A Digital Architecture Employing Stochasticism for the Simulation of Hopfield Neural Nets

DAVID E. VAN DEN BOUT AND THOMAS K. MILLER III

Abstract—**A digital architecture which uses stochastic logic for simulating the behavior of Hopfield neural networks is described. This stochastic architecture provides *massive parallelism* (since stochastic logic is very space efficient), *reprogrammability* (since synaptic weights are stored in digital shift registers), *large dynamic range* (by using either fixed or floating-point weights), *annealing* (by coupling variable neuron gains with noise from stochastic arithmetic), *high execution speeds* ($\approx N \cdot 10^8$ connections per second), *expandability* (by cascading of multiple chips to host large networks), and *practicality* (by building with very conservative MOS device technologies). Results of simulations are given which show the stochastic architecture gives results similar to those found using standard analog neural networks or simulated annealing.**

I. Introduction

ARTIFICIAL neural networks are a family of massively parallel architectures that solve difficult problems via the cooperation of highly interconnected but simple computing elements. The speed and solution quality obtained when using neural networks for solving specific problems in visual perception [2] and dynamic control [5] make specialized neural network implementations attractive. For instance, the Hopfield network of Fig. 1 can be used as an associative memory [11] or for solving various combinatorial problems [9] by the programming of synaptic weights stored as a conductance matrix.

Analog implementations of the Hopfield network containing up to 512 neurons have been built with matrices of fixed resistors and nonlinear amplifiers fabricated on a single chip [8]. Variable resistors are needed in order to change the problem constraints, but the increased complexity of such interconnections reduces by an order of magnitude the number of neurons that can be built on a chip [14], [18], [16]. Real-world applications will require many more neurons than this, so finding a method of interconnecting these chips to form larger networks will be a primary concern. This task is made difficult by the large number of analog signals which must pass between chips and by the external parasitic capacitances which will distort the charging characteristics of the network and possibly cause erroneous results.

Manuscript received June 20, 1988; revised October 12, 1988. This paper was recommended by Guest Editors R. W. Newcomb and N. El-Leithy.

The authors are with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911.

IEEE Log Number 8826718.

The limitations of analog computing have led researchers of neural networks to rely upon digital simulation. The Hopfield network minimizes objective functions of the form

$$E = -\frac{1}{2}\sum_{j=0}^{N-1}\sum_{i\neq j}^{N-1} G_{ij}v_i v_j + \sum_{j=0}^{N-1} I_j v_j \qquad \text{where } G_{ij} = G_{ji}$$

and will converge to a stable solution if $dE/dt < 0$. This can be rewritten as

$$\frac{dE}{dt} = \sum_{j=0}^{N-1} \frac{\partial E}{\partial v_j}\cdot\frac{dv_j}{du_j}\cdot\frac{du_j}{dt} < 0$$

which can be guaranteed if $dv_j/du_j > 0$ and $\partial E/\partial v_j = -du_j/dt$. The first condition is satisfied by using any strictly increasing transfer function relating v_j to u_j. The second condition is satisfied by making

$$\frac{du_j}{dt} = \sum_{i\neq j} G_{ij}v_i + I_j \Rightarrow u_j(t+\Delta t) \approx u_j(t) + \left\{\sum_{i\neq j} G_{ij}v_i(t) + I_j\right\}\cdot\Delta t \tag{1}$$

which resembles the Hopfield network charging equation minus the capacitive decay current. The current source term, I_j, can also be absorbed into the summation by adding a constant *bias neuron* (i.e., $v_B = 1$) that charges each neuron j through a conductance of $G_{Bj} = I_j$. Single-chip digital signal processors (DSP's) excel at inner product computations such as those in (1) and have been used to build neural network architectures [17]. Since most DSP's contain only one hardware multiplier, a large and expensive system employing hundreds of DSP chips would be needed in order to utilize all the potential parallelism available in a neural network.

Stochastic systems [6] use binary signals which randomly assume either the value 0 or 1. The average of a stochastic signal can be viewed as an analog value in the range [0,1] and can be changed by altering the probability distribution function (PDF) of the signal. Neural networks built from stochastic logic elements [7,15] avoid the problems of the preceding implementations and have the following advan-

0098-4094/89/0500-0732$01.00

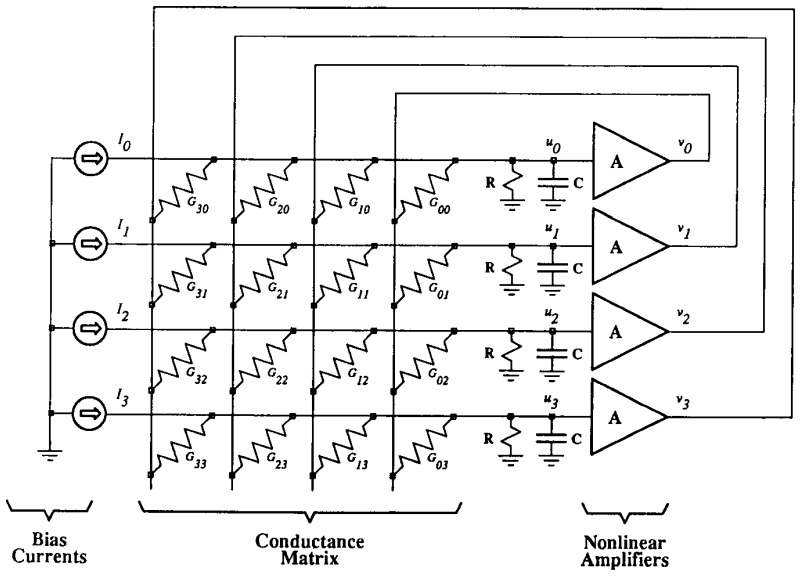


Fig. 1. A Hopfield neural network.

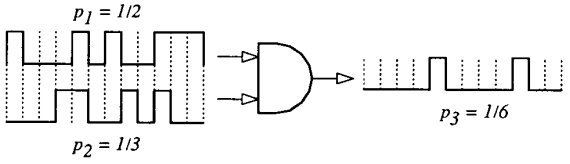


Fig. 2. An example of stochastic multiplication.

tages:

1) Pseudoanalog computations are easily performed on stochastic signals using very space-efficient digital logic. For instance, stochastic signals can be multiplied using only a simple AND gate (Fig. 2), and capacitors can be emulated using digital counters.
2) The simplicity and small size of stochastic circuitry permits massive parallelism.
3) Digital stochastic signals are more easily passed between chips and suffer less from noise and parasitic capacitance than analog signals.

II. A Simple Stochastic Architecture: Theory of Operation

A stochastic neural network architecture [4] will now be presented which distributes the summation in (1) over N time slices [1] of length δt. During time slice n ($0 \leqslant n < \infty$), the output of amplifier $i = n \bmod N$ is used to send charge to all the capacitors. The charging equation for u_j in this altered network is

$$u_j(n\,\delta t + \delta t) = u_j(n\,\delta t) + G_{ij} v_i(n\,\delta t)\,\delta t, \quad \text{with } i = n \bmod N. \quad (2)$$

This expression behaves similarly to (1) as long as the time slices are much smaller than the main integration period (i.e., $\delta t \ll \Delta t$) so that the capacitor voltages do not change too much during a time slice. No deleterious effects caused by this time-multiplexing have been noted in any of our experiments.

Fig. 3 shows the translation of the above idea into a small, all-digital, stochastic neural architecture containing four neurons ($N = 4$). On the right hand side of the figure are four counters which are connected as a circular shift register. The counters contain the neural state vector arranged such that the jth counter initially contains the jth component of the state vector, u_j. The 16 synaptic conductance coefficients are stored in four circular shift registers such that the ith cell of the jth circular shift register initially contains $G_{i,(i+j) \bmod N}$.

Once initialized, the contents of counters $C_0, C_1, \cdots, C_{N-1}$ are continuously rotated such that counter C_j contains $u_{(n+j) \bmod N}$ at clock cycle n. Simultaneously, the synaptic shift registers are also continuously rotated such that the output of the jth shift register is $G_{n \bmod N,(n+j) \bmod N}$. Thus the neural firing signal derived from the output of C_0 (which contains $u_{n \bmod N}$) arrives at the same time as the weights through which it influences the other neurons (Fig. 4).

Stochastic arithmetic is now used to compute the component of charge to be added to each simulated neural capacitor. During each clock cycle, the output of counter C_0, u_i, is compared with a random number, R_1, uniformly distributed over $[R_{\min}, R_{\max}]$. The output of the comparator will pulse if $R_1 < u_i$, thus creating a stochastic neural firing signal v_i whose mean is proportional to u_i provided $u_i \in [R_{\min}, R_{\max}]$. If u_i is outside this range, the mean will saturate at either 0 or 1, providing the needed nonlinearity in the network amplifiers. Similarly, the outputs of the N synaptic shift registers are compared to another uniformly distributed random number, R_2, to provide N stochastic signals with means proportional to the corresponding

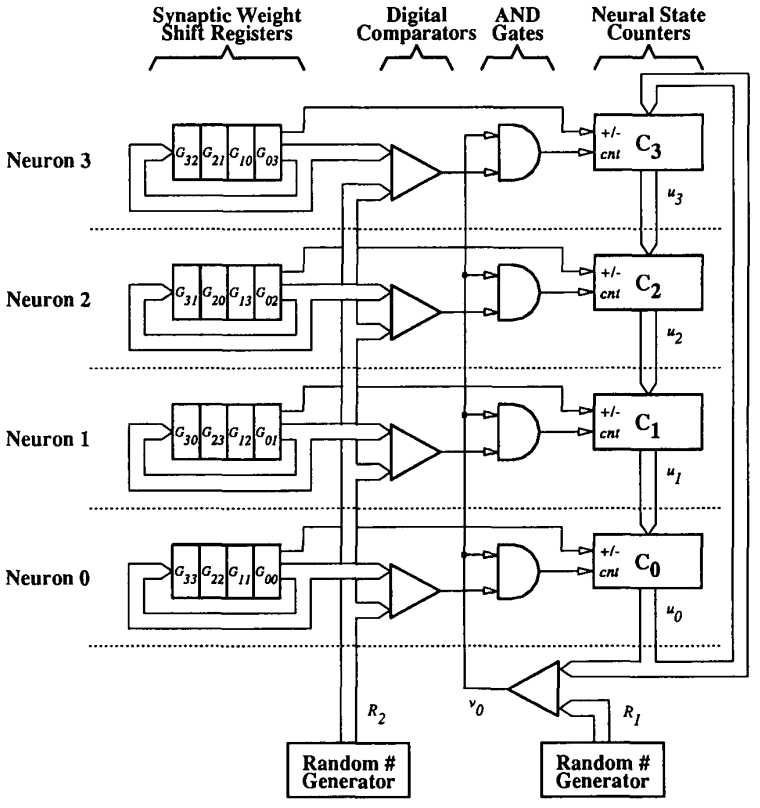


Fig. 3. The stochastic neural network architecture at cycle $n = kN$.

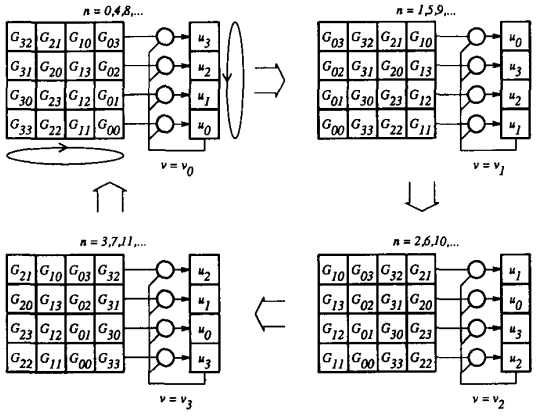


Fig. 4. The flow of data in the stochastic architecture.

synaptic weights. ANDing the neural firing signal with each synaptic weight signal creates a third set of stochastic signals that pulse with probabilities of $|G_{ij}v_i|$. A pulse from an AND gate will cause the connected counter to either increment or decrement u_j depending upon the sign of G_{ij}. So the charging equation for this network is

$$u_j(n+1) = u_j(n) + G_{ij}v_i(n) \qquad \text{with } i = n \bmod N$$

which is the same as (2).

A fully parallel neural network of N neurons can process all N^2 connections in a single cycle, while the pipeline-ring architecture described above requires N clock cycles due to its time-multiplexed nature. The advantage, however, is that each neuron is connected to the rest of the system by only the following signals: two buses for inputing and outputing counter values, a bus for receiving random number R_2, and a single wire which carries the neural firing signal. If the synaptic weight shift registers are built to hold kN_c coefficients, where N_c is the number of artificial neurons per chip and k is an integer, then it becomes possible to cascade k chips and construct much larger networks without an increase in the number of I/O pins (Fig. 5).

The use of random numbers also confers significant advantages. For example, the neural firing comparator pulses only when $R_1 < u$ and thus has a mean output voltage of

$$\langle v \rangle = 1 \cdot \Pr\{R_1 < u\} + 0 \cdot \Pr\{R_1 \geq u\} = \Pr\{R_1 < u\},$$

which is the definition of the cumulative distribution function (CDF) of R_1. Therefore, the neural transfer function can be altered by adjusting the PDF of R_1 (Fig. 6). While a uniform PDF gives a linear transfer function with hard limits, a PDF of

$$\Pr\{R_1 = k\} = \frac{e^{-k}}{1+e^{-k}} \cdot \frac{1}{1+e^{-k}}$$

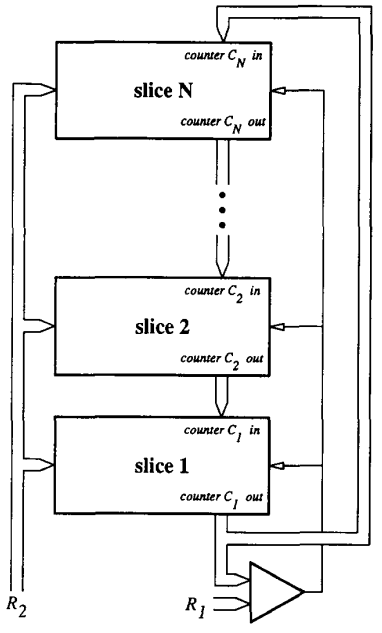


Fig. 5. Interconnecting the stochastic neural network chips.

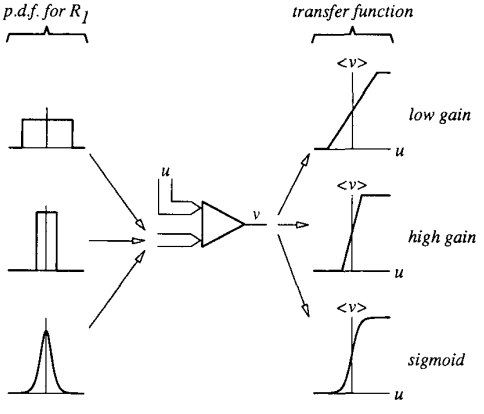


Fig. 6. Altering the PDF of R_1 to get different neural transfer characteristics.

gives the more familiar sigmoidal transfer characteristic

$$\langle v \rangle = \int_{-\infty}^{u} \frac{e^{-k}}{1+e^{-k}} \cdot \frac{1}{1+e^{-k}}\, dk = \frac{1}{1+e^{-u}}.$$

Also, tightening the interval over which R_1 ranges effectively increases the gain of the amplifier. If the capacitive decay current is also simulated, this permits the stochastic architecture to do *annealing* since increasing the gain acts in a manner analogous to lowering the temperature in the simulated annealing process [9].

The PDF for R_2 can also be altered to more efficiently encode a set of synaptic weights. Assuming the stochastic signal generated by the largest synaptic weight, g_M, should pulse with a probability of 1, then a weight g_k should

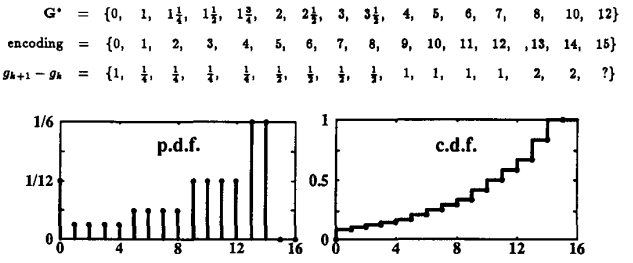


Fig. 7. Encoding synaptic weights and computing the PDF for R_2.

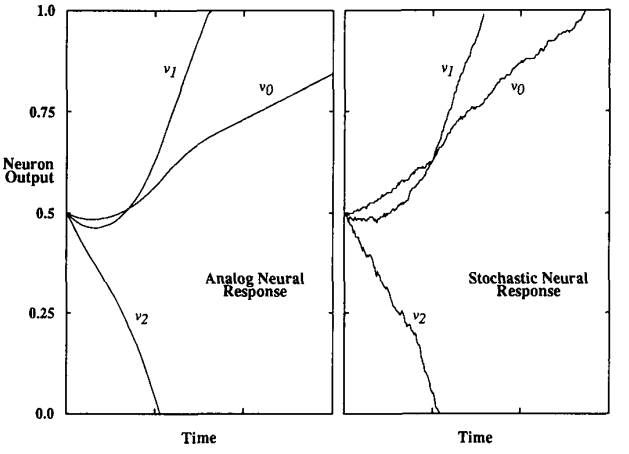


Fig. 8. Analog and stochastic neuron outputs for the A/D problem.

generate pulses with a probability of g_k/g_M. The following procedure constructs a discrete PDF for R_2 based on the differences between these probabilities:

1) Arrange the absolute values of all the synaptic weights plus a weight of zero into ascending order and eliminate all duplicate entries to create a list $\boldsymbol{G}^* = \{0, g_1, \cdots, g_M\}$.
2) Encode each of the synaptic weights by its position within the sorted list plus an additional sign bit.
3) Create a discrete PDF for R_2 where,

$$\Pr\{R_2 = k\} = \begin{cases} \dfrac{g_{k+1}}{g_M} - \dfrac{g_k}{g_M}, & \text{for } 0 \leqslant k < M \\ 0, & \text{otherwise.} \end{cases}$$

The use of this algorithm to encode floating-point synaptic weights with 2-bit mantissas and exponents is illustrated in Fig. 7. Note that the interpretation of the encoding is a function of the PDF of R_2 alone—the circuitry need not be changed in order to support floating-point weights!

III. Simulation Results

Fig. 8 compares the evolution of the neural state vector in the stochastic network to that of an analog Hopfield network when both networks are programmed to act as analog-to-digital converters [10] and are given an input voltage corresponding to a digital code of **011**. As can be

TABLE I
A COMPARISON OF A/D CONVERGENCE TIMES AND PROGRAMMING FLEXIBILITY

Type of Architecture	Convergence Time	Programming Flexibility
VLSI analog Hopfield neural network	$1\mu s$	low
Stochastic neural network (100 MHz)	$200\mu s$	moderate
Hopfield network simulated on a μVAX	$1,300,000\mu s$	high

seen, the state vectors evolve similarly except for the natural addition of some noise in the stochastic version. The speed of convergence and flexibility of programming for this problem on several architectures is shown in Table I. The stochastic architecture offers a good compromise between the high speed of dedicated analog VLSI networks and the flexibility of a general-purpose computer.

As a second test, the stochastic architecture was used to divide a graph containing N nodes into two subgraphs containing $\approx N/2$ nodes while cutting as few edges as possible. Solving such a bipartitioning problem [12] involves minimizing the objective function

$$E = \sum_{i=0}^{N-1} \sum_{j \neq i} E_{ij}(1-v_j)v_i - r \sum_{i=0}^{N-1} \sum_{j \neq i} (1-v_j)v_i$$

where the variables have been defined so that $v_i = 0$ if the node i is in the first subgraph and $v_i = 1$ otherwise. Individual nodes i and j are attracted into the same subgraph by edges of strength E_{ij} while clustering is discouraged by the amorphous repulsive force, r. One hundred bipartitionings of an arbitrary graph containing 84 nodes and 115 unit-weight edges were done using both the stochastic architecture and simulated annealing [3]. Solutions found using simulated annealing had an average of 3.07 cut edges while 5.85 cut edges existed in solutions generated by the stochastic architecture (a randomly generated solution would contain 57.5 cut edges, on average). However, simulated annealing required an average of 329 seconds to converge to a solution versus just 4.2×10^{-3} s for the stochastic architecture. Obviously, a dedicated analog VLSI network would be even faster (just as it was in the previous example), but no hard data is available.

A comparison of the stochastic architecture to other neural network implementations [13] is given in Fig. 9 in terms of the storage capacity and processing speed (in connections/second). In general, a stochastic architecture of N neurons will have storage for N^2 synaptic weights and will process connections at a rate of $N \times f_s$, where f_s is the system clock speed. Even at a modest 10 MHz, the stochastic architecture outperforms the other implementations.

IV. IMPLEMENTATION ISSUES

The stochastic system shown in Figs. 3 and 5 is plagued by signal propagation delays (when transmitting v and R_2 to each artificial neuron) and computational delays (caused by the cascaded comparison, logical AND, and increment/

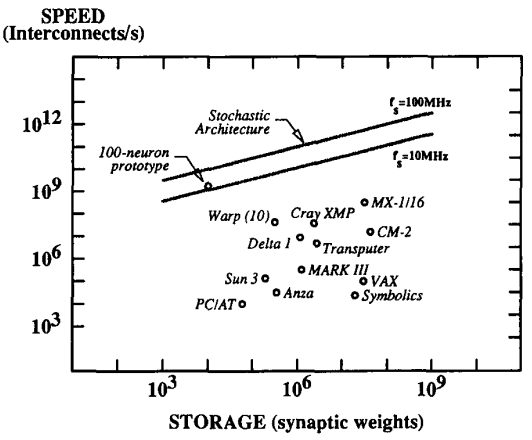


Fig. 9. A comparison of various neural network implementations.

decrement operations). The R_2 propagation delay can be eliminated once it is realized that each neuron does not have to receive the same random number, only one which has the *same characteristics*, i.e., the same PDF. Thus multiple random number generators with the same PDF could be placed on the circuit board to minimize the wiring length to each chip. Internal to each chip, interspersed pipeline registers along the R_2 bus (R_{20}, R_{21}, R_{22}, and R_{23} in Fig. 10) increase the system speed by reducing the wire length which must be driven during a clock cycle while still allowing each neuron to receive random numbers with the same probability distributions. Pipeline delays can also be introduced on the neural firing signal wire, but the counting and shifting operations of the neural state registers must be separated in order to maintain correct operation. (This exacts a very small penalty since the chip area is dominated by the shift registers which store the synaptic weights.) During operation, the neural state vector stored in the counters is transferred into the shift registers and is then shifted out to create the neural firing signal. The result of each neuron firing passes through the delay line and updates each neural state counter. Once the shift register has been emptied, the new neural state is transferred into the shift register and the process begins again.

The computational delay can be significantly reduced by using bit-level pipelining in the comparator and increment/decrement circuitry. By skewing the storage of the sign bit and p magnitude bits of each weight $G_{ij} = s_i g_{ip} \cdots g_{i1} g_{i0}$, then the comparison with R_2 can be done by a series of single-bit comparators. Single-bit registers exist between each comparator stage to store the intermediate borrow bits between cycles, which permits the processing of p comparisons simultaneously. The final borrow output is ANDed with the firing signal from neuron i and the result is used along with the sign bit to control a bit-level pipelined counter. (Additional logic prevents the counter from overflowing or underflowing during the charge integration process.) The computational delay is now determined solely by the time required to do 1-bit arithmetic.

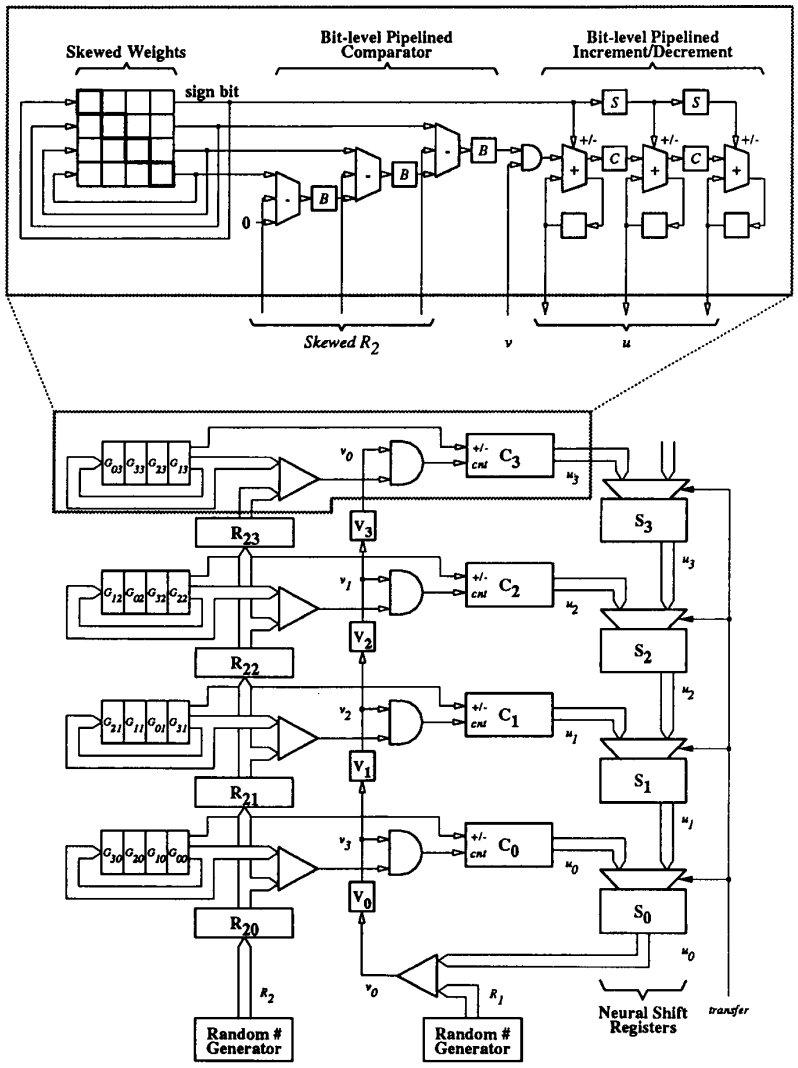


Fig. 10. Adding registers to create pipelining.

The combination of the changes described above with a modern silicon process should allow system clock speeds of 50–100 MHz.

V. Conclusions and Further Work

A neural network implementation based upon stochastic arithmetic has been described. This stochastic architecture is dynamically reprogrammable, is easily expanded using multiple chips, and uses a constant number of I/O pins no matter what the size of the neural network being simulated. Simulations show that the solutions produced by the stochastic neural net do not suffer any ill effects due to its probabilistic, time-multiplexed nature, yet its speed exceeds those of a wide variety of other neural network implementations by 2–6 orders of magnitude.

A prototype of the stochastic architecture which supports 100 neurons has been designed and is being fabricated. A more advanced version is being designed which accesses synaptic weights stored in external RAM's and is capable of learning by dynamically adjusting these weights.

References

[1] A. Agranat and A. Yariv, "A new architecture for a microelectronic implementation of neural network models," in *IEEE First Int. Conf., Neural Networks*, pp. 403–410, 1987.

[2] G. Bilbro, M. White, and W. Snyder, "Image segmentation with neurocomputers," in *Neural Computers*, Rolf Eckmiller and Christopher v. d. Malsburg, Eds., pp. 71–79, Springer-Verlag, 1987.

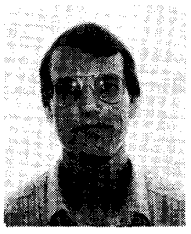
[3] D. E. Van den Bout and T. K. Miller III, "Graph partitioning by automated simulated annealing," *IEEE Trans. Computers*.

[4] ____, "A stochastic architecture for neural nets," in *IEEE Second Int. Conf. Neural Networks*, 1988.

[5] R. Eckmiller, "Neural networks for motor program generation," in *Neural Computers*, Rolf Eckmiller and Christopher v. d. Malsburg, Eds., pp. 359–370, Springer-Verlag, 1987.

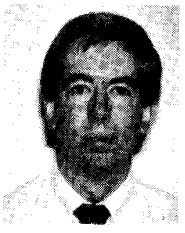
[6] B. Gaines, "Stochastic computing systems," in *Advances in Information and Systems Science*. New York: Elsevier, pp. 37–172, 1969.

[7] ____, "Uncertainty as a foundation of computational power in neural networks," in *IEEE First Int. Conf. on Neural Networks*, pp. 51–57, 1987.
[8] H. Graf, L. Jackel, R. Howard, B. Straughn, J. Denker, W. Hubbard, D. Tennant, and D. Schwartz, "VLSI implementation of a neural network memory with several hundreds of neurons," in *Amer. Inst. Physics Conf. Proc. 151*, pp. 414–419, 1986.
[9] J. Hopfield and D. Tank, "Neural computations of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.
[10] ____, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 533–541, May 1986.
[11] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci., U.S.A.*, vol. 81, pp. 3088–3092, 1984.
[12] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, pp. 291–307, Feb. 1970.
[13] Lincoln Laboratory, DARPA neural network study—executive summary, July 8, 1988.
[14] S. Mackie, H. Graf, and D. Schwartz, "Implementation of neural network models in silicon," in *Neural Computers*, Rolf Eckmiller and Christopher v. d. Malsburg, Eds., pp. 467–476, Springer Verlag, 1987.
[15] D. Nguyen and F. Holt, "Stochastic processing in a neural network application," in *IEEE First Int. Conf. Neural Networks*, pp. 281–292, 1987.
[16] J. J. Paulos and P. W. Hollis, "Neural networks using analog multipliers," in *Proc. IEEE Int. Conf. Circuits and Systems*, Helsinki, Finland, 1988.
[17] P. Penz and R. Wiggins, "Digital signal processor accelerators for neural network simulations," in *Amer. Inst. of Physics Conf. Proc. 151*, pp. 345–355, 1986.
[18] M. Sivilotti, M. Emerling, and C. Mead, "A novel associative memory implemented using collective computation," in *1985 Chapel Hill Conf. on Very Large Scale Integration*, pp. 329–342, 1985.



David E. Van den Bout received the B.S. degree from North Carolina State University in 1978 and the M.S. degree from Massachusetts Institute of Technology in 1979, both in electrical engineering, and the Ph.D. degree in 1987 from North Carolina State University where he is now a Research Assistant Professor.

From 1978 to 1983 he was a Member of Technical Staff at Bell Telephone Laboratories. His research interests include VLSI design, computer architecture, neural networks, and optimization theory.



Thomas K. Miller III received the B.A. degree in mathematics and chemistry from the University of North Carolina at Chapel Hill in 1976, the M.S. degree in biomedical engineering and mathematics in 1980, and the Ph.D. degree in 1982, both from UNC-Chapel Hill. Since 1982 he has been employed as an assistant professor in the Department of Electrical and Computer Engineering at North Carolina State University. His research interest include microprocessor architectures, parallel computing, neural networks, and biomedical signal processing.

Dr. Miller is a member of the Engineering in Medicine and Biology Society.