Hardware Implementation of RAM Neural Networks

Eduardo do Valle Simões, Luís Felipe Uebel & Dante Augusto Couto Barone

Informatics Institute - Federal University of the Rio Grande do Sul

Caixa Postal 15064 - Porto Alegre, RS 91501-970, Brazil

E-mail: EDSIM, UEBEL, BARONE@inf.ufrgs.br

Abstract

This work describes an alternative technique for hardware and software implementation of RAM based Boolean neural networks, which describes neurons using the VHDL language. An example of application consisting of the classification problem of the British mail scanned address is attended with a RAM architecture presenting 340 x 12-input neurons. The weights of each neuron are represented by its truth table and described using simple logic gates (AND, OR, and NOT), aiming to make possible the network logic minimisation and its hardware implementation by the ALTERA MAX+PLUS II fast prototyping package (Altera, 1992). The developed software tool allows the specification and training of the network. Then, its VHDL description is generated to be interpreted and minimised by the ALTERA EPLD design system. If it is not necessary to have high speed processing or if pre-processing phases are needed, the ANN can be implemented in software. The software strategy makes use of the direct translation of the VHDL description into a simplified C language code. Once the user has specified and taught the network, this approach makes possible automatic prototyping of RAM neural networks in software and hardware.

Key words: Boolean Neural Networks, Character Recognition, Fast Prototyping, Image Classification, Neural Network Hardware Implementation.

1. Introduction

During the last decade, the development of advanced techniques for microelectronics design have permitted efficient physical realisations of the Artificial Neural Network (ANN) theory (Arostegui, 1994, Verleysen *et al.*, 1994). On an application level, hardware implementations (primary analogue and digital VLSI) have brought high performance ANN systems (Serrano *et al.*, 1994, Woodburn *et al.*, 1994), while VLSI implementations of the neural network recall-mode have being researched and fully adaptive VLSI ANNs (i.e. including learning) are maturing (Lehmann, 1994, Morie and Amemiya, 1994). However, there are several

problems associated to the classical ANN models, related basically to their convergence properties, and to the necessity to define heuristically the proper network structure for a particular problem. Many researchers are addressing these problems, as for example Arostegui (1994) with evolutive neural models, that offer the possibility to reconstruct automatically during the training process the proper ANN structure able to handle a specific task.

The rise of the ANN application in complex problems like artificial vision, character recognition, voice, and signatures has claimed a constant optimisation of the design techniques (Lehmman, 1994). This optimisation aims to improve ANN performance and reduce the necessary computational resources. After all, many times higher speeds are only achieved with hardware implementations (Verleysen and Jespers, 1989, Mead, 1989, Rossetto *et al*., 1989, Lee and Sheu, 1992, Domínguez-Castro *et al*., 1992). The problem is, however, how to build an integrated circuit (IC) without generalising the neural network architecture, in order to justify development costs. This article shows an EPLD based solution that attaches this problem, considering only the recall-mode of RAM ANNs previously trained according to a specific application.

Many ANN applications need to be implemented in hardware due to the advantages that this solution presents: ***i)*** *Parallel Processing*, each ANN neuron processes a small portion of the final result; ***ii)*** *Asynchronous Operation*, ANN neurons can run at the maximum speed of the employed hardware; ***iii)*** *Fault Tolerance*, some connections can be broken and a few neurons can be removed without corrupt neural processing; and ***iv)*** *Regularity*, neurons are built from few simple components that can be repeated and connected in a regular structure (Lehmann, 1994). Some applications, therefore, do not require high performances and pre-processing phases (i.e., image acquisition, filtering) are necessary. For these cases, the software solution is more appropriated, because it is not necessary to build the interface between ANN and the image processing package. Due to these circumstances, it is also presented in this article a new solution based on a C language implementation of ANNs.

Software implementation of RAM based ANN is limited by the amount of available memory since RAM neurons are assembled as binary vectors, addressed by the inputs and stored into system memory (Meyers, 1992). If the number of inputs is high, what is very common in complex problems, each neuron will need a great amount of memory. This fact can make intractable the application of the RAM model in some cases with less available memory like microcontrollers.

Considering the above related problems we offer a solution that uses VHDL language, allowing each neuron to be described in software according to its truth table, instead of being stored in memory as a vector. This approach does not have the necessity of a large memory area and improves system flexibility for it can be executed in many computer platforms with pre-processing phases in order to decrease noise effects. The use of VHDL also permits a good interface with commercial IC design systems. If it is necessary to improve performance, the neural network can be interpreted by a CAD (Computer Aided Design) tool, and a specific IC can be generated. The employment of FPGAs (Field Programmable Gate Arrays) or EPLDs (Erasable Programmable Logic Devices) to implement a neural network is very attractive, since it allows fast IC prototyping and low cost modifications. This approach grants a reduction of network generalisation, decreasing costs and improving performance.

This research is based on the authors' previous works (Uebel and Barone, 1993, Simões *et al*., 1994, Uebel *et al*., 1995).

2. Implementation of RAN based Neural Networks

Despite the chosen technique, before implementing a neural network it is necessary to perform some tests to determine the network architecture (number of neurons, connectivity) that can solve the intended problem with the best relation between cost and recognition level. A software tool was developed to help with this task and the employed methodology used to

choose a specific RAM topology that can be successfully applied to the British mail classification problem will be presented.

2.1. Analyses of Different Topologies of the RAM Model

Many RAM topologies were analysed in order to specify which architecture presents the best recognition level for the British mail data bank classification (Filho *et al.*, 1992), chosen to perform the training of the implemented neural networks in order to facilitate comparisons to other methodologies. The training phase allows to identify the RAM configuration (number of neurons and neuron input number) that shows the best results for the classification problem of the British mail address. Sixty tests with many possible configurations were performed in order to evaluate the average recognition level. A set of 100 patterns with noise inclusion was used to teach each class (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9). It was noticed that the neural network can saturate if the number of presented patterns is high, instead of improving its recognition level. Table 1 presents the main obtained results.

Table 1 shows: ***i*) IN** ⇨ number of neuron inputs; ***ii*) NEU** ⇨ total number of neurons; ***iii*) PAT** ⇨ number of training patterns; ***iv*) REC** ⇨ average recognition level; ***v*) WREC** ⇨ worst recognition level; ***vi*) Error** ⇨ average recognition error; ***vii*) WError** ⇨ worst recognition error; ***viii*) REJ** ⇨ average rejection index, showing the pattern similarity rejection (the lower **REJ**, the greater the recognition level); ***ix*) WREJ** ⇨ worst rejection index.

The tests have begun with 100 x 4-input neurons to cover the 384 (24 X 16) pixels of the British character data bank. As tests went on with 5, 6, and 7 inputs, the worst case result was always below 90.0%. An 8-input neuron brought about better results, but only in a few cases. The same situation has happened to 9 and 10 inputs. For larger neurons, a greater number of patterns was used in the training phase. Table 1 also shows that 12-input neurons have

Tab. 1: Neural Network Recognition Level for Different Configurations.

	PARAMETER							
IN	**NEU**	**PAT**	**REC**	**WREC**	**Error**	**WError**	**REJ**	**WRej**
4	100	64	87.6%	52.0%	1.1%	2.0%	11.3%	45.0%
5	100	50	95.0%	79.0%	1.5%	2.2%	5.6%	24.0%
6	100	50	96.0%	89.0%	1.4%	2.1%	5.0%	22.0%
7	100	50	96.3%	88.0%	1.2%	1.9%	4.9%	15.0%
8	50	100	96.7%	86.0%	1.4%	4.0%	1.9%	10.0%
10	40	100	96.9%	92.0%	1.3%	4.0%	1.8%	4.0%
12	34	100	97.9%	96.0%	1.0%	4.0%	1.1%	3.0%
14	29	100	96.8%	90.0%	2.3%	7.0%	0.9%	3.0%
15	27	100	96.8%	92.0%	2.3%	7.0%	0.9%	2.0%

improved the obtained results (**REC** = 97.9% and **WREC** = 96.0%). Thus, the number of neurons was reduced to 34.

Another important factor for choosing a RAM configuration is the pattern rejection index (**REJ)** (Myers, 1992). This factor shows the pattern similarity rejection of the network, which is responsible for improving the recognition level. It has to be considered that the greater the number of neuron inputs, the higher **REJ**. Larger neurons have a low saturation level which improves the pattern discrimination and the rejection of the cross-link recognition (Myers, 1992). Fourteen-input neurons have had nearly the same effects as 12-input ones, but rejection has risen considerably. Fifteen or sixteen-input neurons have also shown good results, however their hardware implementation becomes difficult to put in practice.

2.2. Choosing the better Topology for Hardware Implementation

The obtained results allow the identification of the best neural network configuration to be implemented in hardware. A reasonable solution for this problem is the 12-input neuron because larger ones make difficult hardware implementation. This configuration has shown the best relationship between cost and recognition level, considering that only 34 neurons have to be implemented for each class.

Tab. 2: Recognition Level for Different Numbers of Training Pattern.

	PARAMETER							
TEST	**NEU**	**PAT**	**REC**	**WREC**	**Error**	**WError**	**REJ**	**WREJ**
1	34	50	95.2%	84.0%	1.4%	7.0%	3.4%	12.0%
2	34	50	96.0%	85.0%	1.3%	5.0%	2.7%	10.0%
3	34	50	96.3%	91.0%	1.3%	4.0%	2.4%	6.0%
4	34	100	96.9%%	93.0%	0.7%	3.0%	2.4%	4.0%
5	34	100	95.9%	84.0%	1.4%	6.0%	2.7%	10.0%
6	34	100	97.2%	90.0%	1.2%	4.0%	1.6%	7.0%
7	34	150	98.3%	94.0%	0.9%	2.0%	0.8%	4.0%
8	34	150	98.1%	91.0%	1.0%	5.0%	0.9%	4.0%
9	34	150	98.6%	95.0%	0.6%	3.0%	0.8%	2.0%
10	34	175	99.1%	95.0%	0.3%	1.0%	0.6%	4.0%
11	34	175	99.1%	95.0%	0.3%	1.0%	0.6%	4.0%
12	34	175	99.3%	98.0%	0.2%	1.0%	0.5%	2.0%
13	34	185	99.8%	98.0%	0.1%	1.0%	0.1%	1.0%
14	34	185	99.7%	97.0%	0.0%	0.0%	0.3%	3.0%
15	34	185	99.9%	99.0%	0.0%	0.0%	0.1%	1.0%
16	34	190	99.9%	99.0%	0.1%	1.0%	0.0%	0.0%
17	34	190	99.7%	97.0%	0.0%	0.0%	0.3%	0.3%
18	34	190	99.9%	99.0%	0.0%	0.0%	0.1%	1.0%
19	34	195	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%
20	34	195	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%
21	34	195	100.0%	100.0%	0.0%	0.0%	0.2%	2.0%

Table 2 presents the performed tests with 12-input neurons. Different training data sets were used in the tests. The results have been different since the neurons are randomly connected to the input pattern in each test. Table 2 shows that it is possible to improve the recognition level by increasing the number of training patterns. The average recognition level has risen from 52% (with 64 training patterns) to 100% (with 195 training patterns).

2.3. Rising Problems from Hardware Implementation of RAM based Neurons

Hardware implementation of neural networks tends to be a complex task due to their characteristic of massive parallel processing. This characteristic leads to several VLSI problems since all neural network neurons should be implemented (Lehmann and Lansnet, 1993).

Tab. 3: Memory Used for Different Topologies.

IN	NEU	Memory (*bits*)	Memory (*bytes*)
4	100	16.000	2.000
5	100	32.000	4.000
6	100	64.000	8.000
7	100	128.000	16.000
8	50	128.000	16.000
10	40	409.600	51.200
12	34	1.392.640	174.080
14	29	4.751.360	593.920
15	27	8.847.360	1.105.920

Boolean neural networks have binary element processing which makes easier their hardware implementation. In this case, neuron operations can be directly mapped into logic gates, without the necessity of complex state machines (Simões *et al*., 1994, Uebel *et al*., 1995).

Memory allocation in RAM based neural networks is a limiting factor of their employment (Myers, 1992). Table 3 presents the requirements for memory allocation of some implemented architectures. As it can be seen in Table 3, the greater the neuron input number, the greater the network memory necessity.

3. Hardware Implementation of the RAM Model

The proposed design strategy consists of two phases: specification and training of the neural network; and logic mapping of the neurons into a hardware description language. All RAM neurons, as well as the sum of the neuron groups and the comparison process were implemented into ALTERA EPLDs (Altera, 1992).

The developed software system is used to generate the VHDL description of the neurons (Simões *et al*., 1994). The VHDL code can be interpreted by the MAX+PLUS II system of the ALTERA package, which performs the network logic minimisation and generates the EPLD configuration (steps ⑥ and ⑦ bellow). The design environment is divided into eight different steps:

① Sizing of the desired RAM based neural network according to the user's specific application;

② Neural network automatic training with the I/O patterns presented by the user;

③ Simulation of the neural net behaviour. The goal of this step is to determine if the net recognition level is sufficiently high to fulfil the application necessities;

④ Logic mapping of each neuron of the trained net, which is treated as a *black-box*;

⑤ Generation of the VHDL description of all trained neurons of the RAM neural network circuit;

⑥ Logic minimisation of the VHDL description;

⑦ Partitioning and implementation of the network circuit into the ALTERA EPLD;

⑧ Translation of the VHDL description into a C language code (if software implementation is chosen);

Neural network weights are mapped into the neuron VHDL description by the software. Therefore, the ALTERA package provides a minimisation of the logic gates where only the significant connections are taken into account. Each neural network neuron is mapped as a *black-box* using this technique. The system stimulates a neuron with all possible input combinations and identifies the output that corresponds to each stimulus. With this procedure it is possible to construct a truth table that leads to a logic description of the neuron. This description is used to perform a schematic diagram of each neuron. Weights are implicit to neuron logic mapping, reducing the number of interconnections and, consequently, increasing circuit performance.

The VHDL description permits a direct interaction of the neural network design software with commercial FPGA prototyping systems. The network simulation phase provides a feedback to the user, making modifications in the network dimensions easier.

According to Table 1 and 2, a RAM architecture with 12-input neurons was chosen to solve the address classification problem. Figure 1 presents the neural network topology containing 340 neurons and the way the circuit is partitioned into 5 EPLDs. Once the neurons are trained and described by the software tool, the ALTERA system can process them as logic gates. This fact leads to a great logic minimisation, due to the redundant characteristics of neural networks. A set of logic equations (see Table 4), which can be directly implemented into an EPLD, is the result of this operation.

Table 4 describes the equations that represent the truth table of a RAM neuron of the class that represents number “Four”. As it can be seen in Table 4, only memory positions containing the logic value "1" are taken into account, instead of storing great vectors in memory. This technique allows a great economy of resources that, in some cases, can permit RAM based neural network employment in very complex applications.

A full-parallel implementation of a RAM neural network, trained according to the British mail data bank, becomes possible with the technique proposed in this work, because it is necessary only few simple logic gates to describe each neuron (as it can be seen in Table 4).

Table 4 presents a trained neuron represented by a minimised and a non-minimised VHDL description. This neuron is one of those that are responsible for the recognition of the class that represents number “Four” (see Figure 2). The developed software just processes memory positions containing the logic value “1”, instead of using great portions of memory to store large binary vectors. It can be observed that the necessary hardware resources can be represented by a combinational circuit of just few simple logic gates. It is shown in the table only one neuron of one class, but all the 340 neurons (34 neurons for each class) were implemented.

Tab. 4: VHDL Description of a Neuron of the Class that Represents Number "Four".

Description	Neuron of the Class "Four"
Non-Minimised ! = NOT & = AND # = OR	Neuron = !e1 & !e2 & e3 & e4 & e5 & e6 & e7 & e8 & e9 & e10 & !e11 & !e12 # !e1 & e2 & e3 & e4 & e5 & !e6 & !e7 & !e8 & e9 & e10 & e11 & !e12 # e1 & e2 & e3 & !e4 & !e5 & !e6 & !e7 & e8 & !e9 & !e10 & !e11 & !e12 # !e1 & e2 & e3 & e4 & e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # !e1 & e2 & e3 & !e4 & !e5 & !e6 & !e7 & e8 & e9 & e10 & e11 & !e12 # !e1 & !e2 & e3 & e4 & e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # !e1 & !e2 & e3 & e4 & e5 & e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # e1 & e2 & e3 & !e4 & !e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # !e1 & !e2 & e3 & e4 & e5 & !e6 & !e7 & !e8 & e9 & e10 & e11 & !e12 # e1 & e2 & e3 & !e4 & !e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # !e1 & e2 & e3 & e4 & e5 & !e6 & !e7 & e8 & e9 & e10 & e11 & !e12 # !e1 & e2 & e3 & e4 & !e5 & !e6 & !e7 & e8 & e9 & !e10 & !e11 & !e12 # e1 & e2 & e3 & e4 & !e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # !e1 & e2 & e3 & e4 & e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # e1 & e2 & e3 & !e4 & !e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # !e1 & !e2 & e3 & e4 & !e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # !e1 & e2 & e3 & e4 & e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12 # !e1 & !e2 & e3 & e4 & e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12;
Minimised ! = NOT & = AND	Neuron = !Aux1; Aux1 = (!e1 & e2 & e3 & e4 & !e5 & !e6 & !e7 & e8 & e9 & !e10 & !e11 & !e12) & (!e1 & e2 & e3 & !e4 & !e5 & !e6 & !e7 & e8 & e9 & e10 & e11 & !e12) & (!e1 & !e2 & e3 & e4 & e5 & e6 & e8 & e9 & e10 & !e11 & !e12) & (!e1 & e2 & e3 & e4 & e5 & !e6 & !e7 & e8 & e9 & e10 & !e12) & (e1 & e2 & e3 & !e5 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12) & (!e1 & e3 & e4 & e5 & !e6 & !e7 & !e8 & e9 & e10 & e11 & !e12) & (!e1 & !e2 & e3 & e4 & !e6 & !e7 & e8 & e9 & e10 & !e11 & !e12) & (e1 & e2 & e3 & !e4 & !e5 & !e6 & !e7 & e8 & !e10 & !e11 & !e12);

Figure 3 shows the implemented circuit of the neural network into 5 EPLDs. The neural network had to be partitioned into five ICs due to the great amount of inputs (see Figure 1). The input pattern has 384 pixels and the greater available EPLD has only 160 I/O pins. The neural network was partitioned into 5 ICs: 4 ICs are responsible for the implementation of the neurons (IC1, IC2, IC3, and IC4); and IC5 calculates the sum of the neuron groups and the comparison process that specifies which class is dominant. IC1, IC2 and IC3 have 96 inputs to load ¼ of the

Tab. 5: C Language Description of a RAM Neuron of the Class "Four".

Description	Neuron of the Class "Four"
C Language **~ = NOT** **& = AND**	Neuron = ~Aux1; Aux1 = (~e1 & e2 & e3 & e4 & ~e5 & ~e6 & ~e7 & e8 & e9 & ~e10 & ~e11 & ~e12) & (~e1 & e2 & e3 & ~e4 & ~e5 & ~e6 & ~e7 & e8 & e9 & e10 & e11 & ~e12) & (~e1 & ~e2 & e3 & e4 & e5 & e6 & e8 & e9 & e10 & ~e11 & ~e12) & (~e1 & e2 & e3 & e4 & e5 & ~e6 & ~e7 & e8 & e9 & e10 & ~e12) & (e1 & e2 & e3 & ~e5 & ~e6 & ~e7 & e8 & e9 & e10 & ~e11 & ~e12) & (~e1 & e3 & e4 & e5 & ~e6 & ~e7 & ~e8 & e9 & e10 & e11 & ~e12) & (~e1 & ~e2 & e3 & e4 & ~e6 & ~e7 & e8 & e9 & e10 & ~e11 & ~e12) & (e1 & e2 & e3 & ~e4 & ~e5 & ~e6 & ~e7 & e8 & ~e10 & ~e11 & ~e12);

total pixels. Each one of these 3 EPLDs can implement 80 neurons (8 per class). They have a 3-bit output per class (30 at all) to inform IC5 the number of neurons per class that could recognise the presented pattern. IC4 has other 96 inputs, but implements 10 neurons per class (100 neurons at all) in order to fulfil the 340 neurons of the network. It needs 4 bits per class (40 at all) to inform IC5 the number of activated neurons per class. IC5 receives 130 bits (30 + 30 + 30 + 40) carrying information from IC1, IC2, IC3, and IC4. Thus, this EPLD uses other 4 bits to determine which class of patterns (0, 1, 2, ..., 9) was recognised by the network.

4. Software Implementation of the RAM Model

For some applications where a pre-processing phase is necessary, an integrated circuit may not be the most appropriated solution. In these cases, a software approach is more flexible because it permits a better integration with other routines of the system. Then, the description of the RAM based neural network can be automatically translated into a C language code. Consequently, the network circuit can be directly executed into the microprocessor ALU (Arithmetic Logic Unit). This fact decreases the execution time of the algorithm. The advantage of this technique is the employment of the microprocessor ALU to execute the logic operation set of the RAM algorithm. Simple logic functions (*NOT*, *AND*, and *OR*) are faster to execute than complex floating point operations, used by a greater number of neural models.

Table 5 presents the implementation in C language command lines of the minimised description of the neuron shown in Table 4. It can be noticed the similarities between these two representations, that make the conversion easier. VHDL is used because it can suffer a logic minimisation that reduces processing time and improves software performance.

The C language implementation is not so fast as the EPLD circuit (hardware implementation is 23,460 times faster than software), but the software implementation is more flexible and portable. Software speed is not so high due to the serial execution of the logic operations of the neurons by the microprocessor, that takes many clock cycles in comparison to the hardware approach, which takes just one. The software solution is, however, many times faster than other continuos neural networks, like Backpropagation (Freeman, 1992).

Table 6 shows the total processing time of the hardware and software implementations. During the execution of the ANN algorithm, approximately 435,000 logic operations are necessary to recognise one character, what means that an average of 128 logic operations have to be executed per neuron. An INTEL 486 DX 33Mhz, with 8 Mbytes main RAM, 256 Kbytes cache memory, using Windows 3.1 and the Watcom C/C++ 10.0 compiler, was used to calculate the total delay of the software implementation. The hardware solution, presented in Figure 3, was implemented with five 50 MHz ALTERA EPLDs (EPM 7256E - MAX 7000). The total amount of neurons was partitioned into IC1, IC2, IC3, and IC4, where the neurons and the output codification process took 40 ns. The comparison process with multiplexers and comparators was performed in 220 ns by IC5.

5. Conclusions

This article has presented an alternative technique for sizing and implementing RAM based Boolean neural networks in hardware and software. The main circumstances of sizing and choosing a RAM architecture for scanned character recognition were shown, as well as their

Tab. 6: Execution Time of Hardware and C Language Implementations.

Type of the Implementation	Execution Time
Hardware	260 ns
C Language	6.1 ms

importance to the network recognition level. Some significant problems related to IC neural network design were discussed and a VHDL based solution was proposed.

The advantages of the VHDL employment are the great facilities of hardware and software implementation of neural networks. The use of EPLDs allows high speed parallel processing of RAM neural networks. This fact increases system performance while it permits neural network appliance on real time image recognition applications. A simplified alternative for network design is the conversion of the network description into C language command lines. This alternative allows large dimension RAM implementation to attain high recognition levels since the user can build a complex network, because it does not have the necessity of storing large binary vectors in system memory.

In order to implement the above described techniques, the GSN (Goal Seeking Neuron (Filho *et al.*, 1990)) based neural network fast prototyping system, proposed by the authors in a recent publication (Simões *et al.*, 1994), was modified to perform sizing, training, and mapping of RAM based neural networks into logic gates. Thus, the VHDL description of these logic gates can be developed on an IC design tool, as for example the ALTERA commercial prototyping system. The high performance of the presented technique in both hardware and software implementations allows the employment of large RAM architectures in real time applications. The presented approach makes available to the users of complex the neural networks the great capacity and training facilities of the RAM model.

6. References

Altera Corporation (1992). MAX+PLUS II - Getting Started, 137.

Arostegui, J. M. M. (1994). VLSI Architectures for Evolutive Neural Models PhD Thesis, Universitat Politecnica de Catalunya, Departament d'Enginyeria Electronica, Barcelona, Spain, 134.

Domínguez-Castro, R., A. Rodríguez-Vázques, J. L. Huertas and Sánchez-Sinencio (1992). Analog Neural Programmable Optimizers in CMOS VLSI Technologies, IEEE Journal of Solid-State Circuits, 27(7), 1110-1115.

Filho, E. C. D. B., D. L. Bisset and M. C. Fairhurst (1990). A Goal Seeking Neural for Boolean Neural Networks, Proc. International Neural Network Conference, Paris, France, 2, 894-897.

Filho, E. C. D. B., M. C. Fairhurst, and D. L. Bisset (1992). Analysis of Saturation Problem in RAM-Based Neural Network, Electronics Letters, 28(4), 345-346.

Freeman, J. (1992). Neural Networks: Algorithms, Application and Programming Techniques, Computation and Neural System Series, USA, 400.

Lee, B. W. and B. J. Sheu (1992). General-Purpose Neural Chips with Electrically Programmable Synapses and Gain-Adjustable Neurons, IEEE Journal of Solid-State Circuits, 27(9), 1299-1302.

Lehmann, T. and J. A. Lansner (1993). An Analog CMOS Chip Set for Neural Networks with Arbitrary Topologies, IEEE Transaction on Neural Networks, 4(3), 441-444.

Lehmann, T. (1994). Hardware Learning in Analogue VLSI Neural Networks, Ph.D. These, Technical University of Denmark, Lyngby, Denmark, 209.

Mead, C. (1989). Analog VLSI and Neural Systems, Addison-Wesley Publishing Company, 371.

Morie, T. and Y. Amemiya (1994). An All-Analog Expandable Neural Network LSI with On-Chip Backpropagation Learning, IEEE Journal of Solid-State Circuits, 29(9), 1086-1093.

Myers, C. E. (1992). Delay Learning in Artificial neural Networks, Chapman & Hall, Londres, 157.

Rossetto, O., C. Jutten, J. Herault and I. Kreuzer (1989). Analog VLSI Synaptic Matrices as Building Blocks for Neural Network, IEEE Micro, 9(12), 56-63.

Serrano, T., B. Linares-Barranco and J. L. Huertas (1994). A CMOS VLSI Analog Current-Mode High-Speed ART1 Chip, Proceedings IEEE International Conference on Neural Networks, Orlando, 3, 1912-1916.

Simões, E. V., L. F. Uebel and D. C. Barone (1994). Fast Prototyping of Artificial Neural Network: GSN Digital Implementation**,** Proceedings IV International Conference on Microelectronics for Neural Networks and Fuzzy System, Torino, Italy, 192-201.

Uebel, L. F. and D. A. C. Barone (1993). GSN Neural Network in Hardware, ANNES' 93 - The First New Zeland International Two-Stream Conference on Artificial Neural Networks and Expert Systems, Dunedin, New Zeland, 126-129.

Uebel, L. F. Uebel, E. V. Simões and D. A. C. Barone (1995). A Comparision between Three Different GSN Model Hardware Implementations with the Appliance of an ANN Fast prototyping System, World Congress on Neural Networks, Washington, USA, in CD-ROM.

Verleysen, M. and P. G. A. Jespers (1989). An Analog Implementation of the Hopfield's Neural Networks, IEEE Micro, 9(12), 46-55.

Verleysen, M., P.Thissen, J. L. Voz, J. Madrenas (1994). An Analog Processor Architecture for a Neural Network Classifier, IEEE Micro, 14(3), 16-28.

Woodburn, R., H. M. Reekie and A. F. Murray (1994). Pulse-stream Circuits for On-chip Learning in Analogue VLSI Neural Networks, Proceedings on IEEE International Symposium on Circuits and Systems, London, 4, 103-106.

List of Figures

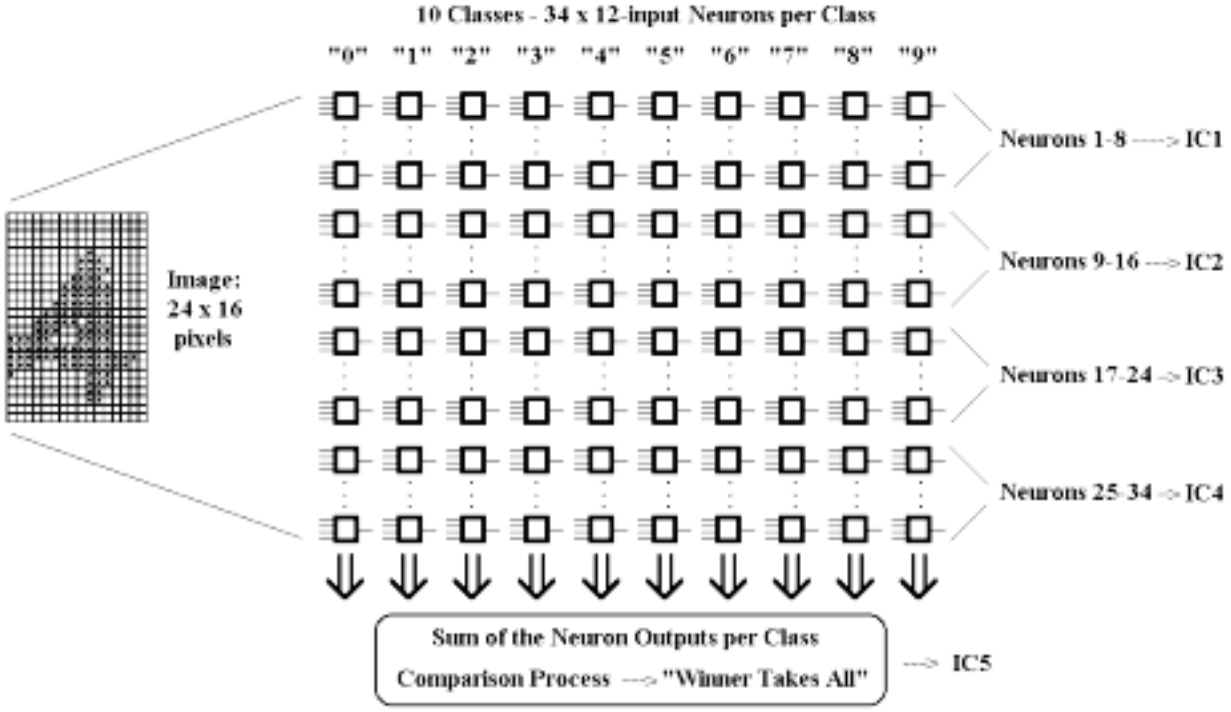
10 Classes - 34 x 12-input Neurons per Class
"0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
Image:
24 x 16
pixels
Neurons 1-8 ---> IC1
Neurons 9-16 ---> IC2
Neurons 17-24 ---> IC3
Neurons 25-34 ---> IC4
Sum of the Neuron Outputs per Class
Comparison Process ---> "Winner Takes All"
---> IC5

Figure 1

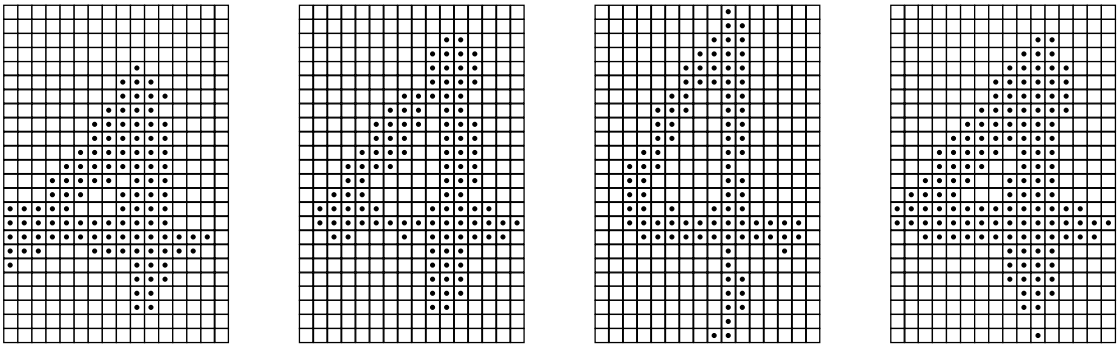

Figure 2

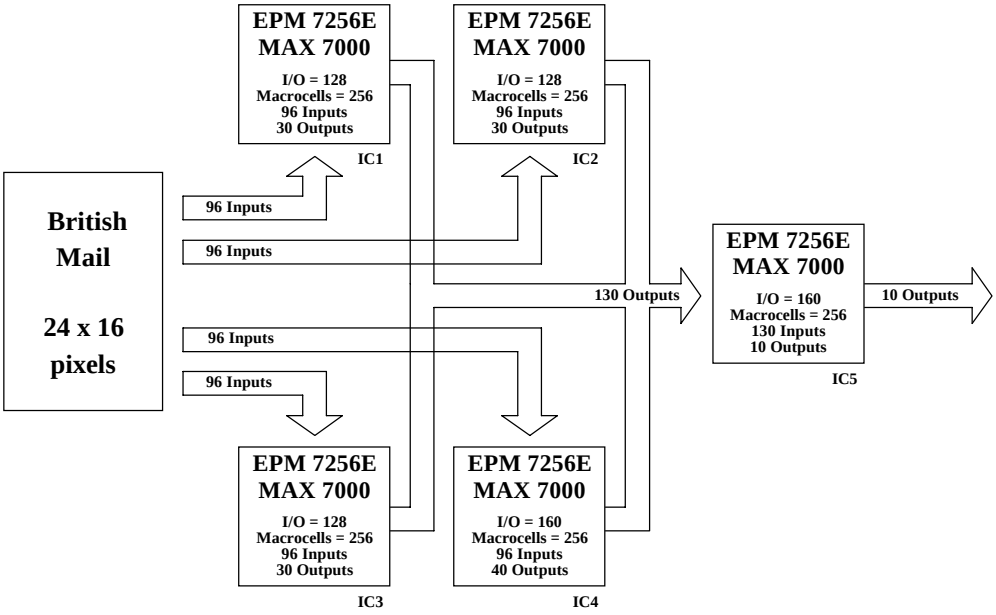
EPM 7256E
MAX 7000
I/O = 128
Macrocells = 256
96 Inputs
30 Outputs
IC1
EPM 7256E
MAX 7000
I/O = 128
Macrocells = 256
96 Inputs
30 Outputs
IC2
British
Mail
24 x 16
pixels
96 Inputs
96 Inputs
96 Inputs
96 Inputs
130 Outputs
EPM 7256E
MAX 7000
I/O = 160
Macrocells = 256
130 Inputs
10 Outputs
IC5
10 Outputs
EPM 7256E
MAX 7000
I/O = 128
Macrocells = 256
96 Inputs
30 Outputs
IC3
EPM 7256E
MAX 7000
I/O = 160
Macrocells = 256
96 Inputs
40 Outputs
IC4

Figure 3