IPHV7I10003X
International Journal Of Advanced Research and Innovation -Vol.7, Issue .I
ISSN Online: 2319 – 9253
Print: 2319 – 9245

FPGA Based Implementation of AES Encryption and Decryption with Verilog HDL

Y.Aruna[1], Prof.S.N.Shelke[2]
M.Tech (Electronics), JDCOE, Nagpur.

Abstract:

Security is the most important part in data communication system, where more randomization in secret keys increases the security as well as complexity of the cryptography algorithms. As a result in recent dates these algorithms are compensating with enormous memory spaces and large execution time on hardware platform. Field programmable gate arrays (FPGAs), provide one of the major alternative in hardware platform scenario due to its reconfiguration nature, low price and marketing speed.In this a hardware implementation of the AES128 encryption and decryption algorithm is proposed.The AES cryptography algorithm can be used to encrypt/decrypt blocks of 128 bits and is capable of using cipher keys of 128 bits wide (AES128). A unique feature of the proposed pipelined design is that the round keys, which are consumed during different iterations of encryption, are generated in parallel with the encryption process.This lowers the delay associated with each round of encryption and reduces the overall encryption delay of a plaintext block. This will leads to an increase in the message encryption throughput. This method will experimentally simulate using Xilinx software with Verilog Hardware Description Language and hardware implementation on FPGA.

Keywords: *- FPGA,AES cryptography, encryption, decryption,pipelined design,throughput,Xilinx.*

I.INTRODUCTION

AES is short for Advanced Encryption Standard and is a United States encryption standard defined in Federal Information Processing Standard (FIPS) 192, published in November 2001. It was ratified as a federal standard in May 2002. AES supersedes Data Encryption Standard (DES). The DES algorithm broken because of short keys (56-bit key).AES can be implemented both on hardware and software.

AES is a symmetric encryption algorithm processing data in block of 128 bits. Under the influence of a key, a 128-bit block is encrypted by transforming it in a unique way into a new block of the same size. AES is symmetric since the same key is used for encryption and the reverse transformation, decryption. The only secret necessary to keep for security is the key. In this project new AES algorithm with encryption and decryption was realized in

Verilog Hardware Description Language.The 128-bit plaintext and 128-bit key, as well as the 128-bit output data were all divided into four 32-bit consecutive units respectively controlled by the clock. The pipelining technology was utilized in the intermediate nine round transformations so that the new algorithm achieved a balance between speed and chip area.

AES-128 may configured to use different key-lengths, the standard defines 3 lengths and the resulting algorithms are named AES-128, AES-192 and AES-256 respectively to indicate the length in bits of the key. In this project AES-128 encryption and decryption with 128-bit key is considered [1-4]. Each additional bit in the key effectively doubles the strength of the algorithm, when defined as the time necessary for an attacker to stage a brute force attack, i.e. an exhaustive search of all possible key combinations in order to find the right one. AES can resist various currently known attacks. Hardware security solution based on highly optimized programmable FPGA provides the parallel processing capabilities and can achieve the required performance benchmarks. The current Area Optimized algorithm of AES are mainly based on the realization of S-box mode and the minimizing of the internal registers which could save the area of IP core significantly [1-5].

The rest of the paper is organized as follows. Section II describes existing model, III describes proposed improved AES Encryption Algorithm with pipeline technology during round transformation, Section IV describes proposed decryption , Section v describes comparision.Finally concluded the paper in section VI.

II.EXISTING MODEL

AES is a specification for the encryption of electronic data. It has been adopted by the U.S. government and is now used worldwide. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

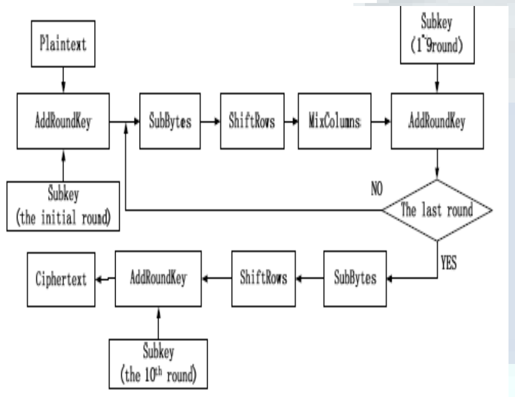
AES is based on a design principle known as a substitution-permutation network. It is fast in both software and hardware.[6] Unlike its predecessor, DES, AES does not use a Feistel network.

AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, whereas Rijndael can be specified with block and key sizes in any multiple of 32 bits, with a minimum of 128 bits. The blocksize has a maximum of 256 bits, but the keysize has no theoretical maximum.

AES operates on a 4×4 column-major order matrix of bytes, termed the state (versions of Rijndael with a larger block size have additional columns in the state). Most AES calculations are done in a special finite field.

The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

The algorithm is composed of three main parts: Cipher, Inverse Cipher and Key Expansion. Cipher converts data, commonly known as plaintext, to an unintelligible form called cipher. Key Expansion generates a key schedule that is used in the Cipher and the Inverse Cipher procedure. Cipher and Inverse Cipher are composed of specific number of rounds (Table 1). For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key length [1]. AES operates on a 4x4 array of bytes (referred to as “state”).



The algorithm consists of four different simple operations.

These operations are:

• Sub Bytes
• Shift Rows
•Mix Columns
•Add Round Key.

- Sub Bytes a non-linear substitution step where each byte is replaced with another according to a lookup table.
- Shift Rows—a transposition step where each row of the state is shifted cyclically a certain number of steps.
- Mix Columns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
- Add Round Key.

III.The Design Of Improved AES Algorithm

The proposed architecture is designed to get maximum speed and lesser area by mapping all the four Logical functions of AES to LUTs, ROMs and Block RAMs. The proposed architecture has three parts

1. Key Generation Module
2. Encryption Module
3. Decryption Module.

The AES encryption and decryption core unit contains key generation module as a common unit. This module gives necessary key expansion for both encryption and decryption functions. Fig.3 presents the block diagram of AES Rijndael encryption and decryption with Key Generation Module as a common unit. The key generation module consists of key register of 128 bits, S-Box and XOR gates for bitwise XOR operation.

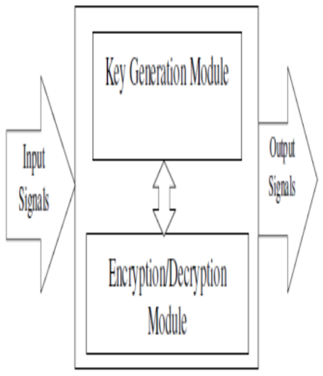


Fig 3.1: AES Encryption and Decryption Unit Block Diagram

It is designed to produce round keys on each positive edge of the clock, when it is enabled. However in the proposed work, the key generation architecture does not require any hardware for shift operation and the port mapping between key register and S-Box is done according to the required shift. Hence the proposed work offers the advantage in area. Also in the proposed work the bits are rearranged on data path from register to S-Box and the round constant required for each rounds are stored in ROM and retrieved on each clock. Fig.3.2 represents proposed architecture of key generation unit.

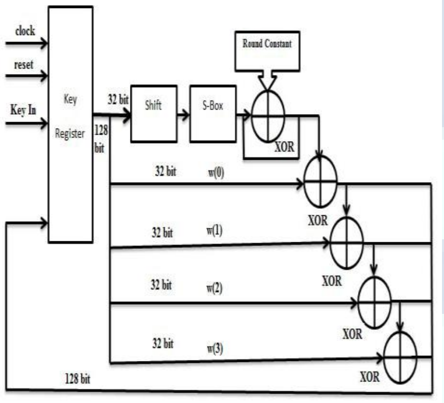


Fig. 3.2. Architecture of Key Generation Module.

3.1 Encryption:

The encryption module takes 128 bit text to be encrypted and receives round key from key generation module to do each round of encryption. Fig. 5 presents the proposed encryption module.

Start, stop_mix, terminate are control signal produced by the control unit. The "done' signal is provided to indicate that encryption is done. Architecture is as shown in Fig. 3.4.

In the proposed work for reducing the hardware of entire architecture, the control unit of encryption module is not designed separately. The control unit of key generation module which is a 4-bit counter is designed to control the entire functioning of encryption module. The sharing of control unit by both encryption and round key generation gives unique advantage of reduction in hardware as compared to other implementations.

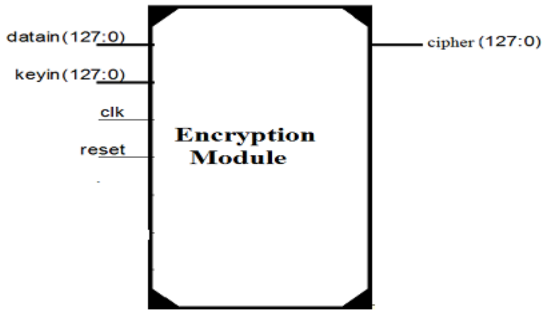


Fig 3.3 Encryption Module

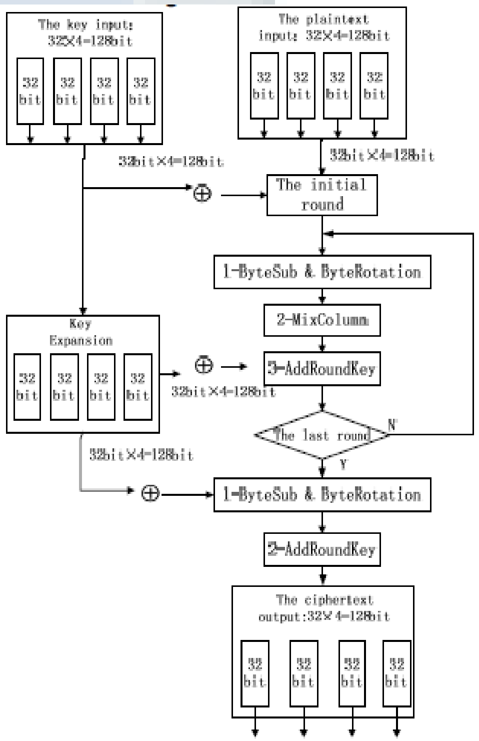


Fig 3.4: Proposed architecture of encryption module

The functions of various parts of the structure shown above are described as follow:

· The initial round of encryption:

The four packets of consecutive 32-bit plaintext (128 bits) have been put into the corresponding registers. Meanwhile, another four packets of consecutive 32-bit initial key (128 bits) have been put into other registers by the control of the enable clock signal. Furthermore, this module should combine the plaintext and initial key by using the XOR operators.

Round Transformation in the intermediate steps:

A round transformation mainly realizes the function of Sub Bytes and Mix Columns with 32-bit columns. Four packets of round transformation are processed independently. Then the results of Mix Columns and the 32-bit keys sourced from Key expansion are combined by using XOR operators. Here, the round transformation is a module with 64 input ports (32- bit plaintext+32-bit key) and 32 output ports.

The function of Sub Byte is realized by Look-Up Table (LUT). It means that the operation is completed by the Find and Replace after all replacement units are stored in a memory (256×8bit = 1024 bit).

The implementation of Mix Column is mainly based on the mathematical analysis in the Galois field GF(28). Only the multiplication module and the 32-bit XOR module of each processing unit(one column) are needed to design, because the elements of the multiplication and addition in Galois field are commutative and associative. Then the function of Mix Column can be achieved.

In the proposed work, the S-Box is implemented by a LUT having 8 bit address (256 addresses) and a data width of 8 bit. This implementation gives higher throughput for the design by significantly decreasing delay in data path. As a result the proposed design takes lesser number of slices when compared with other combinational technique proposed.

Fig 3.5 is a block diagram for the introduction of pipelining technology used in the round transformation.

In the process of pipelining, the 128-bit data is divided into four consecutive 32-bit packets that take round transformation independently. The operation of the above four groups of data can be realized in pipelining technology. In brief, it can be described as follow: store the unprocessed data in the 128-bit register, and control the clock for re-starting the 128-bit register to read the new data when the four groups' operations have been overcome. Thus the 128-bit round-operating unit has been transformed into four 32-bit round-operating elements. The internal pipelining processing should be implemented during the whole nine intermediate Round Transformations of the four packets before achieving the 128-bit ciphertext.

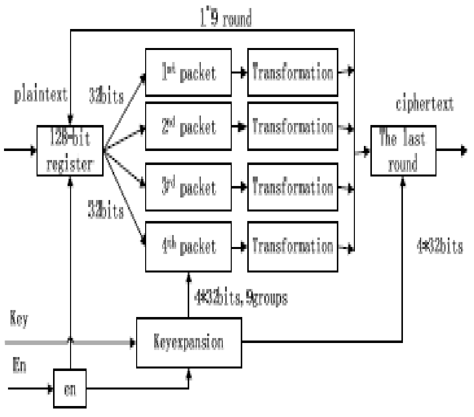


Fig3.5. The round processing with pipeline technology

· The process of the last round

The final round is a 128-bit processor. After nine rounds of operations included Shiftrows, Sub Byte and Mix columns, the 128-bit intermediate encrypted data will be used in XOR operation with the final expanded key(4*32bit), which is provided by the key expansion module. The output of final round in the processor is the desired 128-bit cipher text. Similarly, the cipher text is divided into four packets of 32-bit data by an external enable signal.

· Key expansion and Key extraction

This module is implemented basically the same with the traditional way as another part of the AES encryption algorithm. The only difference lies on the mode of data transmission. The initial key and expanded keys are divided into four 32-bit data before being extracted.

3.2 Decryption:

In the proposed work the Mix Column encryption hardware uses two of such ROM for Galois multiplication of '2' and '3' and for performing 4-Input XOR operation in Mix Column operation, the proposed design use 16 x 1 ROM with the result that Mix Column operation offers higher speed and uses minimum number of slices in the hardware (FPGA).

The decryption unit also uses same design approach for the entire architecture and decrypt the given cipher back to original text. Inverse S-Box architecture uses the same design of S-Box. Entry of LUT is changed according to Inverse Sub Byte transformation. Mix Column operation is implemented using 256X8 ROM. Four such ROMs are designed for the Galois multiplication of 9, 11, 13 and 14. 4-Input XOR operation is designed by 16x1 ROM. Architecture of Decryption module is same as encryption module with all complimentary functions of encryption. Decryption unit contains an extra register for storing Round Keys. Storing key is important since first round decryption use tenth round key and second round use ninth round key and so on.

VI. SIMULATION RESULTS

A. Results

In our test case we have taken 16 byte data by the key board

Input = [36 46 e6 a8 88 5a 30 8c 28 31 98 a2 e0 37 07 34].

Cipher Key= [2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c].

Then after ten rounds of the AES the cipher text will appear as

Cipher text= [27 37 c1 82 83 29 a4 f1 43 93 9c 5e a6 b0 7c e1]

as shown in Fig. 7. For decryption we use cipher text as input and use the same cipher key for decryption algorithm and find

original data= [36 46 e6 a8 88 5a 30 8c 28 31 98 a2 e0 37 07 34].

This decrypted data is sent to the PC through an RS232 port and been verified using the hyper terminal interface (Fig. 8).

Table II shows the resource utilization of the FPGA in our design and Table III shows the execution time of our algorithm and the throughput of encryption and decryption from the equation given below:

Throughput= (bits processed for encryption or decryption)/second

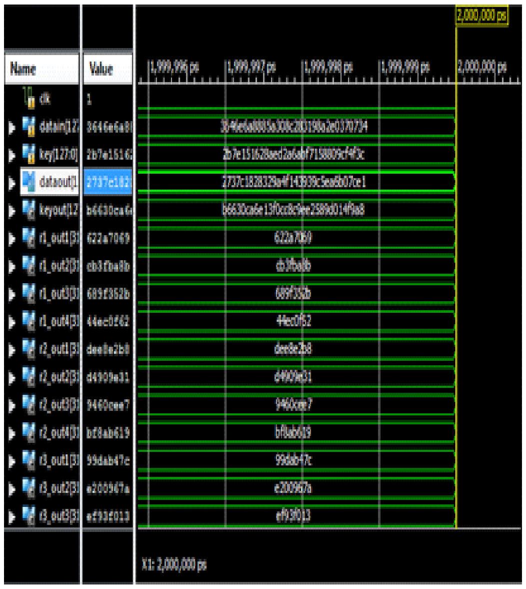
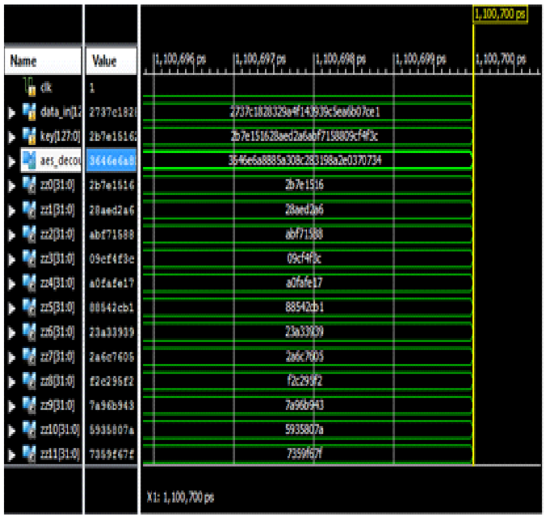


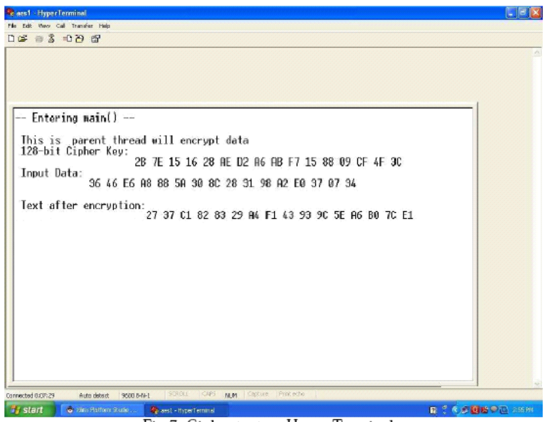
TABLE III: ENCRYPTION AND DECRYPTION TIME

Process	Clock frequency (MHz)	Time (ms)	Throughput (byte/sec)
Encryption	50	4.0274	3972.2
Decryption	50	4.1524	30825.1

TABLE IV: COMPARISION WITH EXSISTING WORKs

Design	Device	Frequency MHz	Slices	BRAMS
Elbirt et al[13]	XCV1000-4	31.8	10992	0
M. McLoone et al[14]	XCV812e-8	93.9	2000	244
K.U.Jarvinen et al[15]	XCV1000e-8	129.2	11719	0
G.P.Saggese [16]	XCV2000e-8	158	5810	0
F. Standaert [17]	XCV3200e-8	154	15112	0
Proposed	Spartan3e Xc3s500e	50	*2,495*	320



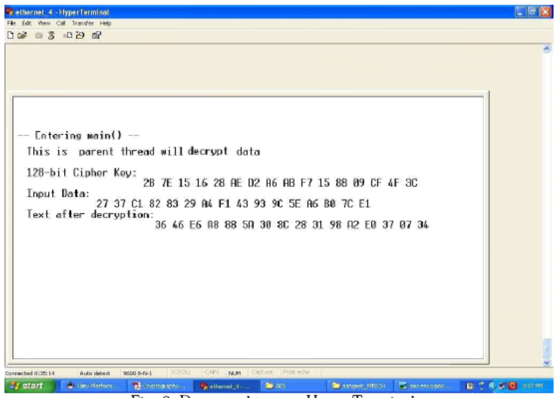


```
-- Entering main() --

 This is  parent thread will encrypt data
 128-bit Cipher Key:
                    2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
 Input Data:
            36 46 E6 A8 88 5A 30 8C 28 31 98 A2 E0 37 07 34

 Text after encryption:
                      27 37 C1 82 83 29 A4 F1 43 93 9C 5E A6 B0 7C E1
```

Fig 7: Cipher text on Hyper Terminal

TABLE II: RESOUCES UNILIZATION

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	2,621	9,312	28%
Number of 4 input LUTs	2,871	9,312	30%
Number of occupied Slices	2,495	4,656	53%
Number of Slices containing only related logic	2,495	2,495	100%
Number of Slices containing unrelated logic	0	2,495	0%



```
-- Entering main() --
 This is  parent thread will decrypt data

 128-bit Cipher Key:
                    2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
 Input Data:
            27 37 C1 82 83 29 A4 F1 43 93 9C 5E A6 B0 7C E1
 Text after decryption:
                      36 46 E6 A8 88 5A 30 8C 28 31 98 A2 E0 37 07 34
```

Fig. 8: Decrypted text on Hyper Terminal

VII. CONCLUSION

The aim of this proposed design is to perform a real time data communication exhibiting a significant level of security and providing a faster processing time where necessary. The bit length of the key used in our experiment is 128 bit and the result is obtained successfully. The key width can be varied

with a little modification in the algorithm. Though we have performed the real time encryption and decryption of data for RS232 serial communication the technique remains the same for Ethernet data communication using the EMAC core. In future we will try to perform the encryption and decryption of data where inputs will be audio, image, video data coming from different multimedia applications performed over FPGA. Usage of single FPGA with dual processor implementation, where one processor will execute the algorithm while other one will be responsible for input data acquisition, so that the executing processor can handle the algorithm without any interruption, will also be a good step in the world of hardware design.

VIII. REFERANCES

[1] Abdel-hafeez.S.,Sawalmeh.A. and Bataineh.S.,“High Performance AES Design using Pipelining Structure over GF(28)” IEEE Inter Conf.Signal Proc and Com.,vol.24-27, pp.716-719,Nov. 2007

[2] J.Yang, J.Ding, N.Li and Y.X.Guo,“FPGA-based design and implementation of reduced AES algorithm” IEEE Inter.Conf. Chal Envir Sci Com Engin(CESCE).,Vol.02, Issue.5-6, pp.67-70, Jun 2010.

[3] A.M.Deshpande, M.S.Deshpande and D.N.Kayatanavar,“FPGA Implementation of AES Encryption and Decryption”IEEE Inter.Conf.Cont,Auto,Com,and Ener., vol.01,issue04, pp.1-6,Jun.2009.

[4] Hiremath.S. and Suma.M.S.,“Advanced Encryption Standard Implemented on FPGA” IEEE Inter.Conf. Comp Elec Engin.(IECEE),vol.02,issue.28,pp.656-660,Dec.2009.

[5] AI-Wen Luo, Qing-Ming Yi, Min Shi. “Design and Implementation of Area-optimized AES on FPGA” ,IEEE Inter.conf.chal sci com engin.,978-1-61284-109-0/2011.

[6] Rizk.M.R.M. and Morsy, M., “Optimized Area and Optimized Speed Hardware Implementations of AES on FPGA”, IEEE Inter Conf. Desig Tes Wor.,vol.1,issue.16,pp.207-217, Dec. 2007.

[7] Liberatori.M.,Otero.F.,Bonadero.J.C. and Castineira.J. “AES-128 Cipher. High Speed, Low Cost FPGA Implementation”, IEEE Conf. Southern Programmable ogic(SPL),vol.04,issue.07,pp.195-198,Jun. 2007.

[8] Abdelhalim.M.B., Aslan.H.K. and Farouk.H. “A design for an FPGAbased implementation of Rijndael cipher”,ITICT. Ena Techn N Kn Soc.(ETNKS), vol.5,issue.6,pp.897-912,Dec.2005.

[9] Fedaral Information Processing Standards publication 197 November 26,2001”ADVANCED ENCRYPTION STANDARD (AES)”.

[10] “Architectures and VLSI Implementations of the AES-Proposal” Rijndael. N. Sklavos and O. Koufopavlou,.IEEE TRANSACTIONS ON COMPUTERS, VOL. 51, NO. 12, DECEMBER 2002.

[11] NIST, “Advanced Encryption Standard (AES)”,NIST,FIPS-197,2001.

[12] http : // Advanced Encryption Standard-wikipedia , the free encyclopedia.html.

[13] P.Chodowiec and K.Gaj.” Very compact FPGA implementation of the AES algorithm”. In Proc.5th Int, workshop on Cryptographic Hardware and Embedded systems (CHES 2003), pages 319-333, cologne,Germany,Sept,8-10,2003.

[14] P.Canright.” A Very Compact S-box for AES”.In Proc.7th Int, workshop on Cryptographic Hardware and Embedded systems (CHES 2005), pages 441-455, Edinberg, UK , Aug.29-Sept,2005.

[15] K..Viswanath, Dr P.v.Rao, Veeresh Patil,”Design and Implementation of Area-Optimized 256-bit Advanced Encryption Standard for real time images on FPGA”, ISSN:2319-1112/V1N2:134-140©IJAEEE.

[16] Satoh .A.Morioka.S,Takano.k. and Munetoh .S,”A Compact Rilndael Hardware Architecture with S-box Optimization”LNCs 2248, ASIA CRYPT 2001,pp.239-254.2001.256.

[17] Ahmed Rady, Ehab EL Sehely, A.M.EL Hennawy,”Design and Implementation of area optimized AES algorithm on reconfigurable FPGA”,IEEE Inter.conf.Comp Elec Engin(IECEE), 978-1-4244-1847-3/07.2007.

[18] S.Sankar Ganesh,J.Jean Jenifer Nesam,”FPGA Based SCA Resistant AES S-BOX Design”,International Journal of Scientific & Engineering Research,volume4,Issue 4,pp.1143-1149,April-2013.