International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.3., Issue.6., 2015
(Nov.-Dec.,)

RESEARCH ARTICLE

ISSN: 2321-7758

DESIGN AND IMPLEMENTATION OF RIJNDAEL ENCRYPTION ALGORITHM BASED ON FPGA

NAVEEN KUMAR PAAYILI[1], GATTADI VINATHA[2]

[1]M.Tech Scholar, St.Martin's Engineering College, Dhoolapally

[2]Associate Professor, St.Martin's Engineering College, Dhoolapally

NAVEEN KUMAR PAAYILI

GATTADI VINATHA

ABSTRACT

Advanced Encryption Standard (AES) is a Federal Information Processing Standard (FIPS) and categorized as Computer Security Standard. The AES algorithm is a block cipher that can encrypt and decrypt digital information. The AES algorithm is capable of using cryptographic keys of 128, 192 and 256 bits. The Rijndael cipher has been selected as the official Advanced Encryption Standard (AES) and it is well suited for hardware. The objective of this paper was to present the hardware implementation of Advanced Encryption Standard (AES) algorithm. This paper proposes an efficient solution to combine Rijndael encryption and decryption in one FPGA design, with a strong focus on low area constraints and high throughput. This Rijndael implementation runs its symmetric

cipher algorithm using a key size of 128 bits, mode called AES128. We have worked with the pipelining structure and modifications such as merging of Subbytes and Shift Rows, and optimization of each clock cycle to incorporate maximum number of operations etc. have been successfully implemented. The encryption and decryption process of Rijndael algorithm was captured in VHDL language and corresponding FPGA implementation resulted in reduced number of slices (6901) and achieved a data throughput of 38.346 Gbps which is implemented on Xilinx 14.2 Virtex5.

Key Words: AES, Cipher text, Cryptography, FIPS, FPGA, Plaintext, pipelining

INTRODUCTION

Cryptography is the art and science of protecting information from undesirable individuals by converting it into a form of non-recognizable by its attackers while stored and transmitted. Data cryptography mainly is the scrambling of the content of data, such as text, image, audio, video and so forth to make the data unreadable, invisible or unintelligible during transmission or storage called Encryption. The main goal of cryptography [1] is keeping data secure from unauthorized attackers. The reverse of data encryption is data Decryption. As we know, the security strength of Data Encryption Standard (DES) [2] has been difficult to adapt to new needs. In October of 2000, the National Institute of Standards and Technology (NIST) selected the Rijndael algorithm as the advanced encryption standard (AES), which was

developed by Joan Daemen and Vincent Rijmen, in order to replace the DES. At present, Rijndael is the most common and widely used symmetric cryptosystem to support bulk data encryption. It offers a good "combination of suppleness, efficiency and safety" As the network transmission speed upgrades to the gigabits per second (Gbps), the software based implementations of cryptographic algorithms cannot meet its needs. The hardware-based implementations using some special optimization techniques (such as pipeline and lookup tables, etc.), can greatly improve throughput and reduce the key generation time. Besides, the processes of cryptographic algorithms and the key generation packaged in chip, which cannot easily be read or changed by external attacker, so hardware-based implementations can get the higher physical security. Some implementations use the field programmable gate arrays (FPGA) and the others use the Application Specific Integrated Circuit (ASIC). ASIC lacks of flexibility and has high development costs and long development cycle. Reconfigurable devices such as FPGA, with hardware of security and high speed and software of flexibility and easy maintenance, have become hardware based implementations research hotspots for block cipher algorithm.

Applications such as smart cards and cellular phones, digital cinema and pay TV, require small area. Applications, such as WWW servers and ATM's require high throughput.In this context, the highest effort was devoted to high throughput Encryption and Decryption designs. Architectural optimization exploits the strength of pipelining, loop unrolling and sub-pipelining. Speed is increased by processing multiple rounds simultaneously at the cost of increased area.

II. ADVANCED ENCRYPTION STANDARD (AES) / RIJNDAEL.

In the late 1990s, the U.S. National Institute of Standards and Technology (NIST) conducted a competition to develop a replacement for DES [1]. The winner, announced in 2001, is the Rijndael (pronounced "rhine-doll") algorithm, destined to become the new Advanced Encryption Standard. Rijndael mixes up the SPN model by including Galois field operations in each round. Somewhat similar to RSA modulo arithmetic operations, the Galois field operations produce apparent gibberish, but can be mathematically inverted. AES have Security is not an absolute; it's a relation between time and cost. Any question about the security of encryption should be posed in terms of how long time, and how high cost will it take an attacker to find a key? Currently, there are speculations that military intelligence services possibly have the technical and economic means to attack keys equivalent to about 90 bits, although no civilian researcher has actually seen or reported of such a capability.

Actual and demonstrated systems today, within the bounds of a commercial budget of about 1 million dollars can handle key lengths of about 70 bits. An aggressive estimate on the rate of technological progress is to assume that technologies will double the speed of computing devices every year at an unchanged cost. If correct, 128-bit keys would be in theory be in range of a military budget within 30-40 years. An illustration of the current status for AES is given by the following example, where we assume an attacker with the capability to build or purchase a system that tries keys at the rate of one billion keys per second. This is at least 1000 times faster than the fastest personal computer in 2004. Under this assumption, the attacker will need about 10 000 000 000 000 000 000 000 years to try all possible keys for the weakest version, AES-128. The comparison between AES, 3DES and DES is shown in table 1**.**

III. THE AES ALGORITHM

The Encryption/Decryption process of Advanced Encryption Standard algorithm is presented below, in fig1 .

This block diagram is generic for AES specifications. It consists of a number of different transformations applied consecutively over the data block bits, in a fixed number of iterations, called rounds, where Nr = 10, 12 and 14. The number of rounds depends on the length of the key used for the encryption/Decryption process, and the blocks are described Figure 1.

Table 1: Comparison between AES, 3DES and DES [2]

Factor	AES	Triple DES	DES
Key Length	128,192 or 256 bits	(k1, k2, k3) 168 bits (k1 and k2 is same) 112 bits	56 bits
Cipher Type	Symmetric Block Cipher	Symmetric Block Cipher	Symmetric Block Cipher
Block Size	128,l 192 or 256 bits	64 bits	64 bits
Developed Yr	2000	1978	1977
Cryptanalysis Resistance	Strong against differential, truncated differential, linear, interpolation and square attacks.	Vulnerable to differential, Brute force Attack could be analyze pain text using differential cryptanalysis.	Vulnerable to differential and linear cryptanalysis; weak substitution tables.
Security	Considered secure	One only weak which is exit in DES	Proven inadequate
Possible Keys	2^{128}, 2^{192} or 2^{256}	2^{112} or 2^{168}	2^{56}
Possible ASCII Printable character Keys	95^{16}, 95^{24} or 95^{32}	95^{14} or 95^{21}	95^{7}
Time to check all possible Keys at 50 billion Keys per sec.	For a 128 bit key: 5x 10^{21} years	For a 112 bit key: 800 days	For a 56 bit key: 400 days

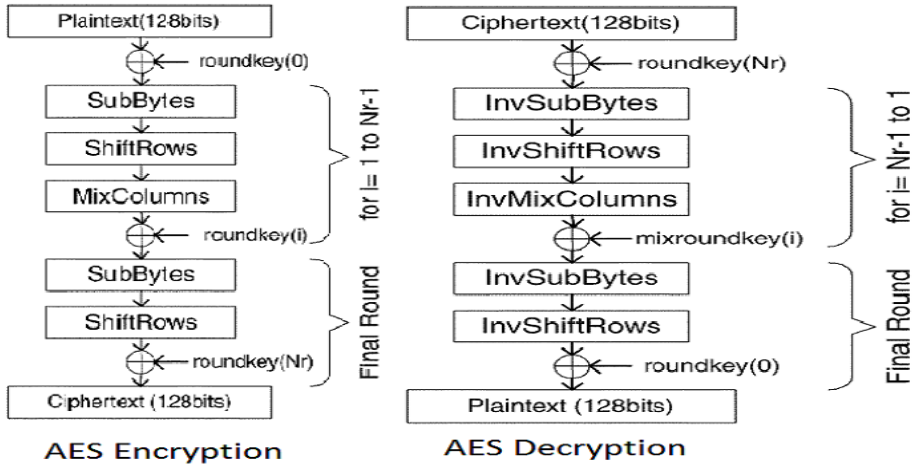

Figure 1: AES Encryption/Decryption [3].

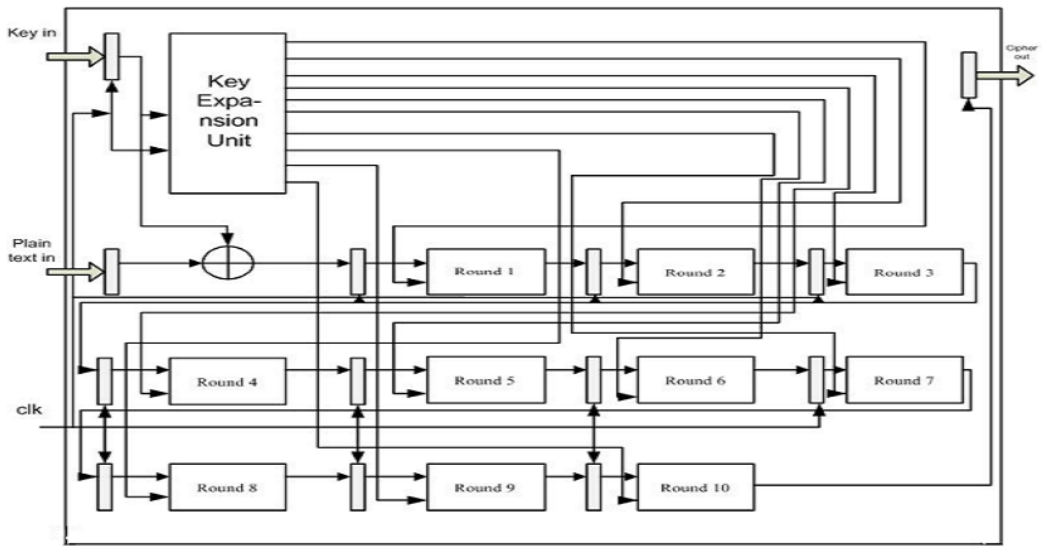

Figure 2: Proposed AES Top Module.

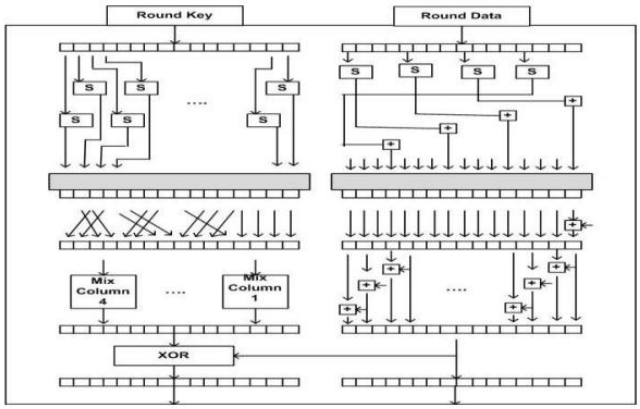

Figure 3: Operations inside Round Component.

A Bytes substitution

The bytes substitution transformation Bytesub (state) is a non-linear substitution of bytes that operates independently on each byte of the State using a substitution table(S-box) and for Decryption inverse (S-Box).

B Shift Rows

In the Shift Rows transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes. The first row is not shifted. The second row is left-shifted circularly one byte. For the third row, a 2-byte circular left shift is performed. For the fourth row, a 3 -byte circular left shift is performed. And for the decryption process the cyclically shifting is to the right.

C Mix Columns Transformation

This transformation is based on Galois Field multiplication. Each byte of a column is replaced with another value that is a function of all four bytes in the given column. The Mix Columns transformation operates on the State column by column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF (28) and multiplied modulo x4 + 1 with a fixed polynomial a(x), given by the following equation.

a(x) = {03}x3+{01}x2+{01} x +{02}.

This can be written as a matrix multiplication. Let

$S'(x) = a(x) \otimes S(x)$

$$\begin{bmatrix} S'_{0,c} \\ S'_{0,c} \\ S'_{0,c} \\ S'_{0,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{0,c} \\ S_{0,c} \\ S_{0,c} \end{bmatrix}$$

For $0 \le c < Nb$.

For decryption process the polynomial is,

$$\begin{bmatrix} S_{0C}' \\ S_{1C}' \\ S_{2C}' \\ S_{3C}' \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} * \begin{bmatrix} S_{0C}' \\ S_{1C}' \\ S_{2C}' \\ S_{3C}' \end{bmatrix}$$

D AES Add Round Key

AddRoundKey operation is designed as a stream cipher; all the 128 bits of state are XORed with 4, 32-bit words of expanded key resulting from key expansion. AddRoundKey is the only operation that involves using the key to ensure security. The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a four word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.

IV. FPGA IMPLEMENTATION OF AES

In this chapter a new architecture for a high speed AES encryption, decryption and combined encryption and decryption using 128-bits key size is presented. This architecture is implemented using fully pipelining method. This new architecture has shown greater performance in terms of throughput and Area comparing to previous pipelined AES Cryptography.The proposed top module of AES Encryption is shown in fig 2, Similar to the existing pipelined designs, the proposed structure first uses loop unrolling to expand the 10- round operations

and adds registers between rounds, forming a one pass data path for plaintexts. For Decryption the rounds will be 10 to1.

In FPGAs, the logic depth, i.e. the number of combinational logic levels, is the number of cascaded LUTs and MUXs that data go through from the output of a register to the input of an adjacent register. To reduce the path delay, the combinational logic paths can be broken by inserting registers. The operations between different pairs of registers can be further pipelined. The routing delay in FPGAs depends on the locations of the hardware logic resources on which the circuit elements are mapped.

Using the embedded hard blocks which lie in fixed locations, e.g. Block RAMs, may increase the routing delay. Therefore, in this paper, the pipelining design of the operations in each round of AES is studied. The four functions of each AES round is shown in Fig 3, where the detailed operations of each function are illustrated further.

A Implementation of Substitution Bytes (S-Box) / Inverse Substitution Bytes.

In the SubBytes step, each byte in the array is updated using an 8-bit substitution box, the Rijndael Sbox. This S-Box is stored in RAM and mapped to the input text whenever needed. The nonlinearity in the cipher is provided by this operation. The procedures to be followed to substitute the bytes in the matrix are:

- Select any element say 18 which is in hexadecimal notation.
- The S-box element in the 1st row and 8th column is to be selected and substituted in its place (i.e., ad).

Similarly for other elements in the matrix. The implementation of Inverse Substitution Bytes is same as s-box implementation in encryption process, except each byte in the array is updated using an 8-bit inverse substitution box, i. e, The Rijndael inverse S-box.

B Implementation of Shift Row/inverse shift row

In the Shift Rows transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes, this shifting is done by modulo operator. For example in the 2nd row one byte left shift has to be done i.e., the byte at the location (1, 2) shifted to the location (1, 1) i.e., by the operator (i, (j -i) mod 4). Similarly for 3 rd row and 4th row the shifting can be done. As the implementation is done in the pipelining methodology, in the same block some part of key expansion is also done and next key generates. The implementation of Inverse Shift Row is same as shift row implementation in encryption process except the shifting is circular to the right.

C Merging of Sub Bytes and Shift Rows

This merging is performed by calling required shifted element from the data matrix, instead of calling element one by one sequentially from the data matrix. Thereby sub-byte and shift row operations are carried out in one-step instead of two figures as shown in figure 4. The 16 elements are stored sequentially after each round in a register file. Using Mux selection, required shifted data elements are called (instead of calling sequentially) from the register file and put into the State. This merging process would increase throughput since elements are not called sequentially and a balance between throughput and area is maintained.

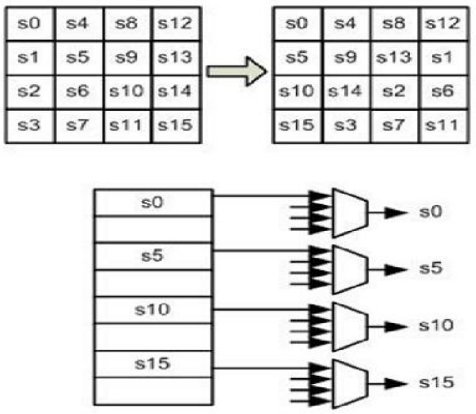

Figure 4: Merging of Shift Row and S-Box.

D Implementation of Mixcolumn

In mix Column, four bytes in the corresponding position in the four “rows” are used for matrix multiplication in GF (28), which involves byte-wise multiplication and addition. Byte wise additions are easily done by XOR, and several tricks are used for multiplications. The architectural view of mix column is shown in fig 5.

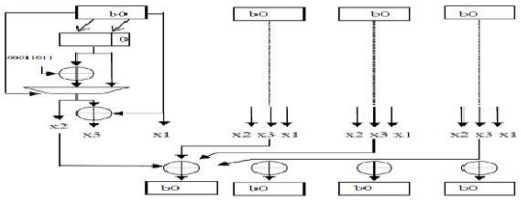

Figure 5: Circuit Architecture of MixColumn [4].

Byte-wise multiplications include multiplying the data by 1, 2, and 3. Multiplying by 1 the data will remains the same. For multiplication by 2, the 8 bit data is left shifted by 1 bit, and the LSB is replaced by 0. Then the MSB of the original data is used for comparison. If it is 0, then the left shifted data is the result; if it is 1, then the left shifted value is XORed with the reduction polynomial, in this case 00011011, to generate the result. For multiplication by 3 the original byte is simply XORed with the result of multiplication by 2. Using the above method, the multiplication by 1, 2, and 3 of each of the bytes in the data are determined.

Then the correct combinations of values are XORed with each other to produce a new byte. The same process goes on until all the 16 bytes in the data are replaced.

E Inversemixcolumn

The IMC transformation used in decryption is more complicated .Constant multiplications used in the Inverse MixColumns transformation can be expressed as

$$\{0b\}X = \{08\}X \oplus \{02\}X \oplus X,$$
$$\{0d\}X = \{08\}X \oplus \{04\}X \oplus X,$$
$$\{09\}X = \{08\}X \oplus X,$$
$$\{0e\}X = \{08\}X \oplus \{04\}X \oplus \{02\}X.$$

S'0, C (0< c <4) can be calculated in the IMC transformation is illustrated in Fig. 6. An XTime block is used.to simplify the diagram. XTime block implements the constant multiplication by {02} in GF (28), each XTime block consists of 4 XOR gates and the critical path includes only one XOR gate. In the IMC transformation, the calculation for the other bytes can be carried out similarly according to (1). As shown in Fig. 8, the critical path is 6 XOR gates, and a total of (10 x 8 +3 x 4) x 4 = 368 XOR gates is needed to implement the inverse MixColumns transformation for one column of the State.

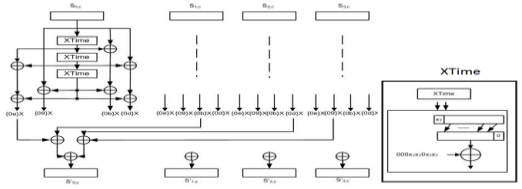

Figure 6: Architecture of inveremixcolumn and XTime[5]

F Key expansion

The key expansion unit is responsible of generating the 10 rounds of keys required during the encryption process, the figure 7. Represents key expansion.

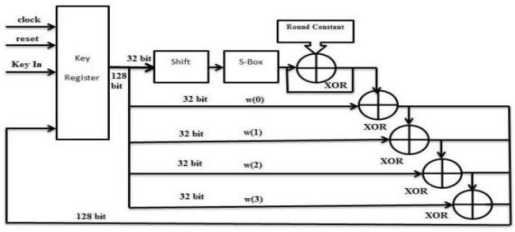

Figure 7: Key Expansion

This unit will be using 4 s-boxes in processing the first column of each group of the round keys, while the XOR gates will be used in processing all the columns by adding the direct previous column with the equivalent column from the previous group. The 4 registers following the multiplexers will be used to store the direct previous group of Round Keys to be used in generation for the ongoing group, while the registers (R1 -R11) will be storing the all the 10 rounds of keys. The original input key will be transformed to the isomorphic mapping so that all generated round keys will also be in the isomorphic transformation.

G Implementation of AES Cryptography.

The combined implementation of encryption and decryption is shown in figure 8. It requires a Plain / Cipher Text which is of 128 bits length. Also Key of length 128 bits is needed to be given. The input data is processed using the key and the resultant data is available at the output terminal named as plain text out. The Key Expansion Module is common for both the Encryption and Decryption module.

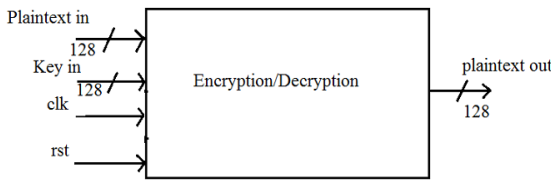

Figure 8: AES Cryptography Top Module

Table 4 summarizes the obtained results and the comparison with previous different implementations.

Looking through Table I, we can find that the performance of our design is improved over old ones to the new ones.

H Comparison with previous work.

Table 2: Comparison of AES Encryption/Decryption with Previous Designs

Author name	Platform	Slices	Frequency	Throughput
Refik Sever et.al [6]	NA	4189CLB/ 4 RAM	65 MHz	1.19 Gbps
Nalini C et.al [3]	Xcv3200	4626 slices	241.313MHz	30.88Gbps
SwinderKaur et.al [4]	Xcv2vp70-7	6279 Slices	119.954MHz	1.18 Gbps
BanraplangJyrwa et.al [7]	Xcv2vp30-ff896	6211Slices	142.5MHz	1458 Mbps
Abhijith.p.s [8]	Xc5vlx110f-3-ff1136	NA	292.40	37.4 Gbps
This work	xc5vlx50-2-ff676	6901 slices	299.58	38.346 Gbps

Table 3: The synthesis report of AES Enc/Dec

Device	xc5vlx50-2-ff676	Xc7v585t
Frequency	299.58(MHZ)	512.82(MHZ)
Throughput	38.346 Gbps	65.64 Gbps
Slices	6901	6569
Critical path delay	3.338	1.95
IOB	386	386

V. RESULTS

Implementation is done on the device: xc5vlx50-2-ff676, Xc7v585t. The behavioural simulation of AES Encryption/Decryption

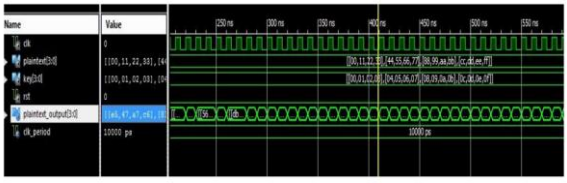

Figure 9: simulation result AES Cryptography

The synthesis report of AES Enc/Dec Design after place and route on the devices xc5vlx50-2-ff676, Xc7v585t.

VI. CONCLUSION

AES-128 bit algorithm for encryption and decryption is implemented in Virtex-5/7 FPGA. The objective of this paper was to present the hardware implementation of Advanced Encryption Standard (AES) algorithm and area-throughput balanced implementations of the Rijindael have examined. We have worked with the pipelining structure and modifications such as merging of Subbytes and Shift Rows, and optimization of each clock cycle to incorporate maximum number of operations etc. have been successfully implemented. The encryption and decryption process of Rijndael algorithm was captured in VHDL language and corresponding FPGA implementation resulted in reduced number of slices and achieved a high data throughput. Since the speed is higher than the already reported systems, hence the proposed design serves as the best high speed AES algorithm and is thus suitable for various applications.

REFERENCES

[1]. William Stalling, Cryptography and Network Security: principles and practice.

[2]. Hamdan O. Alanaji, A. A Jaidan, B. B Jaidan, Hamid A. Jalab and Y. Al-Nabani, "New Comparative Study Between DES, 3DES, AES Within Nine Factors.," *Journal of Computing,* vol. 2, no. 3, pp. 152-157, 2010.

[3]. Nalini C, Nagaraj, Dr. Anand Mohan and Poornaiah D, "An FPGA Based Performance

Analysis of Pipelining and Unrolling of AES Algorithm," *IEEE,* 2006

[4]. Swinder Kaur and Prof. Renu Vig, "Efficient Implementation of AES Algorithm in FPGA Device," in *IEEE*, 2007.

[5]. Shylashree. N, Nagarjun Bhat and V Sridhar, "FPGA implementations of advanced encryption standard": a survey," in *ijaet*, 2012.

[6]. Refik Sever, A. Neslinsmailoglu, Yusuf. C, Tekmen and Burak Okcan, "BurakOkcan A High speed fpga Implementation of the Rijndael Algorithm," in *IEEE*, Proceedings of the EUROMICRO Systems on Digital System Design (DSD'04), 2004.

[7]. Banraplang Jyrwa and Roy Paily, "An Area-Throughput Efficient FPGA implementation of Block Cipher AES algorithm," in *IEEE*, 2009.

[8]. Abhijith P. S and Manish Goswamy, "High performance Hardwre Implementation of AES Using Minimal Resources," in *IEEE*, 2013.