International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Conference Proceedings: Volume: 05 ACME – 2026 DOI: 10.55041/ISJEM.ACME120
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Design and Implementation of 5-Stage pipelined RISC-V Embedded Processor

Mr.Nagaraju Rangappagari
Assistant Professor Dept of ECE
Annamacharya Institute of Technology and Sciences
Tirupati, India
r.nagarajuece@gmail.com

Kunati Sahithi
B Tech Student *Dept of ECE*
Annamacharya Institute of Technology and Sciences
Tirupati,India
sahithis013@gmail.com

Mavilla Triveni
B Tech Student *Dept of ECE*
Annamacharya Institute of Technology and Sciences Tirupati,India
trivenimavilla@gmail.com

Gunduboyana Sravani
B Tech Student *Dept of ECE*
Annamacharya Institute of Technology and Sciences Tirupati,India
gunduboyanasravani@gmail.com

V.SarathChandra
B Tech Student *Dept of ECE*
Annamacharya Institute of Technology and Sciences
Tirupati,India
sarathsai@gmail.com

***Abstract*—**This paper presents the design and implementation of a 32-bit 5-stage pipelined RISC-V embedded processor based on the RV32I instruction set architecture. The processor follows the classical pipeline stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB) to improve instruction throughput and overall computational efficiency. To enhance pipeline performance, a hazard detection unit and forwarding paths are incorporated to resolve data hazards and minimize pipeline stalls. Control hazards are handled through branch target address computation and pipeline flushing mechanisms.

The processor is modeled using modular RTL design and verified through simulation to ensure correct instruction execution and functional validation. Synthesis results indicate logic utilization below 1% and flip-flop utilization of approximately 0.32%, demonstrating efficient hardware resource usage. The normalized silicon area is maintained below 10%, significantly lower than conventional baseline designs. Power analysis shows that dynamic power accounts for nearly 97% of total consumption, while static power contributes approximately 3%. The proposed architecture achieves improved clock efficiency, optimized power-area trade-off, and enhanced throughput compared to traditional non-pipelined implementations, making it suitable for embedded systems and academic research applications.

Keywords— RISC-V, RV32I, Pipelined Processor, 5-Stage Pipeline, Hazard Detection Unit, Forwarding Unit, RTL Design, Power Analysis, Area Optimization, Embedded Systems.

I. INTRODUCTION

Modern embedded systems require processors that provide high performance while maintaining low power consumption and efficient hardware utilization. Pipelining is an effective architectural technique that improves instruction throughput by allowing multiple instructions to execute simultaneously across different stages of a data path.

RISC (Reduced Instruction Set Computer) architectures are widely adopted due to their simple instruction formats and efficient implementation. Among them, RISC-V is an open- source Instruction Set Architecture (ISA) that offers flexibility, scalability, and freedom from licensing constraints. Its modular structure makes it suitable for academic research and embedded processor design.

This work presents the design and implementation of a 32-bit 5-stage pipelined RISC-V processor based on the RV32I instruction set. The processor follows the classical stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB). To improve pipeline efficiency, a hazard detection unit and forwarding logic are integrated to resolve data dependencies and reduce stalls. Control hazards are handled through branch target computation and pipeline flushing mechanisms.

The design is implemented using RTL modeling and verified through simulation and synthesis to evaluate functionality, power, and area utilization.

II. LITERATURE SURVEY

The evolution of Reduced Instruction Set Computer (RISC) architectures has significantly improved processor performance through simplified instruction formats and efficient pipeline organization. One of the foundational works in modern computer architecture is presented by David A. Patterson and John L. Hennessy, who introduced the principles of pipelined processor design and performance optimization. Their work emphasized the importance of dividing instruction execution into multiple stages to enhance throughput while maintaining hardware simplicity. However, early educational models primarily focused on conceptual understanding rather than detailed hazard mitigation strategies.

The introduction of the RISC-V Foundation architecture by Andrew Waterman and collaborators provided an open and extensible Instruction Set Architecture (ISA) suitable for both academic and industrial research. Unlike proprietary architectures, RISC-V allows researchers to implement customized microarchitectures. While the RV32I base instruction set defines the operational framework, it does not

prescribe specific pipeline or hazard handling mechanisms, leaving architectural optimization to designers.

Several researchers have implemented RV32I-based pipelined processors using hardware description languages such as Verilog HDL. For instance, implementations like IndiRA and other FPGA-based designs demonstrated modular datapath structures incorporating five-stage pipelines. These designs typically integrate basic hazard detection units and forwarding mechanisms to minimize pipeline stalls. However, many such implementations primarily focus on functional verification and FPGA realization without comprehensive evaluation of power efficiency and silicon area utilization

Recent academic contributions have addressed data hazards more effectively by incorporating dedicated hazard detection logic and operand forwarding paths between pipeline stages. These techniques reduce load-use delays and improve clock efficiency compared to simple stalling-based approaches. Despite these improvements, advanced features such as dynamic branch prediction, cache integration, and power- aware optimization remain areas for further enhancement in many baseline implementations.

From the reviewed literature, it is evident that while pipelined RISC-V processors improve throughput compared to single- cycle designs, there is scope for balanced optimization in terms of hazard resolution, power consumption, and area efficiency. The proposed work builds upon these foundations by integrating structured hazard detection, forwarding logic, and simulation-based power analysis within a modular RTL framework to achieve improved pipeline performance suitable for embedded applications. handling techniques dependent on individual.

III. EXISTING MODEL

Traditional processor architectures used in embedded systems are often based on proprietary instruction set architectures such as ARM and x86, where internal microarchitectural details are not fully accessible for customization. Early implementations commonly employed single-cycle or multi- cycle datapath designs, where instructions are executed sequentially. Although these approaches simplify control logic, they suffer from inefficient clock utilization and limited instruction throughput.

In later developments, pipelined architectures were introduced to improve performance by overlapping instruction execution. However, many academic and baseline RISC implementations incorporate only basic hazard handling mechanisms. Data hazards are frequently resolved using stalling techniques instead of efficient forwarding paths, which increases the number of pipeline bubbles and degrades overall performance. Load-use hazards, in particular, introduce additional delay when proper hazard detection logic is not implemented.

Control hazards caused by branch instructions further impact pipeline efficiency when branch resolution is delayed or flushing mechanisms are not optimized. Additionally, several existing implementations focus primarily on functional verification without comprehensive evaluation of power consumption and silicon area utilization. The absence of structured power-area analysis limits their applicability in resource-constrained embedded systems.

These limitations highlight the need for a more optimized and modular pipelined processor design that integrates effective hazard detection, forwarding mechanisms, and performance evaluation metrics to achieve improved throughput and efficient hardware utilization.

IV. ARCHITECTURE AND DESIGN

The proposed processor is a 32-bit pipelined architecture based on the RV32I base integer instruction set of the RISC-V ISA. The design adopts a classical 5-stage pipeline structure consisting of Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB) stages. Pipeline registers are inserted between each stage (IF/ID, ID/EX, EX/MEM, MEM/WB) to isolate combinational logic and enable simultaneous execution of multiple instructions.

The overall datapath is divided into two major components:

Datapath Unit – Performs arithmetic, logical, and memory operations.

Control Unit – Generates control signals based on opcode and instruction type.

A. Instruction Fetch (IF) Stage

The IF stage includes the Program Counter (PC), instruction memory, and an adder to compute PC+4. The PC is a synchronous register controlled by clock and reset inputs. Upon reset, the PC is initialized to the base address. During normal operation, the PC updates every clock cycle.

A multiplexer selects the next PC value between:

PC + 4 (sequential execution), Branch target address (from EX stage)

The selected address is forwarded to instruction memory to fetch the corresponding 32-bit instruction. The fetched instruction is stored in the IF/ID pipeline register.

B. Instruction Decode(ID) Stage

The ID stage performs instruction decoding and operand fetching. The instruction is divided into opcode, funct3, funct7, rs1, rs2, and rd fields. The control unit generates signals such as RegWrite, MemRead, MemWrite, ALUSrc, Branch, and MemtoReg based on the opcode.

The register file contains 32 general-purpose registers. It provides two read ports and one write port. The source operands (rs1 and rs2) are read and forwarded to the ID/EX pipeline register.

Hazard Detection Unit:

The hazard detection unit checks for load-use dependencies. If the destination register of a previous load instruction matches the source register of the current instruction, a stall signal is generated. This temporarily disables PC update and inserts a pipeline bubble.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Conference Proceedings: Volume: 05 ACME – 2026 DOI: 10.55041/ISJEM.ACME120
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

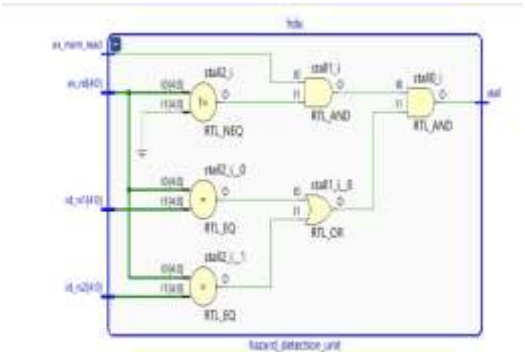


Fig 1: Hazard detection unit

C. Execute(EX) Stage

The EX stage performs arithmetic and logical operations using the ALU. The ALU supports operations such as addition, subtraction, AND, OR, XOR, and comparison based on ALU control signals.

Forwarding Unit:

To resolve data hazards without stalling, forwarding paths are implemented:

From EX/MEM stage to ALU input From MEM/WB stage

to ALU input Multiplexers at ALU inputs select either:

Register file output

Forwarded result from later stage

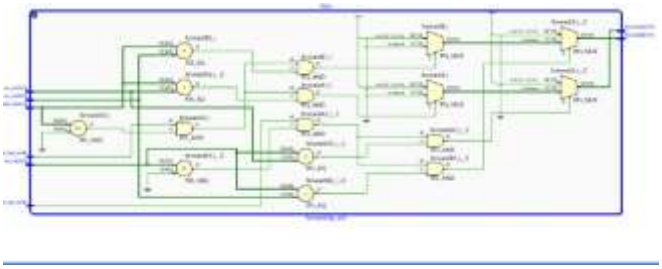


Fig 2:Forwarding unit

Branch Handling Logic:

Branch target address is calculated as: Branch Target = PC + Immediate Offset

Branch condition evaluation is also performed in this stage. If the branch is taken, a flush signal invalidates incorrect
instructions in earlier stages. Arithmetic and Logic Unit(ALU):

The Arithmetic Logic Unit (ALU) serves as the primary execution block in the proposed 32-bit five-stage pipelined RISC-V processor. It is positioned in the Execute (EX) stage, where all computation-related tasks are performed. The ALU processes two 32-bit input operands obtained from the ID/EX pipeline register. Among these inputs, one operand is directly sourced from the register file, while the other is selected either from a register output or an immediate value generated during instruction decoding, based on the ALU Src control signal.

The required operation is selected through a 4-bit ALU control code derived from the main control unit using the opcode, funct3, and funct7 instruction fields.

The implemented ALU is capable of handling a wide range of operations defined in the RV32I base instruction set. These include arithmetic functions such as addition and subtraction, logical operations like AND, OR, and XOR, comparison instructions including signed and unsigned set-less-than (SLT and SLTU), and shift operations such as logical left shift (SLL), logical right shift (SRL), and arithmetic right shift (SRA). In addition to computation, the ALU also performs address calculation for load and store instructions by executing base-plus-offset addition.

To support branch instructions, the ALU generates a Zero status signal that assists in evaluating branch conditions and maintaining correct program flow. The design is realized as a 32-bit combinational module using synthesizable Verilog, ensuring that computation results are produced within a single clock cycle of the Execute stage. Furthermore, the ALU is integrated with operand forwarding paths to reduce data dependency delays, thereby minimizing pipeline stalls and improving overall execution efficiency.

TABLE 1

ALU control lines	Function(Bitwise)
0000	AND
0001	OR
0010	Addition
0011	XOR
0100	SLT
0101	Subtraction
0110	Right shift

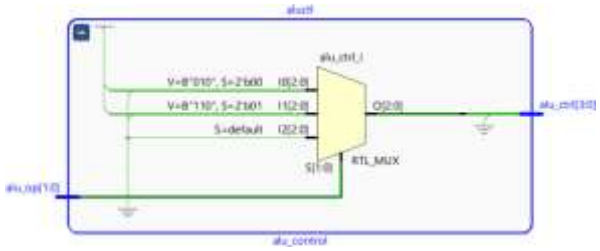


Fig 3:ALU Control

Control
Unit(CU):

The Control Unit (CU) is a fundamental component of the processor architecture responsible for coordinating the overall operation of the data path. In the proposed 5-stage pipelined RISC-V processor, the control unit is implemented in the Instruction Decode (ID) stage, where it interprets the opcode, funct3, and funct7 fields of the instruction. Based on this decoding, the control unit generates appropriate control signals such as Reg Write, Mem Read, Mem Write, ALU Src, Mem to Reg, Branch, and ALU Op. These signals regulate the behavior of various functional blocks including the register file, arithmetic logic unit (ALU), data memory, and multiplexers. The ALU control logic further refines the ALU

Op signal to determine the exact arithmetic or logical operation to be executed in the EX stage. In a pipelined architecture, the generated control signals are propagated through pipeline registers (ID/EX, EX/MEM, MEM/WB) to ensure synchronized execution across stages. This structured control mechanism enables efficient instruction execution while maintaining pipeline stability and correctness.

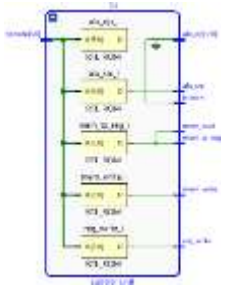


Fig 4: Control Unit

D. Memory Access(MEM) Stage

The MEM stage interfaces with data memory. Based on control signals:

Mem Read enables memory read (Load instruction) Mem

Write enables memory write (Store instruction)

Fig 5:Memory Data

The memory output or ALU result is stored in the MEM/WB pipeline register.

E. Write Back(WB)Stage

The WB stage writes the final result back to the register file. A multiplexer selects between:

ALU result, Memory read data

If Reg Write signal is active, the selected value is written into the destination register (rd).

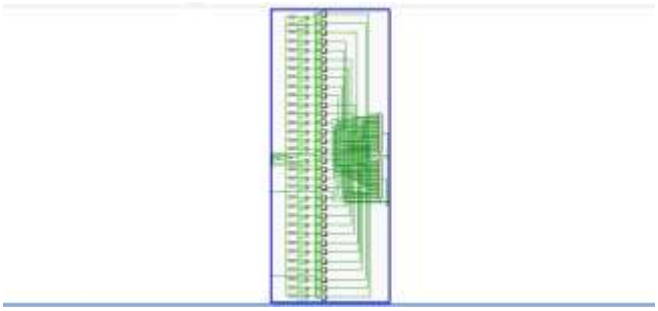


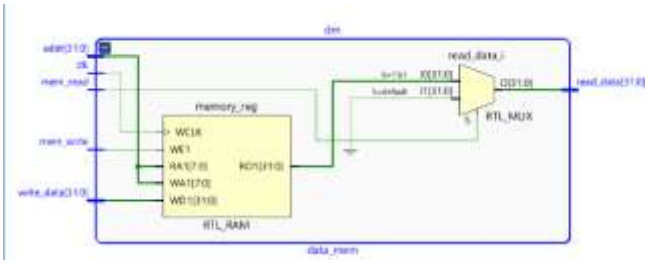
Fig 6:Register File

V. PROPOSED MODEL

The proposed processor is a 32-bit 5-stage pipelined architecture based on the RV32I RISC-V instruction set. The design follows the classical pipeline stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB), enabling parallel execution of multiple instructions to improve overall throughput. A modular RTL-based implementation is adopted to ensure scalability, architectural clarity, and ease of debugging.

To enhance pipeline efficiency, the proposed architecture integrates a dedicated hazard detection unit in the ID stage to identify load-use and data dependency hazards. Instead of relying solely on stalling mechanisms, forwarding paths are implemented in the EX stage to bypass intermediate results directly to the ALU inputs, thereby minimizing unnecessary pipeline stalls. This significantly reduces performance degradation caused by data hazards.

Control hazards are addressed through branch target address computation and pipeline flushing mechanisms, ensuring correct instruction sequencing. The control unit generates structured control signals that propagate through pipeline registers, maintaining synchronization across stages. Furthermore, simulation and synthesis-based evaluation is performed to analyze power consumption and area utilization. The proposed design demonstrates improved clock efficiency, reduced pipeline stalls, and optimized hardware resource usage compared to conventional simplified implementations, making it suitable for embedded system applications and academic research.



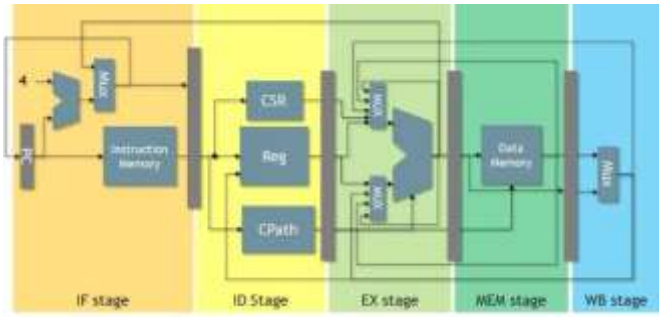


Fig 7:Block Diagram

Fig 8: Schematic diagram of 5-Stage RISC-V Pipelined Processor

VI. RESULT

The proposed 32-bit 5-stage pipelined RISC-V processor was verified using RTL simulation. The waveform results confirm correct instruction **execution** and proper pipeline operation. After reset initialization, the clock-driven processor fetches, decodes, and executes instructions sequentially.

The debug register outputs demonstrate correct register write-back behavior. ALU input signals and control signals (funct3,

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Conference Proceedings: Volume: 05 ACME – 2026 DOI: 10.55041/ISJEM.ACME120
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

funct7, and alu_ctrl) are properly generated based on the decoded instruction. The ALU result matches the expected computed value, confirming accurate arithmetic and logical operation.

The absence of undefined states during normal operation indicates stable control logic. The simulation validates correct data propagation across pipeline stages, ensuring functional correctness of the designed architecture

Fig : Simulation Waveforms

A. GRAPHS AND TABLES

Table 2 : Resource Utilization Table :

Utilization | Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	669	133800	0.50
LUTRAM	128	46200	0.28
FF	868	269200	0.32
IO	130	400	32.50
BUFG	1	32	3.13

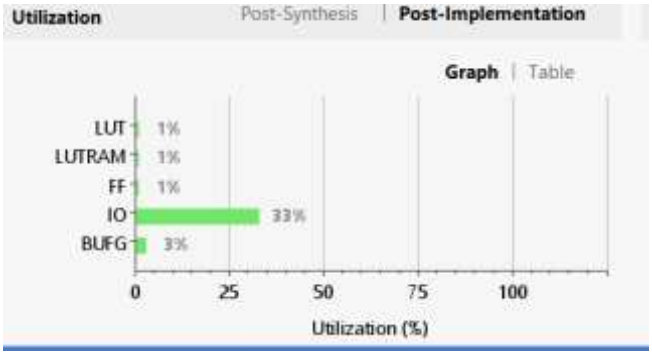


Fig 10: Resource Utilization Graph

Power Utilization :

The power analysis of the implemented 5-stage pipelined RISC-V processor was performed using the Vivado power estimation tool. The total on-chip power consumption is reported as 5.807 **mw** under typical process conditions. Dynamic power dominates the design, contributing 97% (5.657 mw) of the total power, while static power accounts for only 3% (0.150 mw). Among the dynamic components, signal switching consumes the highest portion, followed by logic and I/O power. The junction temperature of 35.9°C confirms stable thermal operation, indicating that the design *operates* within safe limits.

Fig 11:Power

Table 3: Performance comparison Table:

PARAMETER	EXISTING MODEL	PROPOSED MODEL
Logic Utilization	Not Reported	< 1%
FF Utilization	Not Reported	0.32%
LUTs	Not Specified	0.50%
Static Power	Not Specified	3%
Dynamic Power	Dominant	97%

VII. CONCLUSION AND FUTURE SCOPE

This paper presented the design and implementation of a 32-bit 5-stage pipelined RISC-V embedded processor. The architecture was developed using modular RTL design with integrated hazard detection and forwarding mechanisms to ensure efficient pipeline operation. Functional verification through simulation confirmed correct instruction execution and stable control logic behavior. The implemented design demonstrated reliable data propagation across pipeline stages with minimal stalls. Power analysis results indicate that dynamic power dominates overall consumption, while static power remains minimal, ensuring stable thermal performance. The proposed processor achieves efficient performance with optimized area and power characteristics suitable for embedded applications.

The current design can be further enhanced by incorporating advanced features such as branch prediction techniques to reduce control hazards and improve performance. Integration of cache memory (instruction and data cache) can significantly enhance execution speed. Power optimization techniques such as clock gating and operand isolation may further reduce dynamic power consumption. Additionally, extending the processor to support RV32IM (including multiplication and division instructions) or implementing superscalar and multi-core architectures can improve computational capability. The design may also be ported to ASIC technology for improved area and power efficiency in real-time embedded systems.

REFERENCES

[1]. A. Tiwari, P. Guha, G. Trivedi, N. Gupta, N. Jayaraj, and J. Pidanic, "IndiRA: Design and Implementation of a Pipelined RISC-V Processor," in Proc. 33rd Int. Conf. Radioelektronika (RADIOELEKTRONIKA), 2023.

[2]. M. Trivedi, A. Mathur, and K. Mehta,"Design and Performance Evaluation of RISC-V Embedded Processor Using HDL," in Proc. 1st Int. Conf. Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT), 2025.

[3]. B. V. V. Satyanarayana, N. Madhu, and K. Srinivas, "Design and Implementation of High-Performance RISC-V Processor with Five Stage Pipeline on FPGA," in Proc. 3rd Int. Conf. Self Sustainable Artificial Intelligence Systems (ICSSAS), 2025.

[4]. B. W. Mezger, D. A. Santos, L. Dilillo, C. A. Zeferino, and D. R. Melo,"A Survey of the RISC-V Architecture Software Support," IEEE Access, vol. 10, 2022.

[5]. E. Cui, T. Li, and Q. Wei, "RISC-V Instruction Set Architecture Extensions: A Survey," IEEE Access, vol. 11, 2023.

[6]. B. Poduel, P. Kansakar, S. R. Chhetri, and S. R. Joshi, "Design and Implementation of Synthesizable 32-bit Four Stage Pipelined RISC Processor in FPGA Using Verilog/VHDL," Nepal Journal of Science and Technology, vol. 15, no. 1, 2015.

[7]. G. He, Y. Zhao, Y. Xiang, and L. Li, "Design and Implementation of a Prefetcher in a Key Performance Subsystems of RISC-V Processors," Electronics, vol. 15, no. 2, 2026.

[8] 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: 10.1109/VLSISATA.2016.7593047.

[9] K.Stangherlin and M. Sachdev, "Design and Implementation of a Secure RISC-V Microprocessor," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 30, no. 11, pp. 1705- 1715, Nov. 2022, doi: 10.1109/TVLSI.2022.3203307.

[10] A. Oleksiak, S. Cieślak, K. Marcinek and W. A. Pleskacz, "Design and Verification Environment for RISC-V Processor Cores," 2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems", Rzeszow, Poland, 2019, pp. 206- 209, doi: 10.23919/MIXDES.2019.8787108.

[11] China RISC-V Alliance. accessed on Jan. 1, 2023.