VLSI Implementation of a Neural Network Model

Hans P. Graf, Lawrence D. Jackel, and Wayne E. Hubbard
AT&T Bell Laboratories

Models of neural networks are receiving widespread attention as potential new architectures for computing systems. The models we consider here consist of highly interconnected networks of simple computing elements. A computation is performed collectively by the whole network with the activity distributed over all the computing elements. This collective operation results in a high degree of parallel computation and gives the network the potential to solve complex problems quickly. Neural network models have demonstrated functions such as associative memory, adaptive learning from examples, and combinatorial optimization.

To date, the research concerning neural network models has focused mainly on theoretical studies and computer simulations. However, the real promise for applications of the models lies in specialized hardware, in particular specialized microelectronic circuits. Simulations of large networks on serial computers are painfully slow, and only with customized hardware can we hope to realize neural network models with speeds fast enough for applications.

Digital accelerators to simulate neural networks are now commercially available, but they are still orders of magnitude slower than what we can achieve by directly fabricating a network with hardware. Several researchers have built models with discrete electronic components. These implementations help us study properties such as the dynamics of these circuits, but they are too bulky for real applications.

To obtain the full benefit of neural network algorithms, we need special-purpose hardware. An experimental CMOS VLSI circuit was tested as an associative memory and as a pattern classifier.

The most promising approach for implementing electronic neural nets is to fabricate special-purpose very-large-scale-integration chips. With today's integration density, a large number of simple processors can be packed on a single chip together with the necessary interconnections to make a collective computing network. Several groups have initiated experiments with VLSI implementations and demonstrated a few functioning circuits.[1-3]

Attempts are under way to build neural network models with optics.[4] The high interconnectivity of the networks makes optics attractive because interconnections can be made optically in three dimensions. The circuit on a microchip is bound to the two dimensions of the chip's surface. However, optical computing technology is still in its infancy and realizations suitable for applications probably lie further in the future.

We describe a complementary metal oxide semiconductor (CMOS) very large scale integrated (VLSI) circuit implementing a connectionist neural network model that consists of an array of 54 simple processors fully interconnected with a programmable connection matrix. This experimental design tests the behavior of a large network of processors integrated on a chip. We can operate the circuit in several different configurations by programming the interconnections between the processors. We made tests with the circuit working as an associative memory and as a pattern classifier. The results were so encouraging that we interfaced the chip to

0018-9162/88/0300-0041$01.00 © 1988 IEEE

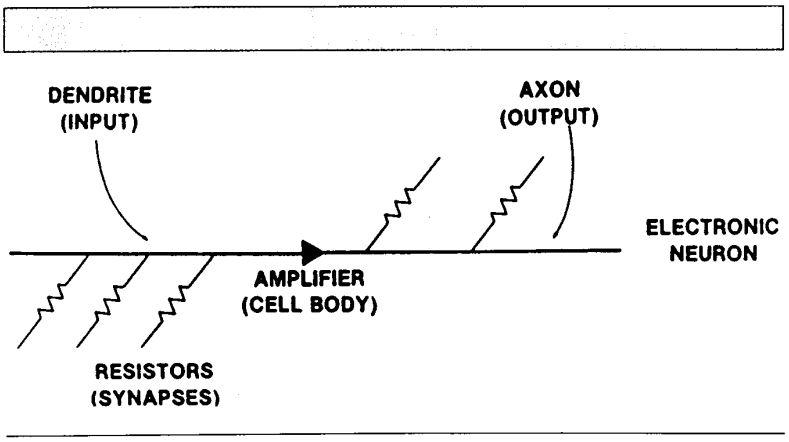


Figure 1. An electronic "neuron."

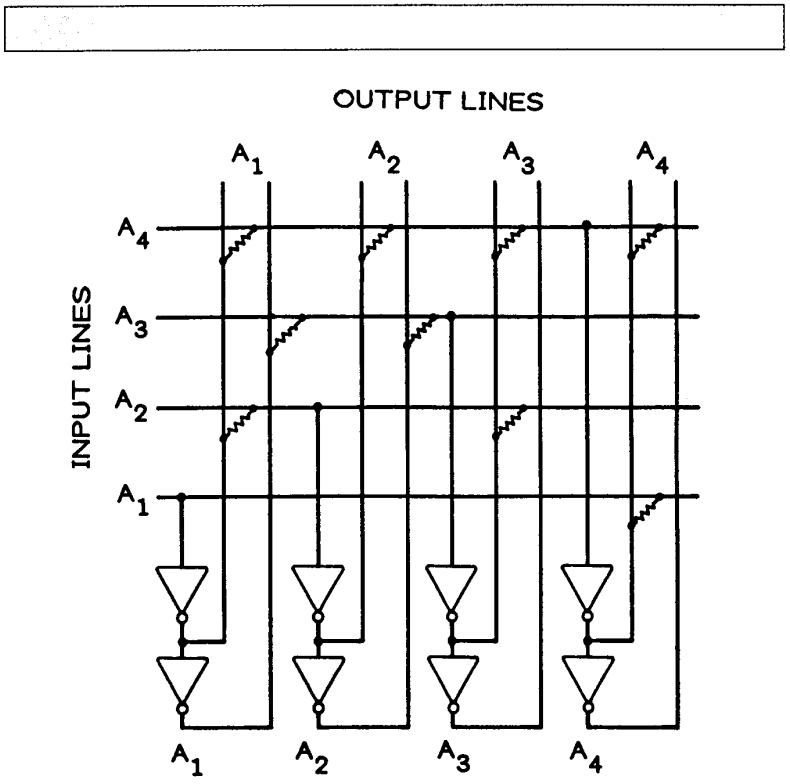


Figure 2. Schematic of the implemented circuit. At each crosspoint of an input and an output line, a resistive connection can be set. All the connections are programmable. The configuration of resistors shown here is just one example. Two inverters are connected in series and work as one amplifier unit. To have the inverted and the noninverted output going into the connection matrix makes it easy to control excitatory and inhibitory connections.

a minicomputer and now use it as a coprocessor in pattern recognition experiments. This mode of operation allows us to test the chip's behavior in a real application and study how pattern recognition algorithms can be mapped into such a network.

Connectionist model

Biological neural networks inspired the models we are implementing in electronics, but we are not attempting to imitate real neurons. Clearly, the models used grossly simplify the biological networks. But even these relatively simple networks have complex dynamics and show interesting collective computing properties. It is this collective computation that we try to exploit by building such networks.

The type of circuit described here is often referred to as a *connectionist model.* In a connectionist model, an individual "neuron" does very little computation, typically just a thresholding of its input signal. The kind of computation performed by the whole network depends on the interconnections among the neurons. Figure 1 shows a possible electronic circuit for such a simple neuron. An amplifier models the cell body, wires replace the input structure (dendrite) and the output structure (axon), and resistors model the synaptic connections between neurons. The amplifier's output voltage replaces the firing rate of a real neuron.

Each of the resistors connects the input line of the amplifier to the output line of another amplifier. Therefore, the state of an amplifier is determined by the states of all the other amplifiers. When the amplifier measures the current flowing into the input line, we can express this as

$$Vout_j = f\left(\sum_i I_i\right)$$
$$= f\left(\sum_i (Vout_i - Vin_j) T_{ij}\right) \qquad (1)$$

where *Vin, Vout* are the input and output voltages of an amplifier; I_i is the current flowing through one resistor; T_{ij} is the conductance of the resistor connecting amplifier i with amplifier j; and $f()$ is the transfer function of the amplifier.

Equation 1 shows how the states of the amplifiers, represented by $Vout_j$, determine how much current flows into the input line and therefore determine the state of this amplifier. The output voltage of the amplifier is then given by its transfer characteristics. Its output connects to the input of many other amplifiers and

influences their states. A network of highly interconnected amplifiers forms in this way. It is very difficult to describe the dynamics of such a system for an arbitrary distribution of the resistive connections, but many special cases can be controlled well and exploited for collective computation. Such networks have been demonstrated as functional associative memory and as optimizers.[5,6]

The implemented circuit well suits work as an associative memory or as a pattern classifier. To have the network perform these functions does not require precise control of the gain of the amplifiers or the ability to fine tune the resistive connections. This makes it possible to design interconnections and amplifiers using only a small area, so many of these components can be integrated on a single chip.

The function described by Equation 1 could be implemented in digital hardware. But this would require a complex multiplier-adder circuit at the input of each neuron, resulting in a large circuit even for a modest number of neurons. The implementation described here uses a mixture of analog and digital CMOS VLSI technology.

Using analog computation, we can achieve a multiplication with a single resistor, and the summing of currents is accomplished "for free" on the input wire of the amplifier. However, designing large integrated analog circuits is a difficult task. There is a strong tendency in signal processing today to avoid analog computation and do everything digitally. Yet the high interconnectivity and relatively low precision needed for the signals in a neural net are well tailored for an analog approach, and the gain in computing power of the network should outweigh the extra design effort.

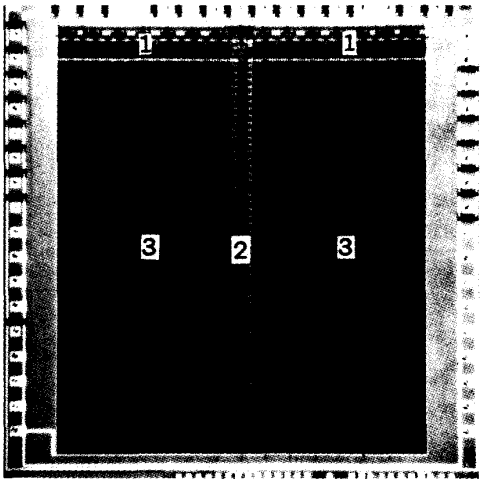


Figure 3. Photomicrograph of the chip. The modules are: 1 = amplifiers, control logic, bit decoder, input bus, and output bus; 2 = decoder to address rows in the interconnection matrix; 3 = interconnection matrix.

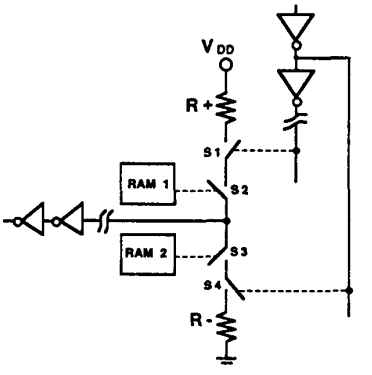


Figure 4. One of the coupling elements connecting the output of an amplifier to the input of another amplifier. S_1 and S_4 are closed when the output of the amplifier controlling them is high (noninverted output high). If a "1" is stored in RAM 1, S_2 is closed and the connection is excitatory. If a "1" is stored in RAM 2, S_3 is closed and the connection is inhibitory. If both RAM cells store a "0," no current flows through this interconnection regardless of the state of the controlling amplifier.

The circuit

Figure 2 shows a schematic of the implemented circuit. It consists of an array of 54 amplifiers with their inputs and outputs interconnected through a matrix of resistive coupling elements. All of the coupling elements are programmable—a resistive connection can be turned on or off.

Figure 3 shows a photomicrograph of the circuit. Fabricated in CMOS technology with 2.5-micrometer design rules, it contains roughly 75,000 transistors in an area measuring 6.7×6.7 millimeters. By far the largest active area of the chip, almost 90 percent, is used for the programmable coupling network.

In this design the circuit shown in Figure 4 replaces the resistors of Figure 2. The output lines of the amplifiers do not feed current into the input lines; instead they control the switches S_1 and S_4. This method reduces the amplifier load to just the capacitance of the output line. For each connection between two amplifiers, two memory cells control switches S_2 and S_3. The content of these memory cells determines the type of connection. One of three connections can be selected: an excitatory (S_2 enabled), an inhibitory (S_3

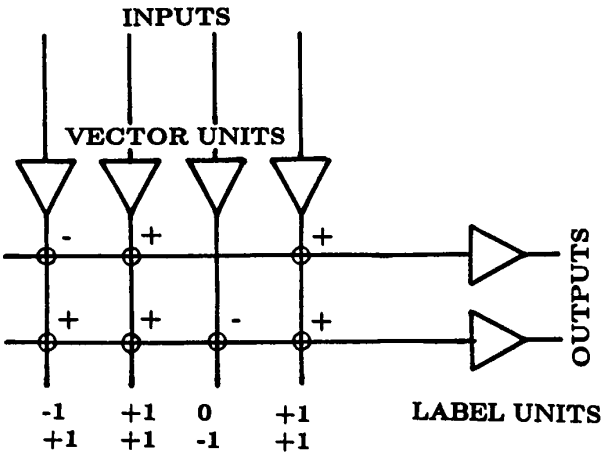


Figure 5. Programming of the interconnections for the pattern classifier circuit. The "+" and the "−" mark excitatory and inhibitory connections, respectively. Below the schematic the stored vectors are shown.

enabled), or an open (both disabled) state. The network contains a total of 2916 such coupling elements. Notice that when an amplifier output is low, the connection turns off completely and no current flows in either direction.

The voltage of an input line of an amplifier is determined by the sum of the currents flowing into the node. This voltage adjusts to a value where the total current is zero. Since the input impedance of the amplifiers is very high, this leads to

$$\sum_{i=0}^{i=N} I_{ij} = \sum_{i=0}^{i=N} \frac{\Delta V_{ij}}{R_{ij}} = 0 \qquad (2)$$

where I_{ij} is the current flowing through a resistor of the coupling element controlled by amplifier i; ΔV_{ij} is the voltage difference across a resistor, ($Vin_j - V_{DD}$, $Vin_j - V_{SS}$); and R_{ij} is the resistor, (R_+, R_-).

Thus, the voltage Vin_j is an analog measure of the sum of the contributions of all the amplifiers connected to input line j. The amplifiers used have a high gain and work essentially as threshold elements, with the switching threshold about halfway between the ground and the supply voltage. Analog computation is used only within the connection matrix; input and output data and all the control signals are digital.

Data input and output are transferred through a register where one memory cell is connected to each amplifier. The input data are first loaded into this register. From there they can be loaded into the memory cells of the connection matrix, or used to initialize the circuit. Initialization of the circuit is done by charging the network with the voltage levels corresponding to the components of the input vector. The amplifiers are turned off during this initialization cycle. For the computation, the amplifiers are turned on and the network evolves to a stable state without any external control or synchronization between the amplifiers. After the circuit has reached a stable state, the output voltage of each amplifier is stored in the register, which can then be read out.

Programming the chip

The circuit's architecture facilitates mapping several different configurations into the network simply by programming the connections between the amplifiers. Figure 5 shows the arrangement of the interconnections for a configuration used to do pattern classification. The amplifiers are divided into two groups: the label units and the vector units. A number of vectors are stored in the circuit, each one along the input line of one label unit. The components of the stored vectors can have the values +1, −1, or 0. An excitatory connection is set for a +1 and an inhibitory connection for a −1.

The input vector is presented on the inputs of the vector units. Its components can have the values +1 or 0. Wherever a +1 in the input vector is set, current can be injected or drawn from the input line of a label unit depending on the type of the connection. As described by Equation 2, the condition for a stable state is that the total current flowing into an input line equals 0. If the input voltage is above the threshold of the amplifier, this label unit turns on; otherwise, it remains off. Whether a label turns on or not is described by Equation 3:

$$\sum_{i=0}^{i=N} \frac{v_i \mu_i}{R_i} = \begin{cases} >0: & Vout = high \\ <0: & Vout = low \end{cases} \qquad (3)$$

where v_i denotes components of the input vector, (+1, 0); μ_i denotes components of the stored vectors, (−1, 0, +1); and R_i is the resistance of the connections, (R_-, R_+).

The input vector is compared in parallel with all the stored vectors and an inner product between the input vector and the stored vectors is evaluated. All the stored vectors closely resembling the input vector turn on their label units.* A +1 in the input vector at the position of a +1 in the stored vector gives a positive contribution to the sum, while a −1 in the stored vector in this position gives a negative contribution. R_+ is about six times larger than R_-. Therefore, a mismatch (a +1 in the input in the position of −1 in the stored vector) counts six times as much as a match. This ratio of R_+ to R_- has no great significance; it simply reflects the ratio of the resistance of the p-channel and the n-channel transistors of the CMOS circuit. For the applications described later, it is convenient to have the inhibition stronger than the excitation, but we could obtain the same effect by using multiple connections.

By using a few amplifiers as bias units, we can shift the threshold of a label unit. We can program the connections between the bias units and the label units to set different thresholds for each stored vector. The right-hand side of Equation 3 is then replaced by the bias value.

*The classification properties of this circuit correspond to those of a single-layer perceptron; it can discriminate between linearly separable patterns. Since we are dealing with binary vectors, the configuration space consists of the corners of an n-dimensional hypercube. Decision regions of any shape can be built in this space as the sum of linearly separable regions.

The number of stored vectors and their lengths are limited by the number of amplifiers in the circuit. Each component of the input vector uses one amplifier, and each label unit needs one. Therefore, the number of bits in a vector plus the number of stored vectors must be smaller than 54. Usually, a few amplifiers are used for biases, reducing this number to approximately 50.

This arrangement uses only the connections between the outputs of the vector units and the inputs of the label units, while the largest part of all the connections in the matrix remains idle. We can achieve a more efficient use of the circuit by using the register cells as the vector units. The arrangement described uses these cells only as intermediate storage for the input vector, but they can also charge the matrix to its initial condition. All the amplifiers then work as label units. Up to 54 vectors, each 54 bits long, can be stored in the network and are compared in parallel to the input vector. To read out the result, the register cells must switch quickly from writing the input vector to reading the result without feeding the result back into the connection matrix. This requires a precisely timed pulse controlling the length of time the amplifiers are turned on. This mode of operation is used for feature extraction from images (see the section "Examples of applications").

Adding inhibitory connections between the label units and connecting the outputs of the label units to the inputs of the vector units can yield an associative memory. This arrangement is shown schematically in Figure 6. In this circuit, vectors with components + 1 and 0 are stored. In addition to the connections along the input lines of the label units described above, there are inhibitory connections between all the label units. Each label unit inhibits all the other label units, but not itself. For the connections between the label outputs and the vector inputs, the same vector is placed along the output line of a label unit stored along its input line. For a + 1 in the vector, an excitatory connection is set, and for a 0, an inhibitory connection is set.

The circuit is initialized with all input lines of the label units discharged to ground. The input vector is given on the input lines of the vector units. Wherever a + 1 in the input vector and in a stored vector occupy the same position, current is injected into the input line of a label unit. The speed at which an input line of a label unit changes state depends on the inner

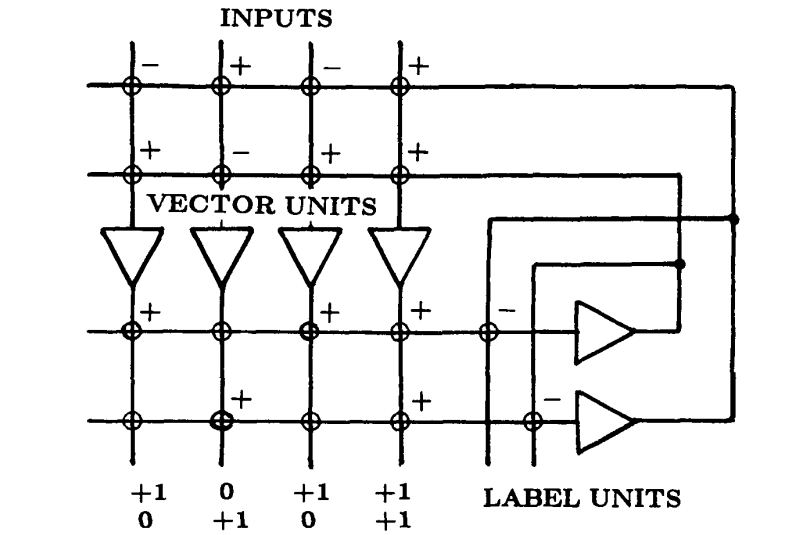


Figure 6. The interconnections set for an associative memory circuit.

product of the input vector and a stored vector. That is,

$$\frac{dV_{in}}{dt} = \frac{1}{CR_+}(V_{DD} - V_{in}) \sum_{i=1}^{N} v_i \mu_i \quad (4)$$

where V_{in} is the voltage of the input line; C is the capacitance of the input node of a label; R_+ is the resistance of the excitatory connection; v_i denotes the components of the input vector, (0, + 1); and μ_i denotes the components of the stored vector, (0, + 1).

The stored vector that has the largest inner product with the input vector will turn on its label unit first. As one label unit turns on, it inhibits all the other label units. If this inhibition is strong enough, no other label unit will turn on.

As mentioned above, an inhibition is six times stronger than an excitation. This limits the allowed mutual overlap (number of common 1's) of two stored vectors to five bits; biases are used to circumvent this limitation. The label unit that comes on first generates the vector stored along its output wire at the inputs of the vector units. In this way, in a stable state one label unit is on and the vector connected to that label appears at the outputs of the vector units.

This circuit performs a minimum distance classification. The input vector is compared with all the stored vectors in parallel, and the circuit determines which of them most closely matches the input vector (common 1's are counted).

Extensive tests with 10 stable states, each 40 bits long, programmed into the associative memory circuit showed that the circuit converges to a stable state in 50 to 600 nanoseconds.[2] Associative recall performed reliably, and we observed only the stable states programmed into the circuit; we saw no spurious stable states. The convergence time depends on how closely the input vector resembles the stored states. Additional tests run with up to 20 stable states, each 30 bits long, programmed into this circuit obtained similar results.

In this circuit the stored data are represented locally in the interconnection matrix. This means that a stored bit can be localized at one or two of the interconnections. In contrast, other circuits based on neural network models use a distributed representation of the data (e.g., the outer product of a stored vector determines the distribution of the connections).

The great interest in associative memory circuits implemented with neural network models has resulted in the description of several programming methods.[5,7] Experiments conducted with this type of associative memory soon made clear the far

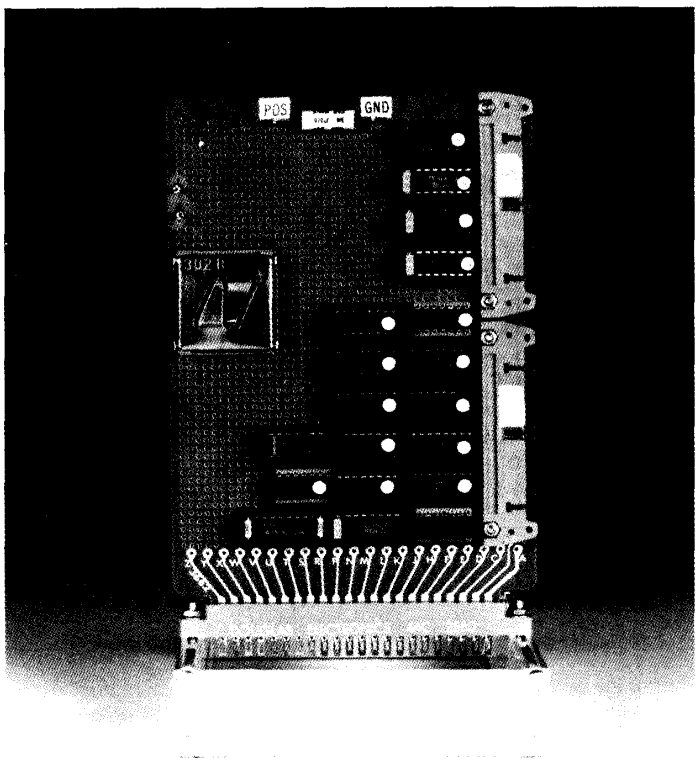


Figure 7. The interface connecting the chip to a minicomputer. The network chip is on the left side of the board. The other chips control the data exchange with the computer.

superior results given by local representation of data. With distributed representation of data, the different stored vectors influence each other because they share the same interconnections. This leads to unwanted spurious stable states and to erroneous recalls of vectors that do not correspond to the nearest neighbor.

These problems do not exist in the circuit we describe here. We observed reliable recall of the nearest neighbor and no unwanted stable states. Moreover, the storage density is considerably higher with this circuit. With a distributed representation of the data, it would hardly be possible to get 20 arbitrary stable states in this circuit. But the arrangement described here still does not use the interconnections of the chip efficiently. None of the connections among the vector units are used. In the case of 10 stored vectors, this means that less than one third of all the connections are used.

With the present chip we do not see a way to further increase storage efficiency. However, a design optimized for this new associative memory circuit can achieve 100 percent storage density.[2] This means that one RAM cell stores one bit. In contrast, associative memory circuits with distributed representation of data typically require five or more RAM cells to store one bit. High storage efficiency is crucial for an electronic implementation. Other researchers have recently proposed circuits similar to the associative memory we have described.[8,9]

We can configure the circuit to move through a sequence of vectors by placing a vector along the output line of a label that differs from the vector along its input line. Mixing connections of two vectors can stabilize the circuit at a vector and make it sensitive to the next vector in the sequence only. When the input lies within a set range of this vector, the circuit will move to the next stable state. Otherwise, it just stays in its present state. Other versions of programming techniques allow for omissions and branches in a parsed sequence of vectors.[10]

Examples of applications

In order to evaluate the behavior of the circuit in an application, we used it in a character recognition experiment. An interface connected the chip to a minicomputer to handle the transfers of the large amounts of data characteristic of image processing requests.

Figure 7 shows the board with the network chip and some additional off-the-shelf integrated circuits to control data flow. Data transfers can be made directly from the minicomputer's memory to the chip at a rate between one and two megabytes per second. This rate is limited by the interface and the minicomputer and not by the chip. One complete processing cycle, which includes loading the input vector, accomplishing a computation with the circuit, and reading back the result into the computer, requires approximately 25 clock cycles or 25 to 50 microseconds. Most of the time is required for reading the data in and out. The processing in the circuit requires only one clock cycle.

The whole process of recognizing a character proceeds as follows: A handwritten character is read with a camera, digitized, and then normalized in size to fill a 128×128-pixel frame. The image is then coarse-blocked into a 16×16-pixel binary image. After this, the character is *skeletonized*—the width of the lines is reduced to one pixel—and the skeletonized picture is searched for a number of geometrical features. The positions of these features are compared to a training set and a best match with one of these training characters is determined. Of this whole process, the line thinning and the feature extraction have been mapped onto the chip; the minicomputer accomplishes the remaining operations.

Figure 8 shows an example of a result of the line thinning operation. The chip is

used in the configuration of Figure 5 for this task. Stored in the circuit are 20 different vectors, each representing a 5×5-pixel area. The input vector to the chip is a 5×5-pixel region of the character image, which we will refer to as a *card*. Based on which label units are on after a run, a decision is made whether or not to delete the center pixel of this card from the image. For the next run the card is shifted by one row or column of pixels on the image; this new card is used as the input vector. In this way the whole picture is scanned. For each pixel position, a processing cycle is accomplished with the appropriate card as the input vector. For each pixel in the image, a decision has to be made whether that pixel makes the line fat or whether its presence is crucial to keeping intact the connectivity of the character.

Figure 9 shows an example of a vector stored in the circuit to make one such decision. If the label connected to this vector turns on, then the center pixel is part of a diagonal line or a corner and that line is at least two pixels wide. In this situation, the pixel can be deleted without destroying the connectivity of the character. The 20 vectors stored on the chip analyze the neighborhood of each pixel for all the configurations that allow its deletion. All the computation needed to decide whether a pixel can be deleted is done in one clock cycle. A total line-thinning operation requires about three scans across the whole character, depending on the width of the lines because only boundary pixels are deleted.

Line thinning is an important step in machine vision, not only in character recognition, but also in tasks like fingerprint analysis and inspection of manufactured parts. Many algorithms have been developed to handle this problem.[11] The algorithm implemented with the network circuit does not differ fundamentally from other pixel-based algorithms. The stored vectors facilitate making the same type of tests as those formulated in the other algorithms with a set of Boolean functions. Most other algorithms base their decision on a 3×3 area around the pixel under test, since the test of larger areas becomes very time consuming. However, with the network it is not a problem to analyze the larger 5×5 pixel area, since it still takes only one clock cycle. Using a larger area makes the algorithm more robust, and it supports integrating some smoothing to enhance the thinning operation.

To accomplish the extraction of geometrical features, 40 vectors are stored in the

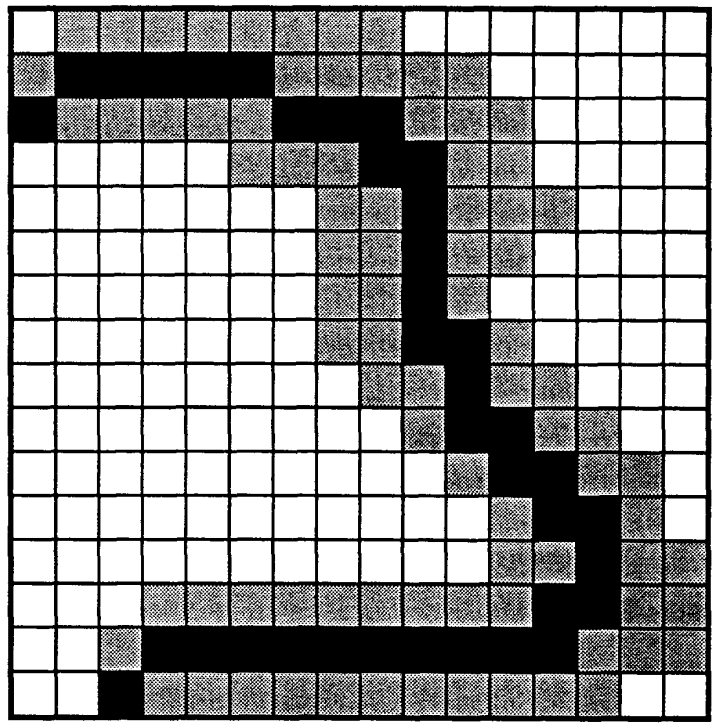
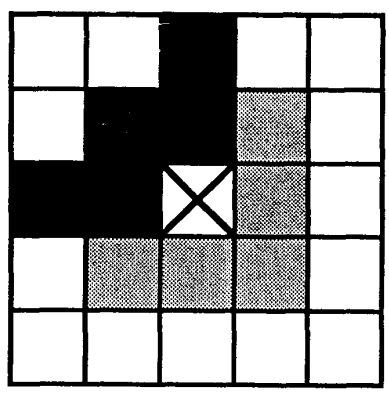


Figure 8. The result of a line-thinning operation on a handwritten "3." The gray area represents the original character, and the black area is the portion that remains after three thinning scans.



00+000++-0+++-00---000000

Figure 9. One of the kernels stored in the chip for the line-thinning operation. If the label connected to this vector turns on, the center pixel is erased in the image. The black pixels are coded as excitatory connections (+) and the gray pixels as inhibitory connections (−). The bias is set to −4. The label turns on whenever five black pixels in the image correspond to the black pixels in this kernel and no black pixel in the image is at a position of a gray pixel in the kernel. Then, the center pixel is a part of the boundary of a thick diagonal line or a corner and can be deleted.

circuit. Each one looks for a feature like a straight line, an endpoint of a line, a corner, crossing lines, or an arc. Whenever one of these features is present in the image, the label of this feature-vector turns on.

The image is scanned sequentially, as described for the line-thinning operation. For every pixel position, a 7×7 pixel neighborhood is searched in parallel for the 40 different features. Such a scan results in 40 feature maps that indicate where particular features occur. To compress this large amount of data, the resolution of the feature maps is then reduced from 16×16 to 3×3 positions. These feature maps are compared with reference characters and the best match is chosen. Currently, the minicomputer accomplishes this last matching operation, but a way to map this final operation onto the network circuit is under development.

The successful recognition rate for hand-written digits is approximately 90 percent. We assume that we can improve this rate considerably by using finer resolution for the character images as well as the feature maps. We now use only local geometrical features of a maximum size of 7×7 pixels. We can expect better results when large features such as long strokes or large circles are generated first from these local features. However, the main purpose of this experiment was to test the circuit's performance and to gain experience programming different algorithms into the network. We have not yet pushed the analysis to find the highest recognition rate.

Character recognition based on feature extraction is more robust against distortions than simple template-matching algorithms that compare the pixel positions of the whole character to reference data. Feature extraction is one of the most versatile functions of low-level machine vision and can be applied to many problems.

Discussion

We have tested the circuit with several programmed algorithms and conclude that a network like this one is reliable and robust enough for applications. We designed this circuit to study the behavior of a large analog network and did not specialize it for any particular application. Since input and output of data limit the processing, we must optimize the I/O structure in new designs for particular applications.

The analog portion of the circuit represents a very powerful processor. With 50 stored vectors, each 50 bits long, the chip completes a processing cycle in less than one microsecond. This means that the circuit evaluates 50 inner products between two 50-bit vectors per microsecond. On a standard microprocessor this computation would require more than one hundred instructions.

A new chip being fabricated was designed specifically for image processing. The input vector is shifted in a shift register along the inputs of the amplifiers. In this way a new input vector is ready for a computing cycle at each clock cycle when a card of 8×12 pixels is scanned over an image. It is possible to store 46 vectors, each 96 bits long, and simulations indicate that a complete program cycle can be accomplished in approximately 100 nanoseconds.

This circuit will evaluate several hundred million inner products between two 96-bit vectors per second. For example, the circuit can do the line-thinning operation described above at a rate of a few hundred microseconds per character compared to the few hundred milliseconds per character a standard computer takes.[10] (With character sizes of 32×32 pixels, each scan requires 1024 cycles if the whole area is scanned; about three scans are required.)

This new design is fabricated with the same conservative 2.5 micrometer CMOS process. Switching to smaller design rules will allow packing considerably more circuitry on a single chip. Also, since this network is implemented with a standard digital fabrication process, it can be combined easily with other memory and processor modules on the same chip to enhance its versatility.

A system concept with a layered structure is also under development. In this scheme a layer of network processors is followed by a memory module, with several of these units stacked in series. In an application like character recognition, the different tasks such as line thinning or feature extraction are then done in different processor layers. Each layer of processors inputs the data from the memory module below it and outputs its results in the memory module above it.

The data flows mainly in one direction, from the raw data at the input of the lowest layer to the output layer that does the pattern identification. However, communication is also provided in the opposite direction, so that results from a higher layer can determine the operation performed in a lower level. This feature facilitates scanning certain areas in the image with a different resolution, or scanning the image for different features when the results in the higher levels are ambiguous. We need additional research on the mapping of different algorithms into the network, and how to format the data optimally to feed from one network into the next with minimal intermediate reformatting.

The network described provides a flexible tool because it can evaluate Boolean expressions and arithmetic equations. Methods of statistical as well as structural pattern recognition can be mapped into the chip. Ideas from the artificial intelligence community on bit-mapped classifiers[12] suggest that expert systems could be made from this network. With all these elements at hand, this network looks promising for building powerful recognition systems. □

Hans Peter Graf is a member of the technical staff at AT&T Bell Laboratories. He is doing research on collective computing systems and has designed several CMOS chips implementing connectionist neural network models. He has also been involved in the development of microfabrication processes of resistive connections with submicron dimensions for neural network circuits.

Graf received the diploma in physics in 1976 and the PhD in 1981 from the Swiss Federal Institute of Technology in Zurich, Switzerland.

Lawrence D. Jackel has been at AT&T Bell Laboratories since 1975 and is now head of the Device Structures Research Department. His current research interests include the physics, fabrication, and application of devices with nanometer scale features and electronic neural networks.

Jackel received the BA degree in physics from Brandeis University in 1969 and the PhD degree from Cornell University in 1976. His thesis research was in the physics and applications of superconducting weak links.

Wayne E. Hubbard is a member of the technical staff at AT&T Bell Laboratories, where he has been involved in such diverse projects as optical recording, vapor phase epitaxial growth systems, and currently, neural networks.

Hubbard began his career in electronics and computer science in 1969 at RCA, then served six years with the US Navy Tactical Data Systems. From 1976 until joining Bell Labs in 1982, he worked for Digital Equipment Corp. as a district support engineer.

Readers may write to the authors at AT&T Bell Laboratories, Room 4G320, Holmdel, NJ 07733.

Acknowledgments

We acknowledge many helpful discussions with Henry Baird, John Denker, Richard Howard, and Sara Solla. We designed the chips with the MULGA design tools, and acknowledge the support of Brian Ackland and Robert Clark in using these tools.

References

1. M.A. Sivilotti, M.R. Emerling, and C.A. Mead, "VLSI Architectures for Implementation of Neural Networks," *Proc. Conf. Neural Networks for Computing*, J.S. Denker, ed., 1986, American Institute of Physics Conf. Proc. 151, pp. 408-413.
2. H.P. Graf and P.G.N. deVegvar, "A CMOS Implementation of a Neural Network Model," *Advanced Research in VLSI, Proc. Stanford Conf. 1987*, P. Losleben, ed., MIT Press, Cambridge, Mass., 1987, pp. 351-367.
3. J.P. Sage, K. Thompson, and R.S. Withers, "An Artificial Neural Network Integrated Circuit Based on MNOS/CCD Principles," *Proc. Conf. Neural Networks for Computing*, 1986, J.S. Denker, ed., American Institute of Physics Conf. Proc. 151, pp. 381-385.
4. Y.S. Abu-Mostafa and D. Psaltis, "Optical Neural Computers," *Scientific American*, March 1987, pp. 88-95.
5. J.J Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two State Neurons," *Proc. Academy of Science USA*, Vol. 81, 1984, pp. 3088-3092.
6. D.W. Tank and J.J. Hopfield, "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and Linear Programming Circuit," *IEEE Trans. Circuits and Systems*, Vol. CAS-33, 1986, pp.533-541.
7. J.S. Denker, "Neural Network Models of Learning and Adaptation," *Physica D*, Vol. 22D, 1986, pp. 216-232.
8. J. Moody, "Perspectives on Associative Memories," *Proc. IEEE First Int'l Conf. Neural Networks*, M. Caudill and C. Butler, eds., IEEE, 1987.
9. R.P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE Acoustics, Speech, and Signal Processing*, Vol. 4, 1987, pp. 4-22.
10. P.G.N. deVegvar and H.P. Graf, "Studies of Associative Memory and Sequence Analyzer Circuits on a Programmable Neural Network Chip," to be published. Contact Graf for preprint copies.
11. N.J. Naccache and R. Shingal, "SPTA: A Proposed Algorithm for Thinning Binary Patterns," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC14, 1984, pp. 409-418.
12. P.W. Frey, "A Bit-Mapped Classifier," *Byte*, Nov. 1986, pp. 161-172.