Communicated by David S. Touretzky

A Neural Net Associative Memory for Real-Time Applications

Gregory L. Heileman
Department of Computer Engineering,
University of Central Florida, Orlando, FL 32816 USA

George M. Papadourakis
Department of Computer Science,
University of Crete, Iraklion, Crete, Greece

Michael Georgiopoulos
Department of Electrical Engineering,
University of Central Florida, Orlando, FL 32816 USA

A parallel hardware implementation of the associative memory neural network introduced by Hopfield is described. The design utilizes the Geometric Arithmetic Parallel Processor (GAPP), a commercially available single-chip VLSI general-purpose array processor consisting of 72 processing elements. The ability to cascade these chips allows large arrays of processors to be easily constructed and used to implement the Hopfield network. The memory requirements and processing times of such arrays are analyzed based on the number of nodes in the network and the number of exemplar patterns. Compared with other digital implementations, this design yields significant improvements in runtime performance and offers the capability of using large neural network associative memories in real-time applications.

1 Introduction

Data stored in an associative memory are accessed by their contents. This is in contrast to random-access memory (RAM) in which data items are accessed according to their address. The ability to retrieve data by association is a very powerful technique required in many high-volume information processing applications. For example, associative memory has been used to perform real-time radar tracking in an antiballistic missile environment. They have also been proposed for use in database applications, image processing, and computer vision. A major advantage that associative memory offers over RAM is the capability of rapidly retrieving data through the use of parallel search and comparison operations; however, this is achieved at some cost. The ability to search the contents

Neural Computation **2, 107–115** (1990)

of a traditional associative memory in a fully parallel fashion requires the use of a substantial amount of hardware for control logic. Until recently, the high cost of implementing associative processors has mainly limited their use to special purpose military applications (Hwang and Briggs 1984). However, advances in VLSI technology have improved the feasibility of associative memory systems.

The Hopfield neural network has demonstrated its potential as an associative memory (Hopfield 1982). The error correction capabilities of this network are quite powerful in that it is able to retrieve patterns from memory using noisy or partially complete input patterns. Komlós and Paturi (1988), among others, have recently performed an extensive analysis of this behavior as well as the convergence properties and memory capacity of the Hopfield network.

Due to the massive number of nodes and interconnections in large neural networks, real-time systems will require computational facilities capable of exploiting the inherent parallelism of neural network models. Two approaches to the parallel hardware implementation of neural networks have been utilized. The first involves the development of special-purpose hardware designed to specifically implement neural network models or certain classes of neural network models (Alspector *et al.* 1989; Kung and Hwang 1988). Although this approach has been shown to yield tremendous speedups when compared to sequential implementations, the specialized design limits the use of such computers to neural network applications and consequently limits their commercial availability. This is in contrast to the second approach to parallel hardware implementation, general-purpose parallel computers, which are designed to execute a variety of different applications. The fact that these computers are viable for solving a wide range of problems tends to increase their availability while decreasing their cost.

In this paper a direct, parallel, digital implementation of a Hopfield associative memory neural network is presented. The design utilizes the first general-purpose commercially produced array processor chip, the Geometric Arithmetic Parallel Processor (GAPP) developed by the NCR Corporation in conjunction with Martin Marietta Aerospace. Using these low-cost VLSI components, it is possible to build arbitrarily sized Hopfield networks with the capability of operating in real-time.

2 The GAPP Architecture

The GAPP chip is an inexpensive two-dimensional VLSI array processor that has been utilized in such applications as pattern recognition, image processing, and database management. Current versions of the GAPP operate at a 10-MHz clock cycle; however, future versions will utilize a 20-MHz clock cycle (Brown and Tomassi 1989). A single GAPP chip contains a mesh-connected 6 by 12 arrangement of processing elements

(PEs). Each PE contains a bit-serial ALU, 128 × 1 bits of RAM, 4 single-bit latches and is able to communicate with each of its four neighbors. GAPP chips can be cascaded to implement arbitrarily sized arrays of PEs (in multiples of 6 × 12). This capability can be used to eliminate bandwidth limitations inherent in von Neumann machines. For example, a 48 × 48 PE array (32 GAPP chips) can read a 48-bit-wide word every 100 nsec, yielding an effective array bandwidth of 480 Mbits/sec (Davis and Thomas 1988; NCR Corp. 1984).

Information can be shifted into the GAPP chip from any edge. Therefore, the ability to shift external data into large GAPP arrays is limited only by the number of data bus lines available from the host processor. For example, Martin Marietta Aerospace is currently utilizing a 126,720 PE array (1760 GAPP chips) in image processing applications. This system is connected to a Motorola MC68020 host system via a standard 32-bit Multibus (Brown and Tomassi 1989).

3 The Hopfield Neural Network

The Hopfield neural network implemented here utilizes binary input patterns — example inputs are black and white images (where the input elements are pixel values), or ASCII text (where the input patterns are bits in the 8-bit ASCII representation). This network is capable of recalling one of M exemplar patterns when presented with an unknown N element binary input pattern. Typically, the unknown input pattern is one of the M exemplar patterns corrupted with noise (Lippmann 1987).

The recollection process, presented in Figure 1, can be separated into two distinct phases. In the initialization phase, the M exemplar patterns are used to establish the N^2 deterministic connection weights, t_{ij}. In the search phase, an unknown N element input pattern is presented to the N nodes of the network. The node values are then multiplied by the connection weights to produce the new node values. These node values are then considered as the new input and altered again. This process continues to iterate until the input pattern converges.

4 Hopfield Network Implementation on the GAPP

Our design maps each node in the Hopfield network to a single PE on GAPP chips. Thus, an additional GAPP chip must be incorporated into the design for every 72 nodes in the Hopfield network. The ease with which these chips are cascaded allows such an approach to be used. When implementing the Hopfield network, the assumption is made that all M exemplar patterns are known a priori. Therefore, the initialization phase of the recollection process is performed off-line on the host computer. The resulting connection weights are downloaded, in signed magnitude format, to the PEs' local memory as bit planes. The local

Let

M = number of exemplar patterns
N = number of elements in each exemplar pattern
x_i^s = element i of exemplar for pattern $s = \pm 1$
y_i = element i of unknown input pattern = ± 1
$u_i(k)$ = output of node i after k iterations
t_{ij} = interconnection weight from node i to node j

Initialization:

$$t_{ij} = \begin{cases} \sum_{s=1}^{M} x_i^s x_j^s, & \text{if} \quad i \neq j \\ 0, & \text{if} \quad i = j \end{cases}$$

$$u_j(0) = y_i, \quad 1 \leq i \leq N$$

Search:

$$u_j(k+1) = f_h\left[\sum_{i=1}^{N} t_{ij} u_i(k)\right], \quad 1 \leq j \leq N$$

where

$$f_h(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

iterate until $u_j(k+1) = u_j(k), \quad 1 \leq j \leq N$

Figure 1: The recollection process in a Hopfield neural network.

memory of the PEs is used to store the operands of the sum of products operations required in the search phase. The memory organization of a PE (node j) is illustrated in Figure 2. For practical applications, the GAPP memory is insufficient for storing all weights concurrently, thus segmentation is required.

The Hopfield network is implemented in parallel with each PE performing N multiplications and $(N-1)$ additions per iteration. However, in practice no actual multiplications need occur since the node values are either +1 or −1. Therefore, multiplications are implemented by performing an exclusive-OR operation on the node bit plane and the sign bit plane of the weights. The result replaces the weights' sign bit plane. These results are then summed and stored in the GAPP memory. The sign bit plane of the summations represents the new node values.

After an iteration has been completed, the input pattern is tested for convergence utilizing the global OR function of the GAPP chips. If the result of the global OR is 1, another iteration is required; thus, it is necessary to transfer the new node values (i.e., the sign bit of the summation) to the host machine. These node values are then downloaded, along with the connection weights, to the GAPP chips in the manner described previously and another iteration is performed.

5 Memory Requirements and Processing Time

The number of bits required to store each weight value and the summation in the search phase are $w = \lceil \log_2(M+1) \rceil + 1$ and $p = \lceil \log_2(NM+1) \rceil + 1$, respectively, where N is the number of nodes in the network and M represents the number of exemplar patterns. Therefore, each PE in the GAPP array has a total memory requirement of $N(w+1)+p$ (see Fig. 2).

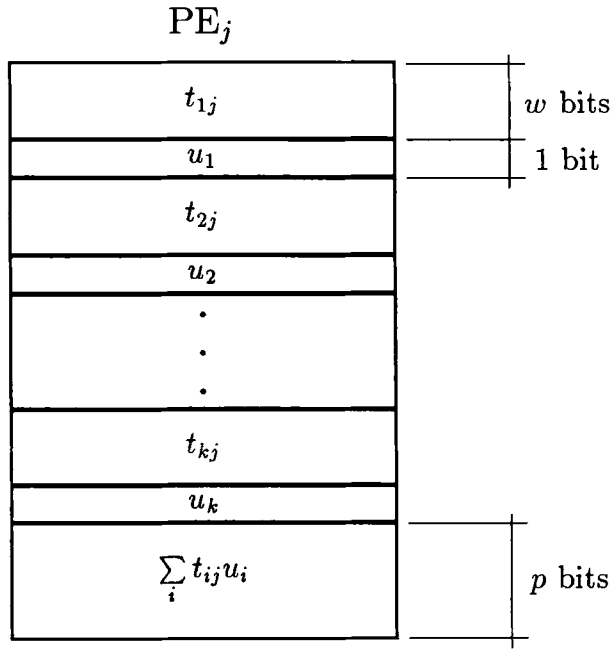

Figure 2: Organization of a single PE's memory in the Hopfield neural network implementation on the GAPP.

If we let B denote the size of a single PE's memory, then each PE has $(B - p)$ bits available for storing weight and node values. If $N(w + 1) > B - p$, there is not enough GAPP memory to store all of the weights at one time, and weights must be shifted into the GAPP memory in segments. The number of weights in each of these segments is given by

$$D = \min\left(\left\lfloor \frac{B-p}{w+1} \right\rfloor, N\right)$$

while the total number of segments is given by

$$S = \left\lceil \frac{N}{D} \right\rceil$$

Letting C represent the number of clock cycles needed to shift a bit plane into GAPP memory, then the number of clock cycles required to download weight and node values to GAPP memory, and to upload new node values to the host is

$$L = [SD(w + 1) + 2]C - 1$$

Furthermore, C depends on the number of data bus lines available from the host and the number of GAPP chips, n. In particular, C can be expressed as

$$C = 12\left\lceil 6n/\#\text{ data lines} \right\rceil + 1$$

The processing time required to implement the search phase of the Hopfield network on the GAPP chips is formulated below. The implementation involves four separate steps. First, the D weights stored in GAPP memory are multiplied by the appropriate node values. As discussed previously, this is performed using an exclusive-OR operation; such an operation requires $3D$ GAPP clock cycles. The second step involves converting the modified weight values into two's complement format; this processing requires $D(4w - 1)$ clock cycles. Next, the D summations required by the search phase are implemented; this can be accomplished in $3Dp$ clock cycles. Finally, 4 clock cycles are required to test for input convergence. The total processing time can now be expressed as

$$P = S[3D + D(4w - 1) + 3Dp + 4] \text{ clock cycles}$$

and the total time required to perform a single iteration of the search phase of the Hopfield network is

$$\begin{aligned} T = L + P \;=\; & SD[C(w + 1) + 3(p + 1) + (4w - 1)] \\ & + 4S + 2C - 1 \text{ clock cycles} \end{aligned}$$

6 Comparisons and Experimental Results

A comparison of the results obtained in the previous section with other digital implementations of the Hopfield network (Na and Glinski 1988) is illustrated in Figure 3. The curve for the DEC PDP–11/70 can be considered a close approximation for the number of clock cycles required by other sequential processing (von Neumann) architectures. Also, the curve for the GAPP PEs assumes the use of a standard 32-bit bus. All of the curves in the figure are plotted with the assumption that $M = \lfloor 0.15N \rfloor$. As more nodes are added, the number of clock cycles required to process the data on the PDP–11/70 and Graph Search Machine (GSM) increases much more rapidly than it does on the GAPP PEs; this can be attributed to the high degree of fine-grained parallelism employed by the GAPP processors when executing the Hopfield algorithm. For example, when implementing a 360 node network, this design requires 7 msec to perform a single iteration.

Extrapolation of the curves in Figure 3 also indicates that for large networks, the ability to implement the network in parallel will easily outstrip any gains achieved by using a faster clock cycle on a sequential processing computer. For example, executing Hopfield networks on the order of 100,000 nodes yields an approximate 132-fold speedup over a sequential implementation. Therefore, a sequential computer with a clock frequency twice as fast as that of the GAPP will still be 66 times slower than the Hopfield network implementation on GAPP processors.

In terms of connections per second (CPS), the 126,720 PE GAPP array discussed earlier can deliver approximately 19 million CPS while running at 10 MHz. The same array running at 20 MHz would yield nearly 38 million CPS, where CPS is defined as the number of multiply-and-accumulate operations that can be performed in a second. In this case, the CPS is determined by dividing the total number of connections by the time required to perform a single iteration of the Hopfield algorithm (the time required to shift in weight values from the host, and the time required to perform the symmetric hard limiting function, f_h, are also included). These results compare favorably to other more costly general-purpose parallel processing computers such as a Connection Machine, CM-2, with 64 thousand processors (13 million CPS), a 10-processor WARP systolic array (17 million CPS), and a 64-processor Butterfly computer (8 million CPS). It should be noted, however, that the CPS measure is dependent on the neural network algorithm being executed. Therefore, in terms of comparison, these figures should be considered only as rough estimates of performance (Darpa study 1988).

To verify the implementation of the Hopfield network presented in Section 4, and the analysis presented in Section 5, a 12×10 node Hopfield network was successfully implemented on a GAPP PC development system using the GAL (GAPP algorithm language) compiler. The exemplar patterns chosen were those used by Lippmann *et al.* (1987) in their

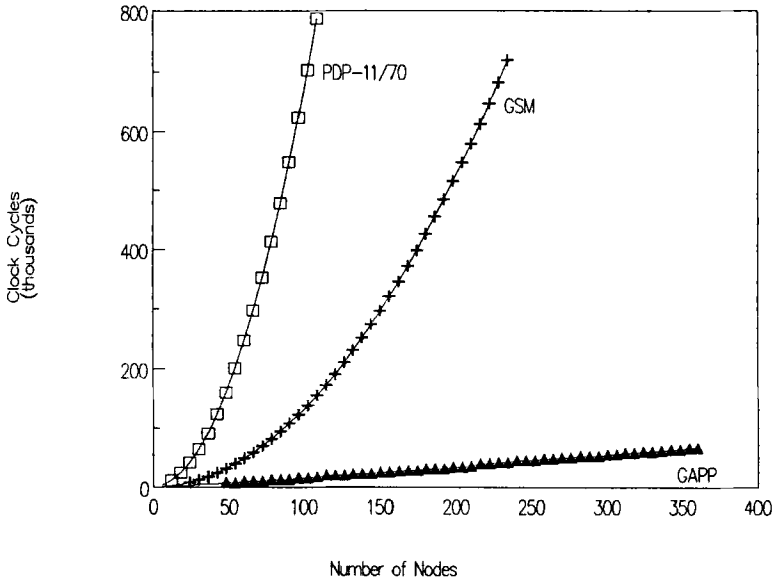

Figure 3: Number of clock cycles required to implement a single iteration of the Hopfield network (search phase) on a PDP–11/70, the Graph Search Machine and GAPP processors. Because of the explosive growth rates of the PDP–11/70 and GSM curves, this graph displays GAPP results for only a relatively small number of nodes. However, the analysis presented here is valid for arbitrarily large networks.

character recognition experiments. The implementation of these experiments in fact corroborated the predicted results.

Acknowledgments

This research was supported by a grant from the Division of Sponsored Research at the University of Central Florida.

References

Alspector, J., Gupta, B., and Allen, R. B. 1989. Performance of a stochastic learning microchip. In *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, ed. Morgan Kaufmann, San Mateo, CA.

Brown, J. R. and Tommasi, M. 1989. Martin Marietta Electronic Systems, Orlando, FL. Personal communication.

Darpa neural network study. 1988. B. Widrow, Study Director. AFCEA International Press.

Davis, R. and Thomas, D. 1988. Systolic array chip matches the pace of high-speed processing. *Electronic Design*, October.

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U.S.A.* **79**, 2554–2558.

Hwang, K. and Briggs, F. 1984. *Computer Architecture and Parallel Processing.* McGraw-Hill, New York.

Komlós, J. and Paturi, R. 1988. Convergence results in an associative memory model. *Neural Networks* **1**, 239–250.

Kung, S. Y. and Hwang, J. N. 1988. Parallel architectures for artificial neural nets. In *Proceedings of the IEEE International Conference on Neural Networks*, Vol. II, San Diego, CA, pp. 165–172.

Lippmann, R. P. 1987. An introduction to computing with neural nets. *IEEE Acoustics Speech Signal Proc. Mag.* **4**(2), 4–22.

Lippmann, R. P., Gold, B., and Malpass, M. L. 1987. *A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification.* Tech. Rep. 769, M.I.T., Lincoln Laboratory, Lexington, MA.

Na, H. and Glinski, S. 1988. Neural net based pattern recognition on the graph search machine. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, New York.

NCR Corp., Dayton, Ohio. 1984. Geometric arithmetic parallel processor (GAPP) data sheet.

Received 5 June 1989; accepted 2 October 1989.