Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 46 No. 10
October 2025

Multicycle RV32IM Microprocessor for Edge Computing Applications

Lekhana Vempa[1], Atsatha B[2], Keerthana Penumaka[3]

[1, 2, 3] M. Tech VLSI Design, School of Electronics Engineering, Vellore Institute of Technology, Tamil Nadu, India
Email Id: [1] lekhanavempa@gmail.com, [2] atsathab@gmail.com, [3] keerthana.penumaka@gmail.com

Abstract

This paper explores the design of a multicycle RV32IM microprocessor for edge computing applications. Leveraging the modularity of the RISC-V architecture, the RV32IM processor is tailored to address the needs of low-power, high-performance devices in real-time computing scenarios. By implementing a multicycle approach, the processor splits instruction execution into stages, allowing for a more efficient use of resources compared to single-cycle designs. This approach results in reduced hardware complexity, improved clock speeds, and better power management. The paper presents the processor's design process, performance metrics, and its suitability for applications such as IoT devices, smart sensors, and low-cost edge computing platforms.

Keywords: RISC-V, RV32IM, Edge computing, Instruction set Architecture(ISA),Jump and Link (JAL), Jump and Link register (JALR).

1. Introduction

In recent years, RISC processors have become increasingly popular because of their simple architecture and flexibility. This is particularly true for low-cost devices like smartwatches, basic smartphones, budget-friendly smart home gadgets, and fitness trackers, where RISC-V processors offer a great balance of performance and efficiency.

The RV32IM microprocessor, based on the open-source RISC-V architecture, has gained attention due to its customizable and modular design. The RV32I core, along with the optional "M" extension for integer multiplication and division, provides a solid foundation for building processors that can be tailored to specific tasks. This paper focus is on the design and its implementation of a multicycle RV32IM processor, optimized for edge computing. Multicycle processor design helps in breaking the execution of instructions into multiple stages while in single cycle processor design the instructions handle all instructions in one cycle. This breaking of instructions helps make the design simple, allows higher clock speed, and makes better use of resources resulting in improved performance and better energy efficiency, both of which are essential for edge computing applications.

2. Edge computing

Edge computing is the process of processing data near its source, at the network's edge, to offer low-latency, energy-efficient, and secure solutions for applications like the Internet of Things, automotive systems, and medical devices. While the centralized cloud computing does the complete work in the centralized cloud, but edge computing works contrary to the cloud computing by reducing the data transmission, latency and bandwidth consumption while improving privacy by processing the private or sensitive data locally. This is really essential for real-time applications where fast data processing is required, like integration of sensors in driverless cars (ADAS), smart agricultural monitoring. By assigning computational tasks to devices with limited resources, edge computing helps to create scalable and efficient systems that are appropriate for a range of latency-sensitive use cases.

Role of Multicycle in Edge Computing

The base integer instruction set (RV32I) and multiplication/division extension (M) of the multicycle RISC-V RV32IM architecture makes it optimal for edge computing applications since it provides energy efficiency and customizability. There can be further design which can be proposed for particular edge applications, like adding unique instructions for machine learning or sensor data processing, because of RISC-V's open-source nature. With this flexibility and a

small size, RV32IM-based systems are ideal for edge devices with limited resources, such as wearables, smart sensors, and industrial controllers.

Why multicycle over single cycle and pipelining processors

In contrast to single-cycle and pipelined architectures, multicycle RISC-V RV32IM processors provide clear benefits for edge computing. Multicycle designs divide instruction execution into multiple cycles, which lowers power consumption and hardware complexity—two factors that are crucial for battery-powered edge devices—in contrast to single-cycle processors, which carry out each instruction in a single clock cycle at the expense of high power consumption and complicated hardware. Pipelined processors, which achieve higher throughput but introduce complexity, increased power draw, and potential pipeline hazards. IoT nodes and medical monitors are examples of edge applications that value energy efficiency over raw performance.

3. RISC-V Processor

The RISC-V processor is based on an open-source instruction set architecture (ISA) that follows the principles of Reduced Instruction Set Computing (RISC). Originally developed at the University of California, Berkeley, RISC-V was designed to be simple, modular, and extensible, making it suitable for a wide range of computing applications—from small embedded systems to high-performance computing. One of its most important advantages is that it is royalty-free and openly available, allowing anyone to implement and customize it without licensing fees or legal restrictions. Its modular nature enables developers to include only the instruction set extensions they need, improving efficiency and reducing hardware complexity. RISC-V supports 32-bit, 64-bit, and even 128-bit implementations and offers standard extensions for tasks like floating-point arithmetic and atomic operations. This flexibility, combined with a growing ecosystem of tools, IP cores, and community support, makes RISC-V an increasingly popular choice for both academic research and commercial product development.

Instruction Set

One of the main benefits of RISC-V is its open-source design, which can be used for free without paying licensing fees. This makes it free from many of the issues and intellectual property (IP) restrictions found in proprietary ISAs like MIPS, SPARC, and x86. Other open-source ISAs like OpenRISC and SPARC V8 are available, but RISC-V is different with its modular architecture. This modularity allows hardware developers to incorporate only the instruction sets necessary for an application, maximizing efficiency through the elimination of unnecessary hardware overhead.

RISC-V consists of a base integer ISA and a variety of optional extensions. The base ISA must be included in every implementation, and the optional extensions are divided between standard and non-standard types. Standard extensions are commonly applicable and are designed to work together without interference, whereas non-standard extensions are more specialized and can overlap with other extensions. This makes it possible to customize implementations for particular use cases.

Four principal standard extensions to RISC-V in addition to the base integer instruction set exist:

'M': Supports integer multiplication and division.

'A': Introduces atomic instructions for synchronization of memory in multi-processor systems.

'F': Supports single-precision floating-point operation.

'D': Supports double-precision floating-point operation.

A configuration with all four of these extensions and the base integer instructions (IMAFD) is usually called 'G', which makes up the RV32G variant for a 32-bit system.

4. Literature Review

The rapid growth of the Internet of Things (IoT) and Wireless Sensor Networks (WSN) has driven the rise of edge computing. This approach moves data processing closer to the source, helping to reduce issues tied to cloud-centric methods, such as high latency, bandwidth limits, and privacy concerns [1], [7], [8]. This distributed computing model is a requirement for real-time processing, quick response(lesser latency), and localized intelligence across various applications, such as smart infrastructure like smart grids [7], wearable devices, and smart homes [1], [3]. However, edge devices often face limitations in resources and power, which requires unique processor designs to balance performance, energy efficiency, and cost [1], [6]. Traditional Instruction Set Architectures (ISAs) already present may not perform well for these low end, power-sensitive applications [1], [3], [4], [6]. As a result, RISC-V has originated as a promising alternative. It is an

open-source, modular, and adaptable Instruction Set Architecture [3], [4], [6], [9], [10]. Its modularity enables designers to customize processor functionality to meet specific applications, also includes adding specialized instruction set extensions, like integer multiplication and division (M-extension) [3], [6], [10]. This project focuses on designing a Multicycle RV32IM microprocessor for edge computing applications. This literature survey provides context and supports our method, by examining existing research on RISC-V processor designs and also the strategies for improving energy efficiency and performance and their usage in various edge computing scenarios including important factors like security.

Foundational RISC-V Architectures

Research into RISC-V core architectures for edge and IoT applications provides a spectrum of design complexities. Simple single-cycle implementations are adopted for its proven low-power and minimal resource usage. Meeradevi et al. [5] demonstrated a basic 32-bit RV32I single-cycle processor design focused on functional verification at a low frequency (2.2 KHz), serving as a foundation for our design. Building upon this simplicity, Shukla et al. [1] proposed an energy efficient single-cycle RV32I microprocessor implemented as an ASIC on a 180 nm CMOS process, achieving competitive power efficiency (0.29 mW/MHz) by rigorously minimizing redundant hardware, highlighting its suitability for low-end edge applications. To address the demand for higher throughput, pipelined RISC-V architectures have been extensively explored. Kumar and Ray [2] introduced an adaptive two-stage pipelined RV32I soft-processor, demonstrating an innovative approach to energy efficiency by dynamically activating the second stage only during memory operations, resulting in a 28.2% power reduction compared to single-cycle RV32I designs on an FPGA. For more generalized performance, Raveendran et al. [4] detailed the micro-architectural design and analysis of a 5-stage pipelined RISC-V processor with advanced micro-architectural elements, including comprehensive data and control hazard handling (with branch prediction and epoch registers) and an out-of-order execution Floating Point Unit (FPU). The need for balanced performance and adaptability in diverse IoT scenarios has led to configurable pipelined designs. Chang et al. [6] proposed a configurable five-stage pipelined RV32IM processor core, enabling operation in low power (RV32I only) or high-performance (with 'M' extension) modes. This design, which includes support for supervisor and user privilege levels and CSRs, exhibited superior performance compared to the ARM Cortex-M3. Concurrently, Singh et al. [9] presented another 5-stage pipelined RV32IM core specifically implemented on FPGA, reinforcing the practical integration of a 2-bit branch predictor for increased throughput. A critical design consideration is the effective integration of multicycle integer multiplication and division. While RV32I is suitable for basic embedded systems, many real-world edge applications benefit significantly from hardware-accelerated multiply/divide operations [3], [10]. Our base for this work, Is lam et al. [10], addresses this by proposing RVCoreP-32IM, an extension of the RVCoreP soft processor to support the RISC V M-extension. This paper provides a crucial quantitative analysis of the benefits and overheads of integrating multicycle hardware multiplication and division into a 5-stage pipelined soft processor using a fork-join method. Their benchmarking demonstrates a significant performance improvement of 1.87 to 3.13 times compared to RV32I, albeit with an approximate 1.5x increase in LUT resource utilization. This comprehensive evaluation is vital for understanding the performance gain versus hardware cost trade-offs associated with incorporating multicycle units.

Table 1: RISC-V Instruction Encoding Format

31	25	24	20	19	15	14	12	0
R-type	funct7	rs2	rs1	funct3	rd	opcode		
I-type	immediate[11:0]		rs1	funct3	rd	opcode		
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode		
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode		
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode		
U-type	immediate[31:12]				rd	opcode		

Table 2: Comparison of Instruction Set Architectures

Features	MIPS	ARMv8	OpenRISC	RISC-V
Free & Open			✓	✓
Base+Ext	✓	✓		✓
Compressed	✓			✓
32-bit	✓	✓	✓	✓
64-bit	✓	✓		✓
128-bit		✓		✓
Privileged ISA		✓		✓
IEEE 754-2008		✓		✓

Specialized Cores and Accelerators for Edge AI

As edge computing increasingly incorporates AI, researchers also focus on specialized hardware to speed up machine learning tasks. This often involves extending the base RISC-V ISA with custom instructions or coprocessors. A review by [3] outlines RISC-V's role in the entire edge intelligence stack. It covers

accelerators, compilers, toolchains, and various AI applications such as object detection, autonomous navigation, and smart healthcare. Extending this idea, the AI-RISC project [11] suggests a scalable RISC-V processor that integrates hardware accelerators (AI Functional Units) directly into the processor pipeline. This design uses hardware, ISA, and software co-design and achieves a 17.63x speedup on vector-matrix multiplication and a 3.93x improvement in energy efficiency compared to a basic RISC-V core. Similarly, a study by Yadav et al. [12] on a RISC-V System on Chip (SoC) with custom DSP accelerators shows a 17% reduction in power usage compared to an ARM Cortex-M0, showcasing the power-saving benefits of including specialized accelerators for edge computing. These strategies are further refined by highly focused designs. A Reconfigurable Approximate Computing RISC-V Platform [13] offers a 3-stage pipelined core that allows configurable trade-offs between accuracy and energy use. This is especially useful for applications requiring fault tolerance, where some imprecision is acceptable in return for power savings. For specific uses, the RisCO2 processor [14] was designed for a low-power CO2 concentration sensor and achieved a 53.5% reduction in energy consumption compared to a reference implementation. This illustrates how customizing RISC-V processors for specialized tasks can lead to significant energy savings.

Multicore Platforms and System-Level Challenges

The massive scale of IoT creates a need for solutions that ex tend beyond single-core designs to manage large, distributed, and parallel workloads. The paper ”Developing a Multicore Platform Utilizing Open RISC-V Cores” [15] addresses this need by focusing on building a multicore platform that takes advantage of RISC-V’s open and flexible nature. This work examines the system-level challenges of coordinating multiple cores and managing shared resources. These factors are crucial for scalable edge systems that need to perform various tasks concurrently. This indicates a significant move toward enabling the complex processing capabilities needed for advanced edge applications. In addition to core architecture, successful edge computing systems also require attention to wider system-level needs. Security and privacy are critical in highly distributed IoT settings [8]. Alrowaily and Lu [8] review secure edge computing in IoT, detailing essential security needs (confidentiality, integrity, availability) and risk-sharing tactics. Their research emphasizes the requirement for processors that can support secure and dependable functions in a distributed edge environment. Moreover, practical applications like smart grids require real-time data processing, multi-protocol adaptation, and immediate response capabilities, often necessitating multi core processing at the edge layer [7].

Research Gaps and Motivation for the Proposed Work

The literature reviewed shows significant progress in RISC Vprocessor design, from ultra-low-power single-cycle cores to performance-focused pipelined and configurable architectures. Studies have quantitatively assessed the benefits of adding multicycle hardware multiplication and division (M-extension) [10], while the trend of using specialized AI and DSP accelerators is growing [11], [12]. However, there remains a need for a unified approach that dynamically optimizes a multicycle RV32IM microprocessor for a variety of edge computing applications, particularly focusing on the complex trade-offs between performance improvement and energy/area costs in a truly adaptive way. While configurable processors [6], [13] offer flexibility, there’s still a need to investigate more refined, real-time adaptive strategies for multicycle units that can adjust processing capabilities based on current workload needs. This would help achieve optimal energy efficiency for irregular or highly dependent edge computing tasks. Furthermore, despite clear performance advantages of RV32IM [10], specific design choices for multicycle implementations that further cut down resource usage while maintaining high performance per watt across varied edge applications (like those requiring strong security features [8] or real-time protocol handling [7]) need more thorough exploration. Building on Islam et al.’s [10] insights regarding the integration of the multicycle ’M’ extension, this project aims to design and evaluate a new Multicycle RV32IM Microprocessor for Edge Computing Applications. Our work will examine advanced micro-architectural optimizations within the multicycle execution framework to boost energy efficiency and adaptability. This will include a thorough analysis of the power-performance-area trade-offs associated with different multicycle unit designs, striving for a design that is not only highly effective for typical RV32IM workloads but also accounts for the dynamic requirements and essential non functional properties demanded by next-generation intelligent edge devices.

5. Proposed Architecture

The previous proposed architectures are either single cycle or pipelined architectures, depending on which increases complexity for high performance. In this paper, a multicycle architecture is being proposed with unified memory and with RV32M support, which is usually missing. The architecture in Fig. 1 combines the submodules including program counter, unified memory, register file, instruction registers, register file, ALU sign extension, branch comparison, and this data path is connected to an external control unit, it supports multicycle operations.

Program Counter (PC): A crucial part of the Datapath, the Program Counter (PC) creates the 32-bit address (pc) for instruction fetches from the shared 4KB memory. Through a 2 bit control signal (pc_src) from the Control Unit, it facilitates conditional branches (PC+immediate), sequential execution (PC +4), and jumps (JAL, JALR).

Unified Memory: In a von Neumann architecture, the Datapath's Unified Memory module offers a 4KB word-addressable memory array (1024 × 32-bit words) for both data accesses and instruction fetches. In order to ensure RISC-V compliance, it supports RV32I load/store instructions (LB, LH, LW, LBU, LHU, SB, SH, and SW) with byte, half-word, and word granularity. Strict alignment is enforced by the module, which returns 32'hxxxxxxxx for misaligned loads and requires word aligned instructions and proper alignment for data accesses. While initialization is done with a program, synchronous read/write operations follow the multi-cycle pipeline. Hex files make simulation easier.

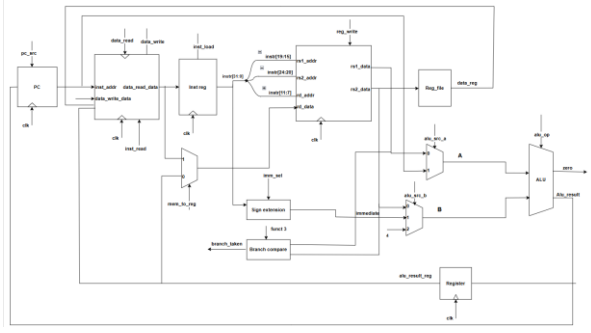

Fig. 2. Proposed Architecture

Instruction Register: The 32-bit instruction fetched from the unified memory for decoding in the multi-cycle pipeline is kept in the Instruction Register (IR). When inst load is active, it loads the instruction from memory and directs it to the Control Unit, Register File, and Sign Extension modules.

Register File: The RISC-V processor's 32 × 32-bit general purpose registers x0–the Register File of the DataPath controls x31, and x0 is hardwired to zero. It employs instruction fields instr[19:15], instr[24:20], and instr[11:7] for addressing and provides two read ports, rs1 data and rs2 data, for source operands and one write port, rd data, for results. Synchronous writes occur on the clock edge if enabled by reg write, and resets set registers to zero. The Register File exchanges data with the Unified Memory for load/store, the Instruction Register for addresses, and the ALU for operands.

Branch and Sign-Extend: The Branch Compare module compares register values rs1 data and rs2 data according to the funct3 field to generate the branch taken signal. It accomplishes this by checking conditions for RV32I branch instructions (i.e., BEQ, BNE, and BLT). The signal is what enables effective control flow in the multi-cycle pipeline by causing the Control Unit to update the Program Counter through pc src. I-type, S-type, B-type, J-type, and U-type instructions are handled by the Sign Extension module, which generates 32-bit immediates from instruction fields depending on imm sel. To perform arithmetic and address computation, it feeds the current input to the ALU; to branch and jump, it feeds the Program Counter.

ALU: Due to its modularity and open-source nature, the RISC-V instruction set architecture (ISA) has grown to be widely used in embedded systems. Part of every RISC-V processor, the Arithmetic Logic Unit (ALU) is responsible for performing arithmetic and logical operations. RV32I (base integer) and RV32M (multiplication/division) instruction sets are supported by the 32-bit ALU discussed in this paper. By employing a finite state machine (FSM) for multiple-cycle operations such as multiplication and division and single-cycle for simple instructions, the design puts a strong emphasis on efficiency. For maximizing division performance, a non restoring division algorithm is employed, carrying out 8 bits per cycle for 4 cycles. To keep latency at a minimum, special cases such as division by zero are also handled by the ALU in combination. Some of the contributions of this work include an RV32I and RV32M, RISC-V compliant ALU, an effective FSM for multi-cycle operations, and a lightweight non-restoring division algorithm.

RV32M instructions are frequently not supported by earlier ALU designs for RISC-V processors, like those in [1], which concentrate on single-cycle operations to

reduce latency. Pipelined architectures are used in other implementations [6], which boost throughput at the expense of increased resource consumption. Using a non-restoring division algorithm to balance complexity and performance, our design stands out for supporting both single-cycle and multi-cycle operations in a small footprint. Our method processes 8 bits per cycle, achieving division in 4 cycles.

The proposed ALU is a 32-bit design with the following inputs and outputs: Inputs: 32-bit operands A and B, a 5 bit opcode, clock (clk), and reset (rst). Outputs: 32-bit result (ALU out), zero flag, and ready signal. The ALU supports 18 operations from the RV32I and RV32M instruction sets, including addition (ADD), subtraction (SUB), logical shifts (SLL, SRL, SRA), comparisons (SLT, SLTU), bitwise op erations (AND, OR, XOR), multiplication (MUL, MULH, MULHSU, MULHU), and division/remainder (DIV, DIVU, REM, REMU). A 2-bit FSM manages three states: IDLE (single-cycle operations), MUL (multi-cycle multiplication), and DIV (multi-cycle division/remainder).

Control unit: This paper presents a Control Unit for a multi cycle RISC-V processor supporting RV32I and RV32M instructions. The design features a five-state FSM with a stalling mechanism for multi-cycle operations, optimizing resource usage for edge computing applications. Multi-cycle designs provide simplicity but may lack support for RV32M multicycle operations. Our Control Unit distinguishes itself with a five-state FSM that stalls for multi-cycle ALU operations (e.g., MUL, DIV) using alu ready, reducing hardware overhead. The integration with a unified memory DataPath further enhances efficiency.

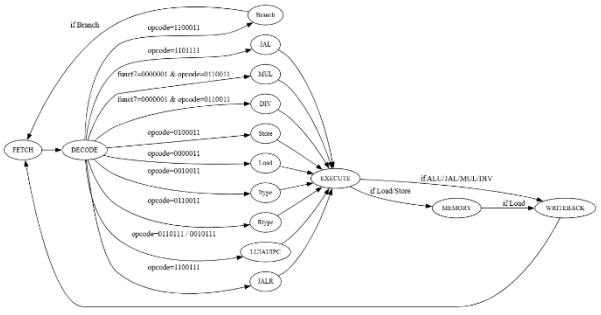

Fig. 3. Control Unit

The Control Unit implements a five-state FSM to manage the multicycle processors as shown in the Fig. 2. The states are: FETCH: Initiates instruction fetch with inst read and inst load. DECODE: Decodes the instruction (op code, funct3, funct7) and sets control signals (e.g., alu op, reg write). EXECUTE: Executes ALU operations, stalling until alu ready for multi-cycle RV32M instructions. MEMORY: Handles load/store operations with data read or data write. WRITEBACK: Writes results to the register file with reg write.

6. Results and Discussion

By simulation on a testbench implemented using UVM, the multi-cycle RV32IM processor design efficiently executes a range of instructions. Multi-cycle operations such as MUL and DIV illustrate the operation of the ALU in handling complex computations in multiple cycles, while single-cycle instructions such as ADD, SUB, and AND are executed efficiently. Program Counter efficiently

```
# [235]   Opcode: 0x13, RS1: x0, RS2: x1, RD: x1
# [235]   Funct3: 0x0, Funct7: 0x00
# [235]   ALU Operation: ADD
# [235]   Operands: RS1=0x00000000, RS2/IMM=0x00000001
# [235] === SCOREBOARD VERIFICATION ===
# [235] VERIFY: Checking instruction 3 at PC=0x00000008
# [235] VERIFY: PC progression check - Expected=0x00000008, Actual=0x00000008,   PASS
# [235] VERIFY: Instruction decode OK - ADDI
# [235] VERIFY: ALU operation ADD - PASS
# [235] VERIFY: Next expected PC = 0x0000000c
# [245] STATE TRANSITION: FETCH -> DECODE
# [245] === REGISTER WRITE ===
# [245] REG: Writing 0x00000002 to register x2
# [255] STATE TRANSITION: DECODE -> EXECUTE
# [255] === ALU OPERATION COMPLETE ===
# [255] ALU: ADD
# [255] ALU: A=0x00000000, B=0x00000002
# [255] ALU: Result=0x00000002, Ready=1
# [255] ALU: 0 + 2 = 2
# [265] STATE TRANSITION: EXECUTE -> WRITEBACK
# [265] === REGISTER WRITE ===
# [265] REG: Writing 0x00000002 to register x2
# [275] STATE TRANSITION: WRITEBACK -> FETCH
# [275] === INSTRUCTION FETCH ===
# [275] PC=0x0000000c, INST=0x00200113 (ADDI)
# [275] Instruction Details:
# [275]   Opcode: 0x13, RS1: x0, RS2: x2, RD: x2
# [275]   Funct3: 0x0, Funct7: 0x00
# [275]   ALU Operation: ADD
# [275]   Operands: RS1=0x00000000, RS2/IMM=0x00000002
# [275] === SCOREBOARD VERIFICATION ===
# [275] VERIFY: Checking instruction 4 at PC=0x0000000c
# [275] VERIFY: PC progression check - Expected=0x0000000c, Actual=0x0000000c,   PASS
```

Fig. 4. Design and Testbench Modules

```
# Loading sv_std.std
# Loading work.processor_if(fast)
# Loading work.uvm_pkg(fast)
# Loading work.uvm_pkg_sv_unit(fast)
# Loading work.questa_uvm_pkg(fast)
# Loading work.processor_testbench_top(fast)
# Loading work.processor_if(fast__2)
# Loading work.RV32IM_MultiCycle(fast)
# Loading work.DataPath(fast)
# Loading work.Program_Counter(fast)
# Loading work.Unified_Memory(fast)
# Loading work.Instruction_Register(fast)
# Loading work.Register_File(fast)
# Loading work.Sign_Extend(fast)
# Loading work.Branch_Compare(fast)
# Loading work.ALU(fast)
# Loading work.Stage_Register(fast)
# Loading work.Control_Unit(fast)
# Compiling /tmp/unknown08c6567358243_dpi_51/linux_x86_64_gcc-10.3.0/exportwrapper.c
# Loading /tmp/unknown08c6567358243_dpi_51/linux_x86_64_gcc-10.3.0/vsim_auto_compile.so
#
# run -all
# UVM_INFO /usr/share/questa/questasim/verilog_src/uvm-1.2/src/base/uvm_root.svh(392) @ 0: reporter [UVM/RELNOTES]
# ----------------------------------------------------------------
```

Fig. 5. Flow of five stage simulation

handles control flow instructions such as JALR, utilizing the pc src mechanism, ensuring proper program flow updates. With transitions such as STATE IDLE to STATE MUL or STATE IDLE to STATE DIV EXECUTE to MEMORY synchronized through the ready signal, the state machines within the ALU and Control Unit work as expected, providing a solid foundation for the processor's multi-cycle architecture. By the simulation

results, the design is dependable since the LSB masking of Program Counter for JALR targets avoids any misalignment possibilities and the multi-cycle support of the ALU for MUL and DIV operations works as intended. With the 4-cycle DIV over the 2-cycle MUL trade-off being an understandable design choice, this verification highlights the processor's viability for prototype or teaching purposes.

Fig. 6. Final processor Statistics

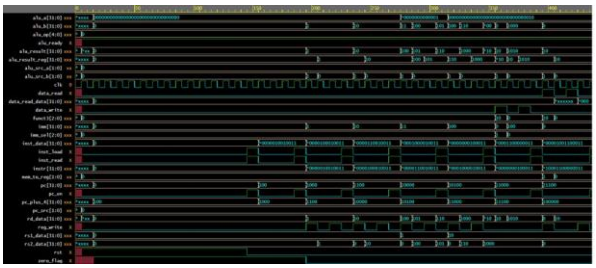

Fig. 7. Instruction execution by proposed architecture

7. Conclusion

By means of efficient simulation validation, the multi-cycle RV32IM processor offers a stable and adaptable platform for RISC-V architecture research. The design effectively maintains a balance between complexity and performance by using a multi-cycle methodology to support the M-extension instructions with regard to feasible dependable control flow.

```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   18
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [DRIVER]     3
# [MONITOR]     1
# [Questa UVM]     2
# [RNTST]     1
# [SCOREBOARD]     2
# [SEQUENCE]     4
# [TEST]     3
# [TEST_DONE]     1
# [UVM/RELNOTES]     1
#
```

Fig. 8.UVM Report summary

```
[515] VERIFY: Checking instruction 10 at PC=0x00000024
[515] VERIFY: PC progression check - Expected=0x00000024, Actual=0x00000024,   PASS
[515] VERIFY: Instruction decode OK - ADDI
[515] VERIFY: ALU operation ADD - PASS
[515] VERIFY: Next expected PC = 0x00000028
=== VERIFICATION PROGRESS ===
Instructions verified: 10
PC progression errors: 0
ALU verification errors: 0
Instruction decode errors: 0
```

Fig. 9. Overall Error Summary

The lack of functional errors in the UVM testbench shows the way the implementation is done. Hardware synthesis may be employed to optimize further performance and scalability in the future, setting this work as a basis for future studies in advanced processors.

Table 3: Processor Verification Statistics

ASPECTS	STATUS
Instruction Fetch	Success
Overall Error rate	0.00%
Overall pass rate	100%
Verification Errors	0
Scoreboard Verification	Passed
Total Processor Verification	Passed

Acknowledgement

We would like to thank Dr. Kumaravel S [VIT Vellore, VLSI Design] for his constant support and guidance during the development of this multi-cycle RV32IM processor project. We are grateful to our peers for their valuable feedback and support in making the project better at every turn.

References

[1] Shukla, Satyam, Punyesh Kumar Jha, and Kailash Chandra Ray. "An energy-efficient single-cycle RV32I microprocessor for edge computing applications." Integration 88 (2023): 233-240.

[2] Kumar, S., Ray, K. C. (2024). An energy-efficient adaptive two stage pipeline RISC-V based soft-processor for edge-computing applications.

[3] Q. Wei, E. Cui, Y. Gao and T. Li, "A Review of Edge Intelligence Applications Based on RISC-V," 2023 2nd International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT), Xiamen, China, 2023, pp. 115-119, doi: 10.1109CCPQT60491.2023.00025. keywords: Trusted comput-ing; Systematics; Image edge detection; Software algorithms; Computer architecture; Software; Hardware; RISC-V; edge intelligence; AI; energy efficiency,

[4] A. Raveendran, V. B. Patil, D. Selvakumar and V. Desalphine, “A RISC-V instruction set processor-micro-architecture design and analysis,” 2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA), Bengaluru, India, 2016, pp. 1-7, doi: 10.1109VLSI-SATA.2016.7593047. keywords: Registers; Pipelines; Computer architecture; Decoding; Reduced instruction set computing; Organizations; Processor Design; Processor Microarchitecture; Out-Of-Order Processor; RISC-V Instruction Set; RISC Processor; IEEE 754-2008 FPU Standard;Floating Point Co-processor,

[5] T. Meeradevi, K. Mohanraj, B. M. Mourissh, V. Santhosh Sivaa, R Samikannu and S. Sasikala, “Design of 32 Bit RISC V Processor,” 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 2024, pp. 1-7, doi: 10.1109/ICCCNT61001.2024.10726132. keywords: Reduced instruction set computing; Memory management; Pipelines; Process control; Software; Registers; Reliability; Signal analysis; Optimization; Testing; PUF; Modeling attacks; Vulnerability; Enhanced Security; Unpredictable responses,

[6] Chang, Y.; Liu, Y.; Peng, C.; Guo, J.; Zhao, Y. Design of a Configurable Five-Stage Pipeline Processor Core Based on RV32IM. Electronics 2024, 13, 120. https://doi.org/10.3390/electronics13010120

[7] S. Song, S. Li, H. Gao, J. Sun, Z. Wang and Y. Yan, “Research on Multi-Parameter Data Monitoring System of Distribution Station Based on Edge Computing,” 2021 3rd Asia Energy and Electrical Engineering Symposium (AEEES), Chengdu, China, 2021, pp. 621-625, doi: 10.1109/AEEES51875.2021.9403026. keywords: Zigbee; Process control; Distribution networks; Power system stability; Power grids; Real-time systems; Security; distribution station; Internet of things; edge computing; multi-parameter acquisition; intelligent terminal,

[8] M. Alrowaily and Z. Lu, “Secure Edge Computing in IoT Systems: Review and Case Studies,” 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, USA, 2018, pp. 440-444, doi: 10.1109/SEC.2018.00060. keywords: Edge computing; Task analysis; Computer architecture; Internet of Things; Sensors; Password; Edge computing, Security, Internet of Things (IoT),

[9] Singh A, Kumar A, Singh A, Reddy RA, Pushpalatha KN. Design and Implementation of RISC-V ISA (RV32IM) on FPGA. SSRG International Journal of VLSI Signal Processing. 2023;10(2):17-21.

[10] Ashraful Islam, M., Miyazaki, H., and Kise, K., “RVCoreP-32IM: An effective architecture to implement mul/div instructions for five stage RISC-V soft processors”, ¡i¿arXiv e-printsi/i¿, Art. no. arXiv:2010.16171, 2020. doi:10.48550/arXiv.2010.16171

[11] Verma, Vaibhav. AI-RISC: Scalable RISC-V Processor for IoT Edge AI Applications. University of Virginia, Electrical Engineering - School of Engineering and Applied Science, PHD (Doctor of Philosophy), 2022-04-27, https://doi.org/10.18130/nj7t-7j93.

[12] Delavari, A., Ghoreishy, F., Shahriar Shahhoseini, H., and Mirzakuchaki, S., “A Reconfigurable Approximate Computing RISC-V Platform for Fault-Tolerant Applications”, ¡i¿arXiv e-printsi/i¿, Art. no. arXiv:2410.00622, 2024. doi:10.48550/arXiv.2410.00622.

[13] Yadav, P., “Design and Implementation of a RISC-V SoC with Custom DSP Accelerators for Edge Computing”, ¡i¿arXiv e-printsi/i¿, Art. no.arXiv:2506.06693, 2025. doi:10.48550/arXiv.2506.06693.

[14] N´u˜nez-Prieto, R., Castells-Rufas, D. and Ter´es-Ter´es, L., 2023. RisCO2: Implementation and Performance Evaluation of RISC-V Processors for Low-Power CO2 Concentration Sensing. Micromachines, 14(7), p.1371.

[15] H. Jang et al., “Developing a Multicore Platform Utilizing Open RISC-V Cores,” in IEEE Access, vol. 9, pp. 120010-120023, 2021, doi: 10.1109/ACCESS.2021.3108475. keywords: Multicore processing; Coherence; Computer architecture; Hardware; Software; Field programmable gate arrays; Multicore platform; RISC-V; system-on-chip (SoC); electronic design automation (EDA),