A RECONFIGURABLE 'ANN' ARCHITECTURE

T.H. Madraswala, B.J. Mohd, M. Ali, R. Premi and Dr. M.A. Bayoumi

The Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504

Abstract : This paper proposes a design of a digital Artificial Neural Network (ANN). The architecture is based on SIMD processing configuration. Communication is done through broadcasting and also by systolical methods. With the help of a microprogrammed control unit, the design is mainly capable of implementing the following three models: 1)Hamming, 2)Hopfield and 3)Carpenter/Grossberg algorithms. The architeture was also designed in order to achieve parallelisim, modularity, adaptability, flexibility, speed, low cost, lesser silicon area, and expandibility.

I. INTRODUCTION

Efficient implementation of neural nets (NN) is one of the main factors which will impact the influence of neural network on digital and computer systems. The difficulty of NN implementation stems from their charecteristics, especially their extremely large number of processing models, very rich interconnection pattern, and different complex learning models. On the technological level, optical, micro electronics and hybrid are viable possibilities for implementing NN. However, micro electronics, especially very large scale integration (VLSI), is the dominant technology. An architecture of a NN is specified by the processing nodes, their interconnection network. and the control structure of the system. The mismatch between the architecture and the structure of the NN, w.r.t both nodes and interconnection,can be managed using flexible control and software.

Implementing synapses has to take into consideration the range of values of synaptic weights, programmability of the synapses, and range of values of neural states. The synapses may take two, three or multiple discrete values from certain range. As an example, consider the single chip neural network from Bell Laboratories [11]. It includes 54 neurons producing a thresholded sum of inputs and a full-interconnection matrix with excitatory and inhibitatory connections, which are adjustable by content of a RAM to three valued values (a,0,-b). There is no learning capability provided in the circuit. At Bell Communications, a chip was designed based on a Boltzmann machine [12]. The neurons computed a weighted sum of inputs from all other neurons and compare it to a nondeterministic threshold to set the output on or off. The weights are adjustable in a wide range of values (8 bits). The learning is implemented in the circuit according to the Boltzmann machine learning model.

In this paper, we have proposed an architecture for a digital Artificial Neural Network (ANN) . Although analog implementation has the potential of high densities and fast operation, it is suscepible to noise, temperature changes and inter-chip variation make manufacturing functionally identical circuits difficult . We therefore opted for digital implementation due to its accuracy, variety of weight-storage realization techniques (e.g static or dynamic RAM, mask ROM, etc.) and of interface for multichip systems. Also, digital implementation can realize more than one network and combine the results in a heirarchial fashion to solve large problems.

The architecture tends to capitalize on the increased flexibility using PLA's as the control structure. By remodeling the PLA, this approach basically redesigns the PU (Processing Unit) for the implemention of different neural models. The proposed architecture can be reconfigured to implement the following NN models: 1) Hamming, 2) Hopfield and 3) Carpenter/Grossberg algorithms. Performance and silicon area have been optimized at the expense of flexibility.

II. ARCHITECTURAL FEATURES

The architecture exhibits the following features : parallelism, modularity,adaptability, flexibility, low cost, speed and expandibility. Certain trade offs were made in attaining these goals.

a) PARALLELISM and MODULARITY Modularity reduces interconnection cost, and leads to replicative PU's. Each PU is composed of localized communications, processor and memory.

b) LEARNING and ADAPTABILITY A weight storage is built on each processor unit. This enhances the adaptive capability of the system by allowing it to learn on-chip without extra hardware [8]

c) LOW COST and SPEED For implementation, we have chosen CMOS because of its inexpensive nature. We have further tried to reduce the cost of CMOS implementation by minimizing silicon area requirements as much as possible. For instance, we decided to do away with any multiplier and divider units in the processor nodes to replace them with firmware. Also, by choosing a centralized digital update, we have reduced the *per-synapse* area to that of a few small memory cells. With small individual processors,large number of it can be accomodated in a small silicon area, hence, paving way for rapid inter-processor communication and increase in overall speed.

d) FLEXIBILITY and EXPANDIBILITY The PLA basically provides flexibility in emulating various neural models. In addition, regular interconnection structure enhances the expansion into larger systems/networks.By using programmable processors, multiplexed digital input and output [1], and by providing significant flexibility on chip, we have made the chip interface more like existing computational devices, which ease the task of integrating this architecture into a complete system.

0-7803-0593-0/92 $3.00 1992 IEEE

e) FAULT TOLERANCE Two mechanism are deviced to include fault tolerance capabilities into the system. The first is used in case of *processing unit* failure, the faulty unit can be bypassed using PU-PU bus. The second mechanism is employed incase of a bus failure, an alternative bus will be used.

III THE ARCHITECTURE

The ANN chip consists of a number of simple, digital signal processor like PUs (Processor Units), operating in an SIMD configuration. Broadcast interconnect is used to create inexpensive, high performance communication. The PU architecture is optimized for traditional neural network applications.

The proposed architecture may be used for a variety of algorithms in both the retrieving and learning phases of an ANN, e.g. Single layer feedback network (Hopfield), Competitive learning network (Carpenter & Grossberg Classifier), and Multilayer feed-forward network (Hamming).

The learning rules supported by the architecture are based on the common recursive weight updating formula.

$$w_{ij}^{(m+1)}(l) = F\Big(w_{ij}^{(m)}(l), \eta_{ij}(l), \delta E/\delta w_{ij}^{(m)}(l)\Big)$$

The new weight value $w_{ij}^{(m+1)}(l)$ at (m+1)th recursion can be determined by the current weight value $w_{ij}^{(m)}(l)$, the updating rate parameter $\eta_{ij}(l)$ and the gradient $\delta E/\delta w_{ij}^{(m)}(l)$.

The recursion index m can be used to represent either type of two possible recursions : pattern as in Carpenter & Grossberg Classifier or sweep as in Hopfield network. If m represents the pattern recursion, then the network updates the synaptic weights after the presentation of each training pattern. On the other hand, if m represents the sweep recursion, then the network updates the synaptic weights after the presentation of all the training patterns.

The iteration index l can represent one of the two possible iterations : time as in Hopfield and layer as in Hamming.

The updating rate $\eta_{ij}(l)$ can regulate the rate of change of each weight at each recursion. This can be set as a global constant as in Hopfield and Hamming networks, or as a locally dependent variable as in Carpenter & Grossberg Classifier.

The E function can be selected so that the weight training can be formulated as a problem of iterative optimization (maximization or minimization) of the function E.

The architecture is built to execute the following generic iterative formula during the retrieving phase of an ANN.

$$u_i(l+1) = \sum_{j=1}^{N_l} w_{ij}(l+1).a_j(l)$$
$$a_i(l+1) = F(u_i(l+1), \theta_i(l+1), a_i(l))$$

where $a_i(l)$ are the stimulus inputs and $\theta_i(l)$ are the external inputs. N_l is the number of neural units at the lth level and F_i is the non linear activation function. F_i can be deterministic function as in Hopfield (Hard Limiter) and Hamming (Threshold Logic) or Winner-take-all mechanism as in Carpenter and Grossberg Classifier [5][6].

A) MAPPING NEURAL NETS The designed archirecture is capable of emulating different types of neural network models by assigning nodes and weights to PUs in a predefined way. Different types of layered network model are shown below [2].

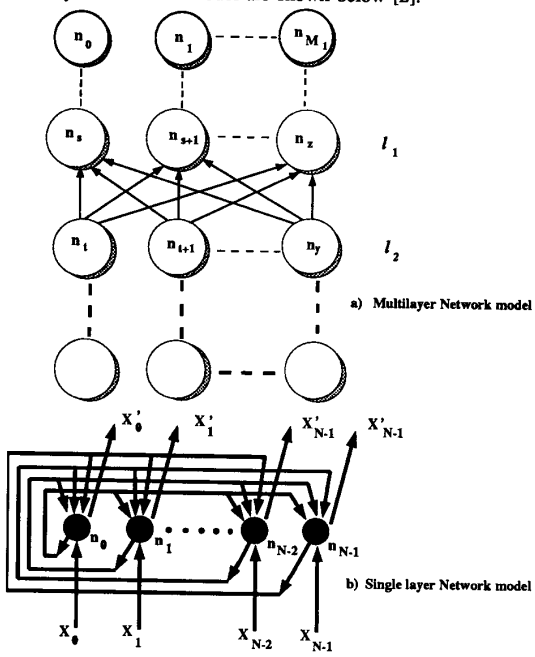

Fig 1 . Single layer and mulilayer neural networks models.

Two possible mapping techniques can be applied :

1) Centralized node computation (CNC)
2) Decentralized node computation (DNC)

Let us assume :

N : *number of PUs*
M_i : *number of nodes in layer i*
r : *relative address offset from the starting location of the algorithm memory block.*
$w_{i,j}$: *the associated with the connection from node j to node i*
Without loosing generality, it is assumed that

$$M_i = N.c \text{ or}$$
$$N = c.M_i$$

for some constant c .

1) Centralized Node Computation (CNC) The Neuron node is assigned to a PU, which performs all the computations of the corresponding node. The PU then, through broadcasting, transmits results to other processors, if required. In addition, more than one node may also be assigned to a PU, in which case, computations of the corresponding nodes are executed in a sequential manner. The mapping function in this method will be as follows :

consider two consecutive layers l_1 and l_2 (see Fig. 1). Node n in layer l_1 (or l_2) is assigned to the K^{th} processor as

$$K = n \bmod N$$

$w_{i,j}$ is stored at processor s and location r as

$$s = j \bmod N$$
$$r = (\lfloor j \div N \rfloor \times M_{l_2}) + i$$

Incase of single layer, M_{l2} = number of inputs.

2) Decentralized Node Computation (DNC) Unlike the CNC, the processing space of a node is not confined to a PU. Instead, it is distributed over all PUs. A PU recieves a part of the computation of a node. After execution, the result is directed to the other processors in a *systolic* manner. Once the process is executed by the last PU, the result is then broadcasted , or, outputed on the Outbus. In this method, process n of node n is initiated at K^{th} processor by

$$K = n \bmod N$$

$w_{i,j}$ is arranged in special mechanism to enhance parallel processing. In the case of multilayer system, $w_{i,j}$ is stored at processor s and location r as

$$s = i \bmod N$$
$$r = \lfloor (i/N) \rfloor \times M_{l_i} + \lfloor (j/N) \rfloor \times N + (j - \lfloor (j/N) \rfloor \times N - s) \bmod N$$

In single layer system

$$r = (j - s) \bmod N$$

CNC has simpler node computation processing, weight arrangements and controller complexity. However, DNC provides parallel processing, which enhances the system's effeciency and performance [3][5]. Consequently, DNC is chosen to be our design mapping technique.

B) PROCESSOR LOOP The system architecture is mainly comprised of Processor Units (PU) and Control Unit (CU) , arranged in a linear array and connected in a circular loop structure.The *processor loop* uses the PU-PU bus (Processor Unit to Processor Unit interconnect bus) for inter-processor communication as shown in Fig.2

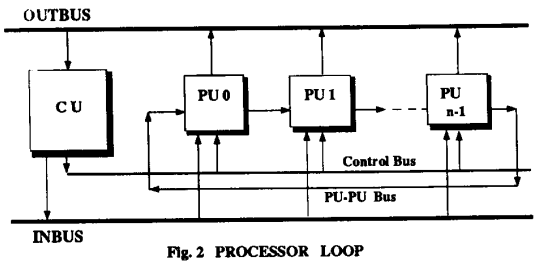

Fig. 2 PROCESSOR LOOP

Neural Networks are often critisized for its large number of connections. A typical network of N nodes will have $O(N^2)$ connections. To overcome this connection cost, broadcast structures were chosen to implement the communication between non-neighbouring PUs. Thus, this communication strategy apart from being simple, saves us from complex routing,control circuitry and the need to program complex interconnect structures, thus reducing silicon area costs. The processor loop mainly consists of the following :

1. BUS SYSTEM The bus system supports systolic communication (PU-PU bus), and broadcasting communication (INBUS and OUTBUS). Control bus transfers commands and/or data between CUs and PUs. The bus system enhances the fault tolerance capability. PU-PU bus is used to bypass any faulty Processing Unit. Incase of a bus failure, other buses can be used instead.

2. CONTROL UNIT (CU) The SIMD architecture is controlled by a microprogrammed Control Unit (CU). Through the Control bus, CU broadcasts commands/data to PUs. Each PU contains a PLA, which gets activated and generate nano instructions, upon recieving information from the CU. The CU monitors PU-PU communication among neighbouring PUs, which is basically systolic in nature. Otherwise, a PU transfers data to the OUTBUS, it is then transfered to the INBUS, from where some or all of the PUs are allowed to read the data. All of the transactions between PUs and the buses are monitored by the the CU. In our design, the CU stores the microinstruction to execute the following algorithms [6]:

1) Hopfield
2) Hamming
3) Carpenter/ Grossberg

*3. **PROCESSOR UNIT (PU)*** The PU is comprised of various units which help in processing the data transfers and execution of operations (see Fig. 3). Communication between the functional units takes place through an inner bus system.The three buses, Control bus, Inbus and Outbus are interfaced with the *decoder* , *outregister* and *inregister* , respectively. Data transfers through buses and, activation of functional units are controlled by control signals $c_0, c_1, \ldots c_n$, generated by the PLA. A brief description of various units which form the PU, follows.

INPUT REGISTER The input register connects the INBUS to the two internal data buses. The sequencer uses this register to latch constants such as threshold values and other initializing parameters required to set up a desired network. It is 16 bits wide.

OUTPUT REGISTER The output register connects the internal data buses to the OUTBUS. The bus arbitration needed for accessing OUTBUS is embedded in the PLA. Control signals from the PLA regulate the flow of information from output register to the OUTBUS. It is also 16 bits wide.

REGISTER ARRAY The register file contains 8 16–bit registers for intermediate storage of constants such as learning rates, threshold values, partial results of multiplication and division etc.

COUNTER This is a modulo 16 synchronous counter used to count number of iterations for the multiplication and division algorithms.

ACCUMULATOR The accumulator forms the output register for the 16–bit adder/subtracter. Together with the Q-register, it is used as an extended 32–bit register during execution of multiplication and division algorithms. Acc(-1) forms the sign bit for the adder/subtracter result.

Q-REGISTER This 16–bit register is used to hold multiplier (divisor), during execution of multiplication (division) algorithms. It is bi-directionally connected to the internal buses so that it may be written into or read from.

ADDER/SUBTRACTER The adder/subtracter takes 16 bit inputs and produces a two's complement 17 bit result. Adder overflow causes saturation to the largest positive or negative number, depending on the sign of the final result. It's inputs come directly from the two internal data buses. This allows all the data stores to directly access the only functional unit in the processor node.

WEIGHT MEMORY Memory is made up of 128x8 bits. Addresses 0–16 are allocated to weight storage for Carpenter/Grossberg algorithm. The next 16 bytes are used for storing weights and constants for Hamming network algorithm and the rest of the memory is used for simulating Hopfield neural network. Weight address generation is done through a counter which can be loaded from the internal bus or via an incrementer. A memory address register serves the purpose of an index register that points to the current working location of the memory. The memory can be read from and written into through a memory data register.

PLA The PLA acts as a microprogrammed control store whose sequencing is done by the control unit (CU). *Nano-operations* are activated by the micro-instruction, sent by the CU. The control signals generated by the *nano-instructions* also monitor the data transfers and communication between various units.

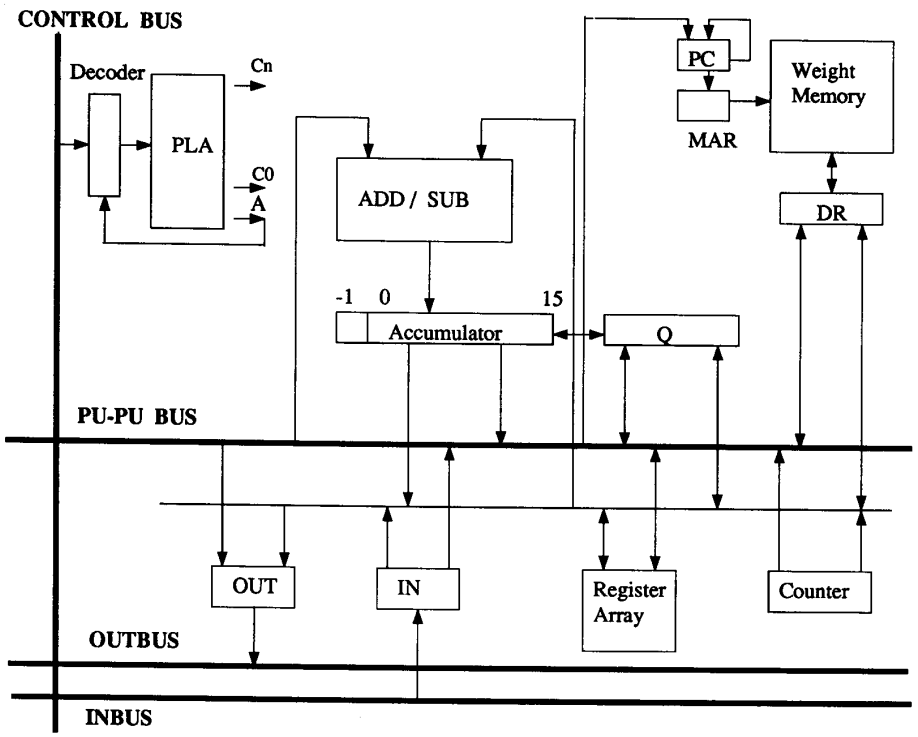

Fig. 3

PROCESSOR UNIT

IV. CONCLUSION

The above architecture provides a robust design for ANNs. The PUs' looped structure and communication methods, enhances modularity, flexibility,expandibility, low cost and parallelism. Bus net and DNC supports concurrent processing as well as fault tolerance. In addition, having a micro-programmed controlled unit and weight memory in the PUs , not only adds the adaptive capability but, also facilitates on-chip learning.

V. REFERENCES

[1] Jim Bailey and Dan Hammerston. Why VLSI implementation of associative require connection multiplexing. *Proceedings of the 1988 International Conference on Neural Networks. June 1988.*

[2] Hans P Graf, Lawrence D Jackel and Wayne E Hubbard. VLSI implementation of a neural network model. *IEEE, Computer, 21(3) : 41–49, 1988.*

[3] H T Kung. Why Systolic Architectures? *IEEE, Computer, 15(1), January 1982.*

[4] S Y Kung. VLSI Array Processors. *Prentice Hall Inc., N.J., 1988.*

[5] S Y Kung and J N Hwang. Parallel architectures for artificial nets. *IEEE, Int'l Conf. on Neural Networks, San Diego, Vol2 : 165–172, July 1988.*

[6] R P Lippman. An Introduction to computing with Neural Nets. *IEEE, ASSP magazine, Vol4 :4–22, 1987.*

[7] C Mead and L Conway. Introduction to VLSI Systems. *Addison-Wesley, 1980.*

[8] B Widrow and R Winter. Neural Nets for adaptive filtering and adaptive pattern recognition. *IEEE, Computer, 21 : 25–39, March 1988.*

[9] J J Hopfield and D W Tank. Computing with Neural Circuits: A Model.*Science, Vol233 : 625–633, August 1986.*

[10] J A Hartigan. Clustering Algorithms. *John Wiley & Sons, New York, 1949.*

[11] H. Graf, L. Jachel, W. Hubbard, VLSI Implementation of a Neural Network Model, *Computer, March 1988.*

[12] J. Alspector and R. Allen: A Neuromorphic VLSI Learning System, *Research in VLSI, Ed. Paul Loseleben, MLT press, 1987.*

[13] Dan Hammerstrom : VLSI Architecture for High Performance, Low Cost, On-chip Learning, *Adaptive Soultions Inc., Beaverton, Oregon, February 28th, 1990.*