Mapping Fpga To Field Programmableneural Network Array(Fpnna)

[1],H Bhargav, [2], Dr. Nataraj K. R
[1],Assistant Professor,.Vidyavardhaka College of Engineering, Mysore, Karnataka, India.
[2],Professor,.SJB Institute of Technology, Bangalore, Karnataka, India.

Abstract

My paper presents the implementation of a generalized back-propagation multilayer perceptron (MLP) architecture, on FPGA, described in VLSI hardware description language (VHDL). The development of hardware platforms is not very economical because of the high hardware cost and quantity of the arithmetic operations required in online artificial neural networks (ANNs), i.e., general purpose ANNs with learning capability. Besides, there remains a dearth of hardware platforms for design space exploration, fast prototyping, and testing of these networks. Our general purpose architecture seeks to fill that gap and at the same time serve as a tool to gain a better understanding of issues unique to ANNs implemented in hardware, particularly using field programmable gate array (FPGA).This work describes a platform that offers a high degree of parameterization, while maintaining generalized network design with performance comparable to other hardware-based MLP implementations. Application of the hardware implementation of ANN with back-propagation learning algorithm for a realistic application is also presented.

Index Terms: Back-propagation, field programmable gate array (FPGA), hardware implementation, multilayer perceptron, neural network, NIR spectra calibration, spectroscopy, VHDL, Xilinx FPGA.

1. Introduction

In recent years, artificial neural networks have been widely implemented in several research areas such as image processing, speech processing and medical diagnoses. The reason of this widely implementation is their high classification power and learning ability. At the present time most of these networks are simulated by software programs or fabricated using VLSI technology [9]. The software simulation needs a microprocessor and usually takes a long period of time to execute the huge number of computations involved in the operation of the network. Several researchers have adopted hardware implementations to realize such networks [8]&[12]**.**This realization makes the network stand alone and operate on a real-time fashion. Recently, implementation of Field Programmable Gate Arrays (FPGA's) in realizing complex hardware system has been accelerated [7]**.**Field programmable gate arrays are high-density digital integrated circuits that can be configured by the user; they combine the flexibility of gate arrays with desktop programmability. An ANN's ability to learn and solve problems relies in part on the structural Characteristics of that network. Those characteristics include the number of layers in a network, the number of neurons per layer, and the activation functions of those neurons, etc. There remains a lack of a reliable means for determining the optimal set of network characteristics for a given application. Numerous implementations of ANNs already exist [5]–[8], but most of them being in software on sequential processors [2]. Software implementations can be quickly constructed, adapted, and tested for a wide range of applications. However, in some cases, the use of hardware architectures matching the parallel structure of ANNs is desirable to optimize performance or reduce the cost of the implementation, particularly for applications demanding high performance [9], [10]. Unfortunately, hardware platforms suffer from several unique disadvantages such as difficulties in achieving high data precision with relation to hardware cost, the high hardware cost of the necessary calculations, and the inflexibility of the platform as compared to software.In our work, we have attempted to address some of these disadvantages by implementing a field programmable gate array (FPGA)-based architecture of a neural network with learning capability because FPGAs are high-density digital integrated circuits that can be configured by the user; they combine the flexibility of gate arrays with desktop programmability.Their architecture consists mainly of: Configurable Logic Blocks (CLB's) where Boolean functions can be realized, Input output Blocks (IOB's) serve as input output ports, and programmable interconnection between the CLB's and IOB's.

2. Motivation

Features of ANN support evaluation implementations of different implementations of networks by changing parameters such as the number of neurons per layer, number of layers & the synaptic weights. ANNs have three main characteristics: parallelism, modularity & dynamic adaptation. Parallelism means that all neurons in the same layer perform the computation simultaneously. Modularity refers to the fact that neurons have the same structural architecture. It is clear from these characteristics that FPGAs are well tailored to support implementation of ANNs, since it has a regular structure based on a matrix of parallel configurable units.Implementations in Application Specific Integrated circuits (ASICs) lack flexibility for evaluating the performance of different implementations. This deficiency can be overcome by using Programmable Logic Devices (PLDs) such as FPGAs. FPGAs provide high performance for parallel computation & enhanced flexibility (if compared with ASICs implementation) & are the best candidates for this kind of hardware implementations. If we mount ANN on FPGA, design should be such that there should be a good balance between the response & area restrictions of ANN on FPGA.FPGAs are programmable logic devices that permit the implementation of digital systems. They provide arrays of logical cells that can be configured to perform given functions by means of configuring bit stream. An FPGA can have its behavior redefined in such a way that it can implement completely different digital systems on the same chip. Despite the prevalence of software-based ANN implementations, FPGAs and similarly, application specific integrated circuits (ASICs) have attracted much interest as platforms for ANNs because of the perception that their natural potential for parallelism and entirely hardware-based computation implementation provide better performance than their predominantly sequential software-based counterparts. As a consequence hardware-based implementations came to be preferred for high performance ANN applications [9]. While it is broadly assumed, it should be noted that an empirical study has yet to confirm that hardware-based platforms for ANNs provide higher levels of performance than software in all the cases [10]. Currently, no well defined methodology exists to determine the optimal architectural properties (i.e., number of neurons, number of layers, type of squashing function, etc.) of a neural network for a given application. The only method currently available to us is a systematic approach of educated trial and error. Software tools like MATLAB Neural Network Toolbox [13] make it relatively easy for us to quickly simulate and evaluate various ANN configurations to find an optimal architecture for software implementations. In hardware, there are more network characteristics to consider, many dealing with precision related issues like data and computational precision. Similar simulation or fast prototyping tools for hardware are not well developed.

Consequently, our primary interest in FPGAs lies in their reconfigurability. By exploiting the reconfigurability of FPGAs, we aim to transfer the flexibility of parameterized software based ANNs and ANN simulators to hardware platforms. All these features of ANN & FPGAs have made me think about giving a hardware(FPGA) platform for ANN. Doing this, we will give the user the same ability to efficiently explore the design space and prototype in hardware as is now possible in software. Additionally, with such a tool we will be able to gain some insight into hardware specific issues such as the effect of hardware implementation and design decisions on performance, accuracy, and design size.

3. PREVIOUS WORKS

In the paper published by Benjamin Schrauwen,1, Michiel D'Haene 2, David Verstraeten 2, Jan Van Campenhout in the year 2008 with the title Compact hardware Liquid State Machines on FPGA for real-time speech recognition have proposed that real-time speech recognition is possible on limited FPGA hardware using an LSM. To attain this we first explored existing hardware architectures (which we reimplemented and improved) for compact implementation of SNNs. These designs are however more than 200 times faster than real-time which is not desired because lots of hardware resources are spend on speed that is not needed. We present a novel hardware architecture based on serial processing of dendritic trees using serial arithmetic. It easily and compactly allows a scalable number of PEs to process larger net- works in parallel. Using a hardware oriented RC design flow we were able to easily port the existing speech recognition application to the actual quantized hardware architecture. For future work we plan to investigate different applications, such as autonomous robot control, large vocabulary speech recognition, and medical signal processing, that all use the hardware LSM architectures presented in this work, but which all have very different area/speed trade-offs. Parameter changing without resynthesis will also be investigated (dynamic reconfiguration or parameter pre-run shift-in with a long scan-chain are possibilities).In the paper published by Subbarao Tatikonda, Student Member, IEEE, Pramod Agarwal, Member, IEEE in the year 2008 with the title Field Programmable Gate Array (FPGA) Based Neural Network Implementation of Motion Control and Fault Diagnosis of Induction Motor Drive have proposed A study of fault tolerant strategy on the ANN-SVPWM VSI performed. This Strategy is based on the reconfiguration on the inverter topology after occurrence. The modified topology for the inverter is proposed. Entire system design on FPGA has been suggested which includes the programmable low-pass filter flux estimation, space vector PWM (neural network based), fault diagnosis block and binary logic block. The paper talks about the fault feature extraction and classification and then how the neural networks can be build on the FPGA. Digital circuits models for the linear and log sigmoid been discussed. This work clearly gives the observer no slight change in operation with any fault in one leg. Study suggests that feature extraction is a challenging research topic still to be exploited.

Still this work has many prospects in multilevel inverters, where the better operating algorithms can be proposed with increase in the level of inverters. The number of redundant states is more they have to be exploited in the near future work. The system behaves as no fault in the system at all.

4. NEURAL NETWROK MODEL DESCRIPTION

There are various hardware implementations based on ASIC,DSP & FPGA.DSP based implementation is sequential and hence does not preserve the parallel architecture of the neurons in a layer. ASIC implementations do not offer reconfigurablity by the user. FPGA is a suitable hardware for neural network implementation as it preserves the parallel architecture of the neurons in a layer and offers flexibility in reconfiguration. FPGA realization of ANNs with a large number of neurons is a challenging task. Selecting weight precision is one of the important choices when implementing ANNs on FPGAs. Weight precision is used to trade-off the capabilities of the realized ANNs against the implementation cost. A higher weight precision means fewer quantization errors in the final implementations, while a lower precision leads to simpler designs, greater speed and reductions in area requirements and power consumption. One way of resolving the trade-off is to determine the "minimum precision" required. In this work we consider the weights as 8 -bit fixed-point values. Direct implementation for non-linear sigmoid transfer functions is very expensive. As the excitation function is highly nonlinear we adopt the Look Up Table (LUT) method in order to simplify function computation. The LUT is implemented using the inbuilt RAM available in FPGA IC.

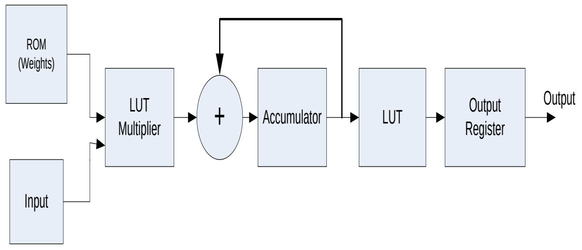


Fig 1. Neuron RTL Block Diagram

The use of LUTs reduces the resource requirement and improves the speed. Also the implementation of LUT needs no external RAM since the inbuilt memory is sufficient to implement the excitation function. The basic structure of the functional unit (neuron) that implements the calculations associated with neuron. Each neuron has a ROM, which stores the weights of the connection links between the particular neuron to the neurons of the previous layer. The multiplier performs high speed multiplication of input signals with weights from ROM. Multiplier is again implemented using an LUT multiplier. Such implementation of a multiplier needs one of the operands to be constant. In this case the other operand addresses the LUT where the result of multiplication is previously stored. Given two operands A&B with n & m bits respectively & B is constant, it is possible to implement their multiplication in LUT of 2^n. Since both multiplier & Activation Function(sigmoid function) are implemented using LUT, cost of implementation is very much reduced. We have tried comparing the model in the reference no. [15], with that of our model with respect to cost of implementation, in the conclusion section. A sixteen bit register is used to hold the weights from the ROM and the input signal from the previous layer. The whole MLP implementation is shown in Fig. 2.The network mainly consists of input registers, control unit, neurons and output register. To provide on neuron output to the next stage at each clock cycle a Multiplexer and a counter is used. The training of the network is done in software and the results loaded into hardware .Weights are updated during the training process, but remain constant during the detection process. The Register Transfer Level design of the system has been carried out using standard VHDL as the hardware description language. This language allows three different levels of description. We have chosen RTL to implement this system. The entire design process was done using the ISE development tool, from Xilinx (ISE development). The system physically implemented on Spartan-3 XC3 S4000 XILINX FPGA device.

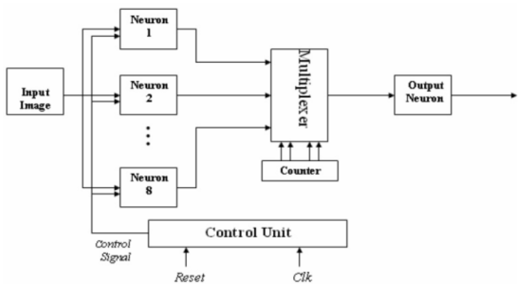


Fig2.The RTL block diagram of MLP neural network

Being a single source for hardware and software expertise, Mistral helps developers save on valuable development time and costs. The software engineers and hardware designers work together in an efficient and seamless manner providing expert design, development and support services.Mistral's professional services include hardware board design, reference designs, driver development, board support packages, embedded applications, codec and DSP algorithms across various domains. These services are delivered through a proven development process, designed specifically for embedded product development.

5. Results:

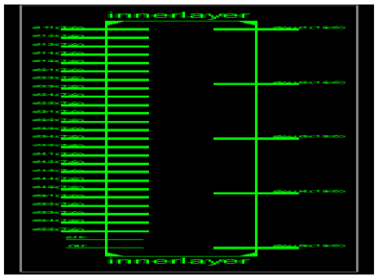


Fig 3. Inner layer top view of neural network.

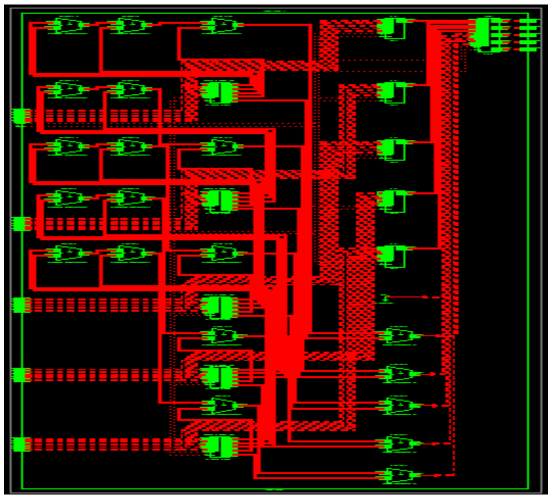
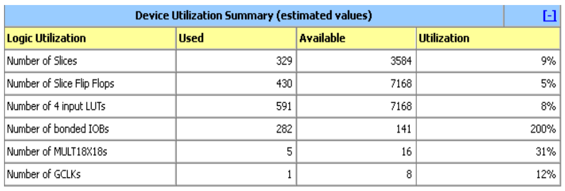


Fig 4. Inner layer RTL Schematic of Neural network.



Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	329	3584	9%
Number of Slice Flip Flops	430	7168	5%
Number of 4 input LUTs	591	7168	8%
Number of bonded IOBs	282	141	200%
Number of MULT18X18s	5	16	31%
Number of GCLKs	1	8	12%

Fig 5 . Inner layer Design summary

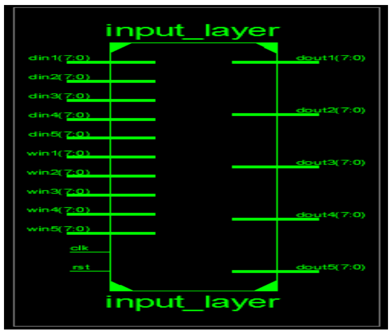


Fig 6. Input layer's top view.

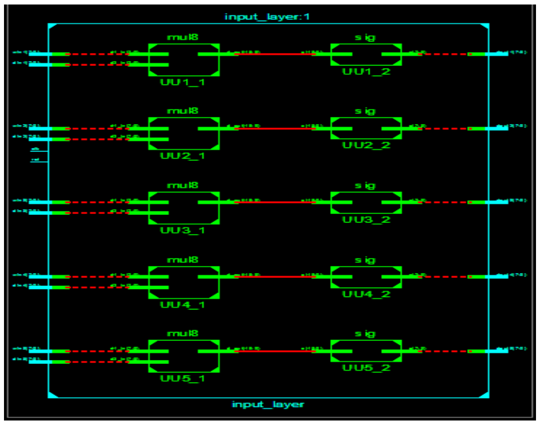
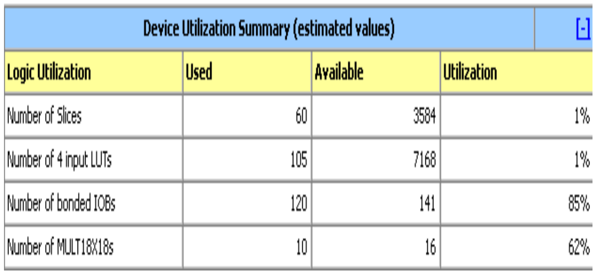
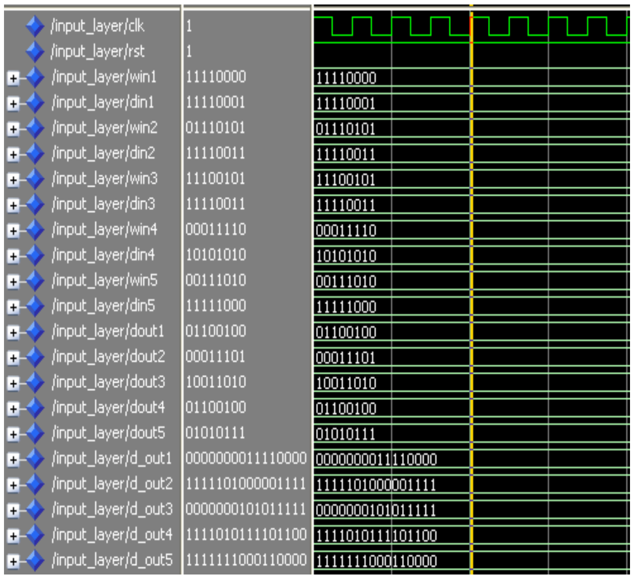


Fig 7.Input layer network of neural network.



Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	60	3584	1%
Number of 4 input LUTs	105	7168	1%
Number of bonded IOBs	120	141	85%
Number of MULT18X18s	10	16	62%

Fig 8. Input layer Design summary.



```
/input_layer/clk      1
/input_layer/rst      1
/input_layer/win1     11110000           11110000
/input_layer/din1     11110001           11110001
/input_layer/win2     01110101           01110101
/input_layer/din2     11110011           11110011
/input_layer/win3     11100101           11100101
/input_layer/din3     11110011           11110011
/input_layer/win4     00011110           00011110
/input_layer/din4     10101010           10101010
/input_layer/win5     00111010           00111010
/input_layer/din5     11111000           11111000
/input_layer/dout1    01100100           01100100
/input_layer/dout2    00011101           00011101
/input_layer/dout3    10011010           10011010
/input_layer/dout4    01100100           01100100
/input_layer/dout5    01010111           01010111
/input_layer/d_out1   0000000011110000   0000000011110000
/input_layer/d_out2   1111101000001111   1111101000001111
/input_layer/d_out3   0000000101011111   0000000101011111
/input_layer/d_out4   1111010111101100   1111010111101100
/input_layer/d_out5   1111111000110000   1111111000110000
```

Fig 9. Simulation Results for Input layer of neural network

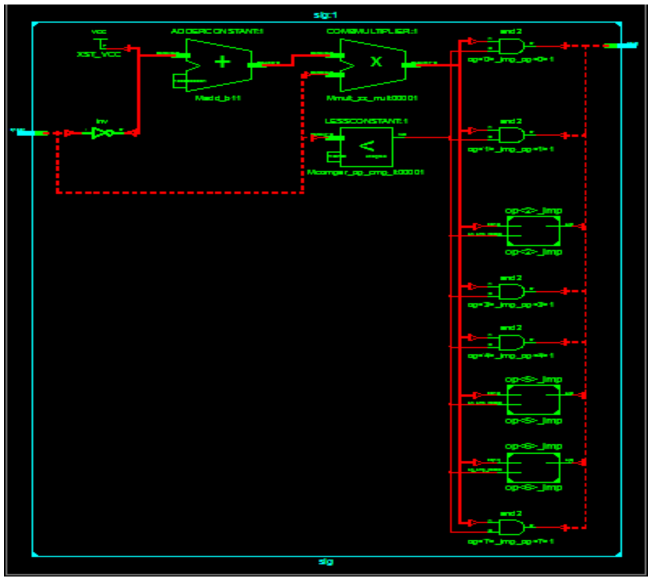
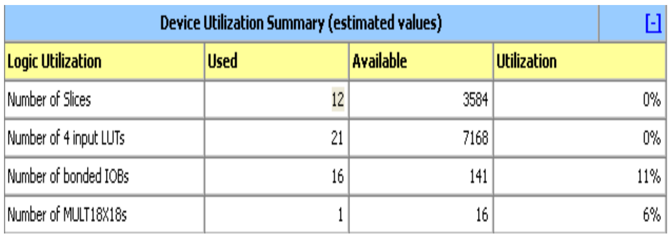


Fig 10. Sigmoid layer RTL Schematic of Neural network



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	12	3584	0%
Number of 4 input LUTs	21	7168	0%
Number of bonded IOBs	16	141	11%
Number of MULT18X18s	1	16	6%

Fig 11.Sigmoid layer Design summary

6. COMPARISION RESULTS

The device used to take Comparison results is Spartan XC3S400-5 PQ208.

Logic Utilization	Available	Proposed system used	Previous system used [15]
No of Slices	3584	12	50
No of 4 inputs LUTs	7168	21	94
No of 4 Bonded IOBs	141	16	21
Number of MULT18X18SIOs	16	1	3

Tab 1: Comparison results of Sigmoid function proposed system and previous system.

7. CONCLUSION

It is seen from the comparison table in section VI that previous system [15] used more number of Slices, LUT's and IOB's for sigmoid function. So our sigmoid function system used less number of resources as mentioned above. So we have effectively reduced the area utilization of these neural network systems. This has increased compactness & reliability of our system. In future our system permits us to achieve maximum level of optimization.Therefore one should aim at giving a hardware platform for ANN like FPGA because of the re-configurability of FPGA, we can develop the prototypes of hardware based ANNs very easily. Mapping FPGA into Field programmable Neural Network Arrays can find vast applications in real time analysis.

REFERENCES

[1] I. A. Basheer and M. Hajmeer, "Artificial neural networks: Fundamentals, computing, design, and application," *J. Microbio. Methods*, vol. 43, pp. 3–31, Dec. 2000.

[2] M. Paliwal and U. A. Kumar, "Neural networks and statistical techniques: A review of applications," *Expert Systems With Applications*, vol. 36, pp. 2–17, 2009.

[3] B. Widrow, D. E. Rumelhart, and M. A. Lehr, "Neural networks: Applications in industry, business and science," *Commun. ACM*, vol. 37, no. 3, pp. 93–105, 1994.

[4] A. Ukil, Intelligent Systems and Signal Processing in Power Engineering, 1st ed. New York: Springer, 2007

[5] B. Schrauwen, M. D'Haene, D. Verstraeten, and J. V. Campenhout, "Compact hardware liquid state machines on FPGA for real-time speech recognition," *Neural Networks*, vol. 21, no. 2–3, pp. 511–523, 2008.

[6] C. Mead and M. Mahowald, "A silicon model of early visual processing," *Neural Networks*, vol. 1, pp. 91–97, 1988.

[7] J. B. Theeten, M. Duranton, N. Mauduit, and J. A. Sirat, "The LNeuro chip: A digital VLSI with on-chip learning mechanism," in *Proc. Int. Conf. Neural Networks*, 1990, vol. 1, pp. 593–596.

[8] J. Liu and D. Liang, "A survey of FPGA-based hardware implementation of ANNs," in *Proc. Int. Conf. Neural Networks Brain*, 2005, vol. 2, pp. 915–918.

[9] P. Ienne, T. Cornu, and G. Kuhn, "Special-purpose digital hardware for neural networks: An architectural survey," *J. VLSI Signal Process.*, vol. 13, no. 1, pp. 5–25, 1996.

[10] A. R. Ormondi and J. Rajapakse, "Neural networks in FPGAs," in *Proc. Int. Conf. Neural Inform. Process.*, 2002, vol. 2, pp. 954–959.

[11] B. J. A. Kroese and P. van der Smagt, *An Introduction to Neural Networks*, 4th ed. Amsterdam, the Netherlands: The University of Amsterdam, Sep. 1991.

[12] J. Zhu and P. Sutton, "FPGA implementations of neural networks—A survey of a decade of progress," *Lecture Notes in Computer Science*, vol. 2778/2003, pp. 1062–1066, 2003.

[13] *"MATLAB Neural Network Toolbox User Guide,"* ver. 5.1, The MathWorks Inc., Natick, MA, 2006.

[14] A. Rosado-Munoz, E. Soria-Olivas, L. Gomez-Chova, and J. V. Frances, "An IP core and GUI for implementing multilayer perceptron with a fuzzy activation function on configurable logic devices," *J. Universal Comput. Sci.*, vol. 14, no. 10, pp. 1678–1694, 2008.

[15] *Rafid Ahmed Khali,l* Hardware Implementation of Backpropagation Neural Networks on Field programmable Gate Array (FPGA)