even for spherical crystals often exhibit differences of shape or integrated area beyond that expected statistically. Contributing errors as a result of multiple scattering, improper sampling of diffuse scattering *etc.* can be investigated programmatically by rotation about the scattering vector or change in the scan mode.

Future trends

Certainly more can be done than is being done. Hopefully, more will be done. Certain trends argue for this.

First, the radical and continued drop in the cost of computing equipment will put this kind of apparatus into the hands of a larger number of experimenters. Those who already have automated diffractometers will find that costs are dropping so rapidly that they will be able to afford substantially larger on-line memories. That, in turn, translates into substantially more sophisticated and versatile control programs.

A second trend is towards the kinds of control and programming languages with which crystallographers are comfortable. The system of which I spoke at the beginning of this paper is now in the process of being extensively upgraded (Dimmler, Greenlaw, Kelley, Potter, Rankowitz & Stubblefield, 1975).

One of the prime motivations was to provide Fortran as the programming language. Others have followed the same route (see Sparks, 1973, 1976).

However, what is needed is a coming together of crystallographers to agree on common conventions and standards for their systems. In this way future systems can be built to stand on the shoulders of their predecessors and not their toes.

In 1967 I wrote a paper (Spinrad, 1967) in which I tried to explain how the research laboratory was becoming automated. I reviewed the underlying principles, offered examples of current systems and extrapolated future trends. One of the examples I used was that of automated diffractometry. Regretfully, of all the fields reviewed, crystallography has made the slowest progress. Other fields have substantially realized the potential that was apparent in the 60's. But, except for some advances in the mechanics of structure determination, crystallography has not. I hope that this will change. There is really no reason why it should not.

It is a pleasure to acknowledge the helpful discussions I had with S. C. Abrahams, R. E. Dickerson, D. G. Dimmler, J. M. Hastings, R. E. Marsh, R. McMullan, B. P. Schoenborn and R. M. Stroud.

References

ABRAHAMS, S. C. (1975). Private communication.

BEAUCAGE, D. R., KELLEY, M. A., OPHIR, D., RANKOWITZ, S. SPINRAD, R. J. & VAN NORTON, R. (1966). *Nucl. Instrum. Meth.* **40**, 26–44.

BECKER, P. J. (1977). *Acta Cryst.* A**33**, 243–249.

DIMMLER, D. G., GREENLAW, N., KELLEY, M. A., POTTER, D. W., RANKOWITZ, S. & STUBBLEFIELD, F. W. (1975). *The Brookhaven Reactor Experiment Control Facility – A Distributed Function Computer Network.* Report BNL 20627, Brookhaven National Laboratory, Upton, New York.

LAWRENCE, J. L. (1977). *Acta Cryst.* A**33**, 232–234.

SCHNEIDER, J. S. (1977). *Acta Cryst.* A**33**, 235–243.

SPARKS, R. A. (1973). *Computational Needs and Resources in Crystallography*, pp. 66–75. Washington: Natl. Acad. Sci.

SPARKS, R. A. (1976). Proc. 1975 Prague International Summer School on Crystallographic Computing. Published as *Crystallographic Computing Techniques*, edited by F. R. AHMED. Copenhagen: Munksgaard.

Acta Cryst. (1977). A**33**, 20–24

Criteria for Selecting and Interconnecting Micro, Mini, Midi and Maxi Computers

BY PETER D. JONES

Network Systems Corporation, 6820 *Shingle Creek Pathway*, *Brooklyn Center*, *Minnesota* 55430, *USA*

(*Received* 4 *March* 1976; *accepted* 23 *July* 1976)

Four commonly accepted computer classes today are: micro, mini, midi and maxi. The dominant parameter which gives rise to this categorization is price, which varies from a ten dollar microprocessor chip to a ten million dollar maxi. This paper attempts to identify other parameters associated with each class in order to provide a mechanism for determining optimization of computer use. Trends towards increased parallelism at both ends of the spectrum are discussed. Finally, a method is described for interconnecting all classes of computers and peripherals at a site to form an installation network.

Introduction

The author's background includes the design and systems application of maxi computers over many years and recently the design and application of microprocessors. This experience led the author to concentrate the main thrust of the paper on the two ends of the computer class spectrum.

The computer structure classification of micro through maxi is one adopted by the trade journals

rather than one of academic significance. Nevertheless, it affords a mechanism for assessing the applicability of various computer types to perform given functions. The failure to select the right computer to do a given job can mean either the task does not get done (if the computer is underpowered) or there is wastage of money (if the computer is overpowered).

Classification of computers into a class structure is a dynamic process. The parameter values of Table 1 will be quickly outdated. At the high end, S. R. Cray claims (unpublished, 1974) he obtains a factor of five in speed every four years. At the low end, the cost of microprocessor chips is falling constantly from $100 in 1973 to around $10 (if purchased in quantity) today (1976). Yesterday's maxis are today's midis and tomorrow's minis. This is illustrated by the increasing number of installations which are using older generation maxis to 'front end' their new ones for file and message handling. Despite the transitory nature of the computer scence, an attempt at some comparison of parameters across the computer spectrum appears beneficial, even if the conclusions are strictly valid only for a single point in time.

Because of the relative newness of microprocessors, readers are referred to the following *Inst. Electr. Electron. Eng. Comput.* issues which are devoted to the design, application and programming of microprocessors: August 1974; August 1975; October, 1975; January, 1976.

Computer classification

Table 1 is a classification summary for computers, covering five basic parameters. To the author's knowledge this classification scheme is not widely used at this time. Price, instruction speed and memory capacity are recognized parameters and found frequently in computer literature. Results per second and input/output bandwidth are not so familiar. The author contends that these latter parameters are important in assessing the functions each computer category can best perform. There are other parameters (such as physical weight, power consumption and size), which are not considered here but which are important for certain applications.

The values chosen for the parameters can vary significantly and it is suggested readers insert values from their own experience. The table serves to illustrate the point that when a computer is being considered for an application, all relevant parameters associated with that application should be analyzed. Quite often this is done in a gross way by executing benchmark programs. A listing of basic parameters affords a second assessment. The following remarks are meant to highlight some of the implications of the parameter values given in the table, in particular the maxi/micro ratios.

(1) Micros are within the budget range of every laboratory, maxis are outside the financial possibility for most. Thus maxis appear best suited to find their place in service centers and shared centralized agencies. The level of computer selected for a site can be based on the minimum necessary to handle the largest program at that site. If there are a number of programs of a similar large size, then there are two possibilities: purchase a number of computers, one for each program, or purchase a single computer to handle all programs. The question of economy of scale is posed. One advantage of the 'more than one' approach is the availability of other computers, should one computer fail. In the 1940's Herb Grosch asserted that the power of a computer system increased as the square of its cost. This principle is now known as 'Grosch's Law'. The law has been empirically tested by a number of investigators and found to be generally true. The reader is referred to Cotton (1975) for a comprehensive survey of this subject.

(2) The instruction rate of maxis to micros is much lower (15 to 1) than one might expect from the price ratio (5000 to 1). This instruction rate reflects a computer's ability to make sophisticated decisions when a small amount of data is involved. Thus micros should find their place in instrumentation and control. However, one should exercise caution in comparing instruction rates between computers; a typical 8 bit instruction in a micro does very little compared with, say, the 64 bit edit instruction in the IBM 370 series.

(3) Results per second relates to work done. There are many ways of defining this, the results per second column in the chart reflects floating-point multiplies per second, which is the base criterion in much scientific analysis. Usually maxis are designed to optimize floating-point arithmetic whereas micros, at least at present, do not attack this area. Note also that vector processing, where one instruction operates on a range of elements, enables the result rate to overtake the instruction rate by a considerable margin.

Table 1. *Classification summary*

	Purchase ($10^3$$)	Million instructions per second	Million results per second	Memory capacity (10^3 bytes)	I/O bandwidth million bits per second
Micro	1	1	0·01	1	10
Mini	20	1	0·1	32	20
Midi	500	3	3	1000	50
Maxi	5000	15	50	8000	1000
Ratio Maxi/micro	5000	15	5000	8000	100

Rozwadowski (1973) has sought ways of formally defining and measuring work output of computers and the concept of 'computer work' is important to discussions on cost performance estimates.

(4) Memory is usually the dominant factor in computer pricing. It typically occupies two thirds of the hardware volume and represents about the same proportion of cost. In the Cray I computer, for instance, eight of the compartments are for memory and four for processing logic. The Cray I machine has one million 64 bit words of memory and an element of data can be entered or removed in 50 ns. Microprocessors usually have $1K$, $2K$ or $4K$ ($K = 1024$) 8-bit bytes of memory with access times the order of a μs. The memory address field length in small computers is usually 8–16 bits, in large ones the common range is 24–48 bits. The point here being that, even if it were physically possible to attach more memory to micro and mini computers, the memory limit is often set by what can be addressed in the instruction format. There are ways of getting around address limitations but usually at the cost of increasing program complexity and delays in accessing the memory extension.

The relationship between memory size and processor speed is important. W. J. Worlton of the Los Alamos Scientific Laboratory has suggested (unpublished, 1970) that, in his experience, a balanced machine requires the ratio of the memory capacity in bits to the processor speed in instructions per second to lie between one and five. This means a one million per second instruction processor should be supported by a one to five million bit memory. Most midi and maxi computers fall within this range, and most mini and micro computers do not. Thus in laboratories such as Los Alamos, simulation of physical phenomena is done on maxi computers, with mini computers providing system-support functions.

(5) Input/output bandwidth is a measure of how rapidly the processor can be fed with data. Most scientific analysis involves processing vast amounts of data and hence the ability to solve many problems is highly dependent on how fast this data can be moved. Micros are extremely limited whereas, for instance, the Control Data STAR-100 computer handles $9{\cdot}6 \times 10^9$ bits per second internally and, independent of this, $3{\cdot}2 \times 10^9$ bits per second is available for input/output. Clearly the more processing per unit of data the less dependent the problem solution is on input/output speed. In the reverse direction, commercial data processing, in which simple comparisons only are often made on the data, is very dependent on peripheral and channel speeds. Input/output bandwidth is the aggregate speed of all the computer channels.

Parallelism

The challenge of parallelism is well expressed by S. R. Cray in the introduction to a book on the 6600 (Thornton, 1970) where he says: 'This book describes one of the early machines attempting to explore parallelism in electrical structure without abandoning the serial structure of computer programs. Yet to be explored are parallel machines with wholly new programming philosophies in which serial execution of a single program is abandoned'.

Another pertinent remark on this subject is by Graham (1970): 'The parallel and pipeline designs will produce computers that, when used to their maximum, unquestionably will be faster than more conventional computers. However, formulating problems and writing programs which will use the new machines to anything approaching their maximum capabilities will prove a severe and, perhaps on occasion, an overwhelming challenge to the creativity of all concerned. Whether these huge machines will become the workhorses of computing hardware or go the way of the dinosaurs has yet to be seen. Their future hinges upon the skill of the users.'

Since these remarks in 1970 the trend towards parallelism has increased, particularly in large computers, but now also at the low end of the range. Maxis obtain their extremely high speed by a combination of high-speed components and parallelism in design. Illiac IV (McIntyre, 1970) has 64 processing elements, Texas Instruments' Advanced Scientific Computer (Watson & Carr, 1974) has four arithmetic pipelines, Goodyear's STARAN (Batcher, 1974) has up to 32 array modules, each containing 256 small processing elements. To obtain their optimum performance the STAR-100 computer (Purcell, 1974) from Control Data and the Cray I machine (*Cray I Reference Manual*, 1975) depend on processing ordered vectors rather than randomly located scalar quantities.

A basic question related to these machines is: 'Is parallelism simply a computer technique for obtaining speed or does it represent a fundamental approach to problem solution?' The question has still not been answered satisfactorily, though an attempt to do so was made last year at an important conference on *Programming Languages and Compilers for Parallel and Vector Machines* (1975). Whilst attempts are being made to develop compilers that are smart enough to extract and apply the parallelism implicit in existing languages, such as Fortran, it appears that, at least in the near future, programmers will be required to explicitly match problem solution and hardware parallelism. What is needed is a satisfactory language for 'parallel thinkers.' APL (Iverson, 1962) appears the best language so far for describing and processing vectors. The issue is an important one. For instance, in the STAR-100 computer a problem solution handled in vector mode can be an order of magnitude faster than the same solution processed in a scalar fashion. Some maxi computers, such as the Control Data 7600 and IBM's 360/195, are not so sensitive to parallel or vector operation, but the trend in more recent maxis is to be much more sensitive to the methods used for problem solution. Thus scientists need to be extremely

cautious in assessing the cost effectiveness of one maxi computer against another. Minis and midis so far have not been sensitive in this area and this is one of their advantages. In terms of growth one approach is to stay within the same computer category, taking advantage of new technology as it comes along. Growth from one computer category to another can involve many changes in programming techniques brought on by a vastly different set of parameter values and new concepts, such as parallel programming. Growth within a category tends to be evolutionary, whereas moving from one category to a higher one contains more of a risk.

So far the discussion has been at the top end of the computer range. Because of current research into microprocessor arrays (Widdoes, 1976; McGill & Steinhoff, 1976) and mini networks (Ashenhurst & Vanderohe, 1975) the same problems associated with parallelism in maxis may come to the micro and mini area.

Site networks

A key question is: 'Given a mixture of computers on a site, how should they be connected together?' It appears there are many benefits from what is called a site network, in which all site elements can communicate with each other. One benefit is the sharing of a common data base by a number of computers. It is interesting to note that, historically speaking, telephone networks have concentrated on switching and not on speed; whereas in computing, channel design has concentrated on speed and not on connectibility of elements. What follows is a description of one method of achieving a site network; this method is geared to the interconnection of high-speed computing elements which generally require high traffic flow.

Table 2. *Transmission performance characteristics*

Speed (nominal) (Megabits per second)	Distance (ft)	Drops
50	500	16
25	700	16
12	1000	16
6		16
3		32
1·5		64

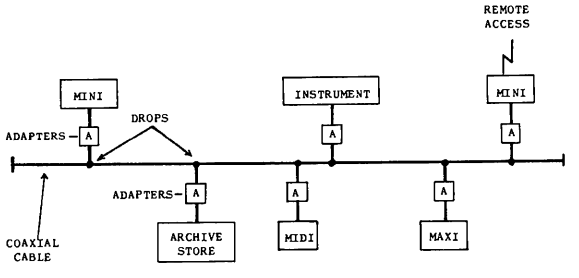

Fig. 1. Sample site network.

Other methods of achieving site or local networks are described by Farber (1975), Wulf & Levin (1975) and Fraser (1975).

The basis of the design is the transfer of serial data along coaxial cable at speeds between one and fifty million bits per second, and distances up to a mile. The number of allowable 'drops' on the cable varies with the speed and distance. 'Drops' are connection points on the cable at which external elements are plugged into the transmission scheme. Some points from a graph of these parameters are shown in Table 2.

Computer channels and peripheral controllers, which include instruments, represent the two main classes of system elements to be interconnected. At each connection there must be an 'adapter', which interfaces the channel or controller to the coaxial trunk. Fig. 1 shows a typical site network. The adapter is physically contained in a small box (20″ × 18″ × 10″) and consists logically of three sections: trunk interface, buffer and control, and device interface.

The trunk interface performs the following functions: transmit and receive cable signals; serial–parallel conversion; process line protocol including error checking; generate automatic responses. Up to four trunks can be connected to an adapter but only one can transmit through the adapter at a time. If another transmission for the adapter occurs at the same time it will receive a 'busy response'. The part of the trunk interface which drives the cable transmission is the transceiver. This operates at a fixed rate which is set by the manufacturer. The line protocol used is a variation of IBM's synchronous data-link control (SDLC). Standard SDLC is an 'electronic envelope' into which data or messages are placed. The transmission medium uses this envelope, which contains a source and destination address, to transfer strings of data bits of varying lengths without regard for the content of the data. In the network system, this link control requires the destination unit to respond to the source after every message, so that all transfers are interlocked, block by block.

In Table 3, the fields marked by asterisk are essentially identical to IBM's SDLC. The three added fields include an access code, a from address, and a checkword following the header. Standard SDLC assumes a single circuit connection from source to destination, either hardwired or established in advance of transmission by a dial-up procedure or equivalent. In contrast, the network has a continuous structure that does not use circuit switching. Since a given link can have more than one source, the from address is necessary.

Table 3. *Transmission frame format*

*Flag	8 Bits	Header checkword	16 Bits
Access code	16	*Data	*n*
*To	8	*Data checkword	16
From	8	*Flag	8
*Function status	16		

* Same as standard SDLC.

Finally, a checkword on the header is important, so that, if the header is faulty, the transmission does not continue wastefully through a useless data field, but instead can be aborted while recovery of the header is attempted. In accordance with standard SDLC, the data has its own checkword which follows the transmission. Thus the proposed line protocol, whilst necessarily different from standard SDLC, is closely related to it and as such should not encounter difficulties in being accepted by users.

The buffer, up to $9K$ ($K = 1024$) bytes, and control logic, which contains a fast microprocessor, coordinates trunk and device activity. The role of the microprocessor is to queue messages, allocate buffer memory and to format 'device messages' into correct trunk protocol.

The device interface matches the hardware cable and signals to the common trunk system. Clearly once a site element is interfaced to the network then it can communicate with every other element in the network. This is important because in the past a great deal of effort has been spent in interfacing different elements, but has stopped short of any generalized local switch system.

Contention between two adapters on a common trunk (the reverse of the previously mentioned contention of two trunks for one adapter) can arise if both units request the trunk within a 'window' of a few μs. If this occurs, neither adapter is likely to receive a response, or possibly both responses will contain checkword errors. Either way, the adapters wait for a prescribed interval of time, then try again. No two adapters have the same retry intervals, which are assigned on the basis of a priority scheme; therefore, contention between two adapters cannot occur on the retry.

One interesting aspect of the adapter for the present discussion is that it contains 1000 chips, of which 10% have something to do with the microprocessor. Of this 10%, about $\frac{1}{5}$ is the processor itself. The point is that in this application where a microprocessor performs an important function, only 2% of the total logic is the microprocessor.

Conclusions

Use of the computers (micro, mini, midi and maxi) in their proper mix will bring immense benefits to the scientist. Used out of their most efficient role they can lead to immense wastage of time, effort and money. If an installation is underpowered, users tend to spend too much time inventing novel methods to circumvent computer shortcomings. At this time the main function of the micro is in instrumentation and control. Minis perform a mixture of functions from special purpose processing to general purpose computing and data management. Midis continue to perform the general computing role cost-effectively. Maxis appear best suited to service bureaux and centralized agencies where they can be accessed by the maximum number of scientists. The advent of large on-line archival stores, such as the IBM 3850 and Control Data's Mass Storage Facility, support the concept of large centralized computer and data bank installations, shared by many geographically remote scientists.

With ever decreasing hardware costs we can expect more performance per dollar in each computer category. Advances in technology seem to favour all categories equally. At the high end, processor speeds and large random-access memories facilitate the accurate simulation of increasingly complex physical phenomena. At the low end, innovative computer structures with arrays of microprocessors are being investigated.

Scientists who keep abreast of trends in computer architecture and who are prepared to adopt new programming techniques at the risk of some initial failures will find themselves moving into increasingly rewarding areas of computation.

Finally, one area of progress is the interconnection of site elements *via* coaxial cable, at high speeds and high reliability, over longer distances (up to a mile) than is now possible with conventional channels. This site network allows for the efficient interconnection of all kinds of processors (including new special purpose ones), peripherals (especially storage) and instruments, enabling computers to perform their most efficient function and hand over to other elements in areas where their efficiency falls off.

References

ASHENHURST, R. L. & VANDEROHE, R. H. (1975). *Datamation*, **21** (2), 40–44.

BATCHER, K. E. (1974). *AFIPS Proc.* **43**, 405–410.

COTTON, I. W. (1975). *Assoc. Comput. Mach. Comput. Surv.* **7** (2), 96–111.

Cray I Reference Manual (1975). Cray Research Inc., 7850 Metro Parkway, Suite 213, Minneapolis, Minnesota 55420.

FARBER, D. J. (1975). *Datamation*, **21** (2), 44–46.

FRASER, A. G. (1975). *Datamation*, **21** (2), 51–56.

GRAHAM, W. R. (1970). *Datamation*, **16** (4), 68–71.

IVERSON, K. E. (1962). *A Programming Language*. New York: John Wiley.

MCGILL, R. & STEINHOFF, J. (1976). *Assoc. Comput. Mach. SIGARCH Conf. Proc.* **4** (4), 46–51.

MCINTYRE, D. E. (1970). *Datamation*, **16** (4), 60–67.

Programming Languages and Compilers for Parallel and Vector Machines (1975). *Assoc. Comput. Mach. SIGPLAN Proc.*

PURCELL, C. J. (1974). *AFIPS Proc.* **43**, 385–387.

ROZWADOWSKI, R. T. (1973). *Assoc. Comput. Mach. SIGMETRICS Conf. Proc.*

THORNTON, J. E. (1970). *Design of a Computer, The Control Data* 6600. New York: Scott, Foresman.

WATSON, W. J. & CARR, H. M. (1974). *AFIPS Proc.* **43**, 389–397.

WIDDOES, K. C. (1976). *Assoc. Comput. Mach. SIGARCH Conf. Proc.* **4** (4), 34–39.

WULF, W. & LEVIN, R. (1975). *Datamation*, **21** (2), 47–50.