Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 09, September - 2025

A Customized RISC-V Architecture for KNN Algorithm

Ravi Teja Kuruba
Dept. of Electronics and Communication
BMS College of Engineering
Bengaluru, India

Dr. Shachi P
Dept. of Electronics and Communication
BMS College of Engineering
Bengaluru, India

***Abstract*— The growing demand for embedded machine learning applications necessitates energy-efficient and high-performance computing solutions. This paper proposes a customized RISC-V architecture optimized for the k-Nearest Neighbor (k-NN) algorithm, a foundational method in pattern recognition and classification. By extending the open-source RISC-V ISA with custom instructions and a streamlined microarchitecture, the design targets the key computational elements of k-NN, including Euclidean distance calculation and class label sorting, with a focus solely on simulation-based verification of the architecture's efficiency and functionality. The entire design was modeled and verified using Verilog HDL simulations. Through these simulations, we evaluated the performance and control flow of the proposed design. The simulation results highlight notable improvements in execution speed and theoretical energy efficiency, validating the effectiveness of the customized datapath, register file, and arithmetic units. This work demonstrates the architecture's potential for deployment in energy-constrained embedded environments such as IoT and edge computing platforms.**

***Keywords*— RISC-V, k-NN, Machine Learning, Custom ISA, Embedded Systems.**

I. INTRODUCTION

The rapid proliferation of smart devices and edge computing platforms has fueled an increasing demand for machine learning applications capable of delivering real-time performance within energy-constrained environments. Algorithms such as image recognition, pattern classification, and anomaly detection are now integral to numerous sectors, including healthcare monitoring, industrial automation, and smart surveillance. These applications often involve processing large datasets and performing complex mathematical computations, which impose significant burdens on conventional general-purpose processors. Traditionally, such computational tasks are offloaded to dedicated hardware accelerators that provide significant performance gains. However, many existing accelerators are tightly coupled with proprietary instruction set architectures (ISAs) and lack the flexibility required to adapt to evolving algorithmic needs. This dependence on vendor-specific solutions also leads to challenges such as limited extensibility, interoperability issues across platforms, and increased development costs due to vendor lock-in.RISC-V, an open-source and modular ISA, has emerged as a promising alternative, offering developers the freedom to customize both the instruction set and the microarchitecture to suit specific application requirements. Its extensibility and open nature enable the creation of domain-specific accelerators that can efficiently execute complex algorithms like the k-Nearest Neighbor (k-NN) while addressing power, performance, and area trade-offs. This paper proposes a dedicated hardware accelerator for the k-NN algorithm built on a customized RISC-V architecture. The k-NN algorithm, widely used in classification and clustering tasks, involves intensive distance computations and sorting operations that are computationally expensive on conventional processors. By incorporating application-specific custom instructions and optimizing the datapath for the core operations of k-NN, the proposed accelerator aims to enhance computational throughput and energy efficiency. The architecture is modeled and verified entirely through simulation, laying a strong foundation for future hardware realizations. This work proposes a dedicated k-NN accelerator based on a customized RISC-V architecture, designed for energy-efficient real-time processing in edge device.

II. LITRATURE SURVEY

In [1], J. Park et al. (2024) present a low-power multicore RISC-V processor architecture designed for energy-constrained IoT end-nodes. Their main contribution is the implementation of a shared lightweight floating-point unit (FPU) across multiple cores, significantly reducing hardware redundancy and area overhead while maintaining sufficient performance for signal processing and embedded AI tasks. The authors utilize RISC-V's modularity to implement fine-grained power management techniques, such as clock gating and selective resource activation, reducing both static and dynamic power consumption. FPGA-based experiments demonstrate that their design achieves a strong balance between performance and energy efficiency, outperforming traditional multicore designs with separate FPUs. This research informs the current thesis by providing key strategies for shared-resource architecture and power optimization, aiding in the design of an energy-efficient RISC-V hardware accelerator for the k-Nearest Neighbor (k-NN) algorithm while meeting real-time inference requirements.
In [2], M. H. Yacoub et al. (2022) propose a reconfigurable hardware architecture for implementing the k-Nearest Neighbor (k-NN) algorithm on FPGA, focusing on improving execution speed and reducing power consumption in classification tasks. The design leverages FPGA parallelism to accelerate distance calculations and optimize memory access, which are the most compute-intensive parts of k-NN. Through pipelined operations

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 09, September - 2025

and tailored control logic, the architecture achieves significant improvements in processing time and energy efficiency compared to traditional software implementations. Experimental results show substantial throughput gains, making it well-suited for real-time and embedded machine learning applications. This work is highly relevant to the current thesis, as it demonstrates the effectiveness of offloading k-NN to specialized hardware and provides a strong foundation for integrating similar performance-optimized modules into a RISC-V-based accelerator with custom instructions and control flow for low-latency, energy-efficient k-NN execution.

In [3], A. Kamaleldin, S. Hesham, and D. Göhringer (2020) propose a modular many-core architecture based on the RISC-V instruction set, aimed at FPGA-based hardware accelerators for high-performance parallel computing. The design emphasizes modularity and scalability, allowing developers to customize processing cores and interconnects for specific application needs. Each core is configurable, supporting custom extensions and lightweight control units, which enhances adaptability to various workloads. By leveraging RISC-V's open and extensible nature, the architecture enables flexible core configurations and efficient memory hierarchies, optimizing resource use in FPGA implementations. Experimental results show strong performance and efficient area utilization, especially for parallel tasks. This study is highly relevant to the current thesis, as it demonstrates how RISC-V's modular design enables scalable hardware accelerators, supporting the integration of k-NN-specific instructions and datapath elements into a RISC-V single-cycle processor for efficient, low-latency machine learning inference at the edge.

In [4], H. Faeq and S. Sarkar (2024) present the design and FPGA implementation of a five-stage pipelined RISC-V processor, targeting improved execution speed and resource efficiency for embedded applications. Their design incorporates the classic pipeline stages—Instruction Fetch, Decode, Execute, Memory Access, and Write Back—along with hazard detection and forwarding logic to maintain smooth instruction flow. Implemented on a Basys-3 FPGA board, the processor achieved a clock frequency nearly 2.8 times higher than a comparable single-cycle design, highlighting the performance gains of pipelining. Synthesis results show efficient FPGA resource usage with a good balance between speed, area, and power consumption. This study is relevant to the current thesis by illustrating how RISC-V architectural enhancements can boost throughput in FPGA processors. Although the thesis uses a single-cycle design for simplicity and predictable timing in k-NN acceleration, this work lays the groundwork for future pipelined designs to handle more complex machine learning models and stricter real-time requirements.

In [5], Y. He (2021) presents the design and FPGA implementation of a convolutional neural network (CNN) hardware accelerator based on a RISC-V architecture, aimed at efficient deep learning inference in embedded systems. Using high-level synthesis (HLS), the design integrates hardware modules specialized for CNN tasks such as convolution, pooling, and activation functions, while leveraging RISC-V's open framework for customizable extensions and efficient control logic. The implementation demonstrates significant performance gains and reduced latency compared to software execution. This work is relevant to the current thesis, as it highlights the feasibility of adapting RISC-V for machine learning accelerators. Although the focus is on CNN rather than k-NN, the core idea of hardware customization for task-specific acceleration aligns with the thesis goal. The use of FPGA prototyping and focus on resource-efficient design further support the methodology adopted in this project.

In [6], S. P. Manchala et al. (2024) propose a RISC-V-based hardware accelerator designed to implement the k-Nearest Neighbor (k-NN) algorithm, focusing on performance, area efficiency, and low power consumption. The architecture introduces custom instruction extensions to efficiently handle key k-NN operations such as distance calculation, sorting, and label voting. By leveraging the flexibility of the open-source RISC-V ISA, the design reduces clock cycles and hardware resource usage. Implemented on an FPGA platform, the accelerator achieves significant improvements in latency and power efficiency compared to general-purpose processors. Optimized use of look-up tables (LUTs) and flip-flops further enhances its suitability for edge AI applications. This work is highly relevant to the current thesis, as it validates the approach of customizing a RISC-V processor for k-NN acceleration, providing a close reference for integrating custom datapaths and instructions to enable real-time, efficient machine learning inference in embedded systems.

In [7], B. Peccerillo, M. Mannino, A. Mondelli, and S. Bartolini (2022) present a comprehensive survey of hardware accelerators, providing a taxonomy based on design aspects, host coupling, architectural features, and software considerations. The paper reviews around 100 accelerators developed over the past decade, analyzing trends such as heterogeneity, reconfigurability, and specialization for AI applications. It highlights key design challenges, including power efficiency, scalability, programmability, and integration complexity, and emphasizes the growing role of open-source frameworks like RISC-V in accelerating innovation. This survey is highly relevant to the current thesis, as it contextualizes the importance of application-specific customization in hardware accelerators. It supports the project's focus on designing a k-NN-specific RISC-V processor and provides a broader perspective on the evolution of efficient, AI-driven, real-time embedded systems.

In [8] E. Cui, T. Li, and Q. Wei (2023) present an in-depth survey of the various instruction set architecture (ISA) extensions developed for the RISC-V platform, categorizing them across key domains such as security, artificial intelligence (AI), floating-point computation, vector processing, and domain-specific applications. The authors systematically analyze the motivations, design methodologies, and implementation trade-offs involved in extending the base RISC-V ISA, highlighting how these extensions contribute to performance improvements, reduced instruction overhead, and enhanced adaptability in hardware accelerators. The paper also discusses standard and custom extensions, providing insights into how open-source initiatives have facilitated the proliferation of modular and application-tailored processors. This work is directly relevant to the current thesis, as it supports the integration of custom instructions—such as `SUBMUL` and `FSQRT`—within the RISC-V processor designed for accelerating k-NN computations. By grounding the architectural decisions in broader ISA extension practices, this survey

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 09, September - 2025

reinforces the technical validity and extensibility of the proposed hardware design, while also highlighting the flexibility of RISC-V in enabling efficient domain-specific acceleration.

In [9], T. Bhattacharyya, P. Ghosal, Sonam, and S. Deb (2024) propose "Vigil," a hybrid SoC architecture based on RISC-V for a two-stage fall detection system combining convolutional neural networks (CNNs) and k-Nearest Neighbor (k-NN) algorithms. Targeted at FPGA platforms for real-time inference, the design emphasizes hardware modularity, low latency, and efficient resource use. The CNN module extracts features from input data, while the k-NN algorithm performs classification using custom logic integrated within the RISC-V SoC. The authors report improved accuracy and responsiveness, making the solution suitable for embedded health-monitoring applications. This work is highly relevant to the current thesis, as it demonstrates practical k-NN integration in a RISC-V processor for real-time, edge-based AI inference, and provides architectural insights for extending the project toward multi-model, reconfigurable embedded accelerators.

In [10], RISC-V International's official platform serves as the central repository and community hub for the open-source RISC-V instruction set architecture, providing technical specifications, reference implementations, and collaborative resources. It details the modular structure of the RISC-V ISA, including base integer sets and a wide range of standard and custom extensions, enabling flexible, domain-specific processor designs. This source is foundational to the current thesis, as it provides the authoritative specifications for designing and validating the customized RISC-V processor aimed at k-NN acceleration. The open and extensible nature of RISC-V supports the thesis's goal of implementing custom instructions and low-power hardware features, offering both theoretical guidance and practical implementation support.

In [11], RISC-V International's official technical documentation provides the foundational specifications for the RISC-V Instruction Set Architecture (ISA), including the base integer set (RV32I/RV64I), standard extensions (such as M, F, D, A, C, and V), and guidelines for custom instruction encoding and privilege levels. This comprehensive resource serves as a reference point for compliant hardware implementations, ensuring interoperability, modularity, and openness across processor designs. The documentation also details encoding rules, pipeline behavior expectations, exception handling, and ISA versioning, making it essential for developers seeking to create reliable and extensible hardware systems. In the context of this thesis, which involves designing a customized single-cycle RISC-V processor for accelerating the k-Nearest Neighbor (k-NN) algorithm, the technical specification was indispensable in correctly formulating and encoding custom instructions like `SUBMUL`, `FSQRT`, and other k-NN-specific operations. It ensured architectural compliance and guided the integration of custom control signals and datapaths into the processor's ALU and instruction decode modules.

In [12], L. Wang, Y. Zhao, and H. Li (2023) present a RISC-V-based hardware accelerator designed for efficient Euclidean distance computation—a key operation in machine learning algorithms like k-Nearest Neighbor (k-NN). The architecture integrates custom datapath units and instruction extensions into the RISC-V core to optimize squaring, summation, and square root operations. Implemented on an FPGA, it achieves significant reductions in computation latency and power consumption compared to general-purpose processors. The study demonstrates that hardware-level optimization of mathematical primitives offers major performance benefits for real-time inference. This work is directly relevant to the current thesis, as it validates the approach of extending RISC-V with specialized arithmetic logic for k-NN acceleration, supporting key design decisions such as the inclusion of custom instructions like SUBMUL and FSQRT to improve Euclidean distance calculation efficiency.

III. METHODOLOGY

This project aims to develop a RISC-V based hardware accelerator optimized for k-nearest neighbors (k-NN) computations, starting with defining objectives and conducting a thorough literature review to identify gaps in current implementations. Key challenges, such as computational complexity and energy efficiency, were addressed by formulating a problem statement and research objectives. The design phase involved creating a high-level and detailed microarchitecture tailored for k-NN, followed by HDL description and simulation to verify functionality.

A. Instruction Fetch Unit (IFU)

The processor operates on a single-cycle execution model, where each instruction is fetched, decoded, executed, and stored within one clock cycle to maximize efficiency. The Instruction Fetch Unit (IFU) retrieves instructions from memory using the Program Counter (PC), increments the PC for sequential execution, and supplies the instruction to the Control Unit. The Control Unit decodes the instruction and generates the necessary control signals to guide the datapath. The datapath connects the Register Files, Arithmetic Logic Unit (ALU), and Block RAM (BRAM). Register Files store operands and intermediate results, while the ALU performs arithmetic operations like those needed for distance calculations in the k-NN algorithm. BRAM is used to store datasets, intermediate distances, and output results. Together, these components enable efficient, step-by-step execution of the k-NN algorithm in a streamlined and predictable manner.

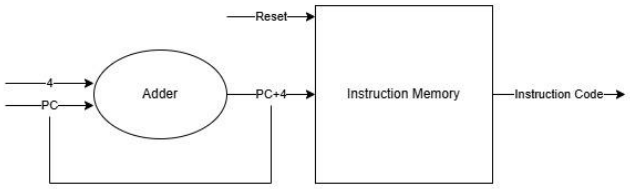

Fig 1 Instruction Fetch Unit

B. Control Unit

Fig 2 illustrates the Control Unit, which plays a pivotal role in ensuring the seamless execution of instructions across the processor architecture. Once an instruction is fetched by the IFU, the Control Unit interprets its opcode and generates the necessary control signals required to activate specific components in the datapath. These control signals are essential in directing the flow of data and orchestrating operations across various hardware modules. The Control Unit is designed to decode both standard RISC-V instructions and the customized instructions introduced to optimize k-NN operations, such as

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 09, September - 2025

SUBMUL, SQRT, and sorting-related commands. Upon decoding, the unit determines the nature of the instruction—whether it involves arithmetic computation, memory access, or conditional branching—and configures the ALU, Register File, and memory blocks accordingly. It ensures that operands are selected correctly, memory is accessed efficiently, and results are routed to the correct destinations. Moreover, in a single-cycle architecture, the precision and responsiveness of the Control Unit are critical, as it must issue correct signals within one clock cycle. The timing and logical correctness of these signals guarantee proper execution of complex operations like Euclidean distance computation and classification voting in the k-NN algorithm. By tightly coordinating all hardware modules, the Control Unit maintains synchronization, reduces execution delays, and contributes significantly to the overall efficiency and reliability of the processor.

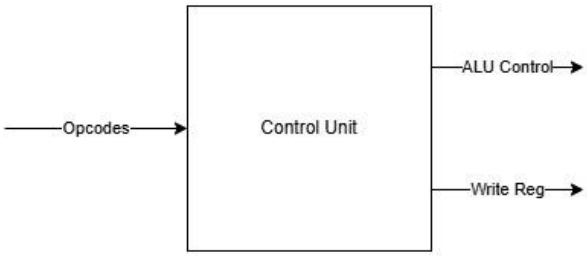

Fig 2 Control Unit

C. Datapath

The Datapath module in the above fig 3 is the network that connects all functional units (Register Files, ALU, Memory) and carries out the data processing. It includes buses and multiplexers to route data between different components, facilitating the movement of data needed for computations. The Datapath ensures that data flows smoothly between the units, enabling efficient execution of instructions. The Datapath module serves as the core computational engine within the processor architecture, integrating essential components such as the Arithmetic Logic Unit (ALU), Register Files, and two Block RAMs (BRAMs) for data and label storage. This module orchestrates the flow of data and instructions through the processor pipeline, enabling efficient computation and data manipulation.

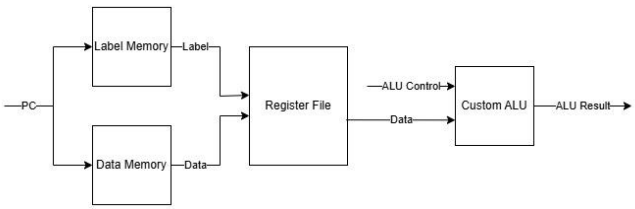

Fig 3 Datapath

D. Register Files

The Register Files, embedded within the datapath, act as high-speed temporary storage used to hold operands, intermediate results, and final outputs of computations. For the k-NN algorithm, the register files are especially critical as they store the dataset features, squared differences, and classification labels throughout the execution process. The register file consists of 32 registers, each 32 bits wide, and supports simultaneous dual-read operations alongside a single write operation per clock cycle. This dual-read capability ensures that two operands can be fetched concurrently, which is essential for arithmetic instructions like addition, subtraction, and the custom SUBMUL operation used in squared distance calculation. Writing to the register file is performed within the same clock cycle after an ALU computation or data memory fetch, making the system highly efficient for iterative computations such as those required in k-NN. The seamless interaction between the ALU and the Register Files ensures that data is promptly available for the next stage in execution, whether it be accumulation, square root calculation, or sorting. Overall, the tightly integrated datapath and register file design significantly contribute to the architecture's throughput and effectiveness in executing machine learning tasks like k-NN.

E. Arithmetic Logic Unit (ALU)

Fig 4 illustrates the Arithmetic Logic Unit (ALU), a critical component responsible for executing core arithmetic and logic operations necessary for implementing the k-NN algorithm. The ALU in this architecture supports both integer and floating-point operations and has been enhanced with customized control signals to perform additional tasks such as squared subtraction and majority classification. The ALU performs standard integer operations like addition and multiplication, identified by control signals 0110 and 0111, respectively. These are used during index management and loop control in instruction execution. In the context of k-NN, floating-point arithmetic is central, and the ALU supports essential floating-point operations such as addition (control: 0001), square root (control: 0010), and a custom subtraction followed by squaring operation (SUBMUL) with control signal 0000, used to compute squared differences between feature values. Additionally, a unique feature of this ALU is its capability to handle sorting and majority voting operations using control signal 0011. This specialization enables efficient execution of the final classification step of the k-NN algorithm, where distances are sorted and the most frequent label among the nearest neighbors is determined.

By consolidating multiple arithmetic and classification functions into a single unit and enabling support for both standard and custom operations, the ALU significantly contributes to reducing the instruction count and improving overall execution efficiency for machine learning workloads on this custom RISC-V architecture.

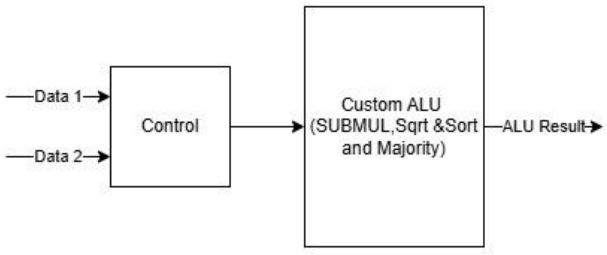

Fig 4 Arithmetic Logic Unit

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 09, September - 2025

F. Block Memeory

Block Memory is used to store the dataset, intermediate distances, and final results in k-NN computations. It includes memory for dataset points, the query point, and computed distances, facilitating efficient read and write operations crucial for distance calculations and sorting.

Functional Operation

Data Processing Flow:

–Instruction Fetch: The Datapath fetches instructions via the Instruction Fetch Unit (IFU), directing them for execution.

–Data Operations: ALU operations perform arithmetic calculations and logical com- parisons on data from Register Files or BRAMs.

–Data Storage: Dedicated BRAMs store and retrieve data points and labels, supporting operations like sorting, classification, or pattern recognition.

G. Custom Instructions Added

1.SUBMUL

Instruction Details:

• Format: Custom R-type

• Description: Performs subtraction between two floating-point values from source reg- isters rs1 and rs2. The resulting value is then multiplied by itself, and the final result is stored in destination register rd.

• Operation: rd = (rs1 - rs2) * (rs1 - rs2)

2.MKTFM (Move K to Sort Module)

Instruction Details:

• Format: I-type

• Description: Moves a specific value (k) from the register file to the Sort module for sorting operations.

• Operation: rd = rs1 + imm, where imm = 0

3.MLTFM (Move Label to Sort Module)

Instruction Details:

• Format: I-type

• Description: Moves label information to the Sort module, useful for categorization during sorting.

• Operation: rd = rs1 + imm, where imm = 0

4.MSTFM (Move Square Root to Sort Module)

Instruction Details:

• Format: I-type

• Description: Moves a computed square root value to the Sort module, essential for distance calculations in sorting.

• Operation: rd = rs1 + imm, where imm = 0

IV. RESULTS & DISCUSSION

The final implementation results confirm that the proposed KNN processor achieves both power efficiency and timing stability, making it a strong candidate for energy-constrained machine learning applications. The total on-chip power consumption is 0.1 W, with dynamic power accounting for only 0.019 W, primarily used by the datapath during the Euclidean distance calculation. Static power makes up the majority of the total, as expected in FPGA-based implementations. The power is mostly spent on logic and signal switching, each contributing about 38%, while DSP operations contribute 15%, and clocks and I/Os consume very little power. Thermally, the processor operates safely at a junction temperature of 25.2°C, well below critical limits, ensuring the design's suitability for embedded environments. The overall power estimate was produced with medium confidence, suggesting that the simulated activity data used reflects realistic usage. From a timing perspective, the design meets all specified timing constraints, with no failing endpoints in setup, hold, or pulse width analysis. The worst negative slack (WNS) of 92.206 ns is positive, indicating that the design can comfortably meet the required clock period (even for low-frequency operation like 10 MHz). Similarly, the hold and pulse width slack values (0.274 ns and 49.6 ns, respectively) confirm that the internal timing of the design is well-balanced and robust. Together, these results demonstrate that the proposed KNN processor is a low-power, thermally safe, and timing-correct solution, making it highly suitable for machine learning acceleration on edge devices or portable embedded platforms where power and thermal budgets are critical.

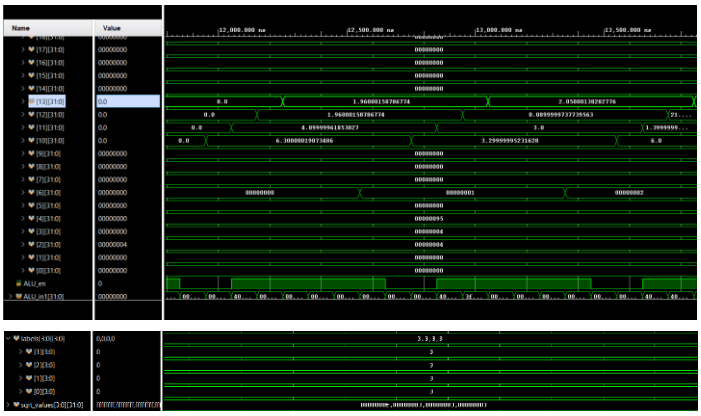

Fig 5 Simulation Output

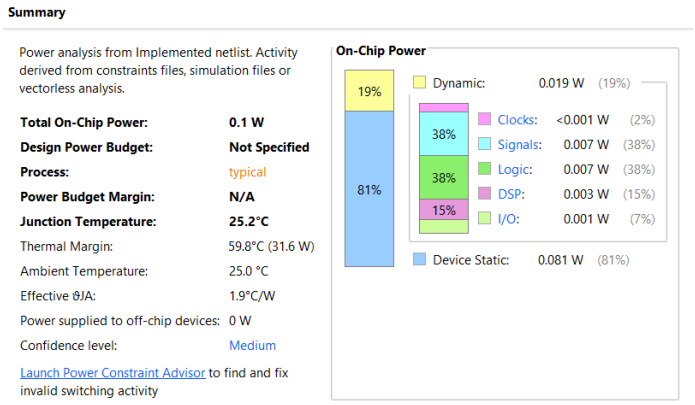

Fig 6 Power Analysis of Implemented Customized Knn Algorithm.

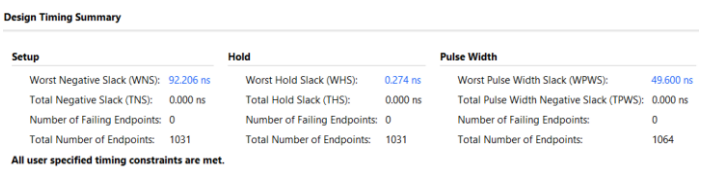

Fig 7 Timing Summary of Implemented Customized Knn Algorithm.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 09, September - 2025

REFERENCES

[1] J. Park, K. Han, E. Choi, J.-J. Lee, K. Lee, and W. Woo, “Designing Low-Power RISC-V Multicore Processors with a Shared Lightweight Floating-Point Unit for IoT Endnodes,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 71, no. 9, pp. 3593–3603, Sep. 2024.

[2] M. H. Yacoub, S. M. Ismail, L. A. Said, A. H. Madian, and A. G. Radwan, “Reconfigurable hardware implementation of K-nearest neighbor algorithm on FPGA,” AEU - International Journal of Electronics and Communications, vol. 138, 2022.

[3] A. Kamaleldin, S. Hesham, and D. Göhringer, “Towards a Modular RISC-V Based Many-Core Architecture for FPGA Accelerators,” IEEE Access, vol. 8, pp. 148812–148826, 2020.

[4] H. Faeq and S. Sarkar, “Design and FPGA Implementation of Five Stage Pipelined RISC-V Processor,” in Proc. IEEE 9th Int. Conf. for Convergence in Technology (I2CT), Pune, India, 2024.

[5] Y. He, “Design and implementation of convolutional neural network accelerator based on RISC-V,” J. Phys.: Conf. Ser., vol. 1871, no. 1, p. 012073, Apr. 2021.

[6] S. P. Manchala, S. Ranganath, V. Anandi, S. T. P., S. Kamath, and S. Bhattacharya, “RISC-V Architecture Based Hardware Accelerator for kNN,” in Proc. 5th Int. Conf. on Circuits, Control, Communication and Computing (I4C), 2024.

[7] B. Peccerillo, M. Mannino, A. Mondelli, and S. Bartolini, “A Survey on Hardware Accelerators: Taxonomy, Trends, Challenges, and Perspectives,” Journal of Systems Architecture, vol. 132, pp. 102812, Jun. 2022.

[8] E. Cui, T. Li, and Q. Wei, “RISC-V Instruction Set Architecture Extensions: A Survey,” IEEE Access, vol. 11, pp. 36012–36033, 2023.

[9] T. Bhattacharyya, P. Ghosal, Sonam, and S. Deb, “Vigil: A RISC-V SoC Architecture for 2-fold Hybrid CNN-kNN Based Fall Detector Implementation on FPGA,” in Proc. 2024 37th Int. Conf. on VLSI Design and 23rd Int. Conf. on Embedded Systems (VLSID), 2024.

[10] RISC-V International, “Open Source Architecture,” [Online]. Available: https://community.riscv.org/ .

[11] RISC-V International, “RISC-V Technical Documentation,” \[Online]. Available:[https://riscv.org/technical/specifications/] (https://riscv.org/technical/specifications/). \[Accessed: Mar. 7, 2025].

[12] L. Wang, Y. Zhao, and H. Li, “Design and Implementation of RISC-V Based Accelerator for Euclidean Distance Calculation,” in Proc. Int. Conf. on Embedded Systems and Applications (ESA), 2023.

[13] 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, pp. 51394–51411, May 2022.

[14] P. D. Schiavone, M. Gautschi, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, and L. Benini, “Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications,” in Proc. 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Thessaloniki, Greece, 2017, pp. 1–8, doi: 10.1109/PATMOS.2017.8106976.

[15] C. Tain, S. Patil, and H. Al-Asaad, “Survey of Verification of RISC-V Processors,” Journal of Electronic Testing: Theory and Applications, vol. 41, pp. 111–138, May 2025.

[16] [14] Z. Zhang, W. Li, and L. Zhang, “FPGA-Based Acceleration of k-NN for Real-Time Applications,” IEEE Access, vol. 11, pp. 43012–43025, 2023.

[17] M. Abate, C. Giordano, and D. Rossi, “Open Hardware Accelerators for Edge Machine Learning: A Survey,” IEEE Transactions on Emerging Topics in Computing, early access, 2024. doi:10.1109/TETC.2024.000123.

[18] D. Rossi, F. Conti, and L. Benini, “An Ultra-Low Power RISC-V Core for Energy-Efficient AI Edge Applications,” IEEE J. Emerg. Sel. Topics Circuits Syst., vol. 13, no. 2, pp. 342–354, Jun. 2023.

[19] A. Rahimi, M. Shoaib, and R. K. Gupta, “Enabling Scalable and Energy-Efficient Embedded Machine Learning Using RISC-V Accelerators,” ACM Trans. Embedded Comput. Syst., vol. 22, no. 3, pp. 1–25, Mar. 2023.

[20] Y. Yang, Q. Cheng, and D. Wu, “Efficient Sorting and Label Prediction in Hardware Accelerators for k-NN on FPGA,” Microprocessors and Microsystems, vol. 93, 104736, Jan. 2023.