Implementation of 128-bit AES Algorithm using Xilinx System Generator

Atryee Bhuyan[1], Manisha Das[2], Abhishek Tamuli[3], Subham Chakrabarty[4] and Smita Sarma[5]

Department of Electronics & Tele-Communication Engineering

Assam Engineering College Guwahati, India

E-mail:- *atreyeebhuyan@gmail.com[1],subham28896@gmail.com[2], mdmanisha385@gmail.com[3] tamuliabhishek@gmail.com[4],smita@aec.ac.in[5]*

DOI: - https://doi.org/10.47531/MANTECH/ECC.2021.22

Abstract

The incredible digital transformation of everyday activities in the Covid-19 pandemic has simultaneously increased the demand for data security and user privacy. Cybersecurity has become a major concern in our country when cashless transactions are at an all-time high along with some other important sectors such as education, work, agriculture, health or entertainment. The cryptographic algorithm known as the "Advanced Encryption Standard (AES)" algorithm is one of the most commonly used symmetric key cryptographic algorithms, where more randomization in secret keys increases the security as well as the complexity of the algorithm. AES is a linear, pipelined and symmetric block cipher where a combination of key generation and S-Box operations make the attackers exhaustive to search for the right one to crack. In this work, a system generator by Xilinx is used to implement 128-bit AES cryptographic algorithm. The synthesis results show the utilization of 121 slice registers and 2.81% LUTs.

Keywords: *- Bandgap, PTAT, CTAT, opamp. AES (Advanced Encryption Standard), Cryptography, LUT (Look-up Table).*

INTRODUCTION

The "Digital India" campaign aims at improving the internet infrastructure in rural and urban areas, thereby making the country digitally enriched in the field of science and technology, and it has ramped up in the period of the Covid-19 pandemic. With an exponential increase in cashless transactions and the adoption of different modes of internet-enabled payment options, the need for cybersecurity and user privacy has been simultaneously ramped up. To avoid unauthorized users from accessing the personal data of a user, various Cryptography algorithms are being practiced. Cryptography is the technique to establish safe and secure communication in the presence of any third parties called 'adversaries. It is the method of protecting the information and communication through encoded data so that the data can be read and processed by only those for whom the information is intended [1]. The word cryptography comes from its Greek origin. The prefix 'crypt' stands for secret, and the suffix 'graphy' means field of study. Encryption is most frequently used in various digital payment methods or transactions to protect important data such as account information, transaction amount or any other privacy concern of a particular user from third parties. Two types of cryptography techniques are available depending on the symmetry of the 'key', namely, symmetric key and asymmetric key cryptography. Key is the core security element in all these algorithms known only to the receiver and the transmitter, and it creates combinations of numbers that confuse the attackers to crack the algorithm. In symmetric-key cryptography, the same key is used for both the encryption and decryption process. AES was published as Federal Information Processing Standards FIPS 197 standard in 2001, and it is based on the Rijndael algorithm that allows symmetric key encryption. It is a block cipher, where block size, length of the key as well as the round number can be varied [1-2]. Data Encryption Standard (DES) algorithm was published as FIPS 46 standard in 1977 and continued till 2005. This algorithm had a fixed key length of 56-bit [3]. It had the disadvantage of fixed key length and was experimentally found to be less resistant against recognized attacks than the AES standard. AES enables the designer to

select a key of 128-bit or 192-bit or 256-bit, giving more choices for key size and larger message bits according to the design requirements [4].

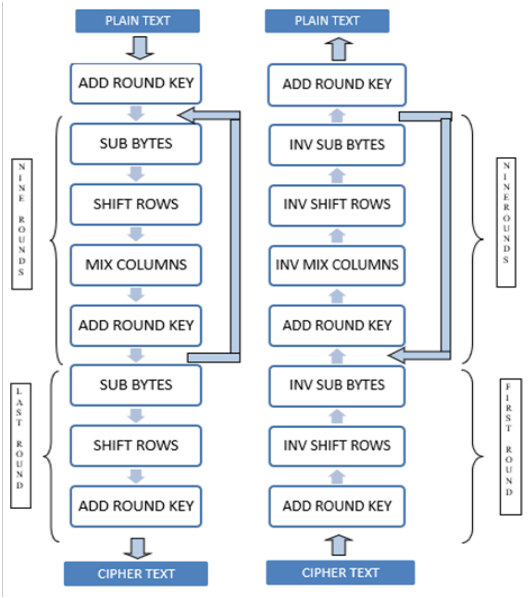

Figure 1: Basic architecture of AES algorithm

AES comes up with a simple design and speed compactness. AES algorithm is suitable to implement because the components, as well as design principles, are completely specified. AES is formulated on linear transformation, and the adoption of the Rijndael algorithm allows the designer to select different block and key sizes [5]. Network security usually depends on the type of algorithm used for encryption, how fast it will take an attacker to find the key, and how long it will take to crack. The more is the key size, the better is its security. AES 256-bit is unbreakable and the most secured one, but even 128-bits brute-forcing is futile [6]. Reconfigurable computation has become a major area of interest among designers. Implementation of different cryptographic algorithms is being carried out using reconfigurable platforms in different work to achieve high speed, minimum area and low power designs. A low power AES algorithm is proposed in [7], which efficiently reduces the overall power consumption of the design by using MLUT based S-Box instead of traditional S-Box. Another low power, high speed and high throughput VLSI architecture are proposed in [8] with 256-bit keys and dual-stage design. Area minimization is carried out in the design proposed in [9], and an overall comparison between various cryptographic algorithms is made based on different parameters in [10].

The paper is organized as follows. Section II and Section III gives an overview of the mathematical operations of the encryption and the decryption stages, respectively. In Section IV, implementation using Xilinx System Generator is explained. The results and analysis are presented in Section IV, followed by the conclusion in section V.

ENCRYPTION OF PLAINTEXT

According to the AES standard, this algorithm can accept 128-bits of the block, and the designer has the freedom to select a key of 128-bit, 192-bit or 256-bit. The number of rounds varies according to the key size. Fig. 1 shows the basic architecture of the AES algorithm. The block diagram is processed in two steps- Encryption and Decryption. In the encryption block, the plain text given as input is converted into a ciphertext with a key. In the decryption block, the ciphertext obtained from the encryption block is converted into plain text so that the user can extract it in a readable form.

A. Byte substituition

There is a concept of S-Box while substituting the bytes, which is a fixed table, and the 16 input bytes are substituted by using this S-Box. The substituted 16 bytes are arranged in a matrix with 4 rows and 4 columns. The S-Box consists of all possible combination of the 8-bits sequence. The static S-Box gets easily exploited, so the Rijndael S-Box is designed to make it resistant to linear and differential cryptanalysis. The values of S-Box are easily available and as shown in Fig. 2.

B. Shift row

In the encryption side of the algorithm, the rows of the matrix generated from byte substitution are cyclically shifted to the left. Any entry that is dropped off is inserted again to the right side. The resulting matrix consists of the same 16 bytes but at different positions. The output matrix is then processed to the next stage, i.e., Mix Columns.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	D7	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	G7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	CA	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	E4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure 2: S-Box values used in the encyption stage

C. Mix column

In the mix column operation, each column of four bytes of the matrix is transformed using a special arithmetic function of Galois field multiplication. This encryption standard uses Galois Field (GF) 28. It employs reducing polynomial for multiplication.

For example: $\{63\}_{16} \cdot \{02\}_{16} = \{C6\}_{16}$

$=> (x^6 + x^5 + x^1 + x^0)(x^1) = (x^7 + x^6 + x^2 + x^1)$

For matrices, Galois Field multiplication is similar to the normal matrix multiplication, but there is one difference, that is, the additional step is replaced by the XOR operation.

D. Add round key

The mix column stage generates a matrix of 16 bytes, and it is considered to be 128 bits combined into 16 inputs of the matrix. In the round key stage, bitwise XOR operation is performed between 128 bits of state and round key of 128 bits. If the result belongs to the last round, then the output is considered to be the ciphertext. Otherwise, the resulting 128 bits are considered as 16 bytes. If the result does not belong to the last round, another round starts with a new substitution process.

DECRYPTION OF CIPHERTEXT

The AES encryption block contains a series of stages with the complex mathematical operation, and finally, it generates the ciphertext. The ciphertext contains all the stages of encryption in reverse order, just like a mirror format, as shown in Fig. 1. The first stage of AES decryption is Inverse Initial Add Round Key. The output of the Round-1 of decryption goes to the Round-10 of encryption, and this goes for all the rounds simultaneously. AES is a type of symmetric cryptography; therefore same keys are used in the decryption part, but in reverse order.

A. Inverse shift rows

In this stage, each row from the matrix generated by the byte substitution is cyclically right shifted. Similar to the shift row operation in encryption, any entry that is dropped off is inserted again to the right side. The resultant matrix consists of the same 16 bytes but at different positions.

B. Inverse byte substituition

The 16 input bytes are substituted by using inverse S-Box. Fig. 3 shows the values of the inverse S-Box of the algorithm. The resultant 16 bytes are arranged in a matrix with 4 rows and 4 columns. The inverse S-Box consists of all possible combination of 8 bits sequence. Transformation in the inverse mixed column is done by using the polynomial of degree less than 4 over Galois field (GF) 28, in which the coefficients are the elements from the column of the state. This is the inverse of the mixed columns in encryption mode but using a different state matrix.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	C8	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	55	D6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	EB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	EC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	6B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Figure 3: S-box values in the decryption stage

C. Add round key

Add Round Key operation is carried out similar to that done in encryption stage. It has its own inverse function since XOR function its own inverse, and the round key must be selected in reverse order.

D. Key generation modulo

The key expansion modulo generates 128-bit keys for each round of AES, based on the initial 128-bit key message. The key expansion modulo contains Sub Bytes, Shift Rows and Round Constant function. Sub Bytes and Shift Rows are the same as that in the encryption stage, but for the Round Constant function, a bitwise XOR operation is performed using a constant array of four bytes. It is given by, [xi-1, 00, 00, 00], where i = 0 to 10. Also, xi-1 represents the powers of x in the field of GF (28). Thus, each round is generated column-wise.

IMPLEMENTATION USING XILINX SYSGEN

The implementation of the AES algorithm is done using Xilinx Block-Set in Simulink. Xilinx System Generator is used to implement the various mathematical blocks of the design with an aim to achieve an efficient, fast, and secure AES algorithm. The System Generator is an additional block set plugged into the MATLAB toolbox that enables the designers to develop high-performance Digital Signal Processor (DSP) architectures. The aim is to emphasize on rapid prototyping of FPGA design without having sufficient knowledge of HDL languages such as Verilog and VHDL. For the design, 128 bits key size is taken, for which the number of rounds is ‘ten’. The use of constant

blocks for 'keys' reduce the number of slices and make the design faster than the traditional approach.

A. Implementation of Encryption

The Sub Byte Substitution is implemented by using ROM block, which stores the address of all the 256 values of S-Box, and the actual values are embedded in the MATLAB command window in a Look-up Table. The Shift Rows implementation is done by changing the positions of the output ports according to the theoretical insight. The third step, i.e., Mix Columns, contains two parts- the multiplier and the XOR operation. The basic multiplier design for encryption uses concatenation, slice registers, multiplexers, constants and logical blocks, as shown in Fig. 4. The design is implemented in such a way that for one input, there are three simultaneous multiplication operations. A subsystem is created for the basic design and is implemented for 16 inputs, which basically allows 128-bits of input. The XOR operation part is implemented by following the second part of Galois field matrix multiplication. All these above steps constitute a Round in encryption, respectively, along with the Add Round Key, which is implemented using only XOR logical blocks having a one-byte message and one-byte key as single input. Similarly, for 16 inputs, 16 XOR logical blocks are used.

Accordingly, ten rounds are implemented with all the four steps referring to the structure of AES as shown in Fig. 1, and the output of the last round gives the ciphertext. The plain text to the main encryption block is given through a code, set to a MATLAB function code block. The input is a 128-bit long ASCII sentence that is converted to decimal values of 16-bytes. Subsequent Keys are generated using MATLAB code.

B. Implementation of Decryption

Decryption means retrieving the input data in its original form. After the ciphertext is generated from encryption, the decryption contains all the stages in encryption in reverse order. Decryption starts with the inverse initial round operation. The matrix obtained after the Inverse Initial Add Round operation is then processed in the next stages, as mentioned in Fig. 1.

The ciphertext of 128-bits is processed through 10 inverse rounds to get the plaintext. Add round key has its own inverse function since the XOR function work in both ways, and the round keys are selected in reverse order. The implementation of Add Round Key in decryption is the same as that in encryption. The inverse Shift Row is implemented exactly in the same way as the shift rows stage, i.e. shifting of output ports' position, but in the opposite direction.

The inverse S-Box values of the AES algorithm are added in the ROM block of the system generator and are embedded in MATLAB Command Window in the format of LUT as well. In the next stage, the inverse Mix Columns is implemented by using inverse Galois matrix multiplication and the basic multiplier block design is shown in Fig. 5.

Further, hardware Co-Simulation Block is created, which determines the results through the hardware, if used. For performance analysis, this work aims only the software simulation and a third-party synthesis tool Xilinx Vivado HLS is used.

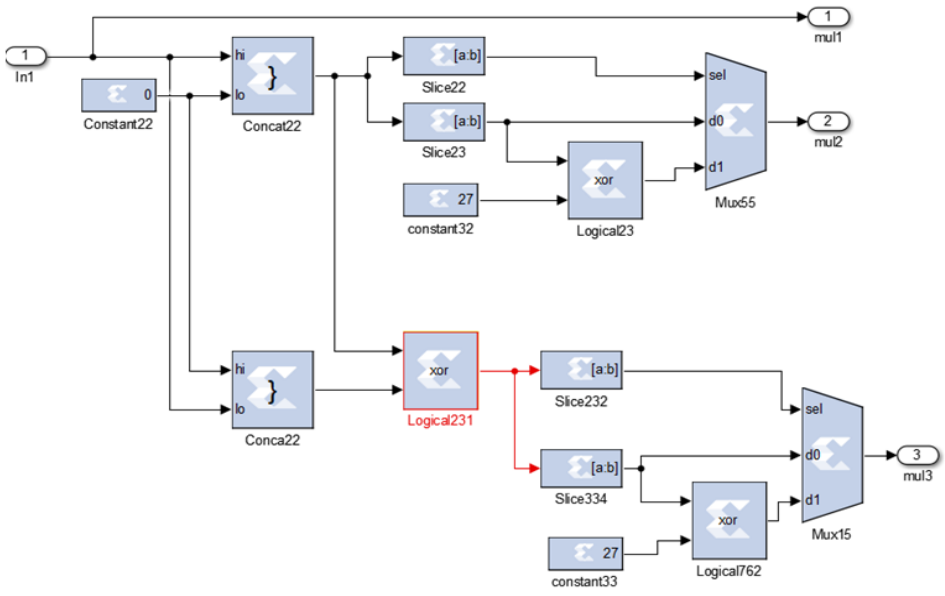

Figure 4: One Multiplication block of mix columns using System Generator

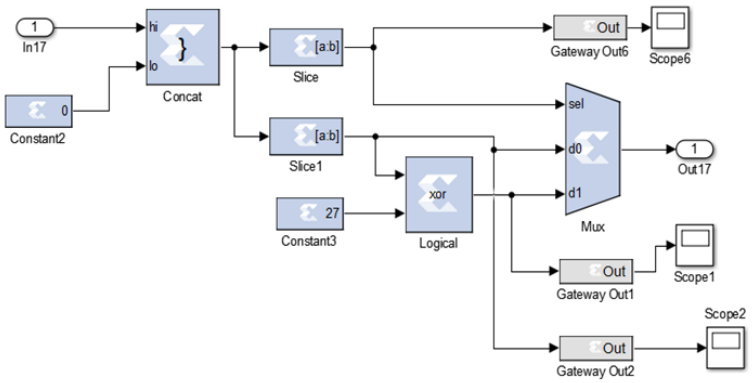

Figure 5: Multiplier design for one input in Decryption stage using Sytem Generator

RESULT AND SIMULATION

Simulation results are error-free and efficient outputs are obtained in the system generator. Multiple text messages and random signals are encrypted using a key into ciphertext and then successfully decrypted into their original message signals. Table I shows the design utilization summary of the circuit. The synthesis result based on the synthesis software shows that the system utilizes 121 slice registers, and only 2.81% LUTs are used from available resources of LUTs. The utilization of LUTs is less, compared to 7% in [1]. Fig. 7 gives the graphical details of the resource utilization by various resources in percentage. The Mixed Mode Clock Manager (MMCM) utilizes around 25% of the available resources, and Block RAM utilizes around 12% of available embedded on-chip memory. The Dynamic power contributes around 54% to the overall power consumption. The graphical representation of the dynamic power consumption summary received from software simulation is shown in Fig. 8. Mixed Mode Clock Manager consumes 85% of the total dynamic power, and 4% of total dynamic power is contributed by clocks. Logic and IO each consumes only 1% of the total dynamic power. To achieve a low power circuit, the overall dynamic power can be minimized by applying various clock power optimization techniques.

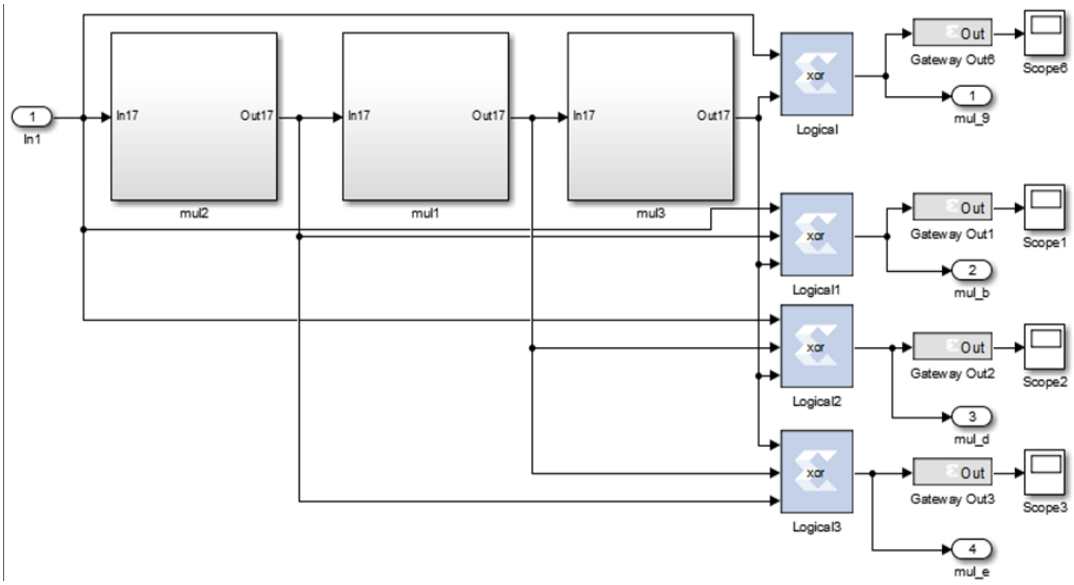

Fig. 6: Inverse Galois filed multiplexing using System Generator

Table I: Resource Utilization Summary

RESOURCE	UTILIZATION	AVAILABLE	UTILIZATION%
LUT	1493	53200	2.81
LUTRAM	5	17400	0.03
FF	2268	106400	2.13
BRAM	18	140	12.86
IO	1	200	0.50
MMCM	1	4	25.00

The Xilinx Vivado IDE (Integrated Design Environment) tools use Static Timing Analysis techniques and provide a timing report after simulation. The summary of the timing report can be used for debugging in case of any timing errors and also to optimize the timing parameters of the design. The worst negative slack (WNS) obtained is 9.815 ns for setup and 0.059 ns for hold, whereas the total negative slack (TNS) obtained is 0 ns. Negative slack comes into the picture when the combinational delay is more than the clock period. From the result obtained, it is clear that the design meets timing.

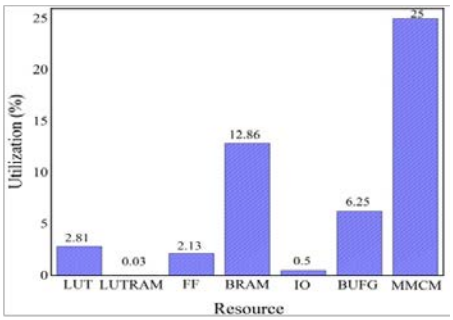

Figure 7: Resource utilization details in percentage

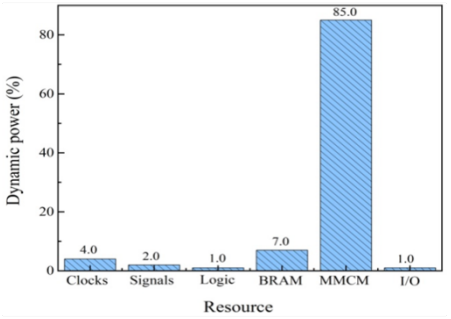

Figure 8: Dynamic power consumption details in percentage

CONCLUSION

The AES algorithm is implemented in Xilinx System Generator for encrypting the plain text and then decrypting the ciphertext. It aims at securing the text message from unauthorized users, thus accomplishing the aim of data security and user privacy. The utilization of LUTs is found to be 2.81% compared to 7% in[1], and it implies that the design is successful in packing the codes into the fewest LUTs possible while maintaining optimized cost. The timing analysis implies that the design meets timing. VHDL codes are developed for key expansion and generation for faster simulation, and signal encryption and decryption is performed successfully. The provision for different messages to be given as input to the system through coding is put forward, so that a voice signal or an image signal can be successfully transmitted and received, maintaining data privacy.

REFERENCES

1. Mulani A. O., and Mane P. B., “High Speed Area-Efficient Implementation of AES Algorithm on Reconfigurable Platform,” Computer and Network Security, DOI: 10.5772/intechopen.82434, June 19, 2019.
2. Mulani A. O., and Mane P. B., “Area optimization of cryptographic algorithm on less dense reconfigurable platform,” IEEE International Conference on Smart Structures and Systems (ICSSS), October 2014.
3. Y. Aruna, and S. N. Shelke, “FPGA Based Implementation of AES Encryption and Decryption with Verilog HDL,” February 2014.
4. P. Deshmukh, “An image encryption and decryption using AES algorithm,” International Journal of Scientific & Engineering Research, Vol. 7, Issue 2, February 2016.
5. Mulani A. O., and Mane P. B., “Watermarking and Cryptography Based Image Authentication on Reconfigurable Platform,” Bulletin of Electrical Engineering and Informatics, Vol. 6, No. 2, June 2017.
6. H. Zodpe, and A. Sapkal, “An efficient implementation of AES algorithm in FPGA using advance key security,” Journal of King Saud University Engineering Sciences.
7. Ratheesh T, and Narayanan S, “FPGA based implementation of AES encryption and decryption with low power multiplexer LUT based S-box,” IOSR Journal of Electronics and Communication Engineering, April 2017, pp.57-61.
8. Kalaiselvi K, and Mangalam H, “ Power efficient and high-performance VLSI architecture for AES algorithm,” Journal of Electrical Systems and Information Technology, September 2015, pp.178-183.
9. N. Shaji, and Bonifus P. L, “Area Optimized Architecture for AES Mix Column Operation,” International Journal of Engineering Research & Technology (IJERT), September-2015.
10. Z. Hercigonja, and D. gimnazija, “Comparative Analysis of Cryptographic Algorithms,” International Journal of Digital Technology and Economy, 2016.
11. A. E, Rabie, and A. Rashed, “Comparative study between DES algorithm, and FRFT for Data Encryption using FPGA,” DOI:10.21608/AUEJ.2019.33702, 2019.
12. https://in.mathworks.com/products/matlab.html
13. https://www.xilinx.com/products/designtools/vivado/integration/sysgen.html