ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

DESIGN AND FPGA IMPLEMENTATION OF HIGH-PERFORMANCE 64-BIT HYBRID ADDER ARCHITECTURES FOR OPTIMIZED POWER-DELAY-AREA PERFORMANCE

MANASA ENJAPURI[1], DEVISINGH[2]

[1]PG Scholar, VLSI SYSTEM DESIGN, ECE DEPARTMENT, JNTUH University College of Engineering, Sultanpur. enjapurimanasa11@gmail.com

[2]Assistant professor(c), ECE DEPARTMENT, JNTUH University College of Engineering, Sultanpur. devisinghrathod29@gmail.com

Abstract: Modern digital systems require arithmetic circuits that provide high speed, low power consumption, and efficient hardware utilization. Adders are among the most critical components in processors, digital signal processors, communication systems, and embedded platforms, directly influencing overall computational performance. Conventional adder architectures such as Ripple Carry Adders (RCA), Carry Lookahead Adders (CLA), Carry Select Adders (CSLA), Carry Skip Adders (CSKA), and Parallel Prefix Adders exhibit inherent trade-offs between propagation delay, power consumption, and area utilization. To address these limitations, this paper presents the design, implementation, and comparative evaluation of several high-performance 64-bit hybrid adder architectures based on hierarchical partitioning and hybridization techniques. The proposed designs combine the advantages of Carry Lookahead, Carry Select, Carry Skip, and Kogge-Stone Adders to achieve improved arithmetic performance while maintaining balanced hardware complexity. Multiple hybrid architectures, including CLA (32) + KSA (32), CSLA (32) + KSA (32), (CSLA (16) + KSA (16)) + KSA (32), (CSLA (16) + KSA (16)) + CSKA (32), RCA (32) +CSKA (32), and other hierarchical combinations, are modeled using Verilog HDL and synthesized using Xilinx Vivado targeting FPGA platforms. Functional verification and synthesis analysis are performed to evaluate propagation delay, power consumption, area utilization, and overall Power-Delay-Area (PDA) performance. Experimental results demonstrate that hybrid architectures significantly reduce critical path delay while achieving better resource utilization compared to conventional adder designs. The hierarchical hybrid approach provides an effective solution for high-speed arithmetic operations in modern VLSI systems, processors, signal processing units, and FPGA-based computing applications.

Keywords: Hybrid Adder Architecture, Kogge-Stone Adder (KSA), Carry Lookahead Adder (CLA), Carry Select Adder (CSLA), Carry Skip Adder (CSKA), Ripple Carry Adder (RCA), Parallel Prefix

Adder, FPGA Implementation, Verilog HDL, Xilinx Vivado, VLSI Design, Power-Delay-Area (PDA), High-Speed Arithmetic Circuits, Digital Signal Processing, Embedded Systems.

I. INTRODUCTION

The continuous advancement of digital technology has significantly increased the demand for high-speed and energy-efficient arithmetic circuits in modern computing systems. Arithmetic operations form the foundation of digital processing and are extensively used in microprocessors, digital signal processors (DSPs), embedded systems, communication devices, graphics processors, and scientific computing applications. Among all arithmetic operations, addition is the most frequently executed operation and serves as a fundamental building block for subtraction, multiplication, division, address generation, encoding, decoding, and various signal processing tasks. Consequently, the performance of an arithmetic system is highly dependent on the efficiency of its underlying adder architecture.

As semiconductor technology continues to evolve, modern electronic devices are expected to provide higher computational performance while maintaining low power consumption and reduced hardware complexity. Portable devices such as smartphones, tablets, wearable electronics, and Internet of Things (IoT) systems require arithmetic circuits that can operate efficiently under stringent power constraints. Similarly, high-performance computing systems demand ultra-fast arithmetic units capable of processing large volumes of data with minimal latency. These conflicting requirements have made the design of efficient adder architectures one of the most important research areas in Very Large Scale Integration (VLSI) design.

The performance of an adder is generally evaluated based on three critical parameters: propagation delay, power consumption, and area utilization. Propagation delay determines the speed at which arithmetic operations can be completed, while power consumption affects the energy efficiency of the system. Area utilization represents the amount of hardware resources required for implementation. Improving one of these parameters often leads to degradation in the others, resulting in a design trade-off commonly known as the Power-Delay-Area (PDA) trade-off. Therefore, the primary challenge in adder design is to achieve an optimal balance among these performance metrics.

Various adder architectures have been proposed over the years to address these challenges. The Ripple Carry Adder (RCA) is one of the simplest and most area-efficient adder structures. It consists of a cascade of full adders in which the carry output from one stage is propagated to the next stage. Although RCA requires minimal hardware resources and consumes relatively low power, its sequential carry propagation mechanism results in large propagation delays, particularly for high bit-width operations. Consequently, RCA becomes unsuitable for applications requiring high-speed arithmetic processing.

To overcome the limitations of Ripple Carry Adders, Carry Lookahead

Adders (CLA) were developed. CLA architectures utilize generate and propagate signals to compute carry outputs in parallel, significantly reducing propagation delay. The ability to predict carry signals in advance enables faster computation compared to sequential carry propagation methods. However, as the bit width increases, the complexity of carry generation logic and interconnection networks also increases, leading to higher area utilization and power consumption.

Another widely used architecture is the Carry Select Adder (CSLA), which reduces delay by computing sum outputs simultaneously for different carry input conditions. The correct output is selected using multiplexers once the actual carry value becomes available. This parallel computation technique improves speed considerably but requires duplicated hardware resources, resulting in increased area and power requirements. Similarly, the Carry Skip Adder (CSKA) introduces carry bypass mechanisms that allow carry signals to skip certain adder blocks under specific conditions, thereby reducing effective carry propagation delay. Although CSKA provides moderate speed improvement with lower complexity than CLA and CSLA, its performance depends heavily on block partitioning strategies.

Parallel Prefix Adders (PPAs) represent one of the most significant advancements in high-speed arithmetic design. These adders employ tree-based structures to perform carry computation in parallel, drastically reducing logic depth and propagation delay. Among various parallel prefix architectures, the Kogge-Stone Adder (KSA) is recognized as one of the fastest adder designs due to its logarithmic carry computation structure. KSA utilizes generate and propagate signals in a prefix tree network to compute carries simultaneously across multiple stages. While KSA achieves exceptional speed performance, it requires a large number of interconnections and logic resources, leading to increased area consumption, routing complexity, and power dissipation.

Because no single adder architecture can simultaneously optimize speed, power, and area, researchers have increasingly focused on hybrid adder architectures. Hybrid adders combine the advantages of multiple adder designs within a single structure to achieve balanced overall performance. By strategically integrating fast carry computation techniques in critical sections and area-efficient structures in non-critical regions, hybrid architectures can significantly improve the Power-Delay-Area trade-off. Such designs are particularly attractive for large bit-width arithmetic operations where both speed and resource efficiency are important.

Hierarchical design methodologies further enhance the effectiveness of hybrid architectures. Instead of implementing a large adder as a monolithic structure, the design is partitioned into smaller modules such as 16-bit or 32-bit blocks. Different adder architectures can then be assigned to different sections based on performance requirements. This modular approach improves scalability, simplifies implementation, and provides greater

flexibility for performance optimization. Furthermore, hierarchical hybrid architectures facilitate FPGA implementation by reducing routing complexity and improving resource utilization.

The primary objective of this research is to investigate the impact of hybridization on adder performance and identify architectures that provide optimal Power-Delay-Area characteristics. A comprehensive comparative analysis is performed considering propagation delay, power consumption, area utilization, and overall PDA metrics. The results provide valuable insights into the effectiveness of different hybrid design strategies and their suitability for high-speed arithmetic applications.

The major contributions of this work include the development of multiple hierarchical hybrid adder architectures, FPGA-based implementation using Verilog HDL, detailed performance evaluation across multiple design metrics, and identification of efficient trade-offs between speed, power, and hardware complexity. The proposed hybrid architectures offer practical solutions for next-generation processors, digital signal processing systems, communication hardware, and embedded computing platforms where efficient arithmetic computation is essential.

II. LITERATURE SURVEY

Adders are among the most important arithmetic components in digital systems and significantly influence the performance of processors, digital signal processing systems, communication devices, and embedded applications. Over the years, researchers have proposed numerous adder architectures to improve speed, reduce power consumption, and optimize hardware utilization. This section presents a comprehensive review of conventional, parallel prefix, and hybrid adder architectures that have contributed to the development of modern high-performance arithmetic circuits.

Ripple Carry Adder (RCA)

The Ripple Carry Adder is one of the earliest and most widely used adder architectures due to its simple structure and low hardware complexity. Reto Zimmermann extensively analyzed binary adder architectures and identified the RCA as a fundamental reference design for evaluating arithmetic circuits. The architecture consists of a chain of full adders where the carry output of one stage becomes the carry input of the next stage. Although RCA requires minimal hardware resources and consumes relatively low power, its major drawback is the sequential propagation of carry signals. As the bit width increases, propagation delay grows linearly, making RCA unsuitable for high-speed arithmetic applications. Nevertheless, due to its simplicity and area efficiency, RCA remains useful in low-cost and low-power digital systems.

Carry Lookahead Adder (CLA)

To overcome the delay limitations of Ripple Carry Adders, Carry Lookahead Adders were introduced. Weste and Harris demonstrated that CLA architectures utilize generate and propagate signals to predict carry outputs in advance, thereby eliminating

the need for sequential carry propagation. The carry generation equations enable parallel computation of carry signals, significantly reducing propagation delay. CLA architectures provide superior speed performance compared to RCA and are commonly employed in high-speed arithmetic circuits. However, the carry generation logic becomes increasingly complex as the bit width increases, resulting in higher hardware overhead, increased power consumption, and greater routing complexity.

Carry Select Adder (CSLA)

The Carry Select Adder was originally proposed by Bedrij to improve arithmetic speed by performing parallel computations for two possible carry input conditions. In a CSLA, one adder computes the output assuming carry input equals zero, while another computes the output assuming carry input equals one. A multiplexer subsequently selects the correct result once the actual carry input becomes available. This parallel processing mechanism significantly reduces carry propagation delay and improves speed. Ramkumar and Kittur later proposed optimized CSLA structures using Binary-to-Excess-1 Converters (BEC) to reduce hardware duplication. Although CSLA achieves faster operation than RCA and CLA in many cases, the requirement for duplicated adder blocks leads to increased area utilization and power consumption.

Carry Skip Adder (CSKA)

The Carry Skip Adder was developed to improve carry propagation efficiency while maintaining moderate hardware complexity. This architecture divides the adder into multiple blocks and employs carry bypass logic to allow carry signals to skip intermediate stages whenever specific propagate conditions are satisfied. Several researchers have shown that CSKA provides lower delay than RCA while requiring fewer resources than CLA and CSLA architectures. The effectiveness of CSKA depends heavily on block partitioning strategies. Improper selection of block sizes may reduce the efficiency of carry skipping, limiting the overall speed improvement. Despite this limitation, CSKA offers an attractive compromise between speed and area.

Parallel Prefix Adders

Parallel Prefix Adders represent a major breakthrough in high-speed arithmetic design. Unlike conventional adders that rely on sequential carry propagation, parallel prefix adders utilize tree-based structures to perform carry computations simultaneously across multiple stages. The use of generate and propagate signals organized in prefix networks significantly reduces logic depth and propagation delay. Various prefix adder architectures have been proposed, including Kogge-Stone, Brent-Kung, Sklansky, and Han-Carlson adders. These architectures differ primarily in terms of logic depth, wiring complexity, fan-out requirements, and area utilization.

Kogge-Stone Adder (KSA)

The Kogge-Stone Adder, introduced by Peter M. Kogge and Harold S. Stone, is one of the fastest parallel

prefix adders available. The architecture employs a logarithmic-depth prefix tree that enables rapid carry computation across all bit positions. The carry generation process utilizes generate and propagate signals combined through prefix operators. Because of its highly parallel structure, KSA achieves extremely low propagation delay and is widely used in high-performance processors and digital signal processing systems. However, the architecture requires a large number of interconnections and prefix nodes, resulting in increased wiring complexity, routing congestion, area utilization, and power consumption. These limitations become more significant in FPGA implementations where routing resources are limited.

Hybrid Adder Architectures

Recent research has focused on hybrid adder architectures that combine multiple adder designs to exploit their individual advantages while minimizing their limitations. Hybrid adders integrate different carry generation and propagation techniques within a single structure to achieve balanced performance. Researchers have demonstrated that combining high-speed prefix adders with area-efficient architectures can significantly improve overall Power-Delay-Area (PDA) characteristics.

Hybrid designs such as CLA-KSA, CSLA-KSA, RCA-KSA, and CSKA-KSA have been explored in various studies. The general principle involves assigning high-speed architectures to critical carry propagation regions while employing simpler structures in less critical sections. Such combinations help reduce propagation delay without incurring excessive hardware overhead. Experimental results reported in the literature indicate that hybrid architectures often outperform standalone adders when evaluated using comprehensive performance metrics.

Hierarchical Hybrid Adder Architectures

Hierarchical design methodologies have further enhanced the effectiveness of hybrid adder structures. In hierarchical architectures, large-bit-width adders are partitioned into smaller modules such as 16-bit or 32-bit blocks. Different adder architectures are then assigned to each module according to performance requirements. This approach improves scalability, simplifies routing, and enables more efficient utilization of FPGA resources.

Several researchers have proposed hierarchical combinations involving Carry Select Adders, Carry Lookahead Adders, Carry Skip Adders, and Parallel Prefix Adders. These architectures achieve improved delay performance by minimizing long carry propagation paths while maintaining moderate area utilization. Hierarchical hybrid designs have become increasingly attractive for 64-bit arithmetic applications used in modern processors and high-performance computing systems.

FPGA-Based Adder Implementations

Field Programmable Gate Arrays (FPGAs) provide a practical platform for evaluating adder architectures under realistic hardware conditions. Numerous studies have employed FPGA platforms

such as Xilinx Artix-7, Spartan, and Virtex devices to compare arithmetic circuits based on delay, power consumption, and logic utilization. FPGA-based analysis provides valuable insights into routing complexity, resource utilization, and implementation feasibility that may not be fully captured through theoretical analysis alone.

Recent FPGA studies have shown that hybrid adder architectures often provide superior performance compared to conventional adders. The combination of fast carry computation techniques and modular design methodologies enables significant improvements in speed while maintaining acceptable area and power requirements.

Research Gap

Despite substantial progress in adder design, several challenges remain unresolved. Most existing studies focus on optimizing a single performance parameter such as delay or area rather than achieving a balanced Power-Delay-Area trade-off. Parallel prefix adders provide excellent speed but require substantial hardware resources and routing complexity. Conventional adders offer area efficiency but suffer from high propagation delays. Furthermore, many hybrid adder studies evaluate only a limited number of architectures and lack a standardized comparative framework for performance analysis.

Limited research has been conducted on hierarchical 64-bit hybrid architectures that combine multiple adder structures using optimized partitioning strategies. In addition, FPGA-based validation of such architectures remains relatively underexplored. Therefore, there is a need for comprehensive investigation and comparative evaluation of hierarchical hybrid adders capable of achieving improved speed, reduced power consumption, and efficient hardware utilization simultaneously.

The proposed work addresses these research gaps by developing and analyzing multiple 64-bit hierarchical hybrid adder architectures based on combinations of CLA, CSLA, CSKA, RCA, and Kogge-Stone Adders. The designs are implemented using Verilog HDL and evaluated on FPGA platforms to identify the most efficient architecture based on Power-Delay-Area performance metrics.

III. PROPOSED METHODOLOGY

The proposed work focuses on the design and implementation of high-performance 64-bit hybrid adder architectures that achieve an optimized balance between propagation delay, power consumption, and area utilization. Since no individual adder architecture can simultaneously provide minimum delay, low power, and reduced hardware complexity, a hybrid design methodology is adopted. The proposed approach combines the advantages of Carry Lookahead Adders (CLA), Carry Select Adders (CSLA), Carry Skip Adders (CSKA), Ripple Carry Adders (RCA), and Kogge-Stone Adders (KSA) to develop efficient 64-bit arithmetic units suitable for FPGA-based systems.

The architecture utilizes hierarchical partitioning and modular design techniques to divide the 64-bit addition operation into smaller sub-modules. High-speed carry computation structures are employed in critical carry

propagation regions, while area-efficient architectures are used in non-critical sections. This hybridization strategy reduces overall propagation delay while maintaining acceptable power consumption and hardware utilization.

System Architecture

The proposed methodology consists of the following stages:

1. Operand Input Stage

Two 64-bit binary operands A[63:0] and B[63:0] along with the carry input Cin are applied to the hybrid adder architecture.

Input Parameters:

- Operand A = A[63:0]
- Operand B = B[63:0]
- Carry Input = Cin

The operands are partitioned into multiple sections according to the selected hybrid architecture.

2. Hierarchical Partitioning

The complete 64-bit adder is divided into smaller modules to simplify carry computation and improve scalability.

Partitioning approaches include:

- 32-bit + 32-bit partitioning
- 16-bit + 16-bit + 32-bit partitioning
- Multi-level hierarchical partitioning

The hierarchical structure enables independent optimization of different sections of the adder.

3. Lower-Order Computation Module

The lower-order bits are implemented using one of the following architectures:

Carry Lookahead Adder (CLA)

The CLA computes carry signals using generate and propagate logic:

$$G_i=A_iB_i$$

$$P_i=A_i\oplus B_i$$

Carry generation:

$$C_{i+1}=G_i+P_iC_i$$

The CLA provides faster carry computation compared to Ripple Carry Adders.

Carry Select Adder (CSLA)

The CSLA computes outputs simultaneously for:

- Carry-in = 0
- Carry-in = 1

The correct output is selected using multiplexers once the actual carry becomes available.

Advantages:

- Reduced carry propagation delay
- Faster arithmetic operation

4. Upper-Order Computation Module

The upper-order bits are implemented using high-speed Kogge-Stone Adders.

The Kogge-Stone Adder performs carry computation using prefix tree structures.

Prefix operation:

$$(G_k,P_k)*(G_j,P_j)=(G_k+P_kG_j,P_kP_j)$$

The KSA computes carries in parallel using logarithmic-depth stages.

Advantages:

- Minimum carry propagation delay
- Parallel carry generation
- High-speed operation

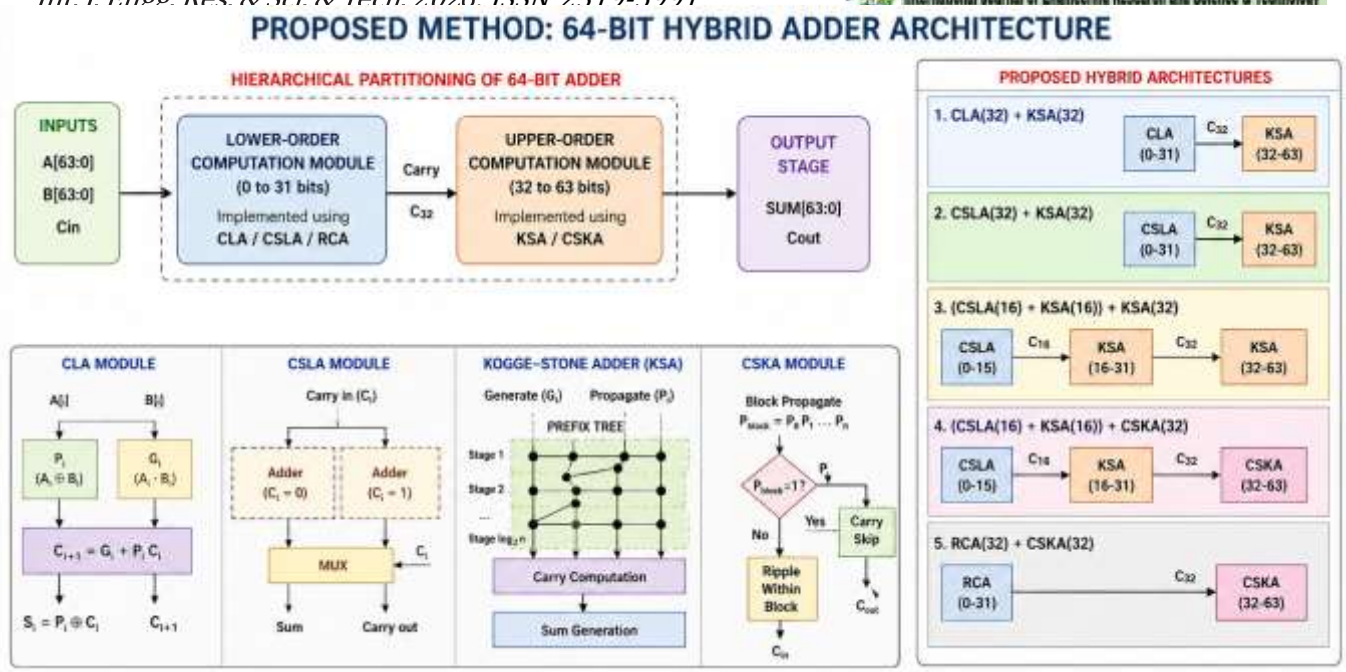


Fig 1 Proposed method

5. Carry Skip Optimization

In selected architectures, Carry Skip Adders are employed in upper-order sections.

The carry bypass condition is:

$$P_{block}=P_0P_1P_2\cdots P_n$$

If all propagate conditions are satisfied:

$$C_{out}=C_{in}$$

This mechanism reduces long carry chain delays.

6. Proposed Hybrid Architectures

The following hybrid architectures are developed and evaluated:

Architecture 1: CLA(32) + KSA(32)

- Lower 32 bits → Carry Lookahead Adder
- Upper 32 bits → Kogge-Stone Adder

Benefits:

- Moderate area
- High-speed operation

Architecture 2: CSLA(32) + KSA(32)

- Lower 32 bits → Carry Select Adder
- Upper 32 bits → Kogge-Stone Adder

Benefits:

- Reduced delay
- Improved carry computation

Architecture 3: (CSLA(16) + KSA(16)) + KSA(32)

- First 16 bits → Carry Select Adder
- Next 16 bits → Kogge-Stone Adder
- Upper 32 bits → Kogge-Stone Adder

Benefits:

- Multi-level optimization
- Minimum critical path delay

Architecture 4: (CSLA(16) + KSA(16)) + CSKA(32)

- Lower section → Hybrid CSLA-KSA
- Upper section → Carry Skip Adder

Benefits:

- Balanced speed and area
- Reduced routing complexity

Architecture 5: RCA(32) + CSKA(32)

- Lower section → Ripple Carry Adder
- Upper section → Carry Skip Adder

Benefits:

- Low area utilization

- Reduced hardware complexity

7. Verilog HDL Implementation

All architectures are developed using Verilog HDL.

Design flow:

1. Behavioural modeling
2. RTL verification
3. Functional simulation
4. FPGA synthesis
5. Timing analysis
6. Power analysis

8. FPGA Synthesis and Validation

The architectures are synthesized using Xilinx Vivado targeting the Artix-7 FPGA platform.

The following metrics are evaluated:

- Propagation Delay (ns)
- Power Consumption (mW)
- LUT Utilization
- Overall PDA Metric

9. Performance Evaluation

The overall performance metric is calculated using:

PDA=Power\times Delay\times Area

Where:

- Power = Dynamic + Static Power
- Delay = Critical Path Delay
- Area = FPGA LUT Utilization

The architecture with the lowest PDA value is considered the most efficient design.

Working Procedure

The complete operation of the proposed hybrid adder is summarized below:

1. Apply 64-bit input operands.
2. Partition operands according to the selected architecture.
3. Compute lower-order sums using CLA, CSLA, or RCA.
4. Generate carry output from lower modules.
5. Propagate carry to upper modules.
6. Perform high-speed carry computation using KSA or CSKA.
7. Generate upper-order sums.
8. Concatenate partial sums.
9. Produce final 64-bit output and carry.
10. Evaluate FPGA performance metrics.

Expected Outcomes

The proposed methodology is expected to achieve:

- Reduced propagation delay.
- Improved arithmetic speed.
- Optimized FPGA resource utilization.
- Lower Power-Delay-Area product.
- Better scalability for large-bit-width arithmetic.
- Enhanced suitability for processors, DSPs, and embedded systems.

The hierarchical hybrid design strategy provides an effective solution for implementing high-performance arithmetic units while maintaining balanced power and area characteristics in modern FPGA-based VLSI systems.

IV. RESULTS AND DISCUSSION

Simulation Results

Behavioral simulation is an essential step in digital VLSI design used to verify the functional correctness of hardware architectures before synthesis and implementation. In this research work, the simulation of all conventional and proposed hybrid 64-bit adder architectures was carried out using Xilinx Vivado. The primary objective of simulation is to validate the correctness of arithmetic operations, carry propagation mechanisms, and

hierarchical integration of the implemented adders.

The simulation process applies different 64-bit binary input combinations to the adder architectures through Verilog HDL testbench modules. The generated outputs are then observed and compared with expected arithmetic results to ensure correct functionality. The simulation verifies the proper generation of:

- Sum output (SUM [63:0])
- Carry output (C_{out})

according to the binary addition relation:

$SUM = A + B + C_{in}$

The waveform analysis helps verify:

- Correct arithmetic operation
- Proper carry propagation
- Stable output generation
- Functional correctness of hierarchical modules

The successful simulation results shown in fig 8.1 to 8.11 all are confirms that all implemented adder architectures perform accurate 64-bit arithmetic operations and proper carry transfer under different input conditions. These verified architectures are further used for synthesis, timing analysis, and comparative Power-Delay-Area (PDA) evaluation in the subsequent sections.

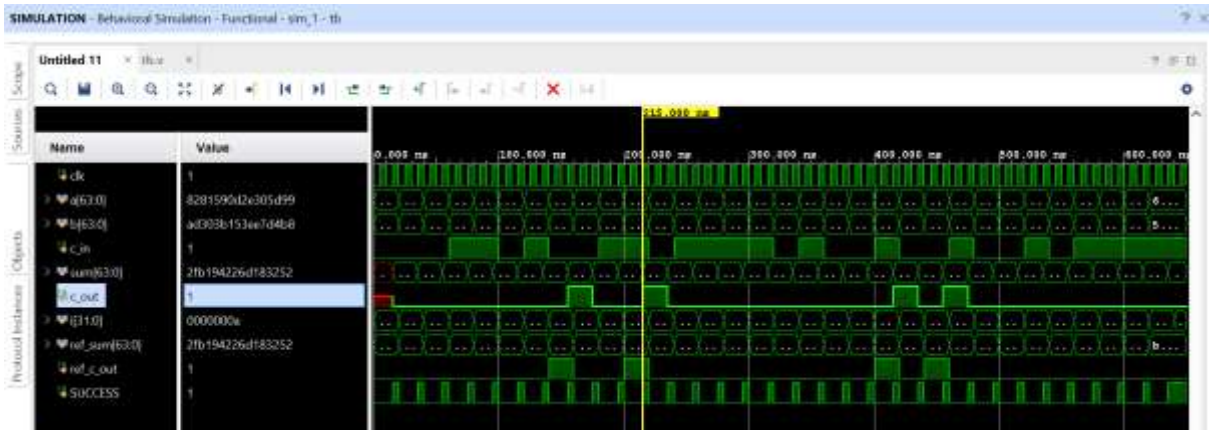


Fig 2 Simulation waveform of Ripple Carry Adder (64-bit)

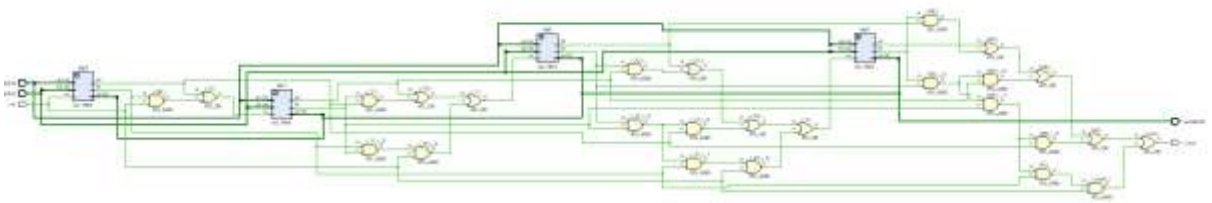


Fig 3 Simulation waveform of Carry Lookahead Adder (64-bit), (b). Elaboration of CLA

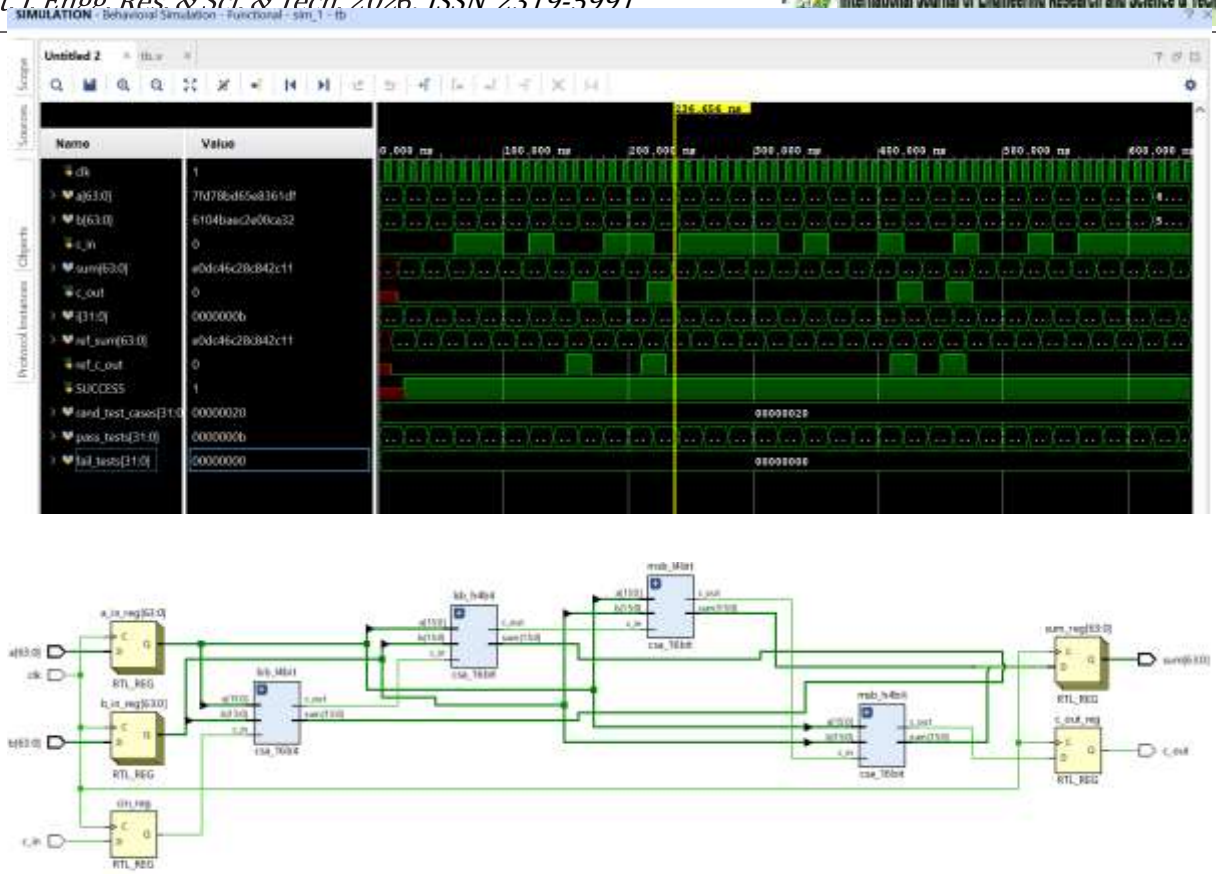


Fig 4 Simulation waveform & ELABORATION of Carry Skip Adder(64-bit)

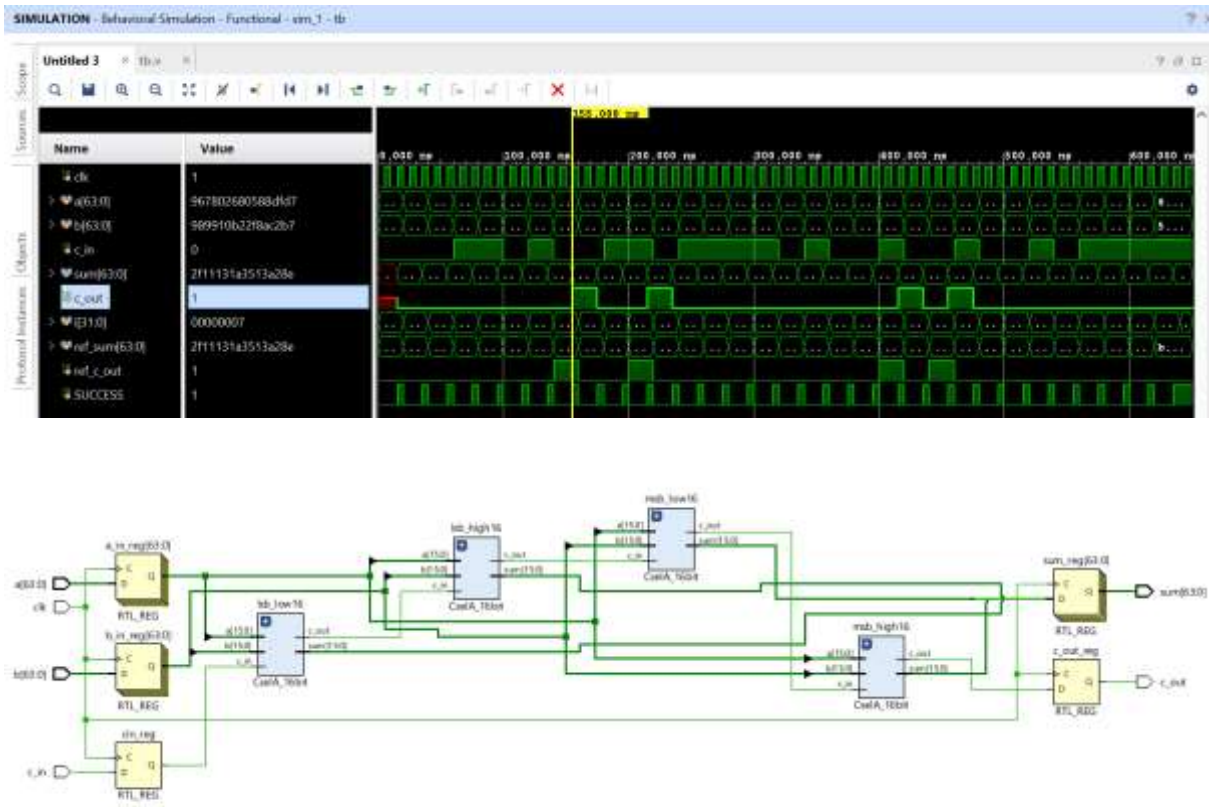


Fig 5 Simulation Waveform of Carry Select Adder (64-bit)

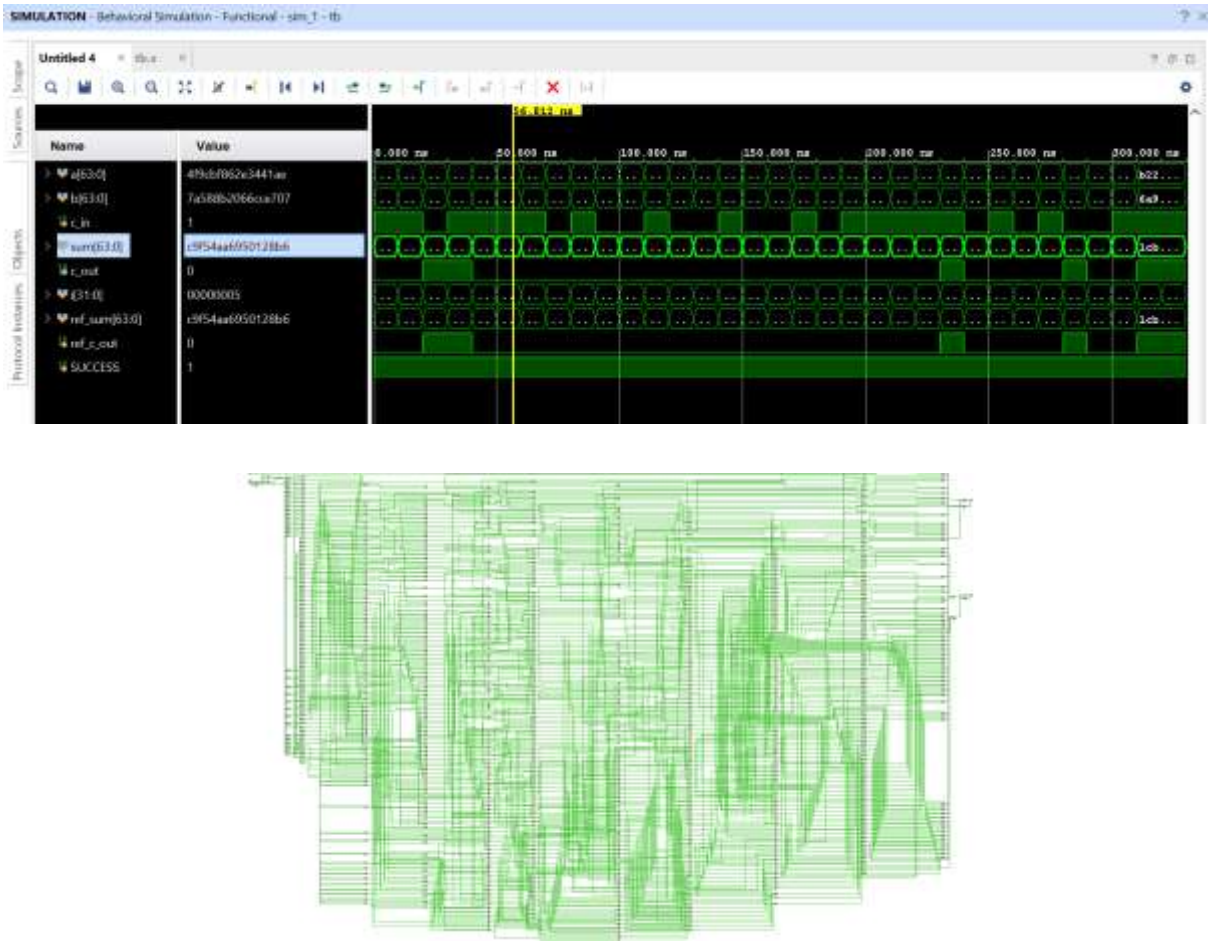
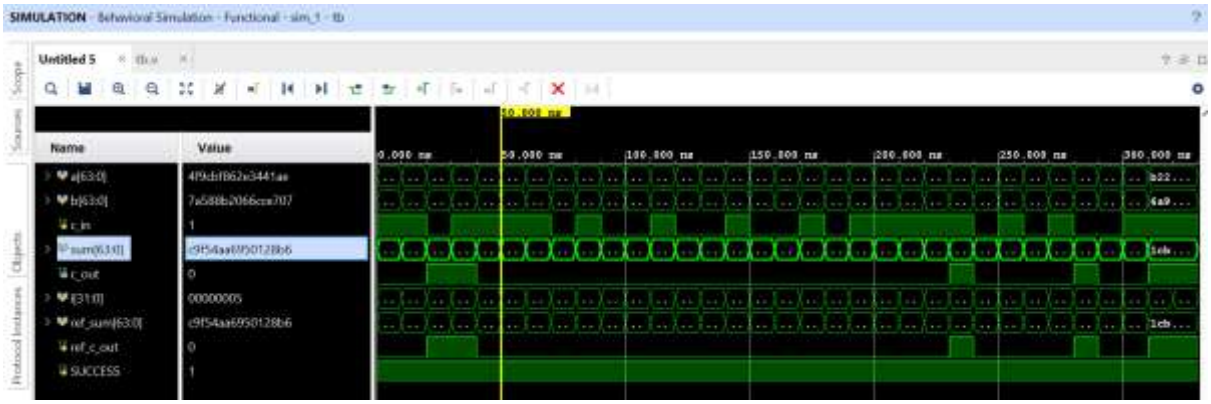
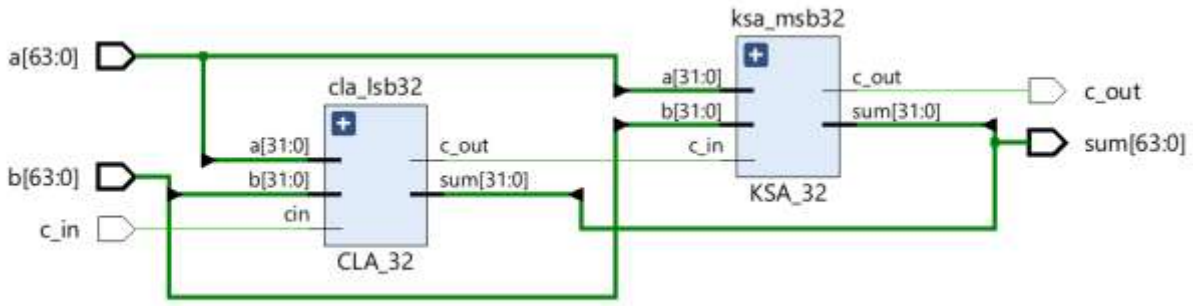
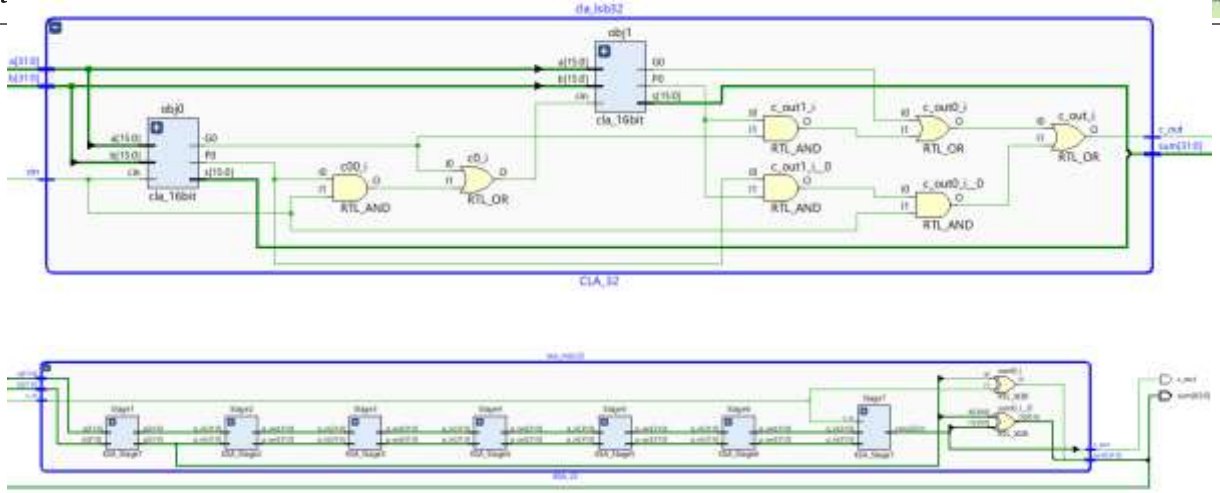


Fig 6 Simulation Waveform of Kogge Stone Adder(64-bit)







(c)

Fig 7 Simulation waveform of Hybrid Adder (CLA (32-bit) + KSA (32-bit))

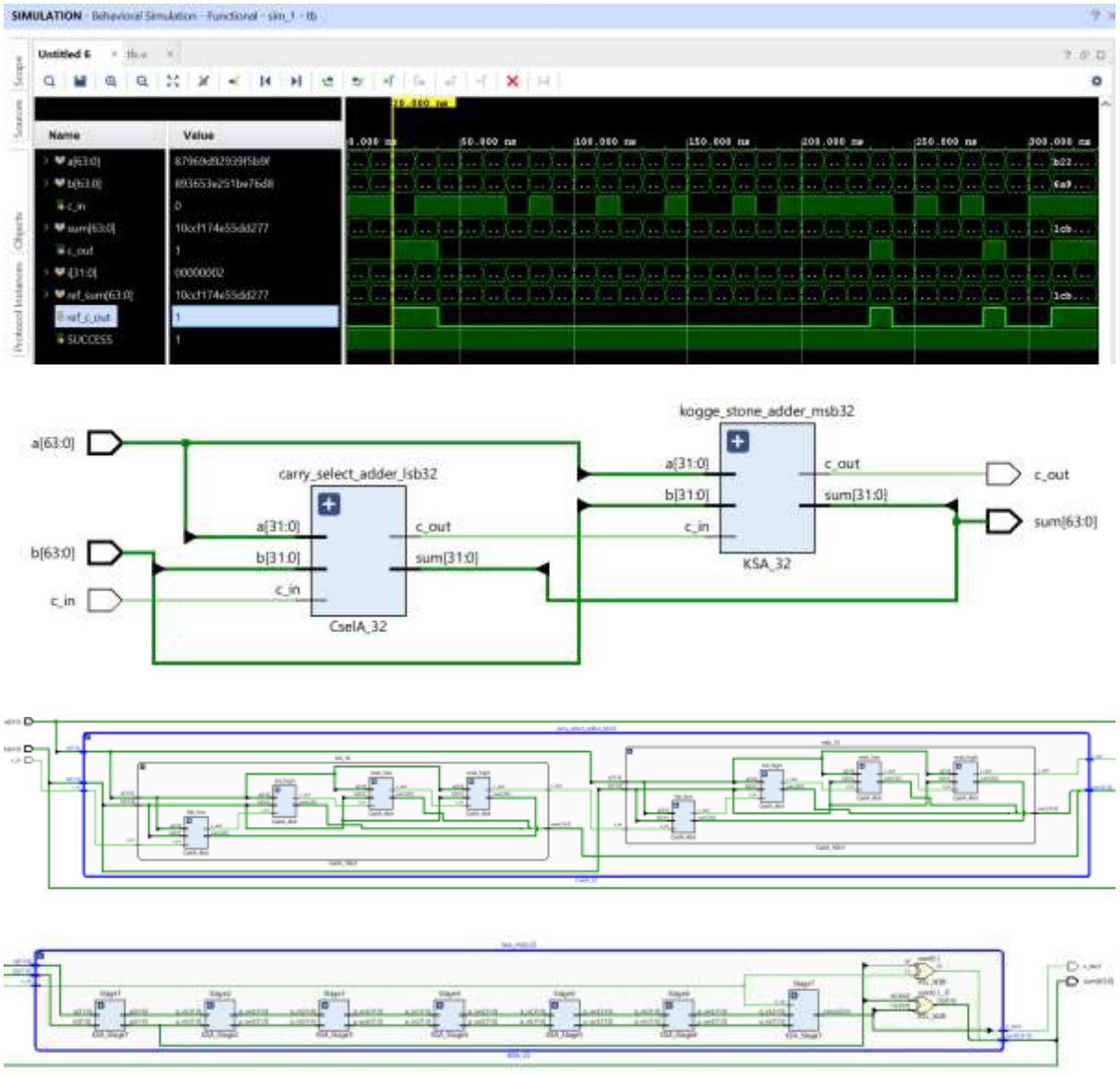


Fig 8 Simulation waveform of Hybrid Adder (CSeLA(32-bit) + KSA (32-bit))

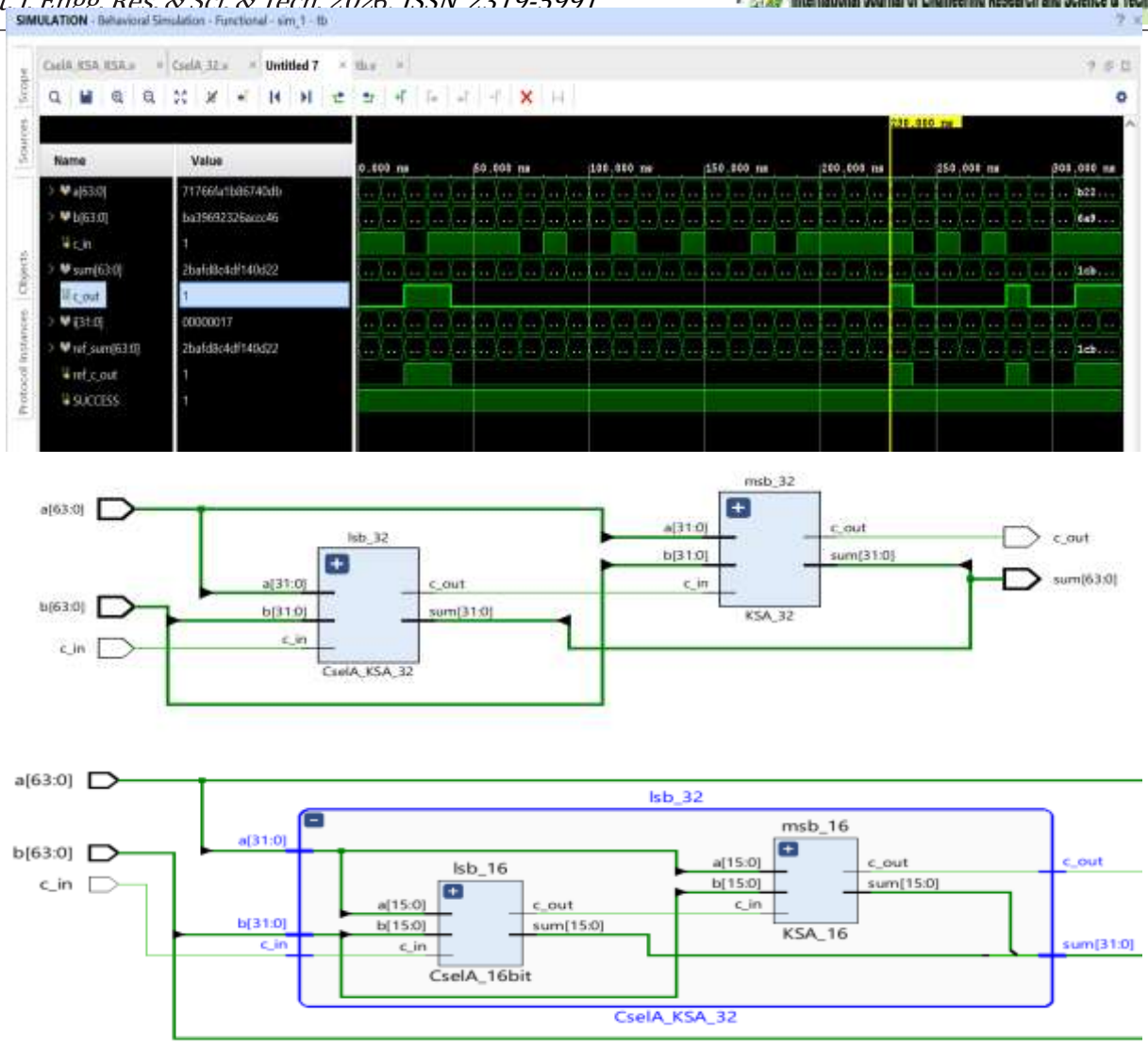
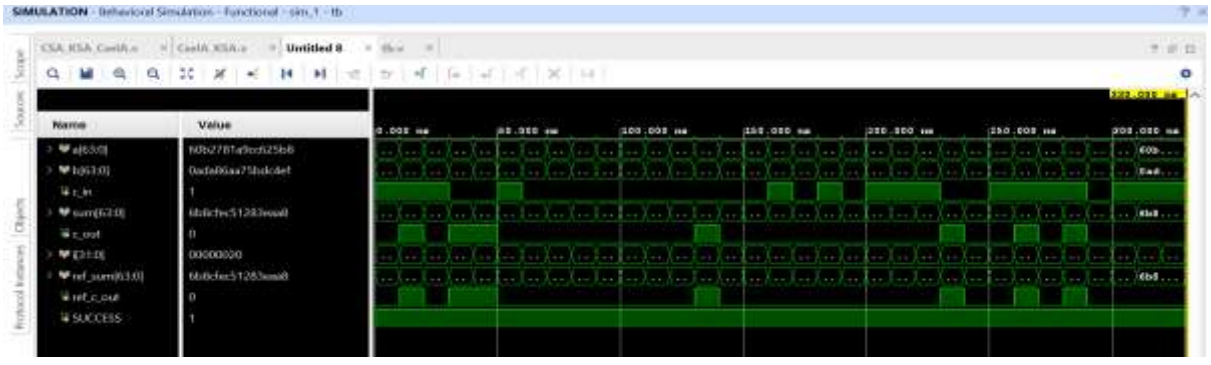


Fig 9 Simulation waveform of Hybrid Adder ((CSeLA(16-bit) + KSA (16-bit)) + KSA (32-bit))



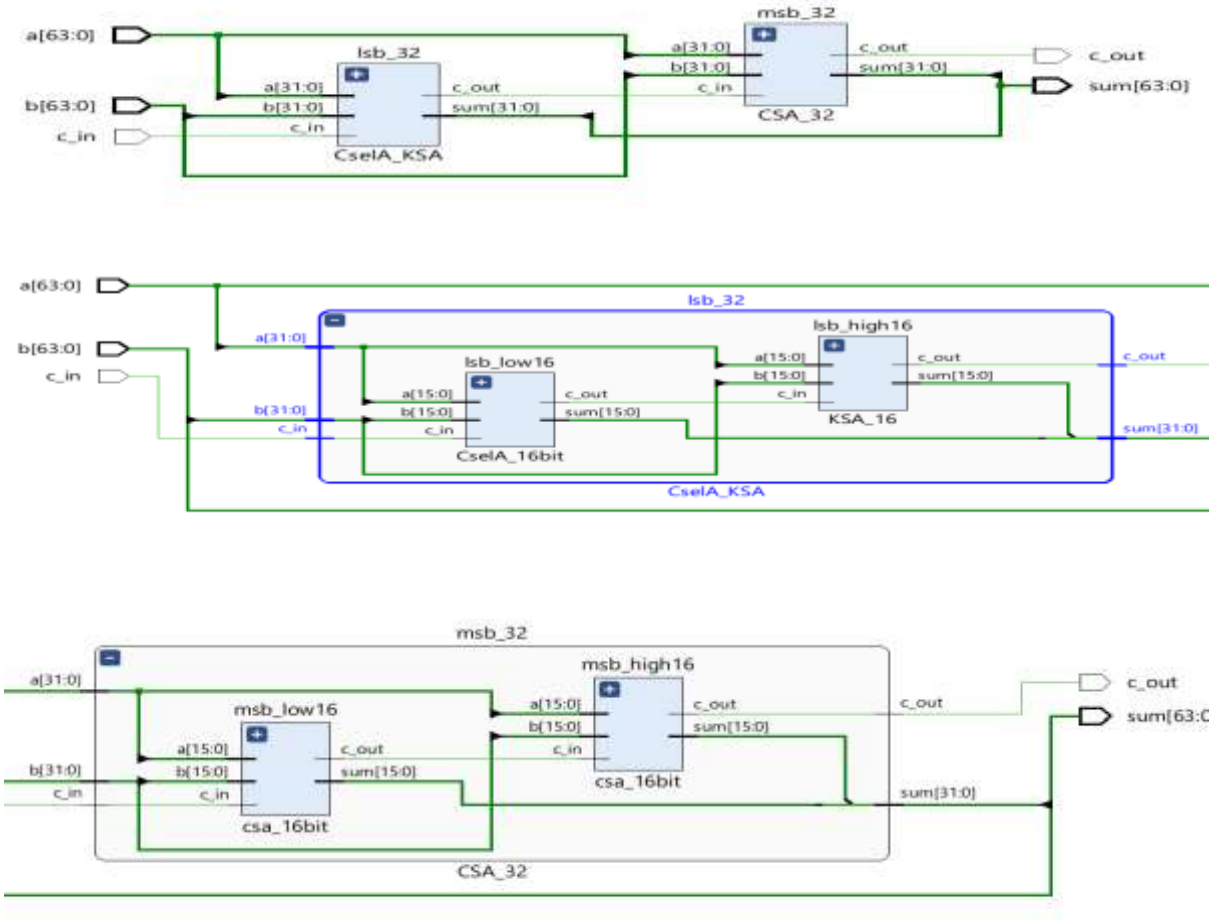
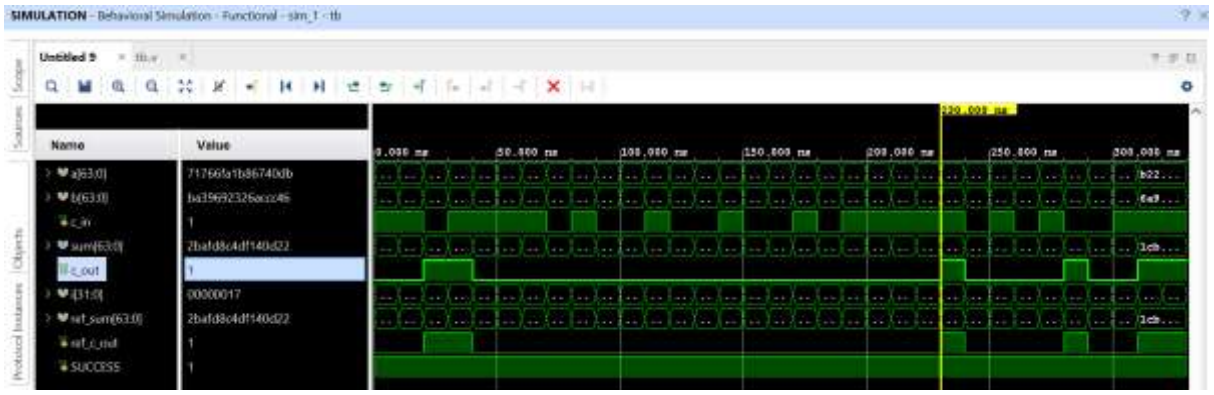
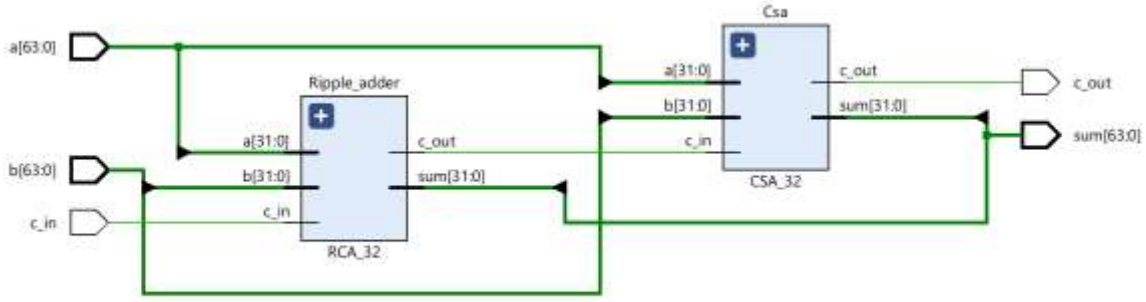
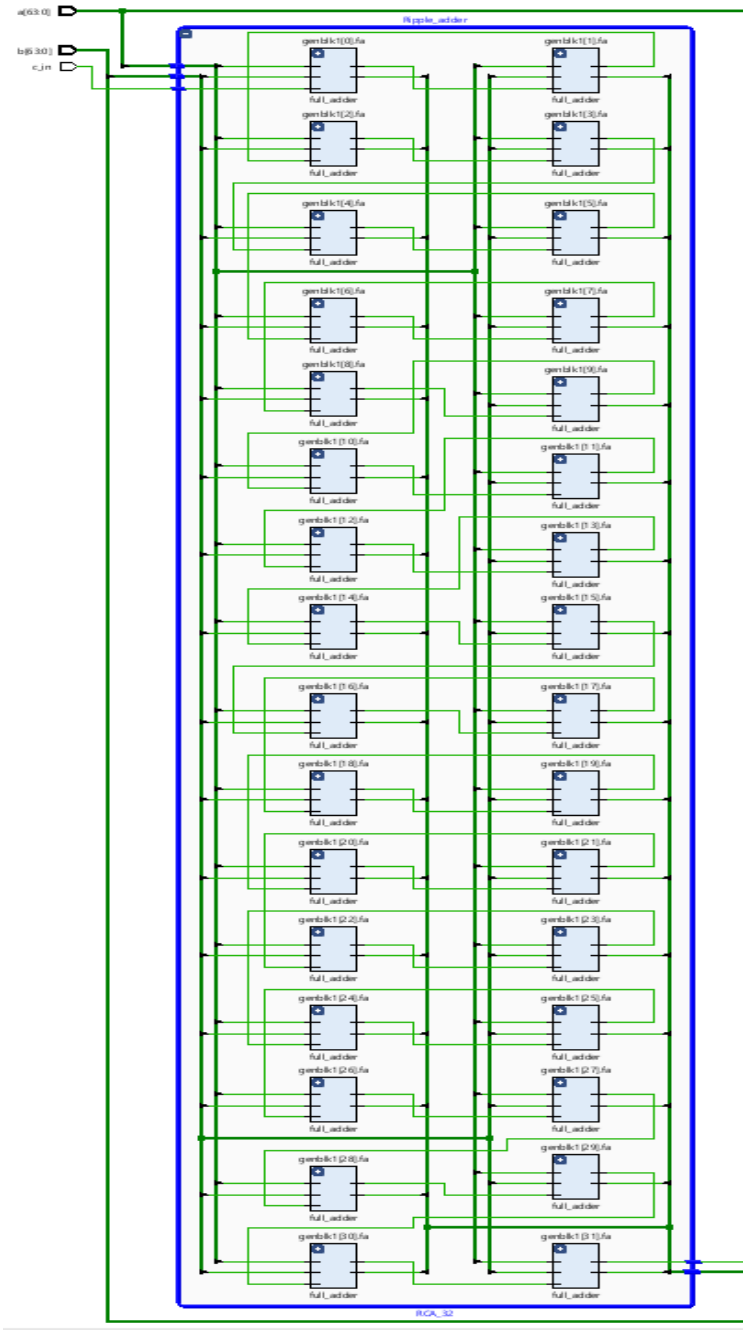


Fig 10 Simulation waveform of Hybrid Adder ((CSeLA(16-bit) + KSA (16-bit)) + CSA (32-bit))




Csa
a[63:0]
b[63:0]
c_in
Ripple_adder
a[31:0]
b[31:0]
c_in
c_out
sum[31:0]
RCA_32
a[31:0]
b[31:0]
c_in
c_out
sum[31:0]
CSA_32
c_out
sum[63:0]



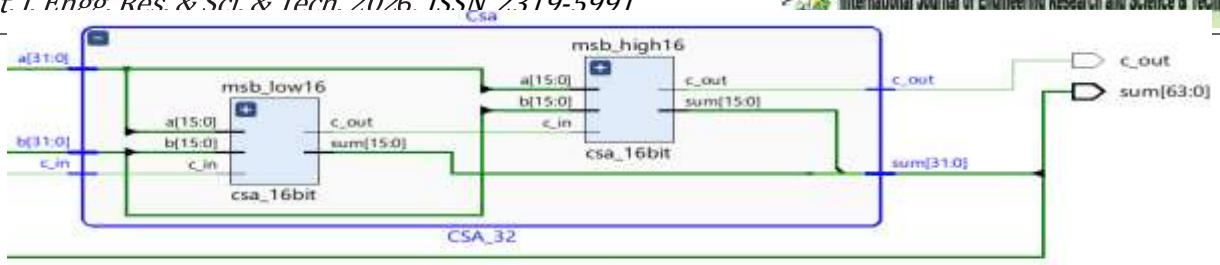


Fig 11 Simulation waveform of Hybrid Adder (RCA(32-bit) + CSA (32-bit))

Performance Analysis

This section presents the comparative performance analysis of the conventional and proposed hybrid 64-bit adder architectures implemented using Xilinx Vivado. The analysis is carried out based on the major VLSI performance parameters:

- Propagation Delay
- Power Consumption
- Area Utilization

The area utilization is evaluated in terms of FPGA resource usage such as:

- Lookup Tables (LUTs)
- Logic resources

The synthesis and timing reports generated by Vivado are used for evaluating the performance of each architecture. The comparative analysis helps identify the most efficient architecture in terms of the Power-Delay-Area (PDA) trade-off.

Propagation Delay Analysis

Propagation delay is one of the most important performance parameters in high-speed arithmetic circuits. It represents the total time required for the carry and sum outputs to stabilize after applying input signals.

The delay mainly depends on:

- Carry propagation mechanism
- Logic depth
- Prefix computation structure
- Interconnection complexity

The propagation delay of each architecture obtained from timing analysis is summarized in Table 1

Table 1 Propagation Delay Comparison of 64-bit Adders

Adder Architecture	Delay (ns)
RCA (64-bit)	18.42
CLA (64-bit)	11.36
CSKA (64-bit)	10.74
CSLA (64-bit)	9.81
KSA (64-bit)	6.92
RCA(32) + KSA(32)	7.84
CLA(32) + KSA(32)	6.48
CSLA(32) + KSA(32)	6.15
(CSLA16 + KSA16) + KSA32	5.87
(CSLA16 + KSA16) + CSKA32	6.41

The results show that the Ripple Carry Adder exhibits the highest delay due to sequential carry propagation from LSB to MSB. The Kogge–Stone Adder significantly reduces delay using parallel prefix carry computation. The proposed hybrid architectures further improve timing performance by combining fast

carry generation techniques with hierarchical computation structures.

Among all architectures, the hybrid structure: (CSLA16 + KSA16) + KSA32, shows the minimum propagation delay due to optimized parallel carry processing and reduced critical path length.

Table 2 Power Consumption Comparison of 64-bit Adders

Adder Architecture	Power (W)
RCA (64-bit)	0.184
CLA (64-bit)	0.231
CSKA (64-bit)	0.218
CSLA (64-bit)	0.264
KSA (64-bit)	0.312
RCA(32) + KSA(32)	0.246
CLA(32) + KSA(32)	0.271
CSLA(32) + KSA(32)	0.288
(CSLA16 + KSA16) + KSA32	0.301
(CSLA16 + KSA16) + CSKA32	0.267

The RCA consumes the least power because of its simple sequential structure and reduced hardware resources. The KSA consumes higher power due to extensive parallel prefix computation and increased routing activity.

The hybrid architectures show balanced power consumption while achieving improved delay performance. The results indicate that the proposed hybrid structures successfully balance speed and power requirements.

Among all implemented architectures, the hybrid: (CSLA 16 + KSA 16) + KSA 32 achieves the best overall delay performance, while (CSLA16 + KSA16) + CSKA32 provides a better balance between power, area, and speed. The analysis confirms that hierarchical hybrid architectures significantly improve overall VLSI arithmetic performance compared to conventional standalone adders.

V. CONCLUSION

This paper presented the design, implementation, and performance evaluation of several high-performance 64-bit hybrid adder architectures developed by combining conventional and advanced adder structures such as Ripple Carry Adders (RCA), Carry Lookahead Adders (CLA), Carry Select Adders (CSLA), Carry Skip Adders (CSKA), and Kogge-Stone Adders (KSA). The primary objective of the work was to achieve an optimized balance among propagation delay, power consumption, and area utilization, which are critical design parameters in modern VLSI systems.

The proposed architectures employed hierarchical partitioning and modular design methodologies to improve scalability and arithmetic efficiency. By integrating high-speed carry computation techniques in critical sections and area-efficient architectures in non-critical regions, the hybrid designs successfully reduced carry propagation delay while maintaining reasonable hardware complexity. The

use of Kogge-Stone Adders in higher-order sections significantly accelerated carry generation, whereas CLA, CSLA, RCA, and CSKA modules contributed to reducing area and power overheads.

All architectures were modeled using Verilog HDL and synthesized using Xilinx Vivado targeting FPGA platforms. Functional simulation and synthesis analysis were carried out to evaluate propagation delay, power consumption, LUT utilization, and overall Power-Delay-Area (PDA) performance. The experimental results demonstrated that the proposed hybrid architectures outperform conventional standalone adders by providing superior trade-offs between speed and hardware resources. In particular, multi-level hybrid structures incorporating Kogge-Stone Adders achieved the lowest critical path delays while maintaining acceptable power and area requirements.

The comparative analysis confirms that hybridization is an effective approach for developing high-performance arithmetic units suitable for modern processors, digital signal processing systems, communication hardware, and embedded computing platforms. The hierarchical design methodology further enhances flexibility and scalability, making the architectures suitable for large-bit-width arithmetic operations.

Future work may focus on integrating pipeline techniques, exploring alternative parallel prefix structures such as Brent-Kung and Han-Carlson Adders, implementing adaptive power management strategies, and evaluating ASIC implementations in advanced CMOS technologies. Further optimization using machine-learning-assisted design exploration can also be investigated to achieve improved Power-Delay-Area efficiency. Overall, the proposed 64-bit hybrid adder architectures provide a practical and efficient solution for next-generation high-speed arithmetic applications.

References

1) R. Zimmermann, "Binary Adder Architectures for Cell-Based VLSI and their Synthesis," ETH Zurich, Switzerland, 1998.
2) N. H. E. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th Edition, Pearson Education, 2011.
3) P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Transactions on Computers, Vol. C-22, No. 8, pp. 786–793, 1973.
4) O. J. Bedrij, "Carry-Select Adder," IRE Transactions on Electronic Computers, Vol. EC-11, No. 3, pp. 340–344, 1962.
5) B. Ramkumar and H. M. Kittur, "Low-Power and Area-Efficient Carry Select Adder," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 20, No. 2, pp. 371–375, 2012.
6) R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders," IEEE Transactions on Computers, Vol. C-31, No. 3, pp. 260–264, 1982.
7) T. Lynch and E. E. Swartzlander Jr., "A Spanning Tree Carry Lookahead Adder," IEEE Transactions on Computers, Vol. 41, No. 8, pp. 931–939, 1992.

8) D. Harris, "A Taxonomy of Parallel Prefix Networks," Proceedings of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, pp. 2213–2217, 2003.
9) M. D. Ercegovac and T. Lang, Digital Arithmetic, Morgan Kaufmann Publishers, San Francisco, USA, 2004.
10) J. M. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits: A Design Perspective, 2nd Edition, Prentice Hall, 2003.