ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

MULTI-KEY IMPLEMENTATION OF AES ALGORITHM IN VERILOG

K. Reddy Rani*, Kuppam N Chandrasekar
* M.Tech scholar, GVIC, Madanapalle, A.P, India.
Assistant Professor, Dept. of ECE, GVIC, Madanapalle, A.P, India.

ABSTRACT

Increasing need of high security in communication led to the development of several cryptographic algorithms hence sending data securely over a transmission link is critically important in many applications. NIST in the beginning selected Rijndael within October 2000 and formal adoption as being the AES standard started in December 2001. FIPS PUB 197 explains a 128-bit block cipher making a use of a 128, 192, or 256-bit key. This paper presents the Rijndael algorithm ([3] and [4]), a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits & each program simulation results are verified with ModelSim PE and are synthesized in Xilinx ISE 14.3 Project Navigator.

KEYWORDS: Cryptography; Rijndael, Key Schedule; Encryption; Decryption.

INTRODUCTION

Cryptography is the science of information and communication security. Cryptography is the science of secret codes, enabling the confidentiality of communication through an insecure channel. It protects against unauthorized parties by preventing unauthorized alteration of use. It uses a cryptographic system to transform a plaintext into a cipher text, using most of the time a key.

The Advanced Encryption Standard, in the following referenced[1] as AES, is the winner of the contest, held in 1997 by the US Government, after the Data Encryption Standard was found too weak because of its small key size and the technological advancements in processor power. Fifteen candidates were accepted in 1998 and based on public comments the pool was reduced to five finalists in 1999. In October 2000, one of these five algorithms was selected as the forthcoming standard: a slightly modified version of the Rijndael.

There are many architecture proposals for AES Rijndael algorithm [1], but many of them are poor in terms of area and speed. This paper proposes a different approach to increase speed by utilizing lesser resources available in FPGA. This paper is structured as follows: Section2 describes AES Rijndael algorithm. The result and conclusion are described in Section 3 and 4 respectively.

AES RIJNDAEL ALGORITHM

The AES Rijndael is a block cipher, which operates on different keys and block lengths: 128 bits, 192 bits, or 256 bits. The input to each round consists of a block of message called the state and the round key. It has to be noted that the round key changes in every round. The state can be represented as a rectangular array of bytes. This array has four rows; the number of columns is denoted by Nb and is equal to the block length divided by 32. The same could be applied to the cipher key. The number of columns of the cipher key is denoted by Nk and is equal to the key length divided by 32. The cipher consists of a number of rounds - that is denoted by Nr - which depends on both block and key lengths. Each round of Rijndael encryption function consists mainly of four different transformations: SubByte, ShiftRow, MixColumn and key addition. On the other hand, each round of Rijndael decryption function consists mainly of four different transformations: InvSubByte, InvShiftRow, InvMixColumn, and key addition.

The AES Encryption

The 128-bit data block and key are considered as a byte array, respectively called "State" and "RoundKey", with four rows and four columns. The description of the four transformations of the Rijndael cipher and their inverses will be given below.

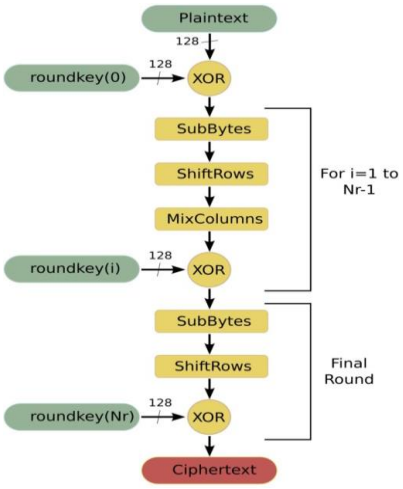

Fig. 1: Algorithm for AES Encryption

The AES is a computer security standard from NIST intended for protecting electronic data. Federal Information Processing Standards (FIPS) Publication 197 gives the specification of AES[1].

Rijndael encryption consist of four operations

1. Key addition
2. Substitution
3. Shift Row
4. Mix Column

Key addition

Add round key

State is represented as follows (16 bytes):

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes called the State. It consists 4 rows, each containing Nb bytes, Nb columns, costituted by 32-bit words. Sr,c denotes the byte in row r and column c. The array of bytes in input is copied in the State matrix. At the end, the State matrix is copied in the output matrix

Add round key (state, key):

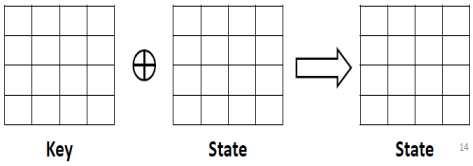

Fig. 2: Add round key

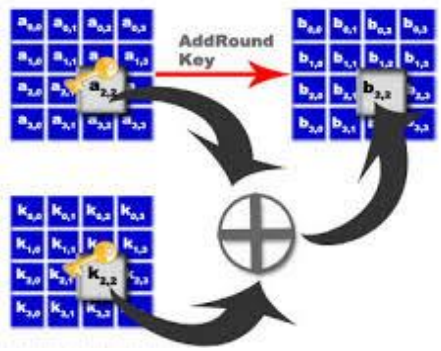

Fig. 3: Round Key is added to the State using an XOR operation

Substitution

Sub bytes transformation:

Bytes are transformed using a non linear s-box

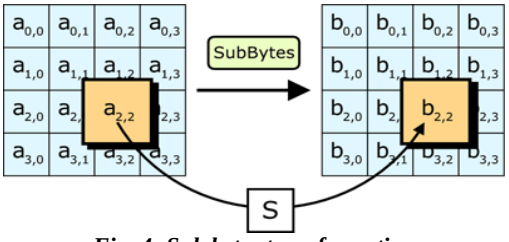

Fig. 4: Sub bytes transformation

Byte substitution using a non-linear (but invertible) S-Box (independently on each byte). S-box is represented as a 16x16 array, rows and columns indexed by hexadecimal bits. 8 bytes replaced as follows: 8 bytes define a hexadecimal number rc, then Sr,c = binary(S-box(r, c))

x \ y	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	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. 5: S-box substitution values for the byte xy (in hexadecimal format).

Shift rows

Shift row transformation

The rows of the state matrix is shifted Circular to Left of a number of bytes equal to the row index. The 1st row is shifted 0 positions to the left. The 2nd row is shifted 1 position to the left. The 3rd row is shifted 2 positions to the left. The 4th row is shifted 3 positions to the left.

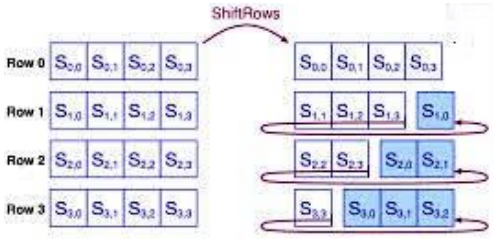

Fig. 6: Shift rows

Mix column

Mix column transformation

Bytes in columns are combined linearly. Interpret each column as a vector of length 4. Each column of State is replaced by another column obtained by multiplying that column with a matrix in a particular field (Galois Field).

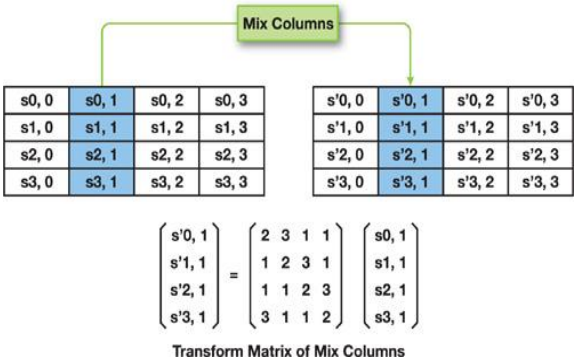

Fig. 7: Mix column transformation

The MixColumns () transformation operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF (2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial a(x), given by

$a(x) = \{03\}x3 + \{01\}x2 + \{01\}x + \{02\}$

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

For $0 \leq c < Nb$

$S'_{0,c} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$

$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$

$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$

$S'_{0,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c})$

The AES Decryption

The Rijandael decryption consists of four inverse operations of encryption which are compliment functions of encryption. They are

1. Inverse Substitution

2. Inverse Shift Row
3. Inverse Mix Column
4. Key addition

Inverse Substitution
Inverse Substitution Transformation
The InvSubByte transformation is done using a once-pre-calculated substitution table called InvS-box. That table (or InvS-box) contains 256 numbers (from 0 to 255) and their corresponding values.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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

Fig 8: Inverse S-box substitution values for the byte xy in Hexadecimal format

Inverse shift row
InvShiftRow Transformation
In InvShiftRow transformation, the rows of the state are cyclically right shifted over different offsets. Row 0 is not shifted, row 1 is shifted over one byte, row 2 is shifted over two bytes and row 3 is shifted over three bytes.

Inverse mix column
InvMixColumns () Transformation
InvMixColumns () is the inverse of the MixColumns () transformation. InvMixColumns () operates on the State column-by-column, treating each column as a four term polynomial. The columns are considered as polynomials over GF (2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial $a^{-1}(x)$, given by

$a\text{-}1(x) = \{0b\}\ x3 + \{0d\}\ x2 + \{09\}\ x + \{0e\}$

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

For $0 \leq c < Nb$
As a result of this multiplication, the four bytes in a column are replaced by the following

$S'_{0,c} = (\{0e\} \bullet S_{0,c}) \oplus (\{0b\} \bullet S_{1,c}) \oplus (\{0d\} \bullet S_{2,c}) \oplus (\{09\} \bullet S_{3,c})$

$S'_{1,c} = (\{09\} \bullet S_{0,c}) \oplus (\{0e\} \bullet S_{1,c}) \oplus (\{0b\} \bullet S_{2,c}) \oplus (\{0d\} \bullet S_{3,c})$

$S'_{2,c} = (\{0d\} \bullet S_{0,c}) \oplus (\{09\} \bullet S_{1,c}) \oplus (\{0e\} \bullet S_{2,c}) \oplus (\{0b\} \bullet S_{3,c})$

$S'_{0,c} = (\{0b\} \bullet S_{0,c}) \oplus (\{0d\} \bullet S_{1,c}) \oplus (\{09\} \bullet S_{2,c}) \oplus (\{0e\} \bullet S_{3,c})$

Key addition
Inverse of the Add Round Key Transformation
AddRoundKey (), it has its own inverse, since it only involves an application of the XOR operation.

The Rijndael Key Schedule

The Key Schedule is responsible for expanding a short key into a larger key, whose parts are used during the different iterations. Each key size is expanded to a different size:

- An 128 bit key is expanded to an 176 byte key.
- An 192 bit key is expanded to an 208 byte key.
- An 256 bit key is expanded to an 240 byte key.

There is a relation between the cipher key size, the number of rounds and the Expanded Key size. For an 128-bit key, there is one initial AddRoundKey operation plus there are 10 rounds and each round needs a new 16 byte key, therefore we require 10+1 Round Keys of 16 byte, which equals 176 byte. The same logic can be applied to the two other cipher key sizes. The general formula is that:
ExpandedKeySize = (nbrRounds+1) * BlockSize

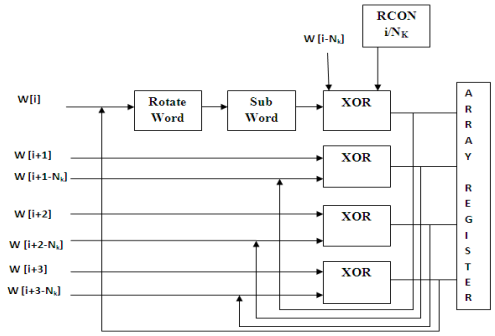

Fig. 9: Key expansion algorithm

The AES algorithm takes the Cipher Key, ***K***, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of ***Nb*** (***Nr*** + 1) words: the algorithm requires an initial set of ***Nb*** words, and each of the ***Nr*** rounds requires ***Nb*** words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [*wi*], with *i* in the range 0< = *i* < ***Nb***(***Nr*** + 1).

SIMULATION RESULTS

The AES simulation outputs is as follows by using Xilinx l4.3 ISE software,

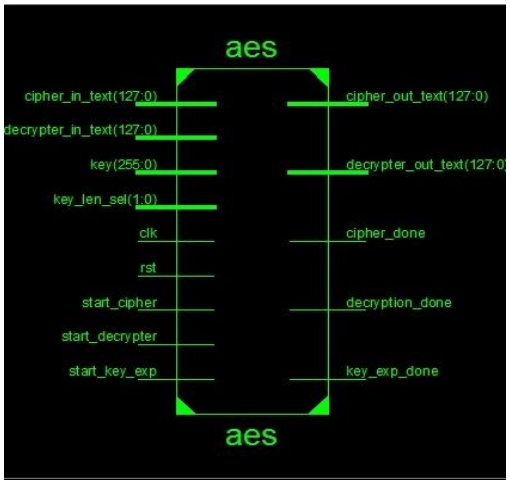

Fig. 10: AES RTL structure using Xilinx ISE Project Navigator

This simulation result shows that RTL schematic black box design of overall design in which cipher_in_text, decrypter_in_text, key acts as basic input for process along with that clk, rst, key_len_sel, start_cipher, start_decrypter, start_key_exp respectively acts as system control inputs. Cipher_out_text, decrypter_out_text are outputs either cipher text or plain text depend on status of cipher_done, decryption_done

Advanced Encryption Standard (AES) is a symmetric key cipher technique used to secure and encrypt operating systems, hard drives, networking systems, files, e-mails, and other similar data. In cryptography, AES consist of three block ciphers taken from a larger collection published originally as Rijndael. Each cipher has a 128-bit block size with three different key sizes of 128, 192, and 256 bits. After expansion of Key length 128 or 192 or 256 bits stored in the memory.

AES Encryption and decryption simulation results

This is the encryption simulation did in Xilinx ISE Project navigator in which Isim Simulator is used.
Initially design is reset; by making reset low key expansion is started. After key expansion done cipher (encryption) process is start by asserting the start_cipher as shown in figure 11 after 11 rounds of iterations in encryption block cipher_done become high, decipher is asserted by giving cipher_text_in to the decryptor to get back the plain text out (original data).

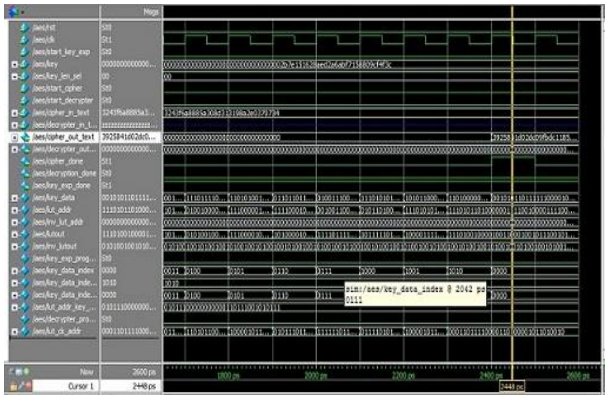

Fig. 11: Encryption with 128 bit plain text and 128 bit key length

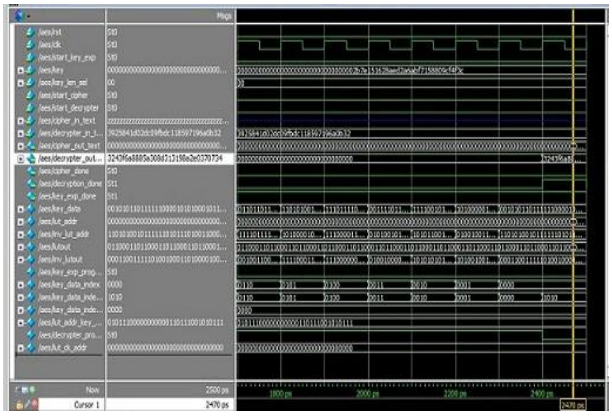

Fig. 12: Decryption with 128 bit plain text and 128 bit key length

Figure 13 show with 192 bit key length encryption process is same as with 128 bit key shown in figure 11 in this 192 bit key length no of rounds increased by 2 from 11 to provide sufficient key data for 13 rounds from 11 rounds of encryption. Decipher is asserted by giving cipher_text_in to the decryptor to get back the plain text out (original data) shown in figure 14.

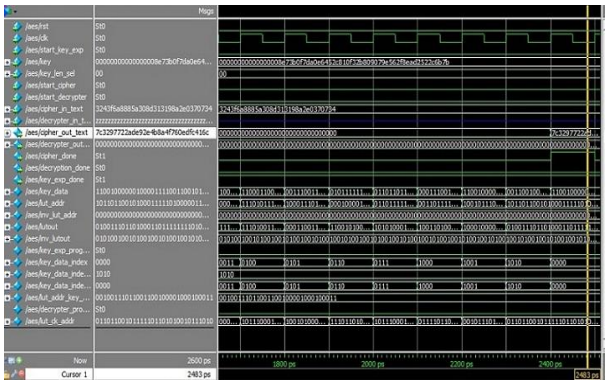

Fig. 13: Encryption with 128 bit plain text and 192 bit key length

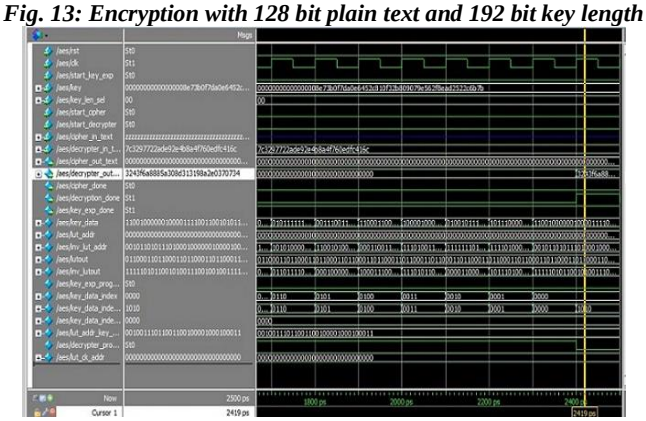

Fig.14: Decryption with 128 bit plain text and 192 bit key length

Figure 15 shows with 256 bit key length makes even strong encryption than the 192, but the process is same as shown in figure 14 except the number rounds in key expansion, encryption as well as decryption increased to 2.

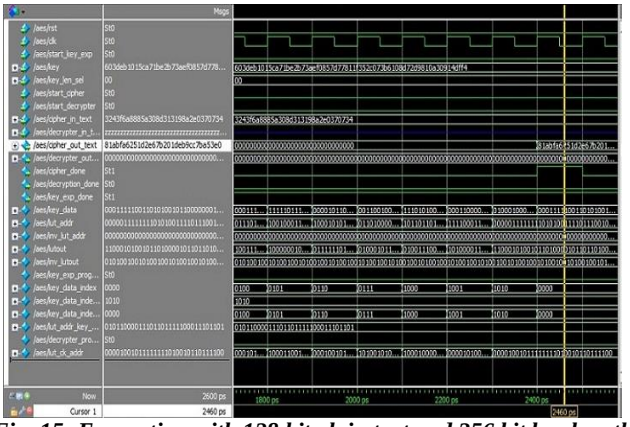

Fig. 15: Encryption with 128 bit plain text and 256 bit key length

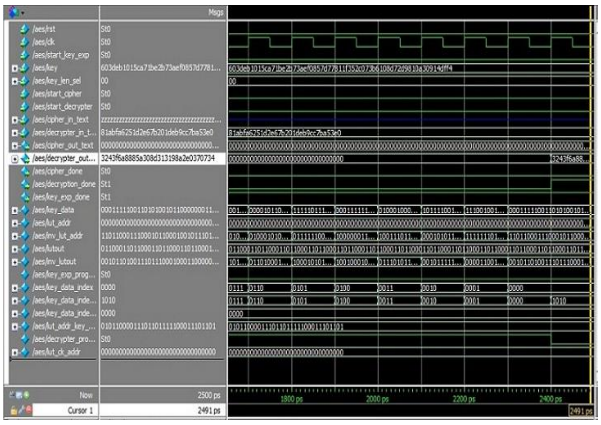

Fig. 16: Decryption with 128 bit plain text and 256 bit key length

CONCLUSION

The combination of a simple, portable and efficient AES cryptographic algorithm implemented in Verilog source code provides an excellent platform for high security applications. A synthesizable Verilog code is developed for the implementation of both encryption and decryption process with different modes. Each program simulation results are verified with ModelSim PE and are synthesized in Xilinx ISE.

REFERENCES

[1] FIPS-197, NIST - National Institute of Standards and Technology," Announcing the ADVANCED ENCRYPTION STANDARD (AES)," http://csrc.nist.gov/publications/fips/fips197/fips-97.pdf, 2001.

[2] H. Trang and N.V. Loi, "An efficient FPGA implementation of the Advanced Encryption Standard algorithm," IEEE Int. Conf. on Computing and Communication Technologies, Research, Innovation and Vision for the Future (RIVF), 2012, pp. 1-4.

[3] J. Daemen and V. Rijmen, AES Proposal: Rijndael, AES Algorithm Submission, September 3, 1999, available at [1].

[4] J. Daemen and V. Rijmen, The block cipher Rijndael, Smart Card research and Applications, LNCS 1820, Springer-Verlag, pp. 288-296.

[5] W. Stallings, "Cryptography and network security principles and practice," Pearson edition 2009, pp. 135-160.

[6] An FPGA Implementation of 30Gbps Security Module for GPON Systems BY Truong Quang Vinh1, Ju-Hyun Park1, Young-Chul Kim1, Kwang-Ok Kim2 2008 IEEE.

[7] FPGA Implementation of AES Encryption and Decryption by Ashwini M. Deshpande, Mangesh S. Deshpande and Devendra N. Kayatanavar, International Conference On "Control, Automation, Communication And Energy Conservation -2009, 4th-6th June 2009. October 2, 2000.

[8] Rijndael: Beyond the AES by Joan Daemen ERG Group – Proton World Belgium Vincent Rijmen Cryptomathic NV, Belgium, and IAIK, Graz University of Technology, Austria.

[9] Report on the Development of the Advanced Encryption Standard (AES) by James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, Edward Roback , October 2, 2000.

[10] AES Proposal by Rijndael Joan Daemen, Vincent Rijmen Joan Daemen Proton World Int.l Zweefvliegtuigstraat 10 B-1130 Brussel, Belgium Vincent Rijmen Katholieke Universiteit Leuven, ESAT-COSIC K. Mercierlaan 94 B-3001 Heverlee, Belgium.

[11] A Versatile Pipelined Hardware Implementation for Encryption and Decryption using Advanced Encryption Standard Nadia Nedjah1 and Luiza de Macedo Mourelle2 by i. Department of Electronics Engineering and Telecommunications, Faculty of Engineering, State University of Rio de Janeiro, Brazil ii. Department of Systems Engineering and Computation, Faculty of Engineering, State University of Rio de Janeiro, Brazil.

[12] Reconfigurable Implementation for the AES Algorithm by Raoel Ashruf, Georgi Gaydadjiev, Stamatis Vassiliadis Computer Engineering Laboratory, Electrical Engineering Department, Delft University of Technology, Delft, The Netherlands.

[13] Cipher and its Verification with FPGA-based Simulation Accelerator by Jae-Gon Lee, Woong Hwangbo, Seonpil Kim and Chong-Min Kyung Department of EECS Korea Advanced Institute of Science and Technology Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea.
[14] Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael Very Well Suited for Small Embedded Applications) by Ga¨el Rouvroy, Franc¸ois-Xavier Standaert, Jean-Jacques Quisquater and Jean-Didier Legat UCL Crypto Group Laboratoire de Micro´electronique Universit´e catholique de Louvain Place du Levant.