ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

System-Level PPA Trade-Off Analysis of Adder-Based MAC Architectures Using Vedic Multipliers on Zynq-7000 FPGA

Subhadip Ghanta[1]; Sanket Shaw[2]; Abdul Barish Khan[3]; Debjyoti Sarkar[4]; Soumik Das[5]

[1,2,3,4,5]Student, Department of Electronics and Communication Engineering
[1,2,3,4,5]Meghnad Saha Institute of Technology, Kolkata, India

[1]ORCID Id: 0009-0000-4753-6679; [2]ORCID Id: 0009-0009-0976-2099

Publication Date: 2026/03/19

Abstract: Multiply-Accumulate units are basic components of digital signal processing systems as well as embedded systems, where computational throughput as well as power constraints are of great importance. In this paper, a Power Performance Area trade-off of Vedic Multiplier-based Multiply-Accumulate units with five different adder styles, namely Ripple-Carry Adder, Carry-Skip Adder, Carry-Look-Ahead Adder, Brent-Kung Adder, and Kogge-Stone Adder, is presented. In this paper, a novel Multiply-Accumulate architecture based on a hierarchical 16×16 Vedic Multiplier with a 32-bit accumulation path is presented, which has been implemented on a Zynq-7000 FPGA platform based on the Zed Board platform. The architectures are tested with both 16-bit and 32-bit datapath configurations in terms of power consumption, propagation delay, Power Delay Product, as well as Energy Delay Product, along with hardware utilization. The experimental results show that the Kogge-Stone Adder-based Multiply Accumulate architecture has the least propagation delay of 3.2 ns as well as 5.2 ns for 16-bit as well as 32-bit datapath configurations, respectively, achieving the least PDP of 304 pJ as well as 728 pJ, respectively, compared to other adder styles. In contrast, the Ripple Carry Adder consumes the least power of 40 mW as well as 83 mW for 16-bit and 32-bit datapath configurations, respectively, utilizing the least hardware resources, but has the highest propagation delay of 12.5 ns and 21 ns, respectively. Among all architectures, the Brent-Kung Adder offers the best trade-off among power, delay, and hardware utilization. The adder-based Multiply-Accumulate architectures are suitable for implementing various high-performance as well as power-efficient applications, including digital signal processing, image processing, and machine learning accelerators.

Keywords: *Multiply–Accumulate (MAC), Vedic Multiplier, Adder, Power–Performance–Area (PPA), Zynq-7000 FPGA.*

How to Cite: Subhadip Ghanta; Sanket Shaw; Abdul Barish Khan; Debjyoti Sarkar; Soumik Das (2026) System-Level PPA Trade-Off Analysis of Adder-Based MAC Architectures Using Vedic Multipliers on Zynq-7000 FPGA.
International Journal of Innovative Science and Research Technology, 11(3), 1374-1393.
https://doi.org/10.38124/ijisrt/26mar690

I. INTRODUCTION

Multiply–Accumulate (MAC) units are fundamental computational blocks widely used in modern digital systems, including digital signal processors (DSPs), communication processors, control systems, and edge Artificial Intelligence (AI) devices. In such systems, MAC operations form the core arithmetic workload, performing repeated multiplication and accumulation of data streams. As these operations are performed at extremely high rates, the performance of the system, power consumption, as well as utilization of the hardware, are significantly affected by the effectiveness of the MAC architecture. In various high-throughput industrial applications, such as real-time signal processing, image/video processing, radar systems, wireless communication systems, as well as embedded machine learning systems, the rate of performing the MAC operations can be extremely high, thereby necessitating the optimization of the MAC architecture.

As a result, the optimization of internal arithmetic components within MAC architectures has become an important research and industrial challenge.

Among various multiplier designs, Vedic multipliers based on ancient Indian Vedic mathematics have gained considerable attention in digital hardware design. The Urdhva Tiryakbhyam (vertical and crosswise) algorithm enables highly parallel multiplication with a regular and scalable architecture. This structure offers reduced

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

multiplication latency, efficient hardware mapping, and suitability for FPGA-based implementations. Due to these advantages, Vedic multipliers are frequently used in high-speed arithmetic units and signal processing architectures.

However, in the context of the Vedic multiplier-based MAC architecture, the accumulation stage involving the adder circuit plays a vital role in determining the performance of the system. Since the MAC operation involves the repeated addition of the results obtained from the multiplication operations, the performance of the adder circuit has a considerable effect on the overall efficiency of the MAC operation. Various adder circuits provide trade-offs in terms of power consumption, speed, and hardware complexity. For instance, the use of simple adder's results in area complexity at the cost of increased delay time.

In order to bridge the research gap in this area, this paper presents a systematic evaluation of different adder architectures as they are integrated with a Vedic multiplier-based MAC architecture. In this regard, the proposed MAC architecture is designed with a 16x16 Vedic multiplier and a 32-bit accumulation path with different adder architectures such as the RCA, CSKA, CLA, Brent-Kung Adder, and the Kogge-Stone Adder. The proposed MAC architecture is implemented and analyzed using the ZedBoard platform with the Xilinx Zynq-7000 FPGA device. The performance of the different MAC architectures is evaluated in terms of power consumption, critical path delay, area utilization, and PDP in order to determine the efficient MAC architecture for low-power and high-speed MAC applications.

The results of this study provide valuable insights into the design trade-offs between different adder architectures when used in practical MAC systems, offering useful guidelines for developing energy-efficient arithmetic units in modern embedded and edge-computing hardware.

This paper is organized as follows: Section II presents the related work, Section III describes the proposed Vedic multiplier-based MAC architecture, Section IV explains the RTL design and simulation analysis, Section V presents the implementation results and performance evaluation, Section VI provides the discussion of the results, and Section VII concludes the paper with future research directions.

II. LITERATURE REVIEW

The Brent-Kung Adder (BKA) offers a balanced alternative for improved speed with moderate area [1, 2, 3, 4, 10]. The Kogge-Stone Adder (KSA) offers the fastest operation in applications where speed is essential, but at a cost of significantly higher area and power consumption [2,3,4,9]. The Ripple Carry Adder (RCA) is the easiest option, but it is slow and area and power. To achieve a balanced performance, Carry Select Adders (CSLA) are often preferred, offering better speed than RCA while being more compact than KSA. Further area reduction can be achieved by replacing RCA blocks with Binary-to-Excess-1 Converters (BEC) and other optimized logic structures [5,6,10].

Beyond conventional exact adders, recent research has introduced hybrid and application-specific architectures that combine different adder types or integrate approximation techniques in specific bit ranges. For example, modified CSLAs using BKA and BEC in two-level structures have shown delay improvements of up to 44.7% [10], making them attractive for FPGA implementations. Additionally, parallel prefix adder variants have been explored in hybrid approximate forms, offering further flexibility in balancing accuracy and hardware cost for signal processing and machine learning workloads [4].

An increasingly adopted approach is approximate computing, which trades computational accuracy for gains in area and power efficiency. For instance, an approximate 128-bit Vedic multiplier demonstrated a 93.95% reduction in power [8], and an approximate adder achieved a 25% reduction in LUT usage compared to exact counterparts [7]. Hybrid designs, which combine accurate and approximate techniques—such as truncating the least significant bits—maintain acceptable output quality while significantly improving area and power metrics [4].

Beyond individual adder topologies, recent research has focused on hybrid and application-specific architectures that strategically combine different techniques. A 32-bit MAC unit integrating a Vedic multiplier with a Brent-Kung adder achieved a significant delay reduction to 16.094 ns [11]. Similarly, architectures combining Booth encoding with Vedic multiplication leveraged concurrent partial product generation to achieve a 35.6% speed improvement over conventional Booth multipliers [11, 12]. More advanced optimizations, such as hybrid adders combining Fast Carry Switching Adder (FCSA) and Kogge-Stone Adder (KSA) with modified Nikhilam Vedic multipliers, have demonstrated a 30% speed improvement and 25% power reduction [13]. The integration of spanning tree adders within Vedic multipliers has also yielded simultaneous improvements in delay and power consumption [14], while modified carry select adders using binary-to-excess-1 converters (BEC) have shown notable area efficiency [15]. Collectively, these studies underscore a clear trend toward hybrid and co-optimized architectures that strategically blend arithmetic blocks to meet the stringent power, speed, and area constraints of modern digital systems[11, 12, 13,14 15,16].

Further architectural innovations address MAC unit performance at multiple levels [18-25]. A 16×16 Vedic multiplier achieves 10.50 ns delay on Virtex FPGA [18], while comparative analysis identifies Vedic-based designs as most efficient in speed and dynamic power [19]. A modified Vedic architecture reduces area by 12.60% and delay by 21.13% for 32-bit MAC units [20]. For larger operands, compressor-based adders manage critical path in 256×256 multipliers [21]. Majority logic gate implementations demonstrate reduced delay using parallel-prefix adders [22]. A parallel MAC design integrating carry-save adder trees with accumulators minimizes critical path and improves gate count [23]. FPGA and ASIC implementations in 180nm technology validate significant improvements in delay and power [24]. Two novel 32-bit Vedic multiplier architectures

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

employing partial product concatenation achieve superior Area-Power-Delay Product across 5nm to 180nm nodes [25].

III. RESEARCH GAP

Present literature indicates that there is a compartmentalized design strategy wherein adder and multiplier optimization have been carried out individually without any holistic co-optimization of all parts of the MAC block. Although various topologies of adders and various architectures of vedic multipliers have shown significant optimization potential individually, the interaction of all parts of MAC block has not been fully investigated.This fragmented focus creates a significant knowledge gap, as true MAC performance depends on synergistic or antagonistic interactions between its constituent blocks.

A further disconnect exists between optimization techniques applied to smaller bit-width designs and their scalability to larger operand sizes. While 16-bit and 32-bit MAC units show remarkable results, scaling to 256-bit and beyond introduces exponential increases in critical path delay, routing complexity, and power dissipation. The inherent parallelism of Vedic multiplication is often negated by inefficient partial product accumulation structures at higher widths, while emerging technologies like QCA-based designs lack validation for seamless integration with conventional CMOS arithmetic blocks.

The present work addresses these gaps through a holistic co-optimization approach that systematically evaluates the interaction between a 32-bit Vedic multiplier and three parallel-prefix adder architectures: Kogge-Stone, Brent-Kung, and an optimized Brent-Kung design. Through analysis of delay, power, and area for all combinations of multipliers and adders, this research identifies optimal combinations for maximum MAC efficiency. This work is an extension of existing techniques in concatenation and adder optimization and shows how architectural design can achieve performance without exponential degradation like larger designs, making it a potential solution for future generations of DSP devices.

IV. METHODOLOGY

The proposed work presents a system-level Multiply–Accumulate (MAC) architecture implemented using a 16-bit Vedic multiplier and a 32-bit accumulation datapath, where the impact of different adder designs is systematically evaluated. In this architecture, the 16-bit Vedic multiplier computes the product of the input operands, generating a 32-bit result that is accumulated using a single 32-bit adder. Multiple MAC variants are independently designed by integrating one adder architecture at a time, including Ripple Carry, Carry-Skip, Carry Look-Ahead, Brent–Kung, and Kogge–Stone adders. All MAC implementations share an identical multiplier, control logic, and operating conditions, ensuring that any variation in power consumption, propagation delay, and area utilization is solely due to the choice of adder architecture. This methodology enables a fair and consistent Power, Performance, and Area (PPA) trade-off analysis across different adder-based MAC designs.

A. Adder Architectures

Five popular adder architectures are developed and assessed in order to determine how adder selection affects MAC performance:

➢ *Ripple Carry Adder (RCA)*

Binary addition is carried out through the Ripple Carry Adder (RCA), which flows incrementally from the least significant bit to the most significant bit using multiple full adders, as illustrated in Fig. 1. While the RCA is highly area-efficient and simple to implement, its critical path delay increases linearly with bit width due to serial carry propagation. Owing to its minimal hardware overhead, the RCA is commonly used as a baseline for evaluating more advanced adder architectures.

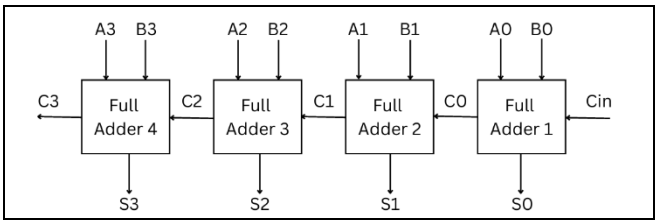

Fig 1 Block Diagram of 4-Bit Ripple-Carry-Adder

➢ *Carry Look Ahead Adder (CLA)*

The Carry Look-Ahead Adder (CLA) eliminates the linear carry propagation delay of the Ripple Carry Adder by computing carry signals in parallel using generate and propagate terms, defined as:

$$g[i] = A[i] \cdot B[i], p[i] = A[i] \oplus B[i] \quad (1)$$

The carry for each bit is directly computed as:

$$c[i+1] = g[i] + p[i] \cdot c[i] \quad (2)$$

Thereby enabling concurrent carry generation rather than sequential propagation, as illustrated in Fig. 2. The final sum bits are obtained by XORing the propagate signals with the computed carries. While the CLA significantly reduces addition delay compared to RCA-based designs, it incurs increased area and wiring complexity due to additional carry generation logic. In this work, a modular 4-bit CLA with group generate and group propagate logic is employed to balance performance and hardware overhead.

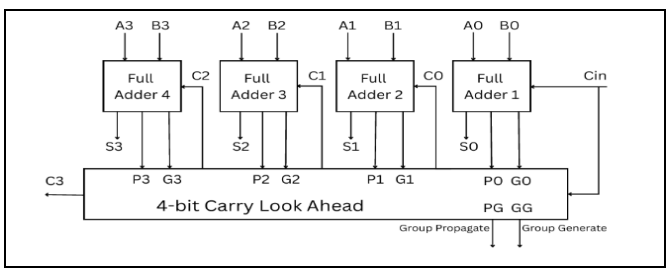

Fig 2 4-Bit Carry Look Ahead (CLA) Adder with Group Propagate (PG) and Group Generate (GG) Logic

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

➢ *Carry Skip Adder (CSKA)*

The Carry-Skip Adder (CSKA) enhances the Ripple Carry Adder by introducing a carry bypass mechanism that reduces worst-case propagation delay. For each bit position, a propagate signal is generated as:

$$P[i] = A[i] \oplus B[i] \quad (3)$$

The entering carry c_in skips the ripple-carry chain and is sent straight to the output carry c_out if all propagate signals within the adder block are set; otherwise, the carry propagates properly through the RCA block, as shown in Fig. 3. This selective skipping significantly accelerates addition during long propagate chains while maintaining low hardware complexity. The sum computation remains identical to that of the RCA, ensuring functional correctness.

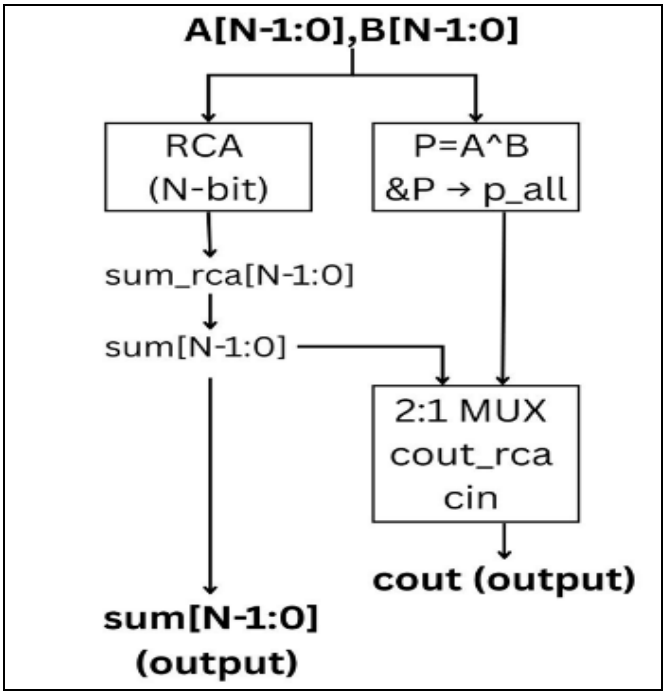

Fig 3 Block Diagram of N-Bit Carry-Skip Adder (CSKA) with Single RCA Block and Skip Logic

➢ *Kogge–Stone Adder (KSA)*

The Kogge–Stone Adder (KSA) is a high-performance parallInput gating el prefix adder that computes carry signals using a logarithmic-depth prefix tree, enabling fast addition. Initially, bitwise generate and propagate signals are computed as:

$$G_0 = A \cdot B, P_0 = A \oplus B \quad (4)$$

These signals are combined across multiple prefix stages, where carry information is propagated over increasing spans, achieving a computation time of $O(\log_2 N)$. The final sum bits are obtained as:

$$\text{Sum}[i] = P_0[i] \oplus C[i] \quad (5)$$

As illustrated in Fig. 4, the KSA offers very low delay and balanced fanout, making it suitable for high-speed arithmetic units, though at the cost of increased area and routing complexity.

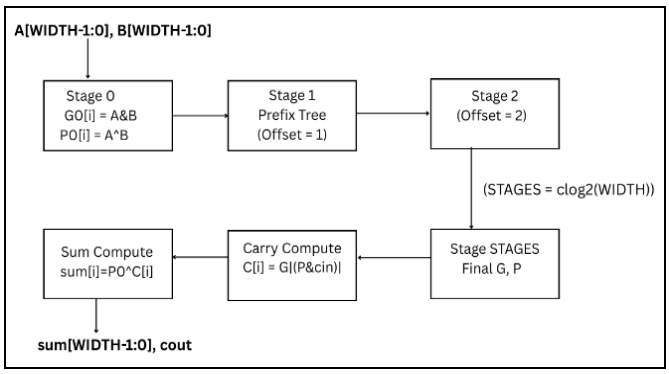

Fig 4 Block Diagram of WIDTH-Bit Kogge-Stone Adder

➢ *Brent–Kung Adder (BKA)*

The Brent-Kung Adder (BKA) is a parallel prefix adder that provides a fair trade-off between speed and hardware complexity by computing carry signals using a hierarchical tree structure. It has slightly higher processing depth at cost of less wiring and expansion than the Kogge-Stone Adder. In this work, the N-bit BKA is implemented using modular 4-bit Brent–Kung blocks, where internal carries within each block are generated in parallel, while inter-block carries propagate sequentially. This hybrid architecture, illustrated in Fig. 5, provides an efficient balance of power, performance, and area for medium- to large-word-length adders.

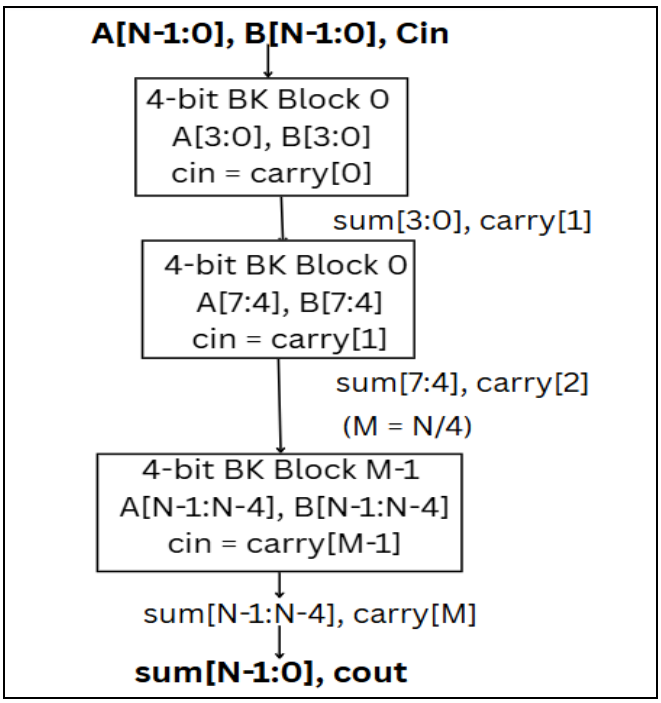

Fig 5 Block Diagram of N-bit Modular Brent-Kung Adder

B. Hierarchical Design of the Vedic Multiplier

The proposed MAC architecture utilizes a hierarchical Vedic multiplier, which is based on the Urdhva-Tiryagbhyam algorithm of Vedic mathematics. Using this algorithm, multiplication is achieved by generating parallel partial products, which reduces propagation delay in comparison to

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

other multiplication techniques. The 16x16 Vedic multiplier is implemented by using a hierarchical design style, starting from the 2x2 multiplier module. Four 2x2 multiplier blocks are combined to obtain a 4x4 multiplier, and again four 4x4 multipliers are combined to obtain an 8x8 multiplier. Finally, four 8x8 multiplier blocks are combined to obtain the 16x16 Vedic multiplier architecture.

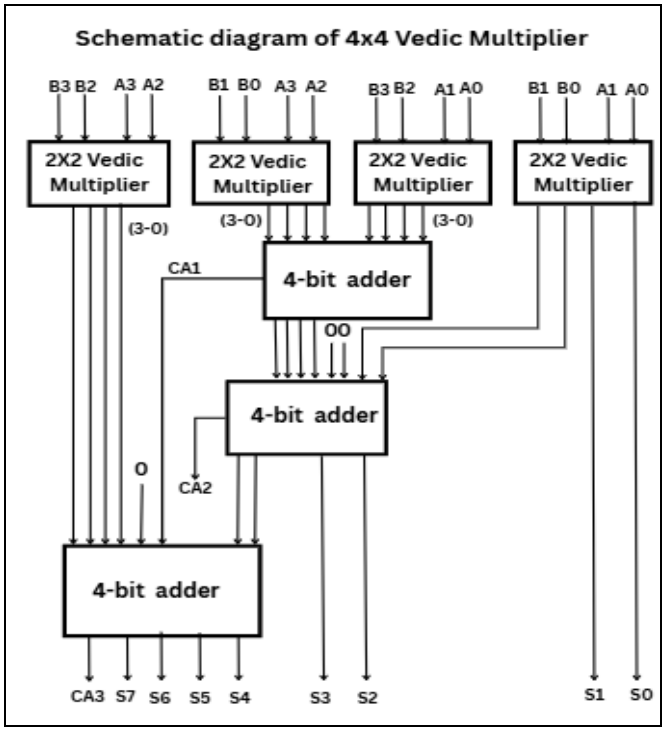

Fig 6 Hierarchical Architecture of the 4×4 Vedic Multiplier Using 2×2 Multiplier Modules.

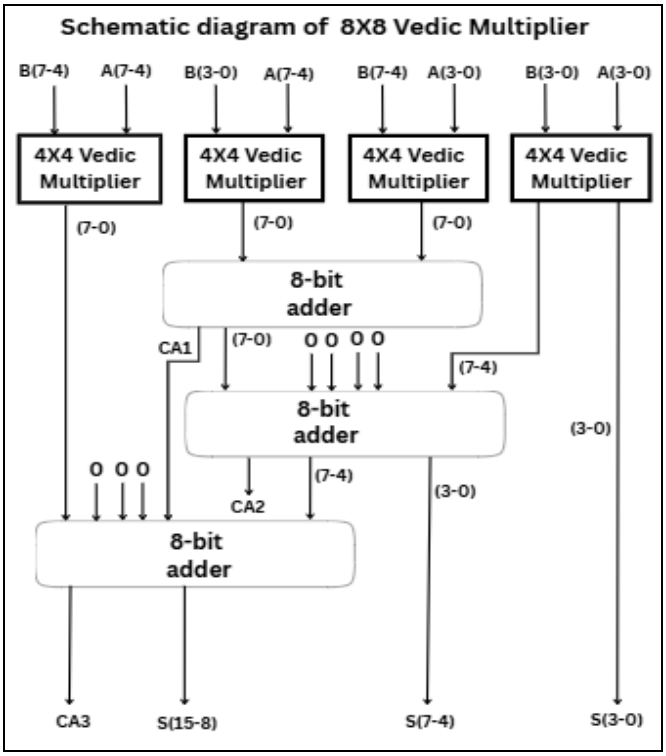

Fig 7 Hierarchical Architecture of the 8×8 Vedic Multiplier Using 4×4 Multiplier Modules.

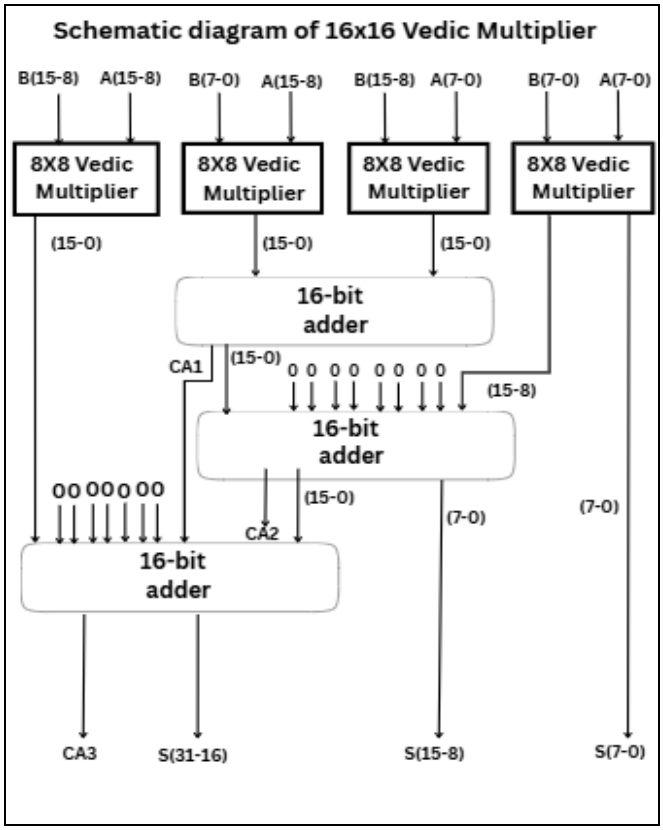

Fig 8 Hierarchical Architecture of the 16×16 Vedic Multiplier Using 8×8 Multiplier Modules.

Figures 6–8 illustrate the hierarchical design of the proposed Vedic multiplier. The architecture is constructed using a modular approach where smaller multipliers are used to build larger ones. A 2×2 Vedic multiplier forms the basic building block, which is used to create a 4×4 multiplier. Similarly, 4×4 modules are combined to form an 8×8 multiplier, and four 8×8 modules are integrated to implement the 16×16 Vedic multiplier. This hierarchical structure enables efficient and scalable multiplier implementation.

In the proposed MAC architecture, the result from the hierarchical Vedic multiplier is accumulated by using different adder architectures. Five different adder architectures are implemented in the accumulation phase of the MAC architecture to observe their effect on performance.

C. Vedic Multiplier–Based MAC Architecture

The Multiply–Accumulate (MAC) unit performs combined multiplication and accumulation operations and is widely used in signal processing and neural network applications. In this work, a 16×16 Vedic multiplier computes the product of two input operands, which is subsequently added to the stored accumulated value using a 32-bit adder. Separate MAC implementations are realized using different adder architectures, including Ripple Carry, Carry Look-Ahead, Brent–Kung, and Kogge–Stone adders, enabling a fair comparison of power, performance, and area trade-offs, as illustrated in Fig. 9. the MAC correctly performed the operation.

$$S_{out} = S_{in} + A \times B \quad (6)$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

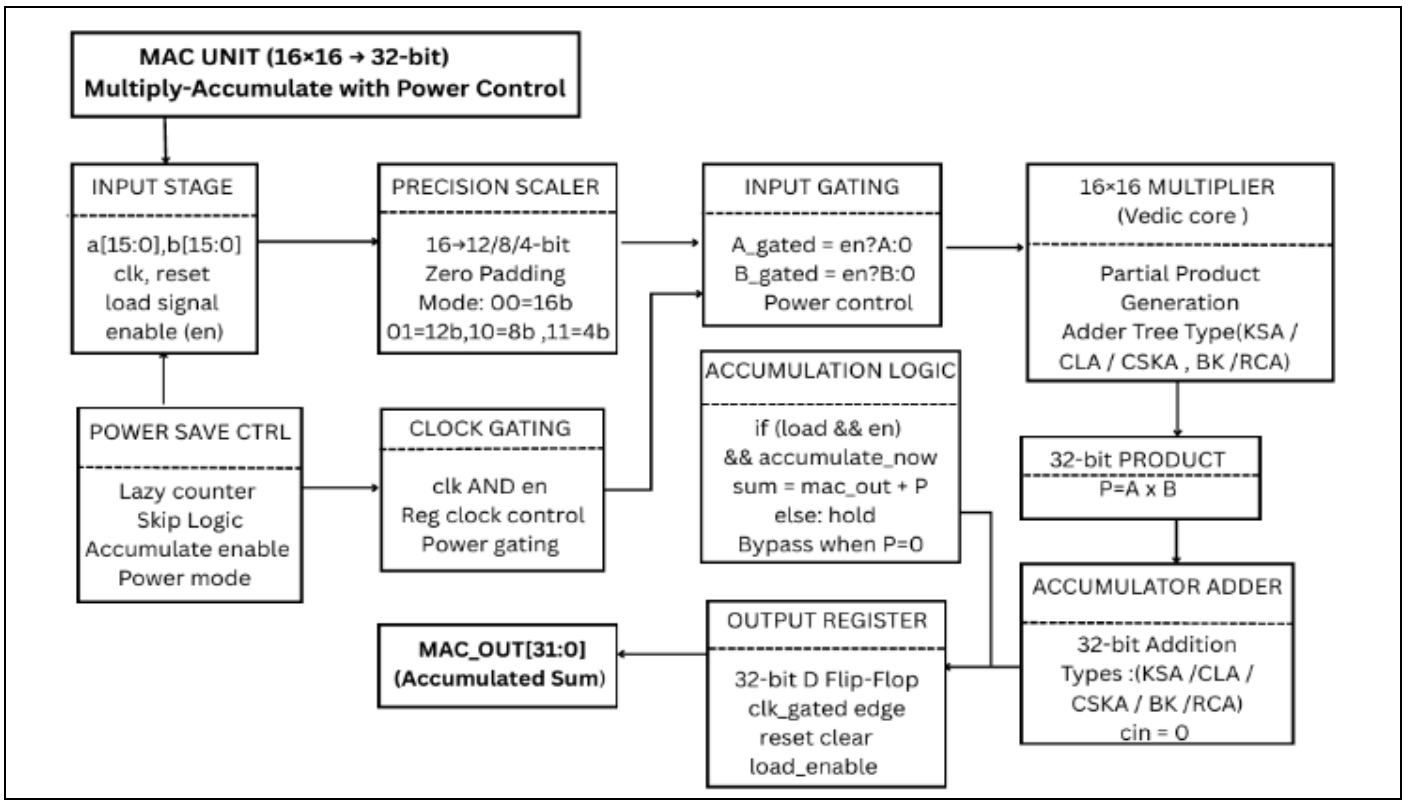

Fig 9 Block Diagram of the 16×16 MAC Unit with Power Control and Precision Scaling

D. Power Optimization and Precision Scaling

To improve energy efficiency, the proposed MAC architecture incorporates several power-aware optimization techniques aimed at reducing unnecessary switching activity and computational overhead. Precision scaling is employed to dynamically adjust the operand bit width from 16-bit down to 4-bit depending on the required computational accuracy. By operating on reduced precision when full accuracy is not required, the number of active logic elements and switching transitions is minimized, thereby lowering dynamic power consumption.

In addition, input gating techniques are applied to disable the multiplier inputs when the MAC unit is inactive or when new data is not available. This prevents unnecessary signal transitions within the multiplier circuitry. The design also incorporates a lazy accumulation strategy, in which accumulation operations are selectively skipped during low-power operating modes when updates to the accumulator are not required. This approach significantly reduces redundant arithmetic operations and switching events.

Furthermore, an accumulator bypass mechanism is introduced to detect cases where the multiplier output is zero. In such situations, the addition stage is bypassed to avoid redundant accumulation operations. Conditional clocking, implemented through clock gating of the accumulator registers, further reduces dynamic power by disabling clock activity when register updates are unnecessary. Collectively, these techniques enhance the overall energy efficiency of the MAC unit while maintaining functional correctness and computational reliability.

E. Evaluation Methodology

To achieve this objective of a fair and consistent comparison, all the adder-based MAC architectures are implemented and synthesized under the same design and tool conditions. This includes the use of the same operand width, synthesis constraints, clock frequency, and hardware platform. This way, the differences in the performance of the architectures are entirely based on the performance characteristics of the adder-based architectures.

The performance characteristics of the adder-based MAC architectures are evaluated based on three main performance parameters: power consumption, critical path delay, and hardware area utilization. The power consumption parameter is used to measure the power consumption characteristics of the adder-based MAC architectures. Critical path delay is the longest combinational path in the adder-based MAC architectures and determines the maximum frequency at which the adder-based MAC architectures can be implemented. Hardware area utilization is the hardware resources required in the implementation of the adder-based MAC architectures.

The performance characteristics are extracted from the implementation and synthesis results of the adder-based MAC architectures and then analyzed based on the Power–Performance–Area (PPA) evaluation methodology.

V. RESULT AND ANALYSIS

In this section, the implementation along with the performance analysis of Multiply Accumulate (MAC) architectures using Vedic Multiplier and five different

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

adders: Kogge Stone Adder (KSA), Carry Lookahead Adder (CLA), Brent Kung Adder (BK), Carry Skip Adder (CSKA), and Ripple Carry Adder (RCA). The MAC architectures using these multipliers and adders were implemented for both 16-bit and 32-bit systems on Zynq-7000 FPGA platform using ZedBoard. The evaluation focuses on hardware resource utilization, functional verification, and system-level performance metrics including power consumption, propagation delay, area utilization, Power–Delay Product (PDP), and Energy–Delay Product (EDP). The results obtained from RTL analysis, simulation verification, tabulated comparisons, and graphical representations provide a comprehensive understanding of the trade-offs between performance, power efficiency, and hardware complexity among the different adder-based MAC architectures.

A. RTL Analysis of the Five Adders (16-Bit)

The RTL schematics show the internal hardware structure of the proposed MAC architecture using different adder architectures. The RTL views obtained from the synthesis tool represent the datapath of the MAC unit using five different adder architectures. The structure of the MAC unit is the same in all the RTL views, with the Vedic Multiplier block being the same and the adder being different in the accumulation path. The RTL views show the datapath structure of the MAC unit using five different adder architectures. Since the structure of the MAC unit is the same in both the 16-bit and 32-bit MAC architectures, the RTL schematics of the proposed MAC architecture using the 32-bit MAC unit is presented here. The MAC architecture using the 16-bit MAC unit has the same structure with reduced datapath width. The RTL schematics of the individual adder architectures are presented in the following subtopics.

➢ *RTL Analysis of Kogge–Stone Adder (KSA) (16-Bit)*

RTL Schematic of the MAC Architecture with Kogge-Stone Adder RTL schematic of the Kogge-Stone Adder describes the parallel prefix adder structure used for efficient carry computation. In the adder design, the generate and propagate signals form a tree structure used for efficient carry look-ahead addition. This structure results in efficient reduction of the carry look-ahead time, thus the KSA is one of the fastest adders. However, the parallel prefix structure increases the hardware utilization since it requires a larger number of logic resources. The RTL structure of the Kogge-Stone Adder is presented in Fig. 10 below.

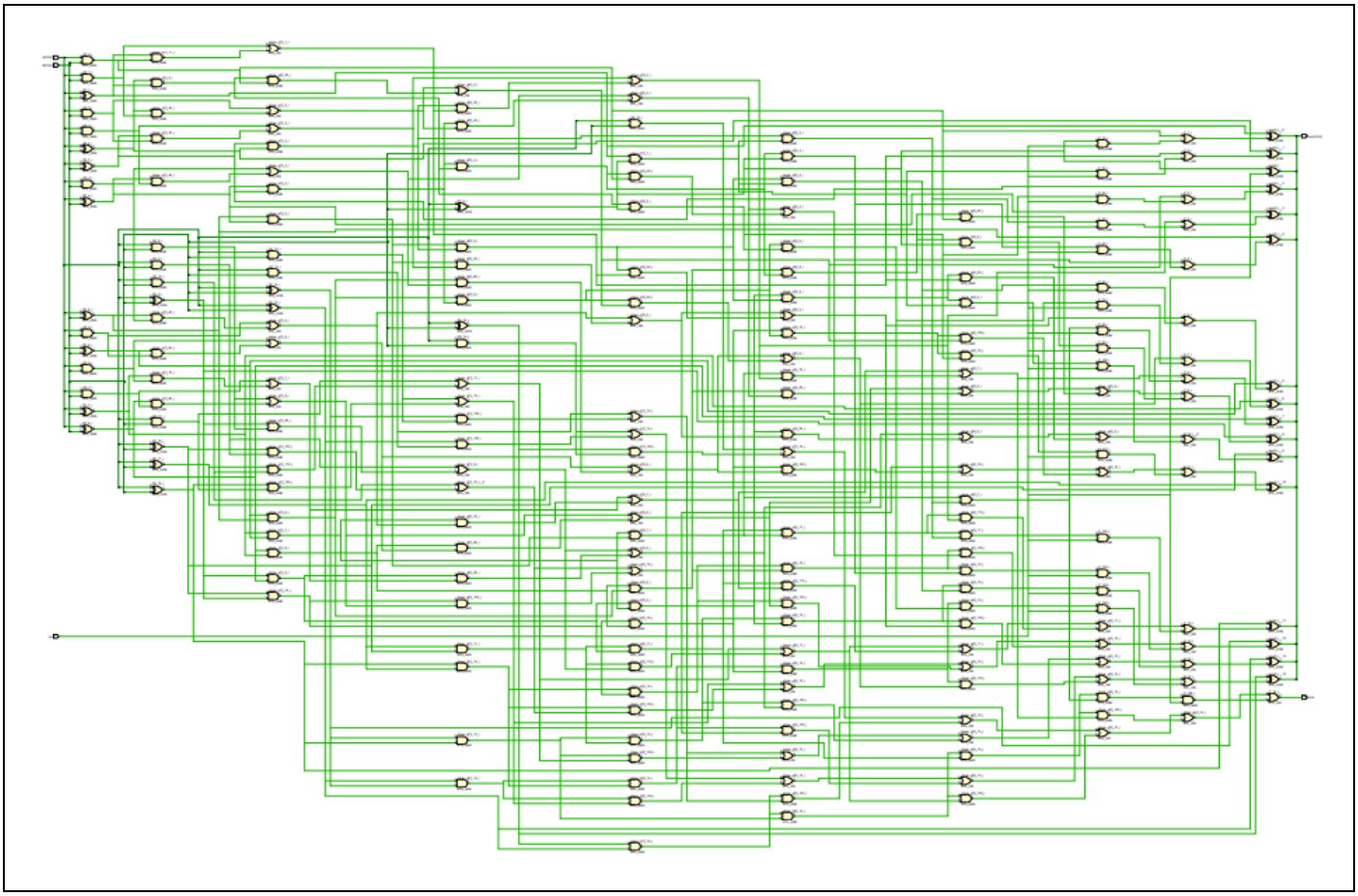

Fig 10 RTL Schematic Kogge–Stone Adder (KSA).

➢ *RTL Analysis of Carry Lookahead Adder (CLA)*

The RTL design of the RTL schematic of the Carry Lookahead Adder illustrates the utilization of the "generate" and "propagate" logic to determine the carry value. Unlike the ripple adders, the "CLA" adder employs parallel determination of the carry value utilizing the "lookahead logic blocks." This enhances the adder's speed compared to the normal adders. However, the hardware complexity is moderate. The RTL diagram for the CLA-based MAC architecture is shown in Figure 11.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

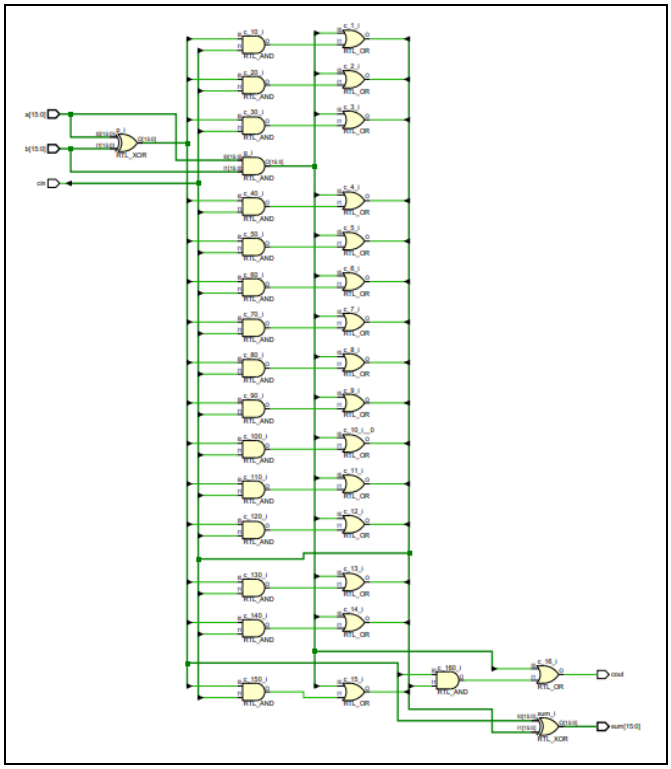

Fig 11 RTL Schematic of Carry Lookahead Adder (CLA).

➢ *RTL Analysis of Brent–Kung Adder (BK)*

The RTL diagram for Brent Kung Adder shows a well-balanced parallel prefix adder that decreases the number of logic nodes compared to KSA. The carry signals for this adder are generated by a hierarchical tree structure with a logarithmic depth, which enables fast propagation of carry signals while minimizing hardware resources compared to KSA.

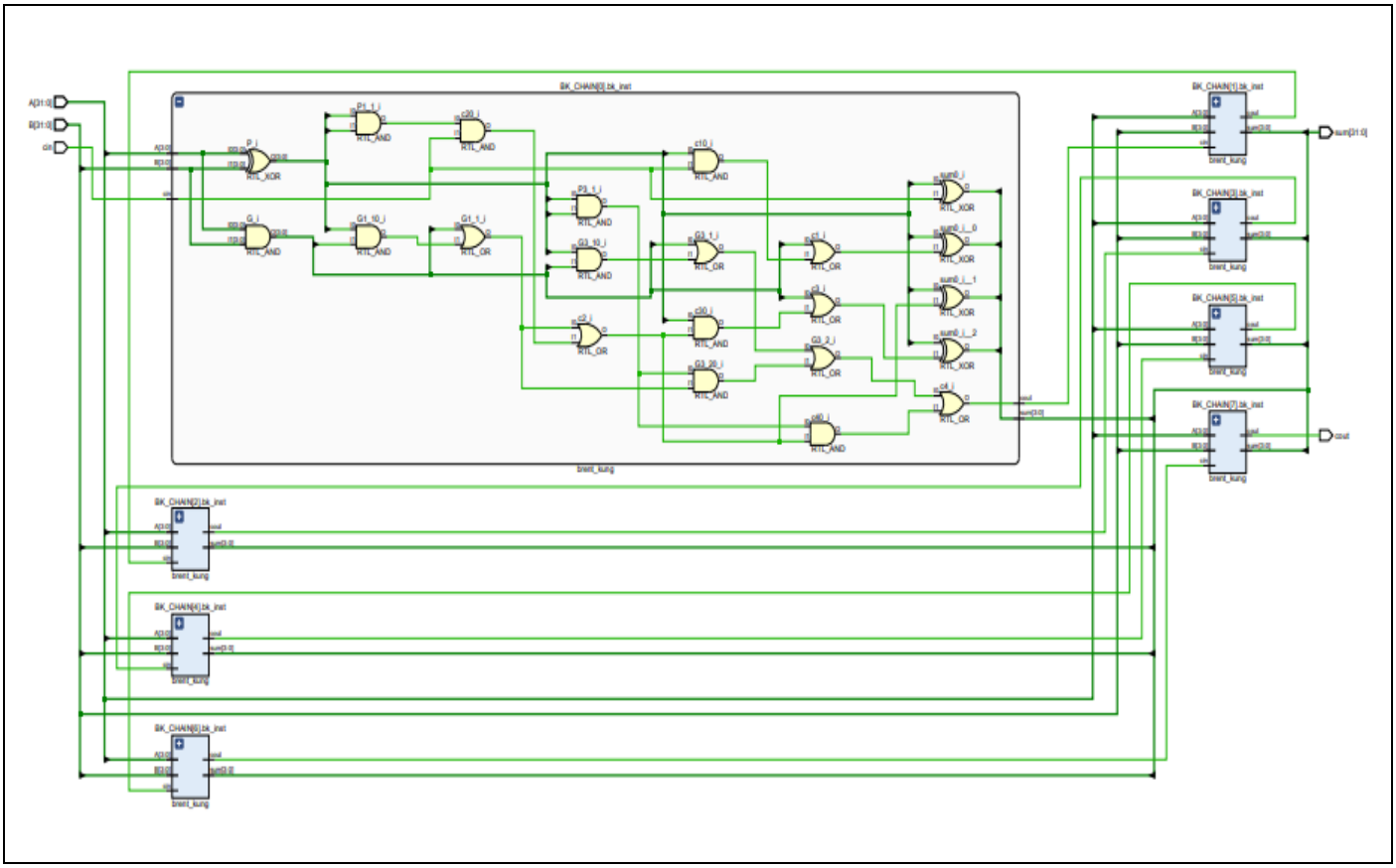

Fig 12 RTL Schematic of Brent–Kung Adder (BK).

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

➢ *RTL Analysis of Carry Skip Adder (CSKA) Based MAC (32-Bit)*

The RTL schematic of the Carry Skip Adder shows the application of skip logic blocks that enable the bypassing of the carry signal through some bits. In the skip blocks, the carry signal is able to skip the block when the propagate condition is met, which improves the performance compared to the Ripple Carry Adder due to the reduced propagation delay. This design has relatively low hardware complexity. The RTL schematic of the CSKA architecture is shown in Fig. 13.

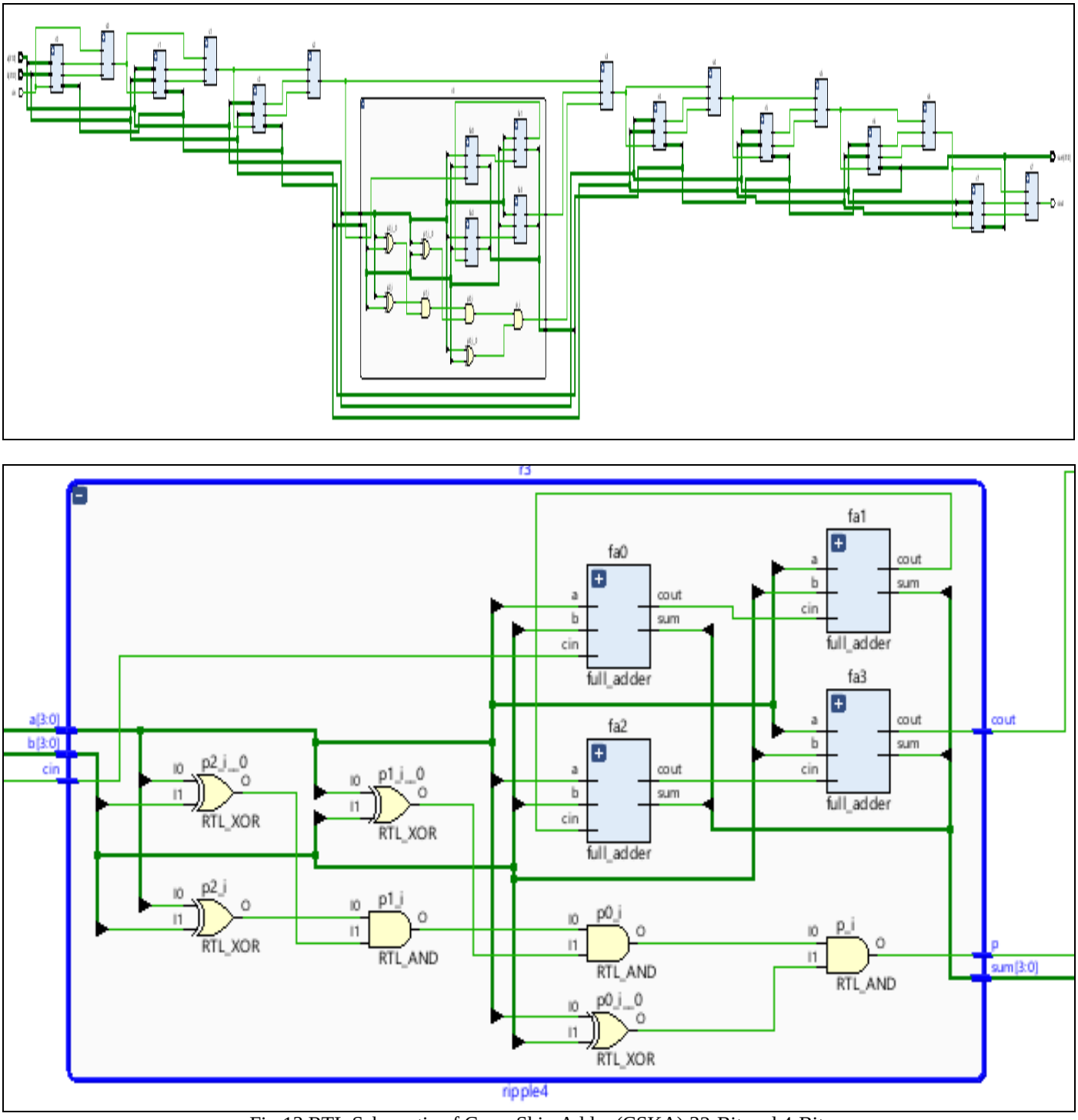

Fig 13 RTL Schematic of Carry Skip Adder (CSKA) 32-Bit and 4-Bit.

➢ *RTL Analysis of Ripple Carry Adder (RCA)*

The RTL schematic diagram of Ripple Carry Adder signifies the simplest form of adder architecture amongst the five implementations. In this adder, the carry signal is propagated sequentially from the least significant bit position to the most significant bit position. Though this adder architecture consumes fewer hardware resources, the propagation delay is the highest in this adder due to the presence of the carry propagation path. The RTL diagram of the RCA architecture is shown in Figure 14.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

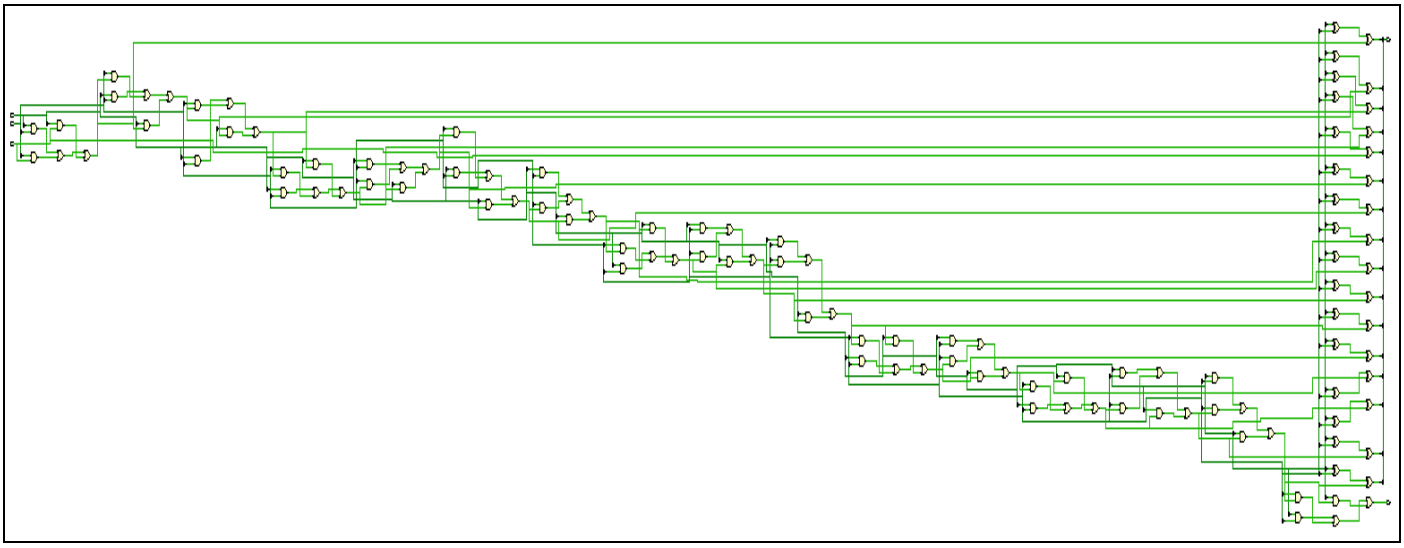

Fig 14 RTL Schematic of Ripple Carry Adder (RCA).

➢ *RTL Analysis of the Proposed MAC Architecture (32-Bit)*

The RTL level of analysis shows the internal hardware structure of the proposed 32-bit MAC architecture based on the Zynq-7000 FPGA platform. The MAC architecture includes a Vedic multiplier followed by an accumulation block, which is implemented based on different adder architectures. Since the datapath of the MAC architecture remains more or less the same, the RTL level of the complete MAC architecture is shown only once.

However, the structural variations of the different implementations are based on the adder used in the accumulation block of the MAC architecture. Therefore, the RTL level of the individual adders used in the accumulation block of the MAC architecture is shown separately to highlight the carry propagation mechanism of the individual adders, which are based on different architectures. The five different adders used in this work are Kogge-Stone Adder (KSA), Carry Look Ahead Adder (CLA), Brent-Kung Adder (BK), Carry Skip Adder (CSKA), and Ripple Carry Adder (RCA). These adder architectures are critical in defining the performance parameters of the MAC architecture, such as the delay, power, and utilization of the MAC architecture.

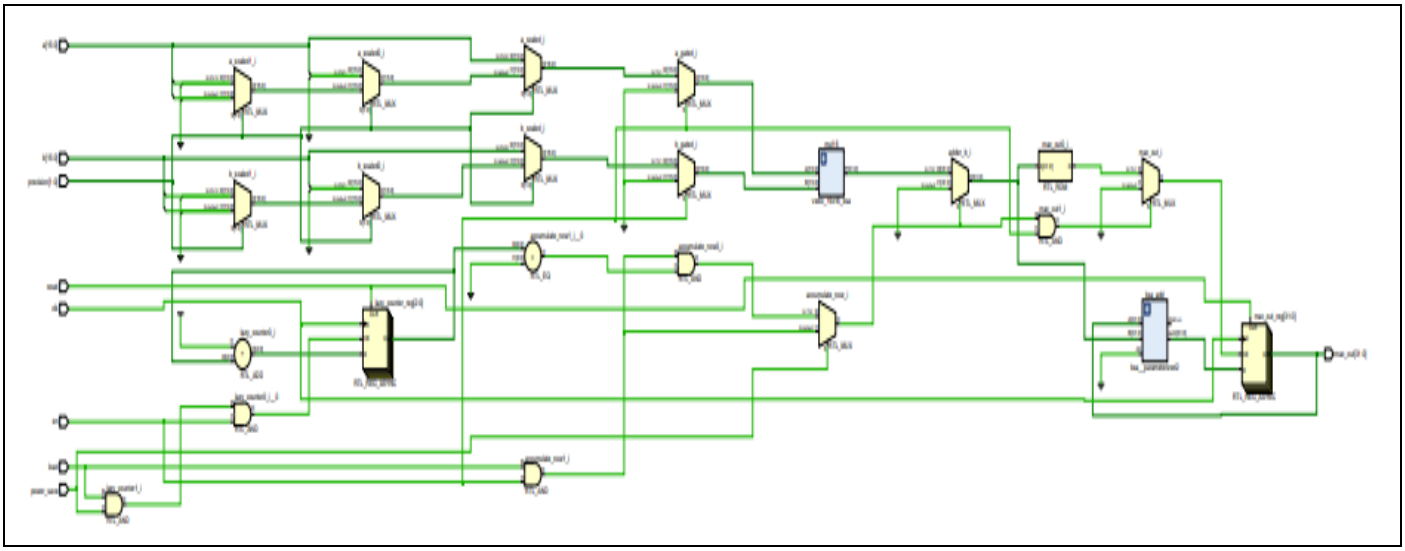

Fig 15 Elaborated Adder Implementation in Vedic MAC Architecture

B. Simulation Analysis of MAC Architectures

Functional simulation was carried out to verify the correctness of the proposed MAC architectures implemented with different adder designs. The simulations were performed using the HDL simulation environment to validate the functional behavior of the Vedic multiplier-based MAC unit. For each architecture, the multiplier output is accumulated through the adder stage under clock control, ensuring correct multiply–accumulate operation.

The simulation results confirm that all five adder-based MAC implementations produce identical functional outputs for the same input combinations. The differences among the architectures are primarily related to structural complexity and propagation delay rather than functional behavior.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

Simulation waveforms for both 16-bit and 32-bit MAC architectures are presented in the following subsections.

The functional behaviour of both 16-bit and 32-bit MAC architectures was verified through simulation. Since the operational behaviour is identical for both configurations and differs only in datapath width, the simulation waveforms of the 32-bit MAC architecture are presented for illustration.

➢ Simulation of KSA-Based MAC *(32-bit)*

The KSA-based MAC implementation was functionally verified through simulation across all supported precision modes. In 4-bit precision mode, inputs A = 0x000F and B = 0x0002 correctly produce mult_out = 0x0000001E and mac_out = 0x0000001E, confirming accurate multiply–accumulate operation, as shown in Fig. 16. The simulation results validate proper precision scaling, timing synchronization, input and clock gating, and correct accumulation behaviour, including lazy accumulation, for the operation.

$$S_{out} = S_{in} + A \times B \quad (7)$$

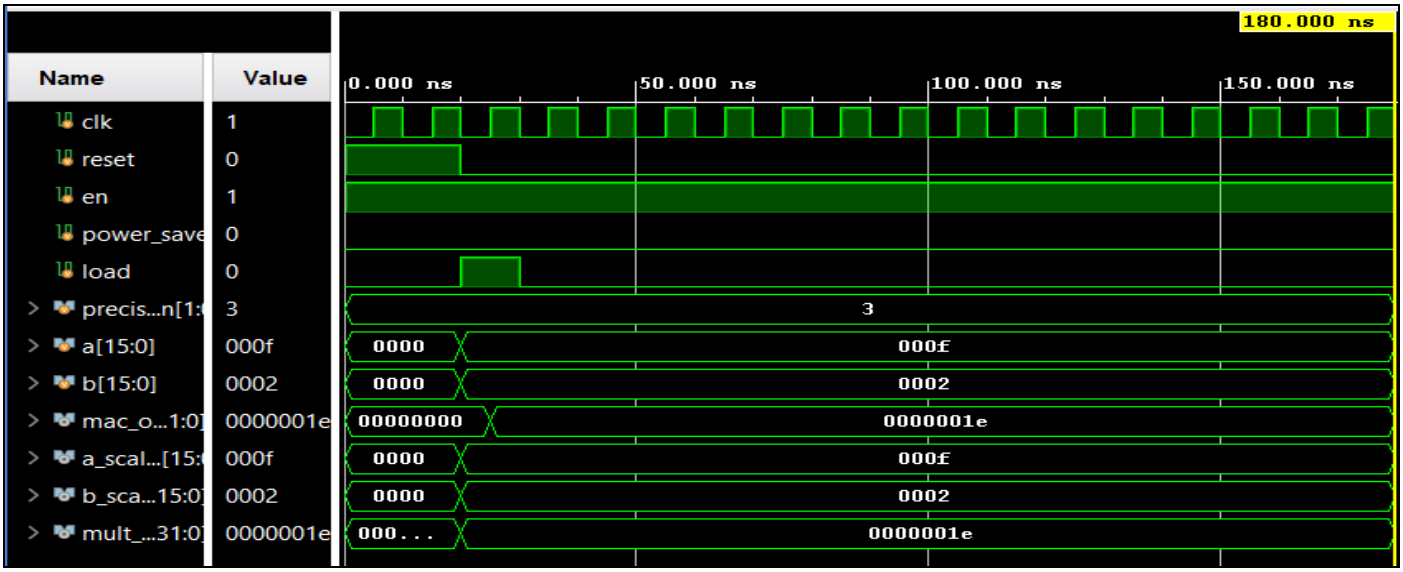

Fig 16 MAC Implementation and Verification with Kogge-Stone Adder.

➢ *Simulation of CLA-Based MAC (32-Bit)*

Through simulation, the functionality of the CLA-based MAC implementation was confirmed in all available precision settings (16, 12, 8, and 4-bit). As illustrated by Fig. 17, the MAC successfully generates mult_out = 0x00062747 and mac_out = 0x000ECA42 for inputs A = 0x029F and B = 0x0259 in 16-bit mode, showing accurate multiply–accumulate operation. The simulation results also verify proper enable/disable control, precision scaling, timing synchronization, and reset functionality for the operation.

$$S_{out} = S_{in} + A \times B. \quad (8)$$

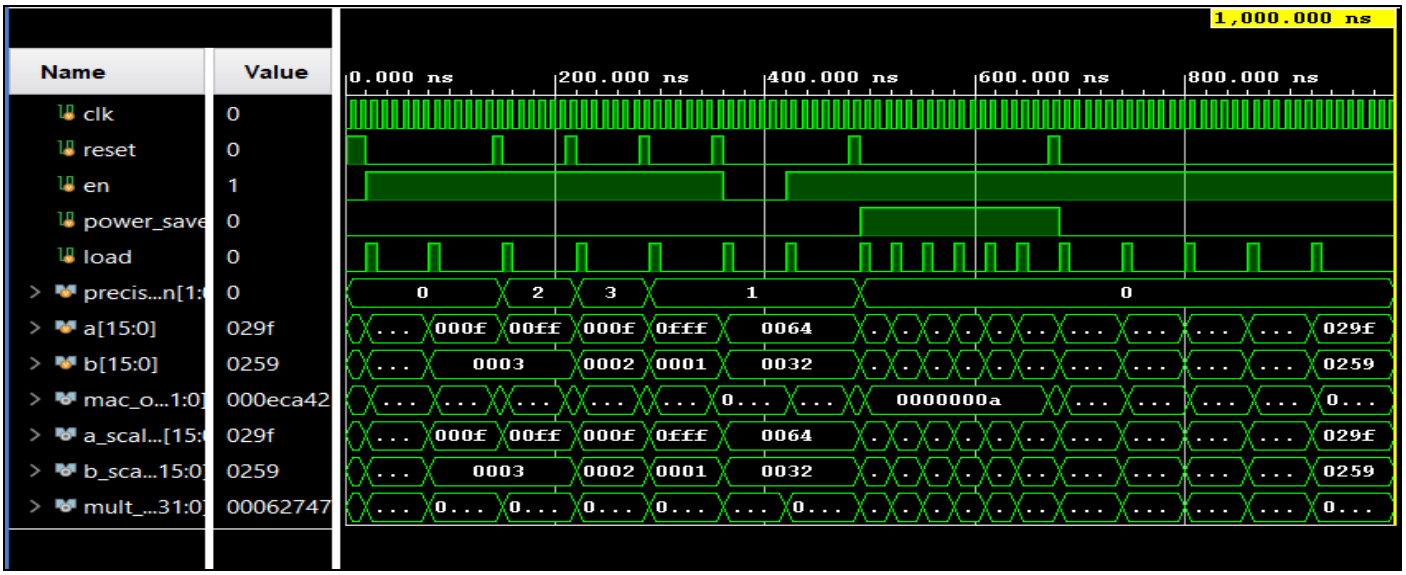

Fig 17 MAC Implementation and Verification with Carry Look-Ahead Adder.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

➢ *Simulation of BKA-Based MAC (32-Bit)*

The Brent–Kung Adder (BKA)–based MAC implementation was functionally verified through simulation. In 16-bit mode, inputs A = 7 and B = 3 correctly produce mult_out = 0x00000015, and subsequent operation in 8-bit precision mode yields the expected accumulated output, confirming accurate multiply–accumulate functionality, as shown in Fig. 18. The simulation results validate correct timing behavior, proper precision scaling, reliable accumulation, and correct control logic operation.

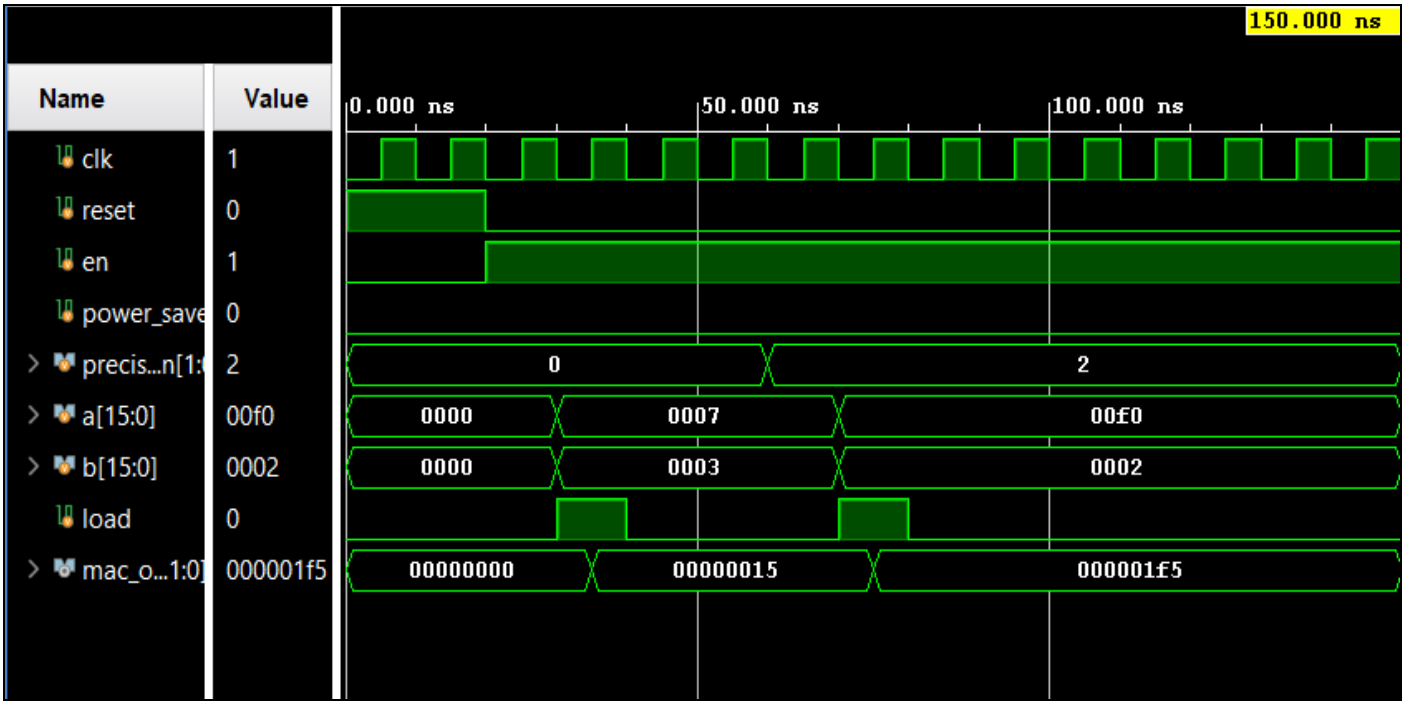

Fig 18 MAC Implementation and Verification with Brent-Kung Adder.

➢ *Simulation of CSKA-Based MAC (32-Bit)*

The CSKA-based MAC implementation was functionally verified through simulation across multiple precision modes. When the enable signal is denied (en = 0), the MAC correctly inhibits all computations and maintains a zero output despite applied inputs, while in active mode it produces accurate multiply–accumulate results for all test cases, as shown in Fig. 19. These results confirm correct control logic operation, proper precision scaling, and seamless integration of the 16×16 Vedic multiplier with the Carry-Skip Adder.

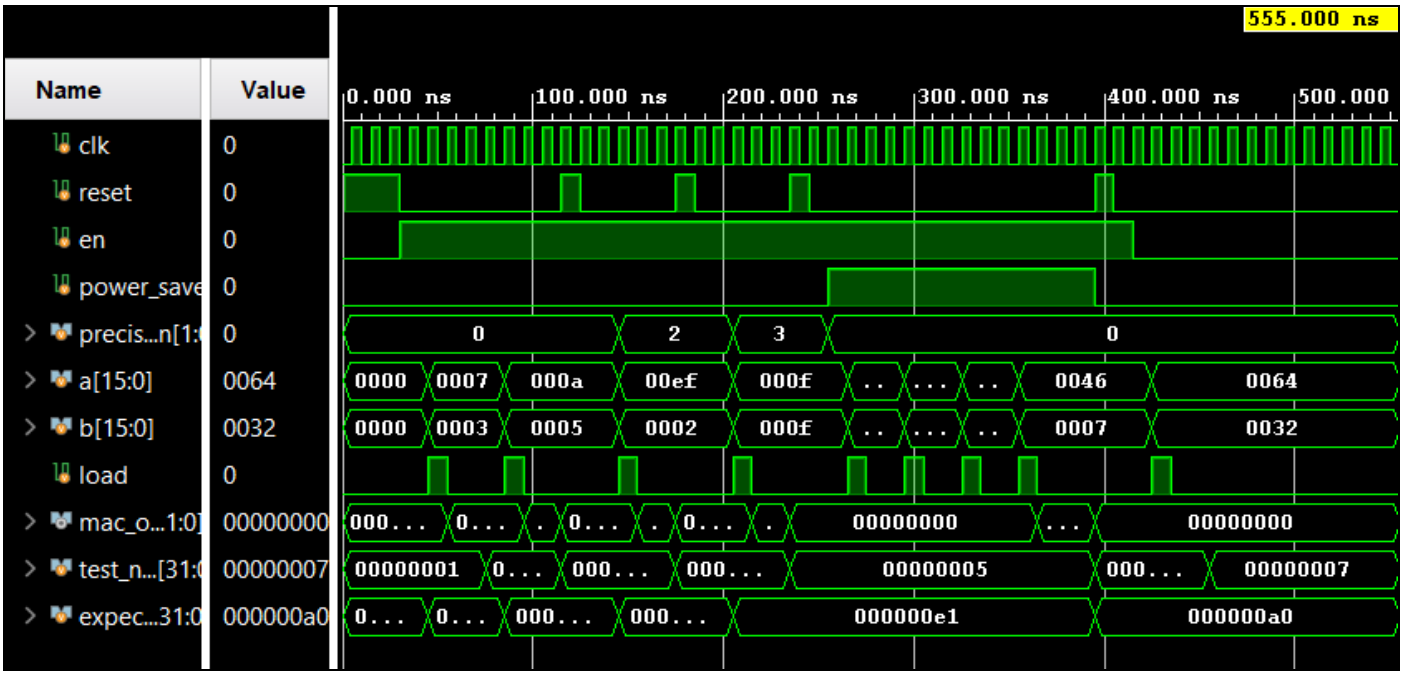

Fig 19 MAC Implementation and Verification with Carry-Skip Adder.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

➢ *Simulation of RCA-Based MAC (32-bit)*

The simulation results confirm correct operation of the RCA-based MAC implementation. Starting from an initial accumulator value of 0x00000000, the MAC produces 0x00000018 after the first multiplication and 0x00000216 after the subsequent accumulation. The expected product of inputs A = 0x00FF and B = 0x0002 in 8-bit precision mode is 0x01FE, which, when combined with the previous accumulator value (0x0018), provides 0x0216, which is consistent with the simulation results given in Fig. 20. These results validate correct integration of the multiplier, adder, and accumulation logic.

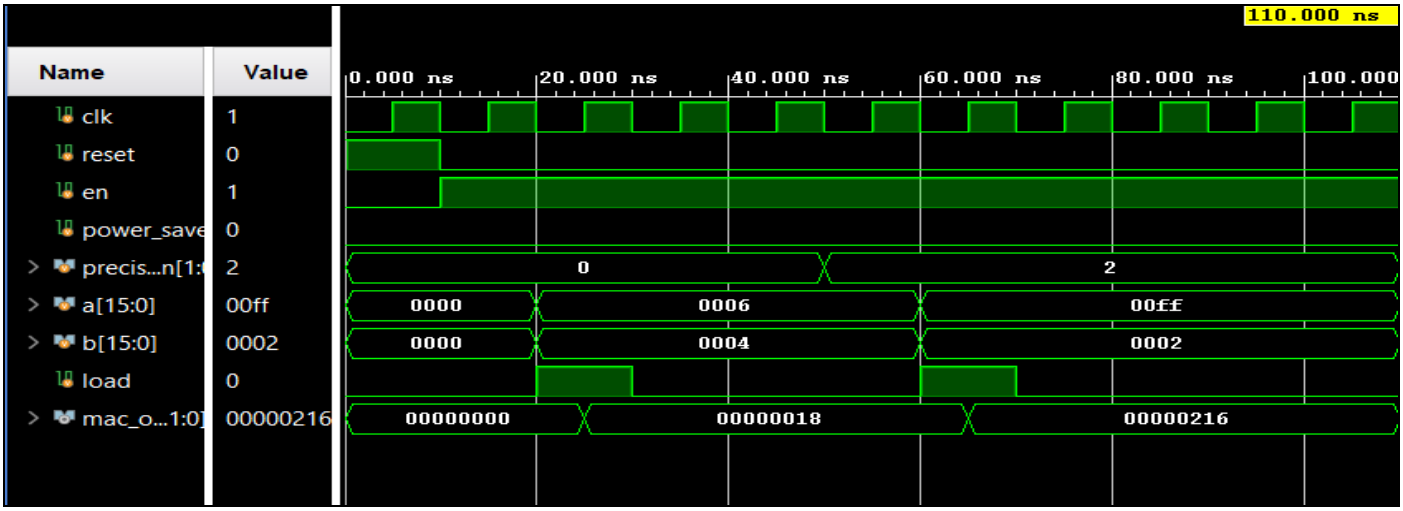

Fig 20 MAC Implementation and Verification with Ripple Carry Adder.

C. Table Analysis

This section discusses the implementation results achieved with the proposed MAC architectures using five different adder architectures in terms of hardware resource utilization such as Look-Up Tables (LUTs), Flip-Flops (FFs), and Input/Output (IO) ports, as well as performance parameters such as power consumption, propagation delay, Power-Delay Product (PDP), and Energy-Delay Product (EDP). The table summarizes the implementation results achieved with the proposed MAC architectures using five different adder architectures after the synthesis and implementation process on the Zynq-7000 FPGA platform.

➢ *Area Analysis of 16-bit MAC Architectures.*

Table 1 shows the hardware utilization of the 16-bit MAC architectures. It is observed that the Ripple Carry Adder requires the least amount of LUTs, 380, because of the simple structure in the adder. The Carry Skip Adder and Brent-Kung Adder require moderate LUTs because of the block structure in the adders. The Carry Lookahead Adder requires more LUTs, 540, because of the additional logic in the adder for generate and propagate signals. The Kogge-Stone Adder requires the most amount of LUTs, 620, because of the complex parallel prefix structure in the adder for faster carry computation.

Table 1 Hardware Resource Utilization of 16-bit MAC Architectures Using Different Adders

ADDER	LUT (16-bit)	FF (16-bit)	IO (16-bit)
KSA	620	35	71
CLA	540	35	71
BK	500	34	71
CSKA	450	22	71
RCA	380	9	71

➢ *Power and Delay Analysis of 16-Bit MAC Architectures*

Table 2 shows the results of the power consumption and delay of the 16-bit MAC architectures designed using different adder architectures. The Kogge–Stone Adder has the least delay of 3.2 ns due to its parallel prefix carry propagation architecture. The Brent–Kung and Carry Lookahead Adders have less delay due to the optimized logic for carry generation. The Carry Skip Adder has moderate delay due to its block-based carry skipping architecture. In contrast, the RCA has the maximum delay of 12.5 ns since the carry is propagated sequentially through each bit stage of the RCA architecture.

Table 2 Power Consumption and Propagation Delay of 16-Bit MAC Architectures Using Different Adders

ADDER	Power (mW) (16-bit)	Delay (ns) (16-bit)
KSA	95	3.2
CLA	72	6.1
BK	65	4.8
CSKA	58	8.4
RCA	40	12.5

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

➢ *PDP and EDP Analysis of 16-Bit MAC Architectures*

Table 3 presents the PDP and EDP comparison using 16-bit MAC architectures, as per the PDP and EDP diagram. The Kogge–Stone Adder has the lowest PDP, i.e., 304 pJ, and EDP, i.e., 972.8, due to the lowest delay in the adder circuit. The Brent–Kung Adder also has competitive PDP and EDP values because of the balanced structure in the adder circuit. The Carry Lookahead and Carry Skip adders have high PDP and EDP values because of the high delay and high-power consumption in the adder circuit. The Ripple Carry Adder has the highest PDP and EDP values because of the high propagation delay in the adder circuit.

Table 3 Power–Delay Product (PDP) and Energy–Delay Product (EDP) of 16-Bit MAC Architectures

ADDER	PDP (Power–Delay Product) pJ (16-bit)	EDP (Energy–Delay Product) pJ (16-bit)
KSA	304	972.8
CLA	439.2	2679.1
BK	312	1497.6
CSKA	487.2	4092.5
RCA	500	6250

➢ *Area Analysis of 32-Bit MAC Architectures*

Table 4 describes the hardware resource usage for the 32-bit MAC architectures. It is evident that as the width of the datapath increases, the number of LUTs also increases for all the architectures. The highest number of LUTs (1135) is required for the implementation of the Carry Lookahead Adder because of the complex carry generation circuitry. The Kogge-Stone Adder also requires a large number of hardware resources (1020 LUTs) because of its parallel prefix adder architecture. The Brent-Kung and Carry Skip adders require moderate hardware resources. The Ripple Carry Adder once again requires the least hardware resources (620 LUTs) because of its simple sequential carry generation circuitry.

Table 4 Hardware Resource Utilization of 32-Bit MAC Architectures Using Different Adders

ADDER	LUT (32-bit)	FF (32-bit)	IO (32-bit)
KSA	1020	75	71
CLA	1135	72	71
BK	890	66	71
CSKA	770	64	71
RCA	620	67	71

➢ *Power and Delay Analysis of 32-bit MAC Architectures*

Table 5 presents a summary of the results for the 32-bit MAC architectures in terms of power and delay. It is observed that the Kogge–Stone Adder has the least delay (5.2 ns) since it has a parallel carry computation technique. The Brent–Kung Adder and the Carry Lookahead Adder have slightly more delay since they have a hierarchical carry design. The Carry Skip Adder has moderate delay reduction using its block skipping technique for carry computation. The RCA has the maximum delay (21 ns) since the carry is propagated sequentially through all 32 bits of the operands.

Table 5 Power Consumption and Propagation Delay of 32-Bit MAC Architectures Using Different Adders

ADDER	Power (mW) (32-bit)	Delay (ns) (32-bit)
KSA	140	5.2
CLA	125	7.8
BK	115	6.9
CSKA	105	9.8
RCA	83	21

➢ *PDP and EDP Analysis of 32-Bit MAC Architectures*

Table 6 shows the results of the PDP and EDP for the 32-bit MAC architectures. The Kogge–Stone Adder has the lowest PDP and EDP of 728 pJ and 3785.6, respectively, because of its minimum delay and high-power consumption. The Brent–Kung Adder has shown comparable results in terms of energy efficiency with its balanced prefix structure. The Carry Lookahead and Carry Skip Adders have shown higher PDP and EDP because of their increased delay and power consumption. The Ripple Carry Adder has shown the highest PDP and EDP because of its increased propagation delay due to sequential carry propagation.

Table 6 Power–Delay Product (PDP) and Energy–Delay Product (EDP) of 32-bit MAC Architectures

ADDER	PDP (Power–Delay Product) pJ (32-bit)	EDP (Energy–Delay Product) pJ (32-bit)
KSA	728	3785.6
CLA	975	7605
BK	793.5	5475.15
CSKA	1029	10084.2
RCA	1743	36603

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

D. Graph Analysis

To facilitate a better understanding of the difference in the proposed MAC architectures, graphical representations of the implementation results have been provided. These graphs have been used to compare the performance of five different adder-based MAC architectures, namely the Kogge-Stone Adder (KSA), the Carry Lookahead Adder (CLA), the Brent-Kung Adder (BK), the Carry Skip Adder (CSKA), and the Ripple Carry Adder (RCA), implemented for different datapath widths of 16 and 32 bits using the Zynq-7000 FPGA platform. Six different performance parameters have been taken into consideration for the analysis of the implemented MAC architectures. These parameters include power consumption, propagation delay, LUT utilization, flip-flop utilization, Power-Delay Product (PDP), and Energy-Delay Product (EDP).

- *Power Comparison of 16-Bit and 32-Bit MAC Architectures*

The power comparison graph (fig. 21) has shown the power consumption differences among the five adder-based MAC architectures. It has been found that the power consumption increases when the datapath width increases from 16 bits to 32 bits. The Kogge-Stone Adder has the highest power consumption (95 mW at 16 bits and 140 mW at 32 bits) because of the complex parallel prefix network. The Ripple Carry Adder has the least power consumption (40 mW at 16 bits and 83 mW at 32 bits) because of the simple sequential carry propagation network. The power consumption of the CLA, BK, and CSKA architectures falls in the middle range.

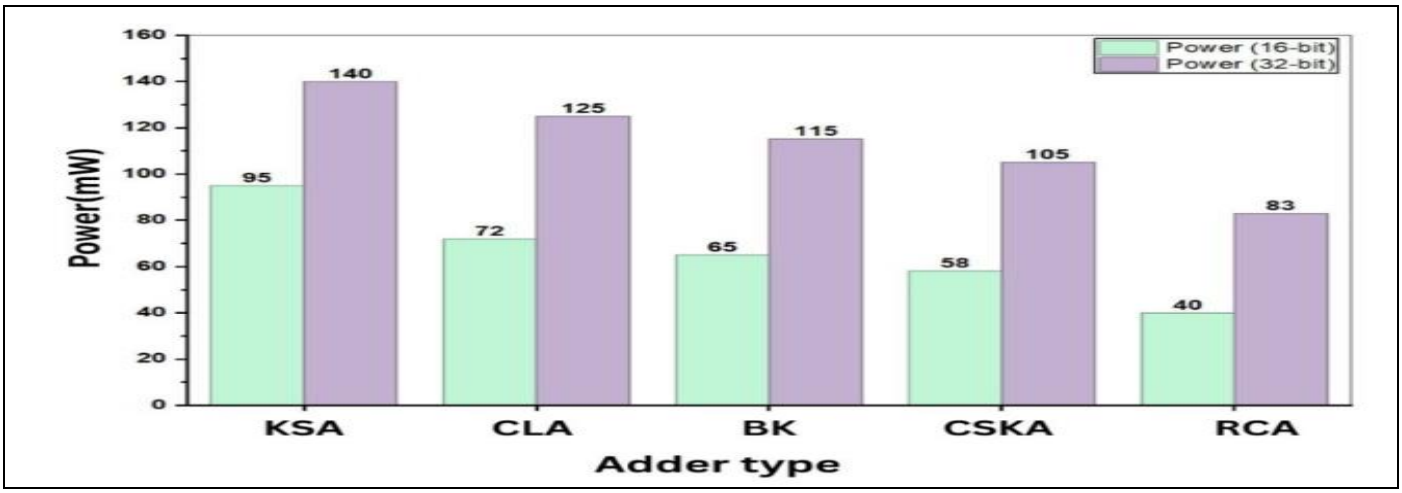

Fig 21 Power Comparison of 16-Bit and 32-Bit MAC Architectures Using Different Adders.

- *Delay Comparison of 16-Bit and 32-Bit MAC Architectures*

The delay comparison graph (fig. 22) is used to represent the propagation delay for each adder-based MAC architecture. The Kogge-Stone Adder has the least propagation delay compared to the others, i.e., 3.2 ns and 5.2 ns for datapath widths of 16 and 32 bits, respectively, due to its parallel carry computation mechanism. The Brent-Kung and Carry Lookahead adders have relatively low propagation delay values due to the optimization of the carry generation mechanism in these architectures. The Carry Skip Adder has moderate propagation delay improvement over the Ripple Carry Adder. The highest propagation delay is observed in the Ripple Carry Adder, i.e., 12.5 ns and 21 ns for datapath widths of 16 and 32 bits, respectively, due to the sequential carry generation mechanism in the adder. The propagation delay is observed to be increased in all the architectures when the datapath width is increased.

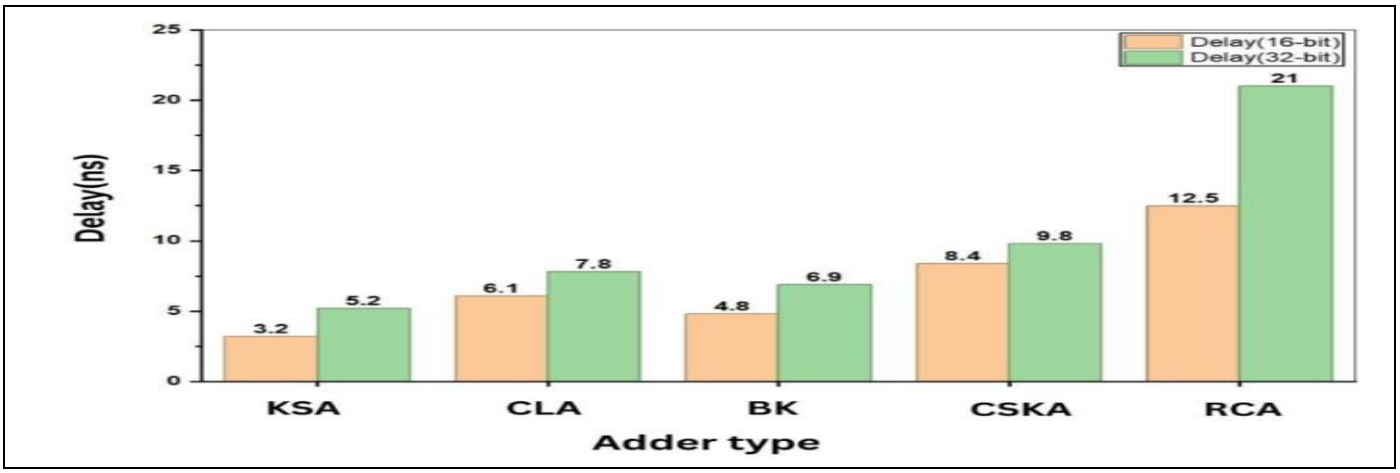

Fig 22 Propagation Delay Comparison of 16-Bit and 32-Bit MAC Architectures.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

➢ *LUT Utilization Comparison of 16-Bit and 32-Bit MAC Architectures*

The LUT utilization graph (fig. 23) represents the hardware resources needed for the implementation of the MAC architectures. The results show that when moving from 16-bit to 32-bit design, there is a significant increase in the number of LUTs needed for implementation. This is due to the increase in datapath width in the 32-bit design. The maximum number of LUTs is needed for the implementation of the Carry Lookahead Adder architecture. It requires 540 LUTs for 16-bit design and 1135 LUTs for 32-bit design since more hardware resources are needed for the implementation of generate and propagate signals. The Kogge-Stone Adder architecture also requires a large number of LUT resources since it is of parallel prefix type. The Ripple Carry Adder architecture requires fewer LUT resources since it is of sequential type. It requires 380 LUT resources for 16-bit design and 620 LUT resources for 32-bit design.

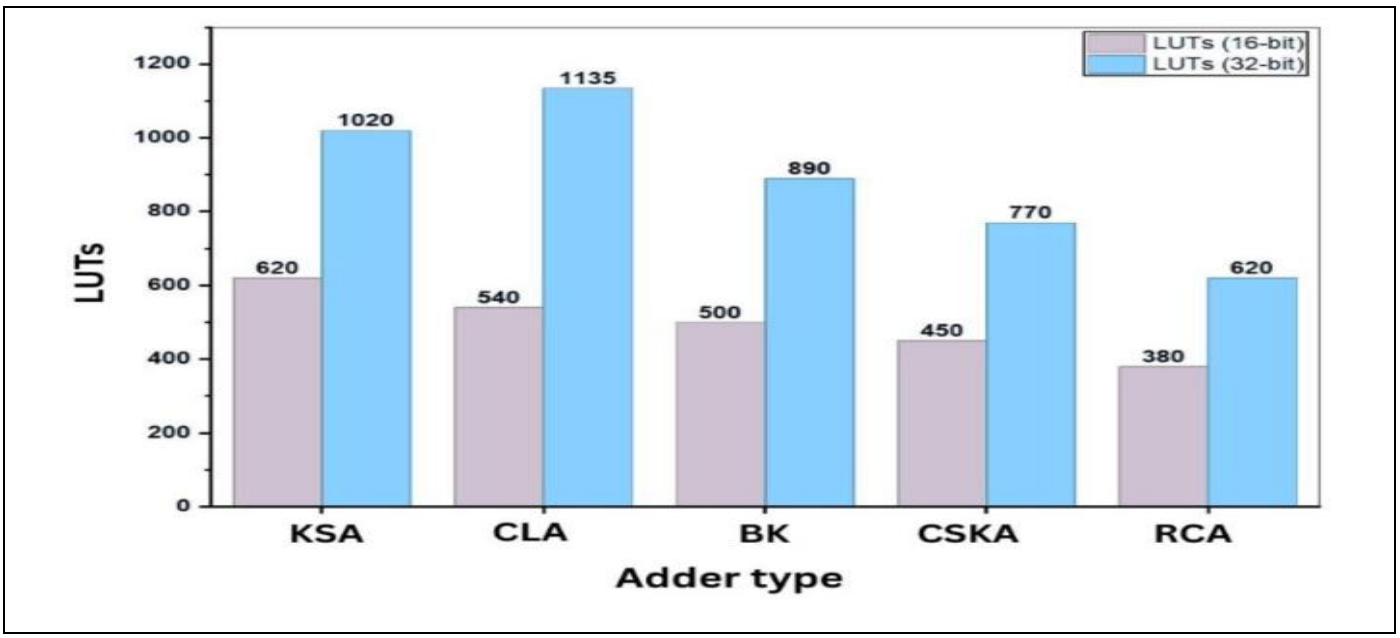

Fig 23 LUT Utilization Comparison of 16-Bit and 32-Bit MAC Architectures.

➢ *Flip-Flop Utilization Comparison of 16-Bit and 32-Bit MAC Architectures*

The utilization graph (fig. 24) for the flip-flop shows the usage of the registers in the different MAC architectures. In the case of the 16-bit implementation, it is observed that all the architectures require approximately 34-35 flip-flops, except the Carry Skip Adder, which requires fewer flip-flops. In the case of the 32-bit implementation, it is observed that all the architectures require a few more flip-flops than in the 16-bit implementation, as the width of the datapath increases in this case. In this case, the Kogge-Stone Adder requires the maximum flip-flops, while the Brent-Kung Adder and Ripple Carry Adder require fewer flip-flops, though not significantly different, as the flip-flops are required to store the results in this case also.

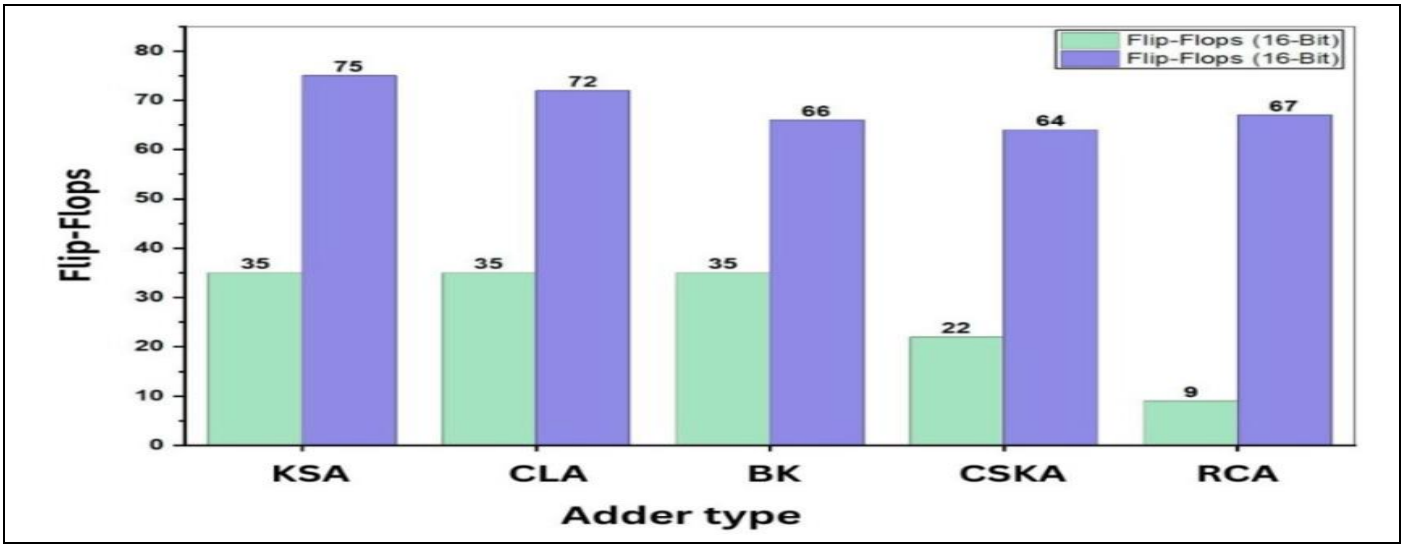

Fig 24 Flip-Flop Utilization Comparison of 16-Bit and 32-Bit MAC Architectures.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

➢ *Power–Delay Product (PDP) Comparison*

The PDP graph (fig. 25) combines the power consumption and propagation delay. In the case of the 16-bit MAC architecture, the minimum value of PDP is obtained by the Kogge-Stone Adder at 304 pJ. This indicates the efficient performance of the adder in terms of high-speed operation. Another adder that has good performance is the Brent-Kung Adder, as the PDP value is 312 pJ. In contrast, the RCA has the maximum value of PDP at 500 pJ because of the large value of the propagation delay and the low power consumption.

In the case of the 32-bit MAC architecture, the PDP values increase for all the architectures because the power consumption and delay values increase. However, the prefix adders have the best performance in terms of speed compared to the other architectures.

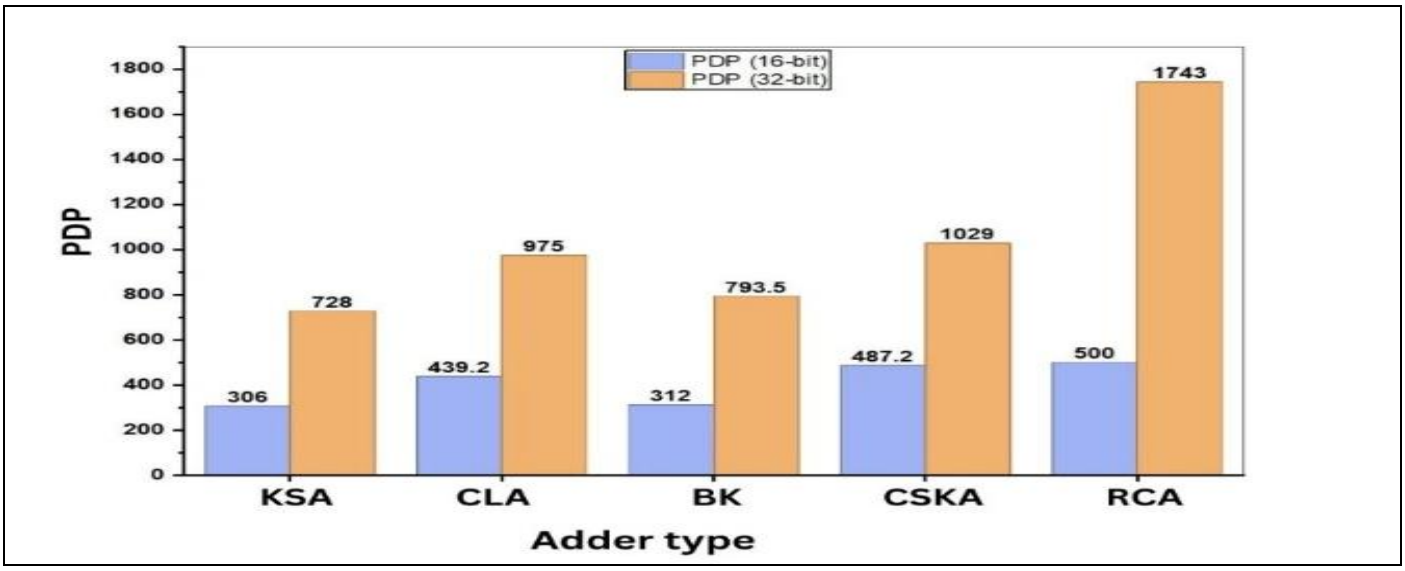

Fig 25 Power–Delay Product (PDP) Comparison of 16-Bit and 32-Bit MAC Architectures.

➢ *Energy–Delay Product (EDP) Comparison*

The EDP graph (fig. 26) is used to evaluate the performance of the energy efficiency and computational speed combined. In the case of the 16-bit implementation, the Kogge-Stone Adder has the least value in the EDP chart, which is 972.8, indicating that the adder is operating efficiently with minimal delay. The Brent-Kung Adder has a good balance between the power consumption and delay values as well. The Ripple Carry Adder has the highest value in the EDP chart, which is 6250, due to the maximum delay value. In the case of the 32-bit MAC implementation, the values in the EDP chart are increased due to the increased datapath width and the consumption of maximum power. Among the architectures, the Brent-Kung and Kogge-Stone adders have good values compared to the other adder architectures.

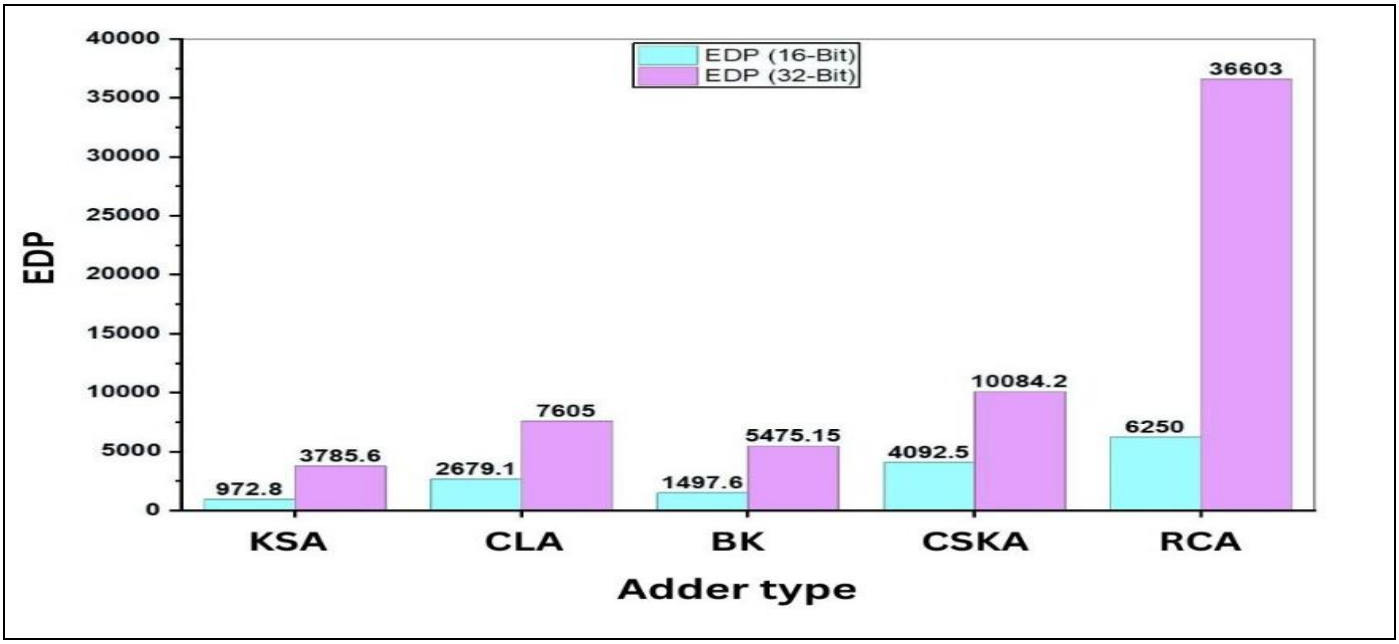

Fig 26 Energy–Delay Product (EDP) Comparison of 16-Bit and 32-Bit MAC Architectures.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

Overall, the implementation results show that the adder architectures indeed have significant effects on the performance of the Vedic Multiplier-based MAC unit in terms of area, power consumption, and speed. As the datapath width is increased from 16-bit to 32-bit, the hardware utilization, power consumption, and propagation delay are increased for all the architectures due to the increased complexity of the logic circuits. Among the architectures, the Kogge-Stone Adder has the least propagation delay compared to the other architectures due to the parallel prefix carry computation method employed in the design. However, the increased propagation delay is achieved at the expense of increased hardware complexity and power consumption. The Ripple Carry Adder has the least hardware complexity and power consumption, but the propagation delay is the highest due to the sequential carry computation method employed in the design. The Brent-Kung Adder has the best trade-off between the hardware complexity, power consumption, and propagation delay and can be efficiently used in the design of the MAC unit using FPGAs.

VI. DISCUSSION

The results obtained from the experiment show that the Vedic multiplier, on its own, requires relatively low power when the adders are designed with different architectures because the multiplication operation is the only operation being performed, and the switching activity is minimal. However, when the Vedic multiplier and the MAC unit are combined, the power consumption increases. This is because the operations are being performed continuously, and the switches are being activated during each clock cycle.

Among the compared structures, the Kogge–Stone Adder has the minimum propagation delay due to its method of computing the carry bits in parallel, while the Ripple Carry Adder has the minimum power consumption and hardware utilization but the highest delay due to its method of computing the carry bits serially. The Brent–Kung Adder offers the optimal trade-offs between delay, power, and hardware utilization, thus suitable for efficient FPGA-based MAC operations.

VII. CONCLUSION

This paper has discussed the implementation and system-level PPA trade-off analysis of the Vedic multiplier-based MAC architectures using the following adders: Kogge-Stone Adder (KSA), Carry Lookahead Adder (CLA), Brent-Kung Adder (BK), Carry Skip Adder (CSKA), and Ripple Carry Adder (RCA). The MAC architectures were implemented on the Zynq-7000-based ZedBoard platform for 16-bit and 32-bit datapath widths.

Based on the results of implementing the architectures, it is observed that the architecture of the adder plays a significant role in determining the performance of the MAC architecture. The Kogge–Stone Adder resulted in the minimum propagation delay of 3.2 ns for 16-bit and 5.2 ns for 32-bit since it has a parallel prefix carry computation technique, thereby resulting in the least PDP and EDP of 304 pJ and 972.8 pJ for 16-bit and 728 pJ and 3785.6 pJ for 32-bit respectively. However, it is observed that the performance of the MAC architecture is achieved with the cost of increased hardware utilization and power dissipation. In addition, the RCA resulted in the least hardware complexity and power dissipation of 40 mW for 16-bit and 83 mW for 32-bit since it has a simple sequential carry propagation technique; however, it resulted in the maximum delay of 12.5 ns and 21 ns for 16-bit and 32-bit respectively. Among all the architectures, the Brent–Kung Adder resulted in a fair trade-off in hardware utilization, power dissipation, and delay performance.

Analysis results have also demonstrated that the increase in datapath width from 16-bit to 32-bit results in the increase of LUT utilization, power consumption, and propagation delay for all architectures. However, the prefix adders still outperform the other architectures in terms of speed performance.

The adder-based Vedic multiplier MAC architectures proposed in the paper can be efficiently used in several high-performance digital signal processing applications like digital filters, Fast Fourier Transform processors, image processors, machine learning processors, and real-time signal processors. The results obtained in the paper can be used to select the appropriate adder architectures according to the requirements of the applications in terms of high-speed performance, low power consumption, and minimum hardware complexity in FPGA-based arithmetic processors.

ACKNOWLEDGMENT

We acknowledge the support and help from the ECE Department, Meghnad Saha Institute of Technology for the execution of the work.

REFERENCES

[1]. Ananthakrishnan, A. Ajit, A. P.V., K. Haridas, N. M. Nambiar and D. S., "FPGA Based Performance Comparison of Different Basic Adder Topologies with Parallel Processing Adder," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2019, pp. 87-92, doi: 10.1109/ICECA.2019.8821925.

[2]. B. Harish, K. Sivani and M. S. S. Rukmini, "Design and Performance Comparison among Various types of Adder Topologies," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2019, pp. 725-730, doi: 10.1109/ICCMC.2019.8819779.

[3]. S. Gauhar, A. Sharif and N. Alam, "Comparison of Parallel Prefix Adders Based on FPGA & ASIC Implementations," 2020 IEEE Students Conference on Engineering & Systems (SCES), Prayagraj, India, 2020, pp. 1-6, doi: 10.1109/SCES50439.2020.9236737.

[4]. J. Anirudh, A. R. S. Babu, K. M. Gopi, S. Karishma and S. Musala, "FPGA Based Low Power Approximate Hybrid Parallel Prefix Adders with Less

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

Area," 2025 International Conference on Information, Implementation, and Innovation in Technology (I2ITCON), Pune, India, 2025, pp. 1-6, doi: 10.1109/I2ITCON65200.2025.11210724.

[5]. R. Priya and J. S. Kumar, "Implementation and comparison of effective area efficient architectures for CSLA," 2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN), Tirunelveli, India, 2013, pp. 287-292, doi: 10.1109/ICE-CCN.2013.6528510.

[6]. R. Sahu and A. K. Subudhi, "An area optimized Carry Select Adder," 2015 IEEE Power, Communication and Information Technology Conference (PCITC), Bhubaneswar, India, 2015, pp. 589-594, doi: 10.1109/PCITC.2015.7438066.

[7]. P. Balasubramanian and D. Maskell, "Hardware Efficient Approximate Adder Design," TENCON 2018 - 2018 IEEE Region 10 Conference, Jeju, Korea (South), 2018, pp. 0806-0810, doi: 10.1109/TENCON.2018.8650127.

[8]. K. S and H. P. N, "Analysis of Low Power Approximate 128x128 bit Vedic Multiplier for Existing Real World FPGAs," 2024 5th IEEE Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2024, pp. 1-6, doi: 10.1109/GCAT62922.2024.10924019.

[9]. G. Thakur, H. Sohal and S. Jain, "Design and Analysis of High-Speed Parallel Prefix Adder for Digital Circuit Design Applications," 2020 International Conference on Computational Performance Evaluation (ComPE), Shillong, India, 2020, pp. 095-100, doi: 10.1109/ComPE49325.2020.9200064.

[10]. B. R. S. Khater, M. A. M. Alshewimy and M. T. Faheem Saidahmed, "An analysis technique for improving delay factor of carry select adder using FPGA," 2017 12th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 2017, pp. 14-18, doi: 10.1109/ICCES.2017.8275269.

[11]. M S. N. Gadakh and A. Khade, "Design and optimization of 16×16 Bit multiplier using Vedic mathematics," 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), Pune, India, 2016, pp. 460-464, doi: 10.1109/ICACDOT.2016.7877628.

[12]. Shing, L. & Hussin, Razaidi & Kamarudin, A. & Mohyar, Shaiful Nizam & Taking, Sanna & Aziz, Hafiz & Ahmad, Norhawati. (2018). 16×16 fast signed multiplier using Booth and Vedic architecture. AIP Conference Proceedings. 2045. 020085. 10.1063/1.5080898.

[13]. A., S., A., S. Modified vedic multiplier architecture using Nikhilam and Karatsuba algorithms with hybrid adders for enhanced performance. Sci Rep 16, 1772 (2026). https://doi.org/10.1038/s41598-025-30966-7

[14]. J. Kuppili, M. Abhiram and N. A. Manga, "Design of Vedic Mathematics based 16 bit MAC unit for Power and Delay Optimization," 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), NaviMumbai, India, 2021, pp. 1-4, doi: 10.1109/ICNTE51185.2021.9487570.

[15]. Pradhan, Manoranjan & Panda, Rutuparna & Sahu, Sushanta. (2009). Speed Comparison of 16x16 Vedic Multipliers. International Journal of Computer Applications. 21. 10.5120/2516-3417.

[16]. Gadakh, Sheetal & Khade, Amitkumar. (2016). Design and optimization of 16×16 Bit multiplier using Vedic mathematics. 460-464. 10.1109/ICACDOT.2016.7877628.

[17]. S. Ravi, M. S. N. V. Mohith, K. Yaswanth Simha, L. Alekhya, and M. Maruthi Sriram, "A Novel High Performance Architecture for MAC Unit Using Vedic Multiplier and Brent-Kung Adder,"International Journal of Research and Scientific Innovation (IJRSI), vol. 12, no. 4, Apr. 2025, doi: 10.51244/IJRSI.2025.12040021.

[18]. Gupta, V., & Kumar, M. (2018). Design of high speed 16x16 bit MAC units using Vedic multiplier. International Journal of Computer Applications, 182(17), 45–48. DOI: 10.5120/ijca2018917895. (ISSN: 0975–8887).

[19]. Nitin Krishna, V. (2020). Performance analysis of MAC unit using Booth, Wallace tree, array and Vedic multipliers. International Journal of Engineering Research & Technology (IJERT), 9(09), 497–504. DOI: 10.17577/ijertv9is090337. (ISSN: 2278-0181; Paper ID: IJERTV9IS090337).

[20]. Paldurai, K., Hariharan, K., Karthikeyan, G.C., & Lakshmanan, K. (2014). Implementation of MAC using area efficient and reduced delay Vedic multiplier targeted at FPGA architectures. In 2014 International Conference on Communication and Network Technologies (ICCNT) (pp. 238–242). IEEE. DOI: 10.1109/ICCNT.2014.7062793.

[21]. Shinde and A. O. Mulani, "A Comprehensive Review on Improving 256x256 Vedic Multiplier Design Using Optimized Adder Architectures," International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), vol. 5, no. 4, pp. 1210–1226, Oct. 2025. DOI: 10.48175/IJARSCT-29467.

[22]. G. Rashmi, P. Sharmila Rani, and S. Nagi Reddy, "Majority Logic Implementation of MAC Unit," Journal of Emerging Technologies and Innovative Research (JETIR), vol. 5, no. 7, pp. 850–856, Jul. 2018. ISSN: 2349-5162. Paper ID: JETIR1807851. DOI: 10.1729/Journal.JETIR1807851.

[23]. S. A. Ahmed and M. Salahuddin, "Design of High Speed Architecture of Parallel MAC Based On Radix-2 MBA," International Journal of Engineering Research and Applications (IJERA), vol. 4, no. 5 (Version 7), pp. 56–61, May 2014. ISSN: 2248-9622. DOI: 10.9790/9622-0405075661.

[24]. K. B. Jagannatha, H. S. Lakshmisagar, and G. R. Bhaskar, "FPGA and ASIC Implementation of 16-Bit Vedic Multiplier Using Urdhva Triyakbhyam Sutra," in Emerging Research in Electronics, Computer Science and Technology, V. Sridhar et al. (eds.), Lecture Notes in Electrical Engineering, vol. 248, pp. 31–38. Springer India, 2014. DOI: 10.1007/978-81-322-1157-0_4

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar690

[25]. M. Mulkalapally, J. Manning, P. Gatewood, and T. Nikoubin, "High Speed, Area and Power Efficient 32-bit Vedic Multipliers," in Proceedings of the 7th International Conference on Computing Communication and Networking Technologies (ICCCNT '16), Dallas, TX, USA, Jul. 6–8, 2016, pp. 1–8. DOI: 10.1145/2967878.2967890.