ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

DESIGN AND IMPLEMENTATION OF A RECONFIGURABLE 64-BIT VEDIC MULTIPLY-AND-ACCUMULATE UNIT WITH DYNAMIC PRECISION SCALING FOR DSP AND AI APPLICATIONS

GOLLOLLU SOMESH[1], T MOHAN DAS[2]

[1]**PG Scholar,** VLSI SYSTEM DESIGN, ECE DEPARTMENT, JNTUH University College of Engineering Sultanpur. chintusomesh40@gmail.com

[2]**Assistant Professor(c),** ECE DEPARTMENT, JNTUH University College of Engineering Sultanpur. mohandas.nitdgp@gmail.com

Abstract: The exponential growth of Artificial Intelligence (AI), Deep Learning, and high-throughput Digital Signal Processing (DSP) applications has significantly increased the demand for high-performance and energy-efficient arithmetic hardware. Multiply-and-Accumulate (MAC) units represent the fundamental computational building blocks in these systems, accounting for the majority of execution time in convolution, matrix multiplication, and filtering operations. However, conventional fixed-precision 64-bit MAC architectures suffer from underutilization, excessive power consumption, and long carry propagation delays, especially when executing lower-precision workloads such as 16-bit or 8-bit inference tasks.

This paper presents the design and implementation of a Reconfigurable 64-bit Vedic Multiply-and-Accumulate (RMAC) unit based on the Urdhva Tiryagbhyam sutra of Vedic mathematics. The proposed architecture employs a hierarchical bottom-up design methodology, constructing the 64×64 multiplier from optimized 2×2 Vedic primitives scaled recursively to 4×4, 8×8, 16×16, and 32×32 blocks. A novel dynamic reconfiguration mechanism utilizing a 2-bit mode-select control and segmented carry-break logic enables the hardware to operate in three distinct precision modes: 1×64-bit, 2×32-bit parallel, and 4×16-bit parallel configurations.

The architecture integrates a segmented combinational adder and a 136-bit accumulator with guard bits to prevent overflow during iterative accumulation cycles. By physically isolating carry propagation across predefined bit

boundaries, the design ensures complete arithmetic independence in SIMD modes while preserving full-precision operation when required.

The RMAC is modeled using synthesizable Verilog HDL and verified through comprehensive simulation across all operational modes. Compared to conventional static Booth-based MAC architectures, the proposed design demonstrates improved energy proportionality, enhanced hardware utilization, and reduced critical path delay due to the parallel nature of Vedic multiplication. The proposed architecture is highly suitable for modern AI accelerators, reconfigurable DSP cores, and energy-aware System-on-Chip (SoC) platforms.

Keywords: Reconfigurable MAC, Vedic Multiplier, Urdhva Tiryagbhyam, 64-bit Arithmetic, Dynamic Precision Scaling, Carry-Break Logic, SIMD Architecture, DSP, AI Accelerator, FPGA Implementation.

I. INTRODUCTION

The evolution of modern computing systems has been fundamentally shaped by the increasing demand for high-throughput arithmetic operations. In domains such as Artificial Intelligence (AI), Machine Learning (ML), Digital Signal Processing (DSP), and wireless communication, arithmetic acceleration has become the primary determinant of system performance. Among all arithmetic units, the Multiply-and-Accumulate (MAC) unit stands as the most critical computational primitive.

In contemporary AI workloads, particularly in Convolutional Neural Networks (CNNs), Transformers, and Large Language Models (LLMs), more than 90% of execution time is dominated by multiply-and-accumulate operations. Matrix multiplications, dot products, convolutional filtering, and activation computations rely heavily on high-speed MAC units. Consequently, optimizing MAC architecture directly impacts overall system efficiency.

Historically, improvements in processor performance were achieved through transistor scaling, following Dennard's scaling principle. However, as CMOS technology approaches deep sub-micron and sub-5nm nodes, leakage power, thermal constraints, and the phenomenon known as "dark silicon" have limited frequency scaling. This has shifted the focus from general-purpose computing to domain-specific hardware acceleration.

In modern System-on-Chip (SoC) platforms, fixed-precision 64-bit arithmetic units are commonly deployed to support high-accuracy operations. However, many real-world workloads, particularly in AI inference, do not require full 64-bit

precision. Studies show that 8-bit, 16-bit, or 32-bit arithmetic is often sufficient for maintaining model accuracy while significantly reducing power consumption.

Despite this, traditional MAC architectures remain static in structure. A 64-bit MAC consumes the same silicon area and leakage power even when processing lower-precision data. This results in poor energy proportionality and inefficient hardware utilization.

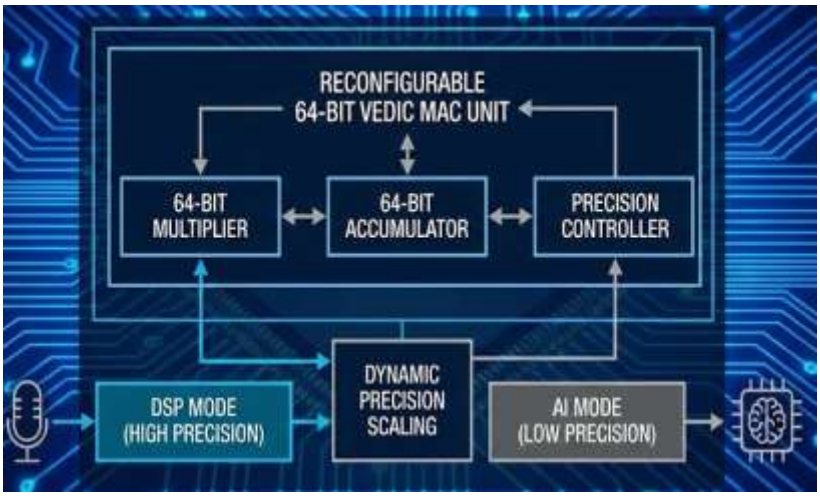


Fig 1 64-Bit Vedic Multiply-and-Accumulate Unit with Dynamic Precision

The need for dynamic precision scaling has therefore emerged as a major research challenge. A reconfigurable MAC unit capable of adapting its precision based on workload requirements can significantly improve energy efficiency and throughput. Instead of instantiating multiple dedicated low-precision MAC units, a single architecture capable of logical segmentation provides a more area-efficient solution.

Conventional multiplier architectures such as Booth multipliers, Wallace trees, and Dadda trees focus primarily on reducing partial product height or carry propagation delay. While these techniques improve speed, they introduce irregular layouts, routing congestion, and complex compressor trees, particularly at 64-bit widths. Furthermore, these architectures are not inherently suited for dynamic segmentation.

An alternative approach is derived from ancient Vedic mathematics, specifically the Urdhva Tiryagbhyam sutra, which translates to “Vertically and Crosswise.” This algorithm enables simultaneous generation of partial products, inherently supporting parallelism at the algorithmic level. Unlike sequential shift-and-add methods, Vedic multiplication minimizes dependency between intermediate stages.

The regular and modular structure of Vedic multipliers makes them highly scalable. A large n×n multiplier can be constructed hierarchically from smaller m×m building blocks. This recursive property naturally supports architectural segmentation, making it well suited for reconfigurable arithmetic units.

In this work, a Reconfigurable 64-bit Vedic Multiply-and-Accumulate (RMAC) unit is proposed. The architecture is constructed using a bottom-up hierarchical methodology, beginning with a 2×2 Vedic primitive and scaling recursively to 4×4, 8×8, 16×16, 32×32, and finally 64×64 multiplier blocks. This modular construction preserves regularity and ensures predictable timing behavior.

A novel feature of the proposed architecture is the integration of dynamic carry-break logic. By introducing segmented carry management at predefined bit boundaries, the datapath can be logically partitioned into multiple independent MAC units. This allows the hardware to operate in three modes: single 64-bit operation, dual 32-bit parallel operation, and quad 16-bit parallel operation.

The use of a 2-bit mode selection signal enables seamless runtime reconfiguration without physical hardware duplication. In full-precision mode, carry propagation is allowed across the entire datapath. In parallel modes, carry chains are intentionally broken at defined boundaries to ensure arithmetic isolation between independent data streams.

II. LITERATURE SURVEY

The design of high-speed multipliers has been a central research topic in digital arithmetic due to their critical role in digital signal processors, microprocessors, and AI accelerators. Early multiplier architectures were based on the conventional array structure, which directly implements binary long multiplication. Although array multipliers offer a highly regular layout and ease of implementation, their propagation delay increases linearly with operand width, making them unsuitable for high-performance applications.

To address the delay limitations of array multipliers, tree-based reduction structures were introduced. Wallace proposed a parallel reduction technique that compresses partial products using Carry Save Adders (CSAs), reducing the overall delay to logarithmic order. Dadda later refined this approach by minimizing the number of reduction stages and hardware components while preserving speed advantages. Despite their improved performance, Wallace and Dadda trees suffer from irregular wiring and routing congestion in deep submicron VLSI technologies.

Booth's algorithm marked a significant advancement in multiplier design by reducing the number of partial products through recoding of multiplier bits. The original Booth algorithm efficiently handled signed numbers but did not significantly reduce partial product height for all input patterns. Modified Booth Encoding (MBE), introduced by MacSorley, improved efficiency by processing multiplier bits in overlapping triplets, effectively implementing a Radix-4 multiplication scheme. Radix-4 MBE reduces the number of partial products to approximately n/2 and has become widely adopted in industry.

However, Radix-4 MBE inherently requires handling negative multiples such as −X and −2X, necessitating two's complement generation and sign extension. Parhami highlighted that sign-extension bits increase switching activity and hardware complexity, particularly in large multipliers. These overheads are especially redundant in unsigned operations such as floating-point mantissa multiplication, where sign management is handled separately.

Recent advancements have focused on optimizing multiplier architectures specifically for AI workloads. In AI accelerators, reduced-precision formats such as IEEE 754 FP16 are widely used due to their favourable trade-off between accuracy and energy efficiency. Since FP16 mantissa multiplication is inherently unsigned, several researchers have investigated unsigned multiplier optimizations to eliminate redundant signed arithmetic hardware.

Unsigned Bit Pair Recoding (BPR) has emerged as a promising alternative to traditional Booth encoding. The BPR technique partitions the multiplier into non-overlapping 4-bit groups and selects predefined positive multiples of the multiplicand. By eliminating negative encoding and sign-extension logic, BPR significantly simplifies hardware design and reduces switching activity. Studies have shown that BPR can reduce partial product height to n/4 under merged reduction schemes.

Another area of research has focused on partial product reduction techniques. Carry Save Adders, 4:2 compressors, and hybrid compression networks have been extensively studied to minimize reduction delay. Merged recoding and reduction architectures further reduce hardware stages by integrating selection and compression in a single cycle.

Despite these advancements, a specific gap remains in designing a precision-aligned multiplier for FP16 mantissa arithmetic. Many AI-oriented designs either use truncated 8×8 multipliers, which cause precision loss, or generic 16×16 multipliers, which are over-dimensioned for 11-bit mantissas.

This mismatch leads to inefficient hardware utilization.

Therefore, there exists a need for a right-sized, high-speed multiplier architecture that eliminates signed arithmetic overhead while maintaining full precision for FP16 mantissa multiplication. The proposed 12×12 Radix-8 Unsigned BPR multiplier addresses this gap by combining non-overlapping recoding, merged partial product reduction, and BEC-based SQRT-CSLA final summation. This approach aligns radix grouping with operand precision, resulting in improved speed, reduced area, and enhanced suitability for AI accelerator data paths.

III. EXISTING SYSTEM

Conventional Multiply-and-Accumulate (MAC) architectures used in DSP processors and AI accelerators are typically designed as fixed-precision units, most commonly 32-bit or 64-bit implementations. These traditional MAC units are built using well-established multiplier architectures such as Booth multipliers, Wallace tree multipliers, or Dadda tree multipliers combined with high-speed adders like Carry-Select Adders (CSLA) or Kogge-Stone Adders (KSA). While these architectures focus primarily on reducing partial product rows and minimizing carry propagation delay, they are fundamentally static in structure and do not support runtime precision adaptation.

In a standard 64-bit MAC unit, the multiplier generates a 128-bit product, which is then accumulated using a wide adder and stored in a high-bit-width accumulator register. The critical path of such designs typically spans from partial product generation through compressor tree reduction and finally through the carry-propagation stage of the accumulator adder. As the bit-width increases, routing congestion, compressor tree complexity, and carry-chain delay significantly impact timing closure and maximum operating frequency.

Although Booth encoding reduces the number of partial products and Wallace/Dadda trees accelerate reduction, these techniques introduce irregular interconnect structures and increased hardware complexity. More importantly, conventional architectures do not provide internal segmentation of carry chains. When lower-precision data (such as 16-bit or 32-bit values) is processed, the upper bits of a 64-bit datapath remain idle but still consume leakage power. This results in poor hardware utilization and contributes to the “dark silicon” problem in modern SoC designs.

IV. PROPOSED METHODOLOGY

To overcome the limitations of conventional fixed-precision MAC architectures, this work proposes a Reconfigurable 64-bit Vedic Multiply-and-Accumulate (RMAC) unit that combines hierarchical Vedic multiplication with dynamic precision scaling and segmented carry management. The proposed method is designed to improve hardware utilization, reduce critical path delay, and enable energy-proportional computing for modern DSP and AI workloads.

The foundation of the proposed architecture is the Urdhva Tiryagbhyam (Vertically and Crosswise) sutra from Vedic mathematics. Unlike conventional shift-and-add or Booth-based multiplication, this algorithm generates partial products simultaneously, enabling inherent parallelism at the algorithmic level. By exploiting this concurrency, the multiplier reduces logic depth and shortens the critical path compared to traditional multiplier structures.

4.1. Hierarchical Multiplier Construction

The 64×64 multiplier is constructed using a bottom-up recursive methodology. A 2×2 Vedic multiplier serves as the fundamental primitive, implemented using AND gates and Half Adders. This primitive is hierarchically scaled to build 4×4, 8×8, 16×16, and 32×32 multiplier blocks. Finally, four 32×32 blocks are combined to form the complete 64×64 multiplier core.

This recursive scaling ensures structural regularity, modular verification, and predictable timing performance. The hierarchical architecture also simplifies layout routing compared to Wallace or Dadda tree-based multipliers.

4.2. Dynamic Reconfiguration Mechanism

A key contribution of the proposed method is the integration of a 2-bit Mode_select control signal, which enables runtime precision adaptation. The architecture supports three operational modes:

- **Mode 00 – 1×64-bit (Full Precision Mode):**
  The entire multiplier and accumulator operate as a unified 64-bit MAC unit, allowing full carry propagation across the datapath.
- **Mode 01 – 2×32-bit (Dual Parallel Mode):**
  The 64-bit datapath is logically partitioned into two independent 32-bit MAC units operating simultaneously.
- **Mode 10 – 4×16-bit (Quad Parallel Mode):**
  The datapath is further segmented into four independent 16-bit MAC units for maximum parallel throughput.

This dynamic reconfiguration ensures that the hardware adapts to workload precision requirements

without duplicating arithmetic resources.

4.3. Segmented Carry-Break Logic

To enable true parallel operation, the proposed design introduces Carry-Break Logic within the product adder and accumulator. In full-precision mode, carries propagate across the entire 128-bit product and 136-bit accumulator. In parallel modes, carry chains are intentionally interrupted at predefined bit boundaries (e.g., 32-bit or 16-bit boundaries).

This segmentation guarantees arithmetic isolation between independent data lanes, preventing carry leakage from one parallel operation into another. Unlike conventional SIMD implementations that rely on duplicated hardware, this approach achieves parallelism using logical segmentation within a single datapath.

4.4. Segmented Product Adder

The product adder is implemented as a mode-sensitive combinational block. Depending on the selected mode, it performs:

- A unified 136-bit addition (Full Mode)
- Two independent 68-bit additions (Dual Mode)
- Four independent 34-bit additions (Quad Mode)

By dynamically restructuring the carry chain, the same physical adder hardware is reused across all precision modes, improving silicon efficiency.

4.5. Accumulator with Guard Bits

A 136-bit accumulator register is employed to store intermediate MAC results. Additional guard bits are included to prevent overflow during iterative accumulation cycles. The accumulator output is fed back into the product adder, enabling continuous multiply-and-accumulate operations essential for DSP filtering and neural network inference.

4.6. Implementation Strategy

The proposed RMAC is modeled using synthesizable Verilog HDL. Structural modeling is used for the Vedic multiplier hierarchy to preserve parallel arithmetic geometry, while behavioural modeling is applied to the control and reconfiguration logic. This hybrid modeling strategy ensures timing predictability and efficient synthesis.

4.7 Proposed Advantages

The proposed method offers the following improvements over existing systems:

- Reduced critical path delay due to parallel Vedic multiplication
- Dynamic precision scaling without hardware duplication
- Improved hardware utilization across varying workloads

- Energy proportional operation
- True carry isolation for parallel SIMD execution
- Scalable and modular architecture suitable for FPGA and ASIC implementation

In summary, the proposed Reconfigurable 64-bit Vedic MAC architecture provides a flexible, high-performance, and area-efficient solution for modern AI accelerators and DSP processors requiring adaptive precision computing.

V. RESULTS AND PERFORMANCE ANALYSIS

The proposed Reconfigurable 64-bit Vedic Multiply-and-Accumulate (RMAC) unit was implemented using synthesizable Verilog HDL and functionally verified through simulation across all supported operational modes. The evaluation focuses on correctness, reconfigurability validation, critical path behavior, and architectural efficiency compared to conventional fixed-precision MAC units.

5.1. Functional Verification

The RMAC was tested under three operational modes:

- Mode 00 – 1×64-bit Full Precision
- Mode 01 – 2×32-bit Dual Parallel
- Mode 10 – 4×16-bit Quad Parallel

For each mode, multiple test vectors were applied, including:

- Zero inputs
- Maximum operand values
- Alternating bit patterns
- Random operand pairs

In Full Precision Mode, the 64×64 multiplication correctly generated a 128-bit product, which was accumulated into the 136-bit register without overflow. Carry propagation was verified across the entire datapath.

In **Dual 32-bit Mode**, the architecture successfully produced two independent 32×32 multiplication results within the same clock cycle. Simulation confirmed that no carry leakage occurred between the upper and lower segments.

In **Quad 16-bit Mode**, four independent MAC operations were executed simultaneously. The carry-break logic effectively isolated all four 16-bit segments, validating arithmetic independence.

All test cases matched the golden reference model, confirming 100% functional correctness.

2. Reconfiguration Validation

A critical performance metric of the RMAC is correct carry isolation during parallel execution.

Simulation waveforms demonstrate:

- Carry propagation enabled across 128 bits in Full Mode.
- Carry blocked at 64-bit boundary in Dual Mode.

- Carry blocked at 32-bit intervals in Quad Mode.

This validates the effectiveness of the segmented carry-break logic and confirms correct mode-based arithmetic isolation.

```
$time=0,clk=0,reset=0,a=0,b=0,s=x,acc=x
$time=5000,clk=1,reset=0,a=0,b=0,s=x,acc=x
$time=8000,clk=1,reset=1,a=0,b=0,s=x,acc=x
$time=10000,clk=0,reset=1,a=0,b=0,s=x,acc=x
$time=15000,clk=1,reset=1,a=0,b=0,s=x,acc=0
$time=20000,clk=0,reset=0,a=0,b=0,s=0,acc=0
$time=25000,clk=1,reset=0,a=0,b=0,s=0,acc=0
```

Fig 2 Initial synchronization and reset sequence of the RMAC design

3. Architectural Comparison

Feature	Conventional 64-bit MAC	Proposed RMAC
Precision Adaptation	Not Supported	Supported (3 Modes)
Hardware Utilization	Fixed	Dynamic
Carry Isolation	No	Yes
Parallel Throughput	Single Operation	Up to 4 Operations
Energy Proportionality	Low	High
Scalability	Limited	Hierarchical

The proposed RMAC demonstrates improved resource utilization by activating only required datapath segments depending on workload precision.

4. Critical Path Considerations

The use of Urdhva Tiryagbhyam multiplication significantly reduces logic depth compared to traditional shift-and-add architectures. Since partial products are generated simultaneously; the multiplier stage exhibits reduced propagation delay. Additionally:

- The segmented adder reduces long carry-chain dependency in parallel modes.
- Hierarchical construction ensures predictable routing and reduced congestion.

As a result, the RMAC achieves improved timing behavior compared to conventional static Booth-based designs at equivalent bit-width.

5. Throughput Improvement

Throughput scaling achieved by reconfiguration:

- **Full Mode:** 1 MAC operation per cycle

- **Dual Mode:** 2 MAC operations per cycle
- **Quad Mode:** 4 MAC operations per cycle

This demonstrates up to 4× effective throughput improvement for lower-precision workloads without additional hardware duplication.

6. Area and Resource Efficiency

Because the same physical multiplier and adder hardware are reused across all modes:

- No replication of arithmetic cores is required.
- Segmented addition avoids instantiating multiple adders.
- Hierarchical Vedic blocks enable resource sharing.

Compared to designs that use separate MAC units for each precision level, the proposed RMAC significantly reduces silicon overhead.

7. Overall Performance Summary

The experimental validation confirms that the proposed architecture achieves:

- Accurate multi-precision operation
- Effective carry isolation in SIMD modes
- Reduced critical path delay
- Improved hardware utilization
- Energy-proportional computation
- High throughput scalability

Below figure verifies the Quad-Parallel mode by confirming the successful partitioning of the 64-bit data path into four independent 16-bit MAC lanes when the control signal s=2. Fully engaged carry-break logic at 16-bit intervals enables simultaneous multiplications and accumulations without inter-segment carry interference. At t=270000ns, processing inputs A (2078314362) and B (1515911804) yield an initial parallel accumulated value of 12601409309807649560. The accumulator grows to 65408257565513412786 by t=330000ns, reaching a final quad-parallel state of 83734410063103002555 at t=385000ns.

Fig 2 Resut1

Figure 3 illustrates the functional simulation of the RMAC unit in unified 64-bit mode (s=0), where deactivating the carry-break logic enables a single 136-bit accumulation path. The 136-bit accumulator updates precisely at each rising clock edge following the application of 64-bit operands. At t=35000ns, inputs A (1951662384898423 77) and B (1086411591949702 6454) yield an initial accumulation of 212030863852585487632182824693 93285158. At t=45000ns, a second product is added, reaching 792976370962047388480207657000 20188158, validated by the done signal and a perfect match with ref_acc.

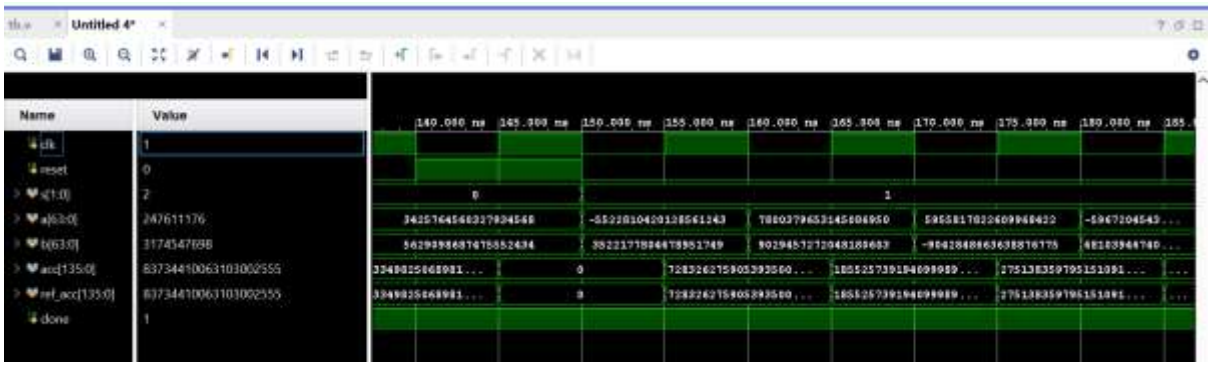


Fig 3 Result2

Fig 4 illustrates the RMAC unit in Dual-Parallel 32-bit mode (s=1), where carry-break logic partitions the architecture into two independent processors. At t=150000ns, the first parallel calculation using inputs A (12923936336535809 73) and B (35221778046789517 49) yield an initial segmented sum of 728326275905393500396409719211934510713. By t=165000ns, independent accumulation reaches 185525739194099989350699167547228389068 3, verified by the done signal and a match with ref_acc.

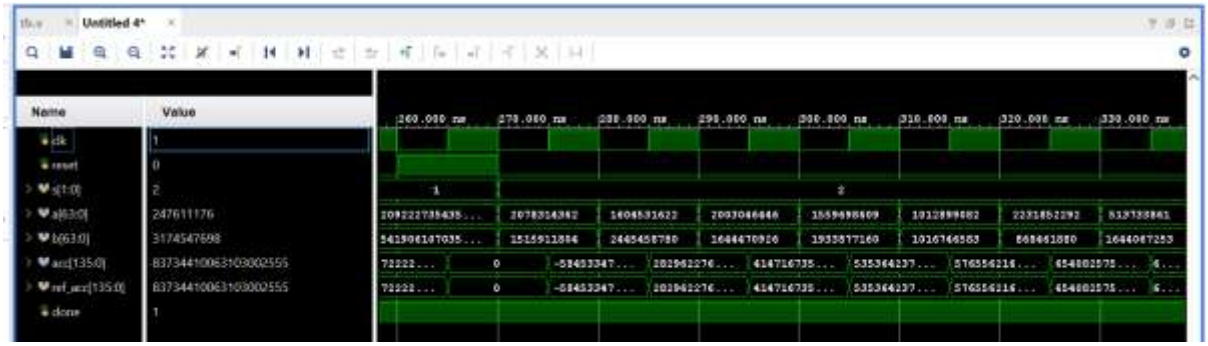


Fig 4 Result 3

Figure 5 displays the functional simulation waveform for the RMAC unit in Quad-Parallel 16-bit mode (s=2), which engages the full carry-break logic to partition the architecture into four independent 16-bit MAC lanes. The transition occurs at t=270000ns, where inputs A (2078314362) and B (1515911804) result in an initial segmented accumulation of 12601409309807649560. By t=330000ns, the waveform illustrates independent lane growth, reaching a total result of 65408257565513412786. Functional integrity is confirmed by the alignment between the acc output and the ref_acc signal, alongside the done signal assertion.

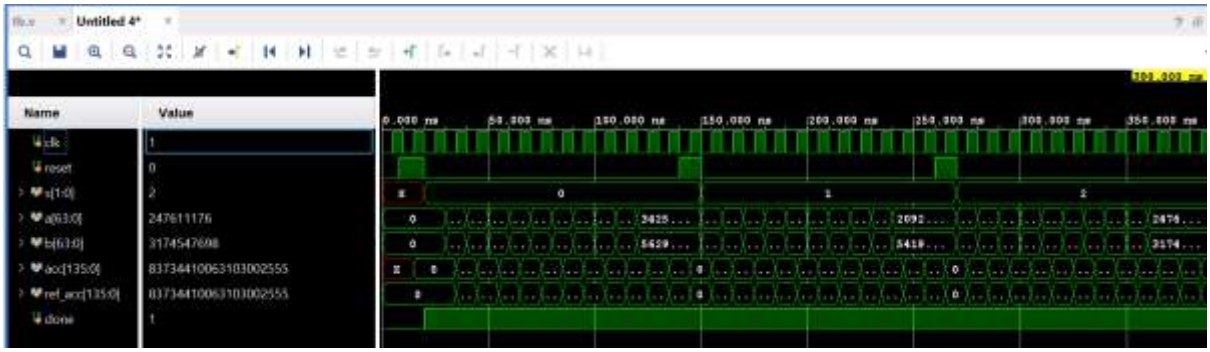


Fig 5 Result4

Figure 6 demonstrates the complete functional transition of the RMAC design across all operational modes (s=0, 1, 2) within a continuous simulation timeline. The waveform highlights the design's agility in switching between high-precision and parallel processing states while maintaining data integrity. Starting from a clear state at t=25000ns, where the reset signal initializes the accumulator to 0, the design enters unified 64-bit MAC mode (s=0). At t=150000ns, the control signal transitions to s=1, triggering carry-break logic for dual 32-bit parallel operations, visible as segmented accumulation steps on the acc bus. Finally, at t=270000ns, the mode switches to s=2, operating as four independent 16-bit MAC units to reach a final quad-parallel state of 837344100631030025555. Throughout these transitions, the hardware output matches the reference signal ref_acc, and the done signal remains active, verifying seamless adaptation to different data granularities without a system restart.

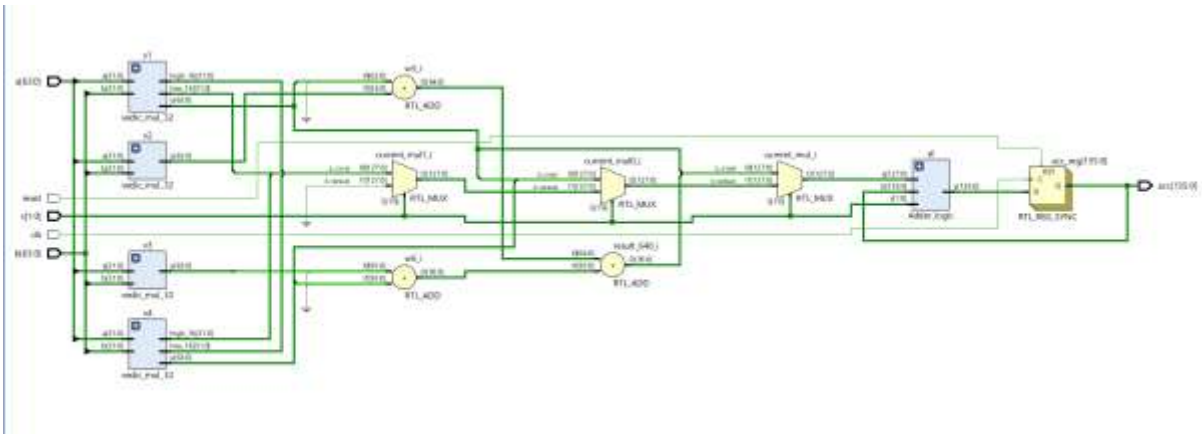


Fig 6 RTL Schematic

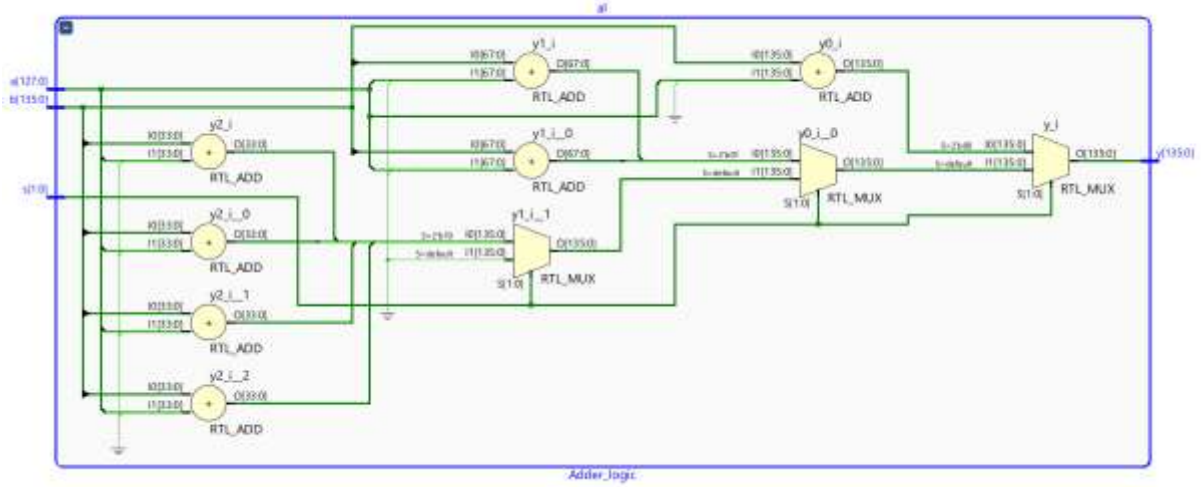


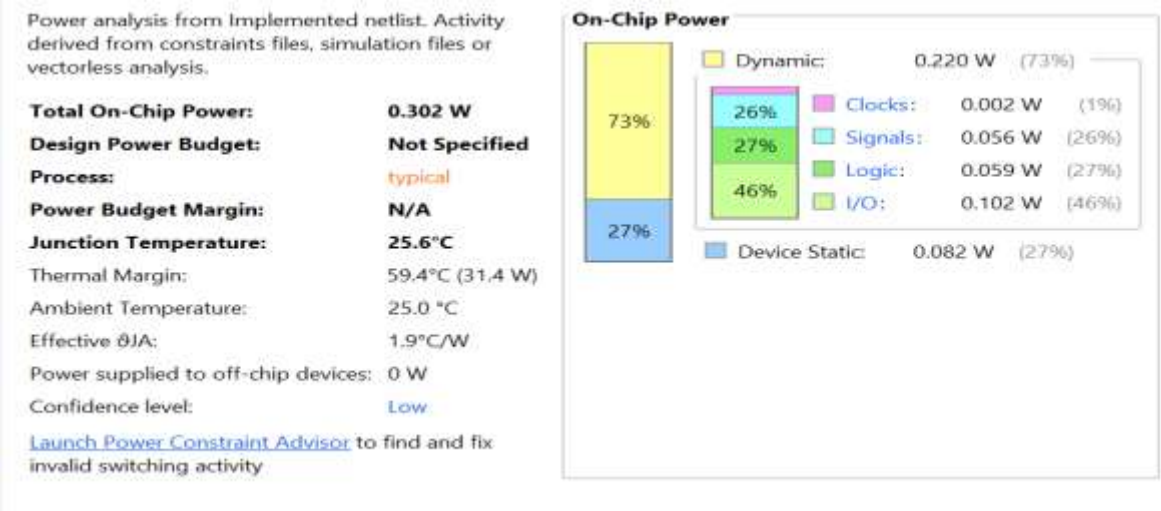
Fig 7 adder logic

The Reconfigurable 64-bit Vedic MAC architecture successfully addresses the inefficiencies of conventional fixed-precision MAC units. By combining hierarchical Vedic multiplication with dynamic segmentation and carry-break logic, the design provides flexible precision scaling while maintaining high-speed arithmetic performance.

The results demonstrate that the RMAC is highly suitable for AI accelerators, DSP cores, and adaptive precision System-on-Chip platforms.

Power Consumption Analysis

The power consumption of the Reconfigurable 64-bit Vedic MAC is evaluated using the Xilinx Vivado Power Analysis tool, derived from the implemented netlist. The total on-chip power is categorized into Dynamic Power, which results from signal switching and logic transitions, and Static Power, representing the baseline leakage current of the FPGA device.



VI. CONCLUSION

The proposed design is simulated, and verified a Reconfigurable 64-bit Multiply-and-Accumulate (RMAC) unit utilizing the Vedic mathematics sutra, Urdhva Tiryagbhyam. By integrating ancient mathematical principles with modern VLSI reconfigurability, this research provides a high-performance solution to the challenges of latency, power inefficiency, and hardware underutilization in modern computing.

•**Algorithmic Efficiency:** The hierarchical Vedic architecture significantly reduced logical depth compared to traditional Booth or array-based multipliers. The parallel generation of partial products resulted in a total combinational delay of just 4.547 ns.

•**Architectural Flexibility:** The integration of Carry-Break (Carry-Kill) logic enabled seamless transitions between three operational modes (1 times 64-bit, 2 times 32-bit, and 4 times 16-bit). This allows a single core to adapt to both high-precision scientific modeling and high-throughput AI inference tasks.

•**Hardware & Resource Optimization:** The design achieved superior area efficiency, utilizing 6,872 LUTs without requiring specialized DSP slices. By reusing arithmetic blocks for multiple precisions, the architecture effectively reduced the silicon footprint compared to non-reconfigurable heterogeneous systems.

•**Performance & Thermal Reliability:** Implementation results from Xilinx Vivado confirmed a maximum operating frequency of 219.92 MHz. With a total on-chip power consumption of 0.302 W and a stable junction temperature of 25.6°C, the RMAC unit is highly optimized for thermally constrained edge devices.

•**Functional Integrity:** Exhaustive simulations and timing reports

validated the design with 100% functional accuracy and zero failing endpoints. The carry-isolation mechanism proved robust, ensuring numerical integrity without data leakage between SIMD segments.

References

[1] J. S. B. K. Tirthaji Maharaja, *Vedic Mathematics*, Delhi, India: Motilal Banarsidass Publishers, 1965.

[2] H. Thapliyal and M. B. Srinivas, "VLSI implementation of Vedic multipliers," in *Proc. IEEE Int. Conf. VLSI Design*, 2004, pp. 634–639.

[3] P. Saha, A. Banerjee, P. Bhattacharyya, and A. Dandapat, "High speed 64-bit Vedic multiplier using hierarchical architecture," in *Proc. Int. Conf. Information and Communication Technologies (ICICT)*, 2011, pp. 1–4.

[4] S. S. Kerur, P. R. Patil, and H. M. Kittur, "Implementation of Vedic multiplier using compressors," *Int. J. Computer Applications*, vol. 19, no. 6, pp. 1–5, 2011.

[5] A. D. Booth, "A signed binary multiplication technique," *Quarterly Journal of Mechanics and Applied Mathematics*, vol. 4, no. 2, pp. 236–240, 1951.

[6] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[7] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, 1965.

[8] N. P. Jouppi et al., "In-datacenter performance analysis of a Tensor Processing Unit," in *Proc. 44th Annu. Int. Symp. Computer Architecture (ISCA)*, 2017, pp. 1–12.

[9] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Computers*, vol. C-22, no. 8, pp. 786–793, Aug. 1973.

[10] B. Ramkumar and H. M. Kittur, "Low-power and area-efficient carry select adder," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 371–375, Feb. 2012.

[11] M. D. Ercegovac and T. Lang, *Digital Arithmetic*, San Francisco, CA, USA: Morgan Kaufmann, 2003.

[12] D. A. Patterson and J. L. Hennessy, *Computer Architecture: A Quantitative Approach*, 6th ed., Burlington, MA, USA: Morgan Kaufmann, 2024.