International Journal of Current Engineering and Technology
ISSN 2277 – 4106
©2013 INPRESSCO. All Rights Reserved
Available at http://inpressco.com/category/ijcet

Research Article

Design of High Speed 128 bit AES Algorithm for Data Encryption

Sumalatha Patil M[a*] and Mala L M[a]

[a]Department of Electronics and Communication Engineering, SDMCET, Dharwad, India

Abstract

With the fast progression of data exchange in electronic way, information security is becoming more important in data storage and transmission. Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication. In data and telecommunications, cryptography is necessary when communicating over any unreliable medium, which includes any network particularly the internet. The Advanced Encryption Standard (AES) is the newly accepted symmetric cryptography standard for transferring block of data securely. The AES algorithm defined by the National Institute of Standard and Technology (NIST) of United States has been widely accepted. AES involves the sequence of four primitive functions: Sub Bytes, Shift Rows, MixColumn and Add Round Key. This paper presents the design of a 128 bit encryptor using AES Rijndael Algorithm for 128 bit data encryption. These designs were described using VerilogHDL. Xilinx ISE 14.2 software is used for synthesis.

Keywords: *AES, Cryptography, Encryption, MixColumn, Key Expansion, S-box, Security, Verilog.*

1. Introduction

Information is significant in every aspect of human life. Like any other property, it needs protection. There are different cryptographic algorithms available to secure information. However, most of them are computationally intensive, either deals with huge numbers and complex mathematics or involves several iterations (G.H Karimian et al, 2011). There are mainly two types of cryptographic algorithms: symmetric and asymmetric algorithms. Symmetric systems such as Data Encryption Standard (DES), 3DES, and Advanced Encryption Standard (AES) use an identical key for the sender and receiver; both to encrypt the plain text and decrypt the cipher text. Asymmetric systems such as Rivest-Shamir- Adelman (RSA) & Elliptic Curve Cryptosystem (ECC) uses different keys for encryption and decryption. Symmetric cryptosystems is more suitable to encrypt large amount of data with high speed. To replace the old Data Encryption Standard, in September 12 of 1997, the National Institute of Standard and Technology (NIST) required proposals to what was called Advanced Encryption Standard (AES).Many algorithms were presented originally with researches from 12 different nations. On October 2nd 2000, NIST has announced the Rijndael Algorithm is the best in security, performance, efficiency, implement ability & flexibility. The Rijndael algorithm was developed by Joan Daemen of Proton World International and Vincent Rijmen of Katholieke University at Leuven (P.Karthigaikumar et al,2011).AES is a simple design, a high speed algorithm, with low memory costs. AES is a symmetric block cipher. The same key is used to encrypt and decrypt the data. The plain text and the cipher text are the same size. AES is an algorithm for performing encryption (and the reverse, decryption) which is a series of well-defined steps that can be followed as a procedure. The original information is known as plaintext, and the encrypted form as cipher text. The cipher text message contains all the information of the plaintext message, but is not in a format readable by a human or computer without the proper mechanism to decrypt it; it should resemble random gibberish to those not intended to read it. The encrypting procedure is varied depending on the key which changes the detailed operation of the algorithm. Without the key, the cipher cannot be used to encrypt or decrypt. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits (K.Rahimunnisa et al, 2012).

A symmetric cryptosystem is shown in Figure.1 and has five ingredients:

a. Plain text: this is the original message or data that fed into the algorithm as input.
b. Encryption algorithm: the algorithm performs various substitutions and transformations on the plaintext.
c. Secret key: this is also an input to the algorithm and its value is independent of the plaintext. The algorithm will produce a different output depending on the specific key.
d. Cipher text: this is the scrambled message produced as output. It depends on the plaintext and the secret key.

*Corresponding author: **Sumalatha Patil M**

e. Decryption algorithm: this is essentially the encryption algorithm run in reverse. It takes the cipher text and the secret key and produces the original plaintext (P.Karthigaikumar et al, 2011).

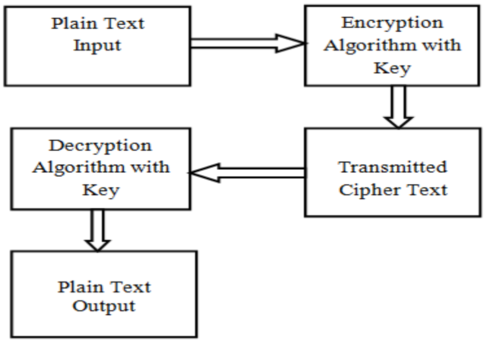


Figure.1 General Block diagram of symmetric cryptosystem

2. Literature Survey

With the use of more handheld wireless devices and increasing networking and wireless data transfer, the issue of security is being addressed from many different directions. The National Institute of Standards and Technology (NIST) selected the Rijndael algorithm as a new Advanced Encryption Standard (AES) (FIPS PUB 197, 2001). This standard was first developed for secure data encryption/decryption for high-end applications. In (K. Rahimunnisa et al, 2012) (G.H Karimian et al, 2011) authors used AES algorithm for a high speed and low power consumption hardware implementation to encrypt the image. For increase speed he used 4 pipeline stages and also for reducing power consumption resource sharing, pipelining and signal gating techniques used. Paper (P.Karthigaikumar et al, 2011) presents the design of a 128 bit encoder using AES Rijndael Algorithm for image encryption to protect the confidential image data from unauthorized access. The paper (Vasamsetti Ramoj et al, 2012) compares the hardware efficiency of different AES implementations with respect to their area, speed and power performance especially in two different styles – one using controller based design can infer that power dissipation is more and the other one is iterative method and can infer that iterative method utilizes less number of hardware units compared to the controller method. In (William Stallings, 2005) authors show how the AES can be programmed in software or built with hardware. Software is used for simulation and optimization of the synthesizable VHDL code and all the transformations of algorithm are simulated using an iterative design approach in order to minimize the hardware consumption. In (Issam Mahdi Hammad, 2010) author introduces new efficient hardware implementations for the Advanced Encryption Standard (AES) algorithm. Two main contributions are presented in this thesis to achieve higher FPGA (Throughput/Area) efficiency comparing to previous loop unrolled designs. The first one is a high speed 128 bits AES encryptor, and the second one is a new 32 bits AES design. In (P. Aatheeswaran et al, 2013) paper mainly focused in implementation of AES encryption and decryption standard AES-128. All the transformations of both Encryption and Decryption are simulated using an iterative design approach in order to minimize the hardware consumption.

3. The Origins of AES

The principal drawback of 3DES (which was recommended in 1999, Federal Information Processing Standard FIPS PUB 46-3 as new standard with 168-bit key) is that the algorithm is relatively sluggish in software. A secondary drawback is the use of 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable. In 1997, National Institute of Standards and Technology NIST issued a call for proposals for a new Advanced Encryption Standard (AES), which should have security strength equal to or better than 3DES, and significantly improved efficiency. In addition, NIST also specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits. In a first round of evaluation, 15 proposed algorithms were accepted. A Second round narrowed to 5 algorithms. NIST completed its evaluation process and published a final standard (FIPS PUB 197) in November, 2001. NIST selected Rijndael as the proposed AES algorithm. The 2 researches of AES are Dr. Joan Daemon and Dr. Vincent Rijmen from Belgium.

1. AES Evaluation

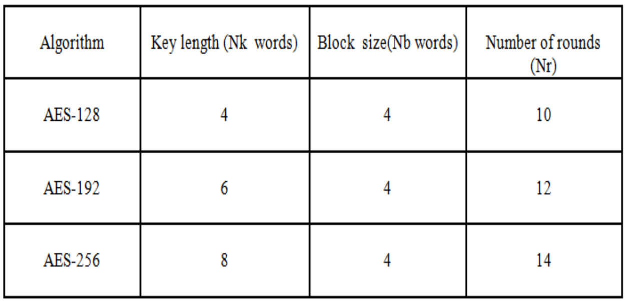
i. Security – 128 minimal key sizes provides enough security.
ii. Cost – AES should have high computational efficiency.
i. Security: This refers to the effort required to crypt analyze an algorithm. The emphasis in the evaluation was on the practicality of the attack. Because the minimum key size for AES is 128 bits, brute-force attacks with current and projected technology were considered impractical. Therefore, the emphasis, with respect to this point, is cryptanalysis other than a brute-force attack.
ii. Cost: NIST intends AES to be practical in a wide range of applications. Accordingly, AES must have high computational efficiency, so as to be usable in high-speed applications, such as broadband links (William Stallings 2005).

2. The AES Cipher

The AES Algorithm is a symmetric-key cipher, in which both the sender and the receiver use a single key for encryption and decryption. The data block length is fixed to be 128 bits, while the length can be 128, 192, or 256 bits. In addition, the AES algorithm is an iterative algorithm. Each iteration can be called a round, and the total number of rounds is 10, 12, or 14, when key length is

128,192, or 256, respectively. The 128 bit data block is divided into 16 bytes. These bytes are mapped to a 4x4 array called the State, and all the internal operations of the AES algorithm are performed on the State (K.Rahimunnisa et al, 2012)(FIPS PUB 197, 2001).

Table I :AES Parameters



Algorithm	Key length (Nk words)	Block size(Nb words)	Number of rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

3. Design Of 128 Bit Encoder

Methodology

The encryption process is iterative in nature. Each iteration is known as rounds. For each round 128 bit input data and 128 bit key is required. That is, need 4 words of key in one round. So the input key must be expanded to the required number of words, which depends upon the number of rounds. The output of each round serves as input of next stage. In AES System, same secret key is used for both encryption and decryption. So it provides simplicity in design.

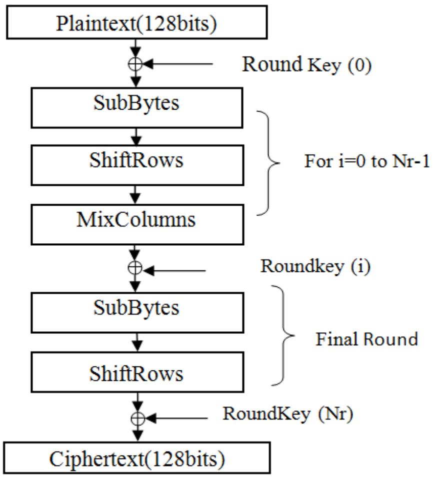
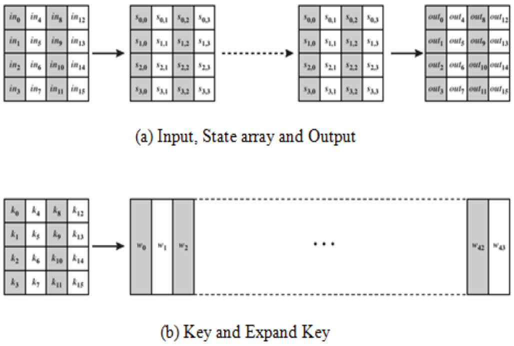


Figure.2 Detailed Block diagram of an encryption

The input to the encryption algorithm is a single 128-bit block, this block, in FIPS PUB 197, is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix. These operations are depicted in Figure.3.a. Similarly, the 128-bit is depicted as a square matrix of bytes. This key is expanded into an array of key schedule words; each word is 4 bytes and the total key schedule is 44 words for the 128-bit key (Figure.3.b). Ordering of bytes within a matrix is by column. Before delving into details, we can make several comments about overall AES structure:


in_0 in_4 in_8 in_{12}
in_1 in_5 in_9 in_{13}
in_2 in_6 in_{10} in_{14}
in_3 in_7 in_{11} in_{15}
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
out_0 out_4 out_8 out_{12}
out_1 out_5 out_9 out_{13}
out_2 out_6 out_{10} out_{14}
out_3 out_7 out_{11} out_{15}
k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 . . . w_{42} w_{43}

(a) Input, State array and Output

(b) Key and Expand Key

Figure.3 AES Data Structures

- This cipher is not a Feistel structure.
- The key that is provided as input is expanded into an array of 44 words (32-bits each), w[i]. 4 distinct words (128 bits) serve as a round key for each round; these are indicated in Figure. 2.
- 4 different stages are used, 1 permutation and 3 of substitution:
  - ➢ Substitute bytes – Uses an S-box to perform a byte-to-byte substitution of the block.
  - ➢ Shift rows – A simple permutation.
  - ➢ Mix columns – A substitution that makes use of arithmetic over GF (2^8).
  - ➢ Add round key – A simple bitwise XOR of the current block with the portion of the expanded key.
- The structure is quite simple. Figure.4 depicts the structure of a full encryption round.

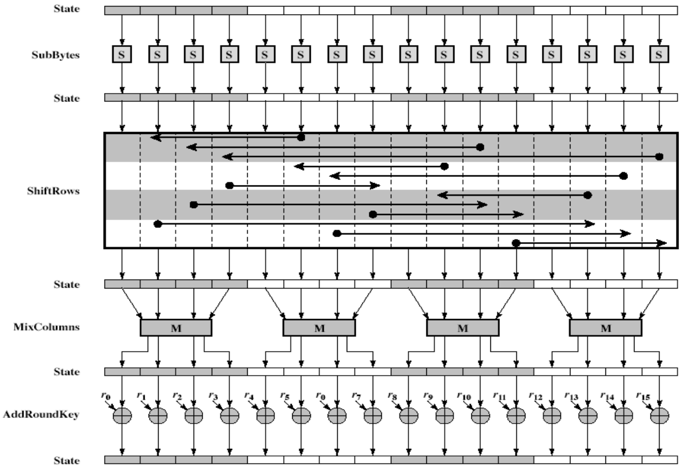

State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

Figure.4 AES Encryption Round

- Only the Add Round Key stage uses the key. Any other stage is reversible without knowledge of the key.
- The Add Round Key is a form of Vernam cipher and by itself would not be formidable. The other 3 stages together provide confusion, diffusion, and

nonlinearity, but by themselves would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (Add Round Key), followed by scrambling of the block.

- The final round of encryption consists of only 3 stages; it is the consequence of the particular structure of AES (William Stallings, 2005).

Design Steps Aes Algorithm

SubBytes Transformation

AES defines a 16 x 16 matrix of byte values called an S-box that is a pre calculated substitution table contains 256 numbers (from 0 to 255) and their corresponding resulting values. S-box table is as shown in Table II. Each byte of State array is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the right most 4 bits are used as a column value (P.Karthigaikumar et al, 2011)(P. Aatheeswaran et al, 2013). These row and column values serve as indexes into the S-box to select a unique 8-bit output value as shown in Figure.5.

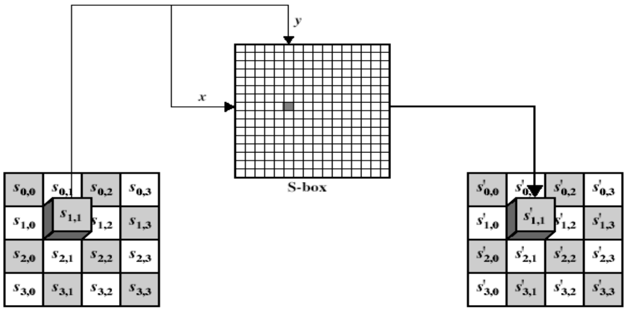
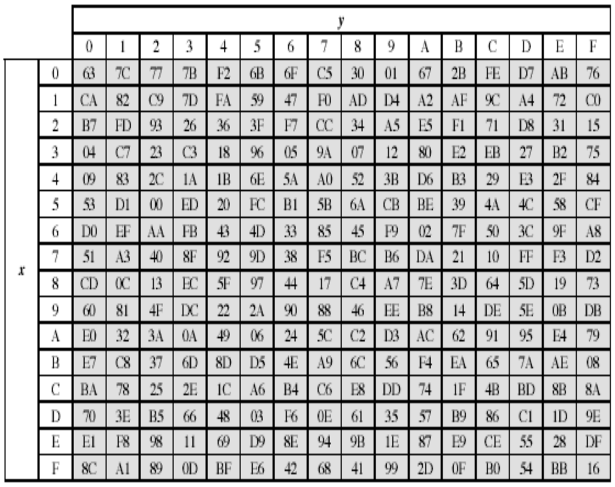


Figure.5 S-box transformation

Table II: S-box table



		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

Shift Rows Transformation

Every row in the state is shifted a certain amount to the left. In this operation, each row of the state is cyclically shifted to the left, depending on the row index. The first row is not shifted, the second shifted 1 byte position, the third 2 byte and the fourth 3 byte position (AES FIPS PUB 197, 2001). A graphical representation of shiftrows transformation is shown Figure.6.

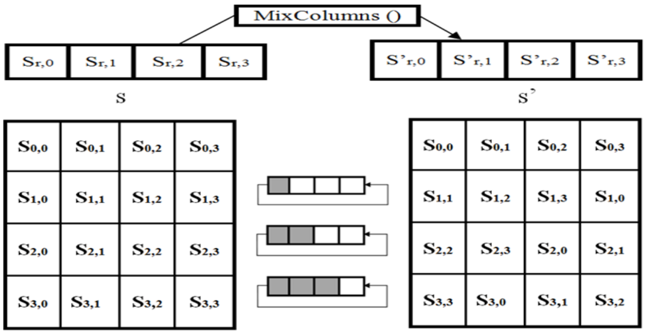


Figure.6 Shift Rows transformation

MixColumns Transformation

MixColumn operation (Figure.7) performs on the state column by column, and each column is treated as a four term Polynomial over GF (2^8) As a result of this multiplication, the new four bytes in a column are generated as follows (Vasamsetti Ramoji et al, 2012):

$$A' = (\{02\}.A)\wedge(\{03\}.B)\wedge(\{01\}.C)\wedge(\{01\}.D)$$
$$B' = (\{01\}.A)\wedge(\{02\}.B)\wedge(\{03\}.C)\wedge(\{01\}.D)$$
$$C' = (\{01\}.A)\wedge(\{01\}.B)\wedge(\{02\}.C)\wedge(\{03\}.D)$$
$$D' = (\{03\}.A)\wedge(\{01\}.B)\wedge(\{01\}.C)\wedge(\{02\}.D)$$

(1)

The operation of '^' is XOR operation modulo 2 and the '.' is a multiplication of polynomials modulo an irreducible polynomial $m(x) = x8 + x4 + x3 + x + 1$ .The operation of $\{02\}.X$ can be computed using VerilogHDL HDL language:

$$\{02\}.X = \{X[6:0], 1'b0\}\wedge(8'h1B \,\&\, \{8\{X[7]\}\})$$

(2)

So $\{03\}\_X$ can be generated as follows (Vasamsetti Ramoji et al, 2012) :

$$\{03\}.X = (\{02\}.X) + \{01\}.X(3)$$

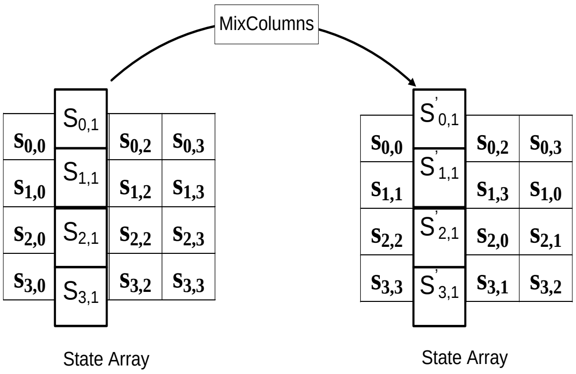


Figure.7 MixColumns transformation

AddRoundKey Transformation

During the AddRoundKey transformation, the round key values are added to the State by means of a simple Exclusive or (XOR) operation (Vasamsetti Ramoj et al, 2012). Each round key consists of N_b words that are generated from the Key Expansion routine. The round key values are added to the columns of the state in the following way:

$$\left[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}\right] = \left[s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}\right] \oplus \left[w_{round*Nb+c}\right]$$

For $0 \le c < N_b$

In the equation above, the round value is between $0 \le round \le N_r$. When round=0, the cipher key itself is used as the round key and it corresponds to the initial AddRoundKey transformation. The AddRoundKey transformation is illustrated in Figure 8.

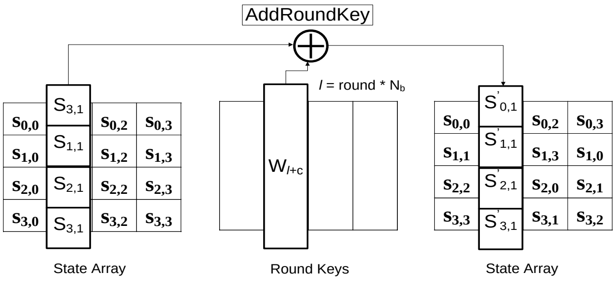


Figure.8 AddRoundKey XORs each column of the state with a word from the key schedule

Key Expansion

The AES algorithm takes the Cipher Key and performs a Key Expansion routine to generate a key schedule (P.Karthigaikumar et al, 2011). This process, as shown in Figure.9, consists of the following sub-functions:

- RotWord performs a one-byte circular left shift on a word.
- SubWord performs a byte substitution on each byte of its input word using the S-box.
- The result of steps i and ii is XOR-ed with a round constant RC[j] is shown in Table III.

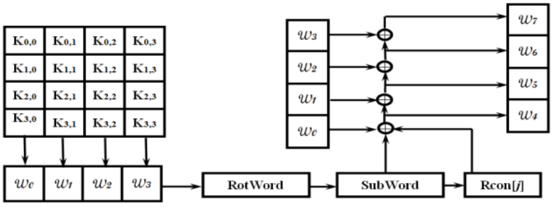


Figure.9 Key Expansion process

Table III: The value RC[j] in Hexadecimal

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

4. Simulation Results

Each round has 4 operations and it is iterative in nature. So the output of first round is fed to the second round as input data and performs the same operations with another set of keys. This process continued until the last round reach. In the last round, there is no MixColumn operation. The State array obtained after the last round is the required cipher text for transmission (Figure.10) and Simulation Results for Different Combination of Input Data as shown in Table IV.

Encryption Process (Cipher):
AES block length/Plain Text = 128bits (Nb = 4)
Key length = 128 bits (Nk = 4);
No. of Rounds = 10(Nr = 10)

Input data

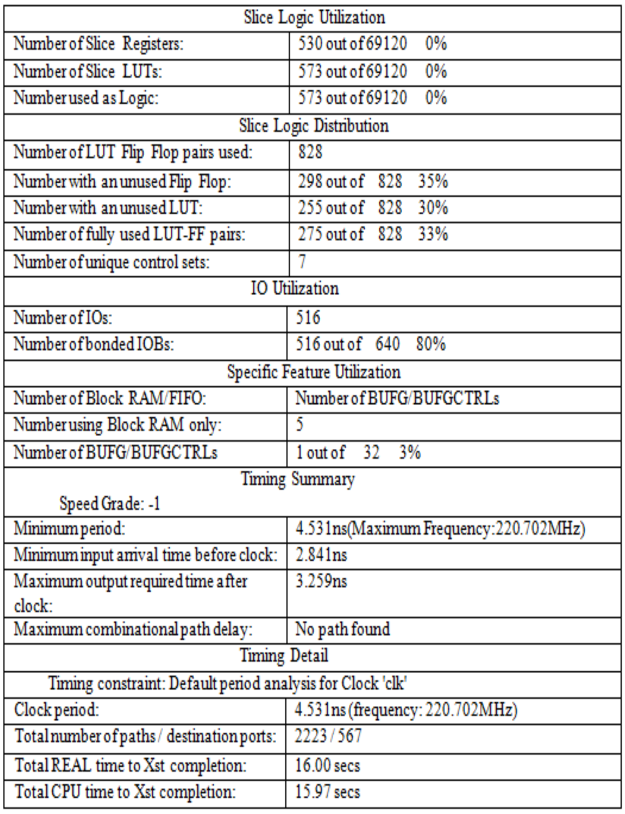
[11223344aabbccddeeffaabbccddeeff]

Input key

[000102030405060708090a0b0c0d0e0f]

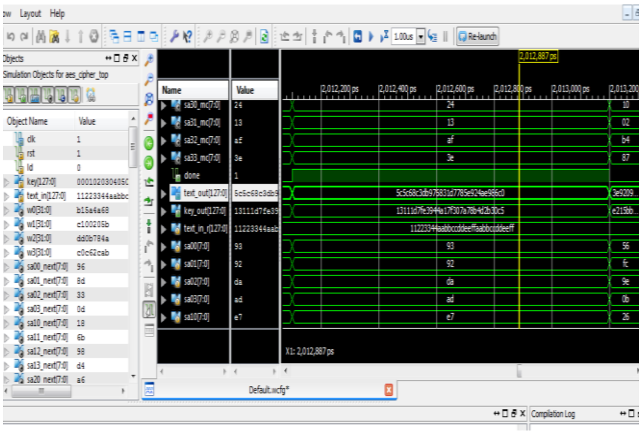
Output/Cipher text
[5c5c68c3db976831d7785e924ae986c0]

Table IV Device utilization summary:



Slice Logic Utilization	
Number of Slice Registers:	530 out of 69120 0%
Number of Slice LUTs:	573 out of 69120 0%
Number used as Logic:	573 out of 69120 0%
Slice Logic Distribution	
Number of LUT Flip Flop pairs used:	828
Number with an unused Flip Flop:	298 out of 828 35%
Number with an unused LUT:	255 out of 828 30%
Number of fully used LUT-FF pairs:	275 out of 828 33%
Number of unique control sets:	7
IO Utilization	
Number of IOs:	516
Number of bonded IOBs:	516 out of 640 80%
Specific Feature Utilization	
Number of Block RAM/FIFO:	Number of BUFG/BUFGCTRLs
Number using Block RAM only:	5
Number of BUFG/BUFGCTRLs	1 out of 32 3%
Timing Summary Speed Grade: -1	
Minimum period:	4.531ns(Maximum Frequency:220.702MHz)
Minimum input arrival time before clock:	2.841ns
Maximum output required time after clock:	3.259ns
Maximum combinational path delay:	No path found
Timing Detail	
Timing constraint: Default period analysis for Clock 'clk'	
Clock period:	4.531ns (frequency: 220.702MHz)
Total number of paths / destination ports:	2223 / 567
Total REAL time to Xst completion:	16.00 secs
Total CPU time to Xst completion:	15.97 secs

The device utilization summary is as shown in Table IV and the Selected Device is 5vlx110tff1136-1.



Object Name	Value
clk	1
rst	1
ld	0
key[127:0]	000102030405C
text_in[127:0]	11223344aabbc
w0[31:0]	b15a4a68
w1[31:0]	c100205b
w2[31:0]	dd0b784a
w3[31:0]	c0c62cab
sa00_next[7:0]	96
sa01_next[7:0]	8d
sa02_next[7:0]	33
sa03_next[7:0]	0d
sa10_next[7:0]	18
sa11_next[7:0]	6b
sa12_next[7:0]	98
sa13_next[7:0]	d4
sa20_next[7:0]	a6

Name	Value		
sa30_mc[7:0]	24	24	10
sa31_mc[7:0]	13	13	02
sa32_mc[7:0]	af	af	b4
sa33_mc[7:0]	3e	3e	87
done	1		
text_out[127:0]	5c5c68c3db9	5c5c68c3db976831d7785e924ae986c0	3e9209...
key_out[127:0]	13111d7fe39	13111d7fe3944a17f307a78b4d2b30c5	e215bb...
text_in_r[127:0]	11223344aab	11223344aabbccddeeffaabbccddeeff	
sa00[7:0]	93	93	56
sa01[7:0]	92	92	fc
sa02[7:0]	da	da	9e
sa03[7:0]	ad	ad	0b
sa10[7:0]	e7	e7	26

Figure.10 Simulation Waveform of Encrypted data

Table V: Simulation Results for Different Combination of Input Data

Sr No	Input data	128 Bit key	Encrypted data
1	00112233445566778899aabbccddeeff	aabbccddeeffaabbccddeeffaabbccdd	d2acdc78b8ebace37f2df83a37c75062
2	11223344aabbccddeeffaabbccddeeff	000102030405060708090a0b0c0d0e0f	5c5c68c3db976831d7785e924ae986c0
3	AABBCCDDEEFFaabbccdd001122334455	0011223344aabbccddeeAABBCCDDEEFF	8f7ec00c1f99cc3c8ee48fa4d1e8b868
4	000102030405060708091011121314 15	00010203040506070809101112131415	e08c969f6ca83fd3c12304f373cd3a83

Conclusion

In this paper software implementation of Advanced Encryption Standard (AES) algorithm is used for data encryption that can process with the data block of 128 bit and cipher key length of 128 bit. The usage of 128 bit cipher key to achieve the high security, because 128 bit cipher key is difficult to broken. As result of this secure transmission of data is occurred in encryption. While computing the existing AES, the SubBytes transformation consumes the more memory in AES so to overcome this affine transform is used in AES flow.

Acknowledgment

The authors wish to thank the department of ECE for providing us the facility to do project work in Research and Development lab. The authors are also grateful to the authorities of SDMCET Dharwad, for the facility extended to carry out the present work. The authors also would like to thank the unknown reviewers for their invaluable comments which help in improving the paper.

References

Advanced Encryption Standard (AES) (2001), Federal Information Processing Standards publication 197.

K.Rahimunnisa, M. Priya Zach, S. Suresh Kumar and J.jayakumar (2012) Architectural optimization of AES Transformations and key expansion.

P.Karthigaikumar and Soumiya Rasheed (2011), Simulation of Image Encryption using AES Algorithm.

Vasamsetti Ramoji, P.Ganesh and Ch.Appala Swamy, (2012), Highly Secured High Throughput Efficient VLSI Architecture for AES Implementations.

William Stallings (2005), Cryptography and Network Security Principles and Practices, Fourth Edition: Prentice-Hall.

Issam Mahdi Hammad, (2010) Efficient Hardware Implementations for the Advanced Encryption Standard (AES) Algorithm, Dalhousie University Halifax, Nova Scotia.

G. H. Karimian, B. Rashidi, and A.farmani (2011), A High Speed and Low Power Image Encryption with 128 Bit AES Algorithm

P. Aatheeswaran and Dr.R.Suresh Babu (2013), FPGA can be implemented by using Advanced Encryption Standard Algorithm.