ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

Implementation of Area Optimized Advanced Encryption Standard

P.Kishore Raju[1], G.Nirosha[2], K.Bhargavi[3], S.Anjaneyulu[4]

Student, E.C.E, S.K University College of Engineering, Anantapuramu, India[1, 2, 3]

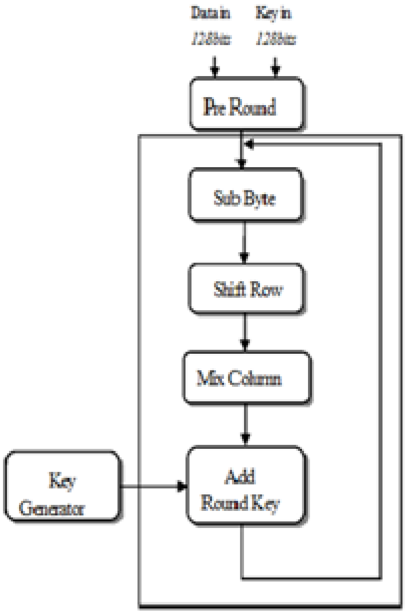
Assistant Professor, E.C.E, SK University College of Engineering, Anantapuramu, India[4]

Abstract: Advanced Encryption Standard (AES) is the current standard for secret key encryption. AES was created by two Belgian cryptographers, Vincent Rijmen and Joan Daemen, to overcome the disadvantages of Data Encryption Standard (DES). The Federal Information Processing Standard 197 used a standardized version of the algorithm called Rijndael for the Advanced Encryption Standard. The algorithm uses a combination of Exclusive-OR operations (XOR), substitution with an S-box transformation, row and column rotations, and a MixColumns. All the transformations of encryption are simulated using an iterative design approach in order to minimize the hardware consumption.To reduce manual operations a verilog code is developed ,synthesis and simulations of code is done by using Xilinx and Modelsim.It was successful because it was easy to implement and could run in a reasonable amount of time on a regular computer.

Key Words: EDK, REAL TIME COMMUNICATION, AGS, SECURITY, XPS, RTOS.

I. INTRODUCTION

The Advanced Encryption Standard, in the following referenced 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



The Rijndael, whose name is based on the names of its two Belgian inventors, **Joan Daemen** and **Vincent Rijmen**, is a **Block cipher**, which means that it works on fixed-length group of bits, which are called blocks. It takes an input block of a certain size, usually 128, and produces a corresponding output block of the same size. The transformation requires a second input, which is the secret key. It is important to know that the secret key can be of any size (depending on the cipher used) and that AES uses three different key sizes: 128, 192 and 256 bits

INPUT	CIPHER KEY
32	2B
43	7E
F6	15
A8	16
88	28
5A	AE
30	D2
8D	A6
31	AB
31	F7
98	15
A2	88
E0	09
37	CF
07	4F
34	3C

DOI 10.17148/IJARCCE.2015.4454

32	88	31	E0
43	59	31	37
F6	30	98	07
A8	8d	A2	34

⊕

19	A0	9A	E9
3D	F4	C6	F8
E3	E2	8D	48
BE	2B	2A	08

⇨

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

SUB-BYTE TRANSFORMATION 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

SUB BYTE TRANSFORMATION:

19	A0	9a	E9
3d	F4	C6	F8
E3	E2	8d	48
Be	2b	2a	08

⇨

D4	*E0*	*B8*	*1e*
27	*Bf*	*B4*	*41*
11	*98*	*5d*	*52*
Ae	*F1*	*E5*	*30*

SHIFT ROW TRANSFORMATION:

D4	E0	B8	1e
27	Bf	B4	41
11	98	5D	52
AE	F1	E5	30

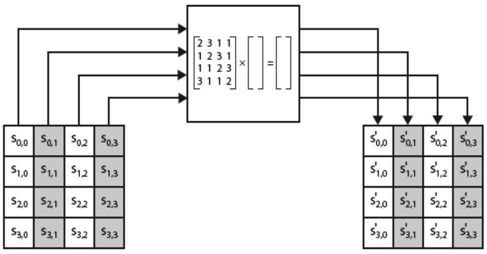
⇨

D4	E0	B8	1E
BF	B4	41	27
5D	52	11	98
30	AE	F1	E5

MIXCOLUMNS:

The MixColumns() transformation operates on the State column-by-column treating each column as a four-term polynomial as described. 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) = $\{03\}x^3+\{01\}x^2+\{01\}x+\{02\}$

The above equation can be described as below:


2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

The following examples are denoted in HEX Mix Column Example During Encryption Input = D4 BF 5D 30

Output(0) = (D4 * 2) XOR (BF*3) XOR (5D*1) XOR (30*1)
= E(L(D4) + L(02)) XOR E(L(BF) + L(03)) XOR 5D XOR 30
= E(41 + 19) XOR E(9D + 01) XOR 5D XOR 30
= E(5A) XOR E(9E) XOR 5D XOR 30
= B3 XOR DA XOR 5D XOR 30
= 04

Output(1) = (D4 * 1) XOR (BF*2) XOR (5D*3) XOR (30*1)
= D4 XOR E(L(BF)+L(02)) XOR E(L(5D)+L(03)) XOR 30
= D4 XOR E(9D+19) XOR E(88+01) XOR 30
= D4 XOR E(B6) XOR E(89) XOR 30
= D4 XOR 65 XOR E7 XOR 30
= 66

Output(2) = (D4 * 1) XOR (BF*1) XOR (5D*2) XOR (30*3)
= D4 XOR BF XOR E(L(5D)+L(02)) XOR E(L(30)+L(03))
= D4 XOR BF XOR E(88+19) XOR E(65+01)
= D4 XOR BF XOR E(A1) XOR E(66)
= D4 XOR BF XOR BA XOR 50
= 81

Output(3) = (D4 * 3) XOR (BF*1) XOR (5D*1) XOR (30*2)
= E(L(D4)+L(3)) XOR BF XOR 5D XOR E(L(30)+L(02))
= E(41+01) XOR BF XOR 5D XOR E(65+19)
= E(42) XOR BF XOR 5D XOR E(7E)
= 67 XOR BF XOR 5D XOR 60
= E5

The multiplication mentioned above is performed over a Galois Field. The mathematics behind this is beyond the scope of this paper. This section will instead concentrate On the implementation of the multiplication which can be done quite easily with the use of the following two tables

E Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	01	03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35
1	5F	E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA
2	E5	34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31
3	53	F5	04	0C	14	3C	44	CC	4F	D1	68	BB	D3	6E	B2	CD
4	4C	D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88
5	83	9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A
6	B5	C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3
7	FE	19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0
8	FB	16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41
9	C3	5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75
A	9F	BA	D5	64	AC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80
B	9B	B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54
C	FC	1F	21	63	A5	F4	07	09	1B	2D	77	99	B0	CB	46	CA
D	45	CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E
E	12	36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17
F	39	4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01

Mix COLUMN OUTPUT:

L Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		00	19	01	32	02	1A	C6	4B	C7	1B	68	33	EE	DF	03
1	64	04	E0	0E	34	8D	81	EF	4C	71	08	C8	F8	69	1C	C1
2	7D	C2	1D	B5	F9	B9	27	6A	4D	E4	A6	72	9A	C9	09	78
3	65	2F	8A	05	21	0F	E1	24	12	F0	82	45	35	93	DA	8E
4	96	8F	DB	BD	36	D0	CE	94	13	5C	D2	F1	40	46	83	38
5	66	DD	FD	30	BF	06	8B	62	B3	25	E2	98	22	88	91	10
6	7E	6E	48	C3	A3	B6	1E	42	3A	6B	28	54	FA	85	3D	BA
7	2B	79	0A	15	9B	9F	5E	CA	4E	D4	AC	E5	F3	73	A7	57
8	AF	58	A8	50	F4	EA	D6	74	4F	AE	E9	D5	E7	E6	AD	E8
9	2C	D7	75	7A	EB	16	0B	F5	59	CB	5F	B0	9C	A9	51	A0
A	7F	0C	F6	6F	17	C4	49	EC	D8	43	1F	2D	A4	76	7B	B7
B	CC	BB	3E	5A	FB	60	B1	86	3B	52	A1	6C	AA	55	29	9D
C	97	B2	87	90	61	BE	DC	FC	BC	95	CF	CD	37	3F	5B	D1
D	53	39	84	3C	41	A2	6D	47	14	2A	9E	5D	56	F2	D3	AB
E	44	11	92	D9	23	20	2E	89	B4	7C	B8	26	77	99	E3	A5
F	67	4A	ED	DE	C5	31	FE	18	0D	63	8C	80	C0	F7	70	07

KEY GENERATION:

04	E0	48	28
66	CB	F8	06
81	19	D3	26
E5	9A	7A	4C

KEY INPUT:

2B	28	AB	09
7E	AC	F7	CF
15	D2	15	4F
16	AB	88	3C

Steps:

1. Take Last Column Of Key Input:
   09 CF 4F 3C
2. S-byte Transformation(Using 'S'Table)
   01 A 84 EB
3. Shift left to One Position
   8A 84 EB 01

ROUND	RCON VALUE
R0	01
R1	02
R2	04
R3	08
R4	10
R5	20
R6	40
R7	80
R8	1B
R9	36

4. Adding RCON Value To Step 3
   [8A] [84] [EB] [01]
   ⟹ [01] [00] [00] [00]

As total is 32 bits '01' is added to MSB and Remaining 0's

8A → 10001010
01 → 00000000
1000 101
8 B

5. 8B 84 EB 01 ⊕olumns
   Each XOR with 4 columns
   8b 84 eb 01
   2b 7e 15 16
   8B → 10001011   84→10000100
   2B → 00101011   7e→01111110
   1010 0000   1111 1110
   A 0   F A
   EB→11101011
   15→ 00010101
   1111 1110
   F E

KEY EXPANDED OUTPUT:

A0	88	23	29
FA	54	A3	6C
FE	2C	39	76
17	B1	39	05

04	E0	48	28
66	CB	F8	06
81	19	D3	26
E5	9A	7A	4C

ADD ROUND KEY OUTPUT:

A4	68	6B	02
9C	9F	5B	6A
7F	35	EA	50
F2	2B	43	49

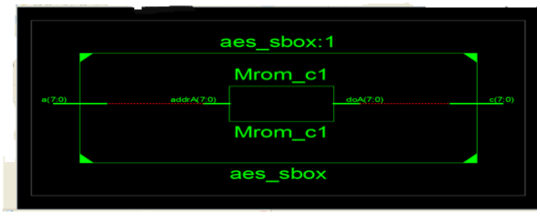
⇩

First Cipher Key Output

Implementing the algorithm manually is diffficult. In order to reduce this, we are converting this algorithm into a verilog code and can be synthesised,simulated using XILINX and MODEL SIM tools.

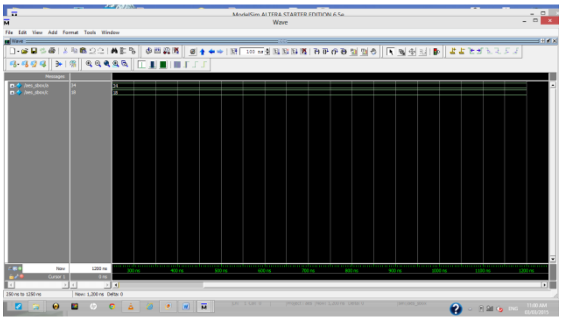
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

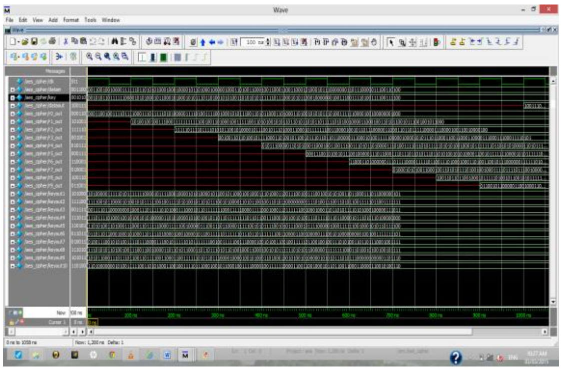


S-BOX BLOCK DIAGRAM :

S-BOX SIMULATION:



CIPHER SIMULATION:



CONCLUSION

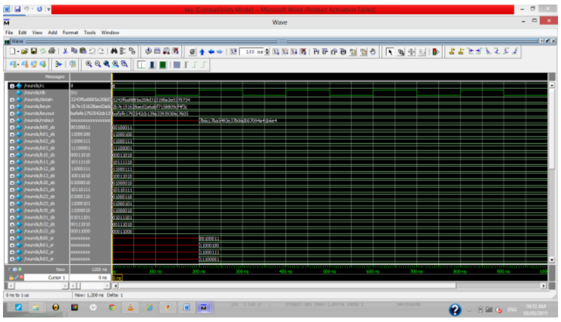
The Rijndael algorithm was chosen as the new Advanced Encryption Standard (AES) for several reasons. The purpose was to create an algorithm that was resistant against known attacks, simple, and quick to code. Choosing to use field GF(2)8 was a very good decision. The block size and key size can vary making the algorithm versatile.

Optimized and Synthesizable VHDL code is developed for the implementation of encryption process. Each program is tested with some of the sample vectors provided by NIST and output results are perfect with minimal delay. Therefore, AES can indeed be implemented with reasonable efficiency on an FPGA, with the encryption taking an average of 320 ns (for every 128 bits).

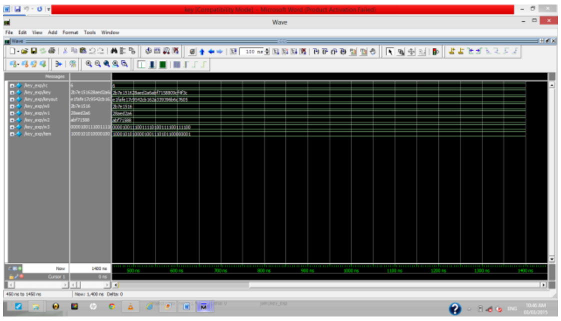
The time varies from chip to chip and the calculated delay time can only be regarded as approximate. Adding data pipelines and some parallel combinational logic in the key scheduler and round calculator can further optimize this design.

AES was originally designed for non-classified U.S. government information, but, due to its success, AES-256 is usable for top secret government information [11]. As of July 2009, no practical attacks have been successful on AES [12].

ROUNDS SIMULATION:



KEY SIMULATION:



REFERENCES

[1]. Deshpande, A.M. Deshpande, M.S. Kayatanavar, D.N. "FPGA implementation of AES encryption and decryption", IEEE Transactions, Print ISBN: 978-1-4244-4789-3 ,Jun 2009.

[2]. Muhammad H. Rais and Syed M. Qausim "Efficient Hardware Realization of Advanced Encryption Standard Algorithm using FPGA", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.9, September 2009

[3]. Marko Mali, Franc Novak and Anton Biasizzo "Hardware Implementation of AES Algorithm", Journal of Electrical Engineering, VOL. 56, NO. 9-10, 2005, 265-269.

[4]. Rajender Manteena, "A VHDL Implementation of the Advanced Encryption Standard- Rijndael Algorithm", College of Engineering University of South Florida, 2004.

DOI 10.17148/IJARCCE.2015.4454