EXPLOITING THE INHERENT PARALLELISMS OF BACK-PROPAGATION NEURAL NETWORKS TO DESIGN A SYSTOLIC ARRAY

Jai-Hoon Chung, Hyunsoo Yoon, and Seung Ryoul Maeng

Department of Computer Science
Korea Advanced Institute of Science and Technology (KAIST)
373-1 Kusong-Dong, Yusung-Gu
Taejon 305-701, Korea

ABSTRACT

In this paper, a two-dimensional systolic array for back-propagation neural network is presented. The design is based on the classical systolic algorithm of matrix-by-vector multiplication, and exploits the inherent parallelisms of back-propagation neural networks. This design executes the forward and backward passes in parallel, and exploits the pipelined parallelism of multiple patterns in each pass. The estimated performance of this design shows that the pipelining of multiple patterns is an important factor in VLSI neural network implementations.

1. Introduction

As simulations of artificial neural networks (ANNs) are time consuming on a sequential computer, dedicated architectures that exploit the parallelisms inherent in the ANNs are required to implement these networks. A systolic array [Kung82] is one of the best solutions to these problems. It can overcome the communication problems generated by the highly interconnected neurons, and can exploit the massive parallelism inherent in the problem. Moreover since the computation of ANNs can be represented by a series of matrix-by-vector multiplications, the classical systolic algorithms can be used to implement them.

There have been several research efforts on systolic algorithms and systolic array architectures to implement ANNs. The approaches can be classified into three groups. One is mapping the systolic algorithms for ANNs onto parallel computers such as Warp, MasPar MP-1, and Transputer arrays, another is designing programmable systolic arrays for general ANN models, and the other is designing a VLSI systolic array dedicated to a specific model. All of these implementations exploit the *spatial parallelism* and *training pattern parallelism* inherent in ANNs, and suggest the systolic ring array or two-dimensional array structures.

The back-propagation neural network [Rume86], in addition to the above two types of parallelism, has one more parallel aspect that the forward and backward passes can be executed in parallel with pipelining of multiple patterns. This paper proposes a systolic design dedicated to back-propagation neural network that can exploit this type of parallelism.

In the following sections, the existing systolic implementations and inherent parallelisms that we can exploit are first reviewed. Then a systolic array that can exploit the forward/backward pipelining of multiple patterns is presented. Finally the potential performance of this design is analyzed.

2. Background

The Back-Propagation Model

Let us consider an L-layer (layer 1 to layer L) network consisting of N_k neurons at the k-th layer. Layer 1 is the input layer, layer L is the output layer, and layers 2 to $L-1$ are the hidden layers. The operations of the back-propagation model are represented by following equations.

The Forward Pass

$$x_{pj}^{(n)} = i_{pj}, \quad (n = 1) \tag{1.1}$$

$$x_{pj}^{(n)} = \sum_{i=1}^{N_{n-1}} y_{pi}^{(n-1)} w_{ji}^{(n)}(t) - \theta_j, \quad (2 \le n \le L) \tag{1.2}$$

$$y_{pj}^{(n)} = x_{pj}^{(n)}, \quad (n = 1) \tag{2.1}$$

$$y_{pj}^{(n)} = f(x_{pj}^{(n)}) = \frac{1}{1 + e^{-x_{pj}^{(n)}}}, \quad (2 \le n \le L) \tag{2.2}$$

CH 3065-0/91/0000-2204 $1.00 © IEEE

The Backward Pass

$$\beta_{pj}^{(n)} = y_{pj}^{(n)} - d_{pj}, \quad (n = L) \tag{3.1}$$

$$\beta_{pj}^{(n)} = \sum_{k=1}^{N_{n+1}} \delta_{pk}^{(n+1)} w_{kj}^{(n+1)}(t), \quad (2 \le n \le L-1) \tag{3.2}$$

$$\delta_{pj}^{(n)} = \beta_{pj}^{(n)} y_{pj}^{(n)} (1 - y_{pj}^{(n)}) = g(\beta_{pj}^{(n)}), \quad (2 \le n \le L) \tag{4}$$

The Weight Increment Update

$$\Delta w_{ji}^{(n)}(u+1) = \Delta w_{ji}^{(n)}(u) + \delta_{pj}^{(n)} y_{pi}^{(n-1)}, \quad (2 \le n \le L) \tag{5}$$

$$w_{ji}^{(n)}(t+1) = w_{ji}^{(n)}(t) + \varepsilon \Delta w_{ji}^{(n)}, \quad (2 \le n \le L) \tag{6}$$

Terminating Conditions

$$E = \sum_p E_p = \sum_p \sum_{i=1}^{N_L} (y_{pi}^{(L)} - d_{pi})^2 < \alpha \tag{7}$$

Inherent Parallelisms

There are three types of parallelisms inherent in back-propagation neural networks. The first is the *spatial parallelism* which can be exploited by executing the operations required in each layer on many processors. The spatial parallelism may be classified into three levels according to the degree of parallelism exploitation.

S1) Partitioning the neurons of each layer into groups
S2) Partitioning the operations of each neuron into sum of products operations and the non-linear function
S3) Partitioning the sum of products operations into several groups.

These partitioned operations can be executed on different processors in parallel. Most of neural network simulators implemented on MIMD computers exploit parallelism S1) and S2), but parallelism S3) is not exploited much due to the communication overhead.

The second type of parallelism is the *pattern parallelism* which can be exploited by executing the different patterns on many processors. The pattern parallelism can be classified as follows:

P1) Partitioning the training pattern set, independent execution on many processors, and epoch update [Pome88, Mill89, Sing90]
P2) Pipelining of multiple training patterns
P3) Pipelining of multiple pattern recalls

The parallelism P2) can be exploited by allowing some processors that completed the parts of operations required to execute training for a pattern to start training for the next training pattern while the other processors are processing for the previous training patterns presented. The parallelism P3) can be exploited by allowing the pipelining of multiple pattern recalls.

The third type of parallelism is the *forward/backward pipelined parallelism* which can be exploited by executing the forward and backward passes of different training patterns in parallel [Sing90]. This parallelism implies the exploitation of the parallelism P2). The depth of pipelining can be maximized to two times the number of neurons in the network, and it is effective if the size of the training set is much larger than the number of neurons in the network. The forward/backward pipelined parallelism can be classified according to the depth of pipelining as follows:

F1) Pass-by-pass pipelining
F2) Layer-by-layer pipelining
F3) Neuron-by-neuron pipelining
F4) Connection-by-connection pipelining

Systolic Implementations of Artificial Neural Networks

There have been several systolic implementations of artificial neural networks, which can be classified as follows:

1) Mapping systolic algorithms for neural networks onto parallel computers [Mann90, Mill89, Pome88, Chin90]
2) Design of a programmable systolic array for general neural networks [Kung88, Rama90]
3) Design of a VLSI systolic array dedicated to a specific neural network model [Blay89, Blay90, Kwan90]

These also can be classified according to the dimension of their resultant systolic array. These implementations are summarized in Table 1. As shown in Table 1, the two-dimensional approaches can exploit more parallelisms. Our approach is distinguished from the other approaches in that it exploits the forward/backward pipelined parallelism described above.

Table 1. Systolic Implementations of Artificial Neural Networks

Dimension	Researchers	Model	Exploited Parallelisms	Target System
1-D Systolic Array	Kung & Hwang [Kung88]	General Model	Spatial S1) Pattern P1)	Transputer Array
	Pomerleau, et al. [Pome88]	Back Propagation	Spatial S1), S2) Pattern P1)	Warp
	Millan & Bofill [Mill89]	Back Propagation	Spatial S1) Pattern P1)	Transputer Array
	Mann & Haykin [Mann90]	Kohonen	Pattern P1)	Warp
	Kato, et al. [Kato90]	Back Propagation	Spatial S1)	Sandy/8
	Ramacher, et al. [Rama90]	General Model	Spatial S1), S2) Pattern P3)	Dedicated VLSI
2-D Systolic Array	Blayo & Hurat [Blay89]	Hopfield	Spatial S1), S2), S3) Pattern P3)	Dedicated VLSI
	Blayo & Lehmann [Blay90]	Hopfield Kohonen	Spatial S1), S2), S3) Pattern P3)	
	Kwan & Tsang [Kwan90]	Back Propagation	Spatial S1), S2),S3) Pattern P3)	
	Chinn, et al. [Chin90]	Back Propagation	Spatial S1), S2), S3)	MasPar MP-1
	Our Approach	Back Propagation	Spatial S1), S2), S3) Pattern P2), P3) Pipelined F4)	Dedicated VLSI

Updating the Weights

The issue of the weight updating time has been controversial with contention between researchers who update network weights continuously after each training pattern is presented (*continuous updating*) and those who update weights only after some subset, or often after the entire set, of training patterns (*batch updating*). To exploit the traing pattern pipelining, we use the batch updating method because the weights must not be changed for the forward pass and the backward pass of a pattern as shown in Equations (1)-(6), although the systolic array proposed in this paper can support both of the weight updating methods.

3. A Systolic Array for Back-Propagation Neural Network

The computation of artificial neural networks can be represented by a series of matrix-by-vector multiplications interleaved with the non-linear activation functions. The matrix represents the weights associated to the connections between two adjacent layers, and the vector represents the input patterns or the outputs of a hidden layer in the forward pass, or the errors propagated in the backward pass. The resultant vector of a matrix-by-vector multiplication is applied to the non-linear activation function and to the next layer.

To exploit the spatial parallelism, we do not consider the one-dimensional arrays. And to efficiently organize the systolic array even for a network in which the numbers of neurons of the adjacent layers are much different, we didn't use the systolic algorithms proposed in [Kung88, Kato90].

Array Organization

In Figure 1(a), the systolic array organization between two adjacent layers is shown. One layer consists of 5 neurons and the other layer consists of 3 neurons. Each basic cell w_{ji} is a computational element which contains the weight value and the weight increment associated to the connection between neuron i of a layer and neuron j of the next layer. The basic cell executes the sum-of-products operations as shown in Equations (1.2) and (3.2) and weight updating operations as shown in Equation (5) and (6).

The weight matrix is shared between the forward and backward passes, and the data paths of the activations forward-propagated and the errors back-propagated are disjoint. The θf unit executes the thresholding and the activation function as shown in Equations (1.2) and (2.2), and the g unit executes the derivative of the activation function as shown in Equation (4). The g unit has a FIFO queue to contain the outputs of a neuron, that are fed back to the basic cells to compute the weight updates in the backward pass. The operations of the basic cell that can be executed in parallel are shown in Figure 1(b).

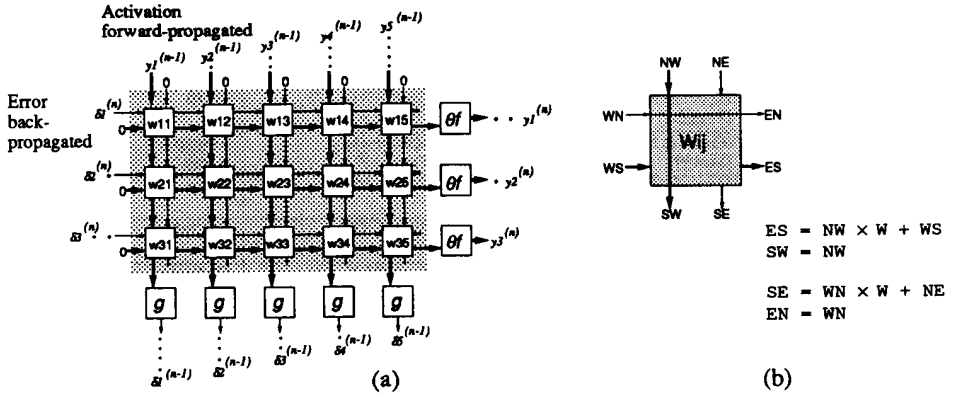

Figure 1. Systolic array organization between two adjacent layers of multi-layer neural networks: (a) combined design for forward and backward passes; (b) basic cell operations.

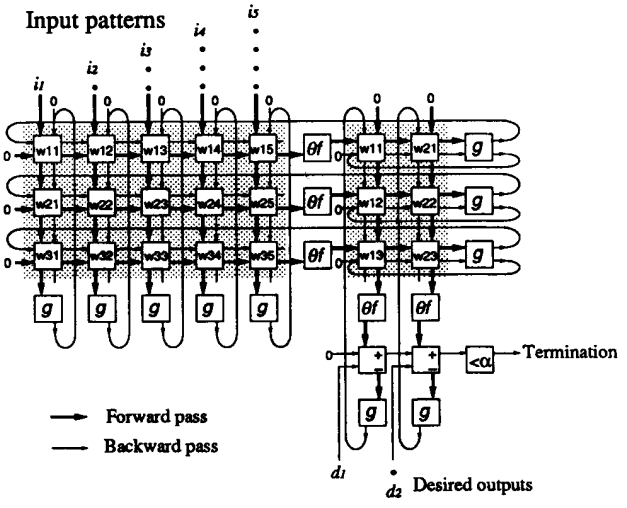

Figure 2. An example of a systolic array organization for (5-3-2) 3-layer neural network.

In Figure 2, an example of a systolic array organization for (5-3-2) 3-layer network is shown. The elements of two weight matrix are set into the two dimensional arrays, the 3-by-5 weight matrix represents the weights associated to the connections between the input layer and the hidden layer, and the 2-by-3 weight matrix represents the weights associated to the connections between the hidden layer and the output layer. The second matrix is transposed in the figure. While the weights are updated, the feeding of training patterns are stopped, and the pipeline is stalled.

Basic Cell Architecture

The internal data path of the basic cell is shown in Figure 3. It consists of three multipliers, three adders, two registers for weight value and weight increment. The communication clock cycle can be two times fast than the computation clock cycle, and the g unit sends the value at idle cycle.

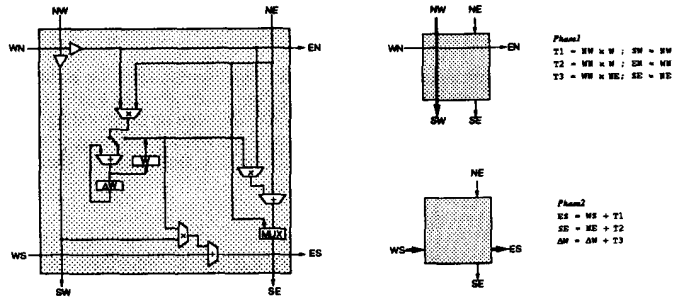

Figure 3. Internal data path of the basic cell

4. Performance Evaluation

Let us consider an L-layer network consisting of N_k neurons at the k-th layer. The required cycles, C, for forward and backward passes of this design are denoted by Equation (8):

$$C_{forward} = \sum_{i=1}^{L} N_i + L - 2 \tag{8.1}$$

$$C_{backward} = \sum_{i=1}^{L} N_i + L - 1 \tag{8.2}$$

When the forward and backward passes are pipelined, the required cycles for a single training pattern are denoted by Equation (9):

$$C_{single} = (C_{forward} + C_{backward}) - N_L + 1 \tag{9}$$

Equation (9) shows only the effects of the spatial parallelism and the forward/backward pipelining. The effects of the forward/backward pipelining on performance is not great even though N_L is much greater than 1. However it makes the pipelining of multiple training patterns possible. When multiple patterns are pipelined, the required cycles for learning p training patterns are denoted by Equation (10):

$$C_p = C_{single} + (p - 1) \tag{10}$$

The speedup, S_p, by the pipelining of p training patterns is denoted by Equation (11). When $p >> C_{single}$, the speedup can be approximated to C_{single} that is the depth of the pipelining and dependent only on the network size.

$$S_p = \frac{p \cdot C_{single}}{C_p} = \frac{p \cdot C_{single}}{C_{single} + (p - 1)} \tag{11}$$

For updating the weights, the cycles that feeding of training patterns is stopped are denoted by Equation (12):

$$C_{update} = 2 \sum_{i=1}^{L} N_i - N_1 - N_2 + L + 1 \tag{12}$$

When we update the weight t times during presentation of p patterns, the total required cycles are denoted by Equation (13):

$$C_{total} = \begin{cases} C_R + C_{update} \times t & (p > t) \\ C_{single} \times p & (p = t) \end{cases} \quad (13)$$

The performance of neurocomputers is measured in *MCUPS (millions of connection updates per second)*. Let N be the number of connections of the target neural network, then the MCUPS is calculated by Equation (14):

$$MCUPS = \frac{\textit{total connections calculated}}{\textit{total elapsed time in } \mu\text{sec}} = \frac{N \times p}{C_{total} \times \text{cycle time}} \quad (14)$$

In this design, the basic clock cycle time is determined as the maximum time among the time required by one multiplication and one addition, the time for thresholding and activation function (table lookup), and the time required to feed the input patterns continuously.

Speed measurements of several high performance neural network implementations have been performed for *NETtalk* [Sejn87] as a benchmark [Pome88, Kato90, Faur90]. Assuming we implement this design with 100 nsec of cycle time for computation and 50 nsec for communication, 248 MCUPS can be obtained for NETtalk for a single training pattern, only by the spatial parallelism and the forward/backward pipelining, and for (128-128-128) three-layer network [Sing90] with 64K training patterns pipelined, the performance that can be obtained is 324,000 MCUPS.

5. Conclusions

In this paper, a systolic array for back-propagation neural network that executes the forward and backward passes in parallel, and exploits the pipelining of multiple patterns in each pass has been presented. The estimated performance of this design shows that the pipelining of multiple patterns is an important factor in VLSI implementations of artificial neural networks. To implement a large network with this design, an efficient mapping method that still allows the exploitation of the pipelining is required as a further study.

6. References

[Blay89] F. Blayo and P. Hurat, "A VLSI Systolic Array Dedicated to Hopfield Neural Network," *VLSI for Artificial Intelligence*, Kluwer Academic Press, 1989, pp.255-264.

[Blay90] F. Blayo and C. Lehmann, "A Systolic Implementation of the Self Organization Algorithm," *Proc. of INNC*, Vol.II, Paris, July, 1990.

[Chin90] G. Chinn, et. al., "Systolic Array Implementations of Neural Nets on the MasPar MP-1 Massively Parallel Processor," *Proc. of IJCNN*, Vol.II, San Diego, California, June, 1990, pp.169-173.

[Faur90] B. Faure and G. Mazare, "Implementation of Back-Propagation on a VLSI Asynchronous Cellular Architecture," *Proc. of INNC*, Vol.II, Paris, July, 1990, pp.631-634.

[Kato90] H. Kato, et. al., "A Parallel Neurocomputer Architecture Towards Billion Connection Updates Per Second," *Proc. of IJCNN*, Vol.II, Washington, D.C., January, 1990, pp.51-54.

[Kwan90] H.K. Kwan and P.C. Tsang, "Systolic Implementation of Multi-Layer Feed-Forward Neural Network with Back-Propagation Learning Scheme," *Proc. of the IJCNN*, Vol.II, Washington, D.C., January, 1990, pp.84-87.

[Kung82] H.T. Kung, "Why Systolic Architectures?," *IEEE Computer*, January 1982, pp.37-46.

[Kung88] S.Y. Kung and J.N. Hwang, "Parallel Architectures for Artificial Neural Nets," *Proc. of ICNN, Vol.II*, San Diego, California, July 1988, pp.165-172.

[Mann90] R. Mann and S. Haykin, "A Parallel Implementation of Kohonen Feature Maps on the Warp Systolic Computer," *Proc. of IJCNN*, Vol.II, Washington, D.C., January, 1990, pp.84-87.

[Mill89] J.R. Millan and P. Bofill, "Learning by Back-Propagation: A Systolic Algorithm and Its Transputer Implementation," *Neural Networks*, Vol.1, No.3, July 1989, pp.119-137.

[Pome88] D.A. Pomerleau, et. al., "Neural Network Simulation at Warp Speed: How We Got 17 Million Connections Per Second," *Proc. of ICNN, Vol.II*, San Diego, California, July 1988, pp.143-150.

[Rama90] U. Ramacher and J. Beichter, "Systolic Synthesis of Neural Networks," *Proc. of INNC*, Vol.II, Paris, July, 1990, pp.572-576.

[Rume86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol.1, Foundations*, MIT Press, 1986, pp.318-362.

[Sejn87] T.J. Sejnowski and C.R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text," *Complex Systems*, Vol.1, 1987, pp.145-168.

[Sing90] A. Singer, "Exploiting the Inherent Parallelism of Artificial Neural Networks to Achieve 1300 Million Interconnects per Second," *Proc. of INNC*, Vol.II, Paris, July, 1990, pp.656-660.