

FPGA-based implementation of a substitution box cryptographic co-processor for high-performance applications

Moulai Khatir Ahmed Nassim^{1,2}, Ziani Zakarya^{2,3}

¹Department of Electrical Engineering and Electronics, Faculty of Technology, University of Tlemcen, Tlemcen, Algeria

²Research Unit for Materials and Renewable Energies (URMER), University of Tlemcen, Tlemcen, Algeria

³Department of Sciences de la Nature et de la Vie, Institute of Sciences of University Center of Salhi Ahmed Naama, Naama, Algeria

Article Info

Article history:

Received Feb 11, 2025

Revised Apr 3, 2025

Accepted Jun 10, 2025

Keywords:

Cryptosystems

Field programmable gate array

Substitution box

VHDL language

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 central processing unit (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.

This is an open access article under the [CC BY-SA](#) license.



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, Shahbazi and Ko [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]–[14].

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 growing demand for secure and efficient encryption in modern embedded systems has led to the development of specialized hardware solutions. Cryptographic coprocessors are dedicated hardware modules that accelerate operations such as encryption, decryption, and key management by offloading these tasks from the main processor. This reduces processing overhead and improves system responsiveness, making them suitable for real-time and resource-constrained environments.

One of the most critical components in encryption algorithms, particularly AES, is the S-Box. It introduces non-linearity and confusion in the transformation of plaintext to ciphertext. However, due to its computational intensity, the S-Box is often a performance bottleneck, especially in software-based systems [15].

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 [9]–[11], [16], [17]. 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. Prior work includes the development of AES accelerators optimized for throughput and area [12]–[14], with some implementations also supporting inverse transformations for decryption [18], [19]. 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 to overcome these limitations, this work proposes an FPGA-based AES cryptographic coprocessor that:

- Implements both encryption and decryption,
- Optimizes S-Box and Inv-S-Box using precomputed LUTs,
- Use a dynamic control mechanism to switch between modes,
- Leverages Spartan-6 FPGA resources efficiently.

The next section presents the detailed methodology of the design and implementation process.

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. 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 architecture of the coprocessor shown in Figure 1 is modular and consists of the following core components:

- 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.
- 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.
- 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 2, this block includes a nonlinear search operation unit, an arithmetic logic unit (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).

- 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.

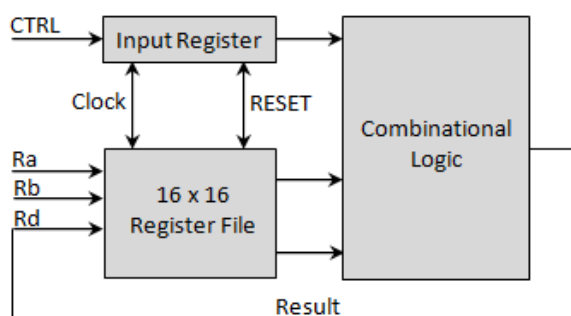


Figure 1. Architecture of the cryptographic coprocessor

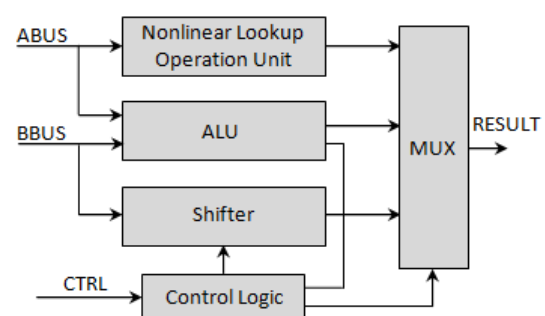


Figure 2. Architecture of the combinational logic block

3.3. Field programmable gate array 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 very high speed integrated circuit 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 ALUs, nonlinear LUTs (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 [20].

For the implementation, the Mimas V2 Spartan-6 FPGA was selected as the target hardware platform. This FPGA shown in Figure 3 offers a balance of performance, flexibility, and cost effectiveness, making it suitable for cryptographic applications. The Spartan-6 architecture provides numerous logic resources, digital signal processing (DSP) blocks and memory units, enabling efficient execution of cryptographic operations. Additionally, its high-speed processing capability and reconfigurability make it an ideal choice for real-time encryption and security applications.

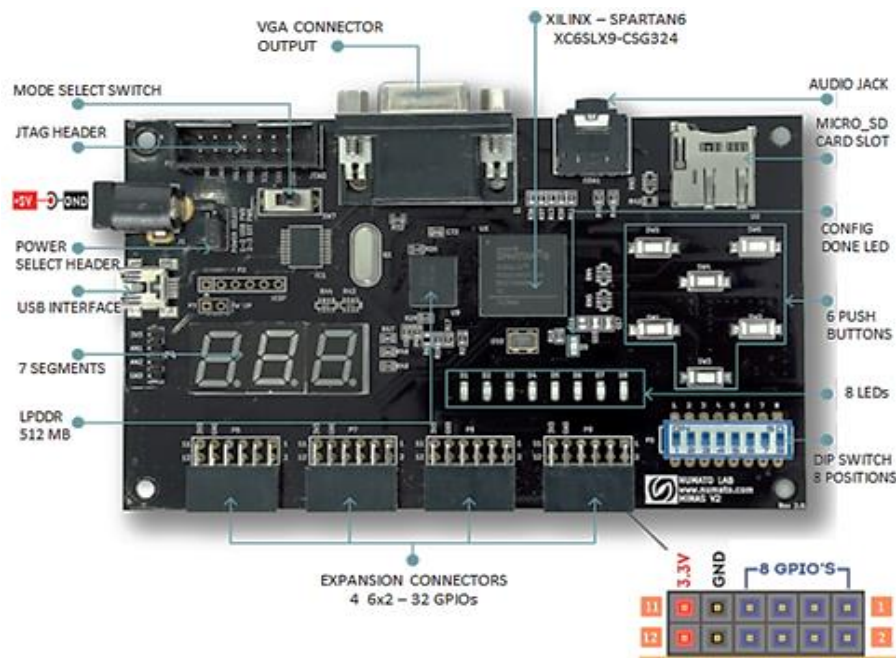


Figure 3. Mimas V2 Spartan-6 FPGA development board [21]

By integrating HDL-based design methodologies with the Mimas V2 Spartan-6 FPGA, this work aims to develop a cryptographic coprocessor that maximizes processing efficiency, minimizes execution time, and optimizes hardware utilization, thereby ensuring a secure and scalable cryptographic solution [22]. The main development environment used to implement our coprocessor is Xilinx ISE Design Suite. It is a software that provides a comprehensive suite of tools for designing, simulating, and implementing VHDL FPGAs, making it an ideal solution for developing hardware-accelerated cryptographic architectures.

Xilinx ISE was used for coding, debugging, and synthesis of the VHDL-based cryptographic coprocessor. The software's integrated simulation environment enabled rigorous testing and validation of key functional units, such as the ALU, S-Box calculations, and control logic blocks. Additionally, the place-and-route (PAR) tool was used to optimize resource allocation and ensure efficient use of the FPGA logic elements [23].

By leveraging Xilinx ISE, the design process was streamlined, enabling efficient verification and implementation of the cryptographic coprocessor on the Mimas V2 Spartan-6 FPGA. This approach ensured a balance between performance, resource utilization and real-time processing capabilities, making it suitable for high-security applications.

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: i) A_BUS (16-bit input): the first data input bus; ii) B_BUS (16-bit input): the second data input bus; iii) CTRL (4-bit input): the control signal that selects the operation; and iv) RESULT (16-bit output): the computed result based on selected operations.

This entity acts as a central processing unit (CPU) within the cryptographic coprocessor. The behavioral architecture consists as described in Figure 4 of three main elements:

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, and 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 [24].

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 5 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.

c. Non linear lookup

This VHDL program implements a substitution operation using a LUT. 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 LUT. 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 6 are generated using a mathematical transformation that ensures non-linearity, diffusion, and resistance to cryptanalysis.

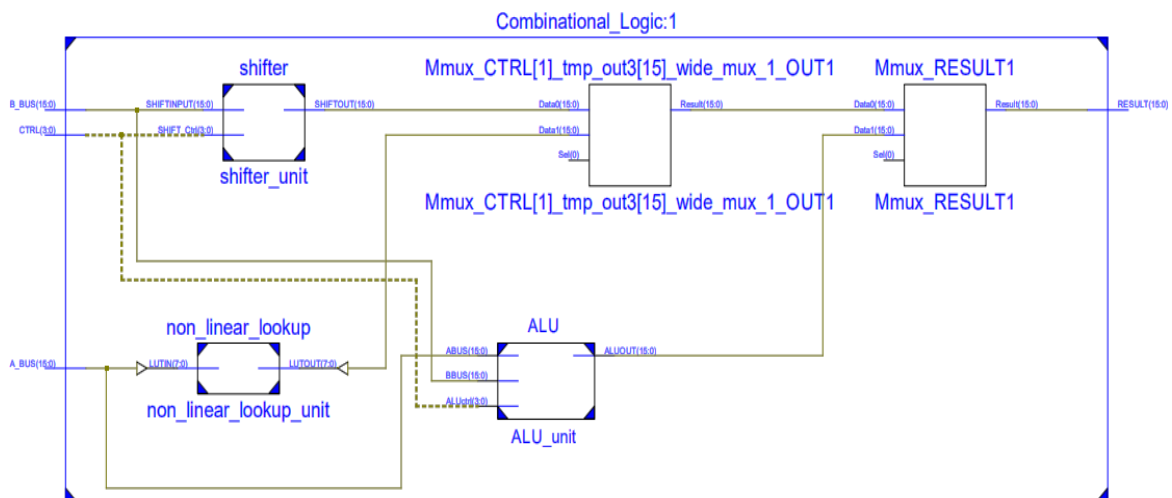


Figure 4. Xilinx block diagram of the combinational logic unit

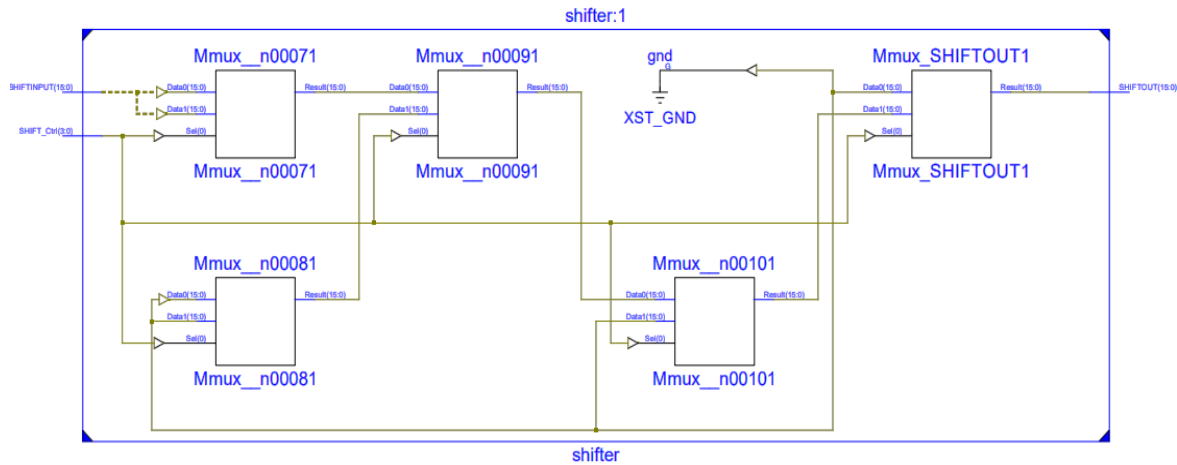


Figure 5. Xilinx internal block diagram of the shifter unit

```

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"E9", 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"1E", 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"9E", 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 6. S-Box LUT 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:

$$S(x) = A \cdot x + C$$

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 0×53 , looking up the S-Box table returns $0 \times ED$, which becomes the new output value. Similarly, if the input is $0 \times 7A$, the program will return $0 \times 3F$ 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 $0 \times B4$, its inverse in $GF(2^8)$ is $0 \times 2D$. However, to avoid complex calculations in real time, these values are precomputed and stored in the LUT [25].

Once the inversion is performed, the affine transformation is applied. This involves matrix multiplication followed by an XOR with a constant (0×63). For example, if the inversion step produces $0 \times 2D$, applying the affine transformation to this value results in 0×95 . 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 [18].

4. RESULTS AND DISCUSSION

To validate the functionality and performance of the cryptographic coprocessor, a testbench simulation was conducted. The waveform in Figure 7 represents the simulation results, showcasing the behavior of key control and data signals over time.

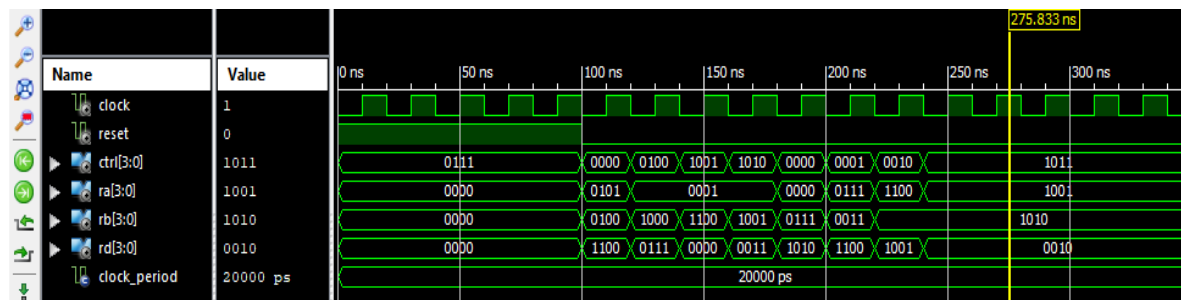


Figure 7. Testbench waveform simulation of the cryptographic co-processor

The signals include: i) clock (clock): a periodic signal that synchronizes operations; ii) reset (reset): initializes the system; iii) control signal (ctrl[3:0]): defines the operation mode; and iv) register addresses (ra[3:0], rb[3:0], rd[3:0]): select registers for processing.

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 [19].

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:

- Inverse S-Box LUT (Inv-S-Box):

The Inv-S-Box is implemented as a precomputed 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 $0 \times 53 \rightarrow 0 \times ED$, the inverse S-Box ensures $0 \times ED \rightarrow 0 \times 53$. The LUT implementation ensures minimal latency while maintaining cryptographic security.

– 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:

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

– 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.

– 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.

The Inverse S-Box (Inv-S-Box) must replace the standard S-Box. Instead of using the LUT, which is used for encryption, we need a precomputed inverse LUT (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 8 show a testbench for our Inverse S-Box VHDL module, which test multiple input values to verify that the correct decryption transformation is applied.

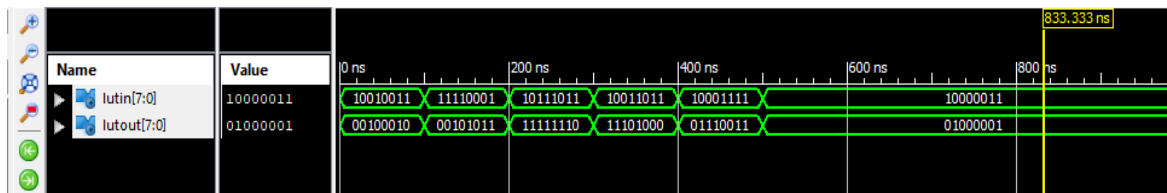


Figure 8. Testbench waveform simulation of decryption transformation

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:

- Execution time reduction due to parallel S-Box computations,
- Efficient resource usage through unified architecture for both encryption and decryption,
- Scalability and adaptability for integration into larger cryptographic systems.

4.4. Comparative analysis

Compared to similar works [9]–[11], 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 [9] or area optimization [10], our design integrates both performance and flexibility. The dual-mode functionality adds versatility not commonly addressed in single-mode accelerators.

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.

4.6. Future work

Future research directions include:

- Implementing the design on more advanced FPGA platforms (e.g., Zynq-7000, Virtex-7),
- Evaluating the power consumption and thermal behavior of the system,
- Extending the architecture to support other cryptographic algorithms (e.g., Rivest-Shamir-Adleman (RSA), elliptic curve cryptography (ECC)),
- Integrating the coprocessor into a complete secure system-on-chip (SoC) or communication system.

This work lays the foundation for further optimization and integration of secure hardware modules in modern embedded 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 and ECC), and integrating the coprocessor into a complete secure SoC architecture.

FUNDING INFORMATION

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.




REFERENCES

- [1] E. C. J and U. A. G, "Analysis of Network Data Encryption & Decryption Techniques in Communication Systems," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 3, no. 12, pp. 17797–17807, 2014, doi: 10.15680/ijirset.2014.0312008.
- [2] V. Shaik and N. K., "Flexible and cost-effective cryptographic encryption algorithm for securing unencrypted database files at rest and in transit," *MethodsX*, vol. 9, Art. no. 101924, 2022, doi: 10.1016/j.mex.2022.101924.
- [3] A. C. H. Chen, "Performance Comparison of Various Modes of Advanced Encryption Standard," *arXiv preprint*, May. 2014, doi: 10.1109/ICSSSES62373.2024.10561385.
- [4] C. Mu, "Application of optimizing advanced encryption standard encryption algorithm in secure communication of vehicle controller area network bus," *Frontiers in Mechanical Engineering*, vol. 10, Jul. 31, 2024, doi: 10.3389/fmech.2024.1407665.
- [5] S. Abib, "Sécurisation de données sensibles à l'aide d'autoencodeur convolutionnel profond pour images," *Doctoral dissertation, 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 *et al.*, "Secure Zigbee Wireless Communication Using AES Encryption," *International Journal of Advanced Research in Science, Engineering and Technology*, vol. 14, no. 4, pp. 592–598, Apr. 2024.
- [8] M. A. Khan, M. Asim, and A. A. Sheikh, "Hardware implementation of AES encryption algorithm on FPGA for 5G communication," *Microprocessors and Microsystems*, vol. 84, 2021.
- [9] K. Shahbazi and S. B. Ko, "High throughput and area-efficient FPGA implementation of AES for high-traffic applications," *IET Computers and Digital Techniques*, vol. 14, no. 6, pp. 344–352, 2020, doi: 10.1049/iet-cdt.2019.0179.
- [10] A. Sideris and M. Dasygenis, "Enhancing the Hardware Pipelining Optimization Technique of the SHA-3 via FPGA," *Computation*, vol. 11, no. 8, pp. 1–15, 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*, 2020, 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*, pp. 730–737, 2024, doi: 10.5220/0012780300003767.
- [13] 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*, vol. 10, no. 16, pp. 1–22, 2021, doi: 10.3390/electronics10162023.
- [14] 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*, vol. 9, no. 3–8, pp. 133–136, 2017.
- [15] M. J. Flynn and W. Luk, "Education and Training for Reconfigurable Systems," *IEEE Transactions on Education*, vol. 49, no. 3, pp. 327–331, Aug. 2006.
- [16] T. Gomes, P. Sousa, M. Silva, M. Ekpanyapong, and S. Pinto, "FAC-V: An FPGA-Based AES Coprocessor for RISC-V," *Journal of Low Power Electronics and Applications*, vol. 12, no. 4, pp. 1–19, Dec. 2022, doi: 10.3390/jlpea12040050.
- [17] H. Kim, Y. Choi, and M. Kim, "Design and implementation of a crypto processor and its application to security system," in *International Technical Conference on Circuits Systems, Computers and Communications (ITC-CSCC)*, 2002, pp. 315–318.
- [18] T. Good and M. Benaissa, "AES on FPGA from the fastest to the smallest," *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th*, 2005, pp. 427–440.
- [19] A. Satoh and S. Morioka, "A compact Rijndael hardware architecture with S-Box optimization," in *Proceedings of ASIACRYPT 2001*, Lecture Notes in Computer Science, vol. 2248, pp. 239–254, doi: 10.1007/3-540-45682-1_15.
- [20] H. Anwar, M. Daneshtalab, M. Ebrahimi, J. Plosila, and H. Tenhunen, "FPGA implementation of AES-based crypto processor," in *Proceedings of the 2010 20th International Conference on Field Programmable Logic and Applications*, 2010, pp. 400–403, doi: 10.1109/ICECS.2010.5615431.
- [21] R. Joshi, N. Naik, N. Kashid, S. Waykar, and C. Rangrass, "VHDL Implementation of 16 Bit ALU," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 4 (ICONET 2014), pp. 1–4, 2014.
- [22] S. Samanta, "FPGA implementation of AES encryption and decryption," *Design & Reuse*, 2008. [Online]. Available: <https://www.design-reuse.com/article/58592-fpga-implementation-of-aes-encryption-and-decryption/#:~:text=FPGA%20Implementation%20of%20AES%20Encryption,and%20Decryption>.
- [23] N. S. S. Srinivas and M. Akramuddin, "FPGA based hardware implementation of AES Rijndael algorithm for Encryption and Decryption," in *Proceedings of the 2nd International Conference on Next Generation Computing and Communication Technologies (NGCCT)*, 2016, pp. 1769–1776, doi: 10.1109/ICEEOT.2016.7754990.
- [24] 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.
- [25] A. Kerbouche, H. Chemali, and M. Ayad, "FPGA-Based implementation of block cipher security using Electronic Codebook mode," *Journal of Electrical Systems*, vol. 20, no. 3, 2024, pp. 8440–8447, doi: 10.52783/jes.7894.

BIOGRAPHIES OF AUTHORS



Moulay 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, Department of Electrical Engineering and Electronics, 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 contact 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 contact at email: ziani@cuniv-naama.dz.