Volume 3, Issue 4, October 2016 ISSN: 2348-8565 (Online)
International Journal of Modern Engineering and Research Technology
Website: http://www.ijmert.org Email: editor.ijmert@gmail.com

Implementation and Optimisation of Advanced Encryption Standard Algorithm

Sonika Gupta
Research Scholar M.Tech Scholar (DC)
Takshshila Institute of Engineering & Technology
Jabalpur (M.P.), [INDIA]
Email: sonikaguptas@gmail.com

Santosh Chouhan
Assistant Professor & Guide,
Department of Electronics & Communication Engineering
Takshshila Institute of Engineering & Technology
Jabalpur (M.P.), [INDIA]
Email: santoshchouhan@takshshila.org

ABSTRACT

With the fast progression of digital data exchange in electronic way, information security is becoming much more important in data storage and transmission. Cryptography has come up as a solution which plays a vital role in information security system against various attacks. This security mechanism uses some algorithms to scramble data into unreadable text which can be only being decoded or decrypted by party those possesses the associated key. Two types of cryptographic techniques are being used: symmetric and asymmetric. This work provides a performance evaluation methodology to estimate how the configuration of any encryption/decryption algorithm affects the performance.

Keywords:— *Plain text, cipher text, Block cipher, S-Box, Encryption, Subbytes.*

I. INTRODUCTION

The rapidly growing number of wireless communication users has led to increasing demand for security measures and devices to protect user data transmitted over wireless channels. Two types of cryptographic systems have been developed for that purpose: symmetric (secret key) and asymmetric (public key) cryptosystems. Symmetric cryptography, such as in the Data Encryption Standard (DES), 3DES, and Advanced Encryption Standard (AES), uses an identical key for the sender and receiver, both to encrypt the message text and decrypt the cipher text. Asymmetric cryptography, such as in the Rivest-Shamir-Adleman (RSA) uses different keys for encryption and decryption, eliminating the key exchange problem. Symmetric cryptography is more suitable for the encryption of a large amount of data. The AES algorithm defined by the National Institute of Standards and Technology (NIST) of the United States has been widely accepted to replace DES as the new symmetric encryption algorithm. The AES algorithm is a symmetric block cipher that processes data blocks of 128 bits using a cipher key of length 128, 192, or 256 bits. Each data block consists of a 4 × 4 array of bytes called the *state*, on which the basic operations of the AES algorithm are performed. The proposed algorithm differs from conventional AES as it has 200 bits block size and key size both. Number of rounds is constant and equal to ten in this algorithm. The key expansion and substitution box generation are done in the same way as in conventional AES block cipher. AES has 10 rounds for 128-bit keys,12 rounds for 192-bit keys, and 14 rounds for 256 -bit keys.

II. LITERATURE SURVEY

A.P. ANUSHA NAIDU et al. [1] have proposed FPGA Implementation of Fully Pipelined Advanced Encryption Standard with worldwide communication of the private and confidential data over the computing networks or internet, there is always a chance of threat of data confidentiality, data integrity and also of data availability. Murtada. M. Abdelwahab et al. [2] VLSI implementation of Advance Encryption Algorithm using index technique have idea of Symmetric algorithm is a popular encryption algorithms. The proposed algorithm is a class of symmetric algorithm providing an advance security because unlike other symmetric algorithms, the key is not available for users, it is covered and fixed inside the lookup table (LUT) as a part of the FPGA, Daniel F. García et al. [3] have proposed the Performance Evaluation of Advanced Encryption Standard Algorithm Generally, confidentiality is obtained by encrypting/ decrypting the information with a symmetric algorithm. Currently, the most used and standardized algorithm is the Advanced Encryption Standard (AES), but the encryption and decryption usually causes undesired delays in the access to information. Rafidah ahmad and widad et al. [4] Implementation of Advanced Encryption Standard Algorithm have design Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication. In data and telecommunications, cryptography is necessary when communicating over any unreliable medium, which includes any network particularly the internet. In this paper, a 128 bit AES encryption and Decryption by using Rijndael algorithm (Advanced Encryption Standard algorithm) is been made into a synthesizable using Verilog code which can be easily implemented on to FPGA.

AES -

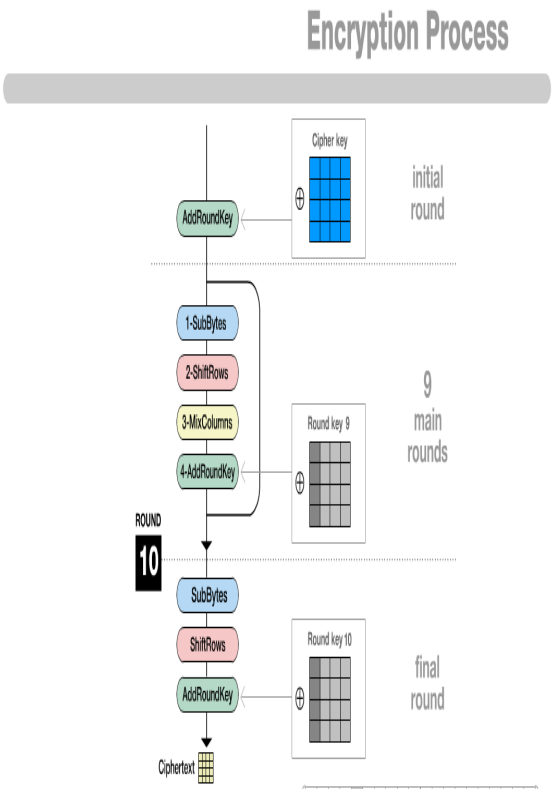

Figure 1 : Encryption Process

Stages in Cipher and Decipher

A. Byte Substitution

Byte Substitution is basically a lookup table utilizing a 16×16 double dimension of byte values known as S-box. This dimension comprises of every conceivable combos of 8 bit sequence (2^8=16 × 16 = 256). Nonetheless, the s-box isn't only an arbitrary stage of these qualities and there is an overall-characterized technique for making the s-box matrix. The architects of Rijndael demonstrated how this was carried out dissimilar to the s-box DES for which not at all justification was given. We won't be excessively interested here how the s-boxes are created and can basically take them as lookup tables. Again the dimension that gets worked upon all around the encryption is called state-matrix.

We need to be interested with how this framework is influenced in every one iteration. For this specific adjust every byte is linked into another 8 bits in the accompanying way: the left-hand side 4 bits of the half word is utilized to determine a specific row of the s-box and the right-hand side 4 bits tags a specific column. For instance, the 8 bits {95} (wavy sections speak to hex values in FIPS PUB 197) chooses line nine segment five that eventually comes out to hold the quantity {2a}, which is utilized to modify the state matrix.

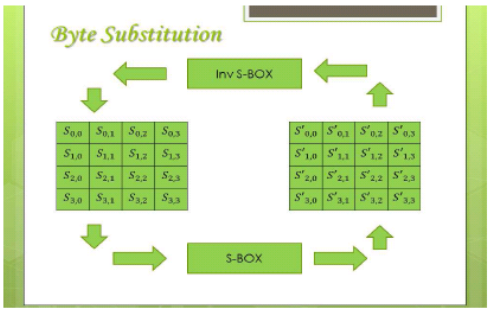

Figure 2- Byte Substitution

The Inverse byte substitution change makes utilization of an Inverse s-box. For this situation what is wanted is to choose the quality {2a} and get the worth {95}. The s-box is intended to be impervious to called cryptanalytic ambushes. Particularly, the Rijndael engineers looked for a plan that has a low connection between data bits and yield bits, and the property that the yield can't be depicted as a straightforward numerical capacity of the information. Also, the s-box has no altered focuses (s-box (a) = an) and no inverse settled focuses (s-box (a) = −a) where (an) is the bitwise compliment of a. The s-box must be invertible if decrypting is to be conceivable (Is-box[s-box (a)] = a) be that as it may it ought not to be its counter directionally toward oneself i.e. s-box (a) 6 = Is-box.

B. Subbytes Transformation:

The first transformation, Sub Bytes, is used for encryption and inverse Sub Bytes used for decryption. The Sub Bytes substitution is a nonlinear byte substitution that operates independently on each byte of the State using a substitution table (S-box). Take the multiplicative inverse in the finite field GF (2^8) and affine transform to do the Sub Bytes transformation. Inverse affine transform have to find for inverse SubBytes transformation then multiplicative inverse of that byte.

Inverse Sub Bytes transformation is inverse of SubBytes transformation. It can find in the similar way only table which is used for mapping the byte is different. The Sub Bytes transformation is done through S-box. There are two techniques to perform substitutions, (i) using S-BOX table, and (ii) using composite field arithmetic.

SubBytes table is also called as S-box and inverse SubBytes table is an Inverse S - box. There are two parts of affine transformation and its inverse; a constant matrix will be multiplied with the data in multiplication part, then the addition part, where a constant vector is added to multiplication result.

Shift Row Transformation

Shift Row Transformation is as displayed in Figure 3. This is a humble permutation and nothing more. It works as below:

- The very initial row (i.e. row 0th) of the **state matrix** isn't modified.
- The 2nd row is left shifted by 1 byte in a round path.
- The 3rd row is left shifted by 2 bytes in a round path.
- The fourth row is left shifted by 3 bytes in a round path.

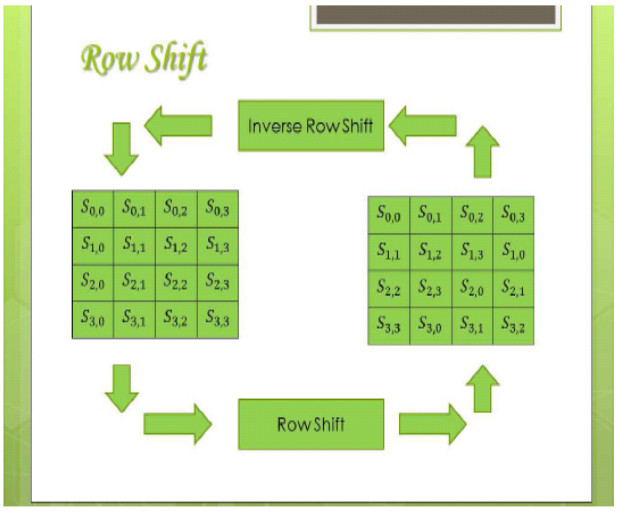

Figure 3 - Row Shift Transformation

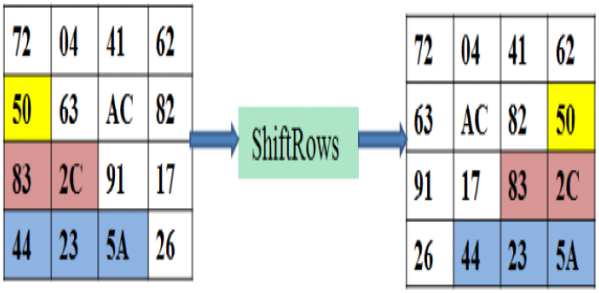

Figure 4:- Row Shift Transformation

The Inverse Shift Rows conversion (called as Inv Shift Rows) performs these round movements in the inverse heading for each of the last three columns (the first column was unchanged in any situation). This process may not seem to do abundant yet in the event that you contemplate how the bytes are requested inside state then it could be seen to have significantly a greater amount of an effect. Keep in mind that state is dealt with as a cluster of four byte sections, i.e. the main section really speaks to bytes 1, 2, 3 and 4. A one byte movement is in this way a direct separation of 4 bytes. The conversion additionally guarantees that the 4 bytes of 1 segment are extent out to 4 separate segments.

Mix Column Transformation

Mix Column Transformation is essentially a substitution yet it makes utilization of math of GF (2^8). Every segment is worked on separately. Every byte of a segment is charted into another esteem that is a capacity of each of the 4 bytes in the section. The conversion might be dictated by the accompanying grid increase on state demonstrated in Figure.

Every component of the item framework is the entirety of results of components of one line and one segment. For this situation the unique augmentations & multiplication are achieved in GF (2_). The Mix Columns change of a solitary segment j (0 _ j _ 3) of state could be communicated as: Where x means multiplication over the finite field GF (2_).

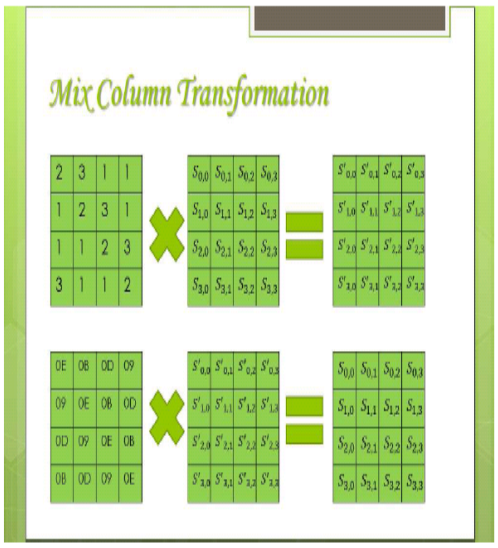

Figure 5: Mix column Transformation

III. Proposed Methodology of S-BOX

The new architecture of S-BOX has proposed after 3 modifications in conventional architecture of S-BOX.

(a) Introduced an operator (op) after merging of some blocks.

(b) Implementation of multiplicative inverse in GF (2^4) using multiplexor.

(c) Reduced the critical path of multiplication in GF (2^2)

The Sub Bytes transformation, done through S-BOX mapping is computationally inefficient when implemented using a ROM. But, it is not efficient for applications requiring very high throughput as ROM accessing involves one complete clock cycle for mapping one 8-bits state element and consequently 16 clock cycles are required to transform the 128 bits data (16 bytes). To increase the throughput, parallel ROMs are required resulting in large size of chip area. Therefore, a more feasible solution is to implement an S - box is by using composite field arithmetic which uses only logic elements in the implementation. The S-BOX substitution starts by finding the multiplicative inverse of the data in GF (2^8), and then applying the affine transformation. Figure shows steps for the one byte forward and inverse Sub Bytes transformation using composite field arithmetic.

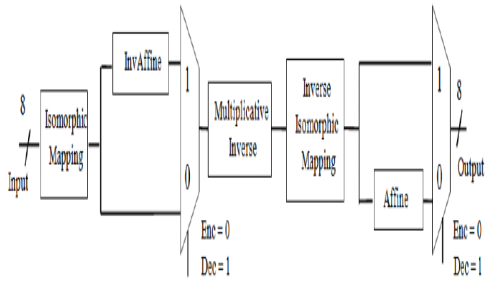

Figure 6 : Sub Bytes and Inverse Sub Bytes transformation

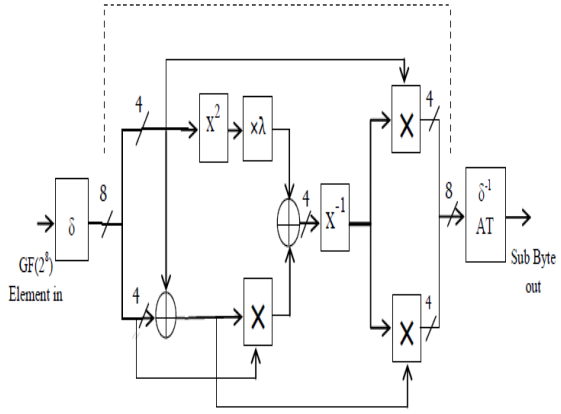

Figure 7: The Proposed S-box architecture in composite field

To find the S-BOX transformation first multiplicative inverse of GF (2^8) then affinetransformation calculated. Similarly, for Inv Sub Bytes first InvAffine transformation then multiplicative inverse has to be calculated. There are one major operation involve here, which is to find the multiplicative inverse in GF (2^8). This can be done by breaking the GF (2^8) elements in GF (2^4) and some more logical blocks. I.e., Any arbitrary polynomial in GF (2^8) can be represented as bx+c using an irreducible polynomial x^2+Ax+B. Here, b is the most significant nibble and c is the least significant nibble. The multiplicative inverse can be found by using the following equation.

IV. Synthesis & Simulation Results

We have designed our AES architecture in VHDL Language using Xilinx 14.1 for Spartan 3E XC3s500e FPGA. Our architecture takes 128 bits of data as input along with the 128 bits of key along with three control signal clk, go_i, and reset signal each of single bit. The block diagram of the AES block is provided below in Figure

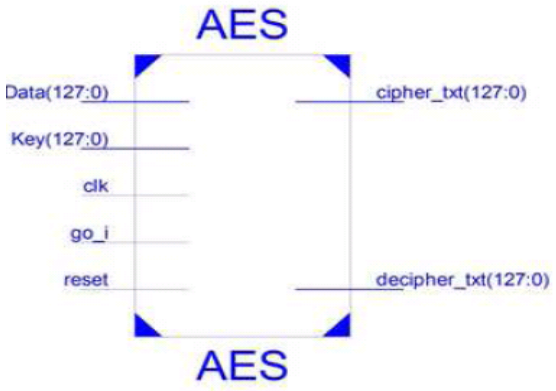

Figure 8: Top view of cipher block.

Simulation: we have implemented the complete encryption and decryption modules of Advanced Encryption Standard that is all the four transformations that are used at encryption and at decryption side. The Simulation results of complete AES encryption and AES decryption is shown below in Figure 5(a) and Figure 5(b). Here All the transformations have been simulated by using Xilinx ISE Design suite.

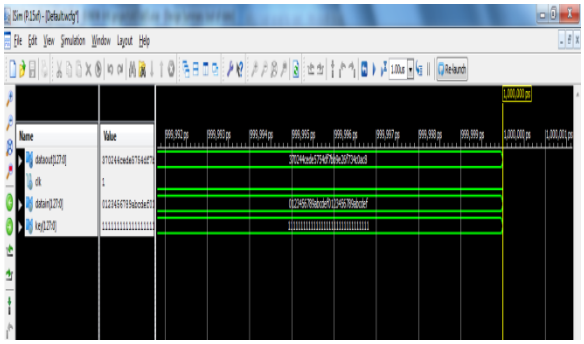

Figure 9: AES encryption

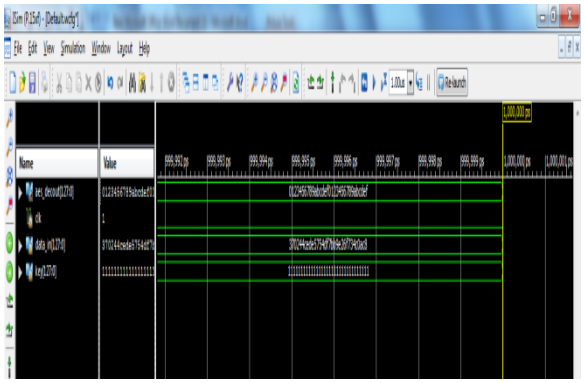

Figure 10 : AES decryption

VI. COMPARATIVE RESULT

Logic Utilization	Murtada et al[2]	Rafidah ahmad et al[4]	Proposed
Number of slices	294	520	241
Number of slice flip flops	512	1994	468
Number of 4 input LUTS	336	1864	472

VII. CONCLUSION

We have proposed optimized VLSI architecture of S-box for AES algorithm. The architecture of s-box in composite field has been modified in order to have high speed and low areas. This thesis was successfully completed with the implementation of AES algorithm on 128 bit message. The encrypted cipher text and the decrypted text are analyzed and proved to be correct. The encryption efficiency of the proposed AES algorithm was studied and met with satisfactory results.

REFERENCES:

[1] A.P. ANUSHA NAIDU, B. Prof (Mrs.) POORVI K. JOSHI, FPGA Implementation of Fully Pipelined Advanced Encryption Standard, IEEE ICCSP 2015 conference.

[2] Murtada. M. Abdelwahab and Abdelrasoul J. Alzubaidi, VLSI implementation of Advance Encryption Algorithm using index technique, International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering, 2015.

[3] Daniel F. García, Performance Evaluation of Advanced Encryption Standard Algorithm, Second International Conference on Mathematics and Computers in

Sciences and in Industry 2015.

[4] Rafidah Ahmad and Widad Ismail, Implementation of high performance Advanced Encryption Standard -128 for wimax application on FPGA, IEEE 2014

[5] Ritu Pahal and Vikas Kumar, Efficient Implementation of AES, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 7, July 2013.

[6] B.A. Forouzan and D. Mukhopadhyay, Cryptography and Network Security, 2nd Ed., Tata McGraw Hill, New Delhi, 2012.

[7] M. I. Soliman, G. Y. Abozaid, "FPGA implementation and performance evaluation of a high throughput crypto coprocessor," *Journal of Parallel and Distributed Computing*, Vol. 71 (8), pp.1075-1084, Aug. 2011.

[8] V. K. Pachghare, Cryptography and information security, E. E. Ed., PHI Learning, New Delhi, 2009.

[9] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A Lightweight High-Performance Fault Detection Scheme for the Advanced Encryption Standard Using Composite Fields," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 19 (1), pp. 85-91, Jan. 2011.

* * * * *