2010 International Conference on Computational Intelligence and Communication Networks

Hardware Implementation of Advanced Encryption Standard

Yogesh Kumar
IITM, Gwalior

Yogeshelex90@yahoo.co.in

Prashant Purohit
RJIT, BSF Academy
Tekanpur, Gwalior
er.prashantpurohit@gmail.com

Abstract: **Advanced Encryption Standard, a federal information processing standard is an approved cryptographic algorithm that can be used to protect electronic data. The AES can be programmed in software or built with pure hardware. However field programmable gate array offers a quicker and more customizable solution. This paper presents the AES algorithm with regard to FPGA and the very high speed integrated circuit hardware description language. Modelsim SE PLUS 5.7g soft ware is used for simulation and optimization of the synthesizable VHDl code. Synthesizing and implementation of the code carried out on Xilinx –project navigator, ISE 8.2i suite. All the transformations of both encryption and decryption are simulated using an iterative design approach in order to minimize consumption. This paper talks of AES 128 bit block and 128 bit cipher key and is implemented on Spartan 3 FPGA.**

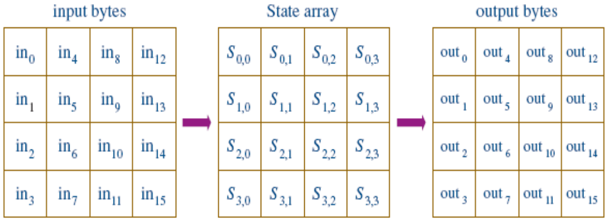
***Index Terms*—FPGA, VHDL**

I. Introduction

Cryptography plays an important role in the security of data transmission. Cryptography is a discipline in mathematics pertaining to providing information security. Cryptographic methods are used for providing three forms of security namely confidentiality, data integrity and authentication. Confidentiality refers to protection of information from unauthorized access [1]. An undesired communicating party, called adversary must not be able to access the private information. Data integrity; ensures that information has not been manipulated in any unauthorized way. Authentication; methods are studied in two groups: entity authentication and message authentication. Message authentication provides means of detecting any modifications to the message. Entity authentication assures the receiver of a message, about both the identity of the sender and his active participation. To have better and advanced algorithm for this purpose, the AES algorithm is considered for efficient software and hardware implementations. This paper presents hardware implementation for the AES algorithm. Compared to software implementations, hardware implementation provided more physical security as well as higher speed. Different application of the AES algorithm may require different speed/area trade-offs. Some application such as smart cards and cellular phones, require small area .other applications WWW severs and ATMs, are speed critical. Some other application such as digital video recorder, require an optimization of speed /area ratio.

II. The AES algorithm

The AES algorithm is a symmetric-key block cipher in which both the sander and receiver use a single key to encrypt and decrypted the information. Although in [1] the block length can be 128, 192, or 256 bits, the AES algorithm [2] only adopted the block length of 128 bits. Meanwhile, the key length can be 128,192 or 256 bits. The AES algorithm, internal operation is performed on two dimensional array of Bytes called state, and each byte consist of 8 bits the state consist of four rows of bytes and each row has Nb bytes. Each bytes denoted by Si, j (0<i<4,0<j<Nb). Since the block length is 128 bits, each row of state contains Nb =128/(4*8)=4 bytes. The four bytes in each column of the state array form a 32 bit word , with the row number as the index for the four bytes in each word. At the beginning of encryption or decryption, the array of input bytes is mapped to the state array as illustrated in fig.1, assuming a 128 bit block can be expressed as 16 bytes: in 0, in 1, in 2 , in 3……………in 15. The encryption / decryption are performed on the state, at the and of which the final value is mapped to the output bytes array out 0 , out 1 , out 2 ,………..out 15.



input bytes

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}

State array

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

output bytes

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}

Fig 1 mapping of input bytes, state array and output bytes

The key of the AES algorithm can be mapped to four rows of bytes in a similar way , except the number of bytes in each row denoted by Nk can be 4,6,or 8 when the length of the key, K, is 128, 192 or 256 bits, respectively. The Aes algorithm is an iterative algorithm . Each iteration can be called a rounds, Nr, is 10 when Nk =4, Nr=12 when NK=6, and Nr 14 when Nk=8.

A. *Step for Encryption and Decryption*

The different step involve in algorithm are as follows:

1. Initial round –modulo 2 addition of the 4X4 state matrix

And the round key, also represented as a 4X4 matrix is performed.

2. Rounds:
   Sub byte: A non linear substitution steps where each byte is replaced with another according to a look up table called S-Box. the AES S-Box is a 256 entry

978-0-7695-4254-6/10 $26.00 © 2010 IEEE
DOI 10.1109/CICN.2010.89

table composed of two transformations such as multiplicative inverse in GF(2^8) and an affine transformation. For decryption, inverse S-Box is obtained by applying inverse affine transformation followed by multiplicative inversion in GF (2^8); where affine transformation is a transformation consisting of multiplication by a matrix followed by the addition of vector. Then shift rows: a transition step where rows of the state is shifted cyclically to the left using 0,1,2 and 3 byte offset for encryption while for decryption , rotation is applied to right, after that mix column: a matrix operation which operates on the columns of the state, combining the four bytes in each column. Each column of the state matrix is multiplied by a constant fixed matrix

$$\begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix}$$

For the Inverse Mixcolumn the multiplication matrix is :

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

AddRound key: each byte of the state is combined with the round key; each round is derived from the cipher key using a key schedule. The above four round are iteratively carried out nine times. GF (2^8). In final rounds: each of the transformation Sub Byte, Shift Rows, and Add Round Key is performed only once. It is noted that the mix columns Operation is not performed in this round.

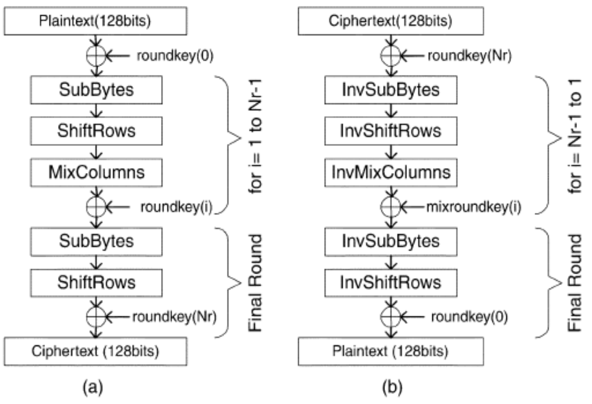


Fig 2 The structure of AES encryption /decryption

III. AES IMPLEMENTATION

A. AES Implementation details

The Algorithm is implemented in ECB mode.

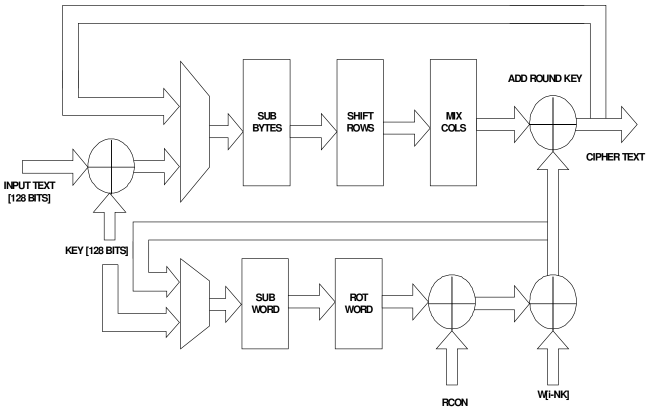


Figure 3 Architectural block diagram

The decryption process follows virtually the same order as encryption except for another round of mix columns on the generated keys before giving them to the add round key step. .

THE ENCRYPTION/DECRYPTION SEQUENCE

Input data and key is fed in two blocks of 64 bits in consecutive clock cycles with the load signal. 64 bits of input and key are read in the pos-edge after the load signal goes high and another block of 64 bits of input and key are read in the pos-edge after the load signal goes low. Hence the complete data and key is loaded only when the load signal makes a low-high-low transition (basically a pulse). The process starts once the start signal is pulsed and the output is validated with 'done' signal 13 clock cycles after the 'start' signal goes low. ‘Done’ remains high until the next start cycle.

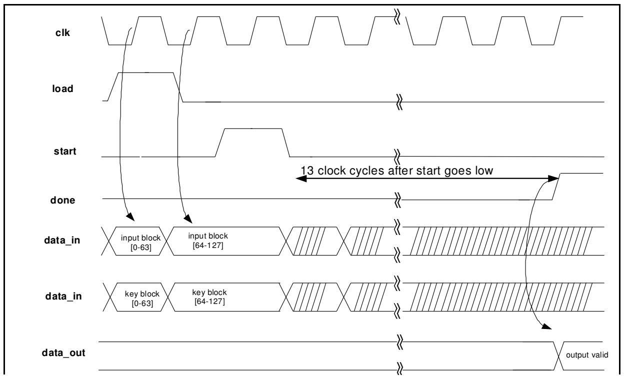
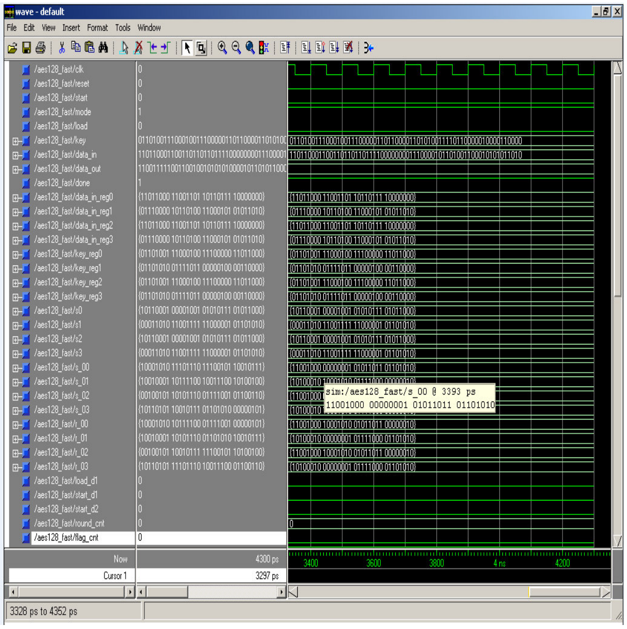


Figure -4 Process sequence for encryption/decryption

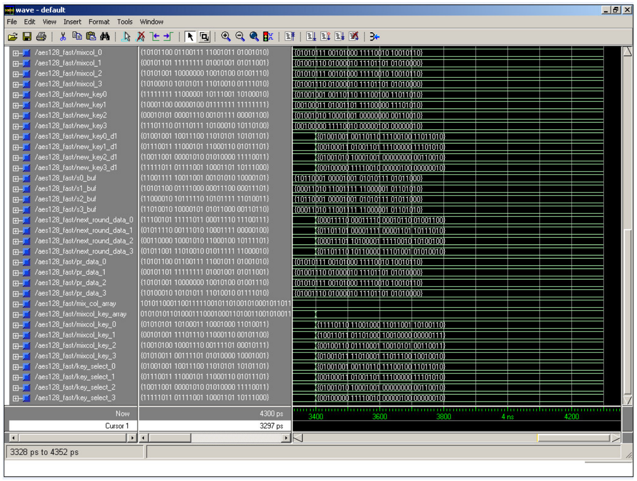
IV. RESULTS

the algorithm has been simulated and the results are shown in figures 5 and 6. The results are showing the reduction of area required when it is implemented on FPGA and having high data rate. The practical results are in accordance to theoretical predictions and satisfy the encryption and decryption methodology. The results are for 192 Kbps and 256 Kbps whereas the earlier implementations are for 128 Kbps.



```
wave - default
File Edit View Insert Format Tools Window

/aes128_fast/clk              0
/aes128_fast/reset            0
/aes128_fast/start            0
/aes128_fast/mode             1
/aes128_fast/load             0
/aes128_fast/key              01101001110001001110000011011000110101001
/aes128_fast/data_in          11011000110011011011011110000000011100001
/aes128_fast/data_out         11001111100110010010101010000101101011000
/aes128_fast/done             1
/aes128_fast/data_in_reg0     {11011000 11001101 10110111 10000000}
/aes128_fast/data_in_reg1     {01110000 10110100 11000101 01011010}
/aes128_fast/data_in_reg2     {11011000 11001101 10110111 10000000}
/aes128_fast/data_in_reg3     {01110000 10110100 11000101 01011010}
/aes128_fast/key_reg0         {01101001 11000100 11100000 11011000}
/aes128_fast/key_reg1         {01101010 01111011 00000100 00110000}
/aes128_fast/key_reg2         {01101001 11000100 11100000 11011000}
/aes128_fast/key_reg3         {01101010 01111011 00000100 00110000}
/aes128_fast/s0               {10110001 00001001 01010111 01011000}
/aes128_fast/s1               {00011010 11001111 11000001 01101010}
/aes128_fast/s2               {10110001 00001001 01010111 01011000}
/aes128_fast/s3               {00011010 11001111 11000001 01101010}
/aes128_fast/s_00             {10001010 11101110 11100101 10010111}
/aes128_fast/s_01             {10010001 10111100 10011100 10100100}
/aes128_fast/s_02             {00100101 10101110 01111001 01100110}
/aes128_fast/s_03             {10110101 10010111 01101010 00000101}
/aes128_fast/r_00             {10001010 10111100 01111001 00000101}
/aes128_fast/r_01             {10010001 10101110 01101010 10010111}
/aes128_fast/r_02             {00100101 10010111 11100101 10100100}
/aes128_fast/r_03             {10110101 11101110 10011100 01100110}
/aes128_fast/load_d1          0
/aes128_fast/start_d1         0
/aes128_fast/start_d2         0
/aes128_fast/round_cnt        0
/aes128_fast/flag_cnt         0

Now                           4300 ps
Cursor 1                      3297 ps

sim:/aes128_fast/s_00 @ 3393 ps
11001000 00000001 01011011 01101010

3328 ps to 4352 ps
```

Fig 5: implementation with 192 Kbps



```
wave - default
File Edit View Insert Format Tools Window

/aes128_fast/mixcol_0              {10101100 01100111 11001011 01001010}
/aes128_fast/mixcol_1              {00101101 11111111 01001001 01011001}
/aes128_fast/mixcol_2              {10101001 10000000 10010100 01001110}
/aes128_fast/mixcol_3              {10100010 10101011 11010010 01111010}
/aes128_fast/new_key0              {11111111 11000001 10111001 10100010}
/aes128_fast/new_key1              {10001100 00000100 01111111 11111111}
/aes128_fast/new_key2              {00010101 00001110 00101111 00001100}
/aes128_fast/new_key3              {11101110 01110111 10100010 10110100}
/aes128_fast/new_key0_d1           {01001001 10011100 11010101 10101101}
/aes128_fast/new_key1_d1           {01110011 11000101 11000110 01011101}
/aes128_fast/new_key2_d1           {10011001 00001010 01010000 11110011}
/aes128_fast/new_key3_d1           {11111011 01111001 10001101 10111000}
/aes128_fast/s0_buf                {11001111 10011001 00101010 10000101}
/aes128_fast/s1_buf                {10101100 01111000 00011100 00011101}
/aes128_fast/s2_buf                {11000010 10111110 10101111 11010011}
/aes128_fast/s3_buf                {11010010 10000101 01011000 00110110}
/aes128_fast/next_round_data_0     {11100101 11111011 00011110 11100111}
/aes128_fast/next_round_data_1     {01011110 00111010 10001111 00000100}
/aes128_fast/next_round_data_2     {00110000 10001010 11000100 10111101}
/aes128_fast/next_round_data_3     {01011001 11010010 01011111 11000010}
/aes128_fast/pr_data_0             {10101100 01100111 11001011 01001010}
/aes128_fast/pr_data_1             {00101101 11111111 01001001 01011001}
/aes128_fast/pr_data_2             {10101001 10000000 10010100 01001110}
/aes128_fast/pr_data_3             {10100010 10101011 11010010 01111010}
/aes128_fast/mix_col_array         10101100011001111100101101001010001011011
/aes128_fast/mixcol_key_array      01010101101000111000100011010011001010011
/aes128_fast/mixcol_key_0          {01010101 10100011 10001000 11010011}
/aes128_fast/mixcol_key_1          {00101001 11101110 11000110 00101100}
/aes128_fast/mixcol_key_2          {10010100 10001110 00111101 00010111}
/aes128_fast/mixcol_key_3          {01010011 00111101 01010000 10001001}
/aes128_fast/key_select_0          {01001001 10011100 11010101 10101101}
/aes128_fast/key_select_1          {01110011 11000101 11000110 01011101}
/aes128_fast/key_select_2          {10011001 00001010 01010000 11110011}
/aes128_fast/key_select_3          {11111011 01111001 10001101 10111000}

Now                                4300 ps
Cursor 1                           3297 ps

3328 ps to 4352 ps
```

Fig 6: implementation with 256 Kbps

IV. CONCLUSION

The hardware implementation of AES is started with the deign and the RTL development of the AES. The modelsim is used for simulation and it is targeted onto the FPGA Xilinx Spartan 3, along with the interfacing kit such as key board and LCD display for input and output display. Coding is done using VHDL as per design 70% of the resources of the FPGA are required and total memory usage is 75404 KB.

V. REFERENCES

[1] J. Daemean and R.Rijmen, "AES Proposal: Rijndeal", version 2, 1999. Available at http: //www.esat.kuleuveb.ac.be/ rijmen / rijndeal.

[2] Advanced Encryption Standrad(AES) »,federal information processing standrad s 197, November 26,2001.

[3] Chitu c, Chien D, Chang F., " A hardware implementation in FPGA of the Rijndael Algorithm ", cirduit and system, vol 1 pp 507- 10 ,2002,

[4] Elbeit A, Yip W ; " Single- chip FPGA implementation of a pipelined memory – based AES rijndeal Encryption Design"IEEE Transactions on very larg scale intregration systems aug 2001,pp 545-557

[5] Hsu , Tsai,Lui, and Lin ," VHDL MOeling for digital design synthsis '', Kluwer Academic press 1995

[6] Sudhkar Yalamanchilli, "introductory VHDl from simulation to synthesis", Xilinx Design series , orentice hall ,2002

[7] Advanced Encryption Standard (RIJNDEAL)http://www.esat.kuleuven.ac.be/rilmen/rijndael/

[8] Xilinx Spartan3 FPGA Family : complete Data sheet , www.xilinx.org