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

***Abstract*—Security vulnerabilities persistently emerge through intricate attacks, frequently targeting both essential and non-essential Internet of Things (IoT) devices as well as embedded systems, posing ongoing threats. These connected devices are vulnerable to communication attacks across public networks and require low power consumption, low area utilization, and high throughput. As a security countermeasure, this study aims to include an integrated approach for confidentiality and availability requirements by implementing a non-pipelined Advance Encryption Standard (AES) with the counter mode of operation (CTR), low-area utilization priority for the Xilinx SoC-FPGA Zynq 7000 (xc7z020clg484-1), and Kintex 7 (xc7k325tffg676-1). Implementation was made using a Very High-Speed Integrated Circuit Hardware Description Language (VHDL) on Vivado 2019-2. The results show their 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 128bits 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), counter mode (CTR), field programable gate array (FPGA), System on Chip (SoC), Zynq7000, Xilinx.

I. INTRODUCTION

Governments and organizations formulate standards and protocols outlining measures to prevent potential incidents that impact public or private entities. Typically, these requirements are met through the deployment of defense systems, network devices, and protocols aimed at bolstering the confidentiality, authentication, and availability of data exchanged between the end nodes. These data can be of variable length, reaching frames at the local link level, up to 2000 bytes, and they may contain private information, supervisory, and control commands, among others. 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 (WPA), IPSec, and Secure Sockets Layer (SSL); also, AES-CTR can be upgraded with Galois/Counter mode of operation (GCM) to add Galois Hash authentication [1] to the security scheme on the device.

The requirements for power consumption, throughput, and security are met more efficiently with hardware implementation than software [2]. 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. In particular, a SoC-FPGA is used in embedded systems, as these require an efficient implementation of hardware and software components within the same device [3], 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, [4] 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-Level Synthesis (HLS), was a pipelined design on a Zynq7000 device with 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., present a pipelined AES implementation with ECB and CTR modes on Virtex 5 [5]. Moreover, Visconti detailed an encryption/decryption implementation using VHDL and a test system scheme [6]. 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-pipelined and 325 MHz for the pipelined [7].

Daoud L. et al implemented, on a Zynq7000 SoC-FPGA, AES-128 using Vivado HLS, achieving utilization of 1417 LUTs and a throughput of 1,29Gbps [8]. Chen et al. and Sikka et al. showed the results of a pipelined AES design implemented in Vivado HLS on a Kintex 7 FPGA [2], [9]. 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 [2] and Chen S.et al at 17.8 Gbps [9]. Finally, Chhabra et al. presented an AES-128 over a Zynq7000 SoC with a testing image processing system [10].

This project presents the implementation of a core for AES-128 [11] with four blocks in the counter mode of operation (CTR) [12]. It was implemented in the VHDL with a non-pipelined strategy to meet the minimum utilization area of the PL. This straightforward iterative design shows the timing and utilization results based on Vivado synthesis and implementation, respectively. These were obtained from the SoC zynq7000 Zedboard (xc7z020clg484-1) and compared

with a Kintex 7 Net FPGA (xc7k325tffg676-1). This article has the following structure: Section 2 presents the AES background; section 3 describes the AES implementation design; section 4 shows the results of the simulation with test vectors and clock cycle count, synthesis timing, and implementation utilization; section 5 compares this work with other AES implementations using Zynq7000 SoC and Kintex 7. Finally, in section 6, conclusions and future work are presented.

II. AES Background

The Rijndael algorithm, developed by Vincent Rijmen and Joan Daemen, became the Advanced Encryption Standard (AES) in November 2001 after a selection process initiated in January 1997 by the National Institute of Standards and Technology (NIST). 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). Plaint 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.

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 Table I.

TABLE I. Description of Modes of Operation [13]

Mode	Description	Typical Application
Electronic CodeBook mode (ECB)	Every block of plaintext bits is encoded autonomously utilizing identical key.	Secure transmission of single values (e.g., an encryption key).
Cipher Block Chaining mode (CBC)	The encryption algorithm takes the XOR operation between the subsequent block of plaintext and the preceding block of ciphertext as input. [14].	General-purpose block-oriented transmission and authentication.
Cipher Feedback mode (CFB)	The input is processed in units of s bits at a time. The preceding ciphertext serves as input to the encryption algorithm, generating pseudorandom output. This output is then XORed with plaintext to produce the next unit of ciphertext. [14].	General-purpose stream-oriented transmission and authentication.
Output Feedback mode (OFB)	Similar to the CFB, the input to the encryption algorithm is the output from the previous encryption, and complete blocks are utilized. [14].	Stream-oriented transmission over noisy channels (e.g., satellite communication).
Counter mode (CTR)	For each block of plaintext, an XOR operation is performed with an encrypted counter. The counter is incremented for each successive block [15].	General-purpose block-oriented transmission and Useful for high-speed requirements.

III. AES Implementation Design

AES design is based entirely on standard FIPS197 [16] and is designed using an iterative sequential combinational method that allows a minimum cycle count for key scheduling and encryption processes (Fig. 1). sBox, Galois, and Rcon were statically implemented. This implementation used three inputs to control the AES process. These are clock signals and the start and reset ports. In addition, it has an extra output that allows the monitoring of process completion. This is known as ready/busy.

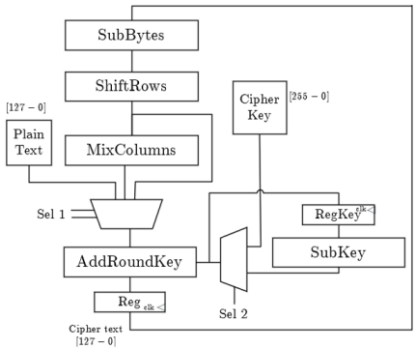

Fig. 1. Iterative AES process block.

Then, the Counter mode (CTR) is configured using the AES Core previously described and as documented SP-800-38A [17] with four blocks that can be expanded or reduced through a VHDL Generic. A single control block, named the CTR block, is shown in Fig. 2.

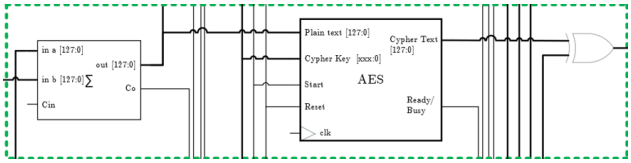

Fig. 2. AES-CTR Block.

Four of these CTR-Blocks are parallelly implemented in VHDL to create an AES-CTR Core.

Plain text and cipher text were 512 bits in length. Although this document shows the results for 128 keys, other key sizes can be programmed through VHDL generic for 192 and 256 bits. The AES-CTR Core also has a 128-bit initialization vector and an IV-base-step input to control the start value for the initialization vector and the step between CTR-Blocks. The output IV-Overflow is used to detect if the initialization vector value, after incrementation by IV-Base-step input, requires an IV-variation to avoid pattern detection in the encrypted data.

IV. Results

The implementation was performed using VHDL and Vivado 2019-2. The simulation, synthesis, and implementation are presented for AES-128 and AES-CTR with four blocks. The simulation results were compared with the NIST/FIPS 197 [16] and NIST800-38A [17] test vectors. Xilinx® Kintex®-7 XC7K325T (xc7k325tffg676-1) and Xilinx® Zynq7000 (xc7z020clg484-1) were synthesized to show the results for maximum frequency, power consumption, and implementation for reporting their utilization.

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 [11]. 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 result of the simulation is shown in Fig. 4. An initialization vector, counter overflow detector, and base counter value were added to the AES-CTR encryption process. Because these four blocks are directly based on AES-128, the same clock cycles are required to show the result with a delay of approximately 60,8nS.

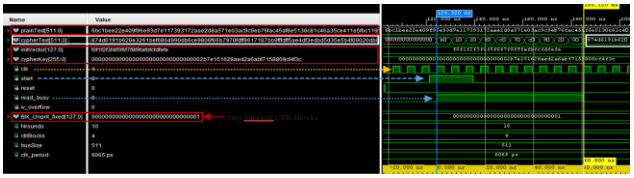

Fig. 4. AES-CTR-128 Vivado simulation with NIST-SP800-38A test vectors.

B. *Synthesis results*

The performance of the AES and AES-CTR cores was based on synthesis timing reports. The Worst Negative Slack (WNS) allows the determination of the maximum frequency, as presented in (1) [18]:

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

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

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

$$Throughput = (output\ bit\ length.)/Delay \quad (3)$$

For a Block length of 128 for AES and 512 for AES-CTR, the resulting values are presented in Table II.

TABLE II. SYNTHESIS TIMING PERFORMANCE

Device	AES Encryption		Throughput (Gbps)	Max Freq (MHz)
	Mode	Blocks		
xc7z020clg484-1	AES	N/A	1.8	155.06
xc7k325tffg676-1	AES	N/A	2.73	237.37
xc7z020clg484-1	AES-CTR	4	7.67	164.9
xc7k325tffg676-1	AES-CTR	4	11.11	238.7

The power consumption results of the synthesized design are presented in Table III.

TABLE III. SYNTHESIS REPORT OF POWER CONSUMPTION

Device	AES Encryption		Power (W)		
	Mode	Blocks	Dynamic	Static	Total
xc7z020clg484-1	AES	N/A	0.25	0.108	0.358
xc7k325tffg676-1	AES	N/A	0.319	0.16	0.478
xc7z020clg484-1	AES-CTR	4	1.047	0.123	1.17
xc7k325tffg676-1	AES-CTR	4	1.534	0.168	1.702

C. *Implemented results*

To establish the utilization report of slices, the implemented design is carried out using a hierarchical partition for AES core analysis, which is instantiated as Out of Context (OOC). This methodology excludes IO but allows for a more accurate estimation of slice placement on the FPGA. Table IV presents the results.

TABLE IV. IMPLEMENTED UTILIZATION REPORT

Device	AES Encryption		Utilization		
	Mode	Blocks	Slice Register	Slice LUT	Slices
xc7z020clg484-1	AES	N/A	269 (0.25%)	1338 (2.52%)	380 (2.86%)
xc7k325tffg676-1	AES	N/A	269 (0.07%)	1251 (0.61%)	348 (0.68%)
xc7z020clg484-1	AES-CTR	4	1095 (1.03%)	8772 (16.49%)	2445 (18.38%)
xc7k325tffg676-1	AES-CTR	4	1076 (0.26%)	7498 (3.68%)	2182 (4.28%)

The emergence of Post-Quantum Cryptography (PQC) as a replacement for ECC/RSA will have a profound impact on security applications spanning from smartphones to blockchains [19-21].

V. COMPARISON OF RESULTS

Regarding the utilization and performance of AES implementations on the Xilinx SoC-FPGA and Kintex 7, other works are presented in Table V, where it can be seen that the two devices in the present work present the lowest GBPS values compared to the other studies found in the literature. Limited information on AES implementations using a non-pipelined method for SoC and Kintex devices is found. This work presents a lower utilization than pipelined implementations, allowing more capacity for logic integration.

TABLE V. COMPARISON OF AES AND AES-CTR FOR SOC-FPGA AND KINTEX7

Reference	Device	AES Encryption		Utilization			Throughput (Gbps)	Max Freq (MHz)
		Mode	Blocks	Slice Register	Slice LUT	Slices		
Daoud et al. [8]	xc7z020-1clg484c	AES	N/A	830	1417	431	1.29	192

Reference	Device	AES Encryption		Utilization			Throughput (Gbps)	Max Freq (MHz)
		Mode	Blocks	Slice Register	Slice LUT	Slices		
This work	xc7z020clg484-1	AES	N/A	269 (0.25%)	1338(2.52%)	380 (2.86%)	1.8	155.06
Visconti et al. [6]	xczu9eg-2ffvb1156e	AES	N/A	0.71%	4.76%	-	28	220
Chen et al. [9]	xc7k325tffg676-2l	AES	N/A	8311	19312	-	17.8	139
This work	xc7k325tffg676-1	AES	N/A	269 (0.07%)	1251 (0.61%)	348 (0.68%)	2.73	237.37
Silitonga et al. [4]	zynq7000 zedboard	AES-CTR	4	6%	46%	-	538.38	-
This work	xc7z020clg484-1	AES-CTR	4	1095 (1.03%)	8772 (16.49%)	2445 (18.38%)	7.67	164.9
Sikka et al. [2]	xc7k70t-fbg676	AES-CTR	1	449	585	-	38.05	297.3
This work	xc7k325tffg676-1	AES-CTR	4	1076 (0.26%)	7498 (3.68)	2182 (4.28%)	11.11	238.7

VI. Conclusion

Interconnected devices like IoT nodes and gateways commonly use SoC devices to reduce dependency on multiple manufacturers and enhance consistency by integrating hardware and software. This integration, however, reduces processor system space and limits integration with different hardware components. Strategies for sequential AES implementation include non-pipelined and pipelined approaches, with the latter offering higher throughput but increased area utilization. Despite the lack of explicit documentation for non-pipelined devices like Zynq7000 and Kintex7, the presented method minimizes area utilization compared to other AES-128 implementations on the Zynq7000 SoC-FPGA. While it has lower throughput, this approach supports various applications for interconnected device communication. Higher throughput is achieved by implementing the CTR mode, improving performance over large packet sizes. The area utilization of AES-128-CTR can be optimized by a common key expansion module for all blocks. The implemented utilization results are deemed more reliable than synthesized results. In applications like IoT and industrial networks, SoC devices' adaptability drives the preference for a combinational sequential implementation, particularly non-pipelined, optimizing hardware space and allowing for enhanced cycle times at higher clock frequencies.

In future work, researchers will improve the utilization of CTR and the integration of an AXI4 interface to perform physical tests with the Zynq7000 ARM Cortex-A9 dual-core processor and Microblaze soft processor.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

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

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

[14] 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].

[15] 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].

[16] NIST, FIPS197, 2001.

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

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

[19] 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

[20] 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

[21] 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.2022.3217006.