[1]Abdelaziz Kerbouche
[2] Hamimi Chemali
[3*] Mouloud Ayad

FPGA-Based implementation of Block Cipher Security Using Electronic Codebook Mode

Journal of Electrical Systems

Abstract: - This paper presents FPGA implementation of the Advanced Encryption Standard using Electronic Codebook Mode. Block ciphers, a cornerstone of cryptographic systems, are employed to encrypt fixed-size blocks of data, ensuring confidentiality in communication. The FPGA platform is utilized to optimize the encryption process, leveraging its parallel processing and high-speed capabilities. The design incorporates the key components of the block cipher algorithm, ensuring efficient execution of encryption process. The implementation is evaluated for its performance, highlighting its potential for real-time secure communication in constrained environments. Two implementation methods are employed for the ECB mode: parallel and pipeline. The parallel method achieved a maximum frequency of 264 MHz, while the pipeline system reached a maximum frequency of 189 MHz.

Keywords: Advanced Encryption Standard (AES), Electronic Codebook (ECB), Encryption algorithm, Field Programmable Gate Array (FPGA), Embedded systems.

I. INTRODUCTION

In secure communication systems, cryptographic techniques are indispensable for protecting sensitive data from unauthorized access. Among these techniques, block ciphers stand out as robust algorithms that encrypt fixed-size blocks of plaintext into ciphertext, ensuring confidentiality and integrity during data transmission. The Electronic Codebook (ECB) mode is a basic operational mode for block ciphers, where each plaintext block is encrypted independently using the same key. This independence makes ECB straightforward to implement and capable of achieving high-speed encryption, especially in hardware systems. Field-Programmable Gate Arrays (FPGAs) offer a compelling platform for implementing cryptographic algorithms like block ciphers. Their reconfigurable nature allows for flexible designs, while their inherent parallelism and hardware-level processing enable high-speed and low-latency operations. These features make FPGAs particularly suited for real-time and resource-constrained applications requiring secure data encryption. In the literature, several works based on FPGA implementation are presented [1-4].

In the past few years, there has been a significant increase in researches on the cipher blocks implementation using different modes. For example, in [5] Wu et al presented a crypto engine in ECB mode for hard-disk data encryption, with a stable speed adapted to read and write throughput of SATA interface. In [6] Al-wattar designed an image encryption approach using AES-ECB mode, and based on DNA alteration, the author presented also some limitation of the ECB mode especially for image encryption. Authors in [7] designed an AES-ECB and AES-CTR (counter mode) on GPU (Graphics Processing Unit). This proposed design achieved high speed due to parallel processing, mainly the table-based approach. Based on Intel core i7 and Intel core 2, authors in [8] present a fast implementation of AES-CTR. The software implementation of this approach (FACE) uses the bitsliced method. The obtained results show 15%-20% more efficiency compared to other works. In another filed of application, a satellite image encryption approach is presented in [9] using CTR mode for AES algorithm. The authors proposed a modification in the counter mode, aiming the enhancement of performance, along with the high level of security, through combining GEFFE generator with AES.

As described above, block cipher algorithms are commonly used in different operation mode, in this work we propose the implementation of a cipher block algorithm, namely the AES, using ECB mode. The aim is providing a continues mode of data encryption for communication purposes using AES-based cipher block. The whole of the paper is presented as follow: In section 2, an overview of functioning mode of cipher blocks is presented. The

[1] PhD Student, L.C.C.N.S Laboratory, Department of Electronics, University of Ferhat Abbas Sétif 1, Algeria. Email: abdelaziz.kerbouche@univ-setif.dz
[2] Professor, Department of Electronics, University of Ferhat Abbas Sétif 1, Algeria. Email: hamimi.chemali@univ-setif.dz
[3] * Professor, Department of Electronics, University of Ferhat Abbas Sétif 1, Algeria. Email: m.ayad@univ-setif.dz

description of the proposed approach and discussion of results are presented in Section 3. Finally, the conclusion is given.

II. Block Ciphers Mode Of Operation

In order to encrypt Information, numerous techniques can be used to transform plaintext into ciphertext, these techniques can be categorized as stream cipher and block cipher. With stream ciphers, data is encrypted and stored one byte at a time, which allows a high-speed encryption, however, it also requires a complex hardware or CPU (Central Processing Unit) infrastructure [10]. This technique uses usually symmetric encryption, where a single key is used for encryption and decryption. One of the challenges of stream ciphers is randomization, where the next byte is unknown until it arrives. this challenge is especially highlighted if multiple identical bytes are input into the stream, resulting to identical bytes on the encrypted side. That's why stream cipher uses an initialization vector (IV), that is added to the stream cipher to introduce some randomization to the encryption process.

The second technique is the block cipher; as the name implies, a block cipher is encrypting a fixed length block of information at a time. so instead of taking a single byte, it takes a block of bytes and encrypt it at one time, where the size of that block has a fix length.

There are several encryption modes of information, each mode has a specific processing method. The basic principle is dividing large data into smaller fixed length blocks before passing through the encryption process.

One of the common modes of operation is the Electronic Codebook (ECB). It consists on performing exactly the same encryption for every block in the series separately, using the same key. The outputs are then concatenated together to get the encrypted data. Based on the definition of this mode, the presence of identical plaintexts, results to identical ciphertext, which makes ECB not adapted to many cases of use such as image encryption. The ECB encryption mode is presented in Fig. 1.

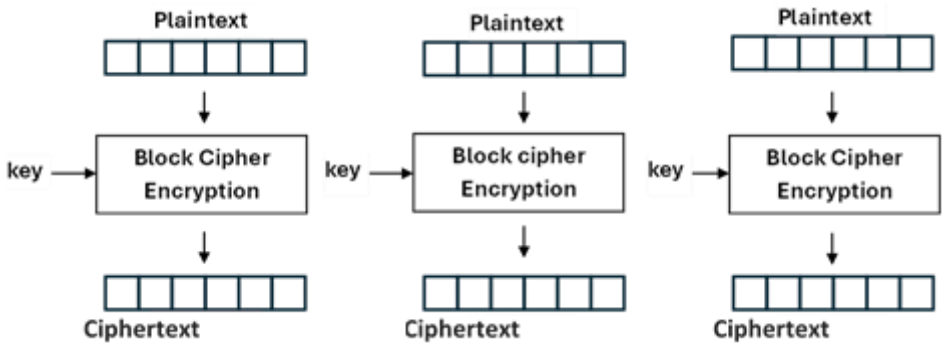

Fig. 1. The Electronic Codebook mode.

Another common operation mode is Cipher Block Chaining (CBC). This mode adds some randomization, to the output, where each block is XORed with the previous ciphertext block, except for the first plaintext in the process is XORed with an initial vector (IV) since there is no ciphertext before. The overall process of encryption is similar to ECB mode, with the simple difference of driving the ciphertext of each block to be mixed with the next plaintext. The CBC encryption mode is presented in Fig 2.

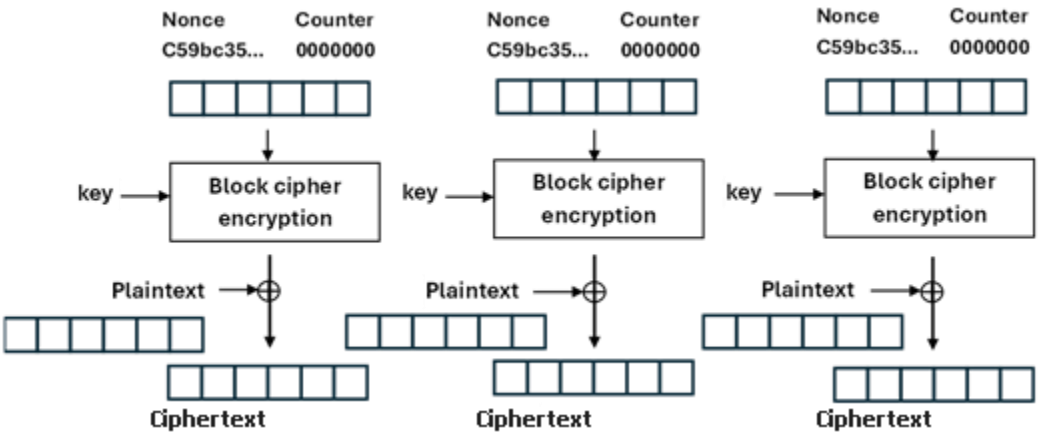

Fig. 2. Cipher Block Chaining mode.

The counter mode (CTR) is another common chaining mode. It uses an incremental counter to be able to add randomization to the encryption process. With counter mode, the incremental counter is encrypted with the block cipher encryption, and the result is then XORed with the plaintext to create finally the ciphertext. In the next block, instead of using ciphertext of the previous block, the counter is incremented and used as in the first block to perform the encryption operation. The CTR mode is presented in Fig. 3.

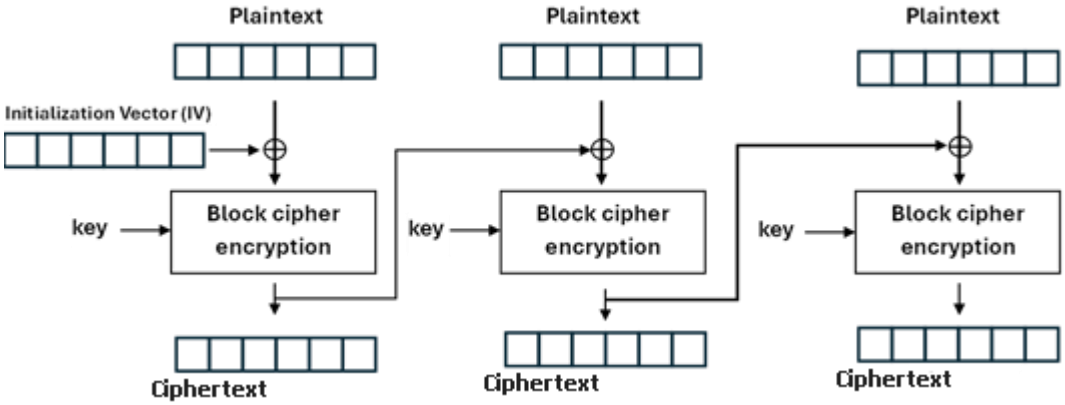

Fig. 3. Counter mode CTR.

III. IMPLEMENTAION OF AES ALGORITHM UNSING ECB MODE

The Advanced encryption standard AES is one of the most commonly used encryption algorithms due to its robustness, security, and ease of implementation in hardware and software. AES is a symmetric block cipher that encrypts 128-bits in a single pass, and it supports 128, 192 and 256-bits key size [11]. It was developed by Joan Daemen and Vincent Rijmen, and is a commonly used algorithm especially in wireless communication. the AES consists of 10 rounds in case of 128 key length, but the number of rounds is 12 and 14 in case of 192-bits and 256-bits respectively. The Round in AES signifies iteration, and each iteration is a set of operations, namely the AddRoundKey SubBytes, ShiftRows, and MixColumns. These operations are executed in every round, except for the last round where there is no MixColumns operation. The AddRoundKey operation is a XOR of the output from the previous operation and the subkey of the active round. This subkey is generated through the Key generation process, or also known as Key expansion. This process uses the initial key to generate a subkey for each round through some mathematical operations.

In this project, we designed two methods for the AES using ECB mode. The first method uses a single AES unit with an input module to divide original data into vectors, and output module to concatenate ciphertext to generate encrypted data. The overall structure of the first method uses an inner pipelining system, while the AES unit itself is an outer-round pipelining, this combination allows achieving a maximum throughput with an optimized area.

The second method uses a similar inner-round pipelining system, but defer in the overall structure, where it uses a parallel data processing instead. The generated ciphertexts are concatenated using a buffer positioned in the output. The diagram of AES module for both methods is illustrated in Fig. 4.

Several implementation techniques of the AES algorithm exist, but not all of them are adapted to chaining mode, since it requires a steady dataflow to guarantee data fluidity and avoid latency. Therefore, pipeline structure is the most convenient for chaining, as it can achieve high performance. The diagram of AES module for both methods is illustrated in Fig. 5

As illustrated in Fig. 5, registers are positioned between rounds for synchronization purpose. The diagram of the first method of AES-ECB is illustrated in Fig. 6.

The encryption unit in Fig. 6 shows the AES-128 module with data formatting units in the input and output to ensure a steady data feeding to the encryption process. The data formatting at the output concatenates ciphertext before generating the encrypted data. The second method of implementation in this project is illustrated in Fig. 7.

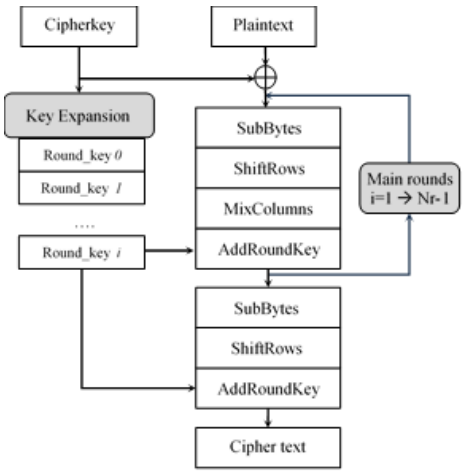

Fig. 4 Advanced encryption algorithm.

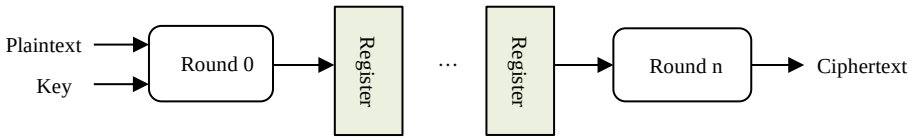

Fig. 5. Advanced encryption standard structure.

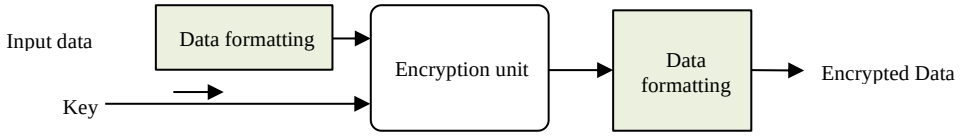

Fig. 6. AES Electronic Codebook mode

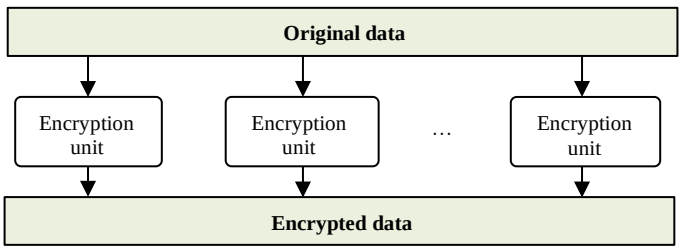

Fig. 7. Parallel AES-ECB mode.

The input data in this method is driven directly to the encryption units without data formatting. It consists of several encryption modules where each module processes a part of the input data.

The development process of this project pass through several phases, starting from the AES modules, progressing toward the top-level structure, using a Hardware Description Language (HDL). The development process is illustrated in Fig. 8.

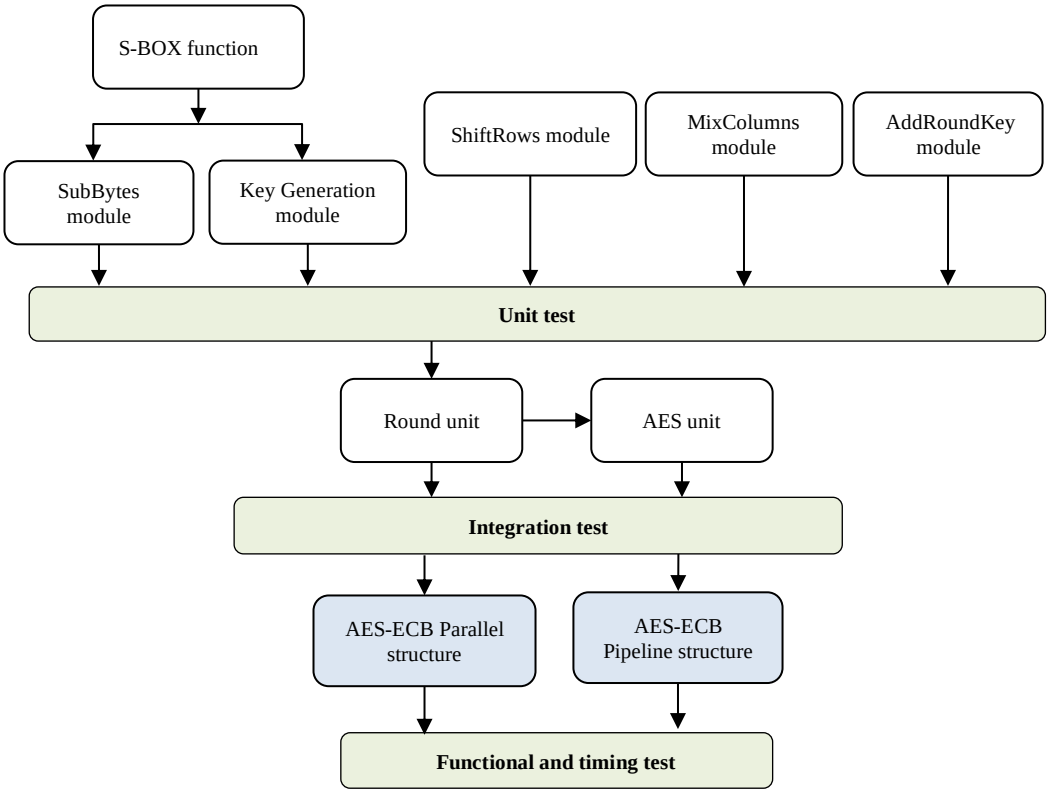

Fig. 8. Development process of the proposed approach.

Each module in the AES is tested separately (unit test) before the assembly of the round unit and the overall structure. The schematic of the round unit is presented in Fig. 9.

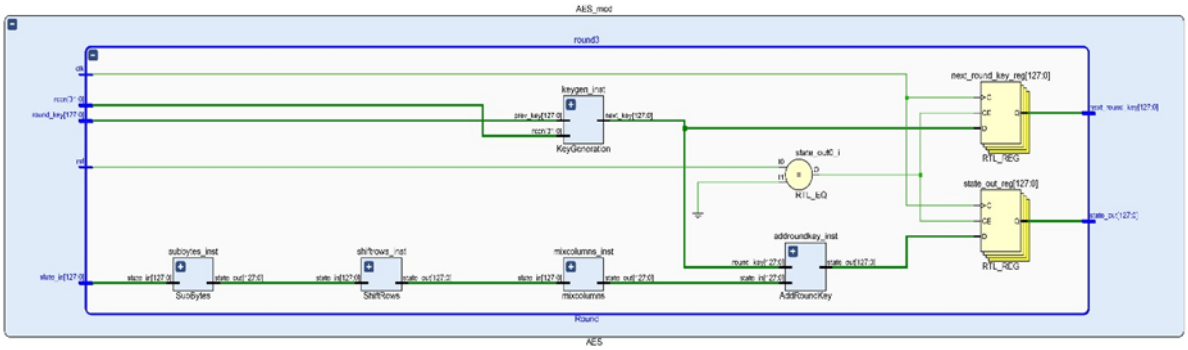

Fig. 9. Round unit schematic.

The Round unit consists of output buffers along with the main operations modules to handle the inner-round pipelining of the AES. The schematic of the AES-ECB pipeline system is illustrated in Fig. 10.

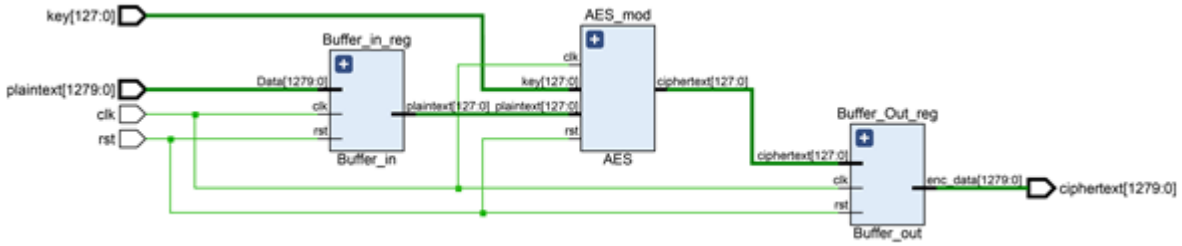

Fig. 10. Pipeline system schematic.

The inputs and outputs of the AES-ECB pass through registers for synchronization. The AES unit processes a new input and generates an output every clock cycle. While the parallel structure processes all the input data simultaneously, where each block is processed with a dedicated AES module. The parallel structure is illustrated in Fig. 11.

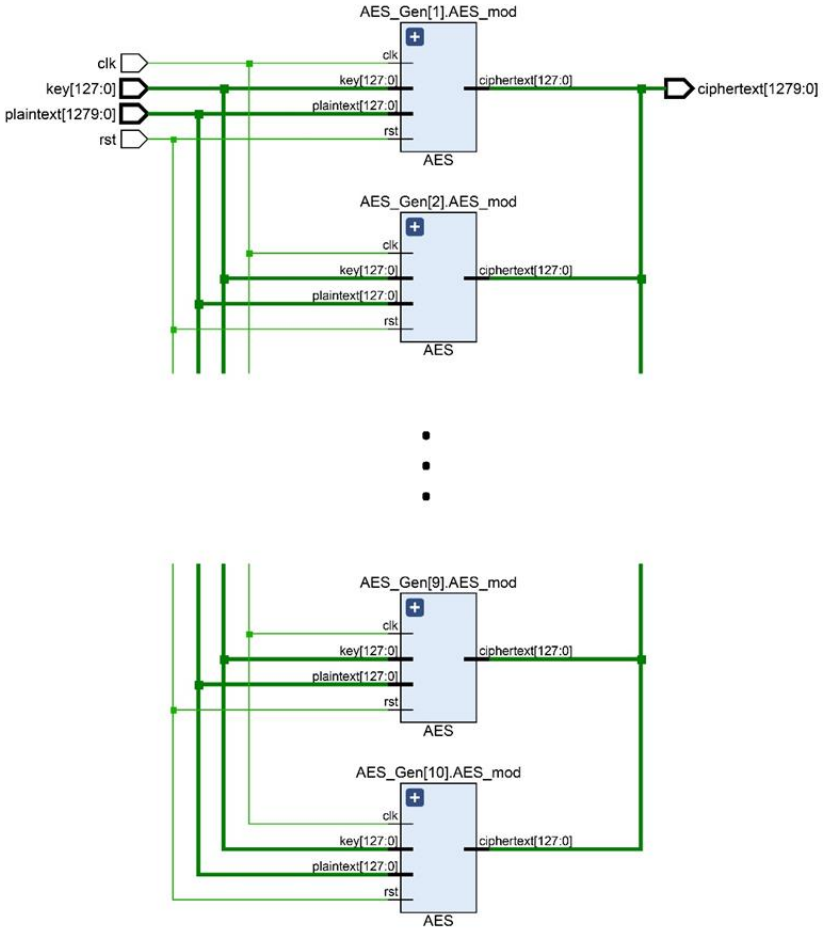

Fig. 11. The parallel structure schematic.

IV. VALIDATION

Both methods parallel and pipeline uses the same AES module with a different overall structure, aiming to ensure a meaningful comparison at the end, as both structures share the same fundamental module.

The maximum frequency F_{max} of a design is computed through running the synthesis and implementation and increasing the timing constraint until getting the smallest Slack violation (WNS <0) in the timing analyses report. The Fmax is calculated using the following equation:

$$F_{max}(MHz) = \max\left(\frac{1000}{T - WNS}\right) \quad (1)$$

Where T is the target clock period (ns) and WNS is the worst negative slack (ns) of the target clock.

The throughput is calculated using the F_{max}, number of clock cycles, and data length. The throughput is defined by the following equation:

$$S(Mbps) = \left(\frac{F_{max} \times dl}{n}\right) \quad (2)$$

Where, F_{max} is the maximum frequency of the design (MHz).

dl = 128 is the block size (bits).

n Number of clock cycles

The resource utilization and throughput of proposed methods is represented in Table 1.

Tab. 1. Resource utilization of proposed methods.

Method	Frequency (MHz)	Throughput	LUT	FF
Parallel structure	264	39.16	28868	7296
Pipeline structure	189.5	24	11328	2503

The parallel design uses 3 AES units as example, but more modules can be placed depending or resources availability in the FPGA used.

The simulation results of parallel and pipeline structures are illustrated in Fig. 12, and Fig. 13 respectively. The pipeline structure processes the first input in 10 clock cycles, while one plaintext is loaded every clock cycle. The parallel structure processes several inputs simultaneously; therefore, it achieves a high throughput at the cost of high resource utilization.

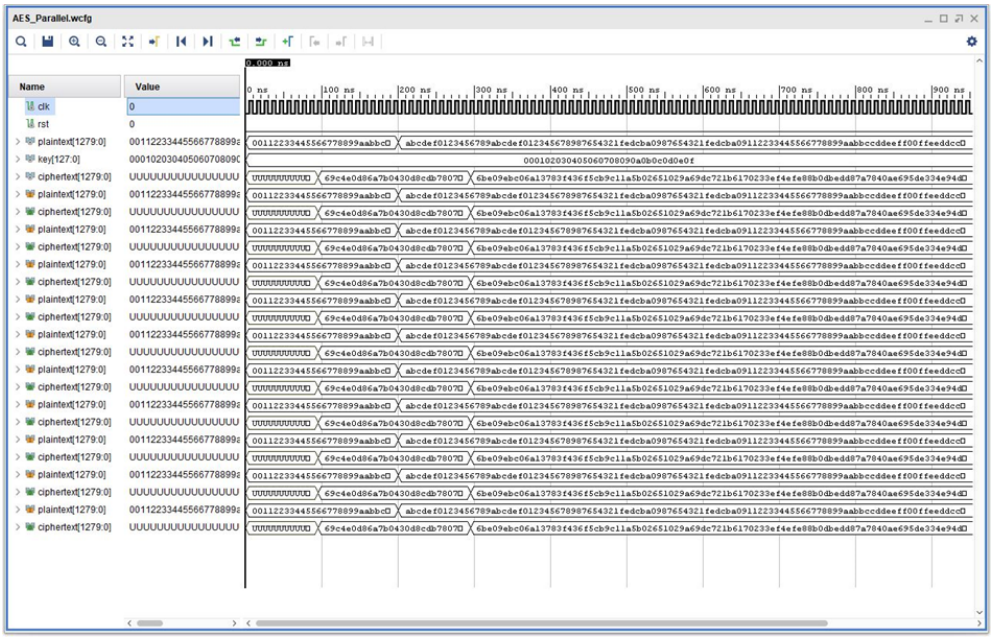

Fig. 13. Simulation results of parallel structure

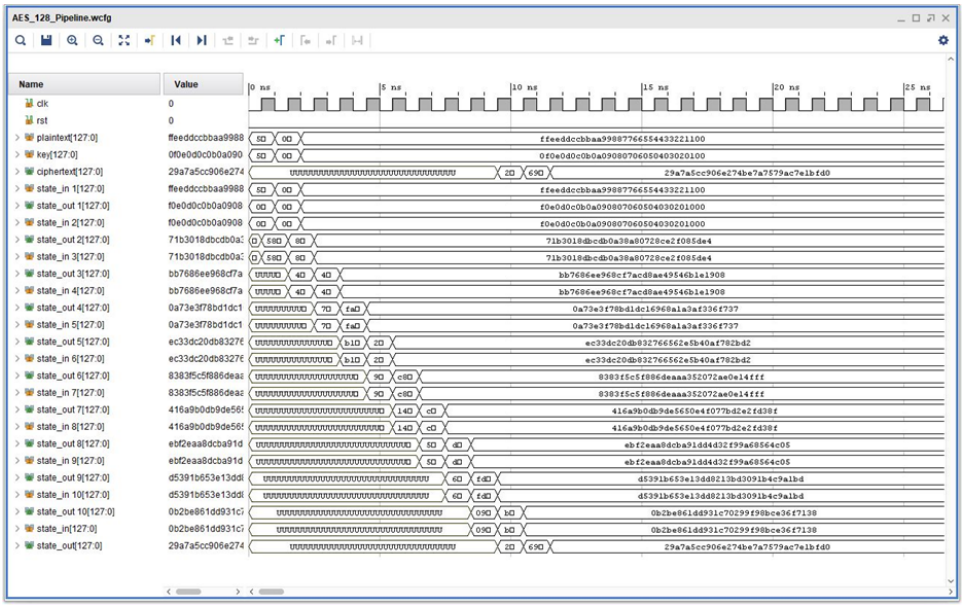

Fig. 14. Simulation results of pipeline structure.

V. CONCLUSION

This work demonstrates the implementation of the Advanced Encryption Standard (AES) in Electronic Codebook (ECB) mode using FPGA technology. Two different methods were implemented to achieve continuous data encryption (parallel and pipeline). The parallel method proved more efficient, achieving a maximum frequency of 264 MHz, compared to 189 MHz for the pipeline system, highlighting the potential of FPGA-based solutions for high-speed encryption applications.

REFERENCES

[1] V. Dahiphale, & al. "Securing IoT devices with fast and energy efficient implementation of PRIDE and PRESENT ciphers", *Cyber Security and Applications*, Vol. 3 (2025): 100055.

[2] M. O. A. Al-Shatari, & al. "FPGA-based lightweight hardware architecture of the PHOTON hash function for IoT edge devices", *IEEE access*, Vol. 8 (2020): pp. 610-618.

[3] A. Poojary, V. G. Kiran Kumar, and H. R. Nagesh. "FPGA implementation novel lightweight MBRISI cipher", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 14, no. 9 (2023): pp. 625-637.

[4] M. Madani and C. Tanougast, "FPGA implementation of an enhanced chaotic-KASUMI block cipher", *Microprocessors and Microsystems*, vol. 80 (2021): 103644.

[5] F. Wu, L. Wang, and Jiguang Wan. "A low cost and inner-round pipelined design of ECB-AES-256 crypto engine for Solid State Disk", *2010 IEEE Fifth International Conference on Networking, Architecture, and Storage*. IEEE, 2010.

[6] A. H. Al-Wattar, "A new approach for the image encryption using AES cipher in ecb mode." *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 14 (2023), no .2: pp. 1061-1074.

[7] Park, Jin Hyung, and Dong Hoon Lee. "FACE: Fast AES CTR mode encryption techniques based on the reuse of repetitive data", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, (2018): pp. 469-499.

[8] J. H. Park et D. H. Lee, "FACE: Fast AES CTR mode Encryption Techniques based on the Reuse of Repetitive Data", *TCHES*, p. 469-499, 2018.

[9] E. H. Bensikaddour & al., "Satellite image encryption method based on AES-CTR algorithm and GEFFE generator", in *IEEE 2017 8th Int. Conf. on Recent Advances in Space Technologies (RAST), Istanbul, Turkey*, 2017, p. 247-252.

[10] W.-K. Lee, & al., "Speed Record of AES-CTR and AES-ECB Bit-Sliced Implementation on GPUs", *EEE Embedded Systems Letters*, vol. 16, no 4, p. 481-484, déc. 2024, doi: 10.1109/LES.2024.3409725.

[11] J. Daemen et V. Rijmen, "The design of Rijndael", Vol. 2. New York: *Springer-verlag*, 2002.