FPGA Realization of Hyperbolic Function

Subha Sri Lakshmi Thiruveedhi[1] and P. Viswanath[2]
[1]Asst. Professor, CVR College of Engineering/ ECE Department, Hyderabad, India
Email: rupashubha@gmail.com
[2] Professor, CVR College of Engineering/ ECE Department, Hyderabad, India
Email: panchagnulaviswanath@gmail.com

Abstract: **Machine Learning (ML) is applied in many real world applications. Artificial Neuron (AN) is the main building block of Artificial Neural Networks (ANN) which is the backbone of ML. ML algorithms for different applications like image and speech recognition etc., speech with large data are implemented on workstations with GPUs. However, the power of ML techniques for real time applications like monitoring, control and diagnostics, typical in power and process plant is still in nascent stage. Most of these applications require FPGA realization of ANNs. Realization of AN with both sigmoid and hyperbolic tangent activation functions along with results is reported in this paper. Design of building blocks consisting of Floating Point Unit (FPU) and Function approximation (FA) are presented. Implementation using Verilog HDL along with results of accuracy with simulated inputs and gate count and timing diagrams are presented.**

Index Terms: **Artificial Neuron, Floating Point Unit (FPU), Exponential Representation, Verilog HDL, Xilinx ISE.**

I. Introduction

Rapid progress in Machine Learning in the last decade is opening opportunities for application of Deep Neural Networks and Machine Learning in real time control, monitoring and diagnostics of power and process applications. Enthalpy based real time control and sensor data reconciliation of power plants require repeated usage of steam and gas properties along with their Jacobian and Hessians. Steam and gas properties are, generally, represented in the form of complex Gibbs free energy or Helmholtz free energy equations. The properties are represented at discrete values and obtaining the function values and their derivatives is carried out by repeated calling of the above complex functions. This is computationally prohibitive. Krishna Dutt et al [1,2] reported a compact ANN representation of these properties for simultaneous obtaining of the properties and their derivatives along with their application to real time data reconciliation of power plant sensor data. Arash Ardakani et al [3] proposed an integer form of stochastic computation for an efficient implementation of a Deep Neural Network based on integral stochastic computing. The proposed architecture was demonstrated on a Virtex7 FPGA, resulting in 45% and 62% average reductions in area and latency. K.P. Sridhar et al [4] studied a testing method for VLSI based single neuron architecture with multiple inputs and one output for bench marking against ISCAS85-C17 circuit. Experimental results were reported on XILINX Spartan III FPGA. Bapuray.D. Yammenavar eta al [5] demonstrated an analog VLSI implementation with memory refresher circuits of a neural network; the design is adopted for digital operations like AND, OR and NOT. Focus of Jeremie Detrey et al [6] is on implementing floating-point elementary functions on FPGAs which is contemporary problem. They report trigonometric function approximation based on last bit accuracy. Derek Nowrouzezahrai et al [7] has presented a computationally efficient algorithm for cosine function based on Taylor's series approximation along with simulation results on Altera Stratix II FPGA. Series approximation is truncated based on accuracy in the last bit weight of the mantissas. P. Graf [8] designed and implemented a VLSI circuit for neural network for classification, particularly for image processing application, consisting of an array of 54 amplifiers with inputs and outputs interconnected through a matrix of resistive elements. All of the coupling elements are programmable resistive connection which can be turned on or off. Ranjeet Ranade [9] et al describes a technique to realize a novel digital multiplier used in Artificial Neural Network (ANN) and studied a generalized `Energy Function' for multiplier and its hardware realization by combining conventional digital hardware with a Neural Network. The design of Neurons, and the digital multiplier are described in this paper along with the simulation results. Valeriu Beiu [10] proposed a method to compute sigmoid function and its derivative in digital hardware by sum of steps and suggested that such algorithms are area-efficient. B.K. Bose [11] research was one of the earliest in floating point operation realizations in VLSI. Focus of research world over in the area of VLSI realization of ANN is on (a) efficient realization of floating point operations with real numbers (b) realization of efficient and accurate representation of functions (c) realization of artificial neuron and the feed forward ANN. Real time realization of the neural net with feed forward mode in FPGA is of recent interest to researchers. Both structural and behavioral models can be considered for implementation of ANN in FPGA. Behavioral Models and realization of an AN with different activation functions along with a floating point algebraic operator for real valued inputs, in FPGA environment is reported in this paper.

II. Artificial Neuron

Fundamental building block of Artificial Networks is Artificial Neuron, also called as Perceptron. Figure 1 describes a typical AN.

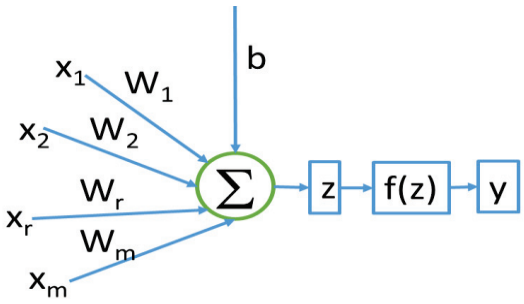

Figure 1. Structure of Artificial Neuron (AN)

AN structure consists of m inputs and their associated weights, a summer of weighted inputs and a mapping function that transforms the sum to an output. This function can be any differentiable function. Sigmoid, Tanh, ReLU are some of the widely used functions. Figure 2 shows the output of these two functions i.e., tanh(x) and sigmoid.

Equation (1) describes the functioning of an AN

$$Z = (\sum w_i x_i + b) \quad (1)$$

$$y = f(Z) \quad (2)$$

In general, an AN has a multivariate input (x_i), weight vector and a mapping function. A summer block which (Z) a weight sum y the input vector (x_i). The mapping function (f) transforms the weighted input to the output (y). The mapping function can be any analytic function of at least first order continuity. The inputs (x) are normalized to be within (-1,1) or (0,1) band an application, however, the weights can be any real number. The output of y is obtained to be within (0,1) or (-1,1) for sigmoid or tanh respectively. There are many other possible functions of which ReLu has become very popular, particularly for dealing with binary inputs, where the above two functions are useful in case of real valued inputs. ReLu is generally used in deep beleif networks. Sigmoid is more popularly used in multi layer neural networks (MLP) or deep neural networks. The structure on learning functions of an AN is based on studies of biological neuron focus in mammalian brains. These biological neurons reveal that learning is either discriminative or generative in nature. A single AN tries to emulate these two capabilities through the learning function.

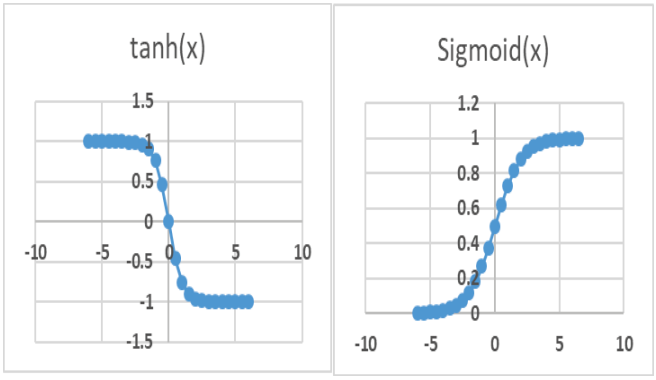

Figure 2. Activation Functions of AN

The output of these two functions is limited to [-1,1]. However, functions like ReLU, figure 3 does not limit the output to be within [-1,1]. From the above general purpose Floating Point Unit (FPU) with basic operations, multiplication, division, addition and subtraction, and a function mapping units are required for realization of an AN model in FPGA. Following describes design of these two with results obtained from the implementation in Verilog Code.

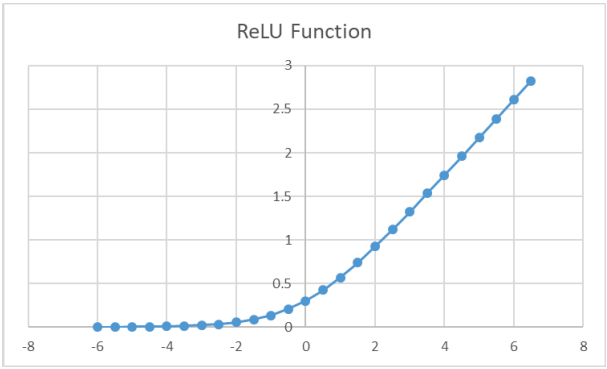

Figure 3. ReLU Activation Function.

III. Floating Point Unit

Floating point unit is defined as math coprocessor. Mainly it is designed to perform various operations such as addition, subtraction, multiplication, division, square root and bit shifting etc. It even can perform other operations such as exponential & trigonometric operations which are defined as transcendental functions. In computer architecture applications one or more floating point units can be inserted within the central processing units (CPU's). Design and realization of FPU are based on IEEE 754 standard as shown in figure 4. Generally IEEE 754 standard specifies the interchange and arithmetic equation formats and methods for binary and decimal floating point arithmetic in computer & signal processing applications. Even this standard interchanges the bit strings that may be used to exchange the floating point data in an efficient form to compact form. Real number inputs (OP1,Op2) are converted to IEEE 754 representation and algebraic operators ADD, SUB, MULTI and DIV are operated on the converted operands. Figure 9 shows the summary details of device utilization, in which highest utilization is on number of slots (slices) to the extent of 35%, which can be considered reasonable.

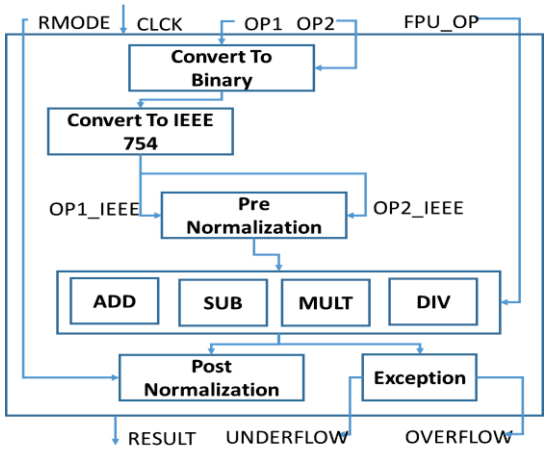

Figure 4. Block Diagram of Floating Point Unit

IV. Function Apprixiamation

Both Sigmoid and tanh functions are modelled as below for realization in FPGA.

$$\text{Sigmoid}(z) = 1.0/(1.0+e^{-z}) \quad (3a)$$

$$\tanh(z) = (e^{z} - e^{-z}) / (e^{z} + e^{-z}) \quad (3b)$$

Common to both equations (3) and (4) is the function block exponentiation of the variable z. This is approximated as

$$e^{z} = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \frac{z^5}{120} + \frac{z^6}{720} \quad (3)$$

Equation 3(a), approximates the exponentiation with a maximum error of 0.04% at the maximum input range. This is considered reasonable approximation for AN. Based on this approximation, equations (1) and (2) are realized in FPGA as shown in figures 5 and 6, respectively.

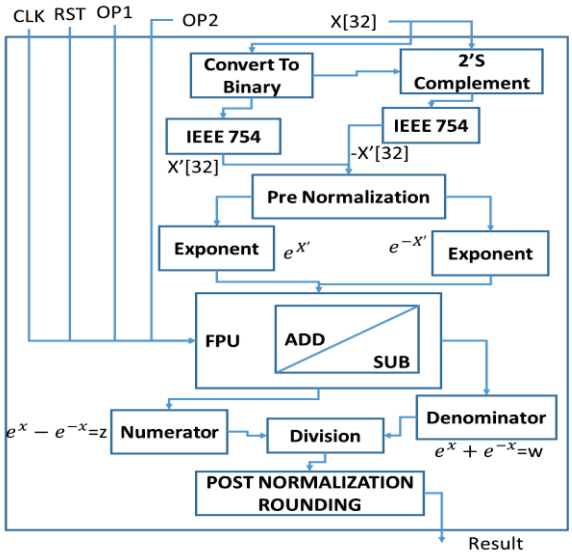

Figure 5. Internal Operational Structure of tanh Function Realization in FPGA

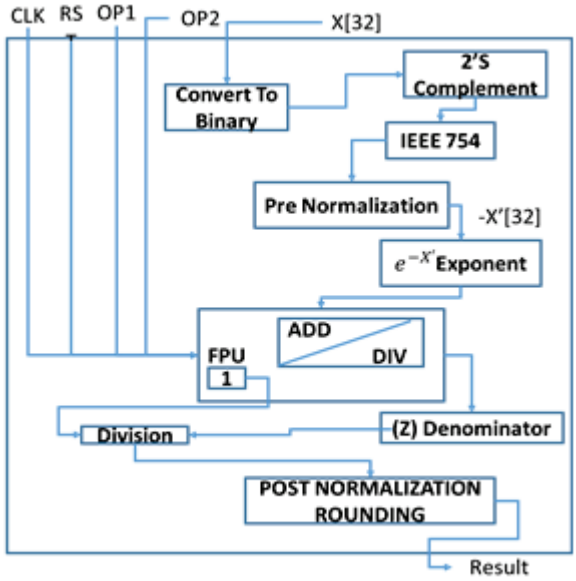

Figure 6. Internal Operational Structure of Sigmoid Function Realization in FPGA

The block level structures of tanh function and sigmoid function i.e., equation 3a and 3b are designed as shown in figures 7 and 8 respectively. The Floating point unit (FPU) is analyzed and designed by using Figure 4. The floating point operations are defined by parameter named as opcode. For example opcode = 00 indicates addition operation, 01 indicates subtraction operation, 11 indicates multiplication operation and finally 10 indicates division operation of floating point arithmetic unit. All the operands are defined in terms of floating point arithmetic and represented with IEEE 754 format.

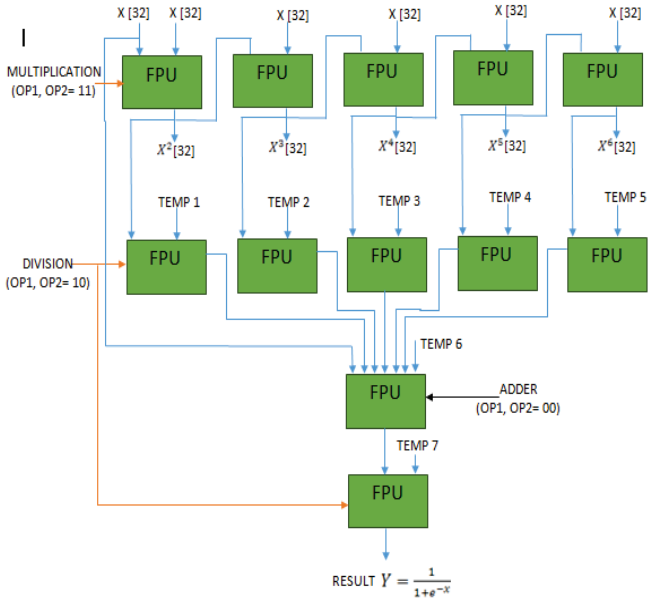

Figure 7. Block Diagram of Sigmoid Function Realization

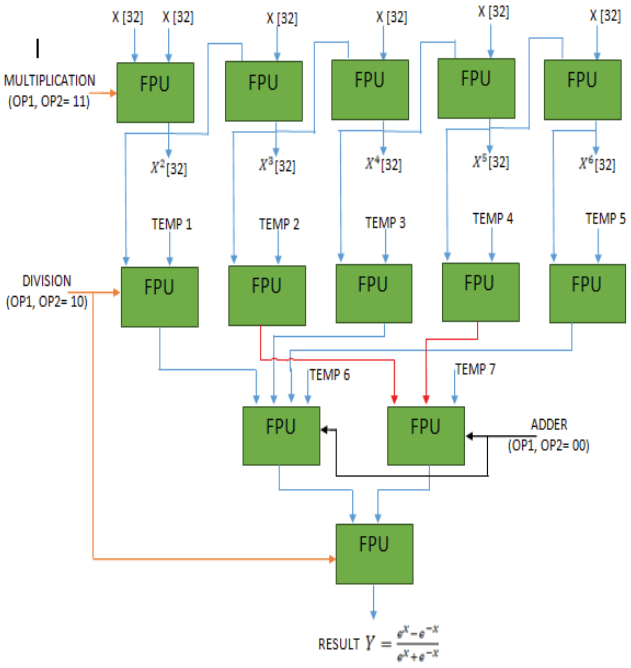

Figure 8. Block Diagram of tanh Function Realization in FPGA

All the TEMP variables in the above figures are defined as respectively, to derive the mathematical expressions such as equation (1), (2) & (3).

Temp 1= 2 -> 32'b 0000 0000 0000 0000 0000 0000 0000 0010
Temp 2= 6 -> 32'b 0000 0000 0000 0000 0000 0000 0000 0110
Temp 3= 24 -> 32'b 0000 0000 0000 0000 0000 0000 0001 1000
Temp 4= 120 -> 32'b 0000 0000 0000 0000 0000 0000 0111 1000
Temp 5= 720 -> 32'b 0000 0000 0000 0000 0000 0010 1101 0000
Temp 6 & Temp 7= 1 -> 32'b 0000 0000 0000 0000 0000 0000 0000 0001

V. Implementation & Results

All the blocks of Artificial Neuron i.e., Floating Point Unit (FPU), Tanh function and Sigmoid Function are designed using Verilog HDL and simulated using Xilinx ISE simulator of 14.7 versions. The device utilization for Floating Point Unit, Tanh and Sigmoid expressions are shown in the figures 9, 10 & 11 respectively. They are shown in terms of no. of slices & no. of Flip-flop's used in the design.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1279	3584	35%
Number of Slice Flip Flops	743	7168	10%
Number of 4 input LUTs	2375	7168	33%
Number of bonded IOBs	100	141	70%
Number of MULT18X18s	4	16	25%
Number of GCLKs	4	8	50%

Figure 9. Device Utilization for FPU in FPGA

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	2703	11440	23%
Number of Slice LUTs	12765	5720	223%
Number of fully used LUT-FF pairs	2415	13053	18%
Number of bonded IOBs	66	102	64%
Number of BUFG/BUFGCTRLs	8	16	50%

Figure 10. Device Utilization for Tanh Function in FPGA

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	2826	11440	24%
Number of Slice LUTs	14626	5720	255%
Number of fully used LUT-FF pairs	2551	14901	17%
Number of bonded IOBs	66	102	64%
Number of BUFG/BUFGCTRLs	8	16	50%

Figure 11. Device Utilization for Sigmoid Function in FPGA

Simulation Waveforms are shown in the figures 12 & 13. Figure 12 shows the simulation waveform of floating point unit with multiplication operation. Input A[31:0] is defined as 01000000001000000000000000000000, input B[31:0] is defined as 01000000101000000000000000000000 and output is observed as

O[31:0]= 01000000110010000000000000000000

Figure 12. Simulation Waveform for FPU in FPGA

Figure 13 shows the Simulation analysis of Tanh mathematical expression in which x is defined in terms of floating point arithmetic values.

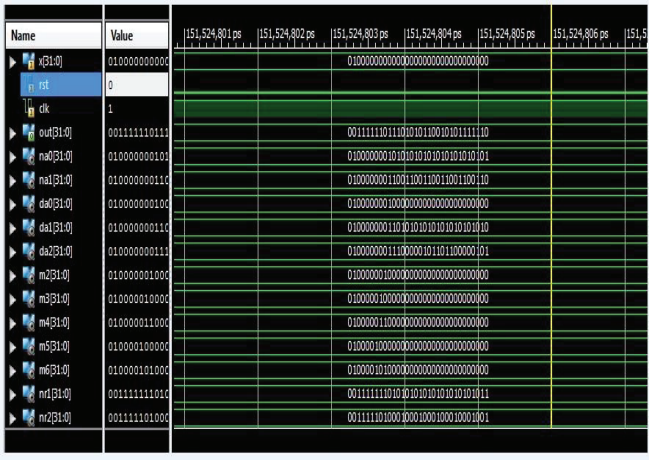

Figure 13. Simulation Waveform for Tanh Function in FPGA

Table 1 gives the details of Floating point unit, Sigmoid expression and tanh expression in terms of gate count (Number of Slices), CPU time and Power usage (in terms of watts).

All the blocks are implemented on FPGA with the respective specifications.

Family – Spartan 3
Device – XC3S400
Package – PQ208
Speed - -4/-5

TABLE I.
AREA VS CPU TIME VS POWER USAGE

Blocks	Gate Count	CPU Time	Power Usage
Floating Point Unit	2432	0.36 sec	0.016
Sigmoid Expression	2826	0.39 sec	0.019
Tanh Expression	2703	0.41 sec	0.025

VI. CONCLUSIONS

Realization of Hyperbolic Function in FPGA is demonstration through Verilog code in Xilinx, ISE 14.7, and simulation tool. Popularly known two activation functions, generally used with real number feature vector, are realized with an overall device utilization of 35%. The design is based on behavioral model. However, it is also possible to realize the Artificial Neuron based on structural model for improving the efficiency of device utilization. The function approximation is reported with an accuracy of 0.5. With an overall cycle time 0.41 seconds with a clock period of 100 MHz. The work reported is considered as building block for realizing a deep neural network of any number of hidden layers with each layer having a finite number of Artificial Neurons. The work can easily be extended to cover other activations functions like ReLU, used in deep belief networks.

REFERENCES

[1] "Neural Nets for Thermodynamic properties", R.V.S Krishna Dutt, J Krishnaiah, Proceedings of the First International conference on Computational Intelligence and Informatics, Advances in Intelligent Systems and Computing, vol 507, Spinger, Singapore., 2017

[2] "System and Neural Net Methods for Signal Validation of Power Plant Measurement Data", R.V.S.Krishna Dutt, J.Krishnaih, Patent No.160123RD

[3] "VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing", Arash Ardakani,, François Leduc-Primeau, Naoya Onizawa, Takahiro Hanyu, Warren J. Gross, arXiv:1509.08972v2 [cs.NE] 24 Aug 2016

[4] "Design and Implementation of Neural Network Based circuits for VLSI testing", K.P. Sridhar, B. Vignesh, S. Saravanan, M. Lavanya and V. Vaithiyanathan, World Applied Sciences Journal 29 (Data Mining and Soft Computing Techniques): 113-117, 2014.

[5] "Design and Analog VLSI Implementation of Artificial Neural Network", Bapuray.D.Yammenavar, Vadiraj.R.Gurunaik,Rakesh.N.Bevinagidad Vinayak.U.Gandage, International Journal of Artificial Intelligence & Applications (IJAIA), Vol.2, No.3, July 2011.

[6] "Floating Point Trigonometric Functions for FPGAS", Jeremie Detrey, Florent de Dinechin, IEEE, 2007.

[7] "Efficient Double-Precision Cosine Generation", Derek Nowrouzezahrai et al, Department of Electrical and Computer Engineering University of Waterloo, 2006.

[8] "VLSI implementation of a Neural Network Model", Hans P. Graf, Lawrence D. Jackel, and Wayne E. Hubbard AT&T Bell Laboratories, IEEE Computer, 1988

[9] "VLSI implementation of artificial neural network based digital multiplier and adder", Ranjeet Ranade, Sanjay Bhandari, A.N.Chandorkar, VLSID '96 Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication, January 03 - 06, 1996

[10] "Close Approximations of Sigmoid Functions by Sum of Steps for VLSI Implementation of Neural Networks", Valeriu Beiu, Jan Peperstraete, Joos Vandewalle, Rudy Lauwereins, The Scientific Annals, Section: Informatics, vol. 40 (XXXX), no. 1, 1994.

[11] "VLSI Design Techniques for Floating Point Computation", B.K.Bose, Ph.D Thesis, UC Berkeley, 1988.