Microelectronics and Computer Science

The functional architecture of the computer has traditionally been shaped by the size of specialized components and concepts of how people think. Microelectronics is now eliminating these constraints

by Ivan E. Sutherland and Carver A. Mead

Computer science has grown up in an era of computer technologies in which wires were cheap and switching elements were expensive. Integrated-circuit technology reverses the cost situation, making switching elements essentially free and leaving wires as the only expensive component. In an integrated circuit the "wires," actually conducting paths, are expensive because they occupy most of the space and consume most of the time. Between integrated circuits the wires, which may be flat conducting paths on a printed circuit board, are expensive because of their size and delaying effect. Computer theory is just beginning to take the cost reversal into consideration. As a result computer design has not yet begun to take advantage of the full range of capabilities implicit in microelectronics. As we learn to understand the changed relative costs of logic and wiring and to take advantage of the possibilities inherent in large-scale integration we can expect a real revolution in computation, not only in the forms of computing machines but also in the theories on which their design and use are founded.

Why is it that computation theory needs to be revised? Suppose one sets out to develop some theories of computation, hoping to put them to work toward two ends: to establish upper bounds on what is computable and to serve as a guide to the design and use of computing machines. Such theories would presumably also advance understanding of computation processes and perhaps shed light on the nature of knowledge and thought. The theories might be based purely on mathematical reasoning or might also be based on fundamental physical principles. By mathematical reasoning alone one can prove many things about computers without resorting to physical principles. Only by attending to physical principles, however, can one make more quantitative statements about how long a computer of given physical dimensions must take to accomplish a given process, based on the fact that information cannot move around in the computer faster than the speed of light and that it takes a certain amount of matter, energy and space to represent one bit, or binary digit, of information with a given reliability.

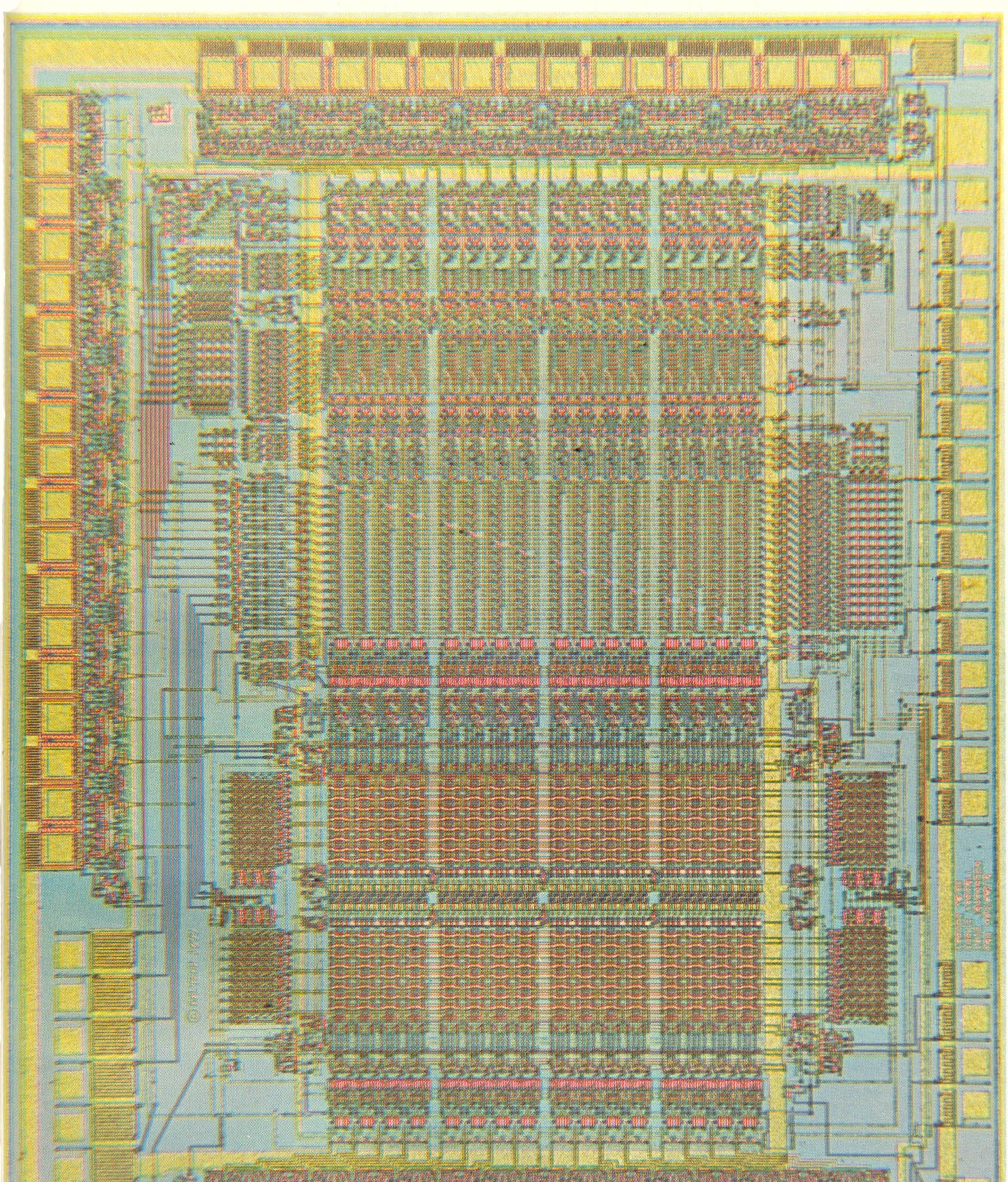
Computer science as it is practiced today is based almost entirely on mathematical reasoning. It is concerned with the logical operations that take place in computing devices. It touches only lightly on the necessity to distribute logic devices in space, a necessity that forces one to provide communication paths between them. Computer science as it is practiced today has little to say about how the physical limitations to such communications bound the complexity of the computing tasks a physically realizable computer can accomplish.

That is so in part because anyone who thinks of a computer as a logical machine that performs logical, numerical or algebraic operations on data will naturally think of the machine in terms of the mathematical notation relevant to those fields. In such notations the symbol x written in one place on the page is identical in meaning with the symbol x written in another place on the page. The idea that communication in space is required if such values are to be identical as represented in a computer storage device has no place in the notation. The notation itself focuses attention on the logical operations, reflecting the fact that human beings think most effectively about only one thing at a time. A mathematical proof is a sequence of steps we absorb over a period of time, and it is easiest to think of computing devices that also do only one thing at a time. The sequential approach to mathematics is not required inside a computer, but the mathematical approach we normally take to problems does not encourage us to think of approaches other than sequential ones for the solution of problems. Nearly all computers in operation today perform individual steps on individual items of data one after another in time sequence.

It was appropriate to ignore the costs of communication when logic elements were slow and expensive and wires were relatively fast and cheap. Sequential machines are appropriate to such technologies because they can be built with a minimum number of switching elements. We have been led—by natural inclination, by our accustomed notations for mathematics and by technology—to develop a style of computing machines and a body of computing theory both of which are rendered obsolete by integrated-circuit technology. We have been able to ignore the limitations placed by physical principles on communications inside computers because those communications did not slow down our operations appreciably and were only a small part of the cost of the machines we built. By making logic elements essentially free and leaving communication cost the dominant factor, integrated-circuit technology forces us into a revolution not only in the kinds of machines we build but also in their theoretical basis.

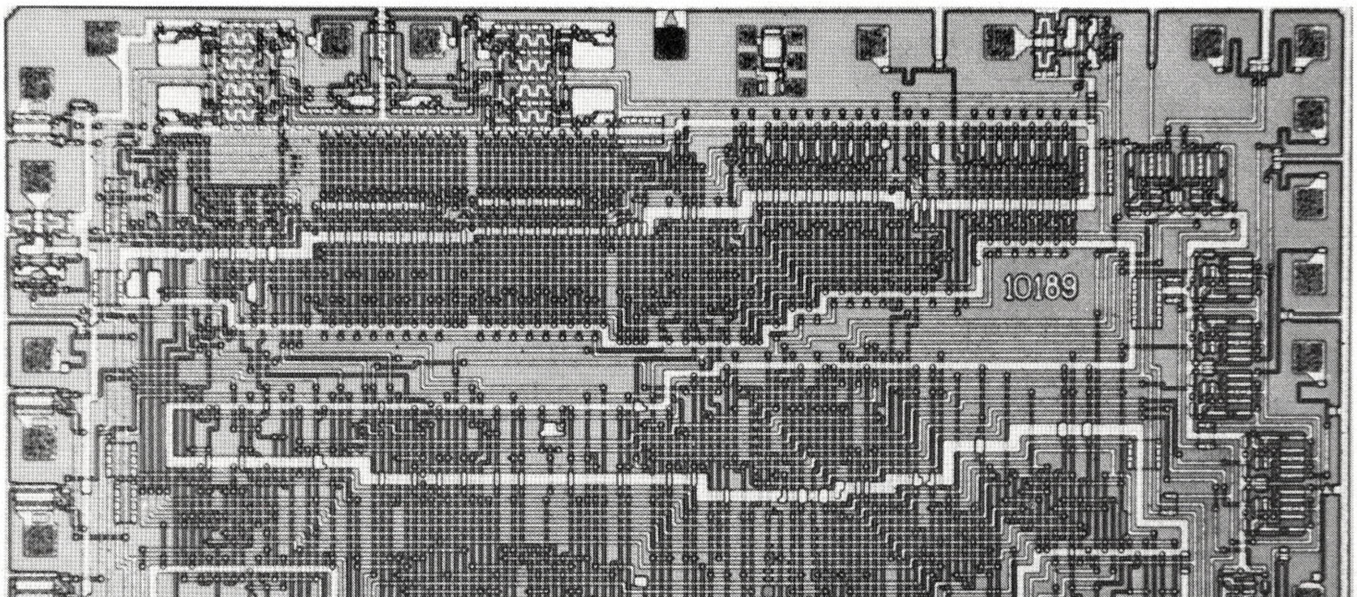
Developing a new theoretical basis

"OM" CIRCUIT, an experimental microprocessor designed by the authors at the California Institute of Technology, is notable for its high degree of regularity, which makes it possible to pack more logic and memory functions on the chip. The main body of the chip (omitting the communication interfaces at top and bottom) is made up of 16 nearly identical columns in four groups of four; each column represents one bit of a 16-bit computer. About 40 percent of the chip (*lower portion*) is memory; middle 20 percent is the "shifter" section and top 20 percent is the arithmetic section.

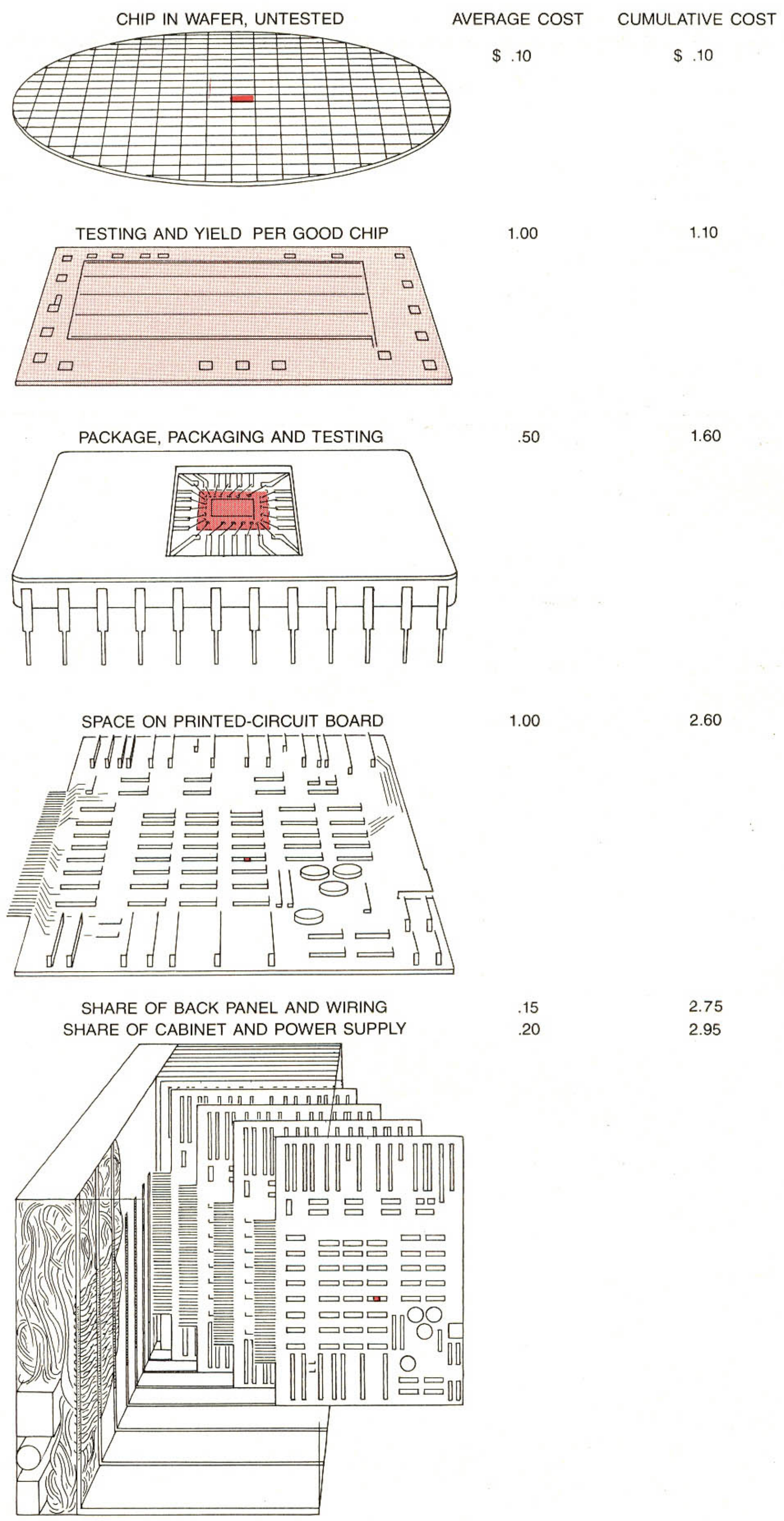


for computer science will not be easy; indeed, the task has been put off in part because it is very difficult to combine notions of logic with notions of topology, time, space and distance, as a new theory will require. In this article we shall outline some of the elements such a theory must include, first by examining the inadequacies of a simple existing theory applicable to small logic networks. Then we shall see how the changes in the relative costs of wiring and logic must change the nature of the computers built in the future. Finally we hope to outline some elements we feel belong in a theory of computation appropriate to the new structures. Such a theory will be quite unlike the present basis of computer science, and so we feel justified in describing as revolutionary the effect of integrated-circuit technology both on the design of computing machines and on the intellectual framework within which such machines are exploited.

Most computer-science curriculums include a course in switching theory, even though it is largely irrelevant to the present-day practice of computer design. Switching theory, which was developed to help design the relay-operated switching networks of automatic telephone systems, provided guides that enabled a designer to formulate a network with the minimum number of relays for accomplishing some given logical operation. It has been extended to the design of networks of newer kinds of logic elements, for example a logic network with



INTERCONNECTIONS among the logic elements of an integrated circuit have become more expensive than the elements themselves. Moreover, as the complexity of a randomly wired array of elements increases, the interconnections become longer and more numerous. Even at modest levels of complexity "wiring" occupies most of the available space on an integrated-circuit chip. This is a comparatively simple integrated circuit dating from about 1971. Note that the linear connectors running between active elements occupy most of the space.



	AVERAGE COST	CUMULATIVE COST
CHIP IN WAFER, UNTESTED	$ .10	$ .10
TESTING AND YIELD PER GOOD CHIP	1.00	1.10
PACKAGE, PACKAGING AND TESTING	.50	1.60
SPACE ON PRINTED-CIRCUIT BOARD	1.00	2.60
SHARE OF BACK PANEL AND WIRING	.15	2.75
SHARE OF CABINET AND POWER SUPPLY	.20	2.95

COST OF AN INTEGRATED CIRCUIT is a small part of the cost of a complete system. As is shown here, the cost of a single typical integrated-circuit die in a wafer is only 10 cents. Given about a 20 percent yield of good chips, after packaging and testing each good chip costs $1.60. Assuming that 100 chips are assembled on each of 20 printed-circuit boards, the cost per chip is almost doubled by each chip's share of board, back panel, cabinet and power supply.

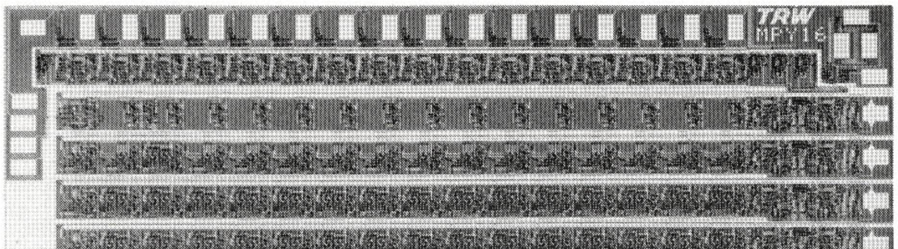
the minimum number of conventional logic gates.

There is no guarantee, however, that such a minimum-number network will occupy the minimum space in an integrated circuit or perform its task in the minimum time. Integrated-circuit designers find they can often add transistors to a design and thereby save space or time, because adding to the minimum number may simplify the pattern of conductors in the design and may speed up its operation. Switching theory does minimize the number of switching components, but it ignores the cost and delay of the communication paths. In today's technology the area of a circuit devoted to communication between elements usually far exceeds the area devoted to switching elements, and communication delays are much longer than logic delays. What is needed, therefore, is a theory that minimizes the cost of computational tasks, considering not only the cost in area and time of the switching elements but also the much larger area and time costs of transporting data from one place to another. Because switching theory as it is known today is based on an obsolete cost function it is largely useless for the design of integrated circuits.

Switching theory is even less useful at the level of design where one is combining integrated circuits into a larger system. In most cases it costs much more to test, package and interconnect integrated circuits than to manufacture the circuits themselves. These costs are largely independent of the particular function of the circuit involved. Even if the cost of the circuits is ignored, communication from one integrated-circuit chip to another is much slower than communication on a single chip. Given a catalogue of standard circuits, there is great motivation to introduce more complex integrated circuits because fewer of them are required, so that the large cost of mounting and interconnecting them is reduced. In fact, designers often specify integrated circuits containing superfluous elements because there is no cost advantage to eliminating the unneeded switching elements. Switching theory has nothing to say about these important issues of cost and speed.

Although the cost of communication has so far found no real place in the theoretical results of computer science, it does play a role in the thinking of practical designers. Seymour Cray, the designer of many of the most powerful computers, cites the "thickness of the mat" and "getting rid of the heat" as the two major problems of machine design. It is obvious that controlling the geometry of the interconnections is essential. If connections can be made to follow regular patterns, they can be produced by less expensive methods and can also be

THICK MAT OF WIRES covers the back panel of a large general-purpose computer, in this case the Cray Research, Inc., CRAY-1. Moving data over the wires takes time and costs money, and the thickness of the mat makes repairs difficult. Arranging elements of a computer so that wires are all parallel would greatly reduce the complexity and thickness of the mat.



REGULARITY is a characteristic of memory circuits and of certain arithmetic circuits, such as this 16-bit multiplier array made by TRW Inc. The chip, about .28 inch square, contains more than 18,000 transistors and resistors. Regularity makes for high logic-element density.

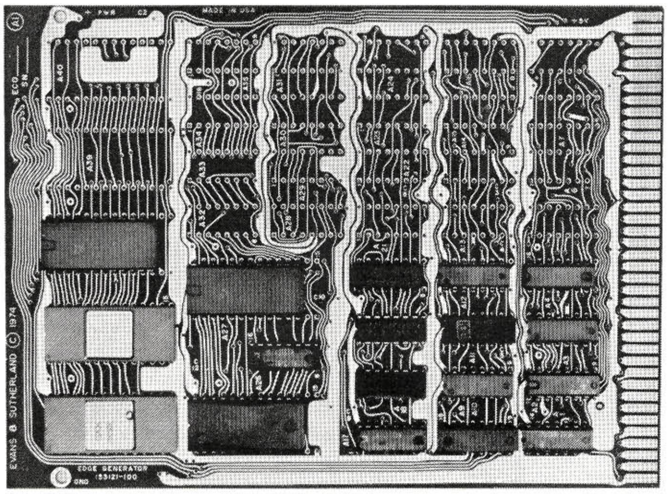
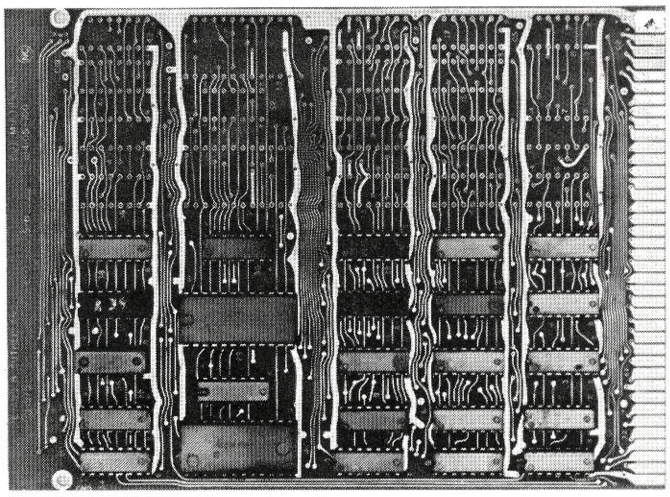
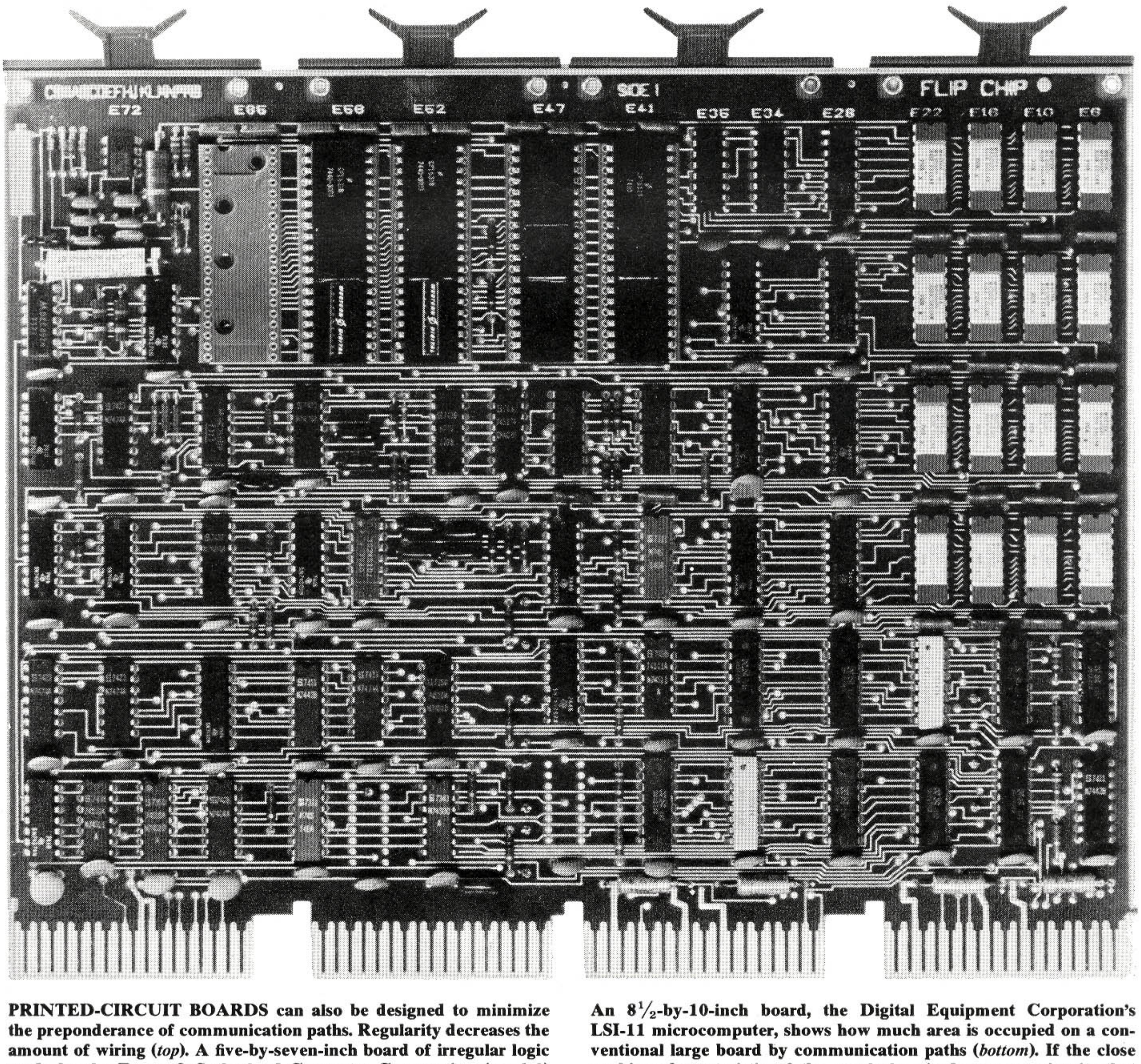
made to occupy less space and so be faster.

If the geometry of interconnection paths is not carefully controlled, the space required for them grows more than linearly as the number of logic elements to be connected is increased. This nonlinear growth comes about because bigger systems require more wires, which are on the average also longer. Because the interconnection paths grow both in number and in length the total area or volume devoted to communication becomes disproportionately larger: to interconnect twice as many randomly placed devices requires four times as much communication space. To accommodate greater wiring space larger printed circuit boards must have wider spacing between components than small boards have; Los Angeles suffers more from freeway congestion than Plains, Ga., does.

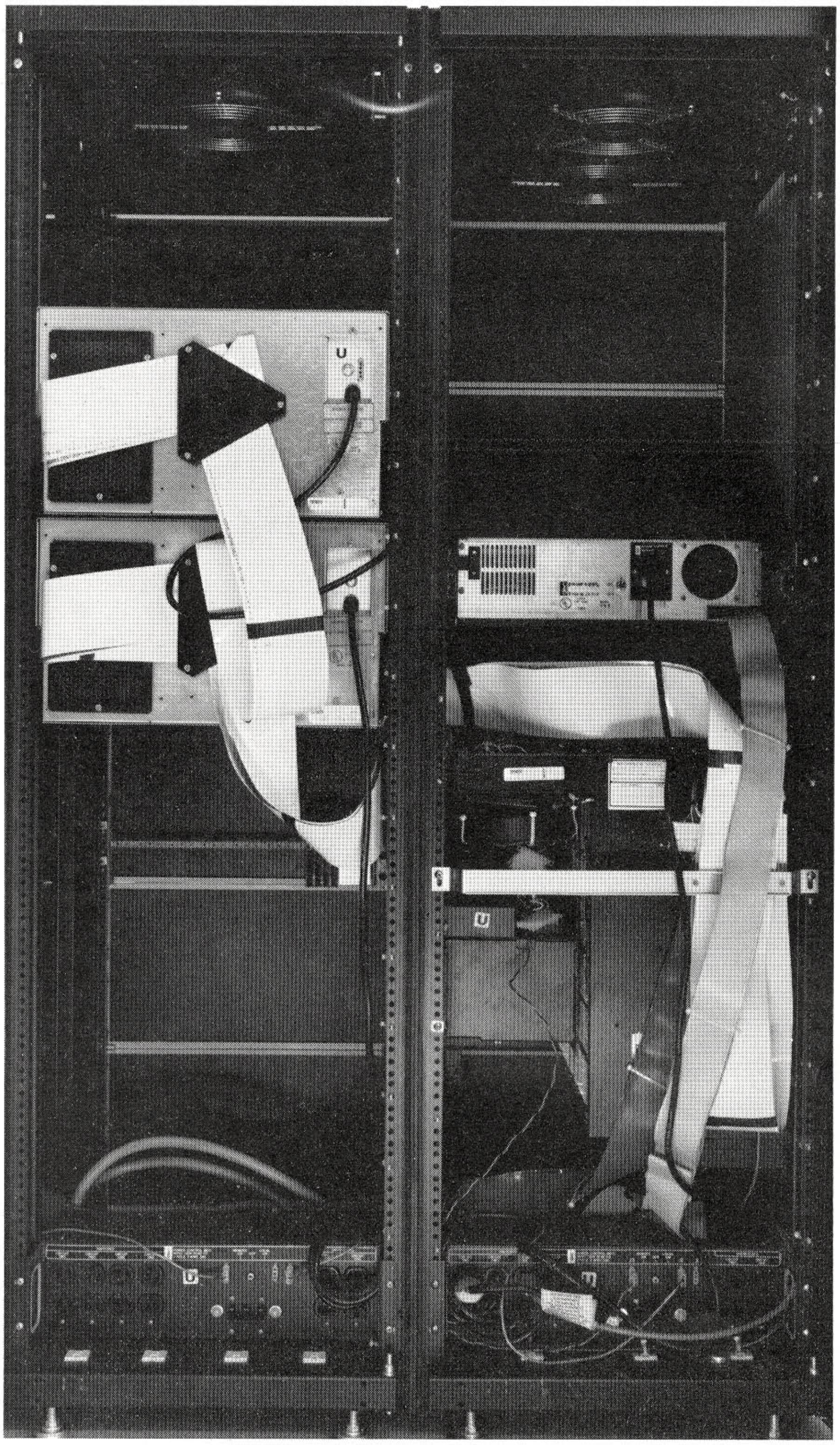
Not only do longer communication paths occupy a disproportionate amount of space but also they function more slowly than short ones. That is because signals traveling even at the speed of light take some time to travel down a path and also because longer paths store more energy. (Inside integrated circuits the speed limit set by the speed of light is not yet an important issue because the distances are short compared with the switching times of the logic elements; the energy-storage delays, however, are important.) Before a signal path can be switched from one electrical state to another, the energy stored in the path must be removed and converted into heat. One must either design a larger driving circuit to provide for the larger power required to switch long wires quickly or suffer the delays of passing the larger amounts of energy through a less powerful driver. More powerful drivers must themselves be driven, and they are therefore not only larger in area but also inherently slower than small drivers.

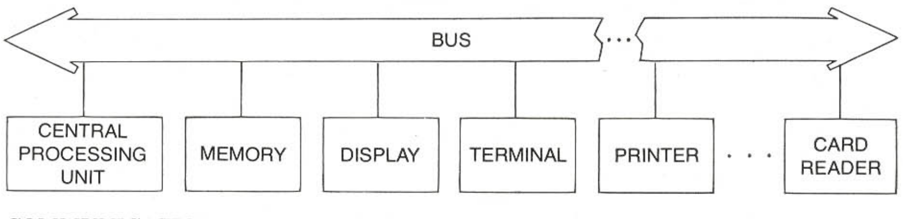
Moreover, the heat generated by the more powerful drivers must be dissipated in some structure, which itself occupies space. It is quite possible that the signaling energies required in a given technology and the size of the structures provided to dissipate heat may set an upper limit to the complexity of the systems that can be built in that technology. Above such a limit the increase in wire length required to provide the space to house what is required to drive longer wires may exceed the original increase in length of wires that was made possible by the larger drivers! There is so far no theory addressing the limits to speed and complexity that may be imposed by this possibility.

The disproportionate growth of interconnections can be avoided by building

PRINTED-CIRCUIT BOARDS can also be designed to minimize the preponderance of communication paths. Regularity decreases the amount of wiring (*top*). A five-by-seven-inch board of irregular logic made by the Evans & Sutherland Computer Corporation (*top left*) is compared with a more regular memory board of the same size (*top right*). The larger the board, the greater the preponderance of wiring. An $8\frac{1}{2}$-by-10-inch board, the Digital Equipment Corporation's LSI-11 microcomputer, shows how much area is occupied on a conventional large board by communication paths (*bottom*). If the close packing characteristic of the regularly wired memory circuits that can be seen in the top right portion of board could have been attained throughout the board, the board would have been only half as big.



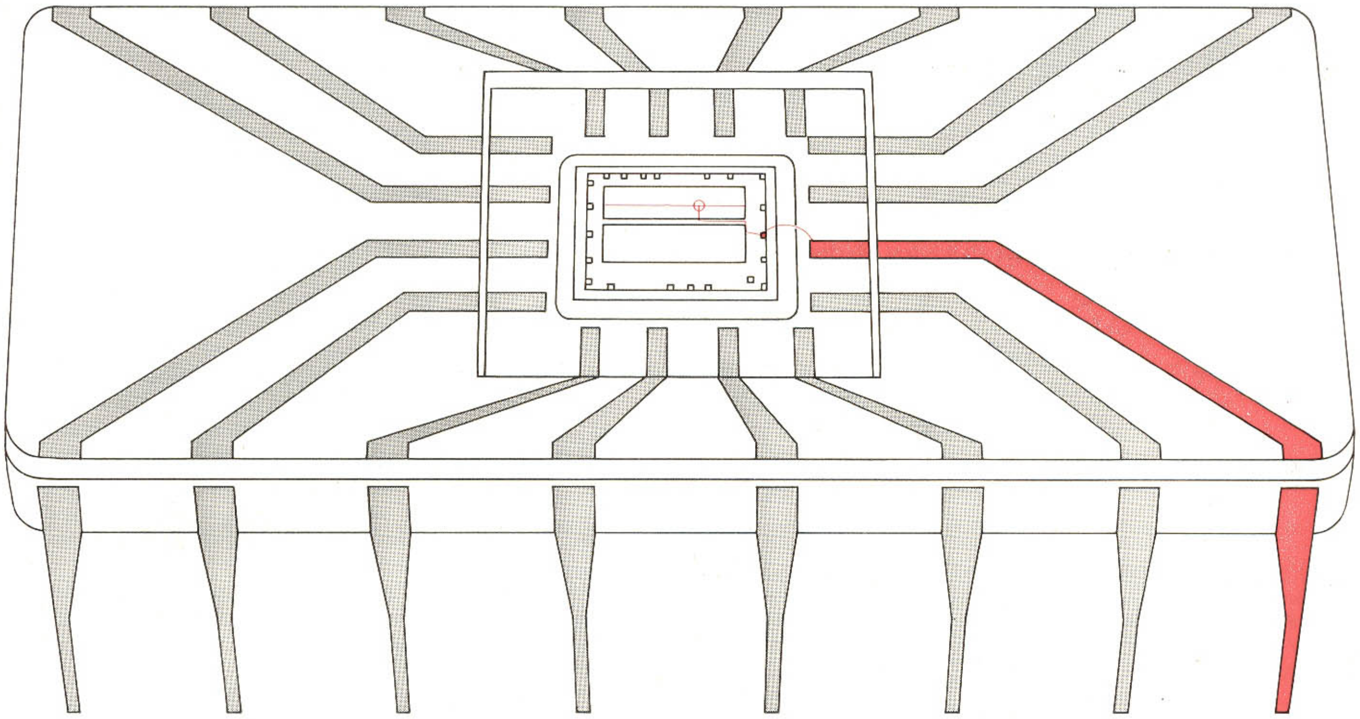


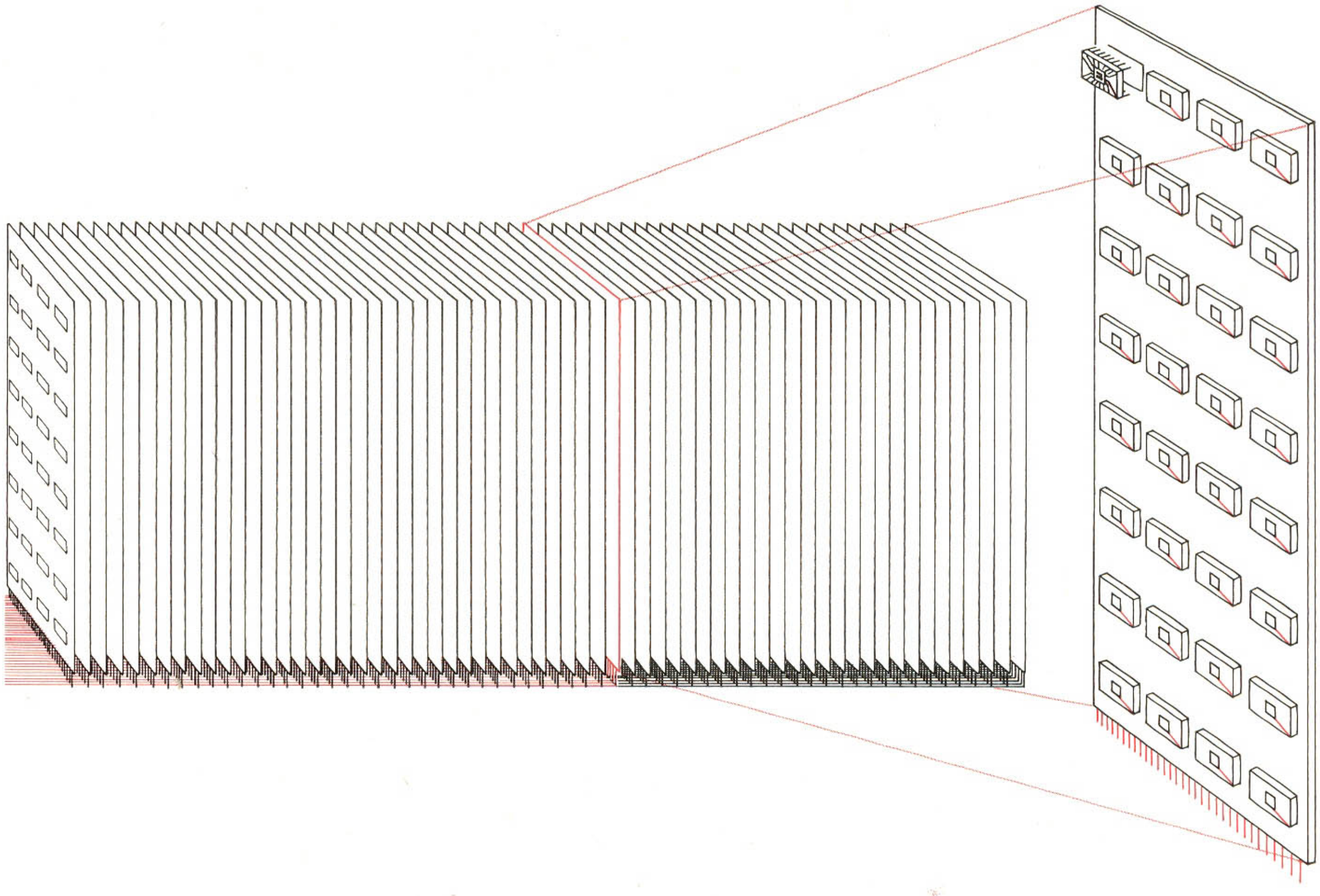
COMMUNICATION "BUS" connects a computer's central processing unit to memory modules and other peripheral units (*top*). It is typically a flat cable of between 20 and 100 long wires that are tapped as they pass through each of the connected units. Photograph shows Digital Equipment Corporation's UNIBUS (*wide, light-colored flat cable*) connecting two disk memory units (*upper left*) with central processing unit (*lower right*) of DEC's PDP-11/40 computer.

very regular patterns of interconnection. There is already a trend toward very regular wiring patterns for integrated circuits and the interconnections among circuits. Read-only memories, for example, implement complex and irregular logic functions with a simple and very regular integrated circuit pattern. This regularity is desirable not only because it makes the specification of such functions simple but also because it may be the most efficient layout from an interconnection point of view. We believe regular patterns of wiring will play an increasing role in future designs. In part, computer science will become the study of the regularity of these structures.

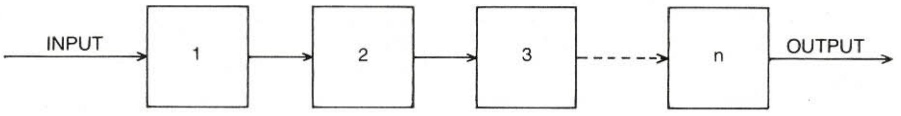
The architecture of a typical computer includes a single logical processing element that communicates with a random-access memory through 20 to 100 long wires combined into a "bus," which, like its namesake, provides public transportation for data but is actually more like a telephone party line. The communication bus is often a flexible cable 50 to 100 feet long. A signaling protocol is specified for the bus so that all the units to which it is connected communicate in a common way and avoid interfering with one another. The great advantage of a bus structure in a computer is that any unit connected to the bus can communicate directly with all other units. Moreover, the protocol and the bus structure may survive several generations of hardware development, so that a line of computing equipment can adopt new storage devices, new input-output units and even new processing elements. In addition, the number of switching elements devoted to communication in each unit on the bus is minimized because each unit needs to communicate with only the one bus to send messages anywhere.

The drawback of the bus structure is that it provides a communication bottleneck. Consider a typical computer with, say, one million words (32 million bits) of integrated-circuit storage built out of 2,048 circuits that store some 16,000 bits each. Any one reference to memory can potentially sense the values of 128 bits on each of the 2,048 integrated circuits constituting the memory. Of these quarter-million bits to which access is obtained on the integrated circuits only 2,048 (one from each integrated circuit) are delivered outside the integrated-circuit package, and of these 2,048 only 32 are delivered over the communication bus to the logical processing unit of the computer. It is assumed that the communication bus connects the memory and the logical processing unit; we assert that in fact it separates them. Each memory access in a large computer wastes access to many thousands of bits

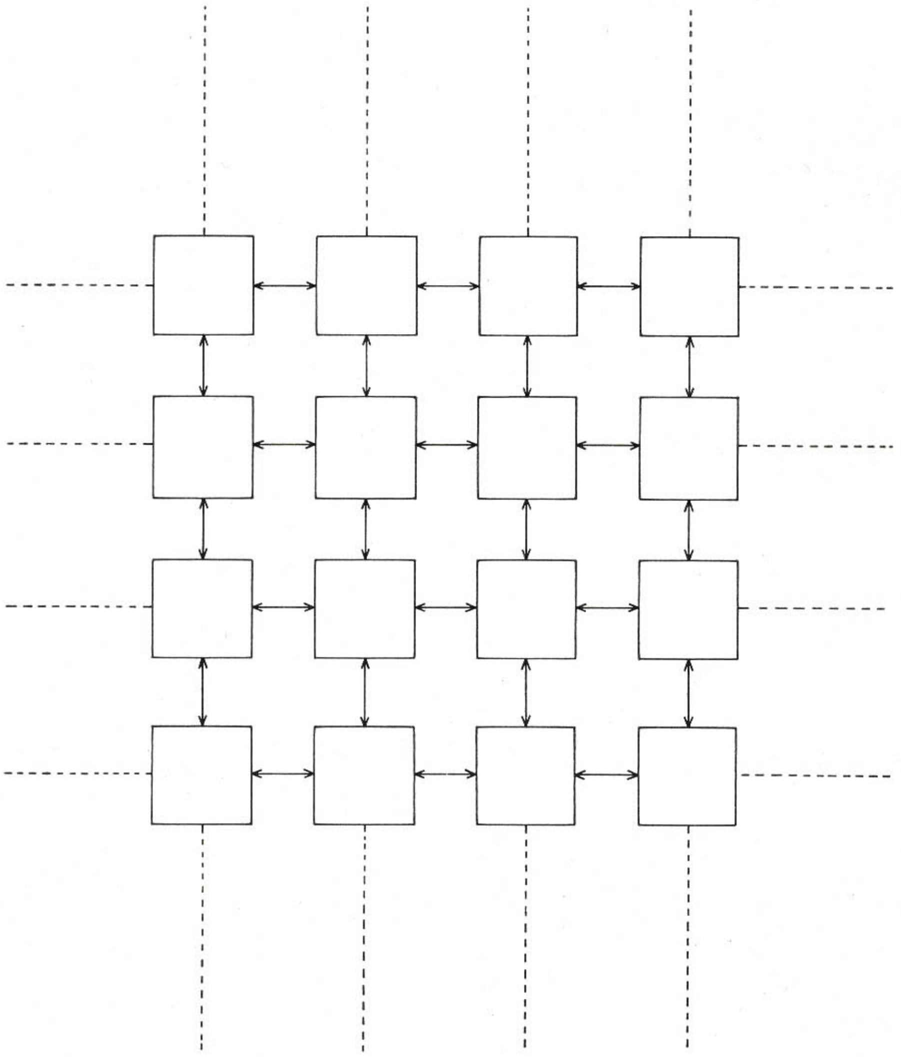




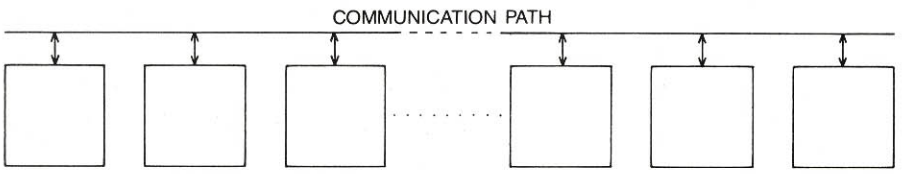
TYPICAL MEMORY CHIP has 16,384 bits arranged in a 128 × 128 array (*top*). An entire row of 128 bits can be accessed at one time, but a selector enables only a single bit to pass to an output pin (*dark color*). A typical memory system is made up of 2,048 such chips, say 64 groups of 32 (*bottom*). Only 32 chips can place their outputs on the 32 wires that join the bus to the central processor. Of the 262,144 (128 × 2,048) bits that moved less than a millimeter on each chip, only 2,048 moved three millimeters to get off their chip and only 32 moved a meter to the processor. In other words, the bus utilizes only about an eight-thousandth of the memory chips' available "bandwidth."



"PIPELINE" PROCESSOR is one of three kinds of parallel processor, illustrated on this page, that have been effective. In a pipeline processor data are passed along from one specialized processing element to the next, with each element performing a successive operation on the data. The pipeline is analogous to an assembly line: all operations are conducted simultaneously but not on the same material. The pipeline configuration is optimum as long as the same basic type of operation is to be performed; it is less effective when the operations are variable.



ARRAY PROCESSOR is effective when much identical processing is to be done on many items of data. All the processors receive the same instructions, like a company of soldiers drilling "by the numbers." The limitation here is that individual computations must depend only on the data in a particular element and its immediate neighbors. This can be effective, however, in operations such as weather simulation, where local atmospheric interactions are significant.



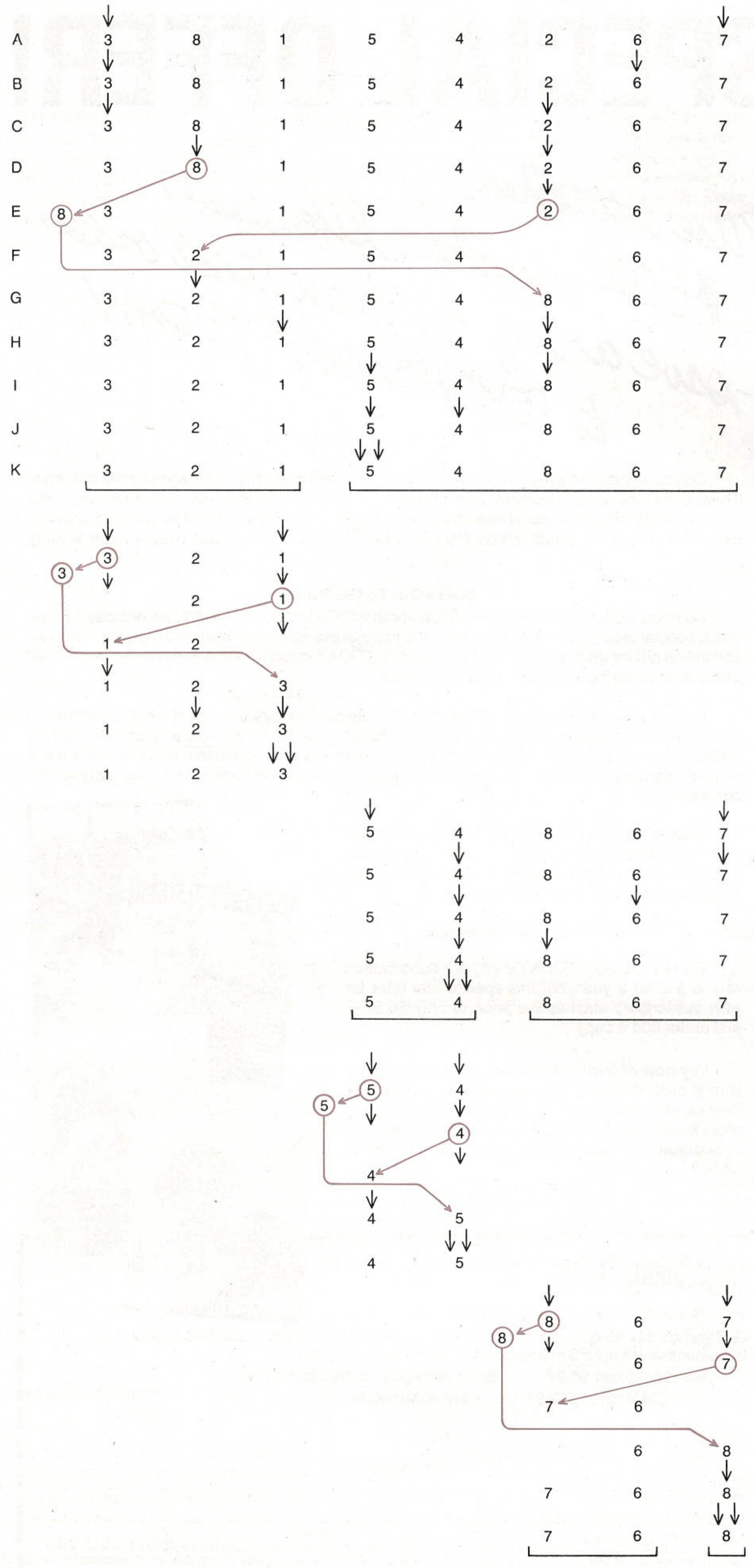
INDEPENDENT PROCESSORS connected by a communication path constitute the most flexible arrangement for parallel execution of different operations. Tasks are given to each processor as is required, as they are to the individual workers in a cottage industry. The system works best when each element can do much processing and need not communicate much with other elements; bottlenecks develop when tasks require elements to wait for the party line.

by selecting only a few bits to send over the memory bus to the central processing unit. This waste is tolerated for two reasons. First, it simplifies our conception of the machine and matches it to our natural inclination to do one thing at a time. Second, it provides a single, simple interface between various parts of the machine.

We pay a high price for this convenience. In an age when memories and logical processing elements were made by different technologies, we had little choice. Now, however, with the silicon integrated circuit dominating both the memory and the logical processing tasks in computers, there is little justification for continuing to accept such waste. Now it is possible to distribute the memory bus over many thousands of integrated circuits, in effect giving each logic element the memory it needs by moving information less than a millimeter from memory to processing facilities located together in the same integrated circuits on each of many thousands of chips. We are just beginning to explore systems with this unconventional architecture. To employ them effectively we must learn how to match the complexities of given problems to the simple fixed patterns of communication provided in the systems we can build.

Machines in which large numbers of logic elements operate simultaneously are called parallel processors. (To some extent, to be sure, every computing machine is a parallel processor. The separate bits that together represent a number are moved simultaneously on parallel communication paths; binary addition is performed by an adder circuit that operates on all the bits of the number at once; multiplication is performed either by sequential addition or, in faster models, by including a number of separate adder circuits and operating them in parallel. Levels of parallelism above basic arithmetic, however, are rare in today's computers.) The simplest form of real parallel processing now available has a few independent processors operating on a common memory; a typical large computing system has from two to a few dozen processors at work. Typically, however, these processors serve quite independent functions and their very existence may be hidden from the user. For example, a separate processor may be involved in communication with the user's keyboard, in operating magnetic-tape or magnetic-disk input-output units or in scheduling the resources of the central processor. Such "multiprocessing" systems have little impact on the user's algorithms.

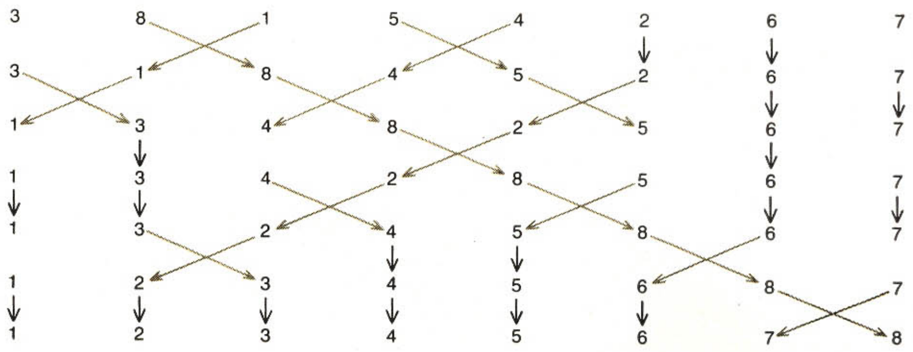
Three kinds of systems that can truly be classed as parallel processors have been built. In one of them, the "pipeline" processor, several processing elements,



each of which is specialized for some particular task, are connected in sequence. The work to be processed flows through these processors much as workpieces move along an assembly line. Communication is simple because information flows along a fixed pathway and has only a short distance to move between processing steps. A pipeline processor gains efficiency for the same reasons an assembly line does: functions are specialized and communications are minimized. Pipelining enables the arithmetic sections of very fast computers to process sequences of numbers with a high overall speed. Pipeline processors are less effective where the tasks to be performed are highly variable.

In a second form of parallel processor many identical processing elements are brought to bear on separate parts of a problem under the control of a single instruction sequence. Several such machines have been built, of which the largest and best known is ILLIAC IV. A modern parallel processor of this type was proposed at a recent Rand Corporation workshop on hydrodynamic simulations. In this hypothetical machine there would be 10,000 processors, each with arithmetic capability and memory, each built on a single integrated circuit and all under the command of a common instruction device. All the processors would execute commands in rigid lock-step. The processors would be arranged in a square array, 100 × 100, and each would communicate data only with adjacent processors to its north, south, east and west in the array, with relatively slow bit-serial communication on a single wire in each direction. We estimate that such a machine would take about five microseconds to communicate a single 64-bit number from one processor to its neighbor, which is very slow by today's standards. Of course, it could communicate 10,000

"QUICKSORT" is a typical sequential algorithm for arranging numbers in ascending order. Numbers pointed to by arrows are compared with the number farthest to the left. If the pointer farthest to the right indicates a number greater than the reference value (*row A*), it is advanced to the left until it rests on a number less than the reference value (*C*). Then the left pointer is advanced to the right until it rests on a number greater than the reference value (*D*). At that stage the numbers pointed to are interchanged (*E–G*). The process is continued until the pointers rest on the same number (*K*); at that stage all numbers to right of pointers are greater than the reference value, and all numbers to left are less than or equal to it. The same algorithm is then applied to each subset; to complete the sorting illustrated here requires five more steps than are shown. To sort *n* numbers requires $n(\log_2 n)$ comparisons of numbers that may be stored in distant locations. The communication cost is the dominant cost of executing the algorithm.


PARALLEL EXECUTION speeds the sorting task. In this algorithm adjacent members of number pairs are compared and are interchanged if the left-hand member is larger than the right-hand one. (In the first row a "pair" is defined as two numbers the left one of which is in an even column; in the second row the left member of each pair is in an odd column, and so on alternatingly.) Sorting the entire set requires $n^2/2$ comparisons, always of nearby numbers. Interchange sorting has been considered slow, but if comparison and interchange elements are attached to each memory element in integrated circuits, comparisons required for one sweep can be accomplished in one memory cycle and the sorting can be completed in n cycles at most.

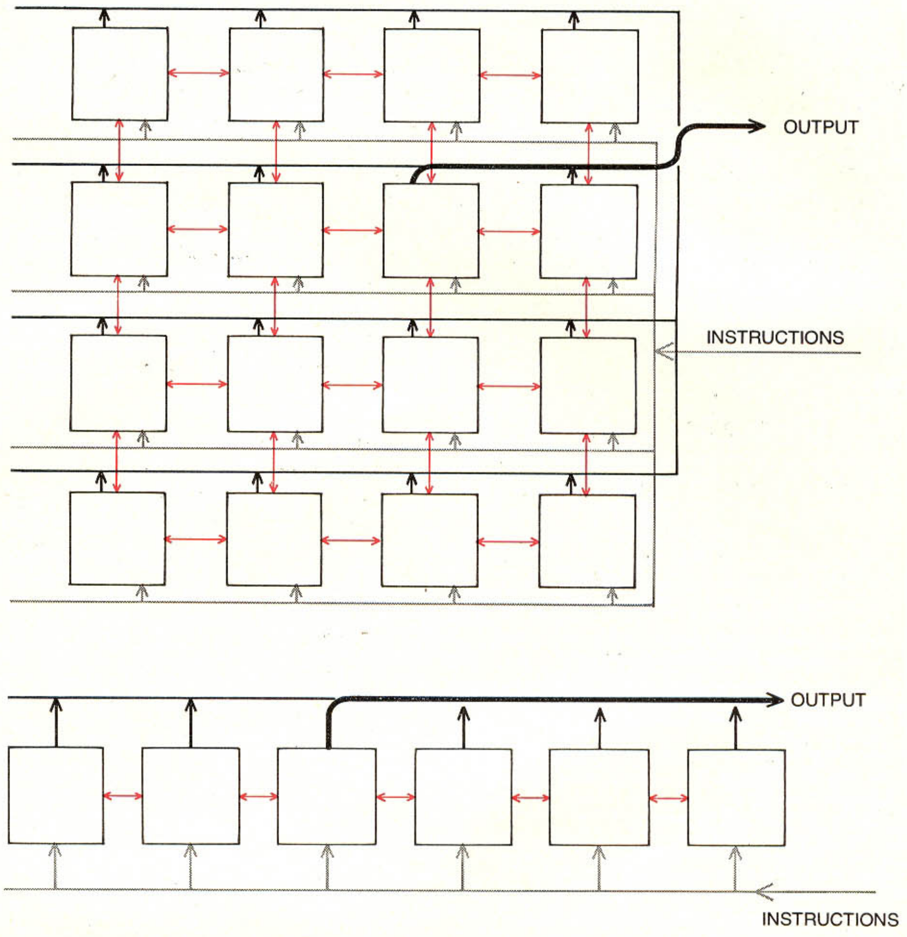
such numbers in any one five-microsecond period. It would also take about five microseconds to perform a multiplication, but again it could perform 10,000 multiplications in that time, for an average rate of two billion multiplications per second. Machines such as this one are called array processors or single-instruction-stream, multiple-data-stream machines. They are most suitable for highly regular tasks such as hydrodynamic computations, numerical simulation of the weather and the inversion of large matrixes.

A third type of parallel processor is one where separate, independent processors under separate, self-contained control structures perform independent parts of the task, communicating data and instructions as is required. The advent of the microprocessor has, of course, suggested to many people the possibility of making systems consisting of thousands of separate microprocessors and having them work in concert on large tasks. Few such multiple-instruction-stream, multiple-data-stream machines have been built, and their properties are poorly understood.

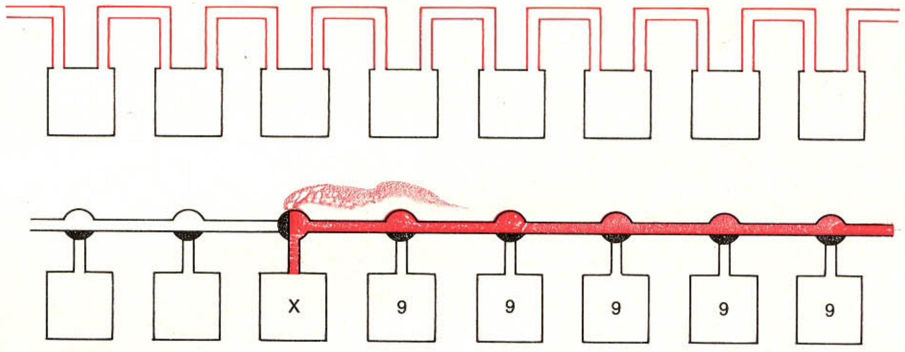
The challenge in designing or using a parallel processor of any of these three types lies in discovering ways in which simple patterns of communication within the processor can be made to match the communication tasks inherent in the problem being solved. As integrated-circuit technology progresses there will be individual circuits of increasing speed and complexity. No relief is in sight, however, for the costs and delays inherent in communicating information from one circuit to another. To provide better communication will require more connections to the integrated circuit, a bigger housing for it, more or larger communication-driving circuits and consequently more heat dissipation. To obtain maximum performance from large computing systems programmers will have to face up to the limitations on communication that are imposed by physical reality. High-performance communication cannot be provided from every element to every other element; the programmer will have to match his formulation of the problem to the available communication paths. Although this is a difficult task, success in accomplishing it will provide unprecedented processing power.

We believe that just as an important part of today's computer science concerns itself with sequences of instructions distributed in time, so an important aspect of computer science in the future will be the study of sets of communications distributed in space. If processors can communicate only with their nearest neighbors, what kinds of arrangements are possible? Obviously one can wire processors in a linear string. Such processors can operate in the pipeline fashion described above, with each one passing data along to the next, or by performing common operations under command from a central instruction device. Such near-neighbor connections are highly effective for tasks such as sorting, in which the local communication of data suffices. Alternatively, one can connect processors in an array, with each processor having more than two neighbors. Such arrays have a basic structure much like the structure of a crystal, and various forms of local communication are possible. In our laboratory at the California Institute of Technology we are considering the properties of the different communication paths that might be included in such structures.

Some years ago R. S. Gaines and C. Y. Lee, who were then working at Bell Laboratories, described three types of interconnection path. One kind of interconnection has connections that are common to all processors; it is effective for sending commands to the processors and for "broadcasting" to all processors certain values that may be important in the course of a computation. A second kind of communication path enables each processor to "talk" simultaneously



WIRES THAT INTERCONNECT MODULES of an array processor can be exploited in three distinct ways. One type of connection "broadcasts" information from a control center to all modules (*gray*). A second type (*black*) moves information from a selected module to a control destination, one module at a time (*heavy black line*). A third type passes data from each module to its nearest neighbor (*color*); in this case all the modules can "talk" at the same time.



TWO SUBTYPES of the third type of wiring are in common use. In one subtype (*top*) information passes into a module during one step, is processed and then passed on to the next module during the next step. The other subtype (*bottom*) is designed merely to "discover" something about a module (or a number of modules collectively) and to do so quickly: information is moved past a module unchanged during a single processing step, provided that the module is in a specified state. Such wiring might discover, for example, where the 9's are in a parallel adder.

to its neighbors. Such a path can handle the communications required in pipelining or, if each processor has its own storage function, open up a space anywhere in the store by moving information simultaneously away from the location of the desired gap. A third kind of communication path enables the processing elements to say something collectively about their results. Such a path can indicate whether no processor, one processor or more than one contains a given condition, which of the processors contains the smallest value or which are between the beginning and end of a particular string.

In our laboratory we have taken on the task of building and using some simple parallel processors involving one-, two- and three-dimensional interconnection patterns. We hope to learn more about the relation of communication paths to the performance of such processors. We have become convinced that the performance of parallel processors can depend critically on the design of communication paths that enable processing elements to make collective statements about their actions. Without such paths how does one find the smallest value stored in the array? How can one identify the set of processors that lie between two processors with designated properties? How does one obtain answers from a number of processors in sequence when more than one of them has something to report? Such paths are electrically complex. Either they involve each local processor as the driving element in a "global" communication task or they require intermediate circuitry specialized for collecting such information, and such intermediate circuitry inevitably introduces time delays and cost. The best structures for this kind of communication appear to be similar to those in the carry circuits of fast parallel adders, but the communication costs of such circuits have not yet been adequately analyzed.

A cornerstone of computer science today is the theoretical analysis of sequential algorithms. There is a large and growing body of theory for selecting efficient algorithms for sequential machines. This body of theory, as one might expect, focuses on algorithms that minimize the number of logical operations required to accomplish some task. For example, it has been shown that for putting numbers into sequence, "quicksort" algorithms are the best to use because they require only $n(\log_2 n)$ comparisons. This body of theory assumes that all data elements in storage are equally accessible and that the movement of data is free.

Data elements in storage are never really equally accessible, although they can arbitrarily be made equally inaccessible in a random-access device by making the access time for all elements as slow as the time required for the most inaccessible element. Because information in a random-access storage device must be moved over long distances the data rate in a random-access storage device of a given technology is inevitably lower than what can be achieved with a more orderly sequential-access mechanism. Moreover, transporting data from a memory cell to a comparison circuit is never really free; in most machines the transportation time far exceeds the comparison time. And so, for example, an analysis of algorithms seeking to show that a particular sorting algorithm is best is based on giving what may be a less than optimum machine the task of performing that algorithm; given a different structure, sorting might be done much faster.

Work on the theory of algorithms has not yet focused on the true relation of computing costs to communication costs. Given that one starts with a blank piece of silicon and is free to place wires, logic gates and so on anywhere one chooses, what choices should one make to accomplish a given computation task in the least time or on the smallest possible area of silicon? We have no basis at all for making sensible choices as to the computing structures we should build. We do have a large body of experience with a particular structure: sequential machines with random-access storage. It may be that such machines are effective because overall they best match a wide variety of computing tasks. There is mounting evidence, however, that a parallel structure can outperform the standard computer by many orders of magnitude on tasks that are suitable for parallel execution. As such unconventional structures have appeared, a wider range of tasks is being discovered for which they are suitable. Certainly one would expect to obtain better performance by paying attention to the real costs of systems, which is to say to communication, than by simply considering the cost—now an almost vanishing cost—of logical processing.

We believe adequate theories that account properly for the costs of communication will be an important guide for designing the machines that have been made possible by the integrated-circuit revolution. We believe such theories will have their basis in the study of regularity, so that computer science will come to include a body of theory akin to that of topology or crystallography. Although such a development is revolutionary in some ways, it is essentially a continuation of the search for regularity in all programming tasks. Computer scientists will simply add geometric regularity to the logical regularity they have already come to know and value.