Performance Improvement of 16-bit RISC Processor Using Vedic Multiplier and High-Speed Adders

Akshay Mhatre[1] Shivam Padwal[2], Mithilesh Kavade[3], Dr. Akhil Masurkar[4]

[1, 2, 4] Vidyalankar Institute of Technology, Mumbai-400037, India

[3] Embedded System Engineer, 1Accord, Mumbai-400070, India

Abstract. This project presents the design and performance enhancement of a 16-bit RISC processor through the integration of a Vedic multiplier and four high-performance parallel-prefix adder architectures: Kogge-Stone Adder, Brent-Kung Adder, Han-Carlson Adder, and Ladner-Fischer Adder. The primary objective is to improve the processor's computational efficiency, reduce latency, and optimize hardware resource utilization by replacing the conventional shift-and-add multiplication unit with a Vedic multiplier and by leveraging advanced adder designs to accelerate arithmetic operations. The Vedic multiplier, inspired by ancient Indian mathematical principles, enables high-speed multiplication through parallel computation, while the selected adders minimize carry propagation delay, collectively enhancing overall processor performance. The processor was implemented using Verilog HDL on Xilinx FPGA platform, including Spartan-7 (XC7S15FTGB196-1). Post synthesis and implementation analysis was conducted to evaluate performance, area, and power metrics before and after the integration of the Vedic multiplier and parallel-prefix adders. The results demonstrate substantial improvements in processing speed, arithmetic efficiency, and resource utilization, validating the effectiveness of combining Vedic arithmetic with advanced adder architectures in modern embedded processor design.

Keywords: 16-bit RISC, Vedic Multiplier, FPGA, Spartan-7, High Speed Adders.

1 Introduction

In modern digital systems, the demand for high-performance processors is rapidly increasing due to applications such as artificial intelligence accelerators, Internet of Things (IoT) devices, robotics, and real-time control systems. Arithmetic operations, particularly addition and multiplication, form the core of processor functionality and significantly influence execution speed, instruction throughput, latency, and power consumption. Conventional designs like shift-and-add multipliers and ripple carry adders, although simple, suffer from high latency, long critical paths, and inefficient resource utilization, which limit overall processor efficiency. To address these challenges, this project focuses on enhancing the performance of the Arithmetic Logic Unit (ALU) in a 16-bit RISC processor by reducing computation time, optimizing hardware usage, and improving speed and energy efficiency.

To overcome the limitations of traditional multipliers, the project integrates a Vedic multiplier based on the Urdhva Tiryakbhyam (Vertically and Crosswise) sutra. This approach enables parallel generation and accumulation of partial products, significantly reducing multiplication delay compared to conventional methods. Its inherent parallelism makes it highly suitable for implementation in FPGA and ASIC platforms, where performance and efficiency are critical. By incorporating this ancient mathematical concept into modern processor design, the project achieves faster arithmetic operations, reduced critical path delay, and demonstrates how classical techniques can effectively enhance contemporary digital systems.

2 Literature Review

The 16-bit RISC processor is designed with a focus on simplicity, efficiency, and modularity, following core RISC principles. It uses a load/store architecture where all computations are performed on register-stored data, and memory access is limited to load and store instructions, reducing complexity and improving execution speed. Based on the von Neumann architecture, it utilizes a single memory space for both instructions and data, ensuring a compact and simplified design. Each instruction is 16 bits long, enabling uniformity and efficient decoding.

The processor operates in a non-pipelined manner with three stages—Fetch, Decode, and Execute ensuring reliable and predictable performance. It includes 15 instructions covering logical operations, program control, data transfer, and HALT functionality. The architecture consists of seven general-purpose registers along with special-purpose registers like the Program Counter (PC), Instruction Register (IR), and flag registers. Designed for applications ranging from embedded systems to platforms like Raspberry Pi, it achieves a balance between performance and cost and is implemented using Verilog HDL.

Instruction Set					
Operation	**Opcode**				**Result**
HLT	0	0	0	0	Stop
ADD	0	0	0	1	$R_z = R_x + R_y$
SUB	0	0	1	0	$R_z = R_x$ - Ry
MUL	0	0	1	1	$R_z = R_x$ * Ry
AND	0	1	0	0	$R_z = R_x$ & Ry
OR	0	1	0	1	$R_z = R_x$ | Ry
XOR	0	1	1	0	R_z = Rx ^ Ry
NOT	0	1	1	1	R_z= ~R_x
SHL	1	0	0	0	Shift left R_x
SHR	1	0	0	1	Shift right R_x
INC	1	0	1	0	Increment R_x
DEC	1	0	1	1	Decrement R_x
MVI	1	1	0	0	Move Immediate
LOAD	1	1	0	1	Read from memory
STORE	1	1	1	0	Write to memory
JUMP	1	1	1	1	Jump to target address

Table 1: Sample Instruction Set for 16 Bit RISC Processor

The 16-bit RISC processor is built around a compact and efficient Instruction Set Architecture (ISA) that follows the core principles of Reduced Instruction Set Computing, comprising 16 fixed-length instructions, each 16 bits wide. These instructions are categorized into arithmetic, logical, data movement, and control operations, ensuring uniformity that simplifies decoding and provides consistent execution time for improved performance and predictability. Arithmetic instructions such as ADD, SUB, and MUL perform operations on register-stored data, with multiplication implemented either through a conventional shift-and-add method or an optimized Vedic multiplier based on performance requirements. Logical instructions including AND, OR, XOR, and NOT enable essential bitwise operations required for control and decision-making processes at the hardware level.

In addition, the processor supports shift and counter operations through SHL, SHR, INC, and DEC instructions, which are useful for bit manipulation, scaling, and loop control. Data transfer is handled using MVI (Move Immediate), LOAD, and STORE instructions, strictly following the load/store architecture where only specific instructions interact with memory, thereby improving execution efficiency. Program control is managed using the JUMP instruction to modify execution flow and the HLT instruction to terminate processing. Each instruction is executed through three sequential stages—fetch, decode, and execute where the instruction is retrieved, interpreted, and executed respectively, ensuring a structured, modular, and efficient operation suitable for embedded and real-time applications.

2.2. Processor Architecture

General Purpose Registers (GPRs): The processor includes a bank of general-purpose registers used to store data, intermediate results, and operands. These registers enable fast access to data compared to memory, enhancing the processor's speed. They also help in reducing the number of memory access operations, which is crucial for RISC efficiency.

Arithmetic Logic Unit (ALU): The ALU performs all arithmetic operations such as addition, subtraction, and logical operations like AND, OR, and NOT. It operates on the data fetched from registers and provides output back to the register or memory. The result of an ALU operation may also update status flags like Zero and Carry.

Control Unit: The control unit manages the execution of instructions by generating appropriate control signals. It decodes the instruction in the Instruction Register and directs the operation of other processor components accordingly. It ensures correct sequencing and synchronization of all operations.

Multiplexers (Mux1, Mux2): Multiplexers control the selection of input data paths within the processor. Mux1 selects between general-purpose register data and the Program Counter value to drive Bus1. Mux2 chooses from ALU output, Mux1 output, or memory data for Bus2, enabling flexible data flow.

Instruction Register (IR): The Instruction Register holds the instruction currently being decoded and executed. It receives data from memory during the instruction fetch cycle. This register ensures that the processor decodes and executes the correct instruction in sequence.

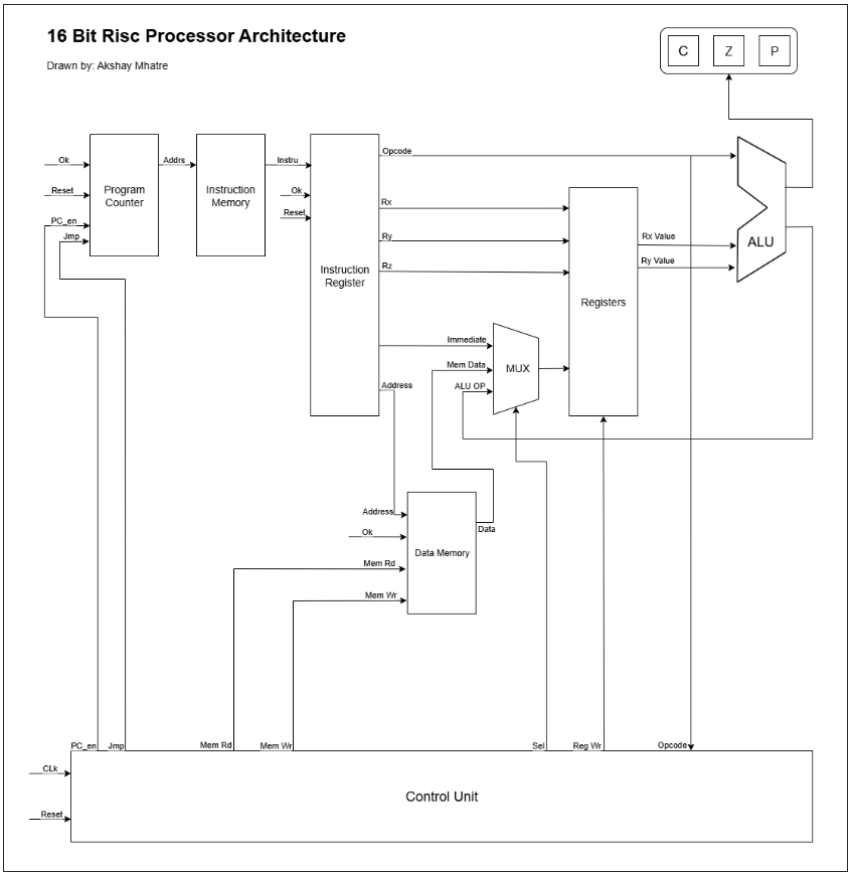

Figure 1: Architecture of 16 Bit RISC Processor

Program Counter (PC): The Program Counter stores the address of the next instruction to be fetched from memory. After fetching, it is usually incremented to point to the subsequent instruction. It plays a critical role in maintaining the program's control flow.

Memory Unit: The memory unit stores both the program instructions, and the data needed for execution. It is accessed by the processor using addresses generated during instruction execution. Memory operations include reading instructions/data and writing results back to specific locations.

Address Register (AddReg): The Address Register temporarily holds the memory address that needs to be accessed. This address could point to either an instruction or data in the memory. It acts as an interface between the control unit and the memory unit during read/write operations.

Operand Register (RegY): RegY is used to store one of the operands required for the ALU operation. The value in RegY is typically loaded from a general-purpose register or directly from memory. It ensures stable operand availability during ALU computation.

Flag Registers (RegZ, RegC): RegZ is the Zero Flag, set when the ALU result is zero, indicating equality or a null result. RegC is the Carry Flag, set when an arithmetic operation results in a carry out, especially in unsigned operations. These flags are essential for decision-making in conditional branch instructions.

Register	Function
PC	Program Counter - instruction address
IR	Instruction Register - current instruction
RegY	Operand Register - ALU input
RegZ	Flag register - zero flag
RegC	Flag register - carry flag
R0-R6	General-purpose registers

Table 2: Table of Register Function

2.3. Vedic Mathematics using Urdhva Tiryagbhyam Sutra

Vedic multipliers are high-speed arithmetic circuits derived from the principles of Vedic Mathematics, particularly utilizing the Urdhva Tiryagbhyam Sutra (Vertically and Crosswise). These multipliers perform parallel generation and accumulation of partial products, significantly enhancing the speed and efficiency of multiplication operations. The modular nature of their design allows scalability from smaller (2×2) units to larger (16×16) configurations, making them highly suitable for RISC processors, DSP applications, and embedded systems that demand high computational throughput and low latency.

Traditional multipliers like the shift-and-add or array-based architectures rely on sequential processing, resulting in long propagation delays. In contrast, Vedic multipliers perform simultaneous partial product generation and addition, enabling a parallel computation flow. This drastically reduces the critical path delay, improves instruction throughput, and enhances power efficiency.

Each higher-order Vedic multiplier is constructed hierarchically by combining smaller modules of the same type, making the design both modular and hardware-efficient. This hierarchical approach simplifies synthesis, testing, and FPGA implementation, as the same basic building block can be reused across different bit-widths.

2x2 Vedic Multiplier: The 2 x 2 Vedic multiplier operates using the ancient Vedic mathematics technique known as "Urdhva-Tiryak" (vertically and crosswise). This multiplier takes two 2-bit binary numbers as inputs (let's call them A and B) and efficiently computes their product using a sequence of simple arithmetic operations. Specifically, it involves multiplying the digits of A and B in a crosswise manner to generate partial products, followed by straightforward addition operations to combine these partial products and obtain the final result. This approach eliminates the need for complex multiplication circuits by leveraging parallelism and optimized arithmetic, making it ideal for hardware implementations where speed and simplicity are key considerations. The 2 x 2 Vedic multiplier thus exemplifies the effectiveness of ancient mathematical techniques in modern computational systems, offering a compact and efficient solution for basic multiplication tasks.

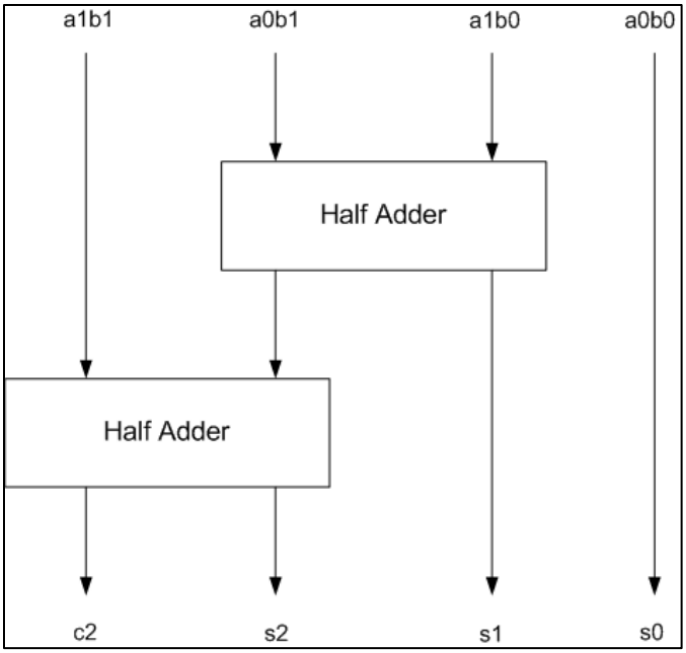

Figure 2: Block Diagram for 2x2 Vedic Multiplier

4x4 Vedic Multiplier: The 4 x 4 Vedic multiplier, based on Vedic mathematics principles, efficiently computes the product of two 4-bit binary numbers using a structured and parallel approach. This multiplier breaks down the multiplication process into several steps. First, it generates partial products by multiplying individual bits of the two input numbers in a crosswise manner. Each multiplication operation results in a partial product that corresponds to a specific position in the final product. Next, these partial products are aligned and added together using simple addition operations, considering their respective positional weights (similar to traditional multiplication but structured differently). Finally, carry propagation is handled to obtain the correct result. The 4 x 4 Vedic multiplier optimizes multiplication through parallel processing of partial products and streamlined addition steps, making it suitable for hardware implementations where speed, simplicity, and efficiency are crucial.

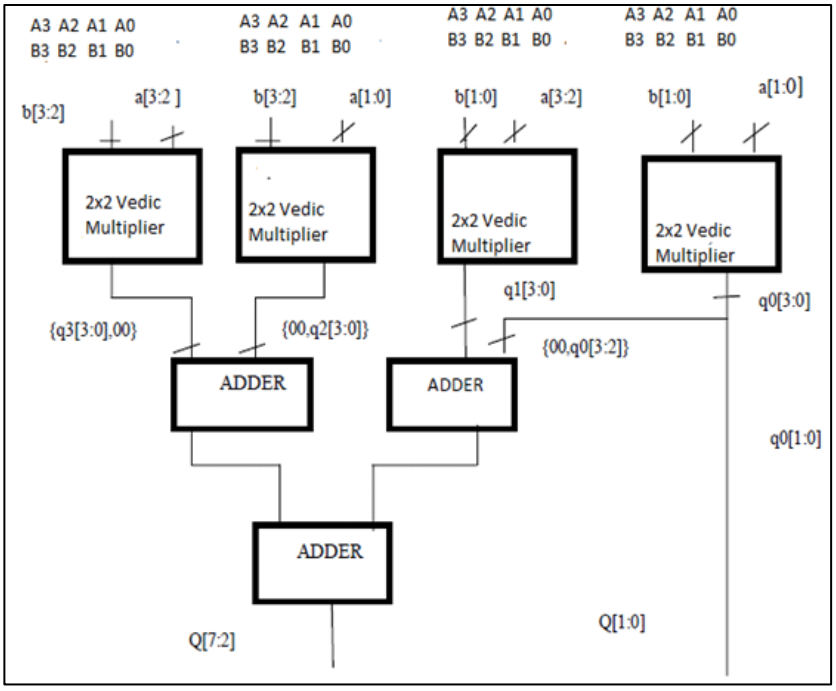

Figure 3: Block Diagram of 4x4 Vedic Multiplier

8x8 Vedic Multiplier: The 8 x 8 Vedic multiplier is implemented using four 4 x 4 vedic multipliers, the two 8-bit inputs are first split into two parts both 4-bit wide and these 4-bit inputs are then fed to the four 4 x 4 multipliers. The product gained from these multipliers is then given to three adders which add the data and give us a final 16-bit answer which is the product of two 8-bit inputs.

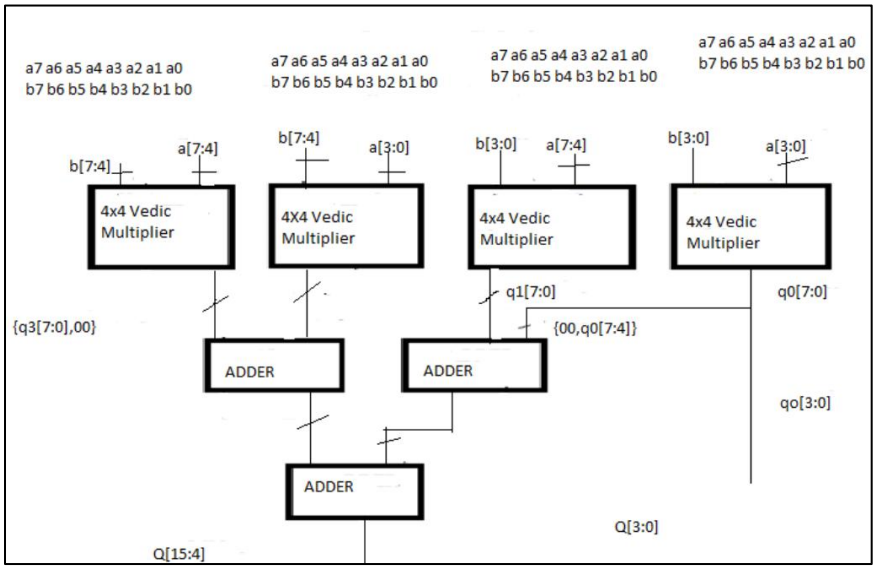

Figure 4: Block Diagram of 8x8 Vedic Multiplier

2.4. Adder & Their Implementation

In digital processor design, adders form the foundation of arithmetic operations within the Arithmetic Logic Unit (ALU). They are responsible not only for performing addition but also for supporting fundamental functions such as subtraction, multiplication, address calculation, and logical comparison. The efficiency of an adder directly influences the overall performance, speed, and power consumption of the processor. Since addition is one of the most frequently executed operations in instruction execution, optimizing adder performance becomes a crucial aspect of processor design.

The performance of an adder is determined primarily by its propagation delay, area utilization, and power efficiency. Traditional adders, such as the Ripple Carry Adder (RCA), are simple in structure but limited by long carry propagation chains, resulting in higher latency. To overcome this, modern architectures employ parallel-prefix adders such as Kogge-Stone, Brent-Kung, Han-Carlson, and Ladner-Fischer, which significantly reduce the carry propagation time through hierarchical computation of carry signals. These adders are implemented in the present 16-bit RISC processor to enhance the Arithmetic Logic Unit (ALU) performance and improve instruction throughput.

Types of High-Speed Adders Implemented:

1. **Kogge-Stone Adder (KSA):** The Kogge-Stone Adder is widely regarded as one of the fastest parallel-prefix adders. It uses a tree-based carry computation structure, generating carriers in a logarithmic number of stages. Although it provides the shortest delay, it requires higher wiring and logic resources, making it area intensive. The KSA is ideal for applications prioritizing speed over area, and in this project, it serves as a performance benchmark for the fastest achievable addition operation.
2. **Brent-Kung Adder (BKA):** The Brent-Kung Adder offers a balance between performance and hardware efficiency. It minimizes the number of logic gates and interconnections compared to Kogge-Stone, trading off slight delay for significant area and power savings. It is well-suited for embedded and FPGA-based processors where hardware resources are limited.
3. **Han-Carlson Adder (HCA):** The Han-Carlson Adder combines features of both Kogge-Stone and Brent-Kung architectures. It achieves a hybrid trade-off by maintaining low delay and moderate resource consumption. Due to its efficient balance between speed and area, it is often used in medium-performance processors and serves as an optimal choice for designs requiring consistent timing characteristics.
4. **Ladner-Fischer Adder (LFA):** The Ladner-Fischer Adder provides high-speed addition with a reduced number of logic levels compared to traditional adders. It offers flexibility in structure, allowing efficient implementation across varying bit-widths. The LFA's design minimizes fan-out and delay, making it suitable for large word-length additions in high-speed processors.

Implementation Approach

In this project, the 16-bit RISC processors ALU has been implemented with multiple adder configurations to analyze and compare their impact on timing, resource utilization, and power consumption. Each adder was designed using Verilog HDL, synthesized, and tested through FPGA implementation. The purpose was to identify the most effective adder that provides optimal performance for arithmetic-heavy operations within the processor.

The evaluation involved:

- **Timing Analysis:** Measuring the critical path delay and worst negative slack (WNS) to assess how quickly arithmetic operations complete.
- **Hardware Utilization:** Comparing Look-Up Table (LUT) and flip-flop usage across different adder designs to determine hardware efficiency.
- **Power Analysis:** Examining dynamic and static power consumption to evaluate energy efficiency under real-time operation.

The results demonstrate that parallel-prefix adders consistently outperform traditional adders in both speed and latency reduction, though with varying degrees of hardware utilization. Among them, hybrid designs like Han-Carlson and Ladner-Fischer achieve a superior balance between delay and logic efficiency, making them highly suitable for mid-range and high-speed embedded systems.

Key Factors in Adder Performance:

- **Carry Propagation Delay**: The time taken for a carry to ripple through stages is the primary factor affecting adder speed. Parallel-prefix adders minimize this delay using hierarchical computation.
- **Fan-Out and Gate Depth:** Efficient adders reduce fan-out and gate depth to minimize signal degradation and delay.
- **Power and Area Trade-Off:** Faster adders typically consume more power and occupy more logic resources. The optimal design depends on the system's power and area constraints.
- **Scalability:** A well-structured adder should maintain proportional performance as the bit-width increases, ensuring reliability for future processor extensions.

2.5. Multiplier Approach

The multiplier unit is a fundamental component of any processor's Arithmetic Logic Unit (ALU), responsible for executing complex arithmetic computations. The efficiency of this unit has a direct impact on the overall processor performance, particularly in applications that demand intensive mathematical operations, such as digital signal processing (DSP), cryptography, artificial intelligence (AI), and real-time embedded systems. In a 16-bit RISC processor, multiplication operations are among the most computation-heavy tasks, often becoming a bottleneck that limits instruction throughput and operating frequency. Therefore, choosing the right multiplier architecture plays a crucial role in achieving the desired balance between speed, area, and power efficiency.

Traditional multipliers, such as Shift-and-Add, Array, and Booth Multipliers, rely heavily on sequential processing.

- **In Shift-and-Add Multipliers**, each bit of the multiplier is processed one at a time. For every '1' encountered in the multiplier, a shifted version of the multiplicand is added to an accumulator. While this design is straightforward, it suffers from high latency due to the serial nature of operations, making it unsuitable for high-speed processing.
- **Array Multipliers** improve upon the shift-and-add method by employing combinational logic to generate and add partial products simultaneously. However, they still face challenges related to propagation delay and large hardware area, especially as bit-width increases. The carry propagation

through multiple adders results in a longer critical path, reducing the achievable clock frequency.

- **Booth's Algorithm** offers partial optimization by encoding the multiplier to reduce the number of additions. While it enhances performance for signed arithmetic, its complex control logic and sequential partial product addition increase design complexity and power consumption.

These conventional designs, though reliable, exhibit limitations in modern high-performance processors where parallelism, timing optimization, and low energy consumption are key requirements.

Need for an Optimized Multiplier:

As technology scales and computational demands increase, processors are expected to perform faster while consuming less power and occupying minimal silicon area. Achieving these objectives requires the adoption of parallel multiplication techniques that reduce the dependency on sequential operations and carry propagation delays.

An ideal multiplier should possess the following characteristics:

- **High speed** – Short critical path delay and parallel computation of partial products.
- **Low power consumption** – Minimal switching activity and reduced clock cycles.
- **Compact hardware footprint** – Efficient use of logic cells, LUTs, and flip-flops.
- **Scalability** – Capability to expand from smaller bit-widths (4-bit, 8-bit) to larger ones (16-bit, 32-bit) with minimal re-design effort.

These criteria form the foundation for the development of Vedic Multipliers, which outperform traditional architectures by leveraging mathematical parallelism.

Vedic Multiplier: A Parallel Computing Approach

The Vedic Multiplier, based on the Urdhva Tiryagbhyam Sutra, utilizes a parallel computation approach where all partial products are generated and added simultaneously instead of sequential processing. This significantly reduces propagation delay and overall multiplication time, resulting in higher operating frequencies and improved throughput. Such parallelism makes it highly efficient for high-speed arithmetic operations in modern digital systems.

The design follows a hierarchical modular structure, where smaller multipliers like 2×2 or 4×4 are combined to form higher-order units such as 8×8 or 16×16, enhancing scalability and simplifying implementation. Its regular and uniform architecture allows easy integration into RISC processors without major modifications. Additionally, reduced dependency on intermediate storage and control logic leads to lower power consumption and easier timing closure during FPGA synthesis.

Design and Implementation Considerations:

In hardware implementation, particularly on FPGA, several key parameters are evaluated to analyze multiplier performance.

- **Critical Path Delay** – The maximum propagation time between input and output signals. Lower delay directly improves operational frequency.

- **LUT and Slice Utilization** – Reflects hardware resource usage. Optimized designs aim for minimal resource consumption while maintaining speed.
- **Dynamic Power Consumption** – Assessed using switching activity analysis to ensure energy-efficient operation.
- **Setup and Hold Timing** – Determines the reliability and stability of the design at higher clock frequencies.

Through FPGA-based testing and synthesis, it was observed that Vedic Multipliers demonstrate significant improvements in timing characteristics, reducing computation delay and enhancing real-time performance. These improvements contribute directly to ALU's overall efficiency, enabling the processor to execute arithmetic-intensive instructions more effectively.

Comparative Analysis:

When compared with conventional multiplication approaches, the Vedic Multiplier achieves a:

- Reduction in propagation delay by enabling parallel computation.
- Improvement in throughput due to minimized sequential dependencies.
- Decrease in power consumption because of lower switching activity.
- Balanced area utilization, maintaining hardware efficiency even for larger bit-widths.

This trade-off between speed and resource optimization makes the Vedic Multiplier particularly suitable for high-performance embedded and low-power systems. It also serves as a foundation for integrating advanced arithmetic circuits into next-generation processors

3 Results and Discussion

3.1 Overview

The results and analysis in this chapter focus on the simulation, synthesis, and performance evaluation of various adder architectures integrated into the Arithmetic Logic Unit (ALU) of a 16-bit RISC processor. The primary objective is to examine how each adder design influences timing performance, operational speed, and computational efficiency across different arithmetic and logical operations. Five adder architectures Ripple Carry Adder (RCA), Kogge-Stone Adder (KSA), Brent-Kung Adder (BCA), Han-Carlson Adder (HCA), and Ladner-Fischer Adder (LFA) were implemented and analyzed, each offering distinct characteristics in terms of carry propagation, logic depth, and interconnection complexity, which directly impact overall processor performance.

The evaluation is primarily based on Worst Negative Slack (WNS), a critical timing parameter in VLSI design that indicates how effectively a circuit meets its timing constraints. Lower WNS values correspond to better performance, allowing operation at higher clock frequencies with

minimal timing violations. By comparing all adder architectures using this metric, the study aims to identify the most efficient design that achieves an optimal balance between speed, area utilization, fan-out, and hardware complexity, making it suitable for high-performance RISC processor implementations.

3.2. Numerical Results and Observations

The measured Worst Negative Slack (WNS) values for all adders under different ALU operations are tabulated below.

Operation	WNS (Worst Negative Slack)				
	RCA (ns)	KSA (ns)	BCA (ns)	HCA (ns)	LFA (ns)
ADD	2.145	2.136	2.331	2.325	2.145
SUB	2.161	2.145	2.264	2.252	2.34
MUL	2.438	2.481	2.45	2.405	2.505
AND	2.161	2.264	2.145	2.277	2.218
OR	2.161	2.145	2.145	2.029	2.221
NOT	2.267	2.253	2.325	1.924	1.947
XOR	2.145	2.239	2.145	1.861	2.254
NAND	2.161	2.253	2.152	2.35	2.105

Table 3: Table of (WNS) for all Adder under different ALU Operations

Adder used	WNS (ns)	WHS (ns)	LUT utilisation (%)	Power (W)
RCA	38.095	0.217	4.28	0.028
Kogge stone	38.189	0.197	4.38	0.028
Hans Carlson	38.212	0.197	4.24	0.028
Brent Cung	38.096	0.217	4.46	0.028
Ladner Fischer	38.034	0.197	4.3	0.028

Table 4: Comparison of different adders based on timing (WNS, WHS), LUT utilization, and power consumption obtained from FPGA implementation.

The results clearly show that the parallel prefix adders outperform conventional adders such as the RCA and BCA. For arithmetic operations, the Kogge-Stone Adder (KSA) achieved the best results for both addition and subtraction, with WNS values of 2.136 ns and 2.145 ns, respectively. This improvement is attributed to KSA's parallel carry generation mechanism, which minimizes delay by using a dense prefix structure. The Ladner-Fischer Adder (LFA) also exhibits strong timing performance (2.145 ns for ddition and 2.340 ns for subtraction), demonstrating its balanced structure and regular interconnection.

For multiplication, where multiple carry paths are generated simultaneously, the LFA achieves a WNS of 2.505 ns, followed closely by HCA (2.405 ns). This indicates that both adders can efficiently handle deep combinational paths, making them suitable for complex arithmetic units.

In logical operations, WNS values are more closely grouped because carry computation contributes less to delay. The Han-Carlson Adder (HCA) performs best for NOT (1.924 ns) and XOR (1.861 ns) operations, while the LFA maintains consistent performance across all logic operations, with values ranging between 1.947 ns and 2.254 ns.

These results prove that prefix adders, especially HCA and LFA, provide high-speed and reliable performance for both arithmetic and logical computations in the 16-bit RISC processor.

3.3. Comparative Discussion

From the results, it can be concluded that each adder architecture exhibits distinct strengths. The Ripple Carry Adder (RCA) is simple and occupies minimal area but has the highest propagation delay due to its sequential carry chain. The Kogge-Stone Adder (KSA) delivers the lowest delay for arithmetic operations owing to its minimal logic depth; however, it consumes more area and introduces routing complexity due to its high fan-out. The Brent-Kung Adder (BCA) achieves a reasonable balance between delay and area but is slower than KSA and LFA.

The Han-Carlson Adder (HCA) and Ladner-Fischer Adder (LFA), on the other hand, provide an excellent balance between speed, area, and complexity. The HCA, with its hybrid prefix structure, efficiently minimizes fan-out, leading to improved performance in logic operations. The LFA provides uniform timing behavior across all types of operations due to its regular prefix structure and logarithmic computation stages. In general, the LFA shows strong performance consistency and ease of implementation, making it ideal for VLSI and FPGA-based processor architectures where timing closure and resource utilization must be balanced effectively.

3.4. Graphical Analysis

To visually interpret the performance results, graphs were plotted to compare the WNS values of all five adders across arithmetic and logical operations.

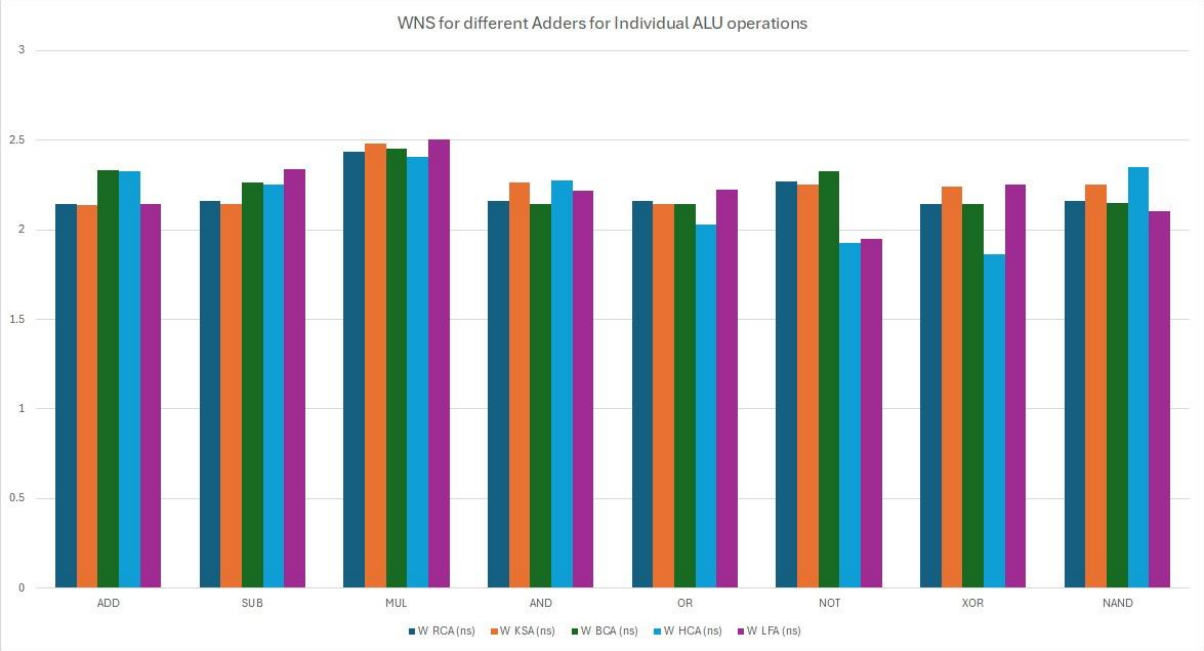

Figure 5: Comparison of (WNS) for Different Adder Architectures across Individual ALU Operations.

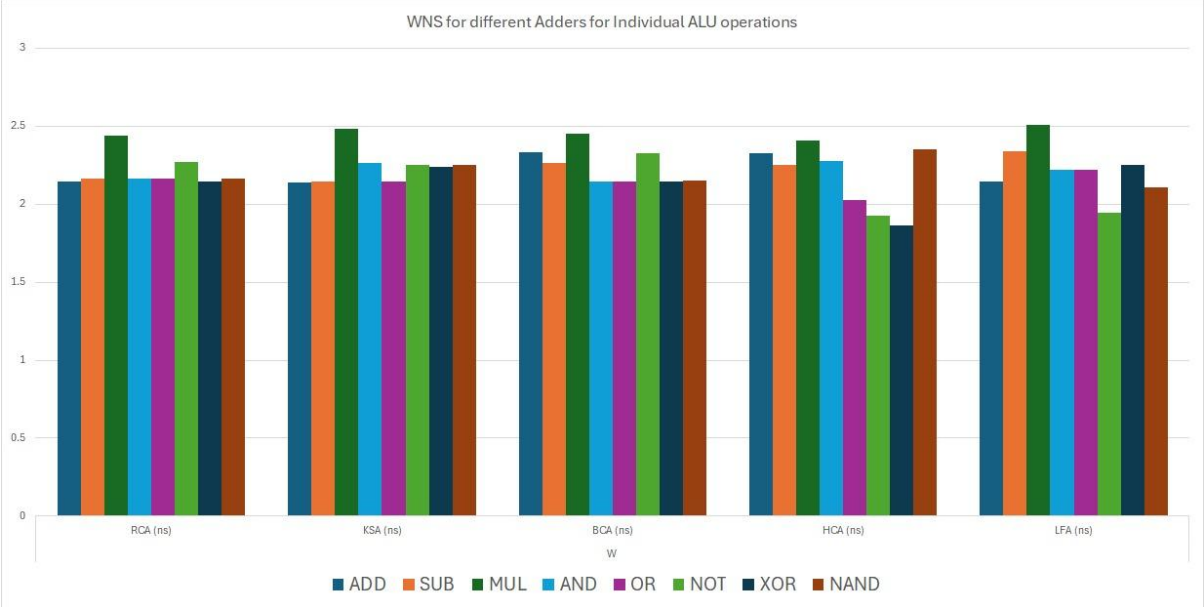

Figure 6: Comparison of (WNS) for Various ALU Operations across Different Adder Architectures.

In the arithmetic operations graph (ADD, SUB, MUL), the Kogge-Stone Adder consistently achieves the shortest bars, indicating the lowest Worst Negative Slack (WNS) values and the fastest performance in addition and subtraction operations. The Ladner-Fischer Adder follows closely, demonstrating stable and efficient performance, particularly in multiplication where it outperforms other architectures. In contrast, the Ripple Carry Adder and Brent-Kung Adder show comparatively taller bars, reflecting higher delays, though they maintain predictable and consistent behavior across operations.

In the logical operations graph (AND, OR, NOT, XOR, NAND), the Han-Carlson Adder stands out by achieving the lowest WNS values, especially in NOT and XOR operations, due to its optimized prefix structure and reduced fan-out. The Ladner-Fischer Adder once again exhibits uniform and balanced performance across all logic operations. The overall line graph of average WNS further highlights that Kogge-Stone and Ladner-Fischer Adders maintain consistently lower timing delays, while the Han-Carlson Adder shows a noticeable dip during logic operations, indicating its strength in that domain. Meanwhile, the Ripple Carry Adder remains the slowest with a higher and flatter trend due to sequential carry propagation. Overall, the analysis confirms that parallel prefix adders provide superior timing performance, with the Ladner-Fischer Adder offering the most balanced and reliable results for high-speed processor design.

4 Conclusion

This project presents a comprehensive study on the design, simulation, and synthesis of various adder architectures integrated into the Arithmetic Logic Unit (ALU) of a 16-bit RISC processor, with the objective of evaluating their impact on performance in terms of timing efficiency, speed, and hardware utilization. Five adders Ripple Carry Adder (RCA), Kogge-Stone Adder (KSA), Brent-Kung Adder (BCA), Han-Carlson Adder (HCA), and Ladner-Fischer Adder (LFA) implemented and analyzed under identical conditions using Worst Negative Slack (WNS) as the key performance metric to assess timing constraints. The results showed that parallel prefix adders significantly outperform conventional designs, with the Kogge-Stone Adder achieving the best timing performance at the cost of higher area and interconnections, while the Han-Carlson Adder offered efficient logic performance due to its hybrid structure. The Ladner-Fischer Adder demonstrated a well-balanced performance across all operations, combining high speed with moderate area and reduced wiring complexity. Overall, the study concludes that the Ladner-Fischer Adder is the most efficient and practical choice for high-performance RISC processors due to its regular structure, low fan-out, logarithmic carry computation, and scalability, highlighting the importance of selecting an optimal adder architecture to balance timing, hardware efficiency, and design complexity in modern digital systems.

5 Future Scope

The current work provides a comprehensive comparison of various adder architectures within the Arithmetic Logic Unit (ALU) of a 16-bit RISC processor, highlighting their impact on timing, area, and efficiency; however, several opportunities remain for future enhancement. A key direction is the implementation on ASIC platforms using tools such as Cadence Virtuoso, which would enable transistor-level analysis for more accurate evaluation of power consumption, propagation delay, and silicon area, along with deeper insights into physical design constraints and integration with Vedic multipliers. Another important extension involves scaling the architecture to higher bit-width processors such as 32-bit or 64-bit systems to study the effects of increased complexity on delay, routing, and area utilization, helping identify the most suitable adder for high-performance or low-power applications. Furthermore, ASIC-level post-layout comparison and optimization can provide precise analysis of delay, power dissipation, area, and speed-power trade-offs, ultimately contributing to the development of optimized, energy-efficient, and high-speed arithmetic units for next-generation processor architectures.

6 Reference

1. M. D, D. S, S. RS, S. P. M, S. K and R. JS, "Energy-Efficient Low-Power Three-Operand Approximate Brent-Kung Adder for VLSI-Based Image Processing," 2025 Second International Conference on Cognitive Robotics and Intelligent Systems (ICC - ROBINS), Coimbatore, India, 2025, pp. 79-84, doi: 10.1109/ICC-ROBINS64345.2025.11086111. keywords: {Surveillance;Very large-scale integration;Gray-scale;Real-time systems; Hardware; Delays; Logic; Adders;Image fusion;Standards;Brent-Kung adder;approximate computing;parallel prefix adder; low power;DSP;image fusion;VLSI},
2. A. P.R, G. I and S. S. R, "Wallace Tree Multiplier Using Reversible Logic With Brent-Kung Adder," 2025 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI), Chennai, India, 2025, pp. 1-6, doi: 10.1109/RAEEUCCI63961.2025.11048246. keywords: {Measurement;Power demand;High performance computing;Computer architecture;Logic gates;Delays;System-on-chip;Logic;Adders;Standards;Wallace tree multiplier;Kogge-Stone adders;brent-kung adder;Compressor;Reverse logic},
3. T. Krishnan, P. Anguraj, M. P, S. S and V. A, "High Performance Reverse Converter Design Using Modified Brent-Kung EAC Adder," 2025 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI), Chennai, India, 2025, pp. 1-6, doi: 10.1109/RAEEUCCI63961.2025.11048213. keywords: {Multiplexing;Simulation;Delays;Adders;Computational intelligence;Brent-Kung Adder (BK);Chinese Remainder Theorem (CRT);End Around Carry (EAC);Parallel prefix adder;Power Delay Product (PDP);Residue Number System (RNS)},
4. K. Karthigadevi, M. Sakthimohan, S. Tamilselvan, B. Thiyagarajan, G. Elizabeth Rani and V. Sanjeevi Kumar, "Enhanced 16 × 16 Dadda Multiplier Using Kung-Brent Adder for Superior Parallel Processing Performance," 2024 8th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2024, pp. 80-85, doi: 10.1109/ICECA63461.2024.10801100. keywords: {Technological innovation;Power

demand;Multicore processing;Scalability;Parallel processing;Throughput;Delays;Low latency communication;Adders;Arithmetic;Dadda Multiplier;Parallel Processing;Kung-Brent Adder;Partial Products},

5. B. Penumutchi and N. Kothapalli, "High-End Approximate Multiplier Using Brent Kung Razor FlipFlop," 2024 5th International Conference for Emerging Technology (INCET), Belgaum, India, 2024, pp. 1-6, doi: 10.1109/INCET61516.2024.10592958. keywords: {Energy consumption;Delays;Adders;Flip-flops;Approximate multiplier;carry mask-able adder (CMA);carry propagation adder (CPA);partial product reduction (PPR)},
6. T. Prudhvi, D. S. Reddy, M. Sarada and K. Satish, "Approximate Brent Kung Adder For Image Processing Applications," 2023 International Conference on Intelligent Technologies for Sustainable Electric and Communications Systems (iTech SECOM), Coimbatore, India, 2023, pp. 480-484, doi: 10.1109/iTechSECOM59882.2023.10435122. keywords: {Digital systems;Image processing;Software;Hardware;Low latency communication;Adders;Matlab;Brent-Kung (BK);Approximate Parallel Prefix Adder (AxPPA);Approximate Brent-Kung (AxBK)},
7. K. A. Rao, M. Pandit and N. Purohit, "Efficient 3-parallel polyphase odd length FIR filter using Brent Kung adder and Booth multiplier for VLSI applications," 2022 IEEE 9th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Prayagraj, India, 2022, pp. 1-5, doi: 10.1109/UPCON56432.2022.9986374. keywords: {Power demand;Program processors;Finite impulse response filters;Digital signal processing;Very large scale integration;Delays;Power dissipation;Digital signal processing;Parallel FIR filter;Booth multiplier;Brent kung adder},
8. A. Sai Kumar, U. Siddhesh, N. Sai kiran and K. Bhavitha, "Design of High Speed 8-bit Vedic Multiplier using Brent Kung Adders," 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2022, pp. 1-5, doi: 10.1109/ICCCNT54827.2022.9984591. keywords: {Signal processing algorithms;Digital signal processing;Logic gates;Software;Mathematics;Hardware design languages;Adders;Vedic multiplier;Brent Kung adder;array multiplier;Wallace tree multiplier;Urdhva-Tiryagbhyam sutra},
9. V. M B, P. Akkamanchi and V. Uttarkar, "Low Power High Speed Brent Kung Adder Using SPST," 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2022, pp. 1-6, doi: 10.1109/GCAT55367.2022.9971848. keywords: {Power demand;Machine learning algorithms;Delays;Digital filters;Adders;Spurious Power Suppression Technique (SPST);Brent-Kung Adder (BKA);Parallel Prefix Adder (PPA);Power;Area},
10. Anju. "Performance Comparison of Vedic Multiplier and Booth Multiplier." International Journal of Electronics and Communication Engineering & Technology, vol. 2, no. 5, June 2013, pp. 336-339. BEIESP, ISSN: 2249-054X. [Retrieval Number: E1846062513/2013©BEIESP]
11. K. A and C. H. Gowda, "An Efficient 32-bit Ladner Fischer Adder derived using Han-Carlson," 2021 IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC), Tumkur, Karnataka, India, 2021, pp. 1-4, doi: 10.1109/ICMNWC52512.2021.9688464. keywords: {Wireless communication;Microprocessors;Conferences;Logic gates;Very large scale integration;Delays;Table lookup;Efficient Ladner Fischer Adder (ELFA);Parallel prefix adder(PPA);Black cell;Gray cell;Ripple carry adder (RCA)},
12. A. S. Reddy, B. K. Rehman, P. Anuradha, K. V. Siva Reddy, M. Basha and M. A. Nasar, "Designing a High-Speed, Low-Area Three-Operand Binary Adder using Kogge-Stone, Han-Carlson, and

Ladner-Fischer VLSI Architectures," 2025 4th International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Tirupur, India, 2025, pp. 98-104, doi: 10.1109/ICIMIA67127.2025.11200640. keywords: {System performance;Signal processing algorithms;Computer architecture;Very large scale integration;Throughput;Hardware;Delays;Cryptography;Adders;Arithmetic;Three-Operand Binary Adder;Modular Arithmetic;Cryptographic Applications;Kogge-Stone Adder;Han-Carlson Adder;Ladner-Fischer Adder;VLSI Architecture},

13. R. Pandey, "Implementation of Approximate Adder circuit of Ladner Fischer Adder (16 bit)," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020, pp. 31-34, doi: 10.1109/ICECA49313.2020.9297540. keywords: {Adders;Logic gates;Conferences;Layout;Very large scale integration;Tools;Libraries;approximate adder;parallel prefix adder;style;architectural approximation;Low power;ASIC flow},
14. S. Akash, M. Ajeeth and N. Radha, "An Efficient Implementation of FIR Filter Using High Speed Adders For Signal Processing Applications," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 1047-1051, doi: 10.1109/ICIRCA48905.2020.9183114. keywords: {Adders;Finite impulse response filters;Delays;Filtering theory;Conferences;Hardware;Propagation delay;Finite Impulse Response;Parallel Prefix Adders etc},
15. K. Papachatzopoulos and V. Paliouras, "Maximum Delay Models for Parallel-Prefix Adders in the Presence of Threshold Voltage Variations," 2020 IEEE 27th Symposium on Computer Arithmetic (ARITH), Portland, OR, USA, 2020, pp. 88-95, doi: 10.1109/ARITH48897.2020.00021. keywords: {Delays;Adders;SPICE;Threshold voltage;Gaussian distribution;Monte Carlo methods;parallel-prefix adders;critical path;threshold voltage variations;variability;delay yield},
16. B. Shyamsundar, R. Venkateswara, H. P. Kumar and H. S. Kumar, "Developing a fast, energy-efficient carry skip adder with configurable delay extension utilizing the Han Carlson adder," 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 2024, pp. 1-5, doi: 10.1109/ICCCNT61001.2024.10724590. keywords: {Power demand;Simulation;Computer architecture;Voltage;Very large scale integration;Logic gates;Energy efficiency;Delays;Adders;Propagation delay;High speed;carry skip adder;energy-efficient;variable latency extension;Han Carlson adder;parallel adder;propagation delays;efficiency;variable latency;adder architectures},
17. B. Shyamsundar, R. Venkateswara, H. P. Kumar and H. S. Kumar, "Development of a Fast and Energy-Efficient Carrier Bypass Adder with Tunable Delay Extension Using the Han Carlson Adder," 2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS), Bangalore, India, 2024, pp. 1-4, doi: 10.1109/ICITEICS61368.2024.10625150. keywords: {Power demand;Simulation;Computer architecture;Voltage;Very large scale integration;Logic gates;Energy efficiency;High speed;carry skip adder;energy-efficient;variable latency extension;Han Carlson adder;parallel adder;propagation delays;efficiency;variable latency;adder architectures},
18. N. Varshney and G. Arya, "Design and Execution of Enhanced Carry Increment Adder using Han-Carlson and Kogge-Stone adder Technique : Han-Carlson and Kogge-Stone adder is used to increase speed of adder circuitry," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2019, pp. 1163-1170,

doi: 10.1109/ICECA.2019.8822194. keywords: {Adders;Delays;Conferences;Hardware design languages;Logic gates;Aerospace electronics;Very large scale integration;CIA;CLA;RCA;HCA;KSA;delay;Verilog HDL},

19. Kumawat, Jyoti, and Sunil Sharma. "A 8x8 bit multiplier using Vedic Mathematics." International Journal of Engineering and Technical Research (IJETR), vol. 2, no. 3, March 2014, ISSN: 2321-0869
20. P. Paliwal and J. B. Sharma, "Efficient FPGA Implementation Architecture of Fast FIR Algorithm Using Han-Carlson Adder Based Vedic Multiplier," 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2018, pp. 643-646, doi: 10.1109/ICIRCA.2018.8597432. keywords: {Finite impulse response filters;Adders;Computer architecture;Hardware;Signal processing algorithms;Delays;FFA;Han-Carlson adder;Urdhva-Triyakbhayam sutra},