The Author

W. DANIEL HILLIS is founding scientist at the Thinking Machines Corporation in Cambridge, Mass., and the architect of the Connection Machine system. He was graduated from the Massachusetts Institute of Technology in 1978 and obtained his master's degree there in 1982. In 1985 he won an ACM Distinguished Dissertation Award for his doctoral thesis, which he did at the M.I.T. Artificial Intelligence Laboratory. Hillis is the author of *The Connection Machine* and many articles on robotics, artificial intelligence and systems architecture.

Bibliography

THE CONNECTION MACHINE. W. Daniel Hillis. The MIT Press, 1985.

MASSIVELY PARALLEL COMPUTERS: THE CONNECTION MACHINE AND NON-VON. Richard P. Gabriel in *Science,* Vol. 231, No. 4741, pages 975–978; February 28, 1986.

SPECIAL ISSUE ON PARALLELISM. *Communications of the ACM,* Vol. 29, No. 12; December, 1986.

SCIENTIFIC AMERICAN

The Connection Machine

by W. Daniel Hillis

SCIENTIFIC AMERICAN
JUNE 1987
VOL. 256 PP. 108-115

The Connection Machine

Most computers have a single processing unit. In this new parallel computer 65,536 processors work on a problem at once. The resulting speed may transform several fields, including artificial intelligence

by W. Daniel Hillis

In the past three decades remarkable changes have taken place in digital computers. The amount of computational power that once required a room full of vacuum tubes can now be found in hand-held devices. Complex computations that would once have taken days to perform can now be done in seconds. Yet in certain fundamental respects the design of the digital computer remained unchanged between the days of the ENIAC (one of the first large-scale digital machines, built at the University of Pennsylvania in the late 1940's) and the current generation of supercomputers. Most modern computers—from supercomputers to microprocessors—are similar to the ENIAC in that the memory and the central processing unit are separate entities. For a computation to be performed, the appropriate data must be retrieved from memory and brought to the central processor; there it is operated on before being returned to memory.

Such a design is called sequential because the processing operations are performed one at a time. The sequential design was adopted mainly for utilitarian reasons. In the early days of digital computing the memory and the central processing unit were made of different materials. Since memory was cheaper than processing, it was desirable to maximize the efficiency of the processing unit at the expense of the memory's efficiency. And that is just what the sequential design does. Today, however, the memory and the central processor are fabricated from the same etched silicon wafers. In a typical computer more than 90 percent of the silicon is devoted to memory. While the central processor is kept wonderfully busy, this vast majority largely sits idle. At about $1 million per square meter, processed packaged silicon is an expensive resource to waste.

Clearly, the general solution to this problem is to find a way to unify processing capacity and memory. But how? One answer is to exploit many small processors, working simultaneously, each accompanied by a small memory of its own. In such a design, which is called parallel processing, memory capacity and processing capacity can both be utilized with high efficiency. This is the approach my colleagues and I have taken in building a parallel computer called the Connection Machine. The Connection Machine incorporates 65,536 simple processors. Each processor is much less powerful than a typical personal computer, but in tandem they can execute several billion instructions per second, a rate that makes the Connection Machine one of the fastest computers ever constructed.

Yet the most interesting thing about the Connection Machine is not its brute speed but its flexibility. Special-purpose devices have been built that exploit parallelism to perform specific tasks quite quickly. Like idiot savants, however, such machines are usually quite awkward outside their specialties. In contrast, the Connection Machine can operate at its peak processing rate in a wide range of applications. As this article will describe, the key to such flexibility is a communications network that enables the multitude of processors to exchange information in the pattern best suited to the problem at hand. The Connection Machine is not just a prototype. About a dozen Connection Machines are already in commercial use, and they have begun to change the way digital computing treats problems in physics, image processing, text retrieval and even artificial intelligence.

In order to understand the benefits of parallel processing, it is helpful to think about the difference between the way a conventional computer deals with an image and the way the same image is treated in the human brain. From the pair of two-dimensional images falling on the retinas a human being is able—without apparent effort—to reconstruct a three-dimensional model of the world and maintain that model as the two-dimensional images change rapidly. Computers can be programmed to carry out part of the task, but even quite fast computers take hours to do what the human brain can do in fractions of a second [see "Vision by Man and Machine," by Tomaso Poggio; SCIENTIFIC AMERICAN, April, 1984]. The brain maintains its advantage in spite of the fact that its components—neurons—are apparently millions of times slower than the computer's transistors.

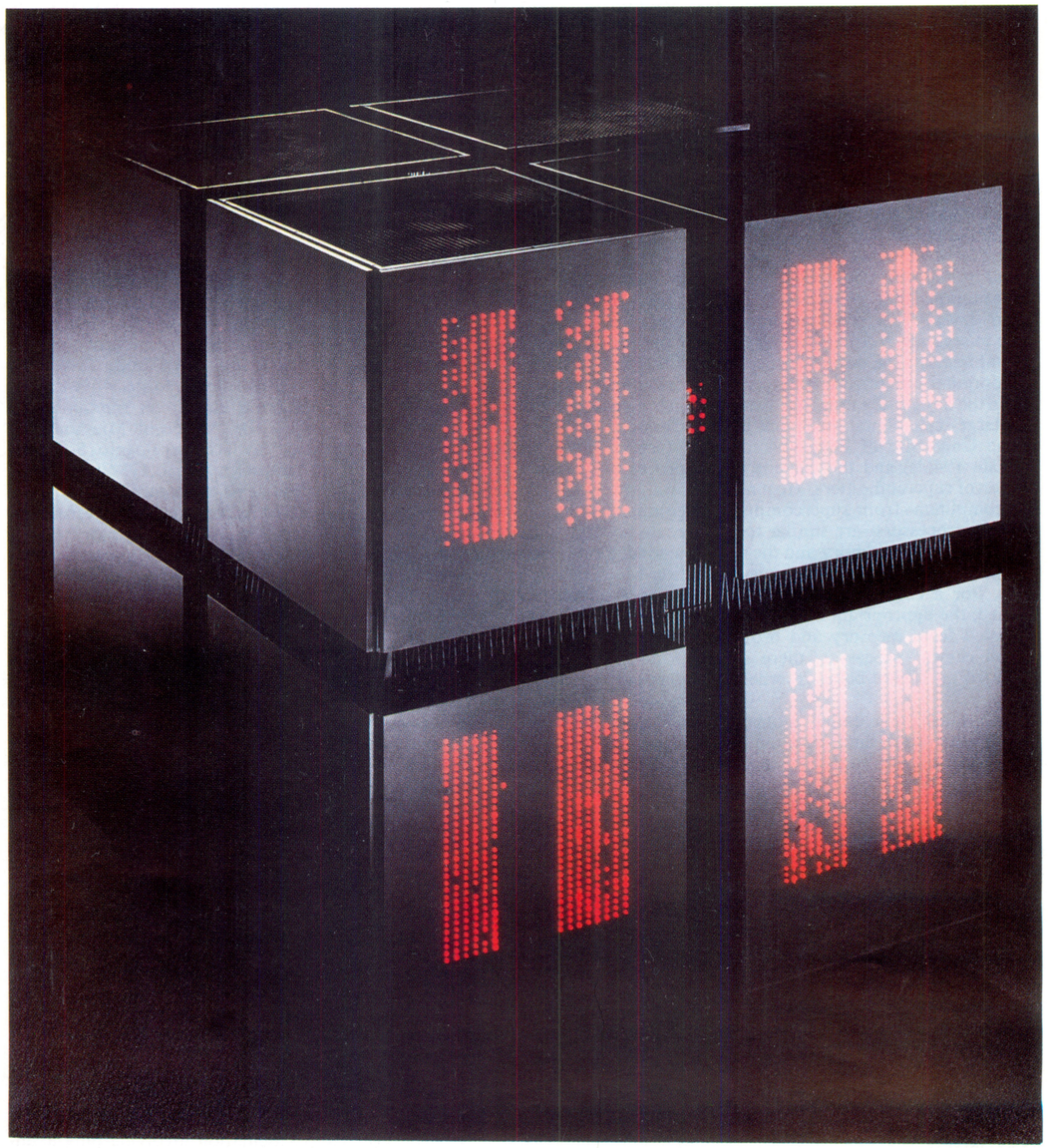
Why, then, is the brain so much faster than the computer? The visual circuitry of the brain is not fully understood, but it is clear that in some areas of the brain the principles of parallel processing are at work. In those parts of the brain the entire image is processed at once. The computer, however, examines the image one tiny spot at a time, as if it were looking through a minute keyhole. In the computer the image is represented as an array of numbers, each of which corresponds to the intensity of the light at a particular point. A typical low-resolution array might be a square with 256 points on a side. A conventional computer operates on only one of the square's 65,536 points at a time. Hence even a simple image-processing operation includes 65,536 steps.

The Connection Machine, on the other hand, assigns a single processor to each point of the image. Since every operation can be performed on all the points simultaneously, a calculation involving the entire image is as fast as a calculation involving only a single point. For example, to find all the points in the image that are brighter than a certain minimum a sequential machine must check the 65,536 tiny el-

ements in succession, comparing each one with the threshold value. In the Connection Machine that comparison is made simultaneously by the 65,536 processors—each one operating on a single element of the image.

The threshold comparison is particularly simple because it can be carried out independently by each processor. Most interesting computations, however, require that the processors exchange information as the operation proceeds. Consider the common image-processing operation called convolution. Convolution blurs an image by averaging each point with its nearest neighbors in the two-dimensional grid. (Convolution, which is analogous to operations carried out in the human visual system, is useful for removing insignificant details and bringing out significant objects.)

To complete a convolution each processor must read a value from the processors that store information about the points to the left, right, above and below the point in question.

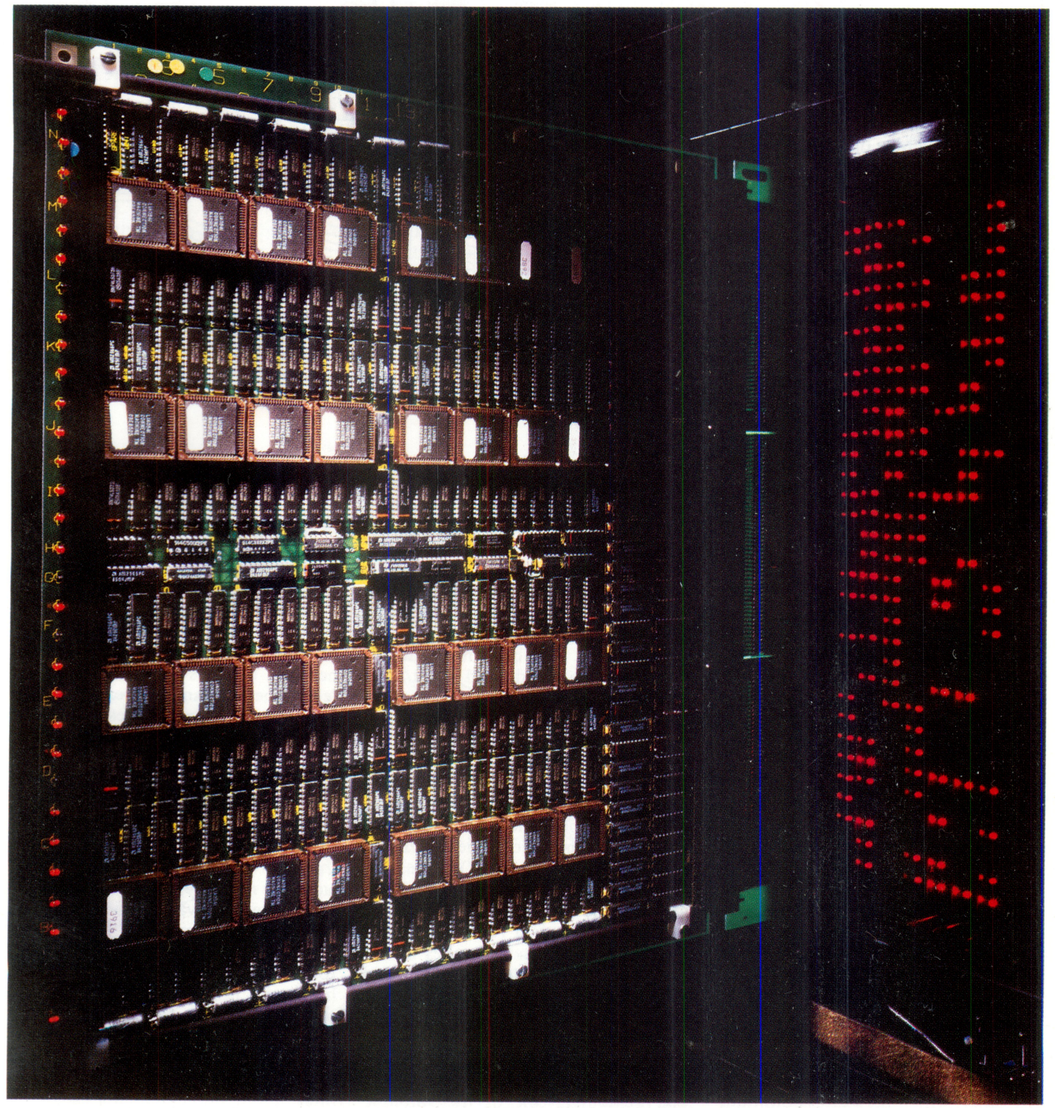


CONNECTION MACHINE is a cube 1.5 meters on a side made up of eight subcubes. Each subcube contains 16 boards arranged vertically. On each board are 32 custom chips. Every chip includes 16 processors, each with a small amount of associated memory. The red lights on the boards indicate the status of the chips; they are for troubleshooting. Operating in parallel, the 65,536 processors can execute several billion instructions per second, making the Connection Machine one of the fastest computers ever built.

In effect the processors must "talk" to each other. One way to accomplish such a pattern of communication is to wire the processors in a two-dimensional grid. Since each processor would be wired to its four nearest neighbors, the grid corresponds directly to the communication paths required for convolution. Indeed, some parallel computers specialized for image processing are wired in a two-dimensional grid. That pattern works well for convolution but not for other computations.

For example, in computing the average intensity of all points in the image a pattern of connections resembling an inverted tree is the most convenient. The average intensity of an image containing 65,536 points can be calculated by first computing the average of every pair of points, then the average of each pair of pairs, and so on. In 16 steps the average can be derived. In its last few steps the computation requires an exchange of information about points that are widely separated in the image; therefore the two-dimensional grid is not a convenient pattern of wiring.

The general principles to be derived from these examples are that each type of computation may require its own pattern of connections and that each processor may need to communicate with any other. Therefore in designing the Connection Machine we chose a communications network in which any processor can communicate with any other. As a result of such flexibility the programmer is free to



BOARD slides out of the Connection Machine much like a book from a shelf. The square objects are the chips, each with its 16 processors. The rectangular objects include memory units and devices for routing communications among the assembled processors.

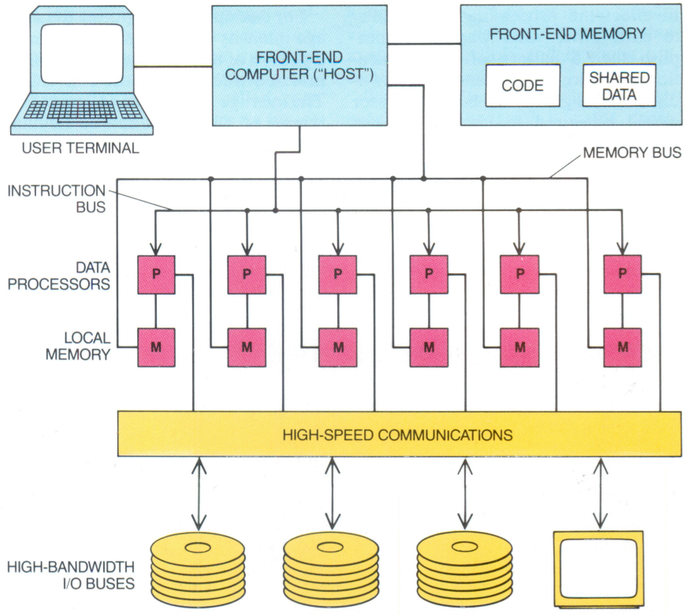
choose the algorithm that is most appropriate for solving the problem at hand without having to worry about the limits imposed by the pattern of wiring.

The basic replicated unit of the Connection Machine is an integrated circuit consisting of 16 small processors and a device for routing communications. Each of the processing units is associated with 4,096 bits of memory. (A typical personal computer has 256,000 bits or more.) The 16 processors are etched on a single chip and 32 of these chips are packaged on a single printed-circuit board. There are 128 such boards in the machine, arranged in a cube 1.5 meters on a side. For purposes of troubleshooting each chip is connected to a light on the edge of its board; the array of lights forms a pattern on the face of the cube as the machine operates.

The 16 processors on each chip are connected by a switch that makes it possible to create a direct connection between any pair of processing units. Implementing such direct connections between every pair of processors among the 65,536 in the system would require more than two billion wires, obviously an impractical figure. Instead, the routing device on each chip is connected to 12 other routers in the system. The routers are wired according to a pattern called a Boolean n-cube. The n-cube is a generalized version of an ordinary three-dimensional cube that has some excellent properties as a network for communications among processors.

The full mathematical detail of the n-cube is somewhat beyond the scope of this article, but its general principles are not difficult to grasp. One can imagine an ordinary three-dimensional cube as one member of a series of "cubes" corresponding to different spatial dimensions. For example, a line segment might be thought of as a "one-cube," or a cube in one dimension. Joining two one-cubes by their ends yields a two-cube, or a square. Joining two two-cubes by their corners yields a three-cube, which is what we ordinarily think of as a cube. Similarly, joining two three-cubes by their corners yields a four-cube [*see illustration on next page*]. The process may be repeated any number of times, and it can readily be shown that a 12-cube has 2^{12} (4,096) corners, or one for each chip in the Connection Machine.

Such a Boolean n-cube is a valuable arrangement for several reasons. In the first place, no processor in the 12-cube is more than 12 wires away from any other, which facilitates communication in the network. Second, the design of the n-cube accords well with the binary logic of the computer. In the digital computer all data are stored as strings of bits, each with a value of either 0 or 1. Now, each cube in the n-cube has two subcubes, which may be designated 0 and 1 respectively. As a result each point in the n-cube has a unique address specified by a string of 12 binary bits. The first bit specifies which of the 11-cubes within the 12-cube contains the desired point. The second bit specifies which of the 10-cubes is in question, and so on until a unique point has been determined.



SYSTEM DIAGRAM shows that the Connection Machine operates in association with a conventional computer, which is called the host. A user of the system interacts with the host by means of a conventional computer language modified for parallel programming. Rather than carrying out repetitive operations one at a time, however, the host delegates them to the Connection Machine, where the operations are done in parallel. The results can be obtained by various input-output devices, including high-resolution visual displays.

These binary addresses can be employed to route messages among the 4,096 chips in the Connection Machine. Each message in the system includes such an address. On receiving the message, the router examines the address one bit at a time, then forwards it to the next router along the way. That router in turn takes up the message, examines the address and forwards it. Thus in no more than 12 steps any message will find its way to the destination.

This communications network has several features that augment its speed and flexibility. One of the valuable properties of the n-cube arrangement is that there are many equally efficient routes of communication between any pair of processors. If one route is already occupied by a transmission in progress, the router is free to select an alternate route merely by processing the bits of the address in a different order [*see illustration on page 8*].

Another type of flexibility is also inherent in the communications system. In some instances the communications network behaves more or less like a telephone exchange: it establishes a circuit between two processors so that they can communicate continuously and exclusively. In complicated cases, however, the messages may be so long and the system so crowded that the routers must behave more like post offices, storing packets of information that are later forwarded. Such decisions are made by the routers based on what wires are available when a transmission must be made.

These properties make it possible for the Connection Machine to establish many different patterns of communication, depending on the problem at hand. An important feature of the system is that such details are invisible to the user, who needs to know no more about Boolean n-cubes than the average user of the telephone needs to know about digital switching. (Indeed, future versions of the machine may have other wiring patterns with no effect on the algorithms that are employed.) The programmer interacts with the Connection Machine through a conventional computer, known as the host, which employs a standard operating system and programming language. The processors of the Connection Machine are connected with the host much as a conventional memory unit would be.

Indeed, in one sense the Connection Machine is the memory of the host. That relationship makes possible a simple integration of parallel computing and existing software. Programs for the Connection Machine are surprisingly similar to conventional programs. The chief difference is that many operations normally carried out by repetitive loops are replaced by single operations corresponding to the simultaneous operation of many processors in the Connection Machine; the routing hardware automatically establishes the necessary communication paths.

It should be noted that nowhere in this system is exotic hardware to be found. In designing the Connection Machine we chose well-tested technologies in order to achieve simplicity and reliability. The individual processors are relatively slow by the standards of today's fastest computers. The custom chip is built by methods similar to those for making personal computers and pocket calculators. Yet the assembled power of the 65,536 processors makes the machine very fast. For many applications the machine can perform more than two billion operations per second; for the most favorable applications the figure is more than 10 billion, or about 1,000 times as fast as a typical mainframe computer.

Putting the machine's speed in a slightly different context, one might consider floating-point operations, which provide a standard for computing power in number-intensive scientific applications. A floating-point operation is the multiplication or addition of two numbers expressed in scientific notation (such as 1.5×10^2). A typical supercomputer can carry out a few hundred million floating-point operations per second; the Connection Machine can average about 2,500 million on a typical problem.

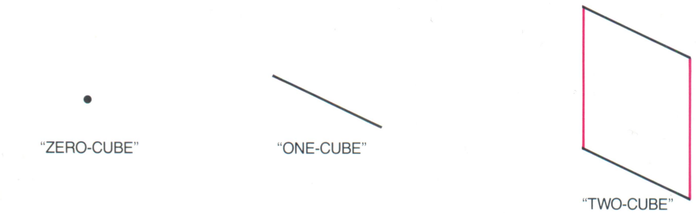
To what uses has this considerable number-crunching capacity been put? As suggested above, some of the initial applications have involved manipulating and processing images. Others have exploited the parallelism inherent in certain physical processes. The engineering problem of calculating the flow of air over an airplane wing or a helicopter rotor provides an example of how the Connection Machine can mirror the parallelism of nature.

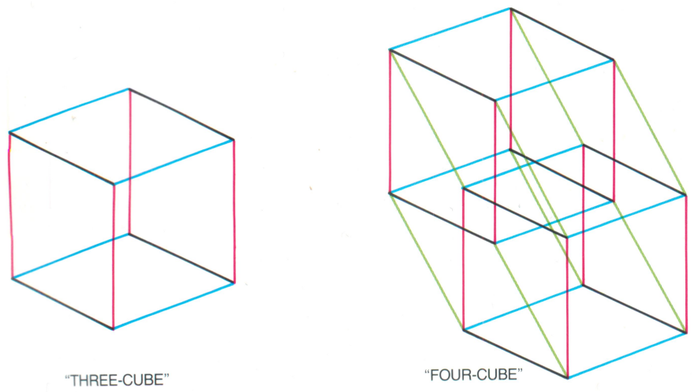
In nature the overall flow pattern emerges from the myriad interactions among air molecules, which bump into one another and into the surface of the wing as they rush along. The engineer (who is interested in the overall flow rather than the specific molecular interactions) uses a simplified, large-scale model consisting of a set of partial differential equations. Yet the equations themselves are set up in parallel: they treat changes in pressure in small volumes of air and sum their interactions to yield the overall flow. Because the equations are parallel, their solution is fast and efficient on the Connection Machine.

With a parallel computer, however, one can also move beyond the equations and come closer to the underlying physical reality. The large-scale behavior of a fluid is for the most part independent of the detailed physical properties of its individual particles. Moreover, the qualitative behavior of the fluid is not changed when the number of particles is greatly reduced. Therefore it is possible to accurately re-create large-scale flows by examining collisions among a few tens of millions of simple, generalized particles.

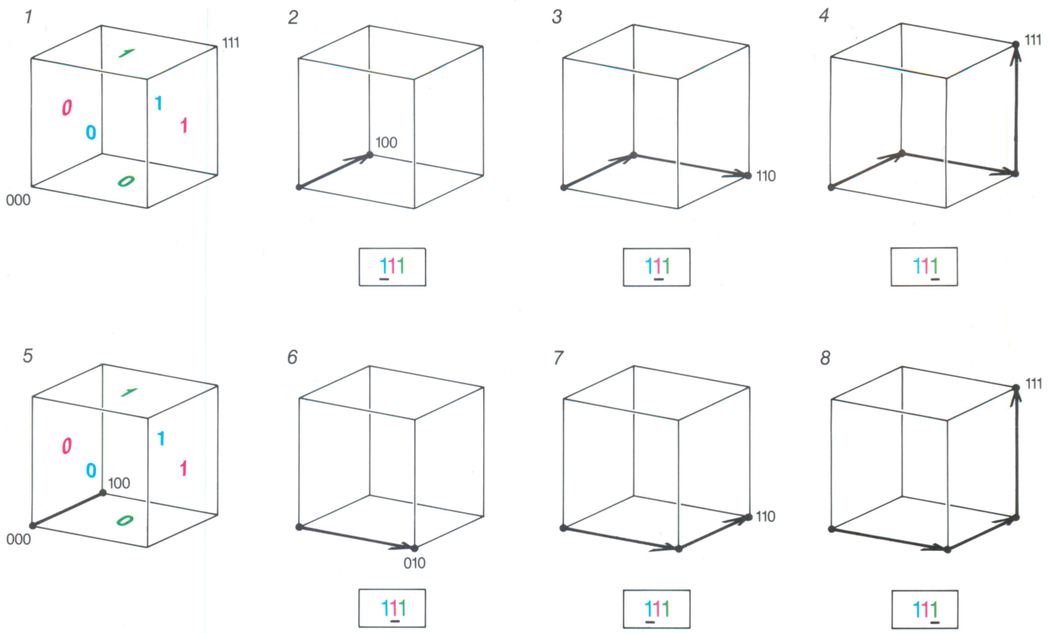
Stephen Wolfram of the Center for Complex Systems Research at the University of Illinois at Urbana-Champaign and my colleague James Salem took advantage of this technique to model fluid flows over complex surfaces. Their simulation entailed only a few tens of millions of particles, and the particles were allowed to move in only six directions at integral velocities. Nonetheless, the system is capable of accurately mimicking the flow of a fluid.

The simplest and most logical way to perform the fluid-flow computation would be to assign each particle its own processor. Yet a typical simulation includes about eight million "particles," and the Connection Machine, vast as it is, includes only some 65,000 processors. The solution to this programming difficulty and analogous ones is to program each processor to act as if it were a string of different





BOOLEAN N-CUBE provides the topology for the network that links the Connection Machine's processors. A Boolean n-cube is a generalized version of an ordinary cube. Such cubes can be constructed in many dimensions, each building on the next-lower dimension. A point can be considered a cube in zero dimensions, or a "zero-cube." Linking two points yields a "one-cube," or a line. Linking a pair of "two-cubes" (squares) yields the familiar three-dimensional cube. Two three-cubes can be joined at their vertexes (corners) to form a "four-cube." Repeating the process would yield a "12-cube" with 4,096 vertexes. The 4,096 chips of the Connection Machine are wired in the form of a 12-cube.



ALTERNATE ROUTES for communication between chips are provided by an *n*-cube. The illustration shows alternate routes in a three-cube, but the same principle applies to the 12-cube of the Connection Machine. Each vertex of the *n*-cube (where the chips lie) can be assigned a unique address as follows. A three-cube includes three pairs of planes. Each plane can be designated 0 or 1, and a vertex is then assigned a three-digit address according to which member of each pair of planes it is found in (*1*). Messages are forwarded by routing devices at each vertex, which read the address and process it one digit at a time. Here a message is sent from 000 to 111. The router reads the first digit and forwards the message to point 100 (*2*). There the second digit is read (*3*). At 110 the third digit is read and the message is forwarded to its destination (*4*). When it comes time to send the message, however, the wire between 000 and 100 may be busy (*5*). In that case the router simply reads the second digit of the address first, choosing an alternate route (*6*). Then the first and third digits are read (*7, 8*) and the message is delivered to the correct address.

processing units, each unit handling one particle at a time. The details of the arrangement are again invisible to the programmer, who simply specifies how many "virtual processors" are required. The hardware and software take care of the rest. Of course, if each processor must simulate 250 units in turn, the computation takes 250 times as long as it would with one actual processor per particle.

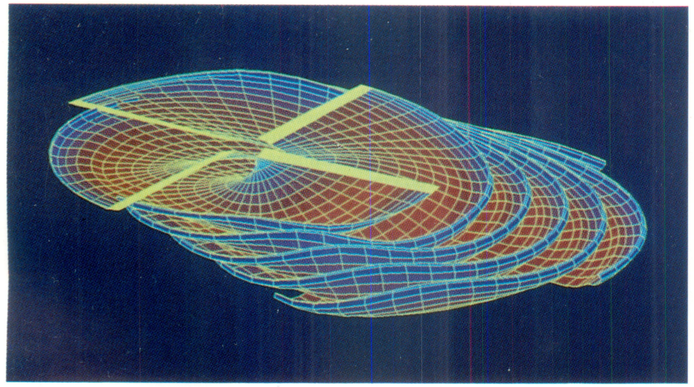
Many interesting applications of the Connection Machine do not involve numbers. My colleagues Brewster Kahle, Craig Stanfill and David Waltz exploited the computer's parallelism to retrieve documents from large collections of texts. The underlying principle of their system is that each processor can be programmed to compare one document in a large data base with a "search sample," a paragraph chosen for its relevance. Once the comparison has been made, the processors exchange information and rank the documents according to how well they match the search sample.

Comparing two pieces of prose to see how well they match is not a simple task. Merely counting the number of words that appear in both samples is useless, because the count is contaminated by words such as "the" and "as," which carry little content. Therefore the document-retrieval system exploits a dictionary and some rules of grammar to extract from each sample the phrases that bear its content. Each processor is loaded with a different article compressed in this way, and the search sample is broadcast to all the processors in the network.

The process of comparison is relatively simple, and since 65,536 documents are checked at once, the entire data base can be examined almost instantaneously. Ranking the articles according to how well they match the search sample is a more difficult operation, because it requires a complex pattern of communication among the processors. Yet in parallel it can be performed in about 50 milliseconds. A few of the highest-ranked documents are then offered to the user of the system, who can choose a new search sample from among them. (Conversely, articles that are clearly irrelevant can be chosen as negative search samples.) Because all the comparisons are done at once, the entire collection of texts can be winnowed repeatedly in a short period, which ensures that all relevant articles will be found.

The document-retrieval program is able to function without anything that approaches an understanding of the contents of the articles. Actually understanding those contents would require considerable background knowledge about the world, which has not been incorporated into the retrieval system. One exciting area of research involving the Connection Machine is the writing of programs that include such background knowledge and are able to mimic certain aspects of human reason.

Like the processing of two-dimensional images to form a three-dimensional world model, "commonsense" reasoning is carried out without apparent effort in the human brain. For example, any child can deduce (with the appropriate affective response) that his mother's favorite vase will fall if it is dropped. The child is able to infer that the vase is more like a plate or a rock, which fall, than it is like a bird



HELICOPTER ROTOR produces a complex airflow that can readily be simulated in parallel on the Connection Machine. Each processor models the circulation within a certain layer of air (*small subdivisions of the image*). The circulation of the air in each section influences the air in each of the other sections. These interactions are computed in parallel. Details such as the wavy distortion at the bottom are important for predicting forces exerted on the helicopter blades. The simulation was developed by T. Alan Egolf of the United Technologies Research Center and the author's colleague J. P. Massar.

or a balloon, which do not. The inference can be made correctly in spite of apparently contradictory information, such as the fact that the vase may be spherical, as a balloon is, or that the child's mother may also own a bird.

Clearly, for human beings the ease and accuracy of such inferences increases with the accumulation of knowledge about the world. The opposite is true for conventional computers. As the number of different concepts increases, the number of possible relations among them increases even more quickly. Because a sequential computer can examine these relations only one at a time, its pace slows dramatically as the quantity of background data grows. Indeed, the slowness of conventional computers in commonsense reasoning was one of the stimuli responsible for the design of the Connection Machine.

In the late 1970's, as a graduate student at the Massachusetts Institute of Technology, I became interested in how commonsense reasoning might be simulated by computers. It seemed to me that one way out of the morass sequential computers found themselves in when asked to make simple, everyday deductions was to build a machine that could examine possible connections among concepts more than one at a time. (In that conclusion I was inspired by the work of Marvin L. Minsky of M.I.T. and Scott E. Fahlman of Carnegie-Mellon University.) That was in 1978. By 1985 the idea had moved to the stage of an actual prototype with the aid of a grant from the U.S. Defense Advanced Research Projects Agency, which offered to buy the first machine. By then I had left M.I.T. and helped found the company—Thinking Machines Corporation—that builds and markets the Connection Machine.

Now that the Connection Machine is a physical reality, investigators of artificial intelligence are making use of it to solve commonsense reasoning problems. The germ of this approach is to assign one fundamental concept to each processor. The connections among processors can then be exploited to represent multiple relations among simple concepts. In the simple example given above, one processor may represent the concept "Vase," another "Mother" and a third "Likes." The connections among these three processors would embody the knowledge that "Mother likes her vase." Other connections might represent the vase's shape, composition and history. When it comes time to decide what the outcome would be if the vase were dropped, the relevant connections can be searched in parallel.

Since there are now about a dozen Connection Machines in operation, there will undoubtedly soon be many new programs for the machine. It is likely that many of them will be in the four general areas touched on above: image processing, simulation of physical processes, searching of data bases and artificial intelligence. One of the greatest challenges in learning to use the Connection Machine lies in beginning to think in parallel terms. Programmers have considerable accumulated experience in programming for sequential machines, and such programming has by now become almost second nature. Learning to write programs for parallel machines requires thinking in ways that are quite different from those demanded by sequential computers.

That challenge will be greater for the Connection Machine than it will be for some other types of parallel machines. It should not be assumed that the Connection Machine is the only representative of its genre. Indeed, many different parallel designs are now in various stages of realization. To generalize greatly, these designs fall into two broad classes: "coarse-grained" and "fine-grained." Coarse-grained machines link relatively few processors, each with a relatively large amount of computational power; fine-grained machines link a great many weak processors.

These two classes of parallel computers form a spectrum. At one end is the conventional sequential computer, which has the minimum number of processors: one. At the other end of the spectrum are designs such as that of the Connection Machine, which include a very large number of small processors. Although some highly qualified investigators and companies are pursuing the coarse-grained approach, I think it is the fine-grained design that will ultimately prove the most fruitful. Yet it is also the one that is the most foreign to our preconceptions about computer programming.

In writing a program for a coarse-grained machine, one can adhere to concepts much like those used for programming sequential computers; the problems arise in attempting to coordinate the programs. In writing a program for the Connection Machine, however, one is faced with an entirely different realm of problems and possibilities. Exploiting the full potential of the machine will require a new way of thinking about computation, which we as programmers have just begun to learn. That learning process will undoubtedly be both rewarding and challenging.

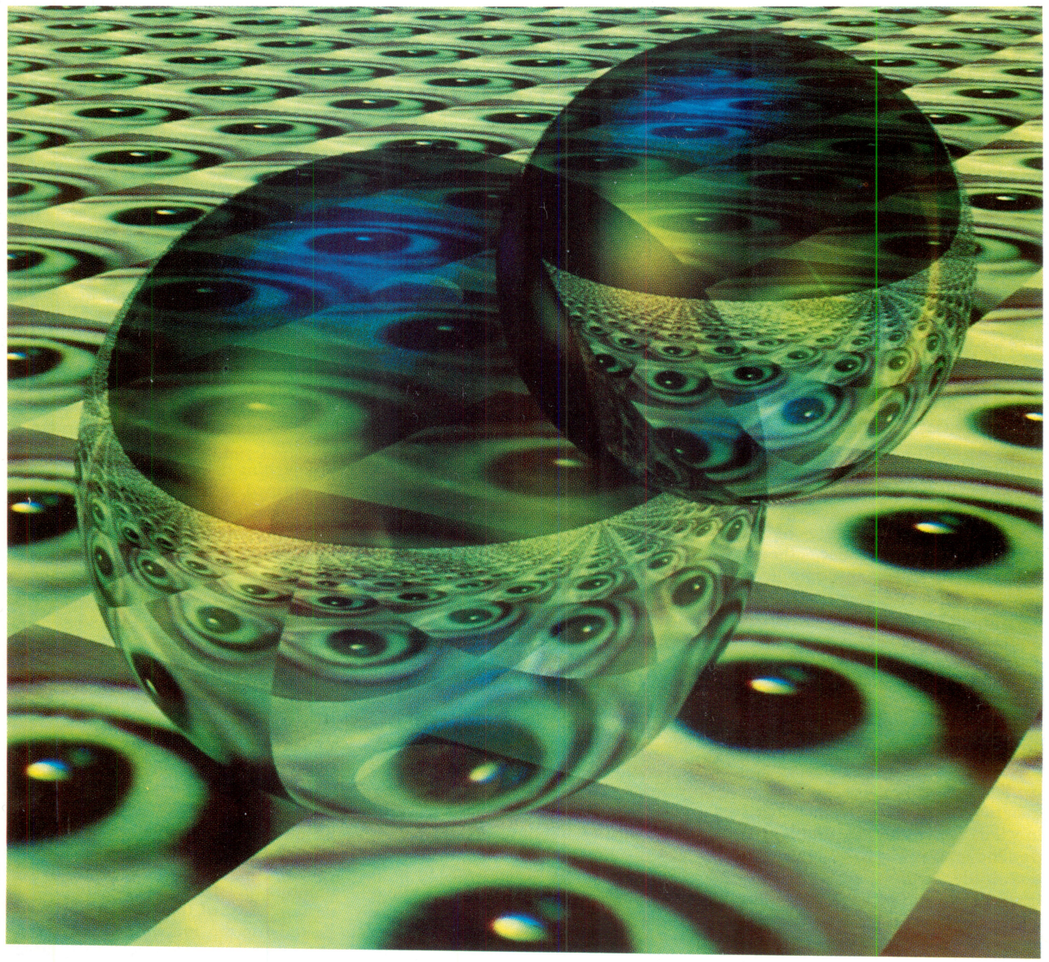
Some of its rewards may come from the fact that the Connection Machine can be expanded to encompass considerably more computational power without any fundamental changes in design. Most of the applications envisioned for the machine could profitably exploit a computer much larger than current versions of the Connection Machine. For this reason the computer has been designed to allow a significant increase in the number of

processors. The Connection Machine can be expanded simply by adding processors, memory and communication devices to an existing machine.

As an extreme example of scaling up, imagine a parallel computer with one billion processors. Such a machine might well incorporate some features of the Connection Machine, although there would undoubtedly be many new problems to solve. If built with current technology, a billion-processor machine would be as large as a building and cost 20 times as much as today's largest computers. It could, however, execute some 100 million million (10^{14}) instructions per second, which is several orders of magnitude greater than the computational power of all existing supercomputers combined.

There are technical problems inherent in building such a computational engine, but they are soluble. The real problems are those of the imagination: conceiving how such power would be used. Some engineering problems, including extrapolations of examples mentioned above, might benefit from such capacity, but they are in a sense trivial. The applications worthy of a billion-processor machine are those that entail a radical change in the way we think about computation.

A parallel computer with a billion processors might provide the basis for a computational utility analogous to existing gas and electric utilities. Just as a coal-fired plant generates electricity that is transmitted to individual appliances, a huge parallel computer could provide computational power to a city's worth of robots and workstations. The design of the parallel machine would enable many users to draw on portions of the total computing capacity for small problems, whereas the total capacity could be applied to large ones. Such a vision is somewhat utopian (at least for the moment), but it is by no means impracticable, which suggests the depth of the changes that parallel computing may ultimately bring.



COMPUTER GRAPHICS is one of the fields in which parallel computers may be most fruitful. The illustration was made by the technique called ray tracing, in which each Connection Machine processor is assigned to a different pixel (picture element). The processors trace rays of light bouncing among imaginary objects, here glass balls and images of eyes. The paths of the rays determine the final color of each pixel. Karl Sims of the Massachusetts Institute of Technology Media Laboratory generated the image.