Neural Network Implementation in Hardware Using FPGAs

Suhap Sahin, Yasar Becerikli*, and Suleyman Yazici

Department of Computer Eng., Kocaeli University, Izmit ,Turkey
suhapsahin@kou.edu.tr, ybecerikli@kou.edu.tr, syazici@kou.edu.tr

Abstract. The usage of the FPGA (Field Programmable Gate Array) for neural network implementation provides flexibility in programmable systems. For the neural network based instrument prototype in real time application, conventional specific VLSI neural chip design suffers the limitation in time and cost. With low precision artificial neural network design, FPGAs have higher speed and smaller size for real time application than the VLSI design. In addition, artificial neural network based on FPGAs has fairly achieved with classification application. The programmability of reconfigurable FPGAs yields the availability of fast special purpose hardware for wide applications. Its programmability could set the conditions to explore new neural network algorithms and problems of a scale that would not be feasible with conventional processor. The goal of this work is to realize the hardware implementation of neural network using FPGAs. Digital system architecture is presented using Very High Speed Integrated Circuits Hardware Description Language (VHDL) and is implemented in FPGA chip. The design was tested on a FPGA demo board.

1 Introduction

Artificial Neural Networks (ANNs) can solve great variety of problems in areas of pattern recognition, image processing and medical diagnostic. The biologically inspired ANNs are parallel and distributed information processing systems. This system requires the massive parallel computation. Thus, the high speed operation in real time applications can be achieved only if the networks are implemented using parallel hardware architecture [1].

Implementation of ANNs falls into two categories: Software implementation and hardware implementation. ANNs are implemented in software, and are trained and simulated on general-purpose sequential computers for emulating a wide range of neural networks models. Software implementations offer flexibility. However hardware implementations are essential for applicability and for taking the advantage of ANN's inherent parallelism [2]. Specific-purpose fixed hardware implementations (i.e. VLSI) are dedicated to a specific ANN model. VLSI implementations of ANNs provide high speed in real time applications and compactness. However, they lack flexibility for structural modification and are prohibitively costly.

* Corresponding author. ybecerikli@kou.edu.tr, ybecer@ieee.org

We are interested in building a different class of hardware environment, i.e. FPGA-based reconfigurable computing environment for implementing ANNs. FPGA offer speed comparable to dedicated and fixed hardware systems for parallel algorithm acceleration, while as with a software implementation, retaining a high degree of flexibility for device reconfiguration as the application demands [3]. With the introduction of FPGAs, it is feasible to provide custom hardware for application specific computation design. The changes in designs in FPGAs can be accomplished within a few hours, and thus result in significant savings in cost and design cycle. A method of implementing a fully connected feed forward network with Xilinx FPGAs for image processing that the single processing node was partitioned into two XC3090 chips is proposed [4]. A neural associative memories implementation based RAMs and XC3090 FPGAs is reported [2].

This paper explores that how to efficiently use 32 bit floating-point numeric representation in FPGA based ANNs. By making use of the features of SpartanIIE series FPGAs. A VHDL library was designed for using ANN's on FPGAs. The library supports to the IEEE-754 standards for single-precision (32-bit) floating point arithmetic, and it is referred to fp_lib.

2 Artificial Neural Network

The concept of ANNs is emerged from the principles of brain that are adapted to digital computers. The first works of ANNs were the models of neurons in brain using mathematics rule [5]. These works show that each neuron in ANNs take some information as an input from another neuron or from an external input. This information is propagated as an output that are computed as weighted sum of inputs and applied as non-linear function.

Architectural ANNs parameters such as number of inputs per neuron and each neuron's conductivity change remarkably from application to application. Thus, for special purpose network architectures parameters must be carefully balanced for efficient implementation.

It is apparent that there are three kinds of parallelism to explain within ANNs when carefully exanimate to the data flow and structure of ANNs. The first is spatial parallelism i.e. every neuron in the same layer runs simultaneously. The second is algorithmic parallelism that is related to the formulation of the algorithm itself. In addition, computation on successive layers can be pipelined [3].

3 Field Programmable Gate Arrays and Very-High Hardware Description Language

FPGAs consist of three basic blocks that are configurable logic blocks, in-out blocks and connection blocks. Logic blocks perform logic function. Connection blocks connect logic blocks with in-out blocks. These structures consist of routing channels and programmable switches. Routing process is effectively connection logic blocks exist different distance the others [6].

FPGAs are chosen for implementation ANNs with the following reason:

- They can be applied a wide range of logic gates starting with tens of thousands up to few millions gates.
- They can be reconfigured to change logic function while resident in the system.
- FPGAs have short design cycle that leads to fairly inexpensive logic design.
- FPGAs have parallelism in their nature. Thus, they have parallel computing environment and allows logic cycle design to work parallel.
- They have powerful design, programming and syntheses tools.

The architecture of ANNs must be specified with schematic or algorithmic at first step of FPGAs based system design. When ANNs based FPGAs system design specify the architecture of ANNs from a symbolic level. This level allows us using VHDL which stands for VHSIC (Very High Speed Integrated Circuit) Hardware Programming Language [7]. VHDL allows many levels of abstractions, and permits accurate description of electronic components ranging from simple logic gates to microprocessors. VHDL have tools needed for description and simulation which leads to a lower production cost.

4 Data Representation

There are two problems during the hardware implementation of ANNs. How to balance between the need of reasonable precision (number of bit), that is important for ANN and the cost of more logic area associated with increased precision. How to choose a suitable number format that dynamic range is large enough to guarantee that saturation will not occur for a general-purpose application. So before beginning ANN's based FPGAs system design with VHDL, number format (floating point, fixed point etc.) and precision which used for inputs, weighs and activation function must be considered. This important that precision of the numbers must be as high as possible are used during training phase. Because, precision has a great impact in the learning phase [9]. However low precision is used during the propagation phase [10]. So especially in classification's applications the resulting errors will be small enough to be neglected [10,5,11].

Floating point offers the greatest amount of dynamic range, making it suitable for any application so it would be the ideal number format to use. The objective of this paper is to determinate feasibility of 32 bit floating point arithmetic in FPGAs based ANNs.

5 Application

In this section FPGA based ANN's architecture, works and system results is represented. The application is implementing fully parallel neural network in FPGA. The network is implemented in Xilinx Spartan IIE chip consist of 200000 typical gates, 2352 slices.

5.1 Arithmetic Architecture for ANN's

The first work must be, trained ANN's is mapped on FPGA in application phase. So ANN's architecture was developed using VHDL with 32 bit floating point arithmetic. Because of floating point have greatest amount of dynamic range for any applications. Unfortunately, there is currently no clear support for floating-point arithmetic in VHDL [4, 12]. As a result, a VHDL library was designed for using ANN's on FPGAs. The library supports to the IEEE-754 standards for single-precision (32-bit) floating point arithmetic, and it is referred to fp_lib. The fp_lib has tree separate library, for floating point addition fp_add, floating point subtraction fp_sub and floating point multiplication fp_mul.

The single precision floating point numeric representation supports to IEEE-754 standard shown in Figure 1.

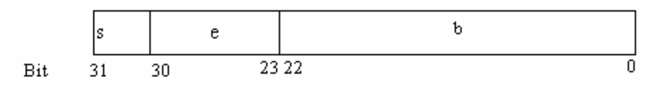

Fig. 1. 32 bit Floating Point Format

The floating point number (n) is computed by:

$$n = -1^{s} 2^{|e-127|} (1.b) \tag{1}$$

In Figure 1, sign field is referred to 's' is bit 31 and is used to specify the sign of the number. Exponent field is referred to 'e' is bits 30 down to 23 are the exponent field. The bias of 127 is used. Because of 8 bit quantity is a signed number representation. To store binary representation (b) of floating point number bits 22 down to 0 are used. The leading one in the mantissa is implicit. So the mantissa is (1.b).

5.2 Network Architecture

By using of the FPGA features hardware implementation of fully parallel ANN's is possible. In the fully parallel ANN's architecture number of multipliers per neuron equals to number of connections to this neuron and number of the full adders equals to number of connections to the previous layer mines one [9]. For example in 2-4-1 network output neuron have 4 multipliers and 3 adders. In this work a VHDL library were designed for floating point addition fp_add and floating point multiplication fp_mul. But most resources of FPGAs are used by multiplication and addition algorithm. So in fully parallel ANN's must be used low number precision (for example 8 bit). With the low number precision fully parallel network is not suitable for any application. With the using fp_lib (32 bit floating point number precision)in ANN's is suitable for any application. But the architecture has one multipliers and one adders per layer and is not full parallel because of area resource of FPGAs.

In this structure there is one multiplier and one adder per layer. The inputs from previous layer enter the layer parallel and multiplier serially with their corresponding weights. The results of multiplication are stored in their neuron area in the addition storage ROM. Multiplied value of per neuron are inputs for adder. The inputs of adder are added serially and each addition are inputs for sigmoid lookup table. The results of look up table are stored for next layer. This ANNs architecture is shown in Figure 2. In this design number of layer and number of neuron are changed easily during the working phase.

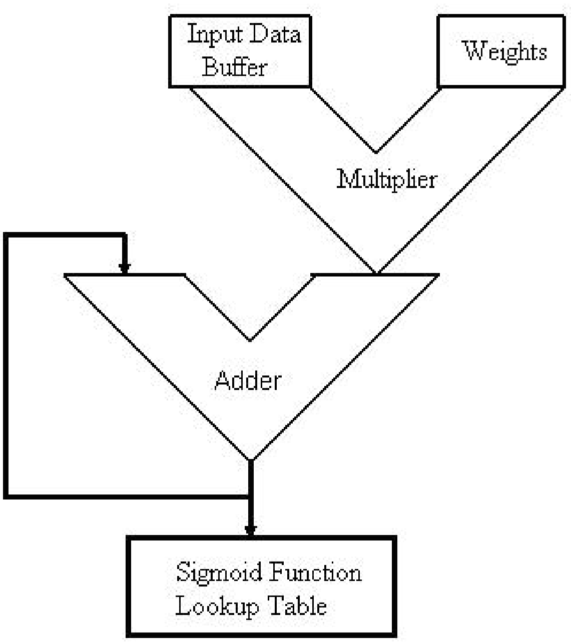

Fig. 2. Block diagram of ANNs

5.3 Modeling Tree Layer (2-3-1) ANNs

ANNs consist of input layer, one hidden layer and output layer as shown in Table 1. Sigmoid function is used as an activation function. Sigmoid function input vector consist of 100 value from -10 to 10 by chosen 0.2 step size. Results of sigmoid function are stored in the ROM

Weights are using this application are shown in Table 1.

Table 1. Weights Used in the Application

w111	w121	w131	w112	w122	w132
0.5	1	0.5	1	0.5	1
w211	w212	w213			
0.5	1	0.5			

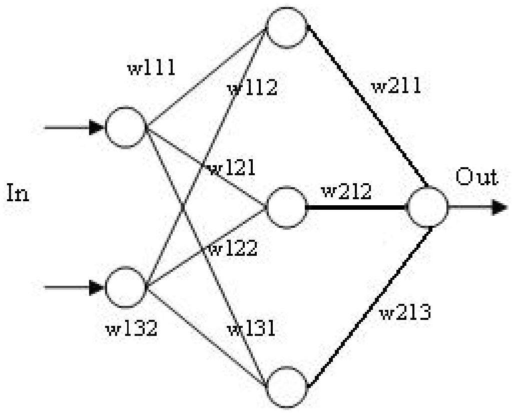

Fig. 3. A three layer MLP

5.4 Tree Layer (2-3-2) ANNs Architecture

In our design the external inputs entered the first layer serially. Input value and tree control signal that are start signal (1 bit), finish signal (1 bit) and neuron count signal (4 bit) must be entered. Before entered input value start signal must be set and after

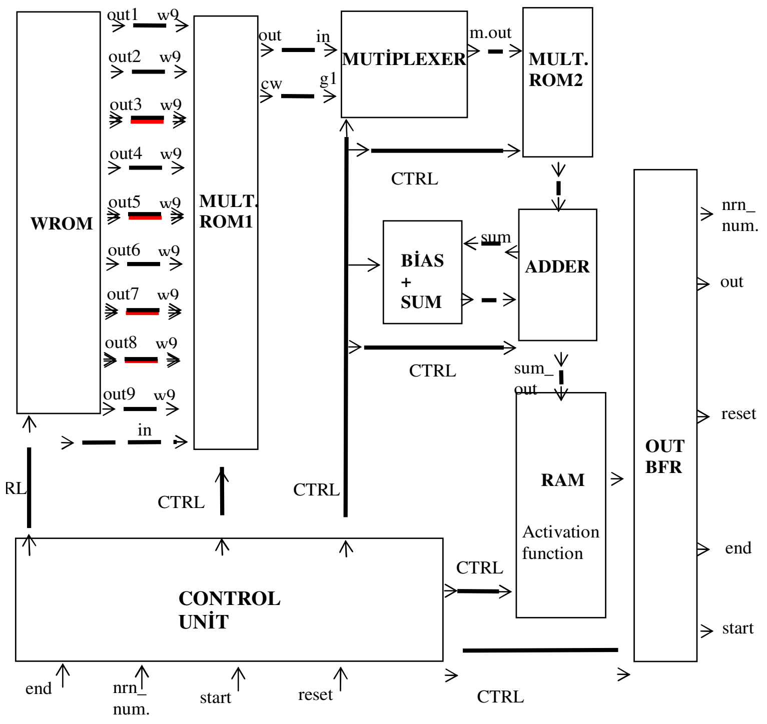

Fig. 4. Block diagram of the reconfigurable ANN

the entered input value finish signal must be set. If finish signal must be set system calculate entered input value number. So, the input number depends on the ones. Weight number is multiplied of input number in the BFR and neuron number.

Then first input, its corresponding and weights are stored in MULT_ROM1 and they are multiplied serially. The results are stored in area of first neuron in MULT_ROM2. The same process is repeat for other inputs. The value and bias in the same neuron area are added serially. The results are input for sigmoid look-up table. Look-up table outputs are stored OUT_BFR. The value that is this layer outputs must be next layer inputs. This working is controlled by control unit. Control unit controls time and makes several signals (for example enable signal) for other unit (see .Fig. 4)

6 Implementation Results

Digilentic demo board is used for implementation. The board has Xlinx Spartan II 2s200epq208-6 and 50 MHz clock. Spartan II chip has 2352 slices and 14 block RAM. VHDL libraries that it is referred to fp_lib were designed for using ANN's on FPGAs. The *fp_lib* has tree separate library are shown in Table 2. The comparison of FPGA based tree layer (2-3-1) ANNs and software based tree layer (2-3-1) ANNs are shown in Table 3 with inputs g1 = 0.5 and g2 = -0.25.

Table 2. Summary of custom arithmetic VHDL libraries

HDL Design	**Description**
fp_lib	IEEE 32-bit single precision floating point library
fp_mul	IEEE 32-bit single precision floating point piplined parallel multiplier
fp_add	IEEE 32-bit single precision floating point piplined parallel adder
Log_rom	IEEE 32-bit single precision floating point Single Port Block Memory

Table 3. Comparison of FPGA base ANNs and software based ANNs

	Software based ANNs	**FPGA based ANNs**	**ERROR**
g1=0.5 g2=-0.25	2.3274321322	2.268109	0.059323

7 Conclusions

In general, it is shown that implementation of neural networks using FPGAs. The resultant neural networks are modular, compact, and efficient and the number of neurons, number of hidden layers and number of inputs are easily changed.

The choice of dynamic range, precision and arithmetic hardware architecture used in neural networks application has a direct impact on the processing density achieved. Using suitable precision arithmetic design, one can be achieve adequately high speed and small size for real time ANNs implementations.

However this study shows that FPGAs are versatile devices for implementing many different applications. The VHDL-FPGA combination is shown to be a very powerful embedded system design tool, with low cost, reliability, and multi-faceted applications. As FPGAs allow the hardware design via configuration software control, the improvement of circuitry design is just a matter of modifying, debugging and downloading the new configuration code in a short time.

References

[1] POLIAC M., ZANETTI J., SALERNO D. 1993. Performance Mesuraments of Seismocardiogram Interpretation Using Neural Networks, Computer in Cardiology, IEEE Computer Society, pp 573-576

[2] RUCKET, U., FUNKE, A. and PINTASKE, C. 1993 Acceleratorboard for Neural Associative Memories, Neurocomputing, Vol.5, No.1, pp 39-49.

[3] ZHU, J., GUNTHER, B.K.,1999. Towards an FPGA Based Reconfigurable Computing Environment for Neural Network Implementations, Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN'99). In IEE Conference Proceedings 470, pp.661-667, IEE.

[4] COX, C., BLANZ, W. 1992. GABGLION-A Fast Field Programmable Gate Array Implementation of a Connectionist Classifier, IEEE Journal of Solid-satate Circuits, Vol.27, No.3, pp 288-299.

[5] HAYKIN, S. 1999. Neural Networks A Comprehensive Foundation. 2nd edition, Prentice Hall Publishing, New Jersey 07458, USA, Vol.1, pp 6-7

[6] BROWN, S.D., FRANCIS, R.J., VRANESIC Z.G. 1992. Field Programmable Gate Arrays, Kluwer Academics Publishers

[7] ASHEDEN, P., J. September-October 2001, VHDL Standards, IEEE Design & Test of Computers, vol. 18, n. 6, pp. 122-123.

[8] SAVRAN A., ÜNSAL S., "Hardware Implementation of a Feed forward Neural Network Using FPGAs", The third International Conference on Electrical and Electronics Engineering (ELECO 2003), 3-7 December, Bursa, Turkey, 2003.

[9] STEVENSON, M., WEINTER, R. and WIDOW, B. 1990 Sensitivity of Feedforward Neural Networks to Weigh Errors, IEEE Transactions on Neural Networks, Vol.1, No 2, pp71-80

[10] BLAKE, J.J., MAGUIRE, L.P., MCGINNITY, T.M., ROCHE, B., MCDAID, L.J. 1998. The Implementation of Fuzzy Systems, Neural Networks using FPGAs, Information Sciences, Vol. 112, pp. 151-168

[11] KRIPS, M., LAMMERT T., and KUMMERT, A. 2002, FPGA Implementation of a Neural Network for a Real-Time Hand Tracking System, Proceedings of first IEEE Internaional Workshop on Electronic Design, Test and Applications.

[12] YU X., DENI D. 1994. Implementing Neural Networks In FPGAs, The Institution of Electrical Engineers, IEE published, Savoy Place, London WC2R 0BL, UK.