International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE) Vol 3, Issue 5, May 2016

Investigation of Logic Level Techniques to Improve AES Throughput

[1]Megha S Hallikeri, [2]Raghuram Srinivasan
[1] Student, [2] Associate Professor,
Dept. of E&C, M. S. Ramaiah Institute of Technology, Bengaluru
[1]meghahallikeri0@gmail.com,[2] raghuram@msrit.edu

***Abstract*— Substitution Box(S-Box) is an important integral part of modern cryptographic cipher techniques like Advanced Encryption Standard (AES). There exists a bulk literature devoted to the implementation of AES. Combinational logic of implementation attains high throughput in terms of parameters like speed, area, delay etc. This paper represents the working principle of AES, Novel algorithmic approaches for S-Box. We mainly concentrated on combinational logic implementation of S-Box in order to Improve Area. AES is programmed and executed on Xilinx 14.7 version, Spartan 3 Family, ModelSim Simulator. Cadence 180nm tool is used to verify the parameters. We examine the current work which exploits the mathematical properties of S-Box. Using this technique throughput is increased by 30%. We verified individual modules separately and waveforms are obtained.**

***Index Terms*—Advanced Encryption Standard (AES),Substitution Box(S-Box)**

I. INTRODUCTION

Cryptography plays a vital role in the network security; security issues have become prominent as technology is developed. The National Institute of Standards and Technology(NIST) declared Rijndael Block cipher method as AES algorithm in 2001 [1]. AES performs based on symmetric key algorithm where in encryption and decryption both uses same key. The length of block used for processing is 128 bit, length of key can be varied to multiples of 32 i.e 128,192 or 256bits. From 128-bit key, algorithm generates 10 keys of 128-bit each, which are placed into 4X4 arrays. Simultaneously plain text is divided into 4X4 arrays each are of 128-bits. Each of 126-bit plain text block is processed in 10 rounds (number of rounds varies with key size). After 10^{th} round new code is generated. Every individual byte is substituted in an S-Box and replaced by reciprocal on Galois Field (GF). AES algorithm undergoes four transformations namely AddRound key, substitution (S-Box), Mix columns, Shift rows [2].

S-Box is the main and costliest block in AES, several different methods for implementation have been proposed in literature. S-Box is a non-linear substitution step where in individual byte is replaced with another byte according to Lookup table [2]. We compared our results with different algorithms and standard lookup table where lookup table (LUT) results with_736 cell numbers, area 2772 square meters._Combinational logic is prominent and emerging technology where implementation of S-Box is achieved by operating on Galois Fields (GF), Computation of S-Box using GF results better than direct implementation or using LUT.

The sections of this paper are analyzed as follows. Section II briefs the basic understanding and functionality of transformations in GF while calculating S-Box, section III elaborate the proposed architecture of S-Box using combinational logic, Section IV provides the obtained results and compares with Base paper.

II. AES WORKING PRINCIPLE

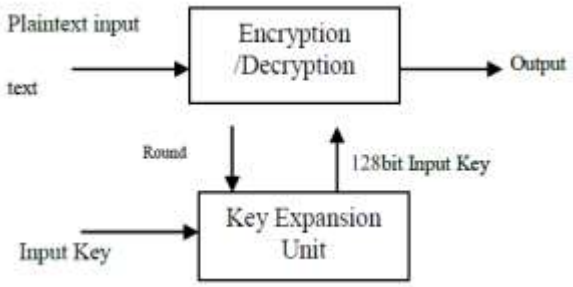

Fig.1 AES basic block diagram

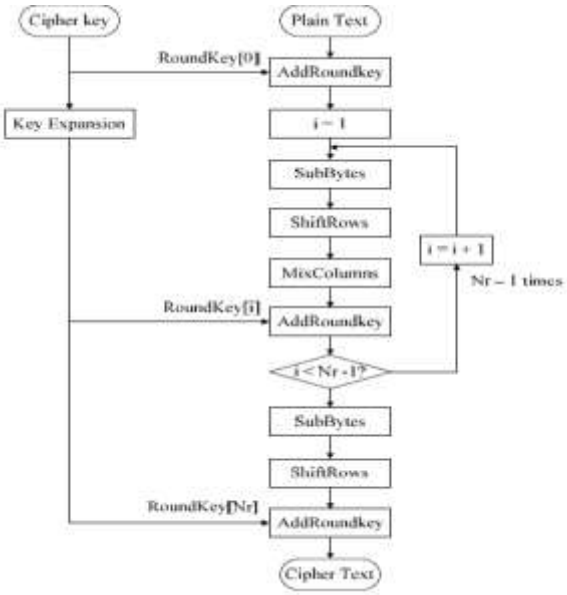

Fig.2 AES flowchart

As shown in above figure AES mainly perform on input which can be text, image or video. And encryption and decryption are done as shown Fig.1. Detailed procedure of AES is shown in Fig.2.

a. Add Round Key

Add Round Key is the initial step, where the sub key is combined and performed with state. Sub key is obtained from main key using Rijndael's key. Size of key and state are kept same. Su bkey is combined with each byte of state with corresponding to bitwise Ex-or[1].

b. Shift Rows

This is the module of AES operates on rows. It operates by shifting each row cyclically, the first row is left unchanged. For shifting certain rows offset is set. 2^{nd}, 3rd rows are shifted by one left shift, two shifts. Shifting rows remains same for different block sizes 128 bits, 192 bits and 256 bits. Row m is shifted left circularly by m-1 bytes[2].

c. Mix Column

Mix Column step is one of the complicated step in AES, four bytes of each column are combined using linear transformation. The function takes four bytes as input. During operation, each column is transformed using matrix. The matrix retains in all AES operation.

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

We are mainly dealing with S-Box and detailed explanation is given in preceding sections.

III. PRELIMINARIES FOR GF TRANSFORMATION

a. Isomorphic Mapping and Inverse Isomorphic Mapping of GF:

In order to attain transformation of higher order Galois Fields into lower order, irreducible polynomials are used as given in equation (1)

$GF(2^4) \longrightarrow GF(2^8)$; $I2(X) = x^2+x+\lambda$;
$GF(2^2) \longrightarrow GF(2^4)$; $I1(X) = x^2+x+\varphi$;
$GF(2) \longrightarrow GF(2^2)$; $I0(X) = x^2+x+1$;

Where $\varphi=\{10\}$ $\lambda=\{1100\}$ (1)

The implementation of S-Box involves 8-bit multiplicative
Inverse module followed by Affine Transform (AT). The inverse SubByte can be generated using Inverse AT. Where both SubByte and Inverse SubByte can be generated together. The bytes are represented in $GF(2^8)$ are viewed in terms of polynomial forms. All the operations in $GF(2^8)$ are obtained by taking Modulo-2 operations using corresponding fixed irreducible polynomial p(x).

$$p(x) = x^8 + x^4 + x^3 + x + 1 \quad (2)$$

Inversion in $GF(2^8)$ has many sublevels. Multiplicative inverse can be calculated by reducing complex $GF(2^8)$ into lower order fields like $GF(2^4)$ and $GF(2^2)$. Multiplicative inverse can be given as in equation (3).

$$(bx+c)^{-1} = b(b^2B+bcA+c^2)^{-1}x+(c+bA)(b^2B+bcA+c^2)^{-1} \quad (3)$$

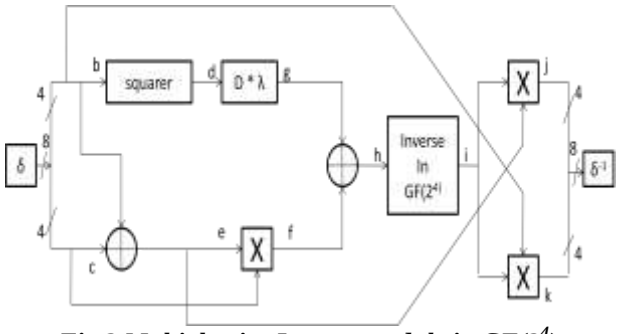

Fig.3 Multiplcative Inverse module in GF(2^4)

The legends of block diagram are as follows

δ	Isomorphic mapping to composite fields in GF
Squarer	Squaring in GF(2^4)
	Multiplication inversion
d* λ	Multiplication with constant λ in GF(2^4)
⊕	Addition operation in GF(2^4)
X	Multiplication operation in GF(2^4)
δ^{-1}	Inverse isomorphic mapping to GF(2^8)

Multiplicative Inverse (MI) in GF

Calculation for the multiplicative inverse in composite fields cannot be directly applied to byte which is based on GF(2^8). That Byte or element has to be mapped to its Composite field representation by applying to an isomorphic function, δ. Followed by performing the MI, the result of MI also have to be mapped back from its composite field to its equivalent in GF(2^8) via the inverse isomorphic function, δ^{-1}. Where δ and δ^{-1} are represented in an 8x8 matrix. Let *q* be the element in GF(2^8), then the isomorphic mapping and its inverse isomorphic mapping can be written as δ**q* and δ-1**q*, which is a obtained by matrix multiplication as shown below, where *q*7 indicates most significant bit(MSB) and *q0* least significant bit(LSB).

$$\delta = \begin{bmatrix} 10100000 \\ 11011110 \\ 10101100 \\ 10101110 \\ 11000110 \\ 10011110 \\ 01010010 \\ 01000011 \end{bmatrix} \quad \delta^{-1} = \begin{bmatrix} 11100010 \\ 11011110 \\ 01100010 \\ 01110110 \\ 00111110 \\ 10011110 \\ 00110000 \\ 01110101 \end{bmatrix} \tag{4}$$

Affine Transformation

The Multiplicative Inverse output has been computed, and affine transformation is carried out on the resulted polynomial[3]. The affine transformation is a simple matrix multiplication and XOR with a constant column matrix. Affine transformation over a finite field i.e. GF (2^8) can be given as below equation,

$$\alpha = \begin{bmatrix} 11111000 \\ 01111100 \\ 00111110 \\ 00011111 \\ 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \end{bmatrix} \times \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{5}$$

IV. PROPOSED WORK

Squarer

Equations for squaring a element in GF(2^4) is taken from [1], where in converting the GF(2^4) elements into GF(2^2) by using irreducible polynomial. Hence the Boolean expression can be given as equation (6)

$$\begin{aligned} d_3 &= b_3; \\ d_2 &= b_3 \oplus b_1; \\ d_1 &= b_2 \oplus b_1 \\ d_0 &= b_3 \oplus b_1 \oplus b_0 \end{aligned} \tag{6}$$

Multiplication with constant (λ)

Modulo reduction can be performed and simplified by substituting $x^2 = x + \varphi$ using the irreducible Polynomial. The equations for multiplication with constant are derived in [4]. The polynomial 'd' is result of squarer of polynomial 'b', is multiplied with the constant $\lambda = \{1\ 1\ 0\ 0\}$. This operation can be further simplified and 'g' could be achieved as simple Boolean expressions, given in equation (7).

$$\begin{aligned} g_3 &= d_2 \oplus d_0 \\ g_2 &= d_3 \oplus d_2 \oplus d_1 \oplus d_0 \\ g_1 &= d_3 \\ g_0 &= d_2 \end{aligned} \tag{7}$$

Multiplication in GF(2^4)

Multiplication in GF(2^4) is the major block in Combinational logic architecture, which consumes more area and complicated to analyze. Multiplication of polynomial in GF is performed as given in fig 4. Which include three major blocks of multiplication. These multiplication blocks carried out in GF(2^2), the equivalent design is given in fig 3. Equations derived for multiplication block in [1], the simplified equation for multiplication in GF(2^2) can be given as below equation(8)

$$\begin{aligned} z(1) &= x(1)y(0) \oplus x(0)y(1) \oplus x(1)y(1) \\ z(0) &= x(0)y(0) \oplus x(1)y(1) \end{aligned} \tag{8}$$

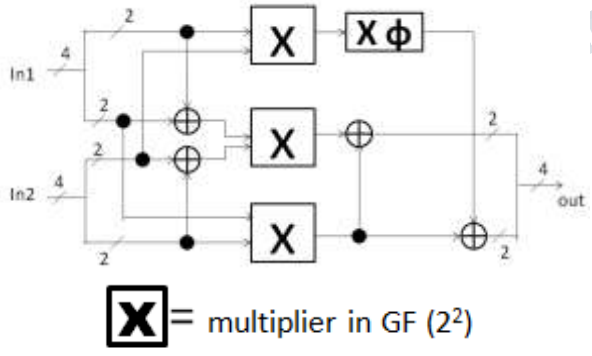

Fig.4 Multiplier In Gf(2^4) Embedded With Multipliers In Gf(2^2)

According to literature this block is directly implemented as shown in Fig.3, and the above equations are used. Using Ex-or gates leads to increase in area and delay. To overcome this hurdle we tried to simplify the equation and proposed a new structural design as shown in Fig.5

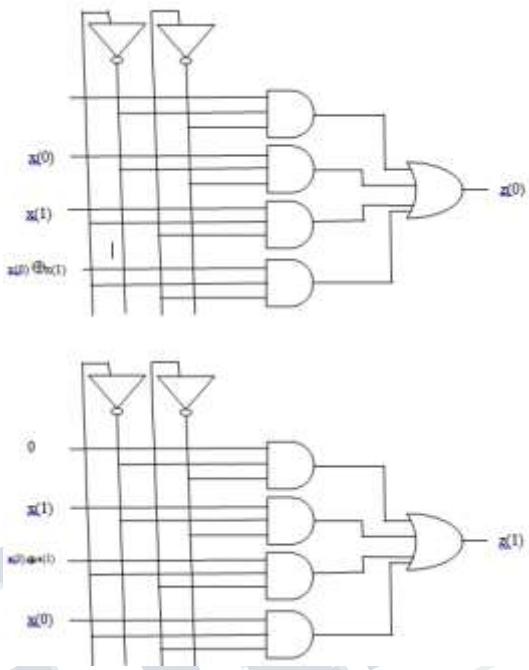

Fig.5 Proposed Architecture for GF (2^2) Multiplier

Where Ex-or gate is replaced by the normal and, or gates. Which has given overall optimized result. This proposed Multiplication design of GF(2^2) is used in Fig.4 The results for each block and module are plotted in next section.

Multiplicative Inverse:

Multiplicative inverse is sub module of combinational logic, can be implemented directly using the standard table as given below. The equations are obtained from [1]

$$\begin{aligned} q^{-3} &= q_3 \oplus q_2 \oplus q_1 \oplus q_3q_2 \oplus q_3q_0 \\ q^{-2} &= q_2 \oplus q_2q_1 \oplus q_3q_0 \oplus q_3q_2q_1 \oplus q_3q_2q_0 \\ q^{-1} &= q_3 \oplus q_1 \oplus q_2 \oplus q_2q_0 \oplus q_3q_1q_0 \oplus q_3q_2q_1 \\ q^{-0} &= q_0 \oplus q_1 \oplus q_2 \oplus q_2q_1 \oplus q_3q_1 \oplus q_3q_0 \oplus q_3q_2q_0 \oplus q_3q_2q_1 \end{aligned}$$

(9)

V. RESULTS

We worked on many methods to achieve the standard table for S-Box, and calculated varies parameters for each methods. Combinational logic method is the efficient method which attained less area. We tried to optimize each sub module of whole architecture shown in fig.1. Could

achieve efficient result in inverse delta module calculation, multiplication in GF(2^2) which in turn enhanced the result of multiplication in GF(2^4). The result of multiplicative inverse module is performed on Affine Transform to obtain Substitution box, and performed on Inverse Affine Transform to get inverse substitution box.

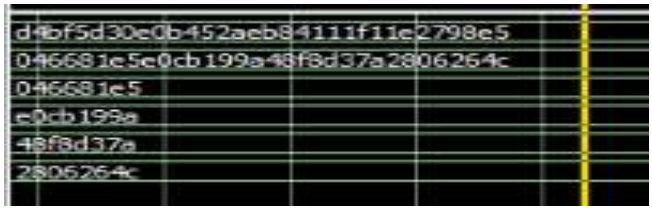

Fig.6 MixColumn output wavefroms

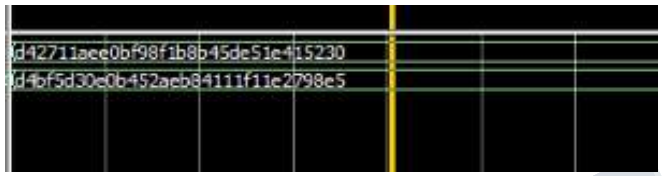

Fig.7 ShiftRows output

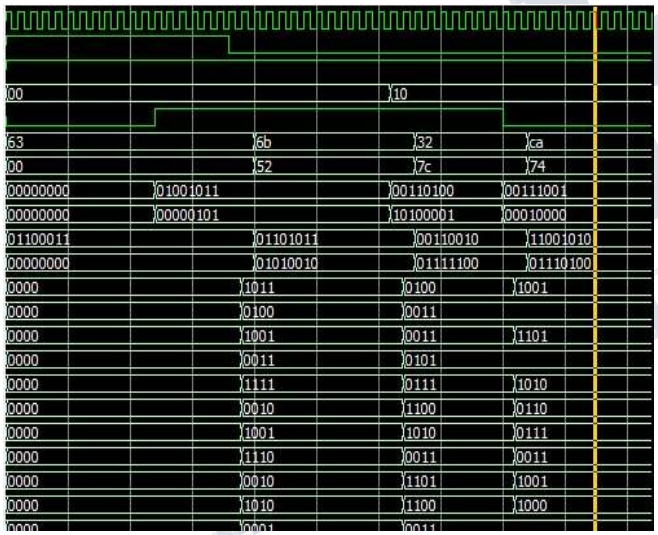

Fig.8 S-Box Output Waveforms In Xilinx

Fig.9 S-Box Output Waveforms Of Lut

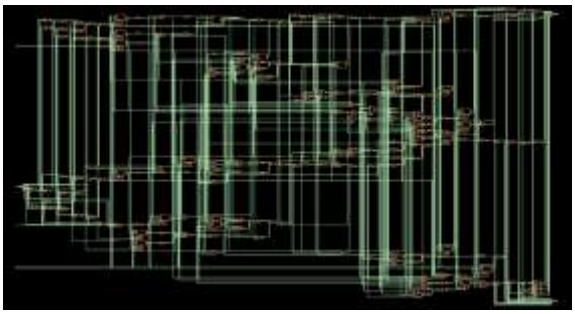

Fig.10 Synthesized Output Of S-Box In Cadence

Architecture	Number of Cells	Area (μm^2)	Timing(ps)	Total Power(nW)
LUT	736	2772	3265	36220
S-Box in GF(16)	227	1255	6467	23694
Using Multiplcative Inverse	485	1821	2114	27486
Using inverse module	167	842	4375	38773
[1]	178	3988.35	-----	61112
Our design	152	785	4068	35667

Table 1: Comparison Of Different algorithms

We considered different algorithms in our work. Where lut method is popular, in which the standard calculated table of s-box is implemented directly. Which consumes large hardware and area. S-box in gf(16) is one of architecture we worked on, where gf(256) is converted to gf(16) and multiplicative inverse is found and processed to affine transform and s-box is obtained. By calculating all possible combination of multiplicative inverse and inverse module, results are tabulating and used in programming. Where in our proposed design implemented by using polynomials in gf(2^8),gf(2^4) and gf(2^2) as explained in above sections.

VI. CONCLUSION

The Results of Our Design Is Verified And tabulated in table.2. Where cell number and area are reduced BY 579, 1987μ^2. And comparing with base paper [1] cell number and area is improved by 30%. the final results obtained are plotted in fig.7. Cadence 180nm technology is used to calculate the parameters. This work can be performed on asic[1].

From table.1 we conclude that combinational logic implementation of s-box gives more efficient and compact platform.

REFERENCES

[1] P.V.S.ShastIy, Anuja Agnihotri, Divya Kachhwaha, Jayasmita Singh, Dr.M.S.Sutaone " A Combinational Logic Implementation of S-box of AES" 978-1-61284-857-0/11/$26.00 @2011 IEEE

[2] Xinmiao Zhang, Keshab K. Parhi "Implementation Approaches for the Advanced EncryptionStandard Algorithm" 1531-636X/12/$10.00©2002IEEE

[3] J. Daeme, V.Rijmen " AES proposal: Rijndael. NIST AES Proposal" April 2003

[4] Bahram Rashidi, Bahman Rashidi " Implementation of An Optimized and Pipelined Combinational Logic Rijndael S-Box on FPGA" I. J. Computer Network and Information Security, 2013, 1, 41-48 Published Online January 2013 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijcnis.2013.01.05

[5] Saleh Abdel-hafeez, Ahmed Sawalmeh, Sameer Bataineh "HIGH PERFORMANCE AES DESIGN USING PIPELINING STRUCTURE OVER GF($(2^4)^2$)" 2007 IEEE International Conference on Signal Processing and Communications (ICSPC 2007), 24-27 November 2007, Dubai, United Arab Emirates