# Arithmetic for Digital Neural Networks

D. Zhang, G. A. Jullien, and W. C. Miller
VLSI Research Group
Department of Electrical Engineering
University of Windsor
Windsor, Ontario, CANADA N9B 3P4

Earl Swartzlander, Jr.
Dept. of Electrical and Computer Engineering
University of Texas
Austin, TX 78712
USA

## Abstract

*This paper describes the implementation of large input digital neurons using designs based on parallel counters. The implementation of the design uses a two-cell library, in which each cell is implemented using "Switching Trees" which are pipelined binary trees of n-channel transistors. Results obtained from initial switching trees realized with a 3-μm CMOS process indicate that the design is capable of being pipelined at 40 MHz sample rates with better performance expected for more advanced technologies. It appears feasible to develop a wafer scale implementation with 2000 neurons (each with 1000 inputs) that would perform over 3 x $10^{12}$ additions/second.*

## 1 Introduction

Over the past several years, neural networks have become a subject of great interest due to their massive parallelism, modularity, and robustness. Current research in the area of neural networks embraces modeling, algorithms, architectures, and implementations [1], [2].

There have been a variety of implementation approaches for neural networks, including software and hardware [3], electronic and optical implementations [4], VLSI [5], and discrete component realizations. The real promise for applications of neural networks currently lies in specialized hardware, in particular VLSI or WSI implementations that achieve high levels of computational performance with modest development effort [6].

The high interconnectivity and relatively low signal precision requirements of neural networks appear well tailored to analog realizations. With the increasing size of neural networks, however, it is becoming increasingly difficult to achieve acceptable matching of gains and thresholds on the analog integrated circuits. On the other hand digital realizations, which superficially appear to be inferior in terms of computational density, have advantages of flexibility in terms of programming for a variety of architectures and learning strategies, as well as simplifying the memory function. Currently, there is a strong development effort, to implement neural networks in VLSI using digital technology [7]-[9].

In digital implementations the requirements for parallel digital arithmetic computation are severe. As an example, for a fully connected layered network with two layers and n neurons per layer, each neuron is required to form an inner product of n elements using 1-bit binary input lines and m-bit weights (e.g., m=16). Thus, a key problem for neuron design is to build an efficient multi-input gated adder structure suitable for m-bit inputs. An interesting approach is to use a parallel counter [10]-[12], and this paper presents such an implementation, using a new pipelined dynamic CMOS logic family based on *Switching Trees*.

## 2 Neural Networks Implemented with Parallel Counters

A three input neural network is shown in Figure 1. The k-th digital neuron on the i-th layer of the n neuron/layer neural network evaluates the following inner product:

$$Y_{i+1,k} = F_{nl}\left(W_{i,n,k} + \sum_{j=0}^{n-1} Y_{i,j}\, W_{i,j,k}\right)$$

Where: $Y_{i,j}$ (for j=0,1,...,n-1) is the input vector and $W_{i,j,k}$ is the weight vector, and $F_{nl}$ is a non-linear function. When the input vector consists of one bit binary components, the sum of products reduces to a multioperand addition. A quasi serial [11], [12] realization of a 1000 input neuron is shown on Figure 2. The weights, $W_{i,j,k}$, are stored in recirculating shift registers that are accessed beginning with the LSB. A 1010 input parallel counter sums the 1000 data inputs, $W_{i,j,k}$ (for j=0, 1,...,999), the weight that sets the neuron's threshold $W_{i,1000,k}$, and the nine carry bits that arise in the counting process. A process was given in [10] for the implementation of 2n + 1 input parallel counters with

CH3015-5/91/0000/0058$01.00 © 1991 IEEE

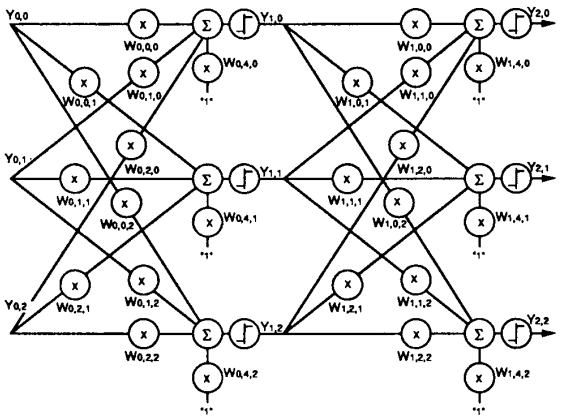


**Figure 1. Example of a Two Layer Fully Connected Neural Network.**

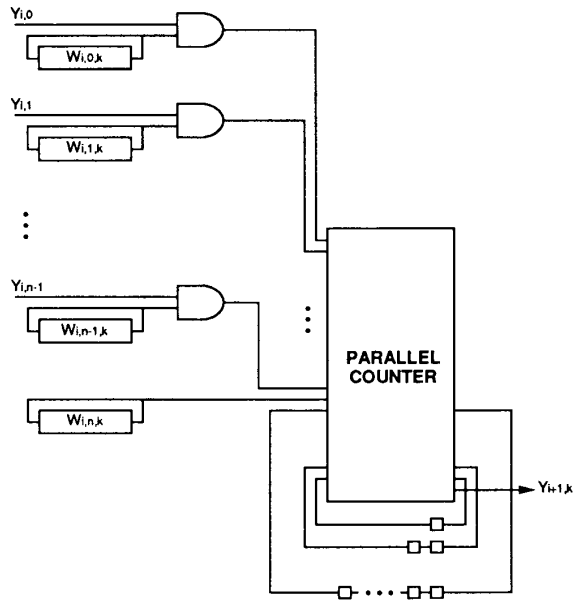



**Figure 2. Artificial Neuron Implementation.**

two n input parallel counters. This process can be applied recursively resulting in an implementation that uses only full adders (which are in fact three input parallel counters). A 1010 input parallel counter constructed with the recursive procedure uses 1000 full adders and requires nine adder delays to generate the LSB of the count and 17 adder delays to generate the MSB of the count. In succeeding sections of this paper, switching trees are used to construct a significantly faster counter that is comparable in complexity.

## 3 Switching Tree Implementation

Implementation of the parallel counter is based on the application of switching trees [13]. An unminimized switching tree is a binary decision tree, directly implemented as an n-channel MOSFET logic block in a complex CMOS pipelined dynamic gate circuit. We denote the tree as a graph, G={X, V, L}, where X is the edge set (NMOS transistors), V is the vertex set (interconnecting nodes), and L is the link set (shorting links) of G. Let $\{x_i, x'_i \mid i=0, 1, ..., n-1\} \in X$, where $x_i$ and $x'_i$ are defined as two different directions, ╲(n-channel transistor driven by the logical complement) and ╱(n-channel transistor driven by the logical true), respectively. Let $\{v_i \mid i=0, 1, ..., n\} \in V$, where $v_0$ is a root and $v_n$ is a leaf. Let $\{v_i \cdots\cdots v_j \mid i,j = 0, 1, ..., n\} \in L$, where a link, $v_i \cdots\cdots v_j$, is denoted as a dotted line from $v_i$ to $v_j$.

A path, P, is a connection from the root, $v_0$, to a leaf, $v_n$, constructed by the edges and the links. A 1-path corresponds to programmed bits in the truth table of the logic function. In a switching tree, all true paths are 1-path and complement paths do not terminate at the bottom edge. A height of the tree is the number of edges of the longest path (only including the maximum edges). A link, $v_i \cdots\cdots v_j$, is an connection between $v_i$ and $v_j$ if these vertices are equivalent. Note that all leaves can be linked.

Given the truth table of a binary function, a switching tree can be constructed, and by applying simple graph theory rules, a minimized tree can be obtained. It is interesting that the same concept was introduced by Shannon [14] for implementing relay logic over 50 years ago. The current minimization procedures and optimization constraints are, however, somewhat different. The switching tree can be directly implemented by a form of pipelined dynamic logic. By using such a direct mapping procedure, the design is optimized for specific silicon cost functions such as area and time. A direct constraint is the height of the switching tree. This height is optimized by balancing the conflicting requirements of area minimization, charge sharing, noise immunity, pipeline cycle time and pipeline latency against the efficiencies inherent in the decomposition of the problem at hand. The designs presented in this paper are based on a switching tree height of seven transistors; this number yields immediate efficiencies in the parallel counter decomposition, with a respectable 25 ns pipeline period for a 3-μm CMOS process.

The pipeline structure used here is based on work by Yuen and Svennson [15], in true single phase (TSP) clocking strategies. The implementation shown here only uses n-

channel transistor logic networks (the switching tree). The p-channel output stage is simply an inverter function instead of a p-channel tree. Although the TSP clock strategy was originally employed to obtain clocking periods of only a few ns, the present implementation trades such high clock rates for increased complexity of switching logic at each evaluation node, and hence a much smaller number of series nodes to complete a given logic function. The latency of these systems will typically be lower than that of applying the TSP clocking to minimum complexity evaluating nodes (e.g., 2-input gates), even though the clock rate is slower. From simulation experiments, it also has been determined that the pipelining of complex single evaluation nodes can be performed at higher throughput rates than by using Domino logic evaluation stages between pipeline latches. These design decisions are based on simulation and a large amount of silicon verification performed over the past several years with the 3-μm CMOS process. The simulations are based on mask extracted data with semi-empirical (level-3) SPICE models tuned from fabrication tests of individual transistors. A more aggressive (i.e., sub-micron CMOS) technology may be expected to improve the performance significantly.

## 4 Counter Cell Design

A single stage dynamic logic cell is defined as a building element for switching tree implementations. Each node of the logic is pipelined; a switching tree structure is used as the complex path to ground for that node. Given the 7-transistor height for the switching tree, two cells serve as the basis for the large parallel counter design.

### 4.1 Seven Input Parallel Counter Cell

A seven input parallel counter with three outputs is the primary building block for large counter implementation. Its switching tree structure and the corresponding domino logic cell are shown in Figure 3. The inputs are $x_0$, $x_1$, $x_2$, $x_3$, $x_4$, $x_5$, and $x_6$, and the outputs are $s_2$ (MSB), $s_1$, and $s_0$ (LSB). This cell is used to perform most of the reduction in the large parallel counter. It is attractive because each use of it reduces the number of bits in the bit matrix by four (there are seven inputs and three outputs) whereas each use of a full adder in [10] only reduces the number of bits by one.

### 4.2 Four Bit Word Adder Cell

The switching tree for a two word by four bit adder and its domino logic cell are shown in Figure 4. This cell is used to perform horizontal reduction in the counter. The cell receives two four bit words X (= $x_3$, $x_2$, $x_1$, $x_0$) and Y (= $y_3$, $y_2$, $y_1$, $y_0$) and a carry $c_0$ as inputs and produces a five bit output word, $c_4$, (MSB) $s_3$, $s_2$, $s_1$, $s_0$ (LSB).

## 5 Parallel Counter Structure

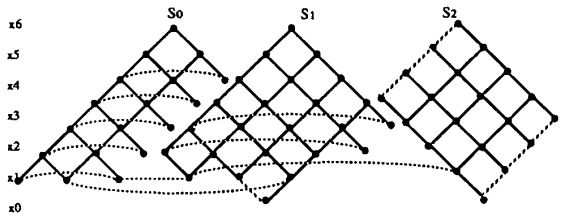
Based on the seven input counter and the four bit word adder cells, a large parallel counter design using switching trees is developed in this section. A two stage reduction process is employed: first the seven input parallel counter cells are used to reduce the initial 1010 row matrix to a matrix with no more than three elements in each column (i.e., no more than three rows where the second and third rows may be sparse). Then the four bit word adder cells are used to complete the reduction to a single row.

### 5.1 Initial Reduction

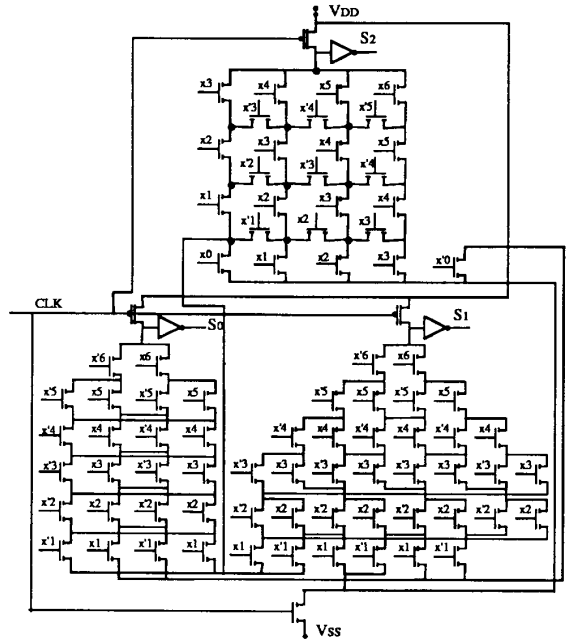
The design process for the large parallel counter is similar to that used by Dadda to develop high speed parallel multipliers [16], [17]. Specifically since the seven input counter has three outputs, the following sequence is used to set the maximum height of the columns in the bit matrices: 3, 7, 15, 35, 79, 183, 427, etc. Each entry is determined by multiplying seven times the integer quotient of the previous entry divided by three and adding any remainder from the division. Thus the third entry was determined as $7 \lfloor 7/3 \rfloor + 1 = 15$. Instead of strictly following Dadda's method which involves reducing the height of each column to no more than the appropriate value from the sequence, greater levels of reduction were performed in the early stages of the reduction. This minimizes the number of latches required to pipeline the counter.

The initial reduction is illustrated by the spread sheet of Figure 5. Each **Bold** entry indicates the number of one bit data located in that column of the matrix. For example, the input matrix (MATRIX 0) has 1010 rows in column 1. Immediately below that entry is the indication that 144 seven input parallel counter cells (indicated as 7:3 counters on Figure 5) are used producing 146 outputs in column 1, and 144 outputs each in columns 2 and 3.

The three sets of outputs from each group of parallel counters are connected with a diagonal line on the diagram. Thus MATRIX 1 has 144 rows in columns 2 and 3 and 146 rows in column 1, etc. Strict application of Dadda's method would have left column 1 with 182 entries and columns 2 and three with 138 entries each. MATRIX 1 for this approach has 434 data whereas Dadda's approach produces a MATRIX 1 with 458 points. Six stages of reduction employing 252 seven input

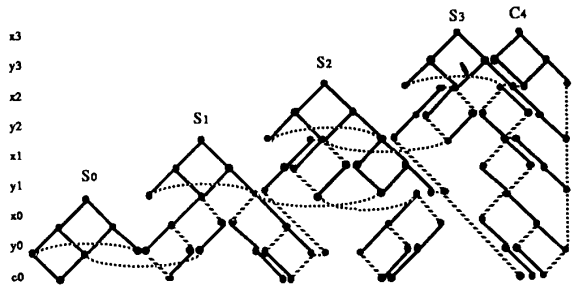



(a) Switching Tree.

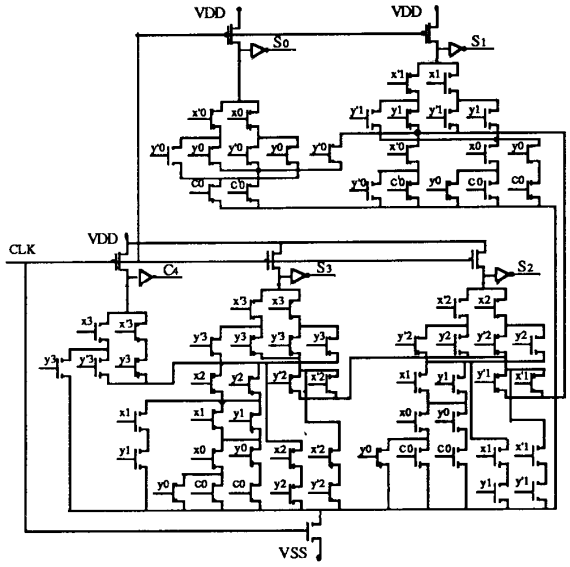



(b) Domino Logic.

Figure 3. Seven Input Parallel Counter.




(a) Switching Tree.




(b) Domino Logic.

Figure 4. Four Bit Word Adder.

parallel counter cells produce MATRIX 6 which is ten columns wide and no more than three rows deep. This output is sufficient to express a count of 2003 which indicates that not all bits can be one simultaneously. This error results from the use of seven input counters in places where there are fewer than seven inputs.

### 5.2 Final Reduction

The final reduction shown in Figure 6 uses four four bit word adders in three steps to reduce MATRIX 6 from Figure 5 to a single eleven bit word (MATRIX 10). At the present time, the design process for this final reduction is a cut and try exercise in geometric reduction.

### 5.3 Counter Performance

This counter is implemented with 252 seven input parallel counters and four four bit word adders. Since each of these cells is three to four times the complexity of a full adder, the switching tree implementation and the full adder implementation (which requires 1000 full adders) are comparable in total complexity. The switching tree cells are comparable in delay to a full adder, so that a non-pipelined implementation of the switching tree counter (which requires ten cell delays) should be nearly twice as fast as the full adder implementation (which requires 17 full adder delays). Pipelined implementations should be able to operate at comparable rates

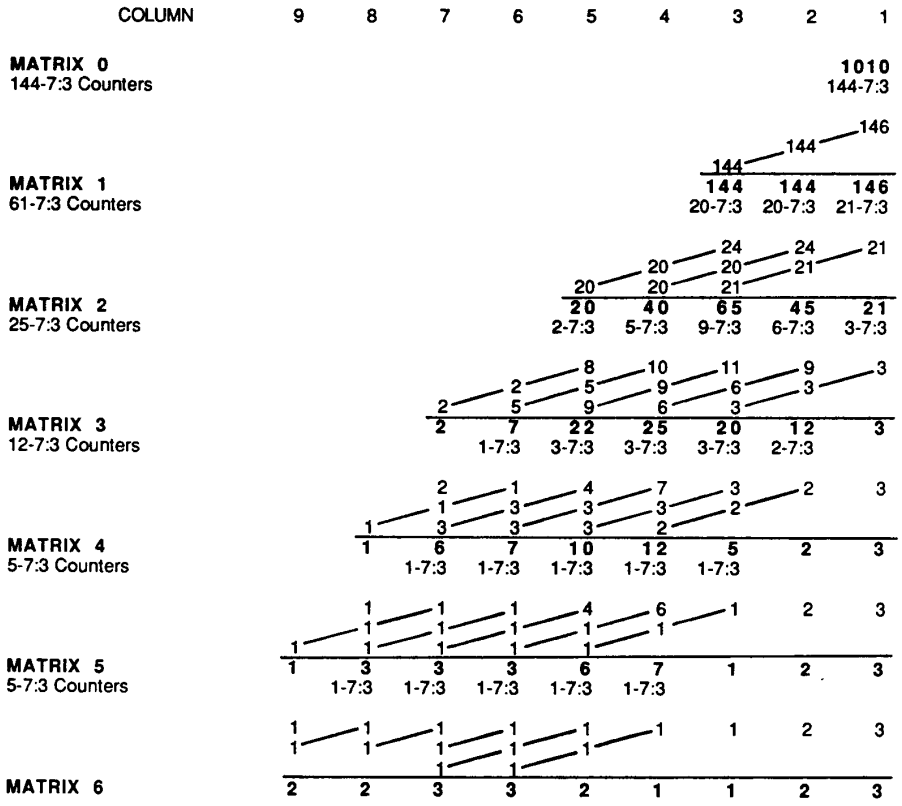



**Figure 5. First Stage in the Bit Matrix Reduction Process.**

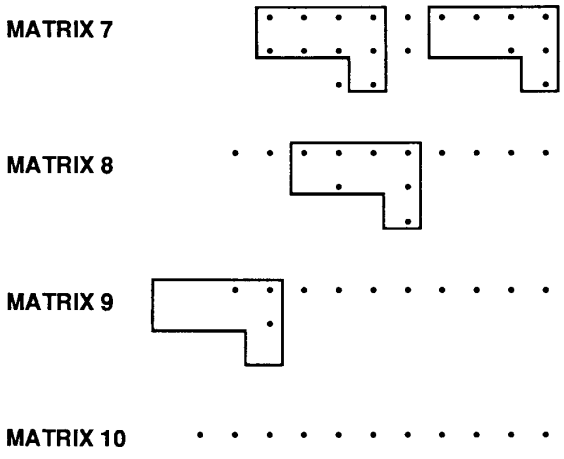



**Figure 6. Second Stage in the Bit Matrix Reduction Process.**

### 5.4 Neural Network Performance

Each cycle of the neural network requires 25 cycles of the parallel counter (16 for the 16 bit weights and nine to assimilate the carries). At a 25 ns clock rate, the time for a neuron cycle is 625 ns. If a wafer scale integration neural network is constructed with 2000 neurons (1000 on each of two layers) as appears feasible [14], the wafer should be able to attain $3 \times 10^9$ neuron cycles/second. Since each neuron cycle involves summing 1000 weighted data, the total computation rate is in excess of $3 \times 10^{12}$ additions/second.

## Conclusions

In this paper, a novel switching tree structure for parallel counter arithmetic of digital neural networks is introduced. Only two types of cells are used in the counters, with a total of 256 cells required (252 seven input parallel

counters and four four bit word adders). Simulation results suggest that the structure can achieve a total computation rate in excess of 3 x $10^{12}$ additions/second.

## Acknowledgements

The first three authors acknowledge support from the Natural Sciences and Engineering Research Council of Canada for funding their portion of this work.

## References

[1] R. Eckmiller and C. V. D. Malsburg, *Neural Computers*, Springer Verlag, 1988.

[2] C. Lau and B. Widrow, eds., "Special Issues on Neural Networks, I and II," *Proceedings of the IEEE* Vol. 78, September and October, 1988.

[3] J. Ghosh and K. Hwang, "Mapping Neural Networks onto Message-Passing Multicomputers," *Journal of Parallel and Distributed Computing*, Vol. 5, 1988.

[4] R. H. Nielsen, "Performance Limits of Optical, Electro-Optical, and Electronic Neurocomputers," *Proceedings of the SPIE*, Vol. 634, pp. 277-306, 1986.

[5] H. P. Graf and P. deVegvar, "A CMOS Implementation of a Neural Network Model," *Proceedings of the Stanford Conference on Advanced Research on VLSI*, MIT Press, pp. 351-367, 1987.

[6] M. A. Sivilotti, M. R. Emerling and C. Mead, "VLSI Architectures for Implementation of Neural Networks," *Proceedings of the AIP Conference*, Vol. 151, P. 408, 1986.

[7] H. P. Graf, L. D. Jackel and W. E. Hubbard, "VLSI Implementation of a Neural Network Model," *Computer*, pp. 41-49, March, 1988.

[8] D. Zhang, G. A. Jullien and W. C. Miller, "Mapping Neural Networks onto Systolic Arrays," *IEEE Transactions on Circuit and Systems*, (in print), 1990.

[9] M. Griffin, *et al.*, "An 11-Million Transistor Neural Network Execution Engine," *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 180, 181, 313, 1991.

[10] E. E. Swartzlander, Jr., "Parallel Counters," *IEEE Transactions on Computers*, Vol. C-22, pp. 1021-1024, 1973.

[11] E. E. Swartzlander, Jr., "The Quasi-Serial Multiplier," *IEEE Transactions on Computers*, Vol. C-22, pp. 317-321, 1973.

[12] E. E. Swartzlander, Jr., B. K. Gilbert, and I. S. Reed "Inner Product Computers," *IEEE Transactions on Computers*, Vol. C-27, pp. 21-31, 1978.

[13] D. Zhang, G. A. Jullien and W. C. Miller, "Switching Tree Structures for VLSI Implementations," submitted to *IEEE Transactions on Computers*, 1990.

[14] C. E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits," *AIEE Transactions*, Vol. 57, pp. 713-723, 1938.

[15] J. Yuan and C. Svennson, "High-Speed CMOS Circuit Technique," *IEEE Journal of Solid-State Circuits*, Vol. 24, pp. 62-71, 1989.

[16] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza*, Vol. 34, pp. 349-356, 1965.

[17] L. Dadda, "On Parallel Digital Multipliers," *Alta Frequenza*, Vol. 45, pp. 574-580, 1976.