COMMUNICATIONS IN
SCIENCE AND TECHNOLOGY
Homepage: cst.kipmi.or.id

A modified MixColumn-InversMixColumn in AES algorithm suitable for hardware implementation using FPGA device

Ragiel Hadi Prayitno[a], Latifah[b,*], Sunny Arief Sudiro[b], Sarifuddin Madenda[a], Suryadi Harmanto[a]

[a]*Doctoral Study Program of Information Technology Gunadarma University, Depok 16424, Indonesia*
[b]*Departement of Computer Science, STMIK Jakarta STI&K, Jakarta Selatan 12140, Indonesia*

Article history:
Received: 24 August 2023 / Received in revised form: 7 December 2023 / Accepted: 18 December 2023

Abstract

This article described the Advanced Encryption Standard (AES) encryption and decryption process without using lookup tables in the MixColumns transformation and parallelizing the transformation process implemented in the Field Programmable Gate Array (FPGA) hardware. Parallelism of the hardware process conducted to the transformation of key schedule, addroundkey, subbyte and shiftrows (subshift) and mixcolumns in the first 5 rounds of the encryption process. The decryption process was parallelized in subshift transformations, both transformations were implemented at the same time. This research produced a modified AES encryption and decryption method and algorithm with the aim of minimizing the resources required for hardware implementation. The method in this article was applied to Xilinx ISE 14.7 software. The experimental results showed that the encryption process required 2,357 slice LUT's, 845 occupied slices and 26 IOB's, while the decryption process required 2,896 LUT's, 1,323 occupied slices and 26 IOB's resources. The encryption and decryption processes each took an average of 2.891 nanoseconds and 3.467 nanoseconds for every 128 bits of data. This approach leads us to obtain a component with minimum resources and enough computational speed.

Keywords: AES; FPGA; mixcolumns; inversemixccolumns; hardware implementation

1. Introduction

Data distributed on the internet can be easily hacked, modified and duplicated. This is a risk to data confidentiality. Confidential information becomes significant to be maintained as part of data security [1]. Data that is converted into digital form is more vulnerable to be compromised [2]. Hackers generally attack digital storage and data transmission media that have the potential to harm certain parties.

One of the security methods that have been existed for centuries is the cryptographic method. This method is used for communicating between the military and commercial. Cryptographic methods are an important part of running the global economy and are used by millions of people every day in electronic commerce through the internet. Sensitive information (bank records, credit card statements, passwords, or personal communications) must be secured with decryption so that it cannot be accessed by other people or unauthorized parties [3], [4].

Cryptographic methods convert plain text messages into encrypted text messages using an algorithm that is known between the sender and recipient. It is possible that sensitive information is stored or transmitted through an insecure network (such as the Internet) therefore it cannot be read by anyone except the intended recipient. The encrypted message can be returned to the original message. Encrypted messages can only be read by the intended recipient because the recipient has the same key as the sender. The process of converting plain text messages into cipher text is called encryption, while the process of converting cipher texts into plain text messages is called decryption [5].

The key used in each encryption and decryption process can be a word or phrase, where the key is part of a cryptographic method that functions to secure data. The National Institute of Standards and Technology (NIST) in 2001 adopted the Advanced Encryption Standard (AES) encryption method as Federal Information Processing Standards (FIPS) [6], [7]. These methods have been tough tested and proven towards hacking by providing a high level of security. This is because the process is quite complicated when performing the AES encryption-decryption method. The method consists of 10 rounds of the transformation process, where in each round there are 4 stages of a complex transformation process, namely SubBytes, ShiftRows, MixColumns and Addroundkey [8].

The application of the encryption-decryption process can be run both software and hardware [9]. The speed of the encryption-decryption process is better hardware than software. It is because software-based applications run at layer 7 OSI (Open Systems Interconnection) or application layer. The application of encryption-decryption on hardware runs on OSI layer 1 or physical layer so that hardware-based applications are more resistant to attacks. The physical layer is

* Corresponding author. Tel.: +62 817-4933-518.
Email: latifah@staff.jak-stik.ac.id
https://doi.org/10.21924/cst.8.2.2023.1257

more difficult to hack and is not based on the existing operating system [10].

Research conducted by Shraddha Soni, Himani Agrawal, and Monisha Sharma [11] compared the DES algorithm and the AES algorithm. The results of this research indicates that the AES Algorithm's time consumption for encryption and decryption processes is faster (99.871 seconds) than the DES Algorithm (215.9359 seconds).

Research conducted by Nia Gella Augoestien, and Agfianto Eko Putra [12] produced an AES encryption and decryption model on FPGA hardware. This model has a hardware efficiency of 1.94 Mbps/Slice and a throughput of 308.96Mbps. This approach has a latency of 83 clocks.

Research conducted by Umer Farooq, and M. Faisal Aslam [8] compared several techniques used in implementing the AES algorithm on FPGA devices. The research showed technique that was proposed can take the advantage of the good available resources and provided a better trade-off as far as use of resource and throughput design.

MixColumns transformation is part of encryption process in AES Algorithm. The Inverse MixColumns transformation is part of decryption process in AES algorithm. This transformation can be used to secure data by shuffling the original data in the state array columns; therefore, new data can be obtained in each state array column.

MixColumns/InverseMixColumns transforma-tion consumed more power and logic so that optimization is important [13]. In traditional AES, MixColumns and Inverse MixColumns are implemented as separate modules, except for some implementations which used sharing resource between MixColumns and Inverse MixColumns to optimize power and space [14], [15]. The design proposed in previous research is based on byte, matrix decomposition and minimization of equations using the generalized sub-expression elimination [14]. MixColumns and InverseMixColumns operations can also be performed by ignoring mathematical analysis by using reference tables L and E, each of which has 256 values [16], [17].

Another study using FPGA in cryptography technic was found in [26]. The approach was to implement BCF (Block Cipher-Four) algorithm in hardware based environment, and again this approach was to obtain faster computation speed (2,847 times faster in this research using FPGA DE2) and the possibility of implementation in embedded system.

The provided studies on FPGA-based AES encryption and decryption models lack detailed information on the specific optimization techniques, the trade-offs achieved between resource utilization and throughput, and the design details of MixColumns and Inverse MixColumns operations. The proposed design based on bytes, decomposition matrix, and minimization equations is mentioned, but without explicit details. The comparative analysis of mathematical analysis versus using reference tables for MixColumns and Inverse MixColumns operations is absent. More comprehensive exploration of these aspects would enhance the clarity and applicability of the research findings.

This research proposed the development of a modified AES algorithm encryption and decryption method thus it can be more easily and efficiently implemented on hardware using FPGA with minimal resources. Modifications are made to the mixcolumns transformation using Galois Field (GF) 2^8 multiplication and the parallelization process of subbyte and shiftrows transformations and each transformation with keyschedule in the first 5 rounds of the encryption process.

2. Materials and Methods

The AES algorithm is an encryption-decryption method in cryptography [15]. AES, also called Rijndael encryption, is a block encryption standard adopted by the United States Federal Government [18,19]. AES is projected to replace the DES algorithm. After several rounds of filtering, AES is then widely used [20]. On November 26, 2011, NIST announced a new encryption standard after five years, and it was implemented in May 2002. After four years of deposition and testing, AES has become one of the most popular symmetric encryption algorithms [19].

The AES algorithm has 3 types of key lengths 128 bits, 192 bits and 256 bits embedded with 128 bits size of packet and the AES algorithm shows a good compliance. This causes the AES encryption system to be widely used in many softwares and hardwares. The key length often used in the AES algorithm is 128 bits. If it is below the key length, 10 calculations will be repeated in the internal algorithm. Besides the final round, each round comprises four parts: SubBytes, ShiftRows, MixColumns, AddRoundKey [19]. Exclusivity occurs between plain text and key extension blocks [21]. In the AES algorithm, there are at least five units of measurement that can be utilized, namely bits, bytes, characters, groups, and status.

2.1. Subbytes

SubBytes transformation is a non-linear and reversible byte transformation. This leads to AES being strong enough in attack. SubBytes transformation is performed by substituting each byte of the matrix with a value stored in the S-box table for encryption. The decryption and substitution process is conducted using the Sbox inverse table for each state. S-box is an algebraic vector Boolean function with 8bit input and output [22]. The S-box table is formed from a 256-byte table search [16].

2.2. ShiftRows

The ShiftRows transformation shifts rows 1, 2 and 3 of the State matrix cyclically to the left at each position 1, 2 and 3. The offset value depends on the row number. The first line is unchanged. Cyclic rotation of the rows gives diffusion properties in the AES algorithm. The ShiftRows transformation is presented in Figure 1.

$$\begin{bmatrix} P_0 & P_4 & P_8 & P_{12} \\ P_1 & P_5 & P_9 & P_{13} \\ P_2 & P_6 & P_{10} & P_{14} \\ P_3 & P_7 & P_{11} & P_{15} \end{bmatrix} \xrightarrow{\text{Shiftrows}} \begin{bmatrix} P_0 & P_4 & P_8 & P_{12} \\ P_5 & P_9 & P_{13} & P_1 \\ P_{10} & P_{14} & P_2 & P_6 \\ P_{15} & P_3 & P_7 & P_{11} \end{bmatrix}$$

Fig. 1. ShiftRows Process

2.3. Mixcolumn transformation

MixColumns transformation, the column in state (which is a word) is performed as a polynomial with the coefficient in GF 2^8. Word [a3, a2, a1, a0] is performed as a polynomial, therefore it becomes $a3X^3+a2X^2+a1X^1+a0$.

The addition of a polynomial is performed because it is added to polynomial ring. So that, the result remains as big as a word, the multiplication is performed modulo polynomial

$$g(x) = X^4 + 1 \quad (1)$$

$X^4 + 1$ is the equation of decimal value 17, where X^4 represents the value 24 and 1 represents 20. It is easy to indicate that

$$X^i \bmod (X^4 + 1) = X^{i \bmod 4} \quad (2)$$

Because -1 = 1 in GF 2^8. Since X4 + 1 is not irreducible polynomial in K[X] where K = GF 2^8, then K[X]/g(X)K[X] is not a field: not all polynomial has inverse. However, the polynomial

$$a(x) = 03X^3 + 01X^2 + 01X + 02 \quad (3)$$

has inverse

$$a^{-1}(x) = 0BX^3 + 0DX^2 + 09X + 0E \quad (4)$$

Values of equations (3) and (4) are in the form of hexadecimal numbers. The MixColumns transformation multiplies (modulo the polynomial g(X)) each column in the state (performed as a polynomial) by the polynomial a(X). The MixColumns state transformation can be formulated in terms of its effect on each column c as follows [6]:

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad (5)$$

$S_{0,c}$, $S_{1,c}$, $S_{2,c}$, and $S_{3,c}$ are the original data. $S'_{0,c}$ is data change in the first row of column c which resulted from MixColumns/InverseMixColumns transformation process between MixColumns matrix and original data. Afterward for $S'_{1,c}$, $S'_{2,c}$, and $S'_{3,c}$ are data change in row 2 to row 4 column c.

MixColumns and InverseMixColumns operations are used in AES for hardware implementation purposes in constrained environments. This research is supported by mathematical analysis of both transformations and leads to efficient serial and parallel decomposition [15].

Previous research created a high-efficiency architecture to perform mixed column operations, which is a key function in the AES method. This research uses prehistoric Vedic Mathematics methods that can give more effective results in AES [23].

Low-power FPGA architecture is based on the use of AES MixColumns/InverseMixColumns transformations. The implementation uses decomposition techniques, pre-computation, and parallel power supply to reduce a power circuit. The circuit consumption is decreased as compared to previous implementations [13].

Instead of using two different mixed and inverse column transformation modules, one module can be implemented to both transformations. This reduces the entire area consumption of the AES algorithm [24].

Previous researchers used memoryless combinatorial design for SB/ISR implementation as an alternative to obtaining higher speed by eliminating memory access delays while maintaining or increasing the efficiency of use of the entire component area. This research also explores the use of sub-pipelining to further increase in the speed and throughput of suggestion implementation. FPGA architecture applies optimization in inverter design and isomorphic mapping using composite field arithmetic to reduce the area requirements of the components used [25].

2.4. Proposed method

AES encryption-decryption method is tested and proven to provide a high level of security towards hack [8], [23]. The weakness of AES encryption is the process is quite complex. The complexity of the standard AES model caused a challenge in hardware implementation. The standard AES process in previous studies was conducted sequentially or sequentially.

The previous research also used tables L and E as a reference in processing MixColumns transformation, where each table consists of 256 reference values [16], while this research used GF 2^8 multiplication and the parallelization process of AES algorithm transformation. Parallelization can increase the speed of the AES encryption-decryption process. The limitation of processing/calculating numbers is one thing that must also be concerned, for example related to integer data and real numbers (floating point) in an FPGA. The algorithm design must also be efficient so that optimization in hardware can be achieved, especially increased processing speed, delayed time, and minimal needed FPGA resources.

Figure 2 shows a flowchart of the modified AES encryption-decryption process proposed in this dissertation research. The development was applied in the form of parallelization between the subbyte and shiftrows transformation processes. The proposed parallelization model has advantages in the form of more efficient use of network resources, faster encryption-decryption process, lower costs, and provided a high level of security.

The concept of implementing encryption and decryption integration process is with the VHDL programming language. The parallelization model between the subbyte and shiftrows transformation processes can be conducted so that it is expected that the use of resources can be more efficient, optimize speed and provide a high level of security. The design of each transformation uses sequential logic techniques because the output depends on the previous input and needs storage element.

The sequence of encryption and decryption processes in the modified AES algorithm is as followed:

1. Key schedule calculation process
   The first process in the AES algorithm is to find a key schedule that will be used for 10 rounds.
2. XOR process between plaintext and cipher key
   The second process is to do XOR between plaintext and cipher key to generate addroundkey.
3. Transforming Subbytes, Shiftrows, MixColumns, and Addroundkey

Subbytes transformation is performed by substitution with the S-Box table that has been stored in memory. The resulting data is a 4x4 matrix. In the shiftrows transformation, there is no shift for row 0, for row 1 of the matrix there is a shift of 1 byte to the left, for row 2 of the matrix there is a shift of 2 bytes to the left, and for row 3 of the matrix there is a shift of 3 bytes to the left. The MixColumns transformation performs the multiplication process (modulo polynomial g(X)) for each column in the state with the polynomial a(X). In the addroundkey transformation there is an XOR process between the state (matrix) and the key schedule. This process is repeated for 9 rounds.

4. Subbytes, Shiftrows, and addroundkey Transformation Process

   In this process there are only 3 transformation processes, they are subbytes, shiftrows and addroundkey. This process occurs in the last loop in the AES algorithm. After it is complete it will produce a cipher text.

Figure 3 is the concept of implementation of encryption integration, in which there is a parallelization process to the key schedule transformation with addroundkey, subbyte and shiftrows, mixcolumns in the first 5 rounds. For the last 5 rounds there is also parallelization, namely the transformation of subbytes and shiftrows.

Figure 4 is the concept of implementation of decryption integration, where the concept is a parallelization process to the transformation of subbytes and shiftrows. Implementation is performed by using FPGA tools as a component development that applies the concept of encryption and decryption. The result of this design is Intellectual Property (IPCore) (component description) which for actual implementation still must go through the development of integrated circuit (IC) in the form of electronic components for the encryption and decryption process. The input data in modified integration of AES encryption algorithm is stated as “textin” and “keyin” with type std_logic_vector(7 downto 0) or 8 bit data. The output is stated as a "cipher" with type std_logic_vector(7 downto 0) or output data 8 bit. The modified AES algorithm integration uses 10 memories consisting of 3 constant memory and 7 memory as temporary storage of the generated data. Constant memory is used for MixColumns data of 16 memory cells, rcon of 10 memory cells and sbox of 256 memory cells of type std_logic_vector(7 downto 0). These 3 constant memories do not change in value so only 3 constant memories are needed. 7 memory as temporary storage is for the values that keep changing.

The first memory as temporary storage is declared as mem2 with type std_logic_vector(7 downto 0) on ram 1 of 16 memory cells used to store input data from textin. The second memory as temporary storage is declared as mem3 with type std_logic_vector(7 downto 0) is 16 memory cells used for store input data from keyin. The third memory as temporary storage is declared as mem4 with type std_logic_vector(7 downto 0) is 16 memory cells which is used for store the result of the addroundkey transformation before starting the loop. The fourth memory as temporary storage is declared as mem5 with type std_logic_vector(7 downto 0) is 16 memory cells which is used for store the result of parallelization of subbytes and shitrows. The fifth memory as temporary storage is declared as mem6 with type std_logic_vector(7 downto 0) is 16 memory cells used for store the MixColumns results. The sixth memory as temporary storage is declared as mem2p with type

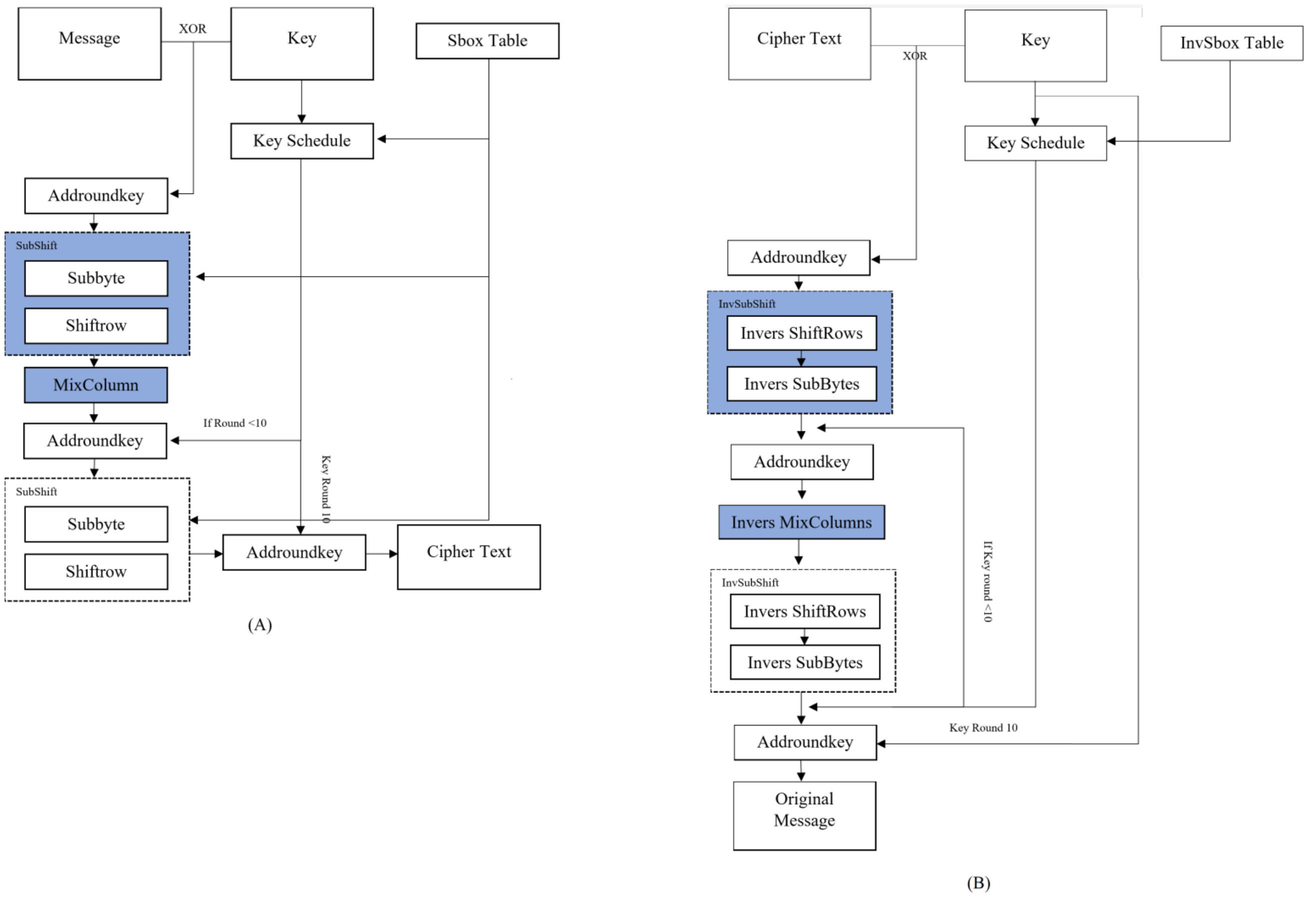


Fig. 2. Flowchart of Encryption (A) and Decryption (B) Processes

std_logic_vector(7 downto 0) of 160 memory cells which is used to store the key schedule results. The seventh memory as temporary storage is declared as mem7 with type std_logic_vector(7 downto 0) of 16 memory cells which is used to store the results of the addroundkey transformation and the encryption results that will be delivered to the "cipher".

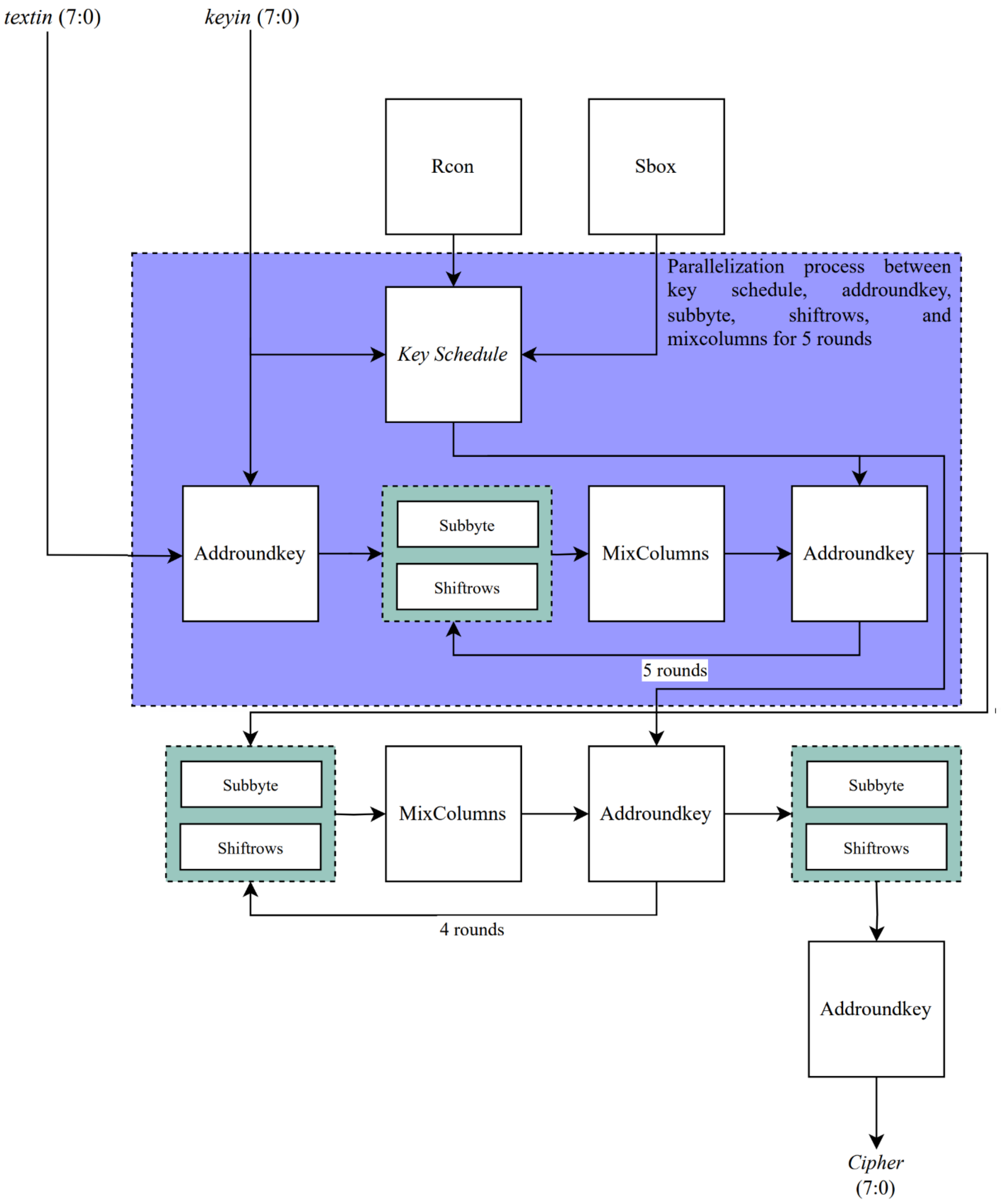


Fig. 3. Encryption integration concept

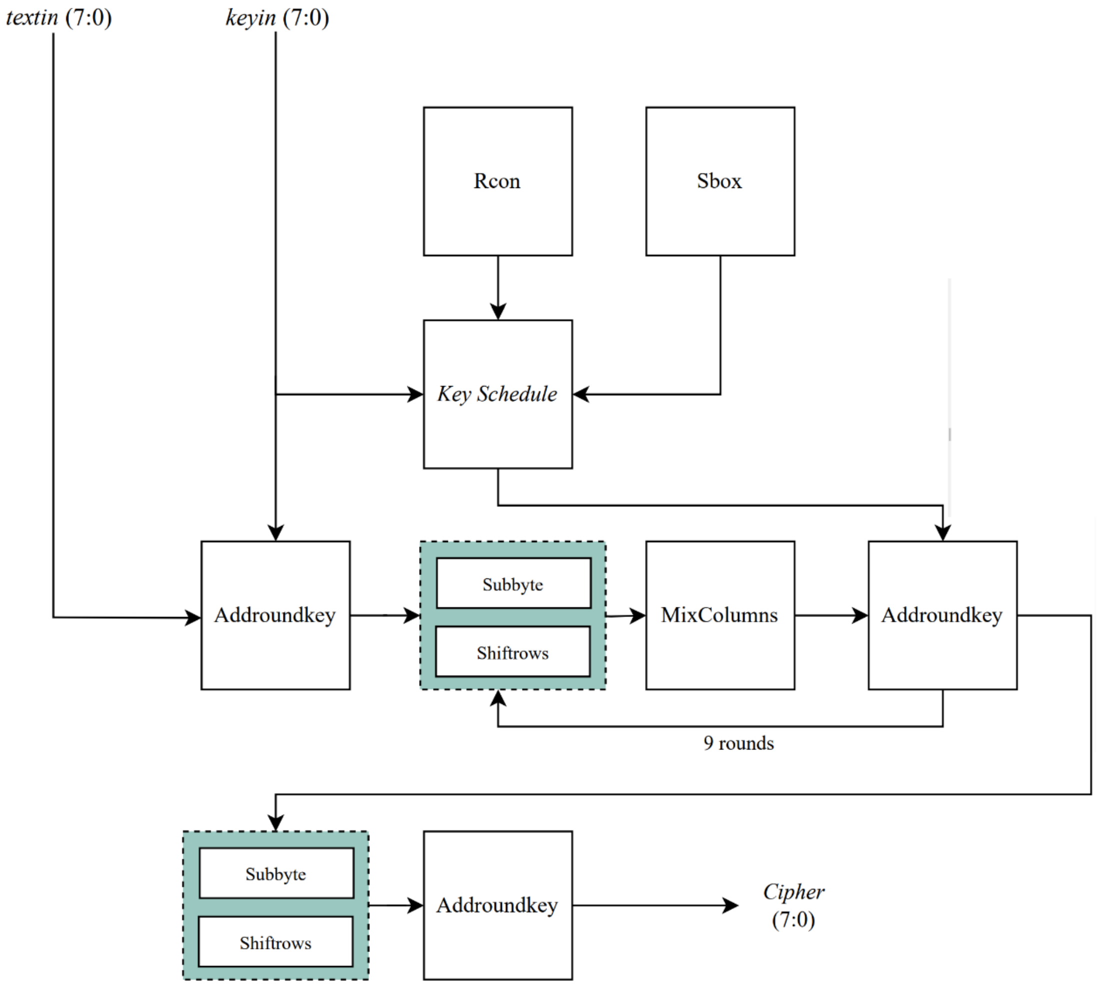


Fig. 4. Decryption integration concept

3. Results and Discussion

This research carried out only 1 type of message (a total of 128 bits of binary data) because each message was carried out per 128 bits. If the message exceeds 128 bits of data (example: 129 bits) then the data will be considered as 256 bits of data and the process will be repeated twice. If it exceeds 256 bits of data (example: 257 bits of data) then it is considered 284 data and the process is repeated 3 times and so on. Focusing for the one message in experiment was to be sure that the value for each cycle in hardware implementation was correctly obtained.

In this research, the encryption and decryption method of AES algorithm with FPGA-based hardware was conducted in sequential logic. The implementation was performed by using Xilinx ISE software and VHDL programming language and simulated with ISim Simulator. The transformations that existed in AES algorithm are being implemented based on the evaluation results and it is simulated with the ISim Simulator to ensure the implementation results are in line with the evaluation results that have been conducted. The result of the implementation of each transformation is integrated by using VHDL programming language according to AES algorithm to produce encryption and decryption.

Figure 5 show that the implementation results have latency of 33 clocks after the last data is stored for reading the output.

Figure 6 shows a summary of encryption resources, which the implementation needs 2,357 slice LUT's and 26 bonded IOBs. Figure 6 also shows timing summary, where the encryption component needs a computation time of 2.891ns and a maximum frequency of 345.853 MHz.

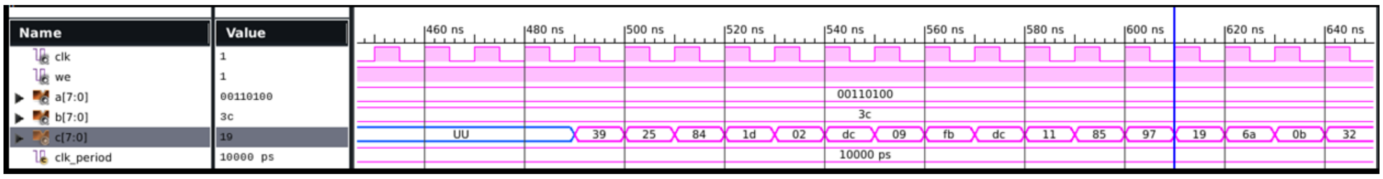
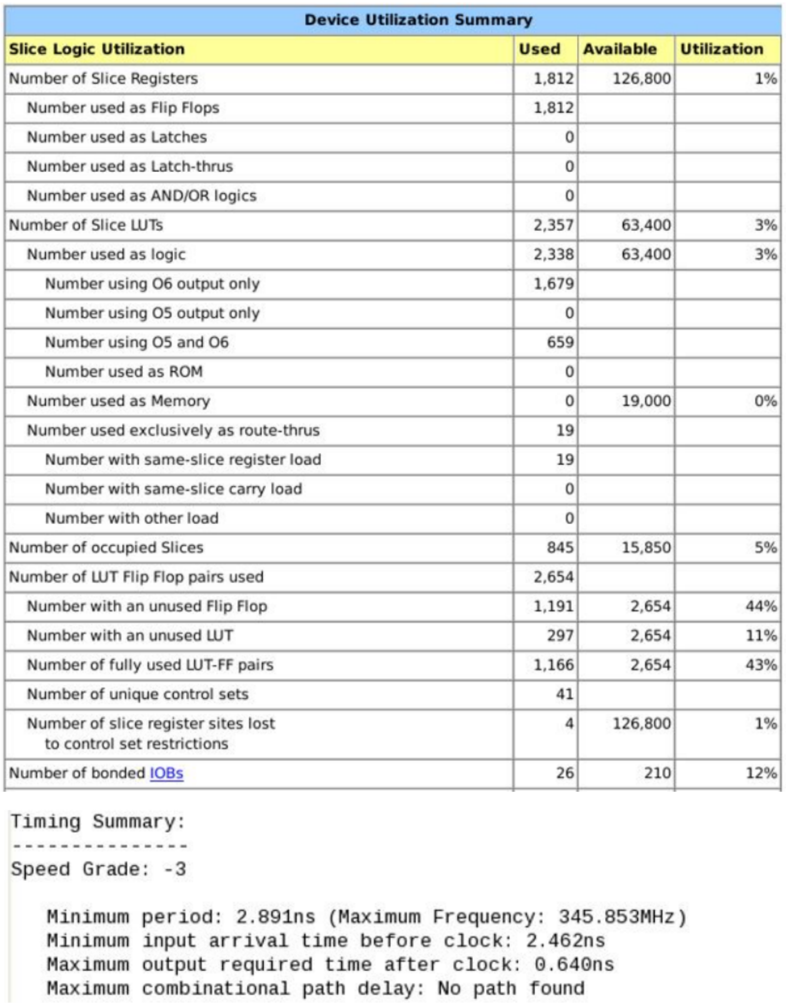


Fig. 5. Encryption simulation results on ISim simulator



Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	1,812	126,800	1%
Number used as Flip Flops	1,812		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	2,357	63,400	3%
Number used as logic	2,338	63,400	3%
Number using O6 output only	1,679		
Number using O5 output only	0		
Number using O5 and O6	659		
Number used as ROM	0		
Number used as Memory	0	19,000	0%
Number used exclusively as route-thrus	19		
Number with same-slice register load	19		
Number with same-slice carry load	0		
Number with other load	0		
Number of occupied Slices	845	15,850	5%
Number of LUT Flip Flop pairs used	2,654		
Number with an unused Flip Flop	1,191	2,654	44%
Number with an unused LUT	297	2,654	11%
Number of fully used LUT-FF pairs	1,166	2,654	43%
Number of unique control sets	41		
Number of slice register sites lost to control set restrictions	4	126,800	1%
Number of bonded IOBs	26	210	12%

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 2.891ns (Maximum Frequency: 345.853MHz)
   Minimum input arrival time before clock: 2.462ns
   Maximum output required time after clock: 0.640ns
   Maximum combinational path delay: No path found
```

Fig. 6. Summary of encryption resources on Xilinx and. Encryption timing summary

Figure 7 shows that the implementation results have a latency of 33 clocks after the last data is stored for reading the output. Figure 8 shows timing summary, the decryption component needs a computation time of 3.467ns and a maximum frequency of 288.403 MHz. Figure 8 also show a summary of decryption, where the implementation needs 2,896 slice LUT's and 26 bonded IOBs.The concept of testing encryption and decryption with VHDL programming language on the Nexys A7-100T FPGA hardware has been conducted. The test is conducted by utilizing the 7-Segment that is contained in the Nexys A7-100T FPGA as the output of the encryption and decryption results. The concept of testing encryption is shown in Figure 9(A) and decryption is shown in Figure 9(B).

The message data that is used in the encryption test is "01020304050607080000000000000000". The key used is "bismillah lancar". Table 1 shows the results of resources that are performed with different devices. Minimizing "Occupied Slice" and "Slice LUT's" in FPGA design is crucial for efficient resource utilization. A smaller footprint of occupied slices and fewer Look-Up Tables (LUTs) signifies a more resource-optimized design. Achieving reduced "Latency" leads to faster system response times, enhancing overall performance. Lower "Timing (ns)" values indicate quicker completion of operations or clock cycles on the FPGA.

The experiment was performed as the same device as the previous researcher to compare the resources that were used. The method used in this research produces latency of 33 clocks on Spartan 6 devices. The latency in this research is lower than the method [12] (83 clocks). The parallelization process that is produced in latency savings so that the latency obtained was slower than the method in [12]. The slice used in this research was 1936 slice on Spartan 6 device.

The occupied slice in this research was still higher than the method in [8] (159 slice). This is due to the technique in designing FPGA-based circuits, method in [8] uses FSM technique while this research uses a sequential logic technique to make the number of occupied slices produced is larger. Slice LUT's used in this research were 3586 slice on Spartan 6 devices. The use of GF2^8 in this research resulted in a slice LUT which was lower than the method in [8] (9276 slice).

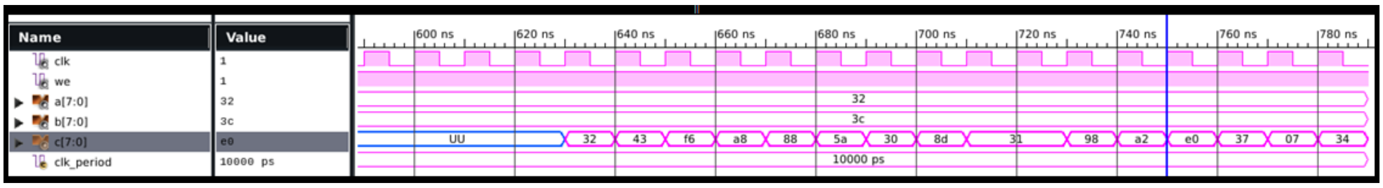
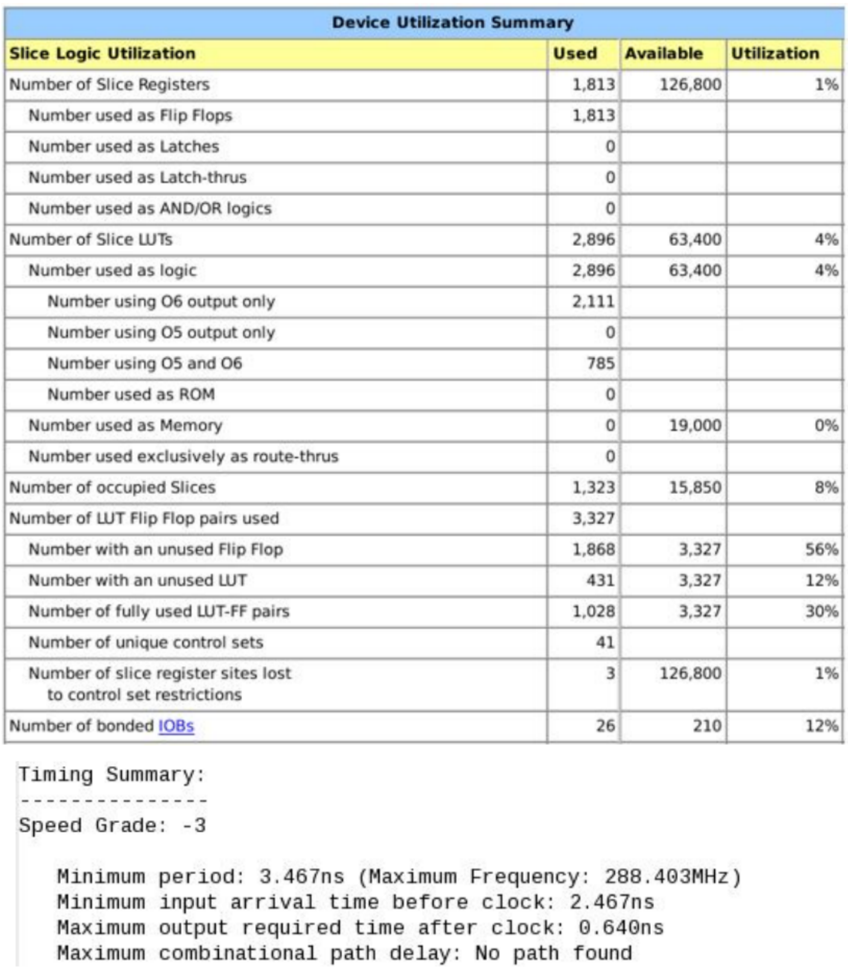


Fig. 7. Decryption simulation results on ISim simulator



Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	1,813	126,800	1%
Number used as Flip Flops	1,813		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	2,896	63,400	4%
Number used as logic	2,896	63,400	4%
Number using O6 output only	2,111		
Number using O5 output only	0		
Number using O5 and O6	785		
Number used as ROM	0		
Number used as Memory	0	19,000	0%
Number used exclusively as route-thrus	0		
Number of occupied Slices	1,323	15,850	8%
Number of LUT Flip Flop pairs used	3,327		
Number with an unused Flip Flop	1,868	3,327	56%
Number with an unused LUT	431	3,327	12%
Number of fully used LUT-FF pairs	1,028	3,327	30%
Number of unique control sets	41		
Number of slice register sites lost to control set restrictions	3	126,800	1%
Number of bonded IOBs	26	210	12%

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 3.467ns (Maximum Frequency: 288.403MHz)
   Minimum input arrival time before clock: 2.467ns
   Maximum output required time after clock: 0.640ns
   Maximum combinational path delay: No path found
```

Fig. 8. Summary of decryption resources on Xilinx and Decryption timing summary

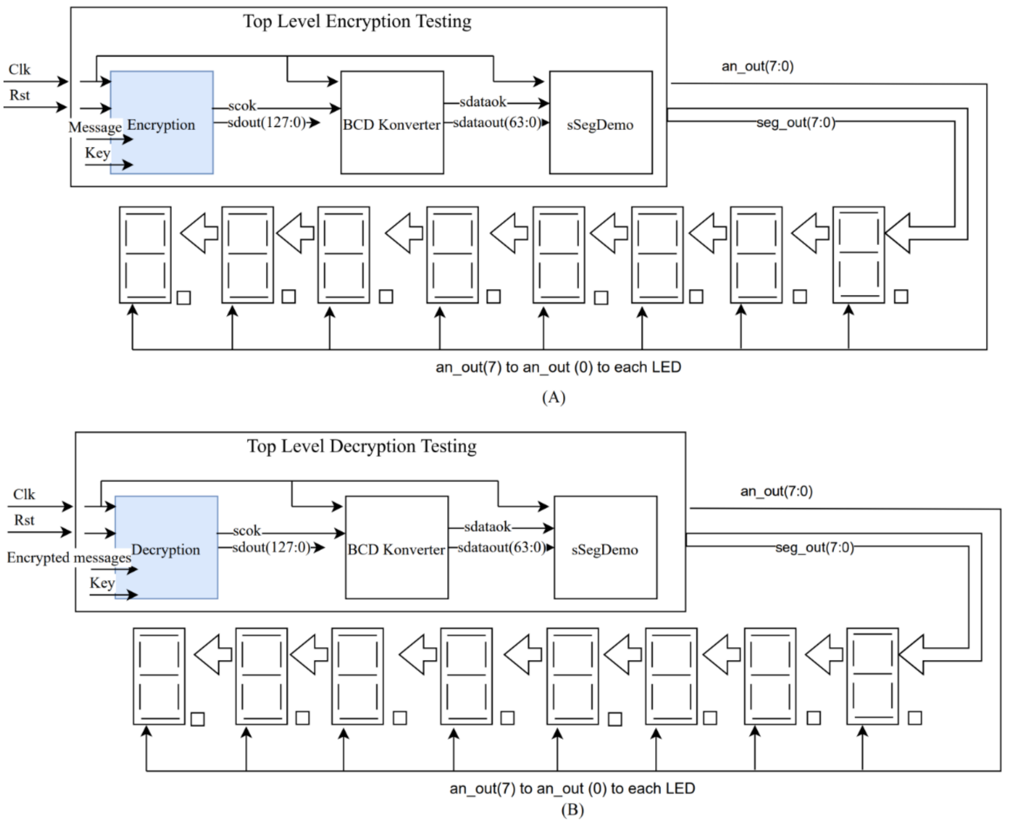


Fig. 9. Concept of testing encryption (A) and decryption (B)

Table 1. Performance Comparison

Performance	Method in [12]	Method in [8]	This work	
Device	Spartan 6	Spartan 6	Spartan 6	Artix 7
Occupied Slice	159	N/A	1936	845
Slice LUT's	N/A	9276	3586	2357
Latensi	83	N/A	33	33
Timing (ns)	4,922	1,127	5,920	2,891

4. Conclusion

The modified AES encryption and decryption algorithm has been successfully implemented into a hardware component in the form of IPCore. Implementation trials were conducted on the Nexys A7-100T FPGA hardware with 7-Segment facilities as the output display. The FPGA resources used in the encryption process are 2,357 slice LUT's, 845 occupied slices, 26 bonded IOBs and a latency of 33 clocks with a computation time of 2.891 nanoseconds for 128 bits of data. The resources needed for the decryption process are 2,896 slice LUT's, 1,323 occupied slices, 26 bonded IOBs and a latency of 33 clocks with a computation time of 3,467 nanoseconds for 128 bits of data. The implementation is suitable for resource-constrained environments such as embedded systems. The impact extends to secure systems, enabling applications like encrypted communication channels and secure data storage. Additionally, the versatility of fpga hardware allows for adaptability in implementing cryptographic algorithms, contributing to both practical applications and educational advancements in the field.

Acknowledgements

.This work is supported by Research Program Kemendikbudristek Indonesia in Hibah Penelitian Disertasi Contract Number: 064/SP2H/LT/DRPM/2021 dated 18 March 2021 and Yayasan Pendidikan Gunadarma Jakarta Indonesia, Contract Number: 05A.26/LP/UG/IV/2021, 5 April 2021

References

1. TM. 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*, ELEC, 10(16) (2021).
2. A. Muslim Djamalilleil, M. Salim, Y. Alwi and H. Herman, *Modified Transposition Cipher Algorithm for Images Encryption*, The 2nd east Indonesia conference on computer and information technology (Eiconcit Makassar, South Sulawesi, (2018) 1-4.
3. Mu. Annalakshmi and A. Padmapriya, *Zigzag Ciphers: A Novel Transposition Method*, IJCA Proceedings on International Conference on Computing and information Technology 2013 IC2IT(2), (2013) 8-12.
4. C.A. Murugan, P. Karthigaikumar and S.S. Priya, *FPGA Implementation Of Hardware Architecture With AES Encryptor Using Sub-Pipelined S-Box Techniques For Compact Applications*, AUTOMATIKA, 61(4) (2020) 682–693.
5. P. Poonia and P. Kantha, *Comparative Study of Various Substitution and Transposition Encryption Techniques*, Int. J. Comput. Appl., 145(10) (2016) 24-27.
6. E. Barker, *Guideline for Using Cryptographic Standards in the Federal Government:Cryptographic Mechanisms*, NIST Special Publication 800-175B Revision 1, (2020).
7. I.C. Guzmán, R.D. Nieto and Á. Bernal, *FPGA implementation of the AES-128 algorithm in non-feedback modes of operation*, Dyna, 83(198) (2016) 37-43.
8. U. Farooq, and M. Faisal Aslam, *Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA*, J. King Saud Univ., 29(3) (2017) 295-302.
9. P. Vijayakumar, P.L. Kishore, K.V.D. Reddy, S. Reddy, R. Rajashree and A. Durai, *FPGA Implementation of High Speed AES Based Authentication Algorithm*, JAT, 12(7) (2020) 112-126.
10. L. Zhao and D. Lie, *Is Hardware More Secure Than Software?*, IEEE Security & Privacy, 18(5) (2020) 8-17.
11. S. Soni, Himani Agrawal, dan Dr. (Mrs.) Monisha Sharma, *Analysis and Comparison between AES and DES Cryptographic Algorithm*, IJEIT, 2(6) (2012) 17-20.
12. N. G. Augoestien and A. E. Putra, *Purwarupa perangkat keras untuk eksekusi algoritma aes berbasis fpga (Hardware Prototype for AES Algorithm Based on FPGA)*, IJEIS, 5(2) (2015) 211–220.
13. M. Sleem, Yousra Alakabani, Ali Rashed and Attif Ibraheem, *Low Power Implementation of AES Mix Columns/ Inverse Mix Column on FPGA*, Adv. Mat. Res., Trans Tech Publications, Ltd, 677 (2013) 311-316..
14. S. Ghaznavi, Catherine Geboty and Reoven Elbaz, *Efficient technique for the FPGA implementation of the AES Mix columns Transformation, International conference on Reconfigurable Computing and FPGAs*, (2009) pp.219-224.
15. V. Fischer, Milos Drutarovsky and Pawel Chodowiec, *InvMixcolumn Decomposition and Multilevel Resource Sharing in AES Implementation*, in IEEE Trans. On VLSI Systems, 13(8) (2005) 989-992.
16. A. Berent, *Advanced Encryption Standard by Example*, Document available at URL https://www.adamberent.com/wp-content/uploads/2019/02/AESbyExample.pdf Accessed: May 2021.
17. H. Zodpe and A. Sapkal, *An efficient AES implementation using FPGA with enhanced security features*, J. King Saud Univ., 32(2) (2020). 115-122.
18. Q. Zhang, Qunding, *Digital Image Encryption Based On Advanced Encryption Standard (AES) Algorithm*, 2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control, (2015) 1218-1221.
19. Y. J. Liand and W. L. Wu, *Improved Integral Attacks on Rijndael C*, J. Inf. Sci. Eng., 27(6) (2011) 2031- 2045.
20. Y. W. Zhu,H. Q. Zhang, Y. B. Bao, *Study of the AES Realization Method on the Reconfigurable Hardware C*, 2013 International Conference on Computer Sciences and Applications, (2013) 72-76.
21. J. Toldinas, V. Stuikys, R. Damasevicius, G. Ziberkas and M. Banionis, *Energy Efficiency Comparion with Cipher Strength of AES and Rijndael Cryptographic Algorithms in Moble Devices*, J. Elektonika IR Elektrotechnika, 108(2) (2011) 11-14.
22. F. Wegener, L.D. Meyer and A. Moradi, *Spin Me Right Round Rotational Symmetry for FPGA-Specific AES: Extended Version*, J. Cryptol., 33 (2020) 1114-1155.
23. M. Kumar, Senthil and DR. S. Rajalakshmi, *High Efficient Modified MixColumns in Advanced Encryption Standard using Vedic Multiplier*,

International Conference on Current Trends in Engineering and Technology (ICCTET), (2014) 462-466.

24. N.E. Abraham and Tibin Thomas, *FPGA Implementation of Mix and Inverse Mix Column for AES Algorithm*, IJSRD, 1(9) (2013) 2321-0613.

25. C. Nalini, P.V. Anandmohan, D.V. Poornaiah and V.D. Kulkami, *Compact Designs of SubBytes and MixColumn for AES*, IEEE International Advance Conputing Conference, (2009) 1241-1247.

26. Y. Kurniawan & M. A. Rizqulloh, *Block cipher four implementation on field programmable gate array*, Commun.Sci. Technol, *5*(2) (2020) 53-64.