ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

High Performance Hardware Implementation of AES

C.Rajendra
M.Tech Student,
Department of ECE,
ACE Engineering College,
JNTUH University, Telangana, India.

M.Ravikumar
Professor,
Department of ECE,
ACE Engineering College,
JNTUH University, Telangana, India.

ABSTRACT:

Increasing need of data protection in computer networks led to the development of several cryptographic algorithms hence sending data securely over a transmission link is critically important in many applications. Hardware implementation of cryptographic algorithms are physically secure than software implementations since outside attackers cannot modify them.

In order to achieve higher performance in today's heavily loaded communication networks, hardware implementation is a wise choice in terms of better speed and reliability. This paper presents the hardware implementation of Advanced Encryption Standard (AES) algorithm using Xilinx– virtex5 Field Programmable Gate Array (FPGA).

In order to achieve higher speed and lesser area, Sub Byte operation, Inverse Sub Byte operation, Mix Column operation and Inverse Mix Column operations are designed as Look up Tables (LUTs) and Read Only Memories (ROMs). This approach gives a throughput of 3.74Gbps utilizing only 1% of total slices in xc5vlx110t-3-ff1136 target device.

Keywords:

AES; Rijndael; Cryptography; FPGA; Verilog; Encryption; Decryption.

I.INTRODUCTION :

The main Objective of this project is to code a Data Encryption System using Advanced Encryption Standard (AES) Algorithm in Hardware Description Language and to test it according to a predetermined standard stimulus so that it meets requirements. This standard specifies the Rijndael algorithm ([1] and [2]), a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits.Rijndael was designed to handle additional block sizes and key lengths; however they are not adopted in this standard.

Throughout the remainder of this standard, the algorithm specified herein will be referred to as "the AES algorithm." The algorithm may be used with the three different key lengths indicated above, and therefore these different "flavors" may be referred to as "AES-128", "AES-192", and "AES-256".

Cryptography is the technique used to avoid unauthorized access of data. For example, data can be encrypted using a cryptographic algorithm in conjunction with the key management.

It will be transmitted in an encrypted state, and later decrypted by the intended party. If a third party intercepts the encrypted data, it will be difficult to decipher. The security of modern cryptosystems is not based on the secrecy of the algorithm, but on the secrecy of a relatively small amount of information, called a secret key. The fundamental and classical task of cryptography is to provide confidentiality by encryption methods.

II.AES ALGORITHM:

The AES is a computer security standard from NIST intended for protecting electronic data. Federal Information Processing Standards (FIPS) Publication 197 gives the specification of AES.

AES use Rijndael algorithm [3] by Joan Daeman and Vicent Rijimen for both encryption and decryption. The AES cryptography algorithm is capable of encrypting and decrypting 128 bit data using cipher keys of 128, 196 or 256bits (AES128, AES196 and AES256) [4].

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

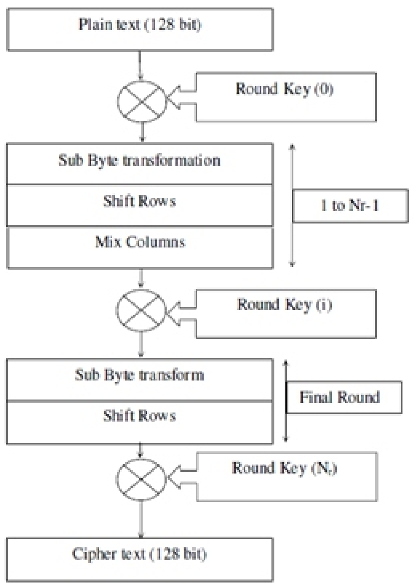

Figure- 1: Algorithm for AES Encryption

Rijndael encryption consist of four operations:

1. Substitution.

2. Shift Row.

3. Mix Column.

4. Key Addition.

The Rijandael decryption consists of four inverse operations Of encryption which are compliment functions of encryption. They are:

1. Inverse Substitution.

2. Inverse Shift Row.

3. Inverse Mix Column.

4. Key addition.

The operations of AES Rijndael algorithm for encryption and decryption is given as follows:

A. Sub Byte and Inverse Sub Byte transformation In the Sub Bytes step, each byte in the state matrix is replaced with a Sub Byte using an 8-bit data from the Rijndael S-Box. In the Inverse Sub Bytes step, each byte in the cipher matrix is replaced with corresponding Inverse Sub Byte. Sub Byte operation provides the non-linearity in the cipher. The S-Box used is derived from the multiplicative inverse over Galois Field known to have good non-linearity properties. Many S-Box implementation use combinational circuit consists of an adder, squarer and constant multiplier. Rijndael S-Box is not shown for brevity.

B. Shift Row Transformation:

The Shift Rows transformation cyclically shifts the bytes in each row by certain offset to the left. For AES, the first row is left unchanged. Each byte of the second row is shifted by one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. Inverse Shift Row transformation does the same shift operation towards right. Fig.2 shows the Shift Row operation.

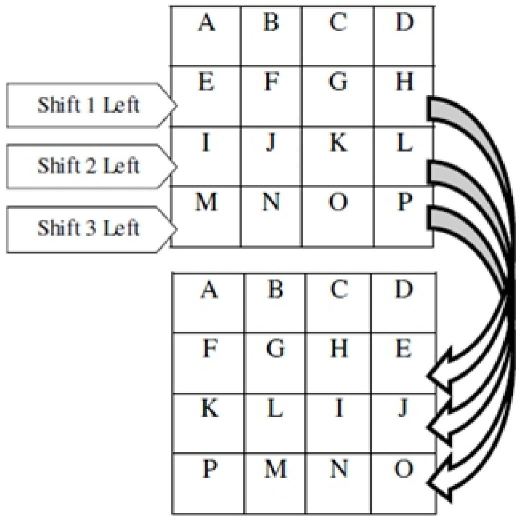

Figure-2: AES Shift Row Operation

C. Mix Column and Inverse Mix Column operation:

In the Mix Column step, the four bytes of each column of the state are combined using an invertible linear transformation.

All entries in the state matrix are considered to be a polynomial and it is multiplied by a fixed polynomial. The Mix Column and inverse Mix Column transformation are represented in matrix form. Where cij and dij are Mix Column input and Output respectively, while aij and bij are respectively the inputs and outputs of Inverse Mix Column operation.

D. Add Round Key operation:

In this operation, bitwise exclusive-or (XOR) operation is performed between outputs from Mix Column and Round Key. For AES-128, 128 bit XOR operations are performed.

III.S-BOX REALIZATION:

The conventional S-box architecture using composite field arithmetic. The meaning of the symbol used in this architecture has been shown in Figure-3. For the S-box mapping, following are the steps. Isomorphic mapping is the first step performed on the 8 bits sub byte input. Output of the isomorphic mapping is given to input of multiplicative inverse (MI) module.

Subsequently, inverse isomorphic mapping and affine transformations are the steps that follow. Detail explanation can be found in [5]. MI in GF represented by the symbol x-1 and multiplication in GF are the two main components falls in the critical path of the design.

$$\left.\begin{aligned} q_3^{-1} &= q_3 + q_3q_2q_1 + q_3q_0 + q_2 \\ q_2^{-1} &= q_3q_2q_1 + q_3q_2q_0 + q_3q_0 + q_2 + q_2q_1 \\ q_1^{-1} &= q_3 + q_3q_2q_1 + q_3q_1q_0 + q_2q_0 + q_2 + q_1 \\ q_0^{-1} &= q_3q_2q_1 + q_3q_2q_0 + q_3q_1 + q_3q_1q_0 + q_3q_0 \\ &\quad + q_2 + q_2q_1 + q_2q_1q_0 + q_1 + q_0 \end{aligned}\right\} \tag{1}$$

We have optimized the delay and area of the MI in GF as well as multiplication in GF (6).

A. Proposed logic for MI in GF:

From eq-1, it is evident that the realization of MI in GF requires a number of exclusive-or (XOR) gates. By eliminating the XOR gates, delay and area can be reduced.

TABLE I shows the input and output combination of MI in GF. The input combinations can be divided into two equal halves. In the first half, MSB will have value '0' and in the second half, MSB will be '1'.

This can be realized by a multiplexer, wherein, for '0' MSB in input, the 4-bit output will be given by combination of three input bits (except MSB). Similarly, for MSB= '1', the 4- bit output will have 3-bits input combination (except MSB).

The combinational logic for first half in terms of three input bits is given by the Eq. (2a). Eq. (2b) represents the combinational logic for the second half. By using a multiplexer, one of the outputs, either from (2a) or from (2b) can be selected depending on the MSB of the input.

It is obvious from Eq. (2a) and Eq. (2b) that the combinational logic contains only OR and AND gates instead of XOR gates.

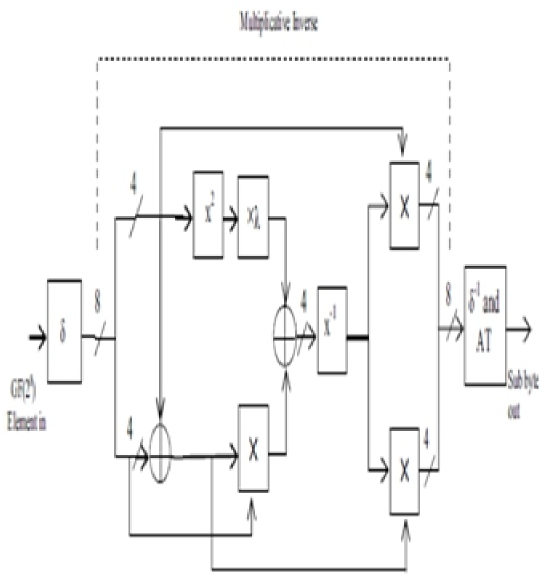

Figure-3: conventional S-box architecture in composite field

$$\left.\begin{aligned}
q_3^{-1} &= \left(q_2\overline{q_1q_0}\right)or\left(q_2\overline{q_1}q_0\right)or\left(q_2q_1\overline{q_0}\right)or\left(q_2q_1q_0\right)\\
q_2^{-1} &= \left(q_2\overline{q_1}\,\overline{q_0}\right)or\left(q_2\overline{q_1}q_0\right)\\
q_1^{-1} &= \left(\overline{q_2}q_1\overline{q_0}\right)or\left(\overline{q_2}q_1q_0\right)or\left(q_2\overline{q_1}\,\overline{q_0}\right)or\left(q_2q_1q_0\right)\\
q_0^{-1} &= \left(\overline{q_2q_1}q_0\right)or\left(\overline{q_2}q_1\overline{q_0}\right)or\left(q_2\overline{q_1q_0}\right)or\left(q_2q_1\overline{q_0}\right)\\
&\quad or\left(q_2q_1q_0\right)
\end{aligned}\right\} \tag{2a}$$

$$\left.\begin{aligned}
q_3^{-1} &= \left(\overline{q_2q_1q_0}\right)or\left(\overline{q_2}q_1\overline{q_0}\right)or\left(q_2\overline{q_1}q_0\right)or\left(q_2q_1\overline{q_0}\right)\\
q_2^{-1} &= \left(\overline{q_2}\,\overline{q_1}q_0\right)or\left(\overline{q_2}q_1q_0\right)or\left(q_2\overline{q_1q_0}\right)or\left(q_2\overline{q_1}q_0\right)\\
&\quad or\left(q_2q_1\overline{q_0}\right)or\left(q_2q_1q_0\right)\\
q_1^{-1} &= \left(\overline{q_2q_1q_0}\right)or\left(\overline{q_2q_1}q_0\right)or\left(\overline{q_2}q_1q_0\right)or\left(q_2\overline{q_1}q_0\right)\\
q_0^{-1} &= \left(\overline{q_2}q_1q_0\right)or\left(q_2\overline{q_1q_0}\right)or\left(q_2q_1\overline{q_0}\right)
\end{aligned}\right\} \tag{2b}$$

IV.SIMULATION REULTS:

A. Key generation:

Initially the 128 bit key is given as the input key. For every rising edge of the clock and rst=1, the key is loaded into the register and key generation is enabled, and finally eleven round keys are generated. After the key generation the key generation which was initially low becomes high indicating the completion of key generation. The Simulation Results of Key Generation is shown in Figure-4.

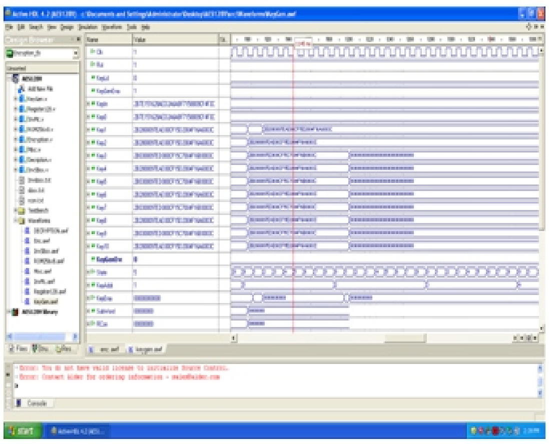

Figure-4: wave form of key generation

B. Encryption:

Functional description: if reset becomes 0 then dreg becomes zero. For every raising edge of the clock, if enable is high then our data will be XOR with the generated keys else the data in the dreg will not change. The Simulation Results of 128 bit ASE–Encryption block is shown in Figure-5.

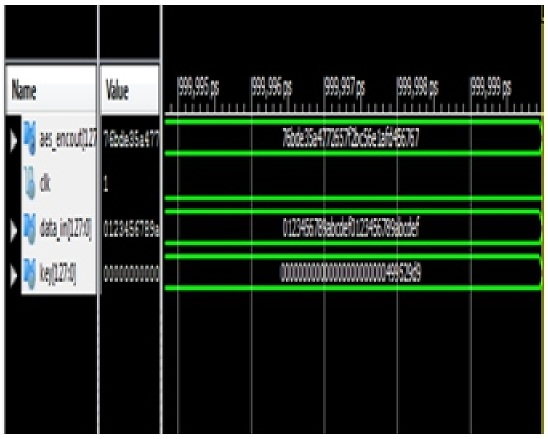

Figure-5: Simulation waveform of 128 bit AES (Encryption)

C. Decryption:

Decryption process is essentially the same as the encryption process; in the decode routine the Cipher text is used as input to the algorithm, but the sub keys K[i] are used in the reverse order.

The intermediate value of the decryption process is equal to the corresponding value of the encryption process with the two halves of the value swapped. Basically, we have inversed the whole encryption and applied all the operations backwards.

As the key schedule stays the same, the only operations we need to implement are the inversed sub Bytes, shift Rows and mix Columns, while add Round Key stays the same. The Simulation Results of 128 bit ASE – Decryption block is shown in Figure-6.

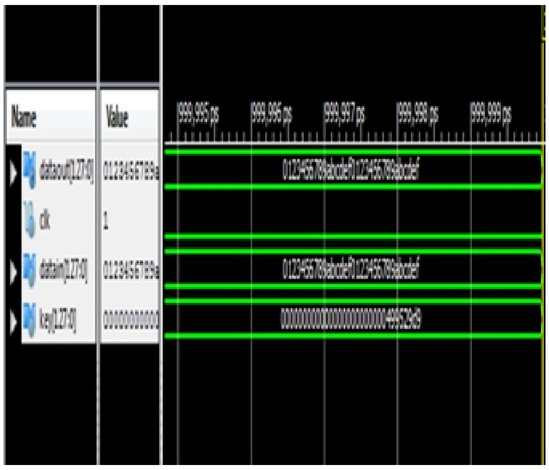
Name
Value
dataout[127:0]
clk
datain[127:0]
key[127:0]
0123456789abcdef0123456789abcdef
0000000000000000000000000449523b9

Figure-6: Simulation waveform of 128 bit AES (Decryption)

V.RTL RESULTS:

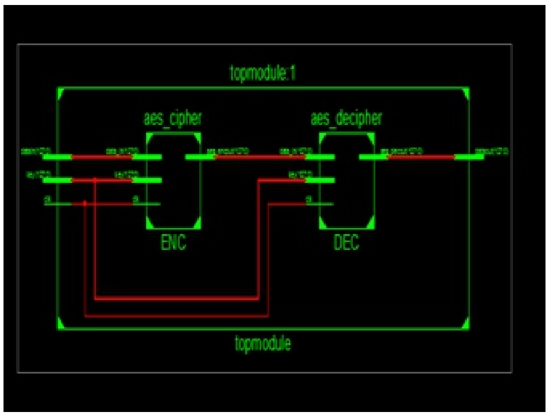

Figure-7: RTL of AES Top Module

The Hardware description languages are most used for is the Register Transfer Level (RTL). Between gate level on the low abstraction side and system level on the high abstraction side, The RT level of abstraction is a good balance between corresponds to actual hardware and ease of description for hardware designers.

At this level of abstraction designs can be simulated with HDL simulators, they are synthesizable and automatic generation of hardware is provided by most hardware design EDA tools. The net list of the proposed architecture is delivered by the Xilinx synthesis tool Xilinx XST 9.2i version. Figure 5.7 shows the RTL Schematic of AES TOP Module showing input and output pins.

VI.CONCLUSION:

Advanced encryption standard is used now a day extensively in many network and multimedia applications to address security issues. Transmission and storage of sensitive data in open networked environments is rapidly growing. it has high efficiency and high performance cipher core based on the proposed multimode multiplier.

The advanced encryption standard round function is regrouped as new linear and nonlinear functions. The multimode multiplier also supports the modular addition, subtraction as well as multiplication; it also supports more scalable key sizes for AES algorithm. As the proposed integration architecture efficiently shares the hardware resources it saves more area and cost.

The proposed AES Algorithm encrypts and decrypts data in 128-bit blocks, using a 128-bit key for encryption it takes a 128-bit block of plaintext as input and a 128-bit block of cipher text as an output. Finally the output plaintext after decryption is also 128-bit data. AES has 10 rounds, meaning the main algorithm is repeated 10 times to produce the cipher text.

REFERENCES:

[1] M. Alam, S. Ray, D. Mukhopadhayay, S. Ghosh, D. RoyChowdhury, and I. Sengupta, "An area optimized reconfigurable encryptor for AESRijndael," in Proc. Conf. DATE, Apr. 2007, pp. 1–6.

[2] Y.-K. Lai, L.-C. Chang, L.-F. Chen, C.-C. Chou, and C.-W. Chiu, "A novel memoryless AES cipher architecture for networking applications," in Proc. IEEE ISCAS, May 2004, pp. 333–336.

[3]. J. Daeme and V. Rijmen, "AES proposal: Rijndael," NIST AES Proposal, June 1998.

[4]. W. Stallings, "Cryptography and network security principles and practice," Pearson edition 2009, pp. 135-160.

[5] Y. Eslami, A. Sheikholeslami, P. G. Gulak, S. Masui, and K. Mukaida, "An area-efficient universal cryptography processor for smart cards," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 14, no. 1, pp. 43–56, Jan. 2006.

[6] A. Hodjat and I. Verbauwhede, "Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors," IEEE Trans. Comput., vol. 55, no. 4, pp. 366–371, Apr. 2006.

[7] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. Boca Raton, FL: CRC Press, Oct. 1993.

[8] P. L. Montgomery, "Modular multiplication without trial division," Math. Comput., vol. 44, no. 170, pp. 519–521, Apr. 1985. [15] V. Rijmen, "Efficient Implementation of the Rijndael S-box," 2001. [Online]. Available: http://www.esat.kuleuven.ac.be/rijmen/rijndael/ sbox.pdf