IJAREEIE

e-ISSN: 2278-8875
p-ISSN: 2320-3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

Volume 14, Issue 12, December 2025

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.807

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE)

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 12, December 2025 ||

| DOI:10.15662/IJAREEIE.2025.1412015|

Design and Implementation of Branch Prediction Unit for Pipeline Processor to Improve CPU Efficiency

Dr. M V Sreenivas Rao[1], Chandana S V[2], Akanksha A[3], Chaya N[4], Anwitha R[5]

Associate Professor, Dept. of ECE, GSSS Institute of Engg. & Tech. for Women, Mysore, Karnataka, India[1]

UG Scholar, Dept. of ECE, GSSS Institute of Engg. & Tech. for Women, Mysore, Karnataka, India[2,3,4,5]

ABSTRACT: Pipelining is a technique used by modern processors to improve performance, but conditional instructions frequently interfere with this seamless execution flow. Branch prediction is essential to increasing the speed and effectiveness of contemporary processors. The processor must choose which path to take when a program comes across a branch instruction, and accurately predicting this helps prevent pipeline delays. The goal of this project is to create a basic Branch predictor using a Branch Target Buffer, a Pattern History Table, and a 2-bit saturating counter. The project shows how intelligent prediction techniques can improve CPU performance in general. The first step in the process is determining how branch instructions affect processor performance and selecting an appropriate prediction method. Following the evaluation of techniques the 2-bit saturating counter approach is chosen due to its precision, speed and minimal hardware demands. The system is segmented into units, like the Branch History Table (BHT) Branch Target Buffer (BTB) prediction logic and update logic. Each module is subsequently crafted using Verilog maintaining modularity and synthesizability. The implementation process starts with writing the RTL code for the branch predictor focusing initially on the components that govern its operation. The Branch History Table (BHT) consists of an array of 2-bit counters that update their states based on actual branch results. Additionally, the Branch Target Buffer (BTB) is designed to hold target addresses along with tags enabling the predictor to fast recognize valid branch records. The prediction mechanism is designed to provide a decision, within one cycle enabling the processor to proceed without causing pipeline delays. To ensure the design works as intended, a testbench is created to apply a variety of branch scenarios Simulations are then run using Cadence Xcelium, where waveforms are reviewed to verify overall prediction accuracy.

KEYWORDS: Branch Prediction, BHT, BTB, RTL, Pipeline Optimization

I.INTRODUCTION

Contemporary processors execute millions of instructions per second. Even a brief delay—such, as waiting to determine if a branch will be followed—can decelerate the whole pipeline. The majority of systems continue to rely on simple static prediction techniques that presume a branch will act identically each time. Sadly, actual programs are not that consistent resulting in mispredictions and avoidable pauses. To tackle this our project develops a VLSI-driven dynamic branch predictor that employs a 2-bit saturating counter combined with a Branch Target Buffer (BTB) to understand and adjust to real program execution. The objective is to minimize pipeline stalls boost instruction throughput and deliver precise predictions while keeping hardware and power costs low. Our objectives include designing the predictor in Verilog, verifying it using realistic testbench patterns, analysing synthesis results, and optimizing the design for better accuracy, lower power, and improved area efficiency. This report is organized in a clear flow, beginning with background and motivation, followed by system architecture and methodology, then implementation, simulation, and synthesis results, and finally concluding with the project's advantages, limitations, and future scope.

II. RELATED WORK

Tang Zhang et.al explored a new way of handling hard-to-predict branches, which traditional predictors like TAGE often struggle with. Even though these branches are rare, they can seriously slow down a processor when predicted incorrectly. Their work introduces a CNN-based branch predictor that looks at patterns in register values to make smarter predictions. By learning hidden relationships between register data and branch behaviour, their model

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 12, December 2025 ||

| DOI:10.15662/IJAREEIE.2025.1412015|

significantly improves accuracy, especially for branches that conventional methods fail to predict well [1]. Wu Yang, Jie Gao et al introduced a RISC-V out-of-order processor architecture employing segmented prediction in the Microelectronics Journal. They addressed the rising performance requirements of processors alongside the constraints of conventional Pattern History Tables (PHT) which often restrict high-performance embedded systems. Their approach enhances branch prediction precision and processor throughput by facilitating efficient out-oforder execution all while ensuring stability through a balance, between prediction latency and execution demands. The study also highlights some limitations: increased hardware complexity, no detailed evaluation of power and area, and limited adaptability across different applications— areas that require further optimization [2].

Pankaj Nair et.al introduced the design and execution of a 5-stage RISC-V processor on FPGA at VDAT 2024 in Trivandrum. They addressed obstacles like the scarcity of RISC-V designs and the substantial FPGA resource needs of microarchitectures. Their research shows that creating a scalable and user-friendly RISC-V processor, on an affordable FPGA is achievable. However, the design has some limitations: it offers restricted instruction-level parallelism, lower accuracy compared to dynamic predictors, and a higher risk of pipeline flushes during branch mispredictions, pointing to areas that could be improved for more advanced applications [3].

Luis A.Q. Villon et.al presented a network-driven branch predictor in Volume 555 Article ID 126637. They noted that conventional neural network predictors typically depend significantly on input data and face challenges with branch patterns resulting in difficulty sustaining steady accuracy, for various branch categories. Their solution uses a WiSARD weightless neural predictor, which improves accuracy and adapts well to certain datasets, making it a promising approach for efficient branch prediction.[4].

Changbiao Yao et.al studied the branch prediction unit of SweRV EH1. They found that low prediction accuracy increases control hazards and hurts processor performance. To improve this, they proposed a hybrid branch predictor achieving 85.9% accuracy on the Power Stone benchmark. While more reliable than traditional designs, it is complex, limited in scope, and depends heavily on sensor inputs, requiring further simplification and validation for real-world use.[5].

Harsha Rastogi et.al, presented "Design Space Exploration of Perceptron Based Branch Predictors for Superconducting CPUs" at ISVLSI 2022. They emphasized that traditional branch predictors are unsuitable for superconducting processors due to constraints, on power, area and temperature. To address this challenge, they introduced SuperBP, a perceptronbased predictor that improves accuracy and decreases misprediction rates by 15.6% to gshare while keeping similar hardware requirements.
However, the design introduces higher complexity, highlighting the trade-off between efficiency and implementation cost.[6].

Hadeel S.H. Mahmood Middle et.al presented “FPGA Configuration of an Alloyed Correlated Branch Predictor Used with RISC Processor” in IJECE. The study tackles performance issues in pipelined RISC processors, where stalls often occur due to nested branch mispredictions that traditional predictors handle poorly. The proposed design combines global history with branch address bits, implemented in VHDL and integrated into a 32-bit MIPS processor. Results show better prediction accuracy, fewer stalls, lower CPI, and higher overall speedup compared to standard branch predictors[7].

Ali S. Al-Khalid et.al introduced "Hybrid Branch Prediction for Pipelined MIPS Processor" in IJECE. Their work targeted the issue of pipeline stalls triggered by branch instructions, which can greatly degrade CPU speed. To solve this, they developed a predictor that integrates various prediction methods and employs a meta-selector to choose the best predictor, for each branch. This method decreases misprediction occurrences accelerates processing and enhances the efficiency of the pipeline when compared to conventional dynamic predictors.
However, the design comes with increased hardware complexity and higher resource demands, making it less ideal for lightweight or low-power systems [8].

Milad Mohammadi et.al presented "Energy On-Demand Dynamic Branch Prediction Models" in IEEE Transactions on Computers. They observed that branch predictor units (BPUs) use an amount of energy since many lookups are, for nonbranch instructions or very biased branches that might be predicted statically. Their approach selectively enables

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 12, December 2025 ||

| DOI:10.15662/IJAREEIE.2025.1412015|

prediction logic only, when necessary, dramatically lowering processor energy consumption while preserving prediction accuracy. Hybrid on-demand approaches also strike a balance between energy conservation and performance. They depend on compiler assistance and introduce complexity in implementation possibly leading to performance compromises, in certain workloads [9].

C. Arul Rathi et.al presented "Design and Development of an Efficient Branch Predictor for an In-order RISC-V Processor" in the Journal of Nano and Electronic Physics. Their work addressed the problem of stalls in, in-order RISC-V pipelines by enhancing prediction accuracy accurately determining branch results from the time onwards. This lowers pipeline stalls and execution time. However, the approach is limited to in-order pipelines, uses a basic prediction architecture, relies only on simulations without power or area analysis, and only partially addresses BTB miss penalties, indicating the need for more comprehensive evaluation[10].

Our study highlights advance in branch prediction units (BPUs) for modern processors. Common challenges include hardware complexity, power consumption, and adaptability across workloads. Performance can be further improved by optimizing predictor selection, balancing accuracy with resource use, and integrating efficient.

III. PROPOSED METHODOLOGY

High-level flow: The Instruction Fetch Unit (IFU) sends the current Program Counter (PC) to the Branch Target Buffer (BTB) and Branch History Table (BHT) to check if the fetched instruction is a known branch and to request a prediction. The BTB provides a cached target address on a hit.

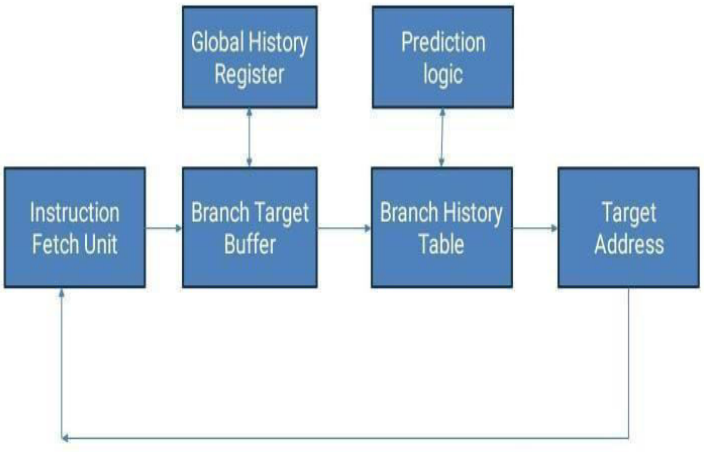


Fig .1: Block Diagram of Branch Prediction Unit for Pipeline Processor

Feeding back to pipeline: If the branch is predicted taken and the BTB hit is valid, the IFU redirects the next fetch to the predicted target address, keeping the pipeline filled and avoiding stalls; otherwise it continues with PC+4 or triggers recovery on misprediction. Actual branch outcomes from the execute stage are fed back to update the Global History Register, BHT counters, and BTB entries, continuously refining prediction accuracy over time as shown in Fig. 1

IV. SOFTWARE DESCRIPTION

1.Coding and HDL Development Tools

These utilities serve the purpose of composing and creating Verilog code, for RTL design.

Typical choices encompass Xilinx Vivado, Cadence Virtuoso VS Code equipped with Verilog extensions and Intel Quartus (when required).

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 12, December 2025 ||

| DOI:10.15662/IJAREEIE.2025.1412015|

2. Simulation Tools

Simulation software assists in verifying if the design functions, by evaluating logic, behaviour and timing. Selected options are Cadence Xcelium, ModelSim/QuestaSim, Vivado Simulator and EDA Playground, for fast online experimentation.

3. Synthesis Tools

These utilities translate RTL code into a level schematic enabling implementation, on FPGA or ASIC. Typical tools consist of Xilinx Vivado for FPGA synthesis along with Synopsys Design Compiler or Cadence Genus, for ASIC processes.

4. Waveform Viewing & Debugging Tools

These tools enable you to observe signal waveforms and troubleshoot problems throughout the simulation. Frequently used options are SimVision and GTKWave.PU faster. Some challenges still exist, such as more complex hardware,higher power use, and difficulty adapting to different workloads.

5. Power and Area Analysis Tools

These tools assist in measuring the power consumption of the design and evaluating the efficiency of hardware resource usage.Typical utilities consist of the Vivado Power Analyzer. When dealing with ASIC tasks, Synopsys PrimeTime PX. They assess factors such as dynamic power, LUT and flip-flop consumption, area as well, as thermal characteristics.

6. Documentation and Reporting Tools

These serve to produce the project report and display the findings clearly. Applications such, as MS Word, Google Docs, LaTeX and several PDF editors assist in structuring and arranging all the project information.

7. Full VLSI Development Flow Support

Together these tools encompass the VLSI design process, which includes:

- Running simulations
- Performing synthesis
- Checking power and area
- Evaluating performance

SOFTWARE IMPLEMENTATION

The software implementation follows the logic represented in the flowchart, where each block is translated into corresponding program statements. The code executes sequentially based on decision and process blocks to ensure correct data processing and desired output.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 12, December 2025 ||

| DOI:10.15662/IJAREEIE.2025.1412015|

1.Flow Chart

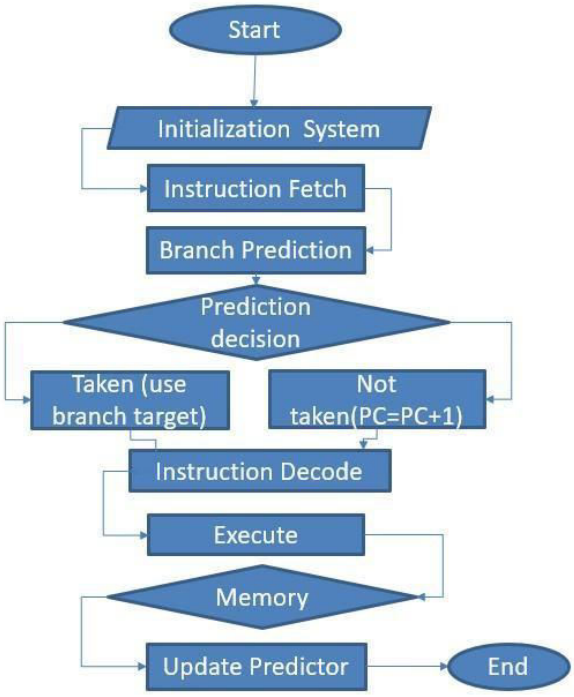


Fig. 2: Flowchart of the Code of the project

Flow Chart Description

1. Start

The operation starts once the processor is powered on or reset. Every pipeline phase, memory block and branch prediction element is initialized to its settings.

2. System Initialization

During this phase essential components of the processor—such, as the Branch History Table (BHT) Branch Target Buffer (BTB) Program Counter (PC) registers and flags—are set up. Predictor inputs have been reset.The computer is configured to the address Pipeline buffers get cleared .This guarantees that the processor begins in an clear state.

3. Instruction Fetch

The processor retrieves the instruction, by utilizing the current PC. At this phase the branch predictor is also consulted to determine if the instruction contains a branch and if a prediction might prevent pipeline delays.

4. Branch Prediction

In this stage the branch predictor evaluates the instruction to determine whether it is a branch. Predicts its potential behaviour. It examines:

Whether the instruction is a branch

The BHT entry for taken/not-taken prediction The BTB for possible target addresses Next the predictor generates:

Prediction direction: Taken or Not Taken

Anticipated destination address: Provided the branch is likely to be taken

BTB hit or miss information

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 12, December 2025 ||

| DOI:10.15662/IJAREEIE.2025.1412015|

5. Prediction Decision

The CPU currently follows the predictor's assumption. If it believes the branch will divert it instantly jumps to that location. Otherwise it proceeds to the instruction.

6. Instruction Decode

The instruction is examined to determine its category and necessary operands. If it is a branch the CPU gets ready to verify the result shortly.

7. Execute

The instruction is processed by the ALU.

For branches this is the point at which the CPU determines whether the branch will be taken and identifies the target.

8. Memory

All memory-related commands are processed at this point. Although branch instructions do not involve memory the CPU monitors them to verify the outcome, against the anticipated one.

9. Update Predictor

The predictor gains knowledge from the outcome. It modifies the BHT and BTB entries to enhance the accuracy of predictions.

10. End

The cycle finishes, and the CPU fetches the next instruction. This loop continues until the program stops.

V. RESULT

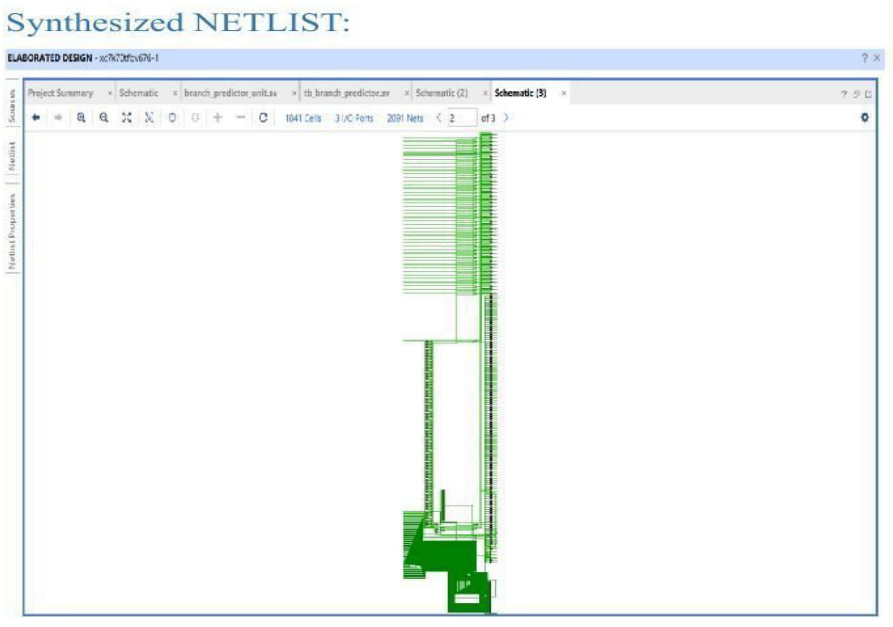


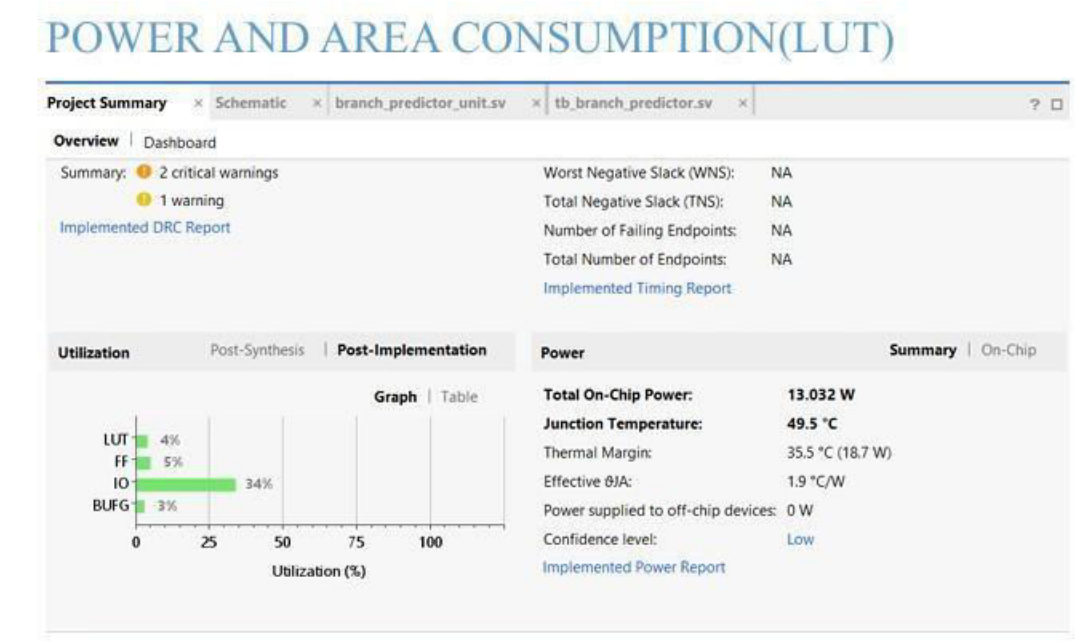
Fig. 3: Synthesized Netlist of Simulation

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 12, December 2025 ||

| DOI:10.15662/IJAREEIE.2025.1412015|

Netlist simulation involves representing the designed circuit as a list of components and their interconnections. This netlist is simulated using appropriate software to verify circuit functionality and analyse performance before hardware implementation.



-

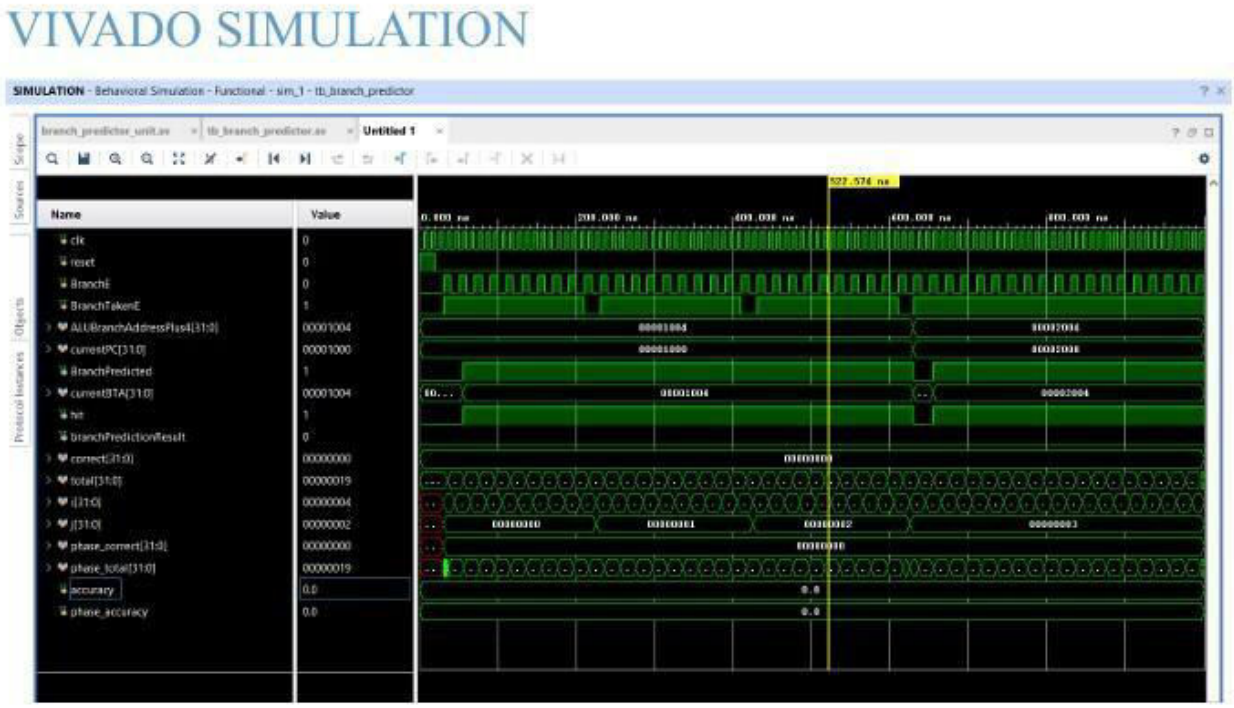


Fig. 4 Power consumption and the output of Vivado Simulation

Power consumption refers to the amount of electrical energy utilized by the system during operation. It is analyzed to ensure efficient performance while minimizing energy usage and heat generation. Optimizing power consumption improves reliability and extends system lifespan.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 12, December 2025 ||

| DOI:10.15662/IJAREEIE.2025.1412015|

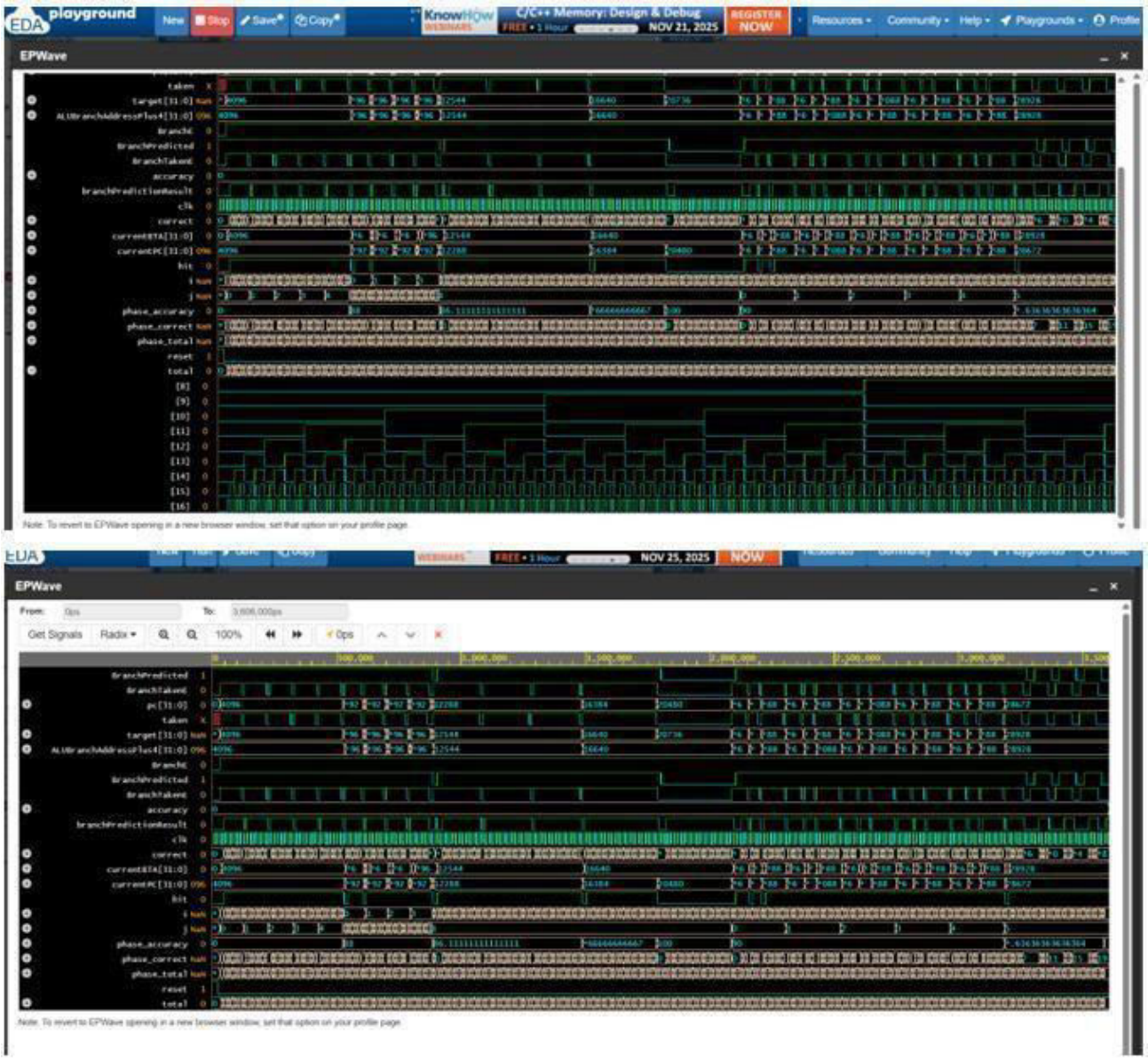


Fig. 5: Actual Output of the project in EDA playground

The actual output in EDA simulation represents the real-time results obtained after executing the design using simulation tools. It is compared with the expected output to verify the correctness of the circuit functionality. Successful matching confirms the accuracy and reliability of the design

VI. CONCLUSION

This Branch Predictor project built on VLSI principles illustrates how careful design can greatly enhance the efficiency of a processor. Utilizing a 2-bit saturating counter BHT combined with a BTB the predictor delivers effective mitigation of control hazards. The complete workflow—from design and RTL implementation in Verilog, to simulation synthesis using Vivado and refinement—demonstrates a methodical engineering strategy. The simulation outcomes demonstrated prediction precision across various branch configurations and synthesis validated that the design meets the necessary power, area and timing constraints. Power reduction strategies like updates and refined indexing contributed to making the design appropriate, for low-power CPUs. The update approach based on feedback guarantees

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 12, December 2025 ||

| DOI:10.15662/IJAREEIE.2025.1412015|

the predictor persistently learns and adjusts to program behaviour. Overall, the project highlights how effective branch prediction can improve CPU flow and reduce stalls, while also setting the stage for future upgrades like global-history or hybrid predictors.

VII. FUTURE WORK

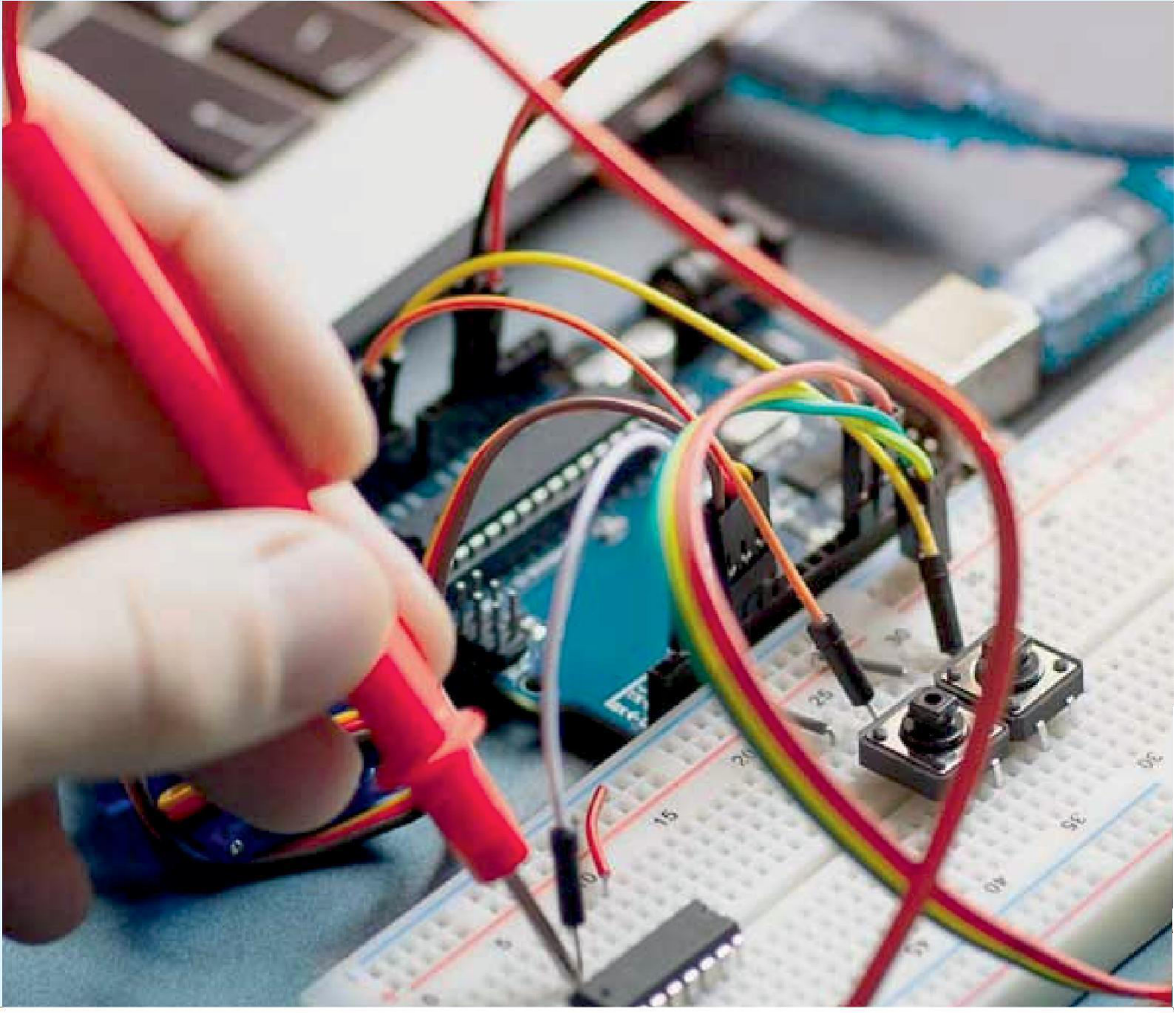
The branch predictor designed in this work establishes a foundation for upcoming improvements. One path forward is incorporating history predictors, which assist in monitoring extended branch dependencies and minimizing mistakes due to related branches. Another beneficial upgrade could be adopting an tournament predictor enabling the system to flexibly select between local and global methods for enhanced precision. Adding support, for Return Address Stacks and indirect branch prediction would further enhance the management of function returns and branches with changing targets. With increasing interest in AI-powered CPU components, integrating perceptron or neural-based prediction models could significantly boost performance for complex workloads. Further improvements in ASIC optimization, power reduction, and latency would help make this design suitable for commercial embedded systems. Overall, the potential for improvement is large, and applying these techniques can lift the predictor to professional and industry-ready standards.

VIII. ACKNOWLEDGMENT

We would like to express our gratitude to the mentors and guide for their unwavering support, constructive criticism, and encouragement throughout this research project. Their advice had a significant impact on the project's course and quality. We are grateful to our institution for providing the equipment, materials, and technical environment required to finish this study.
We would like to thank the co-workers for their support throughout the project by providing guidance, ideas, and conversations.

REFERENCES

1. Yingjie TangZhang, ChenWuyi Fang. “Study of Register Value Branch Predictor Based on CNN” , Microelectronic Research & Development Center, Shanghai University, Published March 2025.
2. Wu Yang, Jie Gao, Qiu Li, Jun Zhang. “Design of RISC-V Out-of-Order Processor Based on Segmented Exclusive Prediction”, Microelectronics Journal, October 2024.
3. Pankaj Nair, V. M. Lalu. “Design and Implement of 5-Stage Pipelined RISC-V Processor on FPGA”, 28th International Symposium on VLSI Design and Test (VDAT), APJ Abdul Kalam Technological University 2024.
4. Luis A.Q. Villon, Zachary Susskind, Alan T.L. Bacellar.” A Conditional Branch Predictor Based on Weightless Neural Networks, Elsevier” Volume 555, Article ID 126637, July 2023.
5. Changbiao Yao, Ziqin Meng, Wen Guo, Jianyang Zhou. “Analysis and Optimization of Branch Prediction Unit of SweRV EH1”, Ningbo Fotile Co., China, December 2022.
6. Harsha Rastogi, Rajat Subhra Chakraborty, Arindam Pal. “Design Space Exploration of Perceptron Based Branch Predictors for Superconducting CPUs”, IEEE ISVLSI Symposium on VLSI, pp. 302–307,2022.
7. Hadeel S.H. Mahmood Middle,” FPGA Configuration of an Alloyed Correlated Branch Predictor Used with RISC Processor”, International Journal of Electrical and Computer Engineering (IJECE), Vol. 11, No. 1, pp. 265–271, 2021.
8. Ali S. Al-Khalid, Safaa S. Omran.” Hybrid Branch Prediction for Pipelined MIPS Processor”, International Journal of Electrical and Computer Engineering (IJECE), Vol. 10, No. 4, pp. 3476–348,2020.
9. Milad Mohammadi, Ehsan Atoofian, Amirali Baniasadi, Tor M. Aamodt. “Energy Efficient On-Demand Dynamic Branch Prediction Models”, IEEE Transactions on Computers, Vol. 69, No. 3, 2020.
10. C. Arul Rathi, Rajkumar G.T., Anil Kumar, Dr. T.S. Arun Samuel.” Design and Development of an Efficient
11. Branch Predictor for an In-Order RISC-V Processor”, Journal of Nano and Electronic Physics, January 2020.



INNO SPACE
SJIF Scientific Journal Impact Factor

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

International Journal of Advanced Research

in Electrical, Electronics and Instrumentation Engineering

9940 572 462 6381 907 438 ijareeie@gmail.com

Scan to save the contact details