An Optimized RISC Core Architecture Employing Vedic Mathematics for Low-Power Applications

Ms. B. PRAVALLIKA, CHOWDARI RAVIKUMAR[2]
[1]Assistant Professor, Dept. of E.C.E, RV Institute of Technology, Guntur- 522212
[2]PG Scholar, Dept. of E.C.E, RV Institute of Technology, Guntur- 522212

Abstract: This paper presents the design and implementation of a high-performance 16-bit Reduced Instruction Set Computer (RISC) processor incorporating a Vedic-based Multiply–Accumulate (MAC) unit, a hybrid Vedic–Karatsuba multiplier, and a parallel prefix adder architecture. The proposed design aims to enhance computational speed, reduce propagation delay, and minimize power consumption for low-power embedded applications. The processor architecture consists of essential modules including the Arithmetic and Logic Unit (ALU), control unit, register bank, program counter, data path, and memory unit, supporting the execution of 14 instructions. To improve arithmetic efficiency, the ALU integrates a high-speed parallel prefix adder that significantly reduces carry propagation delay during addition operations. The MAC unit employs the Urdhva Tiryakbhyam Sutra from Vedic mathematics for efficient lower-bit multiplication, while the Karatsuba algorithm is utilized for higher-order multiplication to decrease partial product generation and hardware complexity. The combination of these techniques provides improved throughput and optimized resource utilization compared to conventional multiplier architectures. The complete design is modeled using Verilog HDL, synthesized using Xilinx ISE 14.7, and functionally verified through Model Sim 6.3 simulations. Experimental results demonstrate notable improvements in speed, area efficiency, and power dissipation, validating the suitability of the proposed processor architecture for next-generation high-speed and energy-efficient digital systems.

Key Words: Reduced Instruction Set Computer, Von-Neumann architecture, Verilog HDL, Vedic Mathematics, and Sutras.

1.Introduction

The increasing demand for high-speed and energy-efficient computing systems led to the development of Reduced Instruction Set Computer (RISC) architectures as an alternative to Complex Instruction Set Computer (CISC) designs. In CISC processors, complex control logic and multiple memory access cycles reduced execution efficiency and increased processing delay. RISC overcomes these limitations by using a simplified instruction set, where most operations are executed within a single clock cycle through efficient pipelining techniques. The reduced complexity of instructions improves throughput, minimizes CPI (Cycles Per Instruction), and enhances processor performance. Modern RISC processors emphasize fast execution, low power consumption, and optimized hardware utilization, making them highly suitable for

Doi:10.48047/ijiee.2026.16.2.60

embedded and real-time applications. The integration of advanced arithmetic techniques such as Vedic multiplication, prefix adders, and hybrid multiplication algorithms further improves computational speed and energy efficiency in contemporary RISC architectures.

- **Pre-fetching:** The process of fetching next instruction or instructions into an event queue before the current instruction is complete is called pre-fetching.
- **Pipelining:** Pipelining allows issuing an instruction prior to the completion of the currently executing one.
- **Superscalar operation:** Superscalar operation refers to a processor that can issue more than one instruction simultaneously.

Vedic mathematics offers an efficient approach for designing high-speed arithmetic circuits, especially in processor-based systems requiring frequent multiplication operations. The Urdhva Tiryakbhyam technique enables parallel generation of partial products, thereby reducing propagation delay and improving computational speed. When integrated with a parallel prefix adder, the ALU achieves faster carry processing and enhanced execution efficiency. In the proposed RISC processor, the hybrid Vedic–Karatsuba multiplier further optimizes performance by balancing speed and hardware complexity.

The designed 16-bit RISC processor is implemented using Verilog HDL and includes essential modules such as the control unit, register bank, program counter, instruction fetch unit, and memory interface supporting a 14-instruction set. Simulation and synthesis results confirm improvements in delay, area utilization, and power efficiency compared to conventional architectures. The proposed design is suitable for embedded and real-time applications requiring high performance with low power consumption. Overall, it provides a scalable and efficient solution for modern processor design.

2.Literature Survey

Mamun B., Shabiul I., and Sulaiman S. presented a single clock cycle MIPS RISC processor design using VHDL for efficient description, verification, simulation, and hardware implementation. The processor supports 32-bit fixed-length instructions based on R-format, I-format, and J-format architectures, along with 32-bit general-purpose registers and memory words. The architecture is divided into five major stages namely instruction fetch, instruction decode, execution, data memory access, and write-back operations, which are coordinated through a centralized control unit. VHDL was utilized for modeling all hardware modules due to its capability to efficiently represent concurrent digital operations. The proposed design was validated through simulation, timing analysis, and hardware realization, demonstrating reliable functionality and effective processor performance.

Takanori M., Satoshi A., and Masaaki I. introduced the Fuce processor architecture based on a continuation-driven multithread execution model derived from dataflow computing principles. The processor aims to combine communication and execution mechanisms to improve thread-level parallelism and concurrency performance. The architecture employs exclusive multi-thread execution units that process events as threads, thereby integrating intra-processor execution with inter-processor communication. The processor was modelled and evaluated using VHDL, and the experimental results demonstrated significant concurrency capability when sufficient parallel threads are available for execution.

Doi:10.48047/ijiee.2026.16.2.60

3. Existing System

The existing processor architecture is designed as a single-chip microprocessor that integrates the major functionalities of a Central Processing Unit (CPU) into a compact and efficient hardware structure. The processor performs arithmetic, logical, control, and data transfer operations through coordinated interaction among several internal modules. The design focuses on achieving reliable processing performance while maintaining simplicity in hardware implementation and reduced system complexity.

The processor architecture consists of several essential functional units, including a register array, Arithmetic Logic Unit (ALU), shifter, program counter, instruction register, comparator, address register, and control unit. The register array contains both 8-bit and 16-bit registers used for temporary data storage and intermediate computations. A 16-bit ALU performs arithmetic and logical operations such as addition, subtraction, comparison, increment, decrement, and logical bit manipulation. The shifter unit supports left and right shift operations required for data processing applications.

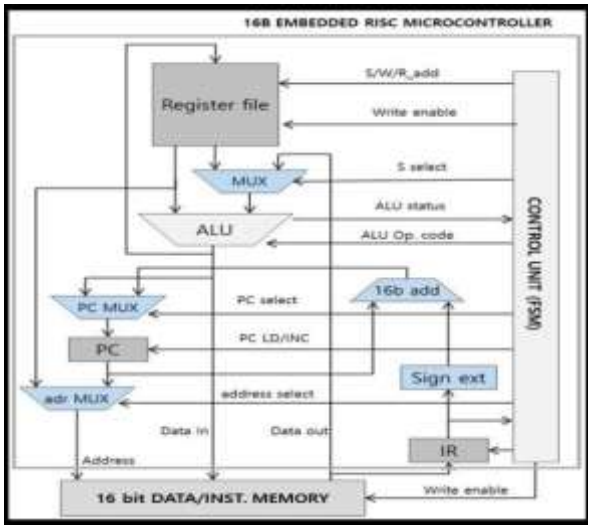

Fig 3.1: RISC Micro Controller

The processor architecture is based on a 16-bit tristate bus system that connects internal modules and manages communication with external memory through address, data, and control buses. It follows a standard instruction cycle involving fetch, decode, execute, and write-back operations controlled by the control unit and program counter. The ALU performs arithmetic and logical operations while results are stored in registers or memory. Despite its structured design, the existing architecture suffers from higher execution delay, limited parallelism, bus congestion, and reduced scalability, leading to lower overall system performance in complex applications.

4. Proposed Method

The proposed system presents the design and implementation of a high-performance 16-bit RISC processor based on Harvard architecture using VHDL. The processor is designed to achieve efficient instruction execution, reduced hardware complexity, improved processing speed, and optimized resource utilization by employing a minimal set of functional units. The Harvard architecture enables separate instruction and data memory paths, allowing simultaneous instruction fetching and data access, thereby improving overall system performance and reducing execution delay.

The proposed 16-bit processor consists of several major functional blocks including a 16-bit Arithmetic Logic Unit (ALU), program counter, instruction register, control unit, flag register, and sixteen 16-bit general-purpose registers. The ALU is capable of performing multiple arithmetic and logical operations such as addition, subtraction, increment, decrement, logical AND, OR, XOR, shifting, and comparison operations. These operations provide efficient data processing capability for embedded and digital computing applications.

Doi:10.48047/ijiee.2026.16.2.60

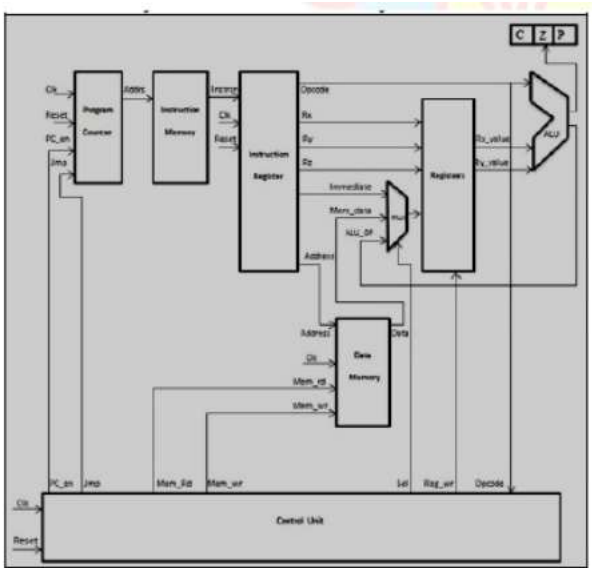

Fig 4.1: Architecture

The proposed 16-bit RISC processor is based on Harvard architecture and uses separate instruction and data memory for efficient operation. Instructions stored in instruction memory are fetched using the address provided by the program counter and loaded into the instruction register for decoding. During decoding, the control unit identifies the opcode, source register, destination register, immediate value, or memory address required for execution.

	Instruction	Opcode				Operation
1	ADD	0	0	0	0	Rd = Rs1 + Rs2
2	SUB	0	0	0	1	If RS1 > RS2 Then Rd = RS1 – RS2 Else Rd = RS2 – RS1
3	AND	0	0	1	0	Rd = Rs1 & Rs2
4	OR	0	0	1	1	Rd = Rs1 | Rs2
5	NOT	0	1	0	0	Rd = ~ Rs
6	XOR	0	1	0	1	Rd = Rs1 ^ Rs2
7	CMP (Equal	0	1	1	0	If Rs1 = Rs2 Then Equal = 1, else Equal =0 If R1 = 0 Then AZ = 1, else AZ=0 If Rs2 =0 Then BZ =1, else BZ =0 If Rs1> Rs2 Then AGB = 1, else AGB = 0 If Rs1< Rs2 Then ALB = 1, else ALB = 0
8	SHIFT LEFT	0	1	1	1	Rd=Rs1 <<1
9	SHIFT RIGHT	1	0	0	0	Rd=Rs1 >>1
10	LOAD	1	0	0	1	Rd = Mem[Rs1]
11	STORE	1	0	1	0	Mem[Rs1] = Rs2
12	JUMP	1	0	1	1	PC = PC+Offset
13	NOP	1	1	0	0	No operation
14	JZ	1	1	0	1	PC = PC+Offset if Rd == 0
15	JNZ	1	1	1	0	PC = PC+Offset if Rd != 1
16	LOAD 8-BIT IMMEDIATE	1	1	1	1	Rd = 8-bit Immediate

Table 1: Operations

The processor is composed of key functional units including a 16-bit ALU, program counter, instruction register, control unit, flag register, and sixteen 16-bit general-purpose registers. It executes arithmetic, logical, and control operations through Fetch, Decode, and Execute states under synchronized control signals. The flag register supports conditional operations by storing Carry, Zero, and Parity status for improved processing efficiency.

ALU:

Inputs “a” and “b” are the two input buses upon which the ALU functions are executed. Output bus “c” returns the outcome of the ALU operation. The ALU can perform arithmetic actions such as add, subtract, increment, decrement, and some logical operations such as AND, OR and XOR, NOT etc.

Opcode (4-bits)	Destination Register (Rd) Address (4-bits)	Source Register 1 Address (4-bits) (Rs1)	Source Register 2 Address (4-bits) (Rs2)

Table 2: Instruction of ALU

Comparator:

The next module described is the comparator. It compares two values and returns either a ‘1’ or ‘0’ depending on the type of comparison requested and the values being compared.

For instance, to compare if inputs “a” and “b” are equal, apply the value eq to port sel. If ports “a” and “b” have the same value, port campout returns ‘1’. If the values are not equal, ‘0’ is returned.

Opcode (4-bits)	Destination Register Address (4-bits) (Rd)	Source Register 1 Address (4-bits) (Rs1)	Source Register 2 Address (4-bits) (Rs2)

Table 3: Instruction of Comparator

Register:

The register array stores intermediate data in sixteen 16-bit registers with dual-read and single-write capability for efficient instruction execution. The shifter, ALU, and fetch unit work together to perform data manipulation, arithmetic operations, and instruction retrieval using the program counter.

Doi:10.48047/ijiee.2026.16.2.60

Decode Unit:

The function of the instruction decode unit is to use the 24-bit instruction provided from the previous fetch unit to index the register file and obtain the register data values. The instruction register, control unit and registers together form decode unit.

Execution Unit:

The execution unit contains an ALU that performs arithmetic and logical operations based on the opcode, supported by a Vedic-based MAC unit for fast multiplication and accumulation. The Urdhva Tiryakbhyam method enables parallel computation, reducing delay and improving overall processor throughput and efficiency.

Urdhava Tiryakbhyam is a Vedic multiplication method that performs vertical and crosswise operations to generate partial products in parallel, reducing delay and improving speed. It offers a simple, scalable architecture with low power and area requirements,

AH AL

BH BL

———

(AH x BH) + (AH x BL + BH x AL) + (AL x BL).

The Vedic multiplier architecture is highly modular and scalable in nature. A 4×4 multiplier is constructed using four 2×2 Vedic multiplier blocks and Ripple Carry (RC) adders to combine partial products efficiently. Similarly, larger multipliers such as 8×8 and 16×16 are implemented using smaller multiplier modules, which simplifies hardware design and improves scalability. During multiplication, partial products and intermediate carries are generated and added in stages to produce the final output. The use of RC adders and modular architecture helps in reducing propagation delay and improving processing speed. Due to its regular structure, the proposed Vedic multiplier achieves better speed, reduced delay, and efficient hardware utilization compared to conventional multiplier architectures.

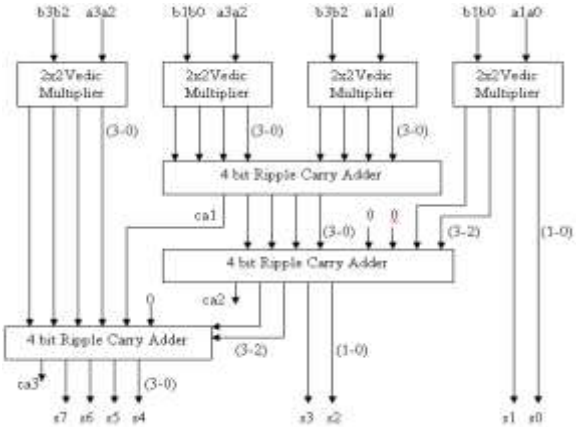

Fig 4.2: Block Diagram of 4x4 bit Vedic Multiplier

The proposed 8×8 Vedic multiplier uses four 4×4 modules with operand decomposition to achieve a modular and scalable design. Partial products are generated in parallel using Urdhava Tiryakbhyam and combined through ripple carry adders to produce a fast and efficient 16-bit output.

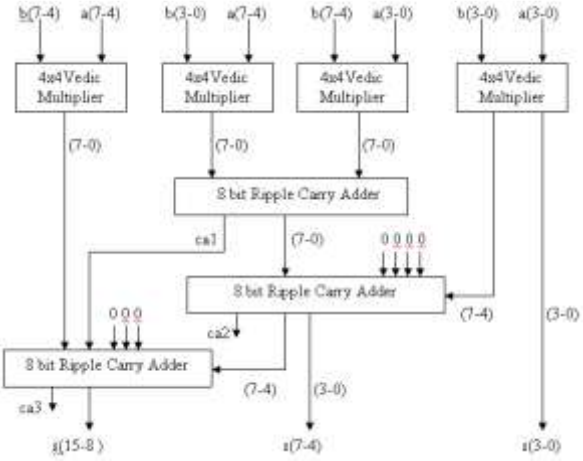

Fig 4.3: Block Diagram of 8x8 bit Vedic Multiplier

The processor improves performance using a parallel prefix adder in the ALU to reduce carry propagation delay and speed up arithmetic operations. A hybrid Vedic–Karatsuba multiplier further enhances efficiency by combining parallel computation with reduced complexity.

Doi:10.48047/ijiee.2026.16.2.60

An efficient MAC unit is used for fast multiply–accumulate operations in DSP applications, while the FSM-based control unit ensures proper synchronization of all operations. The optimized data path and Verilog implementation make the design suitable for FPGA, VLSI, and embedded systems.

5. Results & Discussion

The simulation results show that the proposed RISC core achieves improved speed, reduced area, and lower power consumption compared to conventional architectures due to the use of Vedic multiplication techniques. HDL verification confirms correct functionality with optimized timing, making the design suitable for low-power embedded and real-time applications.

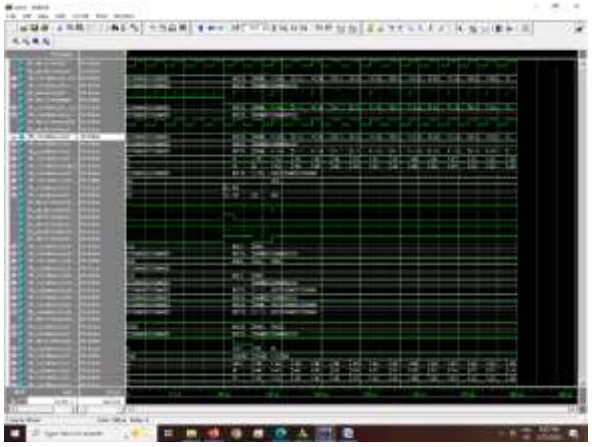

Fig 5.1: Simulation Result

The waveform verifies correct instruction execution with stable timing and improved ALU performance, achieving reduced delay and efficient low-power operation.

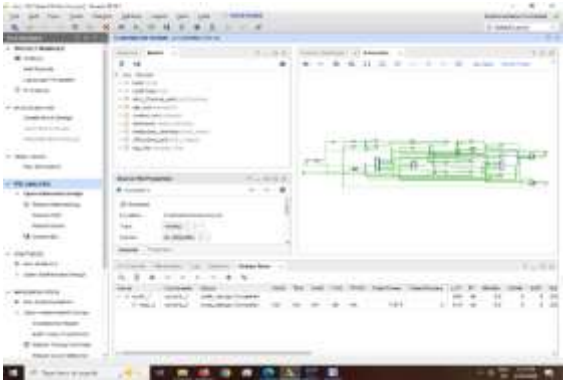

Fig 5.2: RTL schematic of adder

The RTL schematic shows a three-stage adder structure that enables fast carry generation and efficient summation, resulting in reduced delay and improved overall performance.

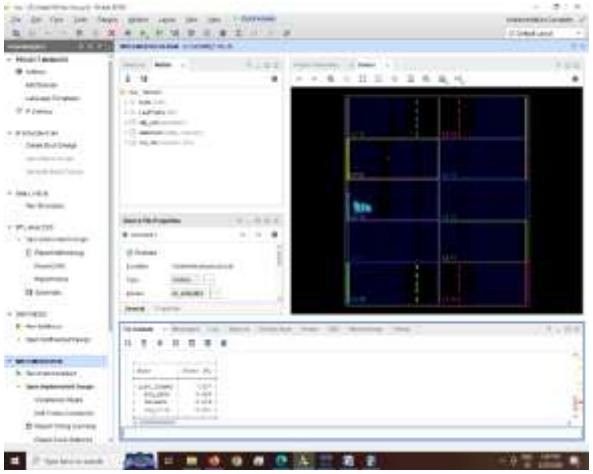

Fig 5.3: Power Report

The delay report indicates that the proposed RISC architecture achieves low latency, confirming faster instruction execution and improved overall processing speed.

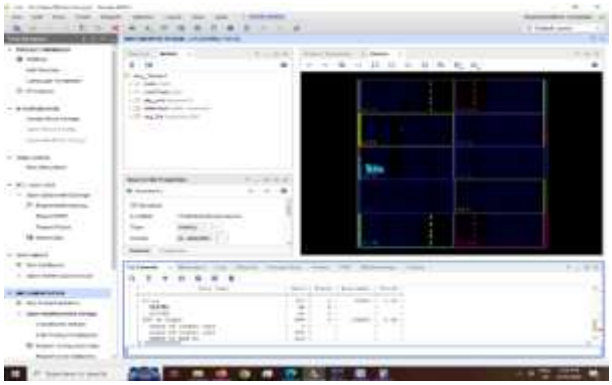

Fig 5.4: Area Report

The area report shows that the proposed RISC processor uses 455 LUTs, indicating efficient hardware utilization and optimized area consumption.

Doi:10.48047/ijiee.2026.16.2.60

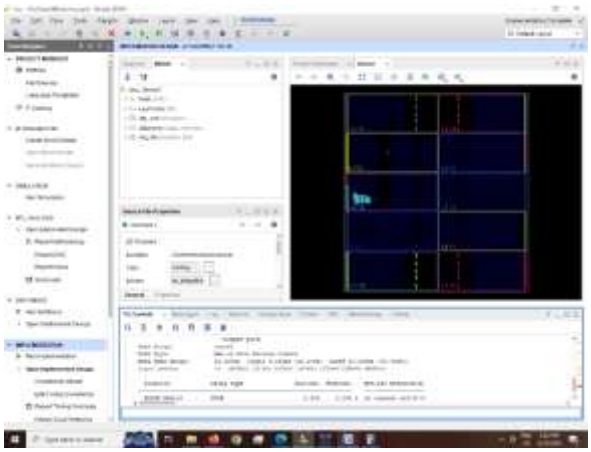

Fig 5.5: Delay Report

The delay report confirms that the processor achieves fast operation with a delay of 23.327 ns, indicating improved timing efficiency and high-speed performance suitable for low-power applications.

	Existing System	Proposed System
DELAY	27.665NS	23.327NS
POWER	0.033MW	0.028MW
AREA	468	455
SPEED	36.14MZ	42.86MZ
PDP	0.9129	0.6531

Table 4: Compression Table

The compression table compares the performance of the existing and proposed RISC processor architectures. The proposed system achieves better timing performance, lower power consumption, reduced hardware area, and improved processing speed, demonstrating the effectiveness of using Vedic Mathematics for efficient and optimized processor design.

6.Conclusion and Future Scope

The proposed 16-bit low-power and low-area RISC processor effectively demonstrates improved performance through the integration of Vedic Mathematics-based arithmetic units and optimized digital design techniques. The use of a parallel prefix adder and hybrid multiplication unit significantly reduces propagation delay and hardware complexity while enhancing overall computational speed. Simulation results confirm reduced power consumption, optimized area utilization, and improved execution efficiency compared to conventional RISC architectures. The design proves to be well-suited for embedded and real-time applications requiring high performance with low energy consumption.

Future Scope: Future work can focus on implementing pipelining and superscalar techniques to enhance processing speed. The architecture can also be extended to higher bit-width designs with cache and floating-point support for advanced computing applications.

References

[1] Ankita Yadav and Varsh Bendre, “Design and Verification of 16-bit RISC Processor Using Vedic Mathematics,” International Conference on Emerging Smart Computing and Informatics, 2021.

[2] Balpande Vishwas V. et al., “Design and Implementation of 16-Bit Processor on FPGA,” Int. J. Adv. Res. Comput. Sci. Softw. Eng., 2015.

[3] Seung Pyo Jung et al., “Design verification of 16-bit RISC processor,” Int. SoC Design Conf., 2008 (widely referenced in later works).

[4] F. Adamec and T. Fryza, “Design-Time configurable processor basic structure,” IEEE DDECS, 2010 (used in extended research literature).

[5] A. Bisoyi, M. Baral and M. K. Senapati, “Comparison of a 32-bit Vedic multiplier with a conventional binary multiplier,” IEEE ICACCCT, 2014.

[6] Mr. Nishant G. Deshpande and Prof. Rashmi Mahajan, “Vedic Mathematics based Multiplier Design for High Speed and Low Power Processor,” IJAREEIE, 2014.

[7] P. Nain and G. S. Virdi, “Multiplier-Accumulator (MAC) Unit,” Int. J. Digital

Doi:10.48047/ijiee.2026.16.2.60

Application and Contemporary Research, vol. 5, no. 3, 2016.
[8] Ram et al., "Area Efficient Modified Vedic Multiplier," IEEE ICCPCT, 2016.
[9] Yogesh M. Motey and Tejaswini G. Panse, "Traditional and Truncation Schemes for Different Multipliers," Int. J. Electron. Comput. Eng., vol. 2, no. 2, 2013 (commonly cited foundation work in later designs).
[10] K. Roy and S. C. Prasad, "Low-Power CMOS VLSI Circuit Design," Wiley, 2000 (standard reference used in 2014–2024 research).
[11] N. H. E. Weste and D. Harris, "CMOS VLSI Design," 4th ed., Pearson, 2011 (still widely used reference in modern VLSI work).
[12] M. Morris Mano, "Computer System Architecture," Pearson, 2007 (core architecture reference).
[13] S. Brown and Z. Vranesic, "Fundamentals of Digital Logic with Verilog Design," McGraw-Hill, 2002 (commonly used design reference).
[14] B. Parhami, "Computer Arithmetic: Algorithms and Hardware Designs," Oxford University Press, 2010 (used in modern arithmetic architecture designs).

Doi:10.48047/ijiee.2026.16.2.60