IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 11, Issue 6, Ver. II (Nov.-Dec .2016), PP 58-66
www.iosrjournals.org

Simulation of Advanced Encryption Standard Algorithm

A Sowmya[1], K Tarakeswara rao[2]
[1]*Pursuing M.Tech (VLSI & ES),*
[2]*Assistant Professor (ECE)*
[1,2]*Vignan's Institute of Engineering for Women,Visakhapatnam-530046, Andhra Pradesh, India*

Abstract: *Advanced Encryption Standard (AES) is an encryption standard used for securing information. AES is a block cipher algorithm that has been analyzed extensively and is now used widely. AES is very secure and has no known weakness. In this work, both encryption and decryption will be carried out with the key length of 128 bits, that is, both AES encrypter and the AES decrypter were integrated. Hence the input block and secret key will be provided for encryption and the cipher block and same secret key will be provided to the decryption to get the proper block as output.*
All the transformations of both Encryption and Decryption will be developed using VHDL language and will be verified with the help of its simulation result using XILINX ISE simulator.
Keywords: *architecture; background; crypto graphical methods; block-diagram; encryption; key expansion; decryption; simulation results.*

I. Introduction

AES (advanced encryption standard) is proposed by NIST (National Institute of Standards and Technology), developed in 2000. AES is a FIPS (Federal information processing standard), which is a crypto graphical algorithm that is used to protect electronic data. AES uses cryptographic keys of 128,192, 256 bits to encrypt and decrypt data in blocks of 128-bits. AES provides combination of security, performance, efficiency and flexibility.

AES algorithm was designed to have characteristics such as resistance against all known attacks, speed and code compactness on a wide range of platforms, design simplicity. To obtain a high throughput in a speed of Gbps, pipelining concept introduced in AES.

II. Background of AES

The older standard, data encryption standard (DES) which is up to 56-bits key size and block size of 64-bits, undergoes four transformations in 16 rounds. DES is vulnerable to differential and linear cryptanalysis, weak substitution tables. To overcome the disadvantage of DES algorithm, the new standard is AES. AES is strong against differential, truncated differential, linear, interpolation and square attacks.

DES algorithm uses a fiestal network, divides block into two halves before going through encryption steps. AES algorithm is based on substitution permutation network, has series of linked mathematical operations. In DES, maximum amount of data that can be transferred with single encrypt key is 32GB and for AES is 256 billion Gbytes.

III. AES Architecture

The Advanced Encryption Standard (AES) specifies a FIPS approved cryptographic algorithm that can be used to protect electronic data. Data that can be read and understood normally called 'plain-text'. Encryption converts plain-text to an unreadable form called 'cipher-text', which is readable only by authorized users. Decryption converts cipher-text to its original form called plain-text.

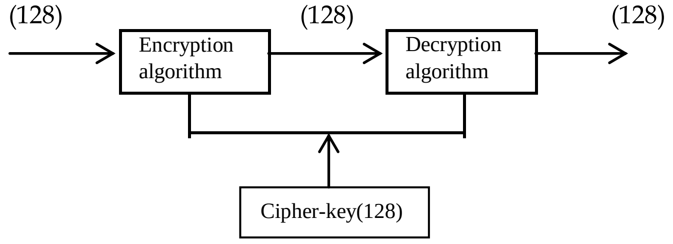


Figure 1: AES 128-bit architecture

IV. Cryptographical Methods of AES

In some encryption methods, the receiver and the sender use the same key and in other encryption methods, they must use different keys for encryption and decryption purposes. They are Symmetric versus Asymmetric Algorithms. Cryptography algorithms use either symmetric keys, also called secret keys, or asymmetric keys, also called public keys.

A. Symmetric cryptography

In a cryptosystem that uses symmetric cryptography, both parties will be using the same key for encryption and decryption. This provides dual functionality. Symmetric keys are also called secret keys because this type of encryption relies on each user to keep the key a secret and properly protected.

If this key got into an intruder's hand, that intruder would have the ability to decrypt any intercepted message encrypted with this key. Each pair of users who want to exchange data using symmetric key encryption must have their own set of keys.

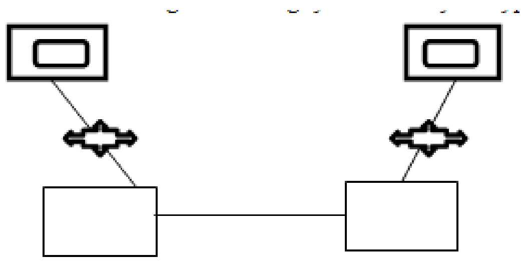


Figure 2: Symmetric cryptography

B. Asymmetric cryptography

In symmetric key cryptography, a single secret key is used between entities, whereas in public key systems, each entity has different keys, or asymmetric keys. The two different asymmetric keys are mathematically related. If a message is encrypted by one key, the other key is required to decrypt the message.

In a public key system, the pair of keys is made up of one public key and one private key. The public key can be known to everyone, and the private key must only be known to the owner. Many times, public keys are listed in directories and databases of e-mail addresses so they are available to anyone who wants to use these keys to encrypt or decrypt data when communicating with a particular person.

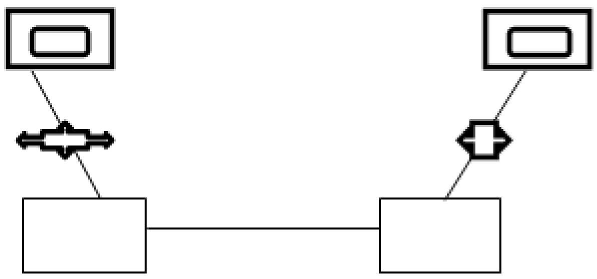


Figure 3: Asymmetric cryptography

V. Advanced Encryption Standard Algorithm

AES algorithm's operations are performed on a two-dimensional array of bytes called 'State'. The state is a rectangular array of bytes and since the block size is 128-bits, which are 16-bytes, the rectangular array is of dimensions 4x4. The basic unit for processing in the AES algorithm is a byte, a sequence of eight bits treated as a single entity. The architecture of AES-128 consists of 128-bit key length; 128-bit plain-text is used for encryption and same for decryption process.

In this version with variable block size, the row size is fixed to four and the number of columns varies. The number of columns is the block size divided by 32 and denoted Nb. The cipher key is similarly pictured as a rectangular array with four rows. The number of columns of the cipher key, denoted Nk, is equal to the key length divided by 32. AES uses a variable number of rounds, which are fixed: A key of size 128 has 10 rounds.

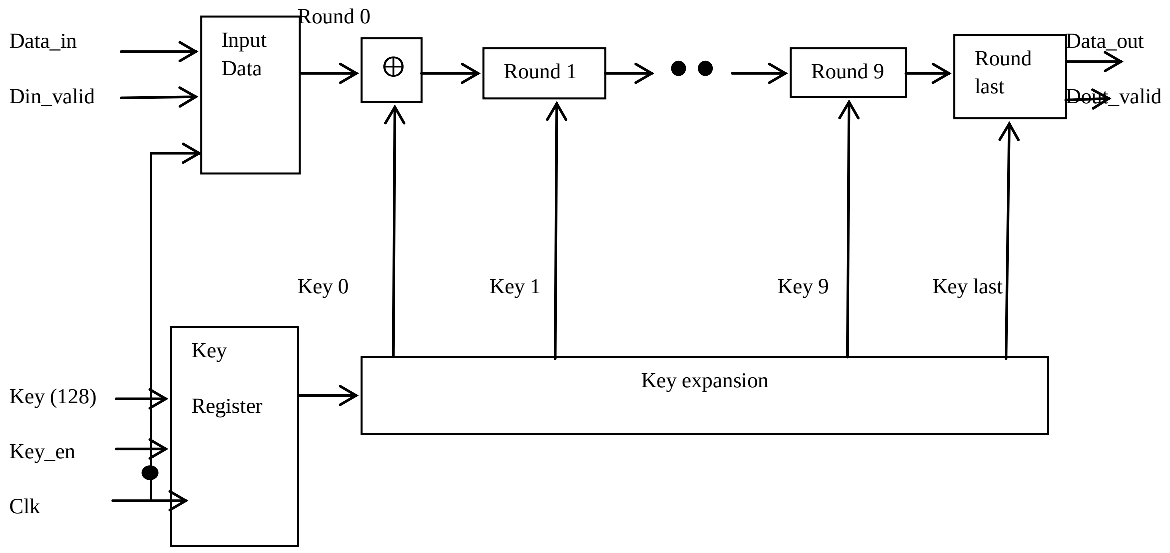


Figure 4: AES block-diagram

VI. Encryption

At the start of the Encryption or Cipher, the input data and the input key were copied to the State array using the conventions. Initially the XOR operation should be performed between each byte of the input data and the input key and the output will be given as the input of the Round-1.

After an initial Round Key addition, the State array is transformed by implementing a round function 10times, with the final round differing slightly from the first Nr–1rounds. The final State is then copied to the output. The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words derived using the Key Expansion routine.

The individual transformations that carried out are listed below.

a) Byte substitution using a substitution table (S-box)
b) Shifting rows of the State array by different offsets
c) Mixing the data within each column of the State array
d) Adding a Round Key to the State

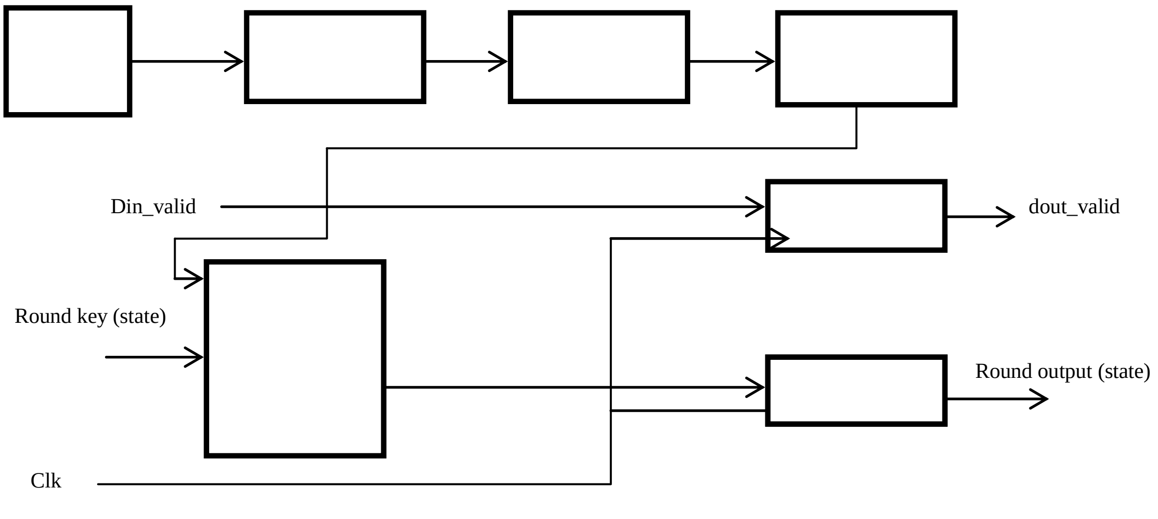


Figure 5: Block-diagram for AES round

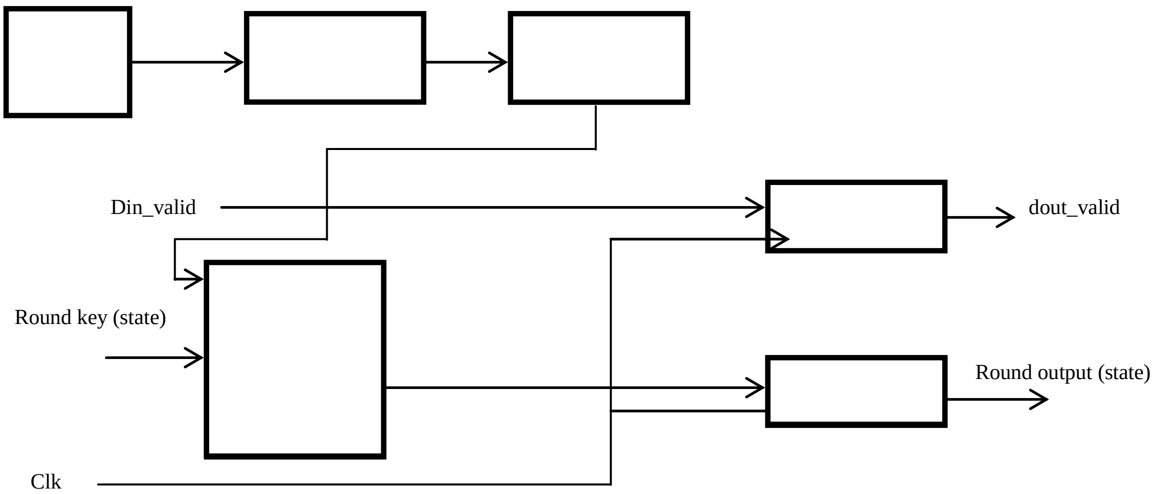
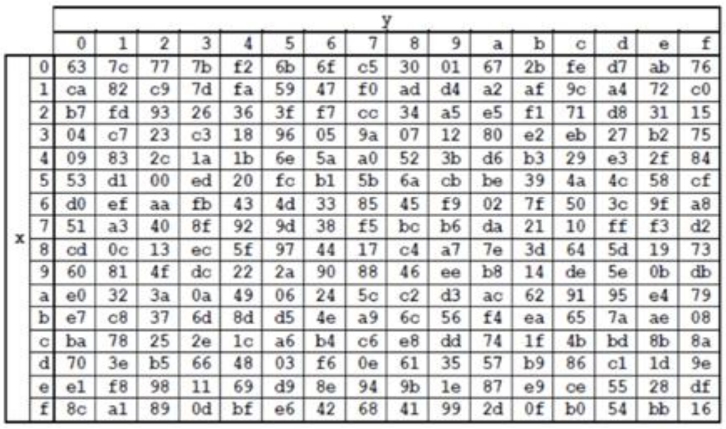


Figure 6: Block-diagram for AES last-round

A. Substitution of bytes

In first stage of substitution of bytes, the each byte of state matrix is replaced by value of byte in defined substitution-box. The S-Box will be of a 16X16 matrix in which the row is represented as "x" and the column is represented by "y".

The S-box used in the Substitution of Bytes transformation is presented in hexadecimal form and hence the substitution value would be determined by the intersection of the row and the column.



		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

S-Box: Substitution Values used in Encryption Process

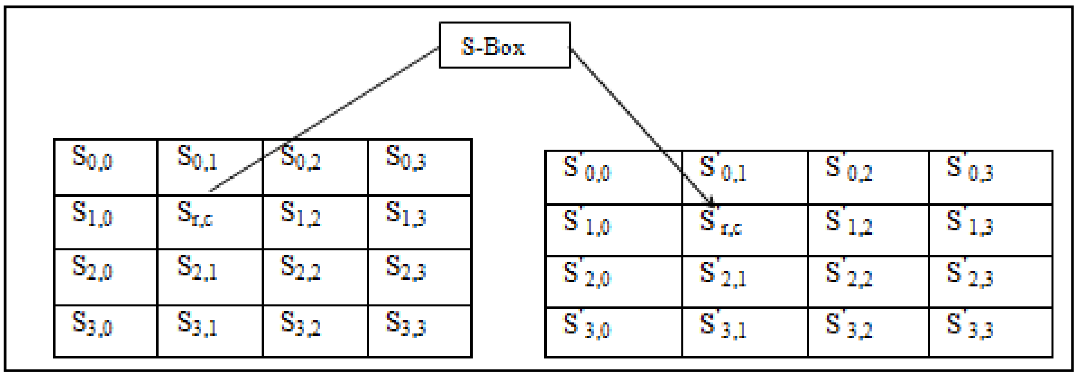

S-Box
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{r,c}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S'_{0,0}$ $S'_{0,1}$ $S'_{0,2}$ $S'_{0,3}$
$S'_{1,0}$ $S'_{r,c}$ $S'_{1,2}$ $S'_{1,3}$
$S'_{2,0}$ $S'_{2,1}$ $S'_{2,2}$ $S'_{2,3}$
$S'_{3,0}$ $S'_{3,1}$ $S'_{3,2}$ $S'_{3,3}$

Figure 7: Sub Bytes Operation of the State

B. Shift rows

The resultant output of first stage which is in form of a state matrix is given as input to second stage called 'shift rows'. In shift rows, the first row of matrix is unchanged but in the second row, each byte has to perform one-byte circular left shift. Similarly, each byte of third row has to perform two-byte circular left shift and each byte of fourth row has to perform three-byte circular left shift. This has the effect of moving bytes to lower positions in the row, while the lowest bytes wrap around into the top of the row.

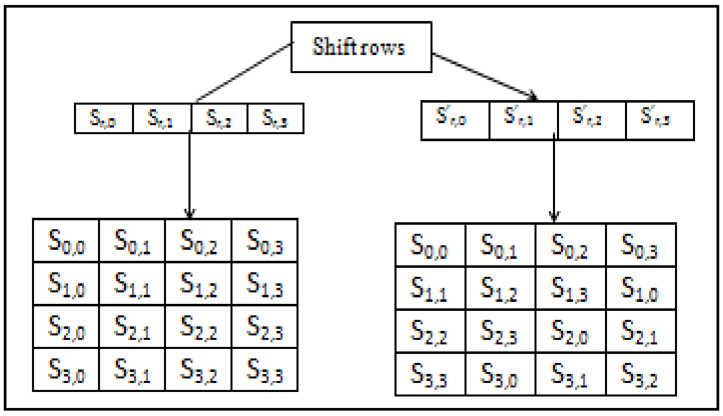


Figure 8: Shift Rows Operation of the State

C. Mix Columns

The resultant matrix of second stage is given as input to third stage called 'mix-columns'. The transformation operates on the State column-by-column. The state is arranged into a 4 row table .The multiplication is performed one column at a time (4 bytes). Each value in the column is multiplied against every value of the matrix (16 total multiplications). The results of these multiplications undergo Exclusive-OR operation together to produce only 4-bytes result for the next state. Therefore 4-bytes input,16 multiplications 12 XORs and 4-bytes output. Themultiplication is performed one matrix row at a time against each value of a state column.

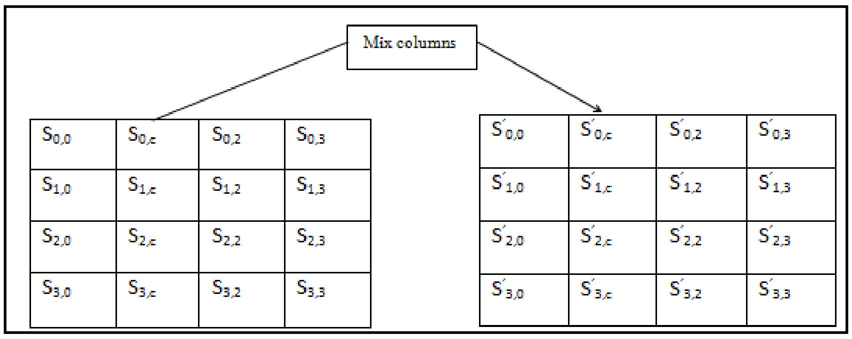


Figure 9: Mix Columns operates on the State column-by-column

The pre-defined 4X4 matrix value and the first column of the Shift Rows state are represented as follows, for the multiplication.

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

Each result byte is calculated by multiplying each 4 values of the each state column against each 4 values of the each first row of the matrix. The results of each multiplication undergo Exclusive-OR operation to produce 1 Byte.

$s'_{0,c} = (\{02\} \cdot s_{0,c}) \oplus (\{03\} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$
$s'_{1,c} = s_{0,c} \oplus (\{02\} \cdot s_{1,c}) \oplus (\{03\} \cdot s_{2,c}) \oplus s_{3,c}$
$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \cdot s_{2,c}) \oplus (\{03\} \cdot s_{3,c})$
$s'_{3,c} = (\{03\} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \cdot s_{3,c})$

D. Addround-key

The resultant matrix is given as input to fourth stage called 'addround-key'. In this last stage, bitwise Exclusive-OR operation is performed between state matrix and round-key of first round. The algorithm process continues till cipher text is obtained at last round. The keys used for each rounds are different from one another. A cipher text is obtained at the end of the last round completing the encryption process.

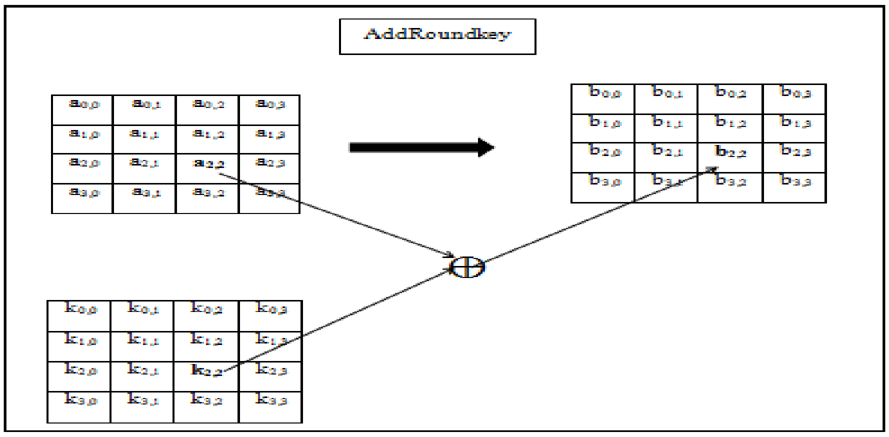

AddRoundkey
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$
$\oplus$

Figure 10: AddRoundKey Operation

VII. Key Expansion

The expanded key is used in the addround-key stage. In each round, the addround-key stage uses different part of the expanded key while performing Exclusive-OR operation against the state. Therefore expanded key must be large enough so that it can provide large key material for every time the addround-key stage is executed. The addround-key stage gets called once for each round at both beginning and ending of the algorithm.

The AES algorithm takes the cipher-key and performs a key expansion routine to generate a key schedule. The key expansion generates a total of Nb (Nr+1) words. The resulting key schedule consists of linear array of 4-byte words.

Since the key size is much smaller than the size of the sub keys, the key is stretched out to provide enough key space for the algorithm. Hence a 128-bit key is expanded into a 176-bit key. There is a relation between the cipher-key size, number of rounds and the expanded key size. For a 128-bit key, there is one initial addround-key operation and then 10 rounds of actual AES algorithm process. Each round needs a new 16 byte key; therefore we require 10+1 Round Keys of 16 byte, which equals 176 byte. An iteration of the above steps is called a 'round'. The amount of rounds of the key expansion algorithm depends on the key size.

Table.1. Key expansion

Key size (bytes)	Block size (bytes)	Expansion Algorithm Rounds	Expanded Bytes /Round	Rounds Key copy	Rounds Key Expansion	Expanded Key (bytes)
16	16	44	4	4	40	176

VIII. Decryption

The cipher text of 128 bits and the same key of 128 bits will be given as the input to the decryption block. The encrypted data will be decrypted and the original plain message will be achieved as the output of the decryption block. The Cipher transformations can be inverted and then implemented in reverse order to produce a straight forward Inverse Cipher for the AES algorithm. The individual transformations used in the Inverse Cipher were listed as follows

a) Inverse substitution of bytes
b) Inverse shift rows
c) Inverse mix columns
d) Inverse addroundkey

AES Inverse cipher functions

The AES Inverse Cipher Function has the same set of transformations as in the encryption but in the inverse form, that is, the predefined values which used for the each transformation will be different.

1) Inverse Substitution of bytes

InvSubBytes is the inverse of the byte substitution transformation, in which the inverse S-Box is applied to each byte of the State. The transformation of this process will be carried out in the similar way as in the Sub Bytes in the encryption such as the substitution value would be determined by the intersection of the row and the column.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	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	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	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	ac	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	3b	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

S-Box: Substitution Values used in Decryption Process

2) Inverse Shift rows

The InvShiftRows is the inverse of the Shift Rows transformation. The bytes in the last three rows of the Stateare cyclically shifted over different numbers of bytes (offsets). The first row, r = 0, is not shifted. The bottom three rows are cyclically shifted by Nb - shift(r, Nb) bytes, where the shift value shift(r, Nb) depends on the row number.

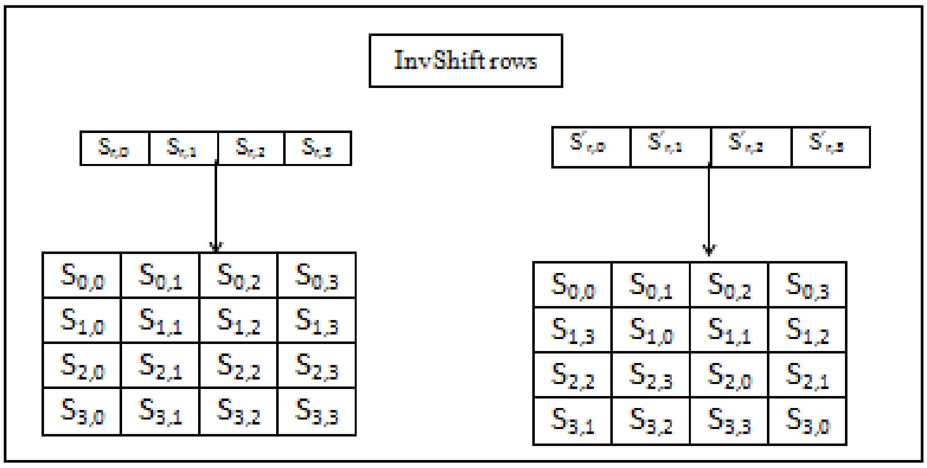

InvShift rows
$S_{r,0}$ $S_{r,1}$ $S_{r,2}$ $S_{r,3}$
$S'_{r,0}$ $S'_{r,1}$ $S'_{r,2}$ $S'_{r,3}$
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,3}$ $S_{1,0}$ $S_{1,1}$ $S_{1,2}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,1}$
$S_{3,1}$ $S_{3,2}$ $S_{3,3}$ $S_{3,0}$

Figure 11: InvShiftRows Operation of the State

3) Inverse Mix Columns

The InvMixColumns is the inverse of the Mix Columns transformation .InvMixColumns operates on the State considering column-by-column. The pre-defined 4X4 matrix value and the first column of the InvShiftRows state are represented as follows, for the multiplication.

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

As a result of this multiplication, the four bytes in a column are replaced by the following.

$s'_{0,c}$ = ({0e}·$s_{0,c}$)⊕({0b}·$s_{1s,c}$) ⊕({0d}·$s_{2,c}$)⊕ ({09} ·$s_{3,c}$)
$s'_{1,c}$ = ({09}·$s_{0,c}$) ⊕({0e·$s_{1,c}$) ⊕({0b}·$s_{2,c}$) ⊕({0d} ·$s_{3,c}$)

$s'_{2,c} = (\{0d\}\cdot s_{0,c}) \oplus (\{09\}\cdot s_{1,c}) \oplus (\{0e\}\cdot s_{2,c}) \oplus (\{0b\}\cdot s_{3,c})$
$s'_{3,c} = (\{0b\}\cdot s_{0,c}) \oplus (\{0d\}\cdot s_{1,c}) \oplus (\{09\}\cdot s_{2,c}) \oplus (\{0e\} \cdot s_{3,c})$
Thus the 4X4 matrix will be obtained which will be given as the input to the next transformation.

4) Inverse Addroundkey
The Inverse of the AddRoundKey is similar to the AddRoundKey in the encryption process. Each element in the resultant matrix of Mix Columns and resultant matrix of Key Expansion will be XORed and the resultant matrix of AddRoundKey will be given as the input to the next round. Hence all the inverse cipher transformations were discussed above and finally, the only thing left to do is putting it all together in one inversed main algorithm. Similarly the forward cipher transformations were combined together to form a Round and combining all the 10 Rounds will constitute a complete AES Encryption and Decryption algorithm

IX. Simulation Results

The simulation of AES algorithm is done using XILINX ISE SIMULATOR OF VERSION 10.1 through VHDL. The plain text and the key of 128 bits will be given as the input to the design and the obtained cipher text undergoes decryption process ensure that the data generated at the end should be equal to given input.

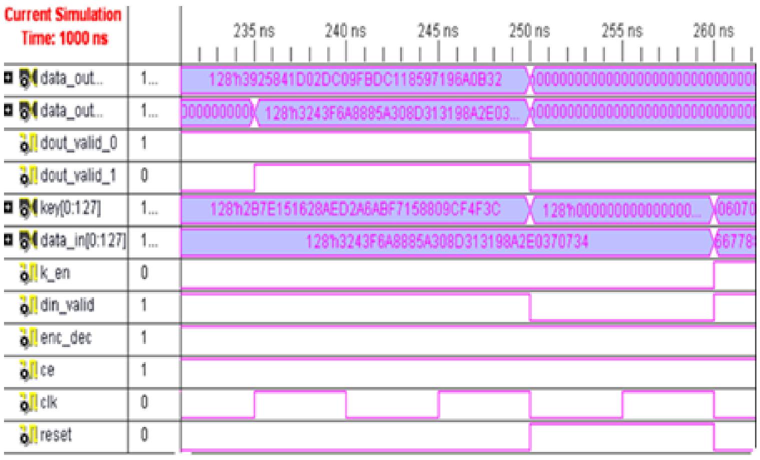


Figure 12: Simulation result

This case deals with the both encryption and decryption for first set of plain text and a key of 128 bits. The basic and common inputs for both encryption and decryption stage were clock (clk), chip enable (ce) and reset (rst). The reset signal is active high, that is, when the reset signal is set to high, the system will be in reset state and hence all the values will be '0'. Once the reset signal is set to low, the system will start its process.

There is signal "enc_dec" which represents that the system is in which operation either in encryption or decryption. When this "enc_dec" is set to high, the encryption process will be carried out with the given inputs and when this signal is set to low, the decryption process will be carried out. The two inputs named as "data_in" and "key_in" which takes the given plain text and the key.

A. Encryption
Here the sets of inputs are taken from the reference as follows.

Input = 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34
Cipher Key = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

The above inputs were represented in the hexadecimal format which contains 16 bytes, that is, 128 bits. So when the proper inputs were given as the input to the system, "din_valid" and "k_en" signals will go high. These signals represents that the valid data and the proper key is given to the system. Hence the output of the encryption process, that is, the cipher text for the given set of inputs is obtained as follows.

Cipher Text = 39 25 84 1d 02 dc 09 fb dc 11 85 97 19 6a 0b 32

B. Decryption
The above cipher text, that is, encrypted data will be given as the input to the decryption stage and the same key should be provided.

Input = 39 25 84 1d 02 dc 09 fb dc 11 85 97 19 6a 0b 32

Cipher Key = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

Here the “din_valid” signal will goes high only after the encryption process. Hence the decryption process will be carried out and the final output, that is, the same plain text which is given as the input to the encryption stage will be achieved.

Final Output = 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34

X. Conclusion

Firstly, understanding the concept of cryptology, flow of AES algorithm is done. Successful simulation of AES algorithm, make to know one of the encryption and decryption standard available in market and it helps to explore the path to simulate such an algorithm using VHDL. This is a 128-bit Key dependent algorithm which has control over the 128-bit input data or plaintext. The original message is taken to 10 round operations which produces the cipher text. This resultant encrypted data is fed as the input to the decryption and 10 rounds operations were carried out and hence the same plain text is achieved. The simulation results have been verified.

References

[1] J. Daemen and V. Rijmen, AES Proposal: Rijndael, AES Algorithm Submission, September 3, 1999, available at via http://www.nist.gov/CryptoToolkit.

[2] Kaur, S., &Vig, R., “Efficient implementation of AES algorithm in FPGA device” Proceedings - International Conference on Computational Intelligence and Multimedia Applications, ICCIMA 2007, 2, 179–187. doi:10.1109/ICCIMA.2007.180

[3] “Design of a High Throughput Crypto Devices Based on AES” , International Journal of Engineering science & Innovative Technology(IJESIT), Volume 2, Issue 1, January 2013

[4] Marcelo B. de Barcelos Design Case, “Optimized performance and area Implementation of Advanced Encryption Standard in Altera Devices.

[5] Tilborg, Henk C. A. van. “Fundamentals of Cryptology: A Professional Reference and Interactive Tutorial”, New York Kluwer Academic Publishers, 2002

[6] Peter J. Ashenden, “The Designer's Guide to VHDL”, 2nd Edition, San Francisco, CA, Morgan Kaufmann, 2002

[7] A. Lee, NIST Special Publication 800-21, “Guideline for Implementing Cryptography in the Federal Government”, National Institute of Standards and Technology, November 1999.

[8] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, New York, 1997, p. 81-83.