Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Design and Implementation of a Self-Adaptive and Performance-Optimized Five-Stage RISC-V Processor on FPGA

P R Prajwal Kumar[1], Yuvaraj Gowda G[2], Raghu Y[3], Jayasurrya M[4], Bharathi Gururaj[5]

[1,2,3,4] Student, Department of Electronics and Communication Engineering, K. S. Institute of Technology, Bengaluru, India

[5] Associate Professor, Department of Electronics and Communication Engineering, K. S. Institute of Technology, Bengaluru, India

Abstract – Pipelining raises processor throughput but introduces data, control and structural hazards, and a fixed optimization strategy often performs unevenly across different workloads. This paper presents the design and FPGA implementation of a self-adaptive, performance-optimized five-stage pipelined RISC-V processor described in Verilog HDL. The processor has Instruction Fetch, Instruction Decode, Execute, Memory Access and Write Back stages separated by four pipeline registers, with forwarding, load-use hazard detection, stalling and branch flushing. A performance-monitoring unit records clock cycles, stalls, forwarding events and taken branches, and an adaptive workload controller uses this information to guide the optimization behavior. The design was simulated and implemented in AMD Vivado 2026.1 on a Xilinx Artix-7 XC7A35T-1CSG324C FPGA and demonstrated on the development board with an OLED display. All individual functional tests and all 25 adaptive-optimization verification cases passed. The implementation uses 1615 slice LUTs (7.76 %) and 1502 slice registers (3.61 %). After optimization, the worst negative slack improved from −98.841 ns to +3.265 ns and the estimated on-chip power fell from 0.084 W to 0.074 W (11.90 %). The adaptive processor reduced execution cycles from 44 to 40 on one workload and from 45 to 41 (8.89 %) on a branch-heavy workload, while four other workloads were unchanged. The benefit therefore depends on workload characteristics, and wider benchmarking is needed to establish general performance.

Keywords – RISC-V, Five-Stage Pipeline, FPGA, Self-Adaptive Optimization, Hazard Handling, Performance Monitoring, Verilog HDL, Artix-7.

I. INTRODUCTION

A processor's performance depends on how efficiently it fetches, decodes, executes and completes instructions. Reduced Instruction Set Computing (RISC) architectures use a small, regular instruction set that suits pipelined implementation. A common organization divides execution into Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM) and Write Back (WB), so that several instructions are in flight at the same time [2], [3].

RISC-V is an open instruction set architecture developed at the University of California, Berkeley [1]. Its modular structure and the absence of licensing fees make it attractive for academic work, embedded systems and hardware research. Field-Programmable Gate Arrays (FPGAs) give a practical way to describe such a processor in a hardware description language, implement it, and measure its timing, resource use and estimated power on real hardware.

Overlapping instructions creates hazards. Data hazards arise when an instruction needs a result that has not yet been written back, control hazards arise when a branch changes the program flow, and structural hazards arise when instructions compete for the same resource. Forwarding, stalls and flushes preserve correct execution, but they cost cycles, and their cost varies with the program being run: a dependency-heavy program and a branch-heavy program stress the pipeline differently. A fixed optimization may therefore help one workload, do little for another, and still add hardware.

This work designs and implements a five-stage RISC-V processor on an Artix-7 FPGA that combines standard hazard handling with a performance-monitoring unit and an adaptive workload controller. The main contributions are: (i) a Verilog implementation of the five-stage pipeline with forwarding, load-use stalling and branch flushing; (ii) a monitoring unit that counts cycles, stalls, forwarding events and taken branches and exposes them on an OLED display; (iii) an adaptive workload controller that uses these measurements to guide optimization; and (iv) an evaluation of timing, resource use, estimated power and execution cycles, with baseline and adaptive configurations compared under the same workloads.

II. RELATED WORK

Five-stage RISC-V processors on FPGAs are well documented. Nair and Lalu [4] built an in-order RV32IM core on an Artix-7 with hybrid one-hot and binary state encoding and static branch prediction, running at 65 MHz with 119 mW. Hussain and Sarkar [5] reported an RV32I core on a Basys-3 board that reached 87.86 MHz, 2.78 times faster than a single-cycle equivalent, but it resolved hazards mainly by stalling. Goswami et al. [6] placed a multiply-accumulate unit inside the ALU, at the price of a slightly longer critical path, and Satyanarayana et al. [7] concentrated on low-power hazard detection and operand bypassing.

Other work makes the pipeline itself flexible. Saussereau et al. [8] presented the AsteRISC core, whose pipeline depth can be configured, although the register barriers add logic area. Avinash et al. [9] combined a flexible MIPS pipeline with dynamic frequency scaling for thermal control. Sudeep et al. [10] implemented forwarding multiplexers and flush logic on a Nexys A7 board, where immediate load-use conflicts still needed a stall.

Branch handling and memory behavior have received much attention. Tiwari et al. [11] paired a gshare predictor with a two-stage fetch and reported a 17.6 % performance

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

gain at a high LUT cost. Prathiyangaran et al. [12] used a four-entry instruction cache to reduce fetch stalls in loops. Bhargavi et al. [13] used a two-bit saturating counter with a branch history table and cut a convolution workload from 830 to 404 cycles, after some warm-up. Li et al. [14] reduced branch aliasing with a TXOR-Gselect predictor and reached 3.11 CoreMark/MHz. Zagan and Gaitan [15] integrated a hardware preemptive scheduler, Kumar and Bhattacharya [16] optimized datapath and program-counter logic, Jin et al. [17] decoupled division from the pipeline to save up to 33 cycles per division, and Dewangan et al. [18] used a comparator-based forwarding unit in a MIPS design with 0.999 W on-chip power.

Table I summarizes selected works. Most of them fix their optimization at design time, and each benefit comes with added hardware or complexity. This motivates a design that watches its own execution and adapts, which is the approach examined here. A literature survey of these studies was presented earlier by the authors [19].

TABLE I. COMPARISON OF SELECTED RELATED WORKS

Ref.	Approach and reported result	Limitation noted
[4]	RV32IM on Artix-7, static prediction; 65 MHz, 119 mW	Static prediction limits adaptability in loops
[5]	RV32I on Basys-3; 87.86 MHz, 2.78× single-cycle	Relies on stalls rather than speculation
[8]	AsteRISC with configurable pipeline depth	Register barriers add logic area
[11]	IndiRA with gshare predictor; 17.6 % gain	High LUT cost for history table
[13]	Two-bit counter with BHT; 830 to 404 cycles	Needs warm-up cycles
[14]	NRP with TXOR-Gselect; 3.11 CoreMark/MHz	Complex indexing limits frequency
[18]	MIPS forwarding unit; 0.999 W	Older MIPS ISA, not RISC-V
This work	Five-stage RISC-V with monitoring and adaptive controller; 0.074 W, WNS +3.265 ns, 8.89 % fewer cycles (branch-heavy)	Limited workload set; see Section VI

III. PROPOSED ARCHITECTURE

The processor is a 32-bit pipelined RISC-V design described in Verilog HDL and shown in Fig. 1. It has five stages, four pipeline registers, hazard handling logic, and the monitoring and adaptive units.

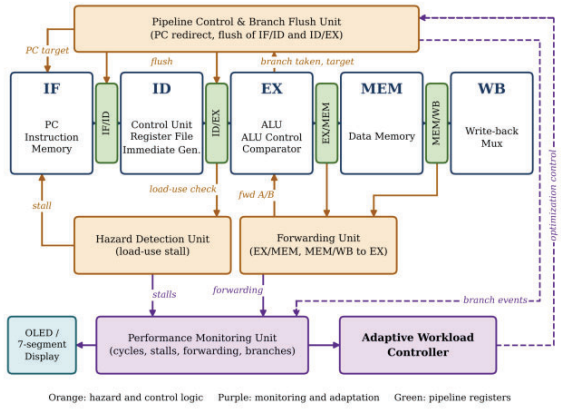


Fig. 1. Block diagram of the self-adaptive, performance-optimized five-stage RISC-V processor.

A. Pipeline Stages

In **IF**, the program counter (PC) addresses the instruction memory, and the PC advances sequentially unless a branch or jump supplies a target. In **ID**, the control unit decodes the instruction, the register file supplies source operands, and the immediate generator extracts and extends the immediate field. In **EX**, the ALU, steered by the ALU control unit, performs arithmetic and logical operations, computes load and store addresses, and a comparator evaluates branch conditions. In **MEM**, loads read and stores write the data memory; other instructions pass their results through. In **WB**, the result, taken either from the ALU or from data memory, is written to the destination register.

B. Pipeline Registers

The IF/ID, ID/EX, EX/MEM and MEM/WB registers carry instructions, operands, immediates, results and control signals between stages. They let a different instruction occupy each stage in every clock cycle.

C. Hazard Handling

A forwarding unit selects operands directly from the EX/MEM or MEM/WB registers instead of waiting for write-back, which removes most read-after-write stalls. When forwarding cannot supply a value, as when an instruction needs data that a preceding load has not yet read, the hazard detection logic inserts a one-cycle stall. When a branch is taken, the pipeline control unit loads the target address into the PC and flushes the wrongly fetched instructions so that they cannot change processor state.

D. Performance Monitoring and Adaptive Control

A performance-monitoring unit counts total clock cycles, stall cycles, forwarding events and taken branches. The adaptive workload controller receives these measurements and uses them to guide the processor's optimization behavior while the hazard logic continues to guarantee correct execution. The measurements are also shown on the OLED display, so the behavior can be observed while the program runs on the board.

IV. FPGA IMPLEMENTATION AND TOOLS

The design was implemented on a Xilinx Artix-7 XC7A35T-1CSG324C FPGA on a development board that also provides a seven-segment display. An external OLED module connected to the board displays the operating mode and the CPI, forwarding, stall, cycle and branch values. The modules (32-bit adder, ALU, register file, program counter, instruction and data memories, immediate generator, control unit, ALU control unit, comparator and multiplexers) were written in Verilog HDL, verified individually, and then integrated first into a single-cycle processor and then into the five-stage pipeline.

AMD Vivado 2026.1 was used for RTL simulation, synthesis, implementation, and for the timing, utilization and power reports. The power values are Vivado estimates from vectorless analysis at medium confidence level, and all slack values refer to the clock constraint used in the Vivado runs.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

V. RESULTS AND DISCUSSION

A. Functional Verification

RTL simulation confirmed the interconnection of the components and the progress of instructions through the pipeline (Fig. 2). On the board, the program counter was seen to update sequentially, confirming instruction fetch and integration of the architecture (Fig. 3). Table II lists the functional tests; all of them passed. The pipeline status also showed forwarding activity, load-use stalls and taken-branch conditions.

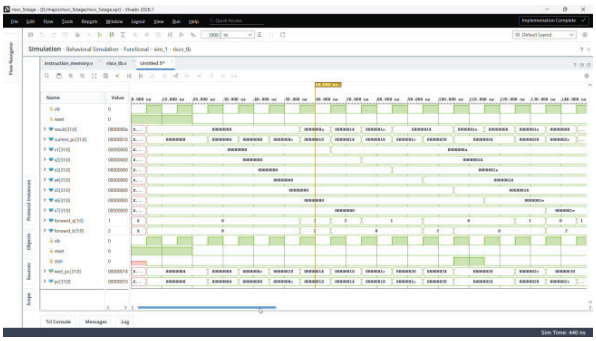


Fig. 2. Vivado behavioral simulation (0–40 ns) showing clock, reset, PC, register values, forwarding selects and the stall signal.

TABLE II. FUNCTIONAL VERIFICATION RESULTS

Test performed	Result
ADD test	PASS
ALU test	PASS
Memory test	PASS
Forwarding test	PASS
Load-use hazard test	PASS
Branch test	PASS
Branch flush test	PASS
Performance monitor test	PASS

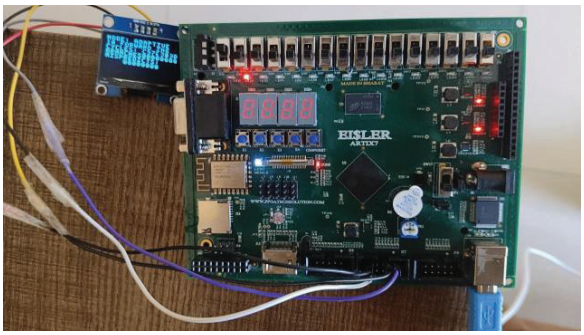


Fig. 3. Five-stage processor running on the Artix-7 board with the OLED display connected.

B. Performance Monitoring

For the monitored program, the unit recorded the values in Table III. One stall cycle and one taken branch were registered, and eight forwarding events show that dependent instructions received their operands from later pipeline stages instead of stalling.

TABLE III. PERFORMANCE MONITORING RESULTS

Parameter	Observed value
Total clock cycles	50
Stall cycles	1
Forwarding events	8
Taken branches	1

C. FPGA Resource Utilization

The synthesized design occupies a small part of the device (Table IV): 7.76 % of the slice LUTs and 3.61 % of the slice registers. Most of the Artix-7 XC7A35T therefore remains available for extensions such as caches, a branch predictor or peripherals.

TABLE IV. FPGA RESOURCE UTILIZATION

Resource	Used	Available	Util. (%)
Slice LUTs	1615	20800	7.76
LUT as logic	1487	20800	7.15
LUT as memory	128	9600	1.33
Slice registers	1502	41600	3.61
F7 muxes	256	16300	1.57
F8 muxes	96	8150	1.18

D. Timing Analysis

Table V compares the Vivado timing reports before and after optimization. Before optimization the worst negative slack (WNS) was −98.841 ns, with two failing setup endpoints, so the timing constraints were not met. After optimization the WNS is +3.265 ns with no failing endpoints and all user-specified constraints met. The improvement,

$$\Delta WNS = WNS_{\text{after}} - WNS_{\text{before}}$$

$$= 3.265 - (-98.841)$$

$$= \boxed{102.106 \text{ ns}} \quad (1)$$

The hold slack remained positive in both runs. The total number of timing endpoints differs between the two runs (605 before, 411 after), which indicates that the optimized implementation is structurally different from the baseline. The clock period used for the constraint is not part of these reports, so a maximum operating frequency is not claimed here.

TABLE V. TIMING SUMMARY BEFORE AND AFTER OPTIMIZATION

Parameter	Before	After
Worst negative slack (WNS)	−98.841 ns	+3.265 ns
Total negative slack (TNS)	−196.932 ns	0.000 ns
Failing setup endpoints	2	0
Worst hold slack (WHS)	0.176 ns	0.164 ns
Total setup endpoints	605	411

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

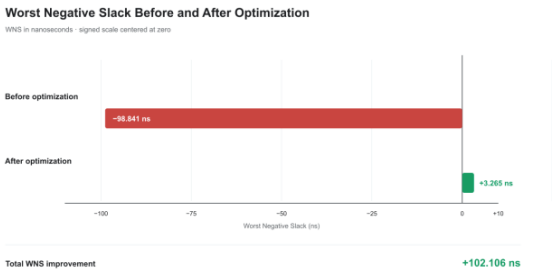


Fig. 4. Worst negative slack before and after optimization.

E. Power Analysis

Table VI gives the Vivado estimates. Total on-chip power fell from 0.084 W to 0.074 W, while dynamic power fell from 0.012 W to 0.002 W. The device-static power stayed at 0.072 W in both cases. The reduction in total power,

$$\text{Power Reduction} = \frac{P_{\text{baseline}} - P_{\text{optimized}}}{P_{\text{baseline}}} \times 100$$

$$= \frac{0.084 - 0.074}{0.084} \times 100$$

$$= \boxed{11.90\%} \quad (2)$$

and the reduction in dynamic power is about 83.33 %. Because static power dominates and is unchanged, the saving in total power is modest, and the values are estimates obtained at medium confidence level, not measurements on the board.

TABLE VI. POWER CONSUMPTION COMPARISON

Parameter	Baseline	Optimized
Total on-chip power	0.084 W	0.074 W
Dynamic power	0.012 W	0.002 W
Device-static power	0.072 W	0.072 W
Junction temperature	25.4 °C	25.4 °C
Ambient temperature	25.0 °C	25.0 °C

F. Workload Execution Comparison

The baseline and adaptive processors were run on five workloads and on a separate branch-heavy workload (Table VII and Fig. 5). Workload 1 took 44 cycles on the baseline and 40 on the adaptive processor, a reduction of 9.09 %. The branch-heavy workload dropped from 45 to 41 cycles, which gives

$$\text{Cycle Reduction} = \frac{45 - 41}{45} \times 100$$

$$= \boxed{8.89\%} \quad (3)$$

Workloads 0, 2, 3 and 4 needed the same 40 cycles in both configurations. The benefit of the adaptive processor therefore depends on the workload: it appears where the optimization is effective and costs no extra cycles elsewhere.

TABLE VII. BASELINE AND ADAPTIVE EXECUTION CYCLES

Workload	Baseline	Adaptive
Workload 0	40	40
Workload 1	44	40
Workload 2	40	40
Workload 3	40	40
Workload 4	40	40
Branch-heavy	45	41

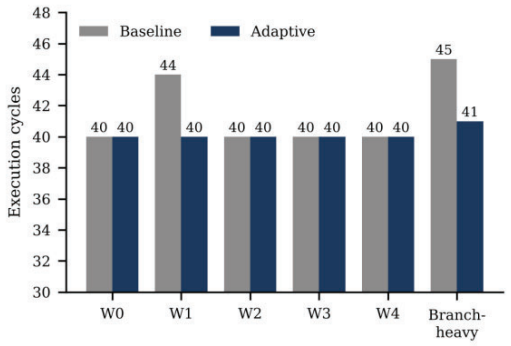


Fig. 5. Execution cycles of the baseline and adaptive processors (W0–W4: workloads 0–4).

G. Adaptive Optimization Verification

The adaptive optimization was verified on 25 test cases comparing execution cycles, stalls, forwarding events, branches and mispredictions. All 25 passed with no failures, which supports the functional correctness of the tested adaptive scenarios. Together with the board demonstration in Fig. 3, the workload results, timing reports and power estimates evaluate the processor at both functional and implementation level.

VI. LIMITATIONS

The evaluation has limits that should be kept in view. The workloads are short programs of about 40 to 45 cycles, and standard benchmarks such as CoreMark were not run, so general performance is not established. Gains were seen on two of six workloads. Power values are Vivado estimates and not board measurements. Slack figures depend on the clock constraint, and the baseline and optimized runs differ in endpoint count. Resource utilization is reported for the synthesized design and does not separate the cost of the adaptive logic from that of the baseline.

VII. CONCLUSION AND FUTURE SCOPE

A five-stage pipelined RISC-V processor with forwarding, load-use stalling, branch flushing, performance monitoring and an adaptive workload controller was designed in Verilog and implemented on an Artix-7 FPGA. All functional tests and all 25 adaptive verification cases passed. The design uses 7.76 % of the slice LUTs. After optimization the worst negative slack improved from −98.841 ns to +3.265 ns and the estimated power fell by 11.90 %, and the adaptive processor reduced cycles by 9.09 % and 8.89 % on the two workloads where it helped, while leaving the others unchanged.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Future work includes dynamic branch prediction, instruction and data caches, adaptive logic driven by more metrics, clock gating and operand isolation to reduce dynamic power, a larger instruction set, evaluation with standard benchmarks and larger workloads, and an ASIC-oriented implementation flow. With such validation, the architecture could be explored for embedded, IoT and edge-computing, robotics and biomedical control applications.

REFERENCES

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanović, "The RISC-V Instruction Set Manual, Volume I: User-Level ISA," EECS Dept., Univ. of California, Berkeley, Tech. Rep. UCB/EECS-2014-54, 2014.

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

[3] D. M. Harris and S. L. Harris, *Digital Design and Computer Architecture: RISC-V Edition*, Morgan Kaufmann, 2021.

[4] P. Nair V. M. and V. Lalu, "Design and Implementation of 5-Stage Pipelined RISC-V Processor on FPGA," in *Proc. 28th Int. Symp. VLSI Design and Test (VDAT)*, Trivandrum, India, 2024.

[5] F. Hussain and S. Sarkar, "Design and FPGA Implementation of Five Stage Pipelined RISC-V Processor," in *Proc. 2024 IEEE 9th Int. Conf. Convergence in Technology (I2CT)*, Pune, India, 2024.

[6] A. Goswami *et al.*, "Design and Implementation of A DSP-Enhanced RISC-V Pipelined Processor Using Verilog HDL," in *Proc. 2025 13th Int. Conf. Intelligent Systems and Embedded Design (ISED)*, 2025.

[7] B. V. V. Satyanarayana, N. Madhu, and K. Srinivas, "Design and Implementation of High-Performance RISC-V Processor with 5 Stage Pipeline on FPGA," in *Proc. 3rd Int. Conf. Self Sustainable Artificial Intelligence Systems (ICSSAS)*, 2025.

[8] J. Saussereau, C. Leroux, J. B. Begueret, and C. Jego, "Design and Implementation of a RISC-V core with a Flexible Pipeline for Design Space Exploration," in *Proc. 30th IEEE Int. Conf. Electronics, Circuits and Systems (ICECS)*, 2023.

[9] Avinash N J *et al.*, "Design and Implementation of 32-Bit MIPS RISC Processor with Flexible 5-Stage Pipelining and Dynamic Thermal Control," in *Proc. 2023 Int. Conf. Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, 2023.

[10] S. Sudeep *et al.*, "Efficient Hazard Handling in a 5-Stage Pipelined 32-Bit RISC-V Processor," in *Proc. 2025 2nd Int. Conf. Software, Systems and Information Technology (SSITCON)*, 2025.

[11] A. Tiwari *et al.*, "IndiRA: Design and Implementation of a Pipelined RISC-V Processor," in *Proc. 33rd Int. Conf. Radioelektronika*, 2023.

[12] K. K. Prathiyangaran, S. Harikrishnan, and R. K. Megalingam, "Optimized 5-Stage RISC-V Pipeline with Lightweight Instruction Cache for Embedded Systems," in *Proc. 4th Int. Conf. Power Electronics and IoT Applications in Renewable Energy and its Control (PARC)*, 2026.

[13] M. B. Bhargavi *et al.*, "Optimizing Convolutional Image Processing on RISC-V Using Branch Prediction on FPGA," in *Proc. TENCON 2024 – IEEE Region 10 Conf.*, 2024.

[14] H. Li, C. Jing, and J. Liu, "Performance-Optimised Design of the RISC-V Five-Stage Pipelined Processor NRP," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 2, 2024.

[15] I. Zagan and V. G. Gaitan, "System Verification and FPGA Implementation of Hardware Preemptive Scheduler for RISC-V Processor," *IEEE Access*, vol. 13, 2025.

[16] R. Kumar and S. Bhattacharya, "Optimal Design for High Performance Computing Systems Based on 5-Stage Pipeline 32-bit RISC-V Processor," *Commun. Appl. Nonlinear Anal.*, vol. 32, no. 3, 2025.

[17] Z. Jin, T. Hu, Z. Jie, and P. Wang, "Research and Implementation of Performance Optimization Methods for RISC-V Level-5 Processors," *Appl. Sci.*, vol. 15, 2025.

[18] G. K. Dewangan, G. Prasad, and B. C. Mandi, "Design and Implementation of 32 bit MIPS based RISC Processor," in *Proc. 2021 8th Int. Conf. Signal Processing and Integrated Networks (SPIN)*, 2021.

[19] P R Prajwal Kumar, Y. Gowda G, R. Y, J. M, and B. Gururaj, "Literature Survey on Design and Implementation of a Self Adaptive & Performance Optimized Five Stage RISC Processor on FPGA," *J. Emerging Technologies and Innovative Research (JETIR)*, vol. 13, no. 4, Apr. 2026, paper JETIR260A4012.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)