Efficient Implementation of Bidirectional Associative Memories on the Extended Hypercube

J. Mohan Kumar*, and L.M. Patnaik*,**

*MicroprocessorApplications Laboratory

**Department of Computer Science and Automation &

Supercomputer Education and Research Center

Indian Institute of Science

BANGALORE 560 012 INDIA

lalit @vigyan.ernet.in

Abstract

Bidirectional associative memories (BAMs) are being used extensively for solving a variety of problems related to pattern recognition. The simulation of BAMs comprising of large number of neurons involves intensive computation and communication. In this paper we discuss implementation of bidirectional associative memories on various multiprocessor topologies. Our studies reveal that BAMs can be implemented efficiently on the extended hypercube toplogy since the performance of the extended hypercube is better than that of the binary hypercube topology.

Keywords: Bidirectional Associative Memories, Hypercube, Extended Hypercube, Simulation, Total Exchange.

I Introduction

One of the important features of artificial neural networks (ANNs) is the associative storage and retrieval of knowledge [1]. BAM is an example of forward and reverse information flow introduced in neural networks to produce two way associative search [2]. The BAM behaves as a two-layer hierarchy of symmetrically connected neurons. Neurons of one layer recieve weighted inputs from all the neurons of the other layer. Simulation of BAMs comprising of large number of neurons is a compute-intensive task. In this paper we discuss simulation of BAMs on multiprocessor topologies, in particular the extended hypercube topology.

Several reserchers have reported implementation of ANNs on multiprocessor topologies [3], [4], [5]. However, in most of the reports [4], [5] implementation of backpropagation network on multiprocessor systems is discussed. Simulation of BAMs involves extensive communication among the processor elements (PEs) of the multiprocessor system. BAM simulation consists of determination of the weight matrix during the learning phase and computation of outputs in either direction during the recall phase. In a multiprocessor implementation,

CH 3065-0/91/0000-2253 $1.00 © IEEE

multinode broadcast and total exchange problems have to be executed efficiently to facilitate implementation of BAMs. It has been proved that the EH topology performs better than ring, mesh, hypercube and other topologies in executing multinode broadcast and total exchange problems [6]. In this paper we compare the performance of the EH topology in implementing BAMs with that of the hypercube topology.

This paper is organized as follows. We discuss the architecture of the BAM network and implementation aspects of BAMs on multiprocessor topologies in the second section. Section III gives an overview of the EH topology, and discusses efficient execution of multinode broadcast and total exchange problems on the EH. We discuss the architecture of the BAM network, and its implementation on multiprocessor networks in section III. Results of the simulation studies are presented in the fourth section. Fifth section concludes the paper with a summary and comments about the work.

II Bidirectional Associative Memories (BAMs)

Bidirectional associative memory is heteroassociative: that is it accepts an input vector on one set of neurons and produces a related but different output vector on another set of neurons. Bidirectionality, forward and reverse information flow, is introduced in neural networks to produce two-way associative search for stored stimulus-response associations (Ak,Bk) for $0 \leqslant k \leqslant t$. Two layers of neurons, input and output layers, are connected by a pX q synaptic matrix. Fig. 2 shows a BAM network comprising of p input neurons and q output neurons. The neurons of the input and output layer are fully interconnected. BAM adopts unsupervised method of learning and functions in two phases - lerning phase and recall phase. A p X q real matrix M is interpreted as a matrix of synapses between the two layers of neurons. The neurons of both layers operate as in a typical artificial neuron paradigm, producing the sum of the weighted inputs and applying it to the activation function: $bj = f(\Sigma a_i m_{ij})$ for the cells of layer B and $ai = f(\Sigma bjmji)$ for the cells of layer A. For a discrete BAM, each neuron ai or bj is 1 or 0 at any time. Hence, a state A is a point on the boolean p-cube $B^p = \{0,1\}^p$ and a state B is a point in $B^q = \{0,1\}^q$.

During the learning phase, an input pattern A is received by the network, passed through M to get an output B'. Then B' is fedback through M^T to produce A', which produces B" when multiplied with M. This procedure is repeated as follows,

$A \rightarrow M \rightarrow B', B' \rightarrow M^T \rightarrow A'$, A'-> $M \rightarrow$ B", $B" \rightarrow M^T \rightarrow$ A" ..... Af $\rightarrow$ M -> Bf, Bf $\rightarrow$ M $\rightarrow$ Af, till this back-and-forth flow of distributed information equilibrates on a fixed pair (Af,Bf) If an associative memory matrix M equilibrates in this fashion for every input pair (A,B), then M is said to be bidirectionally stable. For t pairs of (A,B), M is given by $M = M_1 + M_2 + \ldots + Mt$ or $A^T{}_1B_1 + A^T{}_2B_2 + \ldots + A^T{}_tB_t$. At the end of the learning phase the associative memory matrix M is computed. During the recall phase, information can be recalled in either direction. A X M produces B and B X M^T produces A.

A.Implementing BAMs onto the EH

The BAM shown in fig. 3 comprises of p input cells and q output cells. In a multiprocessor mapping the cells of the BAM network are mapped onto the Processor Elements (PEs) as shown in fig. 3. For a multiprocessor system comprising of N PEs, p/N input cells and q/N output cells are mapped onto each PE. Each PE processes its cells in a sequential fashion. To compute the output of a cell bj (ai), it is required to consider the weighted inputs of all cells of the input (output) layer. In other words the PE containing cell bj has to input messages from all those PEs which are hosts to the neighbors of bj. A connection weight mij, corresponding to the connection between cells ai and bj is used by the cell bj during forward pass and by the cell a_i during reverse pass. In the following sections, we consider the computation and communication aspects of the learning and recall phases.

Learning Phase

Consider PE_x of fig. 3 which is a host to cells a_i for $(x*[p/N] + 1) \leq i \leq [p/N]*(x+1)$ and PE_y which is host to cells bj for $(y*[q/N]+1) \leq j \leq [q/N]*(y+1)$. During the learning phase it is required to compute the synapse matrix $M = M_1 + M2 + \ldots + M_k + \ldots + M_t$, where, Mk $(1 \leq k \leq t)$ is given by $A_k^T B_k$. In our implementation, the vectors Bk $(b_1 b_2 \ldots b_j \ldots b_q)$ are transmitted to all PEs. Processor element PE_x gets ai for $(x*[p/N]+1) \leq i \leq [p/N]*(x+1)$, The PE_x computes $z^k_{ij} = a^k_i * b^k_j$ for $(x*[p/N] + 1) \leq i \leq [p/N]*(x+1)$, $1 \leq j \leq q$ for all pairs of inputs (Ak, Bk) $1 \leq k \leq t$. Then every PE computes elements $m_{ij} = z_{ij}$ for $(x*[p/N]+1) \leq i \leq [p/N]*(x+1)$, $1 \leq j \leq q$, i.e p/N rows or [pXq]/N elements of the matrix. At the end of the learnig phase every PE transmits unique sets of elements to every other PE in the network : PE_x transmits mijs to PE_y for $(y*[q/N]+1) \leq j \leq [q/N]*(y+1)$, and $(x*[p/N]+1) \leq i \leq [p/N]*(x+1)$. This is a total exchange problem[7]. Hence to implement the learning phase of the BAM algorithm it is required to have a topology which can efficiently execute the total exchange problem.

Recall Phase

During the recall phase, a vector A (or B) is presented at the input (or output) layer of the BAM. In a multiprocessor implementation, PE_x(0 x N-1) receives the vector A $(a_1 a_2 \ldots a_{ip})$ and computes b_j for $(x*[q/N]+1)$ j $[q/N]*(x+1)$ using the expression $b_j = a_i m_{ij}$ (the required mijs are transmitted to PE_x during total exchange execution). Thus each PE computes q/N output states and the multiprocessor system as a whole computes vector B $(b_1 b_2 \ldots b_q)$. Similarly if vector B is input, then ouput vector A is computed by multiplying B with M^T.

III. EXTENDED HYPERCUBE

In this section, a formal introduction to the EH architecture is presented. The EH architecture discussed earlier [6], [7], is suited for hierarchical expansion of multiprocessor systems. The basic module of the EH consists of a k-cube and an additional node for handling communication - the Network Controller (NC). There are a total of $[2^k*(k/2+1)]$ links in the basic module, consisting of the $[(2^k*k)/2]$ links of the hypercube and 2^k links between the individual processor elements and the NC as indicated in fig. 1.

An EH consisting of a k-cube and one NC will be referred to as the basic module or EH(k,1) in the rest of this paper. The EH(k, 1) has two levels of hierarchy : a k-cube at the zeroth level and an NC at the first level. The hypercube consisting of the PEs is referred to as the HC(k,0). An EH(k,2) has 2^k k-cubes of PEs at the zeroth level, a k-cube of NCs at the first level and one NC at the second level. In general, an EH(k,l)(l is the degree of the EH), consists of a k-cube of eight NCs/PEs at the (l-l)st level and one NC at the 1th level. The NCs/PEs at the (l-1)st level of hierarchy form a k-cube. We refer to this cube as HC(k,l-1). The NCs at the (l-2)nd level of hierarchy form 2^k distinct k-cubes which are called HC(k,l-2)s. The HC(k,0)s are all computation HCs and the HC(k,l)s (for $1 > 0$) are all communication HCs. The basic module of the EH denoted by EH(k,1) is a constant pre-definable building block and the node configuration remains the same regardless of the dimension of the EH. The EH architecture can easily be extended by interconnecting appropriate number of basic modules. For example, we can interconnect eight EH(3,l)s (basic modules)to get a 64-node EH - an EH(3,2). The topology formed by the 3-cube of NCs at the first level, and the controller at the second level is identical to that of the basic module. Thus we have a hierarchical structure consisting of 64 PEs at the zeroth level (lowest level), eight NCs at the first level, and one NC at the second level (fig. 1 b). The EH has two types of links, viz., hypercube links which form the k-cubes and the EH links which connect the nodes at jth level to the nodes at (j + l)st level. Further, the path between any two PEs of the EH via one or more NCs is called an extended link.

The nodes of the EH are addressed as described below. The address of an arbitrary node at the zeroth level is written as $D_l D_{l-1}$ ... Do, where D_i $(0 \leqslant i \leqslant 1)$ is a k-bit mod 2^k number, e.g.,in an EH(3,1), D_i is 3-bit mod 8 number. In an EH(k,l) a node at the zeroth level has an (1 + l)-digit address, a node at the first level has an l-digit address .., the solitary node at the 1th (topmost)level has a single digit address. The NC at the topmost level retransmits messages to/from the host system from/to the PEs via the hierarchical network of NCs. In an EH(k,l), message passing between neighboring nodes of the hypercube is via the direct communication links among them whereas communication between non-neighboring nodes of the hypercube is via the NC. Message passing operation in local communication (within a k- cube) involves, l)the source PE, 2)upto (k-1) PEs within the k- cube, and 3) the destination PE, whereas, message passing operation in global communication involves l)the source PE, 2)upto 2(1-1) NCs, and 3) the destination PE. The EH has been found to be efficient in implementing multinode broadcast and total exchange problems [6], [7]. In a multinode broadcast problem every node in the system broadcasts a message to all other nodes. The EH efficiently makes use of the NCs in executing the multinode broadcast problem as discussed by Mohan Kumar et. al [7]. The time spent by each node in executing a multinode broadcast is given as,

$T_{mb}(\text{hypercube}) = C * (N-1);$

$Tmb(EH) = C * (k+1)$, where C is the time for communication between two adjacent nodes. Assuming that all PEs work in parallel, T_{mb} is also the time taken to complete the multinode broadcast in a hypercube, whereas in an EH the time taken to complete the multinode broadcast is given by $C * (2^{\cdot})$. In a total exchange problem each node transmits a separate

packet to every other node. The time spent by each node in executing the total exchange problem is given as,

$T_{te}(hypercube) = C * (N-1) + C \cdot (i-1) * {}^{n}C_i$

$T_{te}(EH) = R + C * (N-l) + Max [C * (i-1) *{}^{k}C_i, \delta]$, where, *6* is the switching delay.

IV Experimental Studies

The BAM network is simulated on a reconfigurable multiprocessor system based on 32-bit transputers. Simulation of BAMs involves the following computations. During the learning phase, it is required to perform a total of t*p*q multiplications and additions - i.e $(t*p*q)/N$ multiplications and additions on each PE. At the end of the learning phase every pair of PEs exchange $(p*q)/N^2$ weghts (mijs). During the recall phase (q*p)/N multiplications and additions are performed on each PE. The results of the experimental studies are given in table 1. We compare the performance of the EH with that of the binary hypercube.

V. Conclusions

In this paper we discuss computational and communication issues related to the implemenation of BAMs on a multiprocessor system. Our studies reveal that BAM implementation on a multiprocessor system involves execution of total exchange problem and it is found that the Extended Hypercube topology performs better than the hypercube topology.

References

1. B. Kosko, " Adaptive Bidirectional Associative Memories," Applied Optics, Vol.26, No. 23, 1 Dec.1987, pp. 4947-4960.

2.P.D. Wasserman, "Neural Computing: Theory and Practice," VAN NASTRAND REINHOLD New York 1989, pgs.222.

3. J. Ghosh, and K. Hwang, " Mapping Neural Networks onto Message-Passing Multicomputers," Jourl. of Parallel and Distributed Computing, Vol.6, pp.291-330, 1989.

4. J. Mohan Kumar and L.M. Patnaik, "Mapping Artificial Neural Networks onto the Extended Hypercube" Submitted to IEEE Trans. on Computers.

5. H. Yoon, J.H. Nang and S.R. Maeng, " Parallel Simulation of multilayered Neural Networks on Distributed-memory Multiprocessors," Microprocessing and Microprogramming Vol.29, pp. 185-195, 1990.

6. J. Mohan Kumar and L.M. Patnaik, "Extended Hypercube: A Hierarchical Interconnection Network of Hypercubes," to appear in IEEE Trans. on Parallel and Distributed Systems.

7. J. Mohan Kumar, L.M. Patnaik and Divya K. Prasad, " A Transputer Based Extended Hypercube ," Microprocessing and Microprogramming, Vol.29, pp.225-236,1990.

BAM Network	Uniprocessor	3-cube	EH(3,1)	5-cube	EH(3,2)
512X312 100 Patterns	25.6	16.0	8.8	1.48	0.786
512X512 500 Patterns	128.0	28.6	21.6	3.08	2.34
1024 X 1024 100 Patterns	100.0	36.5	23.5	6.68	4.24
1024 X 1024 500 Patterns	500.0	86.3	73.5	12.93	10.54

Table 1 Time in seconds to implement BAM

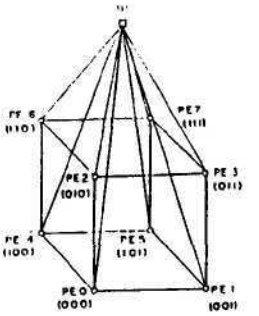

Fig.1a Extended Hypercube EH(3,1)

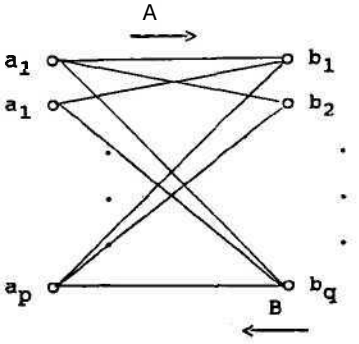

Fig. 2 Bidirectional Associative Memory

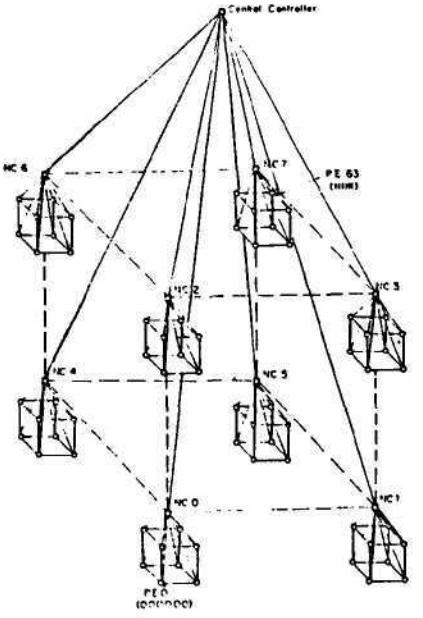

Fig.1b Extended Hypercube EH(3,2)

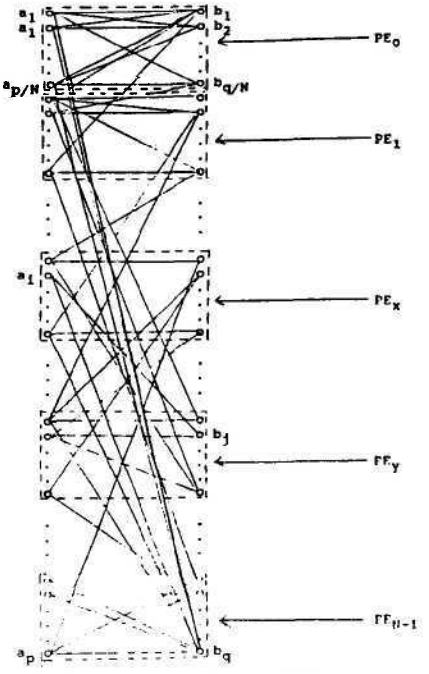

Fig.3 Mapping BAMs onto Multiprocessor Systems