

Risc-V Processor with Integrated 2-Bit Vedic Multiplier for Low-Power Multiply Operations

Meghana S Department of ECE SSIT, Tumakuru Tumakuru, India meghanas23.2005@gmail.com

Pooja P C Department of ECE SSIT, Tumakuru Tumakuru, India poojapc754@gmail.com

Preksha G P Department of ECE SSIT, Tumakuru Tumakuru, India prekshareddy09@gmail.com

Sharanya Reddy V N Department of ECE SSIT, Tumakuru Tumakuru, India Sharanyareddy6523@gmail.com

Abstract— This paper describes a 32-bit RISC-V processor combined with a 2-bit Vedic multiplier to achieve faster arithmetic processing with lower power consumption. The rapid growth of embedded systems, Internet of Things (IoT) devices, and real-time computing applications has increased the demand for processors capable of performing arithmetic operations efficiently with reduced delay and power consumption. Conventional multiplication techniques often increase propagation delay, hardware complexity, and energy usage, making them less suitable for modern low-power computing systems. To overcome these limitations, the proposed system integrates a Vedic multiplier based on the Urdhva Tiryakbhyam Sutra into the Arithmetic Logic Unit (ALU) of the RISC-V processor. The Vedic multiplication technique performs multiplication through parallel generation of partial products, thereby improving computational speed and reducing execution time. The processor architecture supports pipelined instruction execution, optimized hardware utilization, and scalable design implementation for enhanced processor performance. The proposed processor is designed using Verilog HDL and simulated using Xilinx Vivado tools for functional verification and performance analysis. Simulation results confirm improved multiplication efficiency, reduced propagation delay, and reliable processor operation. The proposed architecture provides an efficient, low-power, and high-performance solution for embedded systems, digital signal processing applications, FPGA implementations, and modern IoT-based computing devices.

Keywords— *IoT, Livestock Monitoring, GPS Tracking, ESP32, Health Monitoring, Telegram Bot, Remote Access.*

INTRODUCTION

The growing requirement for efficient and high-speed computing platforms has accelerated developments in processor architecture and VLSI system design. Modern embedded systems, IoT devices, and digital signal processing applications require processors capable of executing arithmetic operations with high speed and low power consumption. Among these operations, multiplication plays a significant role in determining the overall performance of a processor. Conventional multiplication techniques often result in higher propagation delay, increased hardware complexity, and greater power consumption. RISC-V is an open-source instruction set architecture (ISA) known for its simplicity, scalability, flexibility, and reduced hardware requirements. Due to its modular architecture, RISC-V processors are widely used in embedded and real-time applications.

To improve arithmetic performance, Vedic mathematics-based multiplication techniques are increasingly being adopted in modern processor architectures. The Urdhva Tiryakbhyam algorithm is an efficient Vedic multiplication technique that performs multiplication through parallel generation of partial products, thereby reducing computational delay and improving speed.

The proposed architecture aims to achieve faster multiplication operations, reduced power consumption, and optimized hardware utilization. The processor is designed using Verilog HDL and simulated using Xilinx Vivado tools for functional verification and performance analysis. The proposed system provides improved computational efficiency and is suitable for applications such as embedded systems, IoT devices, FPGA implementations, and low-power computing systems.

A. LITERATURE SURVEY

This survey reviews recent research related to RISC-V processor architectures, Vedic arithmetic methods, low-power ALU structures, and FPGA-oriented processor implementation techniques. Researchers have proposed various methods to improve arithmetic performance, reduce propagation delay, and optimize hardware utilization in modern computing systems. F. Elsabbagh et al. [1] discussed RISC-V Instruction Set Architecture extensions and presented Vortex, an OpenCL-compatible RISC-V GPGPU architecture. Their work highlighted the flexibility and scalability of RISC-V processors for high-performance computing applications. Waterman, Y. Lee, David A. Patterson, and K. Asanovic [2] introduced the RISC-V Instruction Set Manual and explained the micro-architecture design and analysis of RISC-V processors. Their study emphasized reduced hardware complexity, modularity, and open-source advantages of the RISC-V architecture. Pratyush Dwivedi, Krishna Kumar, and Aminul Islam [3] proposed the design and implementation of a 2-bit Vedic multiplier using PTL logic at 16nm technology. Their work demonstrated improved multiplication speed and reduced leakage power compared to conventional multiplication techniques. Anvesh Kumar and Ashish Raman [4] presented a low-power ALU design using ancient Vedic mathematics techniques. The study showed that Vedic arithmetic can significantly reduce delay and improve arithmetic efficiency in digital systems. Prathima and K. Harikishore [5] developed a 2×2 Vedic multiplier with different adders using 90nm CMOS technology. Their research focused on improving speed and reducing power consumption for arithmetic operations. Amrita Nanda and Shreetam Behera [6] proposed the design and implementation of an Urdhva Tiryakbhyam-based fast 8×8 Vedic binary multiplier. The study highlighted parallel partial product generation for achieving faster multiplication performance. Vanitha, S. Nagaraj, and B. Sai Kumar [7] implemented a 16-bit Vedic multiplier using Modified Carry Select Adder (CSA) architecture. Their work improved computational efficiency and reduced propagation delay in multiplication operations. Shri Prakash Dwivedi [8] proposed an efficient multiplication algorithm using the Nikhilam method and demonstrated the advantages of Vedic arithmetic techniques in high-speed computation systems. Researchers have also focused on pipelined RISC-V

processor architectures for improved processor performance. In 2023, a flexible 5-stage pipelined 32-bit RISC processor with dynamic thermal control was proposed [9]. The work emphasized efficient instruction execution, thermal optimization, and enhanced processor reliability. Dr. Girish H., Dr. Mangala Gowri, Dr. Keshava Murthy, and Chetan Naik J. [10] presented the design and validation of a 32-bit RISC-V processor incorporating Vedic mathematics. Their study demonstrated improved multiplication performance, reduced delay, and enhanced processor efficiency using Vedic arithmetic techniques.

B. PROPOSED METHODOLOGY

The proposed approach involves the implementation of a 32-bit pipelined RISC-V processor integrated with a compact 2-bit Vedic multiplier for efficient arithmetic execution. Requirement analysis is carried out to understand the need for a low-power and high-speed processor. The RISC-V architecture is selected due to its flexible and scalable design. The problem of high-power consumption and propagation delay in conventional multiplication techniques is identified and analyzed. A 32-bit pipelined RISC-V processor is designed. The processor includes five pipeline stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB). Basic operations such as addition, subtraction, load, and store instructions are implemented. A 2-bit Vedic multiplier based on the Urdhva Tiryakbhyam Sutra is designed to achieve faster multiplication with reduced delay and improved computational efficiency. The designed 2-bit Vedic multiplier is integrated into the Arithmetic Logic Unit (ALU) of the RISC-V processor. The ALU is modified to support efficient multiplication operations along with other arithmetic and logical functions. The proposed processor architecture supports efficient pipelined execution and optimized hardware utilization for high-speed arithmetic computation. The complete design is implemented using Verilog HDL and simulated using Xilinx Vivado tools. Functional verification is performed using simulation waveforms to confirm the correct operation of the processor. Observed Parameters are Power consumption is measured to evaluate low-power performance. Speed and propagation delay are analyzed to determine computational efficiency. Area utilization is evaluated to study hardware resource optimization.



Fig.1: Flow diagram of 5 stage pipelined RISC-V Processor with integrated 2bit Vedic multiplier

1. Instruction Fetch (IF)

The Instruction Fetch stage is the first stage of the pipelined RISC-V processor. In this stage, the processor fetches the instruction from the instruction memory for execution. The Program Counter (PC) generates the address of the next instruction to be executed. The fetched instruction is then transferred to the next stage for decoding and processing. For example, an instruction such as ADD x1, x2, x3 is fetched from memory during this stage.

2. Instruction Decode (ID)

The Instruction Decode stage interprets the fetched instruction to identify the required operation. The control unit analyzes the opcode and determines the type of operation such as addition, subtraction, multiplication, load, or store. It also identifies the source and destination registers involved in the instruction. After decoding, the required operand data is read from the register file and forwarded to the execution stage for further processing.

3. Execute Unit (EX – ALU)

The Execute stage is the main processing stage of the processor where arithmetic and logical operations are performed using the Arithmetic Logic Unit (ALU). Depending on the instruction, the ALU performs operations such as addition, subtraction, shifting, and multiplication. The integrated 2-bit Vedic multiplier performs fast multiplication operations using the Urdhva Tiryakbhyam algorithm. The parallel generation of partial products reduces propagation delay and improves computational speed and arithmetic efficiency.

4. Memory Access (MEM)

The Memory Access stage is responsible for handling memory-related operations. If the instruction is a load instruction, data is read from Data RAM and transferred to the processor. If the instruction is a store instruction, data is written into memory from the processor registers. For arithmetic instructions that do not require memory access, this stage is skipped logically. This stage ensures efficient communication between the processor and memory unit.

5. Write Back (WB)

The Write Back stage is the final stage of the pipelined RISC-V processor. In this stage, the output generated by the ALU or memory unit is written back into the destination register. This updates the register file with the computed result so that it can be used for future instructions. For example, the result obtained after addition or multiplication is stored in the corresponding register during this stage.

Overall Flow:



Fig.3: Pipeline Flow

The pipeline flow represents the pipelined implementation of a 32-bit RISC-V processor architecture in which multiple instructions are executed simultaneously in different stages to improve processor performance and computational efficiency. Pipelining increases instruction throughput by dividing instruction execution into multiple stages such as Instruction Fetch, Instruction Decode, Execute, Memory Access, and Write Back. The integration of a 2-bit Vedic multiplier into the processor further

improves multiplication speed and reduces stages. propagation delay.

Step 1: IF Stage (Instruction Fetch)

The Instruction Fetch stage is the first stage of the pipelined processor. In this stage, the instruction is fetched from the Instruction Memory (IMEM). The Program Counter (PC) provides the address of the current instruction and is incremented to point to the next instruction in memory. The fetched instruction is then transferred to the IF/ID pipeline register for further processing.

Step 2: IF/ID Register

The IF/ID register acts as a pipeline register between the Instruction Fetch and Instruction Decode stages. It temporarily stores the fetched instruction and the corresponding Program Counter value. This stored information is passed to the decode stage during the next clock cycle for instruction interpretation and control signal generation.

Step 3: ID Stage (Instruction Decode)

In the Instruction Decode stage, the fetched instruction is decoded to determine the operation type and identify the source and destination registers. The Register File (RegFile) reads the required operand data from the registers. Simultaneously, the Control Unit generates the control signals necessary for performing arithmetic, logical, memory, or branching operations in the later stages.

Step 4: ID/EX Register

The ID/EX pipeline register stores the control signals and operand values generated during the decode stage. This register transfers the data and control information to the Execute stage, ensuring smooth pipelined instruction execution between

Step 5: EX Stage (Execution / ALU)

The Execute stage is the main processing stage of the processor where arithmetic and logical operations are carried out using the Arithmetic Logic Unit (ALU). Operations such as addition, subtraction, and shift operations are performed in this stage. The integrated 2-bit Vedic multiplier performs fast multiplication using the Urdhva Tiryakbhyam algorithm. The use of

Vedic arithmetic enables parallel partial product generation, thereby reducing propagation delay and improving multiplication speed. The final arithmetic result is generated during this stage.

Step 6: EX/MEM Register

The EX/MEM pipeline register stores the ALU output result and data that may need to be written into memory during memory operations. This information is forwarded to the Memory Access stage for further processing.

Step 7: MEM Stage (Memory Access)

The Memory Access stage handles memory-related instructions. During a load operation, data is read from Data Memory (DMEM) and transferred to the processor. During a store operation, data from the processor registers is written into memory. If the instruction does not involve memory access, this stage is logically bypassed.

Step 8: MEM/WB Register

The MEM/WB pipeline register stores the data received from memory and the ALU result generated in the execution stage. This data is transferred to the Write Back stage for final register updating.

Step 9: WB Stage (Write Back)

The Write Back stage is the final stage of the pipelined RISC-V processor. In this stage, the final processed result is written into the register file. This completes the execution cycle of the instruction and updates the destination register with the computed value for future operations.

Simulation Flow (Vivado Process)



Fig.1: Simulation Flow

Simulation Flow (Vivado Process)

The complete processor architecture is implemented using Verilog HDL and simulated using Xilinx Vivado tools. The simulation process involves writing the Verilog design, developing a testbench, executing functional simulation, and verifying the output waveforms. Performance parameters such as power consumption, propagation delay, speed, and area utilization are analyzed to evaluate the efficiency of the proposed RISC-V processor integrated with the 2-bit Vedic multiplier.

Step 1: Write Verilog Design

The complete 32-bit pipelined RISC-V processor integrated with a 2-bit Vedic multiplier is designed using Verilog HDL. Different functional modules are created separately and integrated to form the complete processor architecture. The major modules include the Program Counter (PC), Arithmetic Logic Unit (ALU), Register File, Control Unit, and the Vedic Multiplier. The ALU is modified to support multiplication operations using the Vedic multiplication technique. The modular design approach improves flexibility, scalability, and easier

Step 2: Write Testbench

A testbench is developed to verify the functionality and correctness of the processor design. The testbench provides necessary input signals, clock signals, and reset conditions required for simulation. It applies

different instruction sequences and test cases to evaluate the processor behavior under various operating conditions. The generated outputs are compared with expected results to confirm the correct functioning of the processor architecture.

Step 3: Add Files in Vivado

All Verilog design files and testbench files are added into the Xilinx Vivado environment for simulation and verification. The project configuration includes setting the target device, simulation settings, and file hierarchy. Proper project configuration ensures smooth execution of the design and accurate simulation results during processor verification.

Step 4: Run Behavioral Simulation

Behavioral simulation is executed in Xilinx Vivado to test the processor design. The simulation runs using the testbench inputs and verifies the working of the RISC-V processor and 2-bit Vedic multiplier.

Step 5: Check Waveform Output

After running the simulation in Xilinx Vivado, the output is checked using waveforms to verify the correct working of the processor. Different signals such as clock, Program Counter (PC), ALU output, and register values are observed in the waveform window.

Comparison of Sutras

LIST OF SUTRAS	NIKHILAM SUTRA	URDHVA TIRYAKBHYAM SUTRA	ANURUPYENA SUTRA	EKADHIKENA PURVENA SUTRA
WORKING PRINCIPLE	Works best when numbers are close to a base (10, 100, 1000, 2 ⁿ). Converts multiplication into base subtraction + multiplication	Generates partial products in parallel using vertical and crosswise multiplication. Suitable for any-sized numbers (8-bit, 16-bit, 2-bit, 64-bit).	Squaring numbers. Constant multipliers ($\times 9$, $\times 11$, $\times 19$, $\times 99$ etc.) Uses "one more than previous digit" logic	Uses scaling to convert difficult multiplication into simpler forms. Example: $(23 \times 22) \rightarrow$ scale both $\rightarrow (46 \times 44) / 4$
USES	Low-area, low-power multipliers. Not suitable for general-purpose multipliers	used in high-speed FPGA/ASIC multipliers.	Used in special-case multipliers (DSP filters, constant multiplication)	Used for scaled multipliers. Only helpful when operands have proportional relationships

Table: Comparison of Sutras

1.SYSTEM DESIGN

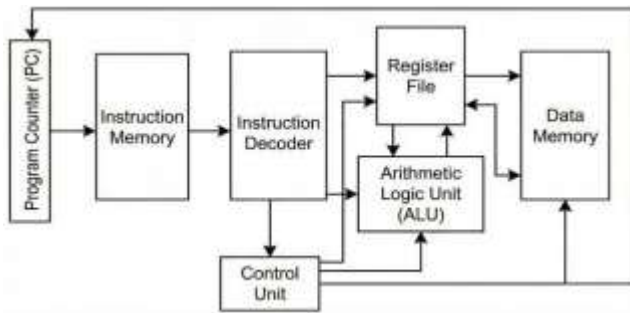


Fig.2: Block diagram of RISC- V Processor.

The block diagram represents the overall architecture of the 32-bit RISC-V processor integrated with an Arithmetic Logic Unit (ALU), register file, control unit, and memory units. It shows how instructions and data move through different parts of the processor during execution.

Program Counter (PC):

The Program Counter stores the address of the next instruction to be executed. It continuously updates after every instruction and sends the address to the instruction memory.

Instruction Memory:

The Instruction Memory stores all the program instructions. Based on the address received from the Program Counter, the required instruction is fetched and sent to the Instruction Decoder.

Instruction Decoder:

The Instruction Decoder interprets the fetched instruction and identifies the type of operation to be performed, such as addition, subtraction, multiplication, load, or store operation. It also identifies source and destination registers. The Register File stores temporary data and operands required for execution. It provides input data to the ALU and stores the output result after execution.

Data Memory:

The Data Memory is used to store and retrieve data during load and store operations. The processor reads

data from memory or writes data into memory whenever required.

2-bit Vedic Multiplier.

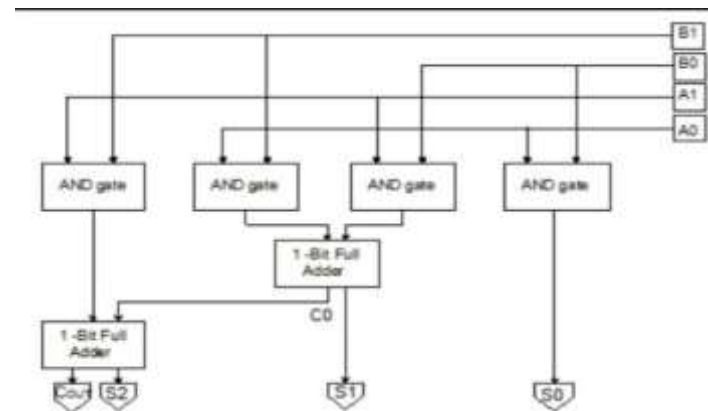


Fig.3: Block diagram 2-bit Vedic Multiplier.

A 2-bit Vedic multiplier is a digital circuit used to multiply two 2-bit binary numbers. It is designed using the Urdhva Tiryakbhyam method from Vedic mathematics, which means “Vertically and Crosswise” multiplication. This method performs multiplication faster by generating partial products in parallel. In this block diagram, the inputs are represented as A1, A0 and B1, B0, where A and B are two 2-bit binary numbers. The multiplication process starts using AND gates. The AND gates generate partial products by multiplying individual bits of the input numbers. The first AND gate generates the least significant bit output S0. Other AND gates generate intermediate partial products required for further calculation. Half adders are used to add these partial products together. The first half adder produces output S1 and carry output C0. Another half adder adds the carry and remaining partial products to generate outputs S2 and S3. The final outputs S3, S2, S1, and S0 together form the multiplication result of the two 2-bit binary numbers. The Vedic multiplier reduces propagation delay and improves multiplication speed compared to conventional multiplication methods. Due to its simple structure and fast operation, it is suitable for low-power and high-speed processor applications.

32-bit RISC-V Processor with 2-bit Vedic Multiplier.

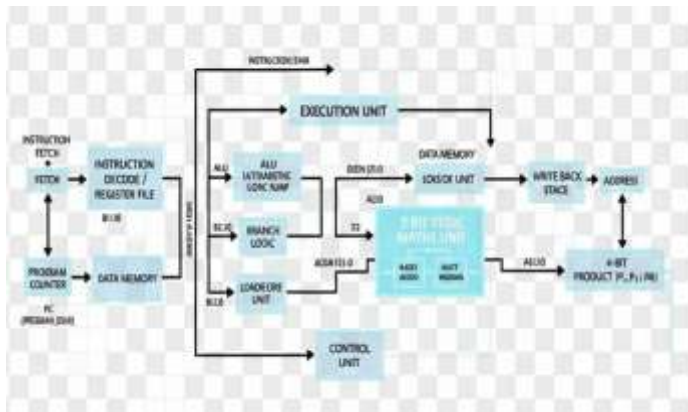


Fig.4: Block diagram 32-bit RISC-V Processor with 2-bit Vedic Multiplier.

This block diagram shows the working of a 32-bit RISC-V processor integrated with a 2-bit Vedic multiplier. The processor is designed to perform arithmetic and logical operations with high speed and low power consumption. The process starts from the Program Counter (PC), which gives the address of the next instruction. The instruction is fetched from the Instruction Memory and

sent to the Instruction Decode stage. In this stage, the processor understands the instruction and reads the required data from the register file. After decoding, the instruction goes to the Execution Unit. The Arithmetic Logic Unit (ALU) performs operations such as addition, subtraction, shifting, and multiplication. The integrated 2-bit Vedic multiplier performs multiplication faster using Vedic mathematics techniques. The Control Unit controls the flow of data between all stages of the processor. If memory access is needed, data is transferred to the Data Memory stage. Finally, the result is stored back in the register during the Write Back stage. The use of the Vedic multiplier reduces delay, improves multiplication speed, and decreases power consumption. This makes the processor suitable for embedded systems, FPGA applications, and low-power computing devices.

1. SYSTEM IMPLEMENTATION AND RESULTS



Fig a: Waveform of PC

he P Program Counter (PC) stores the address of the next instruction to be executed. In the waveform, the PC value increases step by step, usually by +4 for every clock cycle. This shows normal sequential execution of instructions. If a branch or jump instruction occurs, the PC value suddenly changes to a different address using an offset value. Therefore, the waveform shows both regular step increments and sudden jumps during branch operations.

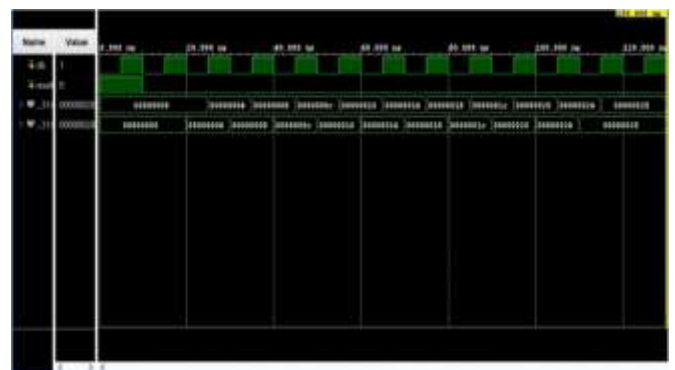


Fig b: Waveform of IMEM

Instruction Memory (IMEM) stores program instructions. The Program Counter (PC) gives the address to IMEM, and the corresponding instruction is read. When the PC changes, the instruction output also changes. The waveform shows instruction fields like opcode, rs1, rs2, rd., and immediate values.



Fig c: Waveform of regfile

The register file stores and reads data in the processor. When the write enable signal is active, data is written into the selected register during the clock pulse. The source registers rs1 and rs2 provide output data through rd1 and rd2. The waveform shows successful read and write operations.



Fig d: Waveform of ALU

The ALU performs arithmetic and logical operations such as ADD, SUB, AND, OR, and MUL. It receives input data from registers and performs operations based on the control signal. The result changes according to the selected operation. If the output becomes zero, the zero flag is set to 1.



Fig e: Waveform of Control

The control unit generates control signals for different processor operations. It receives the instruction opcode and activates signals like regwrite, memread,

memwrite, alusrc, and branch. Different instructions generate different control signals. The waveform verifies correct control signal generation for processor operations.

Fig f: Waveform of DMEM

Data Memory (DMEM) stores data during load and store operations. When memwrite = 1, data is written



into memory, and when memread = 1, stored data is read from memory. The waveform shows correct memory read and write operations.



Fig g: Waveform of if_id_reg

The IF/ID register is a pipeline register between the Instruction Fetch and Instruction Decode stages. It stores the instruction and PC value at every clock edge and forwards them to the decode stage. The waveform shows proper synchronization of instructions between stages.



Fig h: Waveform of id_ex_reg

The ID/EX register transfers decoded data and control signals from the Decode stage to the Execute stage. Inputs remain stable until the next clock cycle, ensuring smooth pipelined operation. The waveform shows correct data transfer and reduced control hazards.



Fig i: Waveform of ex_mem_reg

The EX/MEM register stores ALU results and memory control signals between the Execute and Memory stages. It forwards execute-stage outputs to the memory stage while maintaining timing synchronization. The waveform confirms proper forwarding of data.



Fig j: Waveform of mem_wb_reg

The MEM/WB register transfers memory data or ALU results to the Write Back stage. It helps send final processed data back to the register file. The waveform shows correct data forwarding and proper pipeline flow.



Fig k: Waveform of if stage

The Instruction Fetch (IF) stage fetches instructions from instruction memory using the Program Counter (PC). The PC generates the instruction address and increases by 4 for the next instruction. The waveform shows correct sequential instruction fetching and PC updating.

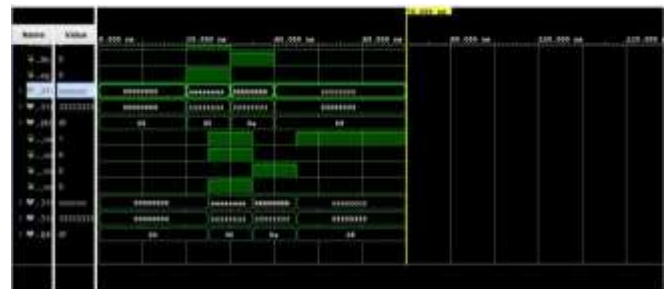


Fig l: Waveform of id stage

The Instruction Decode (ID) stage decodes the instruction opcode, reads register operands, generates immediate values, and produces control signals. The waveform shows correct instruction decoding and successful register reading.



Fig m: Waveform of ex stage

The Execute (EX) stage performs arithmetic and logical operations using the ALU. For multiplication instructions, the integrated 2-bit Vedic multiplier performs multiplication quickly. The result is then forwarded to the EX/MEM register. The waveform confirms fast execution and proper Vedic multiplier operation.



Fig n: Waveform of mem stage

The Memory (MEM) stage handles load and store operations. The ALU result provides the memory address, and the data memory performs read or write operations. The processed data is sent to the MEM/WB register. The waveform shows correct memory access and timing.



Fig o: Waveform of WB stage

The Write Back (WB) stage writes the final result from the ALU or memory back into the destination register. Register write enable controls the update process. The waveform verifies correct result storage and successful completion of pipeline execution.



Fig p: Waveform of RISC-V Processor

This waveform shows the overall operation of the pipelined RISC-V processor. Instructions pass through

all five stages, while pipeline registers synchronize the operations. The ALU and Vedic multiplier perform arithmetic operations correctly, and memory/register updates occur successfully.



Fig q: Waveform of Pipelined RISC-V Processor with Integrated 2-bit Vedic Multiplier

This waveform represents the complete simulation of the pipelined RISC-V processor. The Program Counter (PC) increases sequentially, instructions are fetched correctly, and the ALU performs operations during execution. The integrated 2-bit Vedic multiplier helps achieve faster multiplication with proper synchronization between pipeline stages.

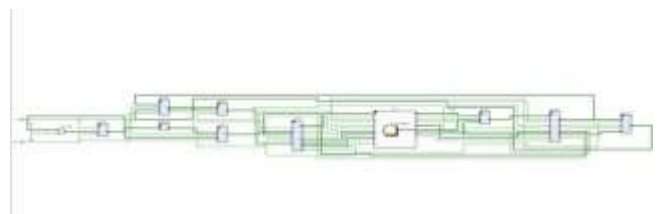


Fig r: Schematic of RISC-V Processor

The schematic shows the complete hardware design of the pipelined RISC-V processor integrated with a 2-bit Vedic multiplier. It includes all five stages: Instruction Fetch, Decode, Execute, Memory, and Write Back. Pipeline registers transfer data between stages, while the Vedic multiplier improves multiplication speed and efficiency.

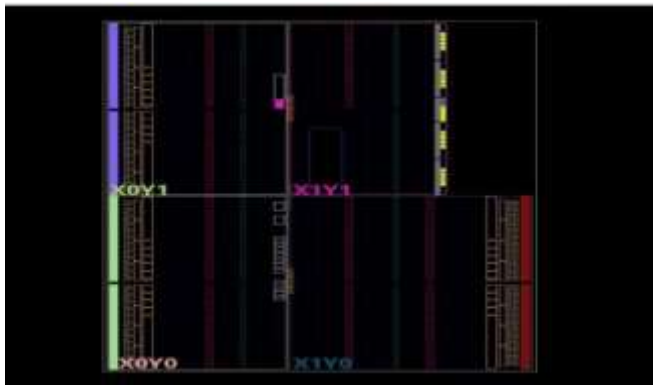


Fig r: Netlist

The netlist represents the synthesized hardware connections of the processor design. It contains logic gates, flip-flops, multiplexers, arithmetic blocks, and interconnections generated from the HDL code. The waveform confirms successful synthesis and shows that the processor is ready for FPGA implementation.

Observed Parameters

Observation	Without integrate	With integrate
CPU Time	2 SEC	6 SEC
Increased Elapsed Time	5 SEC	13 SEC
Increased Memory	2144 MB	1362 MB
Reduced Gain	00%	30%

I. ADVANTAGES

Low Power Consumption – The 2-bit Vedic multiplier reduces power usage, making the processor suitable for low-power applications.

High Processing Speed – Parallel multiplication reduces delay and increases operation speed.

Efficient Hardware Usage – The design uses less hardware complexity and optimizes resources.

Improved Processor Performance – The Vedic multiplier improves overall processor efficiency and computation capability.

Reduced Computational Delay – Faster multiplication decreases execution time for arithmetic operations.

Simple and Scalable Design – The compact architecture can be extended to higher-bit multipliers in future.

Suitable for Embedded Systems – Low power and

efficient performance make it useful for portable and embedded devices.

II. LIMITATIONS

Limited to 2-bit Multiplication – Supports only simple 2-bit multiplication operations.

Scalability Issues – Increasing multiplier size may increase hardware complexity.

Limited Instruction Support – The processor may support only basic RISC-V instructions.

Simulation-Based Verification – Performance is mainly tested through simulation, not real hardware.

Area Overhead – Larger multiplier designs may increase chip area and cost.

Limited Application Scope – Mainly useful for educational and low-power research applications.

Dependence on Simulation Tools – Accuracy and analysis depend on the simulation software used.

III. APPLICATIONS

Embedded Systems – Used in low-power embedded devices for efficient processing.

Portable Electronic Devices – Suitable for battery-operated devices because of low power consumption.

Digital Signal Processing (DSP) – Faster multiplication improves signal processing operations.

IoT Devices – Can be used in Internet of Things applications requiring compact and efficient processors.

Educational and Research Projects – Helpful for studying pipelined RISC-V architecture and Vedic multiplication techniques.

Arithmetic Processing Systems – Useful in systems requiring fast arithmetic calculations.

FPGA-Based Designs – Can be implemented on FPGA platforms for hardware testing and development.

CONCLUSION

This work demonstrated the implementation of a RISC-V processor integrated with a 2-bit Vedic multiplier for efficient and low-energy multiplication operations. By using the Urdhva Tiryakbhyam Vedic multiplication technique, the processor achieved faster arithmetic computation with reduced delay and improved performance compared to conventional multiplication methods. The proposed design successfully demonstrated the advantages of

combining the simplicity and flexibility of the RISC-V architecture with the high-speed capability of Vedic mathematics. Simulation and analysis showed improvements in power efficiency, processing speed, and hardware utilization, making the system suitable for modern embedded and digital signal processing applications. Overall, the project highlights the potential of Vedic multiplier-based processor design in developing compact, energy-efficient, and high-performance computing systems for future real time applications.

REFERENCES:

1. F. Elsabbagh, B. Tine, P. Roshan, E. Lyons, E. Kim, D. E. Shim, L. Zhu, S. K. Lim, and H. Kim, "Vortex: OpenCL Compatible RISC-V GPGPU," arXiv preprint arXiv:2002.12151, 2020.
2. Y. Lee, D. A. Patterson, and K. Asanović, "The RISC-V Instruction Set Manual, Vol. I: Base User-Level ISA," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2011-62, May 2013.
3. Pratyush Dwivedi, Krishna Kumar, and Aminul Islam, "Design and Implementation of 2-bit Vedic Multiplier at 16nm Using PTL Logic," in International Conference on Microelectronics, computing and Communications (MicroCom), 2016.
4. Anvesh Kumar and Ashish Raman, "Low Power ALU Design by Ancient Mathematics," 2010.
5. N. Prathima and K. Harikishore, "A 2×2 Bit Vedic Multiplier with Different Adders in 90nm CMOS Technology," International Journal of Engineering Research and Applications, vol. 3, pp. 1832–1837, 2013.
6. Amrita Nanda and Shreetam Behera, "Design and Implementation of Urdhva Tiryakbhyam Based Fast 8×8 Vedic Binary Multiplier," 2014.
7. B. Vanitha, S. Nagaraj, and B. Sai Kumar, "Implementation of 16-Bit Vedic Multiplier Using Modified CSA," 2023.
8. Shri Prakash Dwivedi, "An Efficient Multiplication Algorithm Using Nikhilam Method," July 10, 2013.
9. M. Lokesh and Manoj Kumar, "Design and Implementation of 32-bit RISC Processor with Flexible 5-Stage Pipelining and Dynamic Thermal Control," in International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), IEEE, pp. 846–849, 2023.
10. Dr. Girish H., Dr. Mangala Gowri, Dr. Keshava Murthy, and Chetan Naik J., "Design and Validation of a 32-Bit RISC-V Processor Incorporating Vedic Mathematics," Gradiva Review Journal, vol. 8, issue 10, pp. 107–115, 2022.