DOI: 10.1587/elex.22.20250329
Received: May 30, 2025
Accepted: July 1, 2025
Publicized: July 9, 2025

ELectronics
EXpress

IEICE Electronics Express, Vol.21, No.1, 1–6

LETTER

AES-RV: Hardware-Efficient RISC-V Accelerator with Low-Latency AES Instruction Extension for IoT Security

Van Tinh Nguyen[1], **Phuc Hung Pham**[1], **Vu Trung Duong Le**[2, a)], **Hoai Luan Pham**[2], **Tuan Hai Vu**[2], **and Thi Diem Tran**[3]

Abstract The Advanced Encryption Standard (AES) is a fundamental cryptographic algorithm widely used to secure data in embedded systems, IoT devices, and cloud computing platforms. However, recent research on AES hardware accelerators face challenges in achieving high performance and hardware efficiency, particularly when supporting multiple modes and key sizes. To address these limitations, this paper proposes a hardware-efficient RISC-V accelerator with low-latency AES instruction extension (AES-RV), designed to enhance both processing speed and energy efficiency across various AES configurations. Specifically, AES-RV incorporates three key optimizations: high-bandwidth internal buffers for continuous data processing, a specialized AES unit with low-latency custom instructions, and system pipelining with a ping-pong memory transfer mechanism. The AES-RV accelerator is implemented and evaluated on a real-time Xilinx ZCU102 FPGA system-on-chip (SoC), utilizing 29,608 FFs, 32,483 LUTs, and 12 BRAMs. Performance comparisons against a baseline RISC-V implementation for multiple AES modes and key sizes demonstrate latency improvements ranging from 195.5 times to 255.97 times. Additionally, evaluations against powerful CPUs and GPUs in real-time AES executions reveal energy efficiency gains of 9.92 times to 453.04 times. Compared to state-of-the-art AES hardware accelerators, AES-RV achieves throughput improvements of 13.56 times to 33.52 times, energy efficiency enhancements of 2.36 times to 58.76 times, and area efficiency gains of 91.42 times to 638.8 times.

Keywords: FPGA, cryptography, RISC-V, SoC, low-power, AES
Classification: Devices, circuits and hardware for IoT and biomedical applications

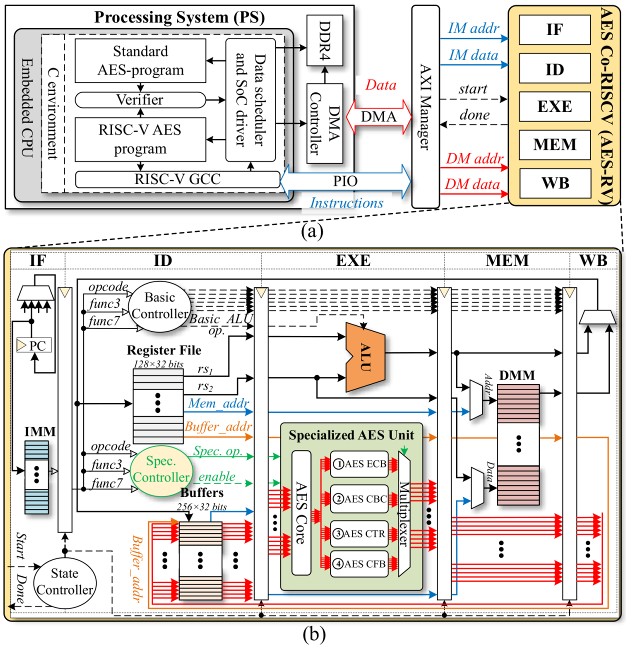
1. Introduction

RISC-V, introduced by the Berkeley research group in the late 2010s, is an open-source CPU architecture designed to promote flexibility and innovation. Its open Instruction Set Architecture (ISA) enables developers to design custom processors without licensing fees, facilitating advancements in specialized hardware. With high compatibility across various applications and superior energy efficiency, RISC-V is an ideal choice for resource-constrained devices [1–6]. Specifically, RISC-V accelerates encryption for IoT devices, supporting the implementation of traditional security algorithms such as SM3, SM4, and SHA-256 [7]. These applications ensure data security and enhance performance in IoT systems. Although RISC-V offers numerous advantages, its ISA continues to evolve to meet emerging demands in fields like artificial intelligence and cloud computing.

The Advanced Encryption Standard (AES) [7–11], standardized by the National Institute of Standards and Technology (NIST), is fundamental to modern digital security. Its adoption has expanded to cost-sensitive systems such as IoT devices, edge servers, fog servers, personal computers, and smartphones, where it secures data, meetings, video content, and confidential files [12–18]. These platforms require AES processing solutions that are energy-efficient to preserve battery life, high-performing to meet server security demands, and flexible to support various AES modes (ECB, CBC, CTR, CFB) and key sizes (AES-128, AES-192, AES-256) [19–26]. Implementing AES processors on the RISC-V architecture addresses these needs by offering customizable, high-speed, and low-cost hardware solutions. RISC-V allows for tailored AES integration, enhancing performance and reducing hardware complexity. Integrating AES capabilities into RISC-V-based systems enables the development of cost-effective, high-performance security solutions tailored to the specific requirements of contemporary information security infrastructures.

Several studies have explored AES acceleration [27], which is crucial for securing IoT, edge, and personal computing devices while maintaining efficiency and flexibility [19]. In [28], an AES-256 accelerator based on a 5-stage RV32IMFC RISC-V core improved speed by 82–84% over software solutions. However, it lacked a dedicated AES ISA extension, limiting integration flexibility. Similarly, [29] proposed a custom ISA for AES on the IBEX core, achieving up to 662 times higher energy efficiency than TinyAES. Yet, its design prevented parallel execution with the processor, reducing throughput in multi-task IoT applications. Meanwhile, [12] introduced a RISC-V cryptographic accelerator with dual concatenable 32-bit ALUs, enabling either two parallel 32-bit operations or a combined 64-bit operation. Despite achieving a 1.7–3.0 times processing speedup, it only supported AES-128 and lacked key expansion optimization, limiting its flexibility. Additionally, [27] proposed a flexible cryptographic unit supporting multiple cryptographic functions. However, it did not optimize AES key expansion or support multiple AES modes, leading to reduced throughput and flexibility. Overall, these RISC-V-based AES implementations still struggle to balance flexibility and hardware efficiency, leaving room for further improvements in integration, parallelism, and mode support.

[1] Le Quy Don Technical University, 236 Hoang Quoc Viet Street, Bac Tu Liem District, Hanoi, Vietnam
[2] Nara Institute of Science and Technology, 8916–5 Takayama-cho, Ikoma, Nara, 630-0192 Japan
[3] University of Information Technology, Vietnam National University, Ho Chi Minh City, 700000, Vietnam
[a)] le.duong@naist.ac.jp



(a)

(b)

Fig. 1 (a) Overview architecture of AES-RV at the system-on-chip level, (b) AES-RV architecture.

To overcome current challenges, this paper proposes AES-RV, a hardware-efficient RISC-V accelerator with low-latency AES instruction extensions for enhanced flexibility and energy efficiency. AES-RV integrates three key optimizations: high-bandwidth buffers for continuous data flow, a specialized AES unit with low-latency instructions, and pipelined processing with a ping-pong memory mechanism. Implemented on a Xilinx ZCU102 SoC FPGA, AES-RV outperforms existing AES platforms by supporting all AES modes and key sizes while meeting edge device constraints.

The remainder of this paper is organized as follows: Section 2. presents the details of the AES-RV proposal. Next, Section 3. shows the evaluation results of the AES-RV. Finally, section 4. concludes the paper.

2. Proposed AES-RV Architecture

2.1 System Architecture Overview

The overview SoC architecture of the AES Co RISC-V, as illustrated in Fig. 1(a), comprises two main components: the Processing System (PS) and the AES-RV module. The PS, controlled by an embedded CPU, executes the C environment to run both standard AES and RISC-V AES programs. The RISC-V AES program is compiled and transferred to the AES-RV for storage in the instruction memory. Meanwhile, the computational data is prepared by the Data Scheduler and SoC driver, facilitating data exchange with the AES-RV's data memory. To optimize data transfer efficiency, instruction data is transmitted via PIO transfer, while large-scale computational data is transferred through Direct Memory Access (DMA).

The AES-RV module consists of two primary components: the AXI Manager and the AES-RV core. The AXI Manager decodes data exchanged between the PS and the AES-RV. Specifically, computational data is stored in data memory (DMM), compiled hexadecimal code is stored in instruction memory (IM), and control signals such as start and done are managed by two controllers within the AES-RV. Similar to conventional RISC-V architectures, the AES-RV core features a five-stage pipeline: instruction fetch (IF), instruction decode (ID), execution (EXE), memory access (MEM), and writeback (WB). However, traditional RISC-V cores utilize a basic ALU with two input sources, performing one operation per instruction, which results in high latency for AES computations. To address this, the AES-RV core, illustrated in Fig. 1(b), integrates a Specialized AES Unit (SAU) to accelerate cryptographic operations. The SAU is a highly flexible ALU designed to support multiple AES modes, including ECB, CBC, CTR, and CFB, with various key sizes (128, 192, and 256 bits). The SAU is controlled via a custom instruction set extension, enabling faster multi-mode AES operations compared to standard RISC-V instructions. To ensure high-performance operation of the SAU, a high-bandwidth buffer system is implemented for intermediate data exchange between the SAU and DMM.

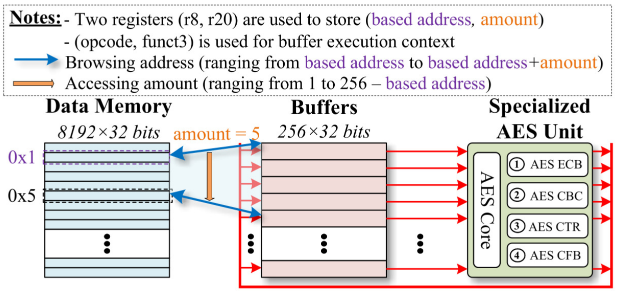
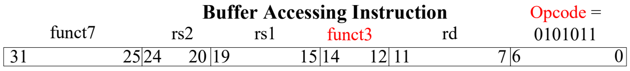

Notes: - Two registers (r8, r20) are used to store (based address, amount)
- (opcode, funct3) is used for buffer execution context
Browsing address (ranging from based address to based address+amount)
Accessing amount (ranging from 1 to 256 – based address)
Data Memory
8192×32 bits
amount = 5
0x1
0x5
Buffers
256×32 bits
Specialized AES Unit
AES Core
① AES ECB
② AES CBC
③ AES CTR
④ AES CFB

Fig. 2 High-bandwidth Buffer Set for Fast Continuous Data Accessing.



Opcode	funct3	Description
0101011	000	Latch the value of base_addr and amount
0101011	001	Set a flag to allow storing into the buffer

Fig. 3 Structure of the Buffer Accessing Instructions.

2.2 High-Bandwidth Internal Buffers for Fast Continuous Data Accessing

The use of the ALU in the RISC-V core, following the traditional RISC-V resource usage rules, requires many instructions to organize the data needed for primary processing. As a consequence, the AES computation performance of the conventional RISC-V core is extremely low since it does not meet the current security requirements. Therefore, the proposed AES-RV is improved by using an SAU that contains specialized components optimized to handle multi-mode AES. Accordingly, the SAU block requires substantial data for continuous computations through the computation loops. To ensure high computational efficiency, a 256×32-bit (8192 bits total) FF-based buffer set is placed before the SAU, holding critical data like keys, IVs, plaintexts, and ciphertexts. Unlike traditional Block RAM, this buffer set is designed for true parallel access: each 32-bit buffer has its own dedicated read/write path. This unique architecture enables the SAU to simultaneously load or store the entire 8192 bits in just one

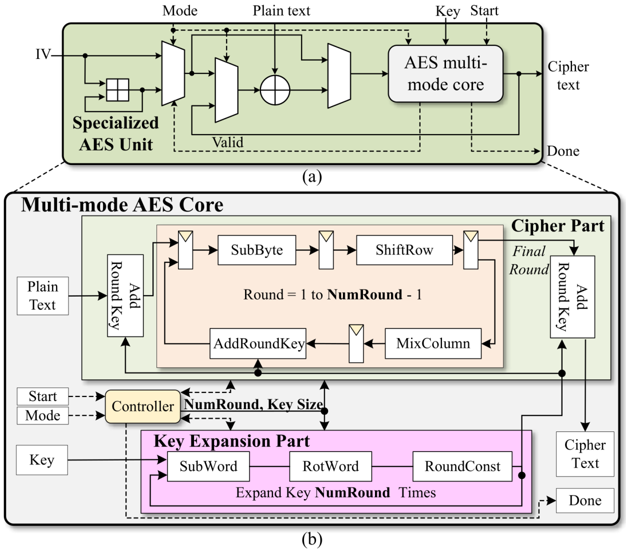


Fig. 4 Specialized AES Unit and Multi-mode AES Core architecture.

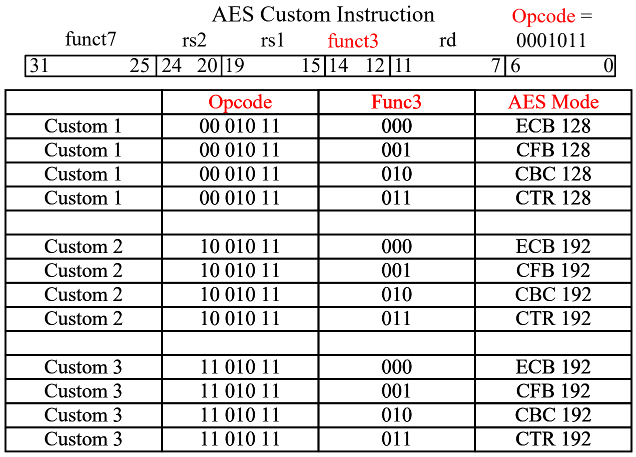
clock cycle, fundamentally bypassing shared-bus memory limitations and providing the extreme parallelism required for continuous, high-speed cryptographic operations.

Fig. 2 shows the detailed communication mechanism of the high-bandwidth buffer set with the data memory. In this scheme, a quantity of data can be transferred between the data memory (DMM) and the buffer based on the management of the values in registers r8 and r20. Register r8 stores the DMM address for reading or writing, while register r20 contains the number of 32-bit values to be read or written. This process is executed by the buffer accessing instruction described in Fig. 3. The buffer consists of one instruction that specifies the address and amount and another instruction to set the flag to start the read/write process. Once the buffer set is fully loaded with data, the SAU can load the entire content and immediately perform the AES Key Expansion and main round computation. When using the original RISC-V instruction set, after one round of computation, the system has to execute many instructions to rearrange the data and store it in the BRAM. By using the high-bandwidth buffer set, this process is completely eliminated. It should be noted that each data rearrangement requires many instructions and takes even longer than the AES computation process.

2.3 Specialized AES Unit (SAU) with Low-Latency Custom Instruction Extension

The use of conventional RISC-V instructions to implement AES results in high latency due to the large number of instructions required. To address this issue, we propose a Specialized AES Unit (SAU) as shown in Fig. 4 to accelerate AES computation. The SAU processes data including plain text, key, IV, and cipher text. Its control signals consist of `start` and `done`, while the configuration data, which contains the operating mode, determines the number of rounds and key size for the AES multi-mode core. The multi-mode AES core comprises three main components: the cipher part, the key expansion part, and the controller. The cipher part executes the main AES computation loop through four steps: AddRoundKey, SubByte, ShiftRow, and MixColumn. The number of computation rounds and the AES key size depend on the input mode, as determined by the controller. To minimize the critical path, the cipher part employs a 4-stage pipeline architecture. The key expansion part is responsible for generating the necessary round keys and consists of three main functions: SubWord, RotWord, and RoundConst. These functions expand the original key for use in the cipher part's computation rounds.

Control of the SAU is achieved through AES custom instructions, as illustrated in Fig. 5. Three groups of custom instructions are defined by opcodes corresponding to key sizes of 128, 192, and 256 bits. By altering the `func3` field, the SAU supports up to four AES modes: ECB, CFB, CBC, and CTR. In summary, the integration of the SAU and the internal buffer set significantly enhances the performance of multi-mode AES computation.



AES Custom Instruction

funct7	rs2	rs1	funct3	rd	Opcode = 0001011
31 25	24 20	19 15	14 12	11 7	6 0

	Opcode	Func3	AES Mode
Custom 1	00 010 11	000	ECB 128
Custom 1	00 010 11	001	CFB 128
Custom 1	00 010 11	010	CBC 128
Custom 1	00 010 11	011	CTR 128
Custom 2	10 010 11	000	ECB 192
Custom 2	10 010 11	001	CFB 192
Custom 2	10 010 11	010	CBC 192
Custom 2	10 010 11	011	CTR 192
Custom 3	11 010 11	000	ECB 192
Custom 3	11 010 11	001	CFB 192
Custom 3	11 010 11	010	CBC 192
Custom 3	11 010 11	011	CTR 192

Fig. 5 Custom instruction for calling Specialized AES Unit.

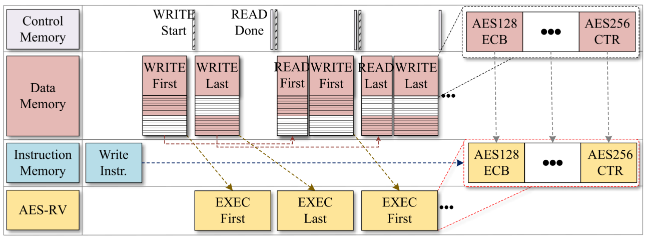


Fig. 6 Timing diagram of the system pipeline using ping-pong memory transfer mechanism.

2.4 System Pipelining with a Ping-Pong Memory Transfer Mechanism

In practice, transferring a large amount of data to the AES-RV core via the AXI interface introduces significant latency. In a straightforward system, the waiting time for reading and writing computational data can exceed the hardware processing time, leading to a system bottleneck. In other words, merely accelerating the AES-RV core cannot substantially improve the overall system performance. To address this issue, a system pipeline with a ping-pong memory transfer mechanism is proposed to mitigate the bottleneck caused by data transfer.

Fig. 6 illustrates the timing schedule of the proposed system pipeline with the ping-pong memory transfer mechanism. The data memory is divided into two equal parts: *first*

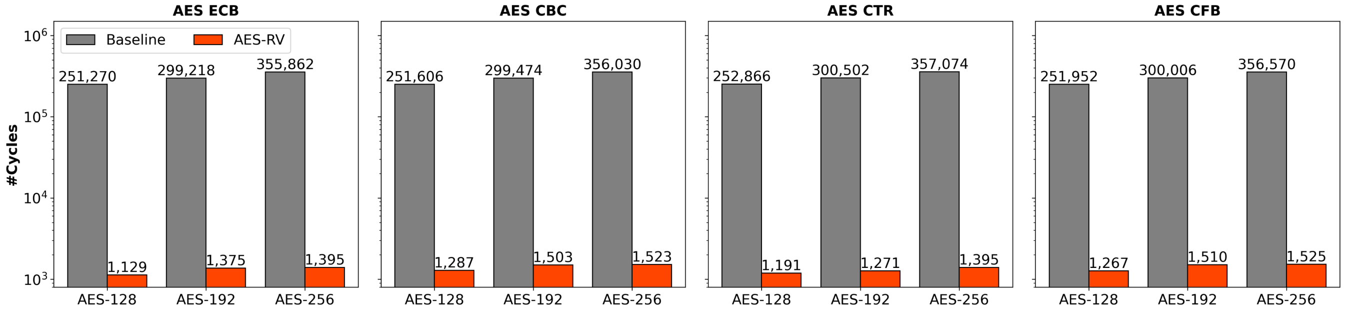


Fig. 7 Execution Cycles of AES-RV vs. Baseline RISC-V for Different AES Modes and Key Sizes

and *last*. Initially, instructions are loaded into the AES-RV core with configurations that support multiple AES modes and key sizes. Subsequently, the input data is filled into the first half of the data memory (WRITE First), followed by a control signal to initiate the AES-RV core for processing this portion (EXEC First). During the EXEC First phase, new data is written into the second half of the data memory (WRITE Last). Notably, if the AES-RV core completes the EXEC First phase, the system performs multiple tasks during the next execution cycle. It simultaneously reads the processed data from the first half of the data memory (referred to as READ First). At the same time, it writes new input data into that same memory region (WRITE First). Meanwhile, the AES-RV core processes the second half of the data memory (EXEC Last). This approach allows the processing system to read the output and write new input data into the alternate memory region while the AES-RV core continues computation. As a result, the entire SoC system eliminates idle wait times for data transfer.

For optimal performance, data transfer time must be shorter than hardware execution time, ensuring that system performance aligns with hardware computing speed, regardless of transfer latency.

3. Evaluation Results

3.1 Implementation results on ZCU102 FPGA SoC

To evaluate the correctness and practicality, AES-RV was implemented on the Zynq UltraScale+ MPSoC ZCU102 FPGA SoC, as illustrated in Fig. 1. The processing system is managed by an ARM Cortex-A53 CPU running Debian GNU/Linux 11, installed via PetaLinux 2022.2. The programmable logic hosts the AES-RV IP, synthesized using Vivado Design Suite 2022.2. AES-RV was verified by executing all AES functions with key sizes from the set $\mathcal{K} = \{128, 192, 256\}$ bits across multiple modes defined by the set $\mathcal{M} = \{\text{ECB}, \text{CBC}, \text{CTR}, \text{CFB}\}$. The real-time SoC verification demonstrated that AES-RV successfully processed 100,000 random plaintext inputs for each mode with 100% accuracy at a frequency of 200 MHz. Utilization reports indicate that AES-RV occupies 29,608 flip-flops (FFs), 32,483 lookup tables (LUTs), and 12 Block RAM tiles (36 KB each). Furthermore, power analysis shows that AES-RV consumes a total power of 4.043 W, with the AES-RV IP contributing 0.043 W in dynamic power.

In general, AES-RV exhibits the ability to operate at high frequencies on real-time SoC systems while consuming minimal power, rendering it an ideal candidate for integration into SoC-based applications.

3.2 Performance Comparison of AES-RV and Baseline RISC-V Implementation

In this section, the execution cycles of AES-RV when computing AES for all key sizes $\mathcal{K}$ and modes $\mathcal{M}$ on four consecutive data blocks are compared in detail with the baseline RISC-V implementation. The comparison results are obtained through waveform simulation extraction, as detailed in Fig. 7.

For the AES-ECB mode, AES-RV achieves execution cycles ranging from 1,129 to 1,395 cycles, outperforming the baseline RISC-V by **217.61 times** (251,270 cycles vs. 1,129 cycles) to **255.10 times** (355,862 cycles vs. 1,395 cycles). In the AES-CBC mode, AES-RV completes encryption within 1,287 to 1,523 cycles, demonstrating a speedup of **195.50 times** (251,606 cycles vs. 1,287 cycles) to **233.77 times** (356,030 cycles vs. 1,523 cycles) compared to the baseline. For the AES-CTR mode, AES-RV executes in 1,191 to 1,395 cycles, surpassing the baseline RISC-V with a performance gain from **212.31 times** (252,866 cycles vs. 1,191 cycles) to **255.97 times** (357,074 cycles vs. 1,395 cycles). Finally, in the AES-CFB mode, AES-RV achieves 1,267 to 1,525 cycles, significantly outperforming the baseline by **198.68 times** (251,952 cycles vs. 1,267 cycles) to **233.82 times** (356,570 cycles vs. 1,525 cycles).

Generally, AES-RV significantly outperforms the baseline RISC-V by leveraging custom instructions and buffer optimizations.

3.3 Performance and Energy Efficiency Evaluation on Real-Time Software Platforms

To assess performance and energy efficiency, the proposed hardware was benchmarked on 8,192 AES-CBC test cases (128-bit key) and compared against high-performance CPUs (Intel Core i7-12700H, Core i9-10940X), GPUs (Quadro RTX 8000, GeForce GTX 1080 with 2048-thread execution), and the ARM Cortex A53 on the ZCU102 FPGA SoC for energy efficiency. Detailed results are shown in Fig. 8.

In the **execution time comparison** shown in Fig. 8 (a), AES-RV demonstrates comparable performance to **Intel Core i7-12700H @ 2.3GHz**, achieving a speedup of **1.06 times** (11.6 ms vs. 10.9 ms). Furthermore, AES-RV outperforms Intel Core i9-10940X @ 3.3GHz by **1.30 times** (14.2 ms vs. 10.9 ms). This difference can be attributed to architectural improvements introduced in the newer Intel Core

Table I Comparison of AES Implementations on hardware platforms in terms of Throughput and Hardware Efficiency.

References	Devices	F_{max} (MHz)	FFs	LUTs	BRAMs	#Slices††	Power (W)	Throughput (Mbps)	Energy Efficiency (Mbps/W)	Area Efficiency (Mbps/Slice)
ATC 2024 [27]	ZCU102 FPGA	210	7,584* 29,644†	7,562* 34,898†	16	2,531* 9,365†	0.136* 4.22†	4.81	3.54E+01* 1.14E+00†	1.90E-04* 5.14E-04†
ICECS 2024 [28]	22nm FDSOI ASIC	1,000	-	-	-	-	0.008	7.07	8.84E+02	-
PRIME 2024 [29]	Nexys Artix-7 FPGA	50	-	609	-	33,650	397	2.86	1.15E+02	8.50E-05
DDECS 2024 [30]	PYNQ Z2 FPGA	100	10,454	15,885	-	7,943	-	4.72	-	5.94E-04
AES-RV	ZCU102 FPGA	241	7,548* 29,608†	5,147* 32,483†	12	1,767* 8,601†	0.046* 4.043†	95.88	2.08E+03* 2.37E+01†	5.43E-02* 1.11E-02†

(*) Denotes results for the AES core only; (†) Denotes results for the entire SoC.

(††) PYNQ Z2: 1 Slice = 6 LUTs, 8 FFs; ZCU102: 1 Slice = 4 LUTs, 8 FFs, and 1 BRAM 36Kb = 40 slices.

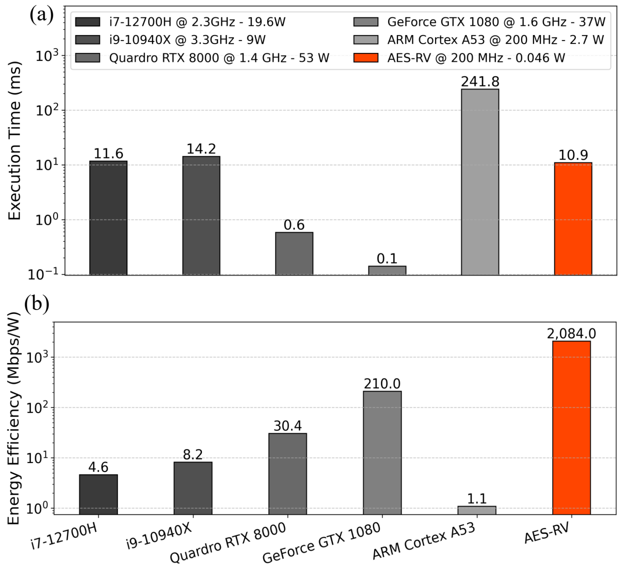


Fig. 8 Comparisons with powerful CPUs/GPUs on (a) execution time and (b) energy efficiency.

i7-12700H, including higher instructions-per-cycle (IPC), more efficient core designs, and improved memory subsystems. Despite its lower clock frequency, the i7-12700H's architectural advantages enable it to outperform the older i9-10940X in real-world tasks. Compared to ARM Cortex A53 CPUs @ 200MHz, AES-RV exhibits a remarkable speedup of **22.18 times** (241.8 ms vs. 10.9 ms). Regarding high-performance GPUs such as Quadro RTX 8000 @ 1.4 GHz and GeForce GTX 1080 @ 1.6 GHz, AES-RV does not surpass them in execution time, as GPUs complete AES operations in 0.6 ms and 0.1 ms, respectively. However, these CPUs and GPUs operate at significantly higher power consumption, resulting in inferior energy efficiency. The energy efficiency comparison, measured in Mbps/W and detailed in Fig. 8 (b), highlights the substantial advantage of AES-RV over traditional CPUs and GPUs. Specifically, AES-RV achieves **453.04 times** (4.6 Mbps/W vs. 2084 Mbps/W), higher efficiency than Intel Core i7-12700H @ 19.6W **254.15 times** (8.2 Mbps/W vs. 2084 Mbps/W) higher efficiency than Intel Core i9-10940X @ 9W , and **1894.55 times** (1.1 Mbps/W vs. 2084 Mbps/W) higher efficiency than ARM Cortex A53 @ 2.7W . Similarly, AES-RV also outperforms high-power GPUs, providing **68.55 times** (30.4 Mbps/W vs. 2084 Mbps/W) better energy efficiency than Quadro RTX 8000 @ 53W and **9.92 times** (210 Mbps/W vs. 2084 Mbps/W) better energy efficiency than GeForce GTX 1080 @ 37W.

Overall, AES-RV matches modern CPUs in speed while vastly outperforming CPUs and GPUs in energy efficiency, making it ideal for low-power, real-time applications.

3.4 Comprehensive Performance and Hardware Efficiency Evaluation for AES Implementations

To demonstrate the improvements in hardware efficiency, key evaluation metrics, including maximum frequency, hardware resource utilization, power consumption, throughput, energy efficiency, and area efficiency, are summarized in Table I for comparison with related work [27–30]. Notably, only AES-RV and [27] support full SoC execution, whereas other implementations focus solely on core-level development. The formulas for throughput, energy efficiency, and area efficiency are defined in Eq. (1)–(3).

$$\text{Throughput} = \frac{\text{F}_{\text{max}} \times \text{Block Size}}{\text{Cycles per Block}} \quad \text{(Mbps)} \tag{1}$$

$$\text{Energy Efficiency} = \frac{\text{Throughput}}{\text{Power Consumption}} \quad \text{(Mbps/W)} \tag{2}$$

$$\text{Area Efficiency} = \frac{\text{Throughput}}{\text{Total Slices}} \quad \text{(Mbps/Slice)} \tag{3}$$

Compared to [27] in ATC 2024, AES-RV supports a maximum operating frequency that is **1.15 times** higher (241 MHz vs. 210 MHz), while also utilizing fewer hardware resources for both the core and SoC by **1.43/1.09 times** (1,767/8,601 vs. 2,531/9,365 slices). Additionally, it consumes **2.96/1.05 times** less power (0.046/4.043 W vs. 0.136/4.22 W). In terms of performance, AES-RV achieves **19.94 times** higher throughput (95.88 Mbps vs. 4.81 Mbps). Moreover, AES-RV surpasses [27] in energy efficiency by **58.76/20.79 times** (2084.0/23.7 Mbps/W vs. 35.4/1.14 Mbps/W) and in area efficiency by **285.26/21.6 times** (0.0543/0.0111 Mbps/Slice vs. 0.00019/0.00051 Mbps/Slice). Similarly, compared to [28] in ICECS 2024, AES-RV delivers **13.56 times** higher throughput (95.88 Mbps vs. 7.07 Mbps) and achieves **2.36 times** greater energy efficiency (2084.0 Mbps/W vs. 884 Mbps/W). When comparing with [29] in PRIME 2024, AES-RV operates at a maximum frequency that is **4.82 times** higher (241 MHz vs. 50 MHz), while also utilizing **19.05 times** fewer core resources (1,767 slices vs. 33,650 slices) and consuming **8,630 times** less power (0.046 W vs. 397 W). In terms of performance, AES-RV achieves **33.52 times** higher throughput (95.88 Mbps vs. 2.86 Mbps) and further demonstrates **18.08 times** greater energy efficiency (2084.0 Mbps/W vs. 115.2 Mbps/W), along with **638.8 times** better area efficiency (0.0543 Mbps/Slice vs. 0.000085 Mbps/Slice). Finally, compared to [30] in DDECS 2024, AES-RV supports a maximum frequency **2.41 times** higher (241 MHz vs. 100 MHz) while requiring **4.5 times** fewer core resources (1,767 slices vs. 7,943 slices). Regarding performance, AES-RV achieves **20.32 times** higher throughput (95.88 Mbps vs. 4.72 Mbps) and outperforms in area efficiency by **91.42 times** (0.0543 Mbps/Slice vs. 0.000594 Mbps/Slice).

In conclusion, AES-RV demonstrates superior throughput and hardware efficiency, benefiting from optimized instruction sets and an efficient architecture tailored for real-time SoC environments.

4. Conclusion

This paper proposes a hardware-efficient RISC-V accelerator with low-latency AES instruction extension (AES-RV), enhancing flexibility and energy efficiency in cryptographic applications. By integrating three key optimizations—high-bandwidth internal buffers, a specialized AES unit with low-latency custom instructions, and system pipelining with a ping-pong memory mechanism—AES-RV significantly improves processing speed and hardware efficiency over conventional AES implementations. FPGA SoC experiments demonstrate notable gains in performance and energy efficiency, making AES-RV a strong candidate for secure, high-performance embedded systems. Future work will extend AES-RV with additional instruction sets for post-quantum cryptography, including CRYSTALS-Kyber and CRYSTALS-Dilithium, further broadening its applicability.

References

[1] M. Sharma and et al., "A Survey of RISC-V CPU for IoT Applications," in *Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2022*, SSRN, February 2022.

[2] J. Park and et al., "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, pp. 4106–4119, 2024.

[3] T.-T. Hoang and et al., "Low-power high-performance 32-bit RISC-V microcontroller on 65-nm silicon-on-thin-BOX (SOTB)," *IEICE Electronics Express*, vol. 17, pp. 20200282–20200282, 2020.

[4] K.-D. Nguyen and et al., "A trigonometric hardware acceleration in 32-bit RISC-V microcontroller with custom instruction," *IEICE Electronics Express*, vol. 18, no. 16, pp. 20210266–20210266, 2021.

[5] M. Liu, "A co-design method of customized ISA design space exploration and fixed-point library construction for RISC-V dedicated processor," *IEICE Electronics Express*, vol. 19, no. 13, pp. 20220244–20220244, 2022.

[6] Q. Yin and et al., "Design and implementation of RISC-V system-on-chip for SPWM generation based on FPGA," *IEICE Electronics Express*, vol. 21, no. 24, pp. 20240603–20240603, 2024.

[7] D. H. A. Le and et al., "High-Efficiency RISC-V-Based Cryptographic Coprocessor for Security Applications," in *International SoC Design Conference (ISOCC)*, pp. 103–104, 2024.

[8] Q. Zou and et al., "28nm asynchronous area-saving AES processor with high Common and Machine learning side-channel attack resistance," *IEICE Electron. Express*, vol. 18, p. 20210309, 2021.

[9] Leurent and et al., "New representations of the AES key schedule," *J. Cryptol.*, vol. 38, p. 1, 2025.

[10] W. K. Lee and et al., "Efficient Implementation of AES-CTR and AES-ECB on GPUs With Applications for High-Speed FrodoKEM and Exhaustive Key Search," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 69, p. 2962, 2022.

[11] Y. Zhang and et al., "A lightweight AES algorithm implementation for encrypting voice messages using field programmable gate arrays," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, p. 3878, 2022.

[12] N. H. Nguyen, , and et al., "LI-RV: A Fast and Efficient RISC-V based Coprocessor for Lightweight Cryptography," in *2024 21st International SoC Design Conference (ISOCC)*, pp. 1–2, 2024.

[13] K. Stangherlin and M. Sachdev, "Design and Implementation of a Secure RISC-V Microprocessor," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 30, p. 1705, 2022.

[14] Pan and et al., "A Lightweight AES Coprocessor Based on RISC-V Custom Instructions," *Secur. Commun. Netw.*, p. 9355123, 2021.

[15] O. Simola and et al., "RISC-V Core with AES-256 Accelerator," in *31st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, p. 1, 2024.

[16] Ignatius and et al., "Power Side-Channel Attacks on Crypto-core based on RISC-V ISA for High-security Applications," in *IEEE Access*, vol. 12, p. 150230, 2024.

[17] Cheng and et al., "A Hardware Security Evaluation Platform on RISC-V SoC," in *IEEE International Test Conference in Asia (ITC-Asia)*, p. 1, 2024.

[18] O. Simola and et al., "RISC-V Core with AES-256 Accelerator," in *31st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, p. 1, 2024.

[19] Salman and et al., "Lightweight Modifications in the Advanced Encryption Standard (AES) for IoT Applications: A Comparative Survey," in *2022 International Conference on Computer Science and Software Engineering (CSASE)*, pp. 325–330, 2022.

[20] S. Jeon and et al., "Cross-Layer Encryption of CFB-AES-TURBO for Advanced Satellite Data Transmission Security," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, p. 1, 2022.

[21] E. Choi and et al., "AESware: Developing AES-enabled low-power multicore processors leveraging open RISC-V cores with a shared lightweight AES accelerator," *Eng. Sci. Technol. Int. J.*, vol. 60, p. 1, 2024.

[22] C. Duran and E. Roa, "A 10pJ/bit 256b AES-SoC Exploiting Memory Access Acceleration," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 69, p. 1612, 2022.

[23] Y. M. Kuo and et al., "RISC-V Galois Field ISA Extension for Non-Binary Error-Correction Codes and Classical and Post-Quantum Cryptography," *IEEE Trans. Comput.*, vol. 72, p. 682, 2023.

[24] P. Nannipieri and et al., "VLSI Design of Advanced-Features AES Cryptoprocessor in the Framework of the European Processor Initiative," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 30, p. 177, 2022.

[25] W. Wang and et al., "An energy-efficient crypto-extension design for RISC-V," *Microelectron. J.*, vol. 115, p. 105165, 2021.

[26] D. Reis and et al., "IMCRYPTO: An In-Memory Computing Fabric for AES Encryption and Decryption," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 30, p. 553, 2022.

[27] V. T. D. Le and et al., "RVCP: High-Efficiency RISC-V Co-Processor for Security Applications in IoT and Server Systems," in *2024 International Conference on Advanced Technologies for Communications (ATC)*, IEEE, 2024.

[28] Simola and et al., "RISC-V Core with AES-256 Accelerator," in *2024 31st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4, 2024.

[29] Zgheib and et al., "Extending a RISC-V core with an AES hardware accelerator to meet IOT constraints," in *SMACD / PRIME 2021; International Conference on SMACD and 16th Conference on PRIME*, pp. 1–4, 2021.

[30] M. N. Rizi and et al., "Optimised AES with RISC-V Vector Extensions," in *2024 27th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pp. 57–60, 2024.