An Analog Neural Network Processor with Programmable Topology

Bernhard E. Boser, *Member, IEEE*, Eduard Säckinger, *Member, IEEE*, Jane Bromley, Yann Le Cun, *Member, IEEE*, and Lawrence D. Jackel, *Senior Member, IEEE*

Abstract —**The architecture, implementation, and applications of a special-purpose neural network processor are described. The chip performs over 2000 multiplications and additions simultaneously. Its data path is particularly suitable for the convolutional topologies that are typical in classification networks, but can also be configured for fully connected or feedback topologies. Resources can be multiplexed to permit implementation of networks with several hundreds of thousands of connections on a single chip. Computations are performed with 6-b accuracy for the weights and 3 b for the neuron states. Analog processing is used internally for reduced power dissipation and higher density, but all input/output is digital to simplify system integration. The practicality of the chip is demonstrated with an implementation of a neural network for optical character recognition. This network contains over 130 000 connections and is evaluated in 1 ms.**

I. Introduction

NEURAL networks have demonstrated their capabilities in numerous applications, including pattern classification, speech recognition, and control [1]–[8]. However, the computational requirements, data rates, and size of neural network classifiers severely limit the throughput that can be obtained with networks implemented on sequential general-purpose computers. Better performance is achieved with special-purpose VLSI processors that employ parallel processing to increase the throughput.

Speed and data rates are not the only challenges faced by specialized hardware designs for neural networks Because of the rapid progress of neural network algorithms, processors must be flexible enough to accommodate a wide variety of neural network topologies. Moreover, the size of networks under investigation is increasing as progressively more difficult tasks are solved with neural networks. Networks with several tens or hundreds of thousands of connections are typical for high-accuracy pattern classifiers, and this number is expected to grow further. To be economical, such networks must be implemented on a small number of chips. Moreover, the high-performance parallel-computing unit must be matched with an equally powerful interface to avoid bottlenecks.

Meeting all these requirements calls for a trade-off. On one side, the hardware should be as general as possible to support a wide range of applications. At the same time, efficiency dictates a design that carefully matches neural network characteristics. In Section II, hardware requirements of neural networks are examined, with special emphasis on classification applications. Issues considered include the regularity of neural network architectures, the feasibility of sharing common hardware for multiple functions, and the effect of limited accuracy computations.

The architecture of the neural network chip is described in Section III, and implementation aspects of the functional blocks are presented. A mixed analog and digital design has been chosen to exploit the low computational accuracy typically required by neural network algorithms, and at the same time address system integration issues, which call for a digital interface.

Experimental results and performance figures are summarized in Section IV. The flexibility and speed are documented with a variety of sample neural network architectures and the corresponding update rate that is achieved with the chip.

The practicality of the design is demonstrated in Section V with results from an implementation of a neural network for handwritten digit recognition with over 133 000 connections. The network fits on a single chip and is evaluated at a rate in excess of 1000 characters per second, which constitutes a speedup of two orders of magnitude over a DSP based implementation. Despite the low resolution of the chip, the error rates of the neural network processor and DSP implementation are very similar at 5.3% and 4.9%, respectively. For comparison, the measured human performance on the same database is 2.5% errors.

II. Neural Network Hardware Requirements

Artificial neural networks that solve difficult problems in areas such as speech recognition and synthesis, or pattern classification, consist of thousands of neurons with tens or hundreds of inputs each. Every neuron computes a weighted sum of its inputs and applies a nonlinear function to its result. Architectural parameters, such as the number of inputs per neuron, and each

Manuscript received May 5, 1991, revised August 16, 1991 This work was supported in part by the U S. Army SDC
The authors are with AT&T Bell Laboratories, Holmdel, NJ 07733.
IEEE Log Number 9103700

neuron's connectivity, vary considerably within a network, and from application to application. A special-purpose neural network processor must be flexible and powerful enough to accommodate a wide range of applications. At the same time, the requirements must be carefully balanced and the special nature of the task exploited to bring an efficient implementation within reach of today's technology. In this section, the characteristics and requirements of neural networks are examined and possible VLSI implementations investigated.

Two phases of operation are distinguished in many neural network applications. During the learning phase, the topology and weights of the network are determined from a labeled set of examples using a rule such as back propagation [1], or a network growing algorithm [9]. In the subsequent retrieval or classification phase, the network parameters are fixed. Patterns are now recognized by the network based on information stored in the architecture and weights during training. Since the computational and infrastructure requirements (e.g., training database) during the learning phase are considerably more complex than those for classification, efficiency considerations call for separate hardware for learning and retrieval. Network parameters determined during learning are downloaded into processors that are specialized for the classification task. This paper focuses on the latter problems. This approach contrasts with implementations of neural network processors with on-chip learning [10], [11]. Those circuits are not suitable for the pattern recognition problems investigated here, both because of limitations of the training algorithms implemented on these chips and because of the limited size of the network that can be trained.

The basic operation performed by a neuron during classification is a weighted sum, followed by a nonlinear squashing function f, typically a hyperbolic tangent or approximation thereof:

$$y = f\left(\sum_i w_i x_i + b\right).$$

The inputs x_i of the neuron are usually referred to as connections, and the w_i as weights. Each input is either tied to the output y of another neuron or to an external input. Optionally, a bias b is added to the weighted sum.

The total number of connections in neural networks for applications such as handwritten character recognition is several ten or hundred thousand [12]. Networks that solve more general problems, for example recognition of entire words instead of isolated characters, require even larger numbers of connections. The speed requirements of typical applications call for a few tens to several thousands of classifications per second. For each classification, one multiplication and one addition must be evaluated for every connection. This translates into up to a few billion multiply-add operations per second. Such computational power can be achieved only with a parallel implementation, where several connections are evaluated concurrently.

The most general network topology permits connections between any two neurons. Such a high degree of (possible) connectivity, combined with the need for parallel processing, results in enormous hardware requirements, and therefore calls for a compromise. Usually, the neurons in a network are arranged in layers. Each layer receives inputs only from neurons in the previous layer. Layers may be fully connected, that is, each neuron may be connected to every neuron in the preceding layer. Often, however, local connectivity is used in order to express knowledge about the problem (e.g., geometric relations, such as neighborhood of pixels in an image) in the network architecture and thus improve the recognition performance [6]. For example, the fact that some pixels in an image are adjacent to each other can be built into the network architecture by constraining neurons to receive inputs only from neighboring pixels. In a fully connected topology, such information must be derived from the training set during the learning phase, usually with only partial success.

A neural network processor could be designed to implement only networks with fully connected topology. Local connectivity would then be realized by simply setting the weights of unused connections to zero. Since, in typical neural networks, the ratio of such unused connections to actual connections is easily 100, such an implementation is unacceptably inefficient. Support for local connectivity adds circuit complexity, but overall a tremendous saving in chip area is realized because of the reduced number of connections, more than compensating for the added complexity.

Another challenge for a compact hardware implementation of a classifier is the amount of memory that is needed for storing several tens or hundreds of thousands of weights. Fortunately, the weights of many neurons in important connection topologies, including time delay or feature extraction neural networks [4], [6], [7], are identical. In those architectures, the connection topology corresponds to a one- or higher dimensional convolution, followed by the nonlinear squashing function. Such a structure can be realized with a single neuron that is time multiplexed, with a corresponding saving of storage and computing devices.

Further optimization of the hardware complexity is possible by matching the computational accuracy of the processor to the requirements of typical neural networks. Both experience and theory [13] indicate that neural network classifiers can be designed to be insensitive to low-resolution arithmetic. Experiments with character recognizers show that the recognition performance remains virtually unchanged when the inputs and outputs of the neurons are quantized to 3 b, and the weights to approximately 5 b. Higher resolution is required in the last layer for the rejection of ambiguous or unclassifiable patterns. Since in typical neural networks the output layer contains only a small fraction of the total number of

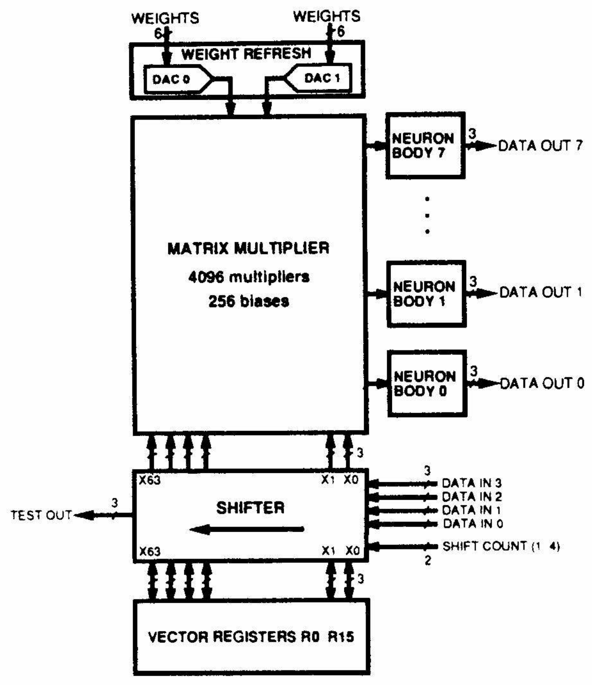


Fig 1 Block diagram of the neural network processor

connections, system complexity can be reduced by evaluating those connections on a different processor with higher accuracy

III. Architecture and Implementation

A. Overview

Fig 1 shows the block diagram of the neural network processor. Data enter the chip through a shifter [14]. This interface is well suited to the convolutional-type network topologies discussed in the section above. The matrix multiplier computes the dot products of input and weight vectors. The weights are programmable and stored locally in the multiplier. The nonlinear squashing function is evaluated by the neuron bodies.

The chip uses analog computation internally to capitalize on the low-accuracy requirements of typical neural network algorithms. All inputs and outputs are digital, however, to simplify system integration. The overhead for D/A and A/D conversion is negligible since both functions are combined with the neural computation. The D/A converters serve simultaneously as multipliers, and the A/D converters in the neuron bodies evaluate the nonlinear squashing function. Neuron inputs and outputs are quantized to 3 b, and weights are represented as 6-b quantities; both are represented in signed-magnitude format. All circuits are designed such that this accuracy is maintained across different chips and fabrication runs. This is particularly important in this design since learning is performed off-line and trimming individual chips is therefore not practical.

The shifter is 64 words (3 b each) wide and reads up to four inputs in each cycle. The use of a shifter limits the number of pins required, and provides direct support for convolutional-type network topologies and multiplexing of neurons with identical weights. This is achieved by scanning the input serially past the matrix multiplier Data loaded into the chip can be buffered temporarily in a file of 16 vector registers to reduce the required input bandwidth, and to evaluate neurons with more than 64 inputs.

The matrix multiplier consists of 4096 cells that perform a four-quadrant multiplication of a digital neuron input and an analog weight that is stored locally using a multiplying D/A converter. The contributions from individual multipliers are accumulated in a current-summing wire. The aspect ratio of the matrix multiplier is programmable to support neurons with 16 to 256 inputs. This configuration can be changed without overhead to permit efficient evaluation of multiple network layers with different topologies on a single chip.

The neuron bodies first scale the outputs from the matrix multiplier by a factor that can be set in the range of 1/16 to 1/2 in eight levels to optimize the useful dynamic range of the circuit. Then the nonlinear squashing function is evaluated and the result converted to the same 3-b signed-magnitude representation as is used in the input of the chip.

The weights in the matrix multiplier are stored as charge packets on capacitors and must be refreshed periodically from an external RAM. Two on-chip D/A converters update two values each clock cycle, concurrently with matrix multiplications.

The neuron transfer function implemented on the chip can be summarized by the following equations:

$$y = f\left(s \cdot \sum_{i=1}^{N} w_i x_i + b\right)$$

where

$$f(x) = \begin{cases} -1, & \text{if } x < -1 \\ x, & \text{if } -1 \leqslant x \leqslant 1 . \\ 1, & \text{if } x > 1 \end{cases}$$

The variables and parameters in the above equations can assume the following discrete values:

Neuron States:	$x_i, y \in \{-1, -2/3, -1/3, 0, 1/3, 2/3, 1\}$
Weights:	$w_i \in \{-1, \cdots, -1/31, 0, 1/31, \cdots, 1\}$
Bias:	$b \in \{-12, \cdots, -1/93, 0, 1/93, \cdots, 12\}$
Scale Factor:	$s \in \{1/2, 1/4, 1/6, 1/8, 1/10, 1/12, 1/14, 1/16\}$
Connections per Neuron:	$N \leqslant 256.$

The implementations of the matrix multiplier and neuron bodies are discussed in more detail below.

B. Matrix Multiplier

The matrix multiplier consists of 4096 individual multiplier cells that are arranged in eight blocks of eight vectors containing 64 multipliers and a variable bias (Fig

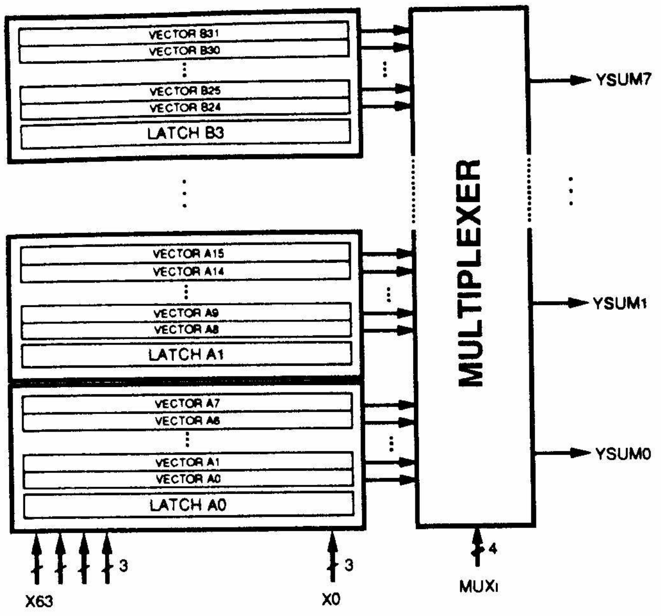


Fig 2 Matrix multiplier configuration

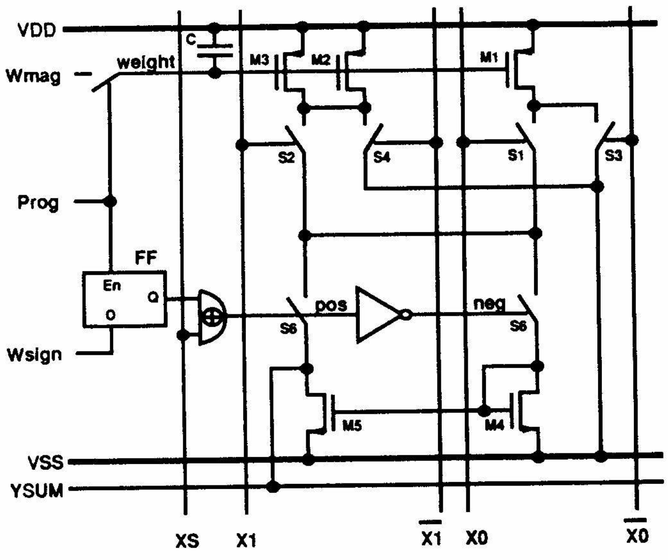


Fig 3 Schematic of the multiplier cell

2). The input to each block is held in a latch that is loaded from the shifter. The multiplexer combines the outputs of one to four vector multipliers and routes the sums to the neuron bodies. This arrangement supports neurons with 64, 128, or 256 inputs.

Fig. 3 shows the diagram of a multiplier cell with the local weight storage. Transistors $M1$ to $M3$ and switches $S1$ to $S4$ constitute a multiplying D/A converter (MDAC) that computes the product of the magnitude of the digital input $(X0, X1)$ and the weight. The magnitude of the weight is stored as a charge packet on capacitor C and the gates of the MDAC current sources [15], [16]. Proper operation requires that the gate capacitances of $M1$ to $M3$ be constant. This is ensured by dumping the current into V_{SS} when it is not needed at the output. The sign of

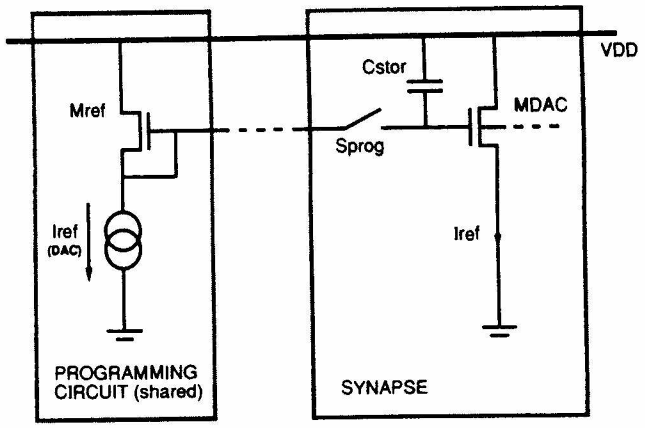


Fig 4 Weight refresh circuit

the product is computed digitally by an XOR gate that controls switches $S5$ and $S6$. These connect the MDAC output either directly to the $YSUM$, or through the mirror $M4$, $M5$. The summing wire $YSUM$ accumulates the contributions from all neuron inputs. It must be held at a constant potential of approximately 1 V by the neuron bodies to ensure proper biasing of the MDAC and current mirror outputs.

Despite increased area, a local current mirror in each multiplier cell has been preferred over a solution where positive and negative contributions from individual cells are summed on separate wires and the difference taken in the neuron body (cf. [17]). This latter solution is not viable for neurons with a large number of inputs because it is difficult to suppress quickly and accurately the large common-mode current in the two summing wires.

The capacitive storage of the magnitude of the weight must be refreshed periodically to compensate charge leakage. This is accomplished by the programming circuit shown in Fig. 4. The desired current is generated by a D/A converter and forced into the load transistor M_{ref} to produce the correct programming voltage, which is then copied onto the storage capacitor of the cell. This arrangement ensures a wide programming range from 0.8 to 4 V and consequent good accuracy. Simulations and measurements show that the error resulting from mismatch between M_{ref} and the current sources in the MDAC is negligible compared to the quantization error of the neural network chip. Two refresh D/A converters are integrated on the chip to achieve a 110-μs update cycle for all weights.

The refresh D/A converters and the A/D converters in the neuron bodies share a common on-chip reference current source. Since chip inputs and outputs are digital, the arithmetic results are independent of the reference, which is designed to maximize the dynamic range of internal analog signals independent of process parameter variations. In particular, the voltage range of the weight storage capacitors is chosen to be as large as possible to minimize errors due to charge leakage. The nominal full-scale current per MDAC is approximately 50 μA.

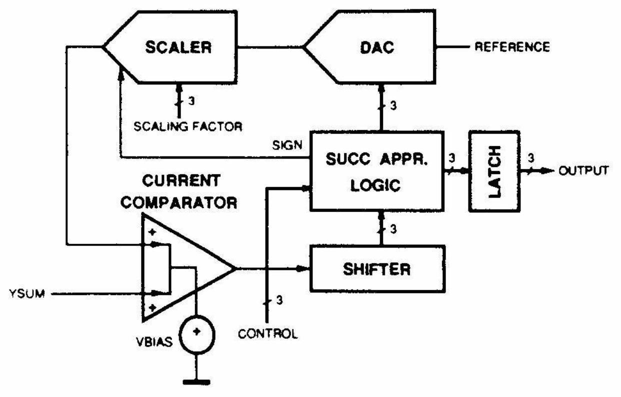


Fig 5 Neuron body

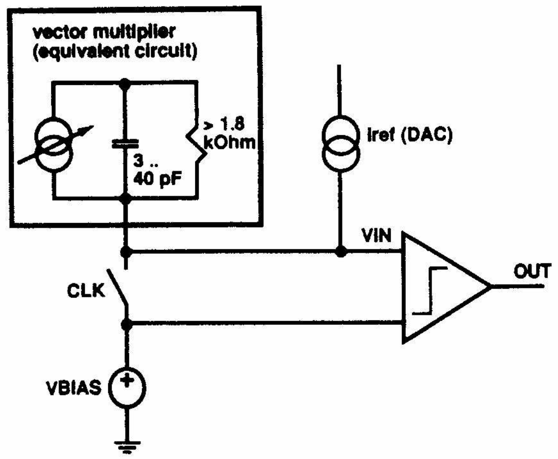


Fig 6 Current comparator principle

C Neuron Bodies

The neuron body circuit, illustrated in Fig 5, scales the current output from the matrix multiplier and converts it to a 3-b digital representation. The nonlinear squashing function is a side effect of the overload characteristics of the converter.

A successive-approximation converter has been chosen over a flash converter to avoid the need to create several copies of the signed output current from the vector multiplier. A programmable scaler in the reference current generator permits the slope of the nonlinear function to be adjusted in eight steps from 1/16 to 1/2 Depending on the setting, the full-scale output of the neuron is generated when at least two to 16 multiplier cells produce their maximum output current

The comparator at the input of the A/D converter senses the sign of the sum of the current from the vector multiplier, and the D/A reference current. The design of the multiplier cells mandates that the output is terminated into a low-impedance voltage source of about 1 V. Conversely, a high-impedance load is preferred in order to reduce the sensitivity requirements on the comparator. These conflicting requirements are reconciled in the circuit shown in Fig 6, where the requirements are met individually in two separate clock phases. The vector multiplier is represented as a variable current source with a parasitic capacitance and resistance. Simulations indicate that these parasitics vary between 3 and 40 pF, with a resistive component that can be as low as 1.8 kΩ, depending on the number of multiplier cells connected to the vector.

The current comparator operates as follows. During clock phase 1, the vector multiplier output is connected to a voltage source $V_{\text{Bias}} \approx 1$ V to minimize the load impedance. It consists now only of the sum of the impedance of the voltage source and the switch resistance During phase 2, the comparison phase, a high load is desired This is achieved by disconnecting the comparator input from V_{Bias}. Now, the parasitic impedance acts as a load. Since its time constant is usually larger than the duration of the clock phase (25 ns), this impedance is mostly capacitive. The current charges the parasitic capacitance and causes a voltage difference to develop at the input of the comparator circuit which is detected with a high-gain amplifier followed by a latch.

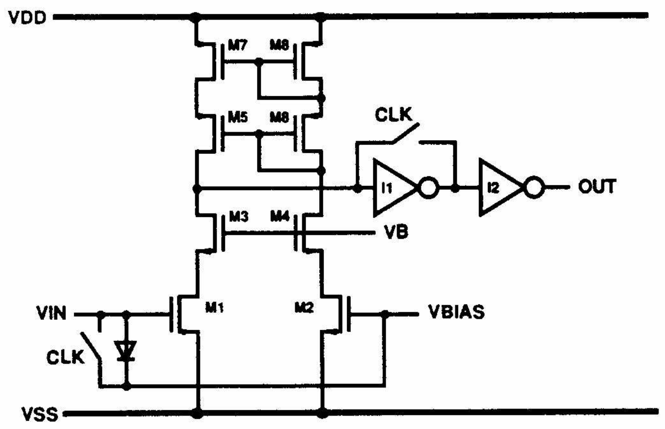


Fig 7 Circuit diagram of the current comparator

The complete schematic of the comparator is shown in Fig. 7. It consists of a differential amplifier with symmetric load, followed by two inverters that amplify the signal to logic levels. In the first clock phase, all switches are closed and the input is connected to V_{Bias}. In this phase, the current flowing from the differential stage into inverter $I1$ is close to zero, since $M1$ and $M2$ have equal gate voltages, and the drain-to-source voltages are forced to be equal by the cascode transistors $M3$ and $M4$. These transistors together with the cascode current mirror ensure a low comparator offset. In the second clock phase, all switches are opened and the comparator detects the sign of the voltage difference that builds up at its input. The diode at the input clamps the voltage to a maximum of about 1.7 V to speed up discharge during the first phase.

IV. Experimental Results

The neural network chip has been fabricated in a single-poly, double-metal 0.9-μm CMOS technology with 5-V power supply. The die measures 4.5×7 mm^2 (Fig. 8)

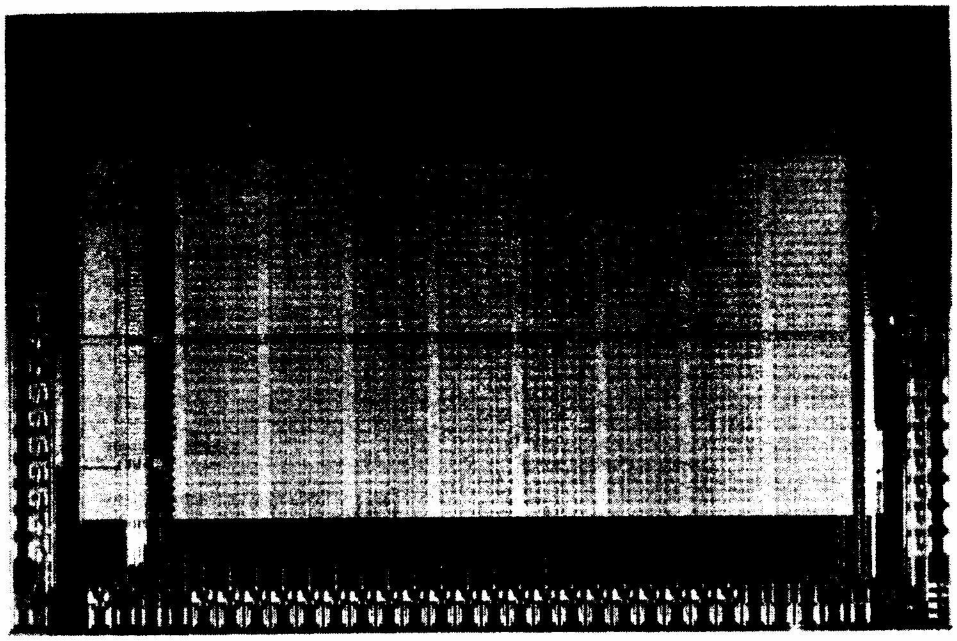


Fig 8 Die photograph of the neural network chip

and is packaged in a 96-pin grid array. The matrix multiplier (center) and the shifter and register file are pitch-matched in order to avoid the need for extra routing channels The typical operating current is less than 100 mA, but reaches 250 mA when all weights are programmed to their maximum value. The chip contains over 180 000 transistors. Despite the lack of redundancy, the yield is high since many devices are larger than minimal size, and since the circuits are insensitive to parameter variations.

Testing is performed in several steps. The overflow of the shifter is connected to pins and used to check the shifter and vector register file for correct operation A test pin provides access to the summing wire of one of the vector multipliers. This feature has been used to functionally test the multiplier cells, neuron bodies, and refresh D/A converters. Finally, statistical techniques are used to determine the overall computation error of the chip, which is comprised of the nonidealities in the multiplier cells, the neuron bodies, and the weight storage and refresh circuits. This error cannot be measured by simply comparing the chip output with a perfectly accurate simulation, since the quantization errors from the ADC's in the actual and simulated neuron body are correlated. Instead, the quantized output of the chip is compared to the simulation result before quantization. Ideally, this signal contains only the quantization error with approximately uniform distribution between plus and minus half the quantizer step size, as is indicated by the simulation result shown in Fig. 9. The measured distribution is wider due to inaccuracies in the analog computation, with a peak error of 0.7 LSB. The measured standard deviation of the error is 0.30 and corresponds to an inaccuracy of less than 0.5 dB (0.1 b) compared to an ideal chip. This

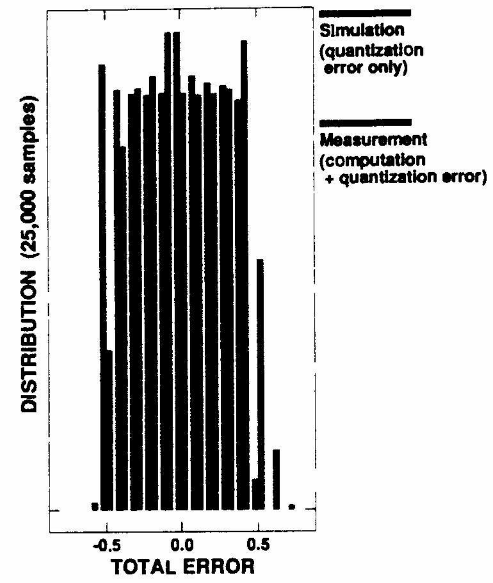


Fig 9 Error statistics

result is obtained consistently across different dies where fabrication tolerances have been simulated with intentional exposure time variations in the manufacturing process.

The chip executes three instructions, CALC, SHIFT, and STORE, to perform computations, load data from an external data source, and transfer data between the shifter, register file, and vector multiplier banks. The CALC instruction takes four cycles of 50 ns; the other two operations execute in a single clock cycle concurrent with an ongoing CALC instruction. In 200 ns the chip can, for example, load eight states and store them in a register and two latches, and evaluate the dot product and nonlinear function of eight vectors with 256 components each.

TABLE I
SYSTEM FEATURES

Weights (physical)	4096
Bias Units	256
Inputs per Neuron	16 to 256
Weight Accuracy	6 b
State Accuracy	3 b
Input Rate	120 Mb/s
Output Rate	120 Mb/s
On-chip Data Buffers	4 6 kb
Computation Rate (sustained)	5 GC/s
Refresh (all weights)	110 μs
Clock Rate	20 MHz

TABLE II
SAMPLE NETWORK ARCHITECTURES AND PERFORMANCE

Network Topology	Average Performance
Fully Connected (single layer)	
64 inputs 64 outputs	2 1 GC/s
128 inputs, 32 outputs	1 2 GC/s
32 inputs, 128 outputs	1 2 GC/s
Local Receptive Fields	
64 × 1 receptive field, 64 features	2 3 GC/s
16 × 16 receptive field, 16 features	4 7 GC/s
16 × 8 receptive field, 32 features	3 6 GC/s
Multilayer Network	
64 inputs, 32 hidden, 32 hidden, 32 outputs	0 8 GC/s
Hopfield Neural Network	
64 neurons	2 1 GC/s

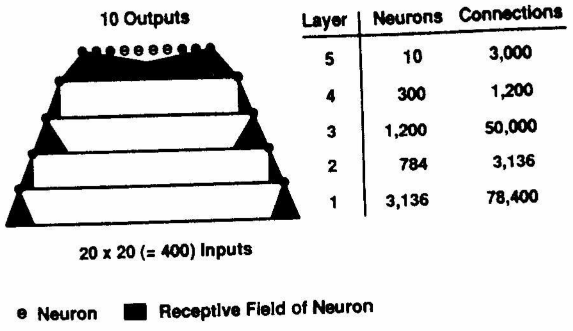
Programming is supported by an assembler and a code generator Table I summarizes the features of the chip.

Programmability is one of the key features of the neural network chip. Table II lists a selection of network topologies that can be implemented and the achieved performance in each case. The chip processes networks with full or sparse connection patterns of selectable size, as well as networks with feedback at a sustained rate of over 10^9 connection updates per second. An external feedback path must be provided for multilayer and Hopfield neural networks.

V High-Speed Character Recognition Application

Speed, capacity, and programmability are important aspects of neural network hardware. Their practical relevance, however, must be proven on a real-world application. In this section, the implementation of an optical digit recognizer [6], [18] on the neural network chip is described. This network has been trained with the back-propagation algorithm [1] on a workstation with floating-point arithmetic to recognize handwritten digits from a 20×20-pixel image The classification error rate on a test set consisting of 2000 handwritten digits is 4.9% miss classifications, compared to a human performance of 2.5% on the same data.

Fig. 10 illustrates the architecture of the network. The more than 3500 neurons with a total of over 133 000 connections are arranged in five layers. The first four layers employ a two-dimensional convolutional type topology with various kernel sizes and subsampling factors.



Layer	Neurons	Connections
5	10	3,000
4	300	1,200
3	1,200	50,000
2	784	3,136
1	3,136	78,400

Fig 10 Architecture of the character recognition network

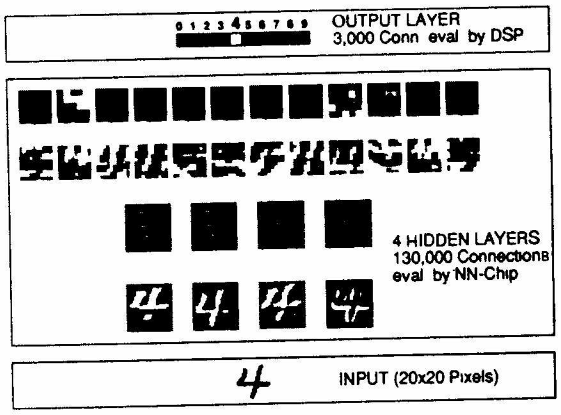


Fig 11 Example chip output for optical character recognition The gray levels encode the neuron state

The last layer is fully connected. This topology has been chosen to maximize recognition performance and classification speed of an implementation on a floating-point digital signal processor (DSP-32C [19]).

Special steps are necessary to adapt the network to the low resolution of the chip. Simple quantization of all weight values from floating-point accuracy to the 6-b signed-magnitude representation of the chip results in an unacceptable loss of accuracy. However, experiments reveal that the computational accuracy provided by the chip is adequate for all but the 3000 weights in the last layer of the network. This last layer is retrained with quantized data obtained from the chip to eliminate performance degradation. After retraining, the classification error rate on the test set is 5.3%, compared to the original 4.9%. This result is obtained consistently with different chips for which the last layer has not been retrained individually. Fig. 11 shows the input, output, and internal states of the neural network for a sample input that has been processed by the neural network chip.

The first four layers of the network with 97% of the connections fit on a single neural network chip. The remaining 3000 connections of the last layer are evaluated on a DSP32C digital signal processor. The throughput of the chip is more than 1000 characters per second or 130 MC/s. This figure is considerably lower than the peak performance of the chip (5 GC/s), a consequence of the small number of inputs of most neurons in the network for which the chip cannot fully exploit its parallelism

Nevertheless, the chip's performance compares favorably to the 20 characters per second that are achieved when the entire network is evaluated on the DSP32C. The recognition rate of the chip is far higher than the throughput of the preprocessor, which relies on conventional hardware. Improvements of both the recognition rate and accuracy can be expected when the network architecture is tuned to take full advantage of the power of the neural network chip.

VI Conclusions

Neural networks are attractive for pattern classification applications but suffer in practice from the limited speed that can be achieved with implementations based on classical sequential processors. This problem can be overcome with highly parallel special-purpose VLSI circuits. While a fully parallel implementation of sufficiently large networks is currently not feasible, adequately high performance can be achieved with an architecture that exploits the limited connectivity and weight sharing that are typical for pattern classifiers. This has been demonstrated with a neural network classifier with over 133 000 connections that has been implemented on a single neural network chip performing more than 1000 classifications per second The availability of fast special-purpose hardware for large applications permits exploration of new neural network algorithms and problems of a scale that would not be feasible with conventional processors.

Acknowledgment

The authors express their thanks to S. Rao for his RAM generator.

References

[1] D E Rumelhart and J L McClelland, *Parallel Distributed Processing — Explorations in the Microstructure of Cognition*, vol 1 Cambridge, MA MIT Press, 1986
[2] R P Lippmann, "An introduction to computing with neural nets,' *IEEE ASSP Mag* vol 4, no 4, pp 4–22, Apr 1987
[3] K. Fukushima and S Miyake, 'Neocognitron A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol 15, pp 455–469, 1982
[4] A Waibel, T Hanazawa, G Hinton, K Shikano, and K Lang, 'Phoneme recognition using time-delay neural networks," *IEEE Trans Acoust Speech, Signal Processing*, vol 37, pp 328–339, Mar 1989
[5] R P Lippmann, "Review of neural networks for speech recognition ' *Neural Computation*, vol 1, no. 1, pp. 1-38, Spring 1989
[6] Y Le Cun *et al*, 'Handwritten digit recognition with a back-propagation network,' in *Neural Information Processing Systems*, D S Touretzky, Ed, vol 2 San Mateo, CA Morgan Kaufmann, 1990, pp 396–404
[7] I Guyon, P Albrecht, Y Le Cun, J S Denker, and W Hubbard ' Design of a neural network character recognizer for a touch terminal, ' *Pattern Recognition*, vol. 24, no 2, pp 105–119, 1991
[8] R. A Milito, I. Guyon, and S A Solla, "Neural network implementation of admission control," in *Advances in Neural Information Processing Systems 3*, D S Touretzky and R Lippman, Eds, vol 3 San Mateo, CA Morgan Kaufmann, 1991
[9] M Mezard and J P Nadal, "Learning in feed-forward layered networks The tiling algorithm," *J Phys A*, vol A-22, pp 2191-2203, 1989
[10] Y Arima *et al*, "A 336-neuron 28k-synapse self-learning neural network chip with branch-neuron architecture," in *ISSCC Dig Tech Papers*, 1991, pp 182–183
[11] J Alspector, R Allen, and A Jayakumar, "Relaxation networks for large supervised learning problems," in *Neural Information Processing Systems*, R Lippmann, J Moody and D Touretzky, Eds., vol 3 San Mateo, CA: Morgan Kaufmann, 1991, pp 1015–1021
[12] L D Jackel *et al*, "VLSI implementations of electronic neural networks An example in character recognition," in *Proc IEEE Int Conf Syst, Man, Cyb*, 1990
[13] M Stevenson, R Winter, and B Widrow, "Sensitivity of feedforward neural networks to weight errors," *IEEE Trans Neural Networks*, vol 1, no 1, pp 71–80, Mar 1990
[14] H P Graf and D Henderson, ' A reconfigurable CMOS neural network," in *ISSCC Dig Tech Papers*, Feb 1990, pp 144–145
[15] W Groeneveld, H Schouwenaars, and H. Termeer, "A self-calibration technique for monolithic high-resolution D/A converters," in *ISSCC Dig. Tech Papers*, 1989, pp 22–23
[16] T Morishita, Y Tamura, and T Otsuki, "A BiCMOS analog neural network with dynamically updated weights," in *ISSCC Dig Tech Papers*, 1990
[17] T H Borgstrom, M Ismail, and S B Bibyk, "Programmable current-mode neural network for implementation in analogue MOS VLSI," *Proc Inst Elec Eng Pt G*, vol 137, no 2, pp 175–184, 1990
[18] B E. Boser *et al*, "An analog neural network processor and its application to high speed character recognition," in *Proc Int Joint Conf Neural Networks*, July 1991, pp 415–420
[19] M L Fuccio *et al*, "The DSP32C AT&T's second-generation floating-point digital signal processor,' *IEEE Micro*, pp 30–48, 1988

Bernhard E. Boser (S'79–M'83) received a diploma in electrical engineering in 1984 from the Swiss Federal Institute of Technology in Zurich, Switzerland, and the M S and Ph D degrees from Stanford University, Stanford, CA, in 1985 and 1988, respectively. His thesis was on the design and implementation of oversampled analog-to-digital converters

Since 1988 he has been with AT&T Bell Laboratories, Holmdel, NJ, where he is working on VLSI implementations of artificial neural networks for pattern recognition applications

Eduard Sackinger (S'84–M'91) was born in Basel, Switzerland on August 13, 1959 He received the M S and Ph D degrees in electrical engineering from Swiss Federal Institute (ETH), Zurich, Switzerland, in 1983 and 1989, respectively His doctoral work was on the theory and CMOS integration of a differential difference amplifier

In the fall of 1983 he joined the Electronics Laboratory of the Swiss Federal Institute of Technology where he investigated analog applications to floating-gate devices Since September of 1989 he has been working for AT&T Bell Laboratories in Holmdel, NJ, in the field of artificial neural-network hardware and its applications to character recognition His research interests include analog circuit design and parallel distributed processing

Jane Bromley received the B Sc honors degree in physics in 1983 and the Ph D. degree in biophysics in 1987, both from Imperial College, London University Her Ph D thesis and postdoctoral work involved the study of human vision and visual dysfunction caused by brain lesions using psychophysical techniques

In 1990 she joined AT&T Bell Laboratories, Holmdel, NJ, and has been working on the application of artificial neural networks to human perception tasks, including the reading of handwritten text

Yann Le Cun (S'87–M'87) was born in France in 1960. He obtained the Diplome d'Ingenieur in electrical engineering from the Ecole Superieure d'Ingenieurs en Electrotechnique et Electronique in Paris in 1983, and the Ph.D. degree in computer science from the Universite Pierre et Marie Curie, Paris, in 1987. His thesis was about connectionist learning models, and introduced an early version of the back-propagation learning algorithm.

From August 1987 to October 1988 he was a Research Associate in the Department of Computer Science of the University of Toronto, Canada. Since October 1988, he has been a Member of Technical staff at AT&T Bell Laboratories Holmdel, NJ. His current interests include neural networks and connectionist models, learning theory, pattern recognition, and VLSI implementations of massively parallel systems.

Lawrence D. Jackel (M'82–SM'90) was born in New York City. He received the B.S. degree in physics from Brandeis University, Waltham, MA, in 1969 and the Ph.D. degree in experimental physics from Cornell University, Ithaca, NY, in 1976. His thesis topic was in the field of superconducting devices.

Since 1975 he has been at AT&T Bell Laboratories, Holmdel, NJ, where his initial research was in microfabrication and microscience. In 1984 he became Head of the Device Structures Research Department and began studies in artificial neural networks. In 1990 his department was renamed the Adaptive Systems Research Department and pursues theory, applications, and hardware for machine learning and machine perception.

Dr. Jackel is a member of the American Physical Society.