WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Design of an Energy-Efficient RISC-V Integrated SRAM-Based Computing-in-Memory Accelerator

K. Nageswar Rao[1], A. Bhanu Priya[2]

[1]P.G. Scholar, Department of ECE, Sanketika Vidya Parishad Engineering College, Visakhapatnam, India
[2]Assistant Professor, Department of ECE, Sanketika Vidya Parishad Engineering College, Visakhapatnam, India
Mail Id: nageshkodiboina@gmail.com

ABSTRACT:

The rapid growth of Edge Artificial Intelligence (AI) applications, including IoT devices, wearable electronics, smart sensors and autonomous systems, demands computing architectures that provide high performance with low energy consumption. Conventional von Neumann organizations suffer from a memory bottleneck because weights, activations and intermediate data are repeatedly transferred between processor and memory. This paper presents an energy-efficient RISC-V integrated SRAM-based Computing-in-Memory (CIM) accelerator intended for Edge AI processing. The architecture combines a programmable RISC-V control subsystem with local feature-map SRAM, weight SRAM, an SRAM-CIM macro, input/output buffers and peripheral control circuitry. The processor provides programmable control and invokes CIM-specific operations, while the memory-centric subsystem performs highly parallel computation close to stored data. The design also considers SRAM-cell stability, sensing, write/read behavior, bit-line capacitance, routing, timing closure and technology scaling. The implementation results demonstrate successful synthesis and functional simulation of the SRAM design. The available power-analysis report estimates 0.073 W total on-chip power, consisting of approximately 0.002 W dynamic power and 0.070 W device-static power; the report explicitly identifies the confidence level as low, so these values are treated as synthesis/power-estimation results rather than measured silicon power.

Keywords: *RISC-V, Computing-in-Memory, SRAM, Edge AI, Energy-Efficient Architecture, VLSI, In-Memory Computing, AI Accelerator.*

1. INTRODUCTION

Edge AI moves machine-learning inference closer to the data source. The source material identifies IoT nodes, smart sensors, wearable electronics, speech interfaces and autonomous systems as representative applications in which local processing is desirable because latency, energy and communication bandwidth are constrained. Neural-network inference repeatedly accesses weights, activations and intermediate feature-maps, making data movement a major system-level concern.

In a conventional von Neumann organization, the processor and memory are separated. Data are transferred through buses or interconnects, processed by the computational core and then stored again. For AI workloads, the repeated transfer of operands can consume substantial energy and introduce latency even when arithmetic units are highly parallel. Computing-in-Memory addresses this architectural limitation by allowing computation to occur within or close to the memory array, reducing the physical distance travelled by frequently reused data.

The proposed work combines this memory-centric principle with RISC-V programmability. The architecture uses a RISC-V control subsystem together with feature-map SRAM, weight SRAM, an SRAM-CIM macro, buffers and peripheral control. The resulting heterogeneous organization allows regular control, preprocessing and post-processing to remain programmable while highly parallel convolutional computation is mapped to the CIM subsystem.

The paper is organized as follows. Section 2 summarizes the literature and architectural motivation. Section 3 describes the proposed RISC-V/SRAM-CIM organization and dataflow. Section 4 discusses SRAM, VLSI and HDL implementation considerations. Section 5 presents the simulation and power results available in the supplied documents. Section 6 discusses the findings and limitations, and Section 7 concludes the paper and identifies future work.

2. BACKGROUND AND RELATED WORK

RISC-V is an open instruction-set architecture with a modular base-plus-extension philosophy. The source material emphasizes that modularity can reduce unnecessary hardware by allowing an implementation to retain only instructions needed for a target application. This property is relevant to resource-constrained Edge AI systems, where deeply pipelined or feature-rich processors may introduce unnecessary area and power.

The supplied literature review discusses a range of RISC-V implementations, including Rocket, Roa Logic, ORCA, SiFive E31, SCR-1, BOOM, MRISCV, PicoRV32, Humming Bird, SHAKTI and related open processor projects. The reviewed processors illustrate different trade-offs among instruction support, pipeline depth, multiplication capability, area, power and performance. PicoRV32 is specifically described as a lightweight multicycle processor, whereas the proposed architecture seeks to combine programmable RISC-V control with a memory-centric compute engine.

On the memory side, SRAM is selected because it is compatible with standard CMOS logic and supports fast read/write access. The thesis material considers 6T, 8T and 10T SRAM structures and notes that the reference CIMR-V organization uses a 10T SRAM implementation for its CIM macro. The selection of the cell and array organization must balance stability, sensing overhead, bit-line capacitance, routing complexity, area and parallelism.

Prior CIM work in the supplied references demonstrates the broader potential of computation-in-SRAM and end-to-end energy-efficient AI hardware. The literature also highlights the importance of reducing feature-map movement, fusing adjacent operations and hiding weight-loading latency. The proposed architecture adopts these system-level principles rather than treating the SRAM macro as an isolated block.

Table 1. Comparison of instruction-set architecture features.

Feature	**MIPS**	**ARMv8**	**Open RISC**	**RISC-V**
Free & Open			X	X
Base + Extension				X
Compressed	X			X
32-bit	X	X	X	X
64-bit	X	X		X
128-bit				X
Privileged ISA				X
IEEE 754-2008		X		X

Table 2. CPI values reported in the supplied literature-review material for PicoRV32.

Instruction class	**CPI**
direct jump (jal)	3
ALU register + immediate	3
ALU register + register	3
branch not taken	3
memory load	5
memory store	5
branch taken	5
indirect jump (jalr)	6
branch not taken	3
shift operations	4–14
MUL	40
MULH [SU/U]	72
DIV[U]/REM[U]	40

Table 3. Implemented base-integer instruction groups supplied in the literature review.

I-Type	**Load I-Type**	**R-Type**	**U-Type**	**UJ-Type**	**S-Type**	**SB-Type**
ADDI	LW	ADD	LUI	JAL	SW	BEQ
SLTI	LB	SUB	AUIPC		SB	BNE
SLTIU	LH	SLL			SH	BLT
ORI	LBU	SLT				BLTU
XORI	LHU	SLTU				BGE
SLLI		XOR				BGEU

I-Type	Load I-Type	R-Type	U-Type	UJ-Type	S-Type	SB-Type
SRLI		SRL				
SRAI		SRA				
JALR		OR				
		AND				

3. PROPOSED RISC-V SRAM-CIM ARCHITECTURE

The proposed system is a heterogeneous accelerator in which the RISC-V processor forms the programmable control plane and the SRAM-CIM subsystem forms the memory-centric compute plane. The architecture contains instruction memory, a feature-map SRAM, a weight SRAM, an SRAM-CIM macro, input/output buffers and peripheral control circuits. The reference organization described in the thesis material uses a 512-Kb SRAM-CIM unit, 256-Kb feature-map SRAM, 512-Kb weight SRAM and an Ibex-based RISC-V core; these values are architectural reference information from the supplied material and are not presented here as independently measured results.

The RISC-V core fetches instructions, maintains program state and initiates specialized CIM operations. The feature-map SRAM stores input and intermediate activations, while the weight SRAM stages model parameters before they are written to the compute macro. Buffers adapt the memory organization to the access pattern of the compute array. The CIM macro performs parallel computation and its sensing circuitry converts the array activity into digital outputs.

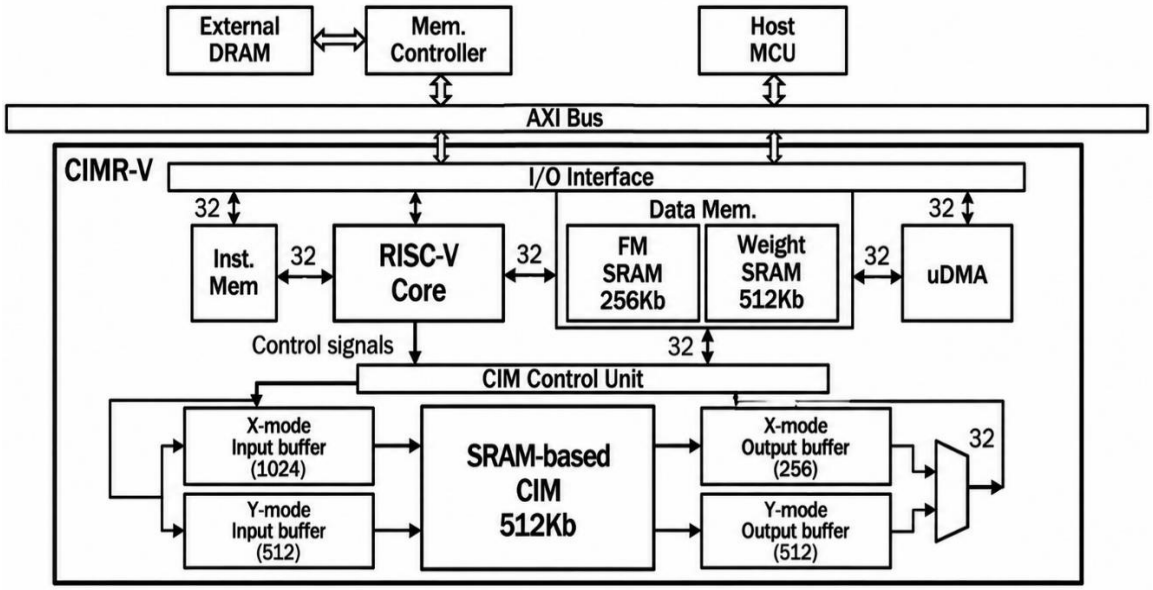


Figure 1. Overall CIMR-V-style architecture showing RISC-V control, feature-map SRAM, weight SRAM, buffers and SRAM-based CIM.

The processor-side organization in the source material also includes a single-cycle RV32IM embedded processor concept with interrupt control, timer, watchdog timer, AHB-Lite interface, SPI, UART and GPIO. This processor work provides the programmable embedded-control context for integrating the CIM subsystem. The supplied diagrams are retained below as supporting architecture figures.

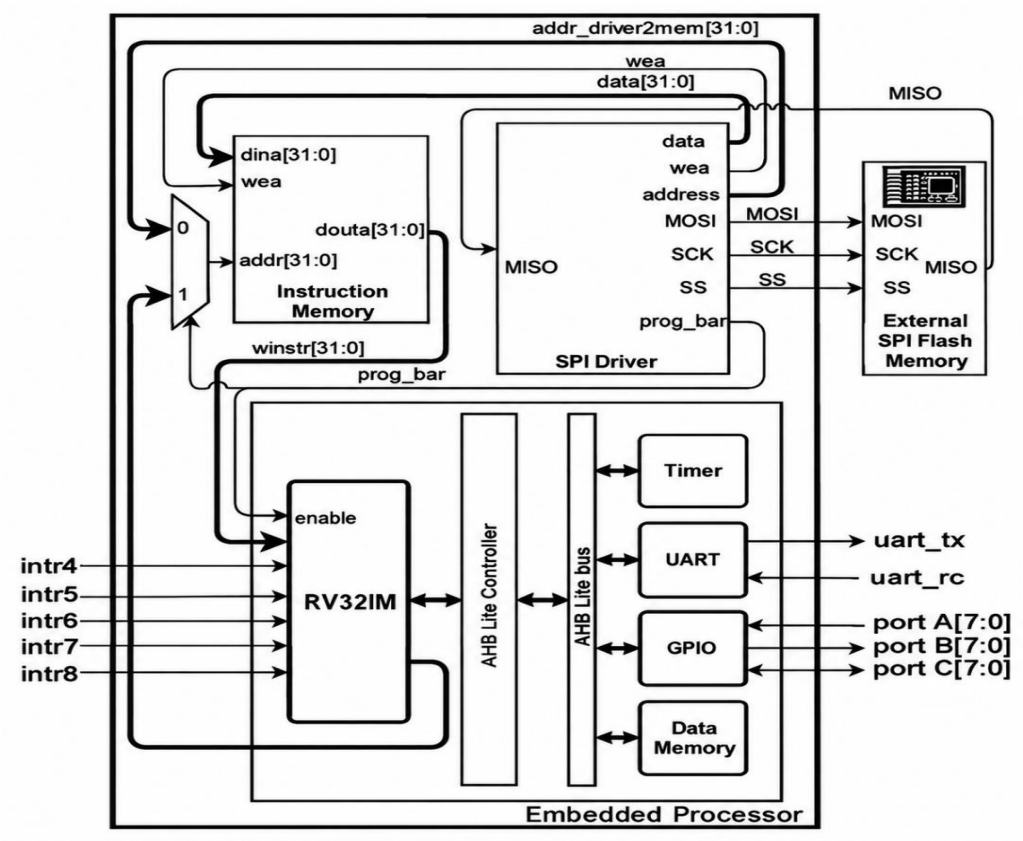


Figure 2. Top-level embedded RV32IM processor organization with peripheral interfaces.

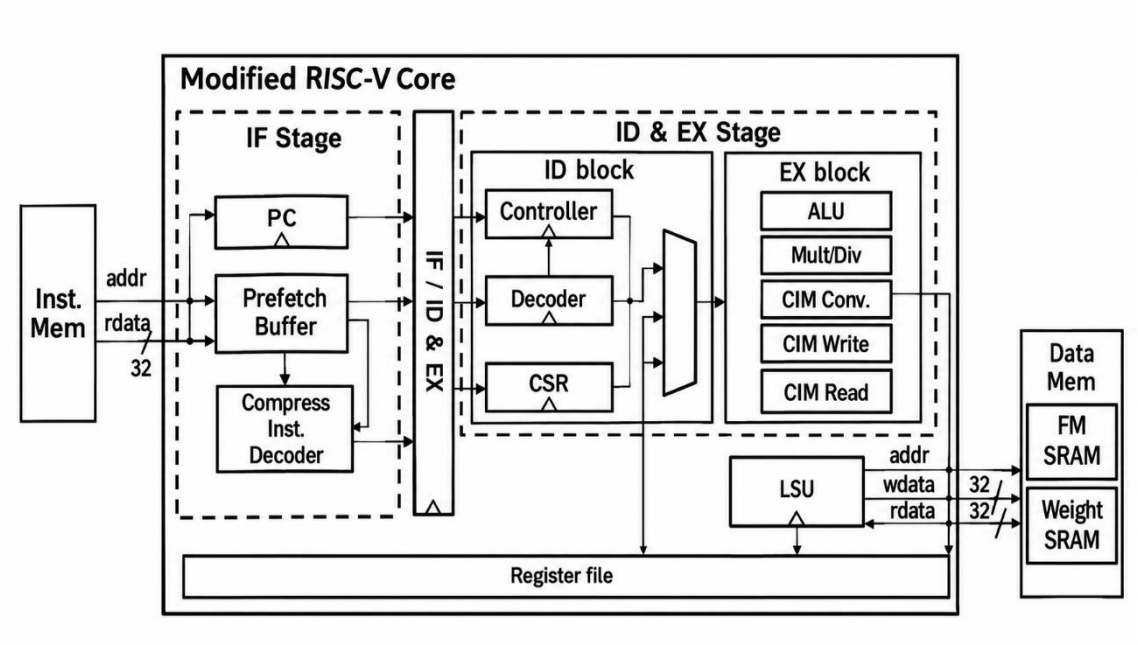


Figure 3. Modified RISC-V core architecture illustrating instruction fetch, decode/execute, register file and CIM-related operations.

3.1 SRAM-CIM Macro Organization

The SRAM-CIM macro is the principal compute engine. Stored weights participate directly in the computation so that multiple operations can be evaluated in parallel. The source material describes X-mode and Y-mode organizations as two possible ways of balancing input and output parallelism. The reference organization uses 1024 wordlines and 512 bitlines in X-mode and 512 wordlines and 1024 bitlines in Y-mode. These dimensions are reported as reference architectural values; the simulation result itself does not expose complete macro-level resource utilization.

3.2 SRAM Cell and Memory Array

The memory array uses a CMOS SRAM organization selected according to the desired compute operation, precision and stability requirements. The thesis considers 6T, 8T and 10T structures and identifies 10T SRAM as the reference CIMR-V implementation. Increasing array size can improve parallelism but also increases bit-line capacitance, routing complexity and sensing overhead. Consequently, array partitioning, bit-line loading, sensing and write drivers must be co-optimized.

3.3 RISC-V to CIM Interface

The RISC-V to CIM interface provides the software-visible mechanism for starting a CIM operation, supplying addresses or control information and receiving completion status. The source material identifies CIM read, write and convolution operations. A dedicated opcode can identify a CIM instruction, while function fields distinguish operation type and source/destination fields identify memory regions. The interface is intentionally compact so that instruction decoding and control overhead do not erode the energy benefits of in-memory computation.

3.4 Feature-Map Buffering and Dataflow

Feature-map buffering keeps frequently reused activation data close to the compute array. The supplied architecture uses input buffering and overlapping data blocks so that convolution windows can reuse data without repeated external transfers. Layer fusion further reduces feature-map movement by allowing the output of one operation to remain local when it can be consumed by the following operation.

3.5 Convolution, Pooling and Weight Fusion

The proposed dataflow overlaps convolution and max-pooling when the dependency structure permits. This is intended to reduce idle periods and improve CIM utilization. A local weight SRAM acts as a staging area so that a new weight block can be transferred while an independent block is being processed. This approach targets the weight-loading latency that can otherwise limit end-to-end CIM efficiency.

3.6 Hybrid Execution and End to End Inference

Not every AI operation must execute inside the CIM array. The proposed heterogeneous model assigns highly parallel convolutional work to CIM while retaining preprocessing, control and post-processing on the RISC-V side where appropriate. An end-to-end inference flow therefore includes model preparation, weight staging, input buffering, CIM convolution, pooling, local feature-map storage and final classification. This system-level view is important because a fast CIM macro alone does not guarantee low total inference energy.

4. SIMULATION RESULTS AND POWER ANALYSIS

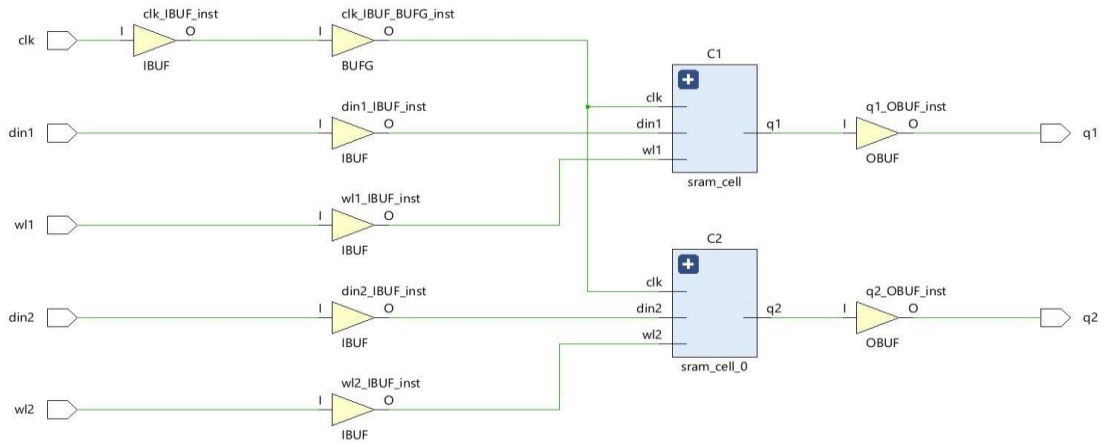


Figure 4. Synthesized schematic of the CIMR-V SRAM-cell test structure containing two storage elements, C1 and C2.

The results provide the directly available implementation evidence for the SRAM portion of the design. The synthesized schematic contains two SRAM cells, C1 and C2. Each cell has a data input, a write-control input, a common clock and an output. The visible interfaces are clk, din1, w1, q1, din2, w2 and q2. Input buffers are visible on the external clock/data/control signals and output buffers drive q1 and q2.

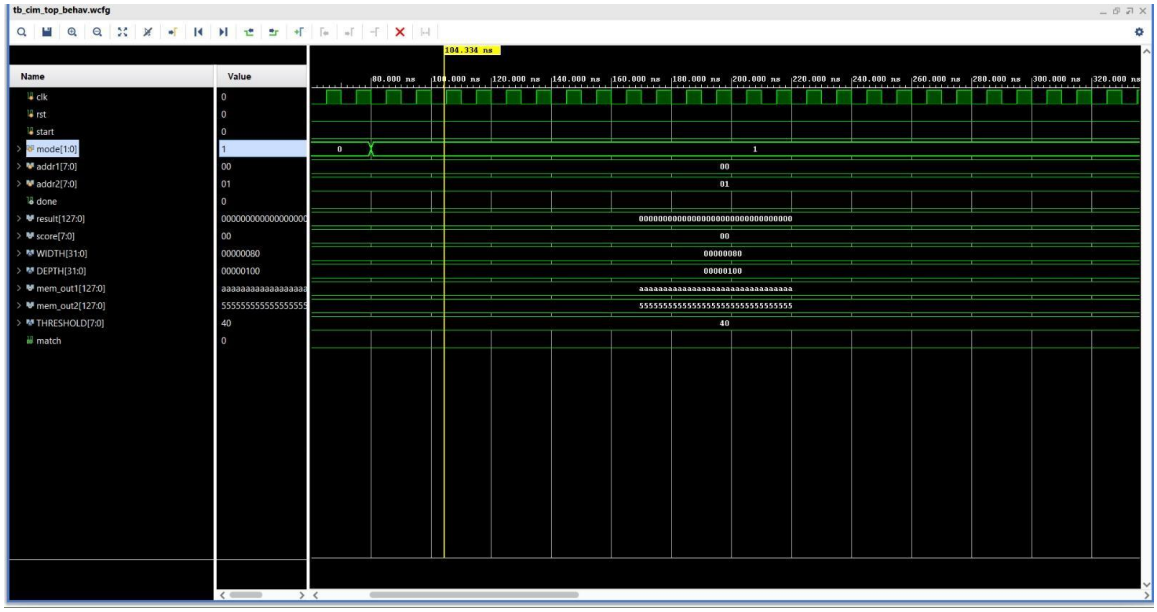


Figure 5. Functional simulation waveform of the SRAM/CIM test structure

The functional simulation waveform shows periodic clock transitions and changes in control, address/data, memory-enable and read/write signals. The supplied report states that the address/data buses assume defined values during active operations and that output/status responses change following the corresponding operations. These observations support functional evaluation of the synthesized SRAM structure.

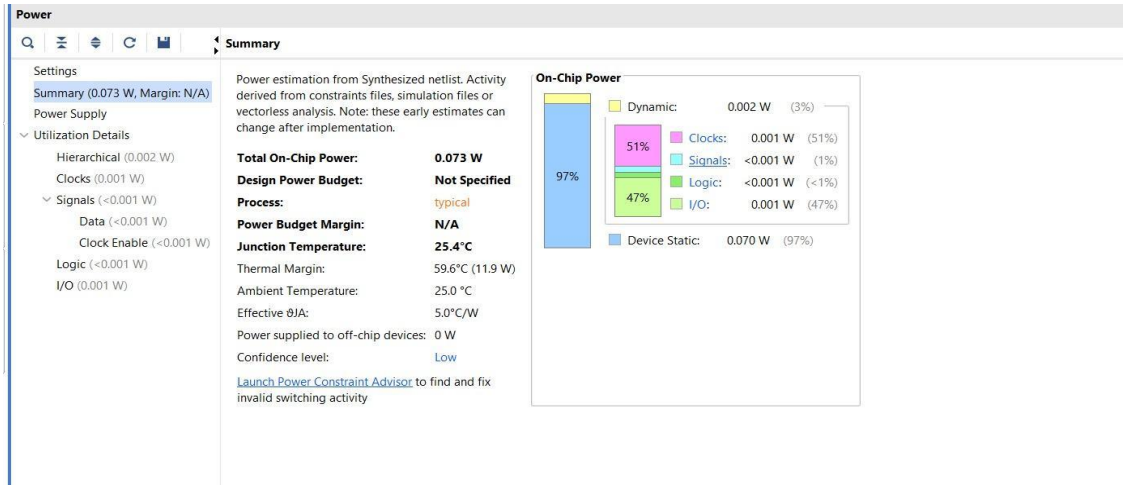


Figure 6. Power-analysis report for the synthesized SRAM design.

The power-analysis screenshot reports an estimated total on-chip power of 0.073 W. Dynamic power is approximately 0.002 W, while device-static power is approximately 0.070 W. The report identifies the result as an early estimate with low confidence; therefore, the values should not be interpreted as measured silicon power.

Table 4. Overall Power Summary Reports

Power category	**Power**	**Contribution**
Dynamic power	0.002 W	3%
Device static power	0.070 W	97%
Total on-chip power	0.073 W	100%

Table 5. Dynamic power breakdown report

Dynamic component	**Power**	**Contribution**
Clocks	0.001 W	51%
Signals	<0.001 W	1%
Logic	<0.001 W	1%
I/O	0.001 W	47%

Table 6. Thermal and operating information report

Parameter	**Reported value**
Junction temperature	25.4 °C
Ambient temperature	25.0 °C
Effective Theta-JA	5.0 °C/W
Thermal margin	59.6 °C
Off-chip power	0 W
Power confidence level	Low

Table 7. Synthesis-result summary

Parameter	**Result**
Design	SRAM-based
SRAM cells shown	2 (C1, C2)
Total on-chip power	0.073 W
Dynamic power	0.002 W
Device static power	0.070 W
Junction temperature	25.4 °C
Ambient temperature	25.0 °C
Effective Theta-JA	5.0 °C/W
Thermal margin	59.6 °C
Off-chip power	0 W
Power confidence	Low

The results do not expose complete LUT, flip-flop, BRAM, DSP, I/O or clock-resource utilization figures, and they do not provide a complete timing report. Accordingly, no such values are inferred in this paper. This distinction is important for reproducibility: the available evidence supports successful synthesis and functional simulation of the SRAM test structure and an estimated power result, but not a complete FPGA implementation benchmark.

5. DISCUSSION

The main architectural contribution described by the source material is the integration of a programmable RISC-V controller with SRAM-based computation rather than relying on a standalone accelerator or a conventional processor-memory organization. This arrangement targets the data-movement bottleneck by retaining feature-maps and weights locally and invoking highly parallel computation in the SRAM-CIM macro.

The architecture also separates responsibilities between programmable and specialized hardware. RISC-V remains useful for sequencing, control and operations that are not naturally mapped to the selected CIM precision, while the CIM array is intended for regular, highly parallel convolution. Feature-map buffering, layer fusion, pooling overlap and local weight staging are system-level mechanisms intended to increase reuse and reduce external traffic.

The available implementation evidence should, however, be interpreted carefully. The power figure of 0.073 W is an estimated on-chip value from the synthesis/power-analysis environment, and the report assigns a low confidence level. In addition, the screenshots do not provide enough information to establish complete timing, area or resource-utilization metrics. The independent implementation therefore remains partially characterized in the supplied material.

The design methodology is consistent with a VLSI-oriented implementation path in which SRAM-cell stability, sense-amplifier behavior, read/write margins, routing, bit-line capacitance and timing closure are considered together. Further post-layout characterization would be needed before making claims about silicon-level energy efficiency or technology-scaled performance.

6. CONCLUSION AND FUTURE SCOPE

This paper consolidated the supplied thesis material into a journal-oriented presentation of an energy-efficient RISC-V integrated SRAM-based Computing-in-Memory accelerator for Edge AI. The proposed architecture combines a programmable RISC-V control subsystem with local feature-map and weight memories, an SRAM-CIM macro, buffering and peripheral control. The organization is intended to reduce unnecessary data movement while preserving software-controlled execution and enabling parallel computation close to stored weights.

The supplied simulation results show successful synthesis and functional simulation of a two-cell SRAM test structure. The reported power analysis estimates 0.073 W total on-chip power, with 0.002 W dynamic and 0.070 W device-static power. Because the report explicitly labels the power confidence as low, these numbers are treated as estimates rather than measured results.

Future work identified in the source material includes optimization for lower power, reduced area and higher throughput; investigation of 8T and 10T SRAM cells; improved sense amplifiers and write drivers; additional customized RISC-V instructions; greater CIM parallelism; technology scaling toward advanced CMOS nodes; layout-aware optimization; improved timing closure; and integration of additional AI operations inside the CIM array. End-to-end evaluation should additionally report complete latency, energy per inference, area, memory utilization and application accuracy under clearly defined workload and technology conditions.

REFERENCES

[1] J. Wang, N. Varshney, C. Gentile, S. Blandino, J. Chuang, and N. Golmie, “Integrated sensing and communication: Enabling techniques, applications, tools and data sets, standardization, and future directions,” IEEE Internet of Things Journal, vol. 9, no. 23, pp. 23416–23440, 2022.

[2] M. Mahbub, M. S. A. Gazi, S. A. Provat, and M. S. Islam, “Multi-access edge computing-aware internet of things: MEC-IoT,” in ETCCE, 2020, pp. 1–6.

[3] X. Kong, Y. Wu, H. Wang, and F. Xia, “Edge computing for internet of everything: A survey,” IEEE Internet of Things Journal, vol. 9, no. 23, pp. 23472–23485, 2022.

[4] Y.-B. Lin, L.-K. Chen, M.-Z. Shieh, Y.-W. Lin, and T.-H. Yen, “CampusTalk: IoT devices and their interesting features on campus applications,” IEEE Access, vol. 6, pp. 26036–26046, 2018.

[5] I. F. Akyildiz and A. Kak, “The internet of space things/cubesats,” IEEE Network, vol. 33, no. 5, pp. 212–218, 2019.

[6] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of things for smart cities,” IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22–32, 2014.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.

[8] A. Ramos, A. Ullah, P. Reviriego, and J. A. Maestro, “Efficient Protection of the Register File in Soft-Processors Implemented on Xilinx FPGAs,” IEEE Transactions on Computers, vol. 67, no. 2, pp. 299–304, 2018.

[9] V. Petrović, G. Schoof, and Z. Stamenković, “Fault-tolerant TMR and DMR circuits with latchup protection switches,” Microelectronics Reliability, vol. 54, no. 8, pp. 1613–1626, 2014.

[10] M. Chiang and T. Zhang, “Fog and IoT: An overview of research opportunities,” IEEE Internet of Things Journal, vol. 3, no. 6, pp. 854–864, 2016.

[11] A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. Asanović, “The RISC-V instruction set manual volume II: Privileged architecture version 1.9,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-129, 2016.

[12] D. A. N. Gookyi and K. Ryoo, “Selecting a synthesizable RISC-V processor core for low-cost hardware devices,” Journal of Information Processing Systems, vol. 15, no. 6, pp. 1406–1421, 2019.

[13] C. Celio, P.-F. Chiu, B. Nikolić, D. A. Patterson, and K. Asanović, “BOOMv2: An open-source out-of-order RISC-V core,” in First Workshop on Computer Architecture Research with RISC-V (CARRV), 2017.

[14] N. Gala, A. Menon, R. Bodduna, G. S. Madhusudan, and V. Kamakoti, “Shakti processors: An open-source hardware initiative,” in 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), pp. 7–8, 2016.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.

[16] G. Chen, C. Parada, and G. Heigold, “Small-footprint keyword spotting using deep neural networks,” in IEEE ICASSP, 2014, pp. 4087–4091.

[17] J.-M. Hung, C.-J. Jhang, P.-C. Wu, Y.-C. Chiu, and M.-F. Chang, “Challenges and trends of nonvolatile in-memory-computation circuits for AI edge devices,” IEEE Open Journal of the Solid-State Circuits Society, vol. 1, pp. 171–183, 2021.

[18] H. Dbouk, S. K. Gonugondla, C. Sakr, and N. R. Shanbhag, “A 0.44-µJ/dec, 39.9-µs/dec, recurrent attention in-memory processor for keyword spotting,” IEEE Journal of Solid-State Circuits, vol. 56, pp. 2234–2244, 2021.

[19] Z. Yue, Y. Wang, Y. Qin, L. Liu, S. Wei, and S. Yin, “BR-CIM: An efficient binary representation computation-in-memory design,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 10, pp. 3940–3953, 2022.

[20] P. D. Schiavone, D. Rossi, A. Pullini, A. Di Mauro, F. Conti, and L. Benini, “Quentin: An ultra-low-power PULPissimo SoC in 22nm FDX,” in IEEE S3S, 2018, pp. 1–3.

[21] C.-S. Lin, F.-C. Tsai, J.-W. Su, S.-H. Li, T.-S. Chang, S.-S. Sheu, W.-C. Lo, S.-C. Chang, C.-I. Wu, and T.-H. Hou, “A 48 TOPS and 20943 TOPS/W 512kb computation-in-SRAM macro for highly reconfigurable ternary CNN acceleration,” in IEEE A-SSCC, 2021, pp. 1–3.

[22] M. Alwani, H. Chen, M. Ferdman, and P. Milder, “Fused-layer CNN accelerators,” in 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 1–12, 2016.

[23] K. Ueyoshi, I. A. Papistas, P. Houshmand, G. M. Sarda, V. Jain, M. Shi, Q. Zheng, S. Giraldo, P. Vrancx, J. Doevenspeck, D. Bhattacharjee, S. Cosemans, A. Mallik, P. Debacker, D. Verkest, and M. Verhelst, “DIANA: An end-to-end energy-efficient digital and analog hybrid neural network SoC,” in IEEE ISSCC, vol. 65, 2022, pp. 1–3.

[24] P. Warden, “Speech Commands: A dataset for limited-vocabulary speech recognition,” 2018.

[25] Y. Kim, W. Yang, and O. Mutlu, “Ramulator: A fast and extensible DRAM simulator,” IEEE Computer Architecture Letters, vol. 15, no. 1, pp. 45–49, 2016.