Original Article

Revista elektron, Vol. 8, No. 2, pp. 71-76 (2024)
https://doi.org/10.37537/rev.elektron.8.2.201.2024

Low Area Implementation of the Advanced Encryption Standard (AES) with Counter Mode (CTR) for System-On-Chip (SoC) - Field-Programable Gate Array (FPGA)

Implementación de Bajo Consumo de Área del Estándar de Cifrado Avanzado (AES) en Modo Contador (CTR) para Sistemas en un Chip (SoC) - Matriz de Puertas Programable en Campo (FPGA)

Marco Andrés Ortiz Niño[#1], Arthur Stink Paipilla Arenas[*2], Hernán Paz Penagos[3]

[#]*Department of Electronical Engineering, University of Escuela Columbiana de Ingeniería Julio Garavito AK 45 (Autonorte) #205-59, Bogotá Colombia*

[1] marco.ortiz@escuelaing.edu.co

[2] arthur.paipilla@mail.escuelaing.edu.co

[3] hernan.paz@escuelaing.edu.co

Recibido: 30/09/24; Aceptado: 04/12/24

Abstract*— Cryptography plays a crucial role in protecting information on public networks. The implementation of AES with CTR on Xilinx SoC-FPGA devices, such as Zynq 7000 and Kintex 7, aims to enhance security in IoT devices and embedded systems. The goal is to ensure data confidentiality and availability in connected environments, prioritizing low area usage, low power consumption, and high performance. Implementation was made using a Very High-Speed Integrated Circuit Hardware Description Language (VHDL) on Vivado 2019-2. The results show its area utilization for AES and AES-CTR implementations, with a throughput of 1.8 and 7.67 Gbps for Zynq 7000 and, 2.72 and 11.11 Gbps for Kintex 7; they are also presented for a 128-bits key size and four CTR blocks. VHDL generics can be configured to be 192-bit and 256-bit lengths with different block sizes. Implemented AES-CTR IP showed correct behavior for 128, 192, and 256 key sizes with four CTR blocks. A cipher process with sizes 192 and 256 requires additional cycles that affect the timing performance and hardware utilization.

Keywords*—Advanced Encryption Standard (AES), CTR, Field Programable Gate Array (FPGA), System-On-Chip (SoC), Zynq7000, Xilinx.

Resumen*— La criptografía juega un papel crucial en la protección de la información en redes públicas. La implementación de AES con CTR en dispositivos SoC-FPGA de Xilinx, como Zynq 7000 y Kintex 7, busca mejorar la seguridad en dispositivos IoT y sistemas embebidos. El objetivo es garantizar la confidencialidad y disponibilidad de los datos en entornos conectados, priorizando un bajo uso de área, bajo consumo de energía y alto rendimiento. La implementación se realizó utilizando el lenguaje de descripción de hardware VHDL en Vivado 2019-2. Los resultados muestran la utilización de área para las implementaciones de AES y AES-CTR, con un rendimiento de 1.8 y 7.67 Gbps para Zynq 7000 y de 2.72 y 11.11 Gbps para Kintex 7; también se presentan para una clave de 128 bits y cuatro bloques CTR. Los genéricos en VHDL pueden configurarse para longitudes de 192 bits y 256 bits con diferentes tamaños de bloque. El IP implementado de AES-CTR mostró un comportamiento correcto para tamaños de clave de 128, 192 y 256 bits con cuatro bloques CTR. Un proceso de cifrado con tamaños de 192 y 256 bits requiere ciclos adicionales que afectan el rendimiento temporal y la utilización de hardware.

Palabras clave*—Estándar de Cifrado Avanzado, CTR, Arreglo de compuertas programables de campo, Sistema en Chip (SoC), Zynq 7000, Xilinx.

I. Introduction

In the literature, various approaches are examined to improve data encryption methods that can be used as hardware accelerators to enhance fundamental AES encryption. Some research publications suggest ways to increase performance, while others recommend ways to save space and energy. Many subsequent developments have also utilized other cryptographic methods such as RSA (Rivest, Shamir, and Adleman) and ECC (Elliptic Curve Cryptography) to improve key management security. Currently, post-quantum cryptography (PQC) and new challenges in secure cryptographic algorithms are being discussed to enhance data security and maintain data confidentiality.

A. Post-quantum cryptography

PQC represents the next evolution in cryptographic security, designed to withstand potential attacks from future quantum computers. Unlike traditional methods such as RSA and ECC, which rely on the difficulty of factoring large numbers or solving the discrete logarithm problem, PQC uses mathematical problems that, to date, are inefficient to solve even for quantum computers [1]. This emerging technology is being developed in response to the threat that quantum computers pose to classical cryptography, where the ability to solve complex problems in a reasonable time could make current security systems obsolete.

The adoption of PQC in smartphones presents a significant challenge due to hardware constraints in these

devi ces. PQC algorithms often require more computational resources and storage space compared to RSA or ECC. For example, keys and signatures in PQC schemes tend to be much larger, which could impact processing capacity and storage on mobile devices. This means that manufacturers will need to optimize the implementation of PQC to avoid a negative impact on device performance and user experience. Additionally, updating security protocols to accommodate PQC could be a complex task, requiring a complete overhaul of mobile applications, operating systems, and secure communication infrastructures, all while maintaining interoperability with older devices that still use RSA or ECC [2].

Blockchains are particularly vulnerable in the post-quantum context due to their reliance on cryptographic algorithms for transaction verification and protection against attacks. Digital signatures based on ECC, which currently secure ide-ntities and transactions in many blockchain networks, could easily be compromised by a sufficiently advanced quantum computer [3]. The transition to PQC in this context not only requires the adoption of new digital signature algorithms but also the adaptation of consensus mechanisms and transaction verification processes. Implementing PQC could result in increased processing time and transaction size, which in turn could affect network speed and system scalability. Furthermore, migrating current cryptocurrencies and smart contracts to a PQC-based system poses considerable challenges, including the need to reevaluate the security of digital assets already in circulation.

Despite the theoretical robustness of PQC against quantum attacks, practical implementations of these algorithms may be exposed to other forms of attacks, such as side-channel attacks [4]. These attacks exploit the physical characteristics of an algorithm's implementation (such as power consumption, execution time, or electromagnetic emissions) to extract critical information without directly compromising the cryptographic algorithm [5]. For example, a fault injection attack could force erroneous behavior in the hardware implementing a PQC algorithm, allowing an attacker to recover private keys or perform unauthorized operations. Additionally, devices implementing PQC will need to be carefully designed to resist these attacks, which could increase development costs and complexity.

The transition from ECC/RSA to PQC will not be an instant or uniform process. During this period, many systems will use a combination of classical and post-quantum cryptography, which could lead to new vulnerabilities if not properly managed [6]. Interoperability between old and new systems could create weak points in the security chain. For example, if a system based on ECC/RSA communicates with one based on PQC and the encryption negotiation is not done correctly, critical information could be exposed to attacks [2]. Additionally, systems that are not updated in time will be exposed to quantum attacks, increasing the risk of compromising sensitive data or valuable assets.

PQC algorithms, although secure against quantum attacks, require a greater number of computational resources, which can be a problem in hardware-constrained devices, such as IoT devices and embedded systems. These devices are designed to be extremely efficient in terms of energy consumption and resources, so implementing PQC could overload them, compromising their functionality or significantly reducing their lifespan. The need for more powerful hardware could lead to increased production costs and the need to completely redesign the existing technological infrastructure [3][5].

Finally, the rise of Post-Quantum Cryptography (PQC) signals a shift away from ECC/RSA encryption algorithms. This shift will impact security applications ranging from smartphones to blockchains, marking a profound change in the cryptographic landscape [7].

B. Advanced Encryption Standard Algorithm

AES is a symmetric byte-oriented cipher that performs 10, 12, or 14 rounds of encryption with key-length sizes of 128, 192, or 256-bits. AES components are a key generator, an input (plain text), four transformation modules, and an output (ciphered text). Plain and ciphered bytes have 128 bit-width lengths arranged in a matrix of 4X4 where each element forms a 4-byte word. The transformation modules were SubBytes, ShiftRows, MixColumns, and AddRoundKey. These form a round of ciphers. The key generator outputs a key per round of the process according to the key size.

For confidentiality, AES and its modes of operation are broadly adopted for secure communication protocols used in IoT and IP networks such as IEEE802.15.4, Wi-Fi Protected Access, IPSec, and Secure Sockets Layer; also, AES-CTR can be upgraded with Galois/CTR of operation to add Galois Hash authentication [7] to the security scheme on the device.

The requirements for power consumption, throughput, and security are met more efficiently with hardware implementation than software [8].

Operation modes are required to secure data greater than 128 bit-length to avoid pattern recognition using the same key in each block. These modes allow the division of data into an AES block layout. The arrangement of these blocks depended on the target application. The results are listed in Fig 1.

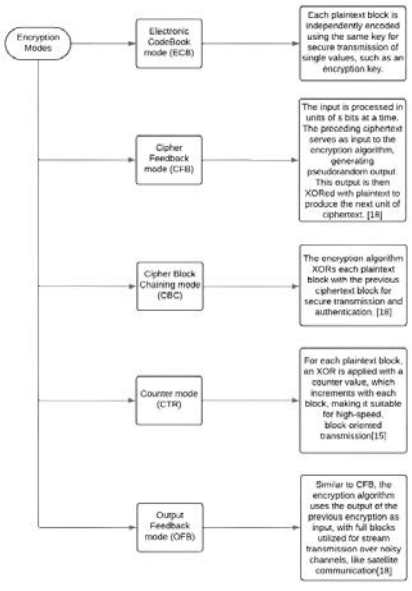

Fig 1. Description of modes of operation [13].

II. STATE OF ART

The efficient implementation of AES in hardware has been a key area of research in recent years due to the need to improve performance and reduce resource usage in embedded systems. FPGA-based architecture has proven to be a viable solution for meeting these requirements. Specifically, optimizing the use of logical resources in FPGAs can minimize the occupied area without compromising system performance. These implementations are essential for applications that require fast and secure encryption in devices with limited resources, such as embedded systems in IoT and telecommunications. For instance, research has demonstrated that improving hardware efficiency by reducing the utilized area enables compact, high-speed implementations [9].

Moreover, the rise of post-quantum cryptography has driven the development of error detection techniques that enhance the robustness of cryptographic systems against faults and attacks. In this context, error detection schemes based on Cyclic Redundancy Check (CRC) have been proposed to ensure the integrity of systems against fault injection attacks and natural hardware faults in FPGAs. These techniques not only improve the reliability of traditional cryptography but are also applied in post-quantum cryptosystems, such as the Niederreiter key generator, to ensure they can withstand failures in deeply embedded systems with resource constraints. Such solutions are critical in sectors like defense and medical electronics, where security and fault detection must be continuously guaranteed.

In addition, advances in the development of lightweight architecture have enabled more efficient implementation of cryptographic S-Boxes, which are key elements in many encryption algorithms. Research in this field has shown that the use of efficient masking techniques can protect S-Boxes from Side-Channel Attacks (SCAs). These architectures, designed to resist both passive and active attacks, strike a balance between hardware complexity and energy efficiency, which is crucial in embedded devices that operate under strict areas and power consumption constraints. These solutions are particularly relevant in lightweight cryptosystems used in mobile devices and low-power systems, where mitigating vulnerabilities is essential without significantly increasing complexity or power consumption.

AES hardware implementations are based on three main strategies: pipelined, non-pipelined, or hybrid. Each has different repercussions regarding throughput, utilization, and power according to the FPGA used. A SoC-FPGA is used in embedded systems, as these require an efficient implementation of hardware and software components within the same device [10], for this, a SoC-FPGA requires components that best match the logic resources that can be used with a wide range of different components. Regarding AES on SoCs for Xilinx, [11] presented a reconfigurable AES for different modes of operation: Cipher Block as Chaining (CBC), Cipher Feedback Mode (CFB), Output Feedback Mode (OFB), counter (CTR), and an extension of the Electronic Codebook (ECB). Their design, based on High- High-level synthesis (HLS), was a pipelined design on a Zynq7000 device with a throughput for AES-CTR of 538.38 Mbps, with a Lookup table (LUT) and Block RAM (BRAM) utilization of 46% and 40%, respectively. Guzman I. et al. [11], present a pipelined AES implementation with ECB and CTR on Virtex 5 [12]. Moreover, Visconti detailed an encryption/decryption implementation using VHDL and a test system scheme [13]. Based on the Zynq UltraScale+ Multiprocessor System-on-Chip (MPSoC), they reported a utilization of 4.76% LUTs and a 28Gbps throughput for AES-

128. In addition, Cowart R. presents an evaluation of a system that integrates the Processing System (PS) with a dual-core ARM Cortex A9 processor and programmable logic (PL) with an AES core for non-pipelined ECB and CBC modes of operation, and a pipelined CTR mode. These have a maximum clock frequency of 125 MHz for the non-pipeline and 325 MHz for the pipeline.

Daoud L. et al. [14] implemented, on a Zynq7000 SoC-FPGA, AES-128 using Vivado HLS, achieving utilization of 1417 LUTs and a throughput of 1,29Gbps. Chen et al. [15] and Sikka et al. [7] showed the results of a pipelined AES design implemented in Vivado HLS on a Kintex 7 FPGA. In addition, Sikka P., et al describe an AES-CTR with LUT utilization of 1,41% for one block and a throughput of 38.05 Gbps and Chen S. et al. [15] at 17.8 Gbps. Finally, Chhabra et al [16]. presented an AES-128 over a Zynq7000 SoC with a testing image processing system.

This project presents the implementation of a core for AES- 128 [17] with four blocks in the CTR of operation (CTR) [18]. It was implemented in the VHDL with a non-pipelined strategy to meet the minimum utilization area of the PL [19]. This straightforward iterative design shows the timing and utilization results based on Vivado synthesis and implementation, respectively [20]. These were obtained from the SoC zynq7000 Zedboard (xc7z020clg484-1) and compared with a Kintex 7 Net FPGA (xc7k325tffg676-1) [21]. This article has the following structure: Section 1 introduces post-quantum cryptography and the AES encryption algorithm; section 2 shows the state of the art of cryptography; Section 3 describes the methodology; Section 4 presents the simulation results with test vectors and clock cycle count, synthesis timing, implementation utilization and compares this work with other AES implementations using Zynq7000 SoC and Kintex 7 Net FPGA. Finally, in section 6, conclusions and future work are presented.

III. METHODOLOGY

The AES design is fully based on the FIPS197 standard [22] and is implemented using an iterative sequential combinational approach, which minimizes the cycle count for key scheduling and encryption processes (as shown in Fig. 2). The sBox, Galois field operations, and Rcon were implemented statically [23]. This design incorporates three input signals to control the AES process: clock signals, along start and reset ports. Additionally, it includes an output that indicates process completion, commonly referred to as ready/busy.

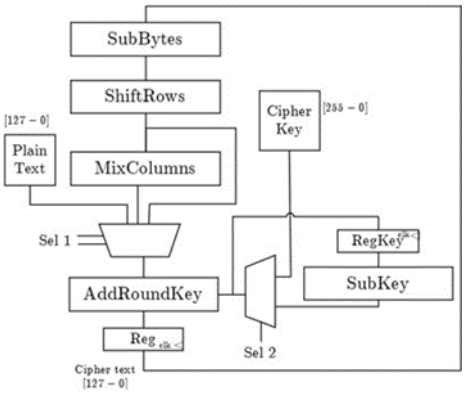

Fig 2. Iterative AES process block.

Then, the CTR is configured using the AES Core previously described and as documented SP-800- 38A [24] with four blocks that can be expanded or reduced through a VHDL Generic. These 4 CTR blocks are implemented in parallel with VHDL to create an AES-CTR core.

The plaintext and ciphertext were both 512 bits in length. Although this document presents results for 128-bit keys, other key sizes can be programmed through a VHDL generic, allowing 192-bit and 256-bit keys. The AES-CTR core also includes a 128-bit Initialization Vector (IV) and an IV-base-step input to control the starting value of the IV and the increment between CTR blocks. The IV- Overflow output is used to detect whether the IV value, after being incremented by the IV-base-step input, needs adjustment to avoid pattern detection in the encrypted data.

The implementation was carried out using VHDL and Vivado 2019.2. The simulation, synthesis, and implementation results are presented for AES-128 and AES-CTR with four blocks. The simulation results were compared to the NIST/FIPS

197 [25] and NIST 800-38A [26] test vectors. Xilinx® Kintex®-7 XC7K325T (xc7k325tffg676-1) and Xilinx® Zynq7000 (xc7z020clg484-1) were synthesized to showcase the results for maximum frequency, power consumption, and resource utilization in the implementation.

IV. Results

A. Simulation Results

The simulation results are obtained from Vivado XSim with a created testbench for an input plain text and key test vectors as seen in [27]. Fig. 3 presents the AES-128 cipher process. This process showed the expected results after 11 clock cycles.

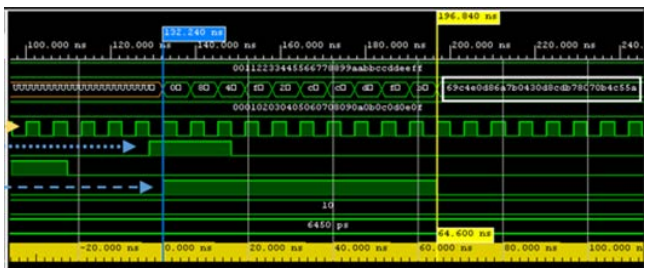

Fig. 3. AES-128 Vivado simulation with FIPS197 test vectors.

For AES-CTR-128, the simulation results are presented in Fig. 4. The AES-CTR encryption process was enhanced by adding an IV, a counter overflow detector, and a base counter value. Since these four blocks are directly derived from AES-128, the same number of clock cycles is required, resulting in a delay of approximately 60.8 ns to display the output.

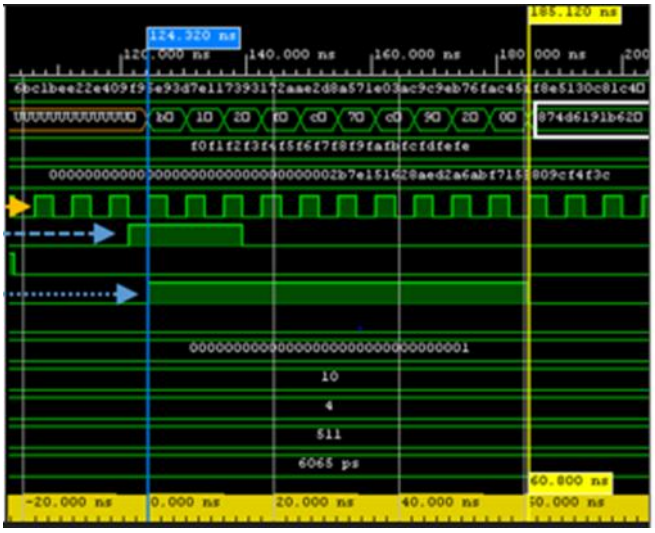

Fig. 4. AES-CTR-128 Vivado simulation with NIST-SP800-38A test vectors.

B. Synthesis results

The performance of both the AES and AES-CTR cores was evaluated using synthesis timing reports. The maximum frequency is determined by the Worst Negative Slack (WNS), following the equation (1).

$$Max.Freq. = \frac{1}{T - WNS} \quad (1)$$

Where T is the reference clock period used during synthesis timing constraints [28]. The throughput can be calculated using two approaches, as shown in (2) and (3):

$$Throughput = \frac{Block\ length\ *\ Freq.Max}{Cycle\ Count} \quad (2)$$

$$Throughput = \frac{output\ bit\ length}{Delay} \quad (3)$$

Two approaches can calculate throughput: one based on the block length and maximum frequency, and the other based on the output bit length and delay. For AES and AES-CTR, with block lengths of 128 and 512, respectively, the calculated throughput reflects significant differences between the two modes.

Regarding power consumption, the dynamic, static, and total power usage for AES-CTR on different devices shows a notable increase in power as the devices scale. For example, the device xc7z020clg484-1 consumes a total power of 1.17W, whereas xc7k325tffg676-1 uses 1.702W. Dynamic power plays a key role in these variations.

The utilization of hardware resources, such as Slice Registers, Slice LUTs, and overall slices, was calculated for the implemented design of AES-CTR across different devices. The xc7z020clg484-1 device shows a Slice Register utilization of 1.03%, while xc7k325tffg676-1 reflects a much lower utilization at 0.26%. Similarly, Slice LUT usage for the devices was 16.49% and 3.68%, respectively. These figures indicate that the implementation efficiency varies based on the device.

In terms of utilization and performance of AES implementations on Xilinx SoC-FPGA and Kintex 7 devices, Table V compares these results with other works. The two devices in this study exhibit the lowest GBPS values compared to those in the literature. There is limited information available on AES implementations using a non-pipelined approach for SoC and Kintex devices. However, this work demonstrates lower resource utilization compared to pipelined implementations, providing more room for additional logic integration.

References		Slitonga et al. [8]	This work	Skka et al. [5]	This work
Devices		zynq7000 zedboard	xc7z020clg484-1	xc7k70t-fbg676	xc7k325tffg676-1
AES Encryption	Blocks	4	4	1	4
Utilization	Slice Register	6	1095 (1.03)	449	1076 (0.26)
	Slice LUT	46	8772 (16.49)	585	7498 (3.68)
	Slices	-	2445 (18.38)	-	2182 (4.28)
Throughput (Gbps)		538.38	7.67	38.05	11.11
Max Freq (MHz)		-	164.9	297.3	238.7

Table I. Comparison AES and AES-CTR

V. Conclusion

The integration of SoC devices in IoT networks improves hardware-software consistency but limits system space and compatibility with other components. Two sequential AES implementation strategies were compared: non-pipelined and pipelined. While pipelined implementations offer higher throughput, they increase area utilization, whereas the proposed method significantly reduces area usage on the Zynq7000 SoC, albeit with slightly lower throughput. This approach is suitable for interconnected device communication applications. Performance can be enhanced by implementing AES in CTR mode, especially for larger packet sizes. Area utilization can further be optimized by using a shared key expansion module. In IoT and industrial applications, a non-pipelined sequential approach optimizes hardware space and improves cycle times at higher clock frequencies. Future research will focus on optimizing CTR utilization and integrating an AXI4 interface for physical testing with the Zynq7000 ARM Cortex-A9 and MicroBlaze processors.

References

[1] J. K. Cheng, E. M. Lim, Y. Y. Krikorian, D. J. Sklar, and V. J. Kong, "A Survey of Encryption Standard and Potential Impact Due to Quantum Computing," The Aerospace Corporation, vol. 1, no. 1, pp. 1-10, 2023.

[2] M. Althoff et al., "CRC-Oriented Error Detection Architectures of Post-Quantum Cryptography Niederreiter Key Generator on FPGA," Proceedings of the International Conference on Field-Programmable Technology (ICFPT), 2022.

[3] P. Werner et al., "Error Detection Schemes Assessed on FPGA for Multipliers in Lattice-Based Key Encapsulation Mechanisms in Post-Quantum Cryptography," IEEE Transactions on Computers, vol. 71, no. 4, pp. 751-764, 2022.

[4] L. Chen, W. Ye, J. He, and X. Ma, "Vulnerability Assessment of Blockchain Using Confrontation Deduction," School of Electronics and Information Engineering, Tongji University, vol. 1, no. 1, pp. 1-10, 2023.

[5] Y. Shen et al., "Lightweight Hardware Architectures for Fault Diagnosis Schemes of Efficiently-Maskable Cryptographic Substitution Boxes," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 10, pp. 4203-4212, 2022

[6] S. Gajbhiye, S. Karmakar, M. Sharma, and S. Sharma, "Paradigm Shift from Classical Cryptography to Quantum Cryptography," CSE-FET, SSGI, SSTC, vol. 1, no. 1, pp. 1-10, 2023.

[7] Circuits and Systems Conference (NorCAS), 2022. DOI: 10.1109/NorCAS57515.2022.9934378

[8] P. Sikka, A. R. Asati and C. Shekhar, "High-throughput field-programable gate array implementation of the advanced encryption standard algorithm for automotive security applications," Journal of Ambient Intelligence and Humanized Computing, 7, 2020

[9] E. M. Benhani, L. Bossuet and A. Aubert, "The Security of ARM TrustZone in a FPGA-Based SoC," IEEE Transactions on Computers, vol. 68, p. 1238–1248, 8 2019

[10] Y. Sovyn, V. Khoma, and M. Podpora, "Comparison of Three CPU-Core Families for IoT Applications in Terms of Security and Performance of AES-GCM," IEEE Internet of Things Journal, vol. 7, p. 339–348, 1 2020.

[11] A. Silitonga, Z. Jiang, N. Khan, and J. Becker, "Reconfigurable Module of Multi-mode AES Cryptographic Algorithms for AP SoCs," in 2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on- Chip (SoC), 2019.

[12] I. C. Guzmán, R. D. Nieto and Á. Bernal, "FPGA implementation of the AES-128 algorithm in non-feedback modes of operation," in DYNA, vol. 83, p. 37–43, 9 2016.

[13] P. Visconti, S. Capoccia, E. Venere, R. Velázquez and R. de Fazio, "10 Clock-Periods Pipelined Implementation of AES-128 Encryption-Decryption Algorithm up to 28 Gbit/s Real Throughput by Xilinx Zynq UltraScale+ MPSoC ZCU102 Platform," in Electronics, vol. 9, p. 1665, 10 2020.

[14] R. Cowart, D. Coe, J. Kulick, and A. Milenković, "An Implementation and Experimental Evaluation of Hardware Accelerated Ciphers in All-Programmable SoCs," in Proceedings of the SouthEast Conference, 2017.

[15] L. Daoud, F. Hussein, and N. Rafla, "Optimization of Advanced Encryption Standard (AES) Using Vivado High-Level Synthesis (HLS)," 2019.

[16] S. Chen, W. Hu, and Z. Li, "High-Performance Data Encryption with AES Implementation on FPGA," in 2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), 2019.

[17] S. Chhabra and K. Lata, "Hardware-Software Co-Simulation of Obfuscated 128-Bit AES Algorithm for Image Processing Applications," in 2018 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), 2018.

[18] R. V. Daemen, "The Design of Rijndael: AES - The Advanced Encryption Standard.," 2002. DOI: 10.1007/978-3-662-04722-4.

[19] H. and R. P. and W. D. Lipmaa, " CTR-Mode Encryption.," 2001.

[20] W. Stallings, "Cryptography and Network Security: Principles and Practice," in Pearson, 2013.

[21] Chegg, "Consider the five block cipher modes of operation shown in Table 6.1. For each mode, consider the case when cipher text block C1 is corrupted. Which plaintext blocks, when decrypted, are corrupted?" [Online]. Available: https://www.chegg.com/homework-help/questions-and-answers/1consider-five-block-cipher-modes-operation-shown- table-61-mode-consider-case-cipher-text--q15451479. [Accessed 21 11 2023].

[22] Course Hero, "1 Consider the five block cipher modes of operation shown in Table 6.docx," 09 20 2019. [Online]. Available: https://www.coursehero.com/file/46728624/1Consider-the-five-block-cipher-modes-of-operation-shown-in-Table-6docx/. [Accessed 21 11 2023].

[23] NIST, FIPS197, 2001.

[24] NIST, SP-800-38A, 2001.

[25] Xilinx, "Vivado Timing - Where can I find the Fmax in the timing report?" 2019.

[26] Lightweight hardware architectures for fault diagnosis schemes of efficiently maskable cryptographic substitution boxes, 2016 IEEE International Conference on Electronics, Circuits and Systems, 2016. DOI: 10.1109/ICECS.2016.7841314.
[27] CRC-oriented error detection architectures of post-quantum cryptography Niederreiter key generator on FPGA, 2022 IEEE Nordic Circuits and Systems Conference (NorCAS), 2022. DOI: 10.1109/NorCAS57515.2022.9934378.
[28] Error Detection Schemes Assessed in FPGA for Multipliers in Lattice-Based Key Encapsulation Mechanisms in Post-quantum cryptography, IEEE Transactions on Emerging Topics in Computing, 2022. DOI: 10.1109/TETC.2022.3217.