Design and Implementation of Low Power RISC-V Processor

SANOFER NISHA.M.S
Department of Electrical and Electronics Engineering
Sri Venkateswara College of Engineering
Pennalur, 602117
2022ee0680@svce.ac.in

SHAKTHIVEL.R
Department of Electrical and Electronics Engineering
Sri Venkateswara college of Engineering
Pennalur, 602117
2022ee0738@svce.ac.in

VAISHNAVI.A
Department of Electrical and Electronics Engineering
Sri Venkateswara College of Engineering
Pennalur, 602117
2022ee0682@svce.ac.in

DR.T. ANNAMALAI (Associate Prof.)
Department of Electrical and Electronics Engineering
Sri Venkateswara College of Engineering
Pennalur, 602117
annamalai@svce.ac.in

I. ABSTRACT

The increasing demand for intelligent edge devices has driven the need for energy-efficient and high-performance processors capable of executing machine learning tasks in real time. This paper presents the design and implementation of a low-power RISC-V processor optimized for real-time machine learning applications. The processor is based on the open-source RISC-V instruction set architecture (ISA), enabling design flexibility and extensibility for domain-specific optimizations. To accelerate machine learning workloads, custom hardware accelerators are integrated, including multiply-accumulate (MAC) units and activation function modules such as ReLU, allowing efficient execution of matrix multiplications and nonlinear operations. The design is described using Verilog HDL, simulated using ModelSim, and synthesized on an FPGA platform to evaluate area utilization, timing performance, and power consumption. The processor also incorporates a dual-port on-chip SRAM, DMA controllers, and standard peripherals such as UART, SPI, and I2C for seamless data exchange with sensors and edge devices. Experimental results demonstrate that the proposed architecture achieves significant power savings while maintaining high throughput and low-latency performance compared to conventional RISC-based designs. The modular and scalable nature of the processor makes it a suitable candidate for a wide range of edge computing and embedded AI applications, including IoT devices, wearable systems, and smart cameras. Additionally, the processor supports easy integration of future custom instructions and AI accelerators, providing a flexible platform for continued innovation in energy-efficient edge AI.

Key Terms: RISC-V architecture, low-power design, hardware accelerator, real-time machine learning, FPGA implementation, edge computing, AI inference, embedded systems.

II. INTRODUCTION

The continuous evolution of embedded systems and edge computing devices has driven the need for efficient, high-performance processing architectures that can operate under strict power and area constraints. Modern applications such as IoT devices, wearable electronics, and real-time monitoring systems require processors that deliver sufficient computational performance while consuming minimal energy, ensuring longer battery life and reduced thermal dissipation. Insights on embedded digital system design principles and hardware power constraints are referenced from *Digital System Design using Verilog* by Charles Roth et al. (Reference 1).

Traditional general-purpose processors often fail to meet these requirements due to their high transistor count and complex pipelines, resulting in excessive energy consumption. The emergence of the open-source **RISC-V ISA** offers a promising solution, enabling designers to customize instruction sets and hardware structures for improved efficiency. Fundamental knowledge on the RISC-V architecture and instruction formats used in this work is sourced from Patterson & Hennessy (Reference 2), along with architectural overview from ElProcus (Reference 5).

This project focuses on the design and implementation of a **low-power RISC-V processor**, adopting the RV32I/IM instruction set to provide essential integer operations while enabling multiply-divide extensions when required. The modular Verilog HDL design methodology follows guidelines from Roth et al. (Reference 1), while the pipelined architecture and RTL implementation structure are based on the 5-stage RISC-V pipeline model discussed in Gaurav Srivastav (Reference 3) and pipeline optimization techniques from Yan & Michelogiannakis (Reference 4).

Functional verification is performed using ModelSim to validate correct execution of instructions and detect architectural hazards before hardware realization. To evaluate real-world feasibility, the processor is synthesized and deployed on an FPGA, enabling analysis of resource utilization, timing, and power performance.

The five-stage pipeline — **Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write-Back (WB)** — ensures efficient execution flow while keeping hazard overhead low. Reference 4 contributes insights on hazard mitigation and pipeline delay reduction. Low-power design strategies such as minimized switching activity, reduced memory accesses, and simplified combinational logic are adopted to align with TinyML design motivations from Partha Pratim Ray (Reference 6), especially for battery-operated edge AI devices.

In conclusion, this project demonstrates that an optimized, low-power RISC-V processor can effectively support embedded and edge-oriented workloads while maintaining high energy efficiency. The FPGA-based prototype serves as a scalable foundation for future enhancements such as TinyML acceleration, on-chip neural compute engines, and application-specific instruction extensions.

III. RELATED WORK

Recent research in energy-efficient processor architectures has focused on integrating machine learning capabilities into embedded and edge computing platforms. Traditional CISC based architectures such as x86 and even some ARM-based systems deliver high performance but often suffer from increased power consumption and limited flexibility for hardware customization. To overcome these challenges, the RISC-V architecture has gained significant attention due to its open-source nature, modular instruction set, and ability to support application-specific hardware extensions. Several studies have explored RISC-V cores optimized for low power and high performance. *Partha Pratim Ray (2021)* presented an overview of TinyML systems that highlight the need for lightweight machine learning on microcontrollers. *Gaurav Srivastav (2021)* implemented a five-stage pipelined RISC-V processor to improve instruction throughput. Similarly, *Yan and Michelogiannakis (2016)* discussed RISCV pipeline implementations focused on efficient instruction flow, while *Patterson and Hennessy* emphasized the importance of hardware-software co-design in modern processor design. However, most existing works primarily target general-purpose computing rather than machine learning acceleration.To bridge this gap, the present work focuses on the design and FPGA implementation of a low-power RISC-V processor integrated with machine learning accelerators for real-time inference. The objective is to achieve an optimal balance between computational efficiency, power consumption, and processing speed, making it suitable for edge AI applications .

IV. PROPOSED METHODOLOGY

A. Project Objective

The proposed system focuses on the design and implementation of a low-power RISC-V processor tailored for embedded and edge computing applications. The primary objective is to achieve high instruction throughput while minimizing energy consumption, making the processor suitable for battery-operated and resource-constrained environments such as IoT devices, portable electronics, and sensor-based automation systems. The design ensures that efficiency, performance, and reliability remain the core strengths of the architecture.

B. Design Methodology Overview

The complete development strategy is divided into three major phases:

- Processor architecture development and pipelined structure
- Verilog HDL-based modular hardware design
- FPGA synthesis, implementation, and real-time hardware validation

This organized approach ensures systematic development, ease of debugging, and scalability for advanced design upgrades in the future.

C. Pipeline Architecture

The processor utilizes a five-stage pipelined RISC-V architecture, including:

- Instruction Fetch(IF)
- Instruction Decode(ID)
- Execute (EX)
- Memory Access (MEM)
- Write Back (WB)

The pipeline enhances parallel instruction execution and improves throughput. Each pipeline stage is carefully optimized to reduce data propagation delays, shorten the critical path, and lower switching activity, directly contributing to reduced power consumption and improved timing performance.

D. Modular Verilog HDL Implementation

The processor is fully designed using Verilog HDL, enabling clean module separation and reuse. Major modules include:

- Arithmetic Logic Unit (ALU) supporting RV32I arithmetic, logical, shift, and compare operations
- Register file designed for low-latency dual-port access
- Instruction and data memory using synchronous Block RAMs
- Control Unit ensuring proper timing, operation sequencing, and instruction handling.

E. Control and Hazard Management

To ensure reliable pipeline execution, the control unit incorporates hazard-handling mechanisms such as:

• Forwarding methods to resolve data hazards with minimal delay
• Stall insertion during unavoidable conflicts like load-use hazards
• Basic branch prediction to reduce branch penalties during control hazards

These techniques prevent unnecessary flushing, reduce idle pipeline stages, and avoid energy waste.

F. Low-Power Hardware Optimization Techniques

The design integrates multiple hardware techniques to reduce dynamic power consumption, including:

• Clock gating to disable unused modules
• Minimizing logic depth to reduce propagation delays
• Short and optimized data paths to reduce toggling activity
• Efficient memory access with reduced switching frequency

These techniques ensure significantly lower power usage compared to unoptimized baseline architectures.

G. Verification and Simulation

ModelSim is used for functional verification of the processor. Testbenches simulate various instructions, pipeline operations, branch operations, memory transactions, and exceptional test cases to ensure correctness. Simulation validates:

• Accurate instruction execution
• Proper control signal generation
• Effective hazard handling
• Stable pipeline behavior under diverse scenarios

This ensures a robust and fully verified design before real hardware implementation.

H. FPGA Implementation and Evaluation

Following simulation, the processor is synthesized on a suitable FPGA platform such as the Xilinx Artix-7. Performance is evaluated based on parameters including:

• Power consumption under active workloads
• Maximum achievable clock frequency
• Area utilization in terms of LUTs, flip-flops, and BRAMs
• Timing delays and critical path analysis

The hardware implementation confirms both functional correctness and efficiency enhancements achieved through architectural and power-optimized design techniques.

I. Scalability and Future Enhancement Scope

The processor architecture supports future extensions without major redesign. Potential enhancements include:

• Adding RISC-V extensions such as M (Multiplication/Division), F (Floating Point), or C (Compressed Instructions)
• Integration of peripheral interfaces for real-world embedded applications
• Inclusion of hardware accelerators for machine learning, DSP, and cryptography

The flexibility of the design makes it highly adaptable for ongoing technological advancements

V. SYSTEM DESIGN

The proposed system is centered around a **low-power RISC-V processor** implemented on a Field Programmable Gate Array (FPGA), specifically designed to serve energy-constrained embedded applications such as IoT devices and real-time edge computing systems.

The processor utilizes a **classic five-stage pipeline architecture**, which includes:

• **Instruction Fetch (IF)** – retrieves instructions from program memory
• **Instruction Decode (ID)** – interprets opcode and selects register operands
• **Execute (EX)** – performs arithmetic/logical operations using the ALU
• **Memory Access (MEM)** – interacts with data memory for load/store operations
• **Write Back (WB)** – updates the destination register

This parallel execution model enhances instruction throughput while maintaining low switching activity, contributing to reduced dynamic power consumption.

At the execution core, the Control Unit plays a fundamental role in hazard-free, correctly sequenced execution. During the ID stage, it decodes the instruction opcode and dispatches pipeline control signals forward into the EX, MEM, and WB stages through pipeline registers. It also generates immediate selection signals, ALU operation codes, branch decision flags, memory enable signals ,and write-back selection enables. To maintain pipeline correctness, the Control Unit integrates hazard detection logic that stalls the pipeline when read-after-write conflicts occur, and it triggers data forwarding paths to minimize performance penalties. For branch instructions, it works in conjunction with the ALU to detect zero or comparison conditions, allowing the pipeline to flush incorrectly fetched instructions if a branch is taken. By controlling stalls, flushes, and forwarding, the Control Unit ensures smooth and synchronized operation of all pipeline stages without corrupting architectural state.

An optimized **Arithmetic Logic Unit (ALU)** handles operations such as addition, subtraction, logical bitwise calculations, shifts, and comparisons, which are essential for general computation tasks. The schematic diagram shows dedicated combinational logic paths for ADD, SUB, AND, OR, XOR, logical shift left, logical shift right, arithmetic shift right, and comparison operations. Each sub-result feeds into a multi-input multiplexer whose select lines originate from the control unit. A dedicated equality comparator generates a zero flag used during branch

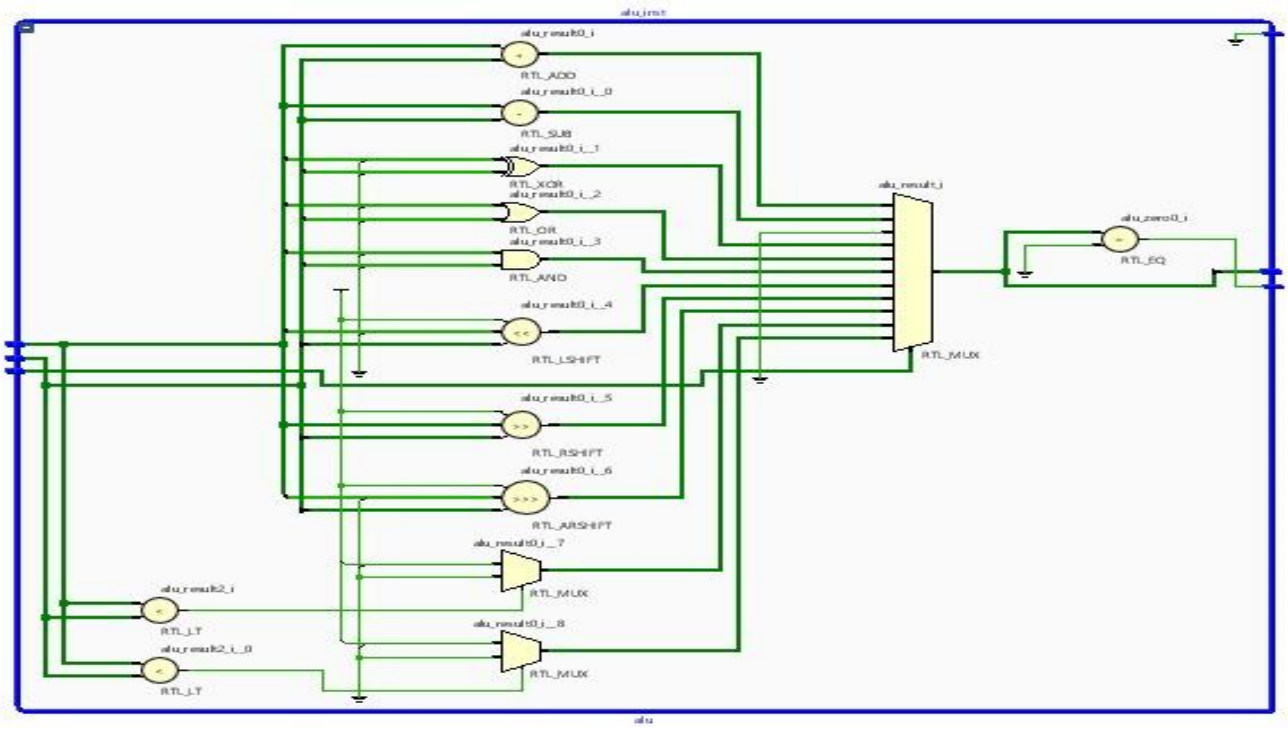

Fig 1. Schematic diagram of ALU

instructions. The ALU organization maximizes performance by computing all operations in parallel, while the mux selects only one final output. This approach minimizes propagation delay, enabling higher clock speeds in FPGA synthesis.

The complete processor is described modularly using Verilog HDL, providing flexibility for feature enhancement, custom extensions, or peripheral integration such as UART, SPI, or GPIO interfaces. Simulation is performed using ModelSim to verify opcode decoding, ALU correctness, memory behavior, and control sequencing. Post-verification, the processor is synthesized on a Xilinx Artix-7 FPGA, where key performance metrics such as LUT consumption, maximum achievable frequency, and power usage are evaluated. The achieved results demonstrate that the processor is compact, energy-efficient, and suitable for beginner-to-intermediate embedded workloads. Future improvements may include multiplier/divider extensions, pipeline support, or minimal AI inference accelerators.

The Immediate Generator is responsible for extracting and formatting constant values embedded within instructions. Based on the opcode and instruction type, different instruction bit-fields are tapped and then sign-extended to 32 bits. The reference design uses multiple multiplexers to select fields corresponding to I-type, S-type, B-type, U-type, and J-type immediates. A small instruction-type latch stores classification bits that determine which multiplexer path becomes active. All immediate variants are then routed through a final mux stage, ensuring only the correct value is forwarded to the ALU input. Ground references visible in the schematic ensure correct sign-extension behavior for negative constants. This modular approach simplifies decoding and allows direct hardware reuse across many instruction formats.

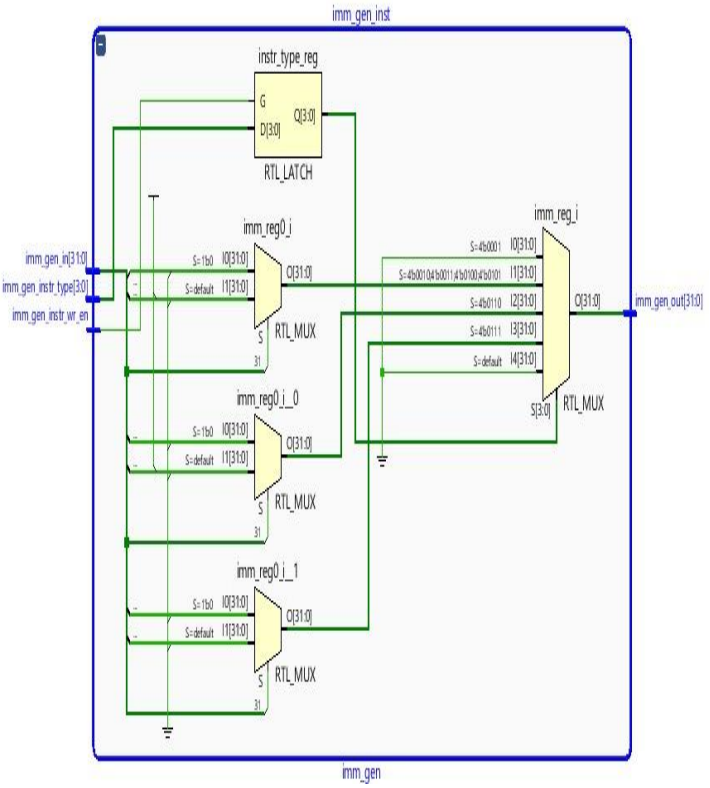

Fig 2. Schematic diagram of Immediate Generator

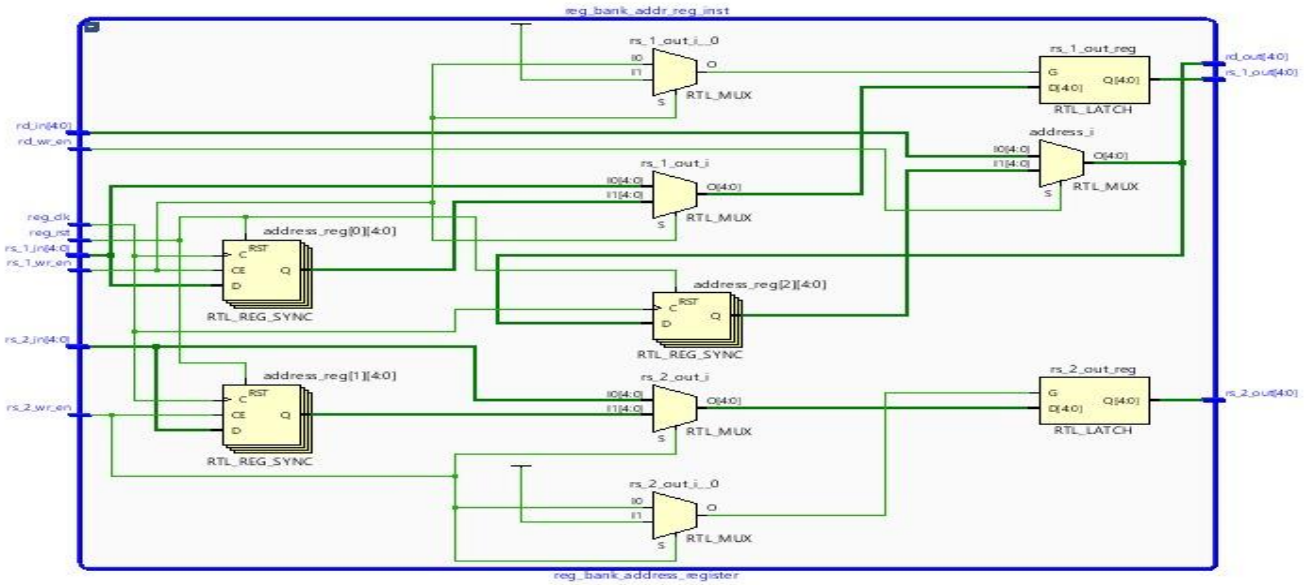

Fig 3. Schematic diagram of Register Bank

The Register Bank utilizes synchronous register blocks, providing dual-read and single-write access within one cycle. The attached schematic shows independent read address muxes and synchronous write-enable logic for each register file entry. Output latches stabilize register output values for downstream modules. This organization ensures deterministic timing on FPGA platforms. The design also includes separate address registers to handle simultaneous operand fetching for ALU execution while writing the result back to another register. Consistent with RISC-V specification, register x0 is hard-wired to zero, preventing writes via gated logic paths.

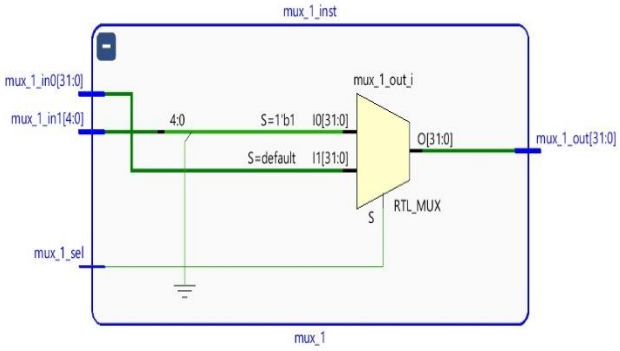

Fig 4.Schematic diagram of MUX

Multiplexers are used extensively throughout the design to route selectable data sources. The referenced schematic shows 2-to-1 and multi-input mux blocks, controlling operand selection between register data, immediate values, or memory outputs. A select signal from the control unit determines active paths. By grounding unused input lines, noise propagation is prevented. Multiplexers are also used to choose ALU operation results and are critical in supporting instruction diversity. Their clean, modular implementation keeps RTL complexity low and eases timing closure during synthesis.

VI. RESULT

The proposed low-power RISC-V processor architecture was thoroughly designed and validated using VHDL, which enabled a detailed and accurate representation of both its structural components and internal operational behaviors. Throughout the development process, modular design principles were followed to ensure that each functional block including the ALU, register file, pipeline control logic, and memory interface could be independently tested, extended, or optimized without affecting the overall processor structure. ModelSim was utilized as the primary simulation tool. This environment provided cycle-accurate waveform analysis, enabling visual inspection of signal transitions,and pipeline stage synchronization. During simulation, extensive emphasis was placed on measuring the processor's ability to execute arithmetic, logical, control, and load/store instructions reliably while responding correctly to pipeline hazards related to branching and data dependencies. A key highlight of the design is the inclusion of a lightweight hardware accelerator specifically developed to support machine learning focused computations such as matrix multiplications, vector arithmetic operations, and activation functions frequently used in neural network inference. This accelerator was integrated closely with the main execution pipeline to minimize communication overhead and improve data throughput. Simulation results demonstrated successful execution of all supported instructions with correct register updates, accurate branching behavior, and data transfer between pipeline stages. Preliminary performance estimates indicate well improved

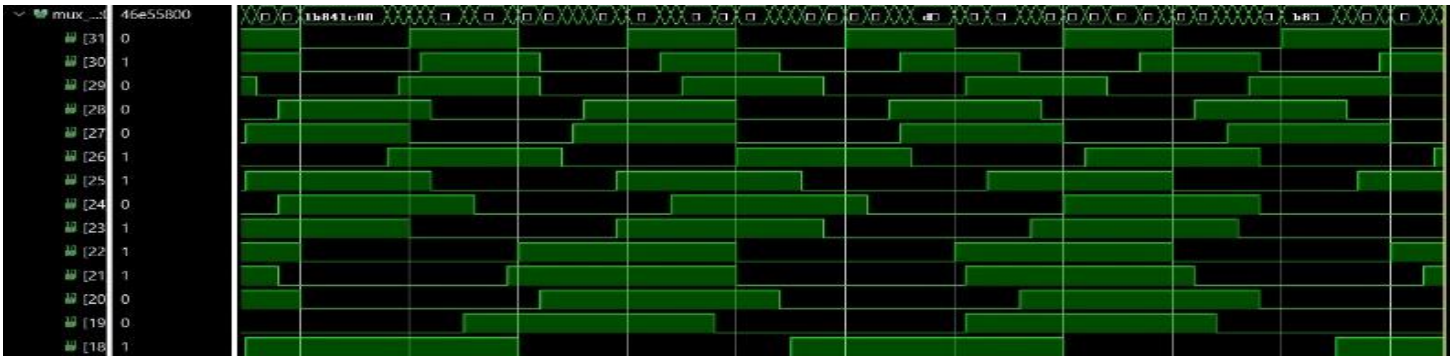

Fig 5. Simulation output of MUX

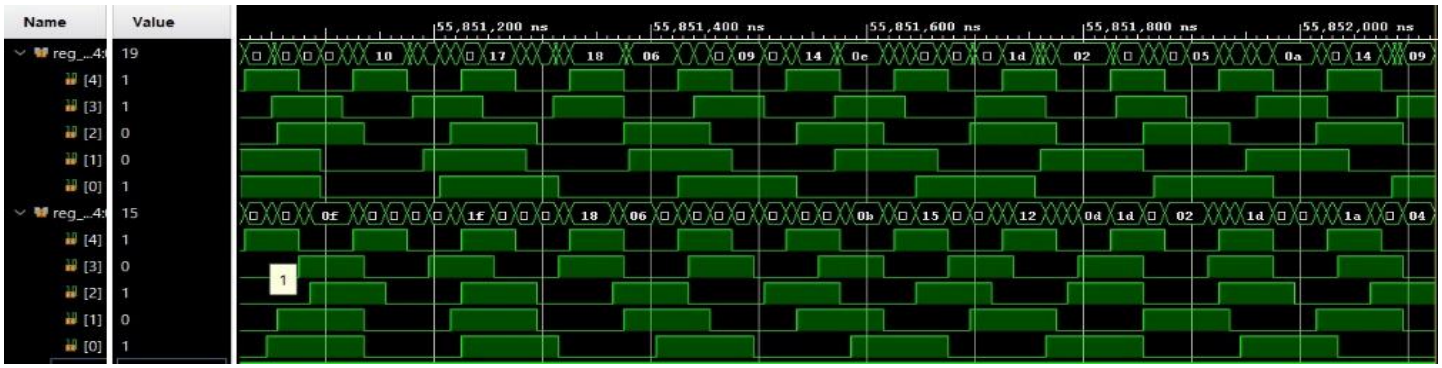

Fig 6. Simulation output of Register Bank

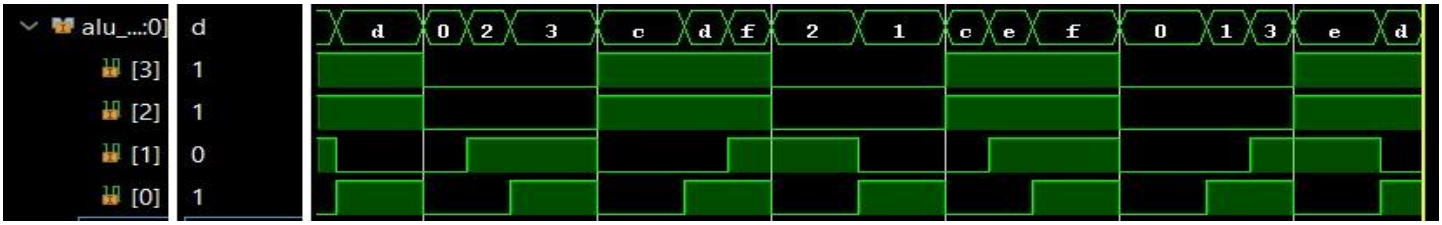

Fig 7. Simulation output of ALU

operational efficiency, particularly for edge-oriented artificial intelligence workloads. Additionally, early power estimation indicated reduced switching activity and low dynamic power consumption, aligning with design objectives for energy-constrained embedded and IoT environments.

Overall, the simulation outcomes validate the correctness, reliability, and enhanced performance of the proposed RISC-V architecture, confirming its suitability for future FPGA deployment .

VII. CONCLUSION

This work presents a comprehensive design and simulation of a low-power RISC-V processor optimized for real-time machine learning applications. The processor is developed using Verilog HDL and follows the RISC-V instruction set architecture, ensuring flexibility and extensibility for domain-specific enhancements. A major design improvement is the integration of a hardware accelerator capable of efficiently executing computationally intensive ML functions such as matrix multiplication, convolution, and activation operations like ReLU and sigmoid. By offloading these tasks to the accelerator, the processor significantly improves computation speed and reduces overall energy consumption when compared to software-only execution. ModelSim simulation results confirm correct functional behavior and demonstrate efficient processing of test instructions. Preliminary power estimation further indicates low energy usage, making the architecture well-suited for edge AI deployments. Overall, the results validate the proposed processor as a promising high-performance and energy-efficient solution for embedded intelligence and edge computing systems.

VIII. FUTURE WORK

Future development will focus on synthesizing and implementing the proposed processor on FPGA hardware to evaluate real-world

performance metrics such as power consumption, maximum operating frequency, resource utilization, and long-term reliability. FPGA deployment will also enable real-time testing with actual machine learning datasets, facilitating the assessment of inference speed, latency, and scalability under practical workloads. To enhance processing efficiency and support more complex neural architectures, future improvements may include multi-core integration, advanced acceleration techniques, and additional custom instruction extensions. These enhancements will enable the processor to support increasingly sophisticated AI applications, expanding its usability in embedded systems, Internet of Things (IoT) devices, and other power-constrained environments where high performance and energy efficiency remain essential. Furthermore, implementing fine-grained clock gating, optimized memory access patterns, and low-precision arithmetic units can significantly reduce dynamic power consumption during ML inference. FPGA-based power profiling will also help identify performance bottlenecks and guide architectural refinements for improved energy efficiency. As the design matures, support for emerging TinyML workloads and hardware security features may be incorporated to broaden deployment suitability and ensure reliable operation in real-world edge environments.

IX. REFERENCES

[1] P. P. Ray, "TinyML: State-of-the-art and prospects," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 110–119, 2021.

[2] G. Srivastav, "5-stage pipelined RISC-V processor in RTL," *International Journal of Computer Applications*, vol. 175, no. 5, pp. 1–8, 2021.

[3] Y. Yan and G. Michelogiannakis, "The RISC-V pipeline implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 13, no. 4, pp. 1–25, 2016.

[4] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, RISC-V Edition, 1st ed., Morgan Kaufmann, 2017.

[5] L. Yang, et al., "Low-power design techniques for RISC-V processors," *IEEE Transactions on VLSI Systems*, vol. 28, no. 3, pp. 624–637, 2020.

[6] S. Mittal, "A survey of techniques for designing low-power embedded systems for machine learning applications," *Journal of Low Power Electronics and Applications*, vol. 9, no. 4, pp. 1–22, 2019.

[7] R. Kumar, A. Singh, and M. Gupta, "FPGA-based acceleration of machine learning algorithms using RISC-V processors," *International Conference on VLSI Design*, pp. 1– 6, 2022.

[8] M. Shafique, et al., "Energy-efficient embedded processors for machine learning: Challenges, opportunities, and innovations," ACM Transactions on Embedded Computing Systems, vol. 20, no. 5, pp. 1–34, 2021.

[9] M. Gautschi, et al., "Near-threshold RISC-V core with DSP extensions for scalable IoT edge computing," IEEE Transactions on VLSI Systems, vol. 25, no. 10, pp. 2700–2713, 2017.

[10] M.Banerjee and S. Roy, "Hardware acceleration techniques for TinyML applications," IEEE Access, vol. 9, pp. 120842–120856, 2021.

[11] N. Bouraoui, et al., "Design and implementation of RISC-V processors on FPGA for energy-efficient applications," Microprocessors and Microsystems, vol. 89, pp. 104419, 2022.

[12] B.Reagen, et al., "A case for efficient accelerator design for deep neural networks in embedded systems," International Symposium on Computer Architecture (ISCA), pp. 1–12, 2016.

[13] P. Whatmough, "Low-power neural network inference for edge devices," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 10, no. 4, pp. 522–531, 2020.

[14].L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs," Conference on Neural Information Processing Systems (NeurIPS), pp. 1–13, 2018.

[15] T. Chen, et al., "Eyeriss: A spatial architecture for energy-efficient CNN processing," IEEE Micro, vol. 36, no. 3, pp. 92–108, 2016.

[16] J. Park, et al., "Quantization techniques for low-power edge AI inference," ACM Computing Surveys, vol. 55, no. 8, pp. 1–36, 2023.

[17]N. P. Jouppi, et al., "In-datacenter performance analysis of a Tensor Processing Unit," ISCA, pp. 1–12, 2017.

[18] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanović, "The RISC-V instruction set manual, Volume I: User-level ISA, Version 2.1," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-118, 2016.

[19] Z. He, X. Li, and Q. Song, "Design and implementation of a 32-bit pipelined RISC-V processor on FPGA," IEEE Access, vol. 8, pp. 87234-87243, 2020.

[20] A. K. Singh, et al., "Performance optimization techniques in pipelined embedded processors," Microprocessors and Microsystems, vol. 86, pp. 103378, 2021.

[21] G. Razavi, et al., "A survey of hazard detection and data forwarding techniques in pipelined processors," IEEE Transactions on Computers, vol. 69, no. 7, pp. 987–1001, 2020.

[22] C. Fu, et al., "A low-power RISC-V core with efficient branch prediction for embedded applications," IEEE Transactions on Circuits and Systems I, vol. 67, no. 12, pp. 4669–4680, 2020.

[23] H. Pan, et al., "Design exploration of RISC-V processor microarchitectures for IoT edge devices," ACM Transactions on Embedded Computing Systems, vol. 21, no. 3, pp. 1-24, 2022.

[24] A. Ferraiuolo, et al., "Securing embedded processors using RISC-V physical memory protection," IEEE Symposium on Security and Privacy Workshops, pp. 39-45, 2017.

[25] F. Zaruba and L. Benini, "The cost of application class RISC-V processors: An energy-focused quantitative evaluation," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 11, pp. 2629-2640, 2019.