Indonesian Journal of Electrical Engineering and Computer Science
Vol. 40, No. 2, November 2025, pp. 1167~1176
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v40.i2.pp1167-1176

FPGA-based implementation of an S-Box cryptographic co-processor for high-performance applications

Moulai Khatir Ahmed Nassim[1,2], Ziani Zakarya[2,3]
[1]Department of Electrical Engineering and Electronics, Faculty of Technology, University of Tlemcen, Tlemcen, Algeria
[2]Research Unit for Materials and Renewable Energies (URMER), University of Tlemcen, Tlemcen, Algeria
[3]Department of SNV, Institute of Sciences of University Center of Salhi Ahmed Naama, Naama, Algeria

Article Info

Article history:

Received Feb 13, 2024
Revised Jul 11, 2025
Accepted Oct 15, 2025

Keywords:

Cryptosystems
FPGA
S-Box
VHDL
XILINX

ABSTRACT

The increasing demand for reliable cryptographic operations for securing current systems has given birth to well-advanced and developed hardware solutions, in this paper we consider issues within the traditional symmetric advanced encryption standard (AES) cryptographic system as major challenges. Additionally, problems such as throughput limitations, reliability, and unified key management are also discussed and tackled through appropriate hierarchical transformation techniques. To overcome these challenges, this paper presents the design and field programmable gate array (FPGA)-based implementation of a cryptographic coprocessor optimized for substitution box (S-Box) operation which is considered as a key component in many cryptographic algorithms such as AES. The architecture of the co-processor proposed in this article is based on the advanced characteristics of FPGAs to accelerate the S-Box transformation, improve throughput and reduce latency compared to software implementations. We discussed carefully the design considerations along with resource utilization, speed optimization, and energy efficiency. The obtained experimental results present significant performance improvements, the FPGA-based implementation ensured higher throughput and lower execution time compared to traditional CPU-based methods. We presented in this work the effectiveness of using FPGAs for the acceleration of cryptographic operations in secure applications which will therefore be a robust solution for the next generation of secure systems.

Corresponding Author:

Moulai Khatir Ahmed Nassim
Department of Electrical Engineering and Electronics, Faculty of Technology, University of Tlemcen
BP 230 – 13000 Chetouane –Tlemcen, Algeria
Email: ahmednassim.moulaikhatir@univ-tlemcen.dz

1. INTRODUCTION

Modern embedded systems, particularly those used in internet of things (IoT) and wireless communication, require high levels of security while maintaining efficiency, flexibility, and adaptability. Reconfigurable platforms such as field programmable gate arrays (FPGAs) have become essential in addressing these requirements due to their parallel processing capabilities and customizable architectures. For data security, encryption is used to hide readable information (plaintext) using a specialized algorithm (cipher), ensuring that only authorized parties with the correct key can decode it [1]. The result of this process is ciphertext, a secure form of data. Decryption reverses the process, converting ciphertext back into plaintext using the appropriate decryption algorithm [2].

Initially applied in defense and governmental communications, encryption now plays a critical role in civil applications to protect both data in transit and at rest. Consequently, integrating cryptographic methods into system design has become essential. Among the various encryption algorithms, the advanced encryption standard (AES) is one of the most reliable and widely adopted [3], [4].

Encryption algorithms can be classified into two categories: symmetric and asymmetric. While asymmetric systems offer strong security, they often suffer from high computational complexity and resource consumption [5]. To mitigate these drawbacks, lightweight asymmetric models have been developed to reduce hardware requirements and simplify key management [6]. The rapid expansion of connected devices and IoT ecosystems has exposed systems to more vulnerabilities, highlighting the urgent need for efficient and secure hardware implementations. AES remains a preferred choice for wireless and telecommunication systems due to its structured key management, strong security, and compatibility with efficient hardware architectures [7], [8].

Recent studies have focused on optimizing AES for better performance and real-time compatibility. For example, [9] proposed architectural modifications to improve throughput, while [10], [11] focused on area and resource efficiency in FPGA-based implementations. A key computational challenge in AES is the substitution box (S-Box), responsible for introducing confusion during encryption. While crucial for security, the S-Box is also computationally intensive and can create latency bottlenecks. To address this, FPGA-based cryptographic co-processors have emerged as a promising solution. By offloading intensive tasks such as S-Box computations, these co-processors exploit hardware parallelism to perform multiple operations simultaneously [12], [13].

In this paper, we propose a cryptographic co-processor implemented on a SPARTAN FPGA, optimized for real-time AES encryption. The design leverages pipelining and parallelism to accelerate S-Box computations, reduce latency, and enhance overall throughput. It also supports scalability and adaptation for future cryptographic needs. The rest of the paper is structured as follows: section 2 presents related work; section 3 details the proposed architecture and methodology; section 4 discusses implementation and performance evaluation; and section 5 concludes the paper and outlines potential future work.

2. BACKGROUND AND RELATED WORK

The need for secure and efficient embedded systems has driven the use of cryptographic coprocessors, which offload tasks like encryption, decryption, and key management from the main processor, enhancing performance in real-time environments. A key component in algorithms like AES is the S-Box, which introduces non-linearity. However, its computational complexity often makes it a performance bottleneck, particularly in software implementations [14].

To address this issue, several studies have focused on hardware-based S-Box implementations using FPGAs. Techniques such as pipelining, parallel processing, lookup tables (LUTs), and dynamic reconfiguration have been employed to optimize speed, reduce area, and enhance flexibility [15]. These approaches significantly reduce latency and improve security by executing transformations in a constant time, thus also mitigating timing attacks.

FPGAs are ideal platforms for implementing cryptographic accelerators due to their parallelism, reconfigurability, and efficiency [16]. Prior work includes the development of AES accelerators optimized for throughput and area, with some implementations also supporting inverse transformations for decryption. Despite these efforts, many designs still struggle to balance resource usage, speed, and scalability. Moreover, few architectures offer unified support for both encryption and decryption using shared hardware resources.

Motivation for this work. This work proposes an FPGA-based AES cryptographic coprocessor that performs both encryption and decryption, optimizes the S-Box and Inv-S-Box using precomputed LUTs, utilizes a dynamic control mechanism for mode switching, and efficiently leverages Spartan-6 FPGA resources. The next section presents the detailed methodology of the design and implementation process [17].

3. METHOD

This section presents the methodological framework used to design, implement, and evaluate the proposed FPGA-based cryptographic coprocessor. It includes a description of the system architecture, hardware tools and platforms, experimental setup, and functional validation through simulation.

3.1. System overview

The proposed cryptographic coprocessor is designed to accelerate AES encryption and decryption operations by optimizing the execution of the S-Box, a core non-linear transformation within AES. The coprocessor aims to address performance bottlenecks found in software implementations by leveraging

hardware parallelism and pipelining techniques on an FPGA platform [18]. The system targets high-speed secure applications in embedded and IoT systems, where low latency and resource efficiency are crucial. It supports both encryption and decryption processes and is scalable for integration into more complex security architectures.

3.2. Architecture description

The Figure 1 illustrates the overall architecture of the proposed cryptographic coprocessor and its main logic components. The architecture is designed in a modular manner to facilitate hardware integration and performance optimization. Figure 1(a) shows the internal organization of the cryptographic coprocessor, including the input register and the 16×16 register file, while Figure 1(b) depicts the combinational logic block, consisting of the arithmetic and logic unit (ALU), the shifter, the control logic, and the nonlinear substitution unit:

a) Input register: the input register plays a vital role in receiving data and control signals from external sources. It acts as a temporary storage unit before processing begins, ensuring proper data alignment. This module is synchronized with the clock signal to manage the timing of operations and is reset as necessary to maintain system stability and avoid erroneous data propagation.

b) 16×16 register file: the 16×16 register file serves as the primary memory storage for cryptographic operations. It provides a structured register matrix that facilitates efficient data manipulation. Ra, Rb, and Rd address entries allow selective access to specific registers, ensuring flexibility in data retrieval and storage. This module interacts with both the input register and the combinational logic block, enabling transparent data flow and optimized execution.

c) Combinational logic block: the combinatorial logic block is responsible for executing the main cryptographic transformations, integrating multiple processing units to ensure efficient data manipulation. As shown in the Figure 1(b), this block includes a nonlinear search operation unit, an ALU, and a Shifter, all of which contribute to different aspects of cryptographic processing as follows:
   - Nonlinear lookup operation unit is primarily used for substitution functions, such as S-Box transformations in AES, ensuring nonlinearity and resistance to cryptanalytic attacks.
   - ALU performs essential arithmetic and logic operations, including modular arithmetic crucial for encryption algorithms.
   - Shifter facilitates bitwise transformations, improving data delivery and strengthening cryptographic security.
   - The final output of these units is selected via a multiplexer
   - MUX to determine the processed result based on control signals. This structured design optimizes speed and efficiency, ensuring that the combinational logic block meets the high-performance requirements of cryptographic operations.

The architecture is designed to support parallel execution of S-Box operations and includes dynamic control logic to toggle between encryption and decryption modes.

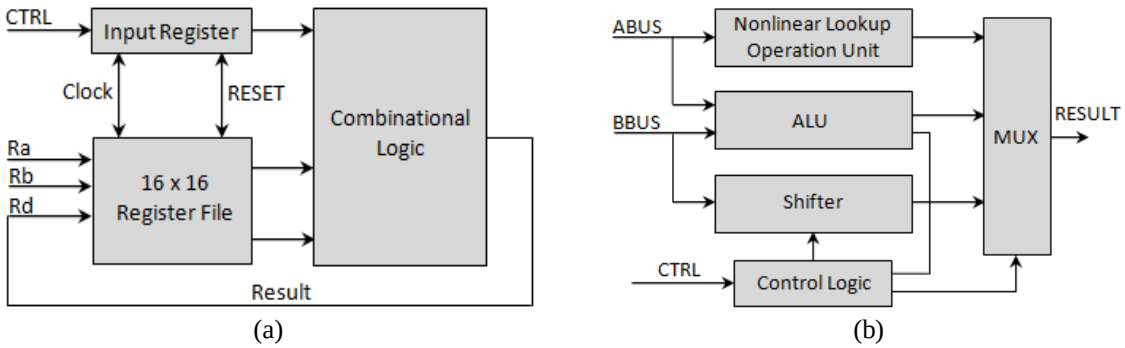

Figure 1. Overall architecture of the proposed cryptographic coprocessor and its core logic components (a) architecture of the cryptographic coprocessor and (b) architecture of the combinational logic block

3.3. FPGA platform and tools

The design and implementation of a cryptographic coprocessor needs a structured approach ensuring efficiency and hardware optimization. Hardware description languages (HDL) such as VHSIC hardware description language (VHDL) and Verilog are new essential tools for the modeling and synthesis of current

digital circuits, allowing very precise control of hardware functionalities. In the context of cryptographic coprocessors, VHDL facilitates the development of key functional units such as arithmetic logic units (ALUs), nonlinear lookup tables (S-Boxes), Shifters, and control logic blocks. By leveraging HDL-based design methodologies, engineers can effectively implement parallelism, pipeline, and resource optimization techniques to improve cryptographic performance. These languages offer engineers the simulation and implementation of complex digital systems [18]. Figure 2 shows the Mimas V2 FPGA development board (Spartan-6 XC6SLX9), which has been used as the implementation platform for the proposed cryptographic coprocessor. It integrates a VGA connector, USB interface, JTAG header, 7-segment display, GPIO expansion connectors, push buttons, DIP switches, LEDs, micro-SD card slot, audio jack, and 512 MB LPDDR memory, providing a versatile environment for hardware prototyping [19].

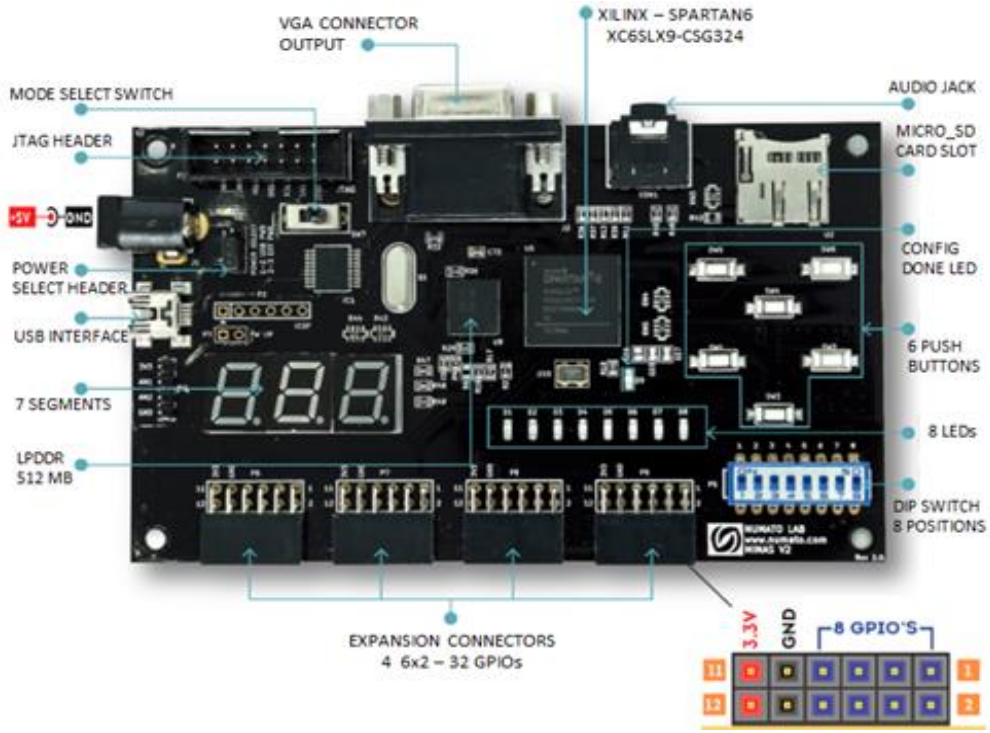

Figure 2. Mimas V2 Spartan-6 FPGA development board [19]

For the implementation, The Mimas V2 Spartan-6 FPGA was chosen for its balance of performance, flexibility, and cost, making it ideal for cryptographic applications. Its rich logic resources, DSP blocks, and reconfigurability support real-time encryption. Using HDL-based design and Xilinx ISE Design Suite, a cryptographic coprocessor was implemented with optimized performance and hardware utilization. Xilinx ISE facilitated coding, debugging, simulation, and resource-efficient synthesis, ensuring a secure and efficient coprocessor suitable for high-security embedded systems [20], [21].

3.4. Experimental setup and performance evaluation

The VHDL program implementing the combinatorial logic unit of our coprocessor is responsible for executing the essential cryptographic operations. It integrates an ALU, a shifter and a nonlinear search unit, with a control logic mechanism that dynamically selects the appropriate calculation.

The entity includes:

– A_BUS (16-bit input): The first data input bus.
– B_BUS (16-bit input): The second data input bus.
– CTRL (4-bit input): The control signal that selects the operation.
– RESULT (16-bit output): The computed result based on selected operations.

This entity acts as a central processing unit within the cryptographic coprocessor. The behavioral architecture consists as described in Figure 3 of three main elements:

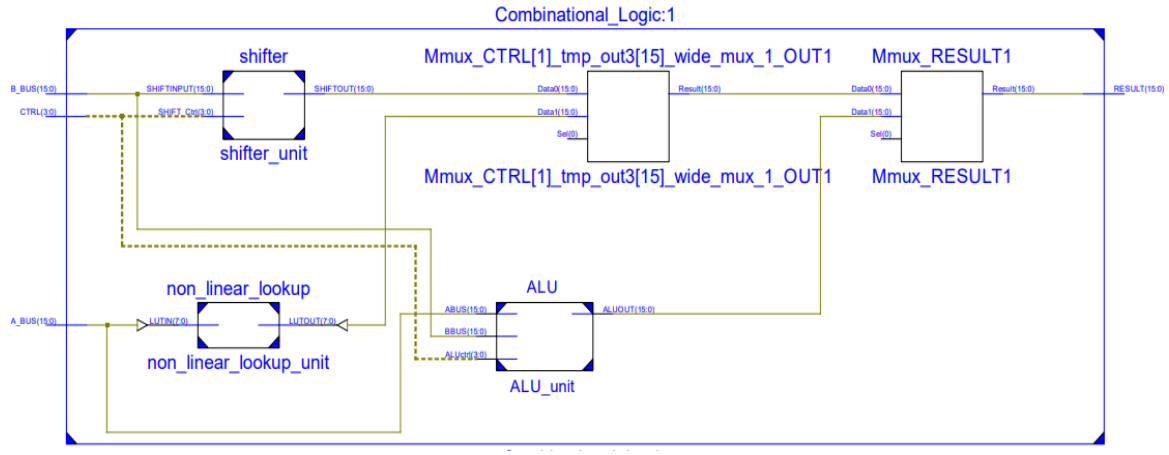

Figure 3. Xilinx block diagram of the combinational logic unit

a) Arithmetic logic unit

The program implements a 16-bit ALU capable of performing fundamental arithmetic and logic operations. The design includes an adder, bit-level logic operations and data manipulation functions, controlled by a 4-bit ALUctrl signal. It integrates addition, subtraction, bitwise operations (AND, OR, XOR, NOT), and data transfer functionalities. The ALU supports addition using an N-bit adder module, as well as subtraction, which is implemented using two's complement representation by inverting BBUS and adding one. It also performs bitwise logical operations, including AND, OR, XOR, and NOT, which are essential for various computational tasks. Additionally, the ALU can execute a move operation, where it simply transfers the value of ABUS to the output without modification [22]. The control logic is implemented using a case statement, which evaluates ALUctrl and selects the corresponding operation to be performed on the input data. The result of the chosen operation is then assigned to the 16-bit output bus (ALUOUT), making the ALU a critical component for digital processing and FPGA-based applications.

b) Shifter

This program defines a shifter module that processes a 16-bit input vector based on a 4-bit control signal. The entity shifter as it's shown in Figure 4 has an input SHIFTINPUT, a control signal SHIFT_Ctrl, and an output SHIFTOUT. The architecture uses a process block to check SHIFT_Ctrl and apply different shift operations:

- "1000" performs an 8-bit right rotation (ROR8).
- "1001" performs a 4-bit right rotation (ROR4).
- "1010" performs an 8-bit left shift (SLL8), filling with zeros.
- Other cases set the output to zero

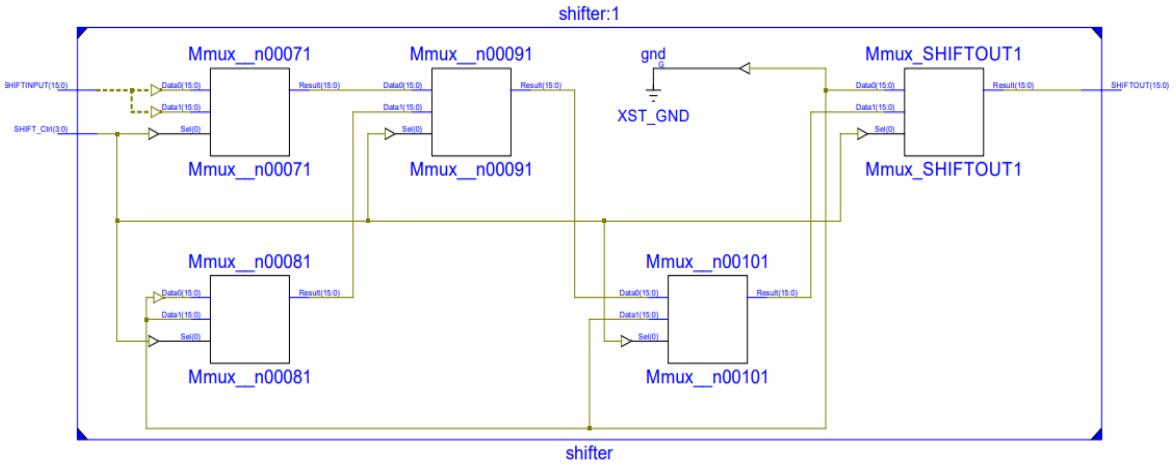

Figure 4. Xilinx internal block diagram of the shifter unit

c) Non_linear_lookup

This VHDL program implements a substitution operation using a lookup table. It takes an 8-bit input and maps it to an 8-bit output using a predefined set of 256 values stored in an array. The mapping follows a non-linear transformation, commonly used in cryptographic applications to introduce security. The input is converted into an integer index, which retrieves the corresponding value from the lookup table. The process operates asynchronously, meaning the output updates as soon as the input changes, without requiring a clock signal. The 256 values in the S-Box shown on Figure 5 are generated using a mathematical transformation that ensures non-linearity, diffusion, and resistance to cryptanalysis.

```
x"63", x"7C", x"77", x"7B", x"F2", x"6B", x"6F", x"C5", x"30", x"01", x"67", x"2B", x"FE",
x"D7", x"AB", x"76", x"CA", x"82", x"C9", x"7D", x"FA", x"59", x"47", x"F0", x"AD", x"D4",
x"A2", x"A6", x"5D", x"85", x"6C", x"32", x"88", x"31", x"5C", x"5A", x"6E", x"52", x"F5",
x"DC", x"9B", x"34", x"8F", x"0D", x"2F", x"28", x"4E", x"81", x"B3", x"F7", x"74", x"92",
x"26", x"36", x"3F", x"F9", x"18", x"23", x"34", x"40", x"1E", x"55", x"73", x"96", x"37",
x"58", x"38", x"F1", x"61", x"D9", x"E4", x"A8", x"A1", x"89", x"0E", x"9C", x"4F", x"A3",
x"72", x"57", x"46", x"45", x"6B", x"9D", x"99", x"66", x"CF", x"C3", x"3C", x"3E", x"11",
x"40", x"B5", x"E7", x"2D", x"5B", x"7D", x"42", x"5E", x"9E", x"6D", x"8A", x"A9", x"82",
x"D6", x"75", x"27", x"DD", x"FD", x"9A", x"6A", x"C1", x"32", x"4D", x"B7", x"60", x"CE",
x"57", x"8D", x"A7", x"8C", x"A6", x"70", x"3D", x"1F", x"39", x"4B", x"59", x"1A", x"A5",
x"81", x"11", x"42", x"6F", x"4A", x"B8", x"E2", x"D5", x"27", x"A4", x"C8", x"F3", x"9F",
x"1B", x"BE", x"6B", x"DB", x"E3", x"29", x"CF", x"2E", x"DA", x"A0", x"95", x"2C", x"F2",
x"AC", x"9A", x"49", x"F8", x"92", x"6A", x"36", x"1B", x"BB", x"5E", x"9F", x"A8", x"73",
x"CE", x"3F", x"0C", x"D3", x"FC", x"11", x"20", x"D1", x"32", x"A2", x"94", x"80", x"E5",
x"5A", x"3A", x"DA", x"29", x"4C", x"B1", x"ED", x"4E", x"55", x"0F", x"AA", x"2D", x"D8",
x"84", x"99", x"67", x"4B", x"B9", x"C5", x"51", x"7E", x"E0", x"87", x"7C", x"B6", x"9B",
x"BD", x"44", x"3C", x"DF", x"DE", x"0A", x"71", x"62", x"CB", x"47", x"8B", x"6C", x"F4",
x"D2", x"C0", x"25", x"C9", x"1E", x"14", x"EA", x"E1", x"2F", x"7A", x"B3", x"5D", x"10",
x"E8", x"AE", x"16", x"7F", x"FF", x"D0", x"3B", x"CC", x"5C", x"68", x"AB", x"19", x"E6",
          x"89", x"76", x"D7", x"65", x"2A", x"79", x"33", x"43", x"90"
```

Figure 5. S-Box lookup table representation

The process typically follows these steps:

– Multiplicative inversion in GF(2^8)

Each byte in the range 0 to 255 is considered an element of the finite field GF(2^8). The corresponding S-Box value is determined by computing its multiplicative inverse within this field, with the exception of 0, which remains unchanged. This transformation guarantees that each value is unique, ensuring a strong cryptographic mapping.

- Affine transformation

After finding the multiplicative inverse, an affine transformation is applied:

$$S(x) = A.x + C$$

Where: x is the 8-bit result from the previous step, A is a fixed invertible matrix over GF(2) and C is a constant vector. When an 8-bit input is provided, the program uses it as an index to access the S-Box, which contains 256 precomputed values. The input byte is replaced with the corresponding value from the table.

For example, if the input is 0x53, looking up the S-Box table returns 0xED, which becomes the new output value. Similarly, if the input is 0x7A, the program will return 0x3F as the output. The multiplicative inversion ensures the non-linearity of the transformation. In AES, each byte is treated as an element of the finite field GF(2^8), and its inverse is determined based on the rules of this field. For example, if the input is 0xB4, its inverse in GF(2^8) is 0x2D. However, to avoid complex calculations in real time, these values are precomputed and stored in the lookup table [23].

Once the inversion is performed, the affine transformation is applied. This involves matrix multiplication followed by an XOR with a constant (0x63). For example, if the inversion step produces 0x2D, applying the affine transformation to this value results in 0x95. This second step adds even more non-linearity and ensures that even a minimal change in the input produces a completely different output.

The VHDL program executes this transformation instantly by storing the results in a LUT. When an FPGA runs this code, it directly accesses the table in a single operation without performing any complex real-time calculations. This significantly optimizes execution speed, making the implementation efficient for real-time cryptographic applications.

Using an LUT also enhances security against certain attacks. For example, in a standard software implementation, the time required to compute the inverse in GF(2^8) may vary depending on the input value,

which could be exploited by a timing attack. Here, since table access occurs in constant time, this risk is eliminated. This design is widely used in cryptographic coprocessors to ensure fast and efficient encryption. On an FPGA, it allows parallel execution of operations, accelerating the processing of data blocks. For example, a full AES encryption process requires multiple S-Box transformations per 128-bit block, and with an LUT, these transformations can be performed simultaneously across multiple processing units within the FPGA [24].

4. RESULTS AND DISCUSSION

To validate the functionality and performance of the cryptographic coprocessor, a testbench simulation was conducted. The waveform in Figure 6 represents the simulation results, showcasing the behavior of key control and data signals over time.

The signals include:

- Clock (clock): A periodic signal that synchronizes operations.
- Reset (reset): Initializes the system.
- Control Signal (ctrl[3:0]): Defines the operation mode.
- Register Addresses (ra[3:0], rb[3:0], rd[3:0]): Select registers for processing.

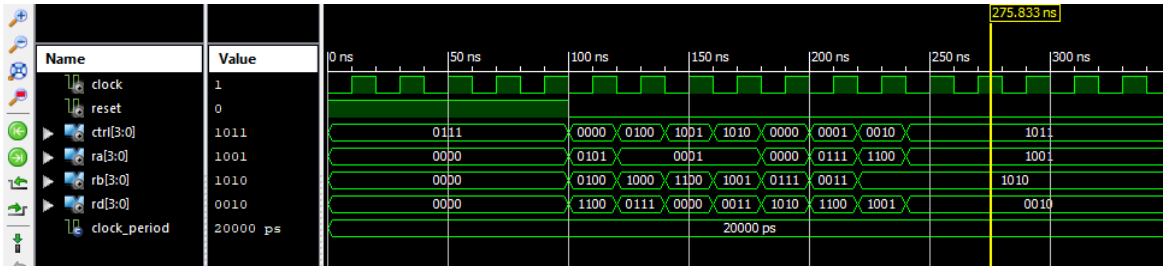

Figure 6. Testbench waveform simulation of the cryptographic co-processor

The simulation was run with a clock period of 20,000 ps (20 ns), aligning with typical FPGA clock frequencies. The timing diagram illustrates how control and data signals evolve over time, confirming correct data flow and synchronization. For instance, at 275.833 ns, the values of ra, rb, and rd indicate successful read/write operations, demonstrating correct register selection and processing. By analyzing these results, we can assess the correct execution of arithmetic operations, S-Box transformations, and data transfers within the FPGA-based cryptographic coprocessor. These simulations play a crucial role in verifying hardware implementation before synthesis and deployment on an FPGA board.

4.1. Hardware implementation of decryption

The FPGA-based cryptographic coprocessor developed in this work is designed to support both encryption and decryption processes. Since AES decryption is structurally similar to encryption but requires inverse transformations, the architecture of the coprocessor has been extended to efficiently handle decryption. The main focus is on implementing inverse transformations while maintaining high performance and resource efficiency on FPGA hardware [25].

4.2. Decryption module architecture

The decryption module is built upon the same hardware structure used for encryption, with additional components for handling inverse transformations. The key elements include:

a) Inverse S-Box lookup table (Inv-S-Box):

The Inv-S-Box is implemented as a precomputed lookup table (LUT) similar to the encryption S-Box but with reversed mappings. Instead of calculating the multiplicative inverse in GF(2^8) in real-time, the LUT approach allows for constant-time substitution.

Example: if encryption maps 0x53 → 0xED, the inverse S-Box ensures 0xED → 0x53. The LUT implementation ensures minimal latency while maintaining cryptographic security.

b) Inverse MixColumns unit:

Since MixColumns in AES encryption spreads the diffusion of bits across a data block, its inverse operation restores the original byte relationships using a different matrix multiplication in GF(2^8). This operation is computationally intensive, but parallelized on the FPGA to minimize processing time. The inverse transformation follows a different matrix:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

c) Inverse key expansion module:

AES decryption requires the round keys to be applied in reverse order compared to encryption. Instead of recomputing round keys, the key expansion unit precomputes and stores them in register memory, allowing for fast retrieval.

d) Arithmetic logic unit (ALU) and control logic:

The ALU, register file, and control logic used for encryption are also utilized for decryption, optimizing resource allocation and minimizing hardware overhead. To differentiate between encryption and decryption operations, a decryption enable flag (DEC_EN) is integrated into the control logic. This flag determines the operational mode of the system, ensuring that the appropriate transformations and key scheduling are applied based on the selected mode.

e) The inverse S-Box (Inv-S-Box):

Must replace the standard S-Box. Instead of using the SBOX lookup table, which is used for encryption, we need a precomputed inverse lookup table (INV_SBOX) that reverses the substitution. By modifying the instruction to: LUTOUT <= INV_SBOX (to_integer(unsigned(LUTIN))); the system will retrieve the correct inverse substitution value, mapping each ciphertext byte back to its original plaintext byte during the InvSubBytes step of AES decryption. This ensures the correct reversal of the non-linear transformation applied during encryption. The Figure 7 show a testbench for our Inverse S-Box VHDL module, which test multiple input values to verify that the correct decryption transformation is applied.

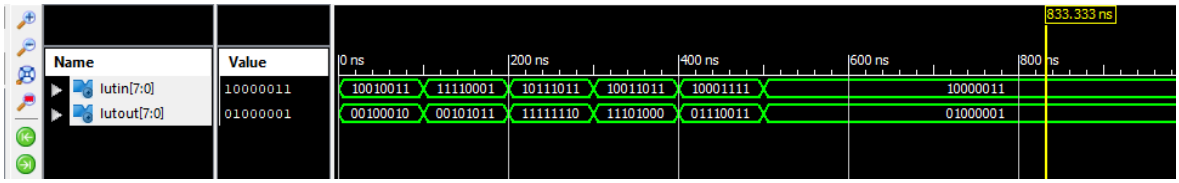

Figure 7. Testbench waveform simulation of decryption transformation

4.3. Performance evaluation

Simulation and synthesis were conducted using Xilinx ISE Design Suite targeting the Mimas V2 Spartan-6 FPGA. The waveform simulations confirmed the functional correctness of the ALU, Shifter, and S-Box modules, including the control logic enabling encryption and decryption modes. The timing diagram demonstrated low-latency data processing with proper synchronization. Key performance metrics include reduced execution time through parallel S-Box computation, efficient resource utilization via a unified encryption/decryption architecture, and scalability for integration into larger cryptographic systems.

4.4. Comparative analysis

Compared to similar works , our architecture achieves:

– Lower latency in S-Box transformation using LUTs,
– Reduced hardware redundancy by sharing ALU and register files across encryption and decryption,
– Improved throughput suitable for high-traffic secure systems.

While previous works have focused on either encryption or area optimization, our design integrates both performance and flexibility. The dual-mode functionality adds versatility not commonly addressed in single-mode accelerators [26].

4.5. Interpretation and implications

These results demonstrate that FPGA-based cryptographic coprocessors can significantly enhance the performance of AES operations in embedded systems. By reducing latency and optimizing resource usage, our implementation is particularly suited for real-time and power-constrained applications such as IoT nodes, secure mobile devices, and industrial controllers. Moreover, the use of precomputed S-Box and Inv-S-Box ensures constant-time operations, which enhances resistance to timing attacks. This contributes to a more secure cryptographic execution pipeline [27].

4.6. Future work

Future work will focus on deploying the design on advanced FPGAs, analyzing power and thermal performance, supporting additional cryptographic algorithms like RSA and ECC, and integrating the coprocessor into full secure systems. This study provides a foundation for optimizing and embedding secure hardware modules in modern platforms.

5. CONCLUSION

In this paper, we presented the design and FPGA implementation of a cryptographic coprocessor optimized for S-Box transformations, a fundamental operation in AES encryption and decryption. Leveraging the parallel processing capabilities of the Spartan-6 FPGA, our architecture significantly reduces execution latency and improves computational throughput compared to traditional software implementations. The proposed coprocessor features a unified design supporting both encryption and decryption modes, with shared resources such as the ALU and control logic, which minimizes hardware overhead. The use of precomputed S-Box and Inv-S-Box LUTs ensures constant-time operation, enhancing security against timing attacks. Simulation results validate the correct behavior of the architecture and demonstrate high-performance cryptographic processing suitable for real-time applications.

Additionally, the system has been designed with scalability in mind, making it adaptable to other cryptographic primitives or more advanced FPGA platforms. The coprocessor is particularly well-suited for secure embedded applications, such as IoT devices, industrial controllers, and mobile systems. Future work will focus on improving power efficiency, extending compatibility to other cryptographic algorithms (e.g., RSA, ECC), and integrating the coprocessor into a complete secure system-on-chip (SoC) architecture.

ACKNOWLEDGEMENTS

The authors would like to thank the Ministry of higher education and scientific research of the Algerian government and the Faculty of Technology–Tlemcen University for providing the funding for this research.

FUNDING INFORMATION

Authors state no funding involved.

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] C. J.Ezeofor and A. G. Ulasi, "Analysis of network data encryption and decryption techniques in communication systems," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 03, no. 12, pp. 17797–17807, Dec. 2014, doi: 10.15680/ijirset.2014.0312008.

[2] V. B. Shaik, "Flexible and cost-effective cryptographic encryption algorithm for non-datafiles," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, pp. 7696–7705, 2022.

[3] A. C. Chen, "Performance comparison of various modes of advanced encryption standard," *arXiv preprint arXiv:2407.09490*, 2024.

[4] Y. Zhang, "Application of optimizing advanced encryption standard algorithms in vehicle controller secure communication systems," *Frontiers in Mechanical Engineering*, vol. 10, 2024, doi: 10.3389/fmech.2024.1407665.

[5] A. Sy, "Sécurisation de données sensibles à l'aide d'autoencodeur convolutionnel profond pour images," *Master's thesis, Université du Québec à Chicoutimi*, 2024.

[6] A. Mansour, K. M. Malik, and N. Kaso, "AMOUN: asymmetric lightweight cryptographic scheme for wireless group communication," *Computer Communications*, vol. 169, pp. 154–167, Mar. 2021, doi: 10.1016/j.comcom.2021.01.019.

[7] A. L. Siridhara *etal.*, "Secure zigbee wireless communication using AES encryption," *International Journal of Advanced Research in Science, Engineering and Technology*, vol. 14, no. 4, pp. 592–598, 2024.

[8] P. Visconti, R. Velazquez, S. Capoccia, and R. de Fazio, "High-performance AES-128 algorithm implementation by FPGA-based SoC for 5G communications," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 5, pp. 4221–4232, oct. 2021, doi : 10.11591/ijece.v11i5.pp4221-4232.

[9] V. T. Nguyen, T. Dang, T. T. Nguyen, and D. Pham, "AES-RV: hardware-efficient RISC-V accelerator with low-latency AES instruction extension for IoT security," *arXiv preprint, arXiv:2505.11880*, 2025. [En ligne]. Disponible: https://arxiv.org/abs/2505.11880.

[10] A. Sideris and M. Dasygenis, "Enhancing the hardware pipelining optimization technique of the SHA-3 via FPGA," *Computation*, vol. 11, no. 8, p. 152, Aug. 2023, doi: 10.3390/computation11080152.
[11] J. T. Grycel and R. J. Walls, "Drab-Locus: an area-efficient AES architecture for hardware accelerator co-location on FPGAS," in *Proceedings - IEEE International Symposium on Circuits and Systems*, Oct. 2020, vol. 2020-Octob, pp. 1–5, doi: 10.1109/iscas45731.2020.9181186.
[12] C. M. Haroldo, N. C. David, M. Madani, and E. B. Bourennane, "FPGA implementation of AES-based on optimized dynamic s-box," in *Proceedings of the International Conference on Security and Cryptography*, 2024, pp. 730–737, doi: 10.5220/0012780300003767.
[13] H. Kim, M.-Kyu Lee, D.-Kyue Kim, S.-Kyoon Chung and K. Chung, "Design and implementation of a crypto processor and its application to security system," *Computational Intelligence and Security*, pp. 1104-1109, 2000, doi: 10.1007/11596981_165.
[14] T. M. Kumar, K. S. Reddy, S. Rinaldi, B. D. Parameshachari, and K. Arunachalam, "A low area high speed FPGA implementation of aes architecture for cryptography application," *Electronics (Switzerland)*, vol. 10, no. 16, p. 2023, Aug. 2021, doi: 10.3390/electronics10162023.
[15] P. Thontirawong and P. Chongstitvatana, "A low-resource AES encryption circuit using dynamic reconfiguration," *Journal of Computer Science and Technology*, 2008.
[16] S. Deshpande, C. Xu, M. Nawan, K. Nawaz, and J. Szefer, "Fast and efficient hardware implementation of HQC," in *Proc. 4th NIST Post-Quantum Cryptography Standardization Conf.*, 2022. https://csrc.nist.gov/csrc/media/Events/fourth-pqc-standardization-conference/documents/papers/fast-and-efficient-hardware-impl-of-hqc-pqc2022.pdf
[17] T. Good and M. Benaissa, "AES on FPGA — from the fastest to the smallest," *Cryptographic Hardware and Embedded Systems – CHES 2005*, 2005, pp. 427–440, doi : 0.1007/11545262_31.
[18] R. Karakchi *et al.*, "Toward a lightweight, scalable, and parallel secure encryption engine," *arXiv preprint arXiv:2506.15070*, 2025.
[19] H. Hamzah, N. Ahmad, M. H. Jabbar, and C. F. Soon, "Optimization AES S-box/Inv S-box using FPGA implementation," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 3–8, pp. 133–136, 2017.
[20] G. P. Saggese, A. Mazzeo, M. Ficco, and L. Romano, "FAC-V: an FPGA-based AES coprocessor for RISC-V," *Journal of Low Power Electronics and Applications*, vol. 12, no. 4, p. 50, Dec. 2022. doi: 10.3390/jlpea12040050.
[21] J.-L. Beuchat, E. Okamoto, et T. Yamazaki, "A low-area unified hardware architecture for the AES and the cryptographic hash function ECHO," *Journal of Cryptographic Engineering*, vol. 1, pp. 101–121, 2011, doi: 10.1007/s13389-011-0009-8.
[22] S. Patel and R. Kumar, "Educational use of FPGA boards in computer engineering," *International Journal of Engineering Education*, vol. 39, no. 4, pp. 1247–1260, 2023 : https://www.ijee.ie.
[23] H. Anwar, M. Daneshtalab, M. Ebrahimi, J. Plosila, and H. Tenhunen, "FPGA implementation of AES-based crypto processor," in *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems*, Dec. 2013, pp. 369–372, doi: 10.1109/ICECS.2013.6815431.
[24] R. Joshi, N. Naik, N. Kashid, S. Waykar, and C. Rangrass, "VHDL implementation of 16 Bit ALU," *International Journal of Engineering Research and Technology*, vol. 2, no. 4, pp. 1–4, 2014, doi: 10.17577/IJERTCONV2IS04050.
[25] S. Samanta, "FPGA implementation of AES encryption and decryption," *Design and Reuse*, 2006.
[26] N. S. S. Srinivas and M. Akramuddin, "FPGA based hardware implementation of AES Rijndael algorithm for encryption and decryption," *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, India, 2016, pp. 1769-1776, doi: 10.1109/ICEEOT.2016.7754990.
[27] P. Kadam and N. D. Parmar, "Combined architecture for AES encryption and decryption using FPGA," *International Conference on Communication Technology (ICCT)*, pp. 14–18, 2015.

BIOGRAPHIES OF AUTHORS

Moulai Khatir Ahmed Nassim received his ingenuity degree in electronics at Faculty of Technology – University of Tlemcen – Algeria, and his Magister and doctorate in microelectronics at Faculty of Technology – University of Tlemcen. Full-time professor of advanced digital electronics (FPGA and VHDL) and electronics graduated program, Electrical Engineering and Electronics Department - Faculty of Technology - University of Tlemcen – Algeria and member of the Research Unit for Materials and Renewable Energies (URMER), University of Tlemcen, BP-119, Tlemcen 13000, Algeria. He can be contacted at email: ahmeddnassim.moulaikhatir@univ-tlemcen.dz.

Ziani Zakarya received his ingenuity degree in physics at Faculty of Science – University of Tlemcen – Algeria, and his Magister and doctorate in energy physics and materials at Faculty of Science – University of Tlemcen. Full-time professor in Department of SNV, Institute of Sciences, University Center of Salhi Ahmed Naama, BP-66, Naama 45000, Algeria. Member of the Laboratory for the Sustainable Management of Natural Resources in Arid and Semi-Arid Zones, University Center Salhi Ahmed, BP-66, Naama 45000, Algeria. He can be contacted at email: ziani@cuniv-naama.dz.