International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-6 Issue-7, December 2016

Highly Secure and Fast AES Algorithm Implementation on FPGA with 256 bit key size

Amrik Singh, Yoginder Talwar, Ajay Prasad

Abstract: ***The Block cipher AES is a symmetric key cryptographic standard used for transferring block of data in secure manner for server based communication networks, SCADA systems for Oil refinery, Oil and Gas Pipe Lines, and Smart Grids based applications. High level security of data transfer needs long key size i.e. 256 bits, analysis of certain ideas of round key expansion mechanisms from given key data are discussed and the same is implemented in FPGA configuration with 128 bits and 256 bits key size to achieve low latency, high throughput with high security.***

Index Terms: Advance Encryption Standard, encryption, decryption, FPGA, VHDL, Virtex-5.

I. INTRODUCTION

In AES encryption, the input plain text and output cipher text with a block size of 128 bits and can be viewed as a 4x4 matrix of 16 bytes arranged in a column major format. It can use a key size of 128, 192, or 256 bits and correspondingly has 10, 12 or 14 iterations of round transformations respectively. Each round transformation has four sub transformations namely; Byte Substitution (BS), Row Shift (RS), Mix Column (MC), and Add Round Key (AK). In the last round Mix Column (MC) transformation is not included. The round keys are derived from the user defined cipher key as per the key schedule involving two components a) Key Expansion mechanism and b) Round key selection. The total number of expanded key bytes required for a complete cipher run is equal to the no. of block length bytes (N_b) multiplied by the number of rounds (N_r) plus one. i. e. $N_b\ (N_r+1)$. Thus the total number of expanded key bytes for key size of 128, 192, and 256 bits is going to be 176, 192, and 240 bytes respectively. The increasing of a given secure key to 256 bit size results in increasing the total no. of possible codes from 2^{128} to 2^{256} and in turn good secured codes also increases accordingly. The brute force code breaking time will also get increased. The key expansion mechanism for 256 bits key size is considered to be the more secure for data block size of 128 bits whose implementation using FPGA will be discussed in this paper.

Manuscript published on 30 December 2016.
*Correspondence Author(s)
Amrik Singh, Department of Electrical Communication Engineering & Electronics and Electrical Engineering, Guru Tegh Bahadur Institute of Technology, GGSIP University, New Delhi, India.
Dr. Yoginder Talwar, Scientist, National Informatics Centre, New Delhi, India.
Dr. Ajay Prasad, Professor, Department of Information Technology, University of Petroleum and Energy Studies, Dehradun (Uttarakhand)-248007, India.

Highly secured AES algorithm implementation in FPGA data system is needed to protect data transmission between SCADA Control Server and Corporate Server of our critical integrated Corporate Industries of Petroleum, Electric Power Grids, Information Centre, Sever water control Infrastructures from cyber-attacks of national enemies, terrorist and disgruntled employees.

FPGA implementation scheme for AES algorithm has been chosen because of its low system development cost and development time, in turn has short marketing time for a product, in comparison to ASIC system designs. The product can be updated for improved performance by reprogramming its software since FPGA has the flexibility in redesign variations in FPGA. An FPGA implementation is an intermediate method between general purpose processors (GPPs) and application specific integrated circuits (ASICs), which is better than both GPPS and ASICs. FPGA scheme has wider applications than ASICs because its configuring software has broad range of functionality supported by reconfigurable nature of FPGAs. This scheme is also faster hardware solution than a GPP [7, 9, 11, and 13].

This paper deals with an FPGA implementation of AES encryption/decryption with data block size of 128 bits and key size of 256 bits, simulation and synthesis report results are compared with the other implementations as listed under [5, 6, 9, 10, 11, 12, and 13]. Our design uses key expansion module to generate round keys calculated as per theoretical calculations given in section 2 for key size of 256 bits, which matches exactly with that the key expansion of 256 bits cipher given in NIST documents. Our design approach uses lookup table approach implementation for S-box to achieve low latency as well as high throughput and is low complexity architecture.

II. MODIFIED KEY EXPANSION OF 128 BIT KEY OF AES IN TERMS OF BYTES

The key expansion of 128-bit key size in AES is defined in the following manner.

The expanded key of $N_b*(N_r+1)$ =44 words is derived from the 4 words of the user defined key.

The first four (=4) words, W [0], …, W(3) of the expanded key are filled with the use defined original cipher key bits.

The subsequent key words for all $N_k \leq i < (N_b*(N_r+1))$i.e. $4 \leq i < 44$ alternatively i = (4,…, 43) are given by:

$$W[i] = \begin{cases} W[i-N_k] \oplus \text{Rotbyte}(\text{bs}(W[i-1])) \oplus \text{Rcon}(i/N_k) & \forall\ i = 0\ (N_k) \\ W[I-N_k] \oplus W[i-1] & \forall\ i \neq 0\ (N_k) \end{cases}$$

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

First 4* N_k (=16) bytes, defined as K^0_j: (k_0, k_1, k_2, …, k_{15}) of the expanded key are filled with the original 128 user defined bits in endian format. For subsequent rounds, the expanded key bytes at
n = {16, …, 175} are given by the following relations:

a. When n =0(mod 4* N_k), the four consecutive key bytes at n to n+3 locations are obtained through:

$K_n = k_{n-16} \oplus bs(k_{n-3}) \oplus Rc(n/16)$

$K_{n+1} = K_{(n+1)-16} \oplus bs((k_{n-2}))$

$K_{n+2} = K_{(n+2)-16} \oplus bs(k_{n-1})$

$K_{n+3} = K_{(n+3)-16} \oplus bs(k_{n-4})$

b. The subsequent expanded key bytes for a particular round i.e. from (n+4)th byte to (n+15)th byte of kn, are obtained through: $k_n = k_{n-16} \oplus k_{n-4}$

Alternatively, these expanded key bytes can be obtained in the form of round keys K^i_j through the following relations with the original key bytes filled at i = 0 & j=0,…, 15 in K^0_j.
For 0≤ i < 10

$K^{i+1}_0 = K^i_0 \oplus bs(K^i_{13}) \oplus Rc(i+1)$

$K^{i+1}_1 = K^i_1 \oplus bs(K^i_{14})$

$K^{i+1}_2 = K^i_2 \oplus bs(K^i_{15})$

$K^{i+1}_3 = K^i_3 \oplus bs(K^i_{12})$

$K^{i+1}_4 = K^i_4 \oplus bs(K^i_{13}) \oplus Rc(i+1) \oplus K^i_0$

$K^{i+1}_5 = K^i_5 \oplus bs(K^i_{14}) \oplus K^i_1$

$K^{i+1}_6 = K^i_6 \oplus bs(K^i_{15}) \oplus K^i_2$

$K^{i+1}_7 = K^i_7 \oplus bs(K^i_{12}) \oplus K^i_3$

$K^{i+1}_8 = K^i_8 \oplus bs(K^i_{13}) \oplus Rc(i+1) \oplus K^i_0 \oplus K^i_4$

$K^{i+1}_9 = K^i_9 \oplus bs(K^i_{14}) \oplus K^i_1 \oplus K^i_5$

$K^{i+1}_{10} = K^i_{10} \oplus bs(K^i_{15}) \oplus K^i_2 \oplus K^i_6$

$K^{i+1}_{11} = K^i_{11} \oplus bs(K^i_{12}) \oplus K^i_3 \oplus K^i_7$

$K^{i+1}_{12} = K^i_{12} \oplus bs(K^i_{13}) \oplus Rc(i+1) \oplus K^i_0 \oplus K^i_4 \oplus K^i_8$

$K^{i+1}_{13} = K^i_{13} \oplus bs(K^i_{14}) \oplus K^i_1 \oplus K^i_5 \oplus K^i_9$

$K^{i+1}_{14} = K^i_{14} \oplus bs(K^i_{15}) \oplus K^i_2 \oplus K^i_6 \oplus K^i_{10}$

$K^{i+1}_{15} = K^i_{15} \oplus bs(K^i_{12}) \oplus K^i_3 \oplus K^i_7 \oplus K^i_{11}$

A. Notations and Notions for 256 keys

We use the data block size of 128 bits and key size of 256 bits here, use 14 rounds of iterations of round transformations.
Let for all round index i= 0,…, 14 and data byte index j=0,…, 14; key byte index l= 0,…,31;

X^i_j : j th text byte of i th round (in particular, X^0_j is the initial input plain text byte and is fixed).

X^{15}_j : j th cipher text byte.

K^i_l : l th expanded key byte of i-th round (in particular K^0_l is the user defined key : k^0_l : (k_0, k_1, k_2, …, k_{31}))
W [i] = i-th key word of 32 bits.

K_n : nth key byte, n= {0, 1, 2, …, 239}

N_k = (key size) /32 =256/32=8.

N_b = (block size) /32=128/32=4.

N_r = No. of cipher rounds =14.

B. Modified Key Expansion of 256 bits key:

The key expansion of 256-bit key size in AES is defined in the following manner.

The expanded key of $N_b*(N_r+1)$ =60 words is derived from the 8 words of the user defined key.
The first 8 words, W[0], …, W[7] of the expanded key are filled with the user defined original cipher key bits stored in big endian format. The subsequent key words for all $N_k \leq i < (N_b*(N_r+1))$i.e. 8≤ i <60 alternatively i = (8,…, 59) are given by:

$$W[i] = \begin{cases} W[i-N_k] \oplus Rotbyte\left(bs\left(W[i-1]\right)\right) \oplus Rcon(i/N_k) & \forall i = 0(N_k) \\ W[i-N_k] \oplus bs(W[i-1]) & \forall i = 4(N_k) \\ W[i-N_k] \oplus W[i-1] & \forall i \neq 0,4(N_k) \end{cases}$$

First 4* N_k (=32) bytes, defined as K^0_j: (k_0, k_1, k_2, …, k_{31}) of the expanded key are filled with the original 256 user defined bits in big endian format. For subsequent rounds, the expanded key bytes at
n = {32, …, 239} are given by the following relations:

a. When n =0(mod 4* N_k), or in particular at n= 32,64,96,128,160,192,224, the four consecutive key bytes at n to n+3 locations are obtained through:

$K_n = k_{n-32} \oplus bs(k_{n-3}) \oplus Rc(n/32)$
$K_{n+1} = K_{(n+1)-32} \oplus bs((k_{n-2}))$
$K_{n+2} = K_{(n+2)-32} \oplus bs(k_{n-1})$
$K_{n+3} = K_{(n+3)-32} \oplus bs(k_{n-4})$

b. When n= 4(mod 32), (or in particular n= 48, 80, 112, 144, 176, 208) the four consecutive key bytes in n to (n+3) locations are obtained through:

$K_n = k_{n-32} \oplus bs[k_{n-4}]$
$K_{n+1} = k_{(n+1)-32} \oplus bs[k_{n-3}]$
$K_{n+2} = k_{(n+2)-32} \oplus bs[k_{n-2}]$
$K_{n+3} = k_{(n+3)-32} \oplus bs[k_{n-1}]$

c. The subsequent expanded key bytes for a particular round i.e. from (n+4) th byte to (n+31)th byte of k_n, (or rest of n=33 to 239) are obtained through:

$K_n = k_{n-32} \oplus k_{n-4}$

These expanded key bytes can be represented in the form of round keys K^I_j with round index i and byte
Index j , through the following relations with original key bytes filled at i = 0 & j = 0, …, 31 in K^0_j.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

The expanded key bytes for the subsequent rounds i.e. $0 \leq I < 8$ are obtained through the following relations:

$K^{i+1}_0 = K^i_0 \oplus bs(K^i_{29}) \oplus Rc\ (i+1)$

$K^{i+1}_1 = K^i_1 \oplus bs(K^i_{30})$

$K^{i+1}_2 = K^i_2 \oplus bs(K^i_{31})$

$K^{i+1}_3 = K^i_3 \oplus bs(K^i_{28})$

$K^{i+1}_4 = K^i_4 \oplus bs(K^i_{29}) \oplus Rc\ (i+1) \oplus K^i_0$

$K^{i+1}_5 = K^i_5 \oplus bs(K^i_{30}) \oplus K^i_1$

$K^{i+1}_6 = K^i_6 \oplus bs(K^i_{31}) \oplus K^i_2$

$K^{i+1}_7 = K^i_7 \oplus bs(K^i_{28}) \oplus K^I_3$

$K^{i+1}_8 = K^i_8 \oplus bs(K^i_{29}) \oplus Rc\ (i+1) \oplus K^I_4 \oplus K^I_0$

$K^{i+1}_9 = K^i_9 \oplus bs(K^i_{30}) \oplus K^I_5 \oplus K^i_1$

$K^{i+1}_{10} = K^i_{10} \oplus bs(K^i_{31}) \oplus K^I_6 \oplus K^I_2$

$K^{i+1}_{11} = K^i_{11} \oplus bs(K^i_{28}) \oplus K^I_7 \oplus K^I_3$

$K^{i+1}_{12} = K^i_{12} \oplus bs(K^i_{29}) \oplus Rc\ (i+1) \oplus K^I_8 \oplus K^I_4 \oplus K^I_0$

$K^{i+1}_{13} = K^i_{13} \oplus bs(K^i_{30}) \oplus K^I_9 \oplus K^I_5 \oplus K^i_1$

$K^{i+1}_{14} = K^i_{14} \oplus bs(K^i_{31}) \oplus K^I_{10} \oplus K^I_6 \oplus K^I_2$

$K^{i+1}_{15} = K^i_{15} \oplus bs(K^i_{28}) \oplus K^I_{11} \oplus K^I_7 \oplus K^I_3$

$K^{i+1}_{16} = K^i_{16} \oplus bs\ \{(K^I_{12} \oplus K^I_8 \oplus K^I_4 \oplus K^I_0 \oplus bs(K^I_{29}) \oplus Rc\ (i+1)\}$

$K^{i+1}_{17} = K^i_{17} \oplus bs\ \{K^I_{13} \oplus K^I_9 \oplus K^I_5 \oplus K^i_1 \oplus bs\ (K^i_{30})\}$

$K^{i+1}_{18} = K^i_{18} \oplus bs\ \{K^I_{14} \oplus K^I_{10} \oplus K^I_6 \oplus K^I_2 bs\ (K^i_{31})\}$

$K^{i+1}_{19} = K^i_{19} \oplus bs\ \{K^I_{15} \oplus K^I_{11} \oplus K^I_7 \oplus K^I_3 \oplus bs\ (K^i_{28})\}$

$K^{i+1}_{20} = K^i_{20} \oplus K^{i+1}_{16}$

$K^{i+1}_{21} = K^i_{21} \oplus K^{1+1}_{17}$

$K^{i+1}_{22} = K^i_{22} \oplus K^{i+1}_{18}$

$K^{i+1}_{23} = K^i_{23} \oplus K^{i+1}_{19}$

$K^{i+1}_{24} = K^i_{24} \oplus K^{i+1}_{20}$

$K^{i+1}_{25} = K^i_{25} \oplus K^{i+1}_{21}$

$K^{i+1}_{26} = K^i_{26} \oplus K^{i+1}_{22}$

$K^{i+1}_{27} = K^i_{27} \oplus K^{i+1}_{23}$

$K^{i+1}_{28} = K^i_{28} \oplus K^{i+1}_{24}$

$K^{i+1}_{29} = K^i_{29} \oplus K^{i+1}_{25}$

$K^{i+1}_{30} = K^i_{30} \oplus K^{i+1}_{26}$

$K^{i+1}_{31} = K^i_{31} \oplus K^{i+1}_{27}$

C. *Expanded Round keys for 256 bits key:*

Upon substituting the values in the expanded individual keys, it is observed that each round has a set of 32 bytes of the expanded key depending on the original 32 key bytes in the following pattern.

K_0 to K_{31} are filled with the user defined key values. Subsequent key values are obtained using the following relation.

$K_{32} = k_0 \oplus bs(k_{29}) \oplus Rc_1$
$K_{33} = k_1 \oplus bs(k_{30})$

$K_{34} = k_2 \oplus bs(k_{31})$

$K_{35} = k_3 \oplus bs(k_{28})$

$K_{36} = k_4 \oplus k_{32}$

$K_{37} = k_5 \oplus k_{33}$

$K_{38} = k_6 \oplus k_{34}$

$K_{39} = k_7 \oplus k_{35}$

$K_{40} = k_8 \oplus k_{36}$
…
…
…
$K_{47} = k_{15} \oplus k_{43}$

$K_{48} = k_{16} \oplus k_{44}$

$K_{49} = k_{17} \oplus k_{45}$

$K_{50} = k_{18} \oplus k_{46}$
$K_{51} = k_{19} \oplus k^{47}$
$K_{52} = k_{20} \oplus k_{48}$

$K_{53} = k_{21} \oplus k_{49}$
…
…
…
$K_{63} = k_{31} \oplus k_{59}$
…
…
…
$K_{239} = k_{207} \oplus k_{235}$

These 32 byte oriented expanded round key of 256 bit may be calculated, stored for immediate use for operations in Mobile hand held systems rather than using look up tables, which will reduce memory requirements, for processing data in low end Spartan FPGA chips unchecked).

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The authors of the accepted manuscripts will be given a copyright form and the form should accompany your final submission.

III. FPGA IMPLEMENTATION OF AES WITH 128 BITS SECURITY KEY

Plain text data of 128 bits is encrypted using 128 bits round key in 10 rounds as shown in Fig.1 on left side and cipher text data is decrypted using the same set of round key but using in reverse order for decryption. For data encryption operation, in round one to round nine we perform BS, SR, MC, and AK transformation during each round and in round ten MC transformations is not included. For data decryption operation, the reverse order of rounds is followed. We perform inverse SR, inverse BS immediately after initial AK transformation using round key 10. During remaining 9 decryption rounds the same order of inverse transformations is used, but including inverse MC transformation in the beginning of the every round with round key number in reducing order. After last of AK transformation we get original plain text output data.

The input secret key of 128 bits is expanded into key for ten rounds of 128 bits each. The 128 bits secret key expansion operation is shown in Fig.2. Round key0 is used for first AK operation with plain text data during start of encryption. Round key1 is used for AK operation during round1 of encryption. Round key2 to round key10 are generated for AK operations, for rounds 2 to 10 as shown in the figure. Round keys generated during encryption are stored and utilized for AK operations of decryption also but are used in reverse direction.

When start pulse is given to the controller module, clock pulse, reset pulse, enable pulse and en/de pulse are generated by controller module. Controller module sends first reset and clock pulses to key generation module and encryption / decryption module, then send 0/1 signal to encryption/ decryption module for encryption or decryption operation depending signal level is 0 or 1 respectively. The input security key of 128 bits and input plain text / cipher text of 128 bits data are entered in key generation module and encryption / decryption module, respectively, on getting enable pulse from controller module as shown in Fig. 3. The encrypted/decrypted data of 128 bits is outputted at output port, and done pulse is generated by encryption/decryption module.

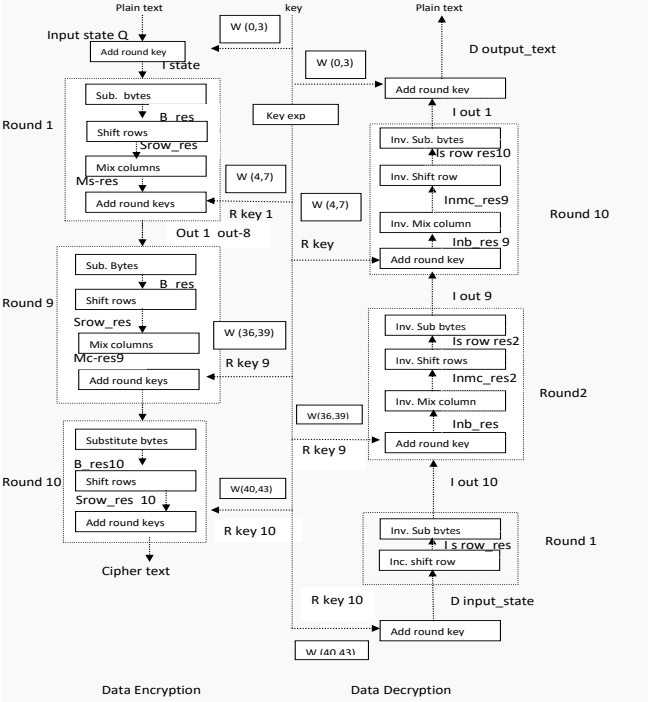


Fig.1. Data Encryption and Decryption -128 Bits

A. FPGA Implementation of AES with 256 bits security key:

Data transmission security level has been enhanced by using a secure key of 256 bit in place of 128 bit size and accordingly 240 bytes round expanded keys will be generated for fourteen rounds in place of 176 bytes for ten rounds respectively. The block diagram scheme for generation of round keys has been modified as shown in Fig. 4 in place of Fig. 2. Plain text data of 128 bits is encrypted in 14 rounds as shown in Fig.3 on

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

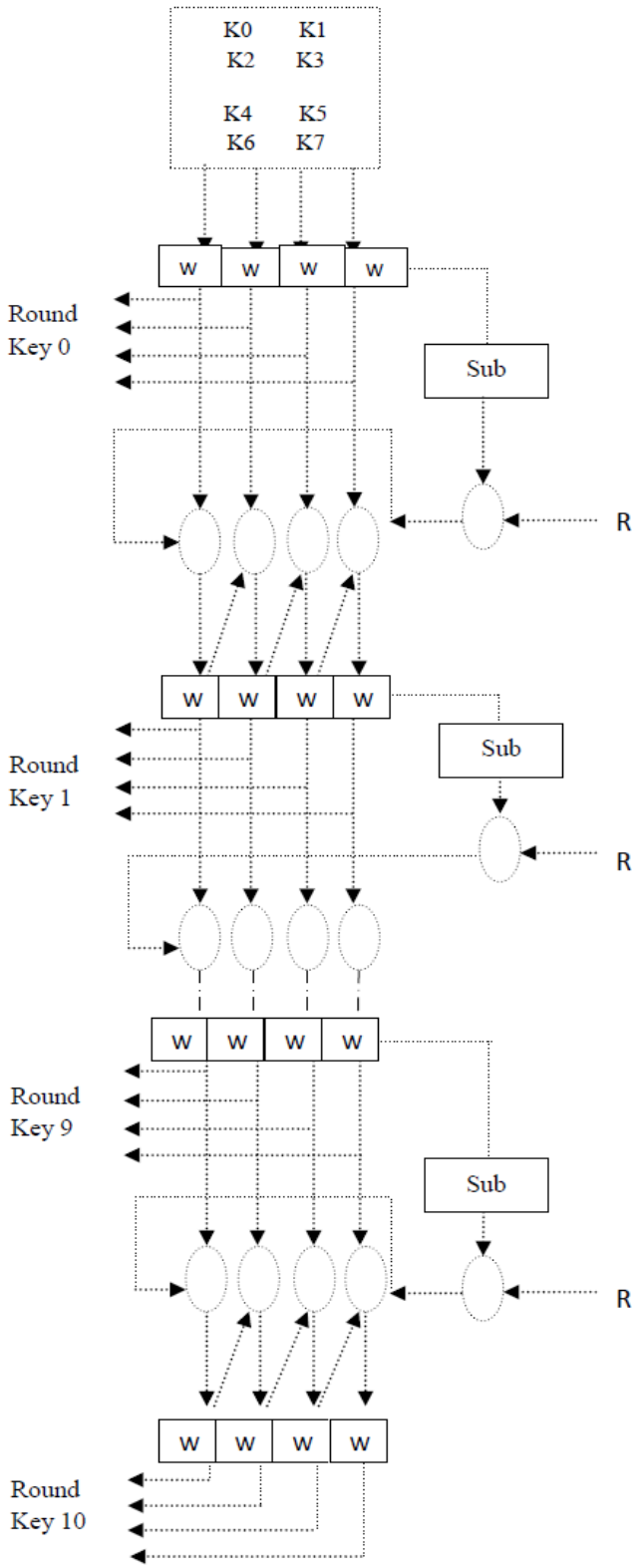


Fig. 2. 128 bits Security key expansion operation

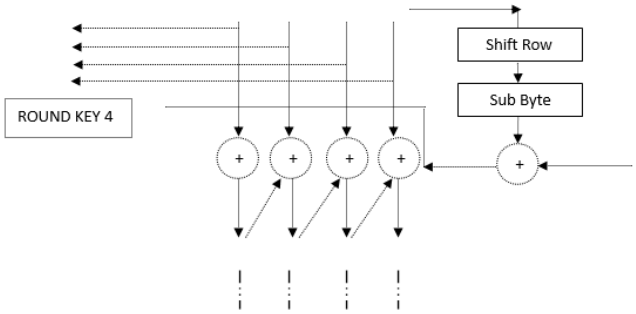
left side and cipher text data is decrypted using the same set of round key but using in reverse order for decryption. For data encryption operation, in round one to round thirteen we perform BS, SR, MC, and AK transformation during each round and in round fourteen MC transformations is not included. For data decryption operation, the reverse order of rounds is followed. We perform inverse SR, inverse BS immediately after initial AK transformation using round key 14. During remaining 13 decryption rounds the same order of inverse transformations is used, but including inverse MC transformation in the beginning of the every round with round key number in reducing order. After last of AK transformation we get original plain text output data.

The input secret key of 256 bits is expanded into key for fourteen rounds of 128 bits each. The 256 bits secret key expansion operation is shown in Fig.4. The first half of 128 bits of given 256 bits security key are termed as round key0 and the second half as round key1. Round key0 is used for first AK operation with plain text data during start of encryption. Round key1 is used for AK operation during round1 of encryption. Round key2 to round key14 are generated for AK operations, for rounds 2 to 14 as shown in the figure. Round keys generated during encryption are stored and utilized for AK operations of decryption also but are used in reverse direction.

When start pulse is given to the controller module, clock pulse, reset pulse, enable pulse and en/de pulse are generated by controller module. Controller module sends first reset and clock pulses to key generation module and encryption / decryption module, then send 0/1 signal to encryption/ decryption module for encryption or decryption operation depending signal level is 0 or 1 respectively. The input security key of 256 bits data and input plain text / cipher text of 128 bits data are entered in key generation module and encryption / decryption module, respectively, on getting enable pulse from controller module as shown in Fig. 5. The encrypted/decrypted data of 128 bits is outputted at output port, and done pulse is generated by encryption/decryption module.

IV. SIMULATION AND SYNTHESIS RESULTS OF 128 BIT KEY

The design has been coded using VHDL and all the results are synthesized based on Xilinx ISE Software 12.4 version and target device used was xc5vtx240t-2-ff1759. The results of simulation of encryption/decryption with security key of 128 bits with 128 bits input data, all 128 bits of one value are shown in Fig. 6. We find encrypted data at transmitter output as quite in random order, since AES algorithm ensures good dispersion and confusion of transmitted data. Simulation results also show that input plain text data is properly ciphered in encryption operation and when ciphered text is given as input to decryption operation, deciphered data is found to be the original input data of encryption operation. All the round keys generated during encryption operation are found to be the same as given in NIST documents for security key of 128
bits.



Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

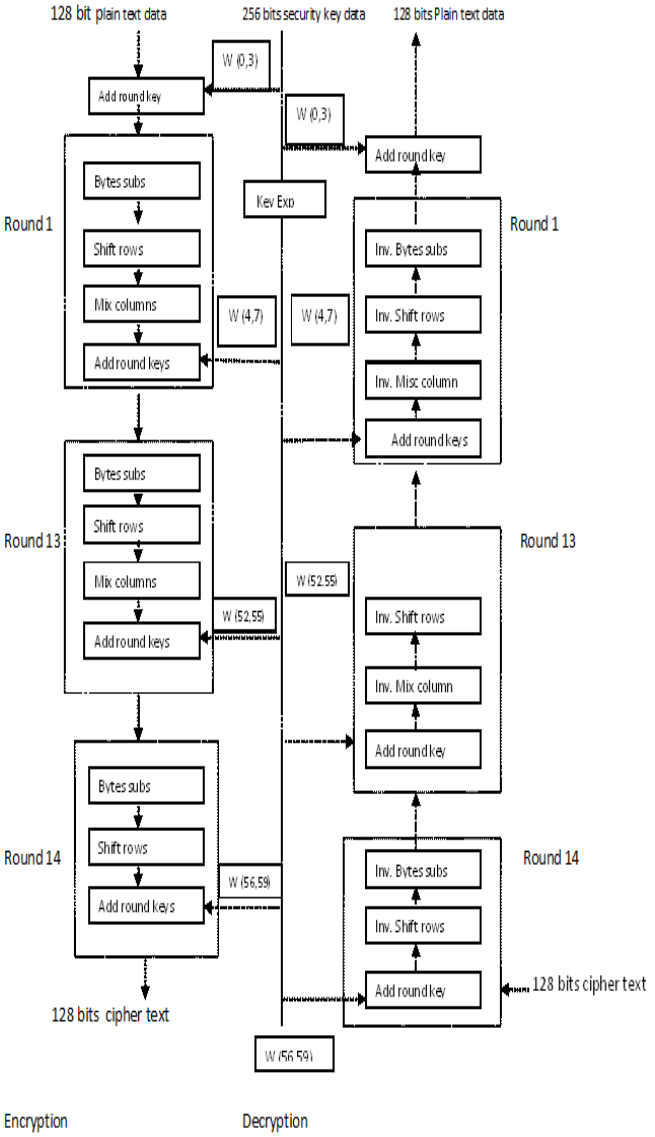


Fig.3. Data Encryption and Decryption with 256 bits security key

Synthesis reports for 128 bits security key are generated for AES algorithm based on Xilinx ISE software 12.4 versions for target device xc5vtx240-2-ff1759 are generated. Synthesis report data generated is given below.

1. No. of ROMs : 360
2. No. of Flip Flops: 10240
3. No. of input and output pins: 515
4. No. of Slice LUT's: 19974
5. Clock period: 2.115nS
6. Maximum Frequency: 472.82 MHz
7. Delay: 2.115nS
8. Throughput: 64 GBPS

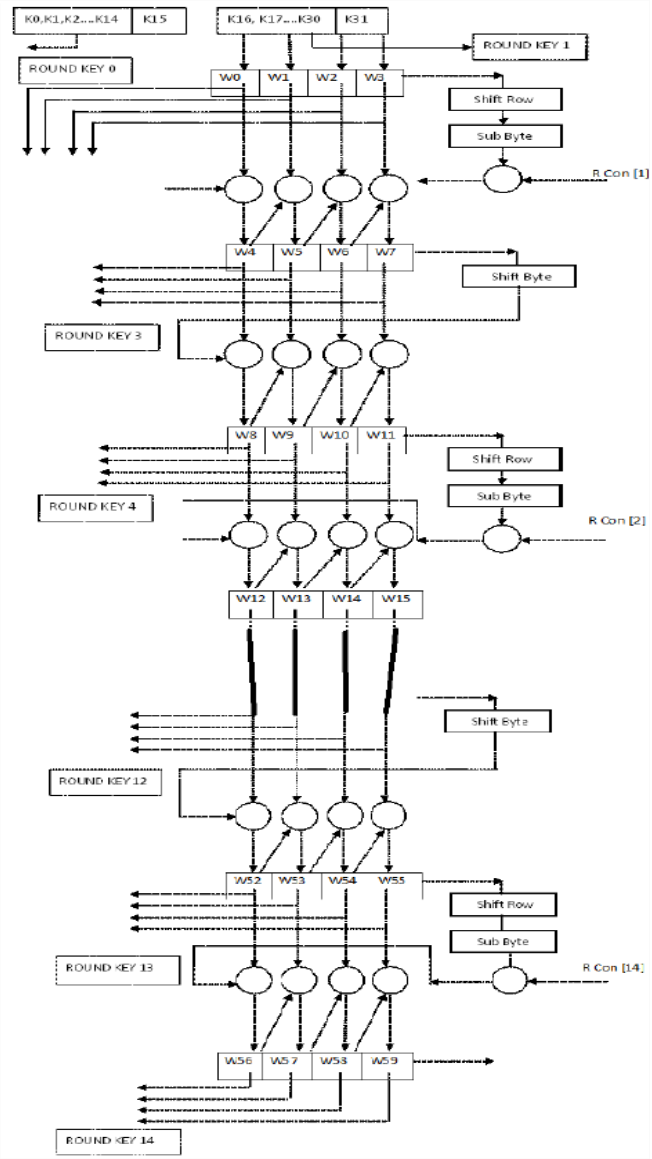


Fig. 4. 256 Bits AES Security Key Expansion Operation

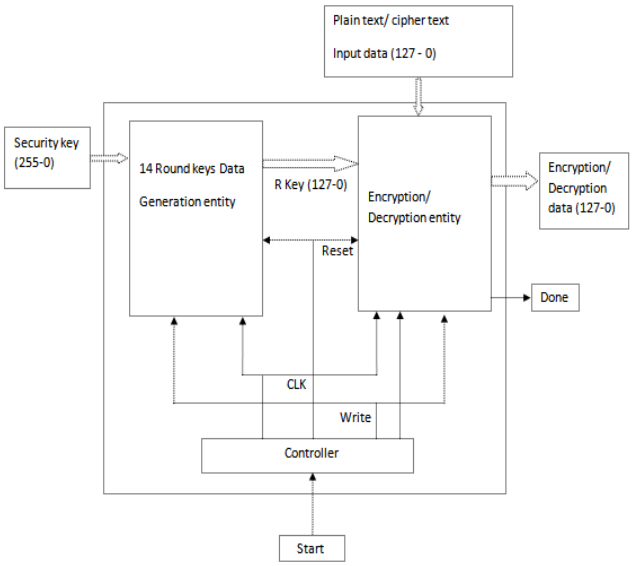


Fig. 5. Top Level Entity of Encryption and Decryption

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

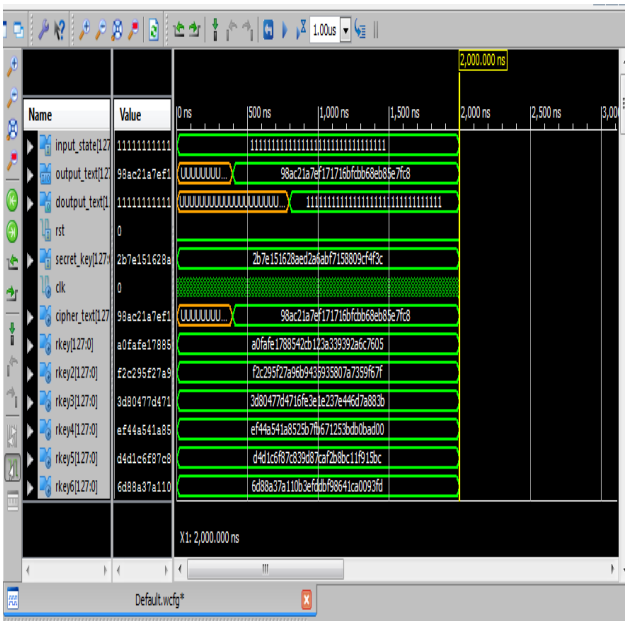


Fig. 6 Simulation results with all the 128 input data bits as "ones".

A. Simulation and Synthesis Results

The design has been coded using VHDL and all the results are synthesized based on Xilinx ISE Software 12.4 version and target device used was xc5vtx240t-2-ff1759. The results of simulation of encryption/decryption with security key of 256 bits with 128 bits input data, all zero value and all 128 bits of one value are shown in Fig. 8 and Fig. 9 respectively. Simulation results shows that input plain text data is properly ciphered in encryption operation and when ciphered text is given as input to decryption operation, deciphered data is found to be the original input data of encryption operation. All the round keys generated during encryption operation are found to be the same as given in NIST documents for security key of 256 bits [1, 2, 4, and 8].

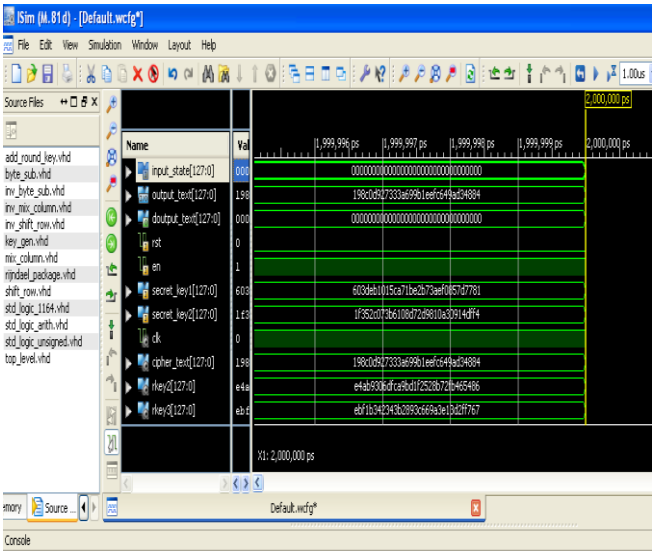


Fig. 7 Simulation results with all the 128 input data bits as ''zeros''.

Synthesis report for 256 bit security key is generated for AES algorithm based on Xilinx ISE software 12.4 versions, for target device xc5vtx240-2-ff1759, the report data is given below.

1. No. of ROMs: 500
2. No. of Flip Flops: 14336
3. No. of input and output pins: 642
4. No. of Slice LUT's: 27517
5. Clock period: 2.115nS
6. Maximum Frequency: 472.82 MHz
7. Delay: 2.115nS
8. Throughput: 64 GBPS

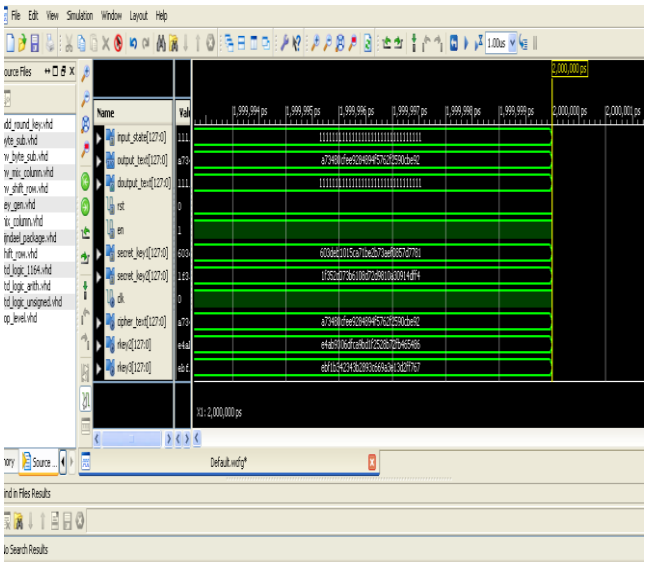


Fig. 8 Simulation results with all the 128 input data bits

Table 1: Comparison of results for FPGA implementation of AES

Design	Device used	Area/Slices used	Throughput Megabits/sec	Throughput Megabits/Slice	Maximum frequency in MHz
1. K. Gaj & P. Chodowiec [5]	XCV1000BG560-6 XC2S30-6	2902 222; GRAM-3	331.5 166	----- 0.132	----- 60
2. Dandalis [ ]	XCV-1000	5673	353.0	0.062	-----
3. Elbirt et.al [10]	XCV1000-4	10992; BRAM-0	-----	-----	31.8
4. Mcloone [12]	XCV812E-8	2000; BRAM-224	-----	-----	93.3
5. Helion	Virtex 4-11	1016	-----	-----	200.0
6. G. Rouvroy	XC3550-4	163 BRAM-3	208	1.26	71
7. Swinder Kaur [9]	Virtex2 p-7	6279; BRAM-5			119.95
8. Amandeep [13]	XC2VP30-5-FF896	1127	-----	-----	247.3
9. Thulasimani [11]	XC-2V600BF-957-6	2943	666.7	0.226	-----
10. Our Design AES-128 bits security key	XC5VTX240T-2FF 1759-2	10240; BRAM-0	**4720**	**0.460**	**472.8**
11. Our Design AES-256 bits security key	XC5VTX240T-2FF 1759-2	14336; BRAM-0	**4720**	**0.329**	**472.8**

V. COMPARISONS OF RESULTS OF AES ALGORITHM WITH 128 BIT AND 256 BIT SECURITY KEYS

Two schemes of FPGA implementations of 128 bit data block size with 128 bits security key and 256 bits security key respectively have been presented in this paper along with results reported by other authors.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

The comparative table clearly shows that our pipe lined architecture using look up tables for S-blocks are better in terms of latency, throughput and higher security with 256 bits security key.

VI. CONCLUSION

This system requires 515 input and output ports for the proposed FPGA implementation. The requirement of input and output ports is very large, which can be reduced considerably by using internal serial to parallel registers for input security key and input data respectively, and parallel to serial register for output data inside FPGA device to reduce pin count from 384 to 3 for I/O ports. A few research papers have been reported with security key of 256 bits, but need is felt for increasing the security level for AES implementation. In this paper an attempt has been made for designing highly secured AES Implementation on FPGA with long size key for data transmission between Server system and other connected corporate business computers for Petroleum Industry and other Industries. Hand held mobile secured system is also suggested for field application design, using S-Box optimized by composite field arithmetic (CFA) method for reducing multiplication inversion calculations to reduce chip area and cost and security enhanced by using masking technique of S-Boxes data.

REFERENCES

1. J. Daemen and V. Rijmen. AES proposal: Rijndael. In AES Round 1 Technical Evaluation, NIST 1998. (see: http:// www.esat.kuleven.ac.be/rijmen /rijndael/, http://www.nist.gov/aes)
2. N ferguson, R. Schroeppel, D. Whiting. A simple algebraic representation of Rijndael Selected Area in Cryptography, SAC 2001, LNCS 2259, Springer-Verlag, 2001, pp.103-111.
3. Courtois, N.T. and J. Pieprzyk: Cryptanalysis of Block Ciphers with over defined Systems of Equations. Accepted by, Asiecrypt 2002, Dec 2002. (See: http://eprint.iacr.org/2002/044).
4. Y. Talwar, C.E. Veni Madhavan, N. Rajpal, "On the key expansion Mechanisms of the AES Ciphers: Rijndael, Serpent".
5. P. Chdowiec, K. Gaj, "Very compact FPGA implementation of the AES algorithm", Cryptographic hardware and embedded systems (CHES 2003), LNCS vol. 2779, pp. 319-333, Springer-Verlog, October 2003.
6. G. Rouvroy, F.X. Standaert, J.J. Quisquater, J.D. Legat, , Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications, Proceedings of the international conference on Information Technology: coding and computing 2004 (ITCC 2004), pp. 583-587, vol. 2, April 2004.
7. Tim Good and Mohammed Benaissa, "AES on FPGA from the Fastest to the Smallest", CHES 2005, LNCS 3659, pp. 427-440, 2005. Springer-Verlog Berlin Heidelberg 2005.
8. Y. Talwar, C.E. Veni Madhavan, Navin Rajpal, "On Partial Linearization of Byte Substitution Transformation of Rijndael-The AES". Journal of Computer Science 2(1): 48-52, 2006, ISSN1549-3636 © 2006 Science Publications.
9. Swinder Kaur and Prof. Renu Vig, "Efficient Implementation of AES Algorithm in FPGA Devices". International Conference on Computational intelligence and Multimedia Applications 2007, DOI 10.1109/ICCIMA -2007.250, pages 179-187,0-7695-3050-8/07, IEEE-(2007) Volume2, pp 179-187.
10. J. Elbirt, W. Yip, B. Chatwynd and C. Paes, "An FPGA Implementation and performance Evaluation of the AES block cipher candidate algorithm analyst", Presented at Proc.3rd AES Conf. (AES). Available: http:// csrc.nist.gov/encryption/AES/round2/conf3/aes3paper.html.
11. Thulasimani L. and Madheswarn, "A Single Chip Design and Implementation of AES-128/192/256 Encryption Algorithms", International journal of Engineering Science and Technology (IJEST); ISSN: 0975-5462, Vol.2(5), 2010, 1052-1059.
12. M. McLoone and J. V. McCanny, "Rijndael FPGA implementation utilizing look-up tables" , in IEEE Workshop on Signal processing systems, Sept. 2001, pp. 349-360.
13. Amandeep Kaur, Puneet Bhardwaj and Naveen Kumar, "FPGA Implementation of Efficient Hardware for the Advanced Encryption Standard", in IJITEE; ISSN: 2278-3075, Volume-2, Issue-3, February 2013.

AUTHOR PROFILE

Amrik Singh got graduation in Electronics and Telecommunication Engineering from Institution of Engineers, Kolkata, India, Master in Engineering in Electronics and Communication Engineering branch, Delhi College of Engg., University of Delhi. Presently he is working as Ph. D Research Scholar (part time) at University of Petroleum and Energy Studies, Dehradun, India. He is working as Associate Professor in ECE Department, Guru Tegh Bahadur Institute of Technology, New Delhi.

Dr. Yoginder Talwar received his graduation in Electronics and Telecommunication Engineering from Institution of Electronics and Telecommunication Engineers, New Delhi, received his Master in Engineering from Delhi College of Engineering, University of Delhi, India in 1998, and received his Ph. D from Guru Gobind Singh Indraprastha University, Delhi, India in 2006. Presently he is working as senior Scientist in Cyber Security Department at National Informatics Centre, New Delhi.

Dr. Ajay Prasad received his Ph.D. in Computer Science and Engineering, in the area of Cloud security, M. Tech. in Computer Science & Engineering, MCA, B.Sc. (PCM), and GATE in 2006. He has more than 14 years teaching experience at various Institutions. He is reviewer in reputed journals and is Life Member of various reputed professional Organizations. Presently he is a Professor in the department of Information Technology, University of Petroleum and Energy Studies, Bidoli, Dehradun.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)