International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-3 Issue-4, April 2014

FPGA Based Hardware Implementation of Encryption Algorithm

M.Natheera Banu

Abstract: ***Reprogrammable devices such as Field Programmable Gate Arrays (FPGA) are used for hardware implementations of cryptographic algorithm. This paper presents an FPGA based Hardware implementation of advanced encryption standard (AES) with 128-bit key as a constant which is used for encrypting the text file and image for secure transmission. Timing report for the files are taken and conclude that text file of 128 bit size is taking less time to encrypt and decrypt compare to the image file. Synthesizing and implementation (Translate, Map and Place and Route) of the VERILOG code is carried out on Xilinx - Project Navigator ISE 12.3 software.***

Index Terms — AES, decryption, encryption, image and text.

I. INTRODUCTION

Cryptography is about the avoidance and recognition of fraud and other cruel activities. Symmetric-key cryptography, also called secret key cryptography. It involves the use of a secret key known only to the users. It is considered by the use of a single key to perform both the encrypting and decrypting of data. On October, 2, 2000, The National Institute of Standards and Technology (NIST) announced Rijndael as the new Advanced Encryption Standard (AES).The Predecessor to the AES was Data Encryption Standard (DES) which was considered to be unsecure because of its weakness to brute force attacks. DES was a standard from 1977 and stayed until the mid1990's and it was clear that the DES's 56-bit key was not enough to prevent attacks. Older cryptographic methods, such as DES, do not have a enough key space to applications where high security is needed. To overcome this situation, the National Institute of Standards and Technology (NIST) created a new encryption standard. The methods were proposed by Joan Daemon and Vincent Rijman, which are called Rijndael. The Rijndael algorithm, the Advanced Encryption Standard (AES) provides a symmetric key cryptography that allows for the encryption and decryption of blocks of data. As a symmetric system, the secret key must be shared between the sender and receiver in order for communication to be possible.AES algorithm is generally applied in the financial field in domestic, such as realizing legal encryption in ATM, magnetism card and intelligence card.

Manuscript published on 30 April 2014.
* Correspondence Author (s)
M.Natheera Banu*, Department of Electronics and Communication, Coimbatore institute of Engineering And Technology, Coimbatore, Anna University, India.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

II. AES RIJNDAEL ALGORITHM

AES is an iterated block cipher with a fixed size of 128 and a variable key length. The state is a rectangular array of bytes and the block Size is of 128 bits, which is 16 bytes; the Rectangular array is of dimensions 4x4.The key is similarly pictured as a rectangular array with four rows. The number of columns of the key, denoted Nk , is equal to the key length divided by 32.It is very important to know that the input bytes are mapped onto the state bytes in the order a0,0,a1,0, a2,0, a3,0, a0,1, a1,1, a2,1, a3,1 ..... and the bytes of the key are mapped onto the array in the order k0,0, k1,0, k2,0, k3,0, k0,1, k1,1,k2,1, k3,1..At the end of the Operation, the output is extracted from the state by taking the state bytes in the same order. AES uses a variable number of rounds, which are fixed: A key of size 128 has 10 rounds. A key of size 192 has 12 rounds. A key of size 256 has 14 rounds. On the encryption algorithm, there will be four processes:

Add Round Key, sub bytes, Shift Rows and Mix Columns. But, on the last stage, the Mix Columns operation is unseen. The decryption algorithm will use the inverse operations:

Inverse Add Round Key, Inverse Sub Bytes, Inverse Mix Columns and Inverse Shift Rows. In the decryption also, the Inverse Mix Columns is unseen on the last stage. The general AES algorithm architecture is shown in fig 1

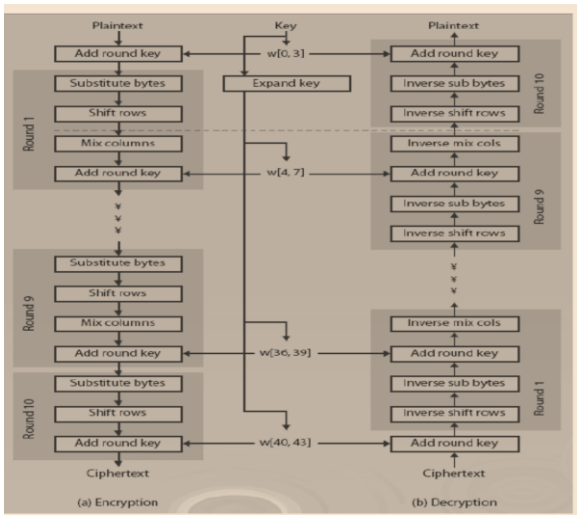

Fig 1: GENERAL AES ARCHITECTURE

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

For both encryption and decryption, the process begins with an Add Round Key, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages. Only the Add Round Key stage makes use of the key. For this reason, the cipher begins and ends with an Add Round Key stage. The final round of both encryption and decryption consists of only three stages (mix column and inverse mix column).

A. Subbytes (substitution bytes)

The substitute byte transformation, called Sub Bytes.AES defines a matrix of byte values, called an S-box that contains all possible 256 8-bit values. Each individual byte of State 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 rightmost 4 bits are used as a column value. These row and column values provide as indexes into the S-box to select an 8-bit output value for the next process. The S-box operation (Fig 2) and look up table of S-box (Fig 3, Fig 4) are shown.

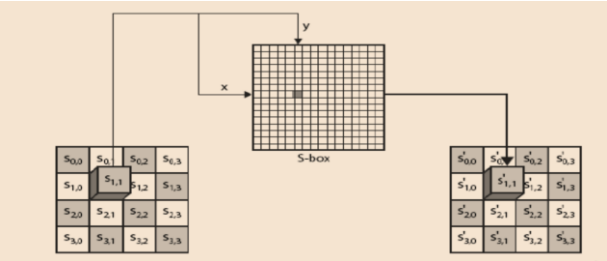

Fig 2: SUBSTITUTION BYTES

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 3: S BOX TABLE

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 4: INVERSE S BOX TABLE

B. Shiftrows

In shift row, the output of sub bytes is shifted as indicated in fig 5.In first Row no shifting takes place. The second Row is shifted by one position. The third Row is shifted by two positions. The fourth Row is shifted by three positions. The decryption algorithm performs the inverse operation.

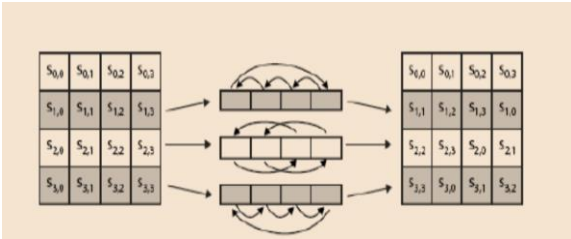

Fig 5: SHIFT ROW OPERATION

C. Mixcolumns

There are two parts to this step. The first will explain which parts of the state are multiplied against which parts of the matrix. The second will explain how this multiplication is implemented over a Galois Field.

D. Matrix multiplication

The state is arranged into a 4 row table. The multiplication is performed one column at a time (4 bytes). Each state in the column is multiplied against every value of the matrix in the row. The results of these multiplications are XORed together to produce only 4 bytes for the next state.

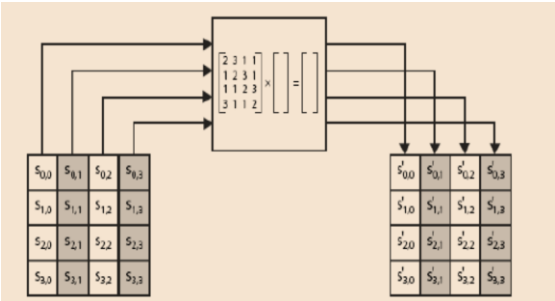

Fig 6: MIX COLUMN OPERATION

E.Galois field multiplication

In the Mix column operation, the state bytes are treated as polynomials of Galois Field algebra GF (2^8). The operation can be implemented using two tables called E Table (fig 7) and L Table (fig 8). The individual block values from the shift row are taken whereas their corresponding values are obtained from L TABLE. The L TABLE values are given as the input for the E TABLE and corresponding output are obtained from the same table. The output of multiplication is given as input to the next state.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

	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	B8	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

Fig 7: E TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D
0		00	19	01	32	02	1A	C6	4B	C7	1B	68	33	EE
1	64	04	E0	0E	34	8D	81	EF	4C	71	08	C8	F8	69
2	7D	C2	1D	B5	F9	B9	27	6A	4D	E4	A6	72	9A	C9
3	65	2F	8A	05	21	0F	E1	24	12	F0	82	45	35	93
4	96	8F	DB	BD	36	D0	CE	94	13	5C	D2	F1	40	46
5	66	DD	FD	30	BF	06	8B	62	B3	25	E2	98	22	88
6	7E	6E	48	C3	A3	B6	1E	42	3A	6B	28	54	FA	85
7	2B	79	0A	15	9B	9F	5E	CA	4E	D4	AC	E5	F3	73
8	AF	58	A8	50	F4	EA	D6	74	4F	AE	E9	D5	E7	E6
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

Fig 8: L TABLE

F. Addroundkey

The AddRoundKey operation performs a simple XOR between the current state data values and the round key for the current round. The round key is obtained from the key selector block based on the value of the key expansion and the current round.

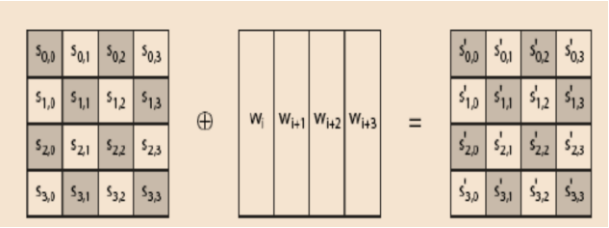

Fig 9: ADD ROUND KEY PROCESS

G. Key expansion

In the AES algorithm, the key expansion module is used for generating keys for every round. The initial key (given key) is divided into Nk words (key0, key1 ... .keyNk-1) which are used as initial words. With the help of these initial words rest of the words are generated iteratively. The key expansion procedure can be described by the pseudo code listed below:

```
KeyExpansion(byte CipherKey[4*Nk], word
w[Nb*(Nr+1)], Nk)
begin
word temp
i = 0
while (i < Nk)
w[i] = word(key[4*i], key[4*i+1], key[4*i+2],
key[4*i+3])
i = i+1
end while
i = Nk
while (i < Nb * (Nr+1)]
temp = w[i-1]
if (i mod Nk = 0)
temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
else if (Nk > 6 and i mod Nk = 4)
temp = SubWord(temp)
end if
w[i] = w[i-Nk] xor temp
i = i + 1
end while
end
```

Fig 10: KEY EXPANSION PSEUDO CODE

The Key Expansion operation consists of three operations. The first operation, Rot Word, makes a one byte circular shifting on the word. The second operation, Sub Word replaces each byte of the input word according to the S-Box. The third operation consists of XOR operations, as shown in Fig 11.

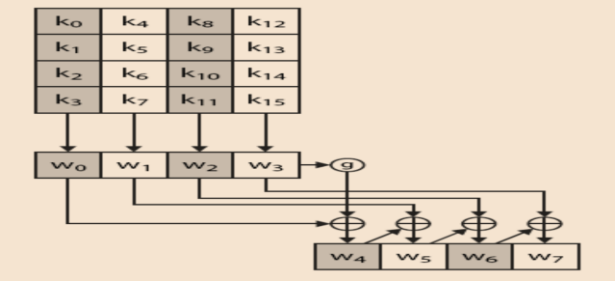

FIG 11: KEY EXPANSION PROCESS

III. IMPLEMENTATION

A.AES encryption

To implement the 128 bit AES algorithm using VERILOG code in Xilinx 12.3.The encryption parameters are the input plaintext, the key size and the output cipher text. First step, we have to map the 16 byte input plaintext in the correct manner to the 4*4 byte state, and calculate the number of rounds based on the key Size and expand the key using key scheduling. At first round, plaintext and key is XORed, the remaining nine rounds which has to apply all four operations such as Substitute Bytes, Shift rows, Mix columns, Add round key. The tenth round Mix column is not included round key was generated each time based on key scheduling algorithm. Here simple XOR of each byte of the key with the corresponding byte of the state is done to get cipher text of the Encryption algorithm.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

B.AES decryption

For AES Decryption, the same encryption process occurs in reverse order. In decryption the key schedule remains the same; the only operations we need to implement are the Inverse subBytes, shiftRows and mixColumns, while addRoundKey stays the same.

C.Text encryption and decryption

A text files which consists of various words. Each word is of 128 bit length which is stored in memory location as a hexadecimal value. This is the input to the AES encryption algorithm which is combined with constant 128 bit key to produce a cipher text. The encrypted values of corresponding text are stored in another memory location which will be an input for AES decryption algorithm to produce an original text.

D.Image encryption and decryption

The 128 bit AES algorithm is used to encrypt and decrypt the image using MATLAB and XILINX.The image is converted into binary numbers in MATLAB which is an input to the 128 bit AES Encryption algorithm to produce encrypted values. These values are stored in memory location and given as an input to MATLAB to produce an encrypted image and same values are given to AES decryption algorithm to produce original values which are then again feed into the MATLAB to produce an original image.

IV. RESULTS

A.Simulation results

Fig 12 shows the simulation output for text file encryption

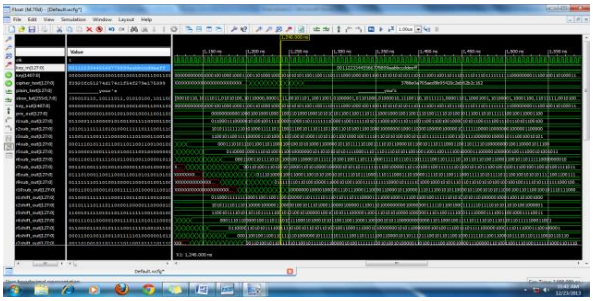

FIG 12: TEXT ENCRYPTION OUTPUT

Fig 13 shows the simulation output for text file decryption

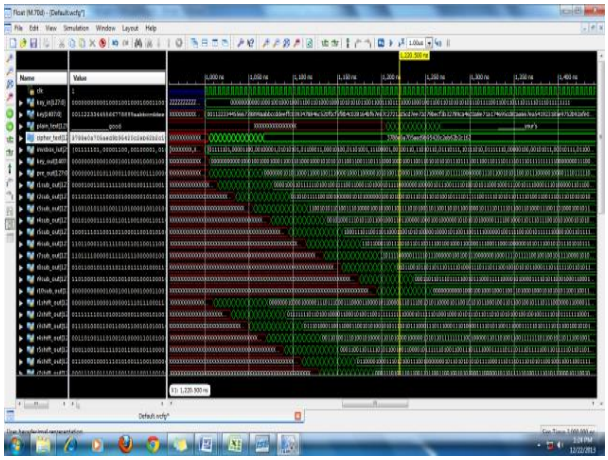

FIG 13: TEXT DECRYPTION OUTPUT

Fig 14 shows the simulation output for image file encryption

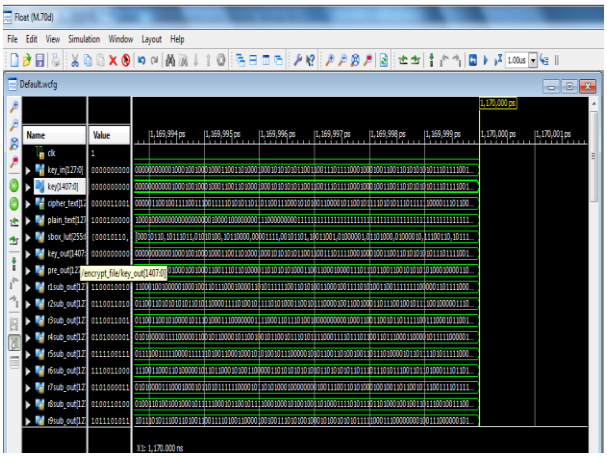

FIG 14: IMAGE ENCRYPTION OUTPUT

Fig 15 shows the simulation output for image file decryption

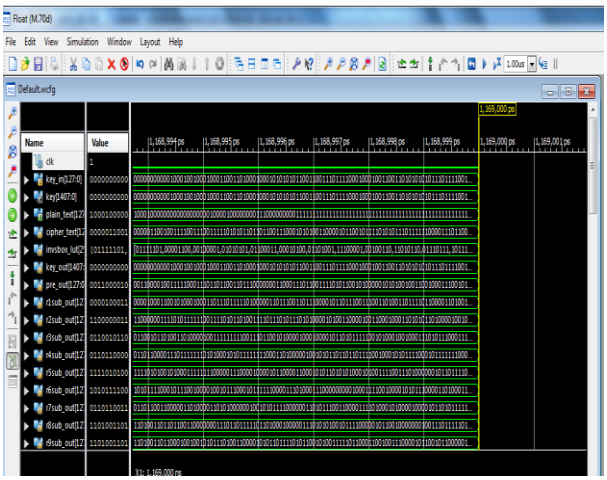

FIG 15: IMAGE DECRYPTION OUTPUT

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

B. Matlab output

Fig 16 shows the output of image encryption and decryption in MATLAB.

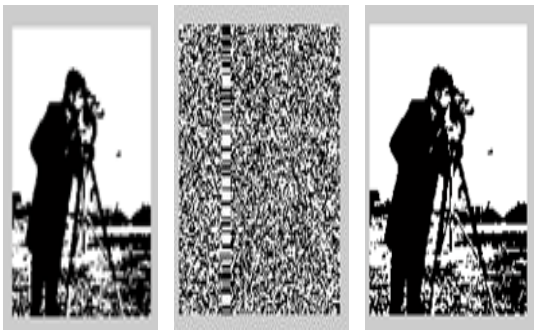

FIG 16 A) INPUT IMAGE B) ENCRYPTED IMAGE C) DECRYPTED IMAGE

C. Text encryption and decryption

Table 1: synthesis utilization

ENCRYPTION: SLICE LOGIC UTILIZATION

Number of Slice Registers	5351 out of 28800 18%
Number of Slice LUTs	6457 out of 28800 22%
Number of LUT Flip Flop pairs used	7745
Number with an unused Flip Flop	2394 out of 7745 30%
Number with an unused LUT	1288 out of 7745 16%
Number of fully used LUT FF pairs	4063 out of 7745 52%
Number of unique control sets	3

IO UTILIZATION

Number of IOs	257
Number of bonded IOBs	257 out of 480 53%

DECRYPTION: SLICE LOGIC UTILIZATION

Number of Slice Registers	5530 out of 28800 19%
Number of Slice LUTs	9199 out of 28800 31%
Number of LUT Flip Flop pairs used	10564
Number with an unused Flip Flop	5034 out of 10564 47%
Number with an unused LUT	1365 out of 10564 12%
Number of fully used LUT FF pairs	4165 out of 10564 39%
Number of unique control sets	3

IO UTILIZATION

Number of IOs	257
Number of bonded IOBs	257 out of 480 53%

D. Image encryption and decryption

Table 2: synthesis utilization

ENCRYPTION: SLICE LOGIC UTILIZATION

Number of Slice Registers	6240 out of 30064 20%
Number of Slice LUTs	9087 out of 15032 60%
Number of LUT Flip Flop pairs used	10660
Number with an unused Flip Flop	4420 out of 10660 41%
Number with an unused LUT	1573 out of 10660 14%
Number of fully used LUT FF pairs	4667 out of 10660 43%
Number of unique control sets	1

IO UTILIZATION

Number of IOs	385
Number of bonded IOBs	385 out of 226 170%

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

DECRYPTION: SLICE LOGIC UTILIZATION

Number of Slice Registers	5376 out of 93120 5%
Number of Slice LUTs	7466 out of 46560 16%
Number of LUT Flip Flop pairs used	8946
Number with an unused Flip Flop	3570 out of 8946 39%
Number with an unused LUT	1480 out of 8946 16%
Number of fully used LUT FF pairs	3896 out of 8946 43%
Number of unique control sets	1

IO UTILIZATION

Number of IOs	385
Number of bonded IOBs	385 out of 240 160%

V. TIMING REPORT

File (bytes)	Encryption time	Decryption time
Text file (2048)	2.595ns	2.606ns
Image (2048)	374.656ns	361.856ns

Table 3: Time taken for 128 bit

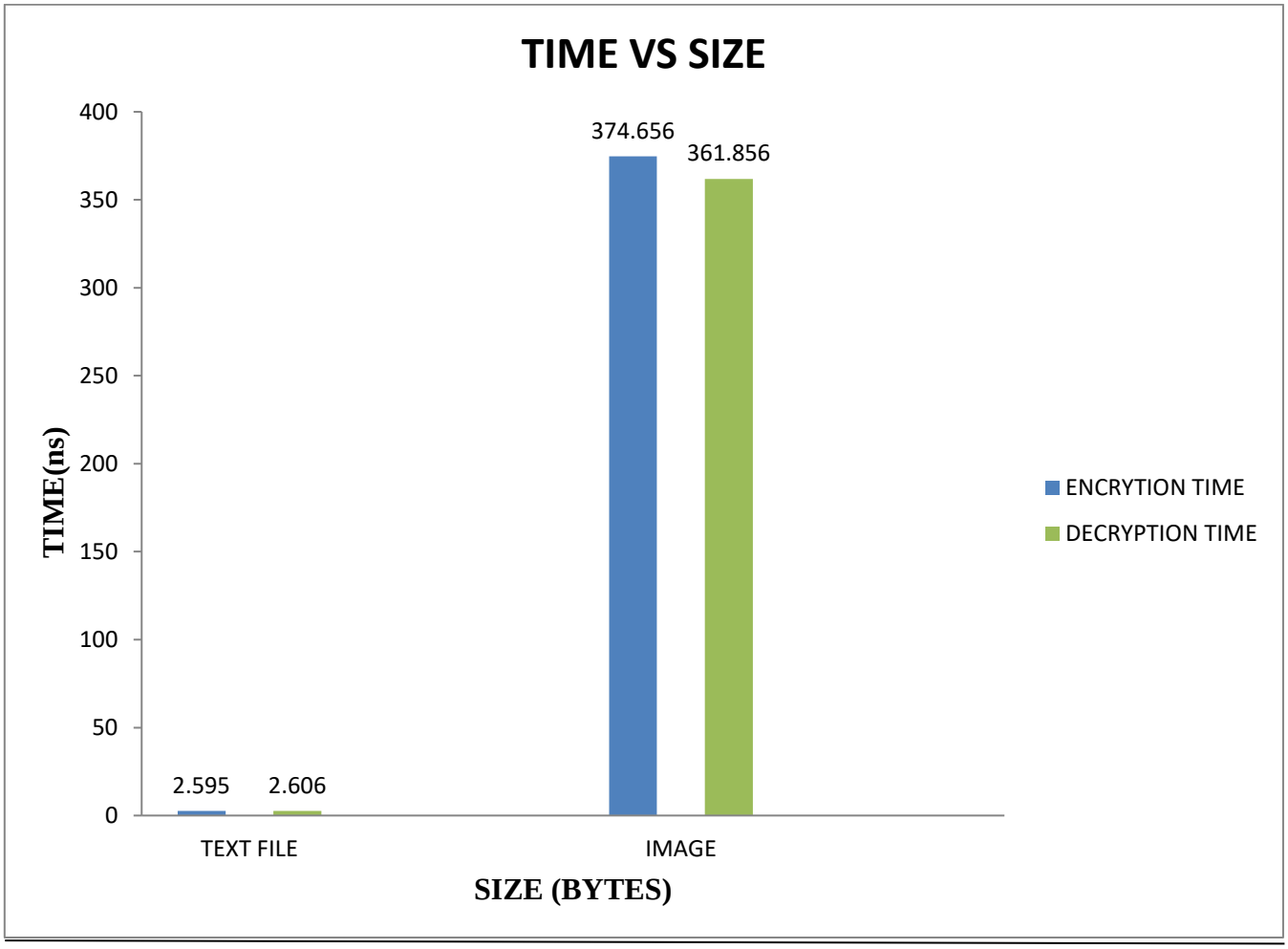

FIG 17: COMPARISION CHART FOR TEXT AND IMAGE

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

Here the time taken for the text and image file are compared which shown in fig 17 and tabulated in table 3. The encryption and decryption time for text file is less compared to image file. The parameters such as no.of.slice registers, no.of slice LUT, no.of.LUT FF pairs used are compared and concluded that text file contains less compared to image file and these are tabulated in table 1 and 2.

VI.CONCLUSION

This paper presents an FPGA based Hardware implementation of advanced encryption standard (AES) with 128-bit key as a constant which is used for encrypting the text file and image for secure transmission. Optimized and Synthesizable VERILOG code is developed for the implementation of both encryption and decryption process. Therefore, AES can indeed be implemented with reasonable efficiency on an FPGA, with the encryption and decryption taking an average of 2.595ns and 2.606 ns respectively (for every 2048 bytes of text file) and 374.656ns and 361.856ns (2048 bytes of image file).

REFERENCES

1. Xinmiao Zhang and Keshab K. Parhi "HIGH-SPEED VLSI ARCHITECTURES FOR THE AES ALGORITHM" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 12, NO. 9, SEPTEMBER 2004.
2. Jia Zhao, Xiaoyang Zeng*, Jun Han, Jun Chen State-Key Lab ofASIC and System,"VERY LOW-COST VLSI IMPLEMENTATION OF AES ALGORITHM" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS,2006
3. Bin Liu, and Bevan M. Baas "PARALLEL AES ENCRYPTION ENGINES FOR MANY- CORE PROCESSOR ARRAYS IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 3, MARCH 2013
4. Stefan Mangard Student Member, IEEE, Manfred Aigner, and Sandra Dominikus "A HIGHLY REGULAR AND SCALABLE AES HARDWARE ARCHITECTURE" IEEE TRANSACTIONS ON COMPUTERS, VOL. 52, NO. 4, APRIL 2003
5. P.Karthigaikumar Soumiya Rasheed "SIMULATION OF IMAGE ENCRYPTION USING AES ALGORITHM" IJCA Special Issue on Computational Science - New Dimensions & Perspectives" NCCSE, 2011.
6. Manoj.B, Manula N Harihar "IMAGE ENCRYPTION AND DECRYPTION USING AES" International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-1, Issue-5, June 2012.
7. Jawad Ahmad and Fawad Ahmed "EFFICIENCY ANALYSIS AND SECURE VALUATION OF IMAGE ENCRYPTION SCHEMES" International Journal of Video & Image Processing and Network Security IJVIPNS-IJENS Vol:12 No:04.
8. Mr. Antosh M. Dyade, Prof. Raafiya Gulmeher "MODIFIED SHIFTROW TRANSFORMATION OF AES FOR IMAGE ENCRYPTION" International Journal of Communications And Engineering Vol. 4 Issue 3, Sept. 2013
9. Sourabh Singh, Anurag Jain "AN ENHANCED TEXT TO IMAGE ENCRYPTION TECHNIQUE USING RGB SUBSTITUTION AND AES" International Journal of Engineering Trends and Technology (IJETT) - Volume4Issue5- May 2013
10. Amandeep Kaur, Puneet Bhardwaj, Naveen Kumar" FPGA IMPLEMENTATION OF EFFICIENT HARDWARE FOR THE ADVANCED ENCRYPTION STANDARD" International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-2, Issue-3, February 2013
11. Chityala Prathyusha, P. Sharmila Rani "IMPLEMENTATION OF FAST PIPELINED AES ALGORITHM ON XILINX FPGA" International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064
13. Mg Suresh, Dr.Nataraj.K.R" AREA OPTIMIZED AND PIPELINED FPGA IMPLEMENTATION OF AES ENCRYPTION AND DECRYPTION" International Journal Of Computational Engineering Research (ijceronline.com) Vol. 2 Issue.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.