HIGH SPEED AND LOW AREA FIR FILTER IMPLEMENTATION BASED ON SHIFT AND ADD MULTIPLIER DESIGN FOR MACHINE LEARNING APPLICATIONS

#1Prof.Ruckmani Divakaran, #2Arshiya Taj A., #3Nikhath Suman, #4Sneha S., #5Sweety Lydia B.

1Dept of Electronics and Communication Engineering, Dr.T.Thimmaiah Institute of Technology, Visvesvaraya Technological University, Belagavi, Karnataka, India

hod.ece@drttit.edu.in

2Dept of Electronics and Communication Engineering, Dr.T.Thimmaiah Institute of Technology, Visvesvaraya Technological university, Belagavi, Karnataka, India

snehashiva97@gmail.com

Abstract

Today every circuit has to face the power consumption issue for both portable device aiming at large battery life and high end circuits avoiding cooling packages and reliability issues that are too complex. It is generally accepted that during logic synthesis power tracks which works well with area. This means that a larger design will generally consume more power. The multiplier is an important kernel of digital signal processors. Because of the circuit complexity, the power consumption and area are the two important design considerations of the multiplier. In this paper a High Speed & low area architecture for the shift and add multiplier is proposed. The simulation result for 8 bit multipliers & four tap Filters shows that the proposed Low Area & Delay architecture lowers the total Area & Delay when compared to the Array Multiplier and Booth Multiplier architecture based Filter. To develop the system blocks in Modelsim 6.4a and Xilinx ISE9.1i, the Spartan3 FPGA tool is used which achieves the simulation and the synthesis of the proposed multiplier. Verilog HDL is the language used for designing the proposed multiplier.

Index Terms: Finite Impulse Response(FIR), Shift and Add Multiplier

I. INTRODUCTION

Shift-and-add multiplication is similar to the multiplication performed manually. The method adds the multiplicand 'A' to itself 'B' times where 'B' denotes the multiplier. To perform the entire multiplication for getting the final product, the conventional architecture that was used for shift and add multipliers required many switching activities. So the power dissipation was more in that. By removing the sources switching activity in the older multiplier, low power architecture of multiplier can be achieved.

This also reduces the energy consumption of the accelerator which satisfies the requirements compared to the previous multipliers. Various different fixed-coefficient multipliers were proposed in the DSP domain[5],it used lookup table but in this proposed multiplier lookup table is not used

NEURAL NETWORK:

Recent study on Neural Network (NN) was showing good advancement over the previous algorithms in machine learning. Different network models, like recurrent neural network (RNN), and convolutional neural network (CNN), have been proposed for video ,image & speech process. Artificial Neural Network. ANNs were usually presented as the systems shown in the below Figure1 is of interconnected nodes called neurons. There are a many articles on architecture for neural acceleration.[1],[2],[3]-[4]. Each and every neuron generates the single output by operating on the vector of inputs, represented by x here. The input vector was associated with the weight vector (represented by w), to identify every entry's numerical significance. Here in the mathematical representation, the neurons will first have to compute a weighted sum, and then they will perform a non-linear activation function on the weighted sum in order to generate the output (x).

$$\mathbf{g}(\mathbf{x}) = \mathbf{f}\left(\sum_{\mathbf{i=1}}^{\mathbf{d}} \mathbf{x_i}\ \mathbf{w_i} + \mathbf{w_0}\right) = \mathbf{f}(\mathbf{w^t}\ \mathbf{x})$$

$$\mathbf{f}(\mathbf{s}) = \begin{cases} \mathbf{1, if\ s \geq 0} \\ \mathbf{-1, if\ s} < 0 \end{cases}$$

Usually, an artificial neural network is defined by three parameters :

1. Interconnection pattern between the different layers of the neurons.

2. The activation function that helps in converting a neuron's weighted input to its output activation;

3. The learning process in updating the weights of the interconnections.

The Error tolerance of the Neural Network (NN) makes appropriate computing the promising technique in order to improve the energy efficiency of neural network inference. Earlier approximate computing focused on balancing the trade-off efficiency-accuracy for the existing pre-trained networks, which would lead to sub-optimal solutions.

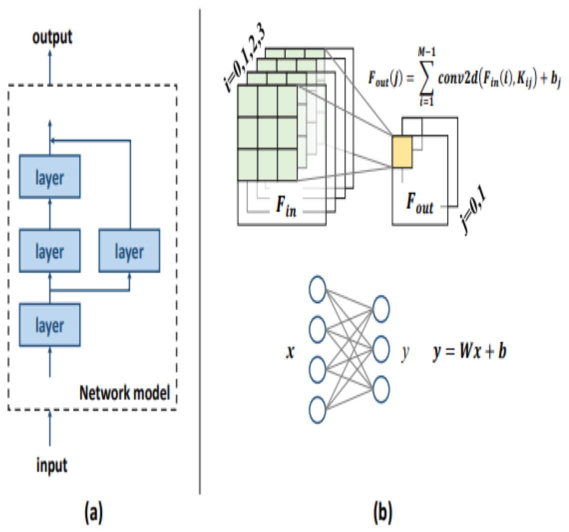


Figure1.Neural Network Model

In this paper the neural network network is considered because it is most widely used in machine learning. The designed multiplier will act as the key hardware component to the neural network. Hence the neural network is implemented using this multilplier and the results obtained are seen.Figure 1 represents the basic general neural network model and the below Figure2 represents the fuzzy storage model of the network including neurons which does processing.

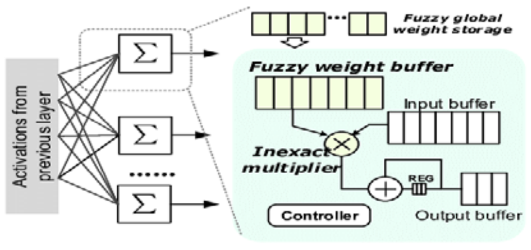


Figure 2. Fuzzy storage model

II.METHODOLOGY

The Architecture of the Simul Multiplier as shown in the below diagram consist of the following components.

Shift and Add Multiplier
Registers
Adder
Shifter
Counter

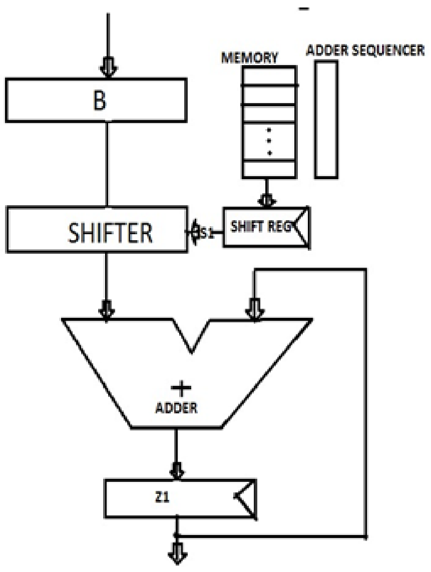


Figure 3.Simul Multiplier Block Diagram

SIMUL is an (Significant - Driven Iterative Approximate Multiplier) as shown in the above Figure 3 is an approximate multiplier features uses control precision that exploits the common characteristics ML Algorithm and it also supports a trade off between the compute precision and energy consumption at run time. Kulkarni et al[17] and Gupta et al[18] they worked together and proposed under-designed multiplier and adder designs in which the minimum and the maximum precision is being fixed at the design time. In the year 2011 Babica et al [11] they have been proposed an ILAM(Iterative Logarithmic multiplier) that can support different precision through the iterations, but our proposed shift-and-add Multiplier is far more energy-efficient with a smaller area than the ILAM. Proposed In the year In the year 1996 K.Chapman [19] proposed a 16 12-bit entries for an 8x8 multiplier per coefficient but our proposed design requires only one 20-bit entry per coefficient.Shift-and-add multiplication is same as the multiplication performed manually by the human. This method adds the 'B' denotes the multiplier. To multiply two numbers by multiplicand 'A' to itself 'B' times, which is performed in a paper, this method takes the digits of the multiplicand by a single digit of the intermediate product in the appropriate positions to the left of the earlier results. To perform the complete operations for getting the final product, the conventional architecture for the shift and add multipliers require many switching activities. In the conventional architecture there is more dissipation of the dynamic power. In this conventional multiplier by eliminating or reducing the sources switching activity, the low power architecture of multiplier can be derived. This is one among digital system of the functional component ,the reduction of power dissipation in multipliers should be as much as possible.

III. TESTS AND RESULTS

The proposed multiplier is based on multiplicands, multiplier, coefficient memory, shift register and adder which is simulated by providing reset, clock and inputs using Modelsim 6.4a and simulation results as shown in the below Figure 4.

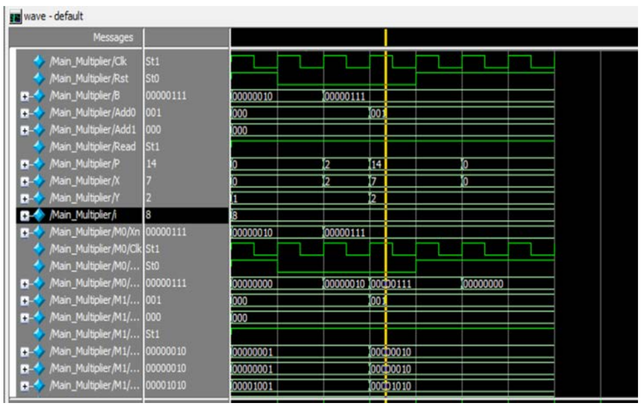

```
wave - default
Messages
/Main_Multiplier/Clk        St1
/Main_Multiplier/Rst        St0
/Main_Multiplier/B          00000111
/Main_Multiplier/Add0       001
/Main_Multiplier/Add1       000
/Main_Multiplier/Read       St1
/Main_Multiplier/P          14
/Main_Multiplier/X          7
/Main_Multiplier/Y          2
/Main_Multiplier/i          8
/Main_Multiplier/M0/Xn      00000111
/Main_Multiplier/M0/Clk     St1
/Main_Multiplier/M0/...     St0
/Main_Multiplier/M0/...     00000111
/Main_Multiplier/M1/...     001
/Main_Multiplier/M1/...     000
/Main_Multiplier/M1/...     St1
/Main_Multiplier/M1/...     00000010
/Main_Multiplier/M1/...     00000010
/Main_Multiplier/M1/...     00001010
```

Figure 4. Simulation results of multiplier

The proposed multiplier is synthesized using Xilinx Synthesis Tool (XST) and Xilinx ISE v9.1i as the development environment. The RTL schematic of the overall design is shown in below Figure 5.

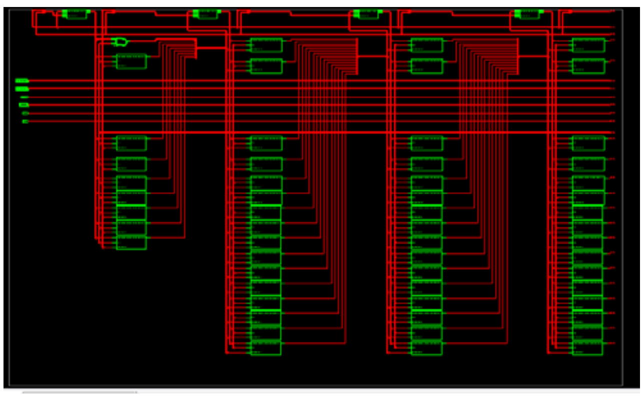
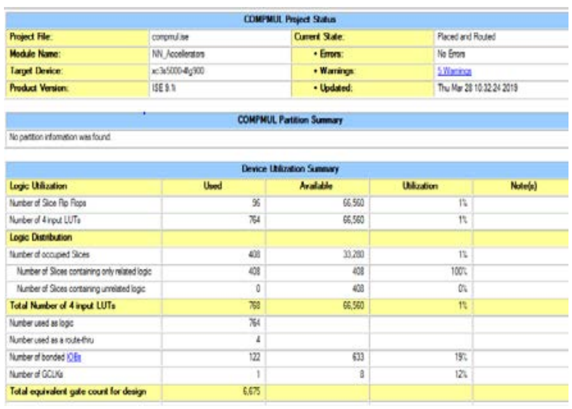


Figure 5. RTL Schematic of Simul Multiplier

The device utilization summary of Shift-and-Add multiplier ,booth and the array multiplier is shown in below Figure 6,Figure 7,and Figure 8.

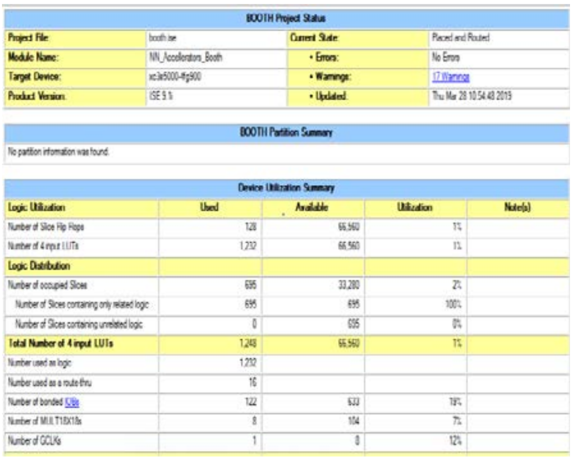


COMPMUL Project Status			
Project File:	compmul.ise	**Current State:**	Placed and Routed
Module Name:	NN_Accelerators	**• Errors:**	No Errors
Target Device:	xc3s5000-4fg900	**• Warnings:**	5 Warnings
Product Version:	ISE 9.1i	**• Updated:**	Thu Mar 28 10:32:24 2019

COMPMUL Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	96	66,560	1%	
Number of 4 input LUTs	764	66,560	1%	
Logic Distribution				
Number of occupied Slices	408	33,280	1%	
Number of Slices containing only related logic	408	408	100%	
Number of Slices containing unrelated logic	0	408	0%	
Total Number of 4 input LUTs	768	66,560	1%	
Number used as logic	764			
Number used as a route-thru	4			
Number of bonded IOBs	122	633	19%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	6,675			

Figure 6. device utilization summary architecture of Shift-and-Add multiplier

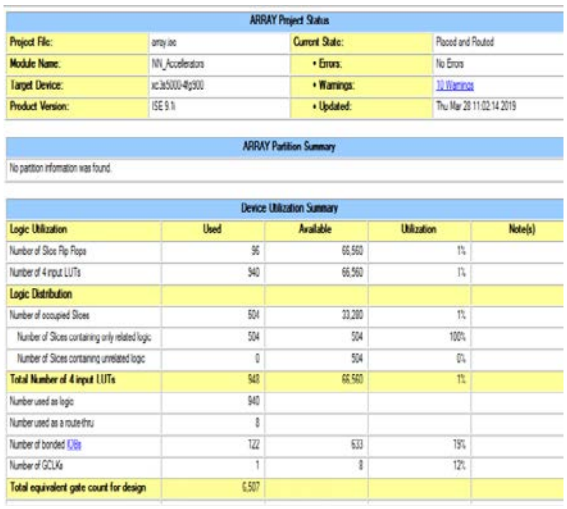


BOOTH Project Status			
Project File:	booth.ise	**Current State:**	Placed and Routed
Module Name:	NN_Accelerators_Booth	**• Errors:**	No Errors
Target Device:	xc3s5000-4fg900	**• Warnings:**	17 Warnings
Product Version:	ISE 9.1i	**• Updated:**	Thu Mar 28 10:54:48 2019

BOOTH Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	128	66,560	1%	
Number of 4 input LUTs	1,232	66,560	1%	
Logic Distribution				
Number of occupied Slices	695	33,280	2%	
Number of Slices containing only related logic	695	695	100%	
Number of Slices containing unrelated logic	0	695	0%	
Total Number of 4 input LUTs	1,248	66,560	1%	
Number used as logic	1,232			
Number used as a route-thru	16			
Number of bonded IOBs	122	633	19%	
Number of MULT18X18s	8	104	7%	
Number of GCLKs	1	8	12%	

Figure 7 device utilization summary architecture of Booth Multiplier



ARRAY Project Status			
Project File:	array.ise	**Current State:**	Placed and Routed
Module Name:	NN_Accelerators	**• Errors:**	No Errors
Target Device:	xc3s5000-4fg900	**• Warnings:**	10 Warnings
Product Version:	ISE 9.1i	**• Updated:**	Thu Mar 28 11:02:14 2019

ARRAY Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	96	66,560	1%	
Number of 4 input LUTs	940	66,560	1%	
Logic Distribution				
Number of occupied Slices	504	33,280	1%	
Number of Slices containing only related logic	504	504	100%	
Number of Slices containing unrelated logic	0	504	0%	
Total Number of 4 input LUTs	948	66,560	1%	
Number used as logic	940			
Number used as a route-thru	8			
Number of bonded IOBs	122	633	19%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	6,507			

Figure 8.Device utilization summary architecture of Array Multiplier

The proposed multiplier is designed as the application part in the Neural Acceleration which can be used in the machine learning as shown in below Figure 9.

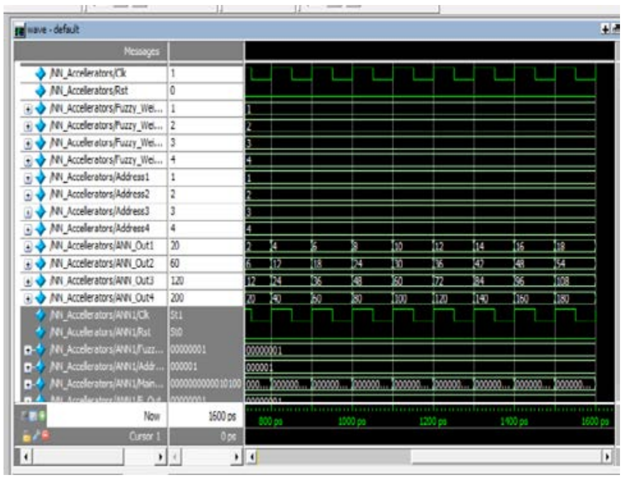

wave - default
Messages
/NN_Accelerators/Clk 1
/NN_Accelerators/Rst 0
/NN_Accelerators/Fuzzy_Wei... 1
/NN_Accelerators/Fuzzy_Wei... 2
/NN_Accelerators/Fuzzy_Wei... 3
/NN_Accelerators/Fuzzy_Wei... 4
/NN_Accelerators/Address1 1
/NN_Accelerators/Address2 2
/NN_Accelerators/Address3 3
/NN_Accelerators/Address4 4
/NN_Accelerators/ANN_Out1 20
/NN_Accelerators/ANN_Out2 60
/NN_Accelerators/ANN_Out3 120
/NN_Accelerators/ANN_Out4 200
/NN_Accelerators/ANN1/Clk St1
/NN_Accelerators/ANN1/Rst St0
/NN_Accelerators/ANN1/Fuzz... 00000001
/NN_Accelerators/ANN1/Addr... 000001
/NN_Accelerators/ANN1/Main... 0000000000010100
2 4 6 8 10 12 14 16 18
6 12 18 24 30 36 42 48 54
12 24 36 48 60 72 84 96 108
20 40 60 80 100 120 140 160 180
Now 1600 ps
Cursor 1 0 ps
800 ps
1000 ps
1200 ps
1400 ps
1600 ps

Figure 9. Simulation result of Neural accelerator

Device name	Area			Delay
FPGA Spartan3XC 3S5000 FG 900-4	Slices	LUT	Gate	Delay
Booth Multiplier	695	1248	41979	11.162 ns
Array Multiplier	504	948	6507	10.979 ns
Shift and Add Multiplier	408	768	6675	10.835 ns

Figure .10 Comparison Table

The Comparison table and the graphical representation of the Shift-and-add multiplier along with the booth and the array multiplier is shown in the below Figure 10, Figure 11 and Figure 12.

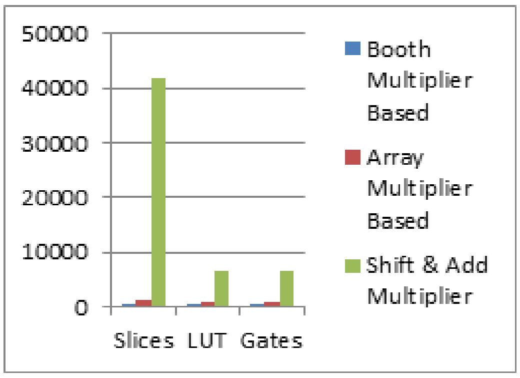


Figure 11. Area Comparision Graph

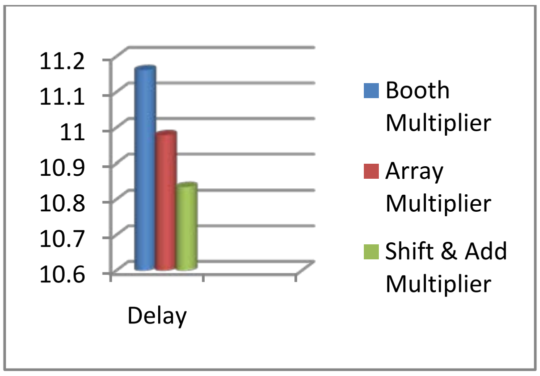


Figure 12. Delay Comparision Graph

IV. CONCLUSION

The proposed architecture Area and Delay Efficiency when compared to a Booth Multiplier and Array Multiplier the proposed architecture makes use of bit width control logic and a low Area and Delay. The simulation result for 8 bit multipliers & four tap Filters shows that the proposed low Area and Delay architecture lowers the total Area & Delay when compared to the Array Multiplier and Booth Multiplier architecture based Filter. The design can be verified using Modelsim 6.4a with Verilog HDL code and Area and Delay is analyzed using Xilinx software. From Table and Comparison Graph, proposed architecture can attain less area and less Delay when compared to the conventional Array Multiplier and Booth multipliers.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our Guide, HOD and Dean Administration Prof. Ruckmani Divakaran for her constant encouragement and excellent guidance without which we wouldn't have accomplished this. We wish to express our hearty thanks to our Principal Dr. Syed Ariff for providing good infrastructure for undergoing this work in college. We would like to thank all faculty members and staffs who were responsible for carrying out this project work successfully. We extend our hearty thanks to our parents and friends for all the moral support provided during the presentation for the project work.

REFERENCES

[1] H. Esmaeilzadeh et al., **"Dark Silicon and the End of Multicore Scaling,"** Proceedings of the 38th International Symposium on Computer Architecture (ISCA), 2011.

[2] J. Park et al., **"Scale-out Acceleration for Machine Learning,"** Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2017.

[3] H. Sharman et al., **"Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks,"** Proceedings of the 45th International Symposium on Computer Architecture, 2018.

[4] J. Albericio et al., **"Bit-Pragmatic Deep Neural Network Computing,"** Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2017.

[5] B. Reagen et al., **"Minerva: Enabling Low- Power, Highly Accurate Deep Neural Network Accelerators,"** Proceedings of the 43rd International Symposium on Computer Architecture (ISCA), 2016

[6] D. Verstraeten et al., **"Isolated Word Recognition with the Liquid State Machine: A Case Study,"** Information Processing Letters, September 2005.

[7] A. Damien, Tensor flow the Examples **"Neural Networks/ multilayer"**_perceptron.py.

[8] Y. LeCun et al., **"The MNIST database of handwritten digits"**.

[9] M. Liberman et al., **"The TI 46-word speech database".**

[10] O. Sakhi, **"Face Detection using Support Vector Machine(SVM)"**.

[11] Z. Babić, A. Avramović, and P. Bulić, **"An Iterative**

Logarithmic Multiplier,” Microprocessors and Microsystems, February 2011.

[12] H. Esmaeilzadeh et al., **“Neural Acceleration for General-Purpose Approximate Programs,”** Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2012.

[13] A. Yazdanbakhsh et al., **“AxBench: A Multi- Platform Benchmark Suite for Approximate Computing,”** IEEE Design & Test,April 2017.

[14] V. K. Chippa et al., **“Scalable Effort Hardware Design: xploiting Algorithmic Resilience for Energy Efficiency,”** Proceedings of the 47th Design Automation Conference (DAC), 2010.

[15] A. Yazdanbakhsh et al., **“Neural Acceleration for GPU Throughput Processors,”** Proceedings of the 48th International Symposium on Micro- architecture, 2015.

[16] A Yazdanbakhsh et al., **“GANAX: A Unified SIMD-MIMD Acceleration for Generative Adversarial Network,”** Proceedings of the 45th International Symposium on Computer Architecture, 2018.

[17] P. Kulkarni, P. Gupta, and M. Ercegovac, **“Trading Accuracy for Power with an Underdesigned Multiplier Architecture,”** Proceedings of the 24th International Conference on VLSI Design, 2011.

[18] V. Gupta et al., **“IMPACT: IMPrecise adders for low-power approximate computing,”** International Symposium on Low Power Electronics and Design (ISLPED), 2011.

[19] K. Chapman, **“Constant Coefficient Multipliers for the XC4000E,”** 1996.