RISC-V core with AES-256 accelerator

Otto Simola*, Aleksi Korsman*, Verneri Hirvonen*, Antti Tarkka*, Julius Helander*,
Kimmo Järvinen†, Marko Kosunen*, Jussi Ryynänen*
*Aalto University, Department of Electronics and Nanoengineering, Finland {firstname.lastname}@aalto.fi
†Xiphera, Finland {firstname.lastname}@xiphera.com

2024 31st IEEE International Conference on Electronics, Circuits and Systems (ICECS) | 979-8-3503-7720-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICECS61496.2024.10848921

***Abstract*—This paper presents a RISC-V core integrated with a cryptographic accelerator for 256-bit Advanced Encryption Standard (AES-256). It supports several block cipher modes and has been integrated as an extension to a 5-stage RV32IMFC RISC-V core implemented in 22 nm FD-SOI. For performance comparison, the hardware accelerator was verified with an extensive verification environment involving simulations and testing with Field Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC) implementations against a C-program solution implemented with the openSSL library, running on the implemented RISC-V core. The accelerator achieves 82–84% faster AES cipher operation compared to the software solution.**

***Index Terms*—RISC-V, AES-256, Accelerator, FPGA, ASIC**

I. Introduction

THE popularity of the open-source RISC-V instruction set architecture (ISA) has sparked growing interest for creating open-source and extensible processors. The growth has allowed adaptation of the RISC-V ISA to several fields, ranging from machine learning to quantum computing. Additionally, the integration of hardware accelerators as extensions of the processor hardware has enabled designers to achieve improvements in computational efficiency. Hence, RISC-V processors provide a great open platform for research and development for development of speed/power consumption optimized algorithm implementations.

The increase of computational needs and software security in various fields, such as telecommunication, has imposed more stringent requirements on encryption and decryption performance. Due to this situation, a generic processor with a software solution alone will not suffice for cryptographic computation tasks, such as the AES-256 cipher, in terms of performance. A solution to this problem is to amend the processor with a custom hardware accelerator optimized for the task, improving the performance of the system on chip (SoC) and enabling the offloading of the cryptographic task from the core to the accelerator with a cost of increased area. Moreover, in combination with the RISC-V core and the cryptographic accelerator form a flexible processing platform, whose functionality can be adapted according to the application needs. In the past, for cryptographic tasks, similar accelerators relying on large process nodes, software based implementations or short key lengths have been implemented such as in, [1], [2] and [3].

To advance these efforts, this research paper proposes to use a cryptographic AES-256 hardware accelerator IP, integrated into a 5-staged RISC-V RV32IMFC compliant processor. The AES-256 hardware accelerator supports the Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) block cipher modes. The aim is to analyse the impact of the experimental hardware prototype design on the cryptographic performance by comparing the result to software implementation. The performance is measured from the computational speed and power consumption. The measurements utilize a comparison between the processor and the accelerator hardware on a C-program implementation of a cryptographic task performed in different block cipher modes. Using the comprehensive FPGA based test methodology introduced in [4], the testing and verification of operation of the processor implementation has been carried out in various platforms: simulation, FPGA and 22 nm FD-SOI ASIC, to emphasize where the performance is affected the most.

The paper is structured as follows. Section II introduces the structure of the processor and the test environment used for the performance measurements. Section III introduces the test software used for the simulation environment and running the tests on FPGA and ASIC after, which the measurement results and performance of the accelerator are shown. Section IV concludes the paper.

II. Processor system description

A. RISC-V core

Shown in Fig. 1, the host processor for AES-256 hardware accelerator is a 5-stage RV32IMFC RISC-V compliant core based on the A-core, which is an open-source RISC-V core implementation [5]. As memory, the core is coupled with 64 KiB of program memory and 64 KiB of RAM.

The hardware description of the processor was written in Chisel [6]. The chisel generator allows flexible generation of the processor system consisting of combinations of RISC-V core, extensions, and accelerators. The processor in this

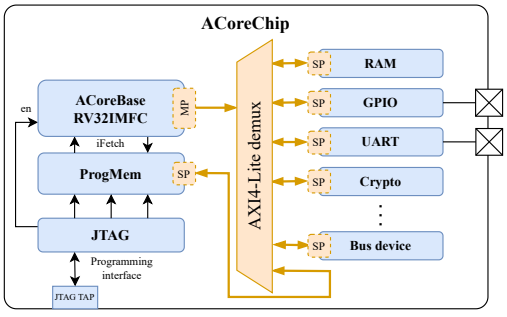

Fig. 1. Structure of the implemented RISC-V core and the integrated AES-256 hardware accelerator.

Standard	Key length, N_k	Block size, N_b	Number of rounds, N_r
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

TABLE I
COMBINATIONS OF N_k, N_b AND N_r OF THE AES ACCORDING TO THE CHOSEN KEY LENGTH.

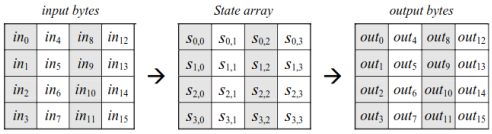

Fig. 2. Example structure of a state in the different phases of the algorithm [7].

work consists of I, M, F, and C extensions, and thus supports both integer and floating point arithmetic. The core controls the integrated hardware accelerators through a AXI4-Lite interface. Furthermore, it supports GPIO and UART interfaces for external interfacing and JTAG for chip programming.

B. AES-256 algorithm

The Advanced Encryption Standard (AES) defines a symmetric block cipher based on the Rijndael algorithm [7]. To date, the standard is used in many commercial applications, such as the IEEE 802.11i standard, TLS and the secure shell network protocol SSH [8]. The AES standard defines three variants of the Rijndael algorithm targeting three different security levels: AES-128, AES-192, and AES-256. The variants differ in the key length N_k and the number of rounding iterations N_r, but all use the same block size N_b. The parameter values of the three variants are collected in Table I.

For instance, AES-256 is implemented by choosing the key length $N_k = 8$ and the number of rounding iterations $N_r = 14$. It requires more iterations and, consequently, achieves lower performance and/or requires a more complex hardware implementation than AES-128 or AES-192. The configurability leaves options for the designer to customize the algorithm and the implementation to the cryptographic task at hand.

In the AES, the 128-bit input is interpreted as a two-dimensional array of bytes, or 8-bit vectors are presented as

$$\begin{aligned} A(x) &= a_7x^7 + \ldots + a_1x^1 + a_0 \\ A &= (a_7,\ a_6,\ a_5,\ a_4,\ a_3,\ a_2,\ a_1,\ a_0). \end{aligned} \tag{1}$$

The resulting array, presented in Fig. 2. is called the state.

Each byte in the state represents an element of the finite field $GF(2^8)$.

A finite field is a set for which addition, subtraction, multiplication and inverse operations are defined. In the case of $GF(2^8)$, addition is defined as,

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Fig. 3. Lookup table used in the byte substitution transformation [8].

$$\begin{aligned} C(x) = A(x) + B(x) &= \sum_{i=0}^{7} c_i x^i, \text{where} \\ c_i &\equiv a_i + b_i \pmod 2 \end{aligned} \tag{2}$$

which equals a bit-wise exclusive-or (XOR) of $A(x)$ and $B(x)$. Notably, addition is its own inverse operation, so subtractions are the same as addition. A multiplication is computed with polynomial multiplication of $A(x)$ and $B(x)$ followed by a reduction modulo $P(x)$, an irreducible polynomial of degree n. The following properties hold for multiplication and inverse operation [8]:

$$\begin{aligned} C(x) &= A(x) \times B(x) \bmod P(x) \\ C(x) &= A^{-1}(x) \times A(x) \bmod P(x) = 1 \\ P(x) &= x^8 + x^4 + x^3 + x + 1. \end{aligned} \tag{3}$$

AES is divided into three main functions named as layers: key addition layer, byte substitution layer and the diffusion layer. To produce the output, these layer operations are repeated for N_r rounds.

In the key addition layer, a round key, which is derived from the main key, is added to the state. Due to the use of $GF(2^8)$, this operation is performed with XOR. The key addition is performed once before the actual rounds and as the last step of each round.

The byte substitution introduces confusion. It transforms each byte of the state by computing an inverse in, $GF(2^8)$ followed by an affine transformation (matrix multiplication and an addition of an 8-bit constant vector). As shown in Fig. 3, this operation can be performed by using a 256×8-bit lookup table. The purpose of adding confusion to the system is to obscure the relation of the ciphertext and the key.

The diffusion layers consists of two operations: row shifting and column mixing. These transform operations add diffusion to the system. The purpose of diffusion is to spread the influence of one plaintext symbol among several ciphertext symbols, hiding the statistical properties of plaintext [8]. In the row shifting operation, the row i is cyclically shifted to the right by i bytes; i.e., the first row is not shifted, the second row is shifted by one byte, etc. The column mixing operation mixes the columns of the state by performing a linear matrix multiplication between the column vector and a constant matrix, as

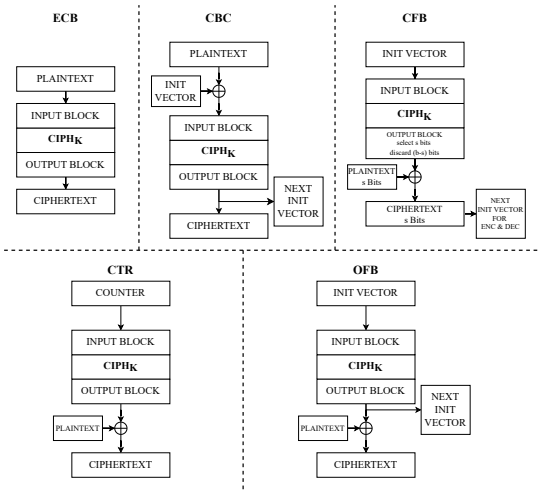

Fig. 4. Generic block diagrams of each cipher mode [9].

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}. \quad (4)$$

The values for the constant transformation matrix are chosen to ease the computation of the column mixing. Addition is again performed with XOR, and multiplications with small elements (02 and 03) can be performed with shifts and XORs.

C. Block cipher modes of operation

The block diagram for the encryption operation in each cipher block mode is shown in Fig. 4. As AES operates only on 16-byte data blocks, it is necessary to use a so-called mode of operation in order to encrypt/decrypt longer messages. A set of modes of operations are defined in [9]. They can be used with AES to further enhance the confidentiality of the AES operations, however they are not mandatory for the standard. These modes include ECB, CBC, CFB, OFB, and CTR modes.

ECB is the simplest possible mode of operation, where each 16-byte block is encrypted separately. ECB is not recommended to be used in practice because all plaintext blocks with the same value result in the same ciphertext block and, thus, full confidentiality is not achieved. The other modes of operation in [9] create dependencies between the blocks by using results from the previous block and/or counter values, and typically use an initialization vector (IV). The modes differ from each other so that some require only AES encryption even for decrypting long messages (for example, CTR and OFB), some allow parallel computations for block encryptions/decryptions (for example, ECB and CTR) whereas some do not (for example, CBC), etc.

D. AES-256 hardware accelerator

The AES-256 compliant hardware accelerator is depicted in detail in Fig. 5. The accelerator implements AES-256, indicating that the key length is 256 bits ($N_k = 8$, $N_b = 4$ and $N_r = 14$). Additionally, it supports ECB, CBC, CFB, OFB and CTR modes of operation. From these modes, the ECB block serves as the basis of the other modes. Moreover, the accelerator is versatile as the IV and the operation mode can be dynamically changed between the 128-bit data blocks. Thus enabling flexible switching between the AES modes of operation. The enable signal allows the RISC-V core to perform other tasks while waiting for the accelerator to finish. The accelerator is wrapped with an AXI4-Lite interface, which enables the accelerator to be included in the design according to the need of the application and provides a simple memory mapped method for the RISC-V core to perform efficient AES-256 operations. Further improving the controls through the core, the status of the accelerator and the controls can be accessed quickly through the use of the control and status registers on the interface.

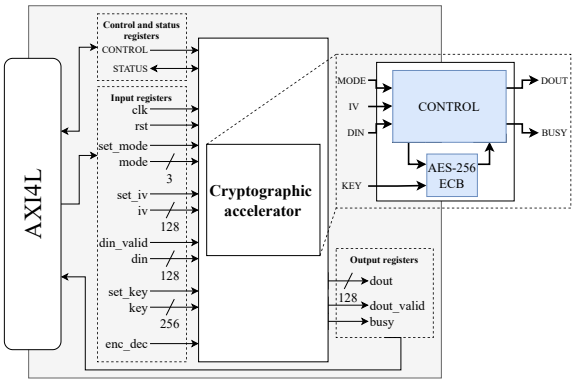

Fig. 5. The external interface and the block diagram of AES-256 cryptographic hardware accelerator.

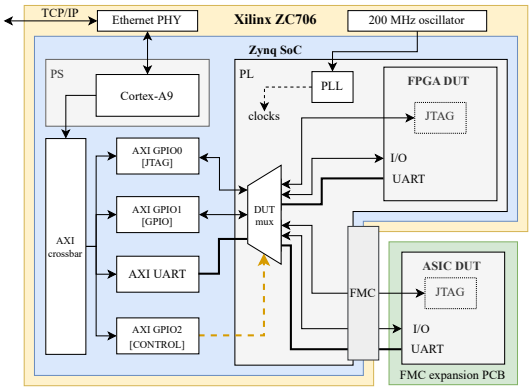

Fig. 6. The main structure of the used FPGA based measurement setup.

III. Verification and Performance Comparison

A. Test and cryptographic software execution environment

Prior to physical implementation of the accelerator on the ASIC, the operation of the system was verified via software and FPGA simulation setup depicted in Fig. 6.

The test environment for the simulation and measurements has been introduced in [4]. It utilizes TheSyDeKick verification framework that is used to control the simulation, FPGA setup and to generate the binary files to be programmed to

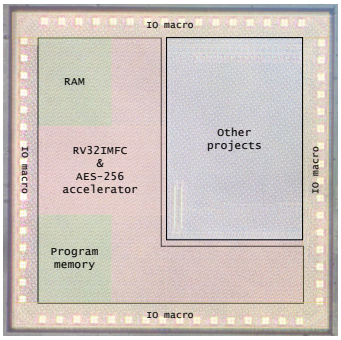

Fig. 7. The physical design of the implemented RISC-V core and the integrated AES-256 cryptographic accelerator.

the design under test. The reference C-program was used to perform encryption and decryption of data to verify that the accelerator can produce the correct output. The FPGA test controller is responsible for directing the program data to the correct design under test (DUT) according to the data received. Additionally, it is used to observe the results and printouts of the core which are directed to the correct TCP/IP-port. The test environment provided various methods to remotely test, verify and compare the hardware performance against the pure software solution.

B. Measurements

The measured processor chip, implemented in 22 nm FD-SOI with the cryptographic accelerator, is shown in Fig. 7. The implemented core is able to achieve 0.628 DMIPS/MHz and 2.07 Coremark/MHz. In the design, the accelerator is synthesized with the RISC-V core and is accessible as a memory mapped device in the AXI4L-bus.

To measure the performance, a self-checking test C-program was implemented to perform eight cipher tasks with 128-bit words in different block cipher modes, described in section II-C. The test involved performing four encrypting operations and four decryption operations on test vectors from [9], on both the RISC-V core using the OpenSSL library, and on the accelerator. To improve the flexibility of measurements, the C-program was parametrized with compilation variables as much as possible, decreasing compilation time and allowing the tester to choose which of the block cipher modes were to be compiled. Hardware counters inside the RISC-V core were used to measure the speed performance of each device during each task. In addition, a trigger signal was used to launch the attached measurement supplies to log the current consumption, which was used for the power consumption measurements. This trigger system simultaneously enabled the calculation of power consumption of power per word ciphered.

Table II shows the resulting clock cycle performance of the RISC-V core and the accelerator hardware when executing the test program. The results do not include the operations for to set the key, the mode, and the IV.

The table shows that the speed performance of the AES-256 succeeds that of the software implementation of the RISC-V core. In terms of operation duration, the cryptographic accelerator performs a ciphering operation by word 82-84 % faster than the software solution. Additionally, the accelerator provides an even performance across the block cipher modes. As the power consumption of the core during the test was measured to be 8 mW at 1 MHz in all block cipher modes with both hardware, the power consumption is relative to the length of the computation. Thus, as similar drop of 82-84 % can be seen in the power consumption of the core when utilizing the cryptographic accelerator.

Method	CBC		CFB		OFB		CTR	
	enc	dec	enc	dec	enc	dec	enc	dec
openSSL	7775	7999	7195	7195	7168	7143	7760	7759
HW-accl.	1311	1311	1311	1311	1311	1311	1311	1311

TABLE II
CLOCK CYCLE COMPARISON OF CYPHER TASKS EXECUTED ON THE RISC-V CORE AND THE AES-256 ACCELERATOR.

The hardware accelerator improves the core performance in terms of power consumption and computational efficiency, with the cost of increased total gate count by ≈ 2 %.

IV. CONCLUSION

In this paper, the integration of an AES-256 hardware accelerator into a custom RISC-V core has been presented. The performance of the accelerator was measured with the introduced FPGA based verification framework and the test C-program. When compared to the performance of a pure software algorithm ran on the RISC-V core, the performance of the accelerator showed a computational speed improvement of 82-84 % and a power consumption lowered by the same amount. This integration can be further improved in the future work by designing a custom RISC-V extension for the accelerator.

REFERENCES

[1] D.-T. Nguyen-Hoang, K.-M. Ma, D.-L. Le, H.-H. Thai, T.-B.-T. Cao, and D.-H. Le, "Implementation of a 32-bit risc-v processor with cryptography accelerators on fpga and asic," in *2022 IEEE Ninth International Conference on Communications and Electronics (ICCE)*, 2022, pp. 219–224.

[2] C. G. de Araujo Gewehr and F. G. Moraes, "Improving the efficiency of cryptography algorithms on resource-constrained embedded systems via risc-v instruction set extensions," in *2023 36th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2023, pp. 1–6.

[3] U. Banerjee, S. Das, and A. P. Chandrakasan, "Accelerating post-quantum cryptography using an energy-efficient tls crypto-processor," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[4] A. Korsman, V. Hirvonen, O. Simola, A. Tarkka, M. Kosunen, and J. Ryynänen, "End-to-end multi-target verification environment for a risc-v microprocessor," in *2023 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2023, pp. 1–4.

[5] A-core, an open-source risc-v processor implementation gitlab repository. [Online]. Available: https://gitlab.com/a-core

[6] Chisel, software-defined hardware. [Online]. Available: https://www.chisel-lang.org/

[7] N. I. of Standards and Technology, "Advanced encryption standard," *NIST FIPS PUB 197*, 2001.

[8] C. Paar and J. Pelzl, *Understanding Cryptography A Textbook for Students and Practitioners*, 1st ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[9] M. J. Dworkin, "Sp 800-38a 2001 edition. recommendation for block cipher modes of operation: Methods and techniques," Gaithersburg, MD, USA, Tech. Rep., 2001.