An RV32IM Processor with a Fuzzy Logic Branch Prediction Unit and Selective Execution

Sonia Munezero
Department of Electrical & Electronics Engineering, Pan African University Institute for Basic Sciences, Technology, and Innovation (PAUSTI), Juja, Kenya
Soniamunezero23@gmail.com (corresponding author)

Nicasio Maguu Muchuka
Department of Electrical & Control Engineering, Egerton University, Nakuru, Kenya
maguunic@gmail.com

Peter Kamita Kihato
Department of Electrical and Electronics Engineering, Jomo Kenyatta University of Agriculture and Technology (JKUAT), Juja, Kenya
pkihato@jkuat.ac.ke

Received: 14 December 2025 | Revised: 8 January 2026 | Accepted: 17 January 2026

 DOI: https://doi.org/10.48084/etasr.16946

ABSTRACT

The Reduced Instruction Set Computer V (RISC-V) 32-bit architecture with an Integer base instruction set and Multiplication and Division extension (RV32IM) is widely adopted in embedded systems; however, pipeline control hazards limit its performance. Conventional branch prediction techniques exhibit limited accuracy under complex, irregular control-flow patterns. This work proposes the integration of the Fuzzy Logic Branch Prediction Unit (FLBPU) into a five-stage RV32IM pipeline. The FLBPU uses an adaptive fuzzy inference system, considering instruction type and global branch history, to calculate prediction confidence dynamically. This confidence metric controls speculative execution via a Selective Execution Technique (SET). The design also incorporates a hazard control unit and enhanced dependency analysis. On a Xilinx Zynq UltraScale+ Field-Programmable Gate Array (FPGA) at 100 MHz, the design was tested with 83 instructions executed over 135 cycles, achieving an Instructions Per Cycle (IPC) of 0.615. The FLBPU processed 32 branch instructions, predicting all 32 branches with 26 correct predictions and 6 mispredictions, yielding an 18.75% misprediction rate and 81.25% branch prediction accuracy. Accuracy progression was observed as follows: branches one to four had 0% accuracy during the initial learning phase; branches five to fifteen showed gradual improvement from 0% to 60% accuracy; branch 24 reached 75% accuracy; and branch 32 achieved 81.25% accuracy. Post-synthesis hardware utilization was 3,385 Lookup Tables (LUTs), 5,387 Flip-Flops (FFs), and 5 Digital Signal Processing (DSP) blocks, with an on-chip power consumption of 0.309 W.

Keywords-RISC-V; branch prediction; fuzzy logic; adaptive systems; FPGA implementation; pipeline hazards; embedded processors; computer architecture

I. INTRODUCTION

The adoption of the open-source Reduced Instruction Set Computer V (RISC-V) Instruction Set Architecture (ISA) has accelerated the design of embedded and real-time systems [1]. The RISC-V 32-bit architecture with an Integer base instruction set and Multiplication and Division extension (RV32IM) is now integrated into embedded platforms that require defined execution time and controlled power consumption [2]. In these designs, the control hazards remain a central limitation. Branch instructions interrupt sequential execution and generate stalls that reduce pipeline throughput [3].

Research on branch prediction began with basic bimodal structures and progressed toward correlation-based and hybrid predictors. Early predictors that used 2-bit saturating counters reached approximately 85% accuracy but showed limited capability with non-uniform branch sequences [3]. Gshare predictors combined global history with branch address indexing to increase accuracy, with segmented versions attaining a 1.2% improvement while retaining compact pattern-history tables suitable for RV32IM cores [2]. Hybrid and tournament predictors integrated local and global components. A recent RISC-V predictor achieved 93.09% accuracy on CoreMark and reduced Misses Per Thousand Instructions

(MPKI) to 14.373 by using multi-source selection logic and speculative update rules [3].

Expanded predictors incorporated longer history lengths and new computational models. Tagged Geometric Length (TAGE) predictors increased accuracy by up to 10% relative to enhanced Gshare variants and reduced performance loss in nested loop conditions by 4.2% [4]. Machine-learning predictors, including perceptron-based designs, produced approximately 93% accuracy but required additional circuit capacity [5]. The Machine-Learning-based Local Virtual Address (ML-LVA) predictor increased memory throughput by 73% with only a 5.09% increase in area [5]. Alternative predictors for real-time use, such as the Simplified Local Branch History Predictor (SLBHP), increased accuracy from 61.3% to 73.96% while preserving deterministic timing characteristics [6].

Architectural work on RISC-V pipelines advanced in parallel with prediction research [7]. Initial RV32IM implementations established configurable pipelines that adjusted functional unit allocation and stage depth to maintain execution-time and power limits in Internet of Things (IoT) workloads [8, 9]. Modified predictor and issue structures provided measurable improvements. A segmented XOR-based Gshare predictor in an out-of-order model produced a 1.2% accuracy increase relative to the standard Gshare [2]. A dual-issue RISC-V architecture for control applications reduced cycle count by 79% compared to commercial reference designs [10, 11]. Superscalar extensions, such as the CVA6S+, improved the number of Instructions Per Cycle (IPC) by 43.5% through refined prediction and memory pipelines [12]. Additional multithreaded microcontroller designs that increased the number of arithmetic-logic units from two to four achieved IPC increases between 76% and 154.3%, with area overheads ranging from 12% to 32.6% [13].

Selective execution and speculation control mechanisms were introduced to reduce unnecessary activity in mispredicted paths. Confidence-based controls limited speculation when prediction confidence dropped below fixed values. In dual-core RISC-V systems, these controls improved instruction fetch and forwarding efficiency while sustaining real-time bounds [10, 14]. Dependency analysis techniques progressed from basic forwarding to classification systems that distinguished data and control hazards with execution-time impact; coordinated compiler scheduling and hardware detection produced performance gains between 11% and 30% while maintaining bounded hardware expansion [15]. Additional methods reduced division-related dependency delays by more than 75% or increased prediction accuracy by 12% through simplified local history tracking with low circuit complexity [6].

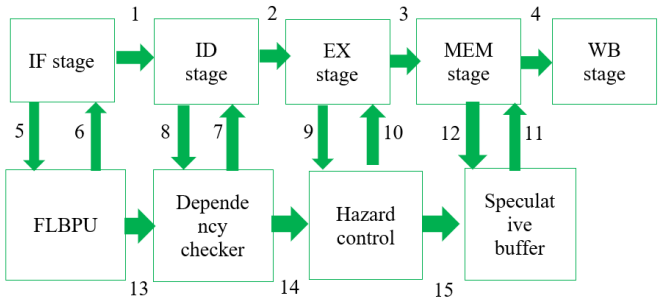
Although these developments improved prediction accuracy, execution timing, and hardware efficiency, significant gaps remain. First, current branch predictors do not adjust internal parameters at runtime to reflect evolving branch-pattern behavior, resulting in misprediction rates of 7-15% in RV32IM pipelines [2, 3]. Second, speculation control methods do not consistently use prediction confidence values when determining whether to execute or halt speculative paths [6, 10]. Third, hazard detection units operate with fixed rules and do not integrate confidence evaluation, which leads to unnecessary stalls or loss of throughput opportunities in embedded systems with strict area and power constraints [4, 5]. Furthermore, speculative execution consumes 25-40% additional energy even when predictions have low reliability, which is critical for IoT platforms with defined energy limits [6, 8].

These issues establish a clear research gap in adaptive control hazard mitigation for embedded RISC-V processors. To address this gap, the present study introduces a unified method that combines prediction confidence evaluation, selective execution, and real-time dependency analysis. The first component, a Fuzzy Logic Branch Prediction Unit (FLBPU), applies fuzzy inference to adjust confidence values using measurable runtime features such as global history length, branch offset, and branch type. This adaptive process is not included in conventional predictors [3, 4]. The second component, a Selective Execution Technique (SET), uses the FLBPU confidence output to restrict speculative execution to high-confidence conditions, thereby reducing unnecessary computation and associated energy cost identified in prior work [6, 10]. The third component is a real-time dependency classification module that works with the FLBPU to coordinate pipeline reactions based on detailed hazard identification. This addresses the limitations of static detection techniques [15]. This integrated system links adaptive branch prediction, confidence-controlled execution, and detailed dependency classification. It reduces branch-related stalls and energy overhead in RV32IM processors while ensuring the feasibility of implementation for embedded and IoT systems [1].

II. IMPLEMENTATION METHODS AND TOOLS

This section describes the implementation of the RV32IM pipelined processor and the method for its evaluation. The processor design builds upon a conventional five-stage pipeline [2, 9, 16, 17] with added units for branch prediction, dependency checking, and hazard control.

The architecture in Figure 1 shows the fetch, decode, execute, memory, and writeback stages. The stages are supported by the FLBPU, a dependency checker, a hazard control unit, and a speculative buffer.



1. Instruction
2. Control signals
3. ALU results
4. Read data
5. Update BTB
6. Prediction
7. Independent flag
8. Dependency
9. Branch Results
10. Forwarding
11. Commit
12. Speculative
13. Confidence
14. Hazard Info
15. Flush/Stall

Fig. 1. Enhanced pipeline architecture.

The processor implementation includes these additions to the RV32IM instruction set:

1. Pipeline registers with confidence scores and dependency metadata.
2. A dual-mode operation for real-time and high-performance execution.
3. A forwarding network with added data paths.
4. A speculative execution infrastructure that includes execution and rollback functions.

Table I presents the enhancements for each pipeline stage.

A. Architectural Components

The core architectural changes from conventional RISC-V implementations [9, 18] are constituted by three components:

- The FLBPU uses a fuzzy inference system instead of conventional predictors.
- The selective execution controller enables dependency awareness and confidence-based speculation.
- The enhanced hazard detection integrates prediction confidence into stall and flush decisions.

1) Fuzzy Logic Branch Prediction Unit

The FLBPU uses a fuzzy inference system that adjusts prediction confidence during operation [2, 3]. This adaptive fuzzy logic approach is inspired by proven Field-Programmable Gate Array (FPGA)-based controllers used in energy management systems, such as hybrid renewable energy controllers [19] and photovoltaic Maximum Power Point Tracking (MPPT) systems [20], which handle dynamic uncertainties through real-time fuzzy inference. Translating this methodology to branch prediction enables the FLBPU to dynamically adapt to runtime branch behavior. The unit contains a Global History Register (GHR), a Fuzzy Inference Engine (FIE), and a Branch Target Buffer (BTB).

TABLE I. PIPELINE STAGE ENHANCEMENTS

No	Stage	Conventional function	Enhancements	Key benefits
1	Fetch	Instruction fetch & PC update	FLBPU integration & BTB	Reduced control hazards
2	Decode	Instruction decode & register read	Dependency analysis & confidence scoring	Better speculation decisions
3	Execute	ALU operations & branch resolution	Enhanced forwarding & prediction detection	Faster hazard resolution
4	Memory	Load/store operations	Speculative buffer interface	Reduced memory stalls
5	Writeback	Register file update	Selective commit & performance monitoring	Energy efficiency

Figure 2 shows the FLBPU architecture. The GHR tracks branch patterns, the FIE applies fuzzy rules to generate confidence levels, and the BTB stores target addresses:

- The GHR classifies branch history into six patterns for prediction decisions, moving beyond saturating counters [2].
- The FIE applies fuzzy rules to generate prediction confidence values.
- The BTB has 64 entries with a Least Recently Used (LRU) replacement policy. The unit caches frequently accessed branch targets to reduce thrashing [3].

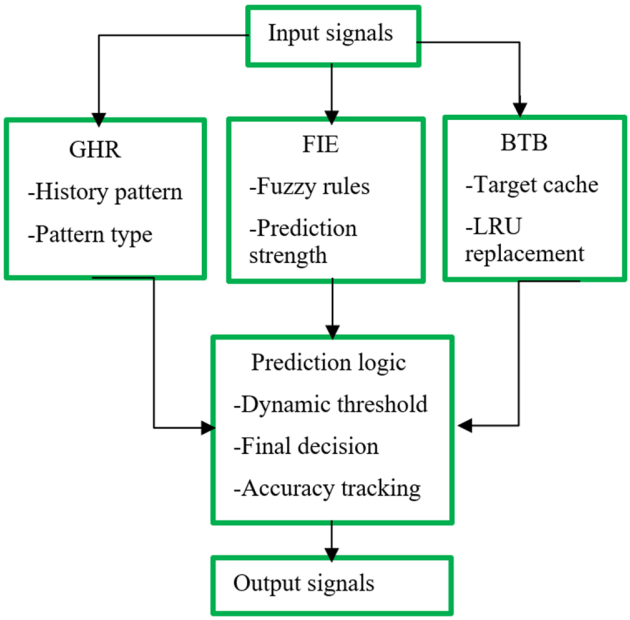


Fig. 2. FLBPU architecture.

Figure 3 shows the FIE process flow. The engine receives inputs, including global history, branch offset, and instruction type. A fuzzification unit computes input strengths. The fuzzy logic rules system evaluates these strengths. The rule firing and aggregation stage selects a dominant rule. A defuzzification process calculates the final prediction strength.

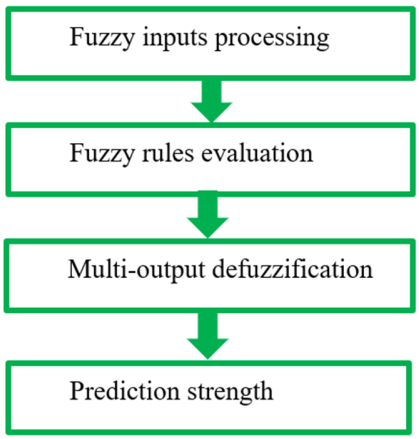


Fig. 3. FIE process flow.

2) Selective Execution and Hazard Control Mechanism

The selective execution mechanism uses a dependency analysis system that classifies operand relationships into seven types for precise speculation control [15]. Its control logic applies a state machine that adjusts speculation strength using confidence values from the FLBPU, together with the dependency classifications, enabling deterministic speculation management [6, 10]. A speculative buffer stores temporary results and applies commit rules that prioritize independent instructions to maintain correct execution order [10]. Combined with these components, the hazard control unit detects data, control, and structural hazards and integrates FLBPU confidence metrics into pipeline regulation. To further reduce stall cycles, an advanced forwarding network provides multiple routing paths for operand values, reducing latency caused by dependency delays [15].

B. Tools

The processor was validated through simulation and FPGA implementation. The simulation used Icarus Verilog (v11.0) for Hardware Description Language (HDL) simulation. It also used GTKWave (v3.3.108) for waveform analysis.

The toolchain configuration was as follows:

- Icarus Verilog v11.0 for simulation.
- GTKWave v3.3.108 for waveform analysis.
- Custom Verilog testbench with modular test cases for testing.
- Hexadecimal memory files (memfile.hex and data_mem.hex) for memory initialization.
- Integrated counters for IPC, branch accuracy, and speculation efficiency for performance monitoring.
- The design was synthesized for the Xilinx Zynq UltraScale+ FPGA using Xilinx Vivado Design Suite to validate resource utilization [21].

The comparison method used identical test programs and simulation infrastructure. Each predictor used consistent Pattern History Table (PHT) sizes. For the FPGA comparison, each predictor was synthesized for the Xilinx Zynq UltraScale+ FPGA under the same timing constraints. The measurement of prediction accuracy, misprediction penalty reduction, and hardware overhead used Lookup Table (LUT), Flip-Flop (FF), and Block Random-Access Memory (BRAM) utilization data.

III. RESULTS AND ANALYSIS

The RV32IM processor, which has an integrated FLBPU and uses SET, was evaluated through Register Transfer Level (RTL) simulation, followed by FPGA implementation. Validation was performed using Icarus Verilog v11.0 for simulation and GTKWave v3.3.108 for waveform analysis at a clock frequency of 100 MHz. The testbench executed an 83-instruction program containing 32 conditional branches, with the instruction memory initialized from hexadecimal files. During simulation, the processor completed the 83 instructions in 135 cycles, achieving an IPC rate of 0.615. The FLBPU processed the 32 branch instructions with an accuracy progression of 0% for the first four branches during the initialization of the learning phase, rising to 60% by the 15th branch, 75% by the 24th branch, and finally reaching 81.25% after all 32 branches had been processed. This progression resulted in 26 correct predictions and six mispredictions, yielding a misprediction rate of 18.75%. Table II documents these runtime metrics, which were extracted from the simulation.

TABLE II. SIMULATION PERFORMANCE METRICS

Metric	Value	Measurement source
Clock cycles	135	Simulation cycle counter
Instructions executed	83	Writeback completion count
IPC	0.615	Calculated (IPC/cycles)
Branch instructions	32	Branch instruction count
Correct predictions	26	Execute verification
Prediction accuracy	81.25%	Calculated (correct/branches)
Misprediction rate	18.75%	Calculated
Learning duration	70 cycles	Learning monitor
BTB hit rate	61%	BTB statistics

Waveform analysis verified FLBPU adaptation mechanisms. During initial learning at cycle 21, the BTB contained a limited number of entries (Figure 4). The value of 3 in the GHR produced pattern classification (Figure 5), whereas the inference engine operated with a prediction strength of 61, which exceeded the dynamic threshold of 53 (Figure 6). This caused initial mispredictions. During this period, the hazard control unit processed data dependencies without activating forwarding paths (Figure 7), thereby maintaining pipeline flow despite the mispredictions. Meanwhile, the prediction unit state (Figure 8) showed 0% accuracy and a confidence value of 61. Meanwhile, Uninterrupted instruction progression and gradual improvement in accuracy were confirmed by complete pipeline waveforms (Figure 9). The progression of prediction accuracy across the 32 branch instructions is illustrated in Figure 10, showing the transition from 0% during initial learning to 81.25% at completion.

The design was synthesized for a Xilinx Zynq UltraScale+ FPGA using the Vivado Design Suite. Post-synthesis analysis measured 0.309 W total on-chip power consumption with 3,385 LUTs, 5,387 FFs, and 5 Digital Signal Processing (DSP) blocks utilized (Tables III–V). Dynamic power constituted 0.088 W (28% of total), with I/O power representing 69% of dynamic consumption. Device static power was 0.221 W (72% of total).

Table VI compares the FLBPU and SET implementation with other contemporary RISC-V approaches. FLBPU demonstrates adaptive learning with an accuracy range of 0% to 81.25% within 32 branches, in contrast to static predictors [2, 3]. This performance surpasses that of simplified real-time predictors (73.96% accuracy) while maintaining moderate hardware overhead. Compared to hybrid predictors [3], which achieve 93.09% accuracy at the cost of increased complexity, the FLBPU strikes a balance between improving accuracy and being efficient with resources. The integrated SET utilizes FLBPU confidence metrics for speculation control and represents an approach that is not implemented in the previous works.

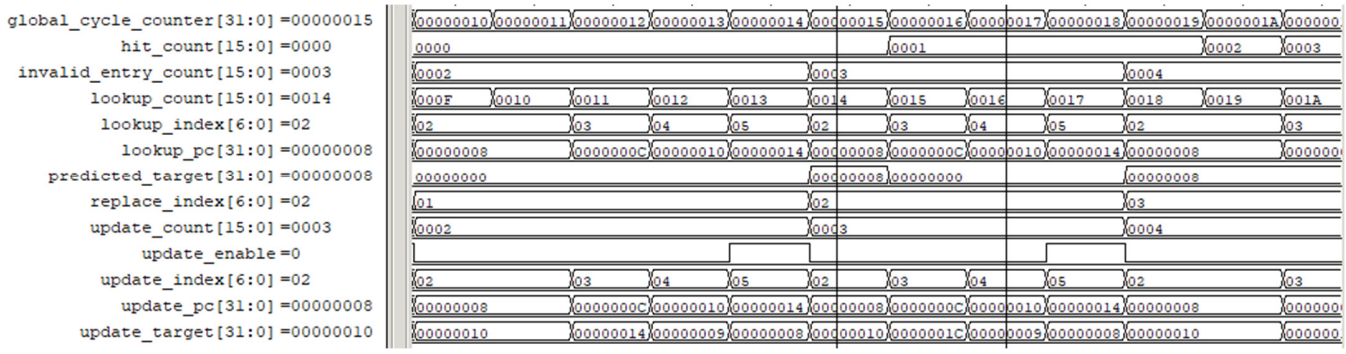


Fig. 4. BTB state.

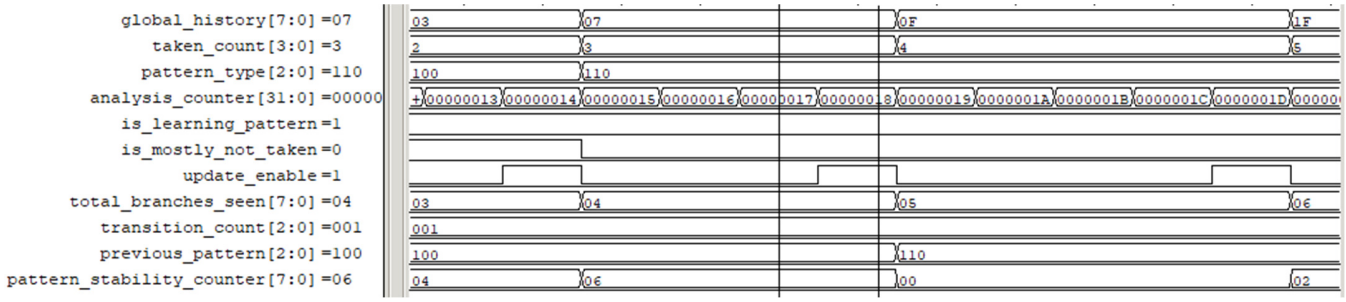


Fig. 5. Global history and pattern analysis.

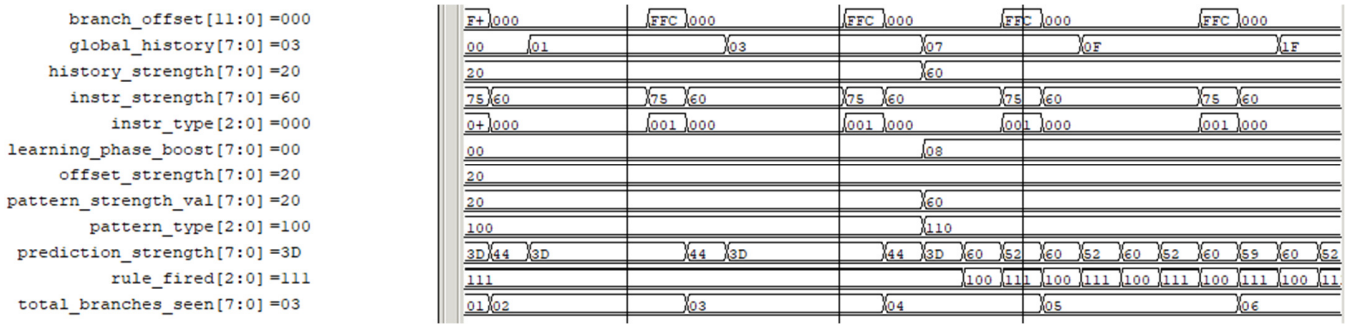


Fig. 6. Inference engine state.

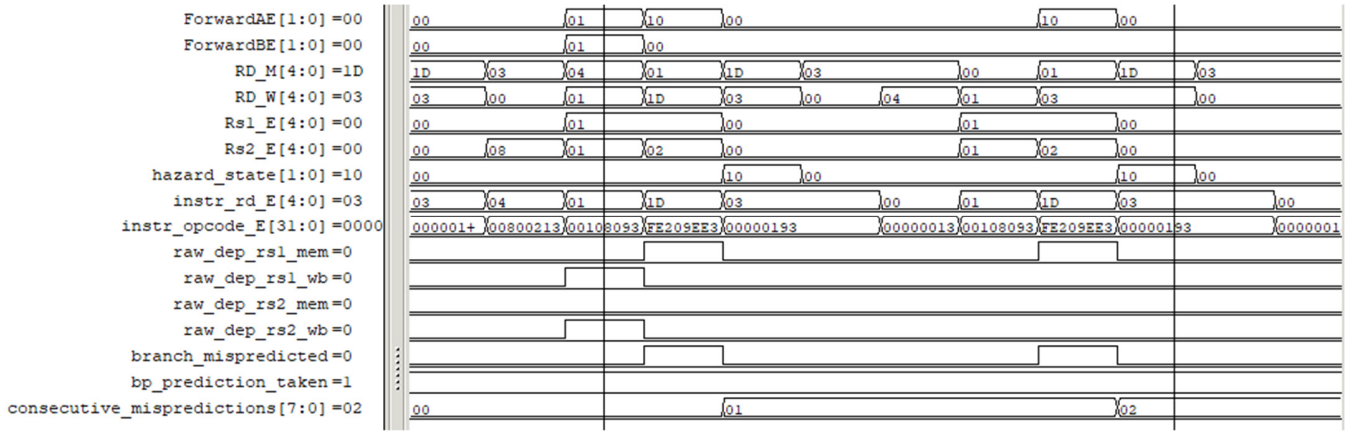


Fig. 7. Hazard and dependency signals.

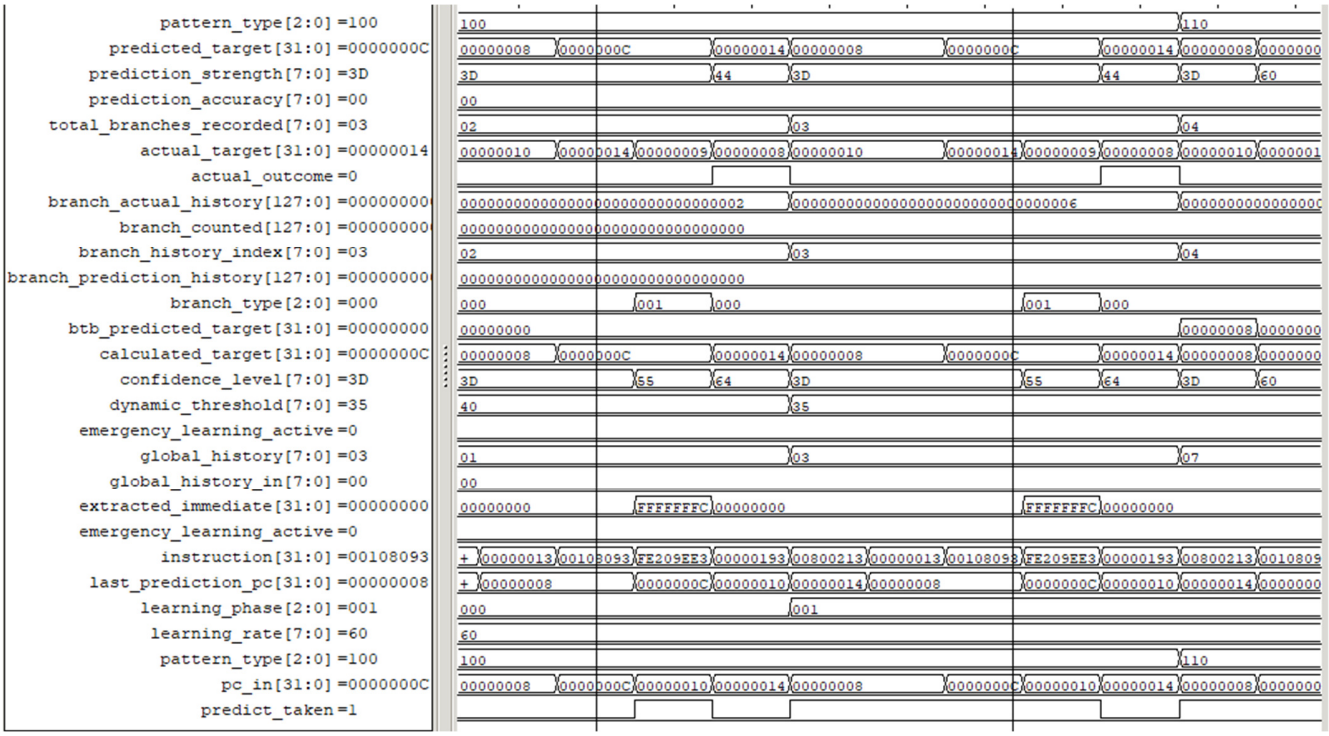


Fig. 8. Prediction unit state.

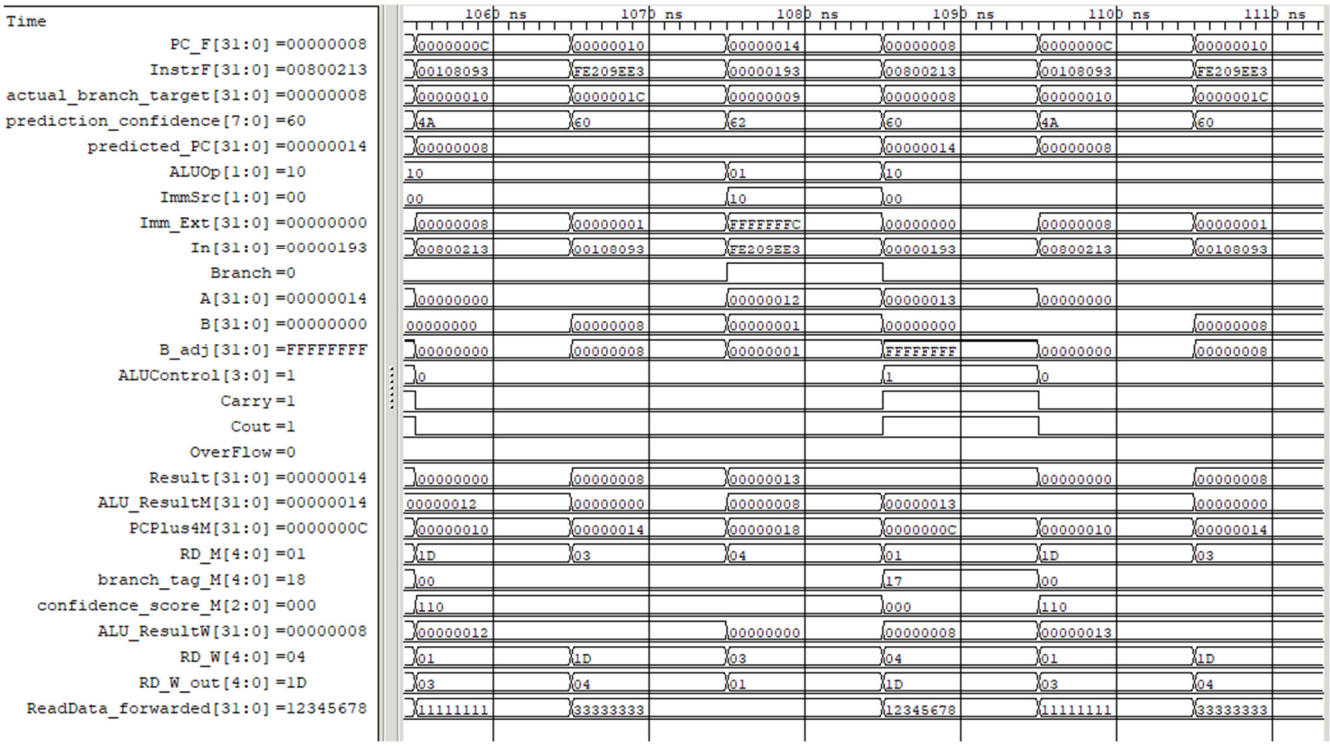


Fig. 9. Full pipeline and ALU waveform.

The FLBPU implementation provides runtime adaptation, contrasting with conventional algorithms [2-4]. The SET enables energy-aware speculation control, addressing the 25–40% additional energy consumption from low-reliability speculation identified in prior work [6, 8]. Design constraints include the BTB learning phase penalty and conservative execution strategy that may limit IPC improvement compared to more speculative approaches.

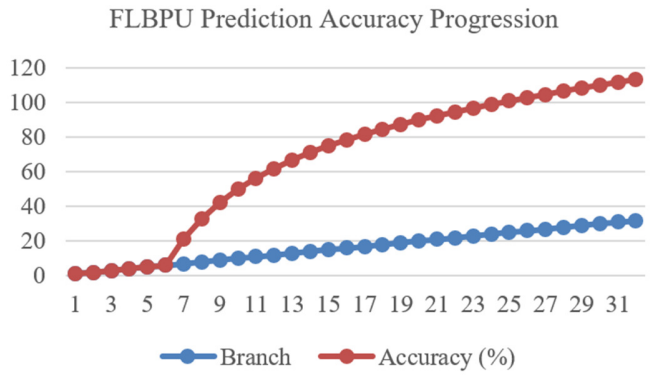


Fig. 10. FLBPU prediction accuracy learning progression (over 32 branches).

TABLE III. POST-SYNTHESIS FPGA POWER SUMMARY

Parameter	Value
Total on-chip power	0.309 W
Junction temperature	25.7 °C
Thermal margin	74.3 °C (31.3 W)
Effective SJA	2.3 °C/W

SJA = Static Junction-to-Ambient thermal resistance.

TABLE IV. FPGA DYNAMIC AND STATIC POWER BREAKDOWN

Category	Power (W)	Percentage (%)
Dynamic power	0.088	28
↳ Clocks	0.013	15
↳ Signals	0.008	9
↳ Logic	0.005	6
↳ DSP blocks	0.001	1
↳ I/O	0.061	69
Device static power	0.221	72

TABLE V. FPGA RESOURCE UTILIZATION

Resource	Available	Utilization
LUT	47,232	3,385
LUTRAM	28,800	32
FF	94,464	5,387
DSP blocks	240	5
I/O	252	90

TABLE VI. COMPARISON WITH RELATED RISC-V WORKS

Work	Focus	Accuracy (%)	Hardware cost	Key metric
This work	Adaptive FLBPU + SET	81.25 (learned)	3,385 LUTs	0.309 W
[2]	Segmented XOR-Gshare	~90–93 (static)	-	Out-of-order performance
[3]	Hybrid predictor	93.09	Higher complexity	MPKI: 14.373
[6]	SLBHP for real-time	73.96	Low complexity	Deterministic timing
[10]	Dual-issue architecture	-	-	79% cycle reduction

IV. CONCLUSION

An enhanced Reduced Instruction Set Computer V (RISC-V) 32-bit architecture with an Integer base instruction set and Multiplication and Division extension (RV32IM) processor, integrating a Fuzzy Logic Branch Prediction Unit (FLBPU) and a Selective Execution Technique (SET) to mitigate control hazards, was implemented and evaluated in this research. The FLBPU employs fuzzy inference to dynamically adjust prediction confidence using global branch history, branch offset, and instruction type, addressing the static parameter limitation of conventional predictors [2-4]. The SET utilizes confidence metrics to restrict speculative execution to high-confidence conditions, reducing computational waste [10].

The design was implemented on a 100 MHz Xilinx Zynq UltraScale+ Field-Programmable Gate Array (FPGA). Simulation results showed that 83 instructions were executed by the processor over 135 cycles, achieving an Instructions Per Cycle (IPC) of 0.615. The FLBPU processed 32 branch instructions, demonstrating adaptive learning with an accuracy range of 0% to 81.25% and a misprediction rate of 18.75%. Analysis of the learning trend revealed a logarithmic regression, estimating a prediction accuracy of 99.4% for 1,000 branches. Post-synthesis, hardware utilization was measured at 3,385 Lookup Tables (LUTs), 5,387 Flip-Flops (FFs), and 5 Digital Signal Processing (DSP) blocks, with an on-chip power consumption of 0.309 W.

Comparative analysis confirms that the FLBPU provides runtime adaptation, which is absent in static predictors [2, 3], while maintaining moderate hardware overhead suitable for embedded systems. The SET enables confidence-controlled speculation, addressing the energy efficiency limitations of conventional pipelines [6, 8]. The integrated architecture strikes a balance between improving prediction accuracy and meeting the resource constraints characteristic of Internet of Things (IoT) and edge computing platforms [1, 8].

Future research directions include: expanding the FLBPU rule base and history length to handle larger workloads, applying confidence-guided execution to dual-issue and out-of-order pipelines; and integrating lightweight machine learning with fuzzy inference to improve pattern recognition while maintaining low hardware complexity.

REFERENCES

[1] E. Cui, T. Li, and Q. Wei, "RISC-V Instruction Set Architecture Extensions: A Survey," *IEEE Access*, vol. 11, pp. 24696–24711, 2023, https://doi.org/10.1109/ACCESS.2023.3246491.

[2] W. Yang, J. Gao, Q. Li, and J. Zhang, "Design of RISC-V out-of-order processor based on segmented exclusive or Gshare branch prediction," *Microelectronics Journal*, vol. 152, Oct. 2024, Art. no. 106334, https://doi.org/10.1016/j.mejo.2024.106334.

[3] D. Li, S. Zhao, X. Zhang, X. Fan, J. Zhou, and Z. Wang, "Design of a Hybrid Branch Prediction Architecture for Risc-V Processors," in *2025 IEEE 14th International Conference on Communications, Circuits and Systems*, Wuhan, China, 2025, pp. 160–164, https://doi.org/10.1109/ICCCAS65806.2025.11102589.

[4] L. Li, L. Xing, X. Peng, and Y. Duan, "Research and Design of a Dynamic Branch Predictor Based on the RISC-V Instruction Set," in *2025 4th International Symposium on Semiconductor and Electronic Technology*, Xi'an, China, 2025, pp. 324–328, https://doi.org/10.1109/ISSET66828.2025.11184914.

[5] A. Aoun, M. Masadeh, and S. Tahar, "ML-based Load Value Approximator for Efficient Multimedia Processing," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 21, no. 7, July 2025, Art. no. 210, https://doi.org/10.1145/3736582.

[6] Z. Jin, H. Di, T. Hu, and P. Wang, "Real-Time Optimization of RISC-V Processors Based on Branch Prediction and Division Data Dependency," *Applied Sciences*, vol. 15, no. 2, Jan. 2025, Art. no. 632, https://doi.org/10.3390/app15020632.

[7] A. Saveau, "Branch Prediction in Hardcaml for a RISC-V 32im CPU." arXiv, Jan. 04, 2024, https://doi.org/10.48550/arXiv.2312.10426.

[8] T. M. Ignatius, S. Bora, and R. P. Palathinkal, "Impact of Pipelining on Low Power IoT applicable RISC-V ISA Core Micro-architectures," in *2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, Kuala Lumpur, Malaysia, 2024, pp. 223–230, https://doi.org/10.1109/MCSoC64144.2024.00045.

[9] S. Ahmed and A. B. M. Harun-Ur-Rashid, "Design, Implementation and Verification of Five Stage Pipeline RISC-V Core (RV32I ISA)," in *2025 International Conference on Electrical, Computer and Communication Engineering*, Chittagong, Bangladesh, 2025, pp. 1–6, https://doi.org/10.1109/ECCE64574.2025.11013913.

[10] Q. Zhai *et al.*, "Instruction Level Parallelism Optimizations in a High-Performance Dual-Issue RISC-V Processor for Real-Time Control Systems," in *2025 IEEE International Symposium on Circuits and Systems*, London, United Kingdom, 2025, pp. 1–5, https://doi.org/10.1109/ISCAS56072.2025.11043760.

[11] M. Rogenmoser *et al.*, "SentryCore: A RISC-V Co-Processor System for Safe, Real-Time Control Applications." arXiv, May 16, 2024, https://doi.org/10.48550/arXiv.2406.06546.

[12] R. Tedeschi *et al.*, "CVA6S+: A Superscalar RISC-V Core with High-Throughput Memory Architecture." arXiv, May 08, 2025, https://doi.org/10.48550/arXiv.2505.03762.

[13] M.-H. Yen, Y.-H. Lin, T.-F. Lin, Y.-H. Chen, Y.-F. Ku, and C.-T. Kao, "Chip Design of Multithreaded and Pipelined RISC-V Microcontroller Unit," *Engineering Proceedings*, vol. 89, no. 1, Feb. 2025, Art. no. 31, https://doi.org/10.3390/engproc2025089031.

[14] Y. Xiang *et al.*, "A Lightweight RISC-V Multi-Core Interaction Framework For Embedded Real-Time Applications," in *2025 IEEE International Symposium on Circuits and Systems*, London, United Kingdom, 2025, pp. 1–5, https://doi.org/10.1109/ISCAS56072.2025.11044187.

[15] R. Olowoniyi, S. Du, R. Baartmans, V. Agostinelli, and L. Chen, "RTL Microarchitecture Optimizations for Reducing Pipeline Hazards in Small RISC Cores." ResearchGate, May 01, 2025, [Online]. Available: https://www.researchgate.net/publication/395173023_RTL_Microarchitecture_Optimizations_for_Reducing_Pipeline_Hazards_in_Small_RISC_Cores.

[16] M. Z. Naveed, M. Amar, A. Hussain, and Z. A. Awan, "Design & Implementation of 5-Stage 32-bit RISC-V Pipeline Processor on FPGA," in *9th Multidisciplinary Student Research & Innovation Conference*, Wah Cantt, Pakistan, 2024.

[17] A. Singh, A. Kumar, A. Singh, R. A. Reddy, and K. N. Pushpalatha, "Design and Implementation of RISC-V ISA (RV32IM) on FPGA," *International Journal of VLSI & Signal Processing*, vol. 10, no. 2, pp. 17–21, July 2023, https://doi.org/10.14445/23942584/IJVSP-V10I2P103.

[18] A. Tiwari, P. Guha, G. Trivedi, N. Gupta, N. Jayaraj, and J. Pidanic, "IndiRA: Design and Implementation of a Pipelined RISC-V Processor," in *2023 33rd International Conference Radioelektronika*, Pardubice, Czech Republic, 2023, pp. 1–6, https://doi.org/10.1109/RADIOELEKTRONIKA57919.2023.10109058.

[19] M. Baloch, A. A. Jumani, A. A. Memon, A. M. Shaikh, and Z. A. Memon, "The FPGA-Based Design of an Optimal Fuzzy Logic Controller for Hybrid Renewable Energy Systems," *Engineering, Technology & Applied Science Research*, vol. 15, no. 6, pp. 28420–28426, Dec. 2025, https://doi.org/10.48084/etasr.12538.

[20] M. Y. Allani, D. Mezghani, F. Tadeo, and A. Mami, "FPGA Implementation of a Robust MPPT of a Photovoltaic System Using a Fuzzy Logic Controller Based on Incremental and Conductance Algorithm," *Engineering, Technology & Applied Science Research*, vol. 9, no. 4, pp. 4322–4328, Aug. 2019, https://doi.org/10.48084/etasr.2771.

[21] A. Batashev, "Dissecting RISC-V Performance: Practical PMU Profiling and Hardware-Agnostic Roofline Analysis on Emerging Platforms," in *17th International Conference on Parallel Computing Technologies*, Almaty, Kazakhstan, 2025, pp. 153–168, https://doi.org/10.1007/978-3-032-06751-7_11.