Hardware Aspects of Artificial Neural Network and Its Applications

Maria Jamal
ECE Department,GGSIPU
Kashmere Gate,Delhi-6,India

Sayed A Imam
ECE Departmente ,JMI
Jamia Nagar,Delhi-25

Abstract- Artificial Neural Networks became a common solution for a wide variety of problems in many fields, such as control and pattern recognition to name but a few. Many solutions found in these and other Artificial Neural Network fields have reached a hardware implementation phase, either commercial or with prototypes. The most frequent solution for the implementation of Artificial Neural Networks consists of training and implementing the Artificial Neural Networks within a computer. In this paper we discussed the hardware aspects of ANN and its applications in various field.

Keywords-Artificial Neural Network, Hardware of ANN, applications of ANN

I. Introduction

The field of Artificial Neural Networks (ANN) has crossed different stages of development. One of the most important steps was achieved when Cybenko (Cybenko, 1989) proved that they could be used as universal approximators. A negative stage was brought by the book of Minsky and Papert called Perceptrons (Minsky et al., 1969). This negative phase was overcome when algorithms for training of multilayer ANN where proposed in the decade of the 80s.Since then much work has been done regarding ANN and their application to many different fields (Dias et al., 2001). Naturally the successful application to some areas led to commercial or specific applications that can work without having a computer attached. The need for leaving the most common implementation of ANN with a computer might arise from a number of reasons: reducing the cost of the implementation, achieving higher processing speed or simpler implementations. Reducing cost or having simpler implementations can be achieved simply by replacing the computer by specific hardware. Unlike the conventional von-Neumann architecture of computers that is sequential in nature, ANN profit from massively parallel processing (Liao). This can be exploited by specific hardware to increase processing speed. For these applications that share the necessity of working without a computer, some dedicated hardware has already been built avoiding the difficulty of producing hardware for each new application. A large variety of hardware has been designed to exploit the inherent parallelism of the neural network models. Despite the tremendous growth in the digital computing power of general-purpose processors, neural network hardware has been found to be promising in some specialized applications, such as image processing, speech synthesis and analysis, pattern recognition, high energy physics and so on. Neural network hardware is usually defined as those devices designed to implement neural architectures and learning algorithms, especially those devices that take advantage of the parallel nature inherent to ANNs.

The hardware produced has been a result of different needs and therefore has different uses. In order to choose hardware for a specific application, details about each circuit will be needed. The different solutions might be useful or not depending on the precision used for the weights, maximum number of weights, type of network implemented, availability of one circuit training algorithms and other characteristics. This article is confined to reporting the commercial chips that have been developed specifically for Artificial Neural Networks, independently of the technology used (Application Specific Integrated Circuits, Field Programmable Gate Arrays, Sea of Gates or others), leaving out others solutions. This option has been made because, aside from some hybrid solutions, most of the other solutions are based on cards which are built either with these chips, Digital Signal Processors or Reduced Instruction Set Computers. The utility of this study can therefore be summarized in two different directions: a short reference for those who need hardware for a specific implementation and information about the existing solutions for those who seek to develop a new implementation.

II. Artificial Neuron Model And Neural Network Structure

The study of artificial neural networks has been inspired in part by the observation that biological learning system are built of very complex webs of interconnected neurons. Typically, the human brain consists of approximately 10^{11} neurons, each with an average of 10^3 - 10^4 connections. It is believed that the immense computing power of the brain is the result of the parallel and distributed computing performed by these neurons [1]. The transmission of signals in biological neurons through synapses is a complicated chemical process in which specific transmitter substances are released from the sending side of the synapse. The effect is to raise or lower the electrical potential inside the body of the receiving cell. The neuron fires if the potential reaches a threshold. This is the characteristic that the artificial neuron model proposed by McCulloch and Pitts [2] attempts to reproduce. This neuron model is widely used in artificial neuron networks with some variations (Figure 1).

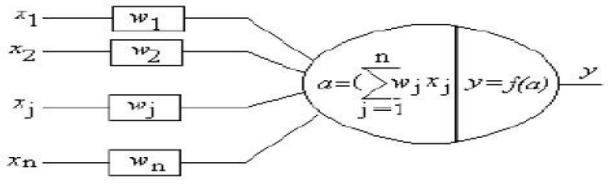

Figure 1: Artificial neuron model

The artificial neuron presented in Figure 1 has N inputs, denoted as $x_1, x_2, \ldots, x_n$. Each line connecting these inputs to the neuron is assigned a weight, denoted as $\omega_1, \omega_2, \ldots, \omega_n$, respectively. The action, which determines whether the neuron is to be fired or not, is given by the formula:

$$a = \sum_{j=1}^{n} w_j x_j \qquad (1)$$

The output of the neuron is a function of its action:

$$y = f(a)$$

Originally the neuron output function *f(a)* proposed in McCulloch-Pitts model was a threshold function. However, linear, ramp and sigmoid functions are also widely used today. An ANN system consists of a number of artificial neurons and a huge number of interconnections among them. According to the structure of the connections, two different classes of neural network architectures are identified [3](Figure 2). In layered neural networks, the neurons are organized in the form of layers. The neurons in one layer get input from the previous layer and feed their output to the next layer. This type of network is called *feedforward neural network*. The first and last layers are *input layer* and *output layer* respectively, and the layers that are not input or output are called {\it hidden layers}. Networks with one or more hidden layers are called *multi-layer networks*. *Multi-Layer perceptron* is a well-known feedforward layered neural network, on which the Backpropagation learning algorithm [4] is implemented.

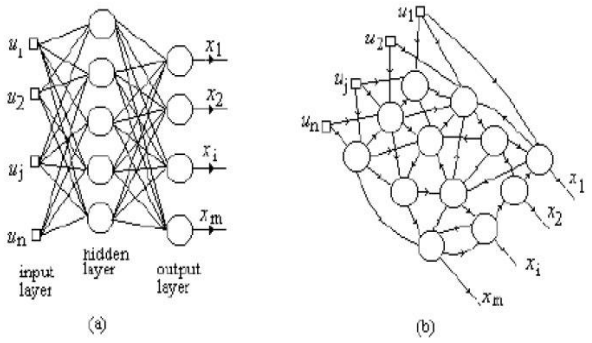

Figure 2: (a) Layered feed forward neural network. (b) Recurrent neural network.

The structure, where connections to the neurons are to the same layer or the previous layers, shown in Figure 2 (b), is called recurrent neural network. Hopfield Neural Network [5] is an example of widely used recurrent networks. Kohonen's selforganizing map (SOM) is another well-known neural network paradigm introduced by Kohonen[6]. Many other ANN learning algorithms have been proposed, including algorithms for more specialized tasks. ANN models have been proved to be successful in a number of applications, including text to speech conversion [7], protein structure analysis, autonomous navigation, game playing, image and signal processing, intelligent vision, pattern recognition, etc. These artificial models rely heavily on highly interconnected computational units functioning in parallel.

III. Neural Network Hardware

Artificial neural networks that solve difficult problems in areas such as speech recognition and synthesis, or pattern classification, consist of thousands of neuron with tens or hundreds of input each. Every neuron computes a weighted sum of its inputs and applies a non linear function to its result. Architectural parameters, such as the number of inputs per neuron and each neuron's connectivity vary considerably within the network, and from application to application. A special purpose neural network processor must be flexible and powerful enough to accommodate a wide range of applications. At the same time, the requirement must be carefully balanced and the special nature of the task exploited to bring an efficient implementation within reach of today's technology.

We can distinguish two phases of operation in many neural network applications. During the learning phase, the topology and weights of the network are determined from the labeled set of examples using a rule such as back propagation, "or a network growing algorithm." In the subsequent retrieval or classification phase, the network parameters are fixed.

The network recognizes pattern based on information stored in the architecture and weights during training. Since the computational and infrastructure requirements (training database) during the learning phase are considerably more complex than those for classification, efficiency considerations call for separate hardware for learning and retrieval. Network parameter determined during learning are downloaded in to processors specialized for the classification task. This approach , which we focus on here, contrasts with implementations of neural network processors with on chip learning. Those circuits are not suitable for pattern recognition problems we investigate here, because of limitations of training algorithms implemented on these chips or because of limited size of the network that can be trained.

The basic operation performed by the neuron during classification is weighted sum, followed by non linear squashing function, typically a hyperbolic tangent or approximation thereof:

$$y = f(\Sigma_i x_i w_i + b) \qquad (3)$$

We generally refer to the input xi of neuron as connection and ω_i parameters as weight. Each input is either tied to the output y of another neuron or to an external input. Optionally, a bias b may be added to the weighted sum.

The total number of connections in neural networks of applications such as hand written character recognition may amount to 10,000 to several hundred thousands "network that solve more general problems, such as recognition of entire words instead of isolated characters, require even larger numbers of connections". The speed requirement of typical applications call for a few tens to several thousands of classifications per seconds. For each classification, the network must evaluate one multiplication and one addition for every connection, which translates to few billion multiply-add operations per second. Only parallel implementations, in which several connections are evaluated concurrently, achieve such computational power.

The most general network topology permits connections between any two neuron. Such a high degree of (possible) connectivity, combined with the need for the parallel processing, result in enormous hardware requirements, and therefore calls for compromise. Usually, the neurons in a network are arranged in layers, each of which receive inputs only from the neuron in the previous layer. Layers may be fully connected; that is , each neuron may be connected to every neuron in the preceding layer. Often, however, we use local connectivity to express knowledge about the problem (geomatric relations such as neighborhood of pixel in an

image) in the network architecture and thus improve the recognition performance.

For example, the fact that some pixel in an image are adjacent to each other can be built into the network architecture by constraining neurons to receive inputs only from neighboring pixels. In a fully connected topology, such information must be derived from the training set during the learning phase, usually meeting with only partial success.

A neural network processor could be designed to implement only networks with fully connected topology. Local connectivity would then be realized by simply setting the weights of unused connection to zero. Since in typical neural networks the ratio of such unused connections to actual connections is easily 100, such an implementation is unacceptable in efficient. The added complexity of hardware required to support local connectivity is no match for the millions of connection saved.

Another challenge for a compact hardware implementation of a classifier is the amount of memory needed for storing several tens of hundreds of thousands of weights. Fortunately, the weights of many neurons in important connection topologies, including time delay or feature extraction neural network, are identical. In these architectures the connection topology corresponds to one or higher dimensional convolution, followed by nonlinear squashing function, as illustrated. We can realize such a structure with a single, time multiplexed neuron with corresponding saving of storage and computing devices.

We can further optimize the hardware complexity by matching the computational accuracy of the processor to the requirements of typical neural networks. Both experience and theory indicate that neural network classifiers can be designed to be insensitive to low- resolution arithmetic.

IV. Hardware Specifications

From the point of view of the user, the first step towards selecting a hardware solution is a short hardware specification. This specification includes the type of ANN (feedforward multi-layer, Radial Basis Function, Kohonen, etc.), the number of neurons, number of external input and outputs, the number of connections to each neuron, the precision, the speed of operation or performance and other characteristics that can be more or less important depending on the application. Most of these characteristics can be derived directly from the application to be developed, while others deserve a closer look. The precision used should be an important parameter to take into account. There might be a different precision for a number of parts: inputs, outputs, weights, internal calculations, accumulators and multiplication. The performance of the circuit can be measured in many different ways and is an issue that is far from being consensual. The most common performance rating is Connection Per Second (CPS) (Lindsey, et al., 1994) which is defined as the number of multiply and accumulate operations per second during the recall or execution phase. An equivalent measure exists for the learning phase: Connection Update Per Second (CUPS) and rates the number of weight changes per second. Other measures exist as well. The value of CPS can be normalized dividing it by the number of weights Nw (equation 1) obtaining the Connection Per Second Per Weight (CPSPW), which was suggested as a better way of rating the performance of each solution (Holler, 1991). CPSPW= CPS/Nw (1) Another measure is Connection Primitives Per Second (CPPS), which can be calculated as: CPPS=binxbwxCPS (2) where bin is the number of bits used for the input and bw is the number of bits used for the weights. This measure allows the precision to be included in the performance measure (Keulan et al., 1994) (Schüffny, et al., 1999). These two measures can also be applied to CUPS. Another parameter that can be used to analyze performance is power dissipation (Schüffny, et al., 1999). Depending on technology, clock frequency, number of processing elements, accuracy, etc, each hardware solution has a measure of power dissipation which cannot be compared directly. An energy per connection measure was proposed in (Schüffny, et al., 1999). This measure is an indicator of the energy efficacy in the circuits and is becoming more important as the integration in the solutions increases because the need to carry away dissipated power limits integration density at system level. Unfortunately this measure is frequently not available for most of the chips, which made it impossible to include it in this survey. For a more detailed specification other information may be taken into account: learning facilities, cascadability, type of storage of the weights, type of implementation of the activation functions, clock rate, number of inputs and outputs and technology or type of implementation of the circuit (analog, digital or mixed).

V. Classification of Neural Network Hardware

Neural network hardware ranges from single stand-alone neurochips to full-fledged neurocomputers. A variety of attributes have been used to classify neural network hardware, such as system architecture, degree of parallelism, inter-processor communication network, general purpose or special purpose device, on-chip or off-chip learning, and so on. Neural network hardware can be categorized into 4 classes by the degree of parallelism: coarse-grained, medium-grained, fine-grained and massive parallelism [8]. The number of processing elements yields the degree of parallelism of a system. The more parallel units there are, the faster data is processed. However, parallelism is expensive in terms of chip area or chip count. Therefore highly parallel systems usually employ simpler processing elements. The parallelism can be rated from only a few processing elements referred to as coarse-grained up to almost a one-to-one implementation of neural processing nodes called massive. There are no definite borders between these different categories. Parallel processing elements only speed up the computation when they do not run idle. Thus, for the system performance it is crucial that the inter-processor communication network provides the processing elements with sufficient data. Broadcast bus, linear array, systolic ring, crossbar and bidimensional mesh are the most frequently encountered communication networks of ANN systems [8]. Here we follow the scheme proposed in [9] and group neural network hardware into four main categories as shown in Figure 3.

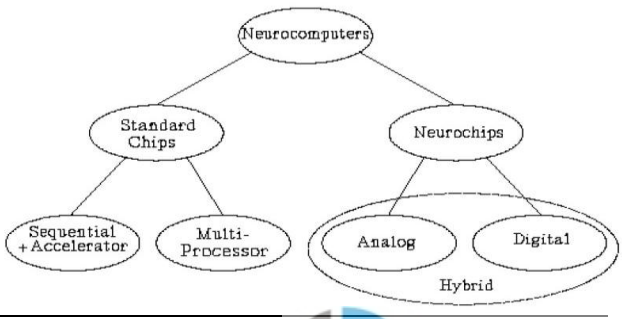

Figure 3: Neural network hardware categories after [5].

The first two main categories consist of neurocomputers based on standard ICs. They consist of Accelerator boards which speed up a conventional computer like a PC or workstation, and parallel multiprocessor systems, which mostly run stand alone and can be monitored by a host computer. The other main categories are neurochips built from dedicated neural ASICs (Application Specific Integrated Circuits). These neurochips can be digital, analog, or hybrid. The rest of this section will look at each of these categories and discuss their advantages and disadvantages.

A. Accelerator Boards

Accelerator boards are the most frequently used neural commercial hardware, because they are relatively cheap, widely available, simple to connect to the PC or workstation, and typically provided with user-friendly software tools. They reside in the expansion slots and are used to speed up the neural network computations. The speed-up that can be achieved is at about one order of magnitude compared to sequential implementations. Accelerator boards are usually based on neural network chips but some just use fast digital signal processors (DSP) that do very fast multiple-accumulate operations. A drawback of accelerator boards is that they are specialized for certain tasks, and thus lack flexibility and do not offer many possibilities for setting up novel paradigms. A good example of accelerator boards is IBM ZISC ISA and PCI Cards. The ZISC036 chip was developed at the IBM Essonnes Lab [10]. A single ZISC036 holds 36 neurons, or prototypes, to implement an RBF network trained with the RCE (or ROI) algorithm. The ISA card holds to 16 ZISC036 chips, giving 576 prototype neurons. The PCI card holds up to 19 chips for 684 prototypes. PCI card can process 165,000 patterns/sec, where patterns are 64 8-bit element vectors. Other accelerator systems include SAIC SIGMA-1 [11], Neuro Turbo [12], HNC [27], etc.

B. Neurocomputers Built from General Purpose Processors

General-purpose processors offer enough programmability for the implementation of neural functions. These implementations will of course never be maximally efficient. But because of their wide availability and relatively low prices, a number of neurocomputers have been assembled from generalpurpose chips. Implementations range from architectures of simple, low-cost elements (for example, the BSP400 [13] and COKOS [14]) to architectures with rather sophisticated processors like transputers, which are unique for their parallel I/O lines [15], or DSPs, which were primarily developed for correlators and discrete Fourier transforms [16]. Much experience has been gained from these implementations, which can be useful for the design of "true" neurocomputers, i.e., dedicated neurocomputers completely built from special purpose elements like neurochips. For instance, in many cases the sigmoid function forms the most computationally expensive part of the neural calculation. A solution for this can be found in using a look-up table rather than calculating the function [17]. Finding an interconnection strategy for large numbers of processors has turned out to be another non-trivial problem. Fortunately, much knowledge about the architectures of these massively parallel computers can be directly applied in the design of neural architectures. The RAP (Ring Array Processor) [18] is an example of neurocomputers built from general-purpose processors. It was developed at the ICSI (International Computer Science Institute, Berkeley, CA) and has been used as an essential component in the development of connectionist algorithms for speech recognition since 1990. Implementations consist of 4 to 40 Texas Instruments TITMS320C30 floating point DSPs containing 256 Kbytes of fast static RAM and 4 Mbytes of dynamic RAM each. These chips are connected via a ring of Xilinx programmable gate arrays (PGAs), each implementing a simple two-register data pipeline. Additionally each board has a VME bus interface logic, which allows it to connect to a host computer. The software support of RAP contains a workstation based command interpreter, tools for the standard C environment and a library of matrix and vector routines. A single board can perform 57 MCPS when computing a multi-layer perceptron network in forward operation, and 13.2 MCPS with backpropagation training.

C. Neurochips

For neurocomputers in Section 2 the neural functions are programmed on general purpose processors. Dedicated circuits are devised in special purpose chips for the neural functions. This will speed up the neural iteration time by about 2 orders of magnitude compared to general-purpose processor implementations. Several implementation technologies can be chosen for the design of neurochips. The main distinction lies in choice of a fully digital, fully analog, or hybrid design. Direct implementation in circuits in many cases alters the exact functioning of the original (simulated or analyzed) computational elements. This is mainly due to limited precision. The influence of this limited precision is of great importance to the proper functioning of the original paradigm. In order to build large-scale implementations, many neurochips have to be interconnected. Some chips are therefore supplied with special communication channels. Other neurochips are to be interconnected by specially designed communication elements.

VI. Application of Artificial Neural Network

Neural networks have been successfully applied to broad spectrum of data-intensive applications. The list below is based on real-world success stories. It will give an overview of the scope of problems that NeuroIntelligence can address.

1. Financial

Stock Market prediction
Credit Worthiness
Credit rating
Bankruptcy Prediction
Property Appraisal
Fraud Detection
Price Forecasts
Economic Indicator Forecasts

2. Medical

Medical Diagnosis
Detection And Evaluation of Medical Phenomena
Patient's length of Stay Forecasts
Treatment Cost Estimation

3. Industrial

Process Control
Quality Control
Temperature and Force Prediction

4. Science

Pattern Recognition
Recipes and Chemical Formulation Optimisation
Chemical Compound Identification
Physical System Modelling

Ecosystem Evaluation
Polymer Identification
Recognising Genes
Botanical Classification
Signal Processing: Neural Filtering
Biological Systems Analysis
Ground Level Ozone Prognosis
Odour Analysis and Identification

5. Data Mining

Prediction
Classification
Change and Deviation Detection
Knowledge Discovery
Response Modelling
Time series Analysis

6. Sales and Marketing

Sales forecasting
targeted Marketing
Service Usage faorecasts
Retail Margins Forecasting

7. Operational Analysis

Retail Inventories optimisation
Scheduling Optimisation
Managerial Decision Making
Cash Flow Forecasting

8. HR Management

Employee Selection and Hiring
Employee Retention
Staff Scheduling
Personnel Profiling

9. Energy

Electrical Load Forecasting
Energy Demand Forecasting
Short and Long-Term Load Estimation
Predicting Gas/Coal Index Prices
Power Control Systems
Hydro Dam Monitoring

10. Others

Sports Betting
Making Horse and Dog Racing Picks
Quantitative Weather Forecasting
Games Development
Optimisation Problems, Routing
Agricultural Production Estimates

VII. Discussions

Among the challenges neural network hardware faces today the competition with purpose hardware is probably the toughest one: computer architecture is a highly competitive domain that advances at an incredible pace. Neural networks in software have become well-established money making tools in a diverse range of pattern recognition and AI applications. The area of ANN hardware on the other hand is not yet as commercialized as general-purpose hardware. Also neural networks hardware tends to be more algorithm-specific. This requires a good knowledge about algorithms as well as system design and leads to a high time-to-market. Therefore, general-purpose computers can profit more often from advances in technology and architectural revisions. Also, in many other respects general-purpose hardware seems to be more user-friendly: it is not bound to algorithmic a-priori-assumptions and therefore offers high flexibility. Uniform programming interfaces exist for general-purpose hardware. This can be important not only to get a better start when programming a system, but also to allow reusability when moving on to the next hardware generation. On the other hand, there are ANN problems, exceeding the computational capabilities of workstations or PCs such as real-time applications, the simulation of large networks or networks employing very complex neuron models. For these applications neurohardware is attractive. Other niche areas for neural hardware are embedded applications of simple, hardwired networks, for example, voice recognition chips, and neuromorphic systems that directly implement a desired function, such as touchpad and silicon retinas. Neurohardware might provide a much better cost-to-performance ratio, lower power consumption and smaller size. The field of neural network hardware has become maturer since it's ``gold rush" period in late 1980s and early 1990s. Clearly an algorithmic success in artificial neural networks would revive the area of neurohardware. As long as conventional hardware can not provide sufficient performance, there is a need for neural network hardware.

VIII. Conclusions

As per the study and investigations, the information collected indicates that few neurochips are available commercially. The appearance of new solutions indicates that this field is still active, but the removal of the market of other solutions does not seem to be good news. As (Heemskerk) indicates, neurocomputer building is expensive in terms of development time and resources, and little is known about the real commercial prospects for working implementations. Moreover, there is no clear consensus on how to exploit the currently available VLSI and even ultra large-scale integration (ULSI) technological capabilities for massively parallel neural network hardware implementations. Another reason for not actually building neurocomputers might lie in the fact that the number and variety of (novel) neural network paradigms is still increasing rapidly. For many paradigms the capabilities are hardly known yet. Paradoxically, these capabilities can only be tested in full when dedicated hardware is available. These might be the reasons for the slow development of the ANN hardware market in the last years, but the authors believe that this situation will change in the near future with the appearance of new hardware solutions. In the user perspective, taking into account the information given here about the existing market, it should be noted that there is no “best” solution for every case but the most suitable solution should be found for each case. This is the reason why the authors decided not to make a performance comparison.

References

[1]. Rumelhart, D. E., McClelland, J. L. and the PDP Research Group, 1986, *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, vol. 1, MIT Press, Cambridge, Massachusetts, 1986.

[2]. McCulloch, W. S. and Pitts, W., 1943, A Logical Calculus of the Ideas Immanet in Nervous Activity. *Bulletin of Mathematical Biophysics*, vol. 5, 115-133, 1943.

[3]. Gelenbe, E. and Halici U., 1994, *Lecture Notes on Neural Networks*, METU.

[4]. Rumelhart, D. E. and McClelland, J. L., 1986, *ParallelDistributed Processing: Exploration in the Microstructure of Cognition (Vols. 1&2)*. Cambridge, MA: MIT Press.

[5]. Hopfield, J. J., 1982, Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences USA*

[6]. Kohonen, T., 1984, *Selforganization and Associative Memory*, Springer-Verlag.

[7]. Sejnowski, T. J. and Rosenberg, C. R., 1987, Parallel Networks That Learn to Pronounce English Text. *Complex Systems*, 1:145-168, 1987.

[8]. Schoenauer, T., Jahnke, A., Roth, U. and Klar, H., 1998, Digital Neurohardware: Principles and Perspectives. *Proceedings of Neuronal Networks in Applications (NN'98)*, Magdeburg, 1998.

[9]. Heemskerk, J. N. H., 1995, Overview of Neural Hardware. *Neurocomputers for Brain- Style Processing. Design, Implementation and Application*, PhD Thesis, Unit of Experimental and Theoretical Psychology, Leiden University, the Netherlands.

[10]. Lindsey, C. S., Lindblad, Th., Sekniaidze, G., Minerskjold, M., Szekely, S., and Eide, A.,1995, Experience with the IBM ZISC Neural Network Chip. *Proceedings of 3rd Int.Workshop on Software Engineering, Artificial Intelligence, and Expert Systems, for High Energy and Nuclear Physics*, Pisa, Italy, April 3-8, 1995.

[11]. Treleaven, P. C., 1989, *Neurocomputers. International Journal of Neurocomputing*, 1, 4-31, 1989.

[12]. Arif, A. F., Kuno, S., Iwata., A. and Yoshita, Y., 1993, A Neural Network Accelerator Using Matrix Memory with Broadcast Bus. *Proceedings of the IJCNN-93-Nagoya*, 3050-3053, 1993.

[13]. Heemskerk, J.N.H., Hoekstra, J., Murre, J.M.J., Kemna, L.H.J.K. and Hudson, P.T.W.,1994, The BSP400: A Modular Neurocomputer. *Microprocessors and Microsystems*, 18, 2, 67-78, 1994. [14]. Speckman, H., Thole, P. and Rosentiel, W., 1993, COKOS: A Coprocessor for Kohonen'sSelforganizing Map. Proceedings of the ICANN-93-Amsterdam, London Springer-Verlag, 1040-1045, 1993. 79.

[15]. Foo, S. K., Saratchandran, P. and Sundararajan, N., 1993, Parallel Implementation of Backpropagation on Transputers. *Proceedings of the IJCNN-93-Nagoya*, 3058-3061, 1993.

[16]. Onuki, J., Maenosono, T., Shibata, M., Iima, N., Mitsui, H., Yoshida, Y. and Sobne., M.,1993, ANN Accelerator by Parallel Processor Based on DSP. *Proceedings of the IJCNN 93-Nagoya*, 1913-1916, 1993.

[17]. Shams, S. and Gaudiot, J., 1992, Efficient Implementation of Neural Networks on the DREAM Machine. *Proceedings of the 11th International Conference on Pattern Recognition*, The Hague, The Netherlands, 204-208, 1992.

[18]. Morgan, N., Beck, J., Kohn, P., Bilmes, J., Allman, E. and Beer, J., 1992, The Ring Array Processor: A Multiprocessing Peripheral for Connectionist Applications. *Journal of Parallel and Distributed Computing*, 14, 248-259, 1992.