Volume: 12 Issue: VI Month of publication: June 2024

DOI: https://doi.org/10.22214/ijraset.2024.63028

FPGA Implementation of 256-Bit Key AES Algorithm

Dr.P.Ravikiran[1], M.Gnaneshwar[2], Enje Heipsiba[3], Sudesh Srivastav[4]
[1]Associate Professor, [2, 3, 4]B.Tech Student, Department of ECE, CMR College of Engineering & Technology, Hyderabad, Telangana

Abstract: The most popular network security technique for secure data transmission between two end users, particularly over a public network, is the AES algorithm, also known as the Rijndael algorithm. It may be found in all kinds of wired and wireless digital communication networks AES, a network security technique, uses 256 bits on an FPGA. An authorized cryptographic technique that can be used to secure electronic data is the Advanced Encryption Standard technique (AES), which is specified by the National Institute of Standards and Technology. Because they provide faster and more flexible solutions, reprogrammable devices like Field Programmable Arrays (FPGA) are a very appealing alternative for hardware implementation of the encryption algorithm AES. The effective FPGA implementation of advanced encryption standards (AES) is proposed in this research. The encryption code is written in the Verilog programming language. The AES Rijndael algorithm performs better than any other when implemented on FPGA. Other works are covered by this implementation to demonstrate its effectiveness. The block and key sizes in the design are 256, and an iterative looping technique is used. Bits, S-box implementation using lookup tables. This results in a low-complexity design that can readily achieve high throughput and low latency

Keywords: Cryptography, Advanced Encryption Standard(AES),Encryption, Decryption, Verilog, Field Programmable Gate Array (FPGA).

I. INTRODUCTION

A key component of data security is cryptography. We can store confidential data on it or send it over unsecured networks, keeping it safe from prying eyes. The need for safe digital data exchange prompted the development of numerous encryption techniques in enormous volumes. Algorithms that fall into two categories: symmetric algorithms using private key algorithms and asymmetric algorithms using public key algorithms. Electronically, symmetric key algorithms typically run far more quickly than asymmetric key algorithms. The US Government organized a contest in 1997 and the Advanced Encryption Standard (AES) emerged as the victor. The Data Encryption Standard was deemed inadequate due to its tiny key size and the increasing processing capacity of devices. In 1998, fifteen candidates were approved; in 1999, the pool was narrowed to five finalists in response to public feedback. One of these five algorithms—a slightly altered Rijndael algorithm—was chosen as the future standard in October 2000.For the Advanced Encryption Standard (AES), the National Institute of Standards and Technology (NIST) issued a call for suggestions. An encryption technique called the AES is a Federal Information Processing Standard (FIPS) that is used to safeguard electronic data. An example of a symmetric block cipher that can both encrypt and decode data is the AES algorithm. Data is transformed into cipher-text, an incomprehensible format, using encryption. The data is returned to plaintext, or its original form, upon decryption of cipher-text.

The AES algorithm may encrypt and decrypt data in blocks of bits using cryptographic keys with 128-, 192-, and 256-bit lengths. In order to obtain more security and enhance confidentiality, we are using the AES-256 algorithm with a 256-bit key length. This is in contrast to AES-128 and AES-192, which have smaller keys than AES-256. Governments and individuals seeking security from data and signal interception now place a high premium on privacy. The desire for a higher level of security on previously unsafe communications has only increased due to the widespread use of personal communications devices. The primary distinction between symmetric key block ciphers like AES and DES (Data Encryption Standard) is the bit length of the key (56 bits for DES). Since both the sender and the recipient utilize the same key in these symmetric key encryption techniques, public keying does not require a verification server. Symmetric keying promotes greater system interoperability by enabling operation without the need for an open network.

II. LITERATURE SURVEY

1) AES-128 cipher. high-speed, low-cost fpga implementation

The Rijndael Algorithm, which replaced DES in October 2000, is known for its fast throughput, low latency, and area optimization for parallel encryption. These goals balance area and speed. AES, integral to the 802.11 standard, is robust but demands additional processing power, often unsupported by older hardware. A high-speed, small-area FPGA implementation can offload processing from the main processor. Implemented on a Xilinx FPGA, the 10-round AES architecture aims for a high-speed, low-area core cipher. This design optimizes hardware and reduces clock cycles compared to the original. Functions related to AES transformations were developed independently and integrated. The 128-bit block encryption takes 200 nanoseconds, utilizing 50% of the FPGA device.

2) Implementation of AES algorithm using VHDL

Cryptography, the science of converting messages into an unrecognizable form using a secret key, protects information from theft. This critical data security measure, in use since 1976, ensures secure information transfer. Changing secret keys enhances security and complexity but increases memory usage and execution time. In November 2001, NIST selected the Advanced Encryption Standard (AES), a symmetric key cipher that uses the same key for both encryption and decryption of 128-bit data blocks. Encryption transforms plain text into cipher text, while decryption reverses this process. Field Programmable Gate Arrays (FPGAs), known for their reconfigurability, affordability, and speed, are ideal for implementing the 128-bit AES algorithm. This paper proposes an FPGA implementation of AES, requiring an understanding of the AES algorithm and coding it in VHDL.

III. THE AES ALGORITHM

The AES algorithm belongs to the iterative, symmetric block cipher class. Given that the same key is used for both the encryption and decryption procedures, it is symmetric. Because each data block has a fixed length of 128 bits and the encryption key has changeable key lengths that can be independently chosen as 128, 192, or 256 bits, it is a block cipher. This means that three alternative key lengths can be employed with this technique, producing three separate forms called AES-128, AES-192, and AES-256. Because this algorithm's steps are repeated multiple times, it is iterative. Another name for these iterations is rounds. The size of the key determines how many rounds or iterations are used in the encryption and decryption operations overall utilized. The link between the key length and the total number of rounds is shown in Table I. The 128-bit data block is divided into 16 bytes, which are then mapped into the State, a size 4 X 4 array.

A 4x4 matrix with 8 bits for each element is created using the 128-bit data. The four operations are carried out on a column basis; therefore, we turn the 128-bit data into a 4x4 matrix with 8 bits for each member.

Tabel 1: Data Structure of 128-bit Message

A_0	A_1	A_2	A_3
A_4	A_5	A_6	A_7
A_8	A_9	A_{10}	A_{11}
A_{12}	A_{13}	A_{14}	A_{15}

Tabel 2:RELATIONSHIP BETWEEN KEY LENGTH AND TOTAL NUMBER OF ROUNDS

AES	Key Length (Nk^a Words)	Block Size (Nb^a Words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

AES ENCRYPTION AND DECRYPTION ALGORITHM

The three primary components of the AES-256 algorithm are the Key Expansion, Inverse Cipher, and Cipher. An inverse cipher returns data to its original form, known as plaintext, whereas a cipher transforms it into an unintelligible form known as cipher text. A Key Schedule is produced by Key Expansion and is utilized in the Cipher and Inverse Cipher procedures. The Cipher and Inverse Cipher consist of a predetermined quantity of rounds. The AES method employs a round function for its Cipher and Inverse Cipher that is made up of four distinct byte-oriented transformations:

1) Using a substitution table (S-box) for byte replacement
2) Modifying the State array's rows by various offsets
3) Combining the information in every State array column.
4) Including the State in a Round Key

To create a simple Inverse Cipher for the AES algorithm, the Cipher modifications can be inverted and then applied in reverse order. The specific modifications applied in the Inverse Cipher.

- Inverse Shift Rows
- Inverse Sub Bytes
- Inverse Mix Columns
- Add Round Key

The key expansion module, key reversal buffer, start permutation module, round permutation module, and final permutation module make up the core of the AES inverse encryption. The keys for every round are first stored in the key reversal buffer, which then presents them to the rounds in reverse order. For 256-bit keys, the round permutation module will execute 14 iterations in a continuous loop.

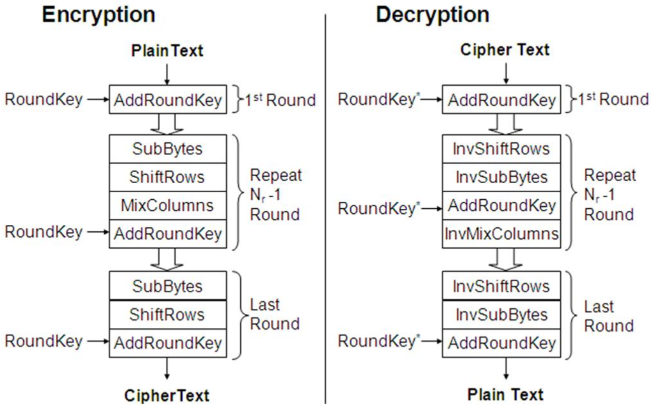


Fig 1: encryption block and decryption block

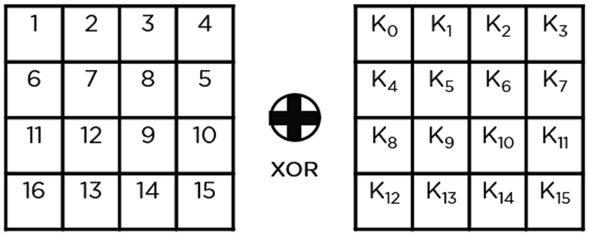
IMPLEMENTATION OF AES 256

A 256-bit key and 128-bit plaintext are used in the encryption procedure. There are fourteen cycles in AES 256. All five operations—preround, sub byte, shift rows, mix columns, and add round key operations—are included in the first round. Four operations—sub byte, shift rows, mix columns, and add round key operations—are included in the rounds two through thirteen. Additionally, there are three operations in the fourteenth and last round: subbyte, shift rows, and add round key operations.

1) Key Expansion round keys are derived from the cipher key using Rijndael's key schedule.
2) Add Round Key each byte of the state is combined with the round key using bitwise xor.
3) Sub Byte a non-linear substitution step where each byte is replaced with another according to a lookup table.
4) Shift Rows a transposition step where each row of the state is shifted cyclically a certain number of steps.
5) Mix Columns is a mixing operation which operates on the columns of the state, combining the four bytes in each column. 6. Add Round Key

- Add Round Key: In AES encryption, the AddRoundKey step is pivotal, serving as a crucial link between the plaintext and the encryption key. During this stage, each byte of the current block undergoes a bitwise XOR operation with a corresponding byte from the round key. This process ensures that the input data is mixed with the encryption key, introducing a key-dependent transformation that enhances the security of the algorithm. By incorporating the key at each round, AES achieves confusion and diffusion, crucial properties for robust encryption.

Through successive rounds of AddRoundKey, the original plaintext is obfuscated, making it increasingly resistant to cryptographic attacks. Ultimately, the AddRoundKey operation fortifies AES by integrating key material into the encryption process, contributing significantly to its strength and resilience against adversarial threats.



1	2	3	4
6	7	8	5
11	12	9	10
16	13	14	15

K_0	K_1	K_2	K_3
K_4	K_5	K_6	K_7
K_8	K_9	K_{10}	K_{11}
K_{12}	K_{13}	K_{14}	K_{15}

Fig 2:Add round key transformation

- Sub-Bytes: In the AES (Advanced Encryption Standard) algorithm, the Sub Bytes transformation operates on each byte of the input data independently. It replaces each byte with another byte from a fixed substitution table called the S-box. This substitution provides non-linearity to the encryption process, enhancing its resistance against cryptographic attacks. The S-box is designed to have desirable cryptographic properties, such as confusion and diffusion. Sub Bytes contributes to confusion by making the relationship between the plaintext and the ciphertext complex, making it harder for attackers to analyze patterns. Additionally, it contributes to diffusion by spreading the influence of each input byte across multiple output bytes, further obscuring any patterns in the data. Overall, the Sub Bytes step is essential for achieving strong encryption in the AES algorithm by introducing non-linearity and enhancing the security of the cipher.

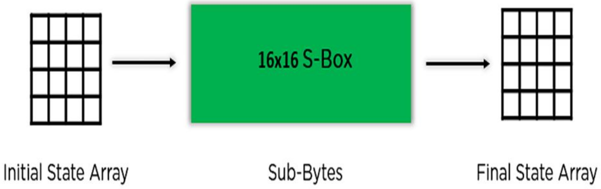
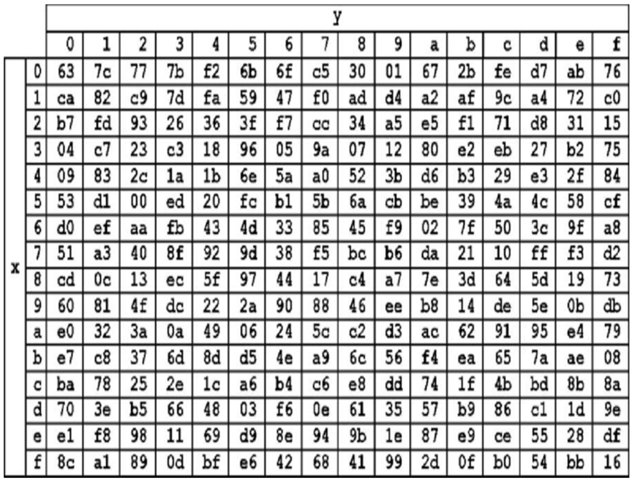


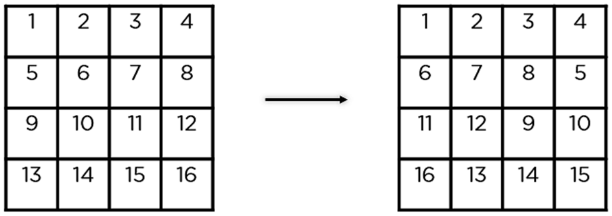
Fig 3:Sub bytes transformation



		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	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	c7	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	c4	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	f4	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

Fig 4: S-box

- Shift Rows: It swaps the row elements among each other. It skips the first row. It shifts the elements in the second row, one position to the left. It also shifts the elements from the third row two consecutive positions to the left, and it shifts the last row three positions to the left.

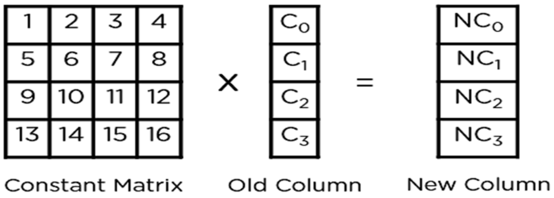


1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	2	3	4
6	7	8	5
11	12	9	10
16	13	14	15

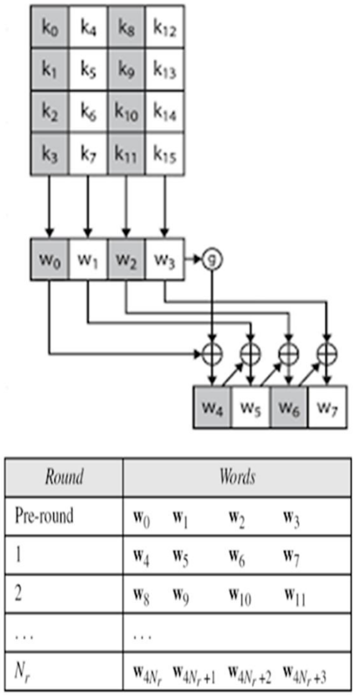
Fig 5: shift row transformation

- Mix Columns: The process involves multiplying a constant matrix with each state array column to create new columns for the next step, which is not done in the final round.



$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \times \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} NC_0 \\ NC_1 \\ NC_2 \\ NC_3 \end{bmatrix}$$

- Key expansion: A four-word round key is acceptable for the initial stage. The initial stage is Add Round Key and each of the 14 rounds of cipher. So, total 14 rounds = 60words. The key is added into the first four words of the expanded key.



Round	Words			
Pre-round	w_0	w_1	w_2	w_3
1	w_4	w_5	w_6	w_7
2	w_8	w_9	w_{10}	w_{11}
. . .	. . .			
N_r	w_{4N_r}	w_{4N_r+1}	w_{4N_r+2}	w_{4N_r+3}

Fig 7:key expansion transformation

The function g defines the complex and that function consist of some sub function as follow:
Step-1: This function performs the one-byte circular left shift.
Step-2: Using S-box each sub word performs a byte substitution.
Step-3: Finally result of Rot word and step 2 is XORed with the round constant called as Rcon [Round].

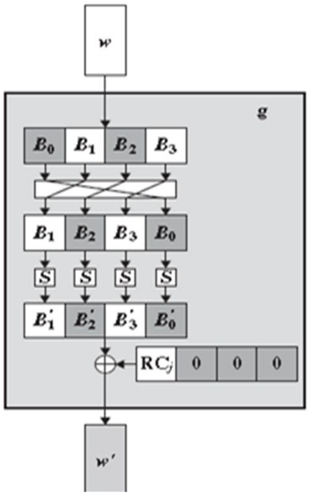
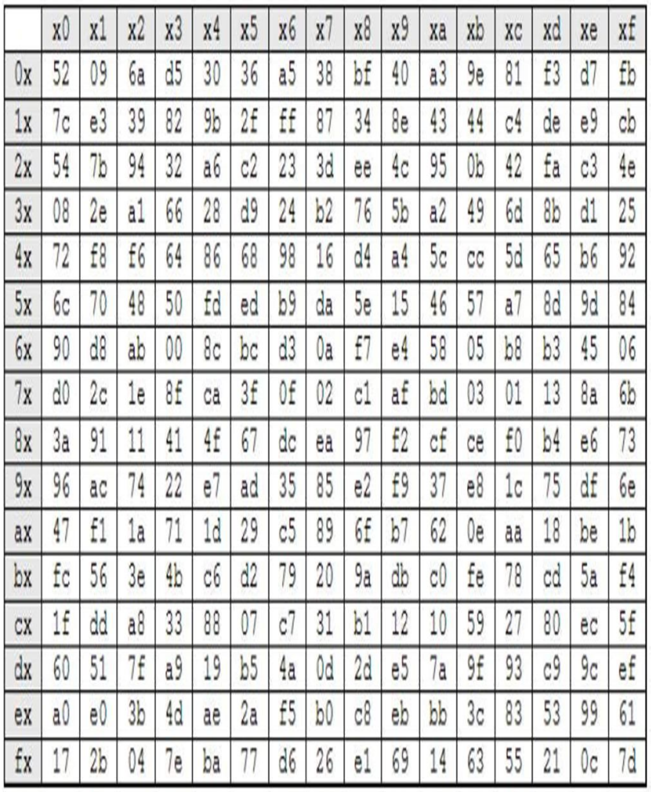


Fig 8: g function module

- AES Decryption: The opposite of AES encryption is the AES decryption process. The input for decryption is the cipher text, which is the result of the encryption process. Another input uses the same 256-bit input key. The key generation module receives the input key and uses it to create new keys, just like we do during encryption. The resulting output keys are used as inputs into inverse mix columns, which provide keys for the decryption process's 14 rounds. Presently, all five operations—preround operation, subbyte, shift rows, mix columns, and add round key operations—are performed in the first round. Four operations—sub byte, shift rows, mix columns, and add round key operations—are used from the second to the thirteenth round. Additionally, there are three operations in the final 14th round: subbyte, shift rows, and add round key operations. Similar to the encryption process, the results of the first round are used as the input for the following rounds, up to the fourteenth, when the results of the last round are used as the final output.

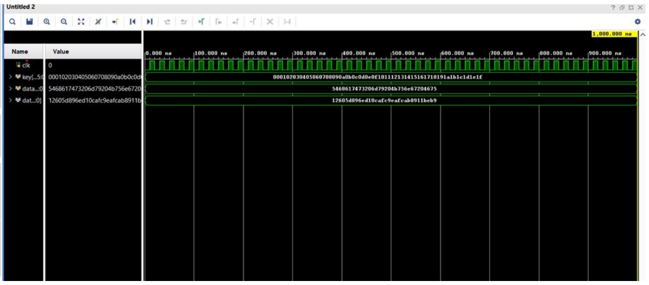


	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1x	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2x	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3x	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4x	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5x	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6x	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7x	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8x	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9x	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
ax	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
bx	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
cx	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
dx	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
ex	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
fx	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Fig 9: Inverse s-box

IV. RESULTS

1) Encryption Process

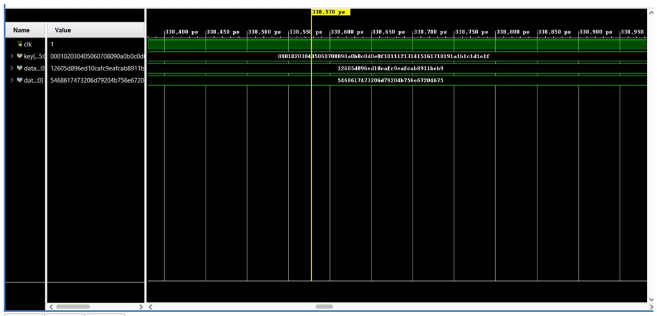


```
Name        Value
clk         0
key[...5:0  000102030405060708090a0b0c0d
data...:0   5468617473206d79204b756e6720
dat...:0]   12605d896ed10cafc9eafcab8911b
```

```
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
5468617473206d79204b756e67204675
12605d896ed10cafc9eafcab8911beb9
```

Fig 10: Simulation results of encryption

2) Decryption Process



```
Name        Value
clk         1
key[...5:0  000102030405060708090a0b0c0d
data...:0   12605d896ed10cafc9eafcab8911b
dat...:0]   5468617473206d79204b756e6720
```

```
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
12605d896ed10cafc9eafcab8911beb9
5468617473206d79204b756e67204675
```

Fig 11: Simulation results of decryption

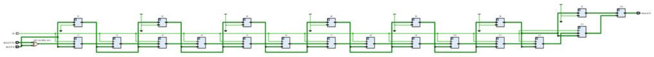


Fig 12:synthesis of encryption

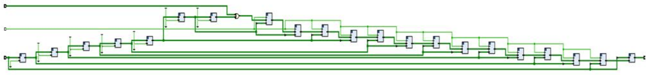
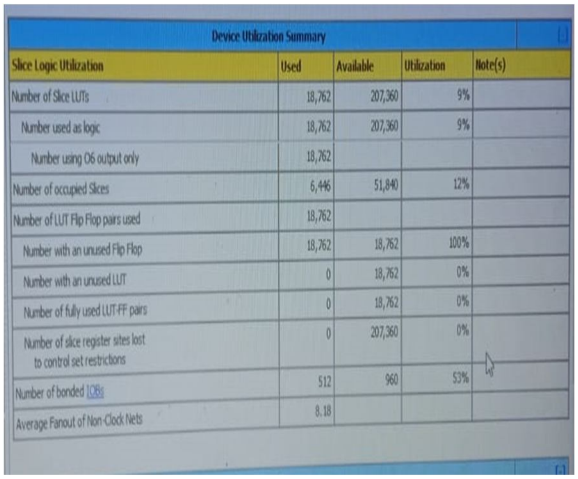


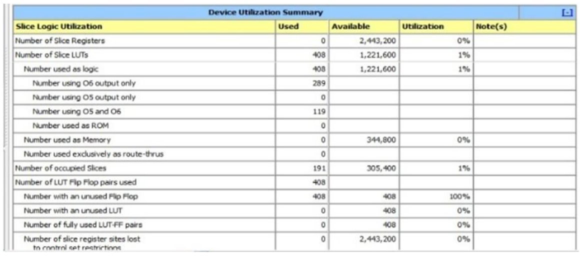
Fig 13:Synthesis of Decryption



Device Utilization Summary

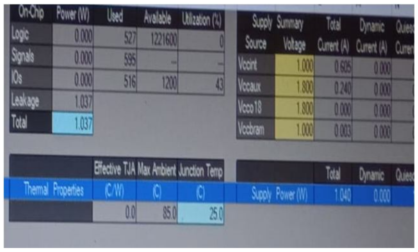
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice LUTs	18,762	207,360	9%	
Number used as logic	18,762	207,360	9%	
Number using O6 output only	18,762			
Number of occupied Slices	6,446	51,840	12%	
Number of LUT Flip Flop pairs used	18,762			
Number with an unused Flip Flop	18,762	18,762	100%	
Number with an unused LUT	0	18,762	0%	
Number of fully used LUT-FF pairs	0	18,762	0%	
Number of slice register sites lost to control set restrictions	0	207,360	0%	
Number of bonded IOBs	512	960	53%	
Average Fanout of Non-Clock Nets	8.18			

Fig 14: Design utilization Summary of encryption



Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	0	2,443,200	0%	
Number of Slice LUTs	408	1,221,600	1%	
Number used as logic	408	1,221,600	1%	
Number using O6 output only	289			
Number using O5 output only	0			
Number using O5 and O6	119			
Number used as ROM	0			
Number used as Memory	0	344,800	0%	
Number used exclusively as route-thrus	0			
Number of occupied Slices	191	305,400	1%	
Number of LUT Flip Flop pairs used	408			
Number with an unused Flip Flop	408	408	100%	
Number with an unused LUT	0	408	0%	
Number of fully used LUT-FF pairs	0	408	0%	
Number of slice register sites lost to control set restrictions	0	2,443,200	0%	

Fig 15: Design utilization Summary of decryption



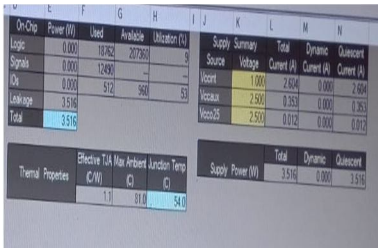
On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	527	1221600	0
Signals	0.000	595	--	--
IOs	0.000	516	1200	43
Leakage	1.037			
Total	1.037			

Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)
Vccint	1.000	0.605	0.000
Vccaux	1.800	0.240	0.000
Vcco18	1.800	0.000	0.000
Vccbram	1.000	0.003	0.000

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	0.0	85.0	25.0

	Total	Dynamic
Supply Power (W)	1.040	0.000

Fig 15: Power report for encryption



On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	18762	207360	9
Signals	0.000	12490	--	--
IOs	0.000	512	960	53
Leakage	3.516			
Total	3.516			

Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	2.604	0.000	2.604
Vccaux	2.500	0.353	0.000	0.353
Vcco25	2.500	0.012	0.000	0.012

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.1	81.0	54.0

	Total	Dynamic	Quiescent
Supply Power (W)	3.516	0.000	3.516

Fig 16: Power report for decryption

Comparison of area utilization summary for AES-128 &192 & 256 Encryption

Parameters	128 bits key used	192 bits key used	256 bits key used	available	utilization
No. of Slice Registers	3760	4435	5356	866400	0%
No. of Fully used LUT-FF Pairs	1465	1831	2039	14831	12%
No. of Bonded IOB's	385	449	512	850	43%
No. of Slice LUT's	10773	12227	18762	207,306	9%
No. of LUT-FF Pairs used	13068	14831	12870		

Comparison of Latency, Throughput, Time for Encryption block

Parameter	Throughput	Latency (Clock Cycles)	Time(ns)
AES-128 Decryption	1.02 Gbps	20	92.5
AES-192 Decryption	1.07Gbps	24	112.5
AES-256 Decryption	1.28 Gbps	28	132.5

Comparison of area utilization summary for AES-128 &192 & 256 Decryption

Parameters	128 bits key used	192 bits key used	256 bits key used	available	utilization
No. of Slice Registers	6088	7286	8768	866400	1%
No. of Fully used LUT-FF Pairs	4479	5468	6543	16849	13%
No. of Bonded IOB's	385	449	512	850	43%
No. of Slice LUT's	15240	17742	19762	433200	9%
No. of LUT-FF Pairs used	16849	19560	20976		

Comparison of Latency, Throughput, Time for Decryption block.

Parameter	Throughput	Latency (Clock Cycles)	Time(ns)

AES-128 Decryption	0.85 Gbps	30	142.5
AES-192 Decryption	0.93Gbps	32	152.5
AES-256 Decryption	1 .03 Gbps	41	197.5

V. CONCLUSION

Using cipher keys of lengths of 128, 192, and 256 bits, the Advanced Encryption Standard algorithm is an iterative private key symmetric block cipher that can handle data blocks of 128 bits. This study presents an effective FPGA implementation of the 128 bit block and 256 bit key AES cryptosystem. For the purpose of implementing 128 bit data encryption and decryption, optimized and synthesizable Verilog ode is produced. The description is confirmed through the use of Xilinx's ISE 9.2i functional simulator. To reduce hardware consumption, an iterative design approach is used to simulate all algorithm modifications.

VI. ACKNOWLEDGMENT

We are very grateful to Dr.P. Ravikiran, Associate Professor in the ECE Department of CMR College of Engineering and Technology for his valuable suggestions and help throughout our project and paper work completion.

REFERENCES

[1] M.Rajeswara Rao and R.K. Sharma, "FPGA implementation of combined s box and inv s box of AES", 2017 4th international conference on signal processing and integrated networks (spin).

[2] M. Liberatori, f. Otero, j. C. Bonadero and j. Castineira, "aes-128 cipher. High-speed, low-cost fpga implementation," 2007 3rd southern conference on programmable logic, mar del plata, argentina, 2007.

[3] Amit Kumar Manoj Kumar P. Balramudu," implementation of AES algorithm using VHDL", proceedings of the IEEE 2017 international conference on computing methodologies and communication (iccmc).

[4] Atul M. Borkar, R. V. Kshirsagar and M. V. Vyawahare, FPGA implementation of AES algorithm.

[5] Xinmiao Zhang and Keshab K. Parhi, "high speed vlsi architectures for the aes algorithm", IEEE, vol. 12, no. 9, september 2004.

[6] Yulin Zhang and Xinggang Wang, "pipelined implementation of AES encryption based on FPGA", IEEE international conference on information theory and information security, 2010.

[7] Vatchara Saicheur and Krerk Piromsopa, "an implementation of aes-128 and aes-512 on apple mobile processor", 14th international conference on electrical engineering/electronics computer telecommunications and information technology (ecti-con), 2017.

[8] "advanced encryption standard (aes)", *fips 197*, November 2001.

[9] Marko mali, Franc novak and Anton Biasizzo, "hardware implementation of aes algorithm", journal of electrical engineering, *vol.* 56, no. 9–10, pp. 265-269, 2005.

[10] N. S. Sai srinivas and MD. Akramuddin, "FPGA based hardware implementation of AES rijndael algorithm for encryption and decryption", international conference on electrical electronics and optimization techniques (iceeot), 2016.

[11] Ashwini M. Deshpande, Mangesh S. Deshpande and Devendra N. Kayatanavar, "FPGA implementation of aes encryption and decryption", international conference on control automation communication and energy conservation, 2009.

[12] J. Nechvatal et al., "Report on the development of Advanced Encryption Standard", NIST publication, Oct 2000

[13] K. Gaj and P. Chodowiec, "Comparison of the hardware performance of the AES candidates using reconfigurable hardware", The Third AES Candidates Conference printed by the National Institute of Standards and Technology.

[14] Behrouz A. Forouzan and Debdeep Mukhopadhyay, "Cryptography and Network Security".

[15] F.X. Standaert, G. Rouvroy and J.D. Legat, "Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware", Improvements and Design Tradeoffs, pp. 334-350, 2003.

[16] Shanxin qu, Guochu Shou and Vihong hu, "high throughput pipelined implementation of aes on fpga" in information engineering and electronic commerce(ieec), ieee press, pp. 542-545, 2009.

[17] Fx. Standaert, G. Rouvoy, JJ. Quisquater, JD. Legat, "efficient implementation of rijndael encryption in reconfigurable hardware: improvements and design tradeoffs". in the proceedings of ches 2003, lecture notes in computer science, vol 2779, pp. 334-350, springer verlag.

[18] M6nica Liberatori, Fernando 0tero, J. C. Bonadero, Jorge castifieira, "aes-128 cipher. high speed, low cost fpga implementation". programmable logic, 2007. spl07. 2007 3rd southern conference on mar. del plata pp. 195-198 feb. 2007.