Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 4 (2024)

Design of High-Performance Dynamically Truncated Approximate Multiplier for VLSI Applications

[1]Ms . Akkim. Sravya, [2]Mrs. Dr .K. Jhansi Rani

[1]M.Tech, VLSI & EMBEDDED SYSTEM, ECE, JNTU Kakinada, Andhra Pradesh, India.
[2]Assistant Professor, Dept.Of ECE, JNTU Kakinada, Andhra Pradesh, India.

Abstract:- Our project aims to create a pipelined-based approximate multiplier, integrating pipelining techniques into its design, inspired by successful approximations such as the high-accuracy approximate compressor. Recognizing the vital role of multipliers in applications needing extensive multiplication, their impact on power consumption is significant. An approximate multiplier proves to be a practical approach for minimizing critical path delay in situations where precision is less important. Balancing speed, space, power, and precision is essential for achieving swift computation. Utilizing an approximate multiplier allows for compromises between accuracy, energy efficiency, and area to improve performance. The seamless integration of Xilinx tools not only streamlines the design, simulation, and verification processes but also ensures that the multiplier adheres to the rigorous standards demanded by real-world applications, particularly within the realm of ALUs. The ALU performs a range of arithmetic operations (addition, subtraction, multiplication, division) and bitwise operations (AND, OR, XOR, etc.). It also includes a logic unit that handles logical operations and numerical tests, such as determining if a number is negative or if the output is zero.

Keywords*:* Concept of approximate computing, pipelining. Approximate multipliers, high precision and reconfigurable approximate designs, VLSI application ALU, Digital Logic Design, Computer Arithmetic.

1. Introduction

The design of dynamically truncated approximate multipliers for Very Large-Scale Integration (VLSI) applications, it's a critical aspect of modern DSP and computing systems. Multipliers are fundamental building blocks in many digital systems, including digital filters, FFT processors, and cryptographic algorithms, among others. The demand for high-performance multipliers in VLSI applications is driven by the need for efficient computation in areas such as wireless communications, multimedia processing, and data analytics, where power efficiency and speed are paramount.

Traditional multipliers, especially those based on full parallel multiplication methods, face challenges in meeting the stringent requirements of modern VLSI applications due to their high computational intricacy and electrical usage. This has led to the exploration of alternative multiplier architectures that offer a balance between performance, power efficiency, and area.

Approximate multipliers represent a class of solutions that aim to trade off exact arithmetic for improved performance or reduced resource utilization. These multipliers are designed to tolerate a certain degree of approximation error, which can be acceptable in many applications where the absolute accuracy of each multiplication is less critical than the overall system performance or power budget.

Dynamically truncated approximate multipliers are a specific type of approximate multiplier architecture that selectively truncates the intermediate products' least significant bits, or LSBs during the multiplication process.

This technique can greatly cut down on the multiplier's processing complexity and power use making it more suitable for VLSI implementations.

The design of dynamically truncated approximate multipliers involves several key considerations, including selecting truncation strategies, optimizing truncation points, and the management of approximation errors. These factors must be carefully balanced Make sure the multiplier satisfies the requirements application-specific requirements for accuracy, speed, and power efficiency.

The development of high-performance dynamically truncated approximate multipliers for VLSI applications is therefore a multidisciplinary endeavor that combines principles from digital logic design, computer arithmetic, and VLSI. These multipliers have the potential to significantly increase the capabilities of the next digital systems through creative design methods and meticulous optimization.

2. Objectives

The objective of designing of high-performance dynamically truncated approximate multiplier for VLSI applications, a high accuracy approximate 4-2 compressor that can be used to construct an approximate multiplier is proposed. The proposed approximate multiplier dynamically truncates partial products to adjust the accuracy and a simple error compensation circuit is used to reduce the error distance. The proposed adjustable approximate multiplier shows a decrease in both delay and average power consumption when compared to the Wallace tree multiplier. In comparison to other approximate multipliers, our proposed multiplier demonstrates the minimal mean error distance and the most reduced average power consumption and finally, we are going to implement an 8-bit ALU which performs some arithmetic and logical operations. In this Arithmetic Operation instead of a Multiplier block replacing with an Approximate Multiplier then we will synthesize and Simulate in Xilinx Vivado.

3. Methods

EXISTING METHOD

Multipliers stand as essential components in computational systems, with a multitude of design strategies available for their creation. Among these, the Wallace tree method emerges as particularly effective, utilizing full-adder and half-adder circuits across three stages to facilitate parallel multiplication operations. This technique breaks down the multiplication process into three key phases:

1. Initial multiplication, where Every bit of the multiplicand is multiplied by each bit of the multiplier, generating n^2 partial products.
2. Reduction of partial products through successive layers of full-adder (FA) and half-adder (HA) blocks until only two sets remain.
3. Final addition of the two resulting n-bit numbers using an n-bit adder.

The reduction phase operates under specific rules: groups of three identical bits enter an FA, producing two bits of differing values; pairs of identical bits are processed by an HA; and individual bits advance to the next layer unaltered.

An innovative design for a reversible 4x4 parallel multiplier, inspired by the Wallace tree, has been proposed. This design efficiently handles the multiplication of two 4-bit numbers, producing 16 partial products that are systematically reduced and added using FAs, HAs, and ultimately, a 4-bit carry ripple adder. The efficiency of a multiplier is closely tied to the speed at which partial products are generated and summed, highlighting the importance of minimizing the number of partial products and employing efficient addition mechanisms.

The reduction phase operates under specific rules: groups of three identical bits enter an FA, producing two bits of differing values; pairs of identical bits are processed by an HA; and individual bits advance to the next layer unaltered. An innovative design for a reversible 4x4 parallel multiplier, inspired by the Wallace tree, has been proposed. This design efficiently handles the multiplication of two 4-bit numbers, producing 16 partial products that are systematically reduced and added using FAs, HAs, and ultimately, a 4-bit carry ripple adder. The efficiency of a multiplier is closely tied to the speed at which partial products are generated and summed, highlighting the importance of minimizing the number of partial products and employing efficient.

The distinction between half-adders, capable of adding only two numbers, and full-adders, designed for adding three 1-bit binary numbers including a carry bit, underscores the complexity of multiplier design. The hardware requirements for array multipliers, in terms of the number of full adders (FA) and the final adder length (FAL), vary depending on the multiplier size, as illustrated in the referenced figure.

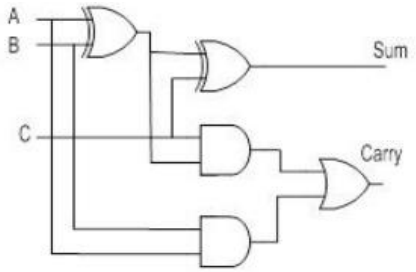

Fig 1: FULL ADDER

When two outputs are produced from three inputs, the adder is called a full adder. Three inputs are carried as follows: A, B, and C are the initial two inputs. SUM denotes the regular output, while CARRY denotes the carry output.

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

TABLE 1: Truth table of Full adder

In the table mentioned earlier:

- 'A' and 'B' denote the input variables, symbolizing the two crucial bits intended for addition.

- 'Cin' signifies the carry input, which is the carry bit obtained from the preceding lower significant position.

- 'Sum' and 'Carry' are the output variables, indicating the resulting values post addition.

- The table encompasses eight rows beneath the input variables, illustrating every conceivable combination of 0s and 1s for these variables.

An array multiplier presents an optimized configuration for a combinational multiplier. It enables the multiplication of two binary numbers through a single micro-operation, leveraging a combinational circuit to generate the product bits simultaneously. This method is particularly swift for multiplying numbers, as the sole delay factor is the propagation time of signals through the gates constituting the multiplication array. Considering two binary numbers, A and B, with m and n bits respectively, an array multiplier employs a series of AND gates to produce summands in parallel. Specifically, an n x n multiplier necessitates n(n-2) full adders, n half-adders, and n^2 AND gates. Furthermore, the maximum delay in an array multiplier scenario is calculated as (2n+1) times the delay of a single gate (td).

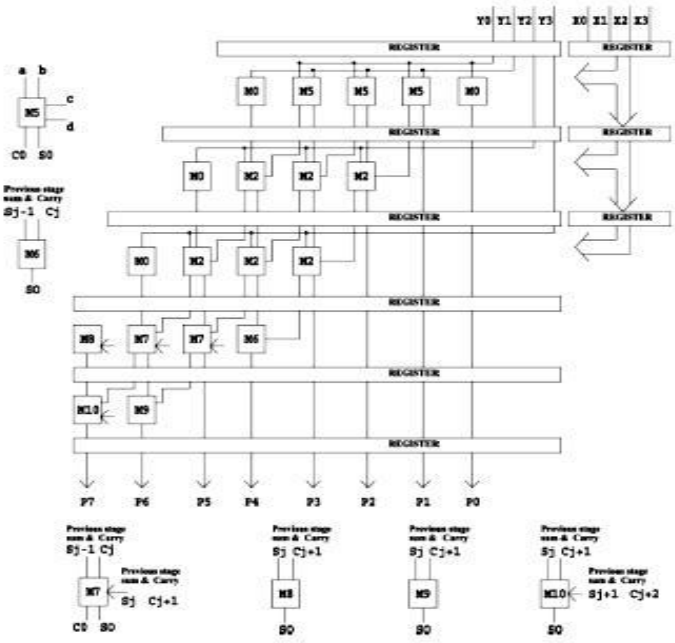

Fig 2: ARRAY MULTIPLIER

Binary multiplication for positive numbers can be achieved using a two-dimensional combinational logic array, as demonstrated by a 4x4 array multiplier. This setup utilizes full adders and AND gates to determine whether multiplicand bits are added to partial products based on multiplier bit values. Each row computes the sum of the multiplicand and incoming partial product, generating an outgoing partial product if the multiplier bit equals 1. The worst-case signal delay occurs from the upper right corner to the bottom left corner of the array. Despite its efficiency, the array multiplier faces challenges with speed and power consumption, leading to the development of various low-power design techniques aimed at reducing dynamic power consumption.

PROPOSED METHOD

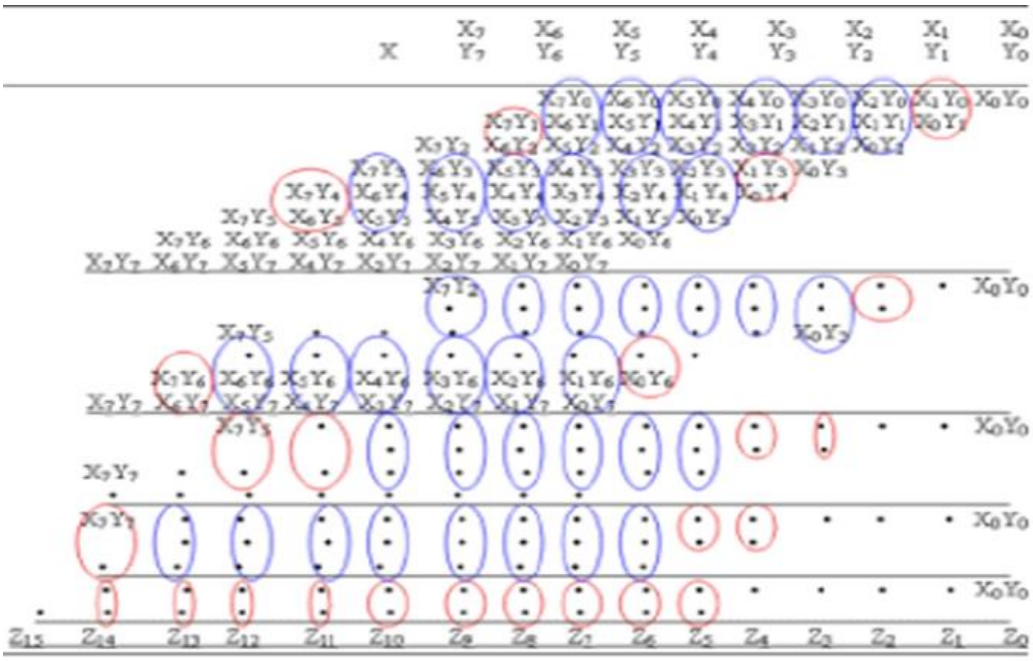

Fig 3: WALLACE MULTIPLIER

The Wallace tree module combines four partial products to produce two intermediate values for the final sum, potentially increasing area and capacitance requirements. Partially Guarded Computation (PGC) segments arithmetic units like adders and multipliers into two sections, deactivating the unused portion to reduce power usage. Reversible circuits maintain an equal number of inputs and outputs, allowing for a unique mapping between them—meaning outputs can be determined from inputs and vice versa. These circuits are assessed based on several factors, including the quantity of gates, constant inputs, unnecessary outputs (garbage outputs), Quantum Cost (QC), latency, and hardware intricacy. The gates count refers to the total number of reversible gates needed for the circuit. Constant inputs are fixed at either 0 or 1. Garbage outputs are those not used in subsequent calculations. Quantum Cost measures the total primitive quantum gates required for the circuit. Delay is calculated as the greatest number of gates along the critical path from input to output.

PROPOSED METHOD

In this initiative, we're integrating the pipeline concept into an approximate multiplier to boost performance and accelerate computational tasks. Our proposal includes an approximation method featuring dynamic truncation of partial products, followed by an error compensation circuit. This adaptable truncation technique facilitates dynamic input truncation, enabling the construction of a versatile multiplier. This contrasts with traditional precise Wallace tree implementations and our novel approach to a dynamically adjustable approximate multiplier process.

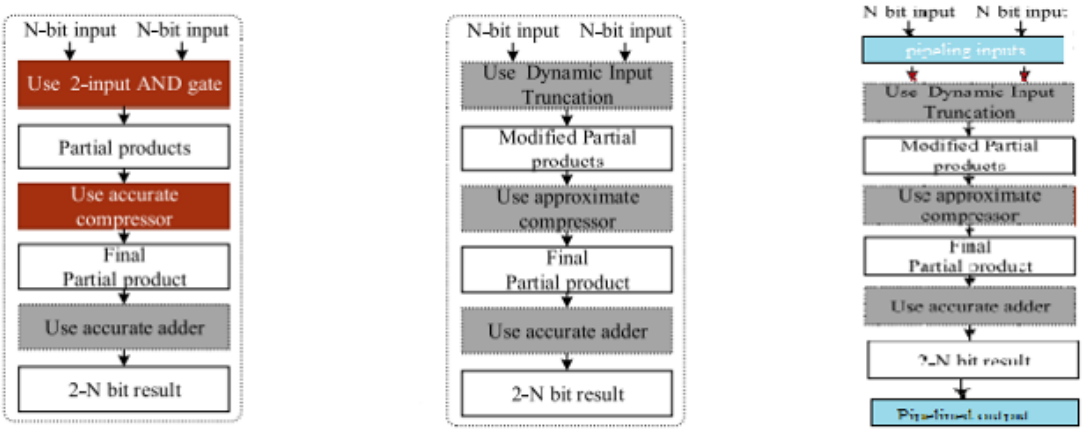

Fig 4.1 (a),(b),(c) OF APPROXIMATE MULTIPLIER AND EXACT WALLACE MULTIPLIER FLOW, PROPOSED PIPELINED MULTIPLICATION FLOW

Figure 4.1(a) depicts the conventional multiplication method that guarantees precise results. The process begins with generating accurate partial products using 2-input AND gates, followed by compression with precise compressors. The final stage involves summing these compressed partial products using accurate adders to produce the result. In contrast, Figure 4.1(b) shows the non-pipelined approach for the proposed approximate multipliers. The main differences between conventional multiplication and the proposed method lie in the generation and compression of partial products. During the generation phase, dynamic input truncation modifies the partial products, as depicted in Figure 4.1(c), showcasing a pipelined base approximate multiplier.

Pipelining Concept: Pipelining involves executing instructions sequentially through a pipeline, which organizes the storage and execution of instructions. This approach increases concurrency in systems that lack feedback loops. By overlapping multiple instructions during execution, pipelining divides the process into stages that form a pipeline structure. Instructions enter at one end and exit at the other, improving overall instruction throughput. Each stage of the pipeline includes an input register followed by a combinational circuit, which holds data and performs operations. The output from one stage's combinational circuit is fed into the input register of the subsequent stage. Pipelining can reduce the critical path by strategically placing latches, thus enabling faster system operation. Essentially, it transforms a topology without feedback loops into one that is suitable for high-

speed applications, overcoming the speed limitations of the original design. Although pipelining reduces the critical path, it introduces additional latches and delays, known as system latency.

Approximate 4-2 Compressor: We present a high-accuracy, low-power approximate 4-2 compressor, as illustrated in Figure 4.2. Since a full-adder functions as a 3:2 compressor, we advance to an approximate 4:2 compressor to improve upon the full adder. The design involves four inputs ($X_1 \sim X_4$) generating $W_1 \sim W_4$ according to Equations (1)-(4). Recognizing that an incorrectly computed carry bit incurs twice the error distance of an incorrect sum bit, the proposed compressor ensures the carry bit is generated with precision. The equations for generating the carry bit are detailed in (5)-(7). The carry bit is set to 1 under three conditions, primarily when X_1 and X_2 are both 1.

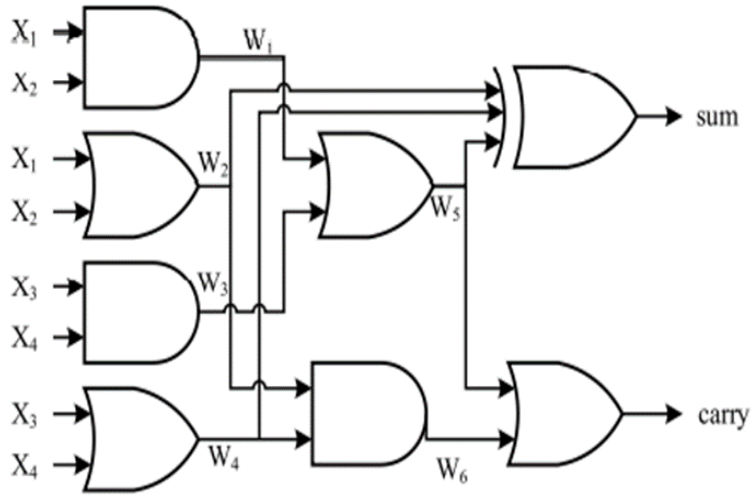

Fig 4.2 GATE-LEVEL IMPLEMENTATION OF PROPOSED 4-2 COMPRESSOR

Another is X_3 and X_4 are both 1. The third is either of X_1 or X_2 is 1 and either of X_3 or X_4 is 1. (5) Checks the first two situations, and (6) checks the third situation. (7) Produces the final carry bit.

$$W_1 = X_1 \& X_2 \quad (1)$$

$$W_2 = X_1 | X_2 \quad (2)$$

$$W_3 = X_3 \& X_4 \quad (3)$$

$$W_4 = X_3 | X_4 \quad (4)$$

$$W_5 = W_1 | W_3 \quad (5)$$

$$W_6 = W_2 \& W_4 \quad (6)$$

$$CARRY = W_5 \& W_6 \quad (7)$$

The formula for generating the sum bit is presented in Equation (8). In a conventional 4-2 compressor, the sum bit is generated using four XOR gates within two full adders. In contrast, our proposed compressor produces the sum bit by feeding W2 and W4 into a 2-input XOR gate, utilizing signals used for the carry bit generation. This method reduces circuit area and static power consumption by sharing common signals. However, we found that relying solely on W2 and W4 in a 2-input XOR gate results in a significant error distance. This issue arises because W2 and W4 are derived from OR gates, causing errors when both X1 and X2 are 1 or both X3 and X4 are 1, which leads to an incorrect sum bit of 1 instead of 0. To improve accuracy, we include W5, a signal that detects these conditions, in the XOR gate. For example, if both X1 and X2 are 1, both W_2 and W_5 will be 1, and the sum bit will be '0 XOR W_4', effectively becoming W_4. In this scenario, only X_3 and X_4 need consideration. Nonetheless, when all four inputs are 1, the sum bit incorrectly results in 1, contributing to an error distance of 1.

$$Sum = W_5 \wedge W_2 \wedge W_4 \quad (8)$$

X3	X2	X1	X0	Carry	Sum	Diff
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	0	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

TABLE 2: Truth table of approximate 4:2 compressor

For error detection, an additional AND gate is required to determine if both W1 and W3 are 1. This is because W1 uses an AND gate to check if both X1 and X2 are 1, and W3 uses an AND gate to verify if both X3 and X4 are 1. The error detection circuit (EDC) is described by Equation (9). Consequently, To implement error compensation for the proposed 4-2 compressor, we can easily add an extra AND gate. The error compensation is defined as

$$\text{Error} = X3 \,\&\, X4 \qquad (9).$$

For an adjustable approximate multiplier in operation, we suggest a dynamic input truncation method that employs two 2-input AND gates, as illustrated in Figure 4.3. This method generates a partial product, defined by Equation (10), where A is the multiplicand and B is the multiplier. The Trunc signal determines whether the partial product PPD should be truncated; if Trunc is set to 1, the PPD is zeroed out. Specifically, Trunc signals help conserve power by reducing PPDs to zero during multiplication, effectively enabling or disabling hardware units in the relevant columns.

$$PPD_{ij} = (\sim Trunc \,\&\, B_i)\&A_j \qquad (10)$$

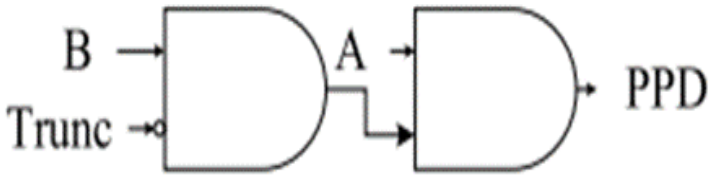

Fig 4.3. MODIFIED PARTIAL PRODUCT

In an 8 × 8 multiplier, each bit of the multiplier is matched with eight bits of the multiplicand, enabling a reduction in hardware costs by utilizing shared gates along with an additional AND gate. For example, PPD00 is computed as ~Trunc0·B0·A0, and PPD01 as ~Trunc0·B0·A1. Here, ~Trunc0·B0 can be pre-computed to form a mask, requiring only three 2-input AND gates, as depicted in Figure 4.4. The handling of Trunc signals in the proposed approximate multiplier provides flexibility for dynamic adjustments.

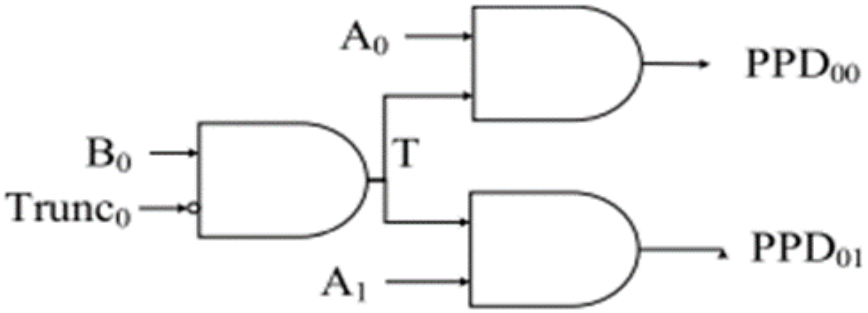

Fig4.4 A GATE SHARING EXAMPLE TO REDUCE THE NUMBER OF GATES

Figure 4.5 depicts an approximate multiplier utilizing the suggested methodology. Although the multiplier's input width is designed for 8 bits, this technique can be scaled to accommodate larger multipliers. The proposed approximate multiplier operates in three phases. In the first phase, each partial product is generated using two 2-input AND gates, as shown in Figure 4.3. The gate-sharing approach illustrated in Figure 4.4 is then applied to further reduce hardware costs. The accuracy of the resulting partial product can be adjusted according to the Trunc signal, depending on the desired outcome. To enhance and optimize control efficiency and lower hardware costs, our proposed approximate multiplier Features a 4-bit Trunc signal, with each bit controlling multiple columns of partial products, referred to as a '3-4-4-4 partition.' Specifically, from the most significant bit (MSB) to the least significant bit (LSB), each bit governs columns 14th to 12th, 11th to 8th, 7th to 4th, and 3rd to 0th, which are indicated by khaki, sky blue, green, and black in Stage 2 of Figure 4.5. For example, if Trunc (3-0) is set to 01012, the columns 14th to 12th and 7th to 4th will be accurate, while the columns 11th to 8th and 3rd to 0th will be truncated.

The second phase delineates the process of compressing the partial products. Following their generation, these products are categorized into two zones: the accurate zone encompassing columns 14th to 8th, and the approximate zone covering columns 7th to 0th. The division into accurate and approximate regions is based on a straightforward half-and-half allocation. Performing For example, a 30-70 split would lead to significant accuracy loss because of the heavy reliance on the approximate multiplier. On the other hand, a 70-30 split would reduce the advantages of approximate computing in terms of power savings.

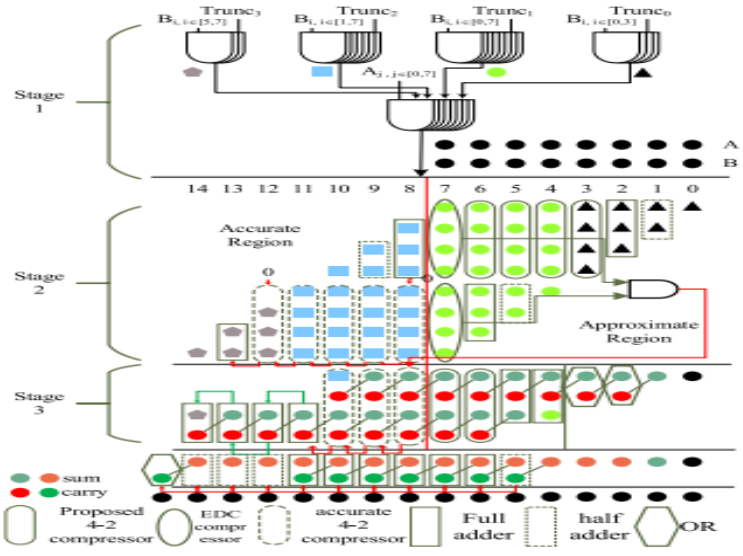

Fig 4.5 PROPOSED APPROXIMATE MULTIPLIER

Since partial products in the accurate region are more significant, we use precise 4-2 compressors to process these products. In the approximate region, we utilize our proposed approximate 4-2 compressors along with an error compensation circuit. In the third phase, outcomes are generated using OR gates in columns from the 3rd to the 0th, without carry propagation, since errors in these columns, being close to the least significant bit (LSB), have a minimal effect on the final output. Errors identified In the second stage, an Error Detection Circuit (EDC), which includes a single AND gate, determines if a compensation bit is needed. For the remaining columns, partial products are compressed using a mix of our proposed approximate 4-2 compressors, accurate 4-2 compressors, full adders, and half adders. After the third stage, the final two rows of partial products are obtained and then summed with accurate adders to produce the final results.

APPLICATION:

Developing an 8-bit ALU that encompasses a wide array of operations, such as addition, subtraction, approximate multiplication, and various logical functions, signifies a considerable advancement in digital computing technology.

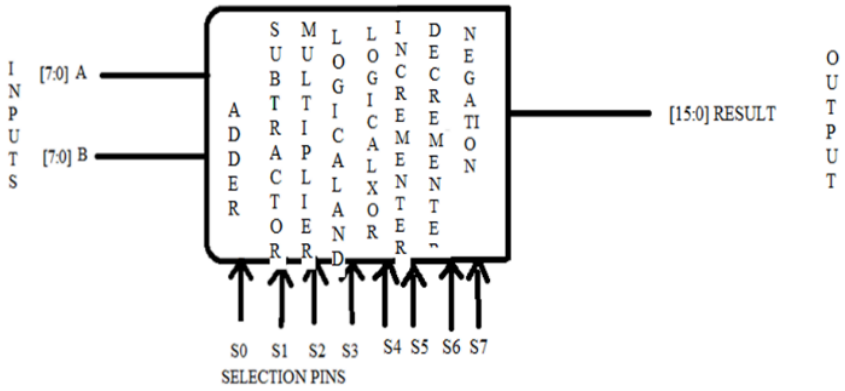

Fig 5: 8-BIT ALU

This unified module not only enhances the ALU's versatility and efficiency in processing binary data within digital systems but also integrates functionalities that extend beyond traditional arithmetic operations. Specifically, the inclusion of approximate multiplication allows the ALU to perform complex calculations with reasonable accuracy, proving particularly advantageous in uses, such signal processing, where exact multiplication is not essential or machine learning algorithms. Moreover, the ALU's design incorporates logical operations such as AND, OR, XOR, negation, incrementation, and decrementation, significantly expanding its utility. These operations, foundational to Boolean algebra, are extensively applied in digital circuit design, data manipulation, and control flow operations, underscoring the ALU's broad applicability.

The ALU's adaptability is further demonstrated by its ability to select specific operations based on control signals, allowing users to customize its functionality to meet a variety of computational demands. This flexibility is crucial across numerous computing scenarios, from arithmetic calculations to bitwise manipulations and data transformations, highlighting the ALU's role in enhancing processing speed and efficiency through parallel operation capability.

In summary, this project underscores the significance of designing and integrating discrete components to create a functional ALU capable of efficiently handling a wide range of computational tasks, emphasizing the importance of such advancements in advancing digital computing technology.

4. Results

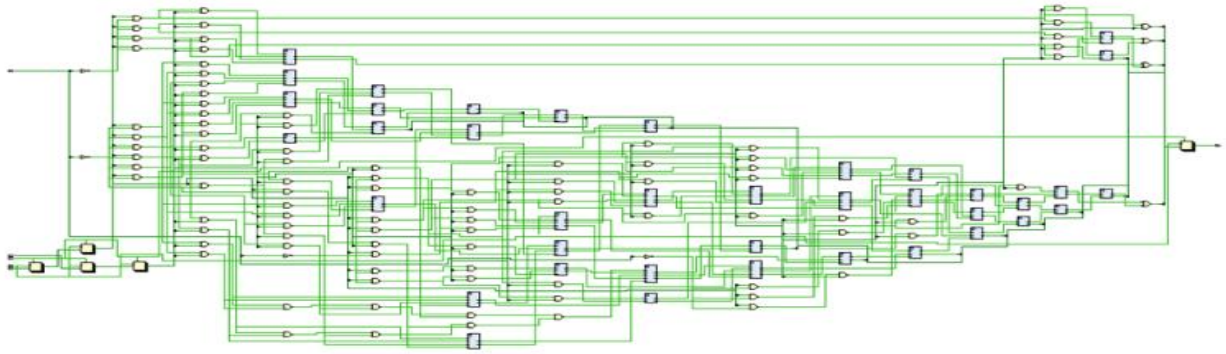

Fig 6: RTL SCHEMATIC OF PROPOSED APPROXIMATE MULTIPLIER

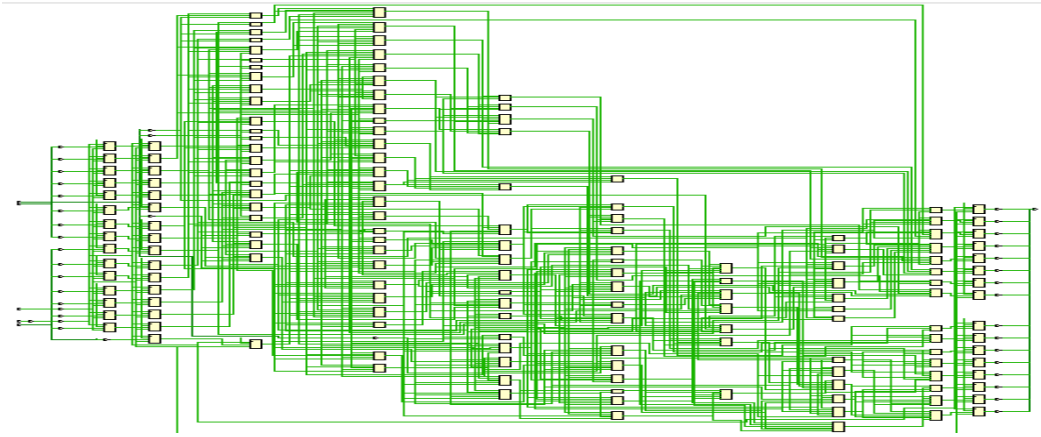

Fig 7: TECHNOLOGY SCHEMATIC OF PROPOSED APPROXIMATE MULTIPLIER

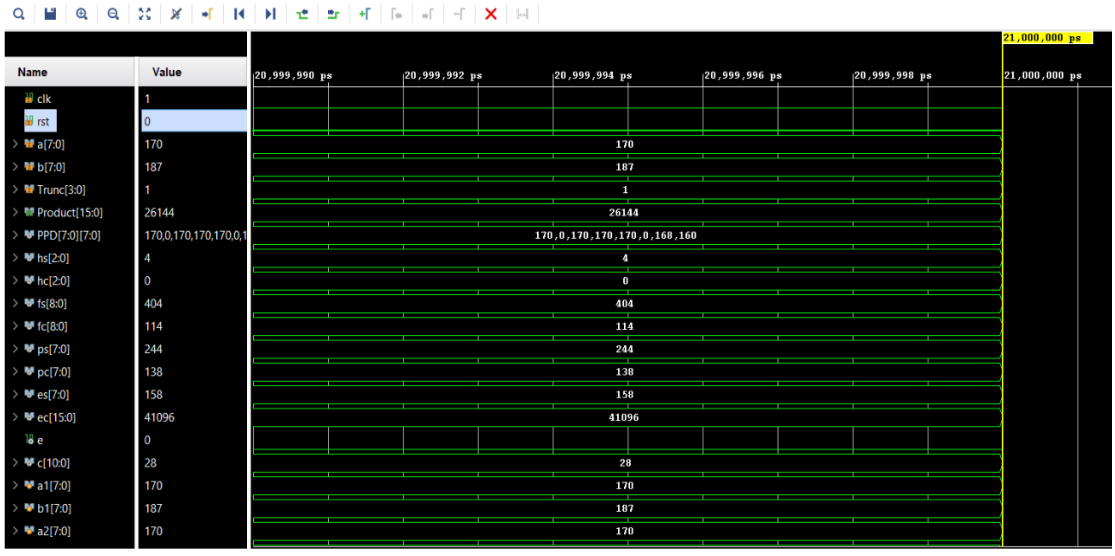

Fig 8: SIMULATION OF PROPOSED APPROXIMATE MULTIPLIER

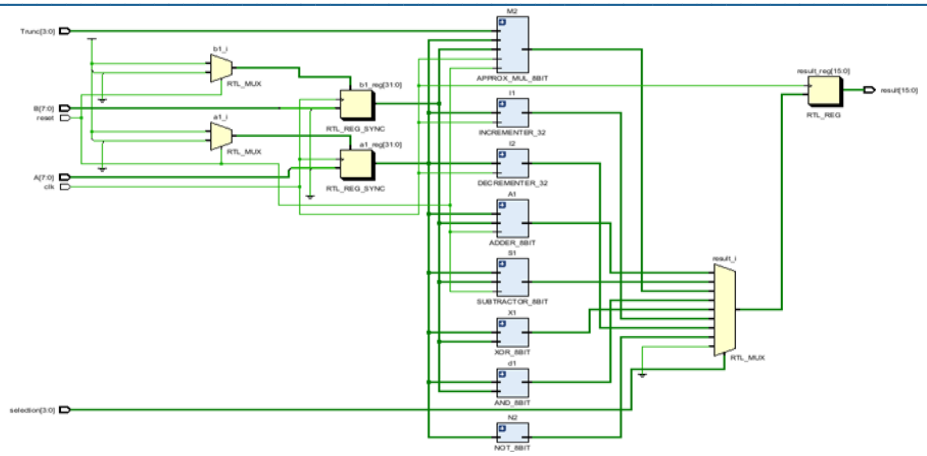

Fig 9: RTL SCHEMATIC OF 8-BIT ALU

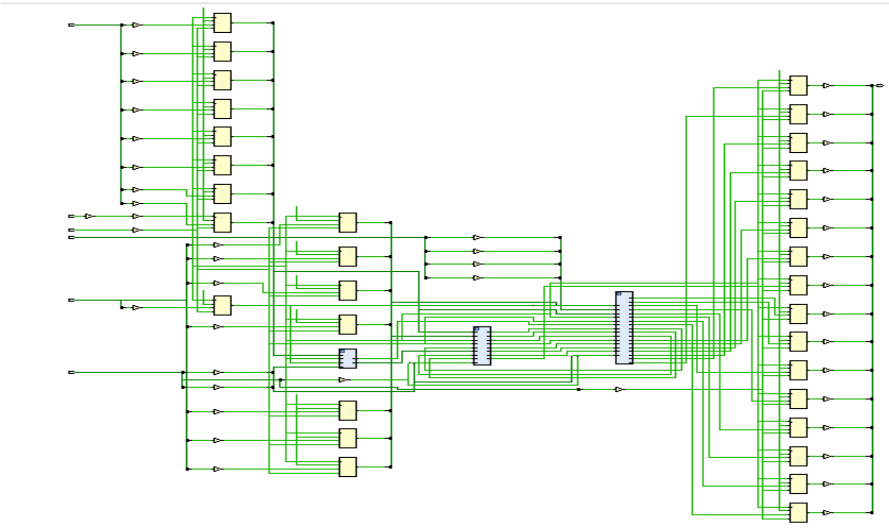

Fig 10: TECHNOLOGY SCHEMATIC OF 8 BIT ALU

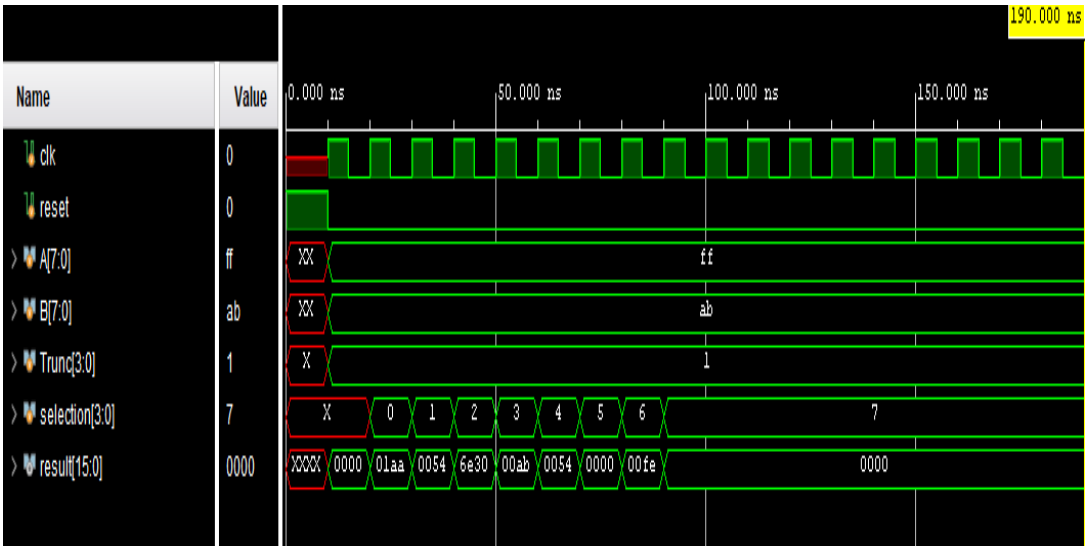

FIG11: SIMULATION WAVEFORM OF 8-BIT ALU

COMPARISON TABLE:

Artix 7	AREA (LUT)	POWER (mw)	DELAY (ns)	**Zynq 7000**	AREA (LUT)	POWER (mw)	DELAY (ns)
APPROX_MUL	86	175	8.563	APPROX_MUL	86	86	1039
Proposed_APP_MUL	86	176	5.540	Proposed_ APP_MUL	86	86	1039
ALU_ APPROX_MUL	142	153	5.548	ALU_ APPROX_MUL	142	142	558
ALU_ Prop_APPROX_MUL	142	156	5.540	ALU_ Prop_APPROX_MUL	142	142	716

TABLE 3: ARTIX7 and ZYNQ7000 boards results

Designing high-performance dynamically truncated approximation multipliers for VLSI applications involves a trade-off between power consumption and delay. Optimizing for one parameter typically means sacrificing another, and this trade-off is inherent in the design process of digital circuits. The goal is to find a balance that meets the specific requirements of the application, such as achieving a desired level of performance with minimal power consumption.

Insights from the Source:

Delay Reduction with Minimal Power Increase: The research outlined in the thesis demonstrates that it is possible to reduce delay by up to 43% with only a 1% to 8% increase in power using optimized algorithms and cell libraries. This indicates that careful design and optimization can mitigate the trade-off between power and delay to some extent

5. Discussion

In this work, the proposed multiplier employs dynamic truncation of partial products to adjust accuracy levels and incorporates a simple error compensation circuit to reduce errors. Compared to the Wallace tree multiplier, the adjustable approximate multiplier in our proposal demonstrates decreased delay and lower average power usage. When contrasted with other approximate multipliers, our design stands out for its smaller mean error distance and lesser average power consumption. Additionally, we intend to develop an 8-bit Arithmetic Logic Unit (ALU) that performs a variety of arithmetic and logical operations. This involves integrating the Multiplier component with the Approximate Multiplier, followed by synthesis and simulation processes in Xilinx Vivado. We apply the principles of pipelining to alleviate delay issues while managing power consumption, all while preserving area efficiency.

References

[1] F. -Y. Gu, I. -C. Lin and J. -W. Lin, "A Low-Power and High-Accuracy Approximate Multiplier With Reconfigurable Truncation," in IEEE Access, vol. 10, pp. 60447-60458, 2022, doi: 10.1109/ACCESS.2022.3179112.

[2] A. Wang, B. H. Calhoun, and A. P. Chandrakasan, Sub-Threshold Design for Ultra Low-Power Systems, vol. 95. New York, NY, USA: Springer, 2006.

[3] Q. Xu, T. Mytkowicz, and N. S. Kim, “Approximate Computing: A Survey,” IEEE Design & Test, vol. 33, no. 1, pp. 8-22, Feb. 2016.

[4] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate Hybrid High Radix Encoding for Energy- Efficient Inexact Multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 3, pp. 421– 430, Mar. 2018.

[5] C.–H. Chang, J. Gu, and M. Zhang, "Ultra-low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," IEEE Trans. Circuits and Syst. I: Reg. Papers, vol. 51, no. 10, pp. 1985-1997, Oct. 2004.

[6] A. Momeni, J. Han, P. Montuschi, and F. Lombardi. "Design and Analysis of Approximate Compressors for Multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984-994, Apr. 2015.

[7] M. Ha and S. Lee. "Multipliers with Approximate 4-2 Compressors and Error Recovery Modules," IEEE Embedded Systems Letters, vol. 10, no. 1, pp. 6-9, Mar. 2018.

[8] [13] D. Esposito, A. G. M. Strollo, E. Napoli, D. De. Caro, and N. Petra. "Approximate Multipliers Based on New Approximate Compressors," IEEE Trans. Circuits and Syst. I: Reg. Papers, vol. 65, no. 12, pp. 4169- 4182, De