DESIGN OF HIGH-SPEED MODIFIED WALLACE TREE APPROXIMATE MULTIPLIER

[1]Mr. Krishna Banavathu, [2]Mrs. N Santoshi

[1]Assistant Professor, Department of Electronics and Communication Engineering, AKNU College of Engineering, Adikavi Nannaya University -AP, India.

[2]Assistant Professor, Department of Electronics and Communication Engineering, AKNU College of Engineering, Adikavi Nannaya University -AP, India.

Abstract

Optimizing multiplication architectures is crucial for improving the speed, power efficiency, and area utilization of VLSI circuits. This Paper explores adder compressors and approximate computing to enhance the performance of multipliers while reducing hardware complexity. The existing model is based on a Wallace Tree or Modified Wallace Tree Multiplier, where partial product reduction is performed using 3:2 compressors (full adders) and 4:2 compressors. These structures facilitate parallel additions, significantly improving multiplication speed over conventional sequential methods. A carry-propagate adder (CPA) is then used to compute the final multiplication result.

This paper presents an energy-efficient multiplier design using adder compressors and approximate computing. By integrating 8:2 and 16-bit compressors, the proposed model reduces partial product reduction stages, lowering delay, power, and area compared to conventional Wallace Tree multipliers. Optimized compression techniques enable faster, low-power multiplication, making the design suitable for DSP and FPGA applications.

Keywords: Wallace Tree Multiplier, Adder Compressors, Approximate Computing, Low-Power, Area Efficiency, DSP Applications, FPGA and ASIC Design.

I. INTRODUCTION

Multiplication is a key operation in digital systems, affecting the performance of processors, DSP units, and hardware accelerators. This paper presents a high-speed modified Wallace tree approximate multiplier using [1]adder compressors and approximate computing to optimize

partial product reduction, reduce carry propagation delay, and lower power and area consumption while maintaining acceptable accuracy. The design is suited for low-power, high-performance applications such as signal processing, AI, and embedded systems.

The aim of this paper is to design a high-speed modified Wallace tree [2] approximate multiplier that enhances computational efficiency while reducing hardware complexity through the use of adder compressors. Further, adder compressors help optimize the partial product reduction stage, significantly lowering delay, power consumption, and area. The optimized architecture improves the partial product reduction stage, enabling faster computation with reduced switching activity. This approach provides an effective solution for low-power, high-performance applications such as signal processing, artificial intelligence, and embedded systems.

II. RELATED WORK

A. **Wallace Tree Multiplier** the Wallace Tree Multiplier, introduced by Chris Wallace in 1964, uses parallel partial product reduction to achieve high-speed binary multiplication, overcoming the delays of conventional multipliers and enabling efficient performance in DSP, image processing, cryptography, and machine learning applications.

1. If more than three wires have the same weight, they are processed through a full adder.
2. If two wires share the same weight, they are input into a half adder, while wires with different weights are carried to the next layer.

The Wallace Tree Multiplier follows a three-step process: Partial Product Generation the two binary numbers to be multiplied are represented in bitwise form. AND gates generate the partial products based on bitwise multiplication. Half adders (HA) and full adders (FA) are used to reduce the number of partial product rows in parallel, minimizing the number of sequential addition stages.

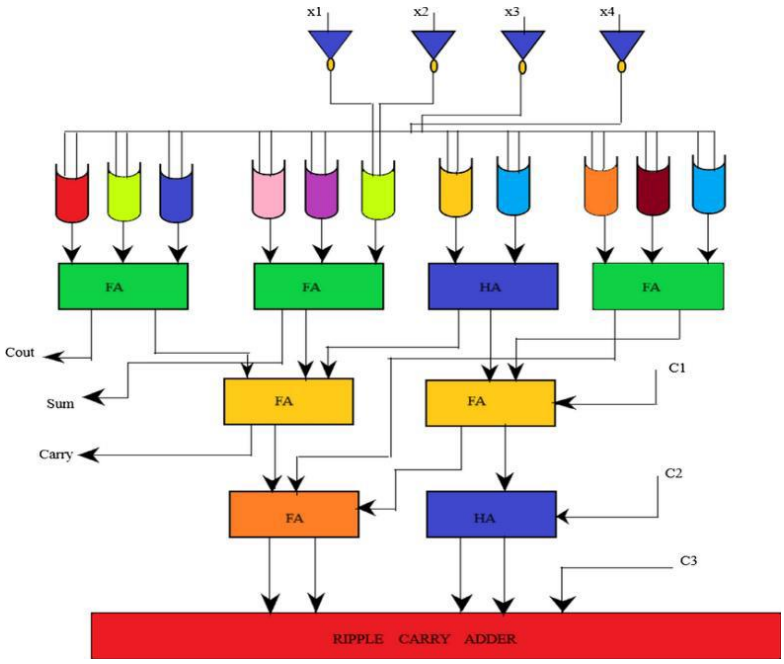

Figure 1 : Structure of Wallace tree multiplier

B. Adder Compressors (ACS) Adder compressors [1] are arithmetic circuits that efficiently sum multiple input bits into fewer outputs, reducing delay, power consumption, and hardware complexity in multiplication operations. Commonly used in high-speed multipliers such as Wallace tree and Dadda architectures, they operate in parallel to minimize critical path delay. Based on input–output configurations, compressors include 3:2, 4:2, 5:2, 7:3, 9:4, 8:2, and 16:2 types, with higher-order compressors enabling faster partial product reduction and improved area and power efficiency in large-scale VLSI designs.

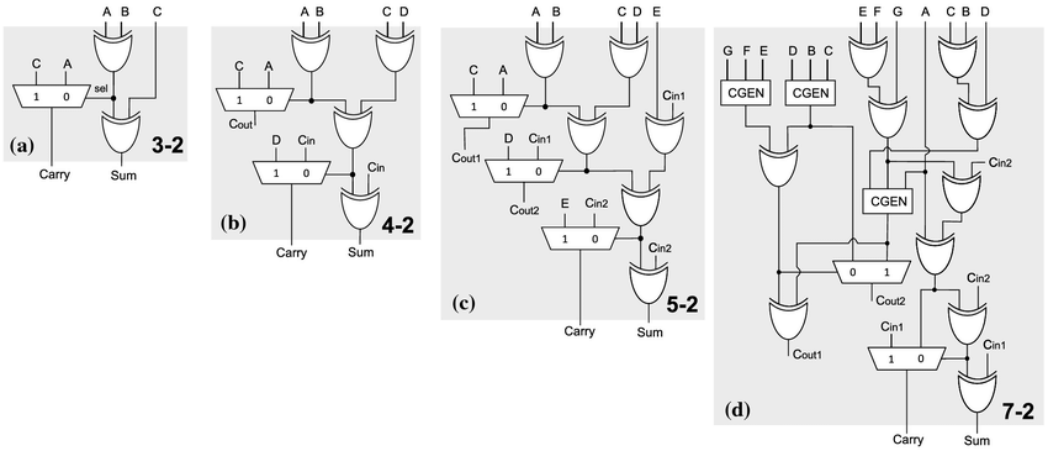

Figure 2 : Internal structures of 3-2, 4-2, 5-2, 7-2 Compressors

An 8:2 compressor is a combinational circuit used in high-speed multipliers to efficiently reduce partial products. It takes 8 input bits and produces 2 output bits, significantly reducing the number of partial product stages in multiplication. The 8:2 compressor sums up 8 input bits ($X1$,$X2$,$X3$,$X4$,$X5$,$X6$,$X7$,$X8$) along with two carry inputs ($Cin1$,$Cin2$), and generates two outputs: Sum (S) and Carry (Cout). This is achieved using multiple full adders (FA) and half adders (HA), reducing delay and improving speed. It can be constructed using a 7:2 compressor and a 3:2 compressor (full adder) for optimized performance. The 7:2 compressor takes seven input bits and produces two outputs (sum and carry), significantly reducing the number of bits at this stage. The remaining eighth input, along with the outputs of the 7:2 compressor, is then processed using a 3:2 compressor (full adder), which further reduces the number of bits by generating another sum and carry. Finally, the combined outputs from these two stages form the sum and carry of the 8:2 compressor.

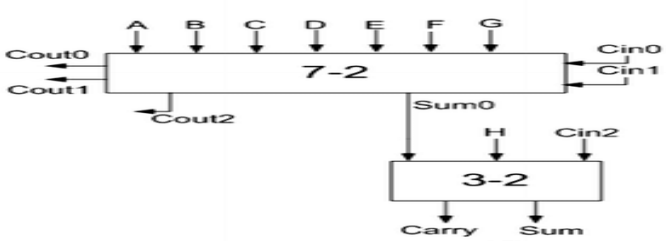

Figure 3: Construction of 8:2 compressor using 7:2 & 3:2 compressors

16- Bit adder using 8-2 compressors [1] if we increase the speed of the any multiplier either reduce the area of the partial products generation or reduce final sum. In this paper, proposed 4-bit Wallace tree multiplier design using 8-bit adder using 4-2 adder compressor and 8-bit Wallace tree multiplier using 16-bit adder using different 8-2 adder compressors and these compressors

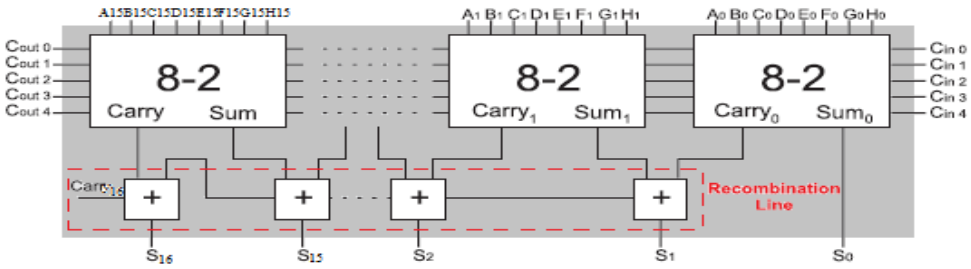

Figure 4 : 16- bit adder using 8-2 compressors

Two multipliers develop a code in Verilog, these delays results were compared with different Wallace tree multipliers developed by various researchers and these results and relevant explanations given.

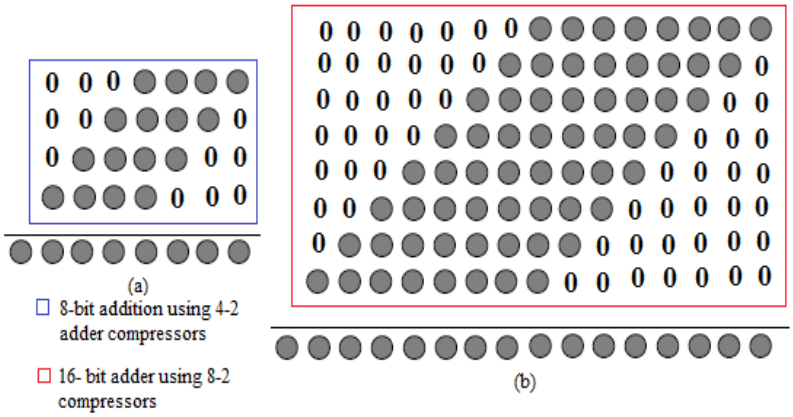

Figure 5: Wallace tree multipliers (a) 4-bit Wallace tree multiplier with 4-2 adder compressor, (b) 8-bit Wallace tree multiplier with different 8-2 adder compressors

The main object in this section we can design high speed multiplier with different high speed adders and these adders design using different adder compressors.

C. Comparison of Parameters

S. No	Title	Implementation	Speed (Delay) (ns)	Area (µm²)	Power Consumption(mW)
01	Design and Analysis of Approximate Compressors for Multiplication	Implemented an 8×8 Dadda tree multiplier with modified 4:2 Approximate compressors	2.1ns - 1.5ns (28.5% improvement)	320µm² - 225µm² (29.7% reduction)	2.1mW - 1.42mW (32.3% savings)
02	Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation	Applied to 16-bit Dadda 4:2 multipliers using Synopsys Design Compiler using Partial Product Perforation	2.0ns - 1.3ns (35% improvement)	450µm² - 248µm² (45% reduction)	2.5mW - 1.25mW (50% savings)

03	Energy-Efficient Approximate Multiplication for DSP and Classification Applications	Implemented on 16×16 multipliers with segment-based approximation using DSM	1.8ns - 1.3ns (27.7% improvement)	400μm² - 260μm² (35% reduction)	3.1mW - 1.3mW (58% energy savings)
04	Low-Power Digital Signal Processing Using Approximate Adders	Applied to DSP applications, including DCT and FIR Filters	2.2ns - 1.5ns (31.8% improvement)	40.66μm² - 22.56μm² (44.5% reduction)	3.0mW - 0.93mW (69% savings)
05	Design of Power and Area Efficient Approximate Multipliers	Implemented using Verilog and CMOS-based circuits	2.3ns - 1.6ns (30.4% improvement)	460μm² - 312μm² (32% reduction)	2.8mW - 1.74mW (38% savings)

These techniques like[4] approximate compressors, Partial Product Perforation, and dynamic segment methods can reduce hardware complexity and power consumption while maintaining acceptable accuracy. These approaches are well suited for error-tolerant applications such as DSP and multimedia. Future work can focus on refining accuracy trade-offs, integrating these methods into FPGA and ASIC designs, and exploring adaptive approximation for optimal performance and energy efficiency

III. METHODOLOGY:

The fig [5] depicts the partial product reduction process in a [3] Wallace Tree or Modified Wallace Tree Multiplier, which is designed to accelerate binary multiplication. At the top, the partial product matrix shows the results of multiplying individual bits from two binary numbers, aligned according to their positional significance. These partial products are then reduced hierarchically using carry-save adders, such as 3:2 and 4:2 compressors, which minimize the number of addition stages. The reduction process transforms the uncompressed sums into a smaller set of intermediate sums and carries, which are finally combined using a carry-propagate adder (CPA) to generate the final product. By performing additions in parallel rather than

sequentially, this architecture significantly improves computation speed compared to conventional multipliers.

The multiplication process is illustrated using partial product formation and their transformation through propagate (P) and generate (G) signals. This method is typically used in high-speed multipliers that implement carry-lookahead addition or approximate multipliers to optimize power and delay.

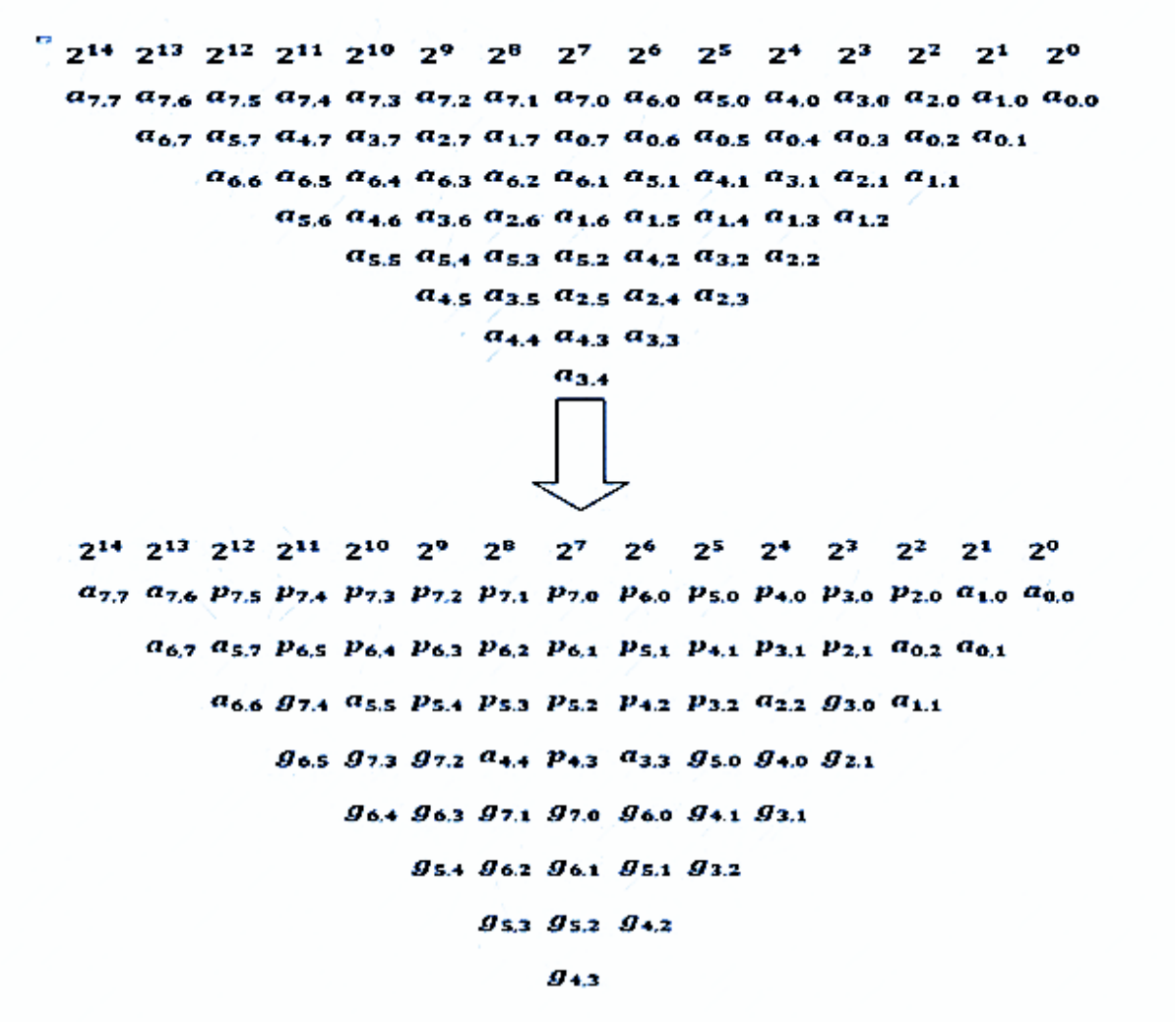

Figure 6: Altered partial products formed by propagate and generate signals

Inputs:

The inputs to the multiplication process are two binary numbers (multiplicand and multiplier).

If we consider an 8×8 multiplier, the two numbers have 8 bits each:

$$A= (a7,a6,a5,a4,a3,a2,a1,a0)\ B=(b7,b6,b5,b4,b3,b2,b1,bo)$$

Outputs: The output is the final product, which is typically 16 bits wide, represented as: $P=(p15,p14,\ldots,p2,p1,p0)$

The intermediate outputs in the diagram include propagate (P) and generate (G) signals that help in reducing delay during addition. The partial products are formed by bitwise ANDing each bit of the multiplier with the entire multiplicand: This results in an array of partial products, which must be summed efficiently. $ai.bj=PPi,j$

To speed up the addition process, propagate and generate signals are introduced: Generate (G) Signal: The generate signal (G) indicates that a carry will be generated regardless of the carry-in bit. It is computed as:

$$Gi= Ai.Bi$$

This means a '1' will be carried to the next stage if both corresponding bits in A and B are 1.

The propagate signal (P) indicates that a carry will be propagated if there is a carry-in from the previous stage. It is computed as:

Propagate (P) Signal:

$$Pi=Ai\oplus Bi$$

This means that if a carry-in is present, it will be passed to the next bit position. The challenges are

- High Power Consumption: The extensive use of logic gates results in significant power dissipation, making the design inefficient for power-constrained applications.
- Increased Delay: The presence of multiple levels of adders and interconnects leads to higher propagation delay, affecting overall computation speed.
- Larger Area Utilization: Due to the complex structure and multiple reduction stages, the design occupies more hardware resources, reducing its suitability for low-power and area-constrained applications.

Approximate Multiplier

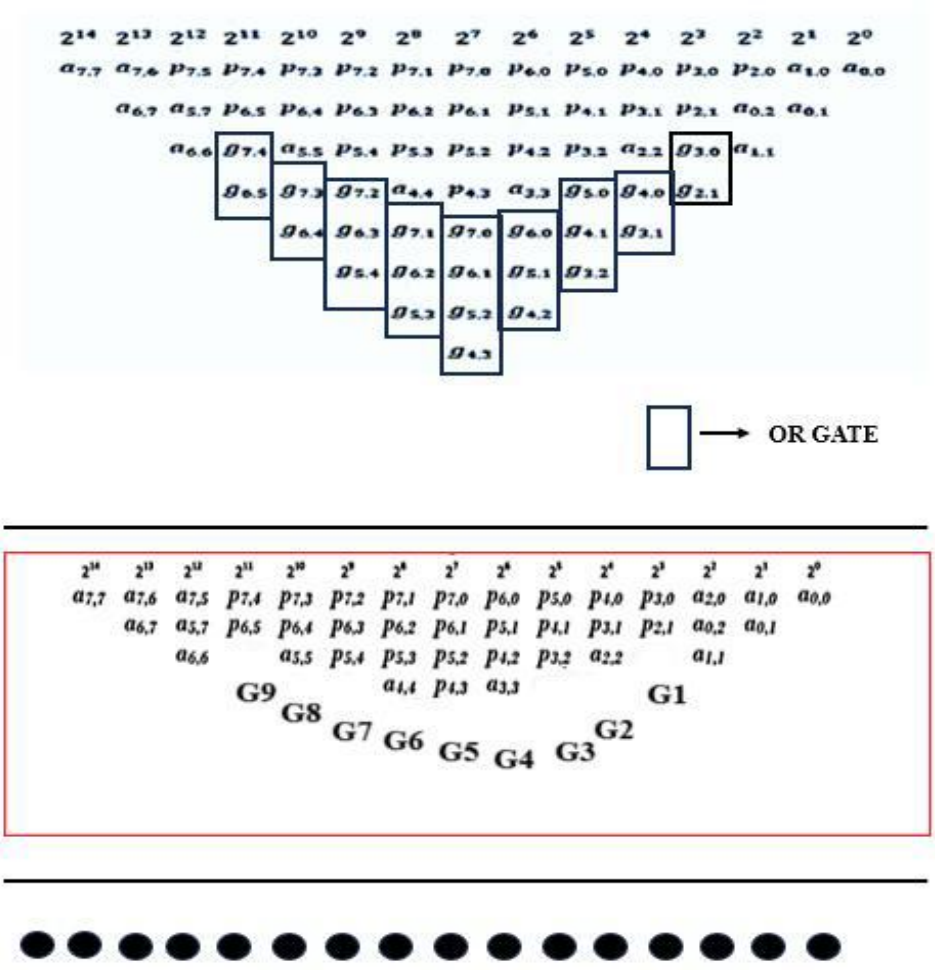

Figure 7: Proposed multiplier using 16-Bit Adder Compressor

The proposed multiplier improves upon the traditional Wallace Tree multiplier by reducing the number of partial product reduction stages using an efficient 8-2 compressor-based architecture and a 16-bit adder compressor. The key innovation is the use of 8-2 compressor structures, which significantly reduce the propagation delay and hardware complexity. Instead of a Wallace Tree, the reduction phase utilizes fewer levels of compression, thereby reducing latency. The final summation stage is handled efficiently using a 16-bit adder, leading to reduced power consumption and area overhead.

- ❖ *Understanding Partial Product Generation*
  - Any binary multiplication follows the principle of ANDing individual bits and shifting them accordingly

- Given two n-bit numbers (multiplicand & multiplier), the first step involves generating partial products, which are then summed up.
- In both the existing and proposed methods, the generation of these partial products remains unchanged.

❖ *Difference in Partial Product Reduction*

- The key distinction between the existing Wallace Tree method and the proposed method lies in how the partial products are summed up. Existing Wallace Tree Approach (Conventional Multiplier) uses a hierarchical compression method, where the bits are grouped and summed in successive stages using full adders (FA) and half adders (HA). Requires more reduction stages, increasing delay and power consumption.
- Proposed 8-2 Compressor-Based Reduction includes the direct compression of eight partial product bits at a time using 8-2 compressors. This reduces the total number of summation levels, leading to: Lower latency (faster computation) and Fewer adders (reducing area and power consumption).

❖ *Role of the 8-2 Compressor*

- The 8-2 compressor is a digital circuit used to add eight binary inputs and generate a smaller number of outputs (typically sum and carry signals).
- This is achieved by using a combination of full adders (FA) and half adders (HA) but in an optimized way.
- Compared to conventional adders, fewer logic levels are required, which improves speed and power efficiency.

❖ *Working of an 8-2 Compressor:*

- It takes 8 input bits and produces 3 outputs: a sum bit, carry bit, and an overflow bit.
- The outputs are then passed to the next stage, where they are added using a 16-bit adder.

❖ *Final Summation Using a 16-Bit Adder*

- After partial product compression, the remaining bits are added using a high-speed 16-bit adder. This further reduces computational delay compared to using multiple smaller adders.

IV. EXPERIMENTS

Schematic Diagram The schematic represents a hardware-level implementation of an optimized multiplier, likely using 8-2 compressors and a 16-bit adder for high-speed multiplication. The AND gates at the top generate partial products, which are then processed through a structured network of compressors in the middle section to reduce the number of addition stages. The final summation is handled by a large-bit adder on the right side, which efficiently computes the product. With 42 I/O ports and 158 nets, the design appears to be optimized for low delay and power efficiency, making it well-suited for FPGA or ASIC implementation. The modular approach, indicated by the multiple Verilog files (proposed1.v, adder416bit.v, adder48to2.v), suggests a hierarchical and reusable design, which can be further analyzed for performance improvements through timing analysis and FPGA synthesis reports.

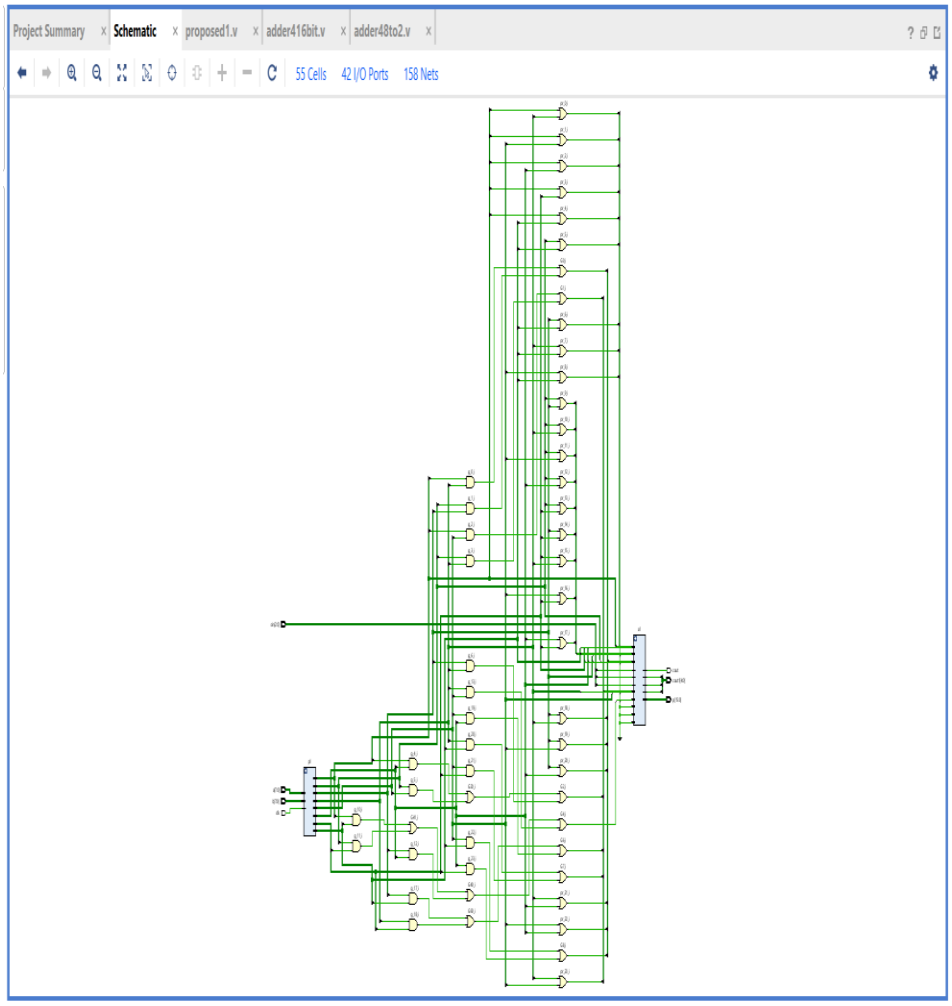

Figure 8: Schematic diagram of proposed multiplier

Simulation Output

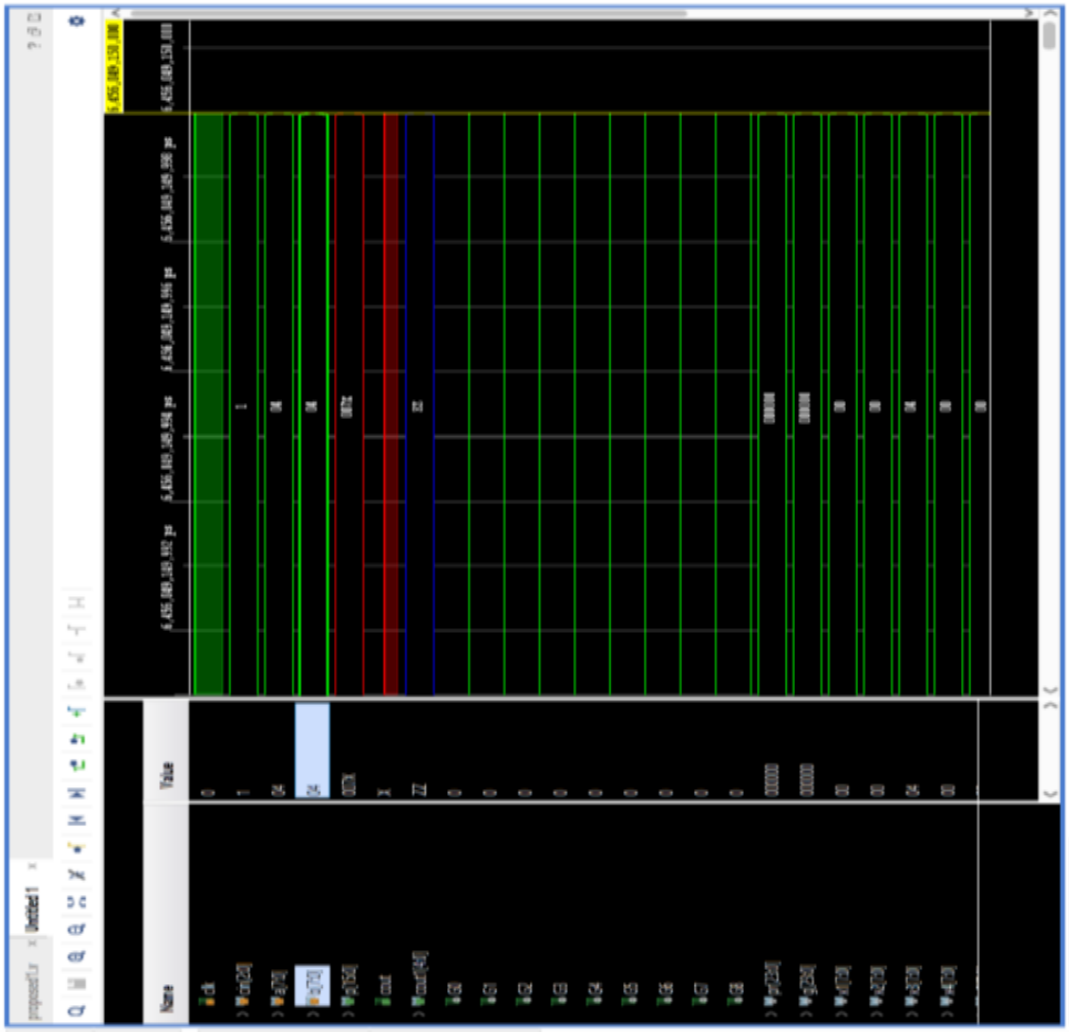

Figure 9 : Output waveform of the Proposed multiplier

The simulated waveform analysis is described above :

The waveform window shows the simulation results of the Verilog implementation.

Inputs:

- a[7:0] = 04
- b[7:0] = 04
- cin[2:0] = 1

Output:

- p[15:0] = 007X → This indicates an undefined (X) bit in the result.
- cout = X (unknown value)
- cout[1:4] = ZZ (high-impedance)

Since this is an approximate multiplier, small errors in the waveform are expected due to the approximation technique. As long as the error is within an acceptable range (low error percentage), the design is valid and usable.

V. RESULT ANALYSIS

	AREA			DELAY (ns)
	SLICES	LUTS	FFS	
Existing Multiplier	43	85	63	16.52 ns
Proposed Multiplier	41	60	64	7.36 ns

The work aimed to design and implement a high-speed approximate multiplier using Verilog HDL and evaluate its performance against a conventional multiplier in terms of hardware utilization and delay. The design was synthesized and functionally verified using Xilinx ISE 14.7 on a Spartan-3E FPGA. The proposed approximate multiplier successfully achieves high-speed computation with lower hardware utilization while maintaining an acceptable level of accuracy. This makes it a viable alternative to conventional multipliers for applications that prioritize speed and power efficiency over absolute precision.

Comparison Graph

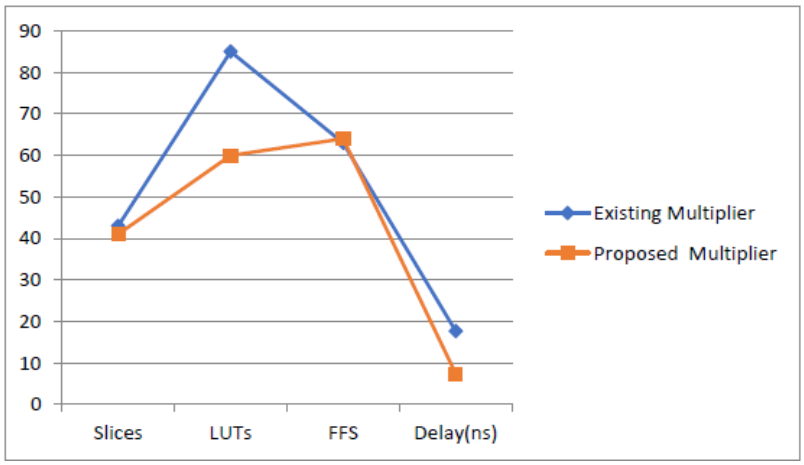

Figure 10: Comparison graph of Existing & Proposed Multipliers

From the given comparison table and graph, we can conclude the following about the proposed approximate multiplier:

Area Efficiency: The proposed multiplier requires fewer slices (41 vs. 43) and significantly fewer LUTs (60 vs. 85), making it more hardware-efficient.

Delay Improvement: The delay is significantly reduced from 16.52 ns to 7.36 ns, improving speed by over 50%.

Negligible Trade-Off: The flip-flop count is slightly higher (64 vs. 63), but this does not significantly affect performance.

Overall Conclusion: The proposed approximate multiplier is faster and more resource-efficient while maintaining acceptable accuracy, making it suitable for high-speed applications.

VI. CONCLUSION

The proposed multiplier efficiently handles numbers up to 14×13, offering high accuracy, low power (79 mW), and reduced area due to fewer transistors. Adaptive voltage control and an 8-transistor full adder improve speed and reduce path delay, while half adders and XOR gates enhance the 4:2 compressor's accuracy. The design is suitable for low-power applications where occasional errors are acceptable. Future improvements could involve optimizing the Wallace Tree multiplier using Booth's algorithm, Kogge-Stone, or Brent-Kung adders for higher-bit multiplication, speed, and area efficiency.

REFERENCES

[1] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and Analysis of Approximate Compressors for Multiplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984-994, 2014. DOI: 10.1109/TC.2014.2308214.

[2] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3105-3117, 2016. DOI: 10.1109/TVLSI.2016.2535398.

[3] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications," *IEEE Transactions on VLSI Systems*, vol. 23, no. 6, pp. 1180-1184, 2014. DOI: 10.1109/TVLSI.2014.2333366.

[4] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-Power Digital Signal Processing Using Approximate Adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124-137, Jan. 2013. DOI: 10.1109/TCAD.2012.2217962.

[5] S. Venkatachalam and S.-B. Ko, "Design of Power and Area Efficient Approximate Multipliers," *IEEE Transactions on VLSI Systems*, vol. 25, no. 5, pp. 1782-1786, 2017. DOI: 10.1109/TVLSI.2016.2643639.

[6] V. Gupta, D. Mohapatra, S.P. Park, A. Raghunathan, K. Roy. “IMPACT: IMPrecise adders for lowpower approximate computing”, in IEEE/ACM International Symposium on Low Power Electronics and Design, pp. 409–414. IEEE, (2011).

[7] A. Momeni, J. Han, P. Montuschi, F. Lombardi, “Design and analysis of approximate compressors for multiplication”. IEEE Trans. Comput. **64**(4), 984-994.14 (2014).

[8] S. Venkatachalam, S.B. Ko,”Design of power and area efficient approximate multipliers. IEEE Trans.Very Large Scale Integration”. VLSI Syst. **25**(5), 1782–1786 (2017).

[9] X. Yi, H. Pei, Z. Zhang, H. Zhou, Y. He, Design of an energy-efficient approximate compressor for error-resilient multiplications, in 2019 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5. IEEE, (2019)

[10] K.M. Reddy, M.H. Vasantha, Y.B. Nithin Kumar, Devesh Dwivedi, Design and analysis of multiplier using approximate 4-2 compressor. AEU-Int. J. Electron. Commun. **107**, 89–97 (2019).

[11] D. Esposito, A.G.M. Strollo, E. Napoli, D. De Caro, N. Petra, Approximate multipliers based on new approximate compressors. IEEE Trans. Circuits Syst. I Regul. Pap. **65**(12), 4169–4182 (2018).