V_2) \frac{1-e^{-ST2}}{S^2 T_2} \quad (24)$$

the turn-on time will be the time to charge the capacitor from zero to V_1 volts. Since the charging rate of the capacitor is determined in the frequency domain

$$\frac{E_0(S)}{E_S(S)} = -\frac{1}{1+RC_1 S} \quad (25)$$

the desired discharge time is the solution of

$$\mathcal{L}^{-1}\left[\frac{V_1-V_2}{T_2} \frac{1-e^{-ST_2}}{S^2(1+RC_1S)}\right] = V_1 \quad (26)$$

or

$$\left[e^{-T_{0D}/RC_1} + \frac{T_{0D}}{RC_1} - 1\right] = \frac{V_1}{V_1-V_2} \frac{T_2}{RC_1} \quad (27)$$

for $T_{0D} \leqslant T_2$

and

$$\frac{T_{0D}}{RC_1}=ln\left[\left(\frac{e^{T2/RC_1}-1}{T_2/RC_1}\right)\left(\frac{V_2-V_1}{V_2}\right)\right] \tag{28}$$

for $T_{0D} \geqslant T_2$

In Fig. 13, the capacitance from base to ground will be initially charged to a voltage

$$V_1=\frac{V_{BB}R_1-V_{CE2}M_3R_2}{M_3R_2+R_1}-\left(R_2\left\|\frac{R_1}{M_3}\right.\right)I_{C0} \tag{29}$$

If P_3 of the inputs are being driven, the effective charging resistance will be the parallel combination of R_2 with $R_1/(M_3-P_3)$ and

$$R_1+\left(R_L\left\|\frac{R_1}{N_2-1}\right.\right)/P_3$$

$$R=R_2\left\|\frac{R_1}{(M_3-P_3)}\right\|\frac{R_1+\left(R_L\left\|\frac{R_1}{N_2-1}\right.\right)}{P_3} \tag{30}$$

Assuming $N_2=N_{21}=N_{22}\ldots$, each input will appear to be turned on with a voltage of

$$E=\frac{\frac{R_1}{N_2-1}}{\frac{R_1}{N_2-1}+R_L}(V_{CC}-V_{BE3})+V_{BE3} \tag{31}$$

applied through a resistance of

$$R_{EQ}=R_L\left\|\frac{R_1}{N_2-1}\right. \tag{32}$$

Thus the total voltage toward which the base capacitance will charge for M_3 inputs, P_3 of which are driven is given by:

$$V_2=-\frac{E\left(R_2\left\|\frac{R_1}{M_3-P_3}\right.\right)}{\left(R_2\left\|\frac{R_1}{M_3-P_3}\right.\right)+\frac{R_{EQ}+R_1}{P_3}}-V_{CE2}\frac{\left(R_2\left\|\frac{R_{EQ}+R_1}{P_3}\right.\right)}{\left(R_2\left\|\frac{R_{EQ}+R_1}{P_3}\right.\right)+\frac{R_1}{M_3-P_3}}-I_{C0}\left[\frac{R_{EQ}+R_1}{P_3}\left\|\frac{R_1}{M_3-P_3}\right\|R_2\right]+\frac{V_{BB}\left(\frac{R_{EQ}+R_1}{P_3}\left\|\frac{R_1}{M_3-P_3}\right.\right)}{\left[R_2+\frac{R_1}{M_3-P_3}\left\|\frac{R_{EQ}+R_1}{P_3}\right.\right]} \tag{33}$$

The second, third, and fourth terms in equation 33 represent the contributions of (M_3-P_3) "on" collector voltages, I_{C0} and the bias voltage V_{BB} respectively to the final open circuit voltage at the base.

Turn on Rise Time

Once the base voltage has fallen to zero the transistor is turned on with a current as shown in Fig. 15.

The Laplace transform of this current is:

$$I_B(S)=I_B\left[\frac{T_{0D}}{T_2}\frac{1}{S}+\frac{1-e^{-S(T_2-T_{0D})}}{S^2T_2}\right] \tag{34}$$

and the transform of the collector current is:

$$I_C(S)=\frac{\alpha_N\omega_0'}{1-\alpha_N}\frac{I_b(S)}{S+\omega_0'} \tag{35}$$

for a load impedance small compared with the output impedance of the transistor. Thus,

$$i_c(t)=\frac{\alpha_N I_B}{1-\alpha_N}\times\left[1-\left(\frac{T_{0D}}{T_2}+\frac{e^{\omega_0'(T_2-T_{0D})}-1}{\omega_0'T_2}\right)e^{-\omega_0't}\right] \tag{36}$$

which equals 0.9 I_C when

$$T_{0N}=\frac{1}{\omega_0'}1n\left[\frac{\frac{T_{0D}}{T_2}+\frac{e^{\omega_0'(T_2-T_{0D})}-1}{\omega_0'T_2}}{1-0.9\,\frac{I_C(1-\alpha_N)}{I_B\alpha_N}}\right] \tag{37}$$

Modification of Moll's Storage Equation for Finite Transition Time of Turn Off Current

Moll computes the time for the component of current collected at the emitter junction, $I_{ER}(t)$, to become zero after the emitter current steps from I_{E1} while the collector current remains at a constant value of I_C. The transform of the component of current collected by the emitter is given as:

$$I_{er}(S)=-\alpha_I(S)\frac{I_C(S)+\alpha_N(S)I_E(S)}{1-\alpha_N(S)\alpha_I(S)} \tag{38}$$

where $\alpha_N(S)$ and $\alpha_I(S)$ are the normal and inverse current transfer functions for the base region.

If instead the turn-off transient is as shown in the Fig. 16, then

$$I_{er}(t)=-\frac{\alpha_I(I_C+\alpha_N I_{E2})}{(1-\alpha_N\alpha_I)}+\left(\frac{\alpha_N\alpha_I}{1-\alpha_N\alpha_I}\right)(I_{E2}-I_{E1})\times\left[\frac{(e^{\omega_A\tau}-1)/\omega_A\tau}{(1-\omega_A/\omega_B)}e^{-\omega_A t}+\frac{(e^{\omega_B\tau}-1)/\omega_B\tau}{(1-\omega_B/\omega_A)}e^{-\omega_B t}\right] \tag{39}$$

which equals zero when

$$\frac{I_{C1}+\alpha_N I_{E2}}{\alpha_N(I_{E2}-I_{E1})}=\frac{\left(\frac{e^{\omega_A\tau}-1}{\omega_A\tau}\right)}{1-\omega_A/\omega_B)}e^{-\omega_A T_S}+\frac{\left(\frac{e^{\omega_B\tau}-1}{\omega_B\tau}\right)}{(1-\omega_B/\omega_A)}e^{-\omega_B T_S} \tag{40}$$

This corresponds to Moll's equation 52 where ω_A and ω_B are roots of:

$$\omega^2-\omega(\omega_N+\omega_I)+\omega_N\omega_I(1-\alpha_N\alpha_I)=0 \tag{41}$$

Then to the approximation that ω_A is large compared with ω_B, the storage time in terms of the base current is

$$T_S=\frac{1}{\omega_B}1n\left[\left(\frac{I_{B1}-I_{B2}}{\frac{I_C(1-\alpha_N)}{\alpha_N}-I_{B2}}\right)\left(\frac{e^{\omega_B\tau}-1}{\omega_B\tau}\right)\right] \tag{42}$$

Nomenclature

M = number of inputs to a TRL circuit
P = number of inputs being energized
N = number of outputs from one TRL circuit
T_M = delay time of a pulse traveling through two cascaded TRL circuits
I_{B_1} = turn-on base current
I_{B_2} = turn-off base current
I_C = collector current at edge of saturation region
I_{C_1} = collector current before the beginning of the turn-off
T_0 = rise time, measured between 0% and 90% points
T_2 = decay time, measured between 10% and 90% points
T_1 = storage time, measured from the time when $I_{B_1}=0$ to the time when $V_{CB}=0$
α_N = ratio of I_C/I_E with the transistor in the normal direction
α_I = ratio of I_E/I_C with the transistor inverted
ω_N = small-signal cut-off frequency in radians per second with the transistor in the normal direction
ω_I = small-signal cut-off frequency in radians per second with the transistor inverted
f_o = characteristic cut-off frequency during a large transient
t_s = characteristic storage time, measured when $I_{B_1}=Ic_1$ and $I_{B_2}=0$
t_r = characteristic rise time, measured as the transistor swings from cut-off point to the edge of saturation, $V_{CB}=0$, 0%–90%
ω_B = characteristic cut-off frequency in the saturated region
I_{C0} = leakage current from collector to base with the emitter open
I_{E0} = leakage current from emitter to base with the collector open
K = Boltzman's constant: 1.38×10^{-23}
T = degrees Kelvin: 273 centigrade
Q = electron charge: 1.6×10^{-19} coulombs
I_E = emitter current: I_C+I_B
β_{A_1} = turn-on circuit beta: I_C/I_{B_1}
β_{A_2} = turn-on circuit beta: Ic_1/I_{B_1}
β_D = turn-off circuit beta: Ic_1/I_{B_2}
r_{C_v} = collector contact resistance
C_C = collector capacity
r_{BIII} = base resistance as measured in the current saturation region
r_{eV} = emitter contact resistance
ω_0 = characteristic cut-off frequency in radans per seconds
$\beta_N=\frac{\alpha_N}{1-\alpha_N}$
V_{BE} = base-to-emitter voltage
V_{CE} = collector-to-emitter voltage
TRL = transistor-resistor logic
Enhancement when $I_{B_1}>\frac{I_C}{\beta_N}$

Note: Other symbols found in the text are specifically defined by the diagram on which they occur.

References

1. Millimicrosecond Transistor Current Switching Circuits, H. S. Yourke. *Transactions* Professional Group on Circuit Theory, Institute of Radio Engineers, New York, N. Y., vol. CT-4 no. 3, Sept. 1957.

2. Surface-Barrier Transistor Switching Circuits, **R. H. Beter, W. L. Bradley, M. Rubinoff.** *Convention Record,* Institute of Radio Engineers, pt. 4, 1955, pp. 139–45.

3. Large-Signal Transient Response of Junction Transistors, **J. L. Moll.** *Proceedings,* Institute of Radio Engineers, vol. 42, Dec. 1954, pp. 1773–748.

4. Junction Transistor Theory (book), **R. D. Middlebrook.** John Wiley & Sons, Inc., New York, N. Y., 1957, pp. 154–63.

5. Effect of Nonlinear Collector Capacitance on Collector Current Rise Time, **T. R. Bashkow.** *Transactions,* Professional Group on Electronic Devices, Institute of Radio Engineers vol. ED-3, no. 4, Oct. 1956, pp. 167–172.

6. The Effect of Collector Capacity on the Transient Response of Junction Transistors, **J. W. Easley.** *Ibid.,* vol. ED-4, no. 1, Jan. 1957, pp. 6–14.

7. Hole Storage Delay Time and Its Prediction, **C. D. Simmons.** *Semiconductor Products,* May/Jun. 1958, pp. 14–18.

8. Large-Signal Behavior of Junction Transistors, **J. J. Ebers, J. L. Moll.** *Proceedings,* Institute of Radio Engineers, vol. 42, Dec. 1954, pp. 1761–772.

The Recording, Checking, and Printing of Logic Diagrams

M. KLOOMOK P. W. CASE H. H. GRAFF

IN THE design and development of today's complex computers, the ratio of routine and repetitive work to creative engineering is getting larger and larger. The factors which make the development of a large scale computer a lengthy and time-consuming process are more and more the sheer mass of detail work that must be performed; the drafting and checking of logic diagrams; the assignment of circuit components to printed cards, of printed cards to panels, of panels to gates and frames; the routing of signal and power wiring between the thousands of circuit components; the production of the many pieces of paper needed to convey to manufacturing, production control, cost engineering, maintenance engineering, etc., the details they require to do their work. There would be every reason to question the feasibility of embarking on a new development project were it not for the fact that a considerable number of these detail tasks are practical computer applications.

International Business Machines Corporation (IBM) is using *700* series computers (*704, 705*) to assist in the design, development, release, and product engineering of the new *7000* series of general-purpose data-processing machines. This system of computer programs is referred to as the "Design Mechanization System." In this system the computer does not replace the design judgment and knowledge of the engineer. Its function is to eliminate the manual repetitive execution of established procedures following rules laid down by the design engineer. IBM is also using *700* series computers to assist the engineer in the design of logic. Computer assistance in logic minimization, logic implementation, logic simulation, and physical placement is the subject of future papers and is not considered here.

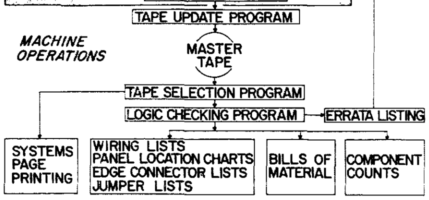


Fig. 1. Component parts of the design mechanization system

Component Parts of the Design Mechanization System

The phases of the design mechanization system discussed herein are shown in Fig. 1. The development engineer in designing computer logic presents his ideas on a system drawing. This rough sketch is then coded for key punching and punched into cards. The cards are read into the computer and the master tape is updated. The master tape contains a complete historical record of the design to date and is the source of all data for succeeding programs. The "master tape select program" selects from the master tape any desired section of the machine for further processing.

The "logic checking program" operates on a complete section of machine logic and checks for adherence to design rules. In addition to checking for proper circuit configurations, many logical checks can be made to test for redundancies and proper page-to-page notation. Most of the rules established in a given machine design program can be inserted into this process. The output, in the form of an errata list, can then be returned to the design engineer for his corrective action.

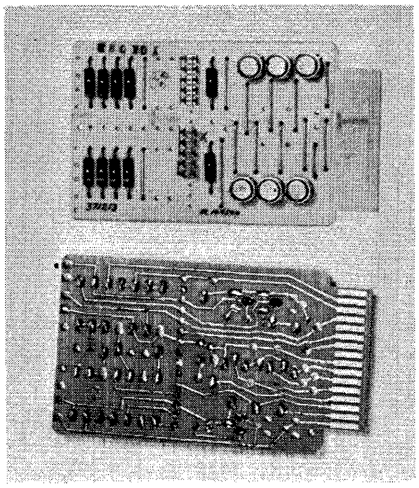


Fig. 2. Basic components of SMS, printed circuit cards

The "systems page print program" converts the coded logic record appearing on the master tape into a computer-prepared systems drawing in block diagram form. Preparing the printed logic page by machine saves large numbers of man-hours over the drafting process formerly used. Other advantages are greater speed of preparation, increased accuracy, a uniform quality of printing regardless of the number of changes to which a given page has been subjected, and the ability to recreate an old engineering level page when needed.

The "panel wiring program," operating on a section of checked logic, routes the necessary panel wires in such a fashion that lead lengths are minimized, taking into consideration capacitive loading and interwire noise. The output is a panel wiring chart to be used as a release document for construction of the panel. A secondary objective is the ability to alter previously computed panel configurations to accommodate "engineering changes," and to make updated panel wiring lists.

M. Kloomok, P. W. Case, and H. H. Graff are with International Business Machines Corporation, Poughkeepsie, N. Y.

The "bill of material generation program" uses the checked logic to generate bills of material for electrical components. This is merely a matter of extracting data from the master tape about the location and quantity of component circuits, and referring to tables to assemble the proper information in the desired format. This program also must provide for handling engineering changes.

The design philosophy which makes design mechanization a practical reality is the far-reaching standards program adopted at IBM. The key to this standards program is the "standard modular system" (SMS). Before going into the details of the design mechanization system it is worthwile dwelling briefly on SMS.

Standard Modular System

The standard modular system is an engineering program for uniform application of solid-state technology in a new generation of IBM data processing machines. SMS makes possible a flexible, standardized packaging system for IBM's new *7000* series transistorized computers.

The basic component of SMS is the printed circuit card shown in Fig. 2. This card is pluggable into sockets on a standard panel. The panels in turn are packaged into one of two standard modular frames. Fig. 3 shows a sliding gate that fits into the larger of these frames. Four standard panels are seen on the gate.

By selecting appropriate quantities of either of the standard frames, packaging is available for data processing equipment ranging from desk size to the giant *7000* series.

A standard range of circuits is used throughout for logical design, so that the logical design process consists of selecting pre-designed standard circuit cards to implememt the logic.

SMS and design mechanization complement each other, since standard packages make possible the use of more generalized computer-aided design techniques. These techniques in turn enable more sophisticated automated manufacturing methods to be used. In addition, the standard circuits lend themselves to computer and checking techniques.

Input

Logic Sketch Form

The input document to the design mechanization system is the logic sketch form. This form has been designed with several objectives in mind:

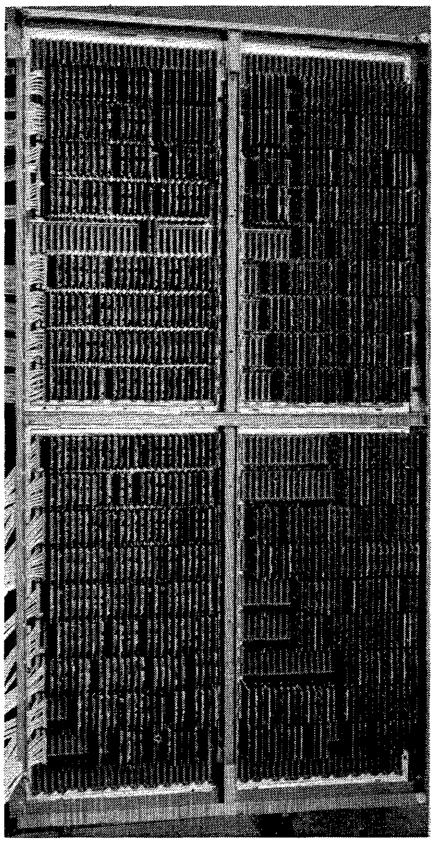


Fig. 3. SMS gate with four standard panels

1. Clarity of logic flow. The page is large enough to enable the development engineer to present a logical entity, and not so large as to appear cluttered, overcrowded, or confusing to the maintenance engineer who must service the equipment using these documents.

2. The page must be of a size to be printed by the standard IBM line printers.

3. Clarity of presentation for the nontechnical people who convert the information into punched cards.

The resultant form is shown in Fig. 4(A). A maximum of 45 logic blocks are permitted, arranged in a 5×9 array. A co-ordinate grid is superimposed for ease of reference to any given circuit block. Provision is made at the top of the page for the required identifying information and at the bottom for comments. The numbers in the background indicate the number of alphanumeric characters permitted for each name. Thus, for example, each line entering or leaving the page can have an identifying name 30 characters long, expressed in two lines of 15 characters each.

The form is printed on vellum with the guiding information printed in nonreproducible blue ink. In using the form, the engineer marks in freehand the blocks he wishes to use, the lines between the blocks, and the off page lines as in Fig. 4(B). He then obtains a facsimile of the drawing for the design mechanization system, retaining the original for his own use until presented with a final drawing, prepared by the computer, see Fig. 5(A). This final drawing is also on vellum.

When a change or correction is made to any page, the computer-produced vellum is marked up as shown in Fig. 5(B) and resubmitted as an engineering change. A new computer-produced vellum master is then prepared.

Transcribing

The conversion of the information on the logic sketch forms into digital data for key punching is effected by a group of clerks called transcribers. The transcribing form is shown in Fig. 6(A). The numerical designations on the transcribing form refer to the card columns into which this information is key punched. No knowledge of circuit logic or convention is required and familiarity with the transcribing rules can be achieved by an average clerk in about one week.

Each logic sketch sheet is encoded into the transcribing form in the following sequence as in Fig. 6(B).

1. Heading Information. Space is provided at the top of the transcription sheet to record the identifying information.

2. Information in the Circuit Blocks. The fields in the body of the form are blocked out to indicate the allowable number of characters and are numbered to indicate the card columns into which the information is to be key punched.

The circuit blocks are identified by their co-ordinates on the sketch sheet and transcribed in order, from top to bottom and right to left. There are six lines of alphanumeric information inside each circuit block and the columns labelled Line 1, Line 2, . . . Line 6 on the transcribing form indicate where this information is to be written.

3. Signal Lines. After all circuit blocks are transcribed, the lines between blocks are encoded. All lines are described in consecutive columns of the transcribing sheet, and the titles have no meaning here.

All lines are considered coming from, rather than to, a block with lines entering the page on the left considered coming from a set of pseudo blocks on the left side of the page. Lines are described by their beginning and ending co-ordinates. Line routing is performed by computer.

4. Supplementary Information. Space is provided at the bottom of the transcribing form for the comments indicated on the logic sketch sheet.

The Master Tape

Master Tape Format

The heart of the design mechanization system is the master tape which contains a complete historical record, block by

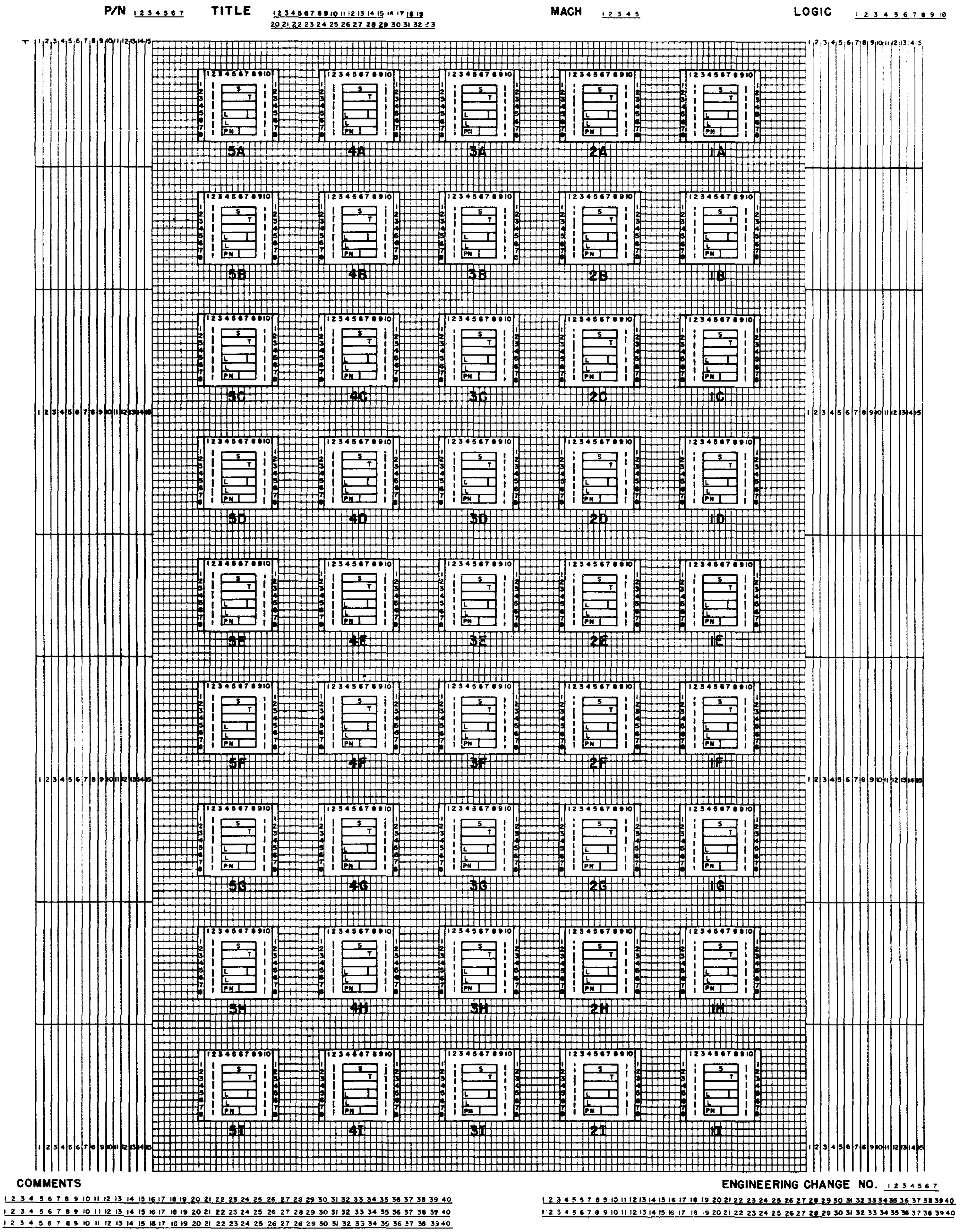


Fig. 4(A). Logic sketch sheet

block and page by page, of the machine logic. This master tape is updated periodically as necessary and is kept current from the early stages of development until the machine is serving customers in the field.

The organization of the master tape is illustrated in Fig. 7. The first record on the tape is a machine identification. Separate master tapes are kept for each machine using the Design Mechanization System. This is followed by a record for each page, including the name, part number, comments, etc. Following the page record is a separate record for each block on the page. Blocks are described in the order in which they are used on the logic sketch sheet. Only the blocks which are actually used are recorded on tape. Block information is kept in order of most recent engineering level, but information at all levels is always retained. There are two reasons for maintaining this historical information.

1. During development, an engineer at his own discretion can discard any logical design and revert to an old configuration.

2. During production there may be machines in the field at many different levels, depending on the date of manufacture or on the customer's choice of special features.

The record length for each logic block is variable, depending primarily on the number of output lines. An average block, with one output line requires 70 characters. A typical machine with 1,000 pages at a representative number of engineering levels requires about 1,200 feet of tape.

Master Tape Updating Program

Updating is performed each day, if any activity took place, by writing a new master tape, merging the information on the old tape with the cards prepared in the transcription process. During the updating run, provision is made to rearrange any section of the master tape, renumber pages, rearrange blocks, etc., at the discretion of the engineer. Of particular interest is a page rearranging routine. This routine can be run wherever it is desired to alter the arrangement of the page in the interest of clarity or appearance.

Master Tape Selection Program

During the master tape updating run, a "request card" is punched for each page affected during the run. The request card contains the page number, part number, and engineering level. A file is kept of these cards for use with the selection program which prepares a new tape of any page or section of machine at any engineering level. This selected tape is the input for the print program or the checking program or both. Provisions are made for selecting all pages at their latest engineering level, when it is desirable to avoid inserting large quantities of request cards.

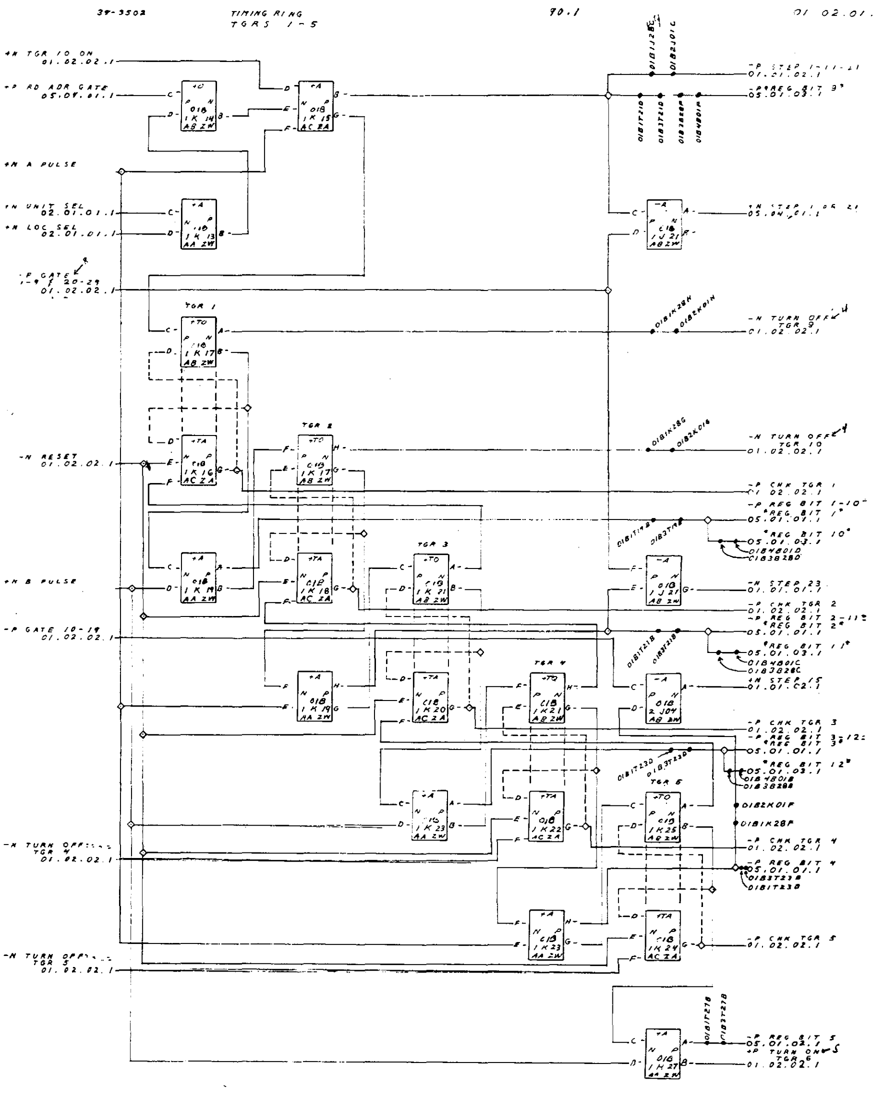


Fig. 4(B). Logic sketch sheet with typical entries by computer

Print Program

Page Size

The logic diagram shown in Figs. 5(A) and (B) was printed from tape on an IBM *717* printer. The only modification to the printer was the addition of a set of six special symbols to each type wheel.

The circuit blocks printed on the page are constant in size, 6 characters wide and 7 lines high. Two factors govern the size of the blocks; the necessity for printing all pertinent data, logical function, transistor type, physical location, circuit part number, etc. in a clear and legible format, and the desire to keep the block size to a minimum in order to increase the number of blocks which could be printed on the page. Space is provided to insert seven vertical lines between adjacent blocks, and ten horizontal lines.

The *717* printer, which is the standard line printer for the *704* and *705*, has 120 type wheels, thus the maximum number of characters that can be printed across the page is 120. The length of the page is determined in a sense by the width, since the final aspect ratio must approximate a "B" size drawing (17×22 inches).

Considering the space that must be allowed for off-page line names, for lines between the blocks, and for other necessary identifying information, the optimum page size was chosen as 120 characters by 186 characters. This results in a 5×9 array of circuit blocks.

The vertical size of the page could be increased without limit, of course, merely by printing more lines. The horizontal dimension could be increased by turning the type faces 90 degrees on the type wheels, thus interchanging the width and length. In either case, however, the desirable aspect ratio is destroyed.

The vellum produced on the printer is 15×24 inches. During development and

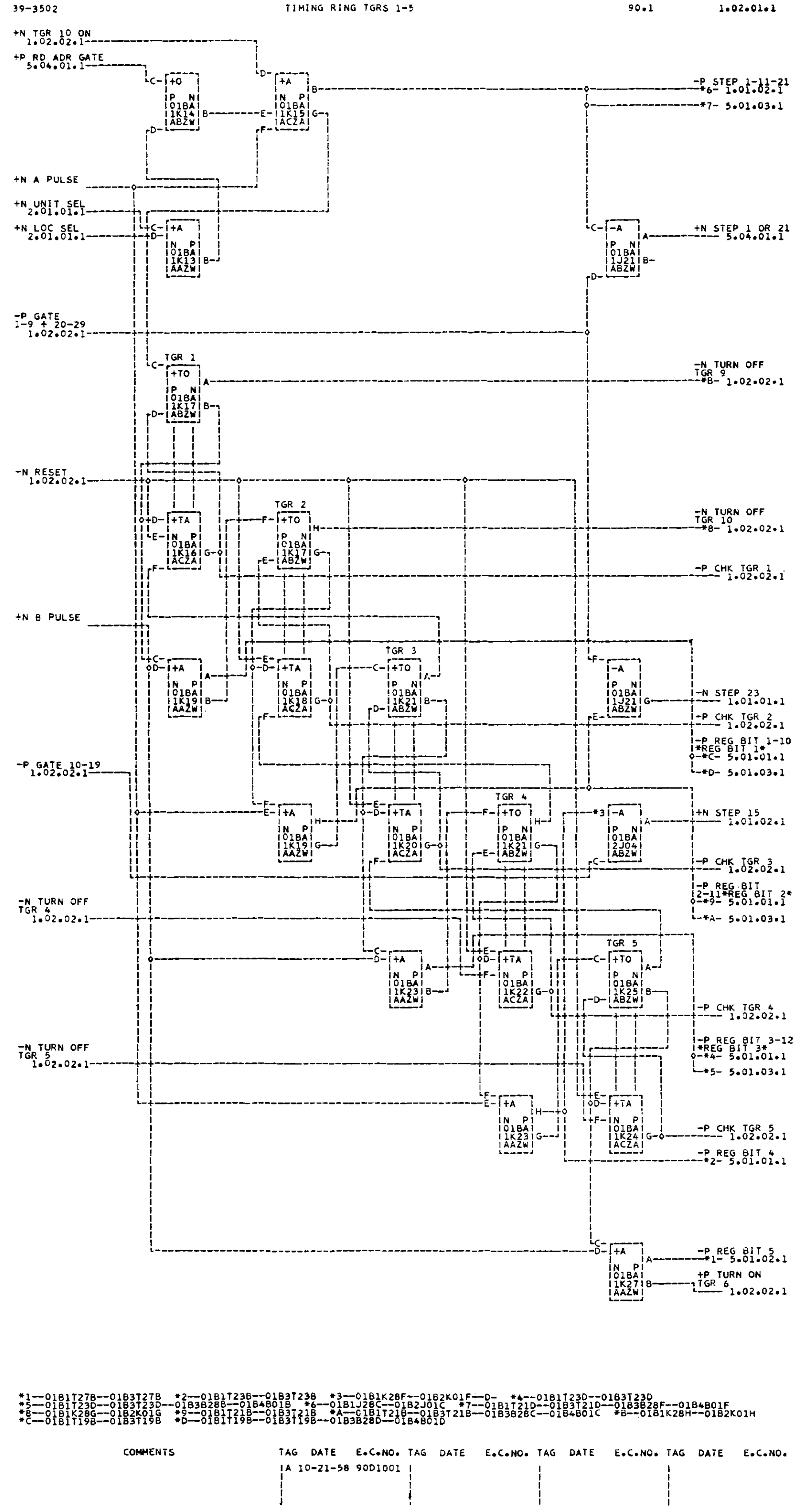

39-3502
TIMING RING TGRS 1-5
90.1
1.02.01.1
+N TGR 10 ON 1.02.02.1
+P RD ADR GATE 5.04.01.1
+N A PULSE
+N UNIT SEL 2.01.01.1
+N LOC SEL 2.01.01.1
-P GATE 1-9 + 20-29 1.02.02.1
-N RESET 1.02.02.1
+N B PULSE
-P GATE 10-19 1.02.02.1
-N TURN OFF TGR 4 1.02.02.1
-N TURN OFF TGR 5 1.02.02.1
TGR 1
TGR 2
TGR 3
TGR 4
TGR 5
-P STEP 1-11-21 *6- 1.01.02.1
*7- 5.01.03.1
+N STEP 1 OR 21 5.04.01.1
-N TURN OFF TGR 9 *8- 1.02.02.1
-N TURN OFF TGR 10 *8- 1.02.02.1
-P CHK TGR 1 1.02.02.1
-N STEP 23 1.01.01.1
-P CHK TGR 2 1.02.02.1
-P REG BIT 1-10 *REG BIT 1* *C- 5.01.01.1
*D- 5.01.03.1
+N STEP 15 1.01.02.1
-P CHK TGR 3 1.02.02.1
-P REG BIT 2-11*REG BIT 2* *9- 5.01.01.1
*A- 5.01.03.1
-P CHK TGR 4 1.02.02.1
-P REG BIT 3-12 *REG BIT 3* *4- 5.01.01.1
*5- 5.01.03.1
-P CHK TGR 5 1.02.02.1
-P REG BIT 4 *2- 5.01.01.1
-P REG BIT 5 *1- 5.01.02.1
+P TURN ON TGR 6 1.02.02.1
*1--01B1T27B--01B3T27B *2--01B1T23B--01B3T23B *3--01B1K28F--01B2K01F--D- *4--01B1T23D--01B3T23D
*5--01B1T23D--01B3T23D--01B3B28B--01B4B01B *6--01B1J28C--01B2J01C *7--01B1T21D--01B3T21D--01B3B28F--01B4B01F
*8--01B1K28G--01B2K01G *9--01B1T21B--01B3T21B *A--C1B1T21B--01B3T21B--01B3B28C--01B4B01C *B--01B1K28H--01B2K01H
*C--01B1T19B--01B3T19B *D--01B1T19B--01B3T19B--01B3B28D--01B4B01D
COMMENTS
TAG DATE E.C.NO. TAG DATE E.C.NO. TAG DATE E.C.NO. TAG DATE E.C.NO.
A 10-21-58 90D1001

Fig. 5(A). Machine-printed systems page

pilot engineering stages, these drawings are used directly or reproduced in small quantities. For larger volume production and for use in the field these drawings are reduced to 11×17 inches ("B" size).

Spatial Arrangement of Circuit Blocks

The print program does not in any way alter the spatial arrangement of the circuit blocks, producing a page identical (as far as the logic blocks are concerned) to the logic sketch sheet which the engineer designed. The task of arranging the blocks on a page by computer is not insurmountable, but two disadvantages would be experienced. First, when the engineer receives a page of logic blocks from the printer, which is arranged differently than the page he submitted in the first place, he would have difficulty in following the logic flow. Second, when a page is changed by the addition or deletion of a block, it would be likely that the computer would again rearrange the page spatially. Thus it was decided to preserve the spatial arrangement chosen by the engineer.

Spatial Arrangement of Circuit Lines

Although the arrangement of the circuit blocks is not altered by the computer, except on request, the print program does route all the electrical lines on the page. It was found to be cheaper and faster to specify the lines by computer than to record the co-ordinates of all line segments as drawn by the engineer and reproduce the lines when the page is printed. Furthermore, the recording of the lines is a manual job, subject to human error.

Line Representation

Six special characters have replaced standard symbols on the *717* type wheel. These are shown in Fig. 8. The corners enable lines to bend in any direction. The diamond denotes a junction of two lines. Line crossings are represented by a conventional plus symbol. Vertical and horizontal dashes makes up line segments.

Page Image in Memory

An exact image of the final page is developed in memory, character by character. This necessitates a memory allocation of 22,320 characters (120×186) independent of computer aud printed instructions. It takes an average of 45 seconds of computer time to develop a page image in memory and write this image on tape for off-line printing. It takes 75 seconds to print each page on the 150-line per minute off-line printer.

Logic Checking Program

Objectives

The checking program is designed to verify that the logical configuration conforms to a set of rules specified by the engineer. Secondarily, of course, the checking program must also detect (and where possible, correct) the errors introduced into the system by the manual processes previously described.

The design rules can be divided into three categories:

1. Circuit rules (Appendix I).
2. Logic format rules (Appendix II).
3. Packaging rules (Appendix III).

Circuit Rules

The circuit rules involved detailed examination of the circuit interconnections to ensure that workable circuit combinations exist. This is called a "circuit compatibility" check. A good deal of effort has been expended in this area to indicate and spotlight the specifics of the incompatibility so that the engineer can identify and correct the difficulty with as little time expenditure as possible. It was felt that a simple yes or no answer to the compatibility question would be too expensive in terms of engineering man-hours.

It should be noted that the philosophy of the program is to identify deviations from the rules and call them to the attention of the designer. When sufficiently sophisticated programs are available, a different philosophy in which the computer does a good deal of the initial design will obviously be in order.

Logic Format Rules

These rules involve checks to ensure that the proper logic format has been followed in minute detail in the finished logic drawings. These rules make possible and useful the machine-produced logic drawing described previously. If this checking were not done, the machine drawing would be only a neat version of the designer's sketch, with the possibility of errors added in preparing it for processing. As a result of this checking, it is assured that the logic is presented in a uniform manner and that it is accurately described. Transpositions of characters and clerical errors are eliminated, making the logic description a reliable document for designing and servicing, and enabling it to serve as an accurate information source for the automatic preparation of necessary secondary documents.

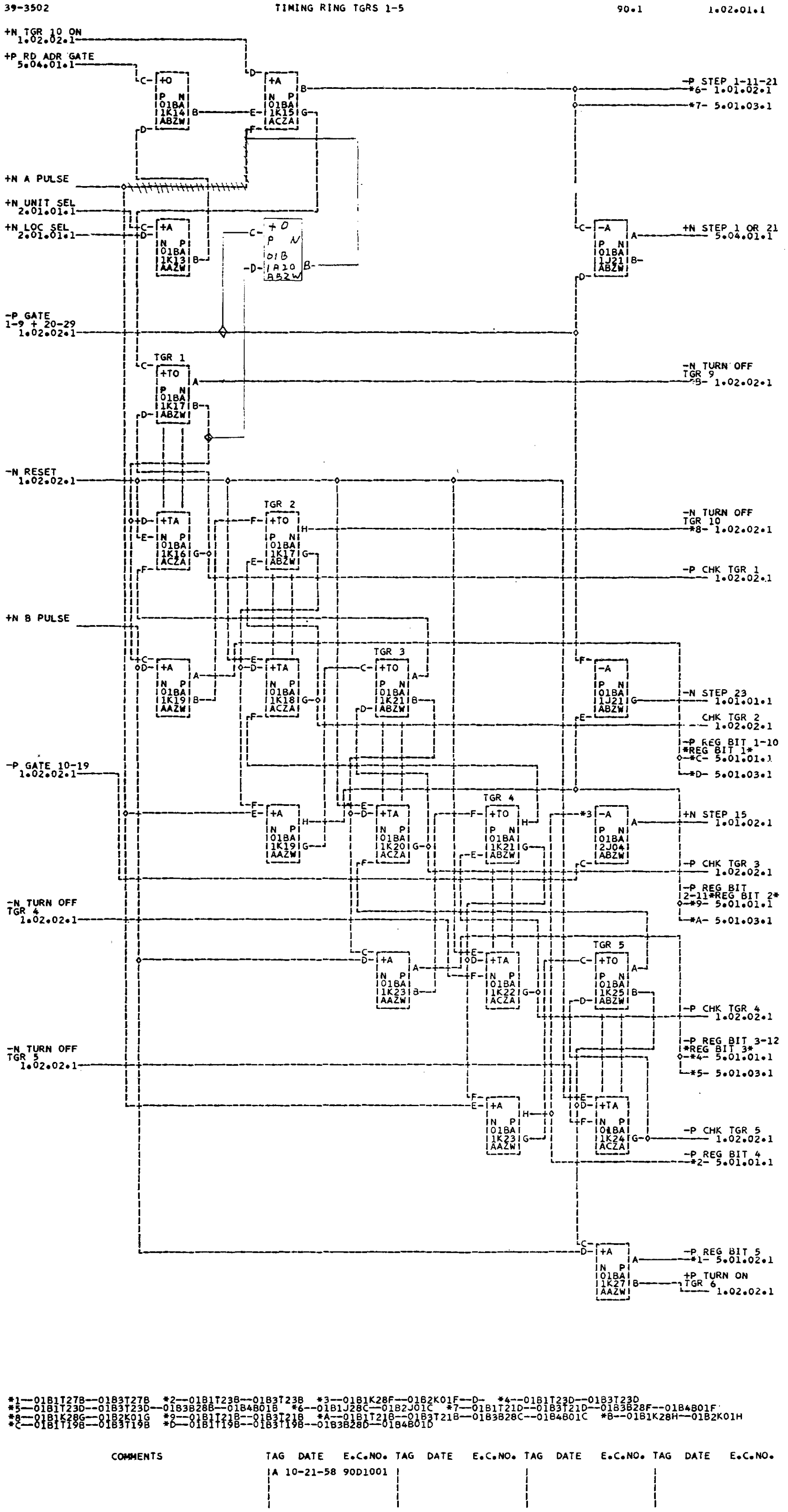
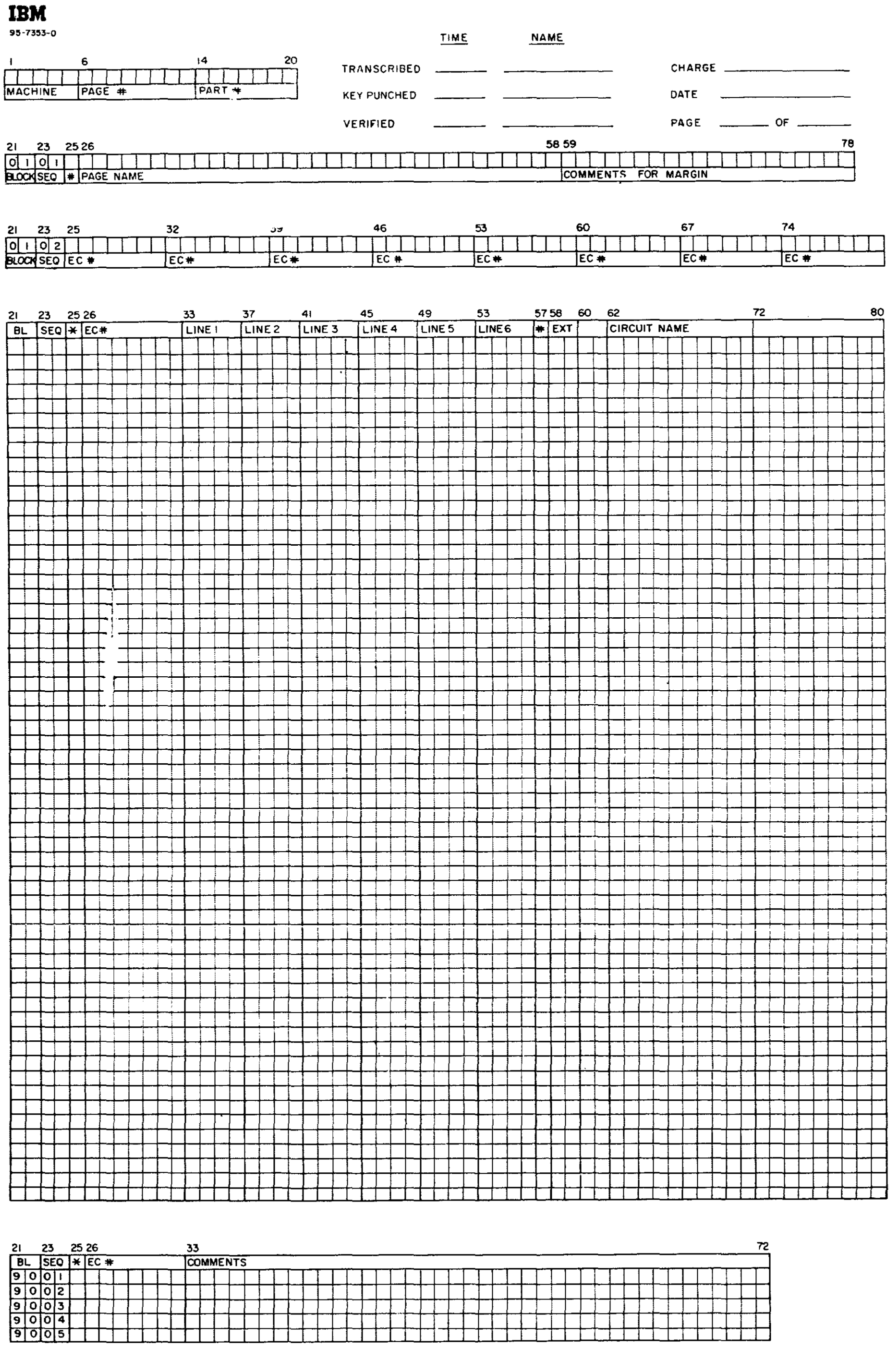

39-3502
TIMING RING TGRS 1-5
90.1
1.02.01.1
+N TGR 10 ON 1.02.02.1
+P RD ADR GATE 5.04.01.1
+N A PULSE
+N UNIT SEL 2.01.01.1
+N LOC SEL 2.01.01.1
-P GATE 1-9 + 20-29 1.02.02.1
-N RESET 1.02.02.1
+N B PULSE
-P GATE 10-19 1.02.02.1
-N TURN OFF TGR 4 1.02.02.1
-N TURN OFF TGR 5 1.02.02.1
TGR 1
TGR 2
TGR 3
TGR 4
TGR 5
-P STEP 1-11-21 *6- 1.01.02.1
*7- 5.01.03.1
+N STEP 1 OR 21 5.04.01.1
-N TURN OFF TGR 9 *B- 1.02.02.1
-N TURN OFF TGR 10 *8- 1.02.02.1
-P CHK TGR 1 1.02.02.1
-N STEP 23 1.01.01.1
CHK TGR 2 1.02.02.1
-P REG BIT 1-10 *REG BIT 1* *C- 5.01.01.1
*D- 5.01.03.1
+N STEP 15 1.01.02.1
-P CHK TGR 3 1.02.02.1
-P REG BIT 2-11*REG BIT 2* *9- 5.01.01.1
*A- 5.01.03.1
-P CHK TGR 4 1.02.02.1
-P REG BIT 3-12 *REG BIT 3* *4- 5.01.01.1
*5- 5.01.03.1
-P CHK TGR 5 1.02.02.1
-P REG BIT 4 *2- 5.01.01.1
-P REG BIT 5 *1- 5.01.02.1
+P TURN ON TGR 6 1.02.02.1
*1—01B1T27B—01B3T27B *2—01B1T23B—01B3T23B *3—01B1K28F—01B2K01F—D- *4—01B1T23D—01B3T23D
*5—01B1T23D—01B3T23D—01B3B28B—01B4B01B *6—01B1J28C—01B2J01C *7—01B1T21D—01B3T21D—01B3B28F—01B4B01F
*8—01B1K28G—01B2K01G *9—01B1T21B—01B3T21B *A—01B1T21B—01B3T21B—01B3B28C—01B4B01C *B—01B1K28H—01B2K01H
*C—01B1T19B—01B3T19B *D—01B1T19B—01B3T19B—01B3B28D—01B4B01D
COMMENTS
TAG DATE E.C.NO. TAG DATE E.C.NO. TAG DATE E.C.NO. TAG DATE E.C.NO.
A 10-21-58 90D1001

Fig. 5(B). Machine-printed systems page, with manual corrections

Packaging Rules

These rules take advantage of the SMS concept to ensure consistent, accurate packaging of circuits. Included is a check of the placement of circuit cards in proper relationship to each other and the frame, as well as conventions of wiring interconnections between circuits. A high degree of cabling standardization has eliminated the need for many special cable designs. The checks insure that



IBM
95-7353-0

	TIME	NAME
TRANSCRIBED		
KEY PUNCHED		
VERIFIED		

CHARGE ______
DATE ______
PAGE ____ OF ____

1	6	14
MACHINE	PAGE #	PART #

21	23	25	26	59
0 1	0 1			
BLOCK	SEQ	#	PAGE NAME	COMMENTS FOR MARGIN

21	23	25	32	39	46	53	60	67	74
0 1	0 2								
BLOCK	SEQ	EC #	EC#	EC#	EC #	EC#	EC #	EC#	EC #

21	23	25	26	33	37	41	45	49	53	57	58	60	62	72
BL	SEQ	*	EC#	LINE 1	LINE 2	LINE 3	LINE 4	LINE 5	LINE 6	#	EXT		CIRCUIT NAME	

21	23	25	26	33
BL	SEQ	*	EC #	COMMENTS
9 0	0 1			
9 0	0 2			
9 0	0 3			
9 0	0 4			
9 0	0 5			

Fig. 6(A). Transcribing form

the designs carefully follow standard rules.

Errata Listing

The primary output of the checking program is the errata listing which is returned with the printed page to the engineer.

Fig. 9 is a compilation of typical errors found by the checking program. It should not be construed to be representative of the numbers of errors made on any given page. The information is organized by logic page, since this is the basic reference document.

Other Output Documents

Another purpose of the checking program is to prepare the many derivative output documents needed to manufacture and service a large computer. The secondary outputs are obtained at this stage in the design mechanization system for two reasons:

1. By its nature the checking program is required to organize the logic data in various formats and sequences.

2. The secondary output documents are useful only if the logic passes the various tests which are imposed by the checking program.

Some of the outputs which are derived from the logic record are as follows:

1. Location charts for the placement of circuit cards.

2. List of points to be connected in wiring panels.

3. Lists of connector jumpers needed to interconnect panels.

4. Lists of cable wires used to interconnect frames and gates.

5. Bills of material for electronic parts.

6. Miscellaneous statistical data.

Operation of Checking Program

The checking program is really a series of individual programs interlocked in a complex manner. The significant components follow.

FORMLO (Format of Logic)

This is a program to reform the master tape records into a format more suitable for the checking process.

CONPAG (Connect Pages)

This program combines lines which are referenced from one page to another. As might be expected, this is a problem area. Three things are done to ease this problem.

1. When matching lines, the names are compressed so that blanks, periods, and commas are eliminated. This prevents mismatches due to the trivial differences in spacing or punctuation.

2. When only one reference is made between two pages, the names are assumed to match. However, a warning message is printed to the effect that this has occurred.

3. When no match can be found, all names are listed together so that the differences are obvious and easily corrected.

LOGVER (Logic Verification)

This program does the circuit checking, such as legality of circuit symbols, accuracy of interconnections, accuracy of representation, etc.

PANLOC (Panel Locations)

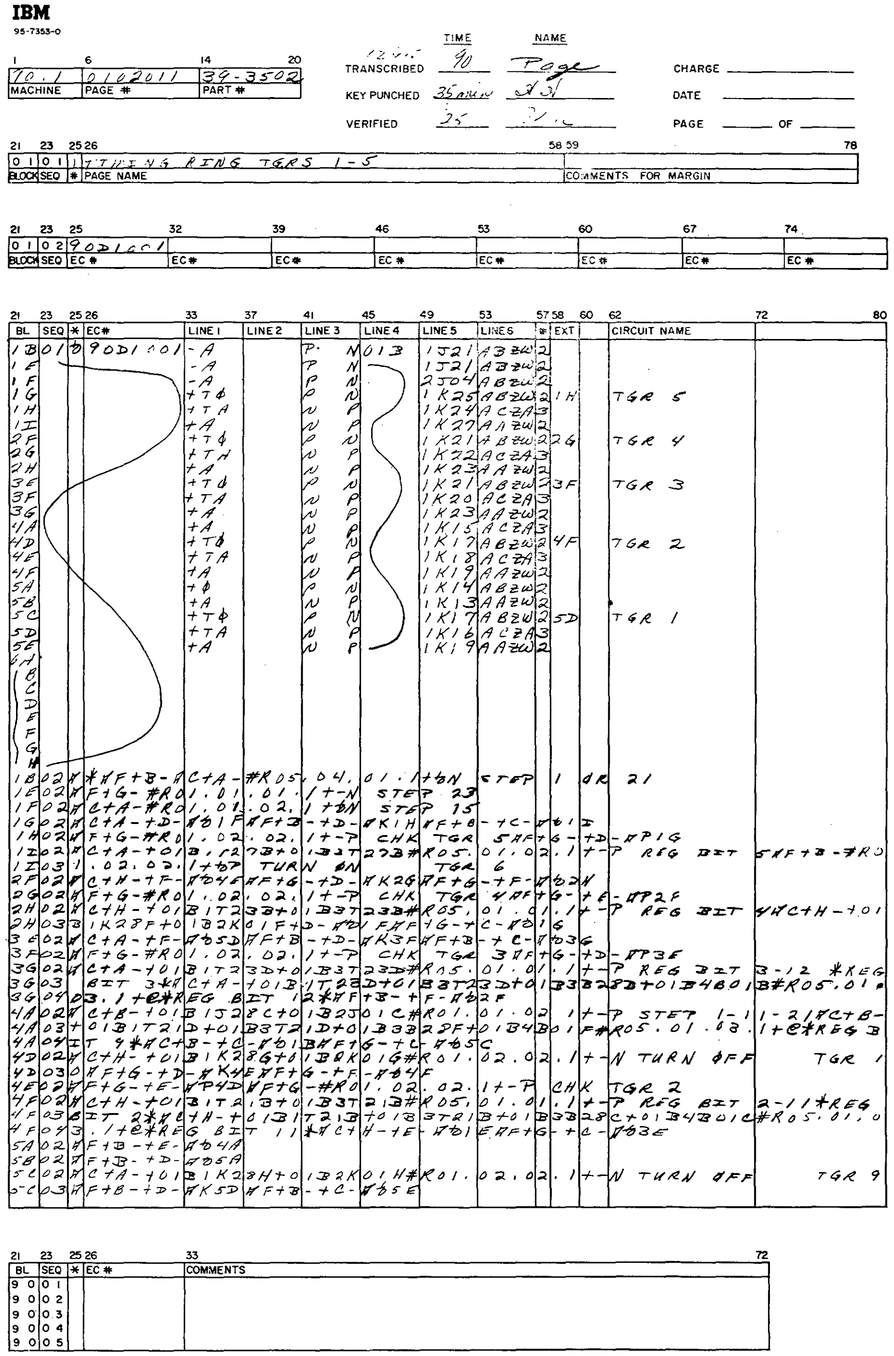
This program furnishes a location chart of circuit cards by panel and bills of material for ordering purposes.

PW Edit

This program provides a list of connector points properly arranged for a subsequent program that will determine the actual wire routing to be used.

Sort Programs

These are for the most part standard generalized sort programs written by the IBM Applied Programming Group.



IBM

95-7353-0

1	6	14 20
70.1	0102011	39-3502
MACHINE	PAGE #	PART #

	TIME	NAME
TRANSCRIBED	90	Page
KEY PUNCHED	35 min	A N
VERIFIED	25	

CHARGE ______

DATE ______

PAGE ____ OF ____

21	23	25	26 58	59 78
0 1	0 1	1	TTIMING RING TGRS 1-5	
BLOCK	SEQ	#	PAGE NAME	COMMENTS FOR MARGIN

21	23	25	32	39	46	53	60	67	74
0 1	0 2	90D1001							
BLOCK	SEQ	EC #	EC #	EC #	EC #	EC #	EC #	EC #	EC #

BL	SEQ	*	EC#	LINE 1	LINE 2	LINE 3	LINE 4	LINE 5	LINE 6	*	EXT		CIRCUIT NAME	
1B	01	Φ	90D1001	-A		P N	013	1J21	ABZW	2				
1E				-A		P N		1J21	ABZW	2				
1F				-A		P N		2J04	ABZW	2				
1G				+TΦ		P N		1K25	ABZW	2	1H		TGR 5	
1H				+TA		N P		1K24	ACZA	3				
1I				+A		N P		1K27	AAZW	2				
2F				+TΦ		P N		1K21	ABZW	2	2G		TGR 4	
2G				+TA		N P		1K22	ACZA	3				
2H				+A		N P		1K23	AAZW	2				
3E				+TΦ		P N		1K21	ABZW	2	3F		TGR 3	
3F				+TA		N P		1K20	ACZA	3				
3G				+A		N P		1K23	AAZW	2				
4A				+A		N P		1K15	ACZA	3				
4D				+TΦ		P N		1K17	ABZW	2	4F		TGR 2	
4E				+TA		N P		1K18	ACZA	3				
4F				+A		N P		1K19	AAZW	2				
5A				+Φ		P N		1K14	ABZW	2				
5B				+A		N P		1K13	AAZW	2				
5C				+TΦ		P N		1K17	ABZW	2	5D		TGR 1	
5D				+TA		N P		1K16	ACZA	3				
5E				+A		N P		1K19	AAZW	2				
6A														
B														
C														
D														
E														
F														
G														
H														

```
1B02# *#F+B-#C+A-#R05.04.01.1+ΦN STEP 1 ΦR 21
1E02# F+G-#R01.01.01.1+-N STEP 23
1F02# C+A-#R01.01.02.1+ΦN STEP 15
1G02# C+A-+D-#Φ1F#F+B-+D-#K1H#F+B-+C-#Φ1I
1H02# F+G-#R01.02.02.1+-P CHK TGR 5#F+G-+D-#P1G
1I02# C+A-+01B1T27B+01B2T27B#R05.01.02.1+-P REG BIT 5#F+B-#R0
1I03 1.02.02.1+ΦP TURN ΦN TGR 6
2F02# C+H-+F-#Φ4F#F+G-+D-#K2G#F+G-+F-#Φ2H
2G02# F+G-#R01.02.02.1+-P CHK TGR 4#F+G-+E-#P2F
2H02# C+H-+01B1T23B+01B3T23B#R05.01.01.1+-P REG BIT 4#C+H-+01
2H03B 1K28F+01B2K01F+D-#Φ1F#F+G-+C-#Φ1G
3E02# C+A-+F-#Φ5D#F+B-+D-#K3F#F+B-+C-#Φ3G
3F02# F+G-#R01.02.02.1+-P CHK TGR 3#F+G-+D-#P3E
3G02# C+A-+01B1T23D+01B3T23D#R05.01.01.1+-P REG BIT 3-12 *REG
3G03 BIT 3*#C+A-+01B1T28D+01B3T23D+01B3B28D+01B4B01B#R05.01.
3G04 03.1+@*REG BIT 12*#F+B-+F-#Φ2F
4A02# C+B-+01B1J28C+01B2J01C#R01.01.02 1+-P STEP 1-11-21#C+B-
4A03+ 01B1T21D+01B3T21D+01B3B28F+01B4B01F#R05.01.03.1+@*REG B
4A04I T 9*#C+B-+C-#Φ1B#F+G-+C-#Φ5C
4D02# C+H-+01B1K28G+01B2K01G#R01.02.02.1+-N TURN ΦFF TGR 1
4D03Φ #F+G-+D-#K4E#F+G-+F-#Φ4F
4E02# F+G-+E-#P4D#F+G-#R01.02.02.1+-P CHK TGR 2
4F02# C+H-+01B1T21B+01B3T21B#R05.01.01.1+-P REG BIT 2-11*REG
4F03B IT 2*#C+H-+01B1T21B+01B3T21B+01B3B28C+01B4B01C#R05.01.0
4F043 .1+@*REG BIT 11*#C+H-+E-#Φ1E#F+G-+C-#Φ3E
5A02# F+B-+E-#Φ4A
5B02# F+B-+D-#Φ5A
5C02# C+A-+01B1K28H+01B2K01H#R01.02.02.1+-N TURN ΦFF TGR 9
5C03# F+B-+D-#K5D#F+B-+C-#Φ5E
```

BL	SEQ	*	EC #	COMMENTS
9 0	0 1			
9 0	0 2			
9 0	0 3			
9 0	0 4			
9 0	0 5			

Fig. 6(B). Sample entries on transcribing form

Operating Times

Experience to date shows that for average quantities of input data, the computer time for processing each logic page averages one minute, or from 3 to 4 seconds per circuit block. This is a complete run through the entire complex of programs.

Wiring Program

As discussed previously, the logic sketch sheets are the only input to the design mechanization system. From these diagrams, the checking program prepares on tape, a list of points to be connected, section by section, with back panel wiring. This tape is used as the input to the wiring program which determines the routing of the back panel wires, observing certain restrictions which have been imposed, and produces the wiring list exhibited in Fig. 10. The program also summarizes the panel wires used by length and type, as shown in Fig. 11, to give an effective bill of material for the panel wiring.

Rules for Back Panel Wiring

The high-speed drift transistor circuits used in the *7000* series computers require extreme care in the point to point wiring. Routing arrangements must be found that satisfy the following criteria:

1. Minimize wire length to avoid electrical delays due to capacitance to ground.
2. Minimize the use of shielding to avoid electrical delays and to decrease cost.
3. Maximize isolation of one circuit from another to minimize noise interference.
4. Minimize the number of connectors to a terminal in order to utilize wire wrapping techniques.

The first pass of the program routes wires so as to minimize total wire length. Yellow single-lead wiring is used throughout. A check is then made on interwire noise, and violations of the specified limits are satisfied automatically by rerouting, by using twisted pairs, or, finally, by resorting to coaxial cable.

Since the number of interconnections for the SMS panels varies from 300 to 1,200, and because of the stringent design restrictions, manual routing taxes human capacity. But the wiring program has been averaging about 20 minutes on the *704* for the typical SMS panel.

Engineering Changes

Since design changes in the logic very often require changes in back panel wiring, the system must accommodate changes as introduced by marked up systems pages. Transcribing of engineering changes is minimized, since only alterations to the previous drawing need to be keypunched. The tape updating program makes the necessary alterations to the tape records, and new diagrams are produced by the print program.

Because a slightly changed set of initial conditions might very well result in an entirely different wiring arrangement, and since it would be prohibitively expensive to rewire a complete panel for each design change, the wiring program must superimpose the required changes on an existing arrangement, without violating any of the rules. Here again the computer reduces to trivia a problem which is excessively complex by manual methods. Changing circuit designs are, generally, readily incorporated into the programs. Consequently, with experience and advances in circuit technology, conformance with the new standards can be checked readily.

Summary

Advantages

The design mechanization system offers the following advantages:

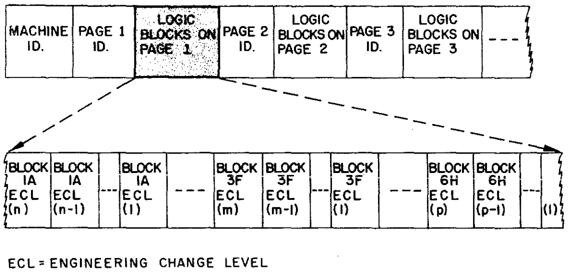


Fig. 7 (left). Master tape format

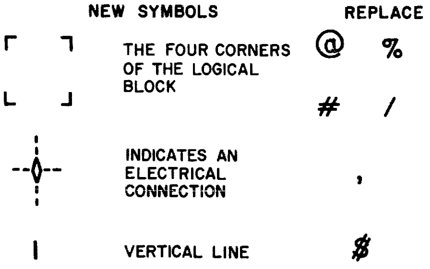

NEW SYMBOLS
REPLACE
THE FOUR CORNERS OF THE LOGICAL BLOCK
@
%
#
/
INDICATES AN ELECTRICAL CONNECTION
,
VERTICAL LINE
$

Fig. 8 (right). Special symbols for the printer type wheel

1. Elimination of error
2. Less repetitive manual detail work
3. Speed of processing
4. More comprehensive information

Elimination of Error

Because of the immense amount of detailed information which must be gathered, preserved, and disseminated during the life of a development project, human errors are inevitable. Many of these errors are not detected in the making, and result ultimately in malfunctions of the machine, scrap and rework, and high product engineering and engineering change activity. The mechanized system not only replaces fallible manual procedures with accurate automatic procedures, but also isolates and rejects many of the errors made in the remaining manual processes.

Less Repetitive Manual Detail Work

Much of the routine work of abstracting details from the basic design is eliminated, leaving professional engineers free for more creative work. This amounts to a more effective utilization of engineering manpower.

Speed of Processing

Waiting periods are eliminated in many stages of the project. Logic diagrams designed by the engineer are checked, "drawn," and returned without delay. Wiring lists are prepared immediately after the design is completed. Bills of material and statistical information are compiled on demand. Changes to circuits and wiring are incorporated without delay. In an era where time is money and overhead is excessive, the elimination of costly delays can reduce over-all project budgets substantially.

More Comprehensive Information

The recording of all logic data on the master tape provides an opportunity to obtain design information heretofore difficult to compile. Most significant to date has been the statistical data so necessary for cost engineering and standardization. A complete count of transistors, circuit cards classified as to type, and various other components can be obtained in a matter of minutes. Moreover, by studying on a statistical basis the utilization of the standard circuits, it becomes possible to eliminate rarely used components and redesign others for more general utility. A reduction in the standard circuit manual is a very desirable objective.

An additional advantage is the availability of logic diagrams at various engineering change levels, thus eliminating a costly storage operation.

Disadvantages

The system as it now exists is not without disadvantages. Some of these are:

1. Programming cost is high, and several man-months of programming effort can be obsoleted by changes in design. This can be a risk when working in a development area where change is the rule rather than the exception.

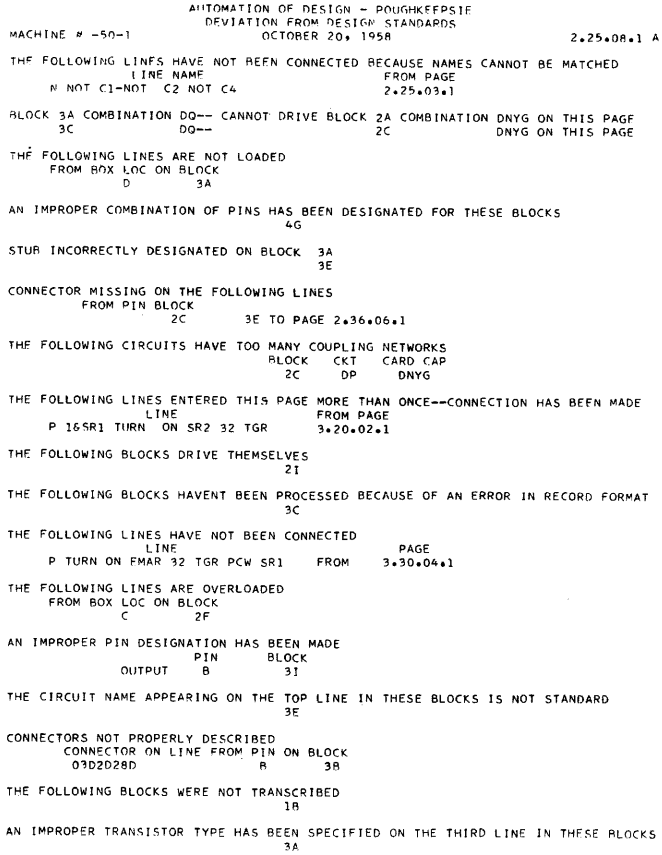
2. The circuit block representation, a somewhat inflexible 5×9 format, is not optimum for representing all the necessary logic. This can sometimes increase the number of logic documents needed.

3. The transcribing and keypunching operation has proven to be extremely time consuming. A large volume of data must flow through this input system with complete accuracy.

Operating Times

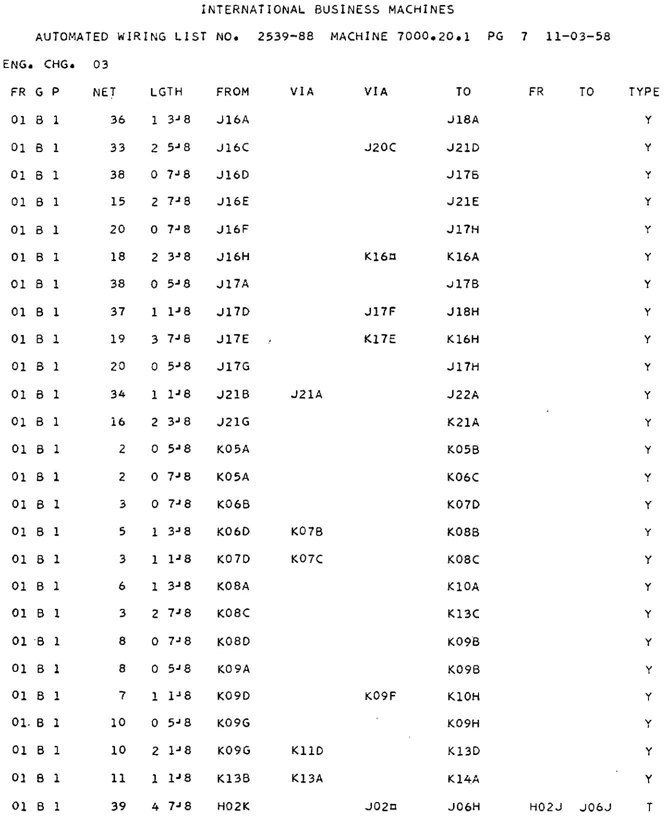
These vary with the volume and complexity of the logic records. Based on experience to date, average figures are as follows:

$2\frac{1}{2}$ to 3 man-hours to transcribe and keypunch a logic page entry. 1 minute of computer time to add to or alter the master tape record. 45 seconds of computer time to prepare the print image on tape. (An additional minute of off-line operation is required for the actual printing.) 70 seconds per logic page checked by the checking program. 20 minutes for computing the wiring of a typical panel.



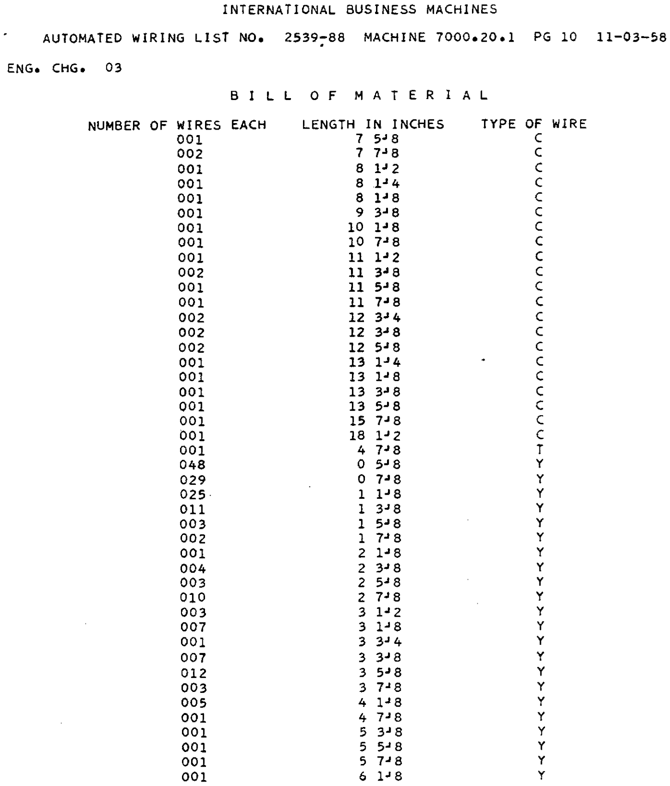
```
                      AUTOMATION OF DESIGN - POUGHKEEPSIE
                        DEVIATION FROM DESIGN STANDARDS
MACHINE # -50-1                OCTOBER 20, 1958                        2.25.08.1 A

THE FOLLOWING LINES HAVE NOT BEEN CONNECTED BECAUSE NAMES CANNOT BE MATCHED
         LINE NAME                                FROM PAGE
   N NOT C1-NOT  C2 NOT C4                        2.25.03.1

BLOCK 3A COMBINATION DQ-- CANNOT DRIVE BLOCK 2A COMBINATION DNYG ON THIS PAGE
      3C             DQ--                    2C                DNYG ON THIS PAGE

THE FOLLOWING LINES ARE NOT LOADED
   FROM BOX LOC ON BLOCK
            D      3A

AN IMPROPER COMBINATION OF PINS HAS BEEN DESIGNATED FOR THESE BLOCKS
                                  4G

STUB INCORRECTLY DESIGNATED ON BLOCK  3A
                                      3E

CONNECTOR MISSING ON THE FOLLOWING LINES
      FROM PIN BLOCK
               2C       3E TO PAGE 2.36.06.1

THE FOLLOWING CIRCUITS HAVE TOO MANY COUPLING NETWORKS
                                BLOCK   CKT   CARD CAP
                                  2C     DP     DNYG

THE FOLLOWING LINES ENTERED THIS PAGE MORE THAN ONCE--CONNECTION HAS BEEN MADE
                LINE                  FROM PAGE
   P 1&SR1 TURN  ON SR2 32 TGR        3.20.02.1

THE FOLLOWING BLOCKS DRIVE THEMSELVES
                                  2I

THE FOLLOWING BLOCKS HAVENT BEEN PROCESSED BECAUSE OF AN ERROR IN RECORD FORMAT
                                  3C

THE FOLLOWING LINES HAVE NOT BEEN CONNECTED
                LINE                          PAGE
   P TURN ON FMAR 32 TGR PCW SR1    FROM    3.30.04.1

THE FOLLOWING LINES ARE OVERLOADED
   FROM BOX LOC ON BLOCK
            C      2F

AN IMPROPER PIN DESIGNATION HAS BEEN MADE
                      PIN     BLOCK
            OUTPUT     B        3I

THE CIRCUIT NAME APPEARING ON THE TOP LINE IN THESE BLOCKS IS NOT STANDARD
                                  3E

CONNECTORS NOT PROPERLY DESCRIBED
     CONNECTOR ON LINE FROM PIN ON BLOCK
      03D2D28D                B      3B

THE FOLLOWING BLOCKS WERE NOT TRANSCRIBED
                                  1B

AN IMPROPER TRANSISTOR TYPE HAS BEEN SPECIFIED ON THE THIRD LINE IN THESE BLOCKS
                                  3A
```

Fig. 9. Machine-printed errata listing



```
                  INTERNATIONAL BUSINESS MACHINES

   AUTOMATED WIRING LIST NO.  2539-88  MACHINE 7000.20.1  PG  7  11-03-58

ENG. CHG.  03

 FR G P    NET   LGTH    FROM    VIA      VIA      TO     FR    TO    TYPE

 01 B 1     36  1 3⌟8    J16A                      J18A               Y

 01 B 1     33  2 5⌟8    J16C             J20C     J21D               Y

 01 B 1     38  0 7⌟8    J16D                      J17B               Y

 01 B 1     15  2 7⌟8    J16E                      J21E               Y

 01 B 1     20  0 7⌟8    J16F                      J17H               Y

 01 B 1     18  2 3⌟8    J16H             K16□     K16A               Y

 01 B 1     38  0 5⌟8    J17A                      J17B               Y

 01 B 1     37  1 1⌟8    J17D             J17F     J18H               Y

 01 B 1     19  3 7⌟8    J17E             K17E     K16H               Y

 01 B 1     20  0 5⌟8    J17G                      J17H               Y

 01 B 1     34  1 1⌟8    J21B    J21A              J22A               Y

 01 B 1     16  2 3⌟8    J21G                      K21A               Y

 01 B 1      2  0 5⌟8    K05A                      K05B               Y

 01 B 1      2  0 7⌟8    K05A                      K06C               Y

 01 B 1      3  0 7⌟8    K06B                      K07D               Y

 01 B 1      5  1 3⌟8    K06D    K07B              K08B               Y

 01 B 1      3  1 1⌟8    K07D    K07C              K08C               Y

 01 B 1      6  1 3⌟8    K08A                      K10A               Y

 01 B 1      3  2 7⌟8    K08C                      K13C               Y

 01 B 1      8  0 7⌟8    K08D                      K09B               Y

 01 B 1      8  0 5⌟8    K09A                      K09B               Y

 01 B 1      7  1 1⌟8    K09D             K09F     K10H               Y

 01 B 1     10  0 5⌟8    K09G                      K09H               Y

 01 B 1     10  2 1⌟8    K09G    K11D              K13D               Y

 01 B 1     11  1 1⌟8    K13B    K13A              K14A               Y

 01 B 1     39  4 7⌟8    H02K             J02□     J06H   H02J  J06J  T
```

Fig. 10. Machine-printed back panel wiring list



```
                        INTERNATIONAL BUSINESS MACHINES

   AUTOMATED WIRING LIST NO.  2539-88  MACHINE 7000.20.1  PG 10  11-03-58

ENG. CHG.  03

                         B I L L   O F   M A T E R I A L

         NUMBER OF WIRES EACH   LENGTH IN INCHES   TYPE OF WIRE
                  001                  7 5⌟8             C
                  002                  7 7⌟8             C
                  001                  8 1⌟2             C
                  001                  8 1⌟4             C
                  001                  8 1⌟8             C
                  001                  9 3⌟8             C
                  001                 10 1⌟8             C
                  001                 10 7⌟8             C
                  001                 11 1⌟2             C
                  002                 11 3⌟8             C
                  001                 11 5⌟8             C
                  001                 11 7⌟8             C
                  002                 12 3⌟4             C
                  002                 12 3⌟8             C
                  002                 12 5⌟8             C
                  001                 13 1⌟4             C
                  001                 13 1⌟8             C
                  001                 13 3⌟8             C
                  001                 13 5⌟8             C
                  001                 15 7⌟8             C
                  001                 18 1⌟2             C
                  001                  4 7⌟8             T
                  048                  0 5⌟8             Y
                  029                  0 7⌟8             Y
                  025                  1 1⌟8             Y
                  011                  1 3⌟8             Y
                  003                  1 5⌟8             Y
                  002                  1 7⌟8             Y
                  001                  2 1⌟8             Y
                  004                  2 3⌟8             Y
                  003                  2 5⌟8             Y
                  010                  2 7⌟8             Y
                  003                  3 1⌟2             Y
                  007                  3 1⌟8             Y
                  001                  3 3⌟4             Y
                  007                  3 3⌟8             Y
                  012                  3 5⌟8             Y
                  003                  3 7⌟8             Y
                  005                  4 1⌟8             Y
                  001                  4 7⌟8             Y
                  001                  5 3⌟8             Y
                  001                  5 5⌟8             Y
                  001                  5 7⌟8             Y
                  001                  6 1⌟8             Y
```

Fig. 11. Machine-printed summary of panel wiring for bill of material use

Extensions of Design Mechanization

For the first time, a complete logic record of a large computer exists in readily available digital form. Problems arising in the post-release period can be programmed readily to make valuable use of this information; for example, to help design special features and to derive proper data for controlling automatic production machinery.

It is obvious that much more progress remains to be made in this direction. By and large IBM has succeeded in mechanizing a record-keeping system. The engineers' effort has been channeled into concern primarily with the basic source document, the logic sketch sheet. Logical extensions of the present program are obvious.

It was not many years ago that "computer-designed computers" seemed like an idle dream. The authors feel that the completion of the record-keeping system described here is a significant step toward this goal.

Appendix I. Checking Program, Circuit Rules

The following lines leaving this page have been defined more than once:

Two or more lines leave a page from different logic blocks but have the same name.

An improper pin designation has been made:

For the specified card/cap combination, the pin given is incorrect.

The circuit name appearing on the top line in these blocks is not standard:

The circuit name given has not been approved. (Also: the name given may be positioned incorrectly within the block.)

An improper transistor type has been specified on the third line in these blocks:

For the specified card/cap combination, the transistor type indicated is in error.

Block xx combination xxxx cannot drive block zz combination zzzz.

Illegal connection between circuits. Existing rules do not permit the indicated circuit to drive the other.

The following lines are overloaded:

Circuit drives more than it should.

The following lines are not loaded:

Coupling network not present.

Nonstandard card/cap combination for driving block no further checking has been done on these blocks:

The card/cap does not appear in the tables used by the checking program.

The following lines to a block with nonstandard card/cap have not been checked:

Since the card/cap of the driven block is not standard, lines connecting cther blocks cannot be checked.

The following pins are not at the proper location on the circuit block:

When a particular pin must appear at a definite location on a logic block, a violation is indicated in this manner.

Driver output line type disagrees with driven input:

Level on output line of driving circuit disagrees with level on input line of driven circuit.

Stubs not allowed on block:

Stub placed on logic block which does not allow a stub.

The following circuits have too many coupling networks:

Two or more coupling networks have been indicated.

Appendix II. Checking Program, Logic Format Rules

These blocks have not been processed because of an error in record format:

These include errors found in the records taken from the master tape by the print-preparation program (e.g. pages which are not numeric, incomplete information, incorrect block designation, etc.)

The following lines have not been connected because names cannot be matched:

More than two lines between pages, some of which may have been connected. The lines indicated are those where it has not been possible to determine whether connection could be made.

The following lines cannot be connected:

Lines that indicate a particular page as a destination, but on the designated page no reference has been made for a line coming to it from the other page.

These lines have no page designation:

Lines that enter or leave a page without specifying the page from which the line is coming or to which it is going.

Names for the following lines are missing:

Lines have not been identified by names where it is necessary to do so.

The following lines leaving this page have been connected, however the line names are unequal:

Occurs when there is only one line reference between two indicated pages even though there are differences in line names (i.e. blank spacing, use of period in abbreviation, additional comments); the program considers it appropriate to connect the indicated lines.

The following lines entered a page more than once:

Lines with the same name and page source have entered a page at two separate places on a page when the line should have entered at one location and branched after entering.

A line from the following blocks leaves the page on the left contrary to drawing standards:

Output lines from logic blocks shown leaving a logic page should be drawn going off the right side of the page.

Stub incorrectly designated on block:

A stub is permitted on the logic block, but it has been shown in the wrong location.

The record character count for the following blocks is incorrect:

The number of characters in a master tape record does not agree with the number of characters received to be checked.

The following block drives itself:

The indicated block is its own input (probably transcription error.)

These blocks have not been transcribed:

One or more lines on a page refer to a block which does not exist. (The block may not have been transcribed, or it may have been deleted but information about a line which drives it was not deleted.)

Appendix III. Checking Program, Packaging Rules

Machine location incomplete for the following blocks:

One or more items of the information identifying frame, gate, panel, row, or column is not present.

The machine location of the following blocks is incorrect:

One or more parts of the indicated machine location have been specified incorrectly.

An improper combination of circuit pins has been designated for these blocks:

One or more of the pins designated for a particular logic block have been indicated incorrectly.

The following blocks have identical machine locations:

Two or more logic blocks have been given the same machine location assignments.

Connector missing on the following lines:

Connectors should be indicated since the line connects blocks in different panels.

Connectors on the following lines have not been checked:

Because of other errors or incomplete information some lines cannot be checked for necessary connectors.

Connectors not properly described:

The information given does not describe any type of connector.

Ground pin used for connector:

Any connector which uses a ground pin is indicated.

The row, column, or pin of the following connectors do not match:

Where it is necessary for the row, column, or pin of a pair of connectors to agree, this has not been indicated.

Edge connectors join the same or adjacent gates:

A connector should be used, but not an edge connector.

Discussion

Frank Segal (Westinghouse Electric Company): Is IBM planning to distribute this through Share?

Mr. Case: This is a series of programs of a high degree of complexity. I doubt if we are going to distribute it through Share, but we are certainly willing to donate it to anybody who feels that they can use it. We have nothing to hide and everything to gain by encouraging the use of computers.

Peter Scola (General Electric Company): What techniques are you using to minimize wire length?

Mr. Case: Wire length is minimized by selecting from a list of possible paths, the paths which will result in a minimum total length. Two points to be connected lie on opposite vertices of a parallelogram, and all paths on the edges or paths within the boundaries with two bend points have equal lengths. This enables a decision to be made selecting the least noisy path without increasing the length above the minimum.

Question (Bell Telephone Laboratories): Briefly, what is your tape update program? Do you have to run off another set of cards to attain a block omitted previously?

Mr. Case: The question refers to program organization. This is a standard file maintenance operation, using the old tape file as input, and adding or deleting any blocks from card input to produce a new output tape.

State-Logic Relations in Autonomous Sequential Networks

W. H. KAUTZ

Synopsis: A sequential network consists of an interconnection of logical elements, such as "and"-gates, "or"-gates, inverters, etc., and storage elements, such as flip-flops and delay lines. These networks process signals, usually binary signals, in the sense that they convert input sequences of 0's and 1's into output sequences functionally related to the input sequences. Typically, these networks occur as parts of digital computers and control systems, and perform operations such as counting, code conversion, program control, addition, comparison, sequence generation, etc. Such networks are presently designed by unsystematic cut-and-try methods. Consequently they are unnecessarily costly, and are often more difficult to test and maintain because of the lack of patterned structure in their realization.

Probably the knowledge most lacking for development of good synthesis techniques for sequential networks is a better understanding of the relationship between the internal logic and the state-sequential behavior of such networks. This paper explores this relationship through the mechanism of a certain mathematical model of the network. In particular, some conditions are derived for the network to be nonsingular; i.e., to have a state diagram which is deterministic even in reversed time, and some important consequences of the nonsingularity condition are demonstrated. Also, the effects on the state diagram of several kinds of constraints imposed on the logic are determined. Several special classes of sequential nets are analyzed: the nonlinear-feedback shift register, the safe-asynchronous net, the fully self-independent net, and the net with cyclically permuted logic. Finally, the realization problem is discussed using various types of binary storage elements, such as set-reset flip-flops trigger flip-flops, delay elements, relays, etc.

IN A RECENT paper Campeau introduced a certain matrix, called in this paper a C-matrix, for the analysis of sequential digital networks.[1] The rows of this matrix constitute a complete expression of the combinational part of the sequential net (the "logic") and the columns indicate the sequence of states which make up the sequential behavior of the network, as usually displayed in a flow table or state transition diagram. Campeau showed how C-matrices may be manipulated for the analysis of digital networks.

Probably the knowledge most lacking for the development of good synthesis techniques is a better understanding of the relationship between the internal logic and the state-sequential behavior of such networks. This paper explores this relationship through the mechanism of the C-matrix. In particular, it derives some conditions for the matrix to be nonsingular; i.e., to correspond to a state diagram which is deterministic even in reversed time, and show some consequences of the nonsingularity condition. Also, the effects on the state graph of several kinds of constraints imposed on the logic are determined. Several special classes of sequential nets are analyzed with the C-matrix: the nonlinear-feedback shift register, the safe asynchronous net, the fully self-independent net, and the net with cyclically permuted logic. Finally, the realization of C-matrices is discussed using various types of binary storage elements, such as set-reset flip-flops, trigger flip-flops, and relays.

Model of a Sequential Net

A sequential network consists of an interconnection of logical elements, such as "and" gates, "or" gates, inverters, etc., and storage elements, such as flip-flops and delay lines. These networks process signals, usually binary signals, in the sense that they convert input sequences of 0's and 1's into output sequences functionally related to the input sequences.

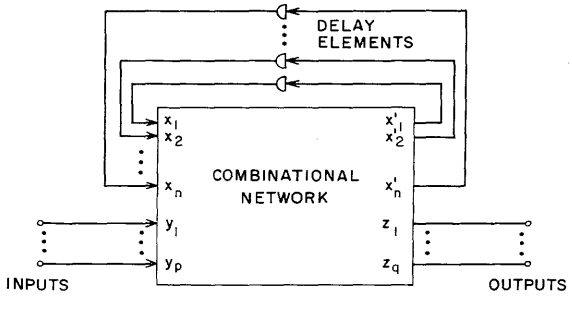


Fig. 1. Model of an arbitrary sequential network

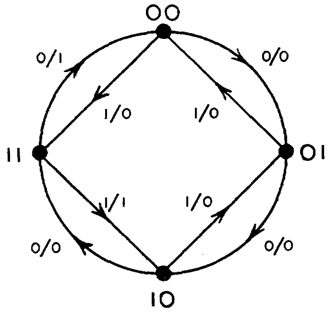


Fig. 2. A typical state graph: a reversible binary 4-state counter

Such a network may be depicted with no loss of generality as shown in Fig. 1. In this model the binary storage elements are given the form of delay elements, and are separated for purposes of analysis from the purely logical elements, which make up the combinational network.[2] The generality of the delay element will be apparent in a later section. This network, being purely combinational, contains no memory of any kind. As an abstract model of a physical network, it may be assumed to respond immediately to changes in its inputs. The network is assumed for the time being to operate synchronously; that is, all changes in the internal variables occur simultaneously, as if under the control of an "advance clock."

For a sequential net with n delay elements, p inputs, and q outputs, the combinational network has $n+p$ input signals and $n+q$ output signals. The terminal behavior may, therefore, be described completely in terms of the $n+q$ switching functions of $n+p$ variables:

$$x_k'(x_1, x_2, \ldots x_n, y_1, \ldots y_p), \quad k=1,2,\ldots n$$

$$z_j(x_1, x_2, \ldots x_n, y_1, \ldots y_p), \quad j=1,2,\ldots q$$

These equations state that the next state and the output of the network are completely determined by the present state and the input.

Techniques for the analysis and synthesis of combinational nets are well known,[3,4] although truly economical synthesis is a goal not yet achieved, particularly for multioutput networks such as that of Fig. 1.

The content of the equations may be expressed in tabular form in a "table of combinations" (see following page).

W. H. KAUTZ is with Stanford Research Institute, Menlo Park, Calif.

The author is much indebted to his colleague, Dr. Bernard Elspas of Stanford Research Institute, who collaborated on many of the proofs and other results, and who participated greatly in the development of the viewpoint toward the solution of digital design problems as presented in the early part of the paper.

y_1	$y_2 \ldots y_p$	x_1	$x_2 \ldots x_n$	x_1' $x_2' \ldots x_n'$	z_1 $z_2 \ldots z_q$
0	0...0	0	0...0		
0	0...0	0	0...1		
.					
.					
.					
0	0...0	1	1...1		
0	0...1	0	0...0		
.					
.					
.					
.					
1	1...1	1	1...1		

The rows of the left section of this table are filled in with all possible $(n+p)$-digit binary numbers, usually in their natural binary order. Thus, there are 2^{n+p} rows. Each row represents the situation of one of the possible inputs in conjunction with one possible internal state. The rows of the right section of the table, when completely filled in, define the sequential network, in that the output and the next state are then specified. Clearly, any manner of completely filling in the right section of the table with 0's and 1's yields the table of a realizable sequential network. The number of such networks is equal to the number of ways in which this part of the table may be filled in: $2^{(n+q)2^{n+p}}$

The C-matrix is now defined as the array of entries in the right-hand section of the table, assuming that the rows of the left-hand section are in their natural binary order. For convenience turn the pattern on its side:

$$C=\left[\begin{array}{cc} & \\ \\ \\ \\ \\ \uparrow & \uparrow \end{array}\right]\begin{array}{l}\leftarrow x_1' \\ \vdots \\ \leftarrow x_n' \\ \leftarrow z_1 \\ \vdots \\ \leftarrow z_q\end{array}$$

output and state following state 00...0, and input 00...0

output and state following state 11...1 and input 11...1

For example, consider a reversible binary counter which counts forward when the single input is "0" and backwards when the input is "1." The output shall be "1" only when the internal state is 11 ... 1. The C-matrix for a 4-state counter of this type is

$$C=\begin{bmatrix} 1010 & 1010 \\ 0110 & 1001 \\ \underbrace{0001}_{y_1=0} & \underbrace{0001}_{y_1=1} \end{bmatrix}\begin{array}{l}\leftarrow x_1' \\ \leftarrow x_2' \\ \leftarrow z_1 \\ \end{array}$$

Thus, when the input y_1 is 0, the counter advances with state 01 following state 00 (column 0), state 10 following state 01 (column 1), state 11 following state 10 (column 2), and state 00 following state 11 (column 3). When $y_1=1$, state 11 follows state 00 (column 4), state 00 follows state 01 (column 5), state 01 follows 10 (column 6), and state 10 follows state 11 (column 7). $z_1=1$ only when the present state is 11.

The expression of a switching function of n variables as a 2^n-digit binary number, this number being a column of the table of combinations (a row of the C-matrix), is sometimes called the designation number or characteristic number of the function. Thus, the characteristic number of x_1' in the matrix is

$$x_1'=[1010 \quad 1010]$$

The 1's and 0's in this representation indicate which items in the "disjunctive normal form" or canonical expansion of the function x_1' are present and absent, respectively. Here

$$x_1'=1\cdot\bar{y}_1\bar{x}_2\bar{x}_1+0\cdot\bar{y}_1\bar{x}_2x_1+1\cdot\bar{y}_1x_2\bar{x}_1+0\cdot\bar{y}_1x_2x_1$$
$$+1\cdot y_1\bar{x}_2\bar{x}_1+0\cdot y_1\bar{y}_2x_1+1\cdot y_1x_2\bar{x}_1+0\cdot y_1x_2x_1$$
$$x_1'=\bar{x}_1$$

The Coding Problem

The major objective of sequential synthesis studies is the development of methods of deriving the set of combinational switching functions x_k' $(k=1, 2, \ldots n)$ and z_i $(i=1, 2, \ldots q)$ from a terminal description of the sequential network as a whole. This problem mainly involves obtaining answers to the following two important questions:

1. Precisely what constitues an adequate terminal description of a sequential network?

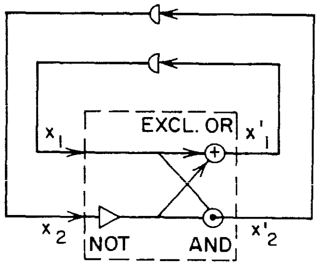
2. What is the essence of the desired relationship between this terminal description and the equations of the combinational network?

From the synthesis viewpoint, question 1 amounts to asking for a mathematical model and language in which to express the specifications of the network to be synthesized. Question 2 gets at the heart of the synthesis process itself, and it is with this question that this paper is concerned.

Virtually all of the past effort on the sequential network problem has been directed towards the network characterization problem 1,[5,6] and the linear transducer.[7]

Two simplifications are in order. First, since question 1 has not yet been fully answered, the "state graph," which is the most suitable terminal description of a sequential net presently available will be used. The state graph, or an equivalent tabular presentation, represents each of the 2^n possible delay-element output signals, or states, $(x_1, x_2, \ldots x_n)$ as a node in a graph, with one interconnecting directed branch for each allowed state-to-state transition. Fig. 2 shows a simple but typical state graph corresponding to the reversible counter previously described. Each transition is conditional on the appropriate input (the number preceding the slant) and gives rise to an output (the number following the slant). For example, if the net is in state 01 with input 0, the next state will be 10, and output 0 will occur.

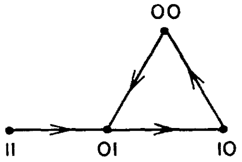
The second simplification is that this paper will consider only the case when the network is autonomous, that is, the case when there are no inputs or outputs.



$$x_1' = x_1 x_2 + \bar{x}_1 \bar{x}_2$$
$$x_2' = x_1 \bar{x}_2$$

Fig. 3 (left). An example of a simple autonomous sequential network

Fig. 4 (right). The state graph and C-matrix of the simple network of Fig. 3



STATE GRAPH

$$C = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

x_2	x_1	x'_2	x'_1
0	0	0	1
0	1	1	0
1	0	0	0
1	1	0	1

TABLE OF COMBINATIONS

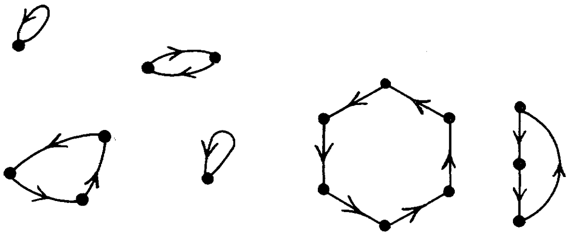


Fig. 5. A typical state graph of an autonomous, nonsingular sequential network, with cycle set ($1_2, 2_1, 3_2, 6_1$)

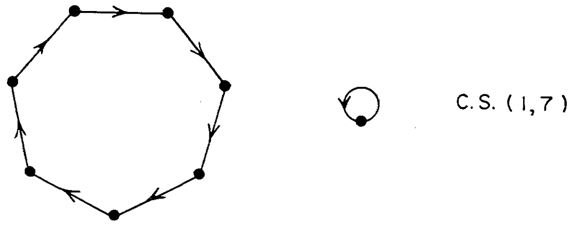


Fig. 6. Typical state graph to be coded for realization in a nonsingular network

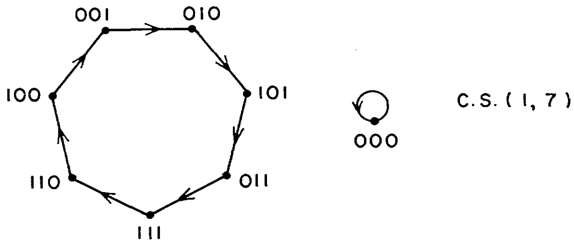


Fig. 7. One possible coding of the state graph of Fig. 6

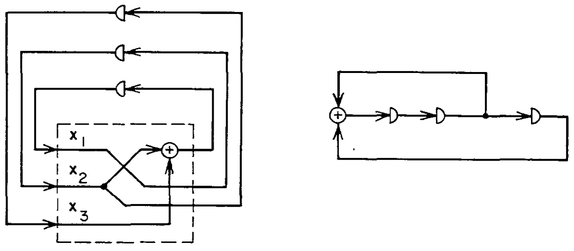


Fig. 8. Network realization of the state graph of Fig. 6, with the coding of Fig. 7

This simpler case is considered principally on the basis of the conjecture that the behavior of the transducer depends strongly on the "natural" behavior of the corresponding undriven network. This conjecture is known to be true at least for the important special case when the combinational network is linear; i.e., when it consists of exclusive "or" gates only. There are good indications that it is true in general.

To eliminate the outputs as well as the inputs is to focus attention on the next-state functions x_k' as the appropriate indications of the internal behavior of the net. Except from the standpoint of economy in synthesis, the output functions z_j may be viewed merely as appendages to the internal state-controlling logic.

With these simplifications, the digits both preceding and following the slant on the branches of the state graph of an autonomous net may be eliminated. Each node (state) then has only one exit branch (transition), which need carry no label.

The analysis of an autonomous sequential network starts with the network arranged in the model of Fig. 1. The switching functions may then be formed by inspection, and these expressed as the rows of the C-matrix (columns of the table of combinations). The state graph can then be derived from the columns of the C-matrix (rows of the table of combinations). These steps are shown in Figs. 3 and 4 for a simple example.

It is apparent that the C-matrix contains implicitly a relationship between the terminal or state-behavior and the internal logic of a sequential network. The state-behavior is contained in the columnar view of the matrix, while its rows express the logic. The remainder of this paper investigates more fully this relationship as expressed in the C-matrix, for the case of autonomous networks.

The synthesis of an autonomous sequential network may be taken to start with a state graph which has no binary labels on the nodes. The first step of the synthesis is the assignment of a unique n-digit binary number to each of the nodes, a process referred to as the coding of the state graph. The C-matrix may then be formed, column by column, by noting the successor states to states 00 . . . 00, 00 . . . 01, 00 . . . 10, . . . , 11 . . . 11. The derivation of the next-state functions x_k' follows from the rows, and from these functions equivalent expressions may be derived appropriate to the binary storage elements which are being employed.

The network and the C-matrix of an autonomous state graph which remains autonomous if the arrows on the branches are reversed are said to be nonsingular. (Huffman terms such a network lossless.) Each state then has a unique preceding state as well as a unique succeeding state. This definition is clearly equivalent to the property of possessing a deterministic time-inverse. Nonsingular networks are worthy of special consideration for two reasons. First, the steady-state behavior of autonomous networks clearly depends only upon the cycles present in its state graph, and not on the "tails" of branches which lead into the cycles. Second, as will be shown in the next section, the most general transformation which carries one coded state graph into another is most naturally represented as an arbitrary nonsingular C-matrix.

It follows directly that the state graph of an autonomous nonsingular net consists entirely of closed loops or cycles. A complete description of the structure of such a state graph can then be given by its cycle set, which is a listing of the lengths ℓ_i of its cycles (the number of states in the cycles), each with its multiplicity r_i. To describe the cycle set, the notation, $c.s. = ((\ell_1)_{r_1}, (\ell_2)_{r_2}, \ldots (\ell_\lambda)_{r_\lambda})$ is used. E.g., for the state graph S of Fig. 5, the expression $c.s.(S) = (1_2, 2_1, 3_2, 6_1)$ indicates that S has two 1-cycles, one 2-cycle, two 3-cycles, and one 6-cycle.

Since the total number of states is 2^n, the numbers ℓ_i and r_i must obviously satisfy the condition:

$$\sum_{i=1}^{\lambda} \ell_i r_i = 2^n$$

This sum condition is clearly necessary for a cycle set to be realizable. It should also be clear that any coding whatsoever of the state graph will lead to a realizable C-matrix so long as: 1. no two nodes are given the same state number, and 2. the state graph has 2^n nodes. Thus, the sum condition is sufficient as well as necessary for a cycle set to be realizable.

For example, realization of a cycle set (1, 7) requires that the state graph of

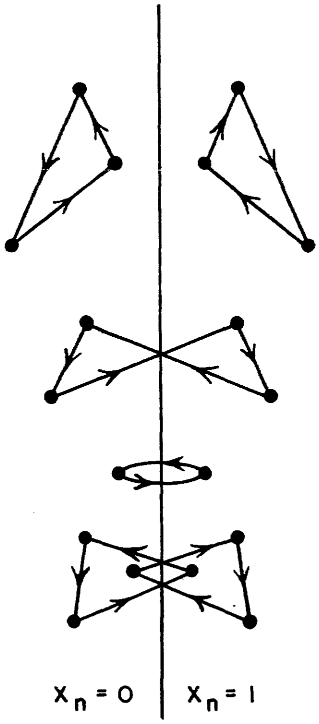


Fig. 9. Typical state graph when all x_k' except x_n' are independent of x_n

Fig. 6 be coded. The eight numbers, 000, 001, 010, 011, 100, 101, 110, and 111 could be allocated to the eight states in a very large number of ways. The particular assignment of Fig. 7 corresponds to the C-matrix:

$$C=\begin{bmatrix} 0011 & 1100 \\ 0101 & 0101 \\ 0011 & 0011 \end{bmatrix}$$

I.e., state 000 is followed by state 000, state 001 is followed by state 010, state 010 is followed by state 101, etc. The rows of this C matrix now give the characteristic numbers of the binary state variables:

$$x_1' = [0011\ 1100]$$

$$x_2' = [0101\ 0101]$$

$$x_3' = [0011\ 0011]$$

from which the circuit equations may be written down immediately:

$$x_1' = x_2 \oplus x_3,\ x_2' = x_1,\ x_3' = x_2$$

The network realizing the given cycle set is that displayed in Fig. 8, drawn in two ways. Almost any other coding would have led to much more intricate circuit logic.

The difficult aspect of synthesis is that of selecting a coding which leads to an economical realization of the multioutput network, in terms of whichever types of binary storage elements and logical elements are prescribed. This appears to be a very difficult problem, and it represents the main reason for this study of the relations between the logic and state behavior of sequential networks.

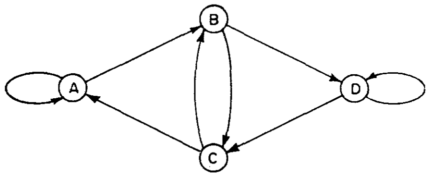


Fig. 10. Transition restrictions pertinent to condition 16

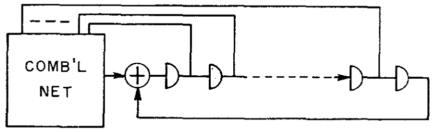


Fig. 11. The nonlinear feedback shift register

Manipulation of C-Matrices

If the state numbers are defined as column matrices,

$$x=\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \qquad x'=\begin{bmatrix} x_1' \\ x_2' \\ \cdot \\ \cdot \\ \cdot \\ x_n' \end{bmatrix}$$

then the formal equation $x' = Cx$ expresses the role of the C-matrix as a transformation from a present state to a next state. The multiplication indicated is clearly not matrix multiplication in the usual sense, but is an expression of the fact that the independent and dependent variables in the row-functions of the C-matrix are the elements of the column matrices x and x', respectively. The identity C-matrix I (Campeau's A-matrix[1]) is therefore the C-matrix for which $x' = Ix = x$.
That is, it is the matrix which expresses the set of equations $x_k' = x_k$:

$$I_1 = [01],\quad I_2=\begin{bmatrix} 0101 \\ 0011 \end{bmatrix},\quad I_3=\begin{bmatrix} 0101 & 0101 \\ 0011 & 0011 \\ 0000 & 1111 \end{bmatrix},$$

etc.

The corresponding state graph then consists of 2^n 1-cycles, since each state is followed by the same state.

Similarly, multiplication of two C-matrices, A and B, also has both a state sequential and a logical interpretation, corresponding to the successive application of the two transformations, viewed individually as state-to-state transformations and as a set of transforming equations. Also, the inverse C^{-1} of a nonsingular C-matrix is that matrix which satisfies the equation $C^{-1}C = CC^{-1} = I$ so that if $x' = Cx$, then $x = C^{-1}x'$. From a state-sequential viewpoint, the inverse C-matrix can be written down immediately from the state graph by imagining all the arrows on the branches to be reversed in direction. From a logic-equation viewpoint, C^{-1} is an expression of the solution of the equations $x_k'(x_1, \ldots x_n)$ for the unprimed variables.

Powers C^m of a C-matrix correspond to simple iteration. C^m transforms each state x into the state m time intervals in the future. The cycle structure of the state graph can be studied through the periodicity properties of the powers of a C-matrix. E.g., it is not hard to show that if a power M can be found such that $C^M = I$ then all cycle lengths in the state graph for C divide M.

An interesting example is afforded by the conventional binary counter, which carries each binary state number into the next number in the natural binary counting sequence. The C-matrix of such a counter is identical to the identity C-matrix I, except that all of the columns are shifted one place to the left; e.g., for $n=3$

$$C_b=\begin{bmatrix} 1010 & 1010 \\ 0110 & 0110 \\ 0001 & 1110 \end{bmatrix}$$

(The first column is shifted around to the last position.) The logic can be read directly from the rows. For a binary counter which counts m at a time instead of *1* at a time, it is only necessary to raise C_b to the mth power. This amounts to shifting the columns of the identity matrix m instead of *1* place to the left. As long as m and 2^n are relatively prime numbers (i.e., as long as m is odd), the state graph will consist of a single 2^n-cycle.

In view of the importance of the coding problem, it is worthwhile to inquire into the nature of the transformation which carries one coding of a state graph into another. Clearly, any such recoding amounts merely to a permutation of the 2^n possible state numbers. The most general such transformation is therefore of the nature of a nonsingular C-matrix itself. If the original state graph is labelled with states x and the recoded state graph with states y, and call the transformation matrix Q, then $y = Qx$.

Similarly, $y' = Qx'$, and from $x' = Cx$, the result is

$$Q^{-1}y' = CQ^{-1}y$$

or, since Q is nonsingular, $y' = QCQ^{-1}y$. The new C-matrix for the y-states, C^*, say, is related to the old C-matrix by the expression $C^*=QCQ^{-1}$, a familiar identity in transformation theory. Note that C need not be nonsingular itself in order for this to hold.

Realization of Storage Elements

Types of binary storage elements other than the delay elements may be related to the delay element by exhibiting the extent of self-dependence in the individual switching functions (individual rows of the C-matrix)

$$x_k'=f_k(x_1,\ x_2,\ \ldots x_n)$$

Expanding x_k' about x_k, results in:

$$x_k'=\bar{x}_k g_k(x_1,\ \ldots x_n)+x_k h_k(x_1,\ \ldots x_n)$$

where g_k and h_k are independent of x_k. In fact,

$$g_k=f_k\Big|_{x_k=0}$$

$$h_k=f_k\Big|_{x_k=1}$$

A further development gives

$$x_k'=\bar{x}_k t_k+x_k q_k+s_k$$

in which the functions t_k, q_k, s_k, and another function r_k, are also independent of x_k, and in addition

$$q_k+r_k+s_k+t_k=1$$

$$q_kr_k=q_ks_k=q_kt_k=r_ks_k=r_kt_k=s_kt_k=0$$

Thus, the *q-r-s-t-* functions are disjunctive; that is, one and only one of them has the value "1" for each combination of values of the $n-1$ variables $x_1 \ldots x_{k-1}$, $x_{k+1}, \ldots x_n$. The *q-r-s-t-* functions, and *g*- and *h*-functions are simply related by the expressions

$$q_k=\bar{g}_k h_k,\quad r_k=\bar{g}_k\bar{h}_k$$

$$s_k=g_k h_k,\quad t_k=g_k\bar{h}_k$$

and

$$g_k=t_k+s_k,\quad h_k=q_k+s_k$$

The *q-r-s-t-* functions may be interpreted in terms of the signals applied to a conventional flip-flop as follows:

$r_k=1$, so $x_k'=0$: reset (i.e., switch the flip-flop to the "0" state)
$t_k=1$, so $x_k'=\bar{x}_k$: complement (trigger)
$q_k=1$, so $x_k'=x_k$: hold (i.e., leave the flip-flop as it is) (quiescent condition)
$s_k=1$, so $x_k'=1$: set (i.e., switch the flip-flop to the "1" state)

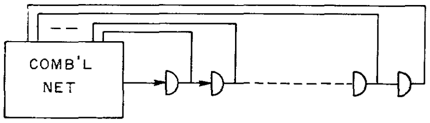


Fig. 12. The nonsingular, nonlinear feedback shift register

From a given function x_k' expanded in terms of the *q-r-s-t-* functions, therefore, one may write down immediately the switching functions for the set, reset, and complementing signals to be applied to a flip-flop realizing the given function x_k':

Case I

set $=S_k=s_k$

reset $=R_k=r_k$

complement $=T_k=t_k$

Case I pertains when the flip-flop has available an input terminal for each of the set, reset, and complement signals, and simultaneous signals on any two of these terminals are not allowed. The following special cases are more likely to be encountered in practice.

Case II

The flip-flop has only two terminals, set and reset. Complementing is accomplished by applying a "1" to both set and reset terminals at the same time. In this case:

$$S_k=s_k+t_k=h_k$$

$$R_k=r_k+t_k=\bar{g}_k$$

Case III

No complementing terminal is available and simultaneous signals on the set and reset terminals are not allowed. Set and reset signals to flip-flop no. *k* are then generally dependent on the outputs of flip-flop no. *k* itself:

$$S_k=s_k+t_k\bar{x}_k$$

$$R_k=r_k+t_kx_k$$

Actually, in this case, application of the "hold" signal to the set or reset terminal can do no harm if the flip-flop is already "on" or "off", respectively. So the preceding equations may be replaced by the following:

$$S_k=s_k+t_k\bar{x}_k+[q_kx_k]$$

$$R_k=r_k+t_kx_k+[q_k\bar{x}_k]$$

where the bracket signifies a "don't care" condition, indicating that any part or all of the bracketed function may be combined with the rest of the expression to effect a possible simplification. (Since the *q-r-s-t*-functions constitute a disjunctive set, however, combinations will be possibly only with the more unusual types of logic elements in the network.)

Case IV

No set and reset terminals are available. Now the complementing signal involves a self-dependence within flip-flop no. *k:*

$$T_k=t_k+s_k\bar{x}_k+r_kx_k$$

It may be seen from these cases that if certain of the functions q_k, r_k, s_k, and t_k can be arranged to be zero in the formation of the switching function x_k', then a simplification of the circuit may result, depending on which case is of interest. E.g., no self-dependence is involved in Case III if $t_k\equiv 0$, and in Case IV if $s_k\equiv r_k\equiv 0$. For Case IV, this leaves

$$T_k=t_k=\bar{q}_k$$

or

$$x_k'=x_k\oplus t_k$$

A safe-asynchronous network is a network in which only one of the *n* variables x_k changes each clock time. In such a circuit, proper behavior is assured if the clock is removed; i.e., if the clock signal is held at "1" continually, since no ambiguity in the internal state can result if two state variables change to new values at different rates, a process which may otherwise produce an unwanted intermediate state. Such race conditions are generally undesirable, although they are frequently allowed in practice, since a significant simplification in the circuit can often result if they are permitted. In such a case, the race is usually made "noncritical" by arranging the circuit so that all possible intermediate states lead to the same final state during a multivariable transition. Unless the saving realized by allowing noncritical races is substantial, however, safe-asynchronous circuits are to be preferred. They are easier to maintain and less likely to lead to inconspicuously incorrect behavior when faults occur.

It is easy to show that, except for a single degeneracy, all functions $t_k(k=1,\ 2,\ \ldots n)$ must be identically zero in a safe-asynchronous circuit. If some t_k were not zero, then for at least one set of values of $x_1 \ldots x_{k-1}, x_{k+1}, \ldots x_n$, the next value x_k' of x_k would be given by $x_k'=\bar{x}_k$. That is, x_k would change next. Since the network is safe-asynchronous, however, none of the state variables other than x_k can change at this time, so that it is still true that $t_k=1$, and $x_k'=\bar{x}_k$. Once the condition $t_k=1$ is reached, therefore, x_k will take on the values 0 and 1 alternately, in the fashion of a doorbell, and there is no way of terminating the oscillation. So except for the possibility of the "doorbell" effect (2-cycles), all functions $t_k(k=1,2,$

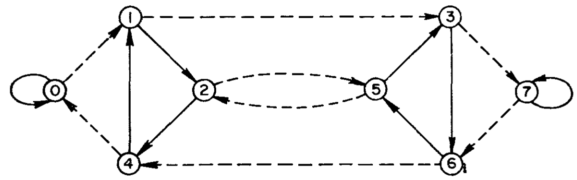


Fig. 13. Generalized state graph for the nonlinear feedback shift register with three stages

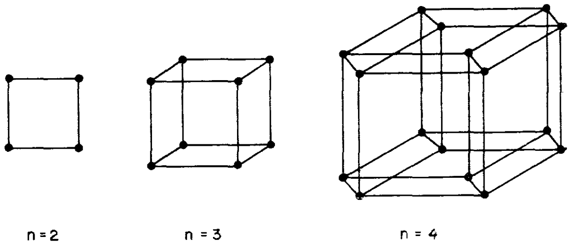


Fig. 14. Generalized state graphs for the safe-asynchronous network

$\ldots$ n) must be identically zero in a safe-asynchronous circuit.

More will be said about safe-asynchronous circuits in a later section.

Neutral Switching Functions

If the number of 1's in the characteristic number of a function f is equal to the number of 0's, the function is said to be neutral. It will be shown now that every switching function in a nonsingular C-matrix is neutral.

From the purely cyclic character of the state graph, it follows that each of the 2^n states has a unique subsequent state. Thus, each of the 2^n binary state numbers must appear once and only once as columns in the C-matrix. The columns of a nonsingular C-matrix are, therefore, merely a permutation of the columns of the identity C-matrix of the same order, for which the switching functions are $x_k'=x_k(k=1,2, \ldots n)$. Since the functions x_k are all neutral, the functions resulting from any column permutation of the identity C-matrix are also neutral.

The only neutral functions of two variables, for example, x and y, are of symmetry type

x

$x\oplus y$

and for three variables, x, y, z:

x

$x\oplus y$

$x\oplus y\oplus z$

$x\oplus yz$

$xy\oplus yz\oplus zx$

$xy\oplus \bar{x}z$

The operation $+$ may be substituted for $\oplus$ in the last two expressions. There are 74 neutral symmetry types of 4 variables. Forty-two of these, along with all neutral symmetry types of 2 and 3 variables, are the same as their complements; the complements of the other 32 are not the same, but form 16 mutually complementary pairs.[8]

If g and h are any neutral functions independent of a variable x, then it may easily be shown that the function $\bar{x}g+xh$ is also neutral. Further, if f_o is any function independent of x, then the function $x\oplus f_o$ is neutral.

Nonsingularity of C-Matrices

Although each of the n switching functions x_k' in a nonsingular C-matrix is neutral, most sets of n neutral functions represent singular C-matrices. It will be shown that a C-matrix is nonsingular if and only if all linear sums

$$\sum x_k'$$

are neutral. (Note: On this page the symbol Σ means the sum of modulo-2.)

Proof: The characteristic number of each of the 2^n-1 sums $\Sigma x_k'$ can be formed by adding (modulo-2) the characteristic numbers of each possible subset of rows of the C-matrix. If the C-matrix is nonsingular, its columns are merely a permutation of the columns of the identity C-matrix. Therefore, each sum $\Sigma x_k'$ will be neutral only according as the sum of the corresponding rows of the identity C-matrix are neutral. Since the individual rows of the identity C-matrix represent the functions x_1, x_2 $\ldots\ldots$ x_n, any sum of these rows has the form Σx_i, and by the last statement of the previous section is certainly neutral. Therefore, all sums $\Sigma x_k'$ are neutral for a nonsingular C-matrix.

To establish the converse, the measure $\mu(f)$ of a logic function f is defined[6] to be the number of 1's in the characteristic number of f. Thus the measure of a neutral function is 2^{n-1}; in fact it is true that:

$$\mu(\bar{f})=\mu(f)=2^{n-1}$$

By adding their characteristic numbers, it is easily seen that for any two functions f_1 and f_2 of n variables

$$\mu(f_1\oplus f_2)=\mu(f_1)+\mu(f_2)-2\mu(f_1f_2) \text{ or}$$

$$\mu(f_1f_2)=\frac{1}{2}\,[\mu(f_1)+\mu(f_2)-\mu(f_1\oplus f_2)]$$

Similarly,

$$\mu(f_1f_2f_3)=\frac{1}{4}\,[\mu(f_1)+\mu(f_2)+\mu(f_3)-\mu(f_1\oplus f_2)-\mu(f_1\oplus f_3)-\mu(f_2\oplus f_3)+\mu(f_1\oplus f_2\oplus f_3)]$$

etc., for more than three functions. If all of these functions and all of their $\oplus$-sums are neutral, each $\mu(\ )$ expression equals 2^{n-1}; and it follows, therefore, that

$$\mu(f_1, f_2\ldots f_r)=2^{n-r}$$

In particular, $\mu(f_1\ldots f_2\ldots f_n)=1$, which is to say that the C-matrix formed from these n functions has only a single column containing all 1's.

If one or more of the functions f_i is complemented, this last equation still holds, since $\mu(f)=\mu(\bar{f})$ for a neutral function. So for all 2^n products,

$$\mu(\dot{f}_1\dot{f}_2\ldots\dot{f}_n)=1,$$

where $\dot{f}_i$ denotes $\bar{f}_i$ or f_i. Thus, the C-matrix has one and only one column in each of the 2^n possible patterns of 0's and 1's. The matrix is therefore a column permutation of the identity C-matrix, and is nonsingular.

It follows directly from this theorem that if a C-matrix is nonsingular, any set of permutations or complementations of the row-functions will preserve the nonsingularity property, although in general its cycle set will be modified drastically. Further, any linearly independent set of n functions selected from the 2^n-1 functions $\Sigma x_k'$ may be assembled to form a new C-matrix, which is then guaranteed to be nonsingular. (With the function "0," these 2^n-1 neutral functions form an abstract group under the $\oplus$ operation. The n functions selected for the C-matrix then constitute a set of generators for the group.)

The set of C-matrices generated in this way from the identity C-matrix constitutes the set of all linear C-matrices; that is, each function x_k' is a linear (modulo-2) sum of the unprimed variables. The linear case has been extensively studied by Huffman[7] and by Elspas.[9]

Huffman has shown in an unpublished memorandum an alternate condition for nonsingularity, which is equivalent to the following:

A C-matrix is nonsingular if and only if all of the 2^n-1 products Πf_i, except the total product, $\prod_{i=1}^{n} f_i$, have even measure, and the total product has odd measure.

The necessity is obvious from the proof of the previous theorem. Sufficiency is established by showing that all of the total products $\dot{f}_1\dot{f}_2\ldots\dot{f}_n$ have a measure which is odd, and must therefore be unity.

State-Logic Relations

In this section some of the simpler relations between state-graph structure and the switching functions x_k' will be derived for the nonsingular C-matrix by showing how certain restrictions on this set of functions reveal themselves as restrictions on the cycle set. When a single one of the functions is to be restricted, this function is taken without loss of generality to be x_n', whose characteristic number is the last row of the C-matrix.

1. If $x_n'=\bar{x}_n$, then all cycles are even; i.e., of even length.

Proof: $x_n'=(1\,1\ldots1\,1\,0\,0\ldots0\,0)$ so that each state whose number is $<2^{n-1}$ is followed by a state $\geq 2^{n-1}$, and each state $\geq 2^{n-1}$ is followed by a state $<2^{n-1}$. Thus, on each cycle the state numbers alternate between these two nonintersecting sets ($<2^{n-1}$ and $\geq 2^{n-1}$), and no cycle length can be odd.

2. If $x_n'=x_n$, then the cycle-set can be halved; that is, the list of cycle lengths may be separated into two distinct subsets, each with half of the total number of states. [E.g., $(1_2, 2, 5, 7)=(1, 2, 5)+(1, 7)$.]

Proof: Here $x_n'=(0\,0\ldots0\,0\,1\,1\ldots1\,1)$, so that each state whose number is $<2^{n-1}$ is followed by a state $<2^{n-1}$, and each state $\geq 2^{n-1}$ is followed by a state $\geq 2^{n-1}$. The stated condition follows directly.

A further consequence of this condition is that the longest cycle length must be $\leq 2^{n-1}$.

3. If all state variables x_k', except x_n', are independent of x_n, then all odd cycles occur with even multiplicity.

Proof: A function x_k' is independent of x_n if and only if the left and right halves of the characteristic number of x_k' are identical. Under the present hypothesis, therefore, the left and right halves of the entire C-matrix, exclusive of the bottom row, are identical. The left and right halves of the bottom row must be complementary, since every state (column) occurs just once. Thus, every state $s_k<2^{n-1}$ is matched by an "image" state $s_i+2^{n-1}\geq 2^{n-1}$. Their successors are also images, since the next-state numbers also differ by 2^{n-1}. Thus the state graph must have complete mirror symmetry, and any odd cycle must be matched by an identical image. Statement 3 directly follows. (See Fig. 9.)

A further consequence of the stated condition follows from the form of the bottom row of the C-matrix: x_n' is fully dependent on x_n; i.e.,

$$x_n'=x_n\oplus g_n$$

where g_n is a function which is independent of x_n. Its characteristic number is given by the left half of the bottom row of the C-matrix.

Each of the three necessary conditions may be interpreted to be sufficient, under some suitable partial encoding of the state graph. Relation 1 for example, may be inverted to read: If all cycles are of even length, then there exists at least one coding of the state graph which makes $x_n'=\bar{x}_n$. The relations 4 through 16 which follow may be similarly inverted. With the exception of relation 16, the proofs of sufficiency of all of these conditions represent merely a reverse of the necessity argument.

Combinations of these conditions lead to further relations. The second function which is restricted may be taken without loss of generality to be x_{n-1}'. The proofs of these relations are direct extensions and combinations of the three previous proofs.

Iterating relations 1, 2, and 3, results in, respectively:

4. If $x_n'=\bar{x}_n$ and $x_{n-1}'=\bar{x}_{n-1}$, then all cycles are even, and the cycle set can be halved.

5. If $x_n'=x_n$ and $x_{n-1}'=x_{n-1}$, then the cycle set can be quartered; that is, it is separable into the sum of four distinct subsets, each with one-fourth the total number of states.

6. If all x_k' except x_n' are independent of x_n, and all x_k' except x_{n-1} are independent of x_{n-1}, then all odd cycles occur with a multiplicity divisible by four, and all even cycles occur with even multiplicity. It also follows that $x_n'=x_n\oplus g_n$ and $x_{n-1}'=x_{n-1}\oplus g_{n-1}$, where g_n and g_{n-1} are independent cf x_n and x_{n-1}, respectively.

Similarly, combinations of conditions 1, 2, and 3 yield other interesting cases

7. If $x_n'=\bar{x}_n$ and $x_{n-1}'=x_{n-1}$, then all cycles are even, and the cycle set can be halved (same as condition 4).

8. If $x_n'=\bar{x}_n$ and all x_k' except x_n' are independent of x_n, then all cycles are even with even multiplicity.

9. If $x_n'=x_n$, and all x_k' except x_n' are independent of x_n, then all cycles are of even multiplicity.

10. If $x_n'=\bar{x}_n$, and all x_k' except x_{n-1}' are independent of x_{n-1}, then all cycles are even, and all cycles not divisible by four occur with even multiplicity.

11. If $x_n'=x_n$, and all x_k' except x_{n-1}' are independent of x_{n-1}, then the cycle set can be halved, each half having no odd cycles with odd multiplicity.

12. If $x_n'=\bar{x}_{n-1}$ and $x_{n-1}'=\bar{x}_n$, then all cycles are even, and the cycle set can be halved. (Same as condition 4).

13. If $x_n'=\bar{x}_{n-1}$ and $x_{n-1}'=x_n$, then all cycle lengths are divisible by four.

14. If $x_n'=x_{n-1}$ and $x_{n-1}'=x_n$, then the cycle-set can be halved such that one half contains even cycles only, and the other half can be rehalved.

15. If $x_n'=x_{n-1}$ and $x_{n-1}'=x_{n-1}\oplus x_n$ then the cycle-set can be separated into two subsets, one accounting for one-quarter of the total number of states, and the remaining containing only cycle lengths which are divisible by three.

Finally, one additional condition is presented:

16. If x_n' depends on x_n according to the equation

$$x_n'=\bar{x}_n g_n+x_n h_n,$$

where g_n and h_n are independent of x_n and individually neutral (actually, the neutrality of either g_n or h_n follows from the neutrality of the other and the nonsingularity of the C-matrix), then the number of odd cycles is no greater than 2^{n-1}.

Proof: The states may be divided into four disjoint sets, as follows:

A: $x_n=0$, $x_n'=0$

B: $x_n=0$, $x_n'=1$

C: $x_n=1$, $x_n'=0$

D: $x_n=1$, $x_n'=1$

The neutrality of g_n and h_n guarantee that the four sets are of equal size; i.e., each contains one-fourth of the total number of states. By definition of the sets, each member state of each set has a successor state in only certain of the other sets. This transition restriction is expressed by the branches on the graph of Fig. 10; e.g., each state in set A or set C ($x_n'=0$) is followed by a state in either set A or set B ($x_n=0$). Clearly, subsets of states only in sets B or C cannot be involved in odd-cycles, but each state in sets A and D can be made part of an odd cycle; so the number of odd-cycles is bounded above by 2^{n-1}, the total number of states in sets A and D.

The sufficiency of this condition is tedious to demonstrate. Its proof provides little insight, and is not given.

Special Cases of Interest

The C-matrix will now be formulated for several interesting special network configurations, and some preliminary observations will be made regarding the conditions of nonsingularity. Each of these cases amounts to a restriction on the logic of the network, and the effect of this restriction on the state behavior will be described in part.

The Feedback Shift Register

For this case, all of the delay elements are concatenated in a single chain, as shown in Fig. 11. The first element is driven by a single-output combinational network, whose inputs are the other state variables. The next-state equations are therefore

$$\begin{aligned} x_n' &= x_{n-1} \\ x_{n-1}' &= x_{n-2} \\ &\cdot \\ &\cdot \\ &\cdot \\ x_2' &= x_1 \\ x_1' &= f_1(x_1, x_2 \ldots x_n) \end{aligned}$$

and all but the top row of the C-matrix may be filled in immediately. E.g., for $n=4$:

$$C=\begin{bmatrix} 01010101 & 01010101 \\ 00110011 & 00110011 \\ 00001111 & 00001111 \end{bmatrix}$$

The top row is arbitrary unless the condition of nonsingularity is imposed. In this case, it follows from theorem 3 that

$$\begin{aligned} x_1' &= \bar{x}_n g_1 + x_n \bar{g}_1 \\ x_1' &= x_n \oplus g_1 \end{aligned}$$

where g_1 is independent of x_n. The characteristic number of g_1 is given by the left half of the characteristic number of x_1', and carries no further restriction. A necessary and sufficient condition for a feedback shift register to be nonsingular is therefore that x_1' depend on x_n in accordance with this relation. The network then takes the form of Fig. 12.

If the combinational logic is entirely linear, the register is then a linear feedback shift register, which is known to be a canonic form for all linear nets.[9] It is known that the nonlinear feedback shift register is not a canonic form for nonlinear nets, but there are good indications that it is a very versatile network, capable of producing a wide variety of different cycle-sets.

From the columns of the C-matrix, it is apparent that there are only two choices for the successor of each state. In fact, since the shifting action of the register amounts to a doubling of the binary number which it contains, then state no. s is followed by either no. $2s$ (modulo-2^n) if the digit of the top row is a "0", or state no. $2s+1$ (modulo-2^n) if the digit in the top row is "1." These constraints on the state behavior may be condensed in a "generalized" state graph, shown in Fig. 13 for $n=3$. Here the solid line transitions result from $g=0$, or $x_1'=x_n$, which is the pertinent equation for a simple loop of n delay elements, and the dashed-line transitions result from $g=1$, or $x_1'=\bar{x}_n$, which is the pertinent equation for a loop of n delay elements and a single inverter.

Many interesting constraints on the possible cycle-sets of the feedback-shift register can be derived from this view of the state graph as a superposition of two simpler state graphs. These results constitute a separate study and are not reported here.

The Safe-Asynchronous Network

From the definition of a safe-asynchronous network in a previous section, it follows immediately that every column of the C-matrix of a safe-asynchronous network differs in no more than one binary digit from the corresponding column of the identity C-matrix. E.g., the C-matrix

$$X=\begin{bmatrix} 0 & 1 & 0 & \underline{0} \\ 0 & \underline{1} & \underline{0} & 1 \end{bmatrix}$$

satisfies this condition, with the differing digits underlined. Thus, the binary number of each state agrees in all but at most one digit with the binary number of the successor of that state. The generalized state graph for safe-asynchronous networks is merely the binary n-dimensional cube, in which vertices (nodes) represent states, and the edges (branches) the permitted transitions. These graphs are shown for $n=2$, 3, and 4 in Fig. 14. (The branches carry no arrows, since transitions in either or both directions are allowed.)

The condition for the realizability of a given state graph in a safe-asynchronous network is now simply stated. If the state graph is otherwise realizable; i.e., if it has 2^n nodes, and has only one branch emanating from each node, then it is realizable in a safe-asynchronous network if and only if it is embeddable in the generalized graph of the n-cube. Many necessary conditions for embeddability can be stated in terms of more obvious features of graphs, but sufficient conditions are not known.

If the condition of nonsingularity is imposed, embeddability requires that the resultant cycle set contain even cycles only. This condition is also sufficient for realizability but the proof will not be given here.

The Fully-Self-Independent Network

If each switching function x_k' is independent of x_k, then the corresponding network will be called fully-self-independent. For the nonsingular case, this condition requires that $g_k=h_k$ in condition 16 of a preceding section so that

$$x_k'=g_k$$

where g_k is independent of x_k, and the result is that the number of odd cycles is bounded by 2^{n-1}. Actually, much stronger conditions of this type follow from extensions of condition 16, and place tight bounds on the multiplicities of short cycles. For $n=2$, the only cycle sets possible are $(1_2, 2_1)$ and (4), and for $n=3$, only $(1, 7)$, $(2, 6)$, and $(1_2, 3_2)$.

The Network with Cyclically Permuted Logic

Consider now the network whose next-state switching functions x_k' are all expressible in terms of a single function ϕ:

$$\begin{aligned} x_n' &= \phi(x_1, x_2 \ldots x_{n-1}, x_n) \\ x_{n-1}' &= \phi(x_n, x_1 \ldots x_{n-2}, x_{n-1}) \\ &\cdot \quad \cdot \quad \cdot \quad \cdot \\ x_2' &= \phi(x_3, x_4 \ldots x_1, x_2) \\ x_1' &= \phi(x_2, x_3 \ldots x_n, x_1) \end{aligned}$$

Thus, the function remains the same, but its arguments are permuted cyclically with the advance of subscript k on x_k'.

Specification of one row of the C-matrix is now tantamount to specification of the entire matrix. For $n=3$, for example, we may designate the bottom row with the characteristic number $[c_0, c_1, c_2 \ldots c_7]$ and then derive the rest of the matrix from the previous equations:

$$C=\begin{bmatrix} c_0 & c_4 & c_1 & c_5 & c_2 & c_6 & c_3 & c_7 \\ c_0 & c_2 & c_4 & c_6 & c_1 & c_3 & c_5 & c_7 \\ c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{bmatrix}$$

It is apparent that the c's fall into distinct groups by columns: all of the c_0's are in column 0; c_1, c_2, and c_4 occupy columns 1, 2, and 4; c_3, c_6, and c_5 are in columns 3, 6, and 5; and c_7 is in column 7. These groups are those defined by the cycles of the simple circulating shift register made up of a loop of 3-delay elements. This register yields the state graph consisting of the nodes and solid line transitions of Fig. 13, with cycle set $(1_2, 3_2)$ and state-groups (0), (7), (1, 2, 4), and (3, 6, 5). That this property is generally true may be easily proved. Indeed, it is not surprising in view of the fact that the simple circulating register itself has cyclically permuted logic: $x_k'=x_{k-1}$ $(k=2,3,\ldots n)$, $x_1'=x_n$.

Conditions for nonsingularity and for realizability of arbitrary cycle sets may be derived, based on this approach.

Conclusions

Campeau's C-matrix and the concepts behind it have been applied to the derivation of specific relationships between the state-behavior and the internal logic of autonomous sequential networks. Relations such as these lie at the core of the coding problem, which is central to the development of synthesis procedures for sequential networks generally. Although achievement of this long-range objective may not be immediately imminent, it is hoped that the preliminary results established here will set a pattern for further exploration of this important subject.

References

1. The Synthesis and Analysis of Digital Systems by Boolean Matrices, **J. O. Campeau.** *Transactions*, Professional Group on Electronic Computers, Institute of Radio Engineers, New York, N. Y., vol. EC-G(4), Dec. 1957, pp. 231–41.

2. A Study of the Memory Requirements of Sequential Switching Circuits, **D. A. Huffman.** *Technical Report No. 293*, Massachusetts Institute of Technology Research Laboratory of Electronics, Cambridge, Mass., 1955.

3. Switching Circuits and Logical Design (book), **S. H. Caldwell.** John Wiley & Sons, Inc., New York, N. Y., 1958.

4. Application of Boolean Algebra to Switching Circuit Design and to Error Detection, **D. E. Muller.** *Transactions*, Professional Group on Electronic Computers, Institute of Radio Engineers, vol. EC-3(3), Sept. 1954, pp. 6–11.

5. Gedanken Experiments on Sequential Machines, **E. F. Moore.** *"Automata Studies"* (book), **C. E. Shannon, J. McCarthy,** Editors, Princeton University Press, Princeton, N. J., 1956, pp. 120–53.

6. A Method for Synthesizing Sequential Circuits. *Bell System Technical Journal*, New York, N. Y., vol. 34, no. 3, Sept. 1955, pp. 1045–079.

7. The Synthesis of Linear Sequential Coding Networks, **D. A. Huffman.** *Proceedings*, Third London Symposium on Information Theory, London, England, Sept. 1955, also Information Theory (book), **C. Cherry.** Academic Press, New York, N. Y., 1956.

8. Synthesis of Electronic Computing and Control Circuits (book). Harvard Computation Laboratory, Harvard University, Harvard University Press, Cambridge, Mass., 1951, pp. 266–70.

9. Theory of Autonomous Linear Sequential Networks, **B. Elspas.** *Transactions*, Professional Group on Circuit Theory, Institute of Radio Engineers, New York, N. Y., vol. CT-6, no. 1, Mar. 1959.

10. On the Measure of Normal Formulas, **R. McNaughton.** *Pacific Journal of Mathematics*, Berkeley, Calif., vol. 7, no. 1, 1957, pp. 969–82.

System Evaluation and Instrumentation of a Special-Purpose Data Processing System Using Simulation Equipment

A. J. STRASSMAN L. H. KURKJIAN

TESTING and instrumentation are essential prerequisities for the completion and operation of any new system. A system can be defined as a number of components that are amalgamated or integrated together to perform a desired operation. Throughout this paper a "component" is considered to be a complete functional part of a data-processing system such as an arithmetic unit or a buffer. To ascertain if a component in the system is going to perform its specific function, it is sometimes necessary for the implementation of tests to be more complex than the component undergoing the testing. This becomes apparent when the component is a part of a large system and has many inputs and outputs.

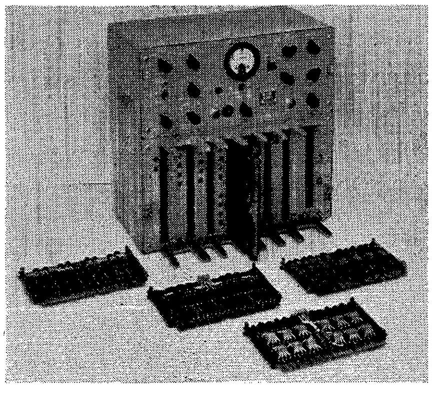


Fig. 1. Standard digital elements and element tester

To prove the system feasibility or operation of the components it is necessary to do either of two things: 1. duplicate and maintain an entire system and use it as one master-test fixture to evaluate each functional component; or, 2. provide individual test facilities for the evaluation of each of the functional components. The second approach requires the design of simulation equipment to provide the necessary inputs (control signals and data) to check out completely the operation of each individual component. It is believed that this approach offers the greatest advantages for large special-purpose data processing systems.

It is necessary to provide the proper work organization for the evaluation of these computer systems. A differentiation can be made between small and large systems and the work organization can be adjusted accordingly. Although the basic philosophy of test remains the same, the details evolved for the testing or evaluation of a small system will be different from that evolved for a large system. For the purpose of this paper in which the evaluation and instrumentation of a large system will be described, a "large" system will be defined arbitrarily

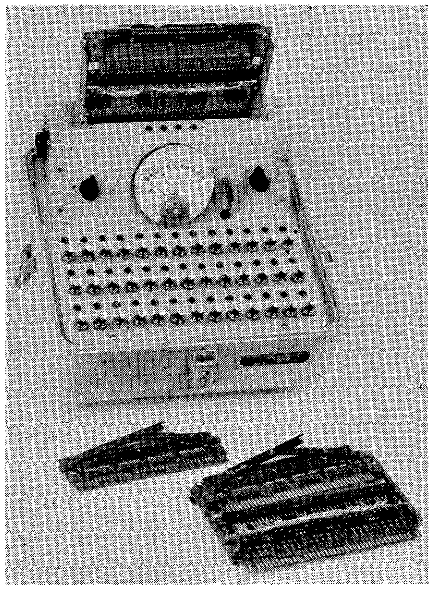


Fig. 2. Matrix assembly and matrix tester

as one that contains more than 500 flip-flops. Since the flip-flop is a basic part of any digital computer, the number of flip-flops can be used as an indication of the size and complexity of the system.

The philosophies contained within this paper led to the basic planning considerations for the test and evaluation of a special-purpose data processing system; parts of which will be described in later paragraphs. This data processing system contains approximately 1,500 tubes, 2,500 transistors, 40,000 logical gating diodes and 3,500 flip-flops. Each flip-flop in the system has four transistors making a total of 16,500 transistors in the entire system. This qualifies the described system to be classified as a large system.

In the case of a small system, all the

A. J. Strassman and L. H. Kurkjian are with Hughes Aircraft Company, Fullerton, Calif.

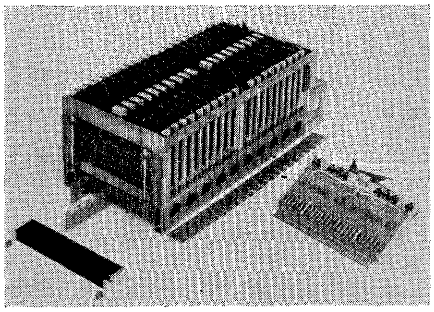


Fig. 3. Digital unit module

work can be handled by a small group which will perform the necessary tasks from system design to final evaluation. This work includes the logical design, circuit design, and testing. It is admitted that to be able to perform all the tasks included, the technical personnel associated with a small system must be those of broader background than those required for the evaluation of a large system. This becomes obvious when the personnel requirements are outlined for the large system. In this case, due to the actual limitations of time, complexity of large systems, and efficiency of utilization of personnel, specialists are needed in each specific work area to perform all the necessary tasks to complete the system. Specialization is indicated by the fact that the system design is done by a group of systems engineers whose function is to define the necessary components needed to implement the system and their interrelationships. Logical design and circuit design are two functions that are performed in an analogous manner by logicians and circuit engineers who are specialists in their respective realms.

The test and evaluation of the system is also handled in a specialized manner. Each component is assigned to a circuit or unit engineer whose responsibility is to 1. design the logical circuitry of the component from the Boolean equations, 2. design the test fixture, 3. write the necessary procedurals, and 4. test the component when it is fabricated.

Types of Tests to Be Performed for Evaluation

The basic parts of any large system can be broken down into five categories which are listed in their order of complexity: 1. elements, 2. units, 3. components, 4. subsystems, and 5. systems. If these basic parts of the system are evaluated and tested in order of complexity, the sequential building of testing integrity yields the understandable advantage of solving small problems first before becoming involved with the intricacies and troubles inherent in any large system. The first type of tests to be performed therefore would be element tests.

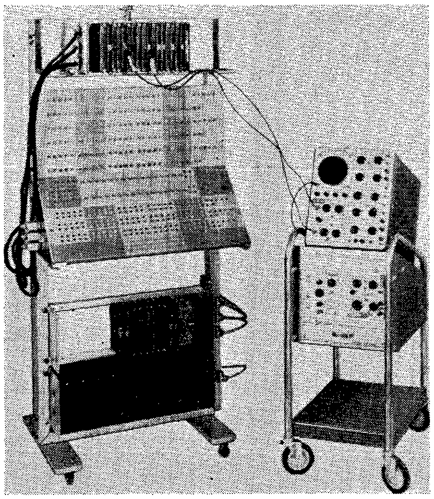


Fig. 4. Digital unit tester

Element Tests

The basic computing elements are usually flip-flops, logical followers, drivers, shift registers, diode cards, and pulse regenerators, etc. These items are referred to as "standard elements." The functional requirements of each of these standard elements determine the design of the test fixture necessary for evaluation. The test fixture for a flip-flop contains the necessary steering circuitry which makes the flip-flop a modulo-2 counter. When more than one flip-flop is built on a standard card, the fixture can be expanded to make the flip-flops count in any prescribed manner. Shift registers, followers, and drivers are most commonly evaluated by inserting specific computer word patterns at the input and observing the appropriate outputs. Simple sequential relay circuits are used to step through the forward and reverse characteristics of diodes on standard diode cards. Typical standard elements of a digital computer system along with a standard-element test fixture are shown in Fig. 1.

The electronic implementation of the digital computer logic that is represented in Boolean notation is formed from the standard diode card and specific resistor networks on the matrix card assembly. The wiring of the resistors to the diodes on each individual matrix card determines its logical function. The logical function of each card can be statically evaluated by a "matrix card tester" which simulates each input term to the card. The output of each gate is monitored as logical true and false levels are placed at the inputs to the gate. A meter is used to indicate to the operator the result of the simulation of any of the logical terms under test. Fig. 2 is a photograph of a matrix assembly and a matrix card tester.

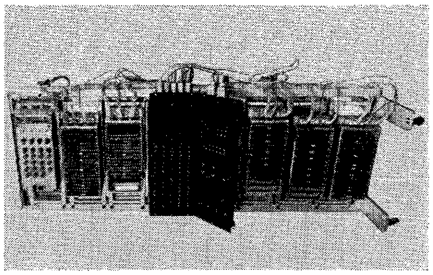


Fig. 5. Component test fixture

Unit Tests

Each module of the system under discussion is called a unit and contains up to 21 element cards. A unit module is shown in Fig. 3. The combinations of, and connections between the elements in the unit provide a portion of an over-all computing function that is to be performed by the system. Evaluation of units are difficult because units are incomplete functional items and therefore the amount of simulation becomes large and complex. However, it is considered that this step in the system evaluation is critical. It is therefore necessary to ascertain that each unit has been tested to the maximum. This obligates one to perform the most exhaustive tests possible on the unit level within the framework of the computer. Provisions to accomplish this can only be done by generating ideal simulated signals that the unit would expect in system operation. This type of simulation has been achieved by the design of equipment referred to as the "unit tester." The unit tester provides combinations of static and dynamic signals to the unit under evaluation. All system timing signals, synchronizing signals, and data inputs are generated in the unit tester. Each connection in and out of the unit under test is available on a patch panel on the unit tester. The choice of four signals is available at each point. The point may be: 1. connected to a logical true signal, 2. to a logical false signal, 3. to a special function (synchronization, timing, data, etc.), or 4. it may be left unconnected if it is an output of the unit that is to be observed. The unit test insures that the interelement wiring and the input-output wiring of the unit is correct and at the same time provides a semidynamic test to the various element

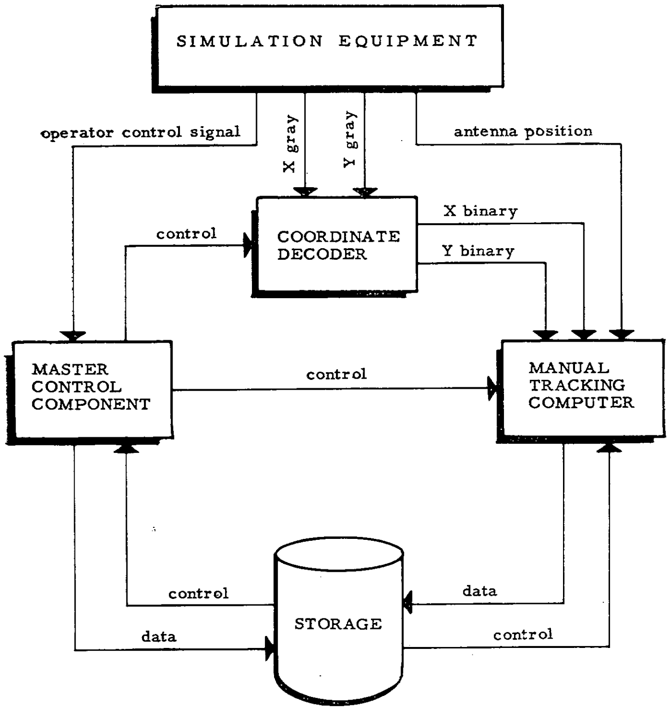


Fig. 6 (left). Typical subsystem block diagram

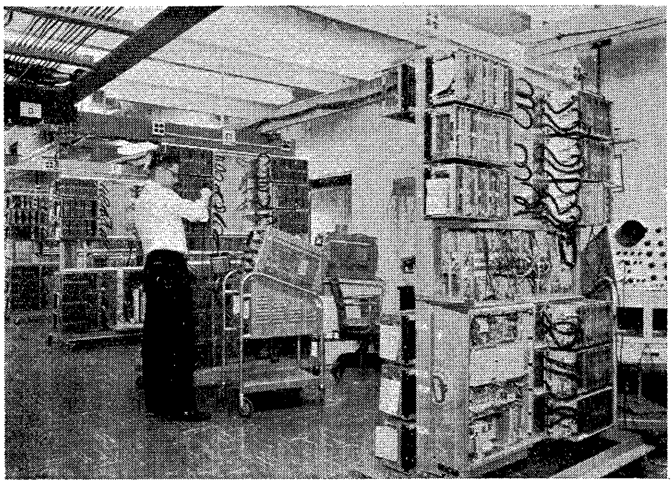


Fig. 7. System test fixtures

configurations. Many of the logical functions can be completely evaluated during this phase of test. The unit test can be easily modified for production testing of each module by simple automation techniques. In Fig. 4, a unit is shown under test in the unit tester.

Component Tests

A system component is a unit or group of units that has been defined in the system to perform a particular computing or data processing function. Examples of typical components are the arithmetic unit, the various buffers, the computer controls, and the buffer controls. It is at this level that the simulation of external signals is very important, as the completeness of testing at the component level determines the ease with which it is possible to system test and evaluate the entire system. The component test provides for the testing of all of the logic contained within the integrated units by means of the external simulated signals. These external signals are developed by a special test fixture that is unique for each component. A component consisting of eight units mounted on its test fixture is illustrated in Fig. 5. The test fixture is designed to simulate the complete system to the component. This test is basically dynamic, and as a consequence, logical errors can be discovered during this phase of evaluation. The simulation equipment consists of the appropriate switches, function generators, and timing and synchronizing signals that the component would operate from if it were in the system. Procedurals are written which outline the detailed steps necessary to evaluate the component function as specified by the system's design. This test actually provides or disproves the component logic with the test fixture as the system simulator. Both the test fixture and the procedurals are designed and written by the cognizant circuit or unit engineer who is charged with the responsibility of this component and has by necessity a complete grasp of the functional operation of this component. Typical examples of system components and an idea of their complexity follow.

1. The co-ordinate extrapolator updates co-ordinates on the basis of velocity stored in the memory. This component requires 12 flip-flops, 8 logical followers, and 180 diodes. The logic written in Boolean notation consisted of 3 typewritten pages and the test procedural was 9 pages long. The control and addition logic in this component were evaluated by means of a component test fixture which simulated the system input co-ordinate data, velocity, and time by means of variable word generators and counters.

2. The computer's control are in the form of a special-purpose wired program computer that controls information from and to three arithmetic units. Its outputs include control signals and generation of appropriate constants needed during the various steps in the wired program. The component was implemented with 15 units which contained 148 flip-flops, 384 logical followers, and 6,350 gating diodes. The logical equations in Boolean notation comprised 26 typewritten pages and the test procedural was 114 pages long. The control signals and the terms of the constant generators checked by a test fixture simulated the essential control signals required to cause the computer to perform each program step.

Subsystem Tests

After the component has been completely evaluated, the next step for system completion is to integrate the components together into the various subsystems as determined by a logical sequential build-up. Fig. 6 demonstrates a simple integration of one subsystem consisting of four components. There is less simulation equipment needed in this phase than for the component test. The example shown shows control and decoder components that have the facility for entering data into a special-purpose computer which steps through a wired program cycle and stores information on a magnetic drum. Parts of this information are used in the control component during system operation. This makes the subsystem a small closed loop within the system. Logical tie-in and timing errors can be found and solved during this part of system completion. Simulation equipment for subsystem tests usually consists of inhibiting signals that effect the closed loop operation and generating all those other signals which are necessary to make the loop operate. In the example shown, X and Y co-ordinate data in Gray Code, simple operator control buttons and radar-antenna position signals were the only signals needed to be simulated. Parts of existing component test fixtures can be used during subsystem tests as they contain the necessary simulation equipment.

System Test and Evaluation

This phase is the culmination of all the test and evaluation effort that has been as magnetic-ink character readers and converters, which the bank ultimately expect to have, approximately 7,000 square feet in total for the equipment was actually allocated.

Since this electronic data-processing system contains its own air-conditioning equipment and, in fact, air conditions the area in which it is situated, with exception of the room containing the magnetic-tape drives, there was a relatively small investment to make for air conditioning. This was to provide for controlling the temperature and humidity in the magnetic file room, which was approximately 1/6th of the total area. There was also provided at the site, the equivalent of approximately 90 gallons of water per minute, at a temperature not higher than 75 degrees, for the compressors included in the system and, of course, the necessary power supplies. Both supplies, that is water and power, were made independent of other requirements within the building, so that any failures resulting from the use of equipment in other areas would not have any effect on the system. The question, which is frequently raised, of whether or not an independent power supply for emergency use was needed was considered and it was resolved that this would be uneconomical in the long run. After all, if the bank should lose power throughout the main office today, it would be "off the air," so to speak, until it were restored. When recalling the multitude of proof machines, bookkeeping machines, tabulating equipment, and other electrical devices upon which the bank relies from day to day, it can be realized that it is not practical to install an emergency supply system of sufficient size to operate them in the event of a main line power supply failure. Hence, it is reasonable to say that exposure with an electronic data-processing system is really no greater, in the event of such a power failure, than it is today with conventional equipment.

Delivery of the equipment began shortly after the middle of April 1958, and on June 2, the engineers had completed assembly of all of the various units involved, and engineering tests and check-outs were commenced.

The contract provided for a minimum of 3 weeks of acceptance tests and these were started later that month. Acceptance tests were run for the required period under the general supervision of the computer expert, who had been retained originally as a consultant. The tests included not only the operation of various routines developed for regular daily use, but also specific programs written for the purpose of testing all of the various components of the system, and each of the mechanical and electromechanical pieces of equipment associated with it. By the middle of July, the bank was convinced that the equipment was satisfactory in all respects, with the exception of one unit, the high-speed printer, which at that time needed some further engineering work. Therefore, the system was placed on a rental basis, with the exception of the printer, on July 17, 1958. The printer and converter, as initially delivered, was essentially a prototype, and after rather extensive field testing a production model, incorporating many improvements in design, is being substituted for it.

When the equipment was delivered in June, the programming for the bank's deposit accounting operation was completed. This program is a comprehensive one intended to eventually handle more than 108,000 checking accounts. Since approximately 32,000 of these accounts are in the nature of special checking accounts where the checks themselves are in the form of punched cards and provide a ready means of input, these accounts were selected for the initial operation. Early in June they were transferred to magnetic tapes and processing on a day-to-day basis behind the old operation was begun. This procedure not only assisted evaluation of the equipment during the test period, but also enabled final polishing of the program itself.

In the middle of July, an attempt was made to operate the new system on a parallel with the old. This attempt eventually led to one conclusion: that it is not possible, as a practical matter, to operate an accounting function designed to make the most effective use of electronic equipment on a parallel basis with manual or automatic systems. The use of a computer system in data processing enables one to approach problems in a considerably different manner than it is otherwise possible to do in other systems. This difference in philosophy, combined with the greater speed and the high degree of accuracy, makes it extremely difficult to draw comparisons in any intermediate stages of handling data. The adjustments that were necessary to balance one system to the other were extremely cumbersome to handle on a day-to-day basis. These tests, however, did serve to convince many employees who were unfamiliar with this new method of data processing of the reliability and accuracy that could be expected of the new accounting procedures, and of the equipment itself. Early in September, procedures were altered so that regular processing was done electronically and the old system followed a day behind, in order that one final check of our cycled statements could be made. At the end of that month, the old system was abandoned entirely and, since that date this work has been done very satisfactorily on the new equipment. It might be of interest to note here, that a normal day's operating time on the computer to edit, sort, and post between 20,000 and 22,000 items to approximately 33,000 accounts, including the preparation of tapes for the printing of statements, lists of overdrafts, and numerous other special reports is in the order of 28 to 32 minutes, figures which compare favorably with the original estimates of the research group as to the time required for this operation.

As indicated earlier, other applications were being programmed and program debugging has continued. In the past few weeks checking out the routines for the Corporate Trust operation have been completed and the records, with respect to all of the bank's stockholders, are on magnetic tape files. These are being processed regularly on the new system. When the conversion of data in old files to tapes is complete, the bank will increase this operation to include in excess of 765,000 accounts. This conversion will take some time, since the practical problem of manually punching the equivalent of more than $5^1/_2$ million cards in order to translate the data in the old files into machine language must be faced.

There are one or two points that are unique about this particular application. First, it is believed that this is the first bank to apply fully automatic techniques, and second, the routine, itself, puts upon the computer the burden of calculating and assigning to each item, a key in lieu of an account number, which when sorted results in arranging the files in alphabetical sequence. In a file of the size with which this bank is concerned, the cost of looking up and assigning numbers to each item in the file, and to each transaction affecting the file, in order to post correctly and to preserve its alphabetical sequence is very high. Some banks have felt that this cost is sufficiently great to wipe out any gains that might otherwise be realized from other semiautomatic or automatic accounting systems requiring the numbering of accounts and have, therefore, stayed with manual systems. While the bank has not as yet had an opportunity to fully check out the application, it is believed that solving this problem of account numbering has been successful. Experiments under actual operating conditions with a file of approximately 25,000 stockholders have led to this conclusion.

It is frequently asked, "If you can

performed previously. All the elements, units, and components have been proved to perform within the framework of the several subsystems. It is now necessary to prove that the integration is complete by operational use of the entire system. Obviously a minimum of simulation equipment is required during the system evaluation. Fig. 7 shows system test fixtures with the components mounted. Since large systems are complex, it is considered good design to incorporate self-test features into the system. This involves the design of simulation equipment to be incorporated within the actual system hardware. This equipment is useful during the initial evaluation as well as during normal system operation.

Conclusions

It is apparent that any complex system can be tested and evaluated by a step-by-step instrumentation. Providing the necessary special purpose instrumentation has proved to be more rapid and economical than the accumulation of general-purpose testing devices. There are many instances in the testing of special-purpose computer components within a system that general-purpose instrumentation devices would not suffice no matter how much and ho𝔍 varied the instruments could be interconnected. In each of the stages of the system integration particular classes of errors and failures can be uncovered. During element tests, electronic part failures and mechanical errors are discovered and corrected. After element testing, each element is considered operative and the troubles found in unit tests cannot be attributed to the elements. During unit testing, logical and timing design errors can be uncovered and intra-unit connections are ascertained to be correct. At the completion of the unit test, each unit is considered to be completely operative. Therefore, during the component test phase, any difficulties discovered cannot be attributed to the unit, but rather to logical tie-in errors between units and interunit wiring. Similarly, the problems within the subsystem test are only related to those difficulties encountered in integrating more than one component because of the completeness of the component evaluation. System testing is merely an extension of the previous statements, but now referring to problems encountered in integrating subsystems. The sequential building of test complexity offers the advantage of solving small problems first before becoming involved with the intricacies and troubles inherent in any large system integration. Finally, the experience of the personnel involved in the test build-up enables a better understanding of the system operation thereby decreasing the time required to integrate a large system made up of many discrete and special components.

Discussion

Howard W. Childs (Sylvania Electronic Systems): I gained the impression that the test equipment was largely manual in operation. Is this true?

Glenn W. Bills (North American Aviation): Why wasn't tape-programmer automatic checkout equipment used in preference to manual checkout by technicians?

Mr. Strassman: These first two questions can be replied to with the same answer. In essence, this was a developmental model and therefore it did not behoove us to spend time and money to design special automatic checkout equipment to do the job. However, we could say that we had automatic technicians because they were provided with special instructions as to exact procedures to be followed for the checkout and what to do when a particular readout was not what it was supposed to be. However, the fact of economics primarily prohibited the design and use of automatic checkout equipment.

G. W. Smith, Jr. (Bell Telephone Laboratories): What provision is made for "in-operation" testing to indicate failures during service?

Mr. Strassman: I did not describe that particular portion of the system, but within the system there is built-in simulation equipment to provide external signals that provide facilities for in-operation testing. For instance, since this particular data processing machine is used for radar data processing, we have actually built into the system a radar target simulator which allows us to provide in-operation testing without an external radar.

W. A. Farrand (Autonetics Division, North American Aviation): To what degree do you find that the dynamic tests of elements provide workability criteria?

Mr. Strassman: Well, I could say it is 100% because the ones that do not work are not used. After testing the elements, it is ascertained completely whether or not they are working, and those that are not working are then repaired, tested again, and then put back into the system. So that we have proven workability of the items before they were actually installed in the system.

APAR, Automatic Programming and Recording

G. R. BACHAND J. L. ROGERS T. F. MARKER

THE AUTOMATED Integrated Data Systems (AIDS) program objectives are to: 1. automate weapon data activities reliably, economically and with realistic accuracies, 2. provide these facilities rapidly with a degree of flexibility for maximum utilization, 3. standardize a systems approach which takes into account technological advances.

One major development area is the automation of information acquisition, and germain to this paper is the problem of automatic acquisition of variables information in testing operations.

The systems needs are based primarily on acquiring data from quasi-static phenomenon at relatively low digitalizing rates and modest accuracies. In addition, building blocks should be made available having higher order capabilities commensurate with other application requirements.

In fulfilling the needs for an economical automated acquisition system, the development of functional modules has been completed which in various combinations, dictated by the application, make up an Automatic Programmer and Recorder (APAR). APAR, when equipped with the proper transducers or test heads, can be used to replace test equipment, the test equipment operator, the recording notebook or inspection sheets, the process of controlling equipment under test, and the process of recording the test data. This acquired information is converted to a machine language and recorded on punched paper tape. The people in the System Studies Department who have generated the concept of APAR and fostered its

G. R. Bachand, J. L. Rogers, and T. F. Marker are with the Sandia Corporation, Albuquerque, N. Mex.

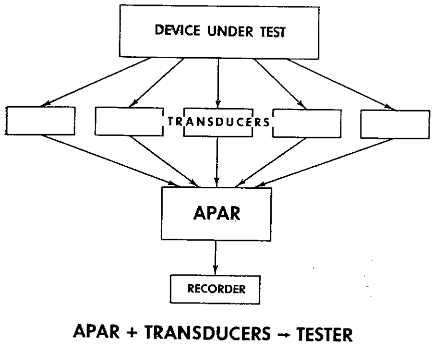


Fig. 1. Information flow diagram

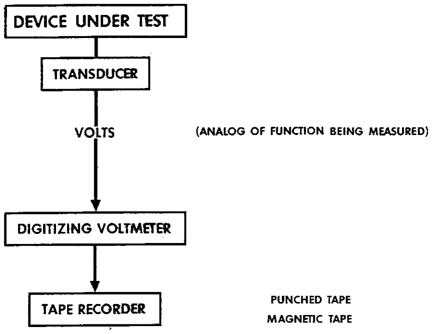


Fig. 2. Basic APAR block diagram

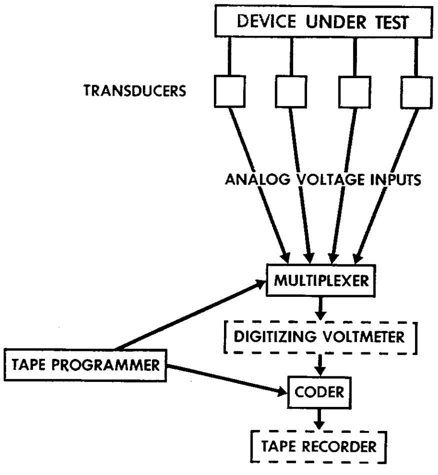


Fig. 3. Addition of multiplexer

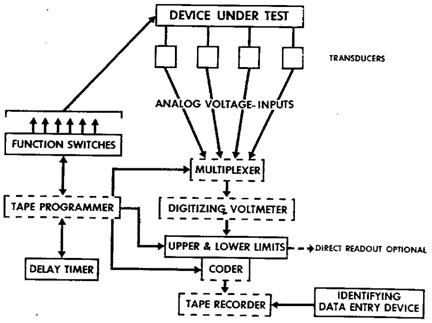


Fig. 4. Addition of control functions

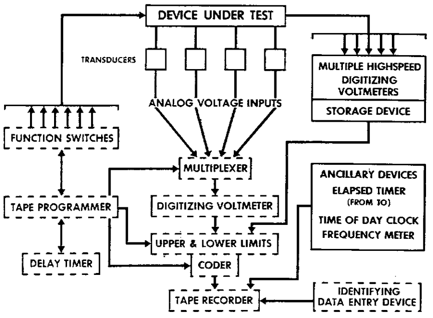


Fig. 5. Addition of multiple high-speed voltmeters

embodiment into hardware do not like to think of APAR as being a single piece of equipment, but rather a series of techniques available in separate modular forms which have a great deal of flexibility so they can be applied in many situations. Some of the illustrations reproduced here show some of the possible combinations that are feasible with the APAR concept. In addition to the APAR, other compatible systems are being developed which will provide immediate reduction of information coming from the APAR.

Fig. 1 shows a flow diagram of information from a device under test into the APAR and its recorder. Notice that between the APAR and the device under test there are groups of transducers which are used to convert the test parameter data into a common analog input language. The APAR then converts this into a digital machine language. The APAR can have provisions for two types of input analog languages: voltage and frequency. An APAR together with its transducers form a tester. For simplicity, the flow of programmed control from an APAR to the device is not shown. The next series of four illustrations will break the APAR into its functional parts to illustrate how it operates.

Fig. 2 shows the most basic parts of an APAR; a digitizing voltmeter and a tape recorder. A digitizing voltmeter is a device which converts the magnitude of a voltage into its equivalent digital code in machine language. The present recorder uses punched tape; magnetic tape may be used later. Notice that the device under test is connected to the digitizing voltmeter through the transducer. The function of this transducer is to take the information which may be a voltage, a current, a pressure, a force, or some other phenomenon and convert it into one of the common analogs previously mentioned which can then be converted into the punched tape code. Voltage has been chosen as the most common analog to be used because there are many transducers whose analog outputs are voltages and currents. If there were many parameters to be measured in the device under test in a simple system of this type, it would be necessary to supply separate digitizing voltmeters and tape punches for each transducer.

Fig. 3 illustrates that by adding a multiplexer to the digitizing voltmeter the various analog inputs can be switched to the digitizing voltmeter and only one digitizer and tape punch will be needed for many data inputs. The tape programmer shown in the left-hand side is a device which reads program commands from prepunched tape and directs the multiplexer to connect various data inputs to the digitizer for sampling. The system is flexible so that any input order can be chosen by punching it into the program tape prepared in advance of the test. Of course, with many inputs the data must be identified on a tape, and so it is necessary to add an encoding device which will identify the data from each input channel with a proper code. In this case, a 2-letter code is used to indicate each channel. There is a possibility of many input channels. At the present, APAR's have 24. This is a modular form, and the number can be increased in multiples of 24 as desired.

In Fig. 4 some other functions have been added. From the tape programmer there are controls to modules of function switches. These are used to control the device under test at the direction of the test program. Any combination of switches can be selected at any time to change the treatment or operation of the device being tested.

In series with the digitizing voltmeter another modular block called upper and lower limits has been added. If one wishes to check a reading from the digitizing voltmeter against a set of limits to determine if it is within specification, the programmer tells the upper and lower limits module what the limits are for the reading to be checked. The module makes a comparison, and if the value is found to be outside the limits selected, the machine will automatically insert a code for a comma in the data tape following the reading so that it can later be identified. There is a provision for an immediate alarm if desired. The test can be stopped at the time an outside-limit event occurs.

It is sometimes desirable to insert a discrete time interval between two events in the test operation. A provision for a delay timer is, therefore, incorporated, as shown in the lower left-hand corner of the

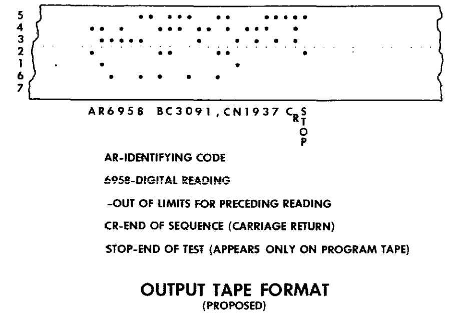

5
4
3
2
1
6
7
AR6958 BC3091,CN1937 CR STOP
AR-IDENTIFYING CODE
6958-DIGITAL READING
-OUT OF LIMITS FOR PRECEDING READING
CR-END OF SEQUENCE (CARRIAGE RETURN)
STOP-END OF TEST (APPEARS ONLY ON PROGRAM TAPE)
OUTPUT TAPE FORMAT
(PROPOSED)

Fig. 6. Proposed output tape format

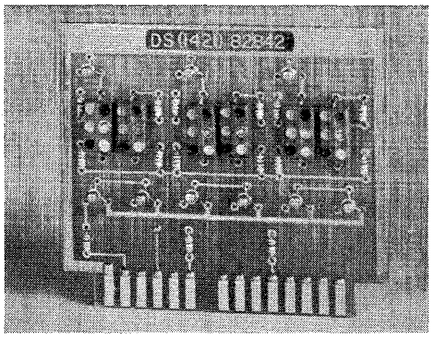


Fig. 7. Logical circuit card

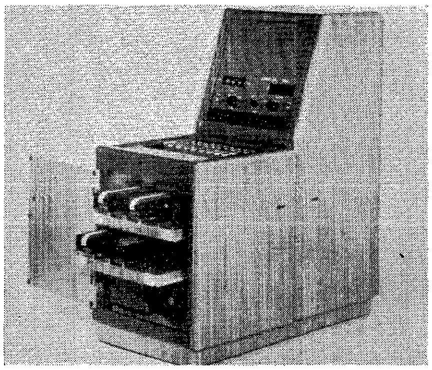


Fig. 9. Cabinet, showing drawers

illustration. The time interval and the when-it-is-to-happen instructions can be programmed.

Shown in the lower right-hand block is the identifying data entry device. Some means of identifying the information on each tape is needed. The device being tested, the tests that are being run, sometimes lot and serial numbers, and similar kinds of information must be known. APAR can be directed to repunch any information on its program tape into the output data tape; also another tape can be inserted into the programmer to read directly into the output tape recorder with this information prepared in advance on a Flexowriter. Generally speaking, each group of identifying information differs from the next only by changes in a lot or serial number, so that it would be considerable nuisance to hand-punch identifying information over and over with only small changes in the information. It is, therefore, desirable to have an identifying data entry device which has provisions for changing only a part of this information at a time. There is such a device which can enter identifying data sequentially into the data tape at any desired time. It has the capability of storing preset alphanumeric word groups, and any letter or numeral in the store can be changed without resetting or erasing all of the information. The store is always on display for the operator.

In Fig. 5 there are added multiple high-speed digitizing voltmeters. It is sometimes desirable to obtain data simultaneously on a number of channels. Because there are limitations to the speed at which the data tape punch and the digitizing voltmeter operate, it is necessary to have special high-speed digitizing voltmeters for parallel operation. The data may be generated faster than the punch can receive it so a temporary storage device is used to hold the information until it can be read into the tape punch. The multiple high-speed digitizing voltmeters are operated on command from the programmer. Their output information is temporarily stored in the memory which may be a tape drum or a small section of continuous magnetic tape, and then at another command from the programmer, the information is fed into the output tape punch.

There are other devices which are sometimes useful. One is an elapsed timer to tell when the test started and stopped. It is sometimes useful to add time-of-day information. The frequency meter is also used for frequency and time interval digitizing.

Fig. 6 shows a section of typical output tape format. The punched tape is 7/8 inch wide and it can store the codes for ten letters or numerals per inch of length. A lateral column on the tape contains the punched code for each letter or numeral. On this sample tape the first two letters, AR, tell that the information following was taken from the input channel identified as the AR channel. The next four digits are the variable data obtained from that channel. At the end of a specified test sequence, a carriage return is inserted. At the end of a group or test sequence, a stop code is inserted which is primarily a signal used to instruct later an International Business Machines Corporation 705 to start processing the previously entered data. After the reading on channel BC, there appears a comma. This indicates that reading 3091 was out of the limits programmed for that channel. This piece of punched tape could then be inserted into a print-out device such as a Flexowriter, and it would print a tab of this information, or it can be used in other types of intermediate level reduction for process control plots. It can also be sent to a data center for later processing.

The construction of this system and others which are coming along with it is on a modular basis. This is to obtain maximum flexibility in order that one can put together whatever combinations of modules are necessary for a particular testing operation. The lowest order module is a card, which is shown in Fig. 7. On one of these printed wiring cards is grouped several logic circuit functions. The cards are about $2^1/_2$ by 3 inches, and they plug into a rack which is called a tray, as seen in Fig. 8. In a tray there are provisions for 30 cards, and a group of cards arranged in the tray to perform some other complete function, such as upper-lower limits, is called a second-

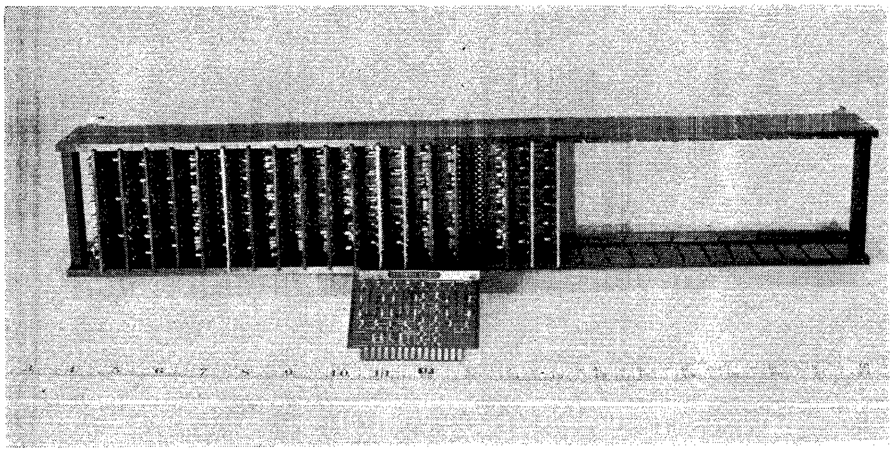


Fig. 8. Card trays in drawer assembly

order module. The trays are mounted in groups of five in an assembly called drawers. The drawers are further assembled in rocks or cabinets as shown in Fig. 9.

All of the computer circuitry and control circuitry is based upon the use of transistors. In the last several years, several types of transistors have proven to be unusually reliable. For example, one type which is frequently used in the circuits has never developed one defect in receiving inspection. More than 50,000 of these have been used and, excepting those damaged in development, it is proving to be one of the most reliable components.

The pioneer application of APAR in one of the weapon component development programs is now being made.

In conclusion, in meeting the physical realization of the AIDS objectives, there are no major technical barriers to overcome to determine adequately the reliability of atomic weapons.

Discussion

G. W. Bills (North American Aviation): What speed is available on the high speed digitizer?

Mr. Marker: Three units can be used for various speeds. One is the Epsco Transicon, which operates at 16,000 readings per second. There are two other designs which operate at 100 and 1,000 readings per second for a 10 per cent change of the input voltage. The accuracies for these two latter types are 0.1 and 1.0 per cent respectively.

Mr. Bills: Are there any provisions for dynamic testing, for example, a response to a step function?

Mr. Marker: A module is being developed for waveform digitizing using an analog sample-and-hold technique where analog storage of the samples is employed.

W. Brandt (International Business Machines Corporation): What is the character rate of the paper tape input and output?

Mr. Marker: Most generally, mechanical readers and punches are employed whose reading and punching rates are 20 and 10 per second, respectively. The programmer is adaptable to photoelectric reading at a rate of 600 characters per second. It is possible, although there is no existing requirement, to use commercial high-speed punches for output tapes whose punch rate is 800 per second.

C. S. Lin (M.I.T. Lincoln Laboratories): Is the 20-kc clock rate, the clock rate or the operating rate?

Mr. Marker: It is the average speed of the flip-flops and other logic circuits in APAR. The device is asynchronous, so it is difficult to specify an operating rate that is meaningful.

A High-Speed Transistorized Analog-to-Digital Converter

R. C. BARON T. P. BOTHWELL

Synopsis: With an increase in the use of digital techniques in the fields of instrumentation, data handling, and control, there has been an ever-increasing need for high-speed high-accuracy, analog-to-digital and digital-to-analog converters.

This paper describes a fully transistorized, reversible, high-speed data converter using the programmed successive approximation technique. Accuracy of 0.05% is obtained in an 11-bit binary encoder, 0.03% in the corresponding decoder. This technique is outlined in some detail as it pertains to the transistorized system.

Conversions are performed at 5 microseconds per bit or 16,000 conversions per second for an 11-bit binary code. Sign is automatically determined and the absolute value of the voltage encoded. A low-voltage reference is used and is electronic-chopper stabilized. The digital-to-analog conversion problem and high-accuracy high-speed techniques using transistor switches are discussed. A number of converter applications are shown.

Finally, over-all system performance of transistorized converters are compared with available vacuum-tube equipment regarding speed, accuracy, size, weight, power consumption, and reliability.

THIS PAPER describes a high-speed solid state converter having a conversion rate of 5 microseconds (μsec) per bit or 16,000 analog-to-digital conversions per second at an accuracy of ±0.05%. The converter can operate as an 11-bit binary or 13-bit binary coded decimal (3 decimal digits and sign) by simple interchange of one card. Stability of all circuits obviates need for zero adjusts or calculation knobs. Both internal and external reference voltages are provided for, and any conversion rate, up to the internal 200,000 bits per second, allows synchronization of the serial output to external tapes, drums or data transmission systems. After a discussion of the problem the converter will be described, then review a few of its many applications.

The many techniques for analog-digital conversion can be classified into three basic types, a sweep, step counter, and successive approximation method. All these basic techniques are widely covered in the literature, so only a brief summary will be given here. The sweep technique compares the unknown voltage with a linearly varying voltage and stops a counter when the two are equal. The step counter approach utilizes a forward-backward counter which incrementally adjusts a known voltage to match the input. The third basic method is the successive approximation technique in which the unknown voltage is compared first with one half full scale, then with one quarter full scale, and so forth.

The sweep technique has the apparent advantage of minimum equipment, but its speed is greatly limited. The step counter has advantages when examining

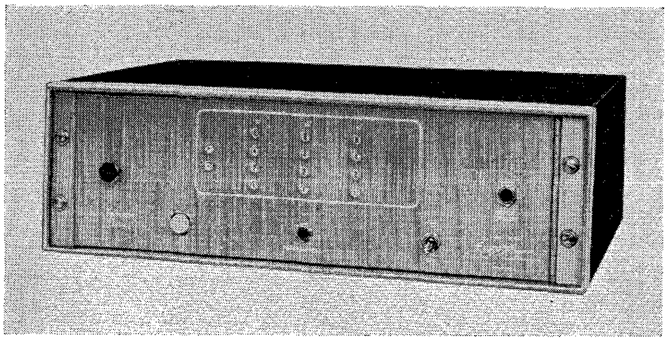


Fig. 1. Reversible voltage - to - digital converter

R. C. BARON and T. P. BOTHWELL are with Epsco, Inc., Boston Mass.

The authors are indebted to Mr. Bruce Smith and Mr. Bernard M. Gordon for ideas and suggestions incorporated in this paper.

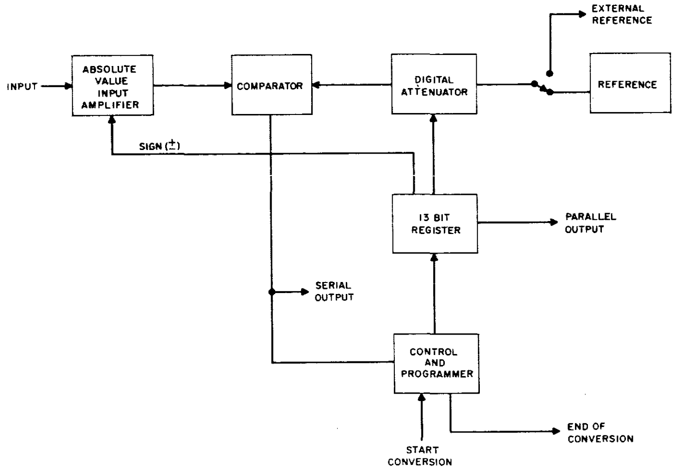


Fig. 2. Block diagram

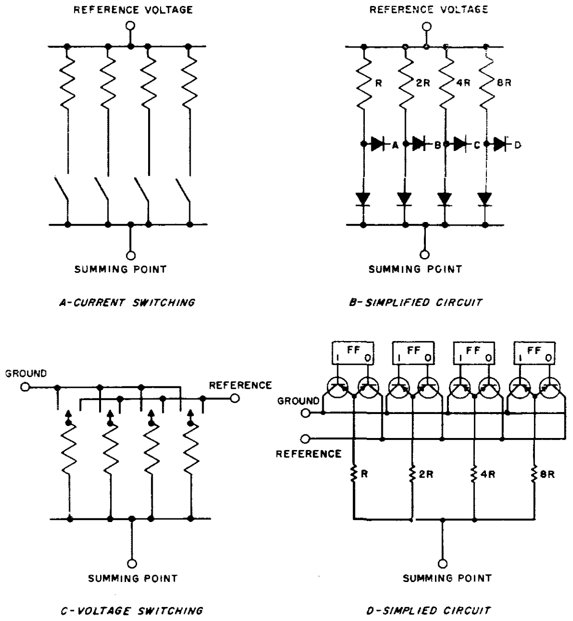


Fig. 3 (right). Basic switching circuit

incremental changes or following a slowly varying voltage but suffers greatly in terms of speed when operating on multiplexed or discontinuous inputs. The successive approximation technique is able to handle the highest rate of independent conversions; every conversion requires a precisely known period of time, and when all is considered, seems competitive in amount of equipment required. A transistorized successive approximation technique was selected as offering an optimum combination of reliability, low cost, and speed of operation.

System Description

Fig. 1 illustrates the Transicon Datrac converter. A block diagram of this converter is shown in Fig. 2. An absolute value input amplifier buffers the input to an impedance level suitable for matching into the comparator. The comparator always sees a positive voltage independent of actual input polarity except during the sign decision interval. An external or manual start pulse clears the register of the previous number and starts the conversion process. An analog voltage is generated by a programmed sequence of events determined by the control circuitry, the programmer, and the register. These set analog current switches which stimulate the analog signal into the comparator. These current switches form a digital attenuator, accurately attenuating the reference supply to the appropriate digital representation of the code in the register. The comparator is interrogated by the control circuitry at a 200-kc rate and a reject pulse is generated if the input current from the switches is greater than the current from the input amplifier. At the end of a conversion, the control circuitry is turned off and an end of conversion pulse is provided at the output.

The basic internal reference is a *IN430A* Zener diode controlling a 0.015% high-precision solid-state voltage regulator. An external reference can be substituted if desired. Both serial and parallel digital outputs are provided which are capable of driving external loads without buffers.

Digital Attenuator

Since the digital-analog converter is the heart of this equipment and strongly influences the design of its other components, it will be discussed first. Prior to discussion of the specific technique used, a discussion of semiconductor switching is in order.

Two basic methods of switching are shown in Fig. 3. For purposes of discussion, they have been called current and voltage switching.

In simplest terms, current switching implies a constant current in the resistors of the digital attenuator (D-A) network, the switching function only determines which of the available currents are selected as the output and which are shunted into a current sink. Voltage switching is the term applied to the case in which the voltage applied across the summing resistors is switched, while the resistors themselves remain tied permanently to the output.

The current switching method is the one most commonly used with vacuum tubes. If points *A*, *B*, *C*, and *D* are high with respect to the summing point, current is drawn from each leg according to the ratio of the resistors. If any of the inputs *A*, *B*, *C*, or *D*, are negative with respect to the summation point, the current in that leg is shorted out and that switch is opened. This is a simple switching scheme although it necessitates the careful selection of diodes or the use of trimming resistors.

A severe disadvantage appears when this scheme is used with solid-state switching devices. The dependence on temperature of the voltage drop across semiconductor diodes, together with their changes in life result in a high reference voltage requirement. Assuming a temperature coefficient of 2 millivolts per degree centigrade (C) and a temperature differential of 40 C, the total uncertainty in voltage drop is 80 millivolts. For 0.05% accuracy, this necessitates a reference supply of 160 volts. Even with the best commercially available silicon diodes, temperature plus life changes imply a reference voltage awkwardly high for a solid state regulator to handle.

If the diodes in Fig. 3B are replaced with germanium transistors in order to improve the quality of the switch, new problems arise. Assuming a leakage current of 0.3 microampere (μa) at 25 C and a maximum operating temperature of 50 C, for 0.05% accuracy, full scale current must equal 30 milliamperes (ma). If used in an A-D converter, this would place severe requirements on the comparator and on the input buffer, and would set impracticably high power levels in all the associated circuitry.

Voltage switching, on the other hand,

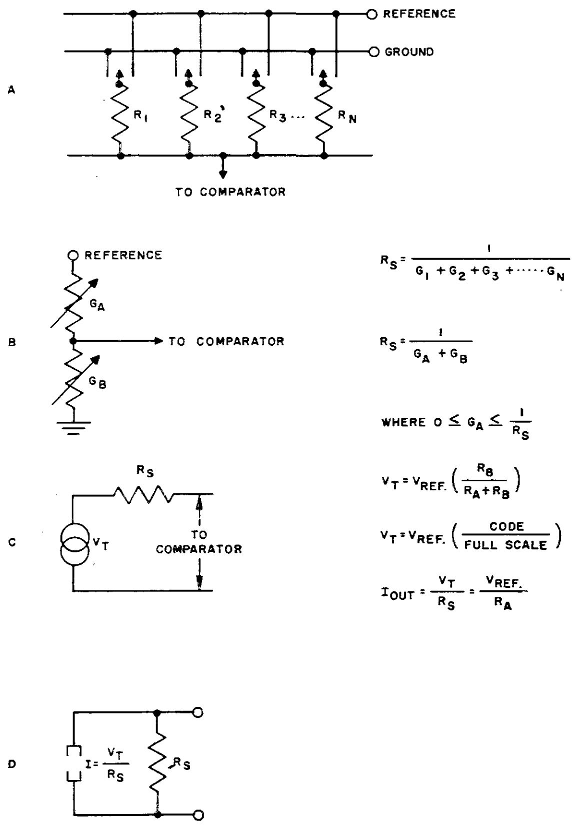

REFERENCE
GROUND
A
R_1
R_2
$R_3 \cdots$
R_N
TO COMPARATOR
REFERENCE
G_A
B
TO COMPARATOR
G_B
$R_S = \frac{1}{G_1 + G_2 + G_3 + \cdots G_N}$
$R_S = \frac{1}{G_A + G_B}$
WHERE $0 \leq G_A \leq \frac{1}{R_S}$
R_S
C
V_T
TO COMPARATOR
$V_T = V_{REF.} \left(\frac{R_B}{R_A + R_B} \right)$
$V_T = V_{REF.} \left(\frac{CODE}{FULL\ SCALE} \right)$
$I_{OUT} = \frac{V_T}{R_S} = \frac{V_{REF.}}{R_A}$
D
$I = \frac{V_T}{R_S}$
R_S

Fig. 4 (left). Thevenin equivalent of switch

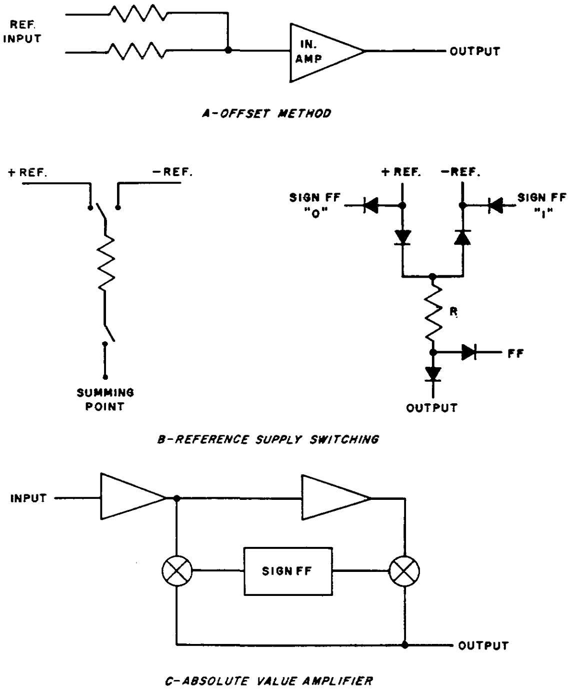


Fig. 7 (right). Three methods of achieving bipolarity

allows a much smaller reference supply voltage to be used. The drop across the switching transistor can be made very small; for example, it is practical to design switching circuits in which the drop from emitter to collector is less than 2 millivolts essentially independent of temperature. This allows use of a 20 volt reference with switching accuracy of 0.01%. Because of the use of two transistors in push-pull, the summing resistor is clamped either to ground or to the reference supply, depending on whether the switch is "on" or "off." As a result, leakage current in the off transistor ceases to be a problem since the cutoff transistor always leaks into a conducting transistor. Comparing the impedance of the conducting transistor to that of the summing resistor, it can be seen that the leakage current is attenuated by a factor of at least 2×10^4. The principal limitation of the voltage switching scheme seems to lie in the breakdown voltage of the transistors used as the switches (they must stand the full reference voltage plus a margin) and their minority carrier storage characteristics which determine maximum switching rate. Both these factors prove to be tolerable with several available transistor types.

A voltage-switched network of the type shown in Fig. 3(D) is also a digital voltage attenuator with respect to the reference voltage, and obviously displays a constant output impedance. Fig. 4 shows the Thevinin equivalent of switch. Fig. 4(C) shows the valid equivalent circuits for a network of this type. When operated into a short circuit or low impedance load, the Thevinin transformation of Fig. 4(D) may be a more useful equivalent.

The constant source impedance property of the digital attenuator makes it a particularly desirable digital-to-analog converter. When the Transicon Datrac converter is operated in the reverse mode, D-A, the digital attenuator may drive the output directly, without buffering, and delivers a current which is exactly proportional to the digital setting of the register regardless of the magnitude of the resistive load. For bipolar outputs, the absolute value amplifier is used as an output buffer. Digital-analog accuracy is ± 0.03%.

The Transicon digital attenuator, Fig. 5, was set to operate with a reference of 16 volts for binary systems. In decimal systems, 16-$^2/_3$ volts is used, giving a Thevinized 10-volt full scale. The digital attenuator is summed with the input into the comparator.

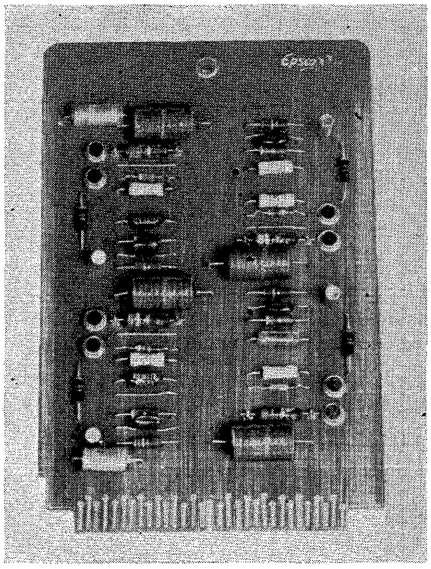


Fig. 5. Digital attenuator

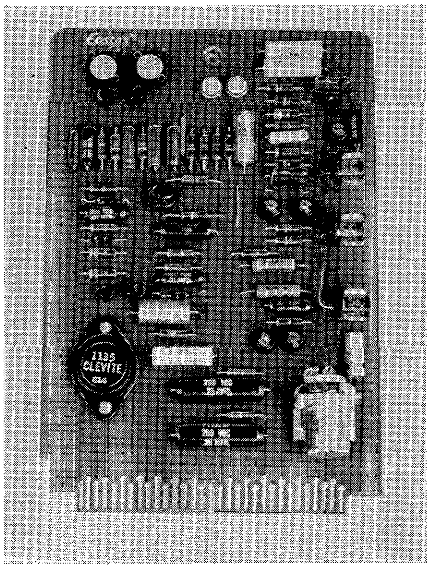


Fig. 6. Reference supply

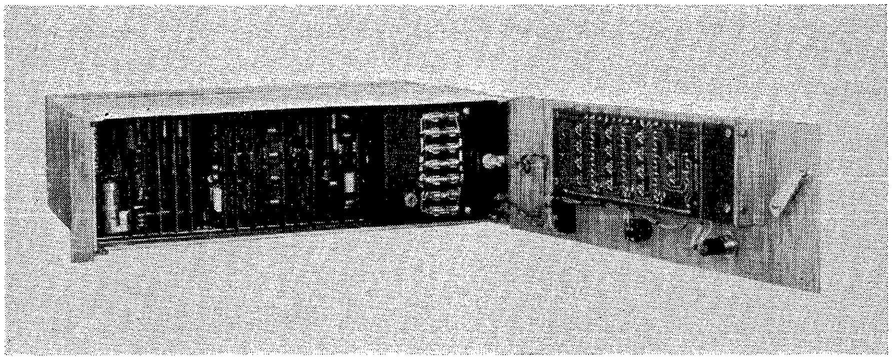


Fig. 8. Interior view of converter

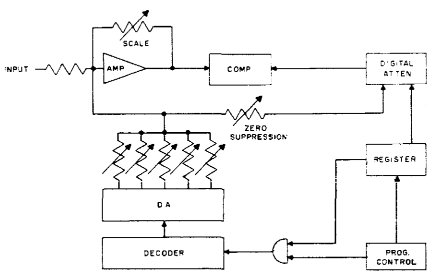


Fig. 11. Span control and linearization

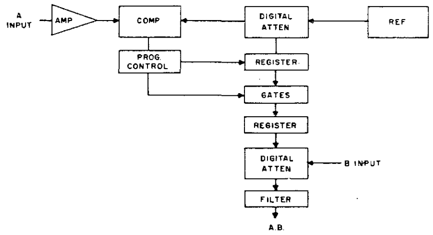


Fig. 12. Analog multiplication

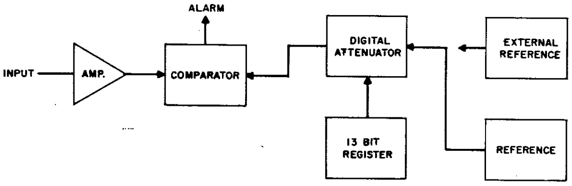


Fig. 9. Alarm limit monitor

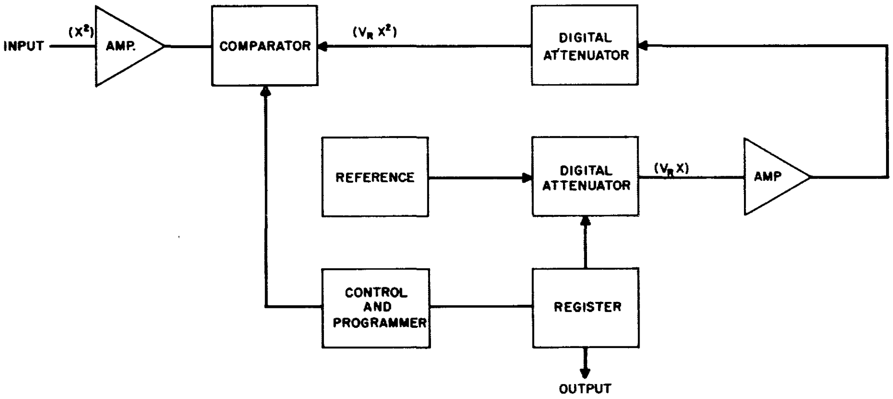


Fig. 10. Square root generation

The solid state comparator is of the pulse amplifier type having a sensitivity of a fraction of a millivolt. When interrogated, it supplies a reject pulse if the sum of the input voltage and its internally generated analog is less than zero. Long term drift of this comparator is ±1 millivolt without any calibration or adjustment. Freedom from calibration is an essential feature in any equipment which must operate in a large system over a long period.

Programmer

A specially designed magnetic programmer was used in the converter. Essentially, it is a magnetic ring counter using a square-loop blocking oscillator per stage. It has two principal advantages. One is low cost due to the use of only one core and one inexpensive transistor per bit. The second is the use of magnetics which permit permanent storage. This enables the converter to operate at any repetition rate from 0 to 200kc. This feature is particularly useful when the serial pulse output is required to be synchronous with the users clock, as for example, in a magnetic drum or tape system.

The reference supply, Fig. 6, is a ±0.015%-precision, solid-state chopper stabilized operational amplifier having an output impedance less than 100 milliohms. In addition to providing current to the encoders, the reference is available externally. If desired, an external reference supply may be switched in for ratiometer measurements.

Absolute Value Amplifier

Several methods of achieving bipolarity in an essentially unipolar converter are shown in Fig. 7. One method is to offset the input amplifier so that ground-in equals half full-scale output. With this method, plus and minus full scale input equal full scale and ground output. This results in complementary outputs for negative inputs and proves disadvantageous for some applications.

Another method of bipolar operation is to switch the polarity of the reference supply, depending on the sign of the input. One means of doing this is shown in Fig. 7(B). This involves diode switching with the associated stability problems. Other schemes are available but all involve the use of two reference supplies with the associated switching problems.

In this, unit bipolarity is achieved by the use of an absolute value amplifier. This involves two cascaded stages of chopper-stabilized operational amplifiers, with the output of the first or second switched to the comparator according to the sign of the input. Use of a transistor switch allows the comparator input to be stabilized within 10 μsec after a change of amplifier input polarity.

Modular Construction

The Transicon Datrac is modular in design. Fig. 8 provides an interior view of one standard model. The use of standard printed-circuit building blocks enable the user to expand his system as the need develops. An 11-bit binary converter can be converted to a 3-decimal digit converter in a matter of hours.

Transicon Datrac System Applications

The modular and flexible basic design of the Transicon Datrac converter allows it to perform a wide variety of arithmetic control, and data processing functions as well as the normal digital-analog and analog-digital conversions. Four examples are blocked out in Figs. 9 through 12. They are "limit monitoring," "square root generation," "span control and linearization," and "analog multiplication."

In the limit monitoring operation, Fig. 9, the digital attenuator is slaved to a digital signal representing the desired limit, and continuous comparisons at 100 kc are made between the number and the input. An alarm pulse is provided when the input exceeds the specified value. A single converter could monitor 100 independent signals at 1,000 samples per second, using multiplexed inputs and plugboard or toggle switch stored constants.

Fig. 10 illustrates a square root generation technique. A feedback analog voltage is generated which is proportional to the square of the number in the register by cascading two digital attenuators both of which are identically controlled by the register. Since the operation of the converter forces the feedback to agree with the input, the number generated during conversion in the register (x) must be proportional to the square root of the input (x^2). This mode of operation may be alternated with normal conversion by switching the additional digital attenuator.

Fig. 11 shows a combined setup for span control and linearization. Zero suppression and scaling are performed in the converter input amplifier. Linearization is accomplished by a two-pass conversion. On the first pass, the raw correction increment is selected; the second pass results in conversion of the corrected voltage. The correction is selected by the most significant bits resulting from the first pass. These bits are applied either to a small external memory which yields the correction, or are decoded and applied as a selection voltage in a simple potentiometer or toggle switch "memory." The great advantage obtained from this technique is that the correction for each interval may be independently selected, making linearization of complex functions nearly as simple as that of monotonic functions.

Fig. 12 shows an analog multiplier which simultaneously digitizes the multiplicand. In the form shown, the multiplier is able to digitize large amplitude complex waveforms where frequency components are below half the sampling rate, or 8 kc. If a forward-backward counter is substituted for the registers shown, (a single counter can replace both registers and the gating), signals with frequency components up to 200 kv may be followed. Because the maximum rate of change of the counter is 200 bits per millisecond, a full scale excursion for a counter-multiplier requires 10 milliseconds.

Comparison with Vacuum-Tube Converters

As the use of digital techniques increases, there will be a need for more and better converters. Transistorized converters have advantages which will enable them to capture much of this market.

During the past few months, not only Transicon but also some of the other available equipment have been evaluated. Based on these tests plus information gained elsewhere, a comparison can be made between equivalent vacuum tube and transistorized converters.

At present, the transistorized converters are somewhat slower than the fastest tube converter, the Datrac. However, there is no intrinsic reason why a transistorized converter could not be operated as fast. The accuracies of the two systems are comparable. The Transicon requires approximately 15% of the volume and weight and 10% of the power consumption of its tube equivalent. The reliability of the Datrac has been proved in over 1,000,000 hours of field operation. Transicon is expected to do as well. Finally, there is the question of cost. Despite the somewhat higher cost of transistors, the transistorized converter is substantially less expensive than an equivalent tube converter. This results from savings in power supply, metalwork, assembly, and simplified circuitry which solid state techniques afford. The next year should witness a definite increase in the use of transistorized systems in the converter field.

Discussion

J. A. Gariocchi (Radio Corporation of America): What are the maximum and minimum values of the ladder resistors in the digital attenuator and how accurate are they?

Mr. Bothwell: The smallest resistor is approximately 13,000 ohms. Accuracy is 0.02 per cent.

The largest resistor in the digital attenuator is about 700,000 ohms, and this is about as large as practical to build in the allowed space.

The accuracy required of the precision resistors, of course, decreases in proportion to their binary weight and their decimal significance. The most significant digits require the greatest accuracy, and so forth.

Mr. Gariocchi: How do you maintain accuracy transistor switching for such a large range of ladder load currents?

Mr. Bothwell: The ladder design is such that the range of load current is restricted to approximately ten to one. In addition, the switching accuracy improves as the current is described so that the switch need only be designed for the largest load current.

Mr. Gariocchi: What transistor types are used in the transistor switch?

Mr. Bothwell: Without going into very specific details, a number of transistor types will give reasonable yield. The General Transistor *2N520*, the Radio Corporation of America *2N269* or *2N404*, and practically any other high-gain medium-frequency alloy junction types will give switching performance in the order of magnitude that we discussed. The specific transistor we use is purchased to our specification at the moment, in order to be sure that we do not have to select. To get optimum performance, examine the equations of Ebers and Moll, and optimize the ratio of forward to reverse gain.

R. Reichard (Computer Control Company): Can you cite statistics on the reliability of the Datrac?

Mr. Bothwell: I think Mr. Reichard is asking for reliability on the Transicon Datrac. For the Datrac, we have had 300 units in the field for several years, and with the normal life time of vacuum tubes, we believe the reliability of that unit has proven to be extremely high. Definitive statistical data on commercially sold instruments is difficult to compile since the unit becomes the property of the customer after it is sold.

The Transicon Datrac reliability statistics are going to be more difficult to give. This unit has only been in production for 6 months. There is not enough live data available to be really significant there but we expect reliability at least as good as the tube equipment gave.

Mr. Reichard: Are failures largely due to the chopper and is this an electromechanical one?

Mr. Bothwell: The answer is that it is an electromechanical chopper used extremely conservatively. The requirements of it are quite modest. We believe the life of this chopper will be extremely long.

Gerald Smith (Daystrom Instruments): Could you briefly describe the type of circuit used for your "compare" functions?

Mr. Bothwell: Essentially, an electronic switch device chops the signal at the error junction between our analog feedback and our input signal. If the analog feedback is not equal and opposite to the input signal, a chopped signal results which passes through a high-gain a-c amplifier and a phase sensitive demodulator. The demodulated pulse signal is, then, the decision signal.

M. Garden (Leeds and Northrup): Is the three millivolt area of the reference supply due to drift or to loading?

Mr. Bothwell: With a 100-milliohm source impedance, we get about a tenth of a millivolt shift due to maximum loading.

The drift which I mentioned is due to two factors. One is temperature drift of the reference Zener diode. The second factor is finite amplifier loop gain and its internal drifts. These factors contribute less than 3 millivolts from 20 to 50 C.

Mr. Garden: Is the external reference required to have the same impedance and recovery time as the internal reference, or is an amplifier provided to match the external reference to the ADC?

Mr. Bothwell: No amplifier is provided unless specifically requested, but it can be provided.

It is not necessary to have quite so low a source impedance as we have, but it is necessary to have a source impedance of less than approximately 0.3 ohms; otherwise, the accuracy will suffer to some extent. The recovery time is very necessary. This can be obtained by putting a good tantalum capacitor across the terminals of the device or across the terminals of the reference supply voltage. The Transicon is not intended to operate from high impedance reference unless a buffering amplifier is included in the equipment.

The Trial Translator, An Automatic Programming System for Experimental Russian-English Machine Translation

V. E. GIULIANO

GRAMMATICAL and syntactic rules designed for the production of "smooth" automatic translations between human languages will be called translation algorithms. Because of the rigid constraints imposed by automatic machines, the formulation of translation algorithms involves unprecedented problems of a linguistic nature. Much of the existing literature in the field of machine translation consists of theoretical treatments of these problems, or of appropriately simplified abstractions from these problems. Several specific algorithms for the machine translation of Russian to English have been proposed, but none have been tested on a large scale.[1] These rules are apparently based on the intuition and linguistic backgrounds of the individual writers, sometimes aided by the manual analysis of short text samples. Until very recently the use of automatic machines has been confined to the application of *ad-hoc* computer programs, tailored to the processing of particular sentences or carefully selected texts.

At the present time it is exceedingly difficult, if not impossible, to evaluate most of the schemes which have been proposed for Russian-English translation. This difficulty is due to the absence of any practical means for testing these schemes on large bodies of text. Many of the proposed algorithms will, doubtlessly, work in the majority of cases. There remains, however, the question of whether they are "fail-safe." That is, do they lead to annoying but safe nonsense when they fail, or do they lead, instead, to smooth but incorrect sentences? It is possible, but highly improbable, that most of the fundamental linguistic problems connected with machine translation are already solved in the literature. At present, however, there is no way of testing the validity of this conjecture.

An automatic programming system, called the Trial Translator, is suggested in this paper for the automatic testing of translation algorithms. This system accepts, as its inputs, a set of experimental translation algorithms expressed in a suitable pseudocode language, and a large body of Russian technical text. The system applies the algorithms to the text and produces, as its output, a readable trial translation, the English text resulting from the application of the given rules to the given text. The Trial Translator contains three main subsystems, the Automatic Dictionary, the Formula Inserter, and the Specifier-Evaluator-Editor (Fig. 1).

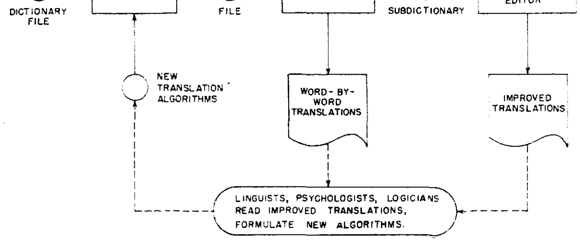

RUSSIAN TEXTS
RUSSIAN-ENGLISH
DICTIONARY FILE
FORMULA INSERTER
AUGMENTED
FILE
AUTOMATIC DICTIONARY
TEXT-ORDERED
SUBDICTIONARY
SPECIFIER-EVALUATOR-EDITOR
NEW TRANSLATION ALGORITHMS
WORD-BY-WORD TRANSLATIONS
IMPROVED TRANSLATIONS
LINGUISTS, PSYCHOLOGISTS, LOGICIANS READ IMPROVED TRANSLATIONS, FORMULATE NEW ALGORITHMS.

Fig. 1. The Trial Translator system

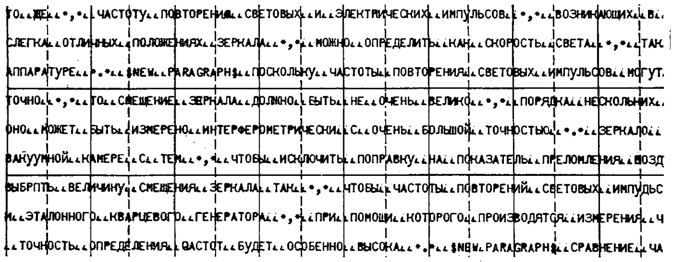
The Automatic Dictionary

Suppose that the entries in a Russian to English dictionary are recorded on a medium which can be read by an automatic computer, say on a reel of Univac magnetic tape. A Russian text can be recorded on another reel of magnetic tape from a manual input device, say a Unityper. A complex of machine programs can be written for the computer, a Univac I in the example considered, which accepts these two reels of tape as inputs, and which produces a tape containing a word-by-word translation of the text as its output. Such a complex of machine programs, together with the dictionary file, will be called an Automatic Dictionary.

An automatic Russian-to-English dictionary was originally proposed by Oettinger in 1954, as a "linear approximation" to a completely automatic translating system.[2] A concrete system of machine programs for the operation of an automatic dictionary, with Univac I equipment, was suggested by the writer in 1957.[3] This system, forming the operating part of the Harvard Automatic Dictionary, has since been programmed and is currently in operation. A semiautomatic dictionary compilation process has been developed, and has been used for the preparation of a Russian-English dictionary file for the fields of

V. E. GIULIANO is with the Computation Laboratory of Harvard University, Cambridge, Mass.

The author is indebted to his colleagues at the Harvard Computation Laboratory for valuable advice and encouragement. Professor A. G. Oettinger, Mr. R. Karp, and Mr. W. Foust gave especially valuable assistance. The writer is also indebted to Professor H. H. Aiken for his continued interest in the machine translation project at the Computation Laboratory. The author's research has been supported in part by the Ohio Oil Company, the United States Air Force, and the National Science Foundation.



```
ТО⌞⌞ЖЕ⌞⌞*,*⌞⌞ЧАСТОТУ⌞⌞ПОВТОРЕНИЯ⌞⌞СВЕТОВЫХ⌞⌞И⌞⌞ЭЛЕКТРИЧЕСКИХ⌞⌞ИМПУЛЬСОВ⌞⌞*,*⌞⌞ВОЗНИКАЮЩИХ⌞⌞В⌞⌞
СЛЕГКА⌞⌞ОТЛИЧНЫХ⌞⌞ПОЛОЖЕНИЯХ⌞⌞ЗЕРКАЛА⌞⌞*,*⌞⌞МОЖНО⌞⌞ОПРЕДЕЛИТЬ⌞⌞КАК⌞⌞СКОРОСТЬ⌞⌞СВЕТА⌞⌞*,*⌞⌞ТАК
АППАРАТУРЕ⌞⌞*.*⌞⌞$NEW⌞⌞PARAGRAPH$⌞⌞ПОСКОЛЬКУ⌞⌞ЧАСТОТЫ⌞⌞ПОВТОРЕНИЯ⌞⌞СВЕТОВЫХ⌞⌞ИМПУЛЬСОВ⌞⌞МОГУТ⌞
ТОЧНО⌞⌞*,*⌞⌞ТО⌞⌞СМЕЩЕНИЕ⌞⌞ЗЕРКАЛА⌞⌞ДОЛЖНО⌞⌞БЫТЬ⌞⌞НЕ⌞⌞ОЧЕНЬ⌞⌞ВЕЛИКО⌞⌞*,*⌞⌞ПОРЯДКА⌞⌞НЕСКОЛЬКИХ⌞⌞
ОНО⌞⌞МОЖЕТ⌞⌞БЫТЬ⌞⌞ИЗМЕРЕНО⌞⌞ИНТЕРФЕРОМЕТРИЧЕСКИ⌞⌞С⌞⌞ОЧЕНЬ⌞⌞БОЛЬШОЙ⌞⌞ТОЧНОСТЬЮ⌞⌞*.*⌞⌞ЗЕРКАЛО⌞⌞
ВАКУУМНОЙ⌞⌞КАМЕРЕ⌞⌞С⌞⌞ТЕМ⌞⌞*,*⌞⌞ЧТОБЫ⌞⌞ИСКЛЮЧИТЬ⌞⌞ПОПРАВКУ⌞⌞НА⌞⌞ПОКАЗАТЕЛЬ⌞⌞ПРЕЛОМЛЕНИЯ⌞⌞ВОЗД
ВЫБРПТЬ⌞⌞ВЕЛИЧИНУ⌞⌞СМЕЩЕНИЯ⌞⌞ЗЕРКАЛА⌞⌞ТАК⌞⌞*,*⌞⌞ЧТОБЫ⌞⌞ЧАСТОТЫ⌞⌞ПОВТОРЕНИЙ⌞⌞СВЕТОВЫХ⌞⌞ИМПУЛЬС
И⌞⌞ЭТАЛОННОГО⌞⌞КВАРЦЕВОГО⌞⌞ГЕНЕРАТОРА⌞⌞*,*⌞⌞ПРИ⌞⌞ПОМОЩИ⌞⌞КОТОРОГО⌞⌞ПРОИЗВОДЯТСЯ⌞⌞ИЗМЕРЕНИЯ⌞⌞Ч
⌞⌞ТОЧНОСТЬ⌞⌞ОПРЕДЕЛЕНИЯ⌞⌞ЧАСТОТ⌞⌞БУДЕТ⌞⌞ОСОБЕННО⌞⌞ВЫСОКА⌞⌞*.*⌞⌞$NEW⌞PARAGRAPH$⌞⌞СРАВНЕНИЕ⌞⌞ЧА
```

Fig. 2. Sample section of proof copy prepared on the Harvard Russian Unityper

electronics and mathematics. This file presently contains 22,000 entries representing 10,500 words, or roughly 120,000 distinct inflected forms of these words. The compilation of the dictionary file has been described elsewhere.[4]

An important output of the Automatic Dictionary is a word-by-word translation, a translation in which one or more English correspondents are simply substituted for each Russian word in the running text. Word-by-word translations of scientific and technical texts might be useful to students, to conventional translators, and to technical editors of conventionally translated texts, as aids in their work. The author's experience with the Harvard Automatic Dictionary, although still very limited, indicates that the production of word-by-word translations is a simple and practical matter.

When a translation is made, the original Russian is entered onto magnetic tape with a special Unityper.[5] This Unityper has a standard Russian keyboard in the lower-case, and a standard English keyboard in the upper-case. It records appropriate alphanumeric pulse patterns for both the Russian and English letters, and produces a proof copy in both Russian and English. The proof copy for a sample section of Russian text is shown in Fig. 2. The Russian is simply copied as it appears in the original printed text. The typist brackets punctuation symbols with asterisks. The typist also inserts English comments to describe equations and illustrations, and brackets these with dollar signs. The Unityper tape and the Russian-English dictionary tapes are the inputs to a single Univac computer run, producing the word-by-word translation.

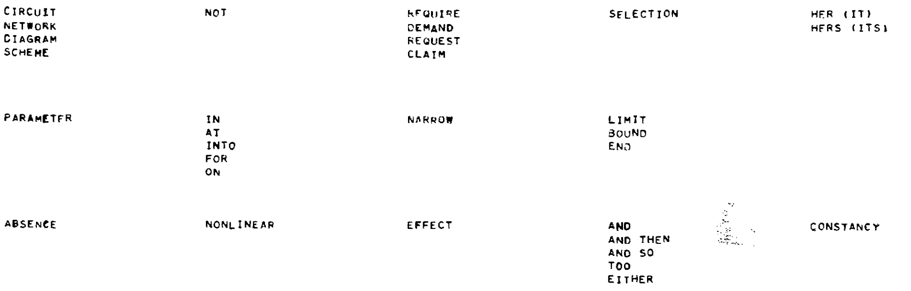
A sample section of a word-by-word translation is shown in Fig. 3. The sample is printed from Univac magnetic tape. The text is read from left to right, in the normal manner. Alternative English correspondents, for the same Russian word, are listed one under another. Provisions exist for printing a maximum of 1,2,3,4, or 5 English correspondents, as desired. Missing Russian words, in the text but not in the dictionary, are simply transliterated into English characters, and are marked with the symbol "###." The sample shown was made with an incomplete dictionary file, and a few words are therefore still transliterated. Punctuation marks appear in their original positions. English comments are marked with the symbol "***." The operating program of the Harvard Automatic Dictionary produces, besides a word-by-word translation, several companion by-products (Fig. 4). For example, a special alphabetic dictionary is produced, containing only words in the particular text.

A word-by-word translation can be converted into a smooth and idiomatic translation by a posteditor, who can work directly on the machine-produced print. A sample page of postedited print is shown in Fig. 5. The illustrated print is produced with the original Russian interlineated with the English, for the benefit of translators knowing some Russian. The posteditor has indicated his choice of meanings, as well as a choice of word-sequence, with the arrows. Besides drawing arrows, a posteditor occasionally supplies meanings missing from the dictionary or better than those originally printed out. A postedited text is eventually transcribed into conventional format by a typist, who copies the words at the heads of arrows.

Experiments are being conducted to determine the value of the dictionary outputs to translators, scientists, and students of technical Russian. Preliminary results are encouraging; several individuals have used the dictionary outputs to translate at considerably accelerated speeds. The outputs seem to offer particular advantages to individuals who are familiar with the technical subject matter of the material being translated, but who know only the rudiments of Russian.

At the time of this writing, over a dozen texts, each 1,000 to 3,000 words in length, have been selected from the Russian technical literature, translated by machine, and postedited. By June of 1959, the number of texts should be approaching 50. New or improved technical terms and meanings, suggested by technical experts who read the dictionary outputs, will be fed back into the dictionary, which is to be periodically updated through a semiautomatic process. The experiments in postediting are extremely important from the long-range viewpoint, since the transformations which are made by the posteditors are the very same transformations which must be eventually automatized.

The main output of an automatic dictionary, insofar as the proposed Trial Translator is concerned, is a by-product called a text-ordered subdictionary. A text-ordered subdictionary is an augmented version of a Russian text. It is made by simply substituting a complete dictionary entry, on magnetic tape, for each Russian word in the original running text. The text-ordered subdictionary produced by the Harvard Automatic Dictionary has a format similar to that shown in Fig. 6, except that the entries are arranged in textual instead of alphabetic order. The entries are separated by heavy horizontal lines. Each entry contains a Russian word (marked α), a complete set of English correspondents (β), and a set of coded information (Υ). The coded information characterizes the grammatical properties of the Russian word in some detail. Information of this nature is needed if smooth and grammatical translations are to be produced automatically. This output combines the original textual information with the available lexical information, and contains all data which might be necessary for an automatic analysis of the text.



```
CIRCUIT            NOT                REQUIRE            SELECTION          HER (IT)
NETWORK                               DEMAND                                HERS (ITS)
DIAGRAM                               REQUEST
SCHEME                                CLAIM


PARAMETER          IN                 NARROW             LIMIT
                   AT                                    BOUND
                   INTO                                  END
                   FOR
                   ON


ABSENCE            NONLINEAR          EFFECT             AND                CONSTANCY
                                                         AND THEN
                                                         AND SO
                                                         TOO
                                                         EITHER
```

Fig. 3. Section of a word-by-word translation

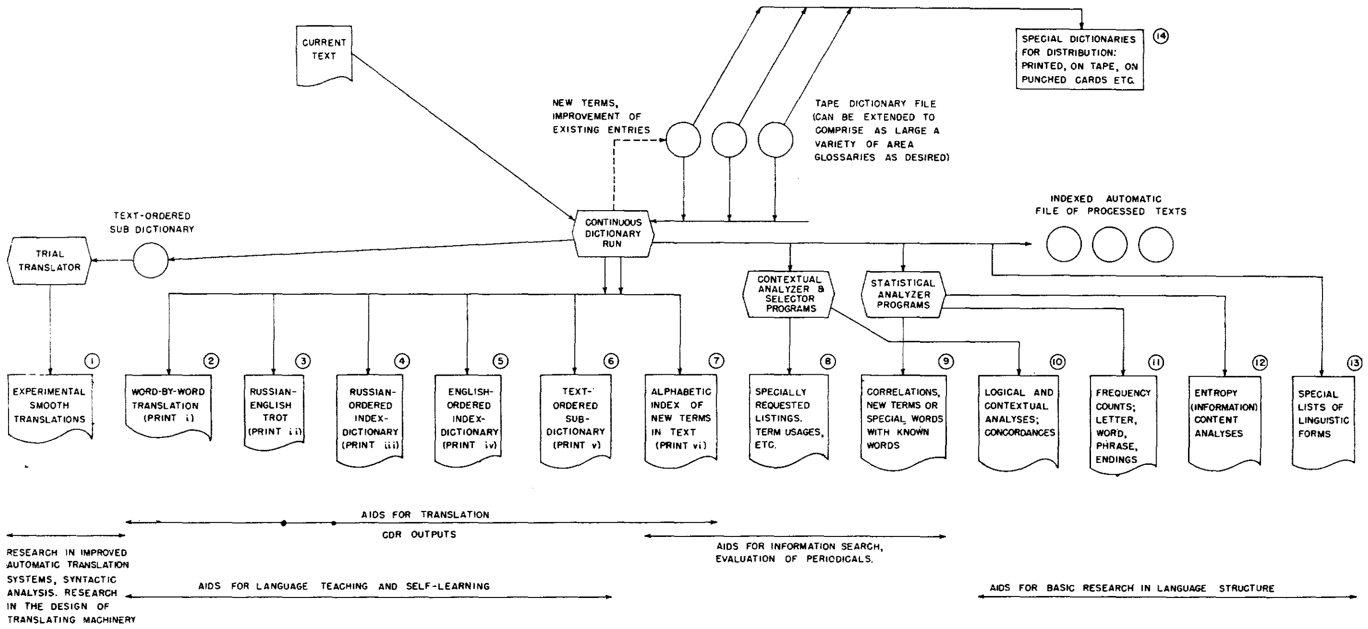

CURRENT TEXT
NEW TERMS, IMPROVEMENT OF EXISTING ENTRIES
TAPE DICTIONARY FILE (CAN BE EXTENDED TO COMPRISE AS LARGE A VARIETY OF AREA GLOSSARIES AS DESIRED)
SPECIAL DICTIONARIES FOR DISTRIBUTION: PRINTED, ON TAPE, ON PUNCHED CARDS ETC.
14
TEXT-ORDERED SUB DICTIONARY
TRIAL TRANSLATOR
CONTINUOUS DICTIONARY RUN
INDEXED AUTOMATIC FILE OF PROCESSED TEXTS
CONTEXTUAL ANALYZER & SELECTOR PROGRAMS
STATISTICAL ANALYZER PROGRAMS
1
EXPERIMENTAL SMOOTH TRANSLATIONS
2
WORD-BY-WORD TRANSLATION (PRINT i)
3
RUSSIAN-ENGLISH TROT (PRINT ii)
4
RUSSIAN-ORDERED INDEX-DICTIONARY (PRINT iii)
5
ENGLISH-ORDERED INDEX-DICTIONARY (PRINT iv)
6
TEXT-ORDERED SUB-DICTIONARY (PRINT v)
7
ALPHABETIC INDEX OF NEW TERMS IN TEXT (PRINT vi)
8
SPECIALLY REQUESTED LISTINGS. TERM USAGES, ETC.
9
CORRELATIONS, NEW TERMS OR SPECIAL WORDS WITH KNOWN WORDS
10
LOGICAL AND CONTEXTUAL ANALYSES; CONCORDANCES
11
FREQUENCY COUNTS; LETTER, WORD, PHRASE, ENDINGS
12
ENTROPY (INFORMATION) CONTENT ANALYSES
13
SPECIAL LISTS OF LINGUISTIC FORMS
AIDS FOR TRANSLATION
CDR OUTPUTS
RESEARCH IN IMPROVED AUTOMATIC TRANSLATION SYSTEMS, SYNTACTIC ANALYSIS. RESEARCH IN THE DESIGN OF TRANSLATING MACHINERY
AIDS FOR INFORMATION SEARCH, EVALUATION OF PERIODICALS.
AIDS FOR LANGUAGE TEACHING AND SELF-LEARNING
AIDS FOR BASIC RESEARCH IN LANGUAGE STRUCTURE

Fig. 4. Outputs of the Harvard Automatic Dictionary

Translation Algorithms

As an example of a translation algorithm, consider a portion of a rule suggested by Koutsoudas and Korfhage.[6] (This specific algorithm is mentioned for illustrative purposes only and the present writer does not assert that it is necessarily valid. It is his contention, in fact, that no translation algorithm can be accepted as truly valid until it has been tested on a large scale with actual language data.)

"Nouns preceded by adjectives:
—If the word preceding the adjective is a noun, and there is no punctuation mark between the first noun and the adjective, then place 'of the' before the adjective."

This rule can be stated in a form analogous to that of a logical implication. Let k be a numeric argument defining the text position of a word on which attention is focused. Let the following propositional variables be defined for all values of k:

$N(k)$ = k is a noun
$A(k)$ = k is an adjective
$\Upsilon(k)$ = a punctuation mark follows k.

When a definite value of k is assigned to a propositional variable, a definite proposition results. For example, $N(26)$ means "word number 26 in the text is a noun." The indicated action can also be abbreviated:

$IN(xxx,k)$ = insert the string "xxx" before the translation of k.

The translation algorithm can then be stated, using the usual connectives of the propositional calculus; "$\sim$" for "not," "$\cdot$" for "and," and "$\vee$" for "or":

$$N(k)\cdot A(k-1)\cdot N(k-2)\cdot \sim\Upsilon(k-2)\rightarrow \text{IN(OF THE, } k-1)$$

This can be abbreviated in the form:

$$P\rightarrow Q$$

where P is a logical function of one or more variables, and Q stands for an action which might be taken in making a translation. The arrow is not a logical implication sign, but stands for an imperative to the translator or translating machine to take the action Q when P is true. Algorithms which can be written in this form will be called basic algorithms. Almost all of the algorithms suggested in the literature of machine translation can be expressed as basic algorithms or as chains of basic algorithms.

The Trial Translator

Many linguists have, in the course of their work, become acquainted with the notation of the propositional calculus. The notation readily lends itself to the use of mnemonic names for logical variables, names such as $N(k)$, $A(k)$, etc. It is simple and well-suited for the formulation of translation algorithms, a fact illustrated by the example. These considerations suggest that the variables and formulas of basic algorithms might serve as the phrases and sentences of a pseudocode language, used as input to an automatic programming system. The system would convert pseudocode instructions phrased in the language of the propositional calculus into computer commands, and execute these commands on Russian texts. This automatic programming system is, of course, the suggested Trial Translator.

Like any other automatic programming system, the Trial Translator might be programmed in a great many different ways.[7] A specific approach is suggested, and is believed to be practical for use on a machine having magnetic tape units, a large internal store, and provision for the handling of alphanumeric characters. Several existing machines satisfy these requirements, among them the International Business Machines Corporation (IBM) *705*, the IBM *709*, and the Univac II, the latter when equipped with the maximum amount of core storage.

The proposed Trial Translator embodies features of both interpretive- and compiler-type systems. It is based upon the selective association of algorithms with dictionary entries. The actual association is effected by the Formula Inserter, a compiler program. The algorithms are later applied to texts by the Specifier-Evaluator-Editor, an interpretive program.

The Association of Algorithms with Dictionary Entries

The Formula Inserter operates upon a Russian-English dictionary file similar to the file of the Harvard Automatic Dictionary. Prior to a computer run, a table of basic algorithms is loaded into an internal memory store. The Formula Inserter selectively inserts coded representations of the translation algorithms into the individual dictionary entries. The algorithms themselves select the entries into which they are inserted. For example, certain algorithms apply uniquely to single Russian words. Among these are algorithms used in the resolution of multiple meaning problems, algorithms used for the interpretation of idioms, and algorithms for the treatment of prepositions. Any one of these algorithms contains a variable stating the presence of a particular word, such as из (k), and is inserted into the entry containing that word. Other translation algorithms apply to broad classes of Russian words, such as those applying to nouns, those applying to verbs, etc. Coded information is assumed to be in each dictionary entry, denoting the part of speech of the Russian word in that entry (noun, verb, preposition, etc.). Many translation algorithms are inserted according to parts of speech: algorithms having $N(k)$ in their formulation apply primarily to nouns, and are inserted into all noun entries, etc. The Formula Inserter is also capable of using other, more complicated criteria for the insertion of algorithms.

The Formula Inserter allows the automatic accumulation and updating of algorithms. It is capable of inserting new algorithms into the entries of a previously prepared dictionary tape, and of deleting old algorithms from such a tape. The output of the Formula Inserter is an augmented dictionary file containing the selected formulas. The augmented dictionary file is prepared only once, and then can be used for any number of texts. Formulas are inserted once and become part of the lexical material in the dictionary. Since the formulas are pseudocoded programs, the Formula Inserter is effectively a compiler-type program. Substantial economies can be realized through the use of this compiler operation, since the program logic needed to associate algorithms with Russian words need be carried through only once.

The second main component of the suggested Trial Translator is an Automatic Dictionary. Russian texts are processed in the usual manner, except that an augmented Russian-English dictionary file containing translation algorithms is used. Both a word-by-word translation and a text-ordered subdictionary are produced for each text. The dictionary search program simply ignores the algorithm codes contained in the dictionary entries, and these codes appear in the entries of the output text-ordered subdictionary.

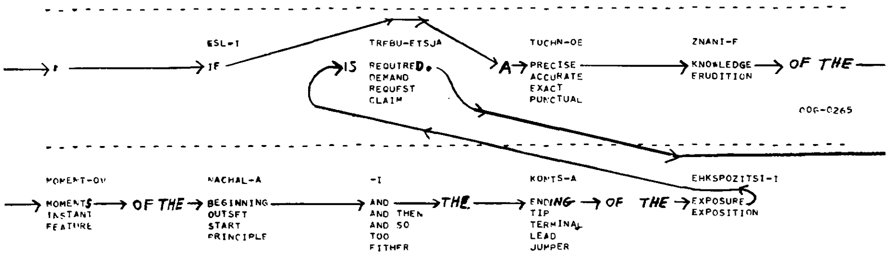


Fig. 5. Postedited section of an interlineated Russian-English word-by-word translation

The Specification of Variables and Evaluation of Formulas

A trial translation is completed by passing the output tape of the automatic dictionary through the Specifier-Evaluator-Editor, the third component in the Trial Translator system. This program operates in an interpretive mode, applying the translation algorithms to the available word-by-word translation. Well-defined contexts, possibly a sentence in length, are processed as separate units. Each context is brought into memory in subdictionary form, is processed and edited, and is recorded as output.

Basic algorithms do not necessarily commute. That is, if A and B are two basic algorithms, the result of applying A and then B to a given context may not be the same as the result of applying B and then A. The order or sequence of algorithm application is, therefore, of great importance. It is necessary, then, to make provisions that enable the linguists who formulate algorithms to exercise control over the order in which these algorithms are applied to a context. These provisions can consist of special sequence-determining algorithms, which determine the order in which other algorithms are executed.

Sequence-determining algorithms can be expressed in the same pseudocode language as other basic algorithms. If they are so expressed, they can be processed by the Specifier-Evaluator-Editor in a like manner. The following procedure for control of the order of algorithm application is envisaged. (The arrangement described is suitable for use during the early stages of experimentation when only a relatively small number of context types will be treated, and when the sequence-determining algorithms will be subject to frequent change. A large number of context types will be treated during the later stages of experimentation, and it will then be necessary to use a more efficient and complicated system of sequence-determining algorithms.) Sequence-determining algorithms are not stored in dictionary entries like other algorithms, but are instead kept in a separate table. This table is available in high speed memory while the Specifier-Evaluator-Editor is operating. The first algorithm executed, whenever a new context is processed, is the first sequence-determining algorithm in the table. The logical portion of a sequence-determining algorithm ascertains whether a given context has a certain structure, for example, noun-verb-preposition-noun. If this is not satisfied by the given context, the next sequence-determining algorithm in the table is evaluated, etc. There is a separate sequence-determining algorithm in the table for each type of context which can be treated. (See preceding parenthetical note.) When the logical portion of such an algorithm is satisfied, the action portion fixes an order in which the algorithms in the entries are to be applied.

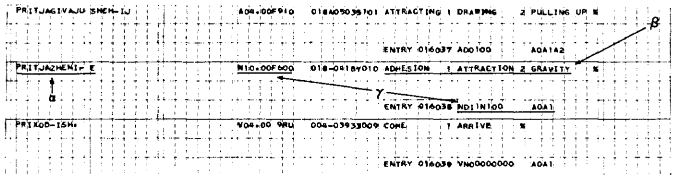
The execution of an algorithm, whether sequence-determining or general, involves the following machine processes:

1. Specification: In a particular context the variables in an algorithm become propositions, which are either true or false. The specification of a variable, in a given context, is the truth value of the associated proposition. This term is used ambiguously, since the process of determining the truth value of a variable is also called the specification of the variable.

2. Evaluation: The truth value of the logical formula portion of an algorithm P, in a given context, is called the evaluation of that formula. The truth value of a formula, of course, depends on the truth values of its component propositions. The term is used ambiguously, since the process of determining the truth value is also called the evaluation of the formula.

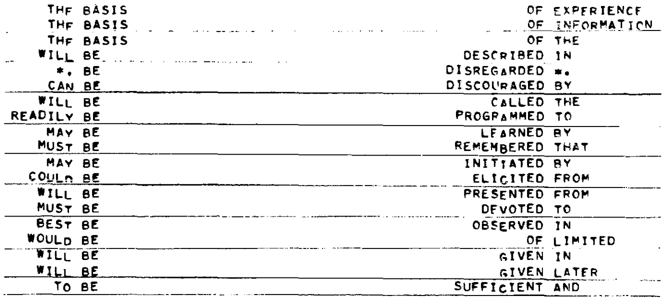
3. Editing: Whenever the evaluation of the logical portion of an algorithm is 1 (true), the corresponding action rule Q is executed. This might be an editorial action, or an action relating to the sequence of algorithm application.

The truth value of a proposition is determined by one of a set of specifier routines. These routines are at the heart of the automatic programming system. The Trial Translator can only handle



```
PRITJAGIVAJU SHCH-IJ          A04.00F910   018A05036101 ATTRACTING 1 DRAWING    2 PULLING UP %
                                                        ENTRY 016037 AD0100      AOA1A2
PRITJAZHENI- E                N10.00F600   018-0418Y010 ADHESION   1 ATTRACTION 2 GRAVITY    %
                                                        ENTRY 016038 ND1IN100    AOA1
PRIXOD-ISH.                   V04.00 9RU   004-03933009 COME       1 ARRIVE     %
                                                        ENTRY 016039 VN0000000   AOA1
```

Fig. 6. Dictionary entries illustrating information available in a text-ordered subdictionary



```
    THF BASIS                       OF EXPERIENCF
    THF BASIS                       OF INFORMATICN
    THF BASIS                       OF THE
    WILL BE                  DESCRIBED IN
      *. BE                DISREGARDED *.
     CAN BE                DISCOURAGED BY
    WILL BE                     CALLED THE
 READILY BE                 PROGRAMMED TO
     MAY BE                    LFARNED BY
    MUST BE                 REMEMBERED THAT
     MAY BE                  INITIATED BY
   COULD BE                  ELICITED FROM
    WILL BE                  PRESENTED FROM
    MUST BE                    DFVOTED TO
    BEST BE                   OBSERVED IN
   WOULD BE                         OF LIMITED
    WILL BE                      GIVEN IN
    WILL BE                      GIVEN LATER
      TO BE                 SUFFICIENT AND
```

Fig. 7. Sample section of a context listing prepared from an English text

variables which can be specified automatically. The breadth of the vocabulary of the input pseudocode language, then, is determined by the degree of flexibility of the specifier routines. Only variables which relate to lexical data, or to the morphology of strings of characters, can be specified automatically. The data, in a given dictionary entry, consist of a Russian word, English correspondents, coded grammatical information, and coded translation algorithms.

Certain variables can be specified by means of the comparison of a known string of characters, given by the variable, with other strings of characters in dictionary entries. Variables of this type will be called string variables. Suppose, for example, that dictionary entries contain coded "part of speech" markers, "N," "A," etc., in a fixed character position, say position 27. In order to specify $N(k+2)$, then, it is sufficient to investigate character position 27 in the second entry following that containing the algorithm under consideration. If the character in this position is "N," the specification is 1 (true), if not the specification is 0. The "part of speech" variables, then, are string variables. Since string variables deal directly with the available morphological and lexical units, it is possible to formulate any admissible basic algorithm in terms of them.

A general-purpose specifier routine is suggested which is capable of specifying any string variable. A string variable can be defined by a set of pseudocode words which this routine is capable of interpreting. These pseudocode words should contain:

1. A key. This is a string of one of more known alphanumeric characters. A string of characters might represent a Russian word, an English correspondent, a grammatical code marker, or the identification code of a basic algorithm.

2. A major co-ordinate. This specifies the text location of the entry in which search is to be made. The major coordinate is a relative co-ordinate, and is zero for the entry containing the algorithm under consideration, -1 for the preceding entry, $+1$ for the following entry, etc. The major co-ordinate might denote a fixed entry, such as $+3$, or a set of entries over which a search must be made. For example, search might be made in all entries following the given entry but preceding the next period. Provisions should exist for both backwards and forward search, with fixed limits or with limits determined by a secondary key.

3. A minor co-ordinate. This specifies the location or locations within an entry which must be checked by the specifier. This can be a number which indicates a specific field in the entry. In the Harvard Automatic Dictionary, for example, Russian stems, English correspondents, and grammatical markers, with minor exceptions, occupy fixed fields. The minor co-ordinate might, instead, denote character positions which are search limits within an entry.

The string in the previous example is "N," the major co-ordinate is $+2$, and the minor co-ordinate is 27.

When a string variable is being specified, the general-purpose routine searches the data positions defined by the major and minor co-ordinates. The strings thus obtained are compared with the key string. When a search is successful, the specification of the variable is 1, otherwise it is 0. A comprehensive routine of this type has been programmed for the Univac I, and successfully checked out.[8] This routine uses four Univac pseudocode words per variable, three of which can contain a key.

Certain variables which will occur commonly in translation algorithms are not, in fact, string variables. For example, consider variables which relate to the interpretations of endings, such as GP(k), standing for genitive plural when k is a noun, or FT(k), standing for future tense when k is a verb. It is not always possible to specify these variables by simply matching strings of characters, and they cannot be handled by the general-purpose specifier routine. A special-purpose specifier routine is therefore suggested, capable of analyzing the ending of a Russian word and determining whether it can represent a given case, number, tense, etc. This routine should be capable of specifying all variables like GP(k) and FT(k).

The operation of this special-purpose specifier routine can be synthesized, using basic algorithms and a general-purpose specifier routine of the first type. That is, each of the variables of the type GP(k) could be replaced by one or more basic algorithms containing only variables of the string type. A separate specifier routine is justified in this case, however, since information about the interpretations of endings will be required very often. Other special specifier routines, handling more complicated variables, can be provided should the need for them arise.

The linguists who formulate translation algorithms should be allowed to communicate these algorithms to the machine in the usual mnemonic notation of the propositional calculus. A linguist should be able to name a variable, for example, by writing "A(k)," "ADJ(k)," "ADJECTIVE (k)," or any other short mnemonic designation. Such mnemonic names need be converted into pseudocode specifier instructions only once, by a programmer, and the correspondence retained in a cross-reference table. The conversion of variables in formulas from mnemonic to full pseudocode form can thereafter be done automatically, as part of the compiler operation of the Formula Inserter.

The evaluation of a logical formula, P, is determined by the specifications of the variables contained in the formula. There are several well-known methods for evaluating logical formulas, any one of which can readily be programmed.[7] If logical formulas are stated in disjunctive normal form, a particularly simple evaluation process can be used. A disjunctive normal form is a sum ($\vee$) of products ($\cdot$) of variables, in which only single variables can be negated ($\backsim$). An evaluator program can consider the products individually and in turn, proceeding from left to right. The specifications of the individual variables are examined, the specification of a negated variable always being reversed. If all of the specifications in a product are 1, the evaluation of the formula is 1 (true), consideration of other products being unnecessary. Whenever a variable with zero specification is encountered, the next product is examined. If all products contain variables with specification 0, the evaluation of the formula itself is 0 (false).

The evaluation of a formula in normal disjunctive form may require the specification of only a few of the many variables in the formula. For example, suppose that the leftmost product in a formula consists of a single variable with specification 1. The evaluation of the formula is immediately seen to be 1, and the specifications of the remaining variables are not needed. The evaluation process, in this case, terminates after a single step. This possibility of early termination suggests that variables be specified only when needed by the evaluator, and that the specifier programs should function as subroutines under control of the evaluator program. The conversion of logical formulas to disjunctive normal form can be done externally, by the linguist-logicians who formulate the algorithms, or internally, as part of the Formula Inserter operation.

A logical evaluator routine, of the type just described, has been checked out on the Univac I.[8] This simple program requires less than a hundred lines of Univac coding. It is designed for use with the previously mentioned general-purpose specifier routine.

Algorithm Actions

When the evaluation of a formula is 1, the indicated action, *Q*, is taken. Each admissible action is represented in the machine by a small subroutine. The actions, like variables, can be named symbolically by the individuals who formulate algorithms. Typical actions might be:

1. Define an order of algorithm execution, for the algorithms contained in the entries of a given context or subcontext.

2. Inflect a given English correspondent into the plural or possessive.

3. Delete the translation of a particular word.

4. Insert an article, such as "the" or "a," before the translation of a given word.

5. Select a particular English correspondent out of the set of those given in an entry.

6. Permute the word order of the translations of two given words.

7. Modify a certain string of characters in a given position, so that the specification of a certain variable is changed.

8. Interrupt the normal sequence of algorithm execution. Take the next algorithm from position *x* in entry *y*.

9. Exit to a fixed routine which edits the output into a format suitable for printing. (This action is taken whenever the processing of a context is completed.) Bring the next context into memory, and transfer control to the first sequence-determining algorithm.

The actions might involve more complicated program logic, controlling the sequence of algorithm application. It might prove economical to construct another interpretive routine, somewhat in the spirit of the logical specifier, which can perform, under control of pseudocode instructions, transformations on arbitrary strings of characters. It should be possible, in the course of experimenting with the Trial Translator, rapidly to build up a library of action subroutines, adequate for all necessary transformations.

Conclusions

The output of the Trial Translator is examined by expert translators, psychologists, and linguists, who determine how well the trial algorithms actually worked. The linguists suggest new algorithms or algorithm modifications. These are coded in mnemonic form, using the language of the propositional calculus, and are fed back into the Formula Inserter. The same human-machine-human cycle is repeated, until a satisfactory set of algorithms is obtained. A large body of Russian text must be used, of course, in order to obtain meaningful results through this feedback process. The linguists are aided, in their choice of algorithms, by the various byproducts of the automatic dictionary operation (Fig. 4). Lists of word contexts, for example, can be readily prepared in a variety of formats.[9] One such format is illustrated, with English text, in Fig. 7.

Research is underway towards automatizing the algorithm finding process. There is reason to believe that certain translation algorithms can be discovered automatically, through the machine comparison of Russian texts and polished professional translations of these texts.

Many successive layers of refinement will exist between the raw word-by-word translations of an automatic dictionary and the smooth translations of an ultimate automatic translator. The first experiments will be modest, and do little more than inflect English correspondents into the plural, etc. Several years will probably be required for the development of a reasonably complete set of fail-safe translation algorithms, adequate for technical Russian.

An important role of a Trial Translator will be the interim mass production of partially improved translations, for actual consumption. These translations will, at every stage, reflect the quality of the best set of translation algorithms then available, without the necessity for expensive reprogramming. It is hoped that these interim translations, like the presently available word-by-word translations, will serve as valuable aids to professional translators, to students of technical Russian, and to scientists interested in the Russian technical literature.

References

1. A great many specific translation algorithms are suggested, for example, in various *Seminar Work Papers (mimeographed)*, Institute of Language and Linguistics, Georgetown University, Washington, D. C.

2. A Study for the Design of an Automatic Dictionary, **A. G. Oettinger.** *Doctoral Thesis*, Harvard University, Cambridge, Mass., 1954.

3. Programming an Automatic Dictionary, **V. E. Giuliano.** *Design and Operation of Digital Calculating Machinery*, Report No. AF-49, Sec. I, Harvard Computation Laboratory, 1957. Also, Papers, *Seminar on Mathematical Linguistics*, vol. III, Harvard University, 1957.

4. Linguistic and Machine Methods for Ccmpiling and Updating the Harvard Automatic Dictionary, **A. G. Oettinger, W. Foust, V. E. Giuliano, K. Magassy, L. Matejka.** *International Conference on Scientific Information*, Washington, D. C., Nov. 1958.

5. An Input Device for the Harvard Automatic Dictionary, **A. G. Oettinger.** *Mechanical Translation*, Massachusetts Institute of Technology, Cambridge, Mass., vol. 5, no. 1, 1958, pp. 2–7.

6. Mechanical Translation and the Problem of Multiple Meaning, **A. Koutsoudas, R. Korfhage.** *Mechanical Translation*, Massachusetts Institute of Technology, Cambridge, Mass., vol. 3, no. 2, 1956.

7. On the Problems of Automatizing the Programming of Translation from One Language to Another (in Russian), **S. N. Razumovskij.** *Doklady Adakimii Nauk USSR*, Moscow, USSR, vol. 113, no. 4, 1957, pp. 760–61.

8. The Calculus on Propositions in Automatic Translation, **J. S. Berson,** *Seminar on Mathematical Linguistics*, Harvard University, Vol. IV, 1958.

9. A Method of Contextual Analysis Applicable to Linguistic Research, **P. E. Jones.** *Ibid.*

Discussion

Allen Walls (Westinghouse Electric Corporation): Does the computer recognize case endings, tenses, etc.?

Mr. Giuliano: Yes. Programs are presently being checked out for this purpose, and variables relating to case endings will be dealt with by the proposed Trial Translator system.

V. Schwab (Remington Rand Univac): To determine English translation, is the Russian word compared letter by letter to a dictionary, or is the word broken down into a root word, with affixes, prefixes, suffixes, and others?

Mr. Giuliano: At present, because of storage considerations, we are using a stem dictionary in which only the stems are stored; however, there is nothing fundamental about this. It just happens to be a question of the large number of bits needed to store a whole dictionary and the limitations of existing storage devices.

W. W. MacWilliams (Bell Telephone Laboratories): Have you had experience with the Trial Translator and, if so, would you summarize it briefly?

Mr. Giuliano: So far, our experience has been with dictionary compilation and producing word-by-word translations. We have processed and postedited about 30 texts. The string specifier and evaluator programs are just now being put into operation. These programs have been written and tested, but they have not been incorporated into a complete Trial Translator system.

G. R. Fiannaca (United States Air Force Intelligence Laboratory, Rome Air Development Center): How do you translate idioms of which a word-for-word translation does not necessarily make much sense, but which contains words which may, at times require a word-for-word translation?

The second part of the question is: Many English translations are sometimes possible for a given Russian word. Some alternatives are not sensible with a given text. How do you deal with the multiple meaning problem?

Mr. Giuliano: Both of these problems relate to advanced phases of research. I can, at best, only indicate how one might approach these problems, since we are just now beginning to tackle them. As far as idioms are concerned, one may store an idiom glossary and use this glossary when certain algorithms show that words translated separately do not make sense or do not properly fit together. It is probable that the multiple meaning problem will have to be approached through the use of an automatic thesaurus; that is, by a dictionary of like meanings. This is about all that can be said at present.

H. Charles Kerpelman (Westinghouse Electric Corporation): What work is being done on "teaching" this machine itself to do the modification and improvements of the translation which is now being done by linguists, psychologists, logicians, etc?

Mr. Giuliano: I am prejudiced, but I think that this is a very good question. I hope soon to publish a paper on the design of a machine system, which I call "Formula Finder," that will automatically compare Russian texts with the postedited word-by-word translations for the same texts. By comparing the two, the system will determine algorithms needed to carry one into the other. The system will only work when given proper clues by a linguist. Outside of this, I know of no work in this area.

J. H. Milsum (National Research Council of Canada): Have you any idea of the expected speed of translation? Will there not be some difficulty in deciding when not to add some more algorithms? Presumably, there will be some point when increased program time is not justified by more accurate translations; in particular, later algorithms will probably involve more work for less effect, I presume.

Mr. Giuliano: I quite agree with your last statement. Our present production rate with the experimental setup is 1,000 words an hour. We feel this can be readily expanded to something like a 100,000 words an hour by reprogramming on a more powerful computer. I must stress the fact that our work is of an experimental nature; it is not production work. I suspect that advancing computer technology will make machine translation more and more economically feasible as time goes on.

E. W. Cannon (National Bureau of Standards): How many bits are required for: 1. the average entry in the dictionary file; 2. a typical augmented file entry?

Will the trial Translator replace the human posteditor now in use?

Mr. Giuliano: The present dictionary contains 2,160 bits per entry. That is, a half of aUnivac data block. Augmented file entries will be of the same size. No efforts have been made to try to achieve storage economy. Our present system is purely experimental.

We hope that the Trial Translator will in time replace the human posteditor. This date, however, is probably some distance in the future.

Paul Benetean (Burroughs Corporation): When do you think you will have a complete system utilizing some of these rules?

What speed do you expect in, say, words per minute?

Mr. Giuliano: I believe that if our research continues at the present rate, it will be possible to have an experimental Trial Translator working, perhaps inefficiently, within 6 months or so.

As far as words per minute, it will probably operate down in the 1,000-words-an-hour range, but at any time it could be reprogrammed to operate as a production system on a faster machine at 10 to 100 times that speed.

DYANA: Dynamics Analyzer-Programmer. Part I: Description and Application

T. J. THEODOROFF

THE PROCEDURE for writing down the equations of motion of certain classes of dynamic systems is explicit, particularly those to which Kirchoff's laws and/or D'Alembert's principle can be applied. The Dynamics Analyzer-Programmer (DYANA) is a program developed for the International Business Machines Corporation (IBM) 704 EDPM which utilizes such a procedure, and from a physical description of a system develops a mathematical model and a resulting program. The objectives in setting up DYANA were:

1. To permit an easy description of dynamic systems to a computer.

2. To provide an automatic analysis by a computer to establish the mathematical model of the behavior of the system.

3. To prepare a complete computer program to numerically solve the mathematical model.

The computer program produced by DYANA is a complete FORTRAN program punched out on cards and ready to be run with the requisite set of numerical data. In addition to the program deck of punched cards, a printed listing of the resultant FORTRAN program is produced along with an input data plan, or map, describing the input data that are needed and the format in which it must be punched on input data cards.

Part I of this paper is concerned with the types of physical systems that are handled by DYANA, how these systems are characterized or expressed, and what the FORTRAN solution of these systems looks like. The internal structure and functioning of DYANA will be given in the second part of this paper (Structure and Function).

T. J. Theodoroff is with General Motors Corporation, Detroit, Mich.

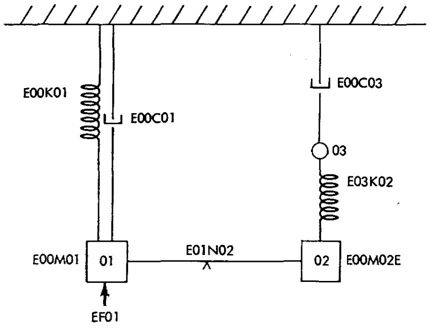


Fig. 1. Simple mechanical system

Types of Systems Which Can Be Studied

The user of DYANA will be able to specify any one of three different kinds of solution for his particular problem. He may ask for a program to compute the natural frequencies and associated modes of vibration of a system. He may ask for a program to compute the frequency response of a system. Finally, he may ask for a program which will give him a time-sensitive description of all displacements and velocities in a system. In this latter case, nonlinear elements may be described in the system as well as any arbitrary excitations at any co-ordinate points. In all three cases branched systems may be described, examples of which would be trains of gears in rotational systems, interconnected levers in translational systems, or transformers in electrical systems. Applications of this program may be made to any system that may be characterized by a single degree of freedom at each co-ordinate point. Some examples might be in thermal processes, hydraulic systems, vibrations in a continuous media which may be approximated by distinct elements occupying discrete positions, as well as electrical and mechanical systems.

Description of Physical Systems

The description of a physical system is given by a convenient notation indicating the elements of the system and their points of connection. The elements are of four distinct types. In mechanical systems they are: 1. masses, 2. viscous dampers, 3. springs, and 4. gears or levers for rotational or translational systems, respectively. All of these elements, except for the masses, have two terminal points of connection. The masses, of course, form a single point of connection. If each point of connection is given a distinct label, then each element can be described in terms of the kind of element it is and what its points of connection are. In DYANA, the labels given to the points of connection are any of the 2-digit numbers 01 to 99, while the 2-digit number 00 is reserved for a fixed terminal point or ground. In this manner DYANA recognizes the symbol *E14K23* as a spring element (from the letter *K*) connected between two points labeled *14* and *23*. The symbol *E14C23* is a damping element connected between the same two points, while *E23C00* is another damping element connected between point *23* and ground. A mass at point *14* is denoted *E00M14*. A gear or level between points *14* and *02* is represented by *E14N02*. Finally, if there are any applied forces in the system, a symbol is given specifying the point at which the force is applied; e.g., *EF12* indicates an applied force at point *12*.

Fig. 1 shows a simple system in which the terminal points have been labeled arbitrarily. Each element in the system, consequently, has a distinct notation as shown. A simple listing of these elements is sufficient to describe the physical system to DYANA.

Input to DYANA

This section will now describe how the complete information pertaining to any specific problem is prepared for DYANA. The system description previously presented is only one of several classes of information that can be given. Suppose that in the simple system of Fig. 1 the spring rate *E03K02*, the viscous damper *E00C03*, and the applied force *EF02* are each to be calculated by some algebraic expression. These algebraic expressions can also be presented as part of the input information, and are in fact prepared in the FORTRAN language itself. Information in regards to the output that is to be printed out may also be given. All classes of information presented to the DYANA program are always preceded by a classification card. Subsequent cards up to the next classification card contain the actual information written in either the DYANA or FORTRAN languages. To illustrate this, consider Fig. 2 which shows the complete input which might be prepared for the simple system. Each line in this figure represents the information that is punched on a standard 80-column IBM card. The classification cards are distinguished by the *X* punched in Column 1.

Card 1 is a classification card which informs the computer now under the control of DYANA that all subsequent cards up to the next classification card contain a description of the ideal model.

Cards 2 and 3 contain the description of the ideal model. The notation appearing on these cards is entirely independent

```
                         INPUT WITH DIAGNOSTIC COMMENTS FLAGGED BY AN *

X     SYSTEM DESCRIPTION                                                  CARD   1
      E00M01,E00M02,E00K01,E00C01,E00C03                                  CARD   2
      E03K02,E01N02,EF01                                                  CARD   3
X     PRE-COMPUTATION                                                     CARD   4
      E03K02 = 2.4 + 0.6*E00K01                                           CARD   5
X     DAMPING RATE, E00C03(X03)                                           CARD   6
      E00C03 = A*X03 + B                                                  CARD   7
X     FORCE, EF01                                                         CARD   8
      EF01 = F*SINF(W*TIME)                                               CARD   9
X     INPUT VARIABLES                                                     CARD  10
      A,B,F,W                                                             CARD  11
X     PRINT PRECOMPUTATION ANSWERS                                        CARD  12
      E00M01,E00M02,E00K01,E00C01,E03K02,E01N02,A,B,F,W                   CARD  13
X     PRINT TIME DEPENDENT ANSWERS                                        CARD  14
      TIME,EF01,X01,X02,X03,DX01,DX02                                     CARD  15
X     TRANSLATIONAL                                                       CARD  16
X     TRANSIENT                                                           CARD  17
X     END                                                                 CARD  18
```

Fig. 2. DYANA problem statement for systems shown in Figs. 1 and 6

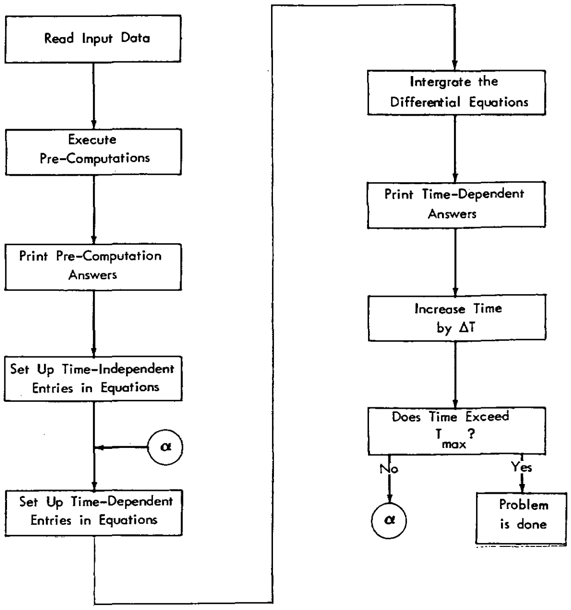
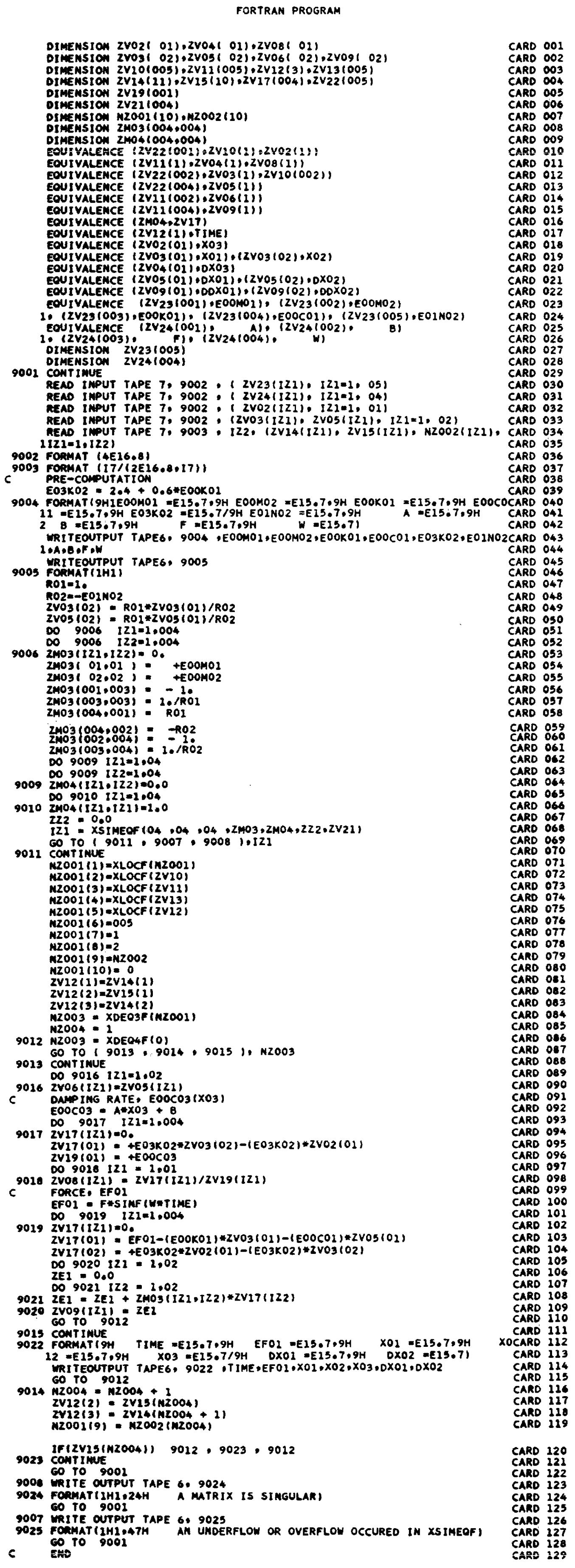

Read Input Data
Execute Pre-Computations
Print Pre-Computation Answers
Set Up Time-Independent Entries in Equations
α
Set Up Time-Dependent Entries in Equations
Intergrate the Differential Equations
Print Time-Dependent Answers
Increase Time by ΔT
Does Time Exceed T max ?
No
Yes
α
Problem is done

Fig. 3 (right). Flow diagram of DYANA-produced program



```
                             FORTRAN PROGRAM

      DIMENSION ZV02( 01),ZV04( 01),ZV08( 01)                           CARD 001
      DIMENSION ZV03( 02),ZV05( 02),ZV06( 02),ZV09( 02)                 CARD 002
      DIMENSION ZV10(005),ZV11(005),ZV12(3),ZV13(005)                   CARD 003
      DIMENSION ZV14(11),ZV15(10),ZV17(004),ZV22(005)                   CARD 004
      DIMENSION ZV19(001)                                               CARD 005
      DIMENSION ZV21(004)                                               CARD 006
      DIMENSION NZ001(10),NZ002(10)                                     CARD 007
      DIMENSION ZM03(004,004)                                           CARD 008
      DIMENSION ZM04(004,004)                                           CARD 009
      EQUIVALENCE (ZV22(001),ZV10(1),ZV02(1))                           CARD 010
      EQUIVALENCE (ZV11(1),ZV04(1),ZV08(1))                             CARD 011
      EQUIVALENCE (ZV22(002),ZV03(1),ZV10(002))                         CARD 012
      EQUIVALENCE (ZV22(004),ZV05(1))                                   CARD 013
      EQUIVALENCE (ZV11(002),ZV06(1))                                   CARD 014
      EQUIVALENCE (ZV11(004),ZV09(1))                                   CARD 015
      EQUIVALENCE (ZM04,ZV17)                                           CARD 016
      EQUIVALENCE (ZV12(1),TIME)                                        CARD 017
      EQUIVALENCE (ZV02(01),X03)                                        CARD 018
      EQUIVALENCE (ZV03(01),X01),(ZV03(02),X02)                         CARD 019
      EQUIVALENCE (ZV04(01),DX03)                                       CARD 020
      EQUIVALENCE (ZV05(01),DX01),(ZV05(02),DX02)                       CARD 021
      EQUIVALENCE (ZV09(01),DDX01),(ZV09(02),DDX02)                     CARD 022
      EQUIVALENCE  (ZV23(001),E00M01), (ZV23(002),E00M02)               CARD 023
     1, (ZV23(003),E00K01), (ZV23(004),E00C01), (ZV23(005),E01N02)      CARD 024
      EQUIVALENCE  (ZV24(001),     A), (ZV24(002),     B)               CARD 025
     1, (ZV24(003),     F), (ZV24(004),     W)                          CARD 026
      DIMENSION  ZV23(005)                                              CARD 027
      DIMENSION  ZV24(004)                                              CARD 028
 9001 CONTINUE                                                          CARD 029
      READ INPUT TAPE 7, 9002 , ( ZV23(IZ1), IZ1=1, 05)                 CARD 030
      READ INPUT TAPE 7, 9002 , ( ZV24(IZ1), IZ1=1, 04)                 CARD 031
      READ INPUT TAPE 7, 9002 , ( ZV02(IZ1), IZ1=1, 01)                 CARD 032
      READ INPUT TAPE 7, 9002 , (ZV03(IZ1), ZV05(IZ1), IZ1=1, 02)       CARD 033
      READ INPUT TAPE 7, 9003 , IZ2, (ZV14(IZ1), ZV15(IZ1), NZ002(IZ1), CARD 034
     1IZ1=1,IZ2)                                                        CARD 035
 9002 FORMAT (4E16.8)                                                   CARD 036
 9003 FORMAT (I7/(2E16.8,I7))                                           CARD 037
C     PRE-COMPUTATION                                                   CARD 038
      E03K02 = 2.4 + 0.6*E00K01                                         CARD 039
 9004 FORMAT(9H1E00M01 =E15.7,9H E00M02 =E15.7,9H E00K01 =E15.7,9H E00C0CARD 040
     11 =E15.7,9H E03K02 =E15.7/9H E01N02 =E15.7,9H      A =E15.7,9H    CARD 041
     2  B =E15.7,9H      F =E15.7,9H      W =E15.7)                     CARD 042
      WRITEOUTPUT TAPE6, 9004 ,E00M01,E00M02,E00K01,E00C01,E03K02,E01N02CARD 043
     1,A,B,F,W                                                          CARD 044
      WRITEOUTPUT TAPE6, 9005                                           CARD 045
 9005 FORMAT(1H1)                                                       CARD 046
      R01=1.                                                            CARD 047
      R02=-E01N02                                                       CARD 048
      ZV03(02) = R01*ZV03(01)/R02                                       CARD 049
      ZV05(02) = R01*ZV05(01)/R02                                       CARD 050
      DO  9006  IZ1=1,004                                               CARD 051
      DO  9006  IZ2=1,004                                               CARD 052
 9006 ZM03(IZ1,IZ2)= 0.                                                 CARD 053
      ZM03( 01,01 ) =   +E00M01                                         CARD 054
      ZM03( 02,02 ) =   +E00M02                                         CARD 055
      ZM03(001,003) =  - 1.                                             CARD 056
      ZM03(003,003) = 1./R01                                            CARD 057
      ZM03(004,001) =  R01                                              CARD 058
      ZM03(004,002) =  -R02                                             CARD 059
      ZM03(002,004) =  - 1.                                             CARD 060
      ZM03(003,004) = 1./R02                                            CARD 061
      DO 9009 IZ1=1,04                                                  CARD 062
      DO 9009 IZ2=1,04                                                  CARD 063
 9009 ZM04(IZ1,IZ2)=0.0                                                 CARD 064
      DO 9010 IZ1=1,04                                                  CARD 065
 9010 ZM04(IZ1,IZ1)=1.0                                                 CARD 066
      ZZ2 = 0.0                                                         CARD 067
      IZ1 = XSIMEQF(04 ,04 ,04 ,ZM03,ZM04,ZZ2,ZV21)                     CARD 068
      GO TO ( 9011 , 9007 , 9008 ),IZ1                                  CARD 069
 9011 CONTINUE                                                          CARD 070
      NZ001(1)=XLOCF(NZ001)                                             CARD 071
      NZ001(2)=XLOCF(ZV10)                                              CARD 072
      NZ001(3)=XLOCF(ZV11)                                              CARD 073
      NZ001(4)=XLOCF(ZV13)                                              CARD 074
      NZ001(5)=XLOCF(ZV12)                                              CARD 075
      NZ001(6)=005                                                      CARD 076
      NZ001(7)=1                                                        CARD 077
      NZ001(8)=2                                                        CARD 078
      NZ001(9)=NZ002                                                    CARD 079
      NZ001(10)= 0                                                      CARD 080
      ZV12(1)=ZV14(1)                                                   CARD 081
      ZV12(2)=ZV15(1)                                                   CARD 082
      ZV12(3)=ZV14(2)                                                   CARD 083
      NZ003 = XDEQ3F(NZ001)                                             CARD 084
      NZ004 = 1                                                         CARD 085
 9012 NZ003 = XDEQ4F(0)                                                 CARD 086
      GO TO ( 9013 , 9014 , 9015 ), NZ003                               CARD 087
 9013 CONTINUE                                                          CARD 088
      DO 9016 IZ1=1,02                                                  CARD 089
 9016 ZV06(IZ1)=ZV05(IZ1)                                               CARD 090
C     DAMPING RATE, E00C03(X03)                                         CARD 091
      E00C03 = A*X03 + B                                                CARD 092
      DO  9017  IZ1=1,004                                               CARD 093
 9017 ZV17(IZ1)=0.                                                      CARD 094
      ZV17(01) = +E03K02*ZV03(02)-(E03K02)*ZV02(01)                     CARD 095
      ZV19(01) = +E00C03                                                CARD 096
      DO 9018 IZ1 = 1,01                                                CARD 097
 9018 ZV08(IZ1) = ZV17(IZ1)/ZV19(IZ1)                                   CARD 098
C     FORCE, EF01                                                       CARD 099
      EF01 = F*SINF(W*TIME)                                             CARD 100
      DO  9019  IZ1=1,004                                               CARD 101
 9019 ZV17(IZ1)=0.                                                      CARD 102
      ZV17(01) = EF01-(E00K01)*ZV03(01)-(E00C01)*ZV05(01)               CARD 103
      ZV17(02) = +E03K02*ZV02(01)-(E03K02)*ZV03(02)                     CARD 104
      DO 9020 IZ1 = 1,02                                                CARD 105
      ZE1 = 0.0                                                         CARD 106
      DO 9021 IZ2 = 1,02                                                CARD 107
 9021 ZE1 = ZE1 + ZM03(IZ1,IZ2)*ZV17(IZ2)                               CARD 108
 9020 ZV09(IZ1) = ZE1                                                   CARD 109
      GO TO  9012                                                       CARD 110
 9015 CONTINUE                                                          CARD 111
 9022 FORMAT(9H   TIME =E15.7,9H   EF01 =E15.7,9H    X01 =E15.7,9H    X0CARD 112
     12 =E15.7,9H    X03 =E15.7/9H   DX01 =E15.7,9H   DX02 =E15.7)      CARD 113
      WRITEOUTPUT TAPE6, 9022 ,TIME,EF01,X01,X02,X03,DX01,DX02          CARD 114
      GO TO  9012                                                       CARD 115
 9014 NZ004 = NZ004 + 1                                                 CARD 116
      ZV12(2) = ZV15(NZ004)                                             CARD 117
      ZV12(3) = ZV14(NZ004 + 1)                                         CARD 118
      NZ001(9) = NZ002(NZ004)                                           CARD 119

      IF(ZV15(NZ004))  9012 , 9023 , 9012                               CARD 120
 9023 CONTINUE                                                          CARD 121
      GO TO  9001                                                       CARD 122
 9008 WRITE OUTPUT TAPE 6, 9024                                         CARD 123
 9024 FORMAT(1H1,24H    A MATRIX IS SINGULAR)                           CARD 124
      GO TO  9001                                                       CARD 125
 9007 WRITE OUTPUT TAPE 6, 9025                                         CARD 126
 9025 FORMAT(1H1,47H    AN UNDERFLOW OR OVERFLOW OCCURED IN XSIMEQF)    CARD 127
      GO TO  9001                                                       CARD 128
C     END                                                               CARD 129
```

Fig. 4. FORTRAN program prepared by DYANA

of the order in which it is presented.

Card 4 is a classification card signaling the computer that subsequent cards contain algebraic expressions in FORTRAN notation which are to be evaluated prior to the solution of the system itself. In this example a spring element is given as a function of some other spring element, as depicted on Card 5.

Card 6 is a classification card indicating that the instantaneous value of damper *E00C03* is a function of time, and so the expression for it, given on the subsequent cards, should be placed in that part of the program where it is evaluated at each time step of the solution. This card also shows that the time-dependent argument in the expression for this element is *X03* representing the instantaneous displacement of point *03* in the system.

Card 7 gives the actual expression for the evaluation of *E00C3*, once again in FORTRAN notation.

Card 8 is a classification card signaling the computer that subsequent cards express in FORTRAN notation the applied force at point *01*.

Card 9 represents the expression for the applied force *EF01*.

Card 10 is a classification card signaling to the computer that the symbols appearing on the subsequent cards are symbols incorporated by the user in some of his FORTRAN coded expressions for which he will provide numerical input data.

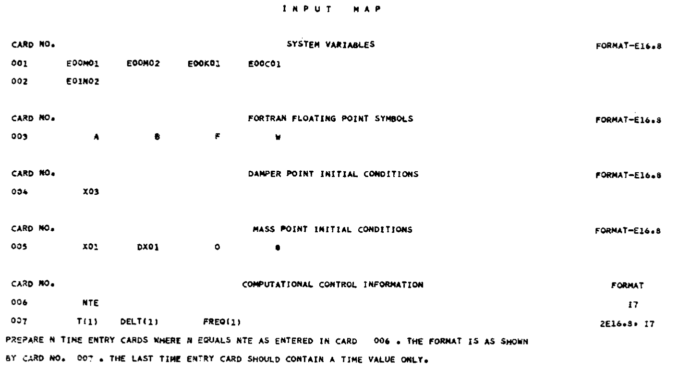
Card 11 is a list of these symbols and is independent of order.

Card 12 is a classification card informing the computer to print out the values represented by the symbols appearing on the subsequent cards prior to the solution of the system equations.

Card 13 is a list of these symbols and is order independent. Some of these values may be input data while others, like *E03-K02*, are results evaluated at the precomputation phase.

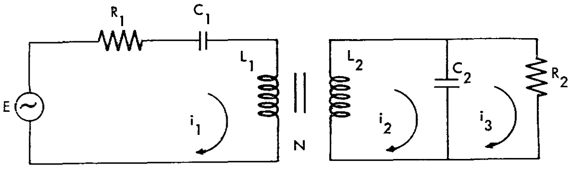
Card 14 is a classification card informing the computer that subsequent cards contain time-dependent variables whose quantities the computer will single out for printing during the computational phase of the problem. They will be printed at regular intervals of the independent variable time.

Card 15 is a list of these variables and is also order independent. In this example, the following items are to be printed out: the independent variable "TIME"; the applied force on point *01* denoted by *EF01*; the displacements at points *01*, *02*, and *03* indicated by the DYANA symbols *X01*, *X02*, and *X03*, respectively; and the first derivatives, or velocities, at points *01* and *02* indicated



```
                                      I N P U T   M A P

CARD NO.                                  SYSTEM VARIABLES                          FORMAT-E16.8
001      E00M01    E00M02    E00K01    E00C01
002      E01N02

CARD NO.                              FORTRAN FLOATING POINT SYMBOLS                FORMAT-E16.8
003           A         B         F         W

CARD NO.                              DAMPER POINT INITIAL CONDITIONS               FORMAT-E16.8
004         X03

CARD NO.                               MASS POINT INITIAL CONDITIONS                FORMAT-E16.8
005         X01       DX01        0         B

CARD NO.                             COMPUTATIONAL CONTROL INFORMATION                 FORMAT
006         NTE                                                                          I7
007        T(1)    DELT(1)       FREQ(1)                                             2E16.8, I7
PREPARE N TIME ENTRY CARDS WHERE N EQUALS NTE AS ENTERED IN CARD   006 . THE FORMAT IS AS SHOWN
BY CARD NO.  007 . THE LAST TIME ENTRY CARD SHOULD CONTAIN A TIME VALUE ONLY.
```

Fig. 5. Input map generated by DYANA



L_1 = E00M01

R_1 = E00C01

C_1 = 1/E00K01

N = E01N02

i_1 = DX01

i_1 = DX03

L_2 = E00M02

R_2 = E00C03

C_2 = 1/E02K03

E = EF01

i_2 = DX02

Fig. 6. Electrical analog of Fig. 1

by the DYANA symbols *DX01* and *DX02*, respectively. The time dependent damping rate *E00C03* may also have been included here if so desired.

Card 16 is a classification card indicating that the system under study is a translational as opposed to a rotational system. The effect of this is in the definition of the lever or gear ratios if any exist.

Card 17 is a classification card which sets up the sequencing within DYANA to produce a program giving a complete time-sensitive solution of the differential equations. If it was desired to produce a program giving a frequency response solution, then the phrase "frequency response" would have been entered on this card. A program to evaluate the natural frequencies of a system would be produced with the entry of the term "natural frequency" on this card.

Card 18 is a classification card which flags the end of the list and signals the computer to begin the processing of the information it has just read in.

Basically, the preparation of these cards is all the effort that the user has to expend to study the behavior of the model shown in Fig. 1.

Output Generated by DYANA

As was stated at the beginning of this paper, the output from DYANA is a complete FORTRAN program punched out on cards and ready to be run with the requisite set of numerical data. Fig. 3 depicts a general flow diagram of the DYANA-produced program. Fig. 4 is a printed listing of the generated program in which each line represents the information punched out on a standard 80-column IBM card. It will be noted that the precoded expressions for one of the spring rates, the nonlinear damper, and the applied force, together with their respective classification cards, have been inserted in their appropriate positions of the final generated program.

In addition to the final program, an input data plan or map is produced which describes the input data that are required and the format in which it must be punched on the input data cards.

Fig. 5 shows the input map that was produced for the simple example. On the basis of the analysis performed by DYANA on the problem description, those elements and variables which require numerical values are listed. The analysis also determines the initial conditions that are needed for the solution of the differential equations. The example shows that the initial displacement at point *03* and the initial velocity and displacement at point *01* are required. The cards which come under the heading "Computational Control Information" list the starting and end values of the independent variable, the step size to be used in the solution, and the frequency of the step size at which the answers are to be printed. The format of all numbers entered as input data is described for each card by the extreme right-hand side of the "input map." Numbers appear on these cards in either the *E*- or *I*-type format provided by the FORTRAN source language.

Electrical Systems

The physical description of electrical networks is readily made using the same notation that was applied to mechanical systems. This of course is possible because of the well-known analogy that exists between electrical and mechanical systems. One method of studying electrical systems is to employ the concept of circulating currents or loop currents. According to this concept, the network is made up of a number of closed loops in which the currents in each loop are the dependent variables and correspond to the velocities at the co-ordinate points in mechanical systems. The basic elements of the electrical system are inductors, resistors, capacitors, and transformers and correspond, in mechanical systems, to masses, dampers, springs, and gears or levers, respectively. In the mechanical system each point of connection was labeled with a 2-digit number and then the location of each element was defined according to its points of connection. In the loop analysis of electrical networks each independent loop is labeled with a 2-digit number and the location of each element is defined in terms of the loops in which the element lies. For example, the symbol *E03K02* would denote a capacitor which is common to the two loops *03* and *02*, *E00C03* denotes a resistor appearing only in loop *03*, and *E00M02* an inductor in loop *02*. The symbol *EF01* denotes an applied voltage in loop *01*.

Except for these conventions for specifying the elements of an electrical network, the complete input information is presented on input cards as it was for the mechanical system previously described. In fact, the input cards shown in Fig. 2 could apply directly to the network shown in Fig. 6.

Conclusions

Experience to date has indicated that DYANA has provided a flexible means of describing dynamic systems to the computer and has eliminated much of the time and effort required in manual programming, coding, and debugging. It has also provided a tool to implement the analytical ability of the engineer and has enabled one unfamiliar with the techniques of computer programming to set up his own problem for the computer.

DYANA: Dynamics Analyzer-Programmer. Part II: Structure and Function

J. T. OLSZTYN

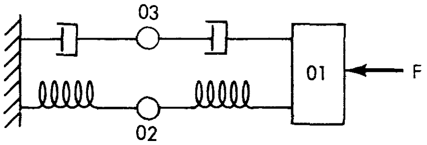


Fig. 2. Schematic of simple system

IN PART I of this paper, it was explained that DYANA is a program which performs the following functions:

1. On the basis of an input language, determines the structure of a dynamic system.
2. Prepares a mathematical model of that dynamic system.
3. Prepares a FORTRAN program which, in turn, is able to numerically solve the mathematical model.

In this paper some of the highlights will be pointed out in the structure and functioning of DYANA which permits it to perform these three functions. This will be done by considering a specific problem formulated in the DYANA language, and by showing how DYANA actually operates on this problem. Consider first the over-all structure and organization of DYANA

Over-all Structure and Functioning

The DYANA program is a collection of *704* SAP-coded routines. These routines are organized into four major groups:

1. Routines of the sequencer
2. Routines of the scanner
3. Routines of the analyzer
4. Routines of the generator

The over-all flow among these major groups is shown in Fig. 1.

At the start of a DYANA run on the *704* computer, a transfer of control is immediately made to the "sequencer." Next, the sequencer passes control to the "scanner." After the scanner has completed its function, it returns control to the sequencer. Fig. 1 shows that the same procedure applies next to the "analyzer" and then to the "generator."

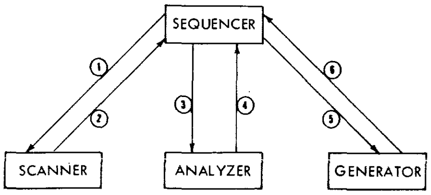


Fig. 1. Over-all flow of control

Briefly, the functions performed by the three processors are:

1. Scanner: The routines of this package have the responsibility of processing the DYANA and FORTRAN language statements and producing a set of primary tables.
2. Analyzer: The routines here operate on the primary tables and produce a set of secondary tables.
3. Generator: The routines in this category have the responsibility of operating on the primary and secondary tables to produce the FORTRAN program (not shown) representing the solution of the problem.

It should be evident in the light of what has just been stated that the major ingredients on which DYANA operates are tables. Therefore, in view of their importance, a few words about them is in order.

Two Basic Classes of Tables

Each table produced by DYANA, whether it is a secondary or primary table, falls into one of two classes. Either it is a member of a class of tables that has meaning only in terms of the content they hold, the ordering of the elements being only of secondary importance, or it is a member of the class of tables where ordering is the important thing, and the content has only secondary significance. As an example of these two classes, consider the following tables:

Table I.	Table II.
TABR	STAB
05	9001
07	9004
08	9016
FLAG	

Table I, TABR, is of the first class mentioned. The entries in this table are labels that the engineer has assigned to the nodal or displacement points in his system. The table is used by DYANA to write out the ordinary linear algebraic equations required by the mathematical model. One equation is written for each node point in the order that they appear in TABR. As far as the mathematical model or the FORTRAN program for that matter is concerned, it makes no difference whether an equation is written first for node *05* or first for node *08*.

Tables of this first class are also characterized by the fact that the number of entries in the table is strictly a function of the size of the physical system being studied. Therefore, these tables always have a special flag which is inserted immediately after the last entry. During subsequent operations on the table, the flag serves as an indicator signalling the end of the table.

STAB, Table II, is a table belonging to the second class. The elements of STAB represent statement numbers which appear eventually in the final FORTRAN program written by DYANA. These numbers have meaning only in terms of their position in the table. For example, the number appearing in the first position of STAB always means that this is the statement number which will begin the block of FORTRAN statements that will read the engineer's data into the computer. Or the number in position two must be the statement number of the block of FORTRAN statements that begin the solution of the differential equations of the mathematical model. From problem to problem, the actual value of the numbers appearing in those two positions may vary, depending on the logical structure and complexity of the final FORTRAN program.

How DYANA handles a particular problem will now be shown. For this purpose, a rather simple problem is selected from the vibrational field.

DYANA Formulated Problem

Assume that the schematic of a physical system is as Fig. 2. Two springs are attached in series, and similarly two dampers. Both series connections are grounded at one end, while the other ends are attached to a common mass. The nodal or displacement points are labled *01*, *02* and *03*. A forcing function acts on the mass.

J. T. OLSZTYN is with General Motors Corporation, Detroit, Mich.

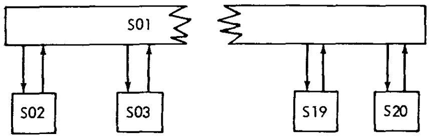


Fig. 3. Flow of control in scanner

Let the DYANA formulated statement of the engineer's problem be as follows

Card 1	*X*	System Description
Card 2		*E00M01, E01K02, E02K00, E00C03*
Card 3		*E03C01, EF01*
Card 4	*X*	Precomputation
Card 5		*E01K02*=2.4+0.6* *E02K00*
Card 6	*X*	Force, *EF01* (*XC3,X01*)
Card 7		*EF01*=*X03*−1.25 * *X01*
Card 8	*X*	Print time answers
Card 9		TIME, *DX01, EF01*
Card 10	*X*	End

Cards 1, 2, and 3 describe the physical system.

Card 5 contains a FORTRAN statement which evaluates the spring rate *E01K02* in terms of the rate *E02K00*.

Card 6 says the forcing function *EF01* is a function of displacement at the nodes *03* and *01*.

Card 7 is a FORTRAN evaluation of *EF01*.

Cards 8 and 9 indicate the output desired.

Card 10 says there are no other statements following.

The Scanning Process

The structure of scan package is shown by the Figure 3. Each *S* with its 2-digit suffix represents a separate routine. Beginning with *S02*, there are as many routines in the scanner as there are "X"-type statements in the DYANA language. One of the functions of *S01* is to pass control to the other routines via a vocabulary table (VOTAB). A portion of VOTAB, Table III, is shown which will be used to explain the functioning of the scanner package:

Table III

VOTAB
System
TRA *S02*
Force,
TRA *S06*
PRE-CO
TRA *S03*
Print *T*
TRA *S11*
End
TRA *S14*

The first word of Table III is a Hollerith word, while the second is a *704* instruction which, if executed, would pass control to *S02*. The other entries in this table have the same general meaning.

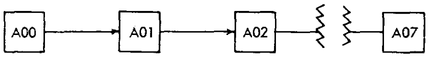


Fig. 4. Flow of control in analyzer

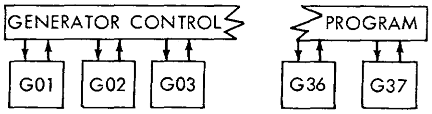


Fig. 5. Flow of control in GCP

Scanning begins with the sequencer passing control to *S01*. *S01* then reads card 1 into core memory. After detecting that there is an *X* on this card, *S01* proceeds to obtain the next word (six nonblank Hollerith characters) after the *X*. *S01* next checks the position of this word in VOTAB. When the position is established, *S01* passes control to the position immediately following. Thus, when the position of the word "system" has been found, *S01* would yield control to the second position in VOTAB. From that point, control would go immediately to *S02*. Whereupon card 1 would be processed and control would go back to *S01*. Card 2 is then read in and control is immediately turned over to *S02*. Upon completion of the processing of card 2, control again goes back to *S01*. Card 3 is handled in a similar manner. When card 4 is encountered by *S01*, control goes via VOTAB to *S06* for the processing of that card and the subsequent one, card 5. One can see that *S01* reads-in all DYANA statements, and through the use of VOTAB, relinquishes control to other routines for the actual processing.

The scanning process is considered completed when *S14* returns control to *S01*. After this occurs, *S01* returns control to the sequencer.

The processing carried out in the individual routines of the SCANNER consist of forming primary tables. Some of the more important of these tables are presented and what their content is after the scanning has been completed is shown.

The tables KTAB (Table IV), CTAB (Table V), MTAB (Table VI), and FTAB (Table VII) are of the first class of tables that was discussed earlier. KTAB contains all of the spring elements of the system, CTAB the damping elements, MTAB the masses, and FTAB the forcing functions in the system. The last entry in each of these tables is a special flag. The flags serve the purpose of terminating the tables. MSCTAB (Table VIII) is a table of the second class. It is used extensively by the generator to determine the structure and flow of the final FORTRAN program. The number 2 which actually occurs in position 39 of that table means that *S03* has processed a set of FORTRAN statements which were preceded by a statement of the type:

X PRECOMPUTATIONS

and that now this set of statements will be found intact as the second item on a certain magnetic tape. This procedure is followed throughout the scanning process whenever original FORTRAN statements are encountered. At generation time, these statements are recalled from tape and inserted into appropriate spots of the FORTRAN program.

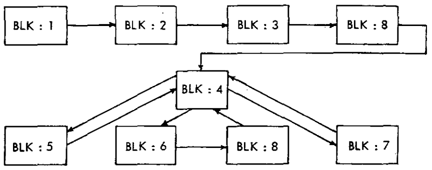


Fig. 6. Invariant flow

Analyzing Phase of DYANA

The purpose of the analyzer is to complete the table structure required by DYANA to:

1. Establish the mathematical model of the physical system.
2. Write the FORTRAN program which will numerically solve the model.

At present, there are a total of eight routines in the analyzer which are assigned this responsibility. The flow of control through these routines is shown in Fig. 4. The flow proceeds sequentially through the routines, starting at *A00*, going next to *A01*, then to *A02*, etc. In general, each routine makes entries into tables which will be used in the formation of the mathematical model and into tables used in developing the final FORTRAN program.

Table IV	Table V	Table VI	Table VII	Table VIII
KTAB	CTAB	MTAB	FTAB	MSCTAB
E01K02	E00C03	E00M01	EF01	
E02K00	E03C01	FLAG	FLAG	2
FLAG	FLAG			

A basic set of tables for the formulation of the mathematical model (shown in Fig. 2) are exhibited next.

Table IX	**Table X**	**Table XI**	**Table XII**
TABR	TABS	TABT	TABF
02 FLAG	03 FLAG	01 FLAG	01 FLAG

These tables are set up by several of the routines shown in Fig. 4. The rules of formation are:

Rule 1: Into TABR (Tables I and IX) go those 2-digit prefixes or suffixes which do not occur in either CTAB (Table V) or MTAB (Table VI).

Rule 2: Into TABS (Table X) go those 2-digit prefixes or suffixes of CTAB (Table V) which do not occur in MTAB (Table VI).

Rule 3: TABT (Table XI) consists of the 2-digit suffixes or prefixes found in MTAB (Table VI).

Rule 4: TABF (Table XII) consists of the 2-digit suffixes found in FTAB (Table VII).

The 2-digit suffix or prefix *00* is ignored. There are no duplicate entries in TABR, TABS, or TABT. In fact, these tables are completely disjoined. Their union, however, is the complete collection of nodal or displacement points to be found in the system.

Each 2-digit number in TABR corresponds to a point in the physical system for which an ordinary algebraic equation must be written. This follows from the fact that TABR represents those points in the system to which only spring elements are attached. For each entry in TABS, one first-order differential equation must be written. For each entry in TABT, one second-order differential equation must be written. The reason for the last statement is that TABT represents points at which there are mass elements.

Tables IV through XII allow DYANA, during the generation phase, to establish the following mathematical model for the physical system shown in Fig. 2.

$$E01K02*(X02-X01)+E02K00*X02=0$$

$$E03C01*(DX03-DX01)+ E00C03*DX03=0$$

$$E00M01*DDX01+E01K02*(X01-X02) +E03C01*(DX01-DX03)=EF01$$

These equations appear in an equivalent form in the final FORTRAN program.

One of the basic tables for developing the FORTAN program is the MSCTAB table. Into this table, the routines of the analyzer store information as to the number and types of equations required to represent the mathematical model, as well as the dimensions of any vectors, matrices, or other arrays that may be needed to conveniently express this model in terms of FORTRAN notation.

Generating Phase of DYANA

At the completion of the analyzing activities, enough information now exists in tabular form to permit the sequencer to pass control on to the generator group of routines to prepare the final FORTRAN program. The organization and function of this group is shown in Fig. 5.

The Generator Control Program (GCP) selects G-routines whose responsibilities are the actual preparation of specific blocks of FORTRAN statements. When a G-routine has finished writing its statements, it returns control to GCP. GCP must then decide what block of FORTRAN statements are needed next, and after this decision has been made to call upon the appropriate G-routine.

The reason that this type of a structure was decided upon for the generator stems from two considerations:

1. The nature of the mathematical model.
2. The type of FORTRAN program required to solve the model.

In its most general form, the mathematical model would be represented by the three matrix equations.

$$K(y,z,\dot{z},t)X=F_1(y.z,\dot{z},t) \tag{1}$$

$$C(x,y,z,\dot{z},t)\dot{Y}=F_2(x,y,z,\dot{z},t) \tag{2}$$

$$M(x,y,\dot{y},z,\dot{z},t)\ddot{Z}=F_3(x,y,\dot{y}.z,\dot{z}.t) \tag{3}$$

X, Y, Z are disjoined vectors whose union would contain all the nodal or displacement points in the sytem. The dot denotes differentiation and t is the independent variable.

If the problem facing the generator was simply that of preparing a FORTRAN program to solve the most general mathematical model, equations 1, 2, and 3, the task could be accomplished in a straight forward manner. The intent in setting up the DYANA program, however, was to be able to prepare FORTRAN programs for every specialization of the set of equations 1, 2, and 3. Furthermore, these programs had to be near optimum from the standpoint of execution time and the amount of storage or core memory they utilized.

A solution to this problem was obtained by considering first those aspects of a FORTRAN flow diagram that would remain invariant for the generalized model in equations 1, 2, and 3, as well as all its specializations. The invariant blocks of this flow diagram are shown in Table XIII.

Table XIII. Invariant Blocks

Block No.	Type of FORTRAN Statements
1	Dimensions, functions, and equivalences
2	Input
3	Statements for computations not depending upon the independent variable-time
4	Statements required by *XDEQIF*, A FORTRAN subroutine for integrating differential equation by the method RUNGA-KUTTA-GILL
5	Statements which solve for the unknowns of the mathematical model
6	Statements for computations which depend upon the solution of the model
7	Statements which control the computational process
8	Output

The invariant aspects of flow through these blocks is shown in Fig. 6.

After the invariant blocks had been established, they were divided into a number of sub-blocks. The division was determined by the types of FORTRAN statements that would be required for the various specializations of equations 1–3, and by the consideration of producing a nearly optimum FORTRAN program. For example, suppose in equation 2, the elements of the coefficient matrix C were all constants. In this case, it would be most desirable to have a sub-block of FORTRAN statements in *BLK:3* which set up the elements of C and invert it, and then another sub-block of statements in *BLK:5* which would use this inverse to solve for $\dot{Y}$.

Corresponding to each sub-block that was finally established there exists a G-routine in the generator. The way in which the GCP program selects the various G-routines depends then upon the invariant flow shown in Fig. 6 and the amount of specialization that exists in the mathematical model which at this point is ascertainable by way of the entries that were previously made in certain tables by the scanner and analyzer.

Some Conventions

In the planning of DYANA a number of conventions were established in regards to the statement numbers and symbols which would be used by the generator. Statement numbers are generated sequentially, starting with the number 9001, by a routine called *STNGN*. Whenever a particular G-routine needs a statement number, it simply transfers to *STNGN*. When control is returned to that G-rouine, the next available statement number appears in the accumulator.

Frequently, it is necessary for one statement of a block to make a reference to a statement of some other block. Since these blocks are set up by different G-routines and at different times, it may happen that a particular statement is being referenced in some block that has not even been generated yet. The table STAB was set up as a solution to this dilemma. Every statement that is generated and for which there is the possibility of it being referenced by statements produced by other G-routines has a specific position assigned to it in STAB. Suppose now that *G24* generates a block of statements the first of which is to be referenced by a "go to"-type statement of a block generated by *G25*. Assume also that GCP selects *G25* to prepare its statements before it selects *G24*, and that position 6 of STAB is to contain the number of the first statement generated by *G24*. The procedure for inter-G-routine referencing in this case is as follows:

At the time *G25* is ready to write the "go to" statement, it interrogates position 6 of STAB. If no statement number appears there, it obtains the next available statement number from *STNGN* and inserts it into that position as well as into the appropriate spot of the "go to" statement. Later when *G24* is ready to write its first statement, it also interrogates position 6 of STAB, finding in this case a number which it then appends to the first statement it writes. In the event that *G24* did not find a number in STAB, it would have followed the same procedure that *G25* did in getting its statement number. The convention that has just been outlined insures proper referencing of all FORTRAN statements, regardless of the order in which GCP selects the G-routines.

An exactly similar convention and referencing procedure was established for a certain class of fixed point variables required in the preparation of the FORTRAN program. Members of this class are designated by the symbol *NZXXX* where the *XXX* is a three-digit numeric taking on sequential values beginning with 001. These symbols are generated by the routine *SYMGN*. For the *NZXXX* symbols the table which corresponds to STAB is SYMTAB.

Other symbol conventions established are:

ZVXX always refers to a vector. The *XX* denotes a numeric which takes on sequential values.

ZMXX is used in referring to matrices.

ZEXX is an erasable floating point symbol. Has a meaning only in the context of statements created by a particular G-routine.

IZXX references fixed point variables, otherwise has the same meaning as *ZEXX*. Most frequently used as the running index of a DO loop.

Summary

The DYANA program consists of the four major components:

1. Sequencer
2. Scanner
3. Analyzer
4. Generator

The sequencer's prime function is to pass control to the other three components. The scanner processes statements written in the DYANA language and prepares primary tables. The analyzer operates on these primary tables and produces a set of secondary tables. At the completion of the analyzing activity enough information exists in tabular form to establish both the mathematical model and the FORTRAN program that will solve the model. The routines of the generator controlled by the GCP program interrogate and interpret various tables and then prepare the final FORTRAN program.

Conclusions

The author has been able to give only a glimpse of the peaks in the development, organization, and functioning of DYANA. It is hoped, however, that this has been sufficient to promote a flow and growth of ideas which will find eventual fruition in programs even more ambitious, and applicable to vastly more complicated systems than DYANA.

Discussion

G. W. Smith, Jr. (Bell Telephone Laboratories): What is the representation for a transformer lever or gear ratio?

Mr. Olsztyn: Let me work with a specific symbol say *E01N02*, in order to answer this question. First of all, such a symbol has a dual role. Appearing in certain DYANA statements, it signals the existence of a certain element connected to specific points of the system. Hence, in this role the symbol could be interpreted as follows: There is an element (*E*) which is a lever (*N*) whose terminals are points (01) and (02) of the system. If we are talking about a gear set, the interpretation would read: A gear set (*N*) exists in the system, with the gear at point (01) meshing with the gear at point (02). The interpretation for a transformer would then be: A transformer (*N*) exists whose primary (secondary) coil is in loop (01) and whose secondary (primary) coil is in loop (02). In the other role; i.e., whenever *E01N02* appears in a FORTRAN arithmetic expression, it refers to the numerical value of a lever ratio, or a gear ratio, or a ratio of windings in a transformer.

Paul H. Hertel (Atlantic Refining Company): What is the number of man-hours consumed in development and programming of DYANA?

Mr. Olsztyn: Four man-years.

Mr. Hertel (Atlantic Refining Company): Is it available to the industry?

Mr. Olsztyn: Not at present. Plans are being made, however, to release it about May of 1959.

J. H. Milsum (National Research Council of Canada): Your direct invasion of the analog computer's very own field (of dynamic problems) cannot pass unquestioned. It is true, of course, that scaling and manual setup have, in the past, prevented the analog computer from fully realizing its great advantages of fast solution time within engineering accuracy requirements. However, many new advantages in analog computers are appearing, simpler programming and servo potentiometer settings are examples. I would like to see figures before conceding that the digital solution is the cheapest all around.

Despite the above comments, however, I believe you have a very valuable new tool in DYANA.

Paul Benetean (Burroughs Corporation): Apart from simplicity of programming, what advantages does this system present over a conventional analog system?

Mr. Olsztyn: Well, I think in part you have answered your own question. Simplicity of programming is intended to be the main feature of DYANA. But more than that, DYANA represents an experiment to get a digital computer to not only do a major share of the programming of a problem, but it also represents an attempt to get a computer to do problem analyses and mathematical model formulation.

Karl Nording (Burroughs Corporation): Can you give an example of problems from other than the vibrational field, that can be studied using the DYANA programming system?

Mr. Olsztyn: Electrical circuits, many acoustical, magnetic, and heat transfer problems can be studied using this system. In general, any physical system whose dynamic equations are formally identical with the equations of motion of vibrating systems can be studied using DYANA.

J. C. Grube (United States Army Signal Research and Development Laboratories): What mathematical method of integration is used to compute results?

Mr. Olsztyn: Gill's variation of a fourth-order Runga-Kutta method.

Nora M. Taylor (David Taylor Model Basin): What configuration of *704* is required for DYANA? Does it use FORTRAN I or II? Can it handle shock vibrations?

Mr. Olsztyn: At present DYANA requires an 8,192-word memory and uses 4 tapes and 2 drums. The version of DYANA that will finally be distributed will be a drumless version requiring 4 tapes and will work on any 8,192-word or larger machine. The FORTRAN source program produced by DYANA can be compiled using either FORTRAN I or II. Shock vibrations can definitely be handled in the DYANA system.

The Univac Air Lines Reservations System: A Special-Purpose Application of a General-Purpose Computer

D. K. SAMPSON V. E. HERZFELD C. W. FRITZE

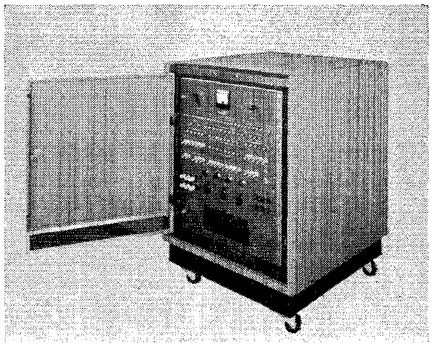


Fig. 2. Low-speed programmer-scanner

THE CONTROL of passenger space on an airline is a key function of greatest importance. It represents to the airline the major contact with its buying public. Essentially, the passenger space control problem is the allocation of a fixed inventory undergoing high activity up to the aircraft departure, at which time the salable commodity vanishes. Passenger space control systems must satisfy the following criteria:

1. From the customers point of view, the system must provide quick service and keep the probability of overbooking to nearly zero.
2. From the airlines point of view, the system must be fast enough to insure that any available seats on an aircraft are sold, thus insuring the highest possible profit margin.

These criteria apply to any reservations system, whether manual or automatic.

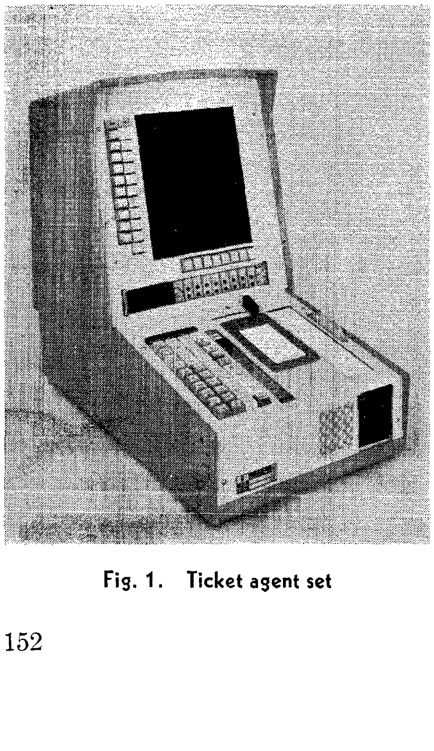


Fig. 1. Ticket agent set

The diligent application of efficiency techniques and time-saving routines have allowed the airlines to meet these criteria and, at the same time, impress the consuming public that excellent service in this category is only to be expected.

Since the performance criteria for these systems stress both speed and accuracy, it is natural to expect that the application of electronic computers should be considered in these systems. There has been an evolution of systems from the purely manual through the semiautomatic to completely automatic special-purpose magnetic drum machines. This paper describes the continuation of that evolution, the application of the general-purpose business-data processor in an on-line inventory control problem. It will attempt to show the soundness of such an application by showing the possibilities of future evolution of such systems by gradual growth of the peripheral system and increased utilization of the input-output power of the central computer.

Functional Characteristics of the System

The basic function of the airlines reservation system is to maintain an up-to-date inventory of seats on aircraft and to accomplish direct communication concerning that inventory between a central computer (or computers) and each reservation agent at the various places of business of the airline. A secondary function is to maintain a current record of flight status, including whether a flight is on time, early or late, the amount of time, and the reason for schedule changes. Communication with the inventory is initiated by the agent and consists of two broad types of transactions; first, those which merely inquire as to the status of a pertinent part of the inventory, and second, those which directly change the inventory. The first type of transaction includes "ask" transactions which merely query the computer as to the availability of seats on one or several flights. Flight information requests concerning arrival and departure times also fit this category. The second type of transaction includes "sell," "cancel," "waitlist," and "cancel waitlist." These transactions directly effect the inventory.

The Agent Set

The agent set (Fig. 1) is the input-output unit which permits the agent to establish communication with the computer. The agent set is basically a register of contact closures and gates comprising an input message register, and a group of lamps comprising an output register, which are illuminated as a result of a response message from the computer. A time-table projected from a slide onto a glass viewing screen is also provided. This satisfies the system need for a catalog of items for sale which are held in inventory in the computer. Photocells behind the upper front panel and adjacent to the viewing screen detect edge coding of the slide, which locates the general area of inventory desired. By means of push-buttons along the edges of the projected image of the portion of the airline time-

D. K. SAMPSON is with Telex Inc., St. Paul, Minn. V. E. HERZFELD is with Remington Rand Univac, Division of Sperry Rand Corporation, St. Paul, Minn., and C. W. FRITZE is with Monarch Electronics Company, Minneapolis, Minn.

table, the agent selects the flight number, origin, and destination of the flight. Numbers of seats, month, and day are entered by means of the appropriate pushbuttons, thus completing all of the required information to locate the inventory desired. The agent may then ask if seats are available on a given flight or on all the flights on the selected timetable, by means of transaction pushbuttons. By indicator lamps, his response reflects the condition of the inventory, indicating that he may sell the inquired seats, or that there is a waiting list and whether the waiting list is open or closed. By pushing a "clear" button and then a "sell" or "waitlist" transaction pushbutton, the agent completes his transaction.

Flight information inquiries are responded to by means of combinations of 36 indicator lamps which indicate that a flight is on-time or late, and the reason for the altered flight status.

Two very important reply lamps are "error" and "re-enter." An "error" reply indicates that a message of the type which can effect the inventory was received at the computer and some error was made during or after the computer processing cycle. Since the inventory could have been effected, supervisory action is required. "Re-enter" on the other hand, indicates that the message was found to be in error but was of a type which could not effect the inventory, or, in the case of inventory effecting transactions the error was detected before the process began, and thus can be re-entered by the agent.

Univac File Computer, Model I

The Model I, Univac File Computer, is the heart of the system. It satisfies the need for an inventory with input-output flexibility and record generation ability. The inventory is kept on random access magnetic drums rotating at 1,800 rpm with 180,000 characters capacity per drum. The usual arithmetic and control means are supplied and programming is stored either on a high-speed magnetic drum or is set up on a plugboard. Input and output to the computer is buffered by means of the high-speed magnetic drum which rotates at 12,000 rpm. The input-output section of this drum is organized into ten demand stations, each of which consists of two magnetic recording tracks of 120 characters capacity. A track switch function is supplied, which enables the computer to be carrying on a transaction on one of the tracks of a demand station while loading or unloading the other. Each demand station has a number of general-purpose control lines; i.e., ten computer to input-output control lines, ten input-output to computer lines and four high-speed input-output to computer control lines which are capable of altering the program. Record generation capabilities are provided in the Univac File Computer by an inquiry typewriter, a 90-column read-punch card unit, magnetic tape units or perforated tape units.

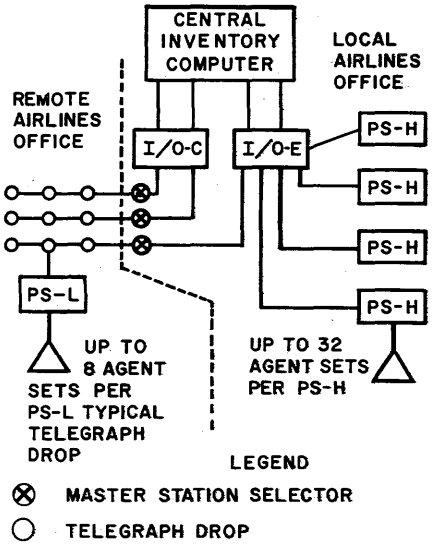

CENTRAL INVENTORY COMPUTER
LOCAL AIRLINES OFFICE
REMOTE AIRLINES OFFICE
I/O-C
I/O-E
PS-H
PS-H
PS-H
PS-H
PS-L
UP TO 8 AGENT SETS PER PS-L TYPICAL TELEGRAPH DROP
UP TO 32 AGENT SETS PER PS-H
LEGEND
⊗ MASTER STATION SELECTOR
○ TELEGRAPH DROP

Fig. 3. Basic airlines reservations system

The System Design Problem

The system design problem is essentially a communications design, switching, and an error control problem. The agent set and the computer described provide the means for manual entry of inquiries to an inventory and automatic response concerning that inventory status. Peripheral devices must be provided, which, when connected to communications facilities, the agent sets, and the computer, allow the agent's inquiries to be quickly and accurately answered.

The speed of response is dependent, to a great extent, on the following system characteristics: First, messages are generated at random times by a large number of agents. Second, each computer makes many types of transactions of varying program lengths. Third, the peak message generating capacity of the agents may exceed the peak handling capacity of the computer. For this reason, congestion may occur at times, resulting in a waiting time for some transactions and thus to some agents.

The error control problem is to guard against erroneous messages effecting the computer inventory and erroneous messages reaching the agents. This error control must be applied in such a way that the system does not "hang up" when an error is detected and thereby aggravate the inherent congestion problem.

The System Solution

The initial applications of this system are representative of the first solution of the system design problem. They are analogous to the "request and confirm" manual reservations systems. In such a system, each inquiring agent in turn has access to the central seat inventory held in a single central computer in order to place a reservation. The inventory is thus completely up to date and the probability of overbooking is zero.

The communication logic is half duplex in nature. That is, a message can be sent only to the computer or from the computer at a given time over a given line. Simultaneous message flow in both directions is not permitted. After acquiring the line, the agent holds the line until the message has been transmitted to the computer. The message waits at the computer for service, is serviced and the answer is transmitted to the agent set. The reply to the agent completes the transaction and confirms the reservation. In this system philosophy where the line is held until a reply is received, it is evident that message addressing is not required.

The Low-Speed Programmer Scanner

The first switching device between the remote agent set and the computer is the low-speed programmer scanner (shown in Fig. 2).

The low-speed programmer scanner is an electromechanical switching device which may connect up to eight agent sets to a telegraph line drop, thus satisfying the need for communications between remote agents and the computer. Its functions are: to seek the agent set with a message, to scan and program the message in the agent set register onto the telegraph line at a rate controlled by the telegraph distributor, and then to wait at the agent set connection until a reply message is received. It then provides light-holding power for the agents reply message and steps on, seeking the next agent with business. If no more business exists, it releases the telegraph line. The low-speed programmer scanner checks character parity and character count of input-output message and causes, under certain conditions, a repeat of the message if found to be in error.

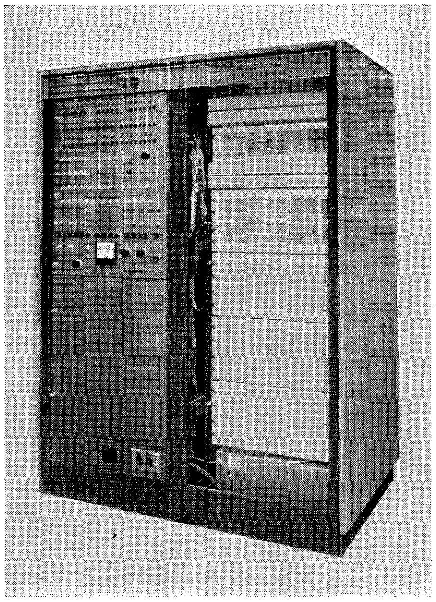


Fig. 4. High-speed programmer-scanner

The Telegraph System

The telegraph system (Fig. 3) consists of a master station control unit, and an outstation control unit or drop at each place of business of the airline. Each telegraph drop accommodates one to two low-speed programmer scanners. Information is transmitted at the rate dictated by the telegraph line, either 10 characters per second or 20 characters per second. Present systems are operating at 10 characters per second. The time to switch between drops in a selective sequence calling (polling) system which polls the line to ask which drops have message traffic is about 475 milliseconds per drop.

The High-Speed Programmer Scanner

The high-speed programmer (Fig. 4), a solid-state switching device at the computer site, satisfies the need for communication between the local agent and the computer. Information is transferred at a rate dictated by the computer, in this case, a character for each high-speed drum revolution time, or 200 characters per second. Each high-speed programmer attaches up to 32 agent sets via an input-output unit to the computer. All of its functions are similar to a low-speed programmer scanner.

The Input-Output Unit

The input-output (I/O) unit provides the major interface connection between the computer and the peripheral equipment by connecting telegraph master-station control units or high-speed scanners to the computer. Two units are now being provided; the Model C, which connects the computer to two master station control units or to two telecommunications systems, and the Model E, which connects the computer to one master station control unit and four high-speed programmer scanners. The track switch feature is not used in these I/O units since this system holds the line and no time would be saved by the time sharing feature of the track switch. The functions of the I/O unit are to record the input message on the I/O track in the proper format and signal the computer when ready. In response to a "computation finished" signal from the computer, a reply message is read from the track and transmitted to the high-speed scanner or the master station control unit. Information rates are again similar to the input. They are dictated by the telegraph line in the case of the connection to the master station control unit and by the computer in the case of connections to the high-speed programmer scanner. The I/O unit checks character parity as it unloads information onto the drum, repeating a character if found to be in error.

System Performance

Three systems of this type have been or are being installed. One of these is operating and is servicing 135 agents sets, 95 local and 40 on the remote telegraph connections. The system is fast enough to accomplish from 1 to $1^1/_2$ transactions per second with response times to the agent of 1 second locally and 10 seconds on the telegraph connections. These response times depend heavily, of course, on computer program time and will vary with each application.

Initial system performance has exceeded expectations with respect to reliability with a performance of 99.7% of scheduled time for the first 6 week period of operation. Actual transactions are numbering in excess of the estimated number. Program times are longer than estimated. Since the Univac Air Lines Reservations System is the application of a general-purpose computer in an on-line inventory processing system, the user has complete freedom in writing his program. For this reason, the program times are determined by the user and as a result, become part of the system environment. If a fixed program, or special-purpose machine were used in this application, the program times could be expected to be described as equipment operation times. The operational speed of each of the component equipments is not sufficient to predict the system speed when the system is placed in its environment. The environment determines the operational speed of the system, and, to this extent, the system user determines the speed of the system.

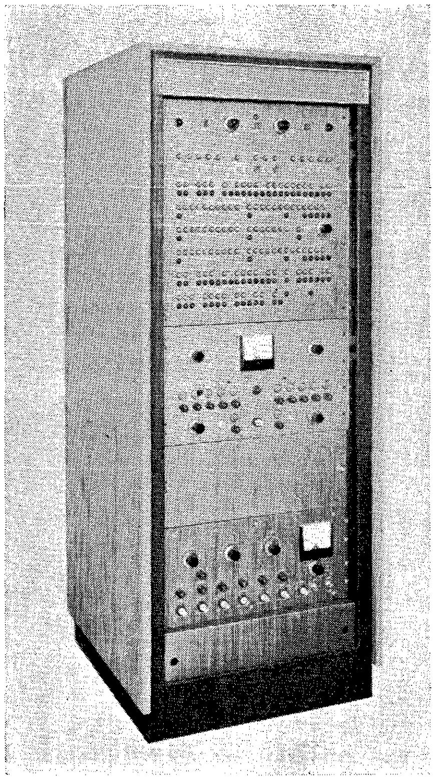


Fig. 5. Input-output unit

System Expansion

With the advent of jet travel and the expected increase in air travel, the airlines find themselves with an increasingly difficult reservations problem. A typical future system might be required to service up to 1,200 agent sets distributed over the entire nation. Message rates from these sets would be as high as 10 to 15 transactions per second. Response times to the agents should be, however, the same. That is, the response to the local agent must be within one second and to a remote agent, 10 seconds.

Systems of the type just described may be expanded by duplications of the basic system. The inventory in such an expanded system would be divided on a geographical basis. The advantage of zero probability of overbooking would be retained if an agent set in city *A* could make sales against the inventory held in a computer in city *B*. The need for rapid communications between computers is clearly indicated for such an expansion.

Another solution of this expanded system design problem, which draws an analogy with the previous expansion techniques of the manual reservations systems, involves changing the system philosophy from a "request and confirm" system to a "sell and record" system. This results in a system which is decentralized by function rather than by geographical distribution of inventory. When it is recognized that the types of transactions which directly affect the inventory are less in number than those which merely ask about the status of an inventory, it is recognized that there is an opportunity for a filter concept. This concept utilizes the fact that several status computers can be operated simultaneously, answering questions only as to status of the inventory, while a single inventory computer changes its inventory in response to sell transactions, in the case of airlines applications. Sales are made against the status record in such a system rather than against the inventory itself. Sell messages must then be transmitted to the central inventory computer in order to change the inventory. Replies are made to the status computers when a status change is made by the transaction. Such a system is vulnerable to overbooking during the time that updatings are being made and the need for high-speed communications between the status computer and the inventory is established.

Initial analysis also indicates that a computer is not needed for each large center of business of the airline, that is, each major city. However, the message traffic generated in such large installations exceeds the capacity of telegraph equipment. Telephone quality lines capable of transmitting the data faster must be provided for information flow between large cities and the status computer. By application of statistical analysis, in particular queuing theory, to such a system, it is soon recognized that a hold-the-line philosophy of half-duplex communications may seriously limit the speed of the system.

The half-duplex or strict "request and confirm" system in the queuing sense is analogous to a series of long corridors leading to a single cash register and a large number of people trying to get through these corridors to be serviced at the cash register. Only one person can occupy the corridor at a time, so that the time the line is held busy is not only the time taken to travel the length of the corridor, be serviced at the cash register, and return, but also the time that is spent in waiting at the cash register. Persons at the input end of the long corridor are then waiting while persons at the other end are also waiting. This represents a highly inefficient use of the line or the corridor. Further queuing analysis indicates that the application of full-duplex telephone speed communications more than doubles the speed of response or, in some cases, more than doubles the speed of the system at a reduced cost compared with half-duplex communications.

This change of communications philosophy from half-duplex to full-duplex in all or part of the system represents a major system philosophy change. The first effect of this change requires that messages sent from the computer over the full-duplex line must be addressed to their origin since the line has been relinquished by the origin after sending the message. The second effect of the change requires that buffering for more than one message must be provided at the computer at the output of a full-duplex line, in order that messages may be sent one after the other, thus fully utilizing the line. The third is a major change in error control philosophy. In the hold-the-line system, each message originating center required that a response be given indicating that a transmitted message has been accurately received before relinquishing the line. The full-duplex system requires that a correct copy of a message be retained at the origin until the functional reply to that message is received, allowing the message to be cleared or erased.

Queuing analysis was applied to determine which portions of the basic system should be full-duplex and addressable. It was found that the agent set, the high-speed scanner, and the low-speed scanner are adequate in either expanded system and may be applied without change. The high-speed scanner and the low-speed scanner may "hold the line" after sending their message at normal agent set traffic rates without limiting the system capabilities. This is true because the low-speed scanner is generally located in areas of low activity and low-message rate transmission; i.e., telegraph, and the high-speed scanner is located in areas of high activity and is served by high-message rate transmission, i.e., 200 character per second lines. The initial specifications required a 10-second response at the low activity centers in the remote areas and a 1-second response in the high activity centers. The probability of blocking traffic at these devices under these conditions can be shown to be small and about equal for each device.

It would seem that addressability should extend also to the telegraph line. Again, queuing analysis shows that this feature would not substantially improve the system performance and would generally increase the cost. This results from the specific nature of this particular application. That is, the airlines problem is one which consists of a high activity of short messages. Since the polling time is comparable with the message transmission time, the system is switching limited thus permitting a hold-the-line solution. In the more usual system, such as air traffic control, it is found that the message activity is lower and messages are of longer length. In such a system full-duplex telegraph communications could be expected to substantially speed up the system.

The Multiplex Unit

Several peripheral equipments are now proposed and are being investigated to implement the incorporation of full-duplex communications in airline-type problems. In general, these new equipments are envisioned as applicable to any on-line inventory problem. The first of these devices, which permits addressability of messages from the computer to the origin, is the multiplex unit, which is merely a switch that connects the computer to the line which terminates at the message origin. This can be accomplished by detecting an address character in the content of the message. In a multilevel system address characters can be added at each input multiplex unit as the message proceeds towards the computer and a character can be subtracted at each multiplex unit as the message is addressed.

The Communications Control Unit

Communications control units (CCU) are being explored as equipment to be associated with the multiplex units. The function of these units would be to simultaneously transmit digital information from and to the digital portions of the system, the computer and the programmer scanners, and to receive and send modulated information on a full-duplex telephone line at a rate of 200 characters per second. The communications control units will send messages of variable length and format. Error control will be implemented by means of character parity checks and message parity checks. The communications control units will signal the next unit in line by means of control lines if messages were found to be in error, causing the source address of the erroneous message to be displayed for maintenance purposes. In contrast to the hold-the-line system or half-duplex system, the CCU's in the full-duplex system would not attempt to repeat a message or a character found in error.

As previously pointed out, the system gain in speed is minor by providing full-duplex telegraph communications in a system which is characterized by large message traffic volume of short messages. For this reason, a telegraph buffer and control unit is being explored which functions in a manner similar to the computer in the basic system. That is, it would hold an input message to be read out through the multiplex unit onto a CCU or an I/O unit, and await the reply message.

The Input Unit

The bulk of the system adjustment to provide for full-duplex communications must be handled by the input-output unit design. It is here that the value of a computer with true versatility of input-output is experienced. A major system-design philosophy change can be made with no modification of the computer, but merely greater utilization of the input-output functions provided.

The new input-output unit being investigated would actually be two boxes; an input unit and an output unit. The input unit would take information from a multiplex unit or a CCU and transmit it to the I/O track. Consistent with the nature of an on-line system, there would probably be many input units for each output unit, since messages arrive randomly at the computer and simultaneous arrival may be expected. Since the computer can handle only one message at a time, messages will be in line as they come out, thus requiring only a single output unit. The functions of the input unit are to arrange incoming message on an I/O track of the high-speed drum and keep a record of the number of messages received since the last service at that track. The input unit would signal "ready" following the receipt of each message and await its turn to be serviced. However, when the number of messages buffered on the I/O track reaches a level close to the full condition of the buffer, the I/O unit must signal the computer that it is in a priority condition and claim precedence over less busy tracks. In the Univac File Computer, sufficient buffering capacity is provided for five messages on each track for a total backlog at each demand station of ten airlines reservations input messages. The four high-speed control lines, previously not used in airlines applications, are well suited for priority indication. These lines, W, X, Y, and Z, may each be associated with an input unit. As input unit X, for example, reaches priority condition, the computer may find input unit Y ready. When input unit Y is demanded, it will respond with signal X, indicating that the computer should skip to demand station X and process messages buffered there.

The Output Unit

The output unit would have a task which is new compared with the basic I/O unit in that it must follow the error control logic of retaining a valid copy of all messages sent from the computer to another computer until the messages have been acknowledged. Again, the buffering capacity of the high-speed drum is sufficient to store a large number of output messages. By means of the track switch feature the computer can write an output message on one track of the demand station, while the other track is busy transmitting a message to other units. By means of the low-speed computer to I/O lines A through J, a computer can record the fact that the output unit has sent a message from storage location A by pulsing control line A and require that a new message may be recorded for transmission at word location A only after control line F is pulsed, signifying that the computer has received a valid answer to the question sent from word location A. If the computer subsequently tries to send a message from word location A without having received F, the message stored at word location A, the valid copy, is then printed out under a subroutine.

Conclusions

In conclusion, it has been found that this system can be expanded to accommodate an increase in traffic by means of equipment duplication or reorganizing the system into one which is decentralized by function or geography, utilizing full-duplex high-speed digital communications and very fully using the input-output versatility of a machine like the Univac File Computer. The author wishes to emphasize that this system design problem has not been solved solely by the company's efforts, but include contributions from customers and prospective customers. The basic solution was inherent in the initial design of the input-output system of a general-purpose machine like the Univac File Computer, which takes in stride a major design philosophy change which may not have been anticipated in the design of a special purpose machine. The effort has been to design peripheral equipment with this lesson in mind. As a result, it is believed that the airlines reservation system peripheral equipment described here, with the exception of the input-output units, is directly applicable to new general-purpose computers. The multiplex unit, the communications control units, the telegraphic buffer and control units, and the proposed full-duplex input and output units are applicable in any Univac File Computer on-line inventory problem.

Discussion

E. Sacks (Teleregister Corporation): Do you propose to handle other airline functions, such as fare quotation, ticketing, and maintenance of passenger records?

Dr. Herzfeld: Fare quotation on an intra-line basis can be easily handled on the slide or groups of slides which can be projected right onto the agent's screen. Inter-line fare computation, which I believe is what Mr. Sacks is referring to, is still being investigated.

Electronic passenger control record storage will arrive when development is complete on our mass storage system.

The automatic ticketing problem has not been completely investigated up to this moment. We are concentrating initially on handling the anticipated volumes of traffic which will arise as the systems are expanded.

The Siemens Digital Computer 2002

H. W. GUMIN

THE SIEMENS Digital Computer *2002* is a medium-scale transistorized computer being developed by the Siemens & Halske AG, Munich, Germany. The *2002* is a general-purpose decimal machine with a word length of 12 decimals plus sign and an average speed of 2,000 operations per second. Special features of the *2002* include three index registers, the use of the instruction location counter for address modifications, the automatic address substitution, and fixed- and floating-point operations. The *2002* has a transistor-driven magnetic core memory of variable size (units of 1,000, 2,500, 5,000 and 10,000 words) and a magnetic drum memory with a capacity of 10,000 words. Input and output data are handled by means of punched paper tape and punched cards. Considerable expansion by magnetic tape equipment is possible.

Characteristics

The *2002* is a decimal machine, where a decimal digit is represented by a 4-digit binary number (excess-three-code).

Word Structure

A word can be interpreted by the machine in four different ways, namely:

1. As an instruction.

±	M	R	O	O	O	S	A	A	A	A	A	I
S	1	2	3	4	5	6	7	8	9	10	11	12

Decimal 1 can be used together with the sign to mark an instruction. Decimal 2 serves several purposes. It indicates for instance, whether the result of an arithmetical or shift operation is to be rounded or not. The operation to be executed is identified by the three decimals, 3 to 5. Both decimal 6 (address substitution) and 12 (index tag) are used for address modifications. The address part of the instructions is given by decimals 7 to 11.

2. As a fixed-point number, with the decimal point being assumed on the left of the most significant digit, the numbers being represented by sign and magnitude.

3. As a floating-point number, where the mantissa occupies ten (decimals 1 to 10) and the characteristic two places (11 to 12).

4. As an alpha numerical expression with two decimal digits characterizing one alphanumerical character.

Address Modification

The usefulness of an instruction code depends greatly on the possibility of performing automatically address modifications. The *2002* allows modification of the address part of an instruction in two different ways, namely by address substitution and by index register modification, this being dependent on the contents of position 6 (substitution) and position 12 (index register modification) of the instruction word. These two types of modification can be combined and are carried out as follows:

The control unit of the *2002* includes three index registers, numbered 1, 2, and 3. When executing indexable instructions, the number in position 12 of the instruction (in the instruction register) determines the index register, the contents of which is to be added to the address part x (position 7 to 11) of the instruction. The number "4" in the index tag indicates that the contents of the instruction location counter is to be added to the address part of the instruction.

After this modification of the address part by the contents of one of the index registers or by the contents of the instruction location counter, resulting in a modified address x_1, position 6 of the instruction word is checked.

In case the number in position 6 is "0," the instruction will be executed in the normal way with the modified address x_1.

If the number in position 6 is "1," then the contents of location x_1 is read out of the memory, and positions 6 to 12 of the contents of location x_1 replace positions 6 to 12 of the instruction in the instruction register address substitution. Then the cycle starts again with a modification of the new address part by the contents of one of the index registers or by the contents of the instruction location counter, dependent on the number in position 12 of the instruction word and so forth.

The process ends after an index register modification, resulting in a modified address x_n, position 6 of the instruction word contains "0." Following this, the instruction (with address part x_n) is executed in accordance with the instruction list.

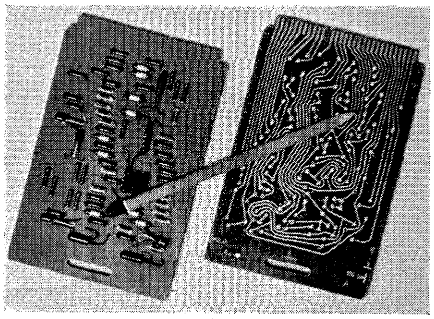


Fig. 1. Circuit board

If the instruction to be executed is not indexable, the modifications by the contents of one of the index registers are suppressed. In case the number in position 6 is "0," the instruction will be executed in the normal way. If position 6 of the instruction word contains "1," only positions 6 to 11 of the contents of location x replace positions 6 on 11 of the instruction word in the instruction register. Again position 6 of the instruction word is checked, etc.

A few examples are to illustrate the foregoing. The (indexable) instruction

+00 ADD 0 99999 4

will effect the addition of the contents of the preceding memory location to the contents of the accumulator, independent of its position in the memory (important for relative programming). If, for instance, the contents of the index register 1 is 100, the contents of the index register 2 is 200, and the contents of memory location 600 is

+00 000 0 3570 2,

then the instruction

+00 ADD 1 00500 1

causes the contents of storage location 3770 to be added to the contents of the accumulator. The (not indexable) instruction

+00 ADI 0 00001 2

effects the addition of 1 to the contents of index register 2, whereas

+00 ADI 1 00001 2

results in an addition of the contents of storage location 1 to the contents of index register 2, provided position 6 of the contents of storage location 1 is "0" (otherwise the process of substitution would be repeated).

Instruction and Speed

The instruction list of the *2002* contains:

1. 28 instructions for arithmetic operations for fixed-point and floating-point numbers, shift operations and other specific operations. In order to increase the calculating speed, devices are provided

H. W. Gumin is with Siemens & Halske, A. G., Munich, Germany.

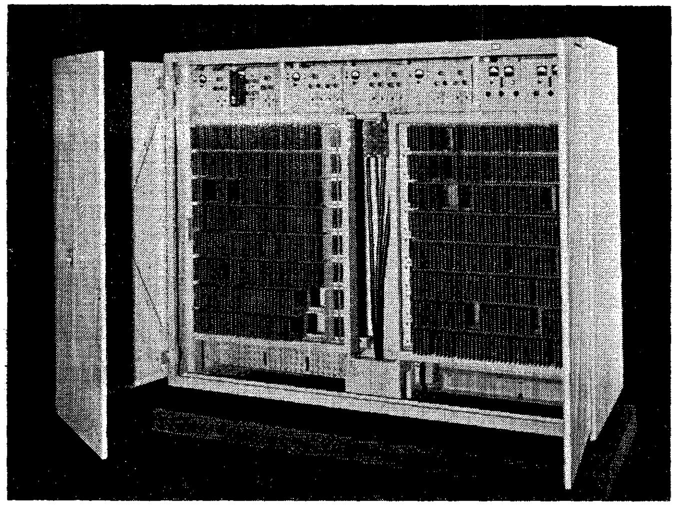


Fig. 2 (left). Central processing unit

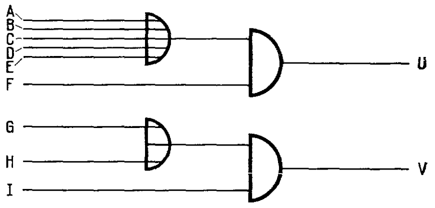


Fig. 3. Circuit board I

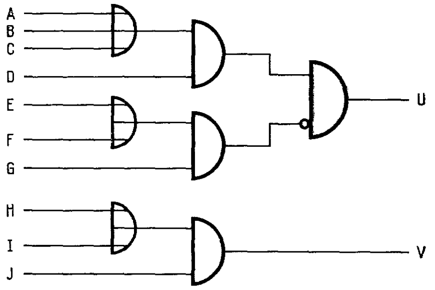


Fig. 4. Circuit board II

which produce multiples of multiplicands and divisions. Special attention has been paid to the built-in unnormalized floating-point arithmetic, so that, roughly speaking, the precision of the results is about the same as the minimal precision of the two operands.

2. 12 jump and other control instructions, one of which, the so called subroutine jump UNT (*unterprogramm*) is executed as follows: suppose the instruction

UNT β

is stored in memory location α, then $\alpha+1$ is stored automatically by this instruction in memory location β (precisely: in positions 7 to 11 all other positions being reset to 0) the next instruction to be performed will be taken from memory location $\beta+1$. The last instruction of a subroutine is a fine example of address substitution. The (unconditional) jump instruction SPR (*springe*) for switching the control to memory location $\alpha+1$ of the main program has the form

SPR 1 β

where position 6 (address substitution) is equal to 1.

3. 9 instructions for index register operations including jump instructions dependent on the contents of an index register.

4. 4 instructions for a transfer of data between drum and core memory in blocks of variable length.

5. 9 instructions for punched paper tape input and output.

6. 9 instructions for punched card input and output.

7. 7 additional instructions for magnetic tape equipment.

Total: 78 instructions

The sign of an instruction is used for program testing. Normally the sign of an instruction is not considered by the control unit. After pressing a button on the control desk, instructions with a positive sign are executed as usual, instructions with a negative sign (which are not performed) initiate the following procedure:

The contents of the instruction location counter is stored in memory location 0.

The next instruction in sequence will be taken from memory location 1.

All operation cycles of the *2002* are integral multiples of the "basic machine cycle," which requires 90 μsec. One basic machine cycle is equal to the time interval necessary for adding the content of a memory location to a number in the accumulator, including reading out of the addend. Reading out of an instruction from the core memory, its interpretation by the control unit and the address modification by index registers also covers one basic machine cycle. Each address substitution requires another 90 μsec.

Reading, interpreting, and address modifications of the next instruction are done by the control unit simultaneously with the carrying out of the preceding instruction provided the control unit is not used for the execution of the preceding instruction and the execution of the latter requires more than 90 μsec. For this reason the numbers in Table I, Operational Speed, do not include the 90 μsec for reading, interpreting, and address modification (by index registers) of the instruction. The next instructions are in some cases even carried out during the execution of a preceding one, as is shown by the sequence of instructions:

MLT	x	Multiply, fixed point
ADI	y	Add to index register
SPR	z	Jump (unconditional)

After initiating the multiplication in the arithmetic unit, the control unit reads and interprets the next instruction in sequence (one machine cycle). Because the instruction "add to index register" can be performed without using the arithmetic unit (the control unit has its own adder), the ADI instruction is executed (two machine cycles). Then, the next instruction is read out of the memory, interpreted and executed, as the unconditional jump can be performed independent of the arithmetic unit, etc. This is done during the execution of the multiplication (14 cycles).

Table I. Operational Speed

(Including Reading Out of the Operand)

Add, fixed point	90 μsec
Multiply, fixed point	1,260 μsec
Divide, fixed point	3,510 μsec
Add, floating point, normal	450 μsec
Multiply, floating point, normal	1,350 μsec
Divide, floating point, normal	3,240 μsec
Shift accumulator, n places	180 μsec

Table II. Physical Characteristics

Internal number system	Decimal Excess-three-code
Mode of operation	Serial Bits of a decimal digit parallel
Word length	12 decimal digits plus sign
Timing	Synchronous
Pulse repetition frequency	200 kc per second
Circuitry	Transistorized mounted on circuit boards
Memory	Core memory Magnetic drum
Power consumption basic computer*	5 kw
Floor space requirements basic computer	600 square feet

* Central processing unit, 1,000-word core memory, 10,000-word magnetic drum, control desk, punched paper tape input and output.

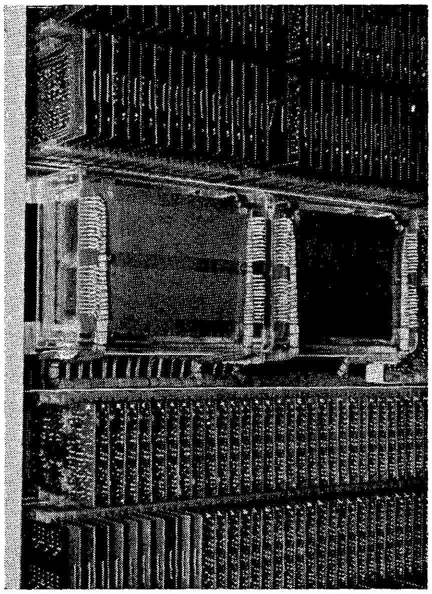


Fig. 5. Core memory unit (1,000 words)

Under certain assumptions concerning the distribution of instructions of a program (25% additions, 25% multiplications, 50% instructions for organization etc.) the average speed of the *2002* amounts to 2,200 operations per second for fixed point and 1,850 operations per second for floating point calculations.

Memory

The basic memory unit for the *2002* is a coincident current ferrite core memory. There are several sizes of the matrix of the core memory, depending on the memory capacity, namely

1. A 20-by-50 matrix for a 1,000-word memory.
2. A 50-by-50 matrix for a 2,500-word memory.
3. A 50-by-100 matrix for a 5,000-word memory.
4. A 100-by-100 matrix for a 10,000-word memory.

52 of the matrices are used; this allows storage of 1,000, 2,500, 5,000 or 10,000 words of 13 decimal digits each with a decimal digit being represented by four binary digits (excess-three-code). The access time of the core memory is 5 μsec, the cycle time 14 μsec. Up to ten memory units can be connected with the *2002*.

A drum memory (capacity 10,000 words) is attached to the ferrite core memory. Data are transferred between these two types of memory in blocks of variable length by a sequence of three instructions giving:

1. The drum address of the first word of the block.
2. The length of the block.
3. The address of the first word in the core memory.

The direction of transfer is also indicated by the last instruction. The average access time per word to the drum memory is greatly reduced when transferring blocks of reasonable length. The drum revolution time is 23 milliseconds (ms), the drum clock frequency 200 kc per second.

Words are written on four parallel tracks with 3.7 bits per millimeter (94 bits per inch) and 250 words per group of four tracks. The track selection is made by dry reed relays in such a way that always two consecutive groups of four tracks are selected. After having finished reading (writing) the first selected group of four tracks, switching to the next group of tracks is done by electronic equipment, so that uninterrupted reading (writing) is possible. While reading (writing) the second group of tracks, the third group of four tracks is selected by relays, etc. Thus, the number of read-write amplifiers is reduced from 4×40 to 4, but the average access time to the first word of the block is increased by 7.5 ms. With a word transfer time of 90 μsec, the transfer time of a block of n words is on the average

$$(11.5+7.5+n\cdot 0.09)\text{ ms}$$
$$=(19+n\cdot 0.09)\text{ ms}.$$

The average transfer time of 1,000 words is therefore 109 ms.

Input-Output

Input and output data are handled by means of punched paper tape, punched card and magnetic tape equipment.

In the case of punched paper tape, data are read into the machine by a photoelectric high-speed tape reader, one character per instruction, with a speed up to 200 characters per second in start-stop mode. Results are punched on paper tape at a speed of 50 characters per second.

If input data and results are to be available in the form of punched cards, a special buffer and control unit can be added. Punched card machines of various makes can be connected with the card control unit.

The punched card control unit contains a core buffer storage for each punched card input or output. The buffer storage has a capacity of 80×12=960 bits, the columns (rows) of the buffer being the equivalent of the columns (rows) of the punched card. The next card is read into the input buffer storage (the contents of the output buffer storage is punched into the next card) while calculating. A special column, numbered 0, is provided in the input buffer storage. Any of the 960 points of a card can be wired into any position of column 0.

Data are read out of the input buffer columnwise, beginning with column 0, continuing with column 1, 2, 3, ... Up to 12 consecutive columns can be read by one instruction, the number of columns as well as the buffer unit are indicated by the address part of the read instruction. The execution time for reading 12 columns is 540 μsec. Data are read into the accumulator. There are two groups of instructions for reading numerical and alphanumerical characters being represented by two decimal digits. The output buffer is filled columnwise in a similar way. If a printer is to be connected with

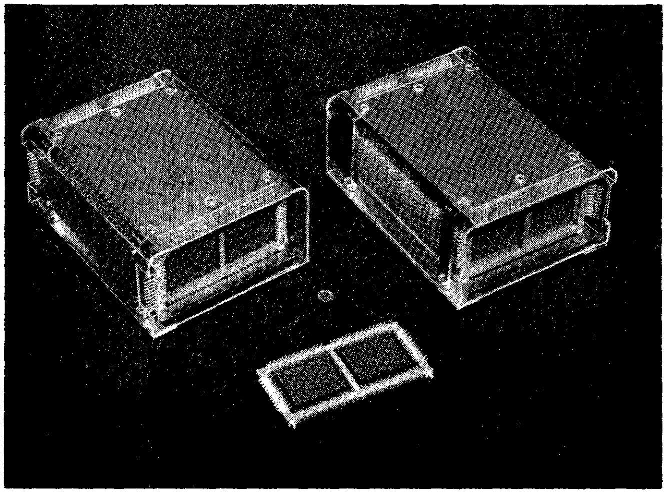


Fig. 6 (left). Core memory matrices

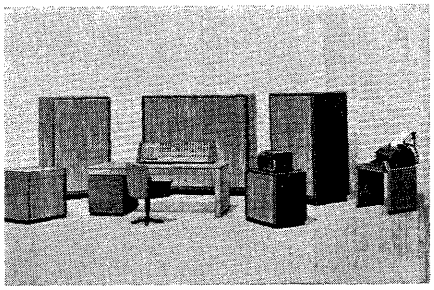


Fig. 7 (right). Siemens Digital Computer 2002 an output buffer, the number of columns can be increased.

The magnetic tape equipment is attached to the *2002* in a similar way. The tape control unit contains one or two buffers with a capacity of 1260×6 bits (one character per column).

The next record on tape is read into the buffer storage (the contents of the buffer storage is written on the tape) by read (write) instructions while calculating. If two buffers are installed, two records can be transferred simultaneously to or from two tape units. Up to 10 tape units can be connected with the tape control unit.

The instructions for reading data out of the tape buffer into the accumulator and for filling the tape buffer from the accumulator are the same as for the punched card buffers. The execution time for reading 12 columns is 270 μsec. Because of the length of the tape buffer, four block transfer instructions are provided which transfer the next n columns of tape buffer k into the core memory, beginning with memory location x, or the data stored in the core memory beginning with memory location x, into the next n columns of tape buffer k; two instructions serve the transfer of alphanumerical and two instructions the transfer of numerical data.

Physical Construction

For the most important physical characteristics see Table II. The circuitry in the arithmetic and control unit is of the dynamic type using gates and delay-lines. The circuitry in the memory units is of the static type. All circuits are composed of 16 types of different units which are dip soldered into etched circuit boards of the plug-in type (size 6.5 ×4 inches), one of which is shown on Fig. 1. Fig. 2 shows a central processing unit. The control and arithmetic units are of five different types, the logical configuration of two of them is reproduced on Figs. 3 and 4. The output of the circuit boards is delayed by 1.25 μsec. Up to 8 inputs can be driven by an output of one of the circuit boards.

On Fig. 5 a part of the rackmounted transistorized core memory unit (1,000 words) together with the read and write amplifiers are to be seen. The 52 matrices are combined within two blocks of 26 matrices each, Fig. 6. In the case of the 2,500-word memory, 160 plug-in units with 1,037 transistors are built into the memory unit (excluding buffer registers).

Fig. 7 gives a general view of the *2002* computer (punched paper tape input and output).

Conclusions

The *2002* has been working since November 1, 1958. A prototype has been operating successfully since October 1956. The *2002* is an enlargement of this prototype

Discussion

E. Goldstein (Bell Telephone Laboratories): Are parity bits included in the core memory? What checks are made to insure accuracy during data transfers between cores and drum, etc.?

Dr. Gumin: One parity bit per word in core memory; and four parity bits per word in drum memory. Data transfers are checked by means of "pseudo tetrads" (forbidden combinations of four bits).

J. Reitman (Teleregister Corporation): Please describe the size of the input and output punch paper tape buffers.

Dr. Gumin: The prototype has punched paper tape buffer of one word (input, one word; output, one word).

J. D. Babcock (The Rand Corporation): Has an assembler-compiler system been devised for the *2002?* If so, would you please give a short description of it?

Dr. Gumin: An assembler-compiler system will be available during the summer of 1959. This will be a generalized formula translating system.

G. M. Wilson (United Shoe Machinery Corporation): What is the card-per-minute speed of reading and punching? Also, what are the read-write speeds of the magnetic tape?

Dr. Gumin: Siemens does not manufacture punched card equipment. Punched card machines and printers of various manufacturers can be attached to the *2002* computer. The system is designed to permit the use of either International Business Machines Corporation or Bull peripheral equipment.

W. R. Haber (Remington Rand Univac): What is the drum speed in milliseconds, and how many drum bands are there?

Dr. Gumin: A drum revolution requires 23 milliseconds, and there are 40 bands each containing 50 words per band.

Design of the RCA 501 System

J. G. SMITH T. M. HUREWITZ

THE RADIO Corporation of America (RCA) *501* System (Fig. 1) was designed specifically for a wide range of data processing applications. This paper describes the system and relates specific functional characteristics of the requirements that are found in data processing activities.

System Highlights

Some of the outstanding features of the RCA *501* Electronic Data Processing System are mentioned here to provide a framework of reference for the remainder of the discussion.

Solid State Technology

The RCA *501* Data Processing System is comprised entirely of solid-state devices. The logic of the system is implemented with transistors, over 90% of which appear in the basic "Nor" circuit configuration. There are approximately 17,000 transistors in a typical system. During the first 6 months of 3-shift operation involving more than forty-four million transistors hours, there were only three transistor failures. These initial data indicate an expected failure rate of less than one transistor failure during 1,000 hours of system operation. The use of transistors has had a potent impact on design particularly in the area of reliability.

Variable Data Length

In keeping with the emphasis on data processing, the *501* system retains the RCA-developed concept of variable word length. This concept of information organization on magnetic tape lends itself naturally to the types of data encountered in processing applications and results in minimum processing times.

Building Block Expandability

Another highlight of the RCA *501* system is its expandability features. The high speed magnetic core storage of the computer may be expanded from a minimum of 16,384 characters (a character

J. G. Smith and T. M. Hurewitz are with Radio Corporation of America, Camden, N. J.

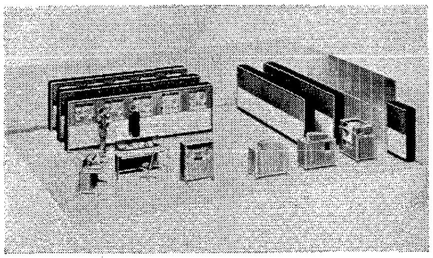


Fig. 1. The RCA 501 system expanded

being 7 bits) to a maximum of 262,144 characters. The expansion occurs in increments of 16,384 characters. The number of magnetic tape mechanisms directly under program control may expand from one (1) to a total of 63. Finally, peripheral equipment may be used on-line or off-line as dictated by machine loads.

For reasons of economy, particular emphasis was given to reducing, or in some cases eliminating, the need for special control devices for operation of peripheral equipment.

Powerful File Handling Capabilities

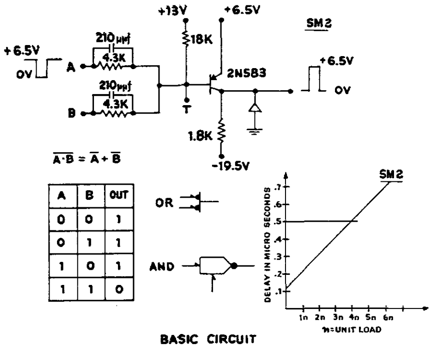
The computer records on magnetic tape at the rate of 33,300 alphanumeric characters per second. The information can be read from the tape while it moves in either direction.

To add to the file handling capabilities of the *501* system, the computer performs certain combinations of operations simultaneously. These combinations are read-compute, write-compute, and read-write. The simultaneity is handled by an interrupt technique which will be discussed in greater detail.

Accuracy Control

Again consistent with the data-processing emphasis, the *501* system incorporates a thorough system of wired-in checks which monitor and safeguard the processing of data. Among these checks are provisions for performing each arithmetic step twice and a complete parity preserving system of information transfer and modifications.

Basic Circuit



A	B	OUT
0	0	1
0	1	1
1	0	1
1	1	0

Fig. 2. Basic transistor circuit

Fig. 2 is the so-called "Nor" circuit which is the basic circuit used repeatedly in the over-all system. (The *SM2* is the RCA designation for this submodule.) The operation of this basic circuit is as follows: If both inputs are sitting high, the emitter-base junction is reverse-biased and hence the transistor is cut-off. This causes the output to be clamped to ground. When either one of the inputs goes low, the divider network yields a potential which is below the emitter potential and the transistor turns on. The output rises to 6.5 volts, less whatever drop occurs through the transistor, which in this case is a maximum of 0.3 volt. This basic circuit is used to achieve signal standardization and to accomplish all logic. The basic logic, in terms of the "Nor" circuit, can be described by a truth table. Using inputs *A* and *B*, and letting "0" represent ground potential and "1" represent +6.5 volts, then when either one of the inputs is at ground the output remains high. Only when both inputs are high does the output go to ground. When both inputs are at +6.5 volts the output is at ground potential. By cross coupling two of these circuits so that the output of one is the input to the other, a flip-flop results. Negative pulses are used to set and reset the circuit.

System Equipment

RCA *501* systems are tailored to specific applications by adroitly choosing "equipment modules." A brief description of the items from which this choice may be made will serve to indicate the degree of flexibility available in arranging system configurations. For purposes of this description equipment is classed into three major types: Computer, magnetic tape, and input-output.

Computer

The basic computer is composed of the following elements: program control, operating console, paper tape reader, monitor printer, high-speed core storage (16,384 alphanumeric characters minimum), controls (switching and buffering for eight magnetic tape stations), and power supply.

The high-speed storage unit, Fig. 3, starts with a basic building block of 16,384 characters. It contains a stack of planes, 28 deep, with each plane having 64×64 cores to give 4,096 cores in a plane. Since one character consists of 7 bits, this stack of 28 planes represents a 4 character-deep array. Hence, information is called from the storage unit in groups of 28 bits or 4 characters in parallel. The method of using these 4 characters will be described in greater detail in the discussion of the block diagram of the computer. Memory access time, the time for retrieving or storing four characters, is 7.5 microseconds (μsec). A complete memory cycle requires 15 μsec.

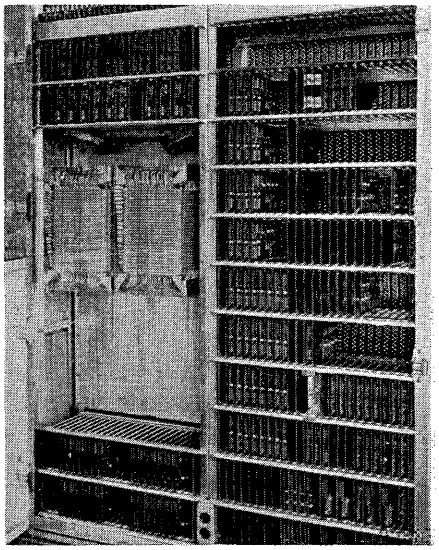


Fig. 3. High-speed memory (32,768-character capacity shown)

The storage unit may be increased in as many as fifteen steps of 16,384 characters to a maximum of 262,144 characters (2^{18}). The addition of a bank of memory in no way affects previously written programs. Further, the only distinction between individual banks are the addresses, which for purposes of programming are considered to be continuous. The unit depicted contains two stacks of memory or a total of 32,768 characters. The same unit can be expanded to a total of 65,536 characters. Four such expanded units may be added to the program control.

The order code consists of 49 instructions covering input-output, data handling, arithmetic, and decision and control. The instruction format, Fig. 4 is as follows: One character is designated the operation code, three characters (18 bits) specify the *A* address; the *N* character specifies the index register to be used as a modifier for either the *A* or *B* address; three characters specify the *B* address. The *N* character is split into two groups of three bits each. The left-hand group specifies how the *A* address is to be modified and the right-hand group specifies how the *B* address is to be modified.

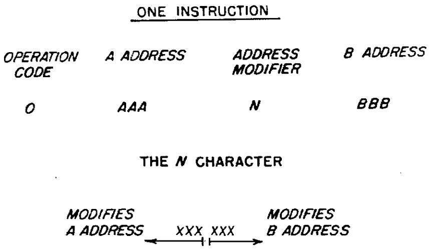


Fig. 4. Instruction format

Particular characteristics of the computer of special interest to programmers will be discussed as a separate topic.

Magnetic Tape

Some specifications of the magnetic tape stations, Fig. 5 serve as an introduction to this topic. Gross data-transfer rate is $33^1/_3$ thousand characters per second, achieved by recording $333^1/_3$ characters per inch and moving tape at the rate of 100 inches per second. The tape is 3/4 inch wide, Mylar base. Reels accommodate up to 2,400 feet of tape. The gaps generated on magnetic tape are 0.35 inch. The tape start time is 3.5 milliseconds. The tape has a lock-out feature which provides for the safety of program and master file tapes. Characters are recorded twice on magnetic tape, substantially increasing reliability.

As many as 63 tape stations may be made available to the computer at one time. Expanding the number of tape stations is done as follows: eight tapes may be connected without any additional equipment. A B-level switching unit is substituted for one of the original eight tape mechanisms. This unit provides for connection of as many as eight additional tape stations. Thus, 15 tapes are available with one B-level switching unit, 22 with two B-level switching units, etc. Two B-level units, together with their power supplies are contained in a single cabinet.

Input-Output Equipment

It has already been mentioned that a 1,000-character per second paper tape reader is available as an on-line device. Similarly, a monitor printer (10 characters per second), with a paper tape punch is used on-line. High-speed output is also available on-line in the form of a 600-line-per-minute, 120-character-wide printer, Fig. 6. This printer uses the computer's electronics for most of the buffering and control functions.

Off-line devices are available which provide for card-to-tape conversion and high-speed printing. The card-feed rate on the card transcriber, Fig. 7 is 400 cards per minute. Characters are written on magnetic tape at a rate of 33,300 characters per second. The input and output hoppers hold up to 1,000 cards for continuous operation. For continuous operation beyond 1,000-card capacity, the cards may be loaded or unloaded during transcription. There is a considerable amount of editing that can be done through the use of the plugboard. As in all RCA *501* equipment, considerable care has been taken to ensure accuracy control. There is a dual sensing station. The card is read twice and the readings are compared. Overpunching which may represent three or more punches in a particular column is sensed and detected. Character parity is checked as is data format. Finally, when information is transferred from the cards to magnetic tape, the echo signal check is used.

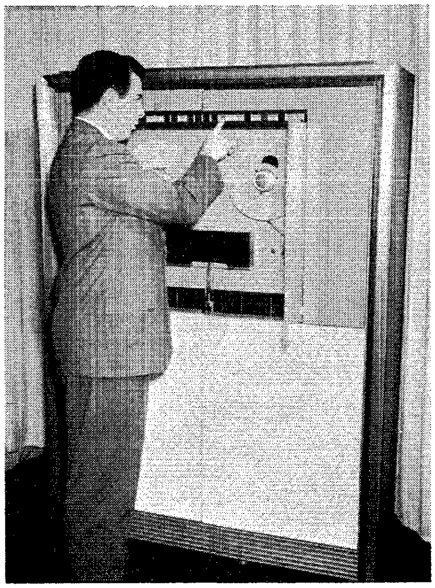


Fig. 5. Tape station

Transcribing card punch, Fig. 8, features are as follows: A maximum of 80 columns can be punched in an 80-column 12-row card. The card punching rate is 150 cards per minute and input-and output-hopper capacity is 1,000 cards each. Again there is extensive editing through the use of the plugboard, and accuracy control features similar to the card transcriber are included.

Off-line printing is accomplished with the Electro-Mechanical Printer, Fig. 6. This device operates with a standard tape station and prints at the rate of 600 lines per minute. Paper skipping can be done at a rate in excess of 70 lines per second. Editing can be accomplished through the use of a plugboard and a paper tape loop. Among the accuracy control provisions for the printer are print-hammer failure check, overlong-line check, tape parity check, warnings for low paper supply, tape station inoperable and printer inoperable.

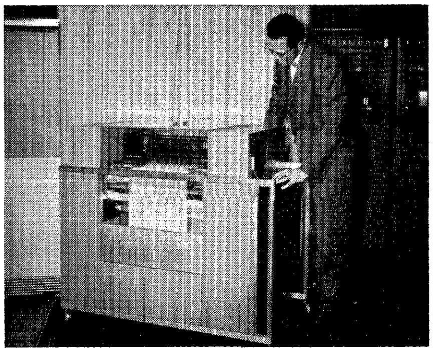


Fig. 6. High-speed printer

Large-scale random access facilities are in the random access file drum, Fig. 9. A file control unit may be connected to any tape trunk and each unit provides for the control of up to 12 drums. A drum has a capacity of 1,500,000 characters and an average access time of 192 milliseconds.

Computer Characteristics

The logic of computer forms the framework of reference for discussion of specific features. The data flow within the computer occurs over a bus system, Fig. 10. Addressing information flows to the memory over the address bus on which are located the P-, A-, B-, T-, and S-registers, and the bus adder. The P-register is the program register and always contains the address of the next instruction. The A- and B-registers correspond to the *A* and *B* addresses of the instruction format. The T-register can be preset by instruction to provide a third address for certain transfer and arithmetic instructions. The S-register is used to address the memory during "interrupt cycles" associated with simultaneous operations. The bus adder is an 18-bit half adder used to modify the contents of the various registers.

A register is made to advance in counter fashion by having its contents transferred to the bus adder, where one is added to the number. The resulting sum is then transferred back to the register. Besides adding to the over-all reliability of the computer, the bus adder also preserves the parity of addresses in accordance with rules independent of those for addition and thus permits a check to be made on all addresses transferred within the machine.

Information flows in groups of four characters from the high-speed memory to the memory register and over the data

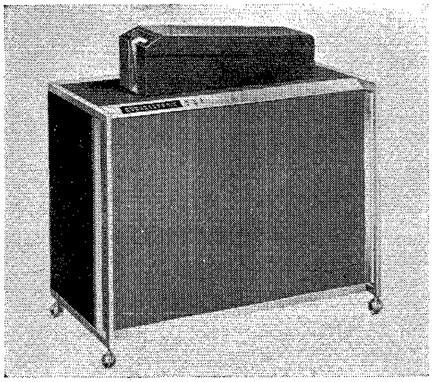


Fig. 7. Card reader

bus into the program control. This same path provides for the flow of data back to the high-speed memory. On the data bus are located the normal operation register, the N-register, used to contain automatic address indicators, and the tape read-write buffers. The interchange is located between the data bus and the single character bus and provides a means of selecting one character from the four on the data bus or transferring a character to one of four paths. By this device, individual characters may be positioned anywhere within a group of four characters comprising a memory word. On the single character bus the L- and R-registers are used to hold an addend character and augend character, respectively. They in turn feed the arithmetic unit. The arithmetic unit returns its results to the adder output register which is connected to the single-character bus. Also on the single-character bus are the SW- and SR-registers which select the write trunk and the read trunk for tape operations.

Many schemes for tape simultaneity were considered for inclusion in the computer. All of the schemes that were considered had their advantages and disadvantages. The one chosen was considered to be the most useful from the programmer's standpoint as well as the least costly in equipment. It is believed that this particular scheme eliminates the need for complex scheduling of special purpose buffer memories, hence, allowing the programmer maximum flexibility.

Simultaneity of tape operation is a demand-type function. The process is initiated when a tape instruction is received from storage. At the time of execution, three tests are made: 1. Is the instruction a type which is "potentially simultaneous?" 2. Is the simultaneous mode free, i.e., not already occupied? 3. Is the program-controlled gate to the simultaneous mode open? If the answer to all three of these questions is positive then the operation code and memory address are transferred to the simultaneous operation register (SOR) and the S-register, respectively, and the computer selects the next instruction sequence. Note that if the answer to any of these questions is negative, the instruction is accomplished in normal fashion.

When a tape operation is being performed in the simultaneous mode, buffering is such as to require the interruption of one 15-μsec memory cycle out of every eight. Thus computation can proceed for 7 cycles before being interrupted. Of course, no interruption is required while tapes are accelerating.

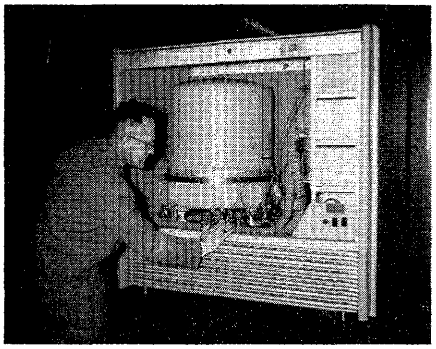


Fig. 9. Random access file

From the programmer's standpoint, it was deemed extremely desirable to include equipment in the computer for utmost flexibility in modifying instructions. For this reason a highly capable set of index registers are incorporated. There are seven index registers which may be used to modify instruction addresses. Of these, four are fixed high-speed memory locations which can be operated upon by any of the normal instructions. In addition, three of the index registers are registers which are dynamically used during the normal course of instruction execution. For example, the program register is also an index register. By this device, programmers may write completely self-relative routines in machine code. Two of the other registers, the A- and T-registers, are used normally for addressing

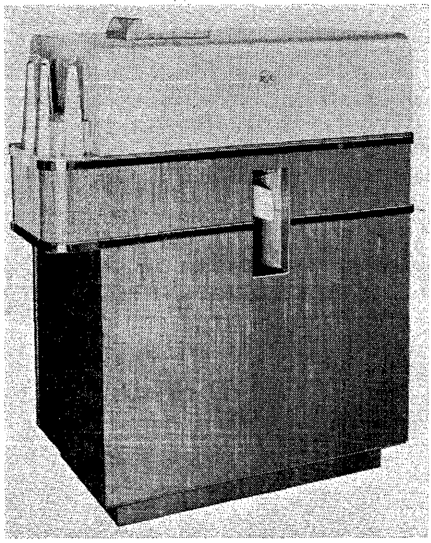


Fig. 8. Card punch

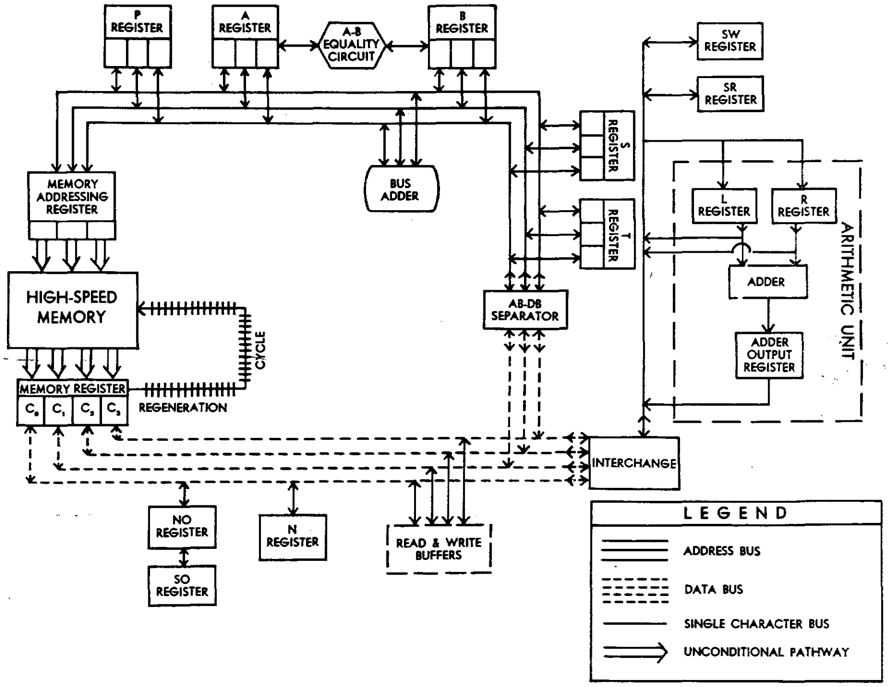


Fig. 10. Computer flow diagram

data in the high-speed memory. Hence, references can be made data-relative, tying-in with the use of variable length operations.

While it is impossible to describe completely the instruction complement at this time, a brief resume is in order. It is natural in a system such as the RCA *501*, which has been designed primarily for data processing, that highly flexible use of magnetic tape be made. Some examples of the facilities available in the *501* for tape use can be cited. First, tape may be read in either direction. Those familiar with sorting will realize that by reading tape in reverse and writing forward on tape, rewind time can be eliminated. A second ability, important in data processing, is the provision for positioning tapes in either a forward or reverse direction by control symbols or by gaps. This last feature permits the scheduling of multiple files on the same reel of tape eliminating both tape costs and operating time. Lastly, data may be stored on tape in either program form or message form. Instructions have been scheduled to offer complete sets of functions for data no matter which of the forms are used.

Data-transfer functions are arranged for either address or symbol control, and either serial or parallel (4 character) operation. In addition, special functions deemed appropriate for data processing problems have been provided. Examples include an instruction which searches consecutive memory locations and stops on the *n*th occurrence of any prescribed character. Another example is an instruction which distributes incoming data to the maximum fields for each item. its converse performs data compression prior to tape write-out.

Arithmetic instructions provide for the four basic decimal functions, in addition to binary and logical functions. The latter two categories are designed to work alphanumeric data as well as program data.

Decision and control functions include the ability to set or store the contents of all of the addressing registers. Conditional jumps can be based on the sign of previous arithmetic results, on tape status, simultaneous availability, program counter exhaust, presence or absence of data, etc. A special tally instruction is included for loop repetition. The instruction decrements a binary quantity, tests for zero, and jumps on non-zero. The jump is ignored for the zero case.

The comments on instruction functions indicated some of the properties available to programmers for flexibility in coding. Another important programming task, that of debugging newly-written routines, has not been ignored. Six breakpoint switches are available on the console which operate in conjunction with bits in the *B* address of the unconditional transfer of control instruction. These bits are used to obtain a "jump or ignore" function. Correspondence with any bit, and an "on" setting of a switch, causes the jump instruction to be ignored.

Availability of a paper tape reader and monitor printer at the computer console facilitates program debugging. These devices are normally operated by any of the magnetic tape instructions having the tape station address of $(77)_8$. A switch on the console is used during debugging to convert all tape station addresses in magnetic tape instructions to $(77)_8$. Thus, the reader and printer may be called without necessitating program modification or, in fact, any provision in programs.

A memory address register is the focal point for the contents of all five register (P, A, B, T, S). Switches have been included on the console which, when used permit stopping at any memory location. Thus, a program may be stopped at a given instruction or whenever the contents of any data location are affected.

Lastly, it has been found useful to establish an audit trail through programs. This function is made possible in the RCA *501* computer. Every jump, either conditional or unconditional is accompanied by the automatic storing of the contents of the program register in a standard memory location, prior to performing the jump. Construction of jump tables is accomplished by transferring the contents of the standard memory locations.

Summary

Many years of experience in engineering and in analysis of data processing requirements have been uniquely combined in the design of the RCA *501* system. Design emphasis has been dictated by the relative sensitivity of the criteria of data processing problems.

Discussion

Edwin Freeman (Remington Rand Univac): Since both the on-line and off-line printers operate at the same speed, it would seem unnecessary to use two printers in the system. Couldn't you use one printer that could be put on-line or off-line, depending on the program?

Mr. Smith: Yes.

H. E. Hortle (Hall Engineering Company): Are all logical functions realized exclusively by "Nor" circuits? If so, how many of these are used?

Mr. Smith: Yes, the logical functions are carried out exclusively by the "Nor" transistor circuits. There are approximately 13,650 of these modules in the computer, not including the high-speed memory.

P. S. Dorn (Sperry Rand Corporation): What is the makeup of the typical system of 17,000 transistors?

Mr. Smith: High-speed memory of 16,384 characters, program control, A-level tape selecting and buffer unit, five tape stations, high-speed on-line printer, monitor printer, and paper tape reader. The A-level selecting unit provides input-output capabilities for the monitor printer, paper tape reader and eight magnetic tape stations. The on-line printer logic is included in the main frame.

W. J. Thielen (Ampex Corporation): What is dual recording?

Mr. Smith: Briefly, each character is recorded on two separate locations on the magnetic tape. There are a total of 16 tracks in line across the tape. Eight tracks provide for a timing track and seven information tracks on one half of the tape. The other eight tracks are the exact duplicates of the first but recorded on the other half of the tape. A series connection of the two magnetic head windings of duplicate tracks supplies the read signal. As little as 40% of normal signal from one track is sufficient for detection by the amplifier. Thus, flaws like oxide holes, oxide nodules, dust particles and the like do not prevent recording or recovering information.

M. A. Powers (Westinghouse Electric Company): What is the approximate additional cost of the duplicated circuitry?

Mr. Smith: Here, I assume the question has to do with the statement that we perform each arithmetic operation twice. The fact is that there are no duplicate arithmetic circuits; we use the same adder twice. The operands pass through the adder the first time to obtain the result. The second time, the complemented operands pass through the adder to obtain the complemented results. The effect we rely on for checking is that a transistor which was "on" the first time must be "off" the second time; and conversely, a transistor which was "off" the first time is "on" the second time. Thus, we have a more powerful check without duplicate circuits than in a two adder system.

If, however, your question relates to duplicate circuitry for dual recording, again no duplicate circuitry is required. Each of the eight channels has one read-write amplifier. For writing on tape, each channel amplifier drives two series windings in the magnetic head to record the information simultaneously on two in-line locations on tape. It is the same for reading. The two windings simultaneously sense the record information at the two locations and supply a single signal for the channel amplifier.

The IBM 7070 Data Processing System

R. W. AVERY S. H. BLACKFORD J. A. McDONNELL

THE INTERNATIONAL Business Machines Corporation (IBM) *7070* is a high-speed solid-state data processing system designed for both commercial and scientific applications. Its versatility and range enables easy expansion from a basic card system to a tape or tape RAMAC system which incorporates the speed and capacity of many large-scale data processing systems.

In both commercial and scientific applications, it is becoming essential that the data processing system be flexible to meet both current and future requirements of the application. The IBM *7070* system has been designed with such flexibility in mind. A wide variety of configurations can be utilized to meet increasing customer needs.

Description of a Typical 7070 System

A typical *7070* system is composed of the machine units shown in Fig. 1. These units are:

Console: This is a separate unit which includes the console typewriter and a small operator's panel. The console unit is designed to simplify and expedite the operator's tasks and insure maximum productive machine time.

The typewriter is the principal operator's tool, and replaces many of the indicator lights and control switches of previous data processing machines.

Operator errors should be minimized by the computer's ability to audit operator commands by a stored program, and by the existence of a printed record from the console typewriter.

Magnetic Tape Units: Two different magnetic tape units are available. The Model *729*II reads or writes tape at a rate of 15,000 characters per second, while the Model *729*IV reads or writes at a rate of up to 62,500 characters per second. Any combination of up to 12 of these units can be employed in the *7070* system.

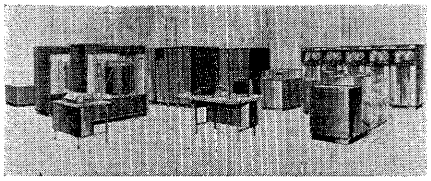


Fig. 1. Typical IBM 7070 system

Each tape unit is attached to one of two independent tape channels. This enables the system to simultaneously read-write compute, read-read compute, or write-write compute.

To insure that the tape record is properly written, an additional set of magnetic reading heads is mounted in the tape drive adjacent to the writing heads, providing an immediate validity check of the written record.

Disk Storage Units: The *7300* disk storage units consist of a magnetic disk storage array with a capacity of six million digits. Each record is 60 words in length, making a total of 600 digits and 60 signs.

Records are read or written by a mechanical access mechanism containing a magnetic recording head. This mechanism moves rapidly to any record in the file. Three of these mechanisms are provided in each disk storage unit to minimize access time by overlapping the operation.

Up to four disk storage units can be utilized in the system, providing a total storage capacity of up to twenty-four million digits. These units are attached to the system by the same two data channels as the magnetic tape units. Each of these data channels is connected to a disk storage unit by program control, thus enabling simultaneous read-write compute, read-read compute, or write-write compute on different disk storage units.

Manual Inquiry Station: The *7900* manual inquiry station permits fast interrogation of the status of data stored in core storage, in a disk storage unit, or on magnetic tape. The station consists of a special typewriter equipped with a solenoid-driven key-board and transmitting contacts. A 16 - channel punched Mylar tape provides format control.

Up to ten manual inquiry stations can be attached to the system, each one separately buffered, and designed for connection to the system by cable up to 2,500 feet long.

Card Reader: The *7500* card reader operates at a rate of 400 cards per minute, with format control by means of a control panel mounted on the reader. Data from a full 80-column punched card may be transferred into the computer.

The card reader is equipped with a front-attended tray-feeding hopper and stacker.

	Bit Code				
Decimal Digit Value	0	1	2	3	6
1	■	■	□	□	□
2	■	□	■	□	□
3	■	□	□	■	□
4	□	■	□	■	□
5	□	□	■	■	□
6	■	□	□	□	■
7	□	■	□	□	■
8	□	□	■	□	■
9	□	□	□	■	■
0	□	■	■	□	□

Fig. 2. Two-out-of-five code

As many as three *7500* card readers can be utilized for card input. Selected cards may be offset in the stacker as desired.

Card Punch: The *7550* card punch operates at a punching speed of 250 cards per minute, with format controlled by control panel wiring. Front-attended hopper and stackers are used. Selected cards may be offset in the stacker.

As many as three *7550* card punches can be utilized for card output.

Printer: The *7400* printer operates at a speed of 150 lines per minute, with format control provided by the control panel. The printed line output consists of a span of 120 characters, spaced ten to the inch.

As many as three *7400* printers can be utilized for printed output.

However, as printed and/or card output, a maximum of three *7550* card punches and *7400* printers can be used in any combination.

Main frame: The main frame of the *7070* system contains most of the system electronics and consists of the following elements:

R. W. AVERY, S. H. BLACKFORD, and J. A. McDONNELL are with International Business Machines Corporation, Endicott, N. Y.

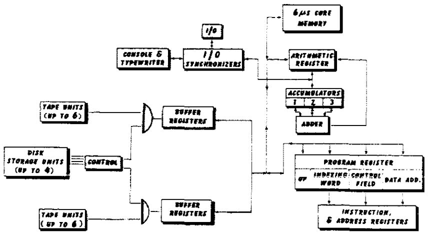


Fig. 3. Major IBM 7070 functional elements

1. Arithmetic registers and core adder.
2. Indexing hardware.
3. Space for optional floating decimal arithmetic.
4. Main core storage consisting of 5,000 or 10,000 words of magnetic cores.
5. Two data channels, code translators, data registers and controls for magnetic tape and disk storage units.
6. Buffers and controls for card input/output and printer.

Machine Characteristics of the 7070 System

Storage address and arithmetic registers are of fixed length of ten digits. However, the execution times of arithmetic operations and data transfers are completely variable.

Digit Code: A word in machine code is composed of 55 bits consisting of ten digits plus sign, each digit of which is represented by five bits in a two-out-of-five code. Fig. 2 illustrates this code.

Modes of Data Transmission: Parallel modes of data transmission include data transmitted to or from core storage via:

1. The program register.
2. The arithmetic register.
3. The tape synchronizing transmission registers, and
4. The auxiliary registers.

Serial modes of data transfer include data moved to and from the tape units. Accumulators, arithmetic registers, and auxiliary registers all have serial paths connecting them to and from the core adder. There is also a serial data path to and from the input/output synchronizers.

Rates of Data Transmission: The *7070* system operates at a 250-kc digit rate, with data transmitted to or from core storage at a 6-microsecond (μsec) rate. Data is transmitted to or from the core shift registers within the main frame at a 4-μsec rate.

On the disk storage unit, access time varies from a minimum of approximately 105 milliseconds (track to adjacent track of the same disk) up to approximately 850 milliseconds (track to track of different disks).

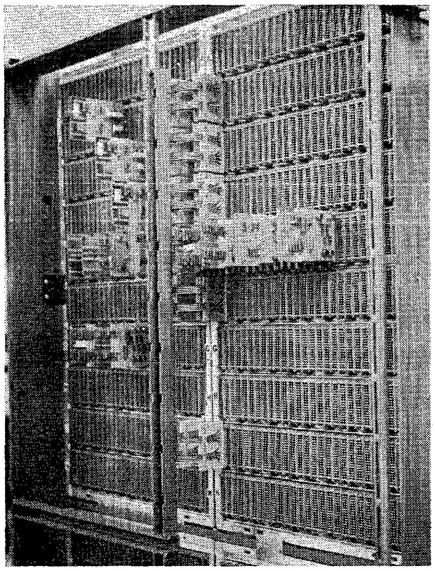


Fig. 4. One of the sliding gate cabinets housing the system electronics

Instruction Format: All operations performed by the *7070* are controlled by numerically coded instructions consisting of 10 decimal digits and sign.

The instruction format is shown in Table I.

Internal Organization of the 7070 System

Fig. 3 illustrates the major functional elements in the machine organization.

Storage: Main Storage consists of 5,000 or 10,000 words of magnetic cores operating on a 6-μsec cycle.

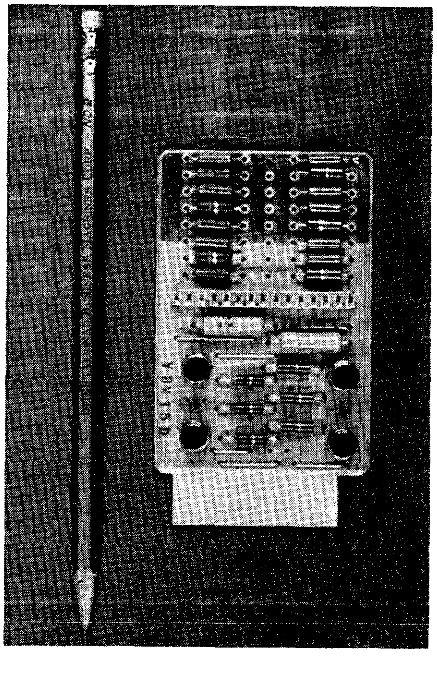


Fig. 5. Sample IBM 7070 printed circuit card

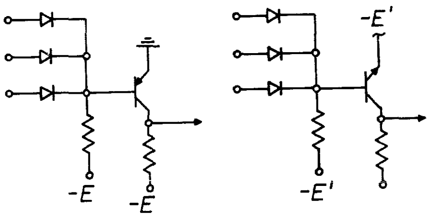


Fig. 6. CTDL transistor circuits

The 55 bits of a word are read into or out of core storage in parallel under control of priority circuits. These parallel data channels interconnect all of the major elements of the system, including arithmetic registers, instruction register, control registers, and input/output registers.

Arithmetic Registers: Arithmetic is performed by three registers with the adder and its controls. A sum is obtained by adding the contents of any two registers serially by digit, with the sum transferred to the specific arithmetic register.

In multiplication, the multiplicand is stored in accumulator 3, the multiplier in accumulator 2, and the product is generated in accumulators 1 and 2.

In division, the divisor is stored in accumulator 3, the dividend in accumulators 1 and 2, and the quotient is transferred to accumulator 2. The remainder is located in accumulator 1.

Program Registers: The program section contains the instruction register, address register, and program register and an arithmetic register (not shown).

Indexing makes use of the main adder and controls.

Up to 99 index words stored in core memory are available for automatic indexing during instruction interpretation.

Magnetic Tape Control: The two magnetic tape channels are buffered by time-sharing main core storage through a tape buffer register of ten digits. When this register is filled, its contants are transferred in parallel into core storage at a location specified by a special control feature called scatter read-write. The scatter read-write feature provides, by means of a sequence of stored control words, the format in memory of the record read from tape. This feature is also used in other input/output operations, including disk storage operations.

In a data processing system containing a multiplicity of input/output devices, there has been a need to provide an effective means of running two or more applications simultaneously. The *7070* solves this problem by a mode of operation known as "automatic priority processing" in which a given input/output device interrupts the sequence of the main routine, temporarily stores machine status information, and causes a program branch to the new routine.

After completion of priority processing routine, the machine returns to the original program.

This mode of operation is provided for in the control of magnetic tape as well as all other input/output devices.

Disk Storage Control: Control of the data format of disk storage records is very similar to the format employed in magnetic tape. Scatter read-write and priority processing are employed.

Card Input/Output and Printer Synchronizers: Each card input/output device is attached by means of a separate synchronizer to permit operation of all these units in parallel with computing. All synchronizers are connected to core storage by a parallel data channel to permit scatter read-write and priority processing functions.

Physical and Electrical Characteristics of the 7070 System

Printed Circuit Cards: The system electronics are housed in a group of sliding-gate cabinets, one of which is illustrated in Fig. 4. These cabinets are designed to accommodate printed circuit cards of a type shown in Fig. 5. These printed cards, which mount up to six transistors, are designed for automated fabrication. The number of different card types utilized is held to a minimum. Approximately 14,000 cards are employed in the *7070* system shown in Fig. 1. A total of 30,000 alloy-junction germanium transistors, and 22,000 germanium diodes, are used in this system.

Chassis intercard signal wiring is provided by jumper wires which are connected to the card socket by wire-wrap connections. This wiring is designed to be accomplished automatically by wire-wrap machinery.

Distribution of power supply voltages to the transistor cards is accomplished by printed circuit strips. This enables a major portion of the electronic system to be fabricated by automatic equipment.

CTDL Transistor Circuits: The logic and control sections of the *7070* utilize complementary-transistor diode logic (CTDL) to provide economical logic circuitry for the system. These circuits are illustrated in Fig. 6. The basic building blocks can be described as using two circuits in each system; each block comprising a p-n-p (N-block) inverter and n-p-n (P-block) inverter. Each inverter has an input circuit that in combination controls the "on" and "off" states of the transistor and, therefore, performs the logic. In addition, outputs can be wired so that logical switching functions are also performed by transistors. Thus, two logical operations may be performed in a single stage with various block connections.

Table I. Instruction Format

S	01	23	45	6789
Sign	Op	IW	C	D

Position S	Contains the sign: 6 minus; 9 plus.
Positions 0–1	Contain the operation code.
Positions 2–3	Contain the indexing word designation. Positions 2–3 of the instruction word specify the indexing word to be used with the instruction. Value 00 indicates no indexing; values 01–99 are directly related to core memory words 0001–0099.
Positions 4–5	Contain the control information, field definition, index word to be operated on, or an extension of the operation code. Position 4 specifies the starting position of the field (high-order position of the field). Position 5 specifies the last position of the field (low-order position of the field). The field specified may not extend beyond the limits of one word to the adjoining word. If the start position is greater than the last position, an error stop occurs. The sign of the specified field is the sign of the word which contains the field.
Positions 6–9	Contain the data address (address of the operand), control information, or branch address.

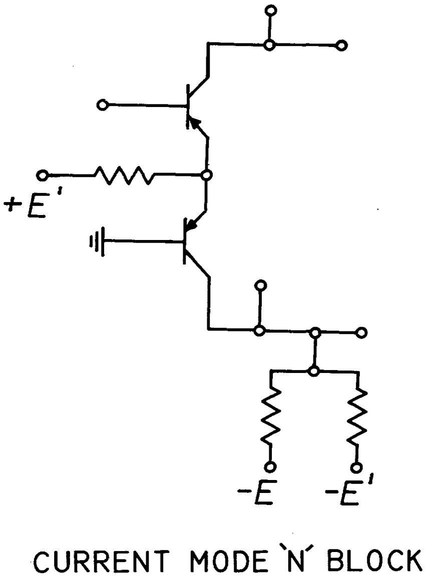


Fig. 7. Typical current mode transistor circuit

Current Mode Circuits: Current mode transistor circuitry is used in the timing storage and core storage portion of the *7070* system. Current mode logic provides a building block in which the flow of sufficient current rather than the voltage levels affects the logical operation. Here, the transistor is considered as a current amplifier. Thus, when X milliamperes of a base current flows, KX milliamperes of collector current will flow, K being the current gain of the transistor. These circuits are employed to represent a "logical 1" when they do not supply base current, and a "logical 0" when they do supply base current. A typical circuit is illustrated in Fig. 7.

Magnetic Core Registers and Translators: Magnetic devices are used in several *7070* units in addition to the core memory.

These units are:

1. Arithmetic registers.
2. Adder.
3. Code Translators.
4. Validity check circuits.

The registers utilize magnetic core shifting registers.

The adder and code translators are coincident current core arrays in the form of truth tables.

Summary and Conclusions

The IBM *7070* system offers facilities required for complete data processing according to the most modern concepts. In addition, by utilizing recent developments in transistor and magnetic technology, the *7070* provides these functions at a minimum cost.

An equally new and modern approach to the manufacture of data processing machines is an important part of this product.

Discussion

H. P. Peterson (Lincoln Laboratory, Massachusetts Institute of Technology): Is arithmetic done in the 2-out-of-5 code?

Mr. Avery: Arithmetic is done in the 2-out-of-5 code using a serial by digit adder.

Mr. Peterson: How long is a full carry time?

Mr. Avery: 4 microseconds.

T. J. Beatson (General Electric Company): How are alphabetic or non-numeric data handled in the 2-out-of-5 code structure?

Mr. Avery: Alphabetic data are represented in the machine by two numeric digits. Alphabetic input-output is automatically translated to and from this code representation.

Alphabetic words in memory are identified by a special symbol in the sign position.

F. B. Pace (Bell Telephone Laboratories): How many different modular types are used in the IBM *7070?*

Mr. Avery: One printed-circuit card design is used for all circuits. About 10 different printed patterns are used. A total of 73 different circuits may be achieved by the manner in which circuit components are mounted on the card.

Mr. Ottollanch: When will IBM make the first commercial installation?

Mr. Avery: The first commercial installation will be in the Spring of 1960.

Mr. Ottollanch: What are add, multiply, and divide time for minimum and full system?

Mr. Avery: Arithmetic times are not dependent upon system size but are dependent upon field size of the record.

A 5-digit add requires 60 microseconds.

Multiply time in microseconds is: 192+ 48. (No. of 1's, 2's and 4's in multiplier) + 2 (No. of 3's, 5's, 6's, and 8's in multiplier) + 3 (No. of 7's and 9's in multiplier) + (No. of zero groups in multiplier)

Divide time in microseconds is: 264 + 48 10 + quotient digits.

H. F. Sherwood (Touche, Niven, Bailey & Smart): Is use of a high-speed printer planned with the *7070* system, and if so, what type?

Mr. Avery: The type *720A* printer with the *760* buffer and *727* tape unit may be used off-line with the *7070*.

The system is designed to be compatible with new IBM printers when available.

W. R. Haber: In the card punch machine, what is the maximum number of holes allowable for punching?

Mr. Avery: A maximum of 80 columns may be punched in each card. Each column may contain all allowable numeric, alphabetic, and special characters together with special control punches in the zone portion of the card.

J. H. Waite (Radio Corporation of America): What are some representative times for matrix inversion?

Mr. Avery: Information on matrix inversion times will become available at a later date.

J. H. Waite: What are program running times for sine, arctan subroutines?

Mr. Avery:

Sine routine:	7.7 milliseconds minimum
	10.3 average
	12.9 maximum
Arctan routine:	9.9 minimum
	14.1 average
	17.2 maximum

Performance Advances in a Transistorized Computer System: The TRANSAC S-2000

R. J. SEGAL J. L. MADDOX P. PLANO

THE THEME of this 1958 Eastern Joint Computer Conference is "Modern Computers." It is significant that all of the new computers described at this conference are transistor machines. In order for a particular transistor machine to be properly described as a "modern computer" it should satisfy the following three criteria:

1. Performance of the transistor machine should be notably better than currently available vacuum-tube machines in the same class. An increase in the speed of arithmetic operation by a factor of 3 (or more) to 1 is an example of this type of performance advance.

2. Performance characteristics should be obtained from a working model of the transistor machine. More than one finely conceived system has been unable to overcome the difficult technical and economic obstacles between the initial concept and its realization in the form of hardware.

3. The existence of a single working model still does not imply the ability to produce many machines with the same performance characteristics. The design specifications, the equipment components, and the production organization must make it possible to produce and deliver many machines on schedule before a particular "modern computer" can be said to truly exist.

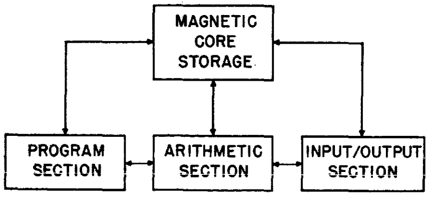


Fig. 1. Magnetic-core storage unit is time shared by other sections of the S-2000 system

In the field of large-scale data processing systems the TRANSAC *S-2000* is a "modern computer" which meets the criteria just outlined. The first *S-2000* system was delivered to the field in November 1958. The Philco production organization is now producing both transistors and additional *S-2000* systems on schedule. As of December 1958, three systems of the *S-2000* class exist in the form of hardware and additional systems are in production. The performance advances of the *S-2000* stem from the versatility of the computing sections, the particular organization of the input-output system with relation to the computing and memory sections, and the new and sophisticated implementation of the *S-2000* logic by mass-produced units of hardware which incorporate the transistor as the active logical element.

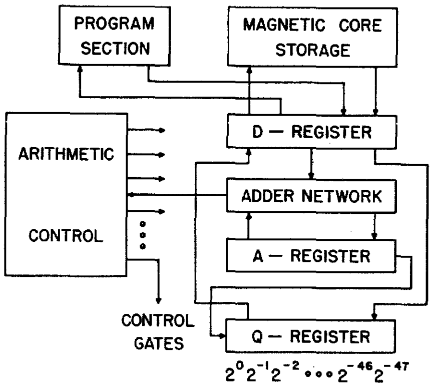


Fig. 2. Arithmetic section block diagram, fixed point

S-2000 System Organization

The TRANSAC *S-2000* organization centers around a magnetic-core storage unit. To the memory are connected the other three essential computer sections:

R. J. SEGAL, J. L. MADDOX, and P. PLANO are with Philco Corporation, Philadelphia, Pa.

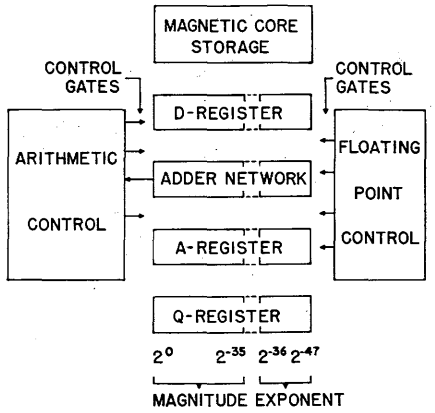


Fig. 3. Arithmetic section block diagram, floating point

the arithmetic section, the input-output section, and the program or control section, as illustrated in Fig. 1. In effect, the magnetic core storage is time-shared by the other three sections.

The capacity of the magnetic core storage unit may be expanded from 4,096 to 32,768 words depending on the requirements of a particular installation. The word length is 48 binary bits. The basic memory module contains 4,096 words. It operates on a coincident current arrangement giving a complete read and write cycle of 10 microseconds (μsec).

The arithmetic section can perform both fixed- and floating-point arithmetic. Two's complement, fractional arithmetic, is performed in both fixed- and floating-point arithmetic. When performing fixed-point arithmetic, the arithmetic section is arranged as shown in Fig. 2. Operands results are transmitted between the magnetic-core storage and the arithmetic section via the D-register which serves both as an operand register for the arithmetic section and as a buffer for the magnetic core storage. The A-register is an accumulator and the Q-register handles the multiplier-quotient operands. The arithmetic control sequences the gating operations with an asynchronous timing device.

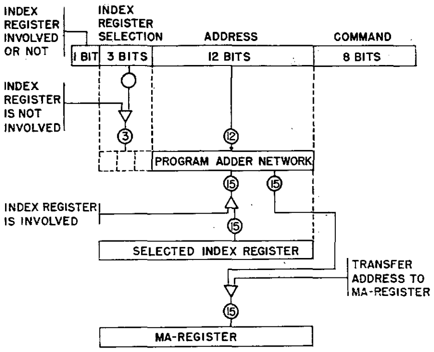


Fig. 5. Computer instruction word format and use of index registers

When performing floating-point arithmetic, the logic of the arithmetic section is arranged as shown in Fig. 3. Functionally, the floating-point arithmetic section is divided into two parts. Bits 2^0 through 2^{-35} (36 bits) form the magnitude portion of the word while bits 2^{-36} through 2^{-47} (12 bits) form the exponent portion of the word. With the exception of pre- and post-normalization, the magnitude portion is operated on in the same manner as in fixed point utilizing the same control. The floating-point control, however, also comes into play and performs the necessary operations on the exponent portion of the word. Table I lists some arithmetic operations and their speeds.

A block diagram of the program section is shown in Fig. 4. The program register, which has a word-length of 48 bits, contains two single-address instructions.

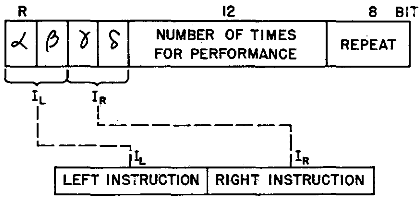


Fig. 7. Repeat instruction format

The program register receives instructions directly from the core storage, hence it also acts as a buffer for the core storage. The remaining registers of the program section (jump address register, program address register, memory address register, and index registers) each have a word length compatible with the magnetic-core storage capacity. For example, the registers are 12 bits long for 4,096 words of core storage, 13 bits for 8,192, etc. The program address register is responsible for sequencing the instructions of the program. The contents of the PA-register behave as a counter by employing the program adder network. The MA-register acts as an accumulator for the program adder network, with the principal responsibility of furnishing addressing information to the core storage selection network. For example, the address portion of the instruction being performed (contained in the PR-register) is transferred to the MA-register which selects the address of the operand or storage location required by the instruction.

The function of the jump address register is to store the return address for a subroutine. The return address locates the next consecutive instruction following the jump instruction to a subroutine. This enables the program to remember where to jump back in order to continue following the subroutine execution.

The index registers may operate in several modes. The most common mode, however, is to add the contents of the selected index register to the address portion of the instruction being performed, using the sum as the effective

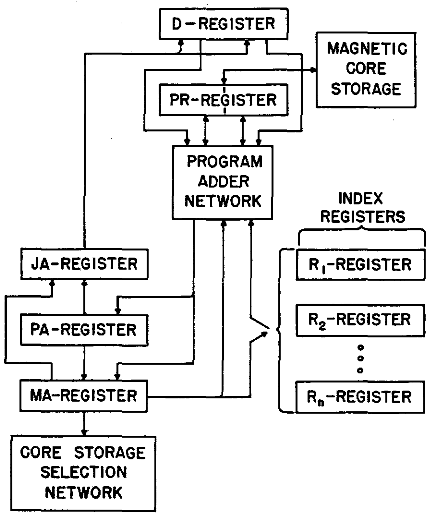


Fig. 4. Block diagram of program section

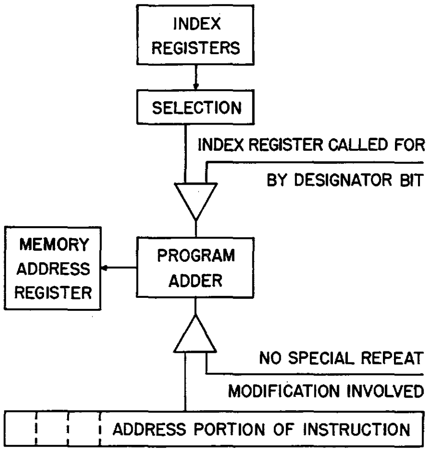


Fig. 6. Use of index registers in obtaining memory address

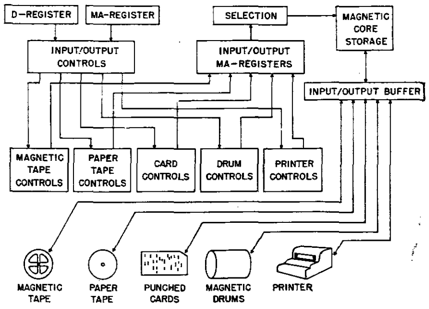


Fig. 8. Input-output system block diagram

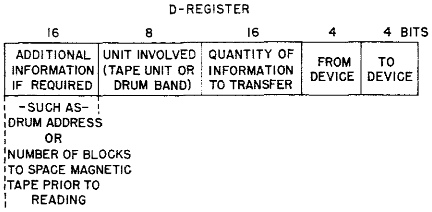

D-REGISTER
16
8
16
4
4 BITS
ADDITIONAL INFORMATION IF REQUIRED
UNIT INVOLVED (TAPE UNIT OR DRUM BAND)
QUANTITY OF INFORMATION TO TRANSFER
FROM DEVICE
TO DEVICE
-SUCH AS-
DRUM ADDRESS
OR
NUMBER OF BLOCKS TO SPACE MAGNETIC TAPE PRIOR TO READING

Fig. 9. Input-output instruction word format

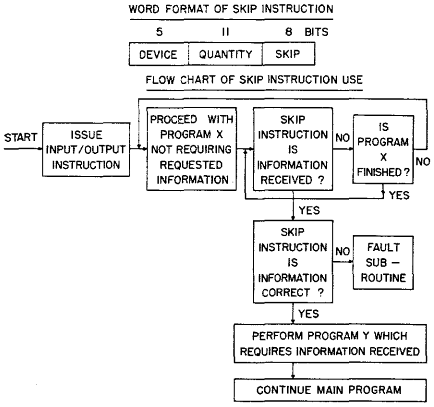

WORD FORMAT OF SKIP INSTRUCTION
5
11
8 BITS
DEVICE
QUANTITY
SKIP
FLOW CHART OF SKIP INSTRUCTION USE
START
ISSUE INPUT/OUTPUT INSTRUCTION
PROCEED WITH PROGRAM X NOT REQUIRING REQUESTED INFORMATION
SKIP INSTRUCTION IS INFORMATION RECEIVED ?
NO
IS PROGRAM X FINISHED ?
NO
YES
YES
SKIP INSTRUCTION IS INFORMATION CORRECT ?
NO
FAULT SUB - ROUTINE
YES
PERFORM PROGRAM Y WHICH REQUIRES INFORMATION RECEIVED
CONTINUE MAIN PROGRAM

Fig. 10. Skip instruction flow chart and word format

address of the instruction. An extra bit (flip-flop) which is assigned to each index register determines whether the index register is to behave as a counter or not. This extra bit is set by a special computer instruction, and remains until the special instruction resets it. If the index register is to behave as a counter, its contents are increased by one after being employed to modify the address of an instruction. The contents of the index register are increased by one by passing through the program adder network. Other operations of the index register and the JA-register will be explained in the section concerning instructions. An *S-2000* system may have up to 32 index registers.

Computer Instructions

The computer instruction format for a system consisting of 8 index registers and 32,768 words of magnetic core storage is illustrated in Fig. 5. The command portion which consists of 8 bits is located at the extreme right. The address portion of a computer instruction consists of 16 bits to the extreme left. The bit on the extreme left designates whether an index register is involved or not. The next 3 consecutive bits specify which index register is involved. If no index register is involved, these 3 bits are then transferred along with the remaining 12 to the MA-register to select a magnetic core storage address.

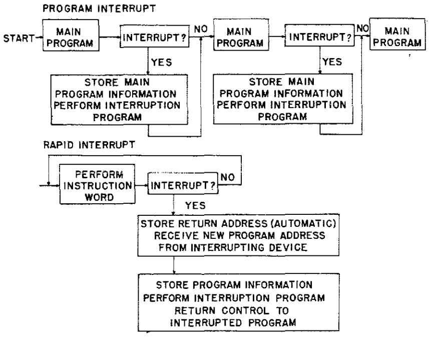

PROGRAM INTERRUPT
START
MAIN PROGRAM
INTERRUPT?
NO
MAIN PROGRAM
INTERRUPT?
NO
MAIN PROGRAM
YES
YES
STORE MAIN PROGRAM INFORMATION PERFORM INTERRUPTION PROGRAM
STORE MAIN PROGRAM INFORMATION PERFORM INTERRUPTION PROGRAM
RAPID INTERRUPT
PERFORM INSTRUCTION WORD
INTERRUPT?
NO
YES
STORE RETURN ADDRESS (AUTOMATIC) RECEIVE NEW PROGRAM ADDRESS FROM INTERRUPTING DEVICE
STORE PROGRAM INFORMATION PERFORM INTERRUPTION PROGRAM RETURN CONTROL TO INTERRUPTED PROGRAM

Fig. 11. Flow chart for interrupt techniques

The instruction catalogue of TRANSAC *S-2000* contains 227 instructions. These are divided into the following types of operation:

Arithmetic and shifts	129 Instructions
Transfers of data	16 Instructions
Transfers of control (jumps)	38 Instructions
Index register operations	20 Instructions
Logical operations	8 Instructions
Special operations	16 Instructions
Total:	227

Among the several instructions in the "special" class is the repeat instruction; it can use index registers in a special way. Fig. 6 shows how an effective memory address can be made from the index register contents alone, if a special repeat modification is used. Fig. 7 illustrates the format of the repeat instruction. The bits labeled α and β refer to the left-hand instruction following (if it exists) and γ and δ refer to the right-hand instruction following the repeat instuction.

If the repeat instruction itself is in a left-half word, only the right-half word following is repeated; if it is in a right-half word, then the two instructions in the next computer word are performed as a pair, the number of times designated by the 12-bit address portion of the repeat instruction itself.

If the α bit which refers to the left instruction following is zero, β is ignored. If the α bit is one, then the contents of the index register designated by that instruction are used alone for the effective memory address of the instruction operand. If $\alpha=1$ and β is zero, the contents of the specified index register are incremented by the contents of the 12-bit address portion of the instruction after its execution.

Table I. Typical Speeds of Arithmetic Operations, TRANSAC *S-2000*

Operation	Fixed Point (μsec)	Floating Point (μsec)
Add-Subtract	15	22
Multiply	65	55
Divide	75	70
Divide and store	78	73
Multiply and add	70	65

Note: These speeds include one operand access and the access of the instruction. Times not under repeat.

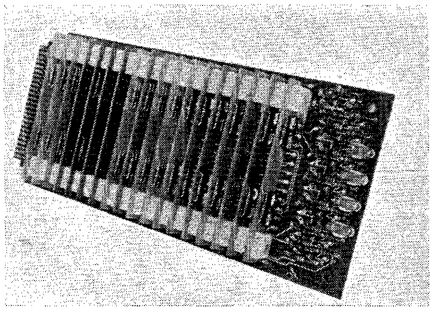


Fig. 12. Typical TRANSAC card

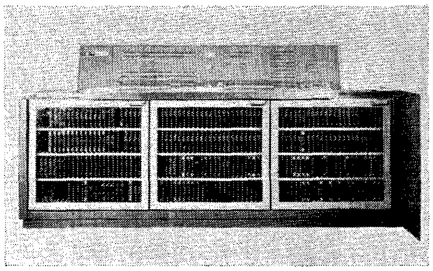


Fig. 13. Modular construction of computer and control unit

If, however, α and β are both one, the contents of the index register specified by the left instruction is decremented by the contents of the 12-bit address following instruction execution. Similarly, γ and δ refer to the right instruction in the same manner as γ and δ refer to the left instruction.

An unusual pair in the instruction catalogue is "larger word," "smaller word." The instruction "larger word" brings the contents of the designated core storage location to the D-register. If the contents of the D-register are greater (in an alphanumeric sense) than the present contents of the A-register, the contents of D are transferred to A and the location of this information is transferred from the memory address register to the JA-register. When this instruction is used in repeat mode, the largest element of a list may be found with a single instruction word.

When the location of the largest element has been stored in the JA-register, it is also accessible for program use. The instruction "smaller word" is used in order to select the smallest element of a particular list.

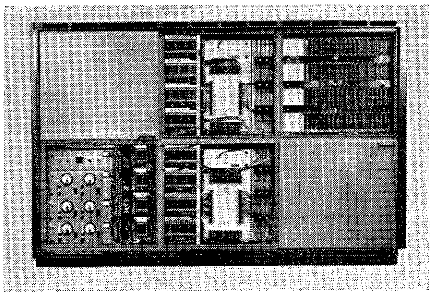


Fig. 14. An 8,192-word magnetic-core storage unit has a volume of 55 cubic feet

Input-Output Section

Input-output equipment available for an *S-2000* system installation includes the units in Table II.

Table II. Input-Output Devices and Speeds

Input-Output Device	Operating Speed
Paper tape reader........	1,000 characters per second
Paper tape punch........	60 characters per second
Magnetic drum..........	491,520 characters per second
High-speed printer........	900 lines per minute
Magnetic tape unit (No... sequencing)	90,000 characters per second
Magnetic tape unit... (Multiple sequencing, 4 units)	360,000 characters per second

The paper tape, magnetic drum, and magnetic tape systems operate in conjunction with their own control units.

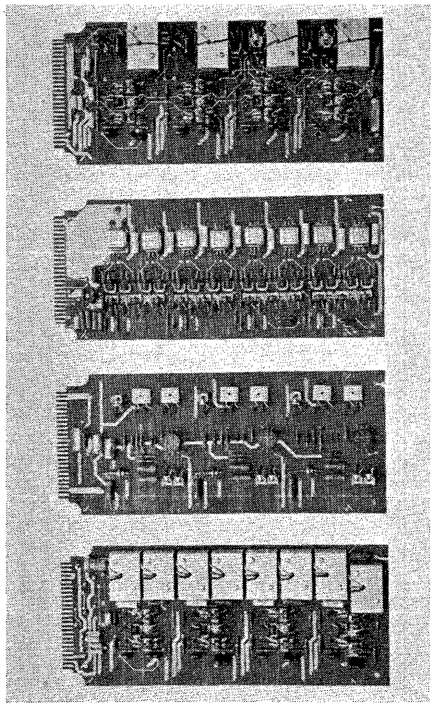
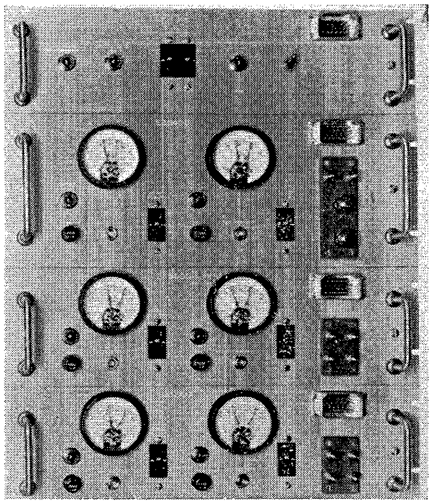


Fig. 15. Memory circuits are packaged on four card types



Courtesy Power Designs, Inc.

Fig. 16. TRANSAC power supplies are completely transistorized

In the case of magnetic tape, up to four tapes may be operated simultaneously through the magnetic-tape multiple-sequencing unit. Each tape is controlled by its own assembly unit. Any assembly unit may be used with any tape transport in the system as selected by the input-output instruction.

Accuracy of reading and recording is ensured by the use of separate read and write heads (to read and check the recording immediately) and parity checks. Every character on tape has a parity bit and every block has two parity characters for horizontal as well as vertical checking. To ensure that recorded data are preserved, when so desired, a physical snap ring is provided with each tape reel. Recording cannot occur without the snap ring and the reel cannot be stored with the ring inserted. In either case data on the tape may be read.

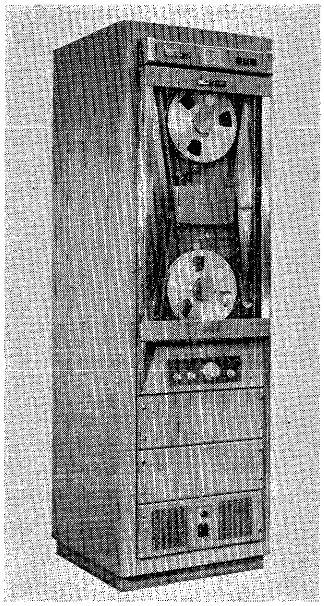


Fig. 17. Magnetic-tape unit

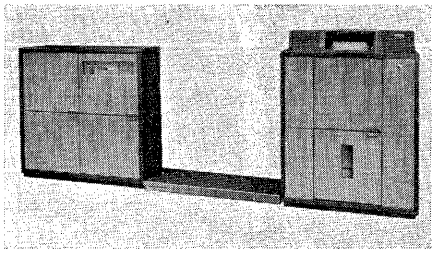


Fig. 18. High-speed printer system

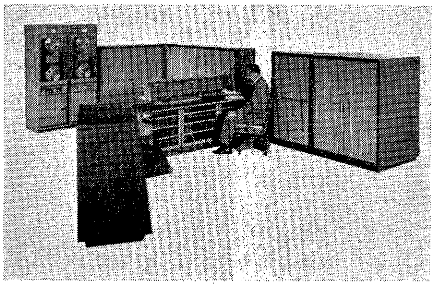


Fig. 19. Typical S-2000 system

Information written on magnetic tape may be binary-coded decimal information, pure binary information, or a combination of both. Regardless of the data form, the central computer accepts six binary digits, plus a parity bit, as a "character." Because all numeric data may be recorded and read in pure binary form, tape and time savings on the order of 50% will be realized. (A 7-place decimal number (2,097,152) in binary-coded decimal form may need as many as 42 bits, and in pure binary form needs 21 bits.)

Up to 256 magnetic tape units may be employed in one system, and every 16 require a magnetic-tape control unit. The magnetic-tape control unit permits reading or writing on one tape while the central computer is processing other data.

Multiple tape sequencing provides simultaneous reading, writing, and computing. In multiple tape sequencing under control of a magnetic-tape control unit, up to four tape units may be operated concurrently while the central computer is processing other data.

One of the important features of the TRANSAC *S-2000* input-output operation is the time sharing of the core storage by the central computer and the various input-output devices. Fig. 8 is a block diagram of the input-output system. Top priority for access to the magnetic-core storage goes to the magnetic tape assembly units. By means of these assembly units and the priorities

Table III. Electrical Component Count for a Typical *S-2000* System (Critical Parts)

S-2000 Units	Card Blanks	Module Blanks	Transistors	Resistors	Capacitors	Diodes	Neons	Transformers	Connectors	Totals
Computer and control unit	260	4,717	19,297	27,125	12,887	159	779		338	
Floating-point option	17	308	1,232	1,489	923		6		0	
Input-output option	18	259	863	1,861	549	6	25		0	
Index register option (8 index registers)	22	554	2,286	3,226	1,563		40		80	
Subtotals (Computer and control unit)	317	5,838	23,678	33,701	15,922	165	850		418	80,889
Magnetic-tape control	125	2,172	8,221	11,705	5,854	39			306	
Paper-tape control	23	327	1,396	2,292	992	53	78		38	
Magnetic-core storage	170	650	5,512	11,936	3,145	339		612	253	
Printer control	11	187	753	1,177	519	11	38		90	
Drum control	21	360	2,048	3,600	1,621	31	24	48	71	
Totals	667	9,534	41,610	64,411	28,053	638	990	660	1,176	147,739

established within the magnetic tape system, up to four magnetic tapes may be operated simultaneously. The central computer assigns the core storage by a priority system such that if magnetic tape does not require immediate memory assignment, slower devices such as cards, paper tape, or the printer are assigned the core storage if it is requested. If no input-output device requires core storage access, the central computer is assigned. Hence, the execution of an instruction such as multiplication, division, or floating-point arithmetic can go on simultaneously with the reception or transmission of information.

Input-Output Instructions

A single computer instruction indicates that an input-output device is involved. In this case, the specific input-output instruction is sensed in the D-register. An input-output instruction word consists of 48 bits as shown in Fig. 9. The command portion of the instruction is in the right eight bits. The number of bits actually used to designate quantity of information depends on the requirements of the particular equipment.

In order to make efficient use of memory sharing, two special computer instructions exist. The first of these is a skip instruction which indicates to the computer whether a specified quantity of information has been received. The skip instruction has several variations depending on the device involved. Word format of the skip instruction is shown in Fig. 10. The left five bits of the skip instruction specify the device involved.

The first variation of the skip instruction involves magnetic tape. In this variation the question is whether there are X or more words left to process. This number is placed in the remaining 11 bits of the instruction address and is compared with the counter of the magnetic-tape assembly unit connected to the tape unit specified in the device portion of the instruction. If the processing has not reached the questioned point, the computer sequences to the next instruction which is always a transfer of control instruction. If the processing has proceeded to the questioned point, the computer skips over the next instruction and executes the routine which depends on the reception or transmission of this data.

The second skip instruction variation involves the paper tape, printer, or cards. In this case, the question is simply, "Is the device actively processing information?" If it is, control passes to the next instruction which is a transfer of control. If the equipment is no longer active, the skip over the next instruction takes place. In the case of a device, such as paper tape, which halts on a stop character, another instruction of this class can then check on whether the device finished because of the quantity of information received or because of the appearance of a stop character.

The third variation is used for program interrupt. This will be explained in the next section.

The second special computer instruction is a skip which checks that the information received is correct as far as can be determined by the various error-detecting features of the input-output equipments. This instruction has two variations. For devices other than magnetic tape, the question is simply, "Did a fault occur?" For magnetic tape there are a number of types of faults recognized. With the use of a proper succession of "skip if magnetic tape fault" instructions, it is possible to isolate which of the errors occurring was most serious.

TRANSAC *S-2000* has two interrupt features. Fig. 11 illustrates these interrupt techniques. The first is a program interrupt which is accomplished by instructions within the program itself. Therefore, the interruption is done at the convenience of the current program. It also means that queueing of the interrupting inputs is determined by the computer program. Hence, any desired scheduling system may be employed. The instruction which is used, "skip the next instruction if the specified device desires to interrupt," as was mentioned previously.

The second feature is a rapid interrupt. The signal from the interrupting device interrupts the program at the completion of the instruction word being performed. In this case the queueing is determined by the interrupting devices. It is possible by this method to have a series of interruptions such that a higher priority device may interrupt a lower priority program and each program can be completed as the higher priority program has finished.

The TRANSAC Assembler-Compiler

The basic programming system for the *S-2000* is the TRANSAC Assembler-Compiler (TAC). The TAC system includes many of the features which have been proved in a number of programming systems on other computers. TAC will serve as the basis for a number of completely automatic programming systems. The basic language of the TAC system is a language of decimal numbers and alphabetic mnemonic instructions. By using TAC, the programmer may, in general, regard the TRANSAC *S-2000* as a decimal computer. The TAC system provides for one-to-one translation of mnemonic pseudocodes into machine codes. Thus, it is possible to make use of all of the features and all of the flexibility of the computer. However, TAC also permits reference to previously coded macro-instructions and subroutines. These are treated just as if they were ordinary machine codes so that in effect, it is possible to expand the instruction catalogue to include almost any function that is desired by the programmer.

S-2000 Hardware and Packaging

The modern transistorized data-processing system inherently offers many hardware advantages to both the producer and user of such equipment. Some of these advantages are:

1. Transistorized hardware is most amenable to mass production techniques. More efficient production and quality control operations may be employed than in the case of vacuum-tube equipment.

2. Transistorized hardware offers significant increases in speeds of operation and in equipment reliability over vacuum-tube systems. Both of these advances follow directly from utilization of the transistor rather than the vacuum tube.

3. Transistorized hardware costs less to install and operate than vacuum-tube hardware. The TRANSAC *S-2000* computer, for example, requires less than 1/5 the power and 1/5 the space of a comparable vacuum-tube computer. Air conditioning requirements are small and in some modern installations no additional air conditioning is required. Signal and power connections are built into *S-2000* equipment, eliminating the need for false floors or ducting of floors.

4. The compactness and modular construction of transistorized hardware as executed in *S-2000* equipment makes it simple to add optional features, or to expand the capacity of an *S-2000* system at any time after installation without the need for rewiring or modification of any of the equipment in use. Wiring to accommodate system expansion is designed into the basic system. Additional building blocks or optional features plug into the basic equipment, virtually eliminating down time for system changes.

S-2000 Production Techniques

A component count for a typical *S-2000* system is given in Table III. Successful production of modern transistorized computers is entirely dependent on the availability of production quantities of reliable switching transistors of uniform predictable quality. The surface-barrier transistor, a Philco development, has been produced by the millions since 1953 by the Philco Corporation at its Lansdale Tube Company plant. The world's most completely automated transistor production facility insures that *S-2000*-production lines will have a constant flow of transistors of uniform and carefully controlled quality. The undesirable process of transistor selection within narrow limits is not required in *S-2000* production.

S-2000 circuits are packed on TRANSAC cards. Fig. 12 shows a typical TRANSAC card. Each TRANSAC card mounts up to 80 transistors, together with their related circuit components in groups of 4 transistors. Each group of 4 transistors and related components is mounted on an electrical module subassembly. The electrical module and TRANSAC card blanks are printed circuit elements. All of the TRANSAC cards for an *S-2000* basic computer and control unit are obtained by first running 4,717 electrical modules of uniform mechanical dimensions through a conventional production line. These 4,717 modules are then assembled to 260 TRANSAC card blanks of uniform mechanical dimensions to produce the computer TRANSAC cards. Production and quality control facilities were brought into the *S-2000* program in the design phase and gradually expanded in the production phase of the first *S-2000* system. Trained factory personnel handle the production of TRANSAC cards without the need of engineering support. Each electrical module and TRANSAC card is statically and dynamically tested on sophisticated factory test equipment. The production operations are so simple that production capabilities may be easily and rapidly expanded in a matter of days.

The modular concept is also applied to the mechanical construction of *S-2000* equipment. The basic computer and control unit is composed of three mechanical modules as shown in Fig. 13. Each mechanical module contains four levels of back panel wiring. Horizontal wiring of each level is accomplished independently and the prewired levels installed in the module. This process speeds up the back panel wiring operation. Power supplies for the computer and control unit are completely transistorized. They occupy a volume of approximately 24 inches by 20 inches by 18 inches and require less than 1,500 watts of input power.

System Expansion and Optional Features

Optional features may be added to a basic *S-2000* system at any time by the simple process of plugging in kits of TRANSAC cards. The floating-point arithmetic kit, for example, contains 17 TRANSAC cards which may be plugged into the designated connectors in the computer and control unit in a matter of minutes. The floating-point arithmetic feature is only a matter of a few minutes. The floating-point arithmetic feature is thus easily incorporated into a basic *S-2000* computer with no rewiring or equipment modification. In a like manner, kits of TRANSAC cards are available for index registers (in groups of 8, up to a maximum of 32), and for the addition of input-output control cards for the various input-output devices.

Fig. 14 is a photograph of a magnetic-core storage unit (8,192 words). The size of the magnetic-core storage unit may be doubled by the addition of a second unit identical to the one shown. This is also accomplished by a simple plug-in operation requiring no rewiring or rework. The memory may be expanded beyond 32,768 words in blocks of 32,768 words if desired. Read-write and selection electronics are, of course, completely transistorized and packaged on four types of TRANSAC memory cards as shown in Fig. 15. The transistorized power supplies for the magnetic core storage unit occupy a volume of 22 inches by 19 inches by 16 inches as shown in Fig. 16.

Examples of TRANSAC *S-2000* input-output hardware are shown in the paragraphs of a magnetic tape unit (Fig. 17) and the high-speed printer system (Fig. 18). A layout for a typical *S-2000* system is shown in Fig. 19.

Conclusions

The conference announcement for the 1958 Eastern Joint Computer Conference promised the answers to the following questions, among others.

1. What is the status of transistors?

2. How will they affect the possible obsolescence of your present computer?

As a result of Philco's experience in placing the transistorized TRANSAC *S-2000* into production, the authors are able to supply these answers:

1. The role of the transistor as the active element in modern computers is established. The transistor is no longer a new device; rather, it is a proven component that has replaced the vacuum tube in the vast majority of computer applications.

2. A modern computer such as the TRANSAC *S-2000* demonstrates important performance advances over present vacuum-tube computers. It has much greater operating speeds, requires much less power, space, and air conditioning. Because of its greater reliability, it is easier to maintain. It costs less to install and operate.

Modern computers are transistorized computers. Vacuum-tube computers are obsolete.

Discussion

A. L. Tritter (Lincoln Laboratory): Is the "Repeat" instruction the only progammable function in which half-word boundaries matter? That is, can jumps occur in either half-word which effect a jump to either half-word?

Mr. Maddox: I may have misled you to some extent by saying that when we form instruction pairs that they did look like two address instructions. The two instructions are certainly independent, and jumps can occur from any one to any other. As far as the programmer is concerned, he is listing single-address instructions.

A. L. Tritter: Are there full-word instructions (except the input-output control words) as such?

Mr. Maddox: There are not, except for the input-output control words. I was only trying to make the point that there was an endeavor to make the single-address instructions match so that they form efficient pairs.

W. W. Bledsoe (Sandia Corporation): Please indicate the range of cost with a "typical example."

Mr. Maddox: I am afraid that this is really beyond the scope of an engineer to answer a question of this sort. This is handled by the Sales Department, and to discuss the cost, rental, or sale, is a matter of discussing a long list of part numbers and then summing them up to decide what a typical example might look like.

A. J. Azzari (Remington Rand Univac): How many *S-2000* units have been delivered?

Mr. Segal: We have delivered one to date.

Mr. Jones (International Business Machines Corporation): Are there any provisions for marginal checking?

Mr. Segal: As a matter of fact, there are provisions for marginal checking in *S-2000* but we do not use them. We are not convinced at this date that marginal checking, at least the conventional variety, really offers an advantage. I think we will have to wait until next year's conference before we know the value of any kind of marginal checking.

I. L. Auerbach (Auerbach Electronics Company): Please report on reliability of the *S-2000* system.

Mr. Segal: There will be a reliability report about a month from now in the next annual Symposium on Reliability and Quality Control.

Programming Design Features of the GAMMA 60 Computer

P. DREYFUS

THE GAMMA *60* is a complete, closed-loop electronic system designed to analyze, process, and produce information. The GAMMA *60* can at the same time read, write, punch, print, calculate, sort, compare, and make decisions. (See Fig. 1.)

In order to do so, the GAMMA *60* is equipped with:

1. Input equipment (card readers, paper tape readers, magnetic tape units).
2. Data processing units (calculators, comparators, transcoders).
3. Storage media (high-speed memory, magnetic drums, magnetic tape units).
4. Output equipment (card punches, paper tape punches, line printers, magnetic tape units).
5. Control elements which allocate instructions and data (program distributor, data distributor).

The control elements and the high speed memory make up the central unit of the GAMMA *60*.

The input-output units, auxiliary storage units, and data processing units are functional elements connected to the central unit.

The fundamental principle in the design of the GAMMA *60* has been the assignment to the functional elements of a completely closed loop mode of operation within the generally autonomous framework of the GAMMA *60*. A functional element attains complete autonomy once it has been furnished with basic data (from cards, punched tape, magnetic tape) and with a medium on which to record the information it produces (blank cards, paper tape, magnetic tape). Fig. 2 illustrates a block diagram of the GAMMA *60*.

The compatibility of the functional elements is made possible by their ability to:

1. Request instructions.
2. Request data transfers.
3. Execute the specialized tasks for which they have been designed completely. Independent of the other components of the GAMMA *60*, once they have received operating instructions and data.

This possibility which permits the simultaneous operation of all the functional elements is the basis of full utilization of the GAMMA *60*.

Within the generally autonomous framework of the GAMMA *60*, made possible by the automatic scheduling of the control element, it is possible to: 1. execute simultaneously data processing functions on items at different stages of evolution in the same problem, and 2. execute simultaneously the data processing functions of several different problems (simultaneous parallel operation).

This makes it possible to obtain considerable improvement in over-all execution times by operating each of the elements at optimum rates.

Information in the GAMMA *60*

Information in the GAMMA *60* is always in coded form. Information may be numeric or alpha numeric. Characters and numbers can be stored in the memory of the GAMMA *60* in the code of an external medium (punched card code, punched paper tape code).

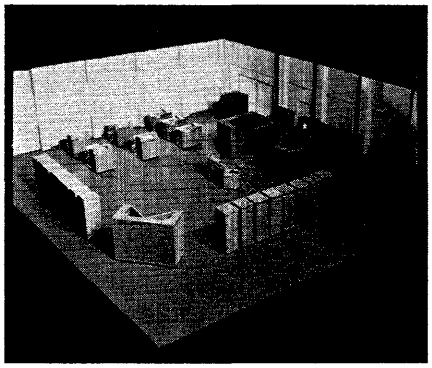


Fig. 1. A GAMMA *60* installation

This model is equipped with (clockwise from the top right) main unit and high-speed memory, power supplies, 2 magnetic drum units, 16 magnetic tape units and their control unit, 4 printers, 2 card readers, and 1 card reader punch

However, information can only be completely processed after having been translated into the internal codes of the GAMMA *60* by the transcoder. The internal codes of the GAMMA *60* are composed as follows:

Numeric information retains its decimal structure, but each decimal digit is represented by a combination of four binary digits (bits), a binary digit being represented by either the presence (1) or absence (0) of a stable electric or magnetic state.

The ten decimal digits are represented in the GAMMA *60* as shown in Table I. This representation of numbers is called the binary coded decimal representation.

P. DREYFUS is with Compagnie Des Machines Bull, Paris, France.

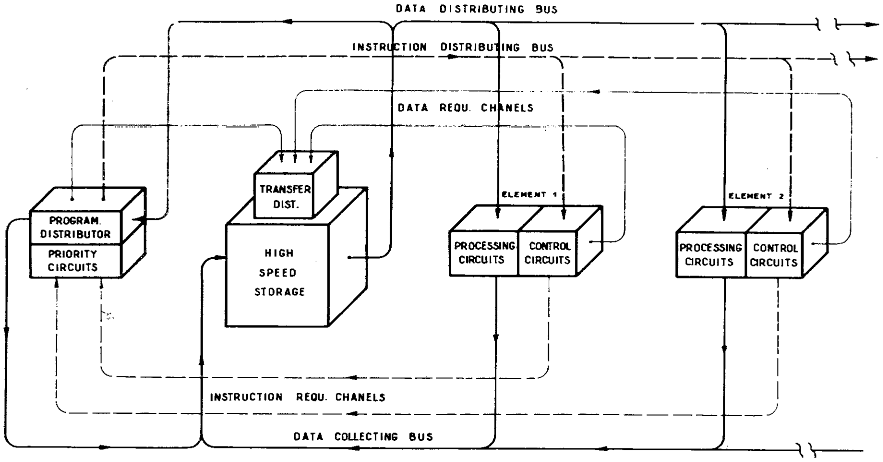


Fig. 2. Block diagram of GAMMA 60

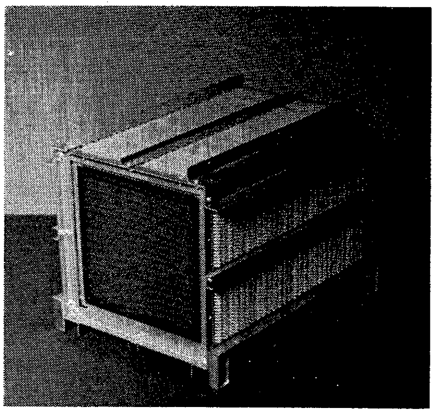


Fig. 3. Coincident magnetic core memory module

Table I. Representation of Decimal Digits in GAMMA 60

Decimal digits	0	1	2	3	4	5	6	7	8	9
Machine code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

In the GAMMA *60* the number 247 would be written as:

001001000111

Alphanumeric information retains its usual structure, but each character (alphabetic or numeric) is represented by six bits.

The 64 possible combinations make it possible to represent certain special characters in addition to the 26 letters of the alphabet and 10 decimal digits.

Letters, numbers and other characters are defined in Table II.

A character is represented by the group of binary bits found in the same line and in the same column.

W is represented in the machine 010111
8 is represented in the machine 101000
= is represented in the machine 110000

The unit of information transfer in the GAMMA *60* is a group of 24 bits called a *catena* (the Latin word for chain).

Thus, 1 catena may contain the following:

24 bits, representing data in binary form
6 decimal digits
4 alphanumeric characters.

Memory capacity and the length of information transfers are expressed in catena.

Description and Operation

The GAMMA *60*, as has been seen, is composed of the central unit to which are connected a variable number of functional elements. The number and type of elements connected is a function of the particular application.

Central Unit

The central unit contains the program distributor, the data distributor and the high-speed memory (HSM). The central unit serves as:

A dispatcher (program distributor and data distributor) which:

1. Distributes instructions.
2. Schedules data transfers.
3. Co-ordinates the activity of the various elements.

A turn table (high-speed memory) through which all information being transferred to and from the elements must pass. The latter do not possess buffers but simply registers of limited capacity in which information is stored only during operational cycles of the elements.

The High-Speed Memory

The high-speed memory, serving as buffer memory between the various elements, may contain at any instant:

1. Program information (instructions) which are the commands to the elements.
2. Quantitative information or data which are to be processed by the elements.

The high-speed memory and the elements are linked by two information channels:

1. Distributor bus over which is transferred all information from the HSM to the elements.
2. Collector bus over which is transferred all information from the elements to the HSM.

The Program Distributor

This unit is designed to:

1. Receive requests for instructions issued by the elements to obtain instructions for themselves or for another element.
2. Choose at a given time from among the requesting elements, one of them, by means of priority circuits, the order of priority having been determined in advance.
3. Read out the instructions destined for the elements as they are selected and interpret them. This interpretation function is very important and will be discussed in the chapter on programming.
4. Give to the various elements the commands and means to begin operational cycles (their type of operation to be carried out, addresses of data to be processed).

The program, which is a list of instructions intelligible to the different elements of the GAMMA *60*, is stored in the auxiliary and large capacity memories (magnetic drum, magnetic tape, etc. . . .) and brought as successive blocks into the

Table II. Character Representation in the GAMMA 60

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
01	P	Q	R	S	T	U	V	W	X	Y	Z	.	,	'	;	|
10	0	1	2	3	4	5	6	7	8	9	+	—	(	)	x	/
11	=	≠	>	<	Δ	&	π	Σ	%	£	$	*				

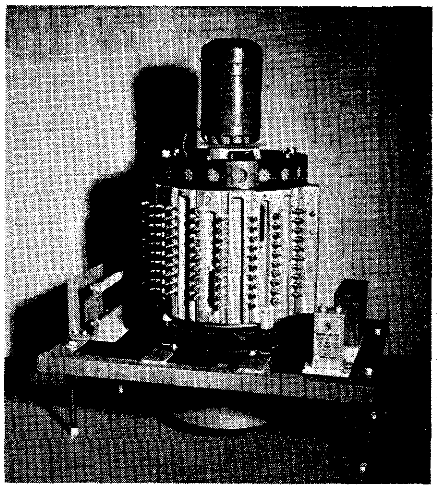


Fig. 4. Magnetic drum auxiliary store

HSM, from which the instructions are then transferred one by one to the program distributor, at its request.

THE DATA DISTRIBUTOR

This unit is designed to:

1. Receive requests for data transfers issued by the various elements to obtain as necessary, data transfers to or from the high-speed memory.

2. Select, in order to effect a data transfer, a given receiver or emitter element and the source or destination zone in the high-speed memory. The choice between conflicting requests is resolved by the priority circuits, the order of priority having been determined in advance.

THE FUNCTIONAL ELEMENTS

Each functional element contains the following:

1. Registers in which are stored the information to be processed.

2. A local built-in program which carries out the specialized functions of the element.

3. Auxiliary registers to store data addresses and instructions.

Every element is capable of:

1. Sending requests for instructions to the program distributor via the instruction request circuits, one line for each element.

2. Sending requests for data transfers from high-speed memory to the data distributor via the data request circuits, one line for each element.

3. Receiving instructions from the program distributor over the unique instruction bus to which every element is connected.

4. Receiving data (input elements excepted) via the unique distributor bus to which all the elements concerned are connected.

5. Sending data (output elements excepted) to the high-speed memory via the unique collector bus to which all the elements concerned are connected.

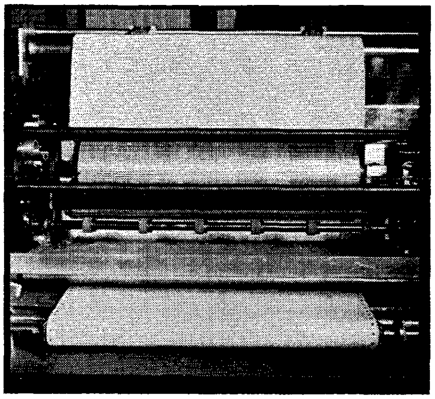


Fig. 5. Output line printer

THE OPERATION OF AN ELEMENT

The operation of every element follows this procedure:

1. Requests instructions from the program distributor.

2. Consideration of the request (depending on the position [rank] of the element in the priority chain and on the number of requests for instructions from other elements waiting to be selected).

3. Request from the program distributor to the data distributor for a transfer from the high-speed memory of instructions relative to the element under consideration.

4. Consideration of this request by the data distributor (depending on the position of the program distributor in the priority chain and on the number of requests for data transfers waiting to be fulfilled) and execution of the transfer.

5. Analysis of the instruction by the program distributor and transfer to the selected element of the instructions and addresses of data to be processed.

6. Requests from the element to the data distributor for a transfer from the HSM of the data to be processed.

7. Consideration of this request and, when the priority considerations are fulfilled, the execution of the data transfer.

8. Closed loop operation of the element and eventual producing of a result.

9. Requests to the data distributor for a transfer of the results to the HSM.

10. Consideration of this request by the data distributor and, when the priority conditions are fulfilled, execution of transfer.

11. Return to operation 1.

For each of the elements Magnetic Drum, Arithmetic Calculator, Printer, etc., the procedure is the same with a few exceptions.

For example steps 9 and 10 are nonexistent during the operation of the printer or of a HSM to magnetic drum data transfer; steps 6 and 7 are nonexistent during the operation of a card reader or a magnetic drum to HSM data transfer.

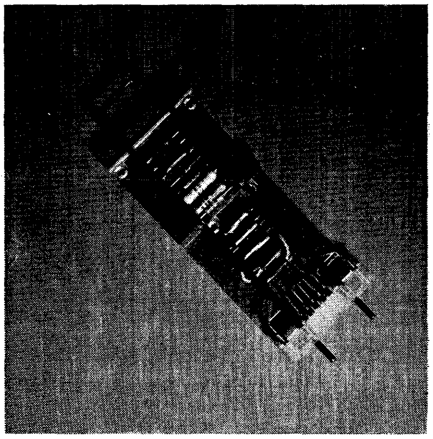


Fig. 6. Transistor amplifier module

Operation 8 is generally much longer than the sum of all other operations. This fact demonstrates how a single program distributor and a single data distributor are sufficient to monitor simultaneous operation of a large number of elements.

Technical Characteristics of the GAMMA *60*

MEMORIES

The GAMMA *60* ~~possesses a set of~~ memory devices for information storage, the functional characteristics of which are closely related to two technological considerations. These are capacity and access times.

Of the types of memory available those with the shortest access times are called high-speed memories. On the other hand storage of information in high-speed memories is expensive and therefore must be temporary.

Growing out of requirements diametrically opposed to the preceding, magnetic tape storage offers the advantage of large capacity but has long access times. This type of memory, which is relatively inexpensive to manufacture, makes it possible to store voluminous information which is organized in the same sequence in which the data are to be processed.

Magnetic drum storage occupies an intermediate position with respect to capacity and access times.

HIGH-SPEED MEMORY

Physically, the high-speed memory is a static memory of saturated magnetic cores. Two stable states of magnetization of the cores can be obtained. These two states are made to correspond to the symbols 0 and 1.

Technologically, the capacity of the

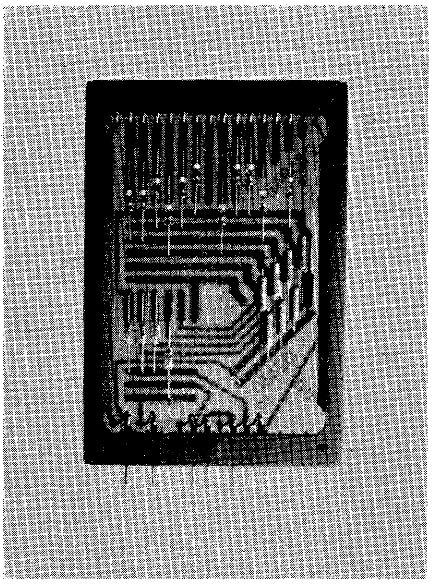


Fig. 7. Logical diode switching card

high-speed memory can vary from 1 to 8 standard blocks of 4,096 catenae.

Table III gives in terms of various units the capacity of the high-speed memory as a function of the number of memory blocks.

Table III. Capacities of High-Speed Memories

Units	1 Block	2 Blocks	4 Blocks	8 Blocks
Binary digits	98,304	196,608	393,216	786,432
Numeric digits	24,576	49,152	98,304	196,608
Alpha numeric characters	16,384	32,768	65,536	131,062
Catenae	4,096	8.192	16,384	32,768

Each block of 4,096 catenae consists of twenty-four 64×64-core planes (Fig. 3). The 32,768 catenae which constitute the maximum capacity high-speed memory are addressable and constitute a common memory pool, with the exception of addresses 0 through 127 which serve a special purpose.

The contents of any address in the high-speed memory can be read out or written in 11 microseconds. This read-write time is called the access time. Thus the frequency of reading or writing from the high-speed memory is approximately 90,000 catenae per second.

The Magnetic Drum

The magnetic drum (Fig. 4) is a steel cylinder coated with a thin layer of magnetic material. The surface of the drum is divided into 128 parallel tracks, each track having a read-write head.

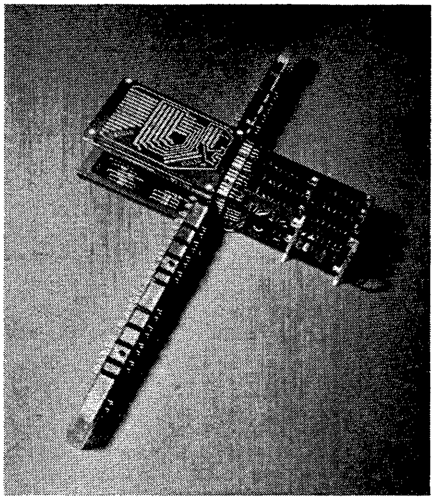


Fig. 8. Subassembly of two switching cards

The magnetic drum turns at a constant speed of approximately 3,000 rpm.

The capacity of each track is 200 catenae. Thus a magnetic drum has a total capacity of 25,600 catenae, 153,600 decimal digits, or 102,400 alphanumeric characters. The magnetic drum is an addressable memory, each catena on the drum having an address in the range from 00000 to 25,599.

The access time to an address on the drum depends on the position of that address with respect to the read and write head at the instant the contents of that address are called for. Thus the access time varies from 0 to 20 milliseconds. The magnetic drum is characterized by:

1. Maximum access time: 20 milliseconds
2. Average access time: 10 milliseconds.

The read or write time for one catena is 100 microseconds (μsec). Data transfers to and from the magnetic drum are completely variable in length.

Magnetic Tape

Magnetic tape provides the GAMMA 60 with a very large capacity memory. The tape itself consists of a plastic (Mylar) tape which is coated on one side with a layer of iron oxide. Magnetic tapes are handled by tape units which consist of:

1. Devices for unreeling and rereeling the tape.
2. A write head.
3. A read head which also serves during a write operation to check the information being written on tape.

Tape Characteristics

Length: 1,100 meters (3,300 feet).
Width: 12.7 millameters (1/2 inch).
Packing Density: (200 per inch).

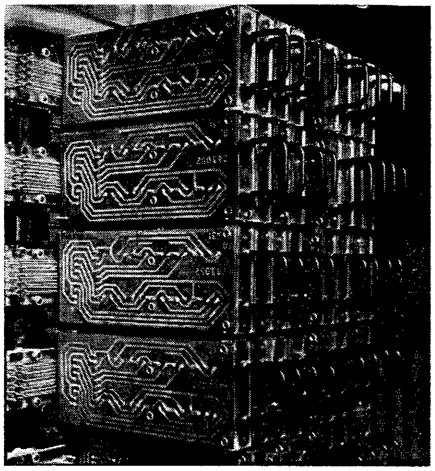


Fig. 9. Complete bay showing amplifier accessibility

Tape speed: 1.9 meters per second (75 inches per second).
Information is recorded on eight 15-kc channels (one channel is for synchronization).
Instantaneous information rate:
22,500 decimal digits per second.
15,000 alphanumeric characters per second.
3,750 catenae per second.
Time to read or write a complete tape: 9½ minutes.
Tape reading in both directions is possible.
Tape can be written in one direction only.

Tape Capacity

1,600,000 catenae in blocks of 128 catenae; Each block is separated from adjoining blocks by an inter-block or start-stop space.
12,500 blocks per tape.
9,600,000 decimal digits.
6,400,000 alphanumeric characters.

Time to Read One Catena

360 μsec.
This time corresponds to an average rate of 2,800 catenae per second.

Tape Security

Physical security is provided for end-of-tape, broken tape, jamming.

Number of Tape Units

Variable.
Generally between 6 and 18 units.
Very high information flow rates can be obtained since it is quite possible to operate several tape units simultaneously.

Data Processing Elements

Arithmetic Calculator

Number representation

The arithmetic calculator operates on numbers in a form similar to logarithms. Such numbers may be either normalized floating decimal point or have a programmed (fixed) decimal point.

Normalized form

Any signed decimal number can be represented by the product of a number between 0.1 and 1.0, a positive or negative power of 10, and the algebraic sign. For example:

$-3242.3 = -0.32423 \times 10^4$
$+0.0034 = +0.34 \times 10^{-2}$

A number can thus be completely specified by its algebraic sign, a power of 10 which represents its order of magnitude and a mantissa which lies between 0.1 and 1.

So as not to have to manipulate negative exponents, a bias of 40 is added systematically. In addition, only the digits of the mantissa to the right of the decimal point need be specified. The two numbers in the previous example are written:

Sign	Exponent	Mantissa
−	44	3242300000
+	38	3400000000

Actually numbers to be processed by the arithmetic calculator are stored in two catenae in the GAMMA *60*.

The first binary position of the first catena of the number contains the sign: +sign = 1, −sign = 0. The seven following positions contain the biased exponent which may vary from 00 to 79. This represents a true variation of the exponent from −40 to +39.

The 40 remaining binary positions of the two catenae permit the representation of ten significant decimal digits in binary coded decimal form.

Programmed Decimal Form

In accounting applications it is particularly useful to be able to write all figures placing the decimal point in a known position. In order to do so, the programmed decimal form is used in the GAMMA *60*.

To represent numbers in this form the position of the decimal point between the ten significant figures of a number is specified.

Writing a number in programmed decimal form with the decimal programmed at 6 means that the decimal point is to be taken as being six places to the left of the least significant figure. In this example, the integral part of the number contains at most four digits. The fractional part contains at most six digits.

The position of the programmed decimal point is specified in the part of the number in its machine coded form reserved for the exponent. The position of the programmed decimal point may vary from 0 to 10. The variation of the position of the programmed decimal point is actually expressed as a number between 40 and 50. When no decimal places are to be retained (programmed decimal at 0), the exponent is actually 50. When no figures to the left of the decimal point are to be retained (programmed decimal 10), the exponent is actually 40.

The number −3242.3 with programmed decimal 6 is represented in the GAMMA *60*:

0 4 4 3 2 4 2 3 0 0 0 0 0

and with programmed decimal 0:

0 5 0 0 0 0 0 0 0 3 2 4 2

and with programmed decimal 10:

0 4 0 3 0 0 0 0 0 0 0 0 0

Operation times for operands of 10 significant figures expressed in floating or programmed decimal form are the following:

Addition	100μsec.
Subtraction	100μsec.
Comparison	100μsec.
Round off	100μsec.
Multiplication	300μsec.
Division	600μsec.

Double Precision Arithmetic

Normally the arithmetic calculator handles operands of ten significant decimal digits. However, in certain scientific applications more precision is required. There exist programs for the GAMMA *60* which make it possible to perform calculations on operands containing 11 to 19 significant figures. This type of arithmetic is called double precision arithmetic.

The Logical Calculator

The logical calculator is the element of the GAMMA 60 designed to perform arithmetic operations on binary operands and to carry out logical operations.

Generalized Comparator

The Generalized Comparator makes it possible to carry out the following operations in the GAMMA *60*.

One way comparisons. Two variable-length operands are compared catena by catena, starting with the left most catena of each operand. The comparison process stops automatically as soon as a difference is detected. Thus both numeric and alphanumeric data may be compared.

Execution time is 44 μsec per catena.

Two-way comparisons. Variable-length operand x is compared with two other operands, A and B. The comparator indicates after the operation the magnitude of x with respect to A and B. For example, if $A<B$ the comparison shows whether:

$X<B$
$X>A$
$A<X<B$
$A \leqslant X \leqslant B$
$X=A$
$X=B$

Execution time for the comparison is 66 μsec per catena.

Variable Length Data Transfers. Transfers of data from one zone in high-speed memory to another are executed using the registers of the generalized comparator. Time to execute HSM to HSM transfers is 22 μsec per catena.

The Transcoder

The transcoder is the element of the machine which makes it possible to translate into GAMMA *60* internal code (numeric or alphanumeric) information in external codes (punched card code, punched paper tape code etc.) and to translate information expressed in GAMMA *60* internal code into external codes for punching cards, paper tape etc., edit data for the printer, placing spacings and printer format controls, rearrange data within a catena (reducing redundancies and eliminating empty character positions to obtain minimum storage volumes and performing the inverse operations before data processing operations).

Input-Output Equipment

Input Elements

The following are the devices which make it possible to bring information in the GAMMA *60*:

1. 80-Column Card Readers. Each card reader consists of three sets of read brushes. The first two sets make it possible to read and check-read a card. If the check fails the third set of brushes makes it possible to re-read the card. Each reader has two receiving bins.

Reading speeds: 300 cards per minute
400 characters per second.

Any number of card readers may be connected to the GAMMA *60*.

2. Punched paper tape readers. The punched paper tape readers can read all 5- and 8-channel paper tape codes.

Each reader has only one sensing element with a speed of 200 characters per second.

Any number of paper tape readers may be connected to the GAMMA *60*.

Output Elements

1. Printers. A printer consists of a rotating-type cylinder of 120 printing positions, each printing position consisting of 60 characters distributed around the circumference of the cylinder, thus forming 120 "type wheels," (see Fig. 5). The edge punched paper is driven past the cylinder by means of sprocket wheels. When a character moves past the printing point a small hammer behind the paper is actuated pressing the paper against the cylinder. Fast paper feeds and line spacing are program controlled. Operating speed is 300 lines per minute. Any number of printers may be added to the GAMMA *60*.

2. 80-Column Card Reader-Punch. A card punch is always equipped with read brushes. A card reader can exist without card punch. A single track serves both units.

Each card reader-punch is equipped with three sets of read brushes. The third set is used to read back and check the information punched. Card handling speed is 300 cards per minute.

Any number of card reader-punches may be connected to the GAMMA *60*.

3. Paper tape punches. It is possible to punch 5 and 8 channel tapes on the Gamma *60*.

Punching speed is 25 characters per second.

Any number of paper tape punches may be connected to the GAMMA *60*.

Programming the GAMMA *60*

Programming the GAMMA *60* consists in writing in symbolic form a list of all the operations to be carried out on one or several items of information for the particular problem at hand.

A program consists of a combination of several elementary instructions called canonical instructions.

Since information transfers in the computer are discontinuous, the length of an elementary instruction has been assigned as 1 catena; that is the length of the unit transfer in the machine.

Canonical Instructions

When an operation is to be executed by an element of the machine the following specifications must be supplied:

1. Which element is called for?
2. The high-speed memory addresses of the operands and/or the address at which the result is to be placed, stored.
3. The operation to be carried out.

As the program distributor reads a program, it decodes instructions which are stored in successive memory locations. A useful possibility is to be able to jump either conditionally or unconditionally to an instruction not stored in the same sequence of consecutive addresses.

From the preceding, one can see the necessity for four types of canonical instructions:

1. The C-instruction which designates the element called for.
2. The A-instruction which designates the addresses of operands in the HSM.
3. The D-instruction which designates the operation to be performed.
4. The B-instruction or branch which modifies the usual order of instruction execution.

The Program

A single canonical instruction is in general not adequate to define an elementary operation in the GAMMA *60*.

A sequence of canonical instructions which completely specify an elementary operation constitute a "complete instruction."

A sequence of complete instructions, all relating to the same element, constitute an elementary program sequence.

A sequence of elementary program sequences which are related to the same program function constitute a subroutine.

Some types of program functions are:

1. Subroutines which perform calculation of functions such as the elementary mathematical functions. . . sine, cosine, exponential, logarithmic. . . etc.
2. Subroutines which perform certain functions. . . input, output, high-speed memory sort routines.
3. Special subroutines which are due to the particular idiosyncracies of the problem.

The first two types of subroutines are known as general-purpose subroutines and can be grouped into libraries. Stored in large capacity memory, they are available to the GAMMA *60* at all times.

A routine program consists of an ensemble of special and general-purpose subroutines to solve a particular problem which are assembled through the use of a master program.

As actually carried out, a segment of a master program, perhaps with parallel sequences, is brought into the high-speed memory. Various subroutines are then brought into HSM as they are needed and jumps back and forth can be executed between the subroutines and the master program as necessary.

Generally to facilitate the coding of problems for the GAMMA *60*, a symbolic instruction code is used. The symbolic instructions are then translated by the computer into canonical instruction.

There are two types of symbolic instructions. The first type is a symbolic representation of one canonical instruction. The second type is the symbolic representation of a sequence of canonical instructions. (Such instructions are processed by the master program and become program jumps to special and general-purpose subroutines.)

The symbolic code has been designed to facilitate as much as possible the work of the programmer. The instruction vocabulary is simple, and mnemonic abbreviations have been used. For example, the instruction which calls for comparison of two program sequences is written in symbolic code COMP. Operation of the arithmetic calculator is programmed ARIT.

As an example, to execute a high-speed memory-to-magnetic drum transfer of 12 catena starting with HSM address 108 to magnetic drum addresses starting with 17,400, the program is written:

```
DRUM           1
   L         108
 DSA      17,400
 TMD          12
```

DRUM 1 selects magnetic drum No. 1. L loads the address 108 in the current address register of the Magnetic Drum. DSA specifies the magnetic drum address where the information is to be stored. TMD is an operation code which specifies the operation and the number of catena to be transferred.

When this program is brought into the GAMMA *60* it is translated from its symbolic form into an internal code which is intelligible to the program distributor.

An important characteristic of the GAMMA *60* is the use of relative addressing in writing subroutines.

As in the case of the design of the symbolic code to facilitate programming, the numbering of symbolic instructions has been designed in such a way that the programmer does not have to be concerned, when he writes his program, with the addresses in high-speed memory from which his program is to operate. When written by the programmer, the instructions of a subroutine are simply numbered from 0 to *n*. It is the master program which causes the GAMMA *60* to carry out the transformation of relative addresses into absolute addresses once the actual high-speed memory allocation is decided upon.

Additional programming flexibility is provided in the GAMMA *60*. It is possible to write instructions which specify, not the actual address of an operand, but the address at which the address of the operand can be found. This operation is called substituted or indirect addressing.

Checking in the GAMMA *60*

Internal checking in the GAMMA *60* is extended to all data processing functions, from the input of data to the output of results.

These distinct problems must be solved.

1. Internal data transfers must be checked.
2. Data processing operations must be checked.
3. Data input and output operations must be checked.

Internal Data Transfer Checks

In internal code in the GAMMA *60* associated with each catena is a check digit. This check digit is the remainder of the division by 7 of the catena considered as a pure binary number of 24 bits. Three binary bits are required to represent the check digit. Every time a catena is transferred the checked digit is recalculated and is compared with check digit accompanying the catena. The check digits must be equal; if not, a faulty transfer has occurred. This type of check provides remarkable security. (Note: It is well to specify that a catena in the high-speed memory, or in any register in the GAMMA *60*, necessitates the memorization of 27 binary bits and not 24 as has been assumed up to now. Each block of the HSM actually contains 27 core planes and not 24.)

Checking of Data Processing Operations

The check of data processing operations is based on the fact that the same operations are carried out on the data and the check digits separately, then the following rule is applied. The check digit of the result of an operation must be identical with the result of the same operation carried out on the check digits of the operands. More briefly stated; the check digits of the result must equal the result of the check digits. If this is not true then the operation is in error.

Checking Input and Output

With the exception of punched paper tape on which each character is punched with a parity check bit which makes it possible to check the information punched as it is being read, information on external media (punched cards, printed characters) not possessing check digits must be checked during read-in on printout by a different procedure from that previously described.

Magnetic Tape Units

When reading magnetic tapes, each catena stored with its corresponding check digit is checked for validity. When writing on magnetic tapes, a read head provides simultaneous reading thus allowing immediate checking. If an error is detected, automatic rewrite occurs; if the drop-out cannot be corrected, the corresponding block on the tape is inhibited to further use. This allows use of imperfect tapes.

Card Readers

Each card is read by two sets of brushes and the results of both reads are compared. In case of disparity, a third set of brushes makes it possible to re-read the card. The information from the third read should validate one of the first two reads. If the information read by the third brush still does not match, the card is rejected into a special bin and an error indicated.

Card Punches

On the card track after the punch there is a set of read brushes which are used to check-read the card just punched. The check is performed by comparing the information read with that just punched.

Printers

The motion of each print hammer trips an electrical pulse, which by comparison with the position on the print cylinder makes it possible to identify the character printed. This "echo" is compared with the information that was to be printed.

Error Routines

All checks are performed automatically and locally and are of no concern to the programmer.

Once an error is detected, it causes the execution of an error routine. The machine pinpoints the error and tries to correct it.

If the error is due to the malfunction of an element, then the computer cannot remedy the situation. The computer is stopped (partially or completely) with the type of error and source indicated on the console of the GAMMA *60*. Rapid trouble shooting and maintenance can then be performed.

Conclusions

The GAMMA *60* is a universal computer, the power, the flexibility, and the speed of which make it an indispensable management tool.

The possibility of connecting to the GAMMA *60* functional elements in proportions and dimensions tailored to the application make it possible to assemble, at any time, a computer suited to the exact needs of the user.

The possibility of executing several problems simultaneously gives the GAMMA *60* great flexibility and power (see Figs. 6–9).

In addition to these features, the sturdy design of the hardware of the GAMMA *60* and the extended use of transistor circuits make it a remarkably trouble free and reliable machine.

Discussion

Question: When will the GAMMA *60* be available? How many machines are already ordered?

Mr. Dreyfus: The prototype will begin operation in June 1959 and will be complete and in full operation in October. The first two deliveries are scheduled to take place before the end of 1959.

As of today we have seven firm orders for systems in Europe.

Question: When will the GAMMA *60* be available in the United States?

Mr. Dreyfus: Taking into account reservations and commitments, deliveries here are about 2 years off.

I. L. Auerbach (Auerbach Electronics Company): What provisions are available for real-time input and output in the GAMMA *60?*

Mr. Dreyfus: Real-time computation can occur both in data processing and industrial control applications, not to speak of military and engineering real-time problems. It is easy to handle these problems on the GAMMA *60* for, at any time, and whatever the machine is actually busy on, one may pick up a different processing routine which will time share the machine with the other programs on hand. This routine has access to all the processing power of the systems, searching files, computing, making decisions, editing, and obviously, input and output. One such application would be a permanent inventory control where the main file need not be on fast, but limited-capacity, random access media but on ordinary magnetic tape, this being up-dated permanently with a maximum of 15 minutes between keyed input and actual posting with eventual response.

Another type of real time computation will occur in control processes. In this case, information is coming in at random through proper input devices. The information handling process will be initiated by the input device without any delay even if other processes have been started almost contemporaneously and are not yet completed. The machine then simulates a multiple processing system although remaining economical in view of the fact that the distribution of processing functions assumed is the most probable instead of the worst. Information flow in these problems is limited by the maximum channel capacity, which is 2,200,000 bits per second.

This does not reflect the maximum input or output capacity but corresponds to the HSM cycle time of 11μsec. per catena.

The actual input-output rate which could be handled would be of about one-tenth of that. This means that 200,000 bits per second as an average input rate could certainly be handled whatever the time distribution may be.

The GE-100 Data Processing System

R. H. HAGOPIAN H. L. HEROLD J. LEVINTHAL

J. WEIZENBAUM

AT THE 1955 Eastern Joint Computer Conference, members of the Stanford Research Institute described a developmental prototype of a business data processor for accomplishing bank checking account bookkeeping.[1] This paper describes the electronic data processing system evolved in the course of converting the developmental prototype into a production system capable of meeting the requirements of both satisfactory operation for banking and manufacturability.

The banking problem may be defined as receiving handwritten checks from a customer, converting the information on the checks to posted entries in a ledger, and periodically sending a copy of the ledger entries to the customer as a statement. The checks must also be sorted and filed so that they may be included with the customer's statement (see Fig. 1).

It is essential that the system be able to process the daily volume of entries in no more than 24 hours so that processing may be kept current. This must be done in spite of business fluctuations or machine down time.

A further time limitation is the determination within a fixed time (imposed by local clearing house regulations) of checks that cannot be honored for reasons such as insufficient funds or stop-payment actions since bad checks must be returned rapidly enough to find the casher and permit the cashing bank to both recover the funds it has expended in cashing the check and to prevent loss of interest on these funds during this period. This deadline also must be met in spite of mutilated checks, checks without identifying account numbers, operator error, or equipment malfunction.

The system design, including operating procedures, programs, and equipment, has been governed to a great extent by the necessity of meeting these two time requirements while working with prime documents that are essentially identical to those now in use. The system's components include input equipment, processing equipment, and output equipment (see Fig. 2). The most important input equipment is the document handler, which reads and sequences the entries. Additional input devices are the Flexowriter and the photoreader for name and address changes. The central processor performs the computing and data processing for the system, and the output is provided by the high-speed printer. Magnetic tape units provide high-speed auxiliary storage.

Document Handler

The development of a paper handling system which sorts the items and converts the transaction data into computer language represents the most important technological breakthrough in the design of a system which will meet bank deadlines. Because this system has the unique feature of utilizing the source document itself as the means of direct input to the central processor, the document handler unit may be properly regarded as the key to the General Electric Company's GE-*100* system. The major components of the document handler are a correlaton character reader, 12-pocket document sequencer, and sequencer control.

In the bank checking account application, all of the necessary transaction data are printed in magnetic ink in a single line along the bottom of each check (see Fig. 3). When a customer writes a check which has been preprinted with the customer's account number, it is ultimately returned to the bank and the dollar amount is encoded with magnetic ink during the proofing operation when the bank verifies the dollar total of checks received. Bundles of checks are then placed in the feeder station of the document sequencer and are automatically fed past the magnetic read head of the character reader. The transaction data are read electronically and transmitted to the central processor by means of the sequencer control unit, which converts the single character output to the modified binary-coded decimal notation used in this system. The checks are automatically distributed to sequencer pockets as designated by the central processor on the basis of validity, readability, and other information on the item.

After the first pass of the bank checks, during which the data are captured by the processor for later processing, the character reader and document sequencer subsystem may be placed in off-line operation for automatic fine sorting of the checks by account number, transaction code, or transit routing code. Pocket decisions are made by the sequencer control during the off-line operation.

The automatic processing of checks requires that the printed information must be read reliably from folded, crumpled, and soiled documents. The system must function even when the printed information is overstamped or overwritten with ordinary ink or pencil, and it must operate satisfactorily regardless of the color or tinted pantograph on the checks. Finally, any one of the conventional printing processes may be used to print the checks.

The GE-*100* character reader is a completely transistorized unit capable of reading 14 characters (ten digits and four special symbols). A block diagram of the character reader is shown in Fig. 4. The printed characters are first magnetized to saturation in order to erase any previous magnetic history. As they pass the reading head, they generate unique voltage waveforms which can be identified with each of the characters.[2]

The waveforms are amplified by a low noise preamplifier and amplifier and then fed into a lumped constant taped delay line. The delay is sufficient to encompass the longest waveform generated by the widest character. From the delay line onward, the circuitry divides into 14 parallel recognition channels and control functions. The stored waveform is sampled at a number of points along the delay line.

R. H. HAGOPIAN, H. L. HEROLD, J. LEVINTHAL, and J. WEIZENBAUM are with General Electric Company, Palo Alto, Calif.

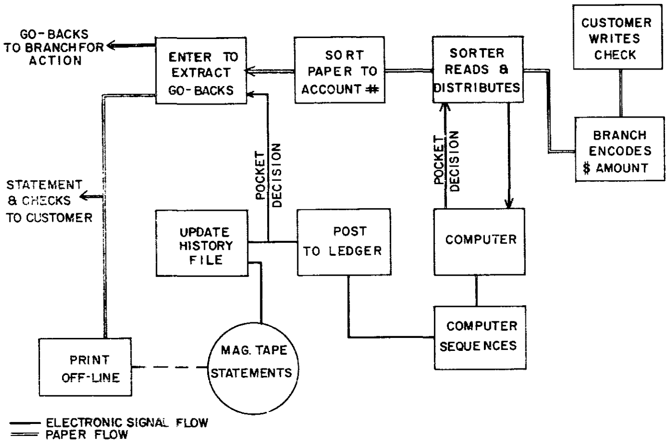
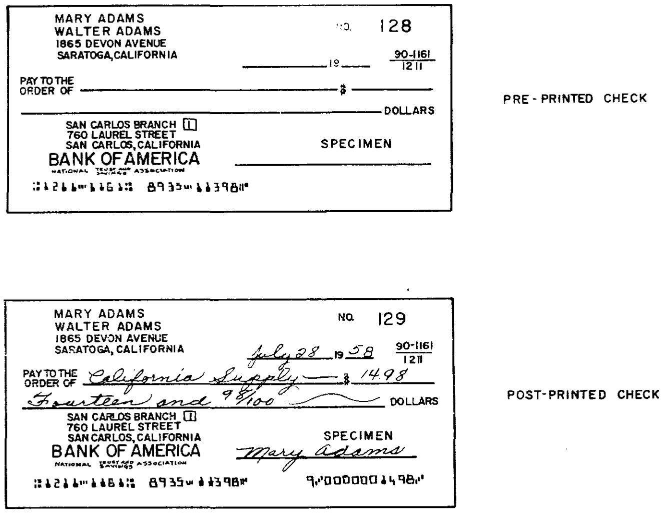


Fig. 1. Processing flow chart



MARY ADAMS
WALTER ADAMS
1865 DEVON AVENUE
SARATOGA, CALIFORNIA

NO. 128

______ 19 ____ 90-1161/1211

PAY TO THE ORDER OF ______ $ ______

______ DOLLARS

SAN CARLOS BRANCH [1]
760 LAUREL STREET
SAN CARLOS, CALIFORNIA
BANK OF AMERICA
NATIONAL TRUST AND SAVINGS ASSOCIATION

SPECIMEN

⑆1211⑆1161⑆ 8935⑈11398⑈

PRE-PRINTED CHECK

MARY ADAMS
WALTER ADAMS
1865 DEVON AVENUE
SARATOGA, CALIFORNIA

NO. 129

July 28 19 58 90-1161/1211

PAY TO THE ORDER OF California Supply $ 14.98

Fourteen and 98/100 DOLLARS

SAN CARLOS BRANCH [1]
760 LAUREL STREET
SAN CARLOS, CALIFORNIA
BANK OF AMERICA
NATIONAL TRUST AND SAVINGS ASSOCIATION

SPECIMEN

Mary Adams

⑆1211⑆1161⑆ 8935⑈11398⑈ ⑈0000001498⑈

POST-PRINTED CHECK

Fig. 3. Checks coded with magnetic ink

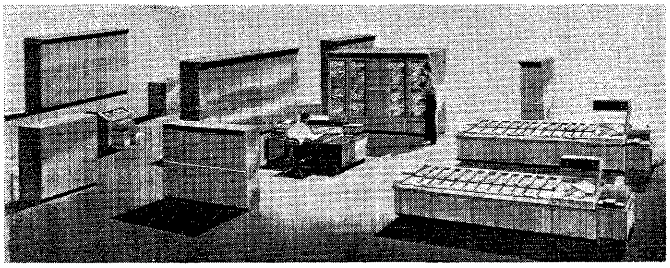


Fig. 2. The GE-100 system

Each sample point goes to 14 resistor matrix boards, one for each parallel channel representing a character.

The resistor matrix for a given character essentially stores the waveform for that character. The waveform in the delay line is compared to these 14 stored waveforms, and the comparison output appears in each of the 14 channels. Assuming that a valid character is being scanned, the delayed waveform will generate an autocorrelation output in its corresponding channel and cross-correlation outputs in the other 13 channels. The autocorrelation function for properly printed characters is significantly greater than cross-correlation functions. The autocorrelation output is selected by means of summing amplifiers and a peak comparator circuit.

Since the recognition process is dynamic, it is necessary to monitor the input in order to know when a valid character is properly registered in the delay line. This is the primary function of the timing and control circuitry shown at the right of the diagram (Fig. 4).

The *E/13-A* character font, which has been approved by the Committee of Office Equipment Manufacturers and recommended to the American Bankers Association as the common language for field evaluation, is shown in Fig. 5. The characters are stylized in order to obtain an optimum compromise between machine readability, esthetics, and printability.

The character reader is designed to read printing from conventional commercial printing processes. These fall into the categories of preprinting and postprinting.

Preprinting refers to printing on the checks prior to issuance by the bank to its customers. Such information as the transit number, routing symbol, account number, and transaction code are preprinted by letterpress, photo-offset, or paper mat offset methods. Large quantity evaluation of commercial printing indicates that good quality commercial printing techniques are sufficient to meet the printing specifications for magnetic character reading.

Postprinting takes place after the checks return to the issuing bank. The dollar amount, and if necessary the account number and transaction code, are postprinted. Several firms are engaged in developing suitable magnetic transfer ribbons and encoders for this purpose.

The document sequencer used in the GE-*100* system was developed by a subcontractor for General Electric. It is designed to feed and stack documents of varying size as normally used in the banking business at the rate of 750 documents per minute. The 12 stacker pockets include ten for numerical sorts, one for rejects, and one for exceptional items as determined by the sequencer control.

The feed station has a 12-inch capacity, and the pockets can accommodate check stacks up to 6 inches deep. With approximately 200 checks per inch, this permits sorting bundles containing as many as 2,400 checks.

Central Processor

The complexity of the banking problem calls for an internally stored program processor which makes use of a core memory in order to achieve the speed required to meet the banking deadlines with the volume of data handled. The necessary speed is enhanced by using a short word length which, in combination with the order structure, allows complete time overlapping of computation and memory access. This short word length permits the use of a single-address machine at high speeds plus the ability to operate on double length words at slightly reduced speed. Memory cycle time is 32 microseconds (μsec), or eight periods of a 250-kc clock. A single-address addition requires two memory cycles for execution, or 64 μsec: one cycle to get out the instruction, and one cycle to get out the operand.

Transistorized circuitry is used throughout. Flip-flops are used both for control and for register storage. Diode networks with intermediate emitter-followers provide the logic implementation.

The central processor operates in a series-parallel mode, using a 4,000-word core memory. Each word is composed of 28 bits, 24 of which are used to form six decimal digits (four bits each in 5 4 2 1 modified binary-coded decimal notation). The remaining four bits make up the seventh or sign digit. One bit represents the sign. One bit serves as designator

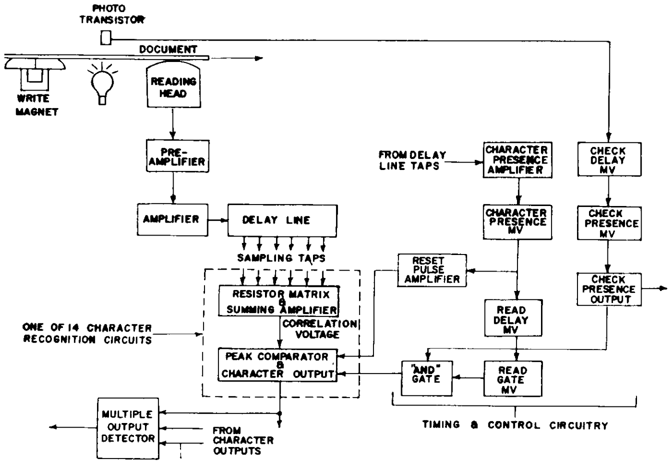


Fig. 4. Block diagram of document reader

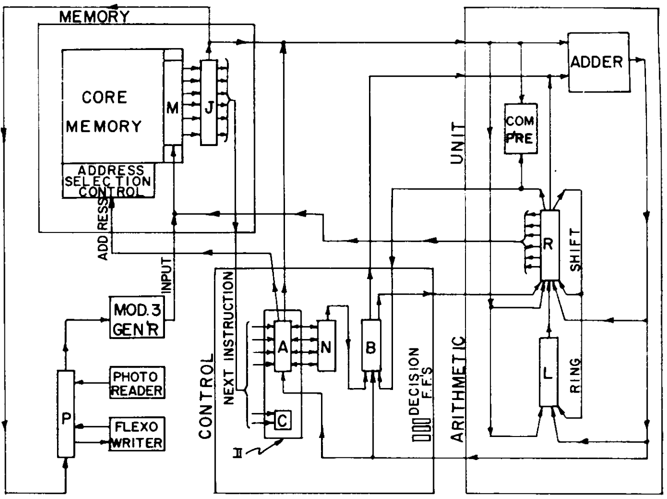


Fig. 6. Processor block diagram

and can be tested by the programmer when in data. In instruction words, the designator bit automatically increases the address portion by two when restored into memory. The additional two bits in the sign digit are used for a modulo-3 check.

The GE-*100* has a fairly standard instruction repertoire. Exceptions to this are the search and tumble instructions, which are very useful in sorting operations, and blockette write, which permits writing 100 words on magnetic tape in 20-word groups from five areas of memory as specified by the programmer. The latter instruction is also useful in sorting operations.

The instruction logic is designed so that the memory and the processor are well matched time-wise. One almost never waits for the other. Exceptions to this are multiply and divide instructions, where the memory essentially idles while the processor works.

The central processor includes the following electronic components (see Fig. 6): two registers used for arithmetic operations, *L* and *R;* one nonaugmenting index register, *B;* a control counter register, *N;* and the instruction address register, *I*, which is subdivided into *C* and *A* for the command and address, respectively. In addition there is a memory register, *M*, which communicates directly with the core matrix, and the memory buffer register, *J*, which allows the memory and the rest of the computer to work simultaneously.

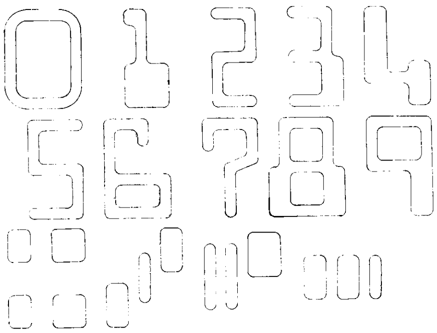


Fig. 5. Series E/13-A character font

Input-output equipment consists of magnetic tape units, a punched paper tape photoreader, Flexowriter, high-speed printer, and the document handler. A one-word buffer, the P-register, buffers input and output to the Flexowriter and input from the photoreader. The magnetic tape buffers allow one tape to be read and another to be written simultaneously with computation. This is accomplished by automatically interrupting computation in order to address the core memory for each word that goes to or comes from magnetic tape. Up to 13 magnetic tape units can be tied in to the processor at one time. A fixed 100-word block is used. Data are recorded on 11-channel tape with two numerical digits per cycle read or recorded in parallel. The basic clock rate is 15 kc, which yields a 30-kc rate for numerics. Row and column parity is checked, but the system is not self-correcting. A trailing read head checks parity during the writing operation.

The printer usually operates off-line from the processor, using magnetic tape as input, but it can be operated on-line. In the latter case the magnetic tape buffer is used with the printer as if it were a writing tape unit.

High-Speed Printer

The GE-*100* system's high-speed printer is a small data processor in itself that takes information from either magnetic tape or the central processor and prints it in an efficient and acceptable manner. Format control is effected by the use of plug boards and paper tape loops.

The printer subsystem relieves much of the burden that would normally be placed on the central processor and on the programmer. The subsystem is composed of a printing unit, thyratron drive unit, and power supply and control unit.

The printing unit is of the revolving drum type and contains a print wheel, operator's control panel, paper tape loop control, and paper feed control. The print wheel rotates at 900 rpm. Nine hundred lines of numeric information or 600 lines of alphanumeric information are printed per minute. The printer recognizes the type of data received and automatically prints at the highest speed possible.

As many as three carbon copies can be produced by the printer. Forms may vary in length from 1 inch (for labels) to 22 inches, and in width from $3^1/_2$ inches to 19 inches. The forms are fed through the printer by means of double sprocket holes.

The print wheel is actually a set of 120 separate type wheels, each of which holds 56 characters. Characters are spaced ten to the inch horizontally and six single-spaced lines to the inch vertically. The 56 characters include 10 numerals, 26 alphabetics, and 20 special symbols.

The thyratron drive unit houses the circuitry which drives the 120 individual solenoid hammers, associated with the 120 print wheels, that cause a line to be printed.

The power supply and control unit includes the power supply, control logic, plug board, main core buffer, type line buffer, and a tape unit.

Data are loaded into the main core buffer a full block at a time from either magnetic tape or the central processor upon demand of the printer. This loading process is initiated during a printing or paper slewing cycle so that no time is wasted in making the data available to the printer. When the print or slew cycle is complete, the printer calls for a line of data from the main core buffer. Data are then transferred from the main core buffer into the type line buffer, where it is routed to a particular column of the 120 possible print columns in accordance with the configuration of the plug board. After the line is set up, the appropriate hammer is energized as the characters on the rotating print wheels reach the print position. After a full revolution of the print wheel (for alphanumeric data) all the characters in the type line buffer will have been printed. Numeric data requires only a half-revolution of the print wheel. The main core buffer is loaded during this operation as it becomes empty and the type line buffer may then be reloaded for each of the lines contained in the block.

Format control includes automatic and controlled tabulation, in which 15 possible tabulations may be set to any of the 120 column positions. These tabs may vary in accordance with eight possible formats. Selective tabulation rearranges the word or item order according to criteria defined by the programmer. For instance, all positive items (deposits, on bank statements) could be sorted to a certain tab position regardless of their original position in the line of data.

The format control also includes automatic page numbering, automatic page heading restoration, automatic file selection (the printer will select any one of five files as specified by the programmer), automatic and selective zero suppression, automatic and selective slew control, and raw tape printout.

Conclusions

In order to insure that the banking job would be finished within the required time schedules in spite of possible downtime, a system has been built employing efficient paper handling techniques and utilizing a data processor that achieves speed through overlapping operation. As a result, standardized circuitry, working at slow speeds, may be used to maintain reliability.

An elaborate printer system is utilized to minimize editing time on the central processor. The outcome of this approach yields a general-purpose machine well suited to the special purpose of banking for which it was originally designed.

References

1. Electronics in Financial Accounting, **B. J. Bennett, K. R. Eldredge, T. H. Morrin, J. D. Noe, O. W. Whitby.** *Proceedings*, Eastern Joint Computer Conference, Institute of Radio Engineers, New York, N. Y., 1956, pp. 26–32.

2. Magnetic Character Reader for Bank Data Processor, **R. H. Hagopian.** *Symposium*, Association for Computing Machinery, Los Angeles, May 9, 1958.

Discussion

H. W. Graham (Sylvania Electric Products, Inc.): How do you overcome the orientation problem of making sure the magnetic writings are right side up?

Mr. Hagopian: During the normal proofing operation, where the dollar amount is postprinted on the check, the checks automatically end up in the proper orientation. Should the operator insert the check in the postprinter incorrectly, the document is rejected on the first pass.

N. J. Dean (Booz, Allen, & Hamilton): What is the reliability of the magnetic ink reader? That is, what is the reject and undetected error rate?

Mr. Hagopian: The reject and error rates are governed primarily by the quality of magnetic ink printing. For this reason, it is difficult to state any figures. We need to obtain statistical data from much larger number of printers. Recently concluded field evaluation tests, sponsored by the Office Equipment Manufacturers Sub-Committee on type design, involving some 47 printers throughout the country, indicated that the printers can meet the printing specifications.

T. Sapino (Datamatic): Another question on reject and error rates: What is the rate of rejects in magnetic character reading? What is the rate of undetected error?

Mr. Hagopian: The same answer as for the preceding question.

Mr. Dean: How many sorter readers can be multiplexed as input at full speed?

Mr. Hagopian: With the use of auxiliary equipment developed by the General Electric Company, up to three sorters may be multiplexed as input at full speed.

Mr. Sapino: What is the speed of preparation of magnetic characters on checks?

Mr. Hagopian: The speed of preparation of checks is determined by the printing process used in preprinting operation; the speed of preparation is no different than when using conventional inks.

Postprinting requires manual operation on the part of the operator performing the proofing operation. This is no different than what the operator would be doing without magnetic printing. In one case, the operator places the check, after entering the amount on the proof machine tape, in one of a number of bins. In the second case, the operator places the check in the post printer. From here on the operation is automatic.

Mr. Sapino: Is only decimal information printed magnetically or is alphabetic also implemented?

Mr. Hagopian: For banking operation, only numerics are printed. The system described here reads numerics and four special symbols. Future character readers will undoubtedly read alphanumerics.