Lightweight Mixcolumn Architecture for Advanced Encryption Standard

K.J. Jegadish Kumar
Associate professor
SSN college of engineering
kalvakkam, Chennai-603110

R. Balasubramanian
Post graduate scholar
SSN college of engineering
kalvakkam, Chennai-603110

ABSTRACT

Lightweight cryptography is an interesting phenomenon that provides the perfect trade-off among security, higher throughput, low-power consumption, and compactness. Designing lightweight cryptography is a challenging issue. In this paper, Mixcolumn operation in the Advanced Encryption Standard (AES) is modified based on cellular automata functions. AES lacks compactness, but have good accessibility than the other algorithms. Security analysis like bent functions, Fast Walsh Transform method is followed to verify the security in modified AES algorithm. Hardware implementation of modified AES offers efficient memory space and area consumption. Comparative study of traditional mixcolumn architecture and Cellular automata based mixcolumn architecture are made through the hardware simulation in Xilinx, to show FPGA implementation of AES results as lightweight cipher, in terms of memory requirement.

General Terms

AES, mixcolumn, Cellular automata, security, FPGA.

Keywords

Lightweight cryptography, Mixcolumn, AES, cellular automata, security, area consumption, FPGA.

1. INTRODUCTION

Cryptography literally means hidden writing, which ensuring the security and privacy of information and communications [1]. In cryptography, U.S government adopted the symmetric-key encryption standard called as Advanced Encryption Standard (AES). The input 128 bit block is given to AES cipher as a 2-dimensional matrix of 16 bytes called State matrix with various key size of 128,192 and 256 bits. For 128-bit key size, there will be 10 rounds of transformations. There are four steps of round transformations: Byte substitution. Shift rows, Mixcolumns and Add round key. All four transformations contribute in AES strength by inducing confusion and diffusion properties. The implementation of Advanced Encryption Standard (AES) in reconfigurable hardware was based on various factors such as power, throughput and area. Based on cryptographic properties, Boolean functions play a vital role in constructing symmetric cryptographic systems. In block ciphers, these functions are used in S-box and mixcolumn design whereas in stream cipher these functions are used as nonlinear filters and combiners. Generally, Boolean functions in cryptography should satisfy various properties simultaneously, mainly high nonlinearity, Balancedness, and good autocorrelation properties, to resist against linear cryptanalysis and differential cryptanalysis attacks [2].

The rest of paper is such that Section 2 explains about the related work for efficient hardware implementation of AES. Section 3 explains about proposed work. Section 4 explains about performance of proposed work and results. Finally, Section 5 gives conclusion and future work directions of the work.

2. RELATED WORK

N. Ahmad and S.M. Rezaul Hasan (2012) [3] discussed that for various security applications, they require a low cost and low power design. AES will adopt for these applications as a symmetric algorithm. A new architecture core of 8-bit stream cipher was proposed for an AES crypto-processor which is an application specific integrated circuit.. Here optimization of power and chip area was carried out, in order to attain high throughput.

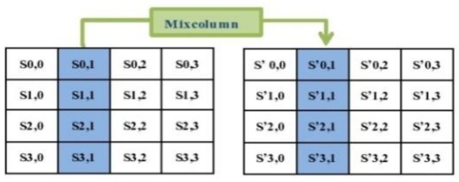
LI Zhen-rong et al (2009) [4] proposed a design on Zigbee system-on-a-chip (SoC) with low power advanced encryption standard (AES) coprocessor. They are able to achieve low cost and low power coprocessor by optimizing the SubBytes/s-boxes, Mixcolumns of AES. The mixcolumns and key expansion consumes the majority of area and account for about 50% of total area. Power consumption was achieved based on high switching activity. Here encryption and decryption procedures are integrated together by the method of resource sharing.

Qiang Liu et al (2014) [5] proposed a design of advanced encryption standard (AES) on field programmable gate array (FPGA) for protection of high speed data. In terms of hardware implementation, a logical operation of AES leads to two efficient pipelining structures with regard to FPGA architectures. The two designs make a trade-off among speed, use of resources and power consumption. A new key expansion scheme was proposed to address the potential issues of existing AES. The proposed scheme increases the complexity of cracking keys by up $2(N - 1)$ times for N-round AES with additional nonlinear operations. The proposed design was implemented on various FPGA devices. The design approach with the analysis of combinational logic operations gains insight into the path delay. Based on the investigation of logic depth, with respect to different FPGA structures, pipelining structures were designed in proper manner, leads to balanced pipeline stages resulting with minimum logic delay.

3. PROPOSED WORK

3.1 Overview of mixcolumn operation

On the verge of the Advanced Encryption Standard (AES), the hardware designers thoroughly studied the AES to achieve the goal of reducing the area consumption and timing delay of the hardware implementation of this cryptosystem. The Mixcolumns transformation is the operation of mixing each column of the square array of the state matrix [6]. This operation uses Galois field (GF) modular multiplication of two byte. Galois field (GF) is the finite field of elements which performs binary operation like addition and multiplication. Multiplication in Galois field is much more difficult and harder to understand and multiplication is distributive over addition in GF (2^8). This transformation provides good diffusion property in the AES algorithm and has a very complex structure compared to Shift-Rows transformation in AES algorithm.



$$\begin{bmatrix} S'0,1 \\ S'1,1 \\ S'2,1 \\ S'3,1 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S0,1 \\ S1,1 \\ S2,1 \\ S3,1 \end{bmatrix}$$

Fig 1: mixcolumn operations in the state array of AES algorithm

Each column of the state matrix is represented as a vector with coefficients in GF (2^8). Thus, this vector is multiplied by 4 ×4 constant matrix N over GF (2^8).

$$N = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

The results of the mixcolumns operation are calculated using GF(2^8) operations. Each element of GF(2^8) can be represented as a polynomial of degree 7. Each term of the polynomial has the coefficients which can take the value 0 or 1. If there are 8 terms in an element of GF(2^8), then each bit in a 8 bit-string represents coefficient of the term in the polynomial.. The least significant bit (LSB) of 8 bit-string is represented as the constant of the polynomial, and starting from right to left, each term is represented by the bit B_i where B_i is i bits to the left of the LSB with coefficient of x^i .

For example, the bit string 10101011 represents $x^7+x^5+x^3+x+1$. If the corresponding coefficient is 1, a term x^i is found in the expression. If the coefficient is 0, the term is omitted from the expression. Galois field GF(2^8) addition of two elements is simply calculated using eight XOR gates to add corresponding bits. The multiplication of two elements in GF(2^8) is simulated with an AND gate. Multiplication in GF(2^8) can be calculated by first multiplying each term of the one polynomial with other polynomial. Each of these products should be XORed together for the final product.

For example:

$$\begin{array}{rr} & x^4+x+1 \\ \times & x^6+1 \\ \hline & x^4+x+1 \\ \oplus & x^{10}+x^7+x^6 \\ \hline & x^{10}+x^7+x^6+x^4+x+1 \\ \hline \end{array}$$

Since the degree of the new polynomial is greater than 7, then it can be reduced using irreducible constant polynomial. In the case of AES, the irreducible polynomial is given by $x^8+x^4+x^3+x+1$.This can be done by synthetic division method. Here the reducible polynomial will be the divisor. The polynomial that is to be reduced will be dividend. In the case of modular multiplication, addition of variables is done in the form of exclusive-OR function. The result will be taken from remainder of the synthetic division until the degree of the polynomial is reduced not greater than 7. From above example,

$$\begin{array}{r|l} & x^2 \\ \hline x^8+x^4+x^3+x+1 & x^{10}+x^7+x^6+x^4+x+1 \\ & x^{10}+x^6+x^5+x^3+x^2 \\ \hline & x^7+x^5+x^4+x^3+x^2+x+1 \\ \hline \end{array}$$

Thus the resultant product for the modular multiplication is given by $x^7+x^5+x^4+x^3+x^2+x+1$. Then in terms of bits, the product will be 10111111. The operation in mixcolumn involves multiplying a constant 4×4 matrix with column vector of GF(2^8) values. Given a 32-bit input word where each w_i is 8-bits, the mixcolumn operation is given as:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} w_3 \\ w_2 \\ w_1 \\ w_0 \end{bmatrix} = \begin{bmatrix} w'_3 \\ w'_2 \\ w'_1 \\ w'_0 \end{bmatrix} \qquad (1)$$

for inverse mixcolumn transformation

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} w'_3 \\ w'_2 \\ w'_1 \\ w'_0 \end{bmatrix} = \begin{bmatrix} w_3 \\ w_2 \\ w_1 \\ w_0 \end{bmatrix} \qquad (2)$$

where each element of the 4 × 4 matrices is a hexadecimal representation of an element in GF (2^8). For example, 0E represents x^3+x^2+x.

3.2 Cellular Automata Based Modular Multiplier

The designers always look for simple, modular, regular and cascadable logic circuit structure to realize a complex function. Here the cellular automata (CA) function is used to design the circuit. The basic transformations are built for additive CA rules having group properties, and are simple in nature. One of the important features of this enciphering method lies in its good measure of strength [7]. Cryptographic secure key stream generators needs three basic requirements are as follows.

1. The period of the key stream must be large.
2. The output bits should be easy to manipulate.
3. The output bits should be hard to predict.

Cellular automata (CA) is a phenomenon of 1-dimensional array of cells where each cell has definite permissible states.

At each time step (clock cycle), the cell value is updated based on some rule (the combinational logic) [7]. Finite field multipliers can be classified into fast bit parallel multipliers which generally requires a maximum area and low-complexity bit serial multiplier which requires minimum area [8]. The bit parallel multipliers have ability to give the result in one clock cycle, whereas bit serial multipliers need m clock cycles to compute the product in GF(2^m). There are some Cellular Automata based multipliers but none of them operates on the field of GF(2^m). A new VLSI array for modular multiplier based on PCA (Programmable Cellular Automata) was designed to overcome this limitations and the canonical (standard, polynomial) basis representation. According to the application environment, the field set of parameters can be changed

3.3 Algorithm for CA Based multiplier

The algorithm for polynomial multiplication in GF(2^m) and the basic theory for Programmable cellular automata has been applied in the modular multiplier such as follows [8]. Let A(x) and B(x) are the two elements in GF(2^m)

$$A(x)= a_{m-1}x^{m-1}+\ldots+a_1x+a_0 \qquad (3)$$

$$B(x)= b_{m-1}x^{m-1}+\ldots+b_1x+b_0 \qquad (4)$$

Then product of two elements A(x)B(x) mod P(x) is given by

$$C(x)= C_{m-1}x^{m-1}+\ldots+C_1x+C_0 \qquad (5)$$

Irreducible polynomial is

$$P(x)=x^m +p_{m-1}x^{m-1}+\ldots+p_1x+p_0 \qquad (6)$$

In general form, for jth cell

$$C_j = a_j b_j \oplus c_{j-1} \oplus c_{m-1} p_j \qquad (7)$$

where j is the position of cell.

a,b,c,p are coefficients of A(x) ,B(x), C(x) ,P(x) respectively.

The PCA was first introduced where the CL of each cell is not fixed but controlled based on control signals such that different functions (rules) can be realized on the same structure.

Table 1 PCA rules based on control signals

C_l	C_m	C_r	PCA rules
1	0	0	X_L
1	0	1	$X_L \oplus X_R$
1	1	0	$X_L \oplus X_S$
1	1	1	$X_L \oplus X_S \oplus X_R$

The combinations of the control signals of C_l, C_m, and C_r , and the corresponding rules are given in Table 1. PCA rules are defined such that X_S represents the value of a cell. X_R is the value of its right neighbor, and X_L is the value of its left neighbor. From Table 1, it can be seen that by allowing control signals in CL, one can ensure immense flexibility into this programmable structure.

3.4 Mixcolumn Transformation by CA Based Multiplier

The results of the mixcolumns operation are calculated using cellular automata based modular multiplication. Each column of the state matrix is assumed as a vector with coefficients in GF(2^8). Thus, this vector was multiplied by 4 ×4 constant matrix N over GF(2^8).

Each element of GF(2^8) is a polynomial of degree 7 with coefficients in GF(2). A(x) is taken from state matrix and B(x) is taken from constant matrix. For example,

$$N = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

$$A(x)=x^7+x^3+x^2+x \qquad B(x)=x$$

Irreducible polynomial $P(x)=x^8+x^4+x^3+x+1$. Obtaining coefficients from the above polynomials and manual computation of CA based modular multiplier is done as shown in fig 2.

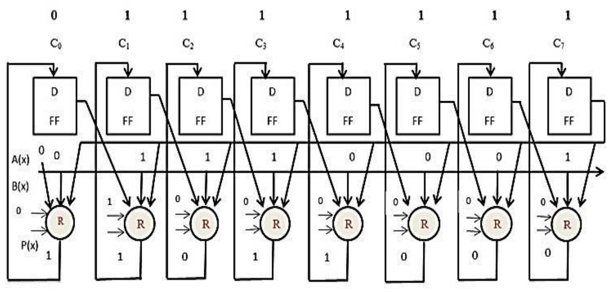


Fig 2: Manual computation of CA based modular multiplication

From the above figure 2, the resultant product obtained is 11111110. Similar process is done for every modular multiplication of two vectors. Modular addition was done with exclusive OR function taken from the four different results of modular multiplication. Similar way of approach has been done for every modular multiplication using cellular automata function. Finally, every columns of the state matrix of AES was changed, reverse process was done in the same manner but the reverse process uses inverse of the constant matrix.

3.4.1 Architecture of CA based mixcolumn multiplier

The proposed architecture consists of D Flip Flop, combinational logic, control signals, wires. D Flip Flop acts as the registers. Architecture of CA based multiplier is shown in fig 3. With the help of registers, the output of modular multiplication has been taken. The combinational logic is used in which specified rule function has been applied to the Mix columns operation [8]. The control signals are used to control the rule function. On comparison with the conventional modular multiplier, the proposed architecture reduces the chip area, gate equivalents and consumes less power than conventional mix column.

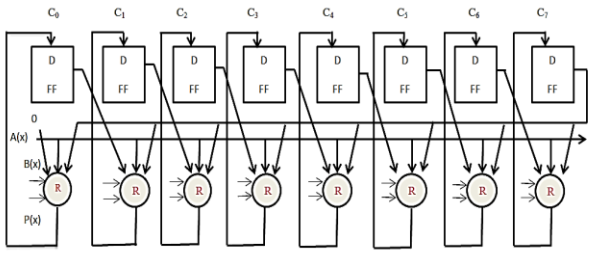


Fig 3: Architecture of CA based modular multiplier

4. SECURITY ANALYSIS

4.1 Nonlinearity

The nonlinearity (NL_f) of a function f is the minimum Hamming distance between function (F) and all affine functions. An example where the function (F) is tested against all affine functions for n=4 is given in Table 2. This function's nonlinearity is six. The proposed CA Function is

$$C_j = a_j b_j \oplus c_{j-1} \oplus c_{m-1} p_j$$

The above CA Function is reformed to

$$F = ab \oplus c \oplus cp \qquad (8)$$

so it will be convenient for bent function analysis.

Table 2: Hamming distance for CA Function (F) with respect to affine function

AFFINE FUNCTION	DISTANCE
0	6
x_1	6
x_2	6
x_3	6
x_4	10
$x_1 \oplus x_2$	10
$x_1 \oplus x_3$	6
$x_1 \oplus x_4$	10
$x_2 \oplus x_3$	6
$x_2 \oplus x_4$	10
$x_3 \oplus x_4$	6
$x_1 \oplus x_2 \oplus x_3$	8
$x_1 \oplus x_2 \oplus x_4$	8
$x_1 \oplus x_3 \oplus x_4$	6
$x_2 \oplus x_3 \oplus x_4$	6
$x_1 \oplus x_2 \oplus x_3 \oplus x_4$	10

4.2 Fast Walsh Transform

The Fast Walsh Transform (FWT) is an efficient method for computing the spectral information of Boolean function. The WHT has a computational complexity of n^2 [9]. The FWT, on the other hand, has a computational complexity of n log (n). This is a significant reduction in the amount of required computations.

4.2.1 Computation

The FWT is a relatively simple computation. Given a valid truth table (TT), pairs of digits from the output of TT are coupled and modified by an "in-place butterfly" module. Here, the term "in-place" means that the values produced by the butterfly module output are placed in the same position from where the butterfly module inputs previously positioned [9]. For inputs a and b, the outputs of the butterfly module will be $a+b$ and $a-b$, respectively as shown in Fig 4.

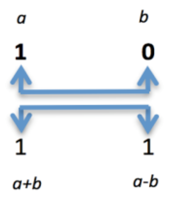


Fig 4: Butterfly module for two variables

The first set of butterfly modules pairs adjacent elements and produces a 2^n element array. This process is repeated a second time, pairing every other element in the first array to produce a second array. The third iteration will pair every fourth element in the second array, and so on. A complete computation of the FWT of a TT with n =4 is shown in fig 5.

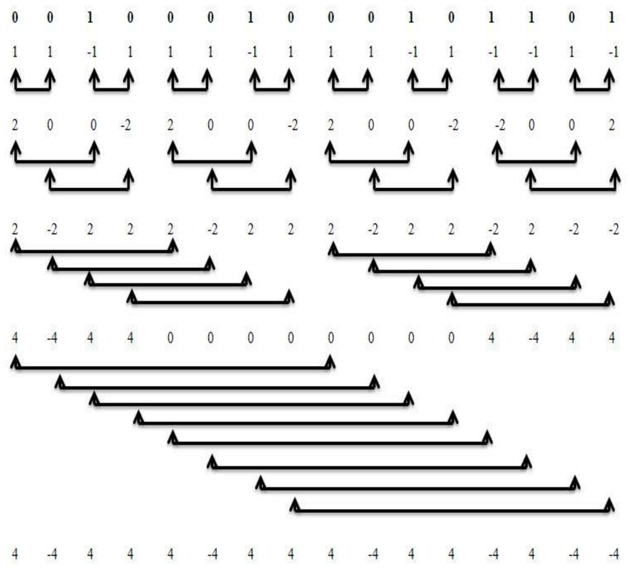


Fig 5: Butterfly method for 16 bit to check nonlinearity

An interesting and important observation is the value of the first element of the FWT, which will be referred to as FWT_0. The value of FWT_0 is equal to the weight of the input TT, which indicates the number of 1's contained in the input TT. This is always true, since the first element of the iterations of the FWT computation always receives the left portion of the butterfly (a +b). Therefore, its output is the sum of all bits in the TT and FWT_0 has a range of values from zero to 2^n.

The other elements of the FWT also have a range that is dependent on n. As n increases, computation of the FWT requires more iteration. Each iteration produces another array and expands the range of each element in the array. For example, the TT elements only have range from 0 to 1. The first array of the FWT computation will have a maximum value of 2 and a minimum value of -1. The second array will have a maximum value of 4 and a minimum value of -3. The third array will have a maximum value of 8 and a minimum value of -5. The fourth array will have a maximum value of 16 and a minimum value of -7. The hamming distance is given by [10]

$$W_H(f) = 2^{n-1} - \frac{\hat{f}_x(0)}{2} \qquad (6.4)$$

Therefore, 6 is the minimum hamming distance. Thus given function has good non linearity.

5. IMPLEMENTATION AND RESULTS

On the verge of designing the CA based mix column operation, the proposed CA based modular multiplication should be compared with the conventional multipliers that are implemented in FPGA. The target device used here is Spartan 3E and concentrate more on analyzed total number of 4 input LUTs and number of slices consumed because the main objective is area consumption in the device. The following are device utilization summary of various multipliers and Mix column operation.

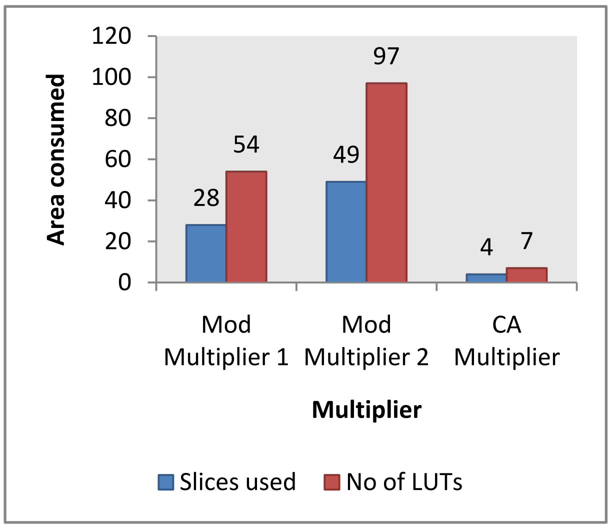


Fig 6: Device Utilization summary for 8-bit multiplier

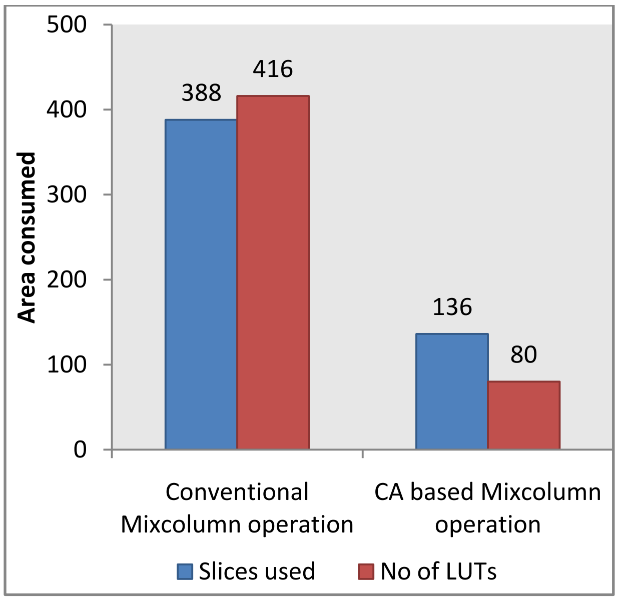


Fig 7: Device Utilization summary for 128-bit mixcolumn operation

In conventional Galois field modular multiplication model 1, the design consists of various AND gates and XOR gates in which it is implemented in mixcolumn operation of AES algorithm. From the Fig 6, the total number of 4 input LUTs used is 54 which is 2% utilization of the available device. Number of occupied slices used is 28 slices which is 2% utilization of the available device.

In conventional modular multiplier of model 2 , the design has less number of AND gates and XOR gates compared to model 1, From the Fig 6, total number of 4 input LUTs used is about 97 which is 5% utilization of the available device. Number of occupied slices used is 49 slices which is 5% utilization of the available device.

In the case of proposed CA based modular multiplier, total number of 4 input LUTs used is 7 which is 1% utilization of the available device. In terms of comparison with other models of modular multiplier, CA based modular multiplier consumes 54.71% less than conventional multiplier model 1 and it also consumes 74.2% less than conventional multiplier model 2. The number of occupied slices in CA based multiplier is 4 slices which is 1% utilization of the available device. In terms of comparison with other models of modular multiplier, CA based modular multiplier consumes 85.71% Less than conventional multiplier 1 and it also consumes 91.83% Less than conventional multiplier 2.

In conventional modular multiplication in Mix column transformation, AES uses modular multiplier model 1. This design is based on complete implementation of conventional modular multiplication for 128-bit data in AES algorithm. From the fig 7, total Number of 4 input LUTs used is 416 which is 21% utilization of the available device. The number of occupied slices used is 388 slices which is 40% utilization of the available device.

In CA based modular multiplication in Mix column transformation, the design is based on complete implementation of cellular automata based modular multiplier for 128-bit input data in AES algorithm. From the fig 7, total number of 4 input LUTs used is 80 which is 4% utilization of the available device. In terms of comparison with conventional mix column operation, CA based mix column operation consumes 80.7% less than conventional one. The number of occupied slices used is about 136 slices which is 14% utilization of the available device. In terms of comparison with conventional mix column operation, CA based mix column operation consumes 64.9% less than conventional one.

6. CONCLUSION AND FUTURE WORK

Security is a major issue in the field of communication systems used to avoid the intruders to steal the information transferred between two parties or in a specific network. Advanced encryption standard is the industrially used security algorithm. The major factors influencing hardware implementation of AES are power consumption, throughput, chip area consumption. Thus, the proposed cellular automata based modular multiplication for AES offers optimized chip area comparing to conventional modular multiplier. On computation of CA based multiplier, security aspects should not be compromised in AES Algorithm. So, security analysis of CA function is carried out to show that CA function has good nonlinearity.

The future work is to implement CA based multiplier of Mix column operation in the complete AES algorithm. Hardware implementation of AES algorithm will be carried out on the FPGA boards through Xilinx software.

7 REFERENCES

[1] Othman 0. Khalifa et al, “Communications cryptography”, presented at the RF and microwave conference, Subang, Selangor, Malaysia, October 5 – 6, 2004.

[2] Deng Tang et al, “Construction of balanced Boolean functions with high nonlinearity and good autocorrelation properties”, in Journal of design, codes and cryptography, china, April 2013, pp. 77-91.

[3] N. Ahmad and S.M. Rezaul Hasan, “Efficient integrated AES crypto-processor architecture for 8-bit stream

cipher", in Electronics letters 2012, ©The Institution of Engineering and Technology 2012. doi: 10.1049/el.2012.2932.

[4] LI Zhen-rong et al, "Low-power and area-optimized VLSI implementation of AES coprocessor for Zigbee system", in The Journal of China Universities of Posts and Telecommunications, Xi'an, China, 2009, pp.89-94.

[5] Qiang Liu et al, "High throughput and secure advanced encryption standard on field programmable gate array with fine pipelining and enhanced key expansion", in IET Computers & Digital Techniques, Tianjin, china, October 2014, pp.175-184.

[6] Hua Li and Zac Friggstad, "An Efficient Architecture for the AES Mixcolumns Operation", in Circuits and systems, IEEE international symposium, Canada, 2005, pp. 4637 – 4640.

[7] S. Nandi et al, "Theory and Applications of Cellular Automata in Cryptography", in IEEE Transactions on computers, 1994 © IEEE. doi: 0018-9340/94.

[8] Hua Li and Chang N. Zhang, "Efficient Cellular Automata Based Versatile Multiplier for GF(2m)", in Journal of information science and Engineering , Canada, 2002, pp. 479-488.

[9] Timothy R. O'Dowd," Discovery of Bent Functions Using the Fast Walsh Transform", M.S. Thesis, department of electrical engineering, Naval Postgraduate School, Monterey, California, December 2010.

[10] Claude Carlet,"Boolean Functions for Cryptography and Error Correcting Codes", Cambridge University Press, Eds Yves Crama and Peter Hammer, October 2010, ch 2, pp.17-20.