IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 12, December 2024

DOI: 10.17148/IJARCCE.2024.131219

Energy-Efficient High Accuracy With Approximate Multiplier Using Adaptive Truncation

Prof Manjula B B[1], Kusuma K S[2], Monika R[3], Noor Fathima[4], Sheetal P[5]

Associate Professor, ECE, EWIT, Bengaluru, INDIA[1]

Student, ECE, EWIT, Bengaluru, INDIA[2]

Student, ECE, EWIT, Bengaluru, INDIA[3]

Student, ECE, EWIT, Bengaluru, INDIA[4]

Student, ECE, EWIT, Bengaluru, INDIA[5]

Abstract: Approximate computing is a paradigm shift in energy-efficient systems design and operation, based on the idea that we are hindering computer systems' efficiency by demanding too much accuracy from them. Interestingly, large number of application domains, such as DSP, statistics, and machine learning. Approximate computing is suited for efficient data processing and error resilient applications, such as signal and image processing, computer vision, machine learning, data mining etc. Approximate computing circuits are considered as a promising solution to reduce the power consumption in embedded data processing. This paper proposes an FPGA implementation for an approximate multiplier based on selective fractional part based truncation multiplier circuits. The performance of the proposed multiplier is evaluated by comparing the power consumption, the accuracy of computation, and the time delay with those of an approximate multiplier based on exact computation presented. The approximate design obtained energy efficient mode with acceptable accuracy. As compared to conventional direct truncation proposed model significantly influences the performance. Therefore, this novel energy efficient rounding based approximate multiplier architecture outperformed other competitive model**.**

Keywords: FPGA,DSP,Approximate Multiplier

I. INTRODUCTION

A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems. Typical DSP applications where a multiplier plays an important role include digital filtering, digital communications and spectral analysis (Ayman.A et al (2001)). Many current DSP applications are targeted at portable, battery-operated systems, so that power dissipation becomes one of the primary design constraints. Since multipliers are rather complex circuits and must typically operate at a high system clock rate, reducing the delay of a multiplier is an essential part of satisfying the overall design.

Multiplications are very expensive and slows the overall operation. The performance of many computational problems is often dominated by the speed at which a multiplication operation can be executed. Consider two unsigned binary numbers X and Y that are M and N bits wide, respectively. To introduce the multiplication operation, it is useful to express X and Y in the binary representation

$$X = \Sigma\, Xi\ 2^{i} \quad i = 0 \text{ to } M$$

$$Y = \Sigma\, Yj\ 2^{j} \quad j = 0 \text{ to } N$$

The simplest way to perform a multiplication is to use a single two input adder. For inputs that are M and N bits wide, the multiplication tasks M cycles, using an N-bit adder. This shift –and-add algorithm for multiplication adds together M partial products. Each partial product is generated by multiplying the multiplicand with a bit of the multiplier – which, essentially, is an AND operation – and by shifting the result in the basis of the multiplier bit's position.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 12, December 2024

DOI: 10.17148/IJARCCE.2024.131219

Similar to the familiar long hand decimal multiplication, binary multiplication involves the addition of. shifted versions of the multiplicand based on the value and position of each of the multiplier bits.

As a matter of fact, it's much simpler to perform binary multiplication than decimal multiplication. The value of each digit of a binary number can only be 0 or 1, thus, depending on the value of the multiplier bit, the partial products can only be a copy of the multiplicand, or 0. In digital logic, this is simply an AND function.

A faster way to implement multiplication is to resort to an approach similar to manually computing a multiplication. The entire partial product are generated at the same time and organized in an array. A multi operand and addition is applied to compute the final product. The approach is illustrated in the figure 1. This set of operation can be mapped directly into hardware. The resulting structure is called an array multiplier and combines the following three functions: partial product generation, partial-product accumulation and final addition.

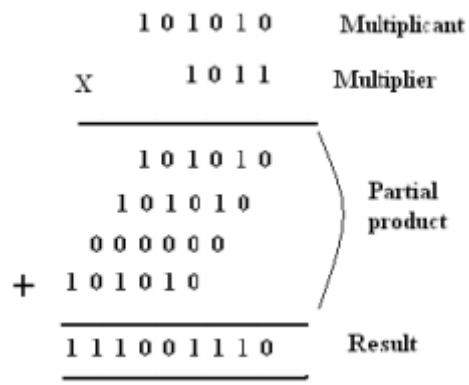


Fig 1.1: Example of manual multiplication

So the adder unit is very important for designing any multiplier(John Rabaey (2003)).The different types of adders and their functions were discussed in (Oklobdzija.V.G et al(1995)),(Pucknell (2004)),(Shalem.R et al (1999)) and (Zimmermann.R and Fichtner.W (1997)). From the results of (Shalem.R et al (1999)) ,we came to know that the new improved 14 Transistor full adder cell shows better result in Threshold loss problem, power dissipation and speed by sacrificing MOS transistor count.

II. METHODS AND MATERIALS

- Step 1: ANN Architecture Design ……..…..(1)

Step 1: ANN Architecture Design

1. Define the layers of the ANN:
   Input layer: 4 neurons (for characters a-z encoded as 4-bit values).
   Hidden layers: 2 or more neurons per layer, depending on the design choice.
   Output layer: 4 neurons, corresponding to the classification results.

2. Initialize Weights and Biases:

Weights for each layer are hardcoded into the VHDL code (e.g., w1[1:4][1:4] for input weights, w2[1:2][1:4] for hidden layer weights, etc.).
Biases for each layer are similarly initialized in the VHDL code.

Step 2: Approximate Multiplier with Adaptive Truncation ..……..… (2)

1. Design the Multiplier:

Implement an approximate multiplier that performs multiplication with reduced precision by truncating the least significant bits (LSBs).

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 12, December 2024
DOI: 10.17148/IJARCCE.2024.131219

The truncation level adapts based on the magnitude of the operands to optimize power consumption without sacrificing too much accuracy.

2. Integrate Multiplier in Neurons:

Use the multiplier within each neuron module to compute the weighted sum of inputs.

- Step 3:VHDL Implementation ……...…(3)

1. Write VHDL Code:
Implement each layer of the ANN as a separate VHDL module.
Include the activation functions and the multiplier in each module to compute the output for each neuron.
Ensure the input data is processed through the network from input to output layers.

2. Implement Reset Logic:
Include reset logic in the VHDL code to initialize the ANN at the start of each operation.

- Step 4:Simulation and Testing ……(4)

1. Simulate Design in ISIM:
Test the VHDL code using Xilinx ISIM to ensure correct functionality. Verify that the network correctly processes input data and produces the expected output.

2. Debug and Optimize:
Use the waveform viewer in ISIM to debug any issues and optimize the multiplier and neuron computations for energy efficiency.

- Step 5: FPGA Implementation ……. (5)

1. Synthesize the Design:
Use Xilinx ISE 14.7 to synthesize the VHDL code and generate the netlist for the Spartan-6 FPGA.

2. Generate Bitstream:
After synthesis, generate the bitstream file for programming the FPGA.

3. Program the FPGA:
Load the bitstream file onto the Spartan-6 FPGA using the Xilinx iMPACT tool and a B-pin USB cable.

Step 6: Real-Time Testing on FPGA …….(6)

1. Provide Input Data:
Test the FPGA by providing input data (characters a-z) and observing the output on the FPGA.

2. Measure Power and Performance:
Measure the FPGA's power consumption and classification accuracy to ensure the design meets the energy-efficiency and accuracy requirements.

Step7: Final Verification and Optimization …….(7)

1. Fine-tune the Design:

Adjust the weights, biases, and truncation parameters based on the performance and accuracy of the design.

2. Document the Project:

Document the design, implementation process, testing results, and optimization steps in a final report.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 12, December 2024

DOI: 10.17148/IJARCCE.2024.131219

MATERIAL USED:

SOFTWARE TOOL:

- XILINX 14.7
- ISIM

HARDWARE MATERIAL:

- FPGA SPARTAN 6 BOARD
- CONNECTING WIRE

HARDWARE AND SOFTWARE IMPLEMENTATION:

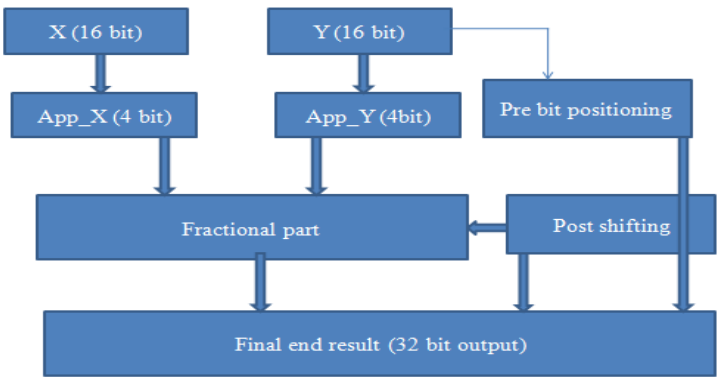
Software implementation:

To implement an energy-efficient, high-accuracy approximate multiplier with adaptive truncation using Spartan-6 FPGA and Xilinx ISE 14.7, the project involves creating a system entirely in VHDL, including an Artificial Neural Network (ANN) for processing input characters (a-z). The approximate multiplier is designed to reduce power consumption by truncating the least significant bits (LSBs) of the operands dynamically, based on input size, while employing error-compensation logic to maintain accuracy. The ANN is implemented in VHDL, with layers performing fixed-point arithmetic using the approximate multiplier for matrix-vector multiplications.

The input characters (a-z) are encoded into binary form and passed to the ANN, which consists of an input layer, one or more hidden layers, and an output layer. Each neuron in the network computes the weighted sum of inputs and applies an activation function (e.g., ReLU or sigmoid). These activation functions are implemented in VHDL using lookup tables (LUTs) or polynomial approximations for efficient hardware execution. The weights and biases for the ANN are pre-defined and stored in the FPGA's block RAM (BRAM).

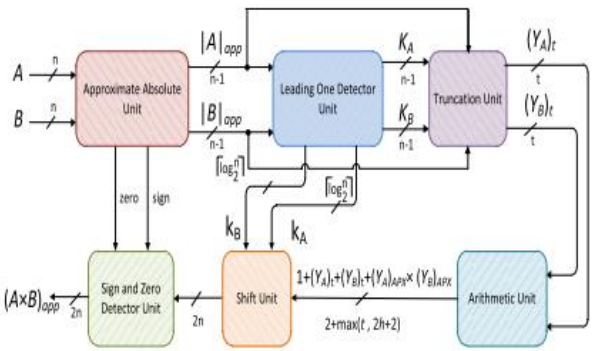
The design process involves writing VHDL code for the approximate multiplier, ANN layers, and data flow control. The system is simulated in ISIM to validate functionality and timing. After synthesis and bitstream generation in Xilinx ISE 14.7, the design is programmed onto the Spartan-6 FPGA using a B-pin USB cable. The system is tested by providing character inputs (a-z) and analyzing the output for classification accuracy. Power consumption is measured to ensure the design meets the energy-efficiency goal.

Fig: 1 – 1.proposed approximate multiplier design multiplier.



1

2.Block diagram of the proposed approximate signed



2

HARDWAREIMPLEMENTATION:

The hardware implementation of an energy-efficient, high-accuracy approximate multiplier with adaptive truncation, integrated with an Artificial Neural Network (ANN) on the Spartan-6 FPGA, involves several critical steps. First, the approximate multiplier is designed to perform energy-efficient multiplication by using adaptive truncation, where the precision of the multiplication result is dynamically reduced based on the magnitude of the input operands. This truncation is implemented in VHDL, where a condition checks the size of the operands, and the result is truncated to a lower bit-width when both operands are small, reducing energy consumption.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 12, December 2024

DOI: 10.17148/IJARCCE.2024.131219

Next, the ANN is constructed in VHDL, consisting of an input layer, multiple hidden layers, and an output layer. Each neuron in the layers computes the weighted sum of its inputs, applies an activation function (such as ReLU or Sigmoid), and forwards the result to the next layer. The weights and biases for the ANN are stored in memory blocks or registers, and the data flows through the layers sequentially. The ANN layers utilize the energy-efficient multiplier instead of a conventional multiplier, enabling adaptive truncation during the weighted sum computation.

The design is synthesized and optimized using Xilinx ISE 14.7 for the Spartan-6 FPGA, where the VHDL code is converted into a netlist. This netlist is then placed and routed, mapping the design onto the FPGA's logic blocks, memory, and I/O pins. A bitstream file is generated, which is used to configure the FPGA with the final design. The design is then tested and validated using ISim, a simulator provided by Xilinx, to verify the functionality and performance of the ANN and the multiplier. After successful simulation, the bitstream is loaded onto the Spartan-6 FPGA, and the design is validated by running it on the actual hardware, ensuring that the ANN operates correctly with the energy-efficient approximate multiplier. This step-by-step approach ensures that the system is both energy-efficient and accurate, optimized for FPGA implementation.

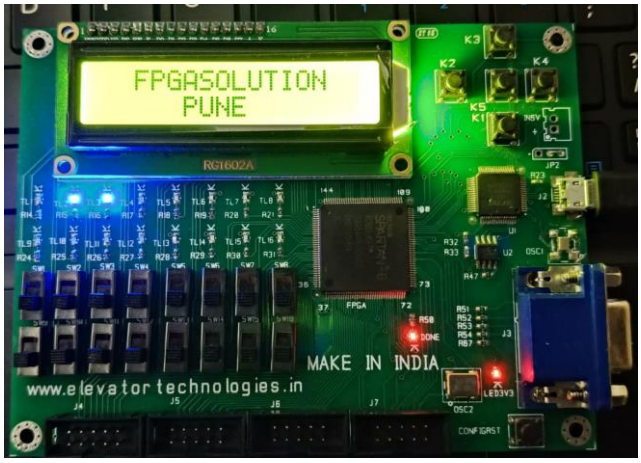


Fig: 2 FPGA SPARTAN-6 BOARD

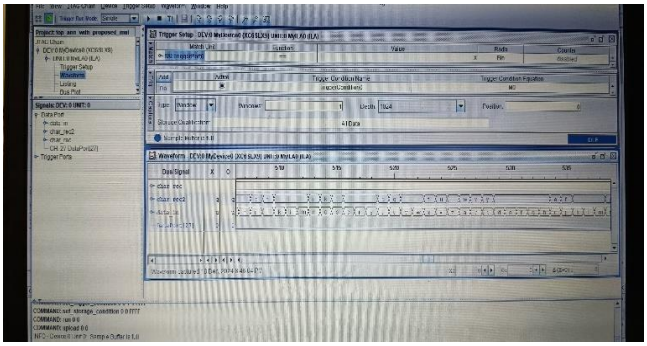


Fig:3 Hardware implementation on FPGA

Table 3.1.1 Features Offered In FPGA

Features	Xilinx virtex II Pro	Altera stratix	Actel Accelerator	Lattice is pXPGA
Clock management	DCM Up to 12	PLL Up to 12	PLL Up to 8	Sys CLOCK PLL up to 8
Embedded memory blocks	Block RAM Up to 10 Mbit	Tri Matrix Memory Up to10 Mbit	Embedded RAM Up to 338K	Sys MEM Blocks Up to 414K
Data processing	CLB and 18-bitx 18-bit Multipliers	LE's and embedded multipliers	Logic modules (C-cell &R-cell)	PFU based
Programmable I/O s	Select IO	Advanced IO Support	Advanced IO Support	Sys IO
Special features	Embedded power PC405 Cores	DSP blocks	Per pin FIFO's for bus application	Sys Hs 1 for high speed serial interface

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 12, December 2024

DOI: 10.17148/IJARCCE.2024.131219

III. RESULT

Simulation result:

This simulation shows digital circuit signals over time. The clock signal controls the circuit's actions, and the reset signal initializes it. Data signals like y1 and y2 demonstrate the circuit's processing.

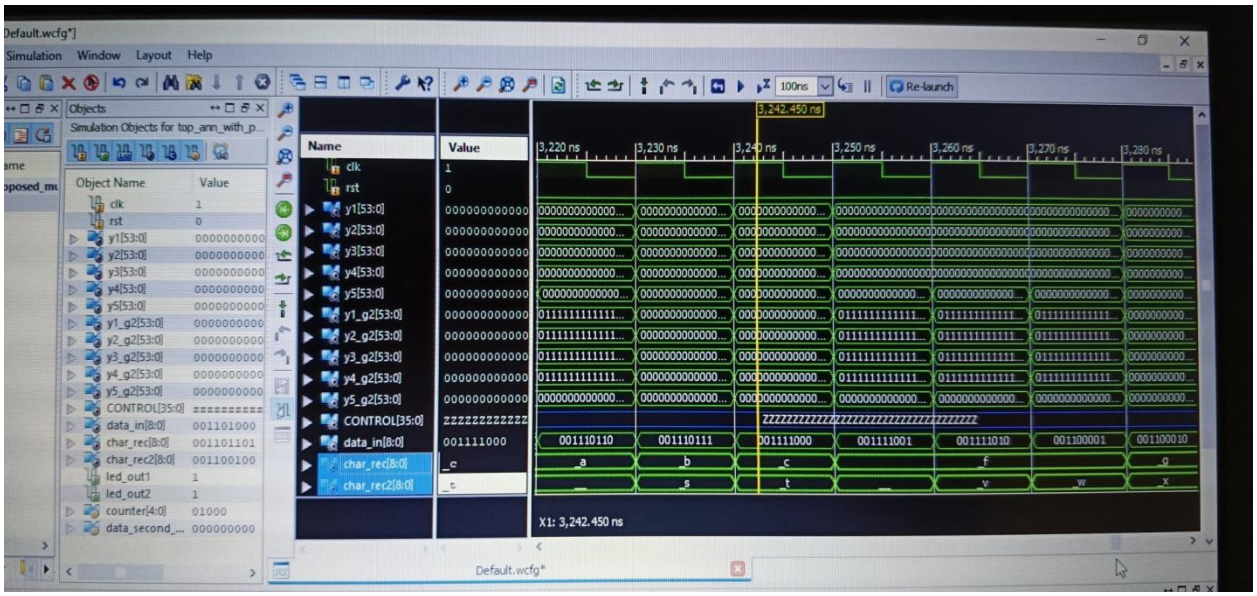


Fig:4 Simulation

Device utilization Plot :

This bar graph shows the Slice Logic Utilization and resource usage of an FPGA design. The key metrics include Slice LUTs, Slice Registers, and Logic Utilization, with red bars representing high utilization. Additional metrics like IOBs, BUFGs, and fanout are also shown for detailed resource analysis.

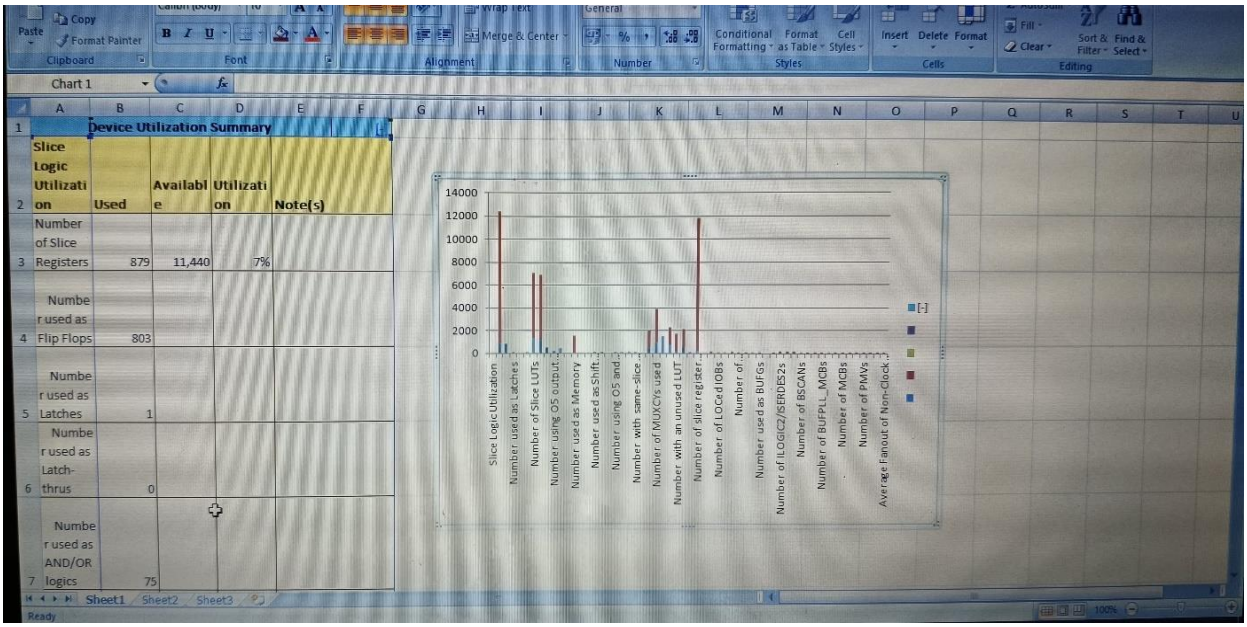


Fig:5 device utilization Plot

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 12, December 2024

DOI: 10.17148/IJARCCE.2024.131219

Power report:

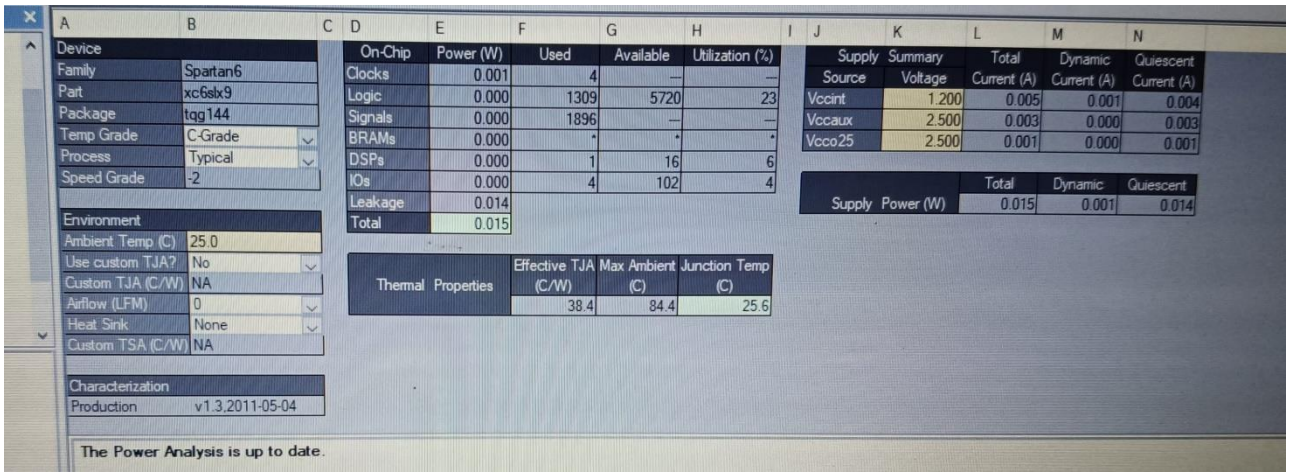


Fig:6 power report

This report provides a power analysis summary for an FPGA design implemented on a Spartan-6 xc6slx9 device:

1. Power Breakdown: It reports total power consumption as 0.015 W, consisting of dynamic power (0.001 W) and quiescent power (0.014 W).
2. Resource Utilization: The utilization of various logic blocks like Clocks, Logic, Signals, and I/Os is presented, e.g., Logic is 23% utilized.
3. Supply Summary: Voltages (1.2V, 2.5V) and corresponding total, dynamic, and quiescent currents are detailed for power sources like Vccint, Vccaux, and Vcco25.
4. Thermal Analysis: With an ambient temperature of 25°C, the junction temperature is 25.6°C, and the effective thermal resistance (TJA) is 38.4°C/W.
5. Environment and Parameters: No airflow, heat sink, or custom thermal values were applied in this analysis, as indicated in the Environment section. The radiation pattern, analysed using HFSS and shown in "Fig 6" depicts energy distribution in space. It highlights directivity and beamwidth, ensuring effective power radiation in desired directions for optimal performance.

IV. CONCLUSION

Here, we investigated the energy and area-efficiency of approximate multiplier in which the input operands were truncated with two different lengths, t and h, and then rounded to the nearest odd numbers to reduce the error resulted by the truncation operation. The proposed multiplier was scalable and outperformed other approximate multipliers in terms of speed, area, and energy using FPGA hardware synthesis**.**

REFERENCES

[1]. M. J. Schulte and E. E. Swartzlander, Jr., “Truncated ultiplication with correction constant,” in *VLSI Signal Processing VI*. Piscataway, NJ: IEEE Press, 1993, pp. 388–396.
[2]. E. J. King and E. E. Swartzlander, Jr., “Data-dependent truncation scheme for parallel multipliers,” in *Proc. 31st Asilomar Conf. Signals, Syst. Comput.*, 1997, pp. 1178–1182.
[3]. M. J. Schulte, J. G. Hansen, and J. E. Stine, “Reduced power dissipation through truncated multiplication,” in *Proc. IEEE lessandro Volta Memorial Int. Workshop Low Power Des.*, 1999, pp. 61–69.
[4]. J. E. Stine and O. M. Duverne, “Variations on truncated multiplication,” in *Proc. Euromicro Symp. Digit. Syst. Des.*, 2003, pp. 112–119.
[5]. J. M. Jou, S. R. Kuang, and R. D. Chen, “Design of low-error fixed-width multipliers for DSP applications,” *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 6, pp. 836–842, Jun. 1999.
[6]. L.-D. Van and C.-C. Yang, “Generalized low-error area-efficient fixedwidth multipliers,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 8, pp. 1608–1619, Aug. 2005.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 12, December 2024

DOI: 10.17148/IJARCCE.2024.131219

[7]. T.-B. Juang and S.-F. Hsiao, "Low-error carry-free fixed-width multipliers with low-cost compensation circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 6, pp. 299–303, Jun. 2005.

[8]. A. G.M. Strollo, N. Petra, and D. De Caro, "Dual-tree error compensation for high-performance fixed-width multipliers," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 52, no. 8, pp. 501–507, Aug. 2005.

[9]. N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G.M. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 6, pp. 1312–1325, Jun. 2010.

[10]. J.-P. Wang, S.-R. Kuang, and S.-C. Liang, "High-accuracy fixed-width

[11]. modified booth multipliers for lossy applications," in *IEEE Trans. Very*

[12]. *Large Scale Integr. (VLSI) Syst.*, Jan. 2011, vol. 19, no. 1, pp. 52–60.

[13]. [11] J.-A. Pineiro, S. F. Oberman, J. M. Muller, and J. D. Bruguera, "Highspeed

[14]. function approximation using a minimax quadratic interpolator,"

[15]. *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 304–318, Mar. 2005.