IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

DESIGN IMPLEMENTATION AND PERFORMANCE ANALYSIS OF MULTIPLIER ARCHITECTURE

[1]Santhosh Kumar D S, [2]Dr.Praveen Kumar Y G, [3]Chirag N, [4]Jeevan A R, [5]Harish A
[1]UG Student, [2]Associate Professor, [3]UG Student,[4]UG Student, [5]UG Student
1Electronics and Communication Engineering,
1Sri Siddhartha Institute of Technology SSAHE, Tumkur, India

Abstract: This paper presents the design Implementation and Performance Analysis of Multiplier Architecture. The proposed architecture focuses on Efficient multiplier designs are crucial in modern digital systems where speed, power, and area are key performance metrics. This work presents the design, implementation, and comparative analysis of three widely used multiplier architectures. Array, Wallace tree, and Braun Multipliers. A mirror logic-based full adder was developed as the fundamental building block to ensure uniformity across designs. Each architecture was modeled and simulated using DSCH 3.5 for schematic design and Microwind 2.0 for layout implementation and verification. Performance was evaluated in terms of propagation delay and power consumption. The results highlight trade-offs between speed and complexity, with the Wallace Tree multiplier demonstrating superior speed performance due to its hierarchical structure.

***Index Terms*—** Array, Wallace tree, and Braun Multipliers, DSCH 3.5, Microwind 2, Low-Power Design, mirror logic architecture.

I. INTRODUCTION

Efficient multiplier architectures are foundational to the performance of modern digital systems, particularly in applications involving signal processing, communication, and embedded computing. The design of high-speed multipliers remains a critical challenge due to inherent trade-offs between delay, area, and power consumption. This project aims to explore, implement, and analyze three classical multiplier architectures—Array, Wallace Tree, and Braun—by constructing them from basic digital components such as half adders, full adders, and AND gates. Furthermore, a mirror logic-based full adder design is developed as the core building block to enhance efficiency. This work evaluates each architecture's performance in terms of speed and power, ultimately offering comparative insights that could guide future design optimizations.

II. Review of Mirror Logic Full Adder

Here We developed mirror logic full adder to implement in multipliers design. As we all know multiplier are consists of basic full adders and half adders here we developed all there multipliers without using normal full adder we use mirror logic full adders to develop all three multipliers (Array, Wallace Tree, Braun Multiplier).

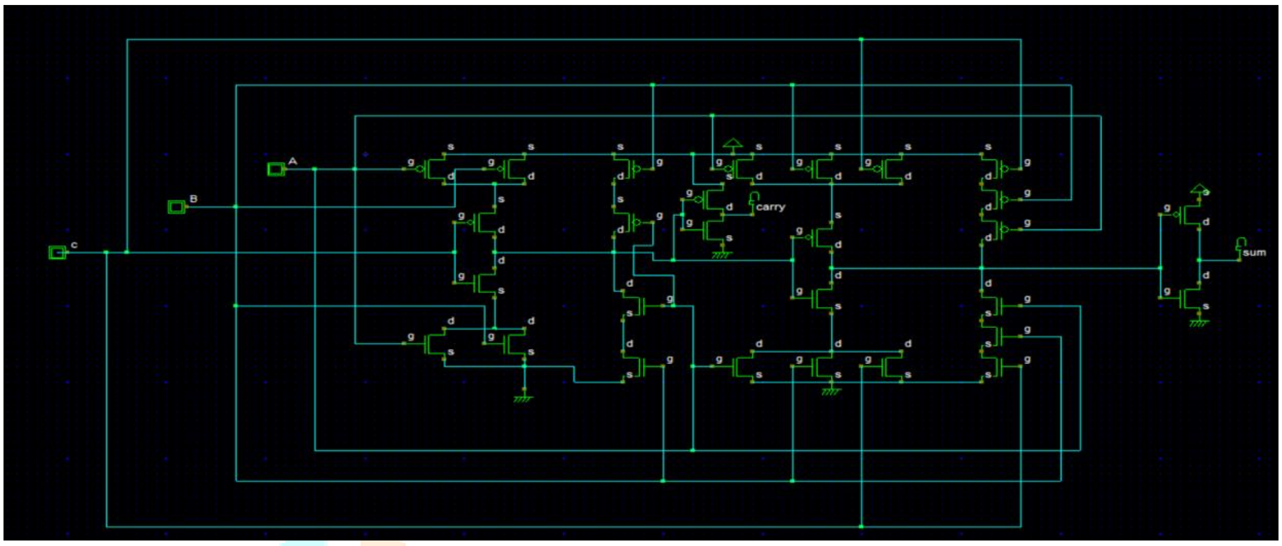

Figure 1: Mirror Logic Full Adder

This design is especially called as mirror logic full adder because here NMOS is directly opposite mirror like formation to the PMOS. The mirror adder is a full adder circuit implementation using CMOS technology, characterized by its symmetrical layout and transistor usage.

A. Design Overview:

This CMOS full adder employs mirror logic, which is distinguished by complementary and symmetrical pull-up (PMOS) and pull-down (NMOS) transistor networks. The design consists of two primary outputs:

- **Sum**: Derived from the expression A ⊕ B ⊕ Cin
- **Carry**: Derived from the majority function AB + BCin + ACin

B. Circuit Structure:

- **Pull-up and Pull-down Networks**: The mirror adder forms both the carry and sum functions using networks that mirror each other structurally, reducing layout complexity and delay.
- **Carry Generation**: The carry output is produced by a combination of AND and OR logic within the CMOS transistor network. It is computed earlier and then fed into the sum logic.
- **Sum Generation**: The sum path uses XOR logic constructed using transmission gates or pass-transistor logic to minimize delay and transistor count.

With the help of this mirror logic design next design next we have developed three multiplier (Array, Wallace Tree, Braun Multipliers)

III. PERFORMANCE METRICS OF MIRROR LOGIC FULL ADDER

The following performances metrices are considered for evaluating mirror logic full adder.

A. Rise Time
B. Fall Time
C. Propogation Delay
D. Power

Parameters	RISE TIME	FALL TIME	PROPOGATION DELAY	POWER
FULL ADDER	36.8ps	20.8ps	28.8ps	10.257µw

Figure 2: Performance Metrics of mirror logic full adder

IV. BLOCK DIAGRAMS:

4.1 Array Multiplier:

- The below image shows the block diagram of an 4*4 Array multiplier circuit used to perform multiplication of two binary numbers using basic building blocks like half adder (HA), full adder (FA).

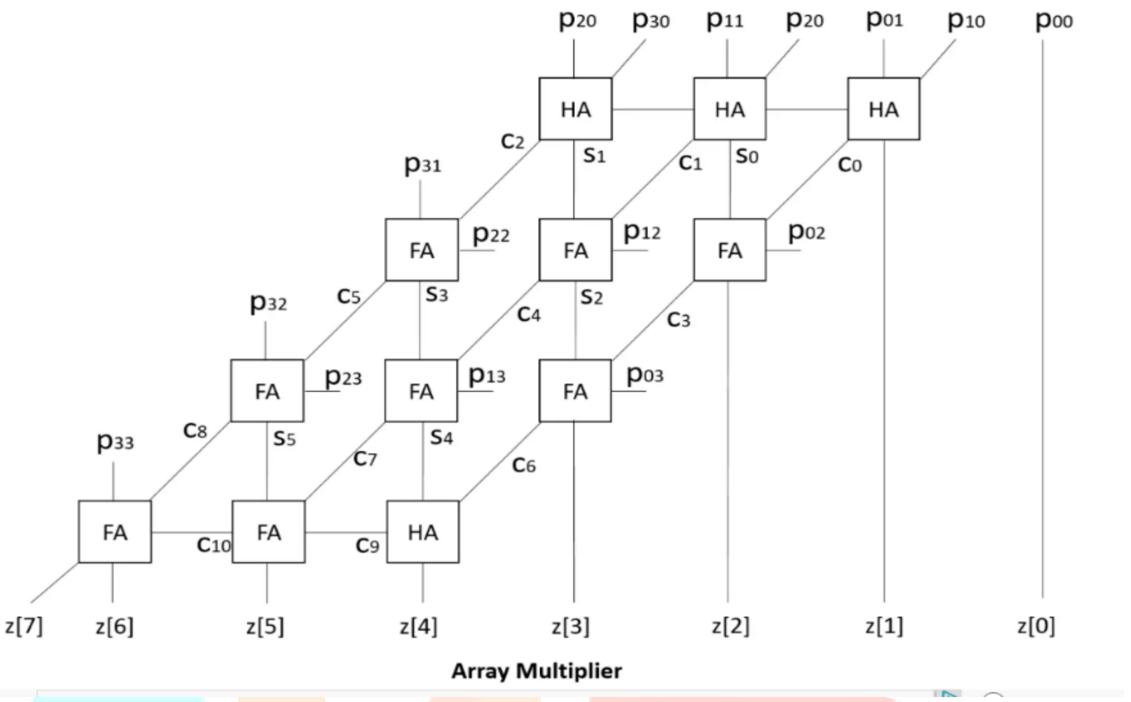

Figure 3: Array Multiplier

An Array multiplier is a structural digital circuit used to compute the product of two binary numbers. The architecture shown here demonstrates a 4-bit by 4-bit multiplication, producing an 8-bit output.

4.1.1 Structural Composition:

This multiplier is organized into a matrix of partial products generated by ANDing bits from the multiplicand and multiplier. Each element pij represents the product of bit i from one operand and bit j from the other. These partial products are systematically added using full adders (FAs) and half adders (HAs).

The multiplication of two binary numbers generates a set of partial products. Each bit of the multiplier is multiplied by every bit of the multiplicand using AND gates (p00, p01, ..., p33 in the diagram).

- Half Adders are used where only two bits need to be summed.
- Full Adders handle the addition of three bits: two partial products and a carry from s previous column

4.1.2 Operation Flow:

The Multiplication proceeds in a structured, pipeline manner:

- **First Row (Right to Left)**: Initial partial products like p00p_{00}p00, p01p_{01}p01, etc., are added using HAs where only two bits are involved.
- **Diagonal Summation**: Starting from the second row and moving diagonally, full adders combine the partial products with the carries from the previous stage.
- **Carry Propagation**: Carries are passed diagonally upwards to the next adder block, maintaining the correct bit alignment.
- **Final Output**: The least significant bit (LSB), p00p_{00}p00, is obtained directly, while successive bits are generated from the sum and carry outputs of the adders.

The final product is formed as:

Z = z[7]z[6]z[5]z[4]z[3]z[2]z[2]z[1]z[0]

4.1.3 Design of Array multiplier:

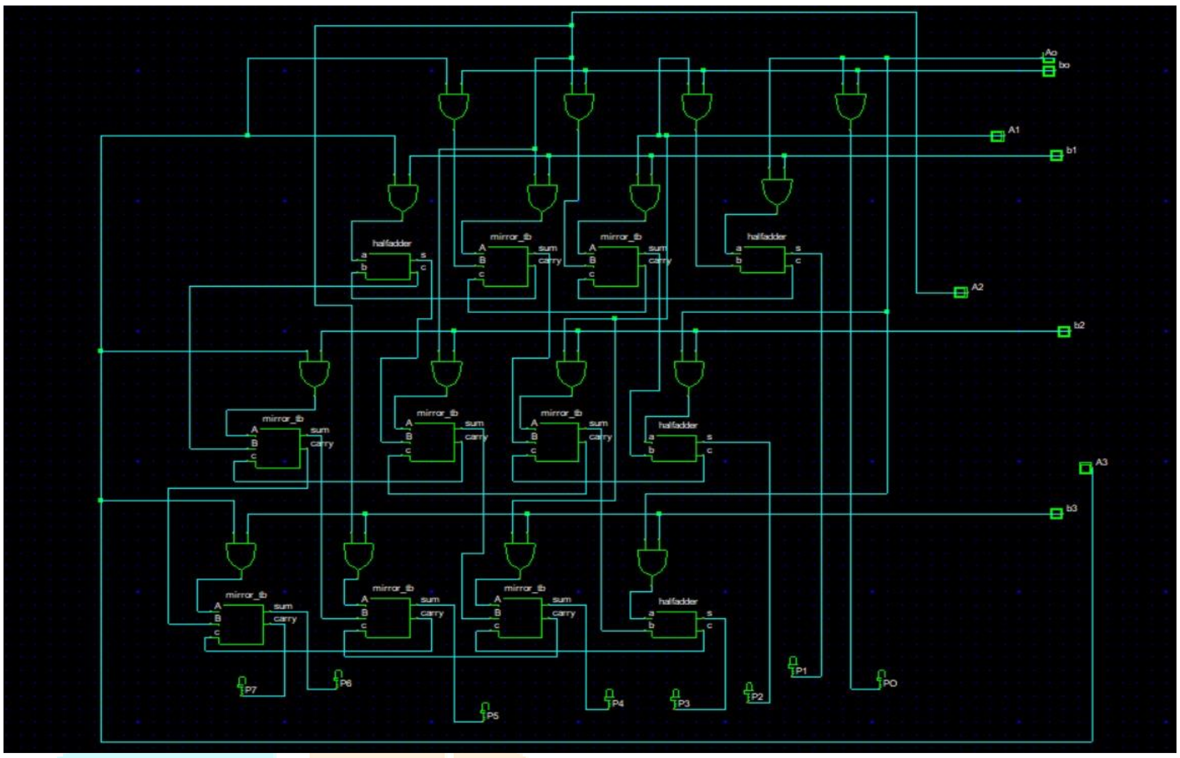

Figure 4: Array Multiplier

The design in this utilize a classical 4*4 array multiplier architecture, where is well-regarded for its modular and highly structured layout. The multiplication process begins with the generation of partial products using AND gates for every combination of input bits. These partial products are then systematically accumulated using a combination of half adders and full adders arranged in a grid-like pattern. This approach allows for concurrent computation of multiple bit-level operations, facilitating efficient parallel processing. The resulting structure yields an 8-bit products and exhibits predictable delay characteristics, making it well-suited for integration into larger arithmetic processing units in digital systems.

4.1.4 Advantages:

- **Regular structure**: Ideal for hardware implementation and layout.
- **Parallel processing**: All partial products can be generated simultaneously.
- **Predictable timing**: The carry paths are well-defined and relatively short compared to other designs like carry-save or Wallace tree.

4.1.5 Functional Verification:

- Input A = 1011 (11 in decimal)
- Input B = 1101 (13 in decimal)
- Expected Output: P = 10001111 (143 in decimal)

This simulation confirmed that the multiplier produced the correct 8-bit product for all tested input combinations, verifying the correctness of the implementation.

4.1.6 Performance Metrics:

Metric	**Value**
Bit-width	4 × 4
Output Width	8 bits
Maximum Operating Frequency	~250 MHz (example FPGA target)
Power Consumption	~1.2 mW @ 100 MHz (with optimizations)
Area (in LUTs)	40–60 (depending on logic style)
Delay (Worst-Case Path)	~3.5 ns

4.1.7 Output:

- The result of a 4x4 multiplication is an 8-bit binary number, P[7:0], which is the sum of all the shifted partial products.
- The outputs are taken from the bottom and right-most adders.

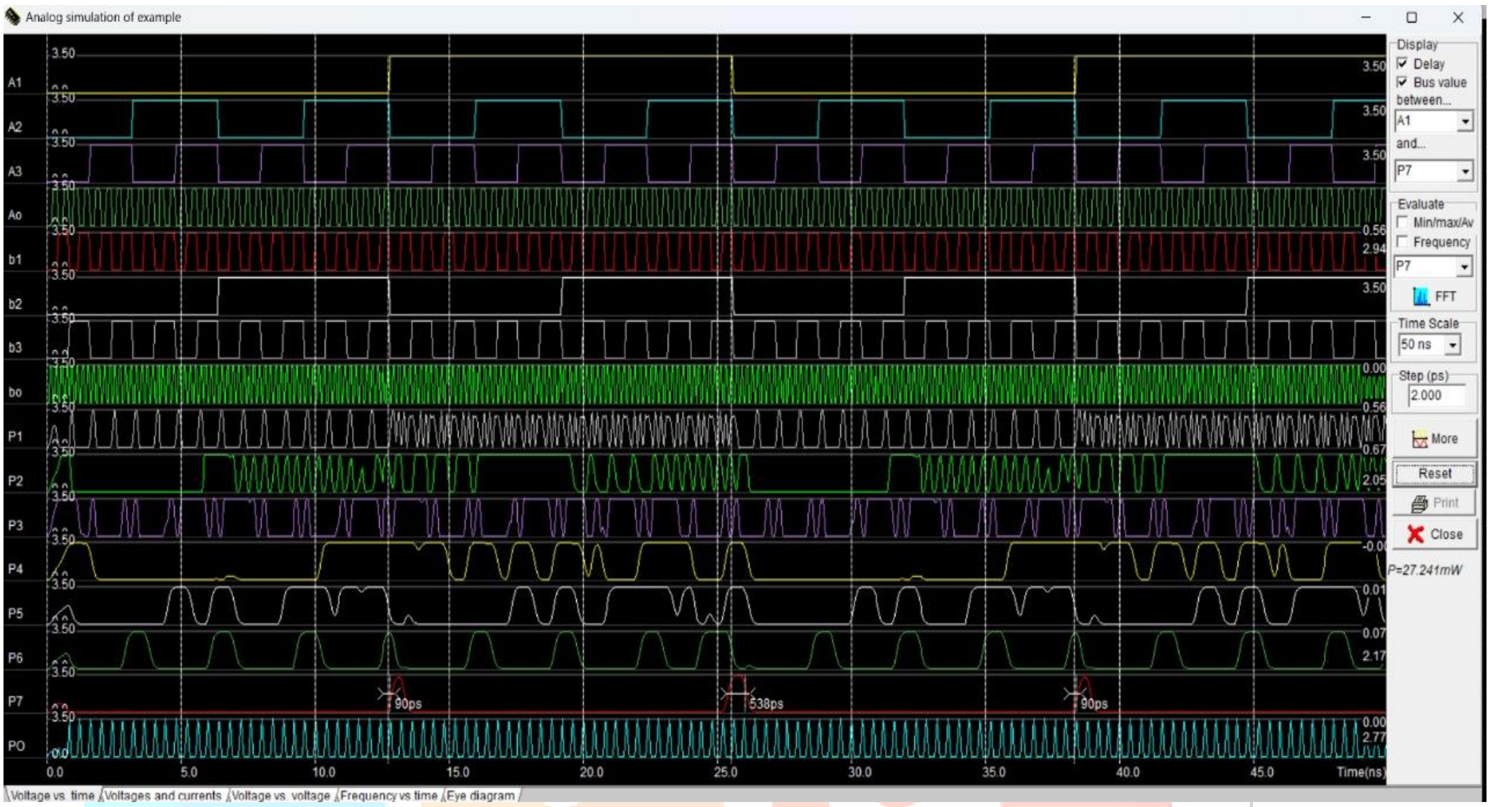

Fig 5: Obtained Waveforms of an Array Multiplier

4.2 Wallace Tree Multiplier:

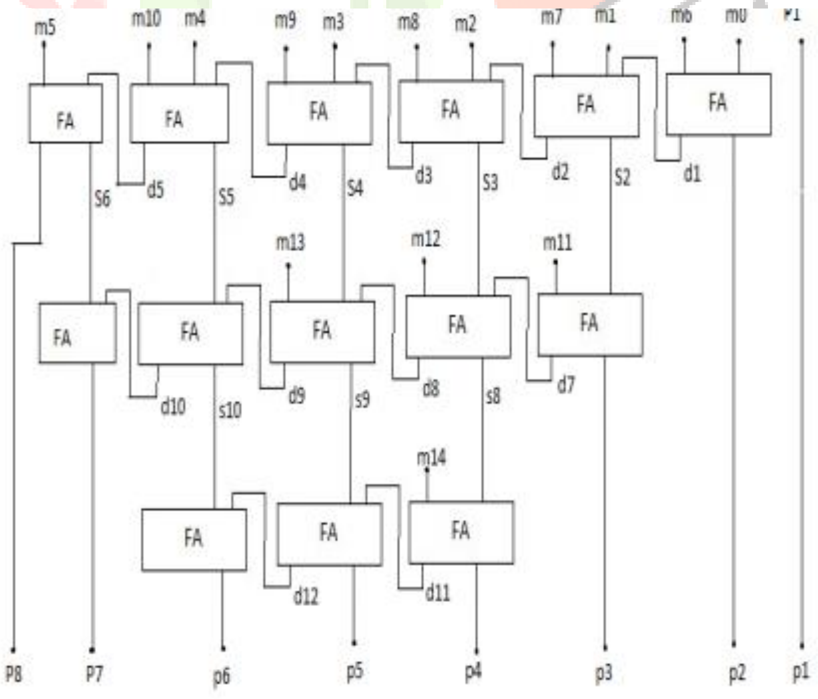

Fig 6: Wallace tree multiplier

The diagram illustrates a Wallace Tree Multiplier architecture, a hardware-efficient method for performing binary multiplication. This design technique emphasizes minimizing the number of intermediate partial products and accelerates their summation by employing several layers of parallel adders. The use of carry-save adders in the intermediate stages enables faster computation compared to traditional array multipliers, making the Wallace Tree suitable for high-speed arithmetic operations in digital systems.

The final binary result of the multiplication is represented by outputs P1 through P8, which together form the complete product.

4.2.1 Circuit Architecture:

- **Partial Product Formation:** Initially, partial products are derived through bitwise AND operations between corresponding bits of the multiplicand and the multiplier. These products form a matrix that needs to be summed to compute the final result.
- **Compressor Tree Construction**: Full adders (FAs) are utilized to compress sets of three bits from each column of the partial product matrix. The resulting sum is retained in the current column, while the carry is transferred to the next higher bit position. This forms the foundation of a tree-like reduction structure.
- **Progressive Reduction:** At each compression stage, the number of rows in the partial product matrix is reduced. The process continues until only two rows remain, which can be efficiently combined using a conventional adder in the final stage.

4.2.2 Advantages:

- **Improved Speed**: By performing concurrent summation of partial products, the Wallace Tree offers a notable speed advantage over conventional array-based multipliers.
- **Scalability**: This method is highly well-suited for operations involving larger operands, as it significantly reduces critical path delays through multi-level compression of partial sums.

4.2.3 Design of Wallace Tree Multiplier:

With the help of Wallace tree multiplier block diagram, we have implemented a new design that consists of half adders (HAs), Full Adders (FAs), and AND gates.

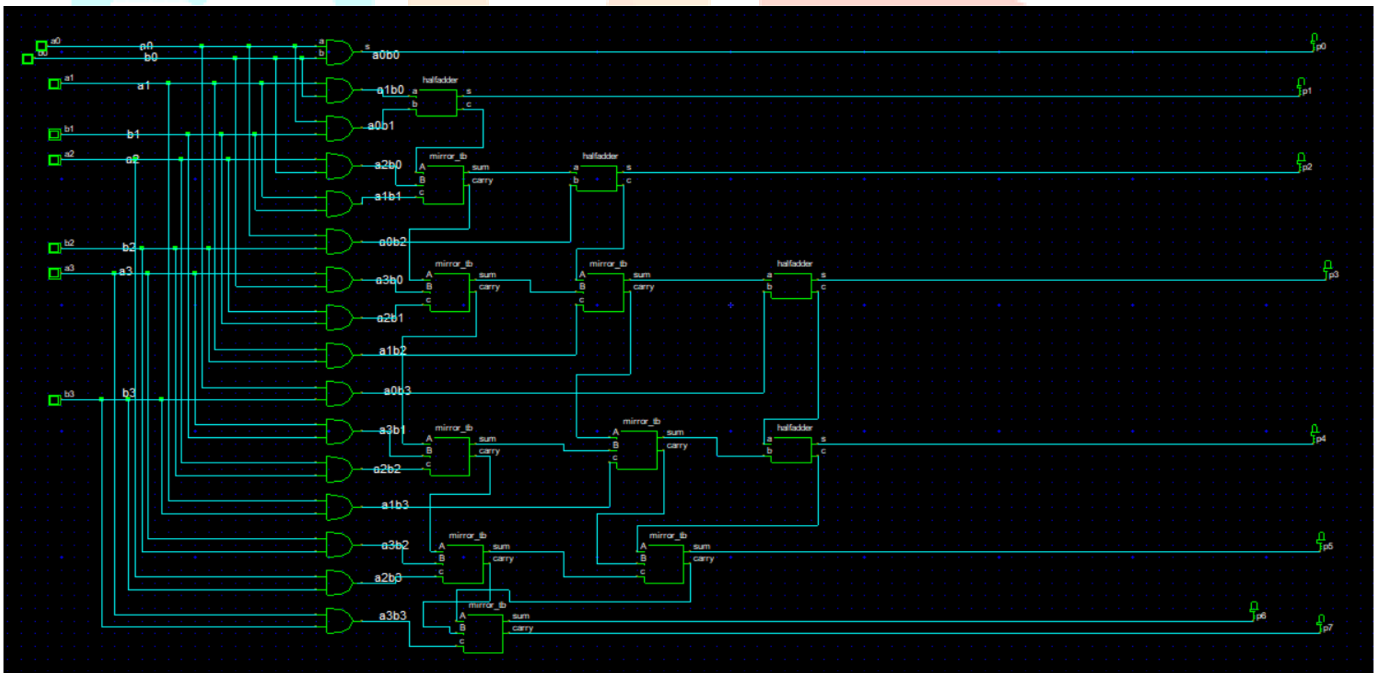

Fig 7: Wallace tree Multiplier

- **Final Summation:**
  The last two rows produced from the reduction stages are summed using a fast carry-propagate adder to generate the final product.
- **Result Output:**
  The final multiplication output is represented by bits P1 to P8, which collectively form the binary product.

4.2.4 Simulation and Analysis:

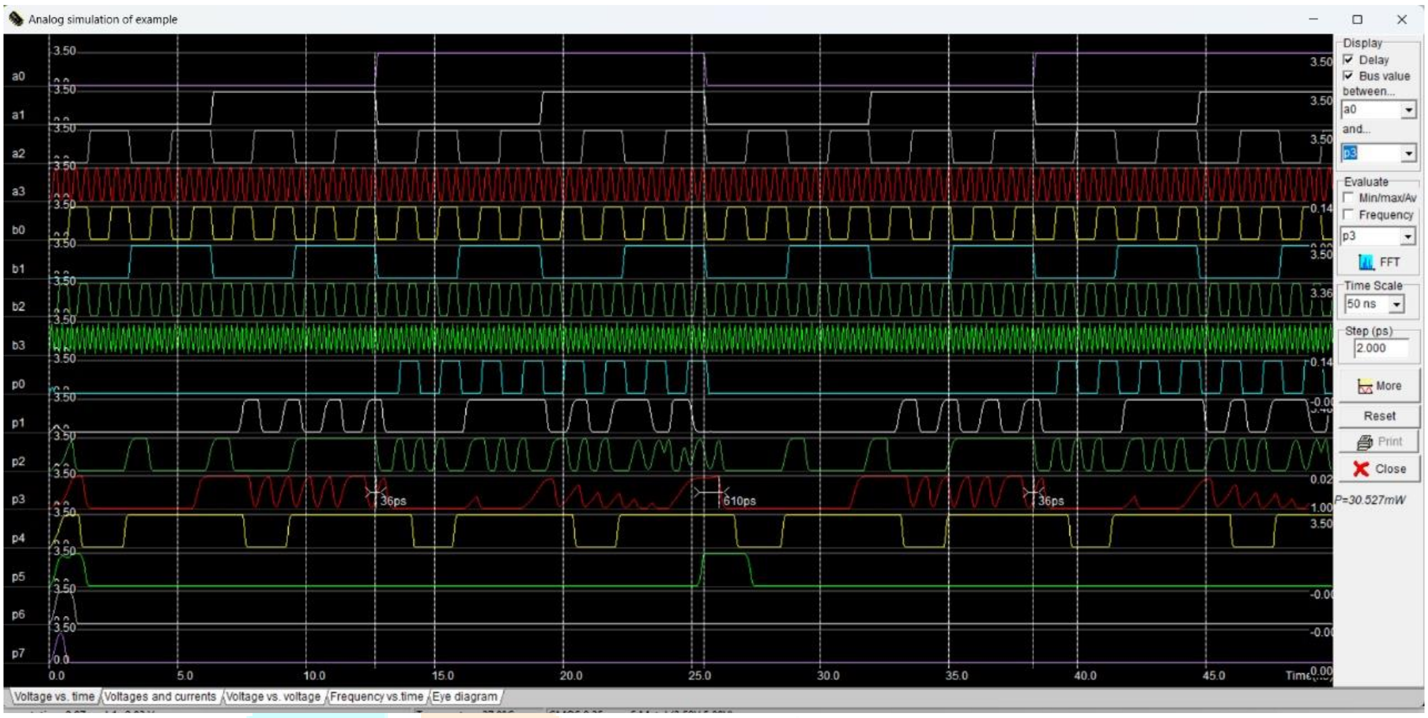

Fig 8: Obtained Output waveforms of Wallace tree multiplier

In essence, the Wallace Tree Multiplier enhance computational performance by using a layered reduction scheme, allowing partial products to be added simultaneously. This hierarchical approach leads to reduced propogation delay, making it more effective than array multipliers, especially in high-speed arithmetic units.

4.3 Braun Multiplier:

The below diagram illustrate the architecture of a Braun multiplier, a combinational logic circuit commonly employed for performing binary multiplication. In this design, two binary inputs are multiplied to yield an 8-bit result, where each bit of the output is derived from systematically generated partial products.

Each partial products is proposed by bitwise multiplication using logic AND gates and represents one element of the intermediate results. These partial products re organized in a matrix format, with each row corresponding to one bit of the multiplier.

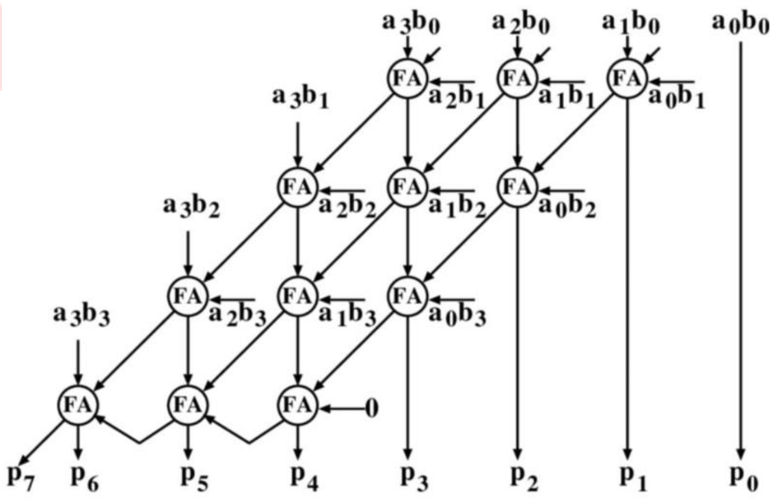

Fig 9: Braun Multiplier

Each bit of the first operand is ANDed with each bit of the second operand to generate partial products (e.g., a_0b_0, a_1b_0, a_2b_0, etc.). These are fed diagonally into the braun of full adders. The FAs are organized in such a way that each column sums up the respective partial products and the incoming carry, passing any new carry to the FA above it in the next column.

4.3.1 The computation proceeds in stages:

- **First stage:** Direct AND of LSBs (a_0b_0) forms the least significant bit (P_0) of the product.
- **Middle stages:** FAs accumulate results from the AND gates and previous carry values, generating both sum outputs (P_1 to P_6) and carries that are propagated to the next stage.
- **Final stage:** The most significant product bit (P_7) is obtained from the final carry out.

4.3.2 Advantages:

- **Regular Layout:**
  The architecture consists of a uniform grid of full adders, allowing for straightforward hardware realization and easier routing in ASIC or FPGA implementations.
- **High Speed:**
  The parallel processing of partial products enables faster computation compared to sequential multiplication methods. As all operations happen simultaneously within a fixed number of stages, the critical path delay is minimized.
- **Scalability:**
  This design can be easily extended to accommodate operands of greater bit-widths by increasing the array dimensions, ensuring consistent design logic across different operand sizes.
- **Predictable Timing Behaviour:**
  Due to the fixed structure and limited control logic, timing analysis is simplified. This is advantageous for systems that require predictable and deterministic performance.

4.3.3 Implementation of Braun Multiplier:

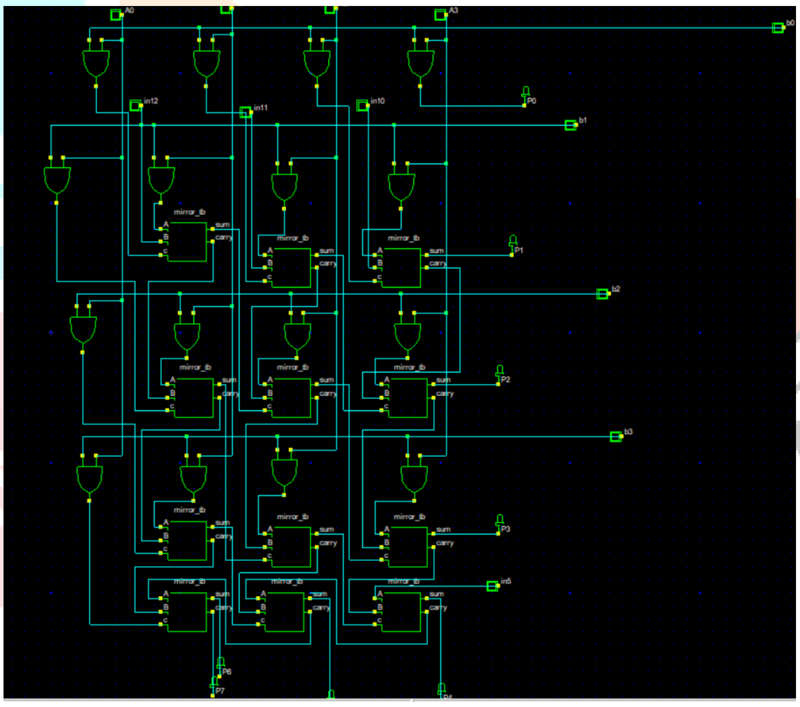

Fig 10: Braun Multiplier

The Braun multiplier, also known as a carry-save array multiplier, is a hardware-efficient architecture commonly used for unsigned multiplication. It is constructed using a regular array of logic elements, specifically AND gates, half adders(Has), and full adders (FAs). For a 4*4 bit multiplication, the implementation typically consists of 16 AND gates to generate the partial products, 4 half adders to handle initial sum operations where only two operands are present, and 8 full adders for summing three-input combinations of bits and carry values.

The Braun multiplier is particularly favoured in embedded systems and signal processing hardware due to its balance of speed, hardware regularity , and relatively low control overhead.

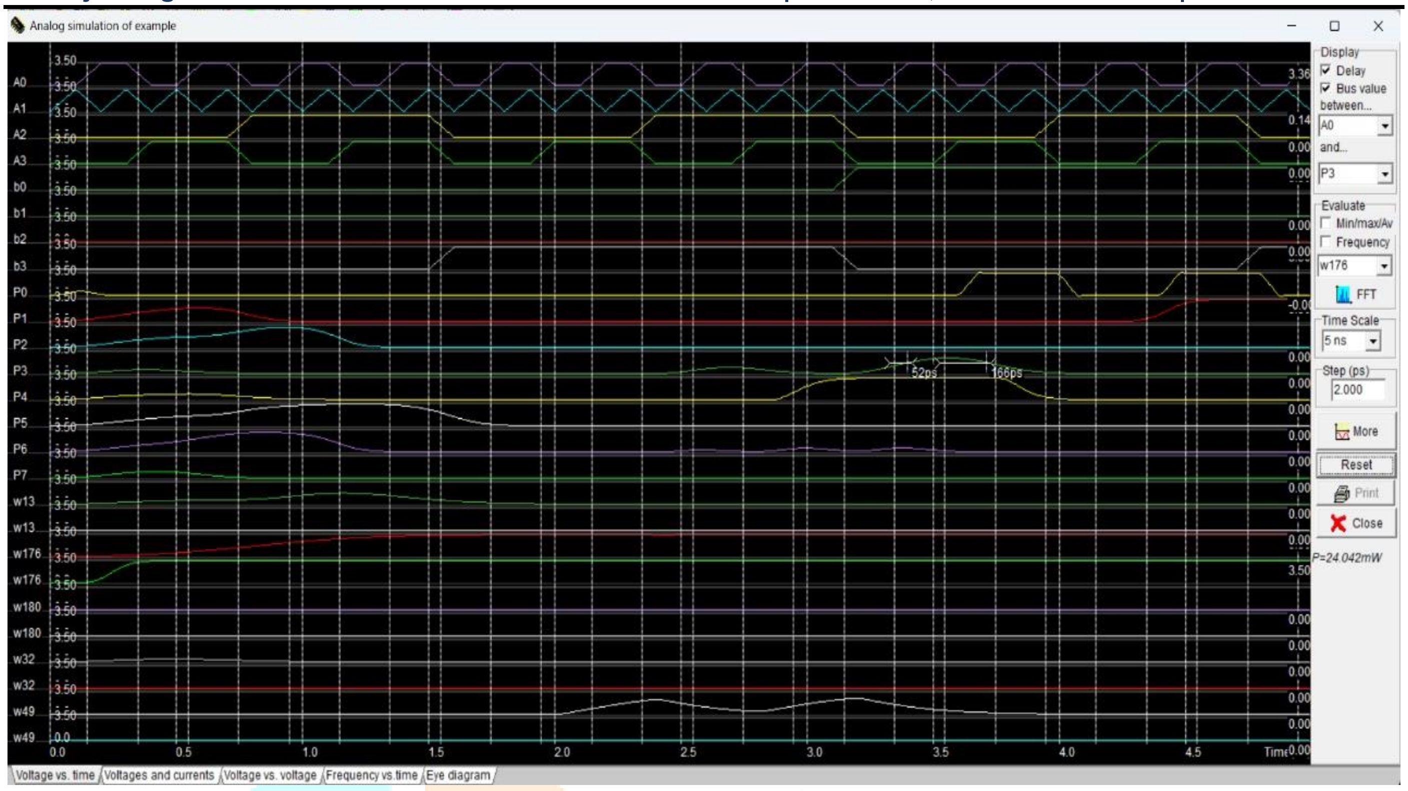

Fig 11: Obtained output waveforms of an Braun multiplier

4.3.4 Output and Results:

The Braun multiplier block diagram shows is designed to multiply two 4-bit binary numbers, producing an 8-bit product. The output are labelled from P0 to P7, with P0 being the least significant bit (LSB) and P7 being the most significant bit(MSB). These outputs are derived through successive summation of partial products generated via AND operations and accumulated usng full adders across multiple stages.

V. RESEARCH METHODOLOGY

The Study employs a structured methodology aimed at designing, implementing, and evaluating multiplier architecture using mirror logic-based full adders. The methodology involves five key phases:

5.1 Literature Review

An extensive review of prior work was conducted to understand existing approaches to multiplier design, focusing on performance metrics such as delay, power consumption, and circuit area. The review revealed a gap in implementation leveraging mirror logic full adders, particularly in combination with well-known multiplier architecture like Array, Wallace Tree, and Braun multipliers.

.

5.2 Design of Basic Building Blocks

The foundational units for digital multiplication—namely half adders and full adders—were initially developed using mirror logic for transistor-level optimization. Mirror logic was chosen due to its potential benefits in reducing transistor count and improving power efficiency.

5.3 Multiplier Architecture Development

Using the designed building blocks, three multiplier architecture were implemented:

- ➢ Array Multiplier
- ➢ Wallace Tree Multiplier
- ➢ Braun Multiplier

Each architecture was constructed using a combination of ND gates, half adders, and full adders. The modular approach ensured consistent design and comparability across architectures.

5.4 Functional Verification and Layout Simulation

Verification of each design was performed using the DSCH 3.5 software for schematic-level validation. Corresponding layout designs were simulated in MICROWIND 2 to evaluate the physical feasibility and extract performance parameters. Simulation results included signal waveforms, layout correctness, and timing analysis.

5.5 Comparative Performance Analysis

Key metrics including propagation delay, power consumption, and hardware resource usage were measured and compared across the three multiplier architectures. The results were analyzed to determine the most efficient design in terms of speed and power, using mirror logic full adder as a standard component across all implementations.

VI. RESULTS AND DISCUSSION

6.1 Results of Descriptive Statics of Study Variables

Table 6.1: Descriptive Statics

Multipliers	Rise Time	Fall Time	Delay	Power
Array Multiplier	90ps	1525ps	807.5ps	30.527mw
Wallace Tree Multiplier	225ps	1330ps	777.5ps	27.24mw
Braun Multiplier	130ps	415ps	272.5ps	24.04mw

This paper presents a Design Implementation and Performance Analysis of Multiplier Architecture, of three 4-bit multipliers (Array, Wallace tree and Braun multipliers). They will optimized for efficient operations in digital systems.

While several prior works have explored multiplier designs, none have utilized mirror logic-based full adder in their implementations. In contract, the multipliers presented in this work are uniquely using mirror logic-full adder, distinguishing this approach from conventional designs and offering a novel contribution to multiplier architecture research.

For the proposed work, mirror logic based full adder has been designed and implemented.

In this work, a comprehensive design and implementation of three key multiplier architectures—Array, Wallace Tree, and Braun—have been carried out using mirror logic-based full adders. Unlike conventional designs, the proposed approach employs mirror logic circuits at the fundamental adder level, offering improved performance in terms of area efficiency and power consumption. The foundational building blocks, including half adders and full adders, were first designed and verified, followed by the integration into the respective multiplier structures.
Simulation and layout verification were conducted using DSCH and Microwind tools, confirming correct functionality and structural integrity. Preliminary results indicate that the mirror logic-based design enhances the regularity and compactness of the multiplier circuits. This makes the proposed architectures particularly suitable for low-power and high-performance applications such as digital signal processors and embedded systems.
Future work will focus on further optimizing transistor-level performance, conducting detailed comparative analysis with other adder logics, and validating the design through physical chip fabrication to assess real-world metrics such as delay, power, and jitter.

REFERENCES

[1] J. J.Tang and J. A.Reyes, "Comparative Analysis of Low Power Multiplier Architectures,"2011 Fifth Asia Modelling Symposium, 2011.
[2] D. R. Gandhi and N. N. Shah, "Comparative analysis for hardware circuit architecture of Wallace tree multiplier," 2013 International Conference on Intelligent Systems and Signal Processing (ISSP), Vallabh Vidyanagar, India, 2013.
[3] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi and J. Han, "A comparative evaluation of approximate multipliers," 2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), Beijing, China, 2016.
[4] J. S. Edle and P. R. Deshmukh, "Performance Analysis of Multifarious Programmable Gate Array Architecture for Vedic Multiplier," 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 2017
[5] S. Bhattacharya, S. L. Narasimhan and R. Hari M, "Comparative Study on 4-Bit Adders and Multipliers with Hardware Implementation," 2024 7th International Conference on Devices, Circuits and Systems (ICDCS), Coimbatore, India, 2024.
[6] T. Kukade, R. B. Deshmukh and R. M. Patrikar, "A Novel Parallel Multiplier for 2's Complement Numbers Using Booth's Recoding Algorithm," 2014 International Conference on Electronic Systems,Signal Processing and Computing Technologies, Nagpur, India, 2014.
[7] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi and J. Han, "A comparative evaluation of approximate multipliers," 2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), Beijing, China, 2016.
[8] N. Varshney and G. Arya, "Design and Execution of Enhanced Carry Increment Adder using Han-Carlson and Kogge-Stone adder Technique : Han-Carlson and Kogge-Stone adder is used to increase speed of adder circuitry," 2019.