FPGA Implementation Of A Fully And Partially Connected MLP-A Dedicated Approach

Er.Jnana Ranjan Tripathy[1], Dr.Hrudaya Kumar Tripathy[2] and Dr.Maya Nayak[3]

[1] Department of Computer Science & Engineering,, Biju Pattnaik University of Technology, Orissa Engineering College Bhubaneswar, Odisha-752050, India

**[2] School of Computing & Technology,
Asia Pacific University College of Technology & Innovation (UCTI)
Bukit Jalil, Kuala Lumpur, 57000, Malasiya**

[31] Department of Information Technology, Biju Pattnaik University of Technology, Orissa Engineering College Bhubaneswar, Odisha-752050, India

Abstract

In this work, we present several hardware implementations of a standard MultilayerPerceptron (MLP) and a modified version called eXtended Multi-Layer Perceptron (XMLP). This extended version is an MLP-like feed-forward network with two-dimensional layers and configurable connection pathways. The interlayer connectivity can be restricted according to well-defined patterns. This aspect produces a faster and smaller system with similar classification capabilities. Furthermore the software version of the XMLP allows configurable activation functions and batched back propagationwith different smoothing-momentum alternatives.

The hardware implementations have been developed and tested on an FPGA prototyping board. The designs have been defined using two different abstraction levels: register transfer level (VHDL) and a higher algorithmic-like level (Handel-C). We compare the two description strategies. Furthermore we study different implementation versions with diverse degrees of parallelism. The test bed application addressed is speech recognition.

Keywords:FPGA, Multi-Layer Perceptron (MLP), Artificial Neural Network (ANN),VHDL, Handel-C, activation function, discretization.

1. Introduction

An Artificial Neural Network (ANN) is an information processing paradigm inspired by the way biological nervous systems process information. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. As in biological systems, learning involves adjustments of the synaptic connections that exist between the neurons.

An interesting feature of the ANN models is their intrinsic parallel processing strategies. However, in most cases, the ANN is implemented using sequential algorithms thatrun on single processor architectures, and do not take advantage of this inherent parallelism.

Software implementations of ANNs are appearing in an ever increasing number of real-world applications OCR (Optical Character Recognition), data mining, image compression, medical diagnosis, ASR (AutomaticSpeech Recognition), etc. Currently, ANN hardware implementations and bio inspiredcircuits are used in a few niche areasin application fields with very high performance requirements (e.g. high energy physics), in embedded applications of simple hard-wired networks (e.g. speech recognition chips), and in neuromorphic approaches that directly implement a desired function (e.g. artificial cochleas and silicon retinas).

The work presented here studies the implementation viability and efficiency of ANNs into reconfigurable hardware (FPGA) for embedded systems, such as portable real-time ASR devices for consumer applications, vehicle equipment (GPS navigator interface), toys, and aidsfor disabled persons, etc.

Among the different ANN models available used for ASR, we have focused on the Multi-Layer Perceptron (MLP).

A recent trend in neural network design for large-scale problems is to split a task into simpler subtasks, each one handled by a separate module. The modules are then combined to obtain the final solution. A number of these modular neural networks (MNNs) have been proposed and successfully incorporated in different systems. Some of the advantages of the MNNs include reduction in the number of parameters (i.e., weights), faster training, improved generalization, suitability for parallel implementation, etc. In this way, we propose a modified version of the MLP called eXtended Multi-Layer Perceptron (XMLP). This new architecture considers image and speech recognition characteristics that usually make use of two-dimensional processing schemes, and its hardware implementability (silicon area and processing speed).

1.1.MLP/XMLP and speech recognition

Automatic speech recognition is the process by which a computer mapsan acoustic signal to text. Typically, speech recognition starts with the digital sampling of the voice signal. The raw waveform sampled is not suitable for direct input for a recognition system. The commonly adopted approach is to convert the sampled waveform into a sequence of feature vectors using techniques such as filter bank analysis and linear prediction analysis. The nextstage is the recognition of phonemes, groups of phonemes or words. This last stage is achieved in this work by ANNs (MLP and XMLP), although other techniques can be used, such as HMMs (Hidden Markov Models), DTW (Dynamic Time Warping), expert systems or acombination of these.

1.1.1. Multi-Layer Perceptron

The MLP is an ANN with processing elements or neurons organized in a regular structure with several layers (Figure 1.1): an input layer (that is simply an input vector), some hidden layers and an outputlayer. For classification problems, only one winning node of the output layer is active for each input pattern.
Each layer is fully connected with its adjacent layers. There are no connections between non-adjacent layers and there are no recurrent connections. Each of these connections is defined by an associated weight. Each neuron calculates the weighted sum of its inputs and applies an activation function that forces the neuron output to be high or low, as shown in Eqn. (1.1).
Zli =f(Sli);Sli =sumjwlijZ(l−1)j) (1.1)
In this equation,^{z}li is the output of the neuron ^{i} in layer ^{l},s_{li} is the weighted sum in that neuron, f is the activation function andwlij is the weight of the connection coming from neuronj in the previous layer (^{l}–1).
In this way, propagating the output of each layer, the MLP generates an output vector from each input pattern. The synaptic weights are adjusted through a supervised training algorithm called back propagation.
Different activation functions have been proposed to transform the activity level (weighted sum of the node inputs) into an output signal. The most frequently used is the sigmoid, although there are other choices such as a ramp function, a hyperbolic tangent, etc. All of these are continuous functions, with a smoothS-like waveform, that transform an arbitrary large real value to another value in a much restricted range.

1.1.2 Extended Multi-Layer Perceptron

The XMLP is a feed-forward neural network with aninput layer (withoutneurons), a number of hidden layers selectable from zero to two, and an output layer. In addition to the usual MLP connectivity, any layer can be twodimensional and partially connected to adjacent layers. As illustrated in Figure 1.2,connections come out from each layer in overlapped rectangular groups. The size of a layerl and its partial connectivity pattern are defined by six parameters in the following form: x(g_x,s_x) ×y(g_y,s_y), wherex andy are the sizes of the axes, andg ands specify the size of a group of neurons and the step between two consecutive groups, both in abscissas (g_x,s_x) and ordinates (g_y,s_y)

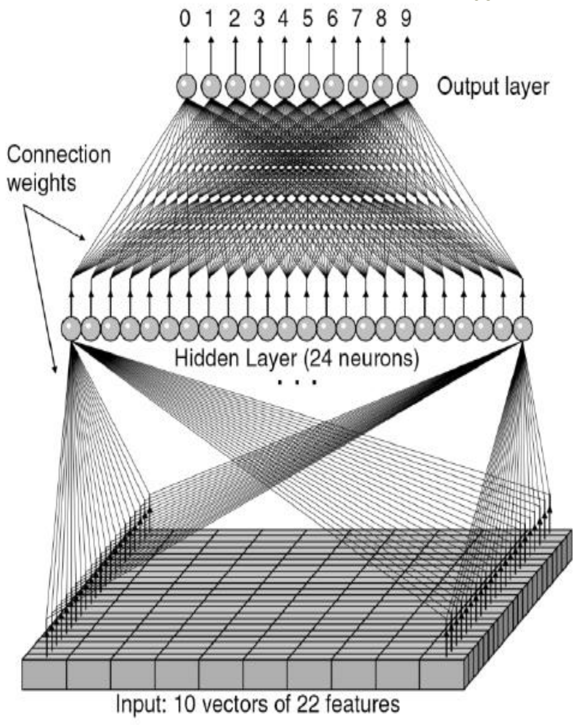

Figure 1.1Example of the MLP for isolated word recognition

A neuron i in the X-axis at layerl+1 (the upper one in Figure 1.2) is fed from all the neurons belonging to thei-the). group in the Xaxis at layerl (the lower one). The same connectivity definition is used in the Y-axis. When g ands are not specified for a particular dimension, the connectivity assumed for that dimension isg_x =x ands_x = 0, or g_y =y and s_y = 0. Thus, MLP is a particular case of XMLP whereg_x =x, s_x = 0,g_y = y and s_y = 0 for all layers.
The second dimension in each layer can be considered as a real spatial dimension for image processing applications or as the temporal dimension for time related problems. Two particularizations of the XMLP in time-related problems are the Scaly Multi-Layer Perceptron (SMLP) used in isolated word recognition, and the Time Delay Neural Network (TDNN), used in phoneme recognition.

1.1.3.Configurations Used for Speech Recognition

To illustrate the hardware implementation of the MLP/XMLP system we have chosen a specific speaker-independent isolated word recognition application. Nevertheless, many other applications require embedded systems in portable devices (low cost, low power and reduced physical size).
For our test bed application, we need an MLP/XMLP with 220 scalar data in the input layer and 10 output nodes in the output layer. The network input consists of 10 vectors of 22 components (10 cepstrum, 10 Δcepstrum, energy, Δenergy) obtained after pre-processingthe speech signal. The output nodes correspond to 10 recognizable words extracted from a multi-speaker database. After testing different architectures, the best classification results (96.83% of correct classification rate in a speaker-independent scheme) have been obtained using 24 nodes in a single hidden layer, with the connectivity of the XMLP defined by 10(4,2)×22 in the input layer and 4× [illegible]n layer.

1.2. Activation functions

For hardware implementations,we have chosen a two’s complement representation and different bit depths for the stored data (inputs, weights, outputs, etc.). In order to accomplish the hardware implementation, it is also necessary to discretize the activation function. Next, we present different activation functions used in the MLP and some details about their discretization procedure.

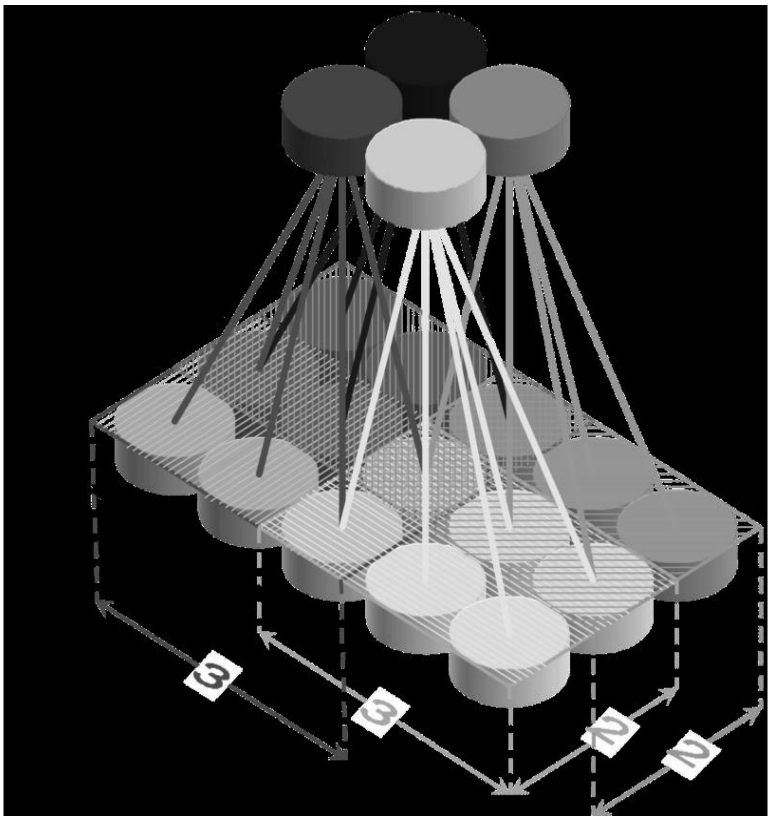

Figure 1.2. Structure of an example XMLP interlayer connectivity pattern defined by the expression 5(3, 2) ×3(2,1)

1.2.1. Activation Functions

One of the difficult problems encountered when implementing ANNs in hardware is the nonlinear activation function used after the weighted summation of the inputs in each node (Eqn.1.1). There are three main nonlinear activation functions: threshold (hard limited), ramp and various sigmoid curves. We have studied different options: classical sigmoid, hyperbolic tangent, arc tangent and ramp activation functions. In the hardware implementation, we have focused on the sigmoid activation function.
In order to achieve generality, an activation function f(x) is defined depending on three parameters: (slope atx = 0),f_{max} (maximum value of f(x)) and f_{min} (minimum value). The generic expressions for the four functions considered and their respective derivatives, needed in the backpropagation algorithm, are given in Table 1.1. An example for three particular values of the parameters is plotted in Figure 1.3.For simplicity,f_R is defined asf_{max}–f_{min}.

1.2.2 Discretization

In this contribution, learning is carried out offline using floating points,while hardware implementations use discrete variables and computations. However, classification results in hardware are very similar to the ones obtained with the software approach. For instance, in phoneme recognition application with the MLP, we obtained 69.33% correct classification with the continuous model and 69% when using the discretized model. In order to use limited precision values, it is necessary to discretize three different sets of variables: network inputs and neuron outputs (both with the same number of bits), weights and the activation function input. Each of these sets has a different range, specified by the number of bits (n) of the discrete variable and the maximum absolute value (M) of the corresponding continuous variable. The expressions for the conversion between continuous (c) anddiscrete (d) values are:

$$d = \mathrm{round}\left[c\,\frac{2^{n-2}}{2M} \right] ;\ c = d\,\frac{2M}{2^{n-2}}$$

1.2.3 Implementation characteristics of MLP with different design strategies

To extract the EDIF files, the systems have been designed using the development environments FPGA advantage, for VHDL, and DK1.1, for Handel-C. All designs have been finally placed and routed onto a Virtex-E 2000 FPGA, using the synthesis tool Xilinx Foundation 3.5i. The basic building blockof the Virtex-ECLB (Configurable Logic Block) is the logic cell (LC). Each CLB contains four LCs organized in two similar slices. An LC includes a 4-input function generator that is implemented as 4-input LUT. A LUT can provide a 16×1synchronous RAM. Virtex-E also incorporates large Embedded Memory Blocks (EMBs) (4096 bits each) that

Complement the distributed RAM memory available in the CLBs. Table 2 presents the results obtained after synthesizing the sequential and parallel versions of the MLP using Handel-C. These results are characterized by the following parameters: number and percentage of slices, number and percentage of EMBs RAM, minimum clock period, the number of clock cycles and the total time required for each input vector evaluation. The percentage values apply to the Virtex-E 2000 device.

As mentioned in Section 1.2.1, obtaining an efficient system requires a detailed analysis of the possible choices. When programming the MLP, there aredifferent options for data storage in the RAM. In orderto analyse the effects of the several techniques for distributing and storing data in RAM. Only distributed RAM for the whole designs has beenused; in (b), the weights associated with synaptic signals (large array) are stored in EMBs RAM modules, while the re

stored in a distributed mode; and finally, (c) onl

grouped in EMBs RAM modules.

Table 1.1.Activation functions and their derivatives

sigmoid	$f(x) = \frac{f_R}{1+e^{-4f_0x/f_R}} + f_{\min}$ $f_0 = \frac{1}{4}, f_{max} = 1, f_{min} = 0$ $(f_R = 1)$ Define the standard sigmoid function
	$g(x) = \frac{1}{1+e^{-4f_0x/f_R}}$, we can obtain a simple expression of f(x) as a function
	of g(x): $f'(x) = f_R g(x) = 4f_0 g(x)(1-g(x))$
	$f_0x + (f_{max}+f_{min})/2$ This function coincides with tanh $f(x) =$ $_2$ tanh$_{f^2_R}$ The sigmoid defined above.
	$f'(x) = f_0\left(1-\tanh^2\frac{2f_0}{f_R}x\right)$
arctan	$f(x) = \frac{f_R}{\pi}\arctan\left(\frac{f_0\pi}{f_R}x\right) + (f_{max}+f_{min})/2$
	$f'(x) = \frac{f_0}{1+\left(\frac{f_0\pi}{f_R}x\right)^2}$
ramp	$f(x) = \begin{cases} f_{\min} & \text{if } x \le -f_R/2f_0 \\ f_0x + (f_{max}+f_{min})/2 & \text{if } -f_R/2f_0 < x < f_R/2f_0 \\ f_{max} & \text{if } x \ge f_R/2f_0 \end{cases}$
	$f'(x) = \begin{cases} f_0 & \text{if } -f_R/2f_0 < x < f_R/2f_0 \\ 0 & \text{otherwise.} \end{cases}$

1.2.4 Discretization

One particular difficulty regarding the migration of ANNs towards hardware is that the software simulations use floating point arithmetic and either double or simple precision weights, inputs and outputs. Any hardware implementation would become unreasonably expensive if incorporating floating point operations and therefore needing to store too many bits for each weight. Fixed point arithmetic is better suited for hardware ANNs because a limited precision requires fewer bits for storing the weightsand also simpler calculations. This causes a reduction in the size of the required silicon area and a considerable speed.

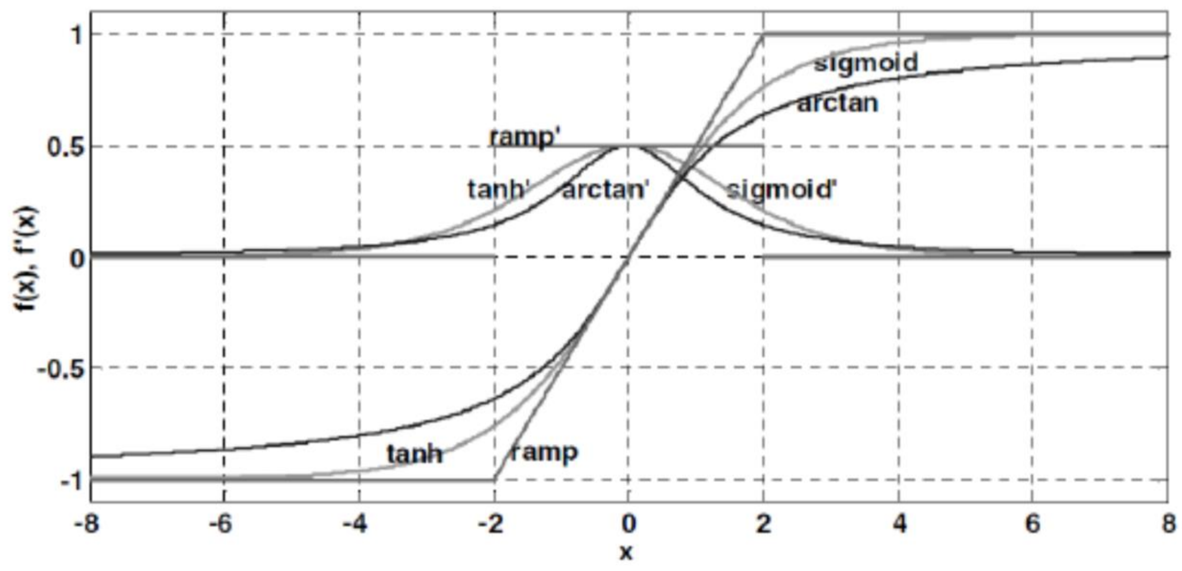

Figure 1.3. Activation functions and their derivatives with f_0= 1/2, f_{max} = 1, f_{min} =−1(f_R = 2)

1.3 Hardware implementations of XMLP

This section introduces a sequential and a parallel version of the XMLP. For reasons of clarity, only the hardware implementation for the parallel version has been described in detail. However, the implementation characteristics of both the sequential and parallel designs are presented.
In this particular XMLP implementation, the connectivity is defined by 10(4, 2)×22 in the input layer and 4×6 in the hidden layer.

1.3.1High Level Description (Handel-C)

From an algorithmic point of view, two-dimensional layers require the use of nested loops to index both dimensions. The following example shows the programmed code equivalent to that of the parallel MLP implementation described in Section 1.4.1.2. Its functionality corresponds to the weighted sum of the inputs ofeach node in the hidden layer. Note that the external loops have been parallelized using the par directive.

```
par (X=0; X<NumHiddenX; X++)
{
FirstX[X] = X*InStepX; par (Y=0; Y<NumHiddenX; Y++)
{

FirstY[Y] = Y*InStepY; Sum[X][Y] = 0 ;

}
} par (X=0; X <NumHiddenX; X++)
{ par (Y=0; Y <NumHiddenY; Y++)
{
for (XGrp=0; XGrp <InGrpX; XGrp++)
{
Mul[X][Y]           =           W[X][Y][XGrp][YGrp]*
In[FirstX[X]+XGrp][FirstY[Y]];
for (YGrp=1; YGrp<InGrpY; YGrp++) par{
Sum[X][Y] = Sum[X][Y]+Mul[X][Y];
Mul[X][Y] = W[X][Y][XGrp][YGrp]*
In[FirstX[X]+XGrp][FirstY[Y]+YGrp];}
}
Sum[X][Y] = Sum[X][Y]+Mul[X][Y];
} }
```

NumHiddenX and *NumHiddenY* are the sizes of the axes in the hidden layer. *InGrp* and *InStep* specify the size of a group of inputs and the step between two consecutive groups (in the input layer), both in abscissas (*InGrpX*, *InStepX*) and ordinates (*InGrpY*, *InStepY*).*W* is the array containing the weight values. *In* is the input array. Finally, *Sum* is a variable that stores the accumulated weighted sum of the inputs.
In order to compare the XMLP to the MLP, similar design alternatives (a), (b) and (c) to the ones considered in the MLP (Table 1.1) have been chosen.

Table 1.1. Implementation characteristics for the XMLP designs described with Handel-C. (a) Only distributed RAM. (b) Both EMBs and distributed RAM. (c) Only EMBs RAM

MLP **Evaluation Time**

Design	**Slices**	**Slices**	**RAM**	**RAM**	**(ns)**	**Cycles**	**(ms)**
a)Serial	2389	12	0	0	44.851	2566	115.087
Parallel	5754	29	0	0	47.964	143	6.858
b)Serial	1700	8	96	60	71.568	2566	183.643
Parallel	5032	26	96	60	64.270	143	9.190
c)Serial	1608	8	140	91	77.220	2566	198.146
Parallel	4923	25	147	91	64.830	143	9.271

As expected, the XMLP approaches result in systems twice as fast compared to the fully connected MLP version. This gain in speed depends on the connectivity pattern defined for the XMLP model. In the case studied, the XMLP requires only

1.3.2 Register Transfer Level Description (VHDL)

The parallel architecture of the XMLP and the MLP are similar. Keeping in mind that the connectivity pattern of the XMLP is different, only modifications related to this feature need to be made, as in Figure 1.4.

Since each hidden neuron is only connected to 88 input values (4×22), the global input RAM module with 220 MLP inputs has been replaced by 24 local RAM modules. These local modules store the 88 necessary input values for each functional unit. As each functional unit only computes 88 input values, local weight RAMs can be reduced to 112-word RAM modules for the first ten units (that also compute the output layer), and 88-word RAM modules for the rest.

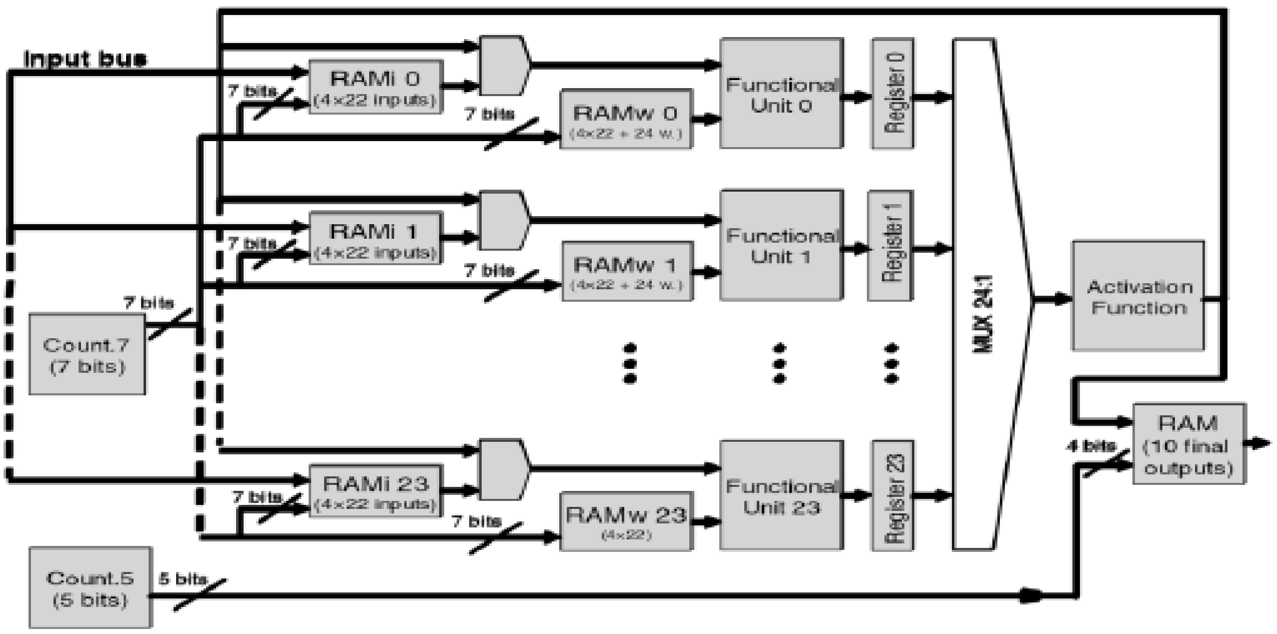

Figure 1.4. Structure of the parallel XMLP

Table 1.2 shows the implementation characteristics obtained after synthesizing both sequential and parallel versions of the XMLP using VHDL. This design corresponds to the option (b) described inTable 1.1

. **MLP** **Evaluation Time**

Design	**Slices**	**Slices**	**RAM**	**RAM**	**(ns)**	**Cycles**	**(ms)**
Serial	267	2	11	6	37.780	2478	93.618
Parallel	1747	9	49	30	53.752	152	8.170

The performance improvement of the XMLP compared to the MLP is similar to that described for the Handel-C approaches.

1.4. Conclusions

We have presented an FPGA implementation of fully and partially connected MLP-like networks for a speech recognition application. Both sequential and parallel versions of the MLP/XMLP models have been described using two different abstraction levels: register transfer level (VHDL) and a higher algorithmic-like level (Handel-C). Results show that RTL implementation produces more optimized systems. However, one of the main advantages of the high level description is the reduction of design time. The Handel-C design is completely defined with less than 100 code lines.

In both (VHDL and Handel-C) described systems, the parallel versions lead to approaches 20 times faster on average for the MLP, and around 18 times faster for the XMLP. This speed-up corresponds to the degree of parallelism (24 functional units). Therefore, it depends on the number of hidden neurons that are computed in parallel.

Finally on comparing the XMLP approaches (Tables 1.2 and 1.3) to the MLP ones (Tables 1.2 and 1.3), we see that the XMLP computes faster than the MLP. In the best case, it reduces the computation time from 13.7 to 6.9 microseconds for the parallel version. The advantages of XMLP are due to the partial connectivity patterns, which reduce the number of multiplications from 5520, with a fully connected configuration (MLP), to 2352 with the XMLP configured as described in Section 1.2.3. It can also be observed that XMLP connectivity reduces the RAM storagerequirements, once more because it requires less connection weights to be stored.

For the speech recognition application we obtain a speaker-independent correct classification rate of 96.83% with a computation time of around 14-16 microseconds per sample. This amply fulfilsthe time restrictions imposed by the application. Therefore, the implementation can be seen as a low-cost design where the whole system, even the parallel version, would fit into low-cost FPGA device. The system could be embedded in a portable speech recognition platform for voice-controlled systems.

A pipeline processing scheme taking one neural layer in each stage would lead to a faster approach. The processing bottleneck is imposed by the maximum neural fan-in, 220 in a hidden node, because of the need for 220 multiplications. With a pipeline structure,we could overlap the computation time of the hidden layer with the computation time of the output layer (24 multiplications per node). This speeds up the data path by a maximum of 10%. Here we did not study the pipeline choice because our design fulfils the application requirements (portability, low-cost and computation time).

References

[1]J. Misraa and I. Sahab, "Artificial neural networks in hardware: A survey of two decades of progress," Neurocomputing, vol. 74, no. 1-3, pp. 239-255, 2010.

[2]J. Zhu and P. Sutton, "FPGA implementations of neural networks -a survey of a decade of progress," in Proceedings of the 13th International Conference on Field Programmable Logic and Applications (FPL 2003), 2003, pp. 1062-1066.

[3]S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[4]M. M. Khan, D. R. Lester, Luis A. Plana, Alexander D. Rast, X. Jin, E. Painkras, and Stephen B. Furber, "SpiNNaker: Mapping neural networks onto a massively-parallel chipmultiprocessor," In International Joint Conference on Neural Networks (IJCNN), pp. 2849-2856, 2008.

[5]Draghici S. On thecapabilities of neural networks using limited precision weights, *Neural Networks*, **15**, 2002, no. 3, pp. 395-414.

[6]Fiesler E. and Beale R. *Handbook of Neural Computation*,, IOP Publishing Ltd and Oxford University Press, 1997.

[7]Ienne P. Cornu T. and Gary K. Special-Purpose Digital Hardware for Neural Networks: An Architectural Survey. *Journal of VLSI Signal Processing*, **13**, 1996, pp. 5-25.

[9]Mentor Graphics, http://www.mentorg.com/

[10]Xilinx, http://www.xilinx.com

[11]Celoxica, http://www.celoxica.com/

About The Author

Er.Jnana Ranjan Tripathy[1]
Pusruing PhD in Centurion University of Technology & Management in "ANN Implementation in Embedded Systems"
M.Tech in Computer Science,Berhampur University
B.Tech in Information Technology, BPUT
Currently working in Orissa Engineering College, Odisha
Worked at Centurion University previously.
Member of IACSIT

Dr.Hrudaya Kumar Tripathy[2]
Ph.D in Computer Science from Berhampur University.
M.Tech in CSE from IIT, Guwahati
B.Tech (Ceramic Technology) from IIC (CG&CRI), Kolkatta
School of Computing & Technology,
Asia Pacific University College of Technology & Innovation
Bukit Jalil, Kuala Lumpur
Published around 20 No.(s) of research papers in reputed international referred journals & IEEE conferences. Technical reviewer and member of technical committee of many International conferences.
Received many certificates of merits and highly applauded in presentation of research papers at International conferences of different Asian countries (Thailand, Singapore, Hong Kong).
Member of International Association of Computer Science and Information Technology (IACSIT), Singapore,
Member of IEEE, India Chapter.

Dr.Maya Nayak[3]
Published around 28 No.(s) of research papers in reputed international referred journals & IEEE conferences. Technical reviewer and member of technical com ny International conferences.