Pages 5-17, Advanced Research in VLSI: Proceedings of the Sixth MIT Conference, 1990.

STONN: A Stochastic Neural Network Chip

William Wike
David Van den Bout
North Carolina State University
Raleigh, NC 27695-7911
devb@csldevb.ncsu.edu

Abstract

A 100,000 transistor digital CMOS Hopfield neural network is presented and the performance is discussed. STONN uses space efficient stochastic logic and bit-wise pipelining to achieve massive parallelism and high operational speeds. The architecture produces solutions to optimization problems with quality equivalent to those produced by analog networks and nearly as good as those found using simulated annealing. The massively parallel nature of STONN increases its speed of convergence by orders of magnitude over uniprocessor implementations. The completely digital STONN design provides dynamically reprogrammable parameters and a practical system implementation which can be expanded using several identical chips.

1 Introduction

Artificial neural networks composed of a large number of highly interconnected but simple computational elements (i.e. *neurons*) have shown promise in solving difficult problems. For example, the Hopfield network has been used to solve simple [12] and complex combinatorial problems [13].

In an N-neuron Hopfield network, the charging equation for the voltage u_j upon the capacitor C at the input of the j^{th} neuron is

$$\frac{du_j}{dt} = \frac{1}{C}\left[\sum_{i=0}^{N-1}\{v_i(t) - u_j(t)\}G_{ij} - \frac{u_j(t)}{R_j} + I_j\right] \qquad (1)$$

where v_i is the output voltage of neuron i, G_{ij} is the conductance value or *weight* connecting the output of neuron i to the input of neuron j, R_j is a resistor which bleeds charge from C, and I_j is a bias current for neuron j. The nonlinear transfer function for a neuron j is *sigmoidal*, $v_j = [1 + \exp(-Au_j)]^{-1}$, where A is the *gain* of the neuron. Without loss of generality, C can be set to unity and the bias currents can all be replaced by a single bias neuron N with $v_N = 1$ that is connected to each neuron j through a conductance value of $G_{Nj} = I_j$. This gives

$$\frac{du_j}{dt} = \sum_{i=0}^{N} v_i(t)G_{ij} - u_j(t)g_j \tag{2}$$

where $g_j = \sum_{i=0}^{N-1} G_{ij} + 1/R_j$. Assuming $G_{ij} = G_{ji}$ and integrating the equations of motion in (2), it can be seen that the Hopfield neurons are performing gradient descent on an *energy surface* of the form

$$E = \underbrace{-\frac{1}{2}\sum_{i=0}^{N}\sum_{j=0}^{N-1} v_i G_{ij} v_j}_{combinatoric} + \underbrace{\frac{1}{A}\left[\sum_{j=0}^{N-1} g_j\{(v_j - 1)\ln(1 - v_j) - v_j \ln v_j\}\right]}_{monadic}. \tag{3}$$

E contains a *combinatoric* portion whose minima correspond to solutions of a complex problem involving many interacting 1–0 integer variables. E also possesses a *monadic* portion (due to the presence of the $u_j g_j$ term in (2)) which lacks such interactions. If $A \to \infty$ or $g_j \to 0$, the monadic term dies away and the combinatoric term dominates. As Hopfield explained in [13], increasing the gain in a neural network is similar to decreasing the temperature in the simulated annealing algorithm [14]. This similarity was used to advantage in [3, 7, 8] to improve the quality of solutions found by Hopfield networks.

The large number of calculations required to evaluate the coupled differential equations in (2) will require that special purpose machines be built before many practical applications can be solved in a reasonable amount of time using neural algorithms. The massive parallelism and simple circuit elements of Hopfield networks makes them attractive for analog VLSI implementations, but the large number of interconnections has limited the size of a neural system to the number of neurons that can be placed on a chip [11]. Multiple chip analog implementations will be plagued by parasitics and noise. Digital implementations require more area for each neuron in order to house parallel multipliers or digital memory and associated logic for storing and evaluating interconnection weights [2, 16, 17, 18].

In the remainder of this paper, we will describe the design, implementation, and performance of STONN [5, 6, 4, 9], a VLSI neural network which uses stochastic processing elements to solve optimization problems in a manner similar to simulated annealing.

2 Stochastic Theory of Operation

The basic stochastic computing element of STONN is the AND gate [10] used as a multiplier to form the inner product of conductance values and amplifier output voltages (see Figure 1). The multiplication requires the multiplicands

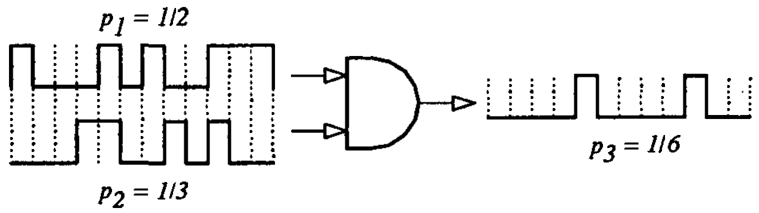


Figure 1: An example of stochastic multiplication.

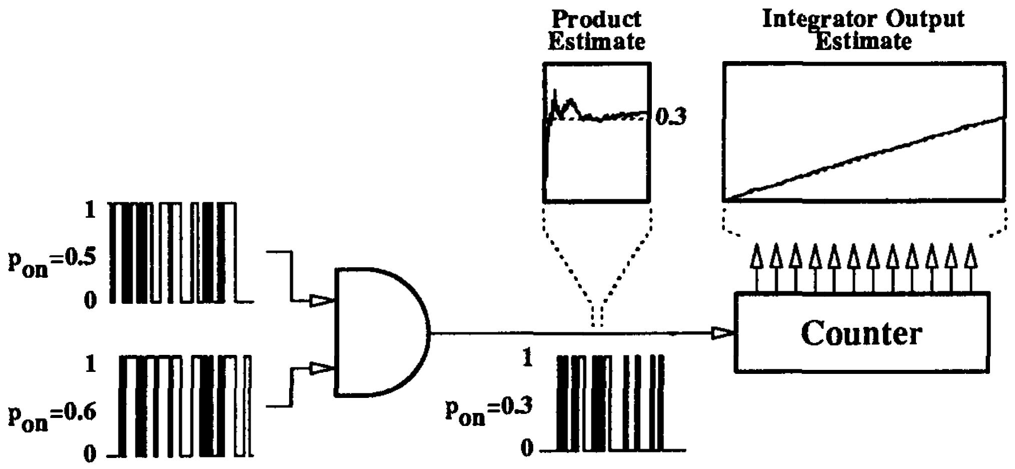


Figure 2: An example of stochastic integration.

to be converted to single-bit, uncorrelated, probabilistic signals that are input to the AND gate. The mean value of the AND gate output will then be the product of the input probabilities. As shown in Figure 2, this method of multiplication does produce high frequency noise in the result. However, it has been shown in [2] and in our own research that noise injected into the network improves solution quality by helping to escape local minima.

In STONN, the network weights and neuron u_j values are stored in digital shift registers which are compared against random variables to produce the single-bit stochastic signals. These signals are then ANDed to produce another stochastic signal representing the charging current for each interconnection. The integration performed by the Hopfield neuron input capacitor is a summation of these charging currents. This summation operation can be implemented with a digital counter which is incremented by positive current pulses and decremented by negative current pulses. (The sign of the current is implemented with a sign bit associated with each conductance value which is directly connected to the counter.) Since the capacitor (or counter) acts as a low pass filter, the high-frequency noise of the stochastic multiplication is reduced. Thus, the counter output is very close to the ideal analog behavior as shown in Figure 2.

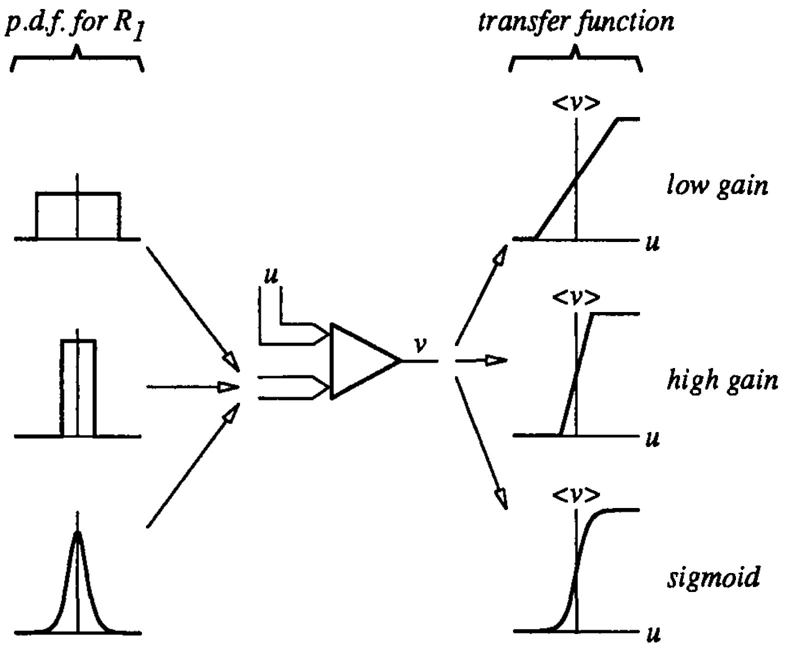


Figure 3: Changing the neural transfer function by altering the PDF.

The digital comparison of a value, u, and a random number will produce a random binary signal with a certain mean value, $\langle v \rangle$. The transfer function relating the input value to this mean value is merely $\langle v \rangle = \int_{-\infty}^{u} p(x)dx$ where p is the probability density function (PDF) of the random number. By changing the PDF, the transfer function can be changed. For example, as shown in Figure 3, the neural gain can be lowered by widening the PDF, while a Gaussian PDF produces a nearly sigmoidal transfer function. This method allows dynamic control of parameters such as the amplifier gain. In an identical manner, it is possible to store the weights as floating point numbers. The lower bits of the random number compared with the mantissa are given a uniform distribution, while the upper bits compared with the exponent are given an exponential distribution. In fact, any monotonic function can be realized with the appropriate PDF. This allows the use of a small number of bits in the weight storage registers without sacrificing precision or dynamic range in the values of the weights. Thus, stochastic processing allows STONN to dynamically alter its neural transfer function and precisely encode its weights without requiring a large silicon area.

3 Stochastic Architecture

In order to implement large, multi-chip neural systems, not only must the primitive computational elements be simple and easy to build, the intercon-

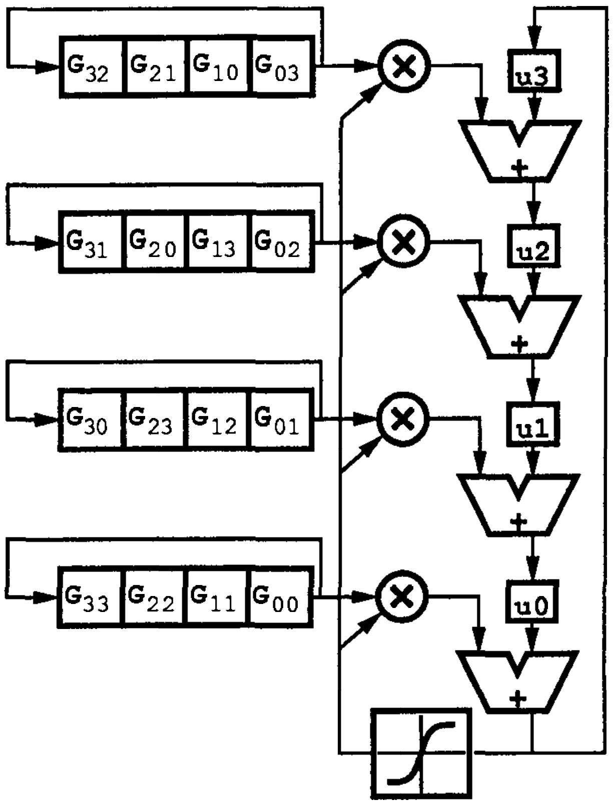
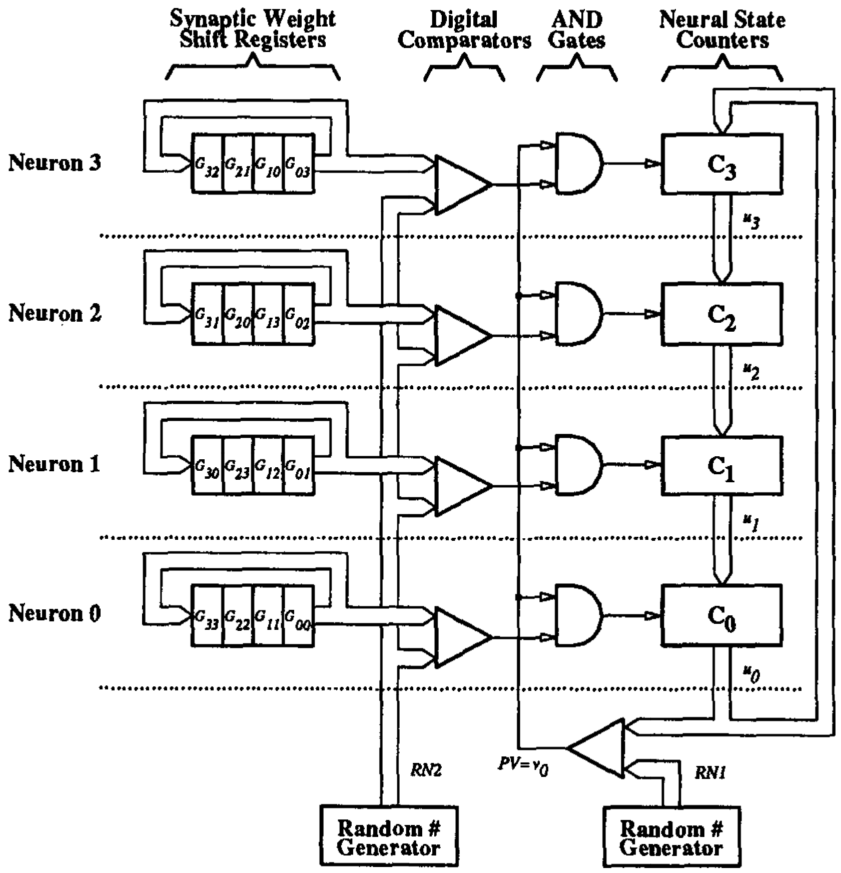

G_{32} G_{21} G_{10} G_{03}
G_{31} G_{20} G_{13} G_{02}
G_{30} G_{23} G_{12} G_{01}
G_{33} G_{22} G_{11} G_{00}
u3
u2
u1
u0
+

Figure 4: Systolic dataflow for the Hopfield network.

nection between the elements must not exceed the practical limits on chip I/O. The STONN architecture was derived from the charging equations and computational data flow of the Hopfield network with an eye toward minimizing communication and control while allowing expandability and pipelining.

The Hopfield network must distribute the outputs from N neurons to N^2 connections to compute the matrix-vector multiplication of (2) in a single cycle. Instead, if during every cycle each neuron accumulates the charging current that results from multiplying the output of a selected neuron by the weights connecting that neuron to every other neuron, then all the required multiplications and accumulations can be performed in N cycles. The data flow for this parallel implementation is shown in Figure 4. The weights and capacitor voltages circulate through shift registers. Note that the weights to each neuron must be distributed across all N memory arrays to align each connection with the proper capacitor voltage as data is cycled. (Similar arrangements were also proposed in [1, 15].) While this method reduces the order of parallelism from $O(N^2)$ to $O(N)$, it makes large networks possible since only the capacitor voltages and some control signals must be routed between chips regardless of the number of neurons simulated.



Synaptic Weight Shift Registers
Digital Comparators
AND Gates
Neural State Counters
Neuron 3
Neuron 2
Neuron 1
Neuron 0
G_{32} G_{21} G_{10} G_{03}
G_{31} G_{20} G_{13} G_{02}
G_{30} G_{23} G_{12} G_{01}
G_{33} G_{22} G_{11} G_{00}
C_3
C_2
C_1
C_0
u_3
u_2
u_1
u_0
RN2
$PV=v_0$
RN1
Random # Generator
Random # Generator

Figure 5: A stochastic implementation for the Hopfield network.

4 Implementation Details

A logical block diagram of the STONN chip is shown in Figure 5. The 7-bit wide synaptic connection weights for each STONN neuron are stored in a dynamic shift register that uses six transistors per shift register cell. There is a multiplexer at the input to each weight shift register that allows two modes of operation. With the multiplexer in the load mode, the input of each shift register is connected to the output of the previous shift register, thus making one long shift register. This permits the weights to be loaded into the chip using only seven input pins and a select line. When performing actual neural computations, the multiplexer is in the cycle mode so that each shift register recirculates its weights. During each clock cycle, a weight is output by each shift register and is compared to a random number, RN2, using a 6-bit comparator. Simultaneously, a signed comparison is performed between the contents of the bottom counter and a signed random number, RN1, to produce the neural firing signal PV. PV is routed to all the neurons and is ANDed with the result of the weight comparisons to produce positive

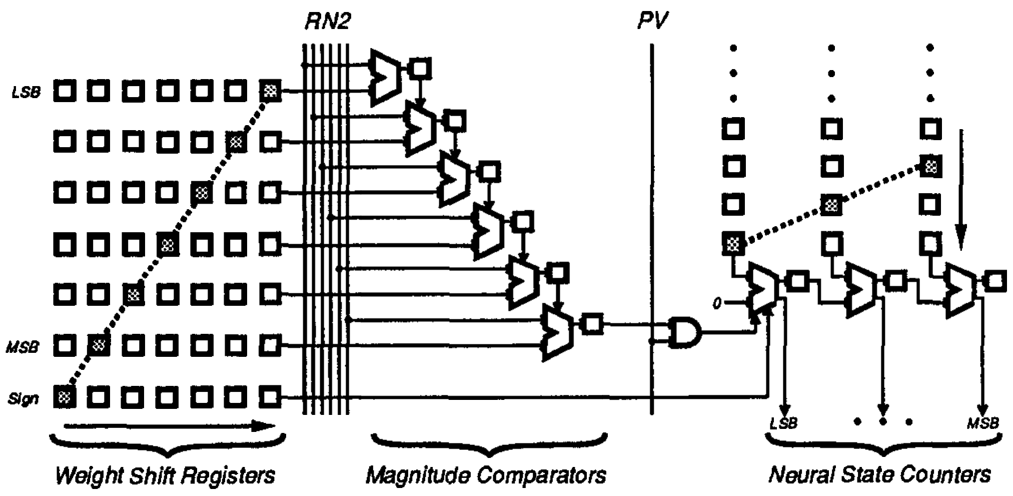


Figure 6: Bit-level pipelining in a single STONN neuron.

or negative neural charging pulses (depending upon the sign of the weight stored in bit 7) which update the counters. Then, the counters and the weight shift registers are all shifted one position and the process is repeated. In this manner, all the charging currents affecting each capacitor voltage are summed over time by the digital counters. Overflow circuitry is built into the counters to set an overflow bit and preserve the sign bit when a counter has reached its positive or negative limit. Convergence to a solution can also be approximately determined by detecting when all overflow bits of the neurons are set.

The regular flow of data in STONN allows the computations to be deeply pipelined down to the bit level. This bit-wise pipelining requires all weight and counter values to be stored in a skewed manner so that more significant bits reach a computation after the less significant bits have been processed. The skewed weights and counters with their associated pipelined comparison and incrementing operations are shown in Figure 6. As can be seen, the comparison result is available six clocks after the shift register outputs the least significant bit of the weight, and the last bit of the counter value is updated after another 10 cycles (the width of the counter). Due to the use of carry-save arithmetic, each comparator is simultaneously handling computations on different bits of 6 different weights, while each counter is concurrently updating 10 capacitor voltages. Thus, no clock cycles are wasted and the duration of a clock pulse is reduced to the time needed to perform a one-bit addition.

In order to save silicon area, counters which could only add 0, 1, or 2 to their contents were alternated with counters that could only perform subtractions of 0, 1, or 2. Such pairs of counters can perform the same computations as pairs of more complex increment/decrement counters. The

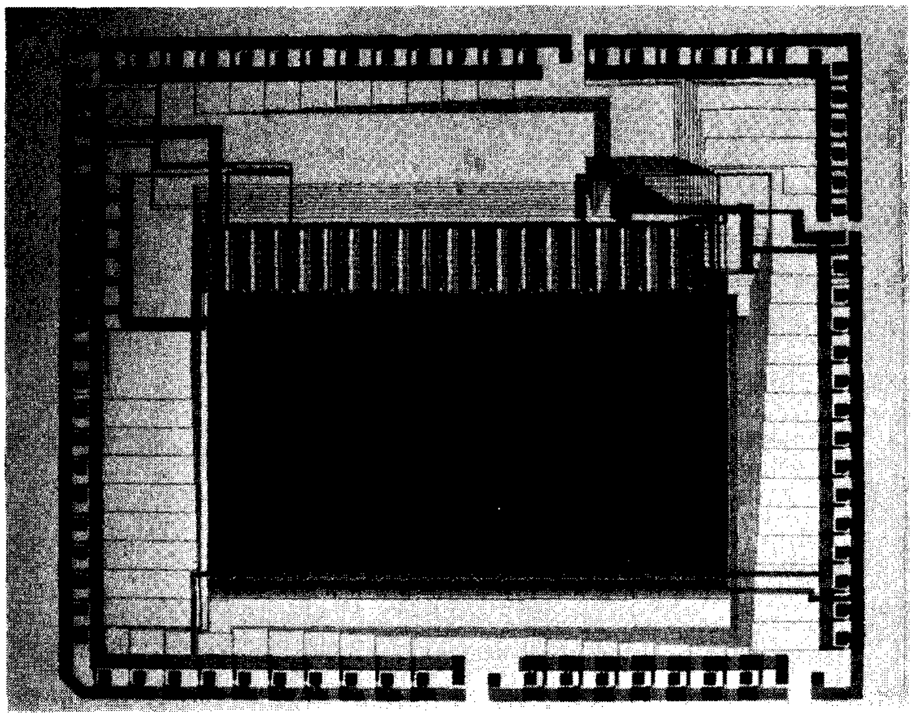


Figure 7: The layout for the STONN chip.

setting and resetting of the overflow bit by the overflow protection circuitry is also simplified since a given counter can now only overflow in one direction instead of both.

In addition to the shift registers for storing the interconnection weights, another shift register is provided which stores the g_j decay weights for each of the neurons. By multiplying stochastic signals proportional to g_j and u_j, the capacitor counter associated with neuron j can be discharged toward zero. As was shown, this decay current gives rise to the simulated annealing properties of the Hopfield network. Annealing can be accomplished in STONN by 1) reducing the values stored in the g_j shift register, 2) decreasing the range of $R1$ (thus increasing the neural gain), or 3) reducing the probability that a decay pulse will reach the counter. The third option is by far the easiest to implement since it requires only the addition of a third stochastic signal, anneal, to the AND gate which multiplies g_j and u_j. Annealing can then be performed as the neural state evolves by decreasing the probability that anneal is high. As will be shown, annealing greatly affects the quality of solutions found by STONN.

The current CMOS version of STONN contains 100,000 transistors organized into 20 neurons (see Figure 7), each of which can receive 100 inputs from other neurons (i.e. the weight shift register can store 100 weights). This allows a completely-connected set of 100 neurons to be constructed

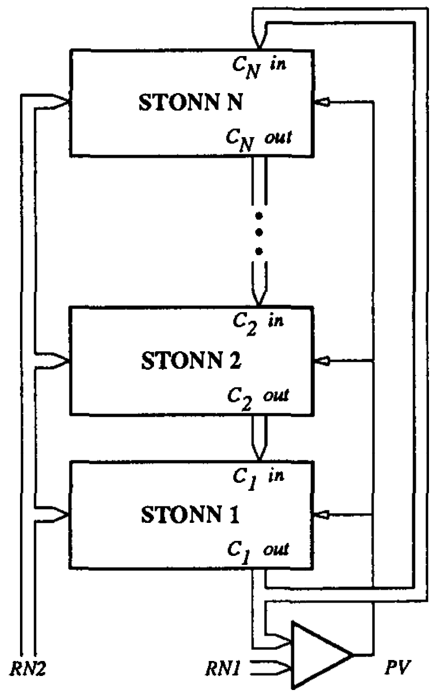


Figure 8: Building larger networks by interconnecting STONN chips.

using five cascaded STONN chips. Cascading chips requires that only the capacitor voltages, PV, and RN2 need be passed between adjacent chips (Figure 8). This reduces the pin count and is the key to building large networks.

5 STONN Advantages

The use of stochastic arithmetic techniques greatly reduces the complexity of logic needed to perform the neural operations as compared to other digital implementations and begins to approach the computational density of analog techniques. Also, the stochastic computing elements naturally accommodate the use of annealing techniques in Hopfield networks. By changing the PDFs of the random numbers input to the STONN chips, neural transfer functions can be easily altered and weights can be stored with high precision in registers containing very few bits.

By accepting $O(N)$ parallelism rather than $O(N^2)$, we have gained the ability to interconnect multiple chips to create networks larger than can be contained on a single chip. Other fully parallel implementations [2, 16, 18] are limited to a small number of neurons since the number of connections needed between two or more chips to maintain complete parallelism far exceeds the available I/O. The need to evaluate each connection during every cycle also increases the complexity and size of the circuitry needed to store the weights. The majority of STONN, however, is composed of dense shift register cells since only one weight per neuron is processed each cycle.

With predicted speeds of the prototype STONN chips between 10Mhz and 25Mhz, the bit-level pipelining and $O(N)$ parallelism of STONN makes it several orders of magnitude faster than the fastest sequential processors. While STONN cannot solve problems faster than fully analog neural networks, our use of digital technology increases the system programmability and noise immunity while easing the difficulties of interfacing to host computers.

6 Experimental Results

STONN was applied to a combinatorial problem involving the partitioning a graph containing N nodes across two bins, $\{b_0, b_1\}$. An optimal solution for this bipartitioning problem minimizes the number of edges crossing between the bins while balancing the number of nodes in each bin. This is expressed by 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 node n_i resides in bin b_0 and $v_i = 1$ otherwise. Each edge in the graph tends to attract the adjacent nodes n_i and n_j into the same bin with a force proportional to E_{ij}. Counterbalancing this attraction is r, an amorphous repulsive force between all of the nodes which discourages them from clustering together.

For a 50 node graph bipartitioning example, the performance of STONN was compared to directly solving the differential equations of (1) or finding an optimal solution using simulated annealing. The average number of crossing edges found using each method are tabulated in Table 1. During 1000 trials on a DECstation 3100, simulated annealing often found the optimal solution having 15 crossing edges. Solving the differential equations using predictor-corrector methods gave a solution having 20 crossing edges. (Since the neural network formulation enforces bin balance with a soft constraint, a solution in which the bin contents differed by less than 5 nodes was considered balanced.) One hundred STONN simulations were run with 4 different operational settings. In the first 25 trials, STONN was simulated with $g_j = 0$ (i.e. the high gain case) and a linear transfer function which saturated at $v = 0$ and $v = 1$. In the next 25 trials, capacitor decay was enabled by setting g_j to a small value. This removed an average of 1 crossing edge from the solutions. For the next 25 trials, g_j was returned to zero and the neural transfer function of Figure 9 was used. The differential gain for this nonlinear function is small when $u \approx 0$ and grows as a neuron begins

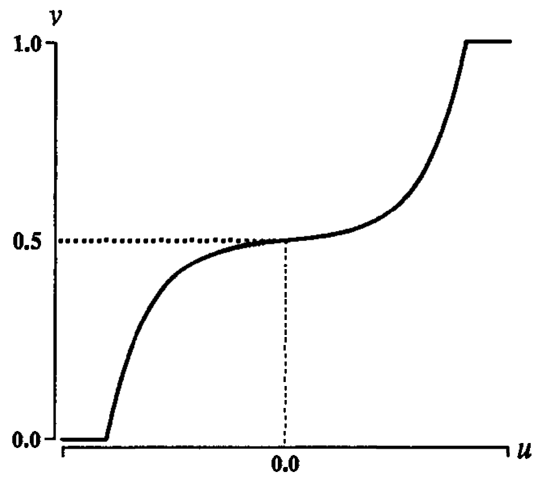


Figure 9: A neural transfer function with variable differential gain.

Simulated Annealing	15.6 ± 0.712
Predictor-corrector	20.0
STONN $g_j = 0$, linear	20.2 ± 2.3
STONN $g_j > 0$, linear	19.4 ± 2.9
STONN $g_j = 0$, nonlinear	19.4 ± 1.9
STONN $g_j > 0$, nonlinear	18.6 ± 2.2

Table 1: A comparison of the number of crossing edges in a bipartitioning found using simulated annealing, predictor-corrector techniques, and STONN.

to saturate. This transfer function lead to approximately the same solution quality as annealing with g_j. For the last 25 trials, both annealing and the new transfer function were enabled, leading to the best solution quality found using STONN.

For execution times, simulated annealing required 1000 sweeps through the nodes to converge to a solution, consuming $\approx$ 2s on average on a DECstation3100. The predictor-corrector technique required 2000 sweeps through the 50 neurons, and evaluating each neuron required 700 floating-point operations. Thus, 70,000,000 floating-point operations were performed. The STONN neurons usually saturated within 4,000 sweeps through the 50 neurons. Assuming a clock speed of 10 MHz, convergence to a solution occurs within 0.02s. In order to reach the same speed, predictor-corrector techniques would require a computational engine capable of 3.5 GFLOPS.

7 Conclusion

A neural network VLSI implementation based on stochastic processing and systolic dataflow has been described, and the performance was reported. STONN offers many advantages, the most important being the ability to build multiple chip systems using a constant number of interconnections. The regular flow of data enables us to increase the speed of a STONN system by adding more chips without incurring any additional communication overhead. Building with stochastic computing elements increases the computational density of STONN and permits the use of annealing techniques. The completely digital design provides dynamically reprogrammable parameters. At a predicted speed of 10 MHz, a 50-neuron prototype STONN system can solve a combinatorial optimization problem in 20 ms. The design can be easily modified using state-of-the-art VLSI technology to greatly improve speed and density.

Acknowledgements

This research was supported in part by DARPA contract MDA972-88-C-0047, DARPA Grant #N0001489J1441, the Neural Engineering Group, the Center for Communications and Signal Processing and the Computer Systems Laboratory at North Carolina State University.

References

[1] A. Agranat and A. Yariv. A new architecture for a microelectronic implementation of neural network models. In *IEEE First International Conference on Neural Networks*, volume III, pages 403–410, 1987.

[2] J. Alspector, B. Gupta, and R. Allen. Performance of a stochastic learning microchip. In *Advances in Neural Information Processing Systems I*, pages 748–760, 1989.

[3] G. Bilbro, R. Mann, T. Miller III, W. Snyder, D. E. Van den Bout, and M. White. Mean field annealing and neurai networks. In *Advances in Neural Information Processing Systems*, pages 91–98. Morgan-Kaufmann Publishers, Inc., 1989.

[4] D. E. Van den Bout, P. Franzon, J. Paulos, T. K. Miller III, W. Snyder, T. Nagle, and W. Liu. Scalable VLSI implementations for neural networks. *The Journal of VLSI Signal Processing*, December 1989.

[5] D. E. Van den Bout and T. K. Miller III. A stochastic architecture for neural nets. In *Proceedings of the IEEE International Conference on Neural Networks*, pages I:481–I:488, 1988.

[6] D. E. Van den Bout and T. K. Miller III. A digital architecture employing stochasticism for the simulation of Hopfield neural nets. *IEEE Transactions on Circuits and Systems*, 36(5):732–738, May 1989.

[7] D. E. Van den Bout and T. K. Miller III. Graph partitioning using annealed neural networks. In *Proceedings of the IEEE International Conference on Neural Networks*, pages I:521–I:528, 1989.

[8] D. E. Van den Bout and T. K. Miller III. Improving the performance of the hopfield-tank neural network through normalization and annealing. Accepted for publication in *Biological Cybernetics.*, 1989.

[9] D. E. Van den Bout and T. K. Miller III. Stochastic architectures for recurrent neural networks. Invited chapter in *Neural Networks: Concepts, Applications, and Implementations* (Prentice-Hall)., 1990.

[10] B. Gaines. Stochastic computing systems. In *Advances in Information and Systems Science*, pages 37–172. Elsevier, 1969.

[11] 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 *American Institute of Physics Conference Proceedings No. 151*, pages 414–419, 1986.

[12] J. Hopfield and D. Tank. Simple "neural" optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, 33(5):533–541, May 1986.

[13] J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. *Biol. Cybern.*, 52:141–152, 1985.

[14] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 13 1983.

[15] S. Kung and J. Hwang. Parallel architectures for artificial neural nets. In *Proceedings of the 1988 IEEE International Conference on Neural Networks*, pages II:165–II:172, 1988.

[16] A. Murray and A. Smith. Asynchronous VLSI neural networks using pulse-stream arithmetic. *IEEE Journal of Solid-State Circuits*, 23(3):688–697, June 1988.

[17] P. Penz and R. Wiggins. Digital signal processor accelerators for neural network simulations. In *American Institute of Physics Conference Proceedings No. 151*, pages 345–355, 1986.

[18] M. Sivilotti, M. Emerling, and C. Mead. A novel associative memory implemented using collective computation. In *1985 Chapel Hill Conference on Very Large Scale Integration*, pages 329–342, 1985.