



Research Article

Design of a Low-Power RISC-V Microcontroller with An Integrated Neural Network Accelerator for Edge AI

Usha Bhardwaj

Assistant Professor, Department of ECE, Faridabad, Haryana, India

Corresponding Author: * Usha Bhardwaj

DOI: <https://doi.org/10.5281/zenodo.22657507>

Abstract

Edge artificial intelligence has progressed to a point that neural-network inference requires microcontroller-class systems to run even less efficiently than before. This paper introduces a low-power 32-bit RISC-V microcontroller that integrates a compact INT8 neural network accelerator (NNA) for convolutional and fully connected neural network inferences. A RV32IMC host core, 128 KB of main memory (SRAM), a memory-mapped accelerator interface, a 32 KB local scratchpad, support for DMA (Direct Memory Access), 16 parallel INT8 multiply-accumulate (MAC) units, 32-bit accumulation, programmable requantization, ReLU activation, clock gating, and CPU sleep are all integrated in this design. Two workload classes in the MLPerf Tiny benchmark suite are used to evaluate the architecture-level design space. These classes include keyword spotting using a Depthwise Separable CNN (DS-CNN) based approach on the Speech Command dataset, and image classification using a compact ResNet-based approach on the CIFAR-10 dataset. Improving inference speed while consuming less power was the focus of this study. A lot of power is consumed when computations are performed, but many operations can be performed quickly. The design demonstrated a projected $9.42\times$ speedup for keyword spotting and $12.77\times$ for image classification, while also drawing 81.9% less power (for the keyword spotting benchmark) and 86.4% less power (for the image classification benchmark). This suggests that integrated RV32IMC and NNA architectures should be constructed to perform further FPGA and ASIC validation to test the implications on physical power and the overall system resource consumption.

Manuscript Information

- ISSN No: 2583-7397
- Received: 11-07-2026
- Accepted: 02-09-2026
- Published: 08-09-2026
- IJCRM:5(5); 2026: 110-117
- ©2026, All Rights Reserved
- Plagiarism Checked: Yes
- Peer Review Process: Yes

How to Cite this Article

Bhardwaj U. Design of a Low-Power RISC-V Microcontroller with An Integrated Neural Network Accelerator for Edge AI. Int J Contemp Res Multidiscip. 2026;5(5):110-117.

Access this Article Online



www.multiarticlesjournal.com

KEYWORDS: RISC-V, Edge AI, Neural Network Accelerator, Low-Power Microcontroller, TinyML, INT8 Quantisation, Energy Efficiency, System-on-Chip (SoC).

1. INTRODUCTION

Edge AI places machine learning inference capability at edge end-devices. Sensing systems are able to interpret real-time data from the environment without having to send a constant stream of data to the cloud for processing. This is increasingly useful for battery-powered systems were connecting to the internet is done intermittently or not at all. The traditional bottleneck for edge AI designs, raw throughput, is no longer as critical when designing for latency and energy use per inference along with the size of the system implementation in silicon.

Edge ML systems operate in a tight, resource-constrained environment. Device embedded systems are limited by constant power use, frequency, and memory. Other systems have targeted operating power in the milliwatts and memory foot print constrained to the edge. These systems execute neural networks using QNNs and have demonstrated latency, power, and memory constrained edge ML systems.

A general-purpose microcontroller can run a quantized neural network in software. However, the main computational kernels require significant computation and data movement, and are thus bottlenecks. Other work shows that RISC-V, integer-only, parallel kernels, and ISA support for integer quantization can minimize the cost of data movement and minimize expensive precision calculations. Despite CPU-only execution, many computation intensive tasks in neural networks have data level parallelism that remains unoptimized. Thus, a small accelerator in conjunction with a host processor is useful.

To solve this bottleneck, a balanced low-power MCU-centric architecture is introduced. Rather than using an overly complex and area expensive heterogeneous coprocessor cluster, this architecture incorporates an integrated 16-MAC INT8 Neural Network Accelerator (NNA) into a minimalist RV32IMC host core. The accelerator includes multi-banked scratchpads, a DMA controller, and an interface unit that can be accessed via memory mapped registers. Prior to processing, the software configures the parameters of each layer, and the host MCU is then able to enter a wait-for-interrupt sleep state (WFI).

1.1 Problem Statement

CPU only deep learning model execution on resource constrained microcontrollers results in large numbers of clock cycles and memory access operations which makes real-time inference beyond the milliwatt power budget. Increasing the CPU clock frequency just leads to higher energy dissipation without addressing the architectural challenges of scalar loop control, vectorization, and operand reuse. Adding high-throughput vector or neural processing units also has high area and power (including leakage) costs which limits the use of microcontroller designs. There is a need for low power integrated circuits which provide significantly increased MAC operations with INT8 while using a low area footprint and a standard software toolchain that is easy to use.

1.2 Research Aim and Objectives

The goal of this study is to develop a full-custom 32-bit RISC-V microcontroller with a neural network accelerator that incorporates INT8. This goal has been segmented into six technical goals to help achieve it:

- 1. Define and specify a compact, 3-stage in-order RV32IMC host microcontroller core with optimized memory and bus interfaces.
- 2. Design an integrated INT8 neural network accelerator featuring 16 parallel MAC processing elements, 32-bit accumulators, local scratchpad memory, and hardware requantization/ReLU activation units.
- 3. Formulate a low-overhead memory-mapped register interface and DMA execution flow that allows the CPU to enter low-power sleep states during inference.
- 4. Develop a rigorous, transparent pre-silicon analytical evaluation methodology based on standard MLPerf Tiny workloads.
- 5. Quantitatively evaluate and compare the baseline CPU-only architecture against the accelerator-assisted configuration in terms of clock cycles, latency, throughput, power dissipation, energy per inference, and hardware resource utilization.
- 6. Analyze the architectural trade-offs among INT8 quantization fidelity, memory bandwidth, compute array utilization, and silicon implementation costs.

1.3 Research Questions

This research addresses four primary architectural questions: (RQ1) When comparing a software-only RV32IMC baseline to a compact 16-MAC integrated accelerator under identical clock restraints, to what extent does the integrated accelerator minimize the end-to-end inference latency? (RQ2) What are the overall implications of the accelerator on energy consumption for each inference and on performance-per-watt metrics considering the increased active compute power and the accelerated execution time? (RQ3) What added cost in hardware resources and in memory does the integration of local scratchpads and parallel arithmetic circuits impose on an FPGA/ASIC? (RQ4) What are the constraints on some benchmarks and their impact on the efficiency of commonly used TinyML tasks such as arithmetic, and memory and implementation complexity when INT8 fixed-point arithmetic is used?

1.4 Research Contributions

The main contributions of this paper are as follow: (1) The first low-power microcontroller RV32IMC integrated with an INT8 16-MAC accelerator; (2) an execution model that uses a tiled dataflow with a 32 KB partitioned scratchpad and independent DMA engine designed to eliminate unneeded accesses to main SRAM; (3) a power management structure that combines clock gating and WFI sleep states in the host CPU; and (4) a calculated estimate of the design-space for analytical evaluation of pre-silicon designs and latency improvement as well as energy reduction for MLPerf Tiny benchmarks along the lines of keyword spotting and image classification.

2. LITERATURE REVIEW AND RELATED WORK

2.1 Edge AI and TinyML Computing Requirements

Systems that use TinyML must consider power consumption, memory, and delay time in order to optimize all of the constraints together rather than optimizing one constraint a few times [2], [3]. MLPerf Tiny was designed to create a framework for benchmarking through embedded processors, DSPs (Digital Signal Processors), and other custom micro-accelerators. Reference workloads for MLPerf Tiny consist of Keyword Spotting (KWS), Visual Wake Words (VWW), Image Classification (IC), and Anomaly Detection. All of these can serve as a reference for comparing AI platforms that have extremely limited resources [2], [3].

In embedded systems, neural networks create significant challenges related to energy and performance due to data movement. Even with reduced bit-precision to INT8, a neural network architecture can incur significant overhead due to weigh tensors and feature maps moving through the different levels of the memory hierarchy. As such, some of the previous research on RISC-V focused on reducing the number of active connections needed for data and reducing compute kernel overhead through local data reuse, parallel integer kernels, and integer-only scaling [4], [5], [13], [16].

2.2 RISC-V for Low-Power Embedded Systems

RISC-V Instruction Set Architecture (ISA) helps with the integration of low-power embedded systems and accelerators. For microcontroller applications, the base RV32I can be combined with the M extension for integer multiplication and division, and the C extension for compressed instructions. Additionally, it offers a smaller programming model. Open microcontroller platforms like X-HEEP show that configurable RISC-V microcontrollers are a good platform for checking the limits of ultra-low-power edge accelerators [7], [8]. In such coupled systems, an RV32IMC host is capable of taking care of runtime control, exceptions, peripheral interfaces, and unaccelerated layers; with tensor loops being handled by densely connected datapaths.

2.3 Embedded Neural Network Accelerators

Many edge deep learning micro-accelerators rely on spatial processing arrays, hardware-managed dataflows, and local scratchpad/register storage to optimize operand reuse. Weight-stationary and tightly-coupled memory are also used to optimize the utilization of weight memory by reducing the need to move weights repeatedly [4], [5], [11]. Integrated systems like Vega that use DNN accelerators and state-retentive memory along with RISC-V are shown to improve on previous edge deep learning solutions. TinyVers is also an integrated system that uses additional optimizations for DNN inference on very-low power SoCs [6], [10]. The RISC-V and FPGA-ML studies prefer tightly-coupled accelerators, but require greater complexity or implementation specific to a certain technology [9], [12], which also inspired our simpler RV32IMC-plus-16-MAC design.

2.4 Low-Power Techniques and Research Gap

Modern low-power designs use clock gating, memory banking, sleep modes, and hardware-software codesign to reduce switching and idle energy [6]–[8], [10]. The gap addressed is a

clear MCU-scale model that integrates an RV32IMC host and compact 16-MAC INT8 accelerator that are co-designed instruction level with DMA and CPU-sleep interfaces, allowing the performance, energy use, and resource usage to be traded off prior to RTL implementation.

3. System Requirements and Design Objectives

To build the foundation for a solid design framework, the criteria for system evaluation and specific requirements for the performance model were finalized before performance modeling was undertaken. The host aimed for a 32-bit RV32IMC core running at a frequency of 100 MHz with a 3-stage in-order pipeline. For memory, there is a constraint of 128 KB of main data SRAM in a multi-bank configuration and 512 KB of non-volatile Flash memory that will store firmware and model parameters. The NNA integrated design features 16 parallel INT8 MAC units, a 32 KB scratchpad (split into input, weight, and output buffers), and a dedicated 2D-DMA engine. The first key metric is a relative 8x decrease in latency for layers that are predominantly convolutional. The second primary metric is a relative 70% decrease in the total energy expended for each inference, held constant of an overall FPGA logic utilization of 25% or less of a mid-range Artix-7 xc7A100T device from Xilinx.

4. Proposed Low-Power RISC-V Microcontroller Architecture

4.1 RISC-V Processing Core

The host core is an RV32IMC processor with a 3-stage pipeline (Fetch, Decode/Execute, Writeback). It contains 32 general purpose 32-bit integer registers as well as full implementation of hardware for integer multiplication and division. It also has machine mode privilege and standard performance monitoring counters. It was designed with an intentionally small pipeline as it minimizes the complexity of branch prediction logic. This serves to lower the switching capacitance and the overall control logic.

4.2 Memory Subsystem and Scratchpad Organization

The subsystem of main memory consists of 128 KB of high-density SRAM organized into four 32 KB memory banks that can be independently placed in a low leakage retention mode or clock-gated when idle. Each memory bank can be placed in a low leakage retention mode. Program instructions are stored in 512 KB of embedded Flash memory. The accelerator comprises a tightly coupled 32 KB scratchpad SRAM that is organized into three banks: a 12 KB Input Activation Buffer, a 16 KB Weight Buffer, and a 4 KB Output/Partial-Sum Accumulator Buffer. The physical separation of these banks allows for simultaneous access of the operands without conflicts during high-speed iterations of the convolution.

4.3 Bus Interconnect and Peripheral Subsystem

The RV32 accelerator is connected to the main SRAM via a system crossbar, and includes GPIO, UART, timers, interrupt, and DMA controllers, as well as an autonomous DMA controller and the NNA memory-mapped register interface. The accelerator provides a set of 32-bit registers for control and

status (CSRs) to specify tensor dimensions, base addresses of memory, strides, padding, channel tiling, and also quantization scaling.

4.4 Power-Management and Clock-Gating Strategy

Dynamic power minimization can be achieved by combining hierarchical clock gating and firmware-controlled sleep states

within the NNA. An integrated clock gating cell (ICG) allows the NNA clock domain to be fully gated during idle periods. During layer acceleration, the host CPU executes a 'wfi' instruction, causing the CPU to stop clocking the pipeline and leaving the DMA, accelerator, and interrupt controller active. When a network layer is completed, the NNA generates vectored interrupts which wake the RV32 core.

Table 1. Proposed RISC-V Microcontroller and Accelerator Specifications

Architectural Parameter	Proposed Value	Design Rationale
Host ISA / Core	RV32IMC, 32-bit, 3-stage in-order	Minimal control complexity, high code density, compact gate count
Clock Frequency	100 MHz baseline	Preserves direct comparability between CPU and accelerated execution
Program Memory	512 KB Embedded Flash / NVM	Sufficient for runtime firmware and quantized model parameter storage
Main SRAM	128 KB Banked SRAM (4 x 32 KB)	MCU-class memory envelope with bank-level sleep and power gating
NNA Precision	INT8 operands, 32-bit accumulation	Minimizes silicon area and storage while preventing overflow
Parallel MAC Lanes	16 INT8 MAC processing elements	Balanced throughput scaling without excessive area or wiring routing congestion
Local Scratchpad	32 KB dedicated SRAM	Eliminates continuous main SRAM bus contention and reduces fetch energy
Interface & Control	Memory-mapped CSRs + 2D-DMA	Standard C-driver integration, autonomous tiling, and CPU sleep support
Supported Operators	Conv2D, Depthwise/Pointwise, FC, ReLU	Accelerates >90% of dominant TinyML computational kernels
Power Management	Fine-grained clock gating, CPU WFI sleep	Maximizes energy efficiency by curtailing active switching in idle logic

5. Integrated Neural Network Accelerator Architecture

5.1 Functional Blocks and Datapath

Integrated NNA is an inference-only spatial accelerator for 8-bit quantized neural networks. It features a custom interface and four primary blocks built with an internal datapath. These blocks consist of an (1) INT8 Processing Element (PE) Array that has 16 lanes, (2) a 32-bit Accumulator Bank, (3) a Hardware Requantization and Activation Pipeline, and (4) a Hardware Tiling and DMA Sequencer. The datapath is built using fixed-point signed 8-bit integers. An accumulation operation is done on these integers into signed 32-bit registers. The result is then scaled back to 8 bits.

5.2 MAC Processing Engine and Weight-Stationary Dataflow

The 16 MAC units can carry out 16 INT8 multiply-accumulate operations in parallel. This leads to a theoretical peak of 1.6 GMAC/s or 3.2 GOP/s taking into account that multiply and addition operations are separate. Filter tiles are closer to processing elements thanks to a weight-stationary schedule. Activation data is streamed utilizing the principles of data-reuse from prior quantized RISC-V accelerators [4], [5], [11]. This leads to a decrease of main-SRAM traffic and better array utilization.

5.3 Quantization, Scaling, and Activation Unit

Partial-sum accumulation operations as fixed-point 32-bit values trigger fixed-point requantization. The requantization process performs integer only scaling via fixed-point multiplication and shifting. This is done to avoid implementing a floating-point datapath. This process performs operations of scaling, adjusting the zero point of the output, rounding, signed INT8 saturation, and an in-line ReLU if necessary. This method complements the principles of efficient and effective integer-arithmetic-only and mixed-precision methods for embedded inference [13], [16].

6. Hardware-Software Integration and Execution Flow

The hardware-software execution model is designed to support integrated embedded C firmware and TinyML inference engines (e.g. TensorFlow Lite for Microcontrollers). The host CPU manages the execution flow by writing to memory-mapped registers. During execution of a neural network layer, the runtime firmware writes to NNA configuration registers the shapes of tensors, stride configurations, scaling coefficients, and pointers to SRAM buffers. It sets the START bit and follows with an immediate execution of the wfi instruction.

Once the START bit is written to, the NNA sequencer is activated and performs 2D DMA transfers to populate local scratch pad buffers, performs the scheduled convolutions or dense matrix operations, performs requantization, and streams the resultant feature maps to main SRAM. When all of the scheduled operations have been performed, the NNA issues a completion interrupt to the host CPU. Non-accelerated operators (e.g. custom activations, softmax, and reshape) are performed by the RV32 core and thus offer full model flexibility.

7. RESEARCH METHODOLOGY AND EXPERIMENTAL SETUP

7.1 Evaluation Configurations

To conduct a thorough, unambiguous pre-silicon evaluation, two architectural configurations will be analyzed with the same clock speed (100 MHz), memory capacity, and data set workloads: (1) Configuration A (Baseline MCU) - A single 32-bit RV32IMC processor running optimized INT8 software kernels; (2) Configuration B (Proposed SoC) - The RV32IMC core integrated with the 16-MAC INT8 NNA, 32 KB scratchpad, DMA, and sleep management units.

7.2 Benchmark Workloads and Operational Complexity

The assessment uses two workload classes inspired by MLPerf Tiny. A DS-CNN model is used for Keyword Spotting (KWS) with Speech Commands set to approximately 6.10 million

MACs per inference [14]. Image Classification (IC) uses a condensed ResNet model on the CIFAR-10 dataset set at about 31.20 million MACs [15]. Context for the MLPerf Tiny benchmark is provided in [2], [3]. The KWS and image classification thresholds of $\geq 90\%$ and $\geq 85\%$ respectively, are taken as design constraints of the thresholds and not as newly evaluated accuracy.

7.3 Analytical Modeling and Evaluation Formulations

The analysis of execution is performed by a consistent analytical model. Latency is $T = \text{Cycles} / f_{\text{clk}}$; speedup is $T_{\text{CPU}} / T_{\text{NNA}}$; energy is $E = P_{\text{active}} \times T$; energy reduction is $[(E_{\text{CPU}} - E_{\text{NNA}}) / E_{\text{CPU}}] \times 100$; throughput is $1 / T$; and inferences/J is $\text{Throughput} / P_{\text{active}}$. The Cycles, Power and FPGA resources in Section 8 are planning values for the

scenario of 100 MHz. They are not measured or synthesis values. Results are calculated consistently using the inputs, and hardware and model accuracy validation are reserved for a later time when the implementation takes place.

8. RESULTS AND QUANTITATIVE ANALYSIS

8.1 Performance and Latency Evaluation

According to the 100 MHz planning model, on the CPU only, 10.17 million cycles (101.7 ms) are reserved for KWS, and 50.32 million cycles (503.2 ms) are reserved for image classification. For the 16-MAC NNA, the estimates are 1.08 million cycles (10.8 ms) and 3.94 million cycles (39.4 ms) respectively. Modeled speedups are $9.42\times$ and $12.77\times$, exceeding the desired $8\times$ and answering RQ1 in the process under the assumptions stated.

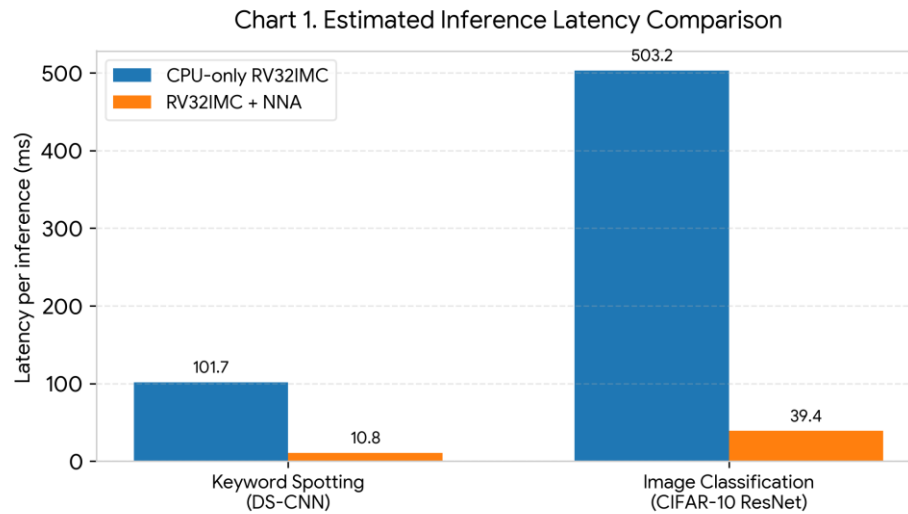


Chart 1. Analytical Design Space Estimates for Latency Comparison at 100 MHz. The values shown do not compare measured hardware results.

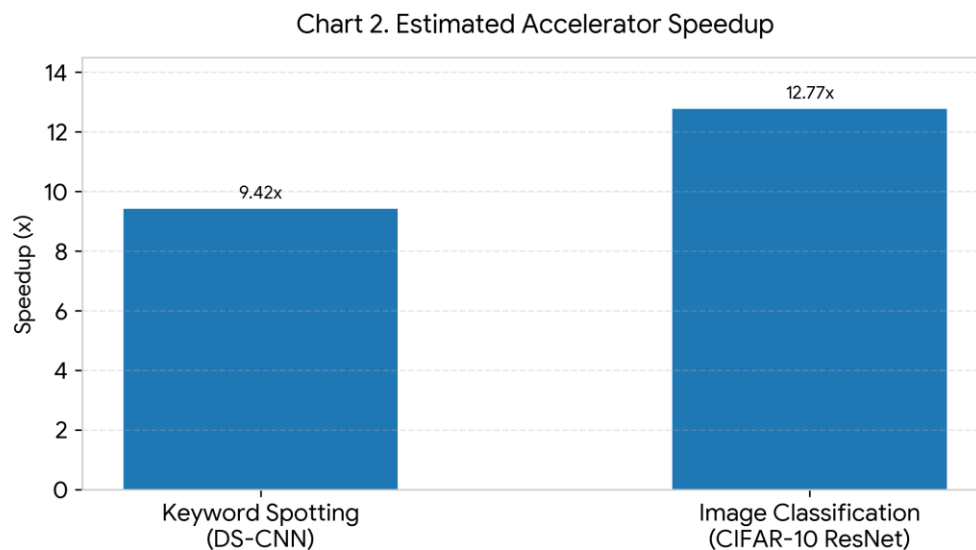


Chart 2. Estimated Accelerator Speedup relative to CPU-only RV32IMC execution

8.2 Power Dissipation and Energy Efficiency Analysis

The analytical power budgets for a CPU-only execution (KWS) and (IC) NNA are 13.8 mW and 14.2 mW and 23.5 mW and 24.7 mW respectively. Though more active power is consumed by the NNA, a modeled latency reduction is responsible for lowering energy consumption from 1.403 to 0.254 mJ in KWS

classification and from 7.145 to 0.973 mJ in image classification, representing an 81.9% and 86.4% reduction, respectively. Energy efficiency also increases to 3,940.1 and 1,027.6 inferences/J respectively. These computations address RQ2 in the analytical setting.

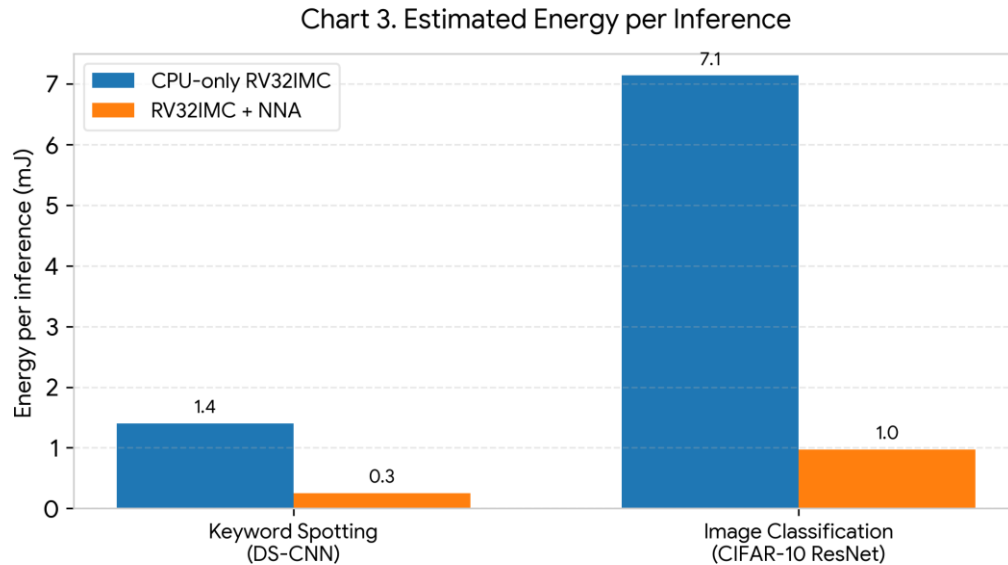


Chart 3. Estimated Energy per Inference derived from active power x latency. Values are analytical estimates.

8.3 Hardware Resource Utilization

The Artix-7 planning model designates 5,200 LUTs, 4,100 flip-flops, 4 DSP slices, and 20 Block RAMs to the baseline MCU. With the addition of the 16-MAC NNA and 32 KB scratchpad, the estimates worsen to 8,700 LUTs, 6,600 flip-flops, and 20

DSP slices, and 32 Block RAMs. This configuration is still contained within the mid-range FPGA prototyping budget stated in the model, thus addressing RQ3; however, it must be stressed that this configuration is fully evaluated only after RTL synthesis and implementation.

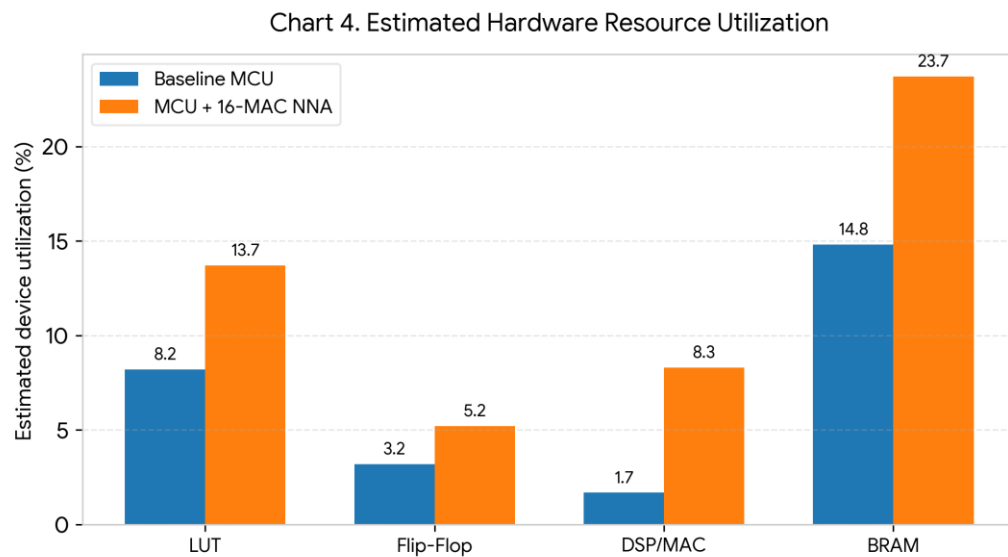


Chart 4. Estimated FPGA Resource Utilization for Baseline MCU and MCU + 16-MAC NNA (Artix-7 XC7A100T planning model).

8.4 Comparative Performance Summary

Table 2. Comparative Performance and Energy Summary

Evaluation Metric	KWS CPU-Only	KWS + NNA	Image CPU-Only	Image + NNA
Latency (ms)	101.7 ms	10.8 ms	503.2 ms	39.4 ms
Execution Cycles @ 100 MHz	10.17 M	1.08 M	50.32 M	3.94 M
Throughput (inf/s)	9.83 inf/s	92.59 inf/s	1.99 inf/s	25.38 inf/s
Active Power Budget (mW)	13.8 mW	23.5 mW	14.2 mW	24.7 mW
Energy per Inference (mJ)	1.403 mJ	0.254 mJ	7.145 mJ	0.973 mJ
Latency Speedup (x)	1.00x (Baseline)	9.42x	1.00x (Baseline)	12.77x
Energy Reduction (%)	— (Baseline)	81.9%	— (Baseline)	86.4%
Performance / Watt (inf/J)	712.5 inf/J	3940.1 inf/J	139.9 inf/J	1027.6 inf/J
Quality Constraint	≥90% Top-1	≥90% Top-1 (INT8)	≥85% Top-1	≥85% Top-1 (INT8)
Evaluation Method	Analytical Estimate	Analytical Estimate	Analytical Estimate	Analytical Estimate

Table 3. Hardware Resource Utilisation and Overhead Summary (Artix-7 Target)

0	Baseline RV32IMC	MCU + 16-MAC NNA	Absolute Overhead	Relative Increase (%)
Look-Up Tables (LUTs)	5,200 (8.2%)	8,700 (13.7%)	+3,500 LUTs	+67.3%
Registers / Flip-Flops (FF)	4,100 (3.2%)	6,600 (5.2%)	+2,500 FFs	+61.0%
DSP Blocks (DSP48E1)	4 (1.7%)	20 (8.3%)	+16 DSPs	+400.0%
Block RAM (36 Kb BRAM)	20 (14.8%)	32 (23.7%)	+12 BRAMs	+60.0%

9. Discussion and Architectural Trade-Offs

This discussion of design-space results demonstrates the interplay of parallel arithmetic, local memory bandwidth, DMA overhead, and power management. Our modeled 16-MAC array achieves an end-to-end acceleration of between 9.42x and 12.77x as opposed to the idealized 16x, due to memory transfers, tiling, and CPU orchestration. This is aligned with accelerator research stating that compute density must be balanced with memory coupling and data reuse [4], [11].

Research on energy consumption is another important contribution of this analysis. We estimate an increase in power consumption of between 70-74%, however we also estimate an improvement in energy of between 81-86% due to shortened time of execution. For IoT nodes that are duty cycled, this demonstrates that it is beneficial to perform computations at a fast pace and then transition into an idle state compared to continuously executing computations on the CPU. To truly demonstrate an improvement in the Operating Life of the system, power consumption must be measured at the implementation level.

INT8 provides an appropriate trade-off for computing and memory at the architectural level. Integer-only quantization can remove floating-point arithmetic and can reduce memory by an order of magnitude as compared to floating-point [13], [16]. This study considers the MLPerf Tiny-style accuracy thresholds as design boundaries instead of new accuracy assessment. Thus, architectural tradeoffs for INT8 are addressed, and remaining the modeling accuracy for INT8 is retained for validation at the implementation level.

9.1 Limitations

These results contain pre-silicon design-space estimates. Post-synthesis, post-route, and measured hardware results should be included. Cycle, power, and resource values are planning inputs and should be used for comparison. These values should be used to evaluate the NNA, which requires the use of a CPU fallback for irregular operators. Speedup, energy, resource, and

accuracy estimates must also be evaluated via RTL synthesis and FPGA implementation, along with testing at the workload level, prior to making any deployments.

10. CONCLUSION AND FUTURE WORK

This paper proposed a low-power RV32IMC microcontroller with a 16-MAC INT8 accelerator. With the assumptions of 100 MHz, the model indicated an expected 9.42x and 12.77x speedups and an expected energy savings of 81.9% and 86.4% for KWS and image classification, respectively. The FPGA planning model falls within the stated budget. therefore, RQ1–RQ3 were assessed as design-space estimates, while RQ4 was assessed as a tradeoff of INT8 complexity and storage within benchmarked accuracies. Accuracy and hardware performance will need to be assessed during the design implementation.

Complete synthesizable SystemVerilog RTL implementation and FPGA emulation on Xilinx Artix-7/UltraScale+ will be the starting focus. Physically implementing the design in a low power 65nm/28nm CMOS process node will be done next, including hardware support for INT4/INT2 quantization with sparse weight to include high precision. Power consumption and stress margins will be measured on high-speed shunt oscilloscopes.

REFERENCES

1. RISC-V International. The RISC-V instruction set manual, volume I: unprivileged architecture. Document Version 20260120. 2026 Jan.
2. Banbury C, et al. MLPerf Tiny benchmark. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (MLHPC); 2021. doi:10.1109/MLHPC54614.2021.00009.
3. MLCommons. MLPerf Tiny inference benchmark rules. Version 1.2. MLCommons Tiny Working Group; 2024-2026.
4. Garofalo A, Rusci M, Conti F, Rossi D, Benini L. PULP-NN: accelerating quantized neural networks on parallel

- ultra-low-power RISC-V processors. *Philos Trans R Soc A*. 2020;378(2164):20190155. doi:10.1098/rsta.2019.0155.
5. Garofalo A, Tagliavini G, Conti F, Rossi D, Benini L. XpulpNN: accelerating quantized neural networks on RISC-V processors through ISA extensions. In: *Design, Automation & Test in Europe Conference (DATE)*; 2020. p.186-191. doi:10.23919/DATE48585.2020.9116529.
 6. Rossi D, et al. Vega: a ten-core SoC for IoT endnodes with DNN acceleration and cognitive wake-up from MRAM-based state-retentive sleep mode. *IEEE J Solid-State Circuits*. 2022;57(1):127-139. doi:10.1109/JSSC.2021.3114881.
 7. Schiavone PD, et al. X-HEEP: an open-source, configurable and extendible RISC-V microcontroller. In: *Proceedings of the ACM Computing Frontiers Conference*; 2023. p.379-380. doi:10.1145/3587135.3591431.
 8. Machetti S, Schiavone PD, Müller TC, Peón-Quirós M, Atienza D. X-HEEP: an open-source, configurable and extendible RISC-V microcontroller for the exploration of ultra-low-power edge accelerators. *IEEE Trans Circuits Syst I Regul Pap*. 2024. arXiv:2401.05548. doi:10.48550/arXiv.2401.05548.
 9. Nadalini A, et al. A 3 TOPS/W RISC-V parallel cluster for inference of fine-grain mixed-precision quantized neural networks. In: *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*; 2023. p.1-6. doi:10.1109/ISVLSI59464.2023.10238679.
 10. Jain V, Giraldo S, De Roose J, Mei L, Boons B, Verhelst M. TinyVers: a tiny versatile system-on-chip with state-retentive eMRAM for ML inference at the extreme edge. *IEEE J Solid-State Circuits*. 2023;58(8):2360-2371. doi:10.1109/JSSC.2023.3236566.
 11. Yi X, et al. OpenGeMM: a high-utilization GeMM accelerator generator with lightweight RISC-V control and tight memory coupling. In: *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*; 2025. doi:10.1145/3658617.3697652.
 12. Borrás H, et al. Open-source FPGA-ML codesign for the MLPerf Tiny benchmark. *IEEE Embed Syst Lett*. 2022. arXiv:2206.11791. doi:10.48550/arXiv.2206.11791.
 13. Bruschi N, Garofalo A, Conti F, Tagliavini G, Rossi D. Enabling mixed-precision quantized neural networks in extreme-edge devices. In: *Proceedings of the ACM International Conference on Computing Frontiers*; 2020. p.217-220. doi:10.1145/3387902.3394038.
 14. Warden P. Speech Commands: a dataset for limited-vocabulary speech recognition. arXiv:1804.03209. 2018. doi:10.48550/arXiv.1804.03209.
 15. Krizhevsky A. Learning multiple layers of features from tiny images. Toronto: University of Toronto; 2009. Technical Report.
 16. Jacob B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2018. p.2704-2713. doi:10.1109/CVPR.2018.00286.

Creative Commons (CC) License

This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution–Non-Commercial–No Derivatives 4.0 International (CC BY-NC-ND 4.0) license. This license permits sharing and redistribution of the article in any medium or format for non-commercial purposes only, provided that appropriate credit is given to the original author(s) and source. No modifications, adaptations, or derivative works are permitted under this license.

About the Author

Usha Bhardwaj is an Assistant Professor in the Department of Electronics and Communication Engineering (ECE), based in Faridabad, Haryana, India. She is actively involved in teaching, academic activities, and research. Her professional interests focus on electronics, communication engineering, emerging technologies, and fostering innovation and technical skills among students.