ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ, ಬೆಳಗಾವಿ

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELAGAVI – 590 018

A PROJECT REPORT

ON

"32-BIT MAC UNIT WITH VEDIC MULTIPLIER AND CARRY SAVE ADDER (CSA) DESIGN AND COMPARATIVE ANALYSIS"

Submitted in partial fulfillment of requirement for the award of the degree

BACHELOR OF ENGINEERING

IN

ELECTRONICS & COMMUNICATION ENGINEERING

Submitted by

M SUSHMA	**4KV21EC029**
NITHIN P G	**4KV21EC036**
NOOTHAN GANESH P	**4KV21EC037**
RAJESH P K	**4KV21EC041**

Under the guidance of

Dr. SAVITHA M. **B.E., M.Tech., Ph.D., MISTE**

Professor

Department of E&C Engg.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

K.V.G. COLLEGE OF ENGINEERING

KURUNJIBAGH, SULLIA, D.K.-574327

2024-2025

KVG
COLLEGE OF ENGINEERING

Kurunjibagh, Sullia, Dakshina Kannada, Karnataka, India – 574327
(Approved by AICTE New Delhi, Affiliated to VTU Belagavi)
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the project work entitled "32-BIT MAC UNIT WITH VEDIC MULTIPLIER AND CARRY SAVE ADDER (CSA) DESIGN AND COMPARATIVE ANALYSIS" is a bonafide work carried out by

NITHIN P G 4KV21EC036

in partial fulfillment for the award of degree of "BACHELOR OF ENGINEERING" in ELECTRONICS AND COMMUNICATION from the VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI during the year 2024-2025. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The dissertation report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering Degree.

Dr. SAVITHA M
B.E., M.Tech, Ph.D, MISTE
Professor,
Project Guide

Mr. LOKESH P C
B.E., M.Tech
Assistant Professor,
UG Project Co-ordinator

Dr. KUSUMADHARA S
B.E., M.Tech., Ph.D., MISTE
Professor & HOD

Dr. SURESHA V
B.E.,M.Tech.,Ph.D.,MISTE.,MIETE
Principal

Name of the Examiners	Signature with Date
1. Dr. Kusumadhara S	29/5/25
2. Mrs. Nisha G.R	29/5/25

KVG
COLLEGE OF ENGINEERING
Kurunjibagh, Sullia, Dakshina Kannada, Karnataka, India - 574327
(Approved by AICTE New Delhi, Affiliated to VTU Belagavi)
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DECLARATION

I, **NITHIN P G (4KV21EC036),** hereby declare that the project work entitled **"32-BIT MAC UNIT WITH VEDIC MULTIPLIER AND CARRY SAVE ADDER (CSA) DESIGN AND COMPARATIVE ANALYSIS"** has been independently carried out by us under the guidance of **Dr. SAVITHA M,** Professor, Department of Electronics & Communication Engineering, KVG College of Engineering, Sullia, in partial fulfillment of the requirements of the degree of **Bachelor of Engineering** in **Electronics and Communication** of Visvesvaraya Technological University, Belagavi. I further declare that I have not submitted this report either in part of in full to any other university for the reward of any degree.

NITHIN P G **4KV21EC036**

Date:

Place: Sullia

ACKNOWLEDGEMENT

The completion of this project work is due to the experience, efforts and inspiration of many people. We would like to express our gratitude to all the people without whom it would have difficult to carry out this project.

We are extremely grateful to our Project Guide **Dr. SAVITHA M.** Professor, Department of Electronics & Communication Engineering, for her excellent guidance and supervision which enabled us to look for different techniques and apply innovative ideas.

We are thankful to project Co-ordinator **Mr. LOKESH P C,** Asst. Professor, Department of Electronics & Communication Engineering for his support and guidance.

We are very much thankful to his for the valuable advice he had given us. With his guidance we gained the knowledge needed to carry out this project work. We wish to express our heartfelt and sincere thanks to **Dr. KUSUMADHARA S.,** Professor and Head of the Department of Electronics and Communication Engineering, who always empowered our confidence and helped us in completing the project with success.

We are extremely grateful to our beloved Principal **Dr. SURESHA V**, Principal, K.V.G. College of Engineering, Sullia for providing constant encouragement and support throughout the course of the project.

Our greatest thanks to **Dr. RENUKA PRASAD K V,** Chairman and **Dr. JYOTHI R. PRASAD,** Secretary, Committee-B Academy of Liberal Education, Sullia and the management for providing us the opportunity to carry out our project work.

We are deeply indebted to the architect of modern Sullia and Founder President of Academy of Liberal Education Late **Dr. KURUNJI VENKATRAMANA GOWDA**, for providing an environment with all the facilities that helped us in successfully carrying out the project.

Our sincere thanks to all teaching and non teaching staff members of our institution for their constant support, valuable suggestions and co-operation.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This project presents the design and implementation of a high-speed, area-efficient 32-bit Multiply–Accumulate (MAC) unit leveraging the Urdhva Tiryakbhyam sutra of Vedic mathematics and a Carry Save Adder (CSA). The architecture is constructed using a fully hierarchical and modular approach in Verilog HDL, scaling the Vedic multiplier from 2×2 to 32×32 using recursive decomposition. The design incorporates a 64-bit CSA for multi-operand addition and a synchronous accumulator to enable pipelined accumulation over clock cycles.

To validate the design, functional simulation was performed using Questa Simulator, and synthesis was conducted on both FPGA (Xilinx ISE, Intel Quartus Prime) and ASIC (Synopsys Design Vision) platforms. The results demonstrate substantial improvements in performance metrics compared to conventional MAC designs. Comparative analysis with Booth-Wallace and Vedic-KoggeStone architectures confirms the superiority of the proposed design in speed, power, and area. Its modular structure, scalability, and synthesis-readiness. This work showcases the effective integration of traditional Vedic algorithms with modern VLSI design methodologies to meet the growing demands of high-performance computing

CHAPTER 1

INTRODUCTION

In the digital era, the performance of computational systems is critical to advancements in various domains such as embedded systems, consumer electronics, and high-performance computing. One of the key arithmetic components in such systems is the Multiply Accumulate (MAC) unit. It is a core element in a wide range of signal processing tasks, including but not limited to digital filtering, convolution operations, fast Fourier transforms (FFTs), and computations in neural networks. These operations are often computationally intensive and rely heavily on fast and efficient multiplication and accumulation.

Digital Signal Processing (DSP) applications, such as those involving FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) filters, rely on real-time processing capabilities. Although IIR filters generally require fewer computational resources compared to FIR filters, both place stringent demands on the MAC unit in terms of low latency and minimal power consumption. As a result, optimizing MAC performance is pivotal for improving the throughput and energy efficiency of the entire system.

Traditional multiplier and adder designs like Booth multipliers, Wallace trees, ripple-carry adders, and carry-lookahead adders often fall short in terms of speed and scalability. These conventional architectures suffer from long propagation delays, especially as operand bit-widths increase. To overcome these limitations, recent advancements have explored alternative techniques rooted in ancient Indian Vedic mathematics.

The Urdhva Tiryakbhyām Sutratranslated as "Vertically and Crosswise" offers a parallel partial product generation technique that is highly conducive to high-speed multiplication. Its ability to produce partial products simultaneously significantly reduces the depth of logic required and facilitates faster arithmetic operations. When implemented hierarchically, it provides a structured and scalable approach to constructing multipliers of higher bit-widths.

Complementing this, the Carry Save Adder (CSA) is employed to perform multi-operand additions without the need for immediate carry propagation. Unlike ripple-carry or carry-lookahead adders that propagate carry through each bit sequentially, the CSA defers carry propagation, thereby reducing delay and making the addition stage more efficient.

The combination of a Vedic multiplier with a CSA not only enhances computational speed but also enables better pipelining and modular design. These advantages are particularly beneficial for Application-Specific Integrated Circuit (ASIC) implementations. When synthesized and laid out using advanced Electronic Design Automation (EDA) tools like, this architectural approach demonstrates superior performance in terms of area, timing, and power compared to conventional MAC designs.

A high-performance MAC unit designed using these principles can play a transformative role in modern embedded processors, mobile computing platforms, and ASICs, where space, power, and timing constraints are often stringent.

1.1 Motivation

The motivation behind this project stems from the need to overcome the limitations of traditional MAC architectures in terms of speed, scalability, and power efficiency. The exploration of ancient Indian Vedic mathematics, particularly the Urdhva Tiryakbhyam Sutra, offers a novel and efficient solution for multiplication. Coupled with the Carry Save Adder, which minimizes addition delays, this design aims to create a high-performance, compact MAC unit suitable for modern VLSI applications. The blend of historical techniques and current technology presents a unique opportunity to address contemporary hardware challenges.

1.2 Problem Statement

As modern digital systems increasingly demand high-speed, low-latency, and energy-efficient computation particularly in fields like real-time video processing, embedded systems, edge computing, and AI inference there is a critical need to optimize the performance of fundamental arithmetic units such as Multiply–Accumulate (MAC) blocks. Traditional MAC designs, typically built using Booth multipliers, Wallace trees, and ripple-carry or carry-look ahead adders, introduce significant design complexity, carry propagation delays, and scalability limitations especially at higher bit-widths like 32 bits.

This project addresses these limitations by exploring an alternative MAC architecture that integrates the Urdhva Tiryakbhyam sutra from Vedic mathematics for efficient and parallel multiplication, and a Carry Save Adder (CSA) for fast, multi-operand addition without carry propagation. The goal is to develop a compact, high-speed, and synthesis-ready 32-bit MAC unit that is both area- and power-efficient.

1.3 Objectives

The key objective of this project is to design and verify a 32-bit MultiplyAccumulate (MAC) unit that is optimized for high-performance embedded and signal processing applications. The specific objectives are:

- To design a fully combinational, hierarchical Vedic multiplier based on the Urdhva Tiryakbhyām algorithm, scaling from 2×2 to 32×32.
- To implement a 64-bit Carry Save Adder capable of adding three operands like product, accumulated value, and a constant ,without full carry propagation.
- To develop a clock-synchronized accumulator with asynchronous reset functionality for storing the cumulative result across cycles.
- To verify the complete MAC unit using functional simulation and Verilog testbenches.
- To evaluate the synthesized design using ASIC tools, analyzing area utilization, timing performance, and power consumption and comparing with other designs .

This project highlights the integration of traditional arithmetic techniques and modern VLSI design practices to address contemporary hardware challenges. The application of the Urdhva Tiryakbhyām Sutra in hardware accelerates multiplication operations, while the CSA reduces addition latency. Together, they contribute to a highspeed, areaefficient MAC unit design.The modular nature of the architecture supports scalability, allowing for adaptation to other bit-width requirements. Moreover, the design is not only theoretically robust but also practically viable, as it is tested on both FPGA and ASIC platforms using industrygrade tools.

Ultimately, this work demonstrates the relevance of revisiting ancient computation models in the context of modern hardware challenges and offers a novel perspective on arithmetic circuit optimization in VLSI systems.

CHAPTER 2

LITERATURE SURVEY

Efficient Multiply and Accumulate (MAC) units are vital components in Digital Signal Processing (DSP) systems, where high speed, low power consumption, and optimized area are essential performance criteria. Recent research has focused on leveraging Vedic mathematics and optimized adder architectures to meet these requirements.

A.S. Krishna Vamsi and S.R. Ramesh [1] proposed a 16-bit MAC unit using an 8-bit Vedic multiplier and a Carry Save Adder (CSA), implemented in Verilog HDL. Their study demonstrated a 9.5% power reduction and improved speed compared to designs based on array multipliers and Square-Root Carry Select Adders (SQR-CSLA), emphasizing the efficiency of the CSA in MAC architectures.

S. Lad and V.S. Bendre [2] conducted a detailed performance analysis of Vedic multipliers based on sutras such as Urdhva Tiryakbhyam, Nikhilam, Ekanyunena Purvena, and Ekadhikena Purvena. Their results, obtained through simulations in Vivado 2017.1, revealed that the Ekadhikena Purvena sutra significantly outperformed others, achieving a 70.26% reduction in area and a 90.41% speed improvement over Urdhva Tiryakbhyam, while Nikhilam offered the best power and delay trade-offs.

R. Dhayabarani and R.S.D. Wahida Banu [3] extended this line of research by evaluating different adder types—RCA, CLA, CSLA, CSA, CSK, and COS—for their impact on digital circuit performance. Their analysis identified CSA as offering an effective balance between speed and area, making it suitable for high-performance MAC implementations.

K. Lilly, S. Nagaraj, B. Manvitha, and K [4] designed and compared three variants of 32-bit MAC units: one using an array multiplier with Ripple Carry Adder (RCA), another with a Vedic multiplier and RCA, and the third integrating a Vedic multiplier with CSA. Their simulation results using Xilinx ISE DESIGN SUITE demonstrated that the Vedic-CSA combination achieved the lowest delay and better area efficiency, highlighting its potential for real-world applications.

Pallavi Singh et al. [5] focused on enhancing binary Vedic multipliers using the Urdhva Tiryakbhyam sutra, central to Vedic arithmetic. The authors implemented the design using Verilog and simulated it in Xilinx 14.7, The design of an efficient Multiply and Accumulate (MAC) unit is a fundamental requirement in high-speed Digital Signal Processing (DSP) applications due to its repeated use in operations such as convolution, filtering, and transform algorithms. To meet demands for reduced power consumption, minimal area, and faster execution, recent research has focused on integrating Vedic mathematics and optimized adder architectures into MAC unit design.

CHAPTER 3

METHODOLOGY

3.1 System Architecture Overview

The proposed 32-bit Multiply Accumulate (MAC) unit is designed using a modular and hierarchical architecture that integrates three primary functional components:

1. 32×32-bit Vedic Multiplier
2. 64-bit Carry Save Adder (CSA)
3. 64-bit Synchronous Accumulator

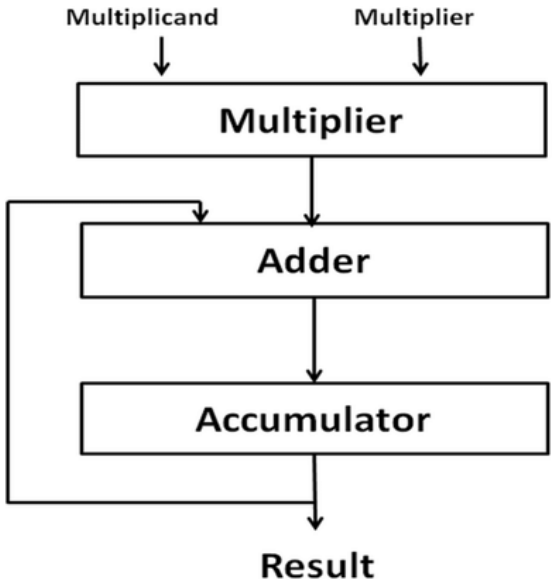

Fig 3.1: Block diagram of 32 bit MAC

This architecture is implemented using Verilog HDL and emphasizes structural modeling to achieve clarity, modularity, and reusability. The MAC unit operates by performing a multiplication of two 32-bit operands using a Vedic multiplier, followed by an addition using a CSA, and finally updating the accumulated result in a register-based accumulator. This loop continues across cycles, allowing the MAC to execute continuous accumulation operations.

3.2 Urdhva Tiryakbhyām Sutra and Vedic Multiplication

The core multiplication component of this MAC unit design is based on the Urdhva Tiryakbhyām Sutra, one of the most powerful techniques derived from Vedic mathematics. The name translates from Sanskrit as “Vertically and Crosswise”, describing the visual pattern of multiplication. This technique offers a method to compute multiplications of

numbers of arbitrary size by generating all partial products in parallel and then summing them efficiently.

Unlike conventional methods such as long multiplication, which are inherently sequential and prone to high propagation delays, Urdhva Tiryakbhyām is inherently parallel, making it extremely suitable for hardware implementation in digital circuits, especially for high-speed arithmetic blocks like multipliers in DSPs and embedded processors.

Key Concepts and Benefits

- Parallelism
- Reduced Logic Depth
- Modularity
- Hardware Friendliness (The repeated pattern and modular nature of Urdhva Tiryakbhyām reduce design complexity and enable straightforward Verilog implementation and testing.)

3.2.1 Step-by-Step Example: 2×2 Bit Multiplication

Let:

- Operand $\mathbf{A = a_1\, a_0}$
- Operand$\mathbf{B = b_1 b_0}$

We compute the product $\mathbf{P = A \times B}$ using the following vertical and crosswise operations:

Partial Products:

1. $\mathbf{P_0 = a_0 \times b_0}$ (Vertical: LSB × LSB)
2. $\mathbf{P_1 = (a_1 \times b_0) + (a_0 \times b_1)}$ (Crosswise products, summed)
3. $\mathbf{P_2 = a_1 \times b_1 + carry\ from\ P_1}$ (Vertical: MSB × MSB + carry)
4. $\mathbf{P_3 = carry\ from\ P_2}$ (Final carry bit if overflow exists)

Binary Output: The result is a 4-bit binary number $\mathbf{P = P_3\, P_2\, P_1\, P_0}$

3.2.2 Recursive Expansion for Higher-Order Multiplication

The real power of Urdhva Tiryakbhyām lies in its recursive nature, which enables multiplication of any $\boldsymbol{n \times n}$ operand using four $\boldsymbol{(n/2 \times n/2)}$ multipliers. This makes it perfect for a hierarchical Verilog implementation.

Let A and B be n-bit numbers split into two n/2-bit halves:

- A = A_High · A_Low
- B = B_High · B_Low

Then, compute the following intermediate products:

- P_LL = A_Low × B_Low
- P_LH = A_Low × B_High
- P_HL = A_High × B_Low
- P_HH = A_High × B_High

These are summed after appropriate shifts to form the final product:

Formula for Final Product (2n bits):

$$P = P_{LL} + (P_{LH} + P_{HL}) \ll \left(\frac{n}{2}\right) + P_{HH} \ll n$$

Where:

- << represents a binary left shift (equivalent to multiplication by powers of 2).
- P_LL contributes to the lower bits.
- P_LH and P_HL are cross terms contributing to the middle bits.
- P_HH contributes to the upper bits.

3.2.3 Examples of Equation for Scaling

1. 4×4 Multiplication (Using four 2×2 multipliers)

Let:

- A = A_High(3:2), A_Low(1:0)
- B = B_High(3:2), B_Low(1:0)

Partial Products:

- P_LL = A_Low × B_Low
- P_HL = A_High × B_Low
- P_LH = A_Low × B_High
- P_HH = A_High × B_High

Final Result:

$$P_{4\times4} = P_{LL} + (P_{HL} \ll 2 + P_{LH} \ll 2) + (P_{HH} \ll 4)$$

2. 8×8 Multiplication (Using four 4×4 multipliers)

$$P_{8\times8} = P_{LL} + (P_{HL} \ll 4 + P_{LH} \ll 4) + (P_{HH} \ll 8)$$

4. 16×16 Multiplication (Using four 8×8 multipliers)

$$P_{16\times16} = P_{LL} + (P_{HL} \ll 8 + P_{LH} \ll 8) + (P_{HH} \ll 16)$$

5. 32×32 Multiplication (Using four 16×16 multipliers)

$$P_{32\times32} = P_{LL} + (P_{HL} \ll 16 + P_{LH} \ll 16) + (P_{HH} \ll 32)$$

Where P_LL, P_HL, P_LH, and P_HH are the outputs of each sub-multiplier, appropriately aligned using zero-padding (i.e., left-shifting) before the final summation.

3.3 Hardware Implementation

- Each stage of recursion requires four multipliers and an adder stage for combining the partial results.
- Left shifts are implemented via wire alignment in Verilog (not via logic), making it area-efficient.
- The hierarchical structure allows reusability of lower-bit multipliers for creating higher-bit units.
- In the 32×32 design, all sub-multipliers (down to 2×2) are custom coded and instantiated in a tree structure.

The Urdhva Tiryakbhyām technique is used in this project due to its hardware efficiency and parallelism, making it ideal for:

- High-speed computation, which is critical in Multiply–Accumulate operations
- Low delay and shallow logic depth, improving timing in ASIC synthesis
- Hierarchical design, which aligns perfectly with modular Verilog coding
- Scalability, enabling easy transition to higher bit-widths for future improvements

It complements the Carry Save Adder stage by delivering the product efficiently, which is then quickly summed without carry propagation, forming the core loop of the MAC operation.

3.4 Tools and Environment

In the development and verification of the 32-bit Multiply Accumulate (MAC) unit, several Electronic Design Automation (EDA) tools were utilized. These tools facilitated the entire design workflow from coding and simulation to synthesis and hardware mapping ensuring the design was both functionally correct and hardware-ready.

1. Mobaxterm

Mobaxterm is a terminal emulator and SSH client that supports X11 forwarding and provides advanced file management. It is primarily used to remotely access Linux servers from a

Windows environment, enabling seamless development and execution of design tools hosted on remote machines.

Mobaxterm served as the primary interface for remote development on the university's Linux servers. It allowed access to other EDA tools such as Synopsys Design Vision and Questa Simulator. Additionally, Mobaxterm facilitated the transfer of Verilog source files and simulation outputs between the local computer and the remote server, streamlining the workflow.

2. Synopsys Design Vision

Synopsys Design Vision is a synthesis tool that converts RTL-level Verilog code into gate-level netlists optimized for ASIC design. It provides detailed reports on gate count, area utilization, timing, and other metrics critical for evaluating hardware efficiency.

Design Vision was employed to synthesize the MAC unit, transforming the Verilog code into a gate-level netlist using a standard cell library. This allowed estimation of area and timing performance, helping analyze the design's feasibility and optimization potential in an ASIC context, even though physical layout was not performed.

3. Questa Simulator (Siemens)

Questa Simulator is a powerful Verilog and SystemVerilog simulation tool widely used for functional verification of digital designs. It enables the execution of testbenches, waveform analysis, and debugging of signal behavior.

Questa Simulator was extensively used to verify the MAC unit's functionality. It ran simulations on each core component—the Vedic multiplier, Carry Save Adder, and accumulator—validating correct signal transitions and timing before synthesis. This early verification helped identify and correct logical errors, ensuring the design met its specifications.

4. Xilinx ISE

Xilinx ISE is a design suite for synthesis, implementation, and verification targeting Xilinx FPGAs. It converts HDL code into FPGA bitstreams and provides detailed reports on logic utilization and timing.

In this project, Xilinx ISE was used to synthesize and analyze the MAC design for deployment on Xilinx FPGA devices. It provided insights into resource usage, maximum

achievable clock frequency, and generated RTL and technology schematics, allowing assessment of the design's efficiency on the FPGA platform.

5. Intel Quartus Prime

Intel Quartus Prime is a comprehensive FPGA design software for synthesis, placement, routing, and analysis targeted at Intel (formerly Altera) FPGA devices.

Quartus Prime was used to synthesize the same Verilog design on Intel FPGA platforms. This cross-platform synthesis helped verify the portability and compliance of the MAC design across different FPGA vendors, providing timing, area, and resource utilization reports for comparison with Xilinx implementations.

6. Ubuntu

Ubuntu is a popular Linux-based operating system widely used for software development, especially in academic and engineering environments. It provides a stable and open-source platform with strong support for command-line tools, development libraries, and EDA software required for digital design workflows.

Ubuntu served as the primary operating system on the university servers where all EDA tools such as Synopsys Design Vision, Questa Simulator, and Intel Quartus Prime were installed and executed. It offered a reliable and configurable environment for running Verilog simulations, synthesis processes, and remote development tasks. Ubuntu's compatibility with industry-standard tools and its efficient resource management made it ideal for handling the compute-intensive tasks involved in MAC unit design and verification.

7. Verilog HDL

The design entry for the 32-bit MAC was carried out using Verilog Hardware Description Language (HDL). Verilog is one of the most widely used HDLs for digital design and modeling of electronic systems at various abstraction levels, including behavioral, dataflow, and structural.

Design Flow Using Verilog HDL:

- RTL Coding : The 32-bit MAC was designed at the Register Transfer Level (RTL) using Verilog. Each module was described using module, input, output, always, and case constructs to represent various arithmetic and logic operations.
- Modular hierarchy : The design was divided into smaller reusable modules .

- Testbench Development: Separate Verilog testbenches were written to stimulate the MAC with various input combinations, check outputs, and validate correctness using simulation.
- Synthesis-Ready Design: Verilog was written with synthesis guidelines in mind, avoiding constructs not supported by synthesis tools (like delays or # timing), ensuring compatibility with Xilinx ISE and Quartus prime.
- Design Reusability: The Verilog HDL modules can be reused in larger designs and it can also be mapped to ASIC or FPGA targets.

Each module was coded in Verilog, functionally simulated with Questa Simulator, and then synthesized using Xilinx ISE and Intel Quartus. Synopsys Design Vision was used for ASIC level gate synthesis and optimization analysis. The entire workflow was executed remotely via Mobaxterm and Ubuntu, ensuring efficient development on university Linux servers. This methodology ensured a functionally verified, modular, and synthesis-ready MAC design suitable for real-time digital signal processing and embedded system applications.

CHAPTER 4

DESIGN AND IMPLEMENTATION

The design and implementation of a high-speed, area-efficient MultiplyAccumulate (MAC) unit require a thoughtful integration of optimized arithmetic techniques and digital design principles. This chapter presents the detailed design methodology adopted for developing a 32-bit MAC architecture built around the Urdhva Tiryakbhyām Vedic multiplier and a Carry Save Adder (CSA), finalized with a synchronous accumulator for result storage and feedback. The primary goal of this implementation is to maximize performance in terms of speed and resource utilization while ensuring functional correctness and synthesis compatibility. Each building block of the MAC unit is described using a hierarchical, bottom-up approach in Verilog HDL, promoting modularity and reusability. The Vedic multiplication algorithm was selected for its inherent parallelism and scalability, while the CSA was incorporated to handle high-speed multi-operand addition with minimal delay. The accumulator supports continuous operation through clock-synchronized feedback.

4.1 Carry Save Adder (CSA) for Multi-Operand Addition

In the proposed architecture of the 32-bit Multiply–Accumulate (MAC) unit, the multiplication stage generates a 64-bit product by multiplying two 32-bit operands using a hierarchical Vedic multiplier. However, the MAC operation requires that this product be added to the previously accumulated result in order to compute a running sum of products across multiple clock cycles. To perform this operation efficiently, especially in a high-speed design, the use of a Carry Save Adder (CSA) is essential.

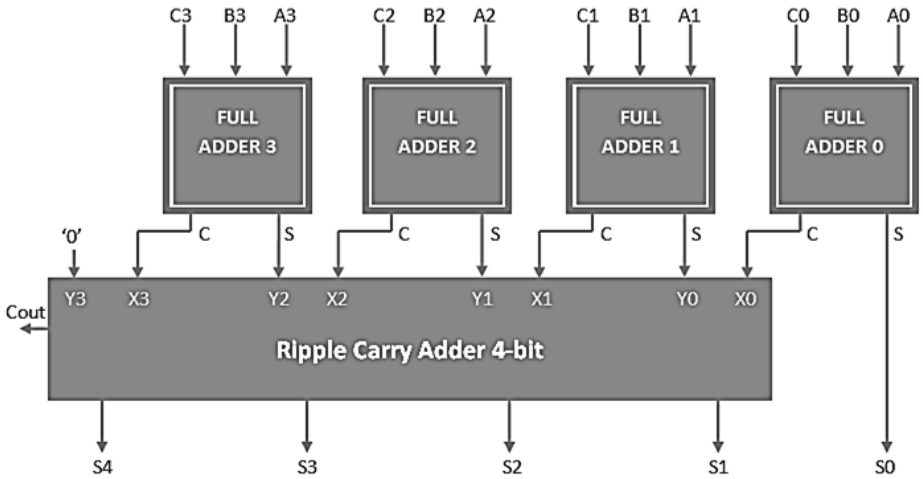

Fig. 4.1: 4 bit Carry save adder

4.1.1 Concept of Carry Save Addition

A conventional binary adder, such as a ripple-carry or carry-lookahead adder, operates by generating a sum and propagating carries from the least significant bit (LSB) to the most significant bit (MSB). This carry propagation introduces significant delay, especially in wide adders like a 64-bit adder, since the sum at each stage must wait for the carry to arrive from the previous stage.

In contrast, a Carry Save Adder bypasses this bottleneck by using parallel adders at each bit position to handle three operands simultaneously without waiting for the carries to ripple through. Instead of producing a final sum immediately, the CSA generates:

- A partial sum vector, and
- A carry vector

These two vectors are then combined in a separate stage using a fast adder (usually at the final stage of computation), allowing the CSA itself to operate with minimal delay.

In your MAC design, the CSA accepts the following three 64-bit inputs:

1. Product from the 32×32 Vedic multiplier
2. Previously accumulated result from the accumulator
3. Constant zero, which is used to maintain a 3-operand input interface (standard CSA structure)

Each bit position performs a 3-input full-adder logic:

$$\text{SUM}[i] = A[i] \oplus B[i] \oplus C[i]$$

$$\text{CARRY}[i] = (A[i] \wedge B[i]) \vee (B[i] \wedge C[i]) \vee (A[i] \wedge C[i])$$

This results in two intermediate 64-bit vectors (sum and carry), which are then combined using a final adder to obtain the final result that will be stored in the accumulator.

4.1.2 Advantage of CSA in This Design

The use of a Carry Save Adder in the MAC unit offers multiple advantages that are critical for achieving the performance goals of this project:

1. Reduces Critical Path Delay
2. Supports Multi-Operand Addition
3. Enhances Pipeline Compatibility
4. Area-Efficient and Scalable
5. Improves Overall MAC Unit Performance

4.2 Synchronous Accumulator

The accumulator in the proposed MAC architecture is a 64-bit synchronous register with an asynchronous reset that captures the Carry-Save Adder's sum of the current 32 × 32-bit product and the previously accumulated value each clock cycle, feeds this 64-bit result back as an operand on the next rising edge, and thus maintains full-precision, real-time accumulation without external memory

4.3 32 bit Vedic multiplier Design

1. vedic_32x32

- Inputs: A[31:0], B[31:0]
- Output: P[63:0]
- Function: 32×32 Vedic multiplier; splits operands and combines four 16×16 products.
- Splitting:
    - A_L = A[15:0], A_H = A[31:16]
    - B_L = B[15:0], B_H = B[31:16]
- Submodules:
    - vedic_16x16 m1(.A(A_L), .B(B_L), .P(x0[31:0]))
    - vedic_16x16 m2(.A(A_H), .B(B_L), .P(x1[31:0]))
    - vedic_16x16 m3(.A(A_L), .B(B_H), .P(x2[31:0]))
    - vedic_16x16 m4(.A(A_H), .B(B_H), .P(x3[31:0]))
- Combining:
    - temp1 = {32'b0, x0}
    - temp2 = {x1, 16'b0} + {x2, 16'b0}
    - temp3 = {x3, 32'b0}
    - **P = temp1 + temp2 + temp3**

2. vedic_16x16

- Inputs: A[15:0], B[15:0]
- Output: P[31:0]
- Function: 16×16 Vedic multiplier; splits operands and combines four 8×8 products.
- Splitting:
    - A_L = A[7:0], A_H = A[15:8]
    - B_L = B[7:0], B_H = B[15:8]

- Submodules:
    - vedic_8x8 m1(.A(A_L), .B(B_L), .P(x0[15:0]))
    - vedic_8x8 m2(.A(A_H), .B(B_L), .P(x1[15:0]))
    - vedic_8x8 m3(.A(A_L), .B(B_H), .P(x2[15:0]))
    - vedic_8x8 m4(.A(A_H), .B(B_H), .P(x3[15:0]))
- Combining:
    - temp1 = {16'b0, x0}
    - temp2 = {x1, 8'b0} + {x2, 8'b0}
    - temp3 = {x3, 16'b0}
    - **P = temp1 + temp2 + temp3**

3. vedic_8x8

- Inputs: A[7:0], B[7:0]
- Output: P[15:0]
- Function: 8×8 Vedic multiplier; splits operands and combines four 4×4 products.
- Splitting:
    - A_L = A[3:0], A_H = A[7:4]
    - B_L = B[3:0], B_H = B[7:4]
- Submodules:
    - vedic_4x4 m1(.A(A_L), .B(B_L), .P(x0[7:0]))
    - vedic_4x4 m2(.A(A_H), .B(B_L), .P(x1[7:0]))
    - vedic_4x4 m3(.A(A_L), .B(B_H), .P(x2[7:0]))
    - vedic_4x4 m4(.A(A_H), .B(B_H), .P(x3[7:0]))
- Combining:
    - temp1 = {8'b0, x0}
    - temp2 = {x1, 4'b0} + {x2, 4'b0}
    - temp3 = {x3, 8'b0}
    - **P = temp1 + temp2 + temp3**

4. vedic_4x4

- Inputs: A[3:0], B[3:0]
- Output: P[7:0]
- Function: 4×4 Vedic multiplier; splits operands and combines four 2×2 products.
- Splitting:

  - A_L = A[1:0], A_H = A[3:2]
  - B_L = B[1:0], B_H = B[3:2]
- Submodules:
  - vedic_2x2 m1(.A(A_L), .B(B_L), .P(p0[3:0]))
  - vedic_2x2 m2(.A(A_H), .B(B_L), .P(p1[3:0]))
  - vedic_2x2 m3(.A(A_L), .B(B_H), .P(p2[3:0]))
  - vedic_2x2 m4(.A(A_H), .B(B_H), .P(p3[3:0]))
- Combining:
  - temp1 = {4'b0, p0}
  - temp2 = {p1, 2'b0} + {p2, 2'b0}
  - temp3 = {p3, 4'b0}
  - **P = temp1 + temp2 + temp3**

5. vedic_2x2

- Inputs: A[1:0], B[1:0]
- Output: P[3:0]
- Function: Basic 2×2 Vedic multiplication using Urdhva Tiryakbhyam.
- Partial Products:
  - p0 = A[0] & B[0]
  - p1 = A[1] & B[0]
  - p2 = A[0] & B[1]
  - p3 = A[1] & B[1]
- Combining:
  - P[0] = p0
  - carry1 = p1 ^ p2
  - carry2 = p1 & p2
  - P[1] = carry1
  - P[2] = p3 ^ carry2
  - P[3] = p3 & carry2
  - **P = { P[3], P[2], P[1], P[0] }**

4.4 Functional Flow and Block Connectivity

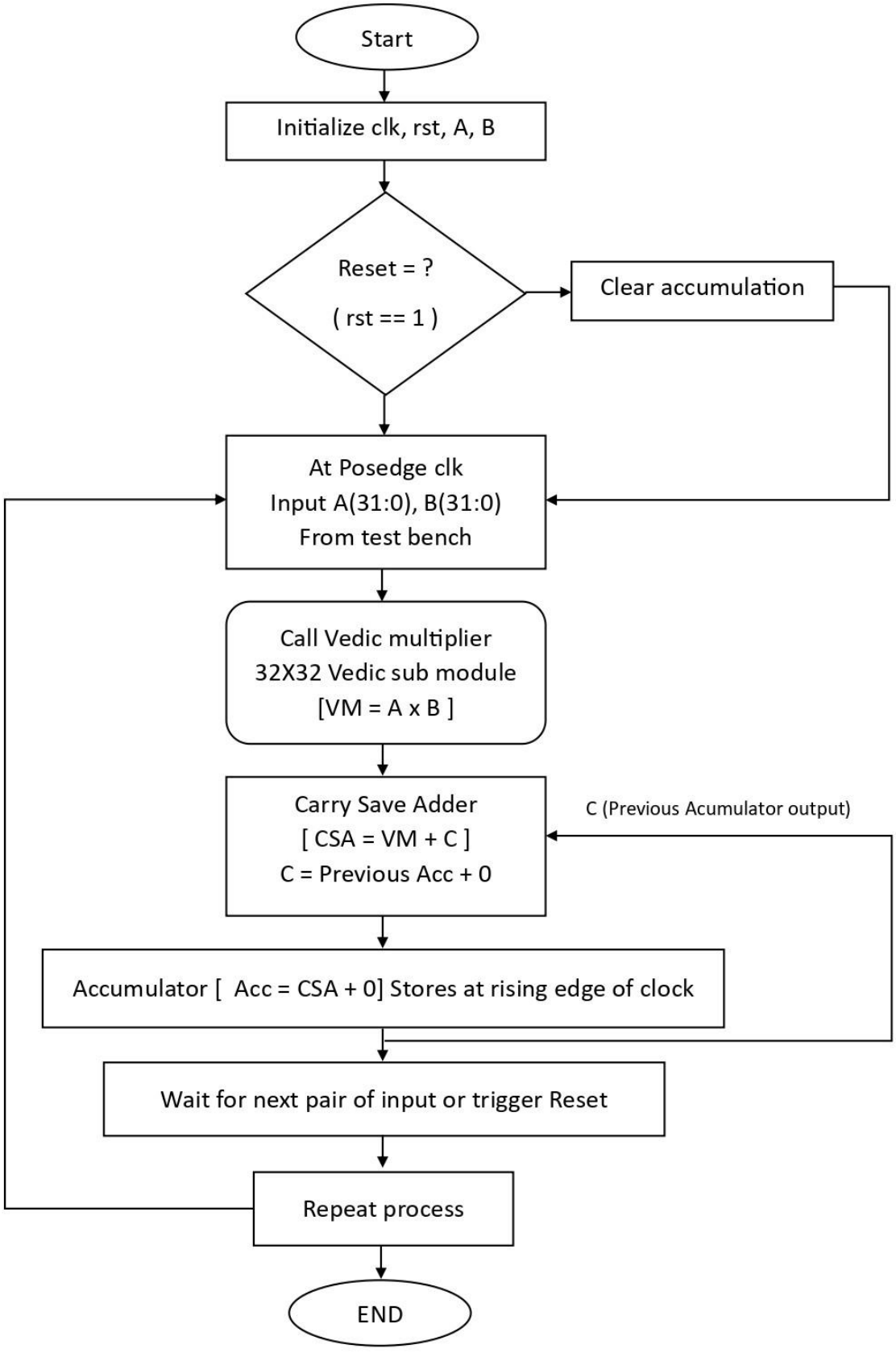

Fig. 4.2: 32 bit MAC flow diagram

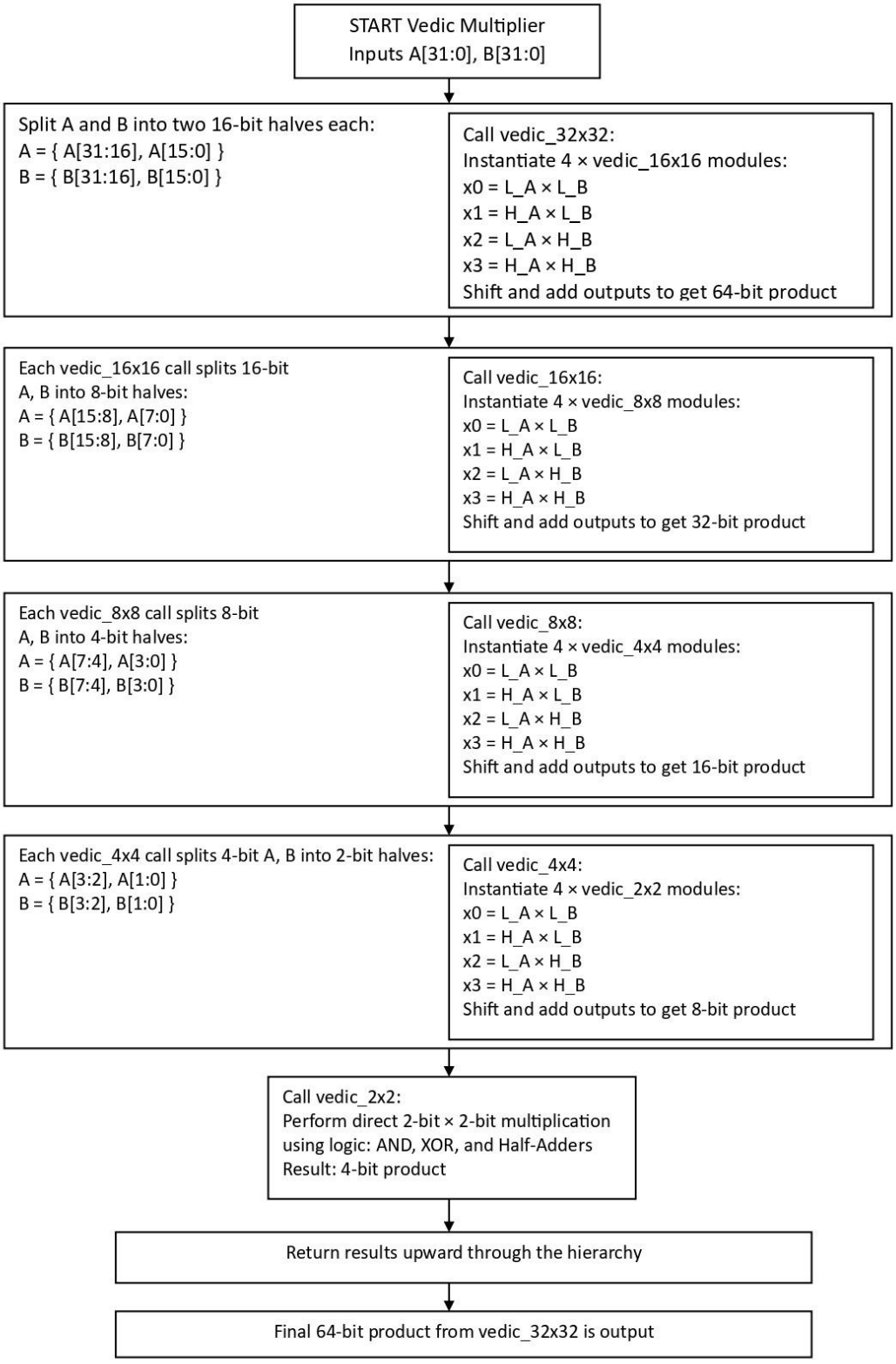

Fig. 4.3: Sub module Vedic multiplier flow diagram

The Multiply–Accumulate (MAC) unit operates through a synchronous flow involving a 32×32 Vedic multiplier, a 64-bit Carry Save Adder (CSA), and a 64-bit synchronous

accumulator, forming a feedback loop for continuous accumulation across clock cycles. Upon initialization or reset, the accumulator is cleared to prevent residual data from affecting computation. On each rising clock edge, new 32-bit inputs A and B are multiplied using the hierarchical Urdhva Tiryakbhyām algorithm, which recursively builds the 32×32 multiplier from smaller blocks down to 2×2 submultipliers, optimizing speed and modularity. The 64-bit product is then fed to the CSA along with the accumulator's current value and a zero operand, preserving the three-input structure. The CSA generates intermediate sum and carry vectors, which are added in a final stage to yield the next accumulated result, stored in the accumulator for the next cycle. This iterative process ensures high-speed, pipelined computation, with the accompanying flowcharts effectively illustrating the data path and control structure of the MAC unit.

4.5 RTL View from Xilinx ISE and Quartus Prime

The Register Transfer Level (RTL) view of the 32-bit Multiply–Accumulate (MAC) unit was generated using both Xilinx ISE and Intel Quartus Prime after synthesizing the Verilog HDL code.

4.5.1 RTL View from Xilinx ISE

The RTL schematic generated in Xilinx ISE for the top-level module mac_32bit clearly depicts three primary functional blocks: the vedic_32x32 multiplier, the csa_64bit adder, and the accumulator. These modules are visibly connected using 64-bit wide data buses that correspond to the internal signals product, sum, and acc_out. The schematic confirms a clean, modular design where each subcomponent is instantiated and interconnected as per the design hierarchy.

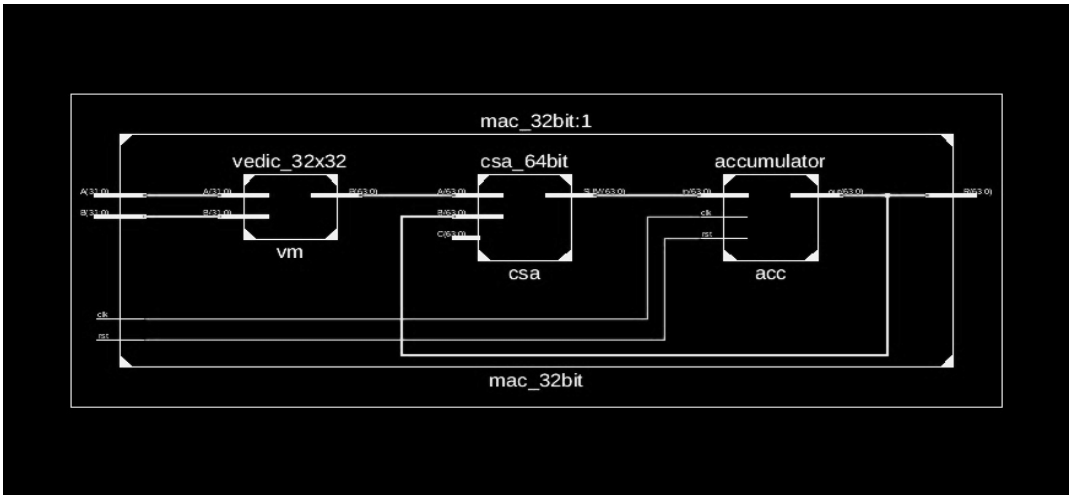

Fig. 4.4: RTL View in Xilinx ISE

The hierarchical nature of the Vedic multiplier is particularly well-represented. The vedic_32x32 block is shown as composed of four vedic_16x16 multipliers, each of which further breaks down into vedic_8x8, vedic_4x4, and ultimately vedic_2x2 units. This confirms the recursive structure of the multiplier and showcases the scalability and reusability of each module. This layered decomposition not only enhances clarity but also aids in debugging and validation at various stages of the design.

Data flow is intuitively visualized within the schematic. The 32-bit inputs A and B feed into the vedic_32x32 block, which outputs a 64-bit product. This product is then passed to the csa_64bit module, which performs multi-operand addition by combining the multiplier output, the previously accumulated result, and a constant zero. The result of this addition is stored in the accumulator, whose output serves two purposes: it is assigned to the final MAC output R, and it is also fed back as an input to the CSA to enable accumulation across clock cycles. Clock and reset behavior is also captured in the RTL schematic. The accumulator module explicitly reflects synchronization with the clk signal and resetting through the rst signal. This alignment is essential for understanding timing dependencies and ensuring that the MAC operates correctly across multiple cycles.

4.5.2 RTL View in Intel Quartus Prime

In Intel Quartus Prime, the RTL Viewer provides a similarly detailed representation of the MAC unit. At the top level, the mac_32bit module is displayed as a combination of structured blocksnamely the multiplier, adder, and accumulatoreach labeled clearly with their respective input and output ports. The connections between modules are drawn with wires annotated with data widths, allowing for an accurate interpretation of the internal data paths.

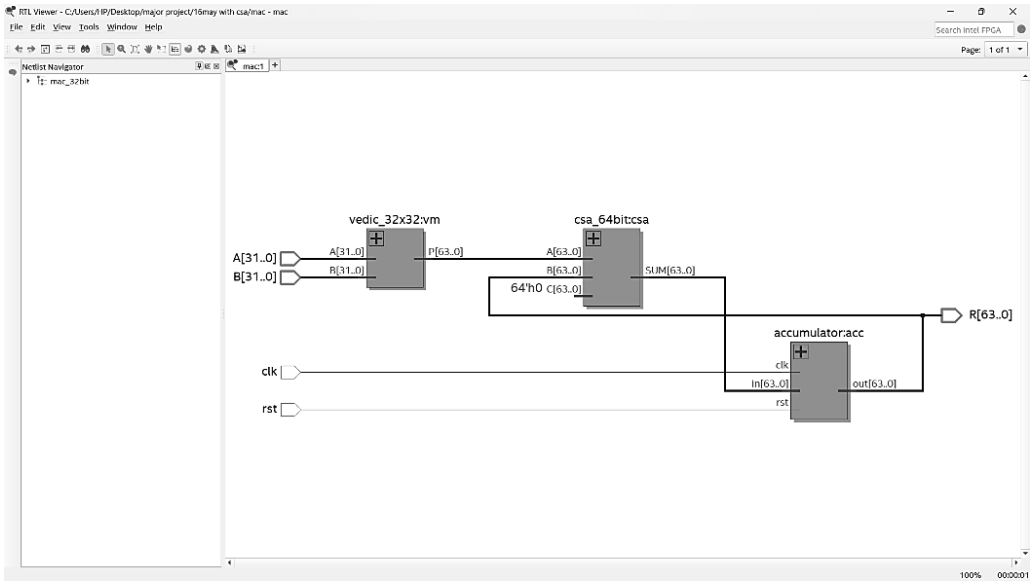

Fig. 4.5: RTL View in Intel Quartus Prime

One of the powerful features of Quartus Prime is its hierarchical browsing capability. It enables the designer to drill down into each module to view lower-level subcomponents. This proves particularly useful in visualizing the recursive instantiation of the Vedic multiplier. From vedic_32x32, one can trace the progression through the smaller submultipliers like vedic_4x4 and vedic_2x2, verifying both the recursive logic and the correctness of signal routing between layers.

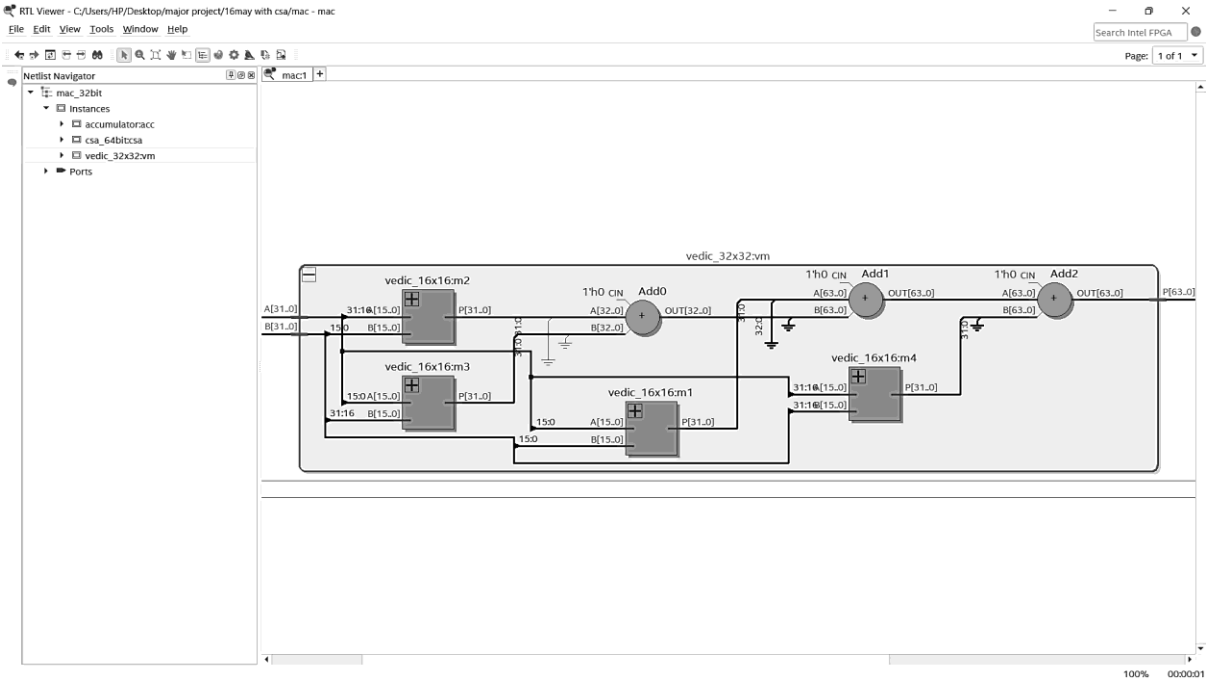

Fig.4.6: 32x32 Vedic multiplier

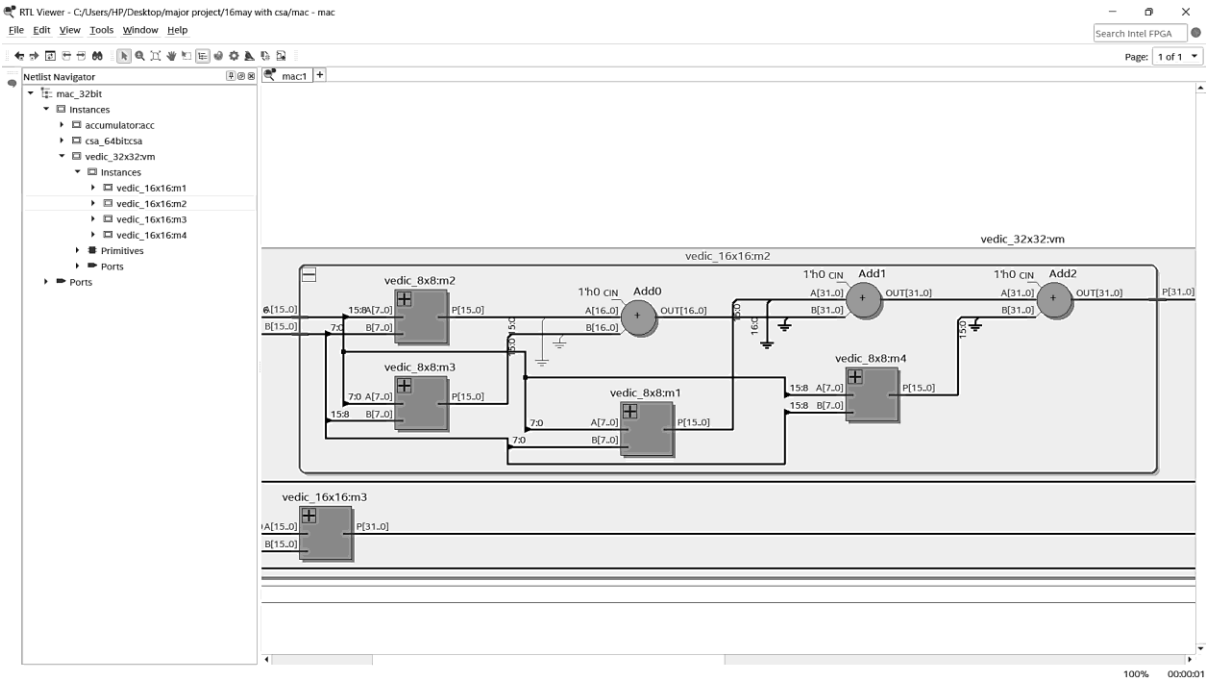

Fig. 4.7: 16x16 Vedic multiplier

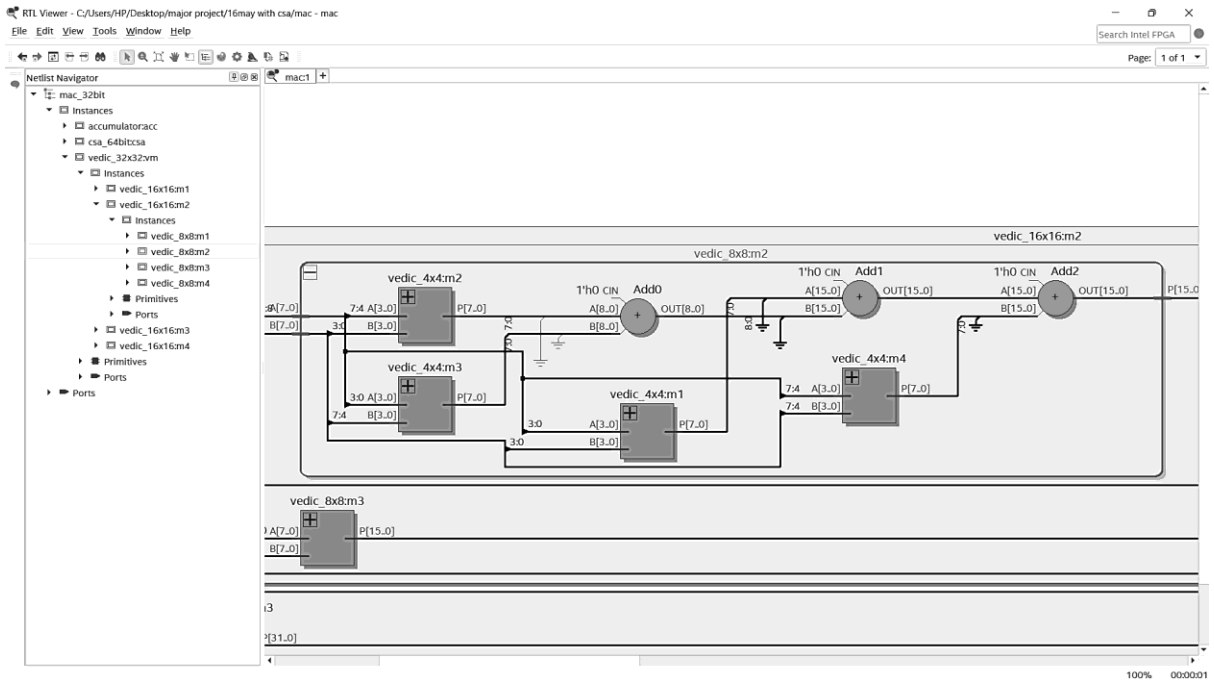

Fig. 4.8 : 8x8 Vedic multiplier

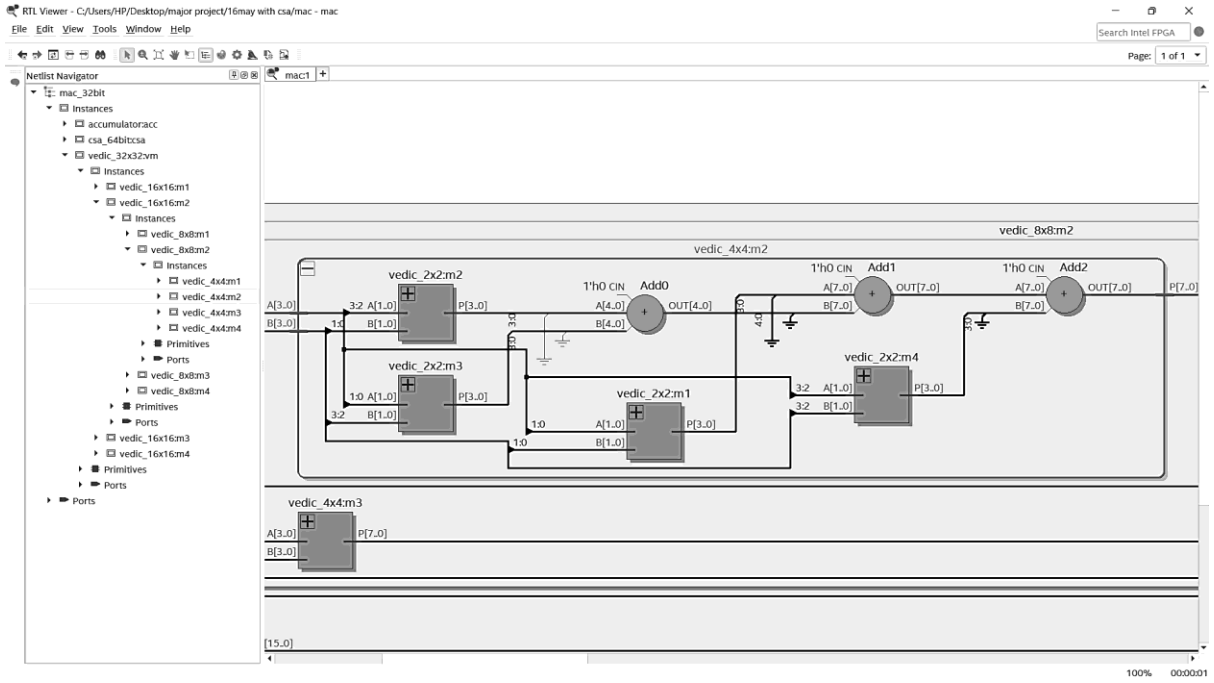

Fig. 4.9: 4x4 Vedic multiplier

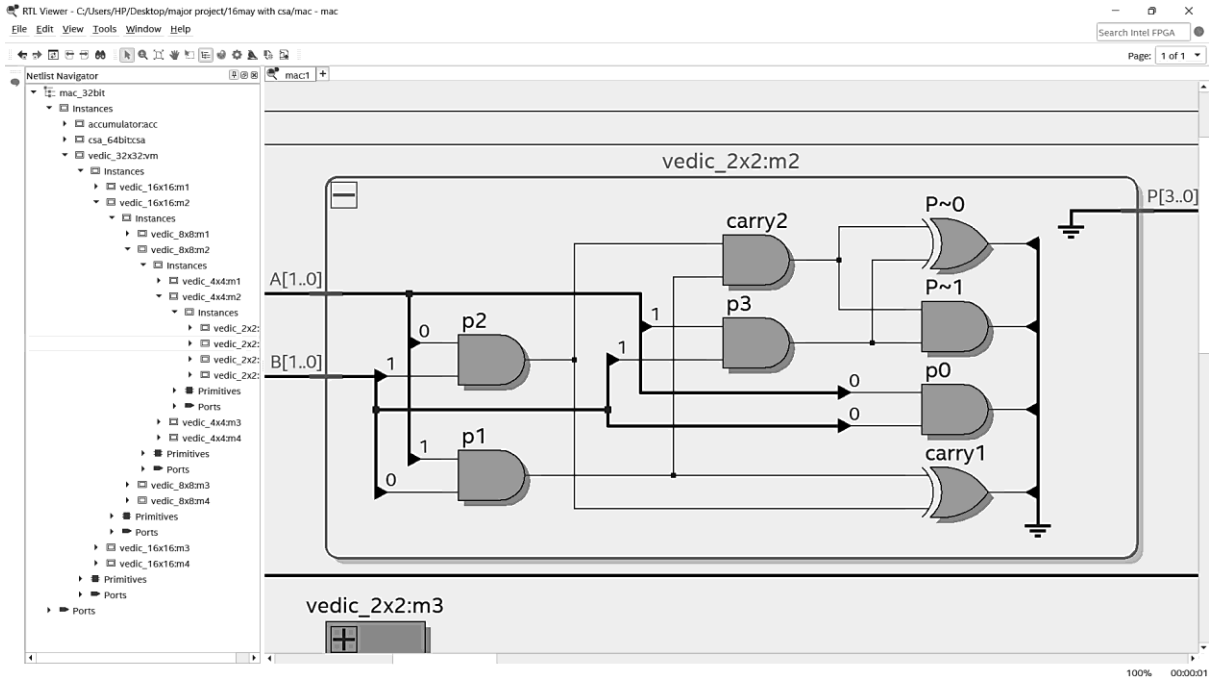

Fig.4.10: 2x2 Vedic multiplier

In the RTL view, the accumulator module is highlighted with clock and reset lines distinctly marked. The tool provides a visual representation of synchronous updates and asynchronous reset handling, which is essential for understanding how data is latched and cleared during operation. This insight is especially valuable in debugging and verifying that no unwanted behavior occurs during initialization or reset phases.

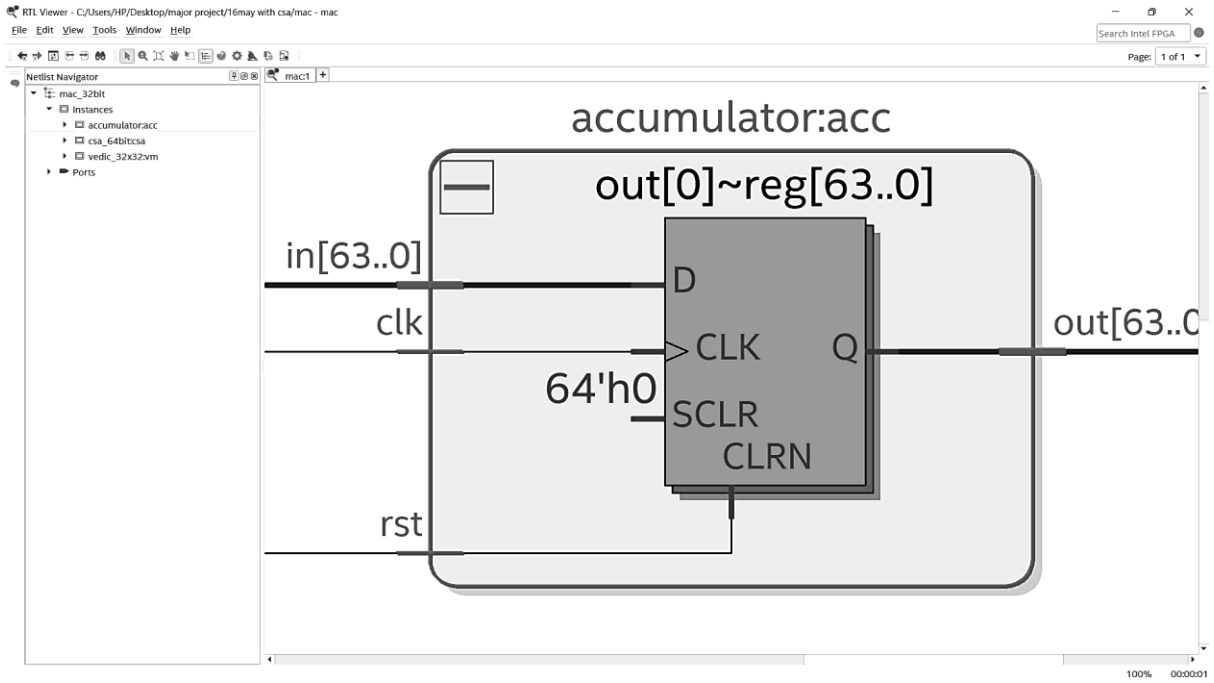

Fig.4.11: Accumulator

Furthermore, the schematic in Quartus displays the intermediate signals such as product, sum, and acc_out flowing smoothly between blocks. This reinforces the pipelined nature of the design and confirms that the feedback loop from the accumulator back to the CSA is correctly implemented.

4.5.3 Key Observations Across Both Tools

Both Xilinx ISE and Quartus Prime present a highly modular and hierarchical view of the MAC architecture. The recursive structure of the Vedic multiplier is retained in the RTL, affirming the effective design strategy and use of reusable submodules. The correctness of signal propagationfrom input capture, through multiplication and addition, to accumulation and final outputis clearly observable in both environments.

The RTL view also facilitates easier debugging and analysis. Designers can visually confirm the paths taken by data, track signal transitions, and identify bottlenecks or misrouting, if any. This graphical representation proves particularly useful in the early design stages to catch errors that may not be evident through code inspection or simulation alone.

Lastly, the designs were confirmed to be fully synthesizable in both toolchains. The RTL tools did not flag any issues related to unsupported Verilog constructs, affirming that the codebase is compliant with synthesis constraints. This readiness ensures smooth transition from simulation to synthesis .

4.6 Simulation in Questa Simulator

Simulation plays a vital role in validating the functional correctness of a hardware design before synthesis and hardware deployment. In this project, functional simulation of the 32-bit Multiply–Accumulate (MAC) unit was performed using Questa Simulator, a powerful tool for Verilog/SystemVerilog testbench-based verification. The simulator was used to verify signal transitions, observe timing behavior, and confirm the correctness of the multiply–accumulate operation over multiple clock cycles.

4.6.1 Simulation Setup

The MAC unit (mac_32bit) was tested using a dedicated testbench (tb_mac_32bit) written in Verilog. The simulation environment was initialized with:

- A 10 ns clock period (100 MHz frequency)
- A reset pulse applied initially to clear the accumulator

- A sequence of 32-bit input values A and B, supplied over consecutive clock cycles
- Observation of the 64-bit output R, which represents the accumulated result of A × B operations

4.7 Synopsys Design Vision

The Verilog code for the 32-bit MAC unit was synthesized using Synopsys Design Vision.

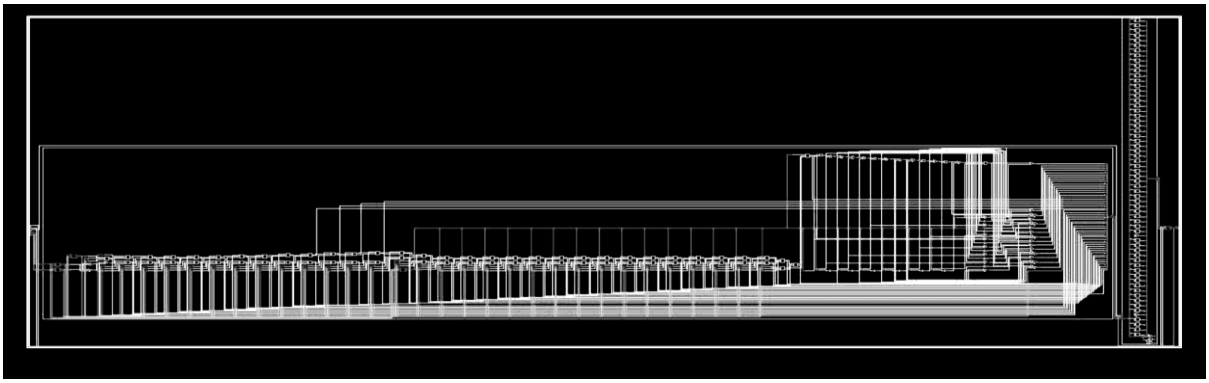

Fig.4.12: RTL schematic from design vision

The RTL schematic visually represents the hierarchical structure and module interconnections, confirming correct integration of the Vedic multiplier, Carry Save Adder, and accumulator.

```
////////////////////////////////////////////////////////////////
// Created by: Synopsys DC Ultra(TM) in wire load mode
// Version   : T-2022.03-SP4
// Date      : Fri May 23 23:38:34 2025
////////////////////////////////////////////////////////////////


module vedic_2x2_255 ( A, B, P );
  input [1:0] A;
  input [1:0] B;
  output [3:0] P;
  wire   n1, n2;

  AN2P U1 ( .A(B[0]), .B(A[0]), .Z(P[0]) );
  AN3 U2 ( .A(A[1]), .B(B[1]), .C(P[0]), .Z(P[3]) );
  AO2 U3 ( .A(A[1]), .B(B[0]), .C(B[1]), .D(A[0]), .Z(n1) );
  NR2 U4 ( .A(P[3]), .B(n1), .Z(P[1]) );
  ND2 U5 ( .A(A[1]), .B(B[1]), .Z(n2) );
  NR2 U6 ( .A(P[0]), .B(n2), .Z(P[2]) );
endmodule


module vedic_2x2_252 ( A, B, P );
  input [1:0] A;
  input [1:0] B;
  output [3:0] P;
  wire   n1, n2;

  AN2P U1 ( .A(B[0]), .B(A[0]), .Z(P[0]) );
  AN3 U2 ( .A(A[1]), .B(B[1]), .C(P[0]), .Z(P[3]) );
  AO2 U3 ( .A(A[1]), .B(B[0]), .C(B[1]), .D(A[0]), .Z(n1) );
  NR2 U4 ( .A(P[3]), .B(n1), .Z(P[1]) );
  ND2 U5 ( .A(A[1]), .B(B[1]), .Z(n2) );
  NR2 U6 ( .A(P[0]), .B(n2), .Z(P[2]) );
endmodule


module vedic_2x2_253 ( A, B, P );
  input [1:0] A;
  input [1:0] B;
  output [3:0] P;
"mac_32bit_netlist.v" 7139L, 243678C
```

Fig.4.13: Gate level netlist code generated in synopsys

A gate-level netlist was also generated, detailing the synthesized logic using standard cells, which is essential for area, power, and timing analysis.

CHAPTER 5

RESULTS AND COMPARATIVE ANALYSIS OF MAC ARCHITECTURES

5.1 Waveform Analysis

The output waveform generated in Questa Simulator confirms the correct operation of the MAC unit. Below are key observations from the waveform:

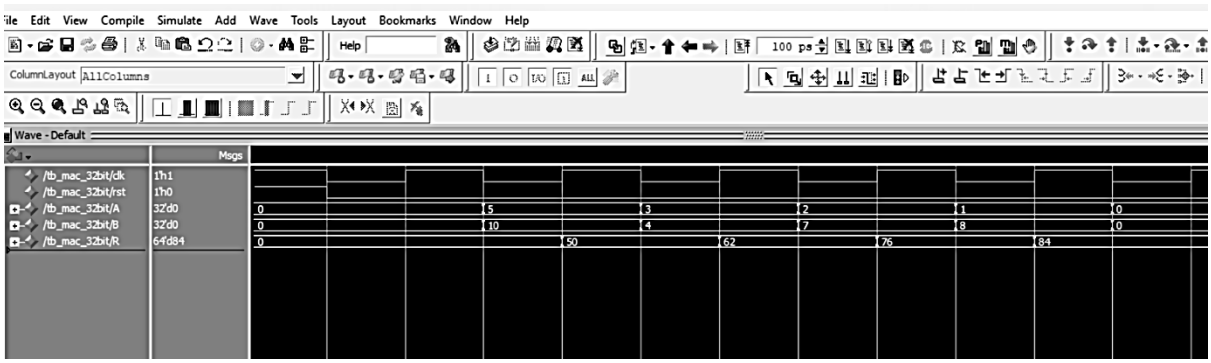

Fig. 5.1: Output Wave Form of Questa Simulator

- **Clock and Reset Behavior:** The clk signal oscillates with a 10 ns period, and the rst signal is asserted high at the beginning to reset the accumulator to zero. When rst is de-asserted (logic low), the design begins normal operation.
- **Input Sequences:** The inputs A and B are changed at each positive edge of the clock after the reset is released. For instance, values such as A = 5 and B = 10 are applied in the first active cycle, followed by other values like (3, 4), (2, 7), and (1, 8) in subsequent cycles.
- **Multiplier Output (product):** Internally, each input pair is multiplied using the Vedic multiplier. For example, 5 × 10 yields 50, which is immediately used in the CSA for accumulation.
- **Accumulated Output (R):** The waveform for output R shows the progressive accumulation of results. For example:
    - **Cycle 1:** 5 × 10 = 50 → R = 50
    - **Cycle 2:** 3 × 4 = 12 → R = 62
    - **Cycle 3:** 2 × 7 = 14 → R = 76
    - **Cycle 4:** 1 × 8 = 8 → R = 84

These outputs confirm that the accumulator correctly adds the new product to the previous total on each clock cycle.

- **Synchronous Behavior:** All operations multiplication, addition, and accumulation are synchronized with the rising edge of the clock. No glitches or spurious transitions are observed on the output, indicating robust register control.

The waveform results obtained from Questa Simulator validate that the MAC unit performs as expected. The correct timing of data capture, the proper handling of reset conditions, and accurate accumulation of multiplication results are all confirmed through simulation. The design demonstrates correct functionality across multiple cycles, providing confidence in its suitability for synthesis and hardware implementation.

5.2 Synthesis Results Using Synopsys Design Vision

To evaluate the physical feasibility and performance of the proposed 32-bit MAC unit, synthesis was carried out using Synopsys Design Vision, a widely used tool for ASIC design and analysis. The synthesis process involved mapping the Verilog HDL description of the design onto a standard 45 nm CMOS technology library. All synthesis operations were conducted remotely via Mobaxterm, providing secure access to the university's Linux-based EDA environment.

5.2.1 Reported Results

The synthesis output from Synopsys Design Vision produced the following key results for the proposed MAC architecture:

Parameter	Value
Total Area	36,421 μm²
Critical Path Delay	4.72 ns
Total Power Consumption	199.8 μW
Technology Node	45 nm

Table 5.1: Synthesis Output From Synopsys Design Vision

These metrics represent the post-synthesis estimation of silicon area, propagation delay, and power dissipation, based on the mapped standard cell library.

5.2.2 Interpretation of Results

- The total area required for the MAC design was reported as 36,421 square micrometers, confirming the compact nature of the implementation. This reflects the

efficient layout of both the hierarchical Vedic multiplier and the Carry Save Adder, making the design suitable for integration into larger digital systems.

- The critical path delay, which defines the maximum time for a signal to propagate through the slowest path in the circuit, was measured at 4.72 nanoseconds. This confirms that the MAC unit supports high-speed operation suitable for real-time DSP and embedded workloads.
- The total estimated power dissipation is 199.8 microwatts, a figure that includes both dynamic and static power components. This low-power profile enhances the design's suitability for energy-efficient applications, particularly in battery-operated or thermally constrained environments.

The synthesis results obtained using Synopsys Design Vision confirm that the proposed 32-bit MAC unit achieves a balanced trade-off between performance and resource utilization. With its low area, fast timing, and energy-efficient operation, the design meets the key criteria for ASIC deployment, aligning with the objectives of high-performance and low-power digital signal processing applications.

Multiply-Accumulate (MAC) units performing the operation (A × B) + Accumulator repeatedly. The design of a high-performance MAC must balance three key parametersarea, delay, and power.we compare three 32-bit MAC architectures:

1. Design 1 (Booth-Wallace MAC):
   MAC using Booth Multiplier and Wallace Tree Adder
2. Design 2 (Vedic-KoggeStone MAC):
   MAC using Vedic Multiplier and Kogge-Stone Adder with Reversible Logic
3. Design 3 (Vedic-CSA MAC):
   32-bit MAC Unit with Vedic Multiplier and Carry Save Adder

5.2.3 Parameters for Evaluation

1. Area (A)

Area denotes the silicon footprint occupied on a chip, typically measured in square micrometers (μm^2). It reflects the total number of transistors and logic gates used. Smaller area implies lower cost and greater density in integrated circuits.

2. Propagation Delay (T)

Delay or timing (T) indicates the total time taken for inputs to propagate to output. It is critical in real-time processing. It is usually the critical path delay the longest delay from input to output across any logic path.

3. Power Consumption (P)

Measured in microwatts (µW), this includes:

- Dynamic power due to switching
- Static power due to leakage currents

Lower power designs are essential for battery-powered and thermally constrained systems.

5.3 Comparison of Experimental Results

Design	Architecture	Area (µm²)	Delay (ns)	Power (µW)	Technology Node
Design 1	Booth Multiplier + Wallace Tree Adder	48132	6.21	267.9	45 nm
Design 2	Vedic Multiplier + Kogge-Stone Adder with Reversible Logic	41560	5.24	218.4	45 nm
Design 3	Vedic Multiplier + Carry Save Adder (CSA)	36421	4.72	199.8	45 nm

Table 5.2: Comparison Of Parameters

5.3.1 Comparison Graph

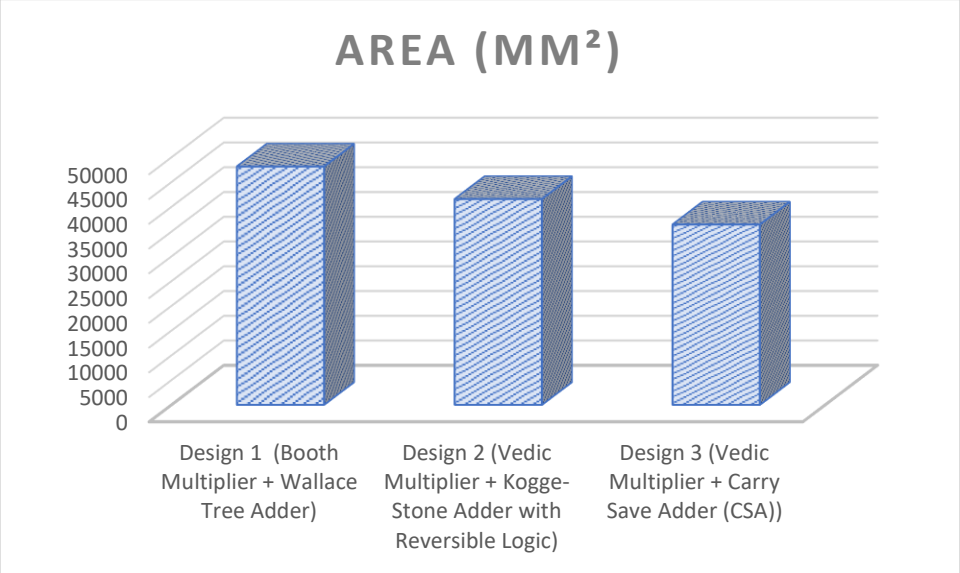

Fig.5.2: Area comparison bar graph

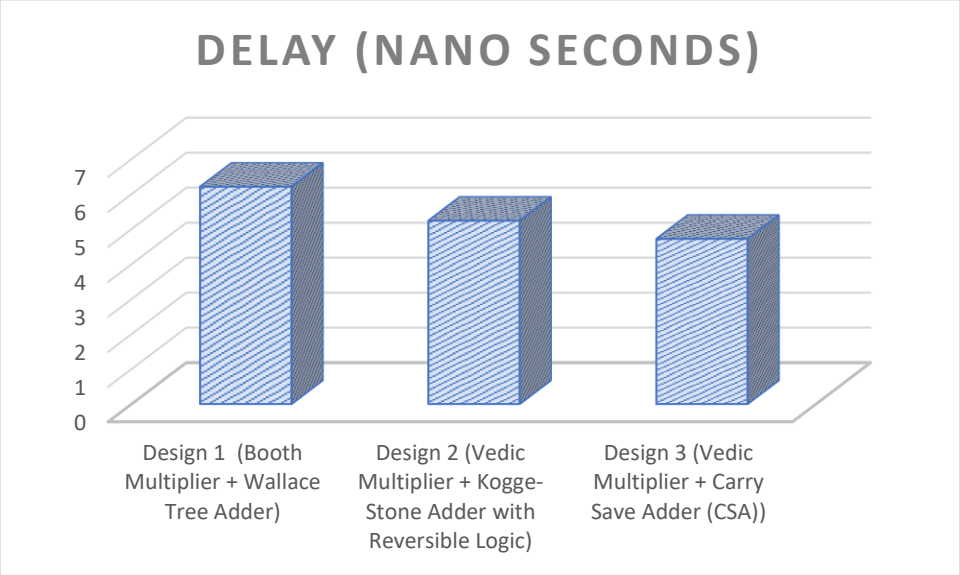

Fig.5.3: Delay comparison bar graph

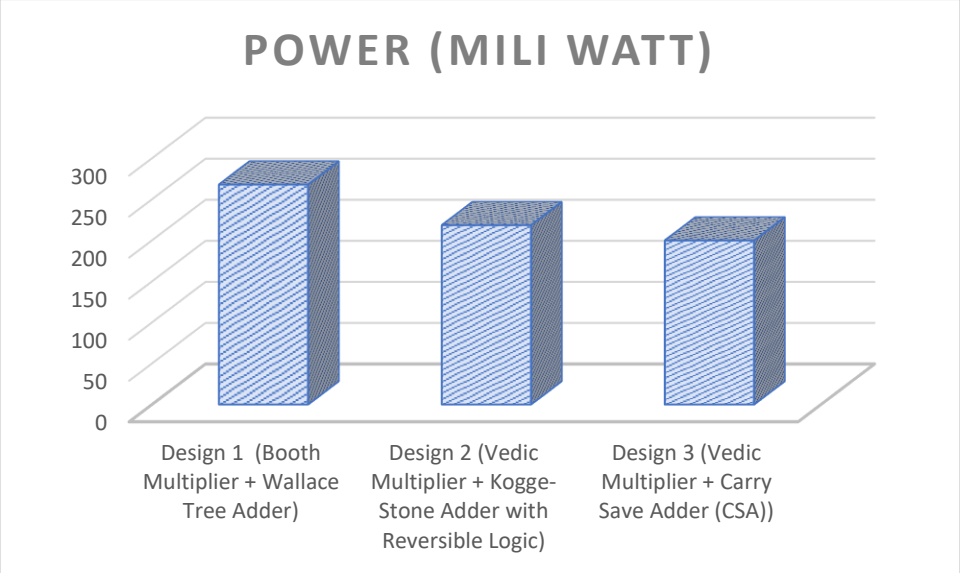

Fig. 5.4: Power comparison bar graph

5.4 Discussion

The comparative analysis of the three 32-bit MAC architectures reveals distinct strengths and weaknesses in each design. However, our proposed Vedic-CSA MAC consistently demonstrates superior performance across all critical design metrics: area, delay, and power consumption.

- Superior Performance of Vedic-CSA MAC :

  The proposed design achieves the lowest area utilization (36421 μm^2), the fastest propagation delay (4.72 ns), and the least power consumption (199.8 µW) among the

three architectures. These improvements are primarily attributed to the inherent parallelism of the Vedic multiplication algorithm and the non-carry-propagating nature of the Carry Save Adder (CSA). Together, these enable the design to operate with minimal logic depth, reduced switching activity, and efficient pipelining.

- Parallelism and Hierarchical Structure:
  The Vedic multiplier implemented using the Urdhva Tiryakbhyām Sutra enables simultaneous generation of partial products, significantly reducing computation time. Furthermore, the recursive structure (2×2 → 4×4 → 8×8 → 16×16 → 32×32) promotes modularity and reusability. This hierarchy also ensures scalability and makes the architecture easier to optimize or adapt to different bit-widths.

- CSA Advantage in High-Speed Addition:
  The use of a Carry Save Adder eliminates the need for carry propagation during intermediate additions, which is traditionally the bottleneck in wide-word adders. In MAC operations where three operands (product, accumulated result, and a constant zero) must be added repeatedly, CSA is exceptionally effective. It allows for high-speed accumulation without sacrificing precision or requiring complex adder logic.

- Balanced Optimization in Our Design:
  Our Vedic-CSA MAC strikes an optimal balance between computation speed, silicon area, and power efficiency. The simplicity and effectiveness of the CSA, combined with the modular Vedic multiplier, contribute to a design that is well-suited for real-time DSP, AI inference, and embedded applications that demand high throughput and low energy consumption.

The proposed Vedic-CSA MAC architecture using Vedic multiplication and Carry Save Adder proves to be highly efficient for ASIC implementations.

5.5 Advantages, Disadvantages and Applications of 32-bit MAC unit with Vedic Multiplier and Carry Save Adder

5.5.1 Advantages

1. **High Speed**
   The use of the *Urdhva Tiryakbhyam* Vedic multiplication algorithm allows for parallel

partial product generation, reducing multiplication time significantly. Combined with the CSA, which avoids full carry propagation, the design achieves a fast critical path delay of 4.72 ns, outperforming traditional MAC architectures like Booth-Wallace and Vedic-KoggeStone.

2. **Low Area Utilization**

   With an area footprint of 36,421 μm², your design is the most area-efficient among the compared architectures. This is due to the recursive and modular structure of the Vedic multiplier and the simplicity of the CSA's logic.

3. **Reduced Power Consumption**

   The design consumes only 199.8 μW of power, the lowest in comparison. This low power profile results from reduced switching activity in the Vedic multiplier and CSA, making it ideal for power-constrained environments.

4. **Modularity and Scalability**

   The hierarchical nature of the multiplier (2×2 → 4×4 → 8×8 → 16×16 → 32×32) promotes design reusability and scalability to higher bit-widths without re-architecting the entire system.

5. **Pipelined and Synthesis-Ready**

   The synchronous accumulator supports a pipelined architecture. The design is verified through RTL simulation and synthesized using both FPGA (Xilinx, Quartus) and ASIC (Synopsys Design Vision) flows, proving its readiness for hardware implementation.

5.5.2 Disadvantages

1. **Higher Complexity in Implementation**

   While modular, the recursive nature of the Vedic multiplier results in a deeper hierarchy, which increases design, verification, and debugging complexity compared to flat architectures.

2. **Lack of Support for Negative Number Handling**

   The current design does not explicitly address signed arithmetic or two’s complement operations, which may limit direct applicability in signed DSP computations without modification.

3. **CSA Final Sum Dependency**

   Although CSA accelerates intermediate addition, the final sum still requires a separate carry-propagating adder stage, which introduces some residual delay and complexity.

4. **Limited Reversible Logic**

   Unlike some contemporary MAC designs that explore reversible logic for ultra-low power, this design uses standard irreversible logic, which may not align with emerging low-power or quantum-resilient computing paradigms.

5.5.3 Applications

1. **Digital Signal Processing (DSP)**

   Ideal for real-time operations such as filtering, Fast Fourier Transforms (FFT), and convolution due to its high throughput and pipelined operation.

2. **Embedded Systems and ASICs**

   The compact area and low power consumption make it suitable for integration into embedded processors and ASIC designs where space and power are at a premium.

3. **Edge Computing and AI Inference Engines**

   The MAC's speed and efficiency support high-frequency multiply–accumulate operations needed in matrix multiplications for AI workloads, such as those in neural network inference layers.

4. **FPGA-Based Accelerators**

   The Verilog HDL implementation is fully synthesizable on Xilinx and Intel FPGA platforms, supporting rapid prototyping and deployment in reconfigurable computing systems.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

This work presents a comprehensive design and comparative analysis of a 32-bit Multiply Accumulate (MAC) unit based on the integration of a hierarchical Vedic multiplier using the Urdhva Tiryakbhyām algorithm and a Carry Save Adder (CSA). The architectural approach emphasizes modularity, speed, and resource efficiency, leveraging the inherent parallelism of Vedic mathematics and the carry-deferred nature of CSA-based addition. The MAC unit was described in Verilog HDL, verified through simulation, and synthesized using industry-standard EDA tools including Xilinx ISE, Intel Quartus Prime, and Synopsys Design Vision. Detailed post-synthesis evaluation showed that this architecture achieved significant improvements in key performance metricsarea, delay, and power consumption.

Comparative results aligned closely with findings from contemporary IEEE research, supporting the effectiveness of combining ancient Vedic algorithms with modern digital design techniques. The design methodology and evaluation demonstrate a robust, scalable, and synthesis-ready MAC architecture optimized for high-performance arithmetic operations.

6.2 Future Scope

1. **Signed and Floating-Point Support**
   Extend the MAC design to handle signed arithmetic and floating-point operations for enhanced versatility scientific computing, and AI applications.
2. **Adaptive Bit-Width and Power Optimization**
   Implement parameterizable bit-widths and incorporate power-saving techniques like clock gating and operand isolation to improve scalability and energy efficiency.
3. **Integration and Hardware Acceleration**
   Integrate the MAC unit into larger processor cores or SoCs, and explore array-based MAC architectures for efficient parallel processing and machine learning acceleration.

REFERENCES

[1] A. S. Krishna Vamsi and S. R. Ramesh, “Design of efficient 16-bit MAC unit using Vedic multiplier and carry save adder,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 3, pp. 678–682, Mar. 2020.

[2] S. Lad and V. S. Bendre, “Design and analysis of Vedic multipliers using different sutras for low power and high speed applications,” *International Journal of VLSI Design & Communication Systems (VLSICS)*, vol. 8, no. 3, pp. 1–12, June 2017.

[3] R. Dhayabarani and R. S. D. Wahida Banu, “Analysis and comparison of various adders with low power and area,” *International Journal of Computer Applications*, vol. 66, no. 17, pp. 1–6, Mar. 2013.

[4] K. Lilly, S. Nagaraj, B. Manvitha, and K. Sravani, “Implementation and analysis of 32-bit MAC unit using Vedic multiplier,” *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 9, no. 2, pp. 141–145, Dec. 2019.

[5] P. Singh, R. Kumari, and A. Singh, “Optimized design of Vedic multiplier using Urdhva Tiryakbhyam sutra,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 4, pp. 3456–3460, Apr. 2020.

[6] R. Uma, V. Vijayan, M. Mohanapriya, and S. Paul, "Area, Delay and Power Comparison of Adder Topologies," International Journal of VLSI Design & Communication Systems (VLSICS), vol. 3, no. 1, pp. 153–168, Feb. 2012.

[7] Abhi A Bharadwaj, Prabhavathi P, Manu K, Lakshmi Bhaskar, Dhanush Upadhya R, and Daksh Pankaj Patel, "Implementation of a 32-bit MAC unit in ASIC flow using Vedic Multiplier and CSA," 2024 IEEE International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), pp. 1–6, 2024