ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 5, Issue 5, May 2018

Implementation of Area and Memory Efficient Combined ByteSub and InvByteSub Transformation for AES Algorithm

[1] Sushma D K, [2] Dr. Manju Devi
[1] Dept. of ECE, TOCE, Bangalore
[2] Professor & Head, Dept. of ECE, TOCE, Bangalore

Abstract- **Efficient implementation of combined ByteSub and InvByteSub transformation for encryption and decryption in advanced encryption standard (AES) architecture using the composite field arithmetic in finite fields GF (256) or GF (28) hence this approach is more advantages than the conventional LUT method that incurs the unbreakable delay, greater amount of memory and area. The proposed architecture which is combined implementing of S-box and InvS-box makes use of an enable pin to perform encryption and decryption in AES. The architecture uses combinational logic, as both S-box and InvS-box are implemented on same hardware reduces the area and gate count by the large amount. Low power consumption due to resource sharing by the multiplicative inverse module of the proposed system. The proposed architecture is accouterment on Spatan6 board using Verilog HDL in Xilinx ISE 14.6.**

***Index Terms*— Composite field arithmetic, AES, Galois field, look-up table, FPGA.**

I. INTRODUCTION

Cryptographic development in recent years has been a challenging and high priority research area in both fields of mathematics and engineering. Due to advancement in embedded system and need of encryption in it has made encryption more resource constraint in terms of power, area and delay. Advanced Encryption Standard (AES) is adopted as the standard for encryption and decryption by National Institute of Standards. AES makes use of larger key sizes (128, 192 and 256bits) to provide strong security to digital data through encryption technique. Encryption algorithms have two types first one is Private key or Symmetric Key and the other is public key. Private Key algorithms uses only one key, for both encryption and decryption whereas, public key algorithms involve two different keys, for encryption and decryption [1]. Symmetric key cryptography is one of the main subjects in cryptography where a key of a certain size will be shared for the encryptor and decryptor processes. The AES algorithm as applications in different fields like banking World Wide Web servers, digital video recorders Automated Teller Machines (ATMs), smart cards, cellular phones and sensor nodes. The four important operations in AES algorithm as four transformations they are S-Box & InvS-Box, MixColumn and InvMixColumn have more priority than the addroundkey and shift row operations.

ByteSub and InvByteSub transformation are non-linear that encounters each byte of the state that is 128 bits to different values by making use of the substitution table for S-box and InvS-box. It can be implemented by using memory method and memory-less method. In the memory method, ROM based LUT (Look-up table) is used to compute the S-box that utilizes more memory, which increases area and power of AES and thus the disadvantage of this is unbreakable delay and low latency. In memory-less method, implementation of S-Box using LUT and SOP approach is fast but effective in cost. The paper is organized as following. Proposed architecture is briefed out in section II. The Composite field arithmetic processes gives details in section III. Hardware design implementation and results is defined in section IV.

II. PROPOSED ARCHITECTURE

The ByteSub & InvByteSub transformation are calculated by the use of multiplicative inverse to the plain text in GF(28) and then the affine transformation is applied to it. For decryption, the InvByteSub transformation is calculated by the use of inverse affine transformation to the cipher text before applying the multiplicative inverse [6]. The multiplicative inverse operation is involved in both the ByteSub and in its inverse transformations too.

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 5, Issue 5, May 2018

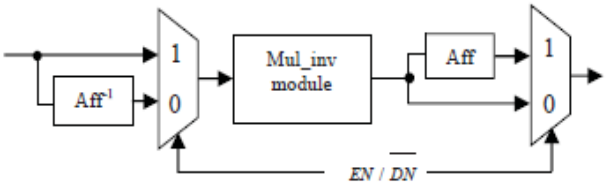

Fig 1: Combined ByteSub and invByteSub transformation

Here 'Aff' block represents affine transform, 'Aff-1' represents inverse affine transform, the EN/DN will act as selection line of s-box and InvS-box, and 'Mul_inv' block represents multiplication inverse in GF(28) .Implementing the architecture of S-Box (and its inverse) using combinational logic has an advantage of small area occupancy and on using pipelined structure and also increases the clock frequency.

A. Affine and inverse affine transform:

The Affine and Affine-1 are the Affine Transformation and its inverse while the vector is the multiplicative inverse of the input byte from the state array. From here, it is observed that both the SubByte and the InvSubByte transformation involve a multiplicative inversion operation. Thus, both transformations actually share the same multiplicative inversion module in a combined architecture. Switching between SubByte and InvSubByte is just a matter of changing the value of EN/DN. EN is 0 for SubByte and 1 is set for InvSubByte operation as desired.

For SubBytes, the multiplicative inverse operates on each byte of the State is followed by an affine transformation. Thus SubBytes can be designated by (1)

$$S_{i,j}^{1}=M \cdot S_{i,j}^{-1} + C \quad (1)$$

Where Si,j (0<i,j<4) is deliberated as an element of GF(28) M is 8x8 binary matrix and c is a 8bit binary vector with only 4 nonzero bits. The transformations in the decryption process performs the inverse of the resulting transformations in the encryption process. Specifically, the InvSubByte performs the subsequent operations on each byte of the State by (2)

$$S'_{i,j} = (M^{-1}(S_{i,j} + C))^{-1} \quad (2)$$

Where S and S' are input and output bytes in 8-D vector formats.

Multiplicative inverse module:

This multiplicative inverse module is a complex operation, such that it is divided which is the major operation in both the ByteSub and in inverse ByteSub transformation. It takes more than 630 gates to implement it with repetitive multiplications in GF (28). So, to reduce the gate count in large amount, composite field arithmetic is used.

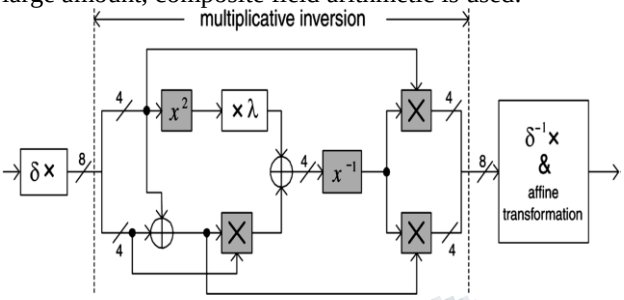

Fig 2: Multiplicative inverse module for AES algorithm

Isomorphic mapping function and its inverse

Composite field is symbolized as GF((2n)m) , that is Isomorphic to the finite field GF(2k), for k = nm. The composite field GF(28) can be made iteratively from minor order fields like GF(2) by making use of irreducible polynomials that are stated in (3):

$$\begin{aligned} &GF(2) \Rightarrow GF(2^2): && P_0(x)=x^2+x+1 \\ &GF(2^2) \Rightarrow GF((2^2)^2): && P_1(x)=x^2+x+\varphi \\ &GF((2^2)^2) \Rightarrow GF(((2^2)^2)^2): && P_2(x)=x^2+x+\lambda \end{aligned} \quad (3)$$

Where φ = {10}2 & δ = {1100}2. To represent an element of finite field GF(28) in its composite field, an isomorphic mapping function is used and after applying the multiplicative inverse for output of isomorphic function, again to convert the result into finite field GF(28), an inverse isomorphic mapping function is used. The 8 × 8 binary matrices of isomorphic (δ) and its inverse (δ-1) functions can be decided by the irreducible Polynomial p(x) = x8 + x4 + x3 + x +1 of the finite field GF (28) and by the irreducible polynomials of its composite fields which are mentioned in (3). Let 'a' be an element (can represent in column matrix of order 8×1) in GF (28), then the isomorphic mapping can be written as a matrix multiplication, δ×a and its inverse as another matrix multiplication δ-1×a, as shown in (4) and (5):

$$\delta \times a = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \quad (4)$$

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 5, Issue 5, May 2018

$$\delta^{-1}\times a = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \quad (5)$$

The isomorphic mapping can be appliance easily by using verilog code and this matrix multiplication can be done using XOR operations.

ii. Multiplicative inversion in GF(28):

In the composite field GF(28) , an element can be expressed as bx + c, where b, c in GF(24) are first and second nibbles of the byte and x is a root of irreducible polynomial P2(x) in (3). The multiplicative inverse of bx + c modulo P2(x) can be computed by using Extended Euclidean algorithm [2] [5] as shown in (6).

$$(bx+c)^{-1} = b\left(b^2\lambda + c(b+c)\right)^{-1}x + (c+b)\left(b^2\lambda + c(b+c)\right)^{-1} \quad (6)$$

From the above equation implies that there are multiply, addition, squaring and multiplication inversion in GF(24) operations in Galois Field. Each of these operators can be transformed into individual blocks when constructing the circuit for computing the multiplicative inverse. From above simplified equation, the multiplicative inverse circuit GF(28) can be obtained.

III. COMPOSITE FIELD ARITHMETIC OPERATIONS

Any arbitrary polynomial can be represented by bx + c where b is upper half term and c is the lower half term. Therefore, from here, a binary number in Galois Field q can be spilt to qH x + qL for instance, if q = {1011}2, it can be represented as {1 0}2x + {1 1}2, where qH is {1 0}2 and qL is {1 1}2. The decomposing is done by making use of the irreducible polynomials introduced at (3). Using this idea, the logical equations for the addition, squaring, multiplication and inversion can be derived.

A. Addition in GF(24):

Addition of two elements in Galois Field is translated to simple bitwise XOR operation between the two elements.

B. Squaring in GF(24):

Let 'q' is an element in GF(24) which can written as qHx + qL and this can be split, let 'k' is another element in GF(24) which is equal to square of q as given in equation (7).

$$kH\ x + kL = (qHx+qL)2 = qH2x2+qL2 \quad (7)$$

The x2 term can be modulo reduced using the irreducible polynomial from 3), x2 + x + φ. By setting x2 = x + φ and replacing it into x2. Doing so yields the new expressions below.

$$k_3 = q_3$$
$$k_2 = q_3 \oplus q_2$$
$$k_1 = q_2 \oplus q_1$$
$$k_0 = q_3 \oplus q_1 \oplus q_0 \quad (8)$$

The logic diagram to the above equations is given in fig:3.

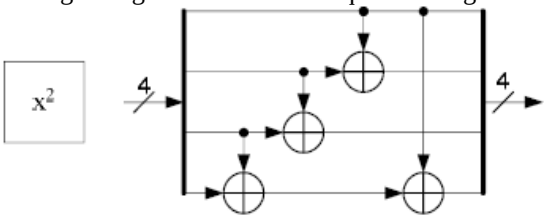

Fig3: Representation of square in GF(24)

C. Multiplication with constant λ in GF(24):

Let q and k are the 4bit elements of GF(24) and let k=qλ, where λ={1100}2 hence neglect lower λL and the equation given by

$$k = qH\ \lambda H\ x2 + qL\ \lambda H\ x \quad (9)$$

Modulo reduction can be performed by substituting x2 = x + φ using the irreducible polynomial in (3) which yields the following equations.

$$k_3 = q_2 \oplus q_0$$
$$k_2 = q_3 \oplus q_2 \oplus q_1 \oplus q_0$$
$$k_1 = q_3$$
$$k_0 = q_2 \quad (10)$$

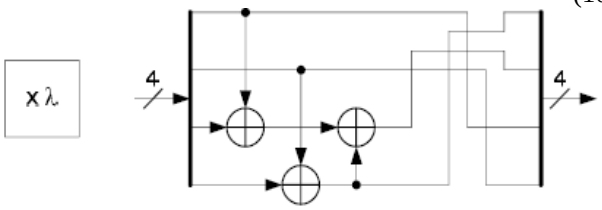

Fig 4: Multiplication with constant λ

D. Multiplication in GF(24):

Let k = qw, where k, q and w are elements of GF(24).

$$k = kH\ x + kL = (qH\ x + qL)\ (wH\ x + wL)$$
$$k = (qHwH)\ x2 + (qHwL + qLwH)\ x + qLwL \quad (11)$$

Substituting the x2 term with x2 = x + φ yields the following

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 5, Issue 5, May 2018

k= (qHwH + qHwL+qLwH) x + qHwH φ+ qLwL ϵ GF(22) (12)

Equation (12) is in the form GF (22). It can be observed that there exists addition and multiplication operations in GF(22). As mentioned in Section III (A), addition in GF(22) is but bitwise XOR operation. Multiplication in GF(22), on the other hand, requires decomposition to GF(2) to be implemented in hardware. Also, it the expression would be too complex if equation (12) were to be broken down to GF(2). Thus, the formula for multiplication in GF(22) and constant φ will be derived instead. Figure 5 below shows the hardware implementation for multiplication in GF(24).

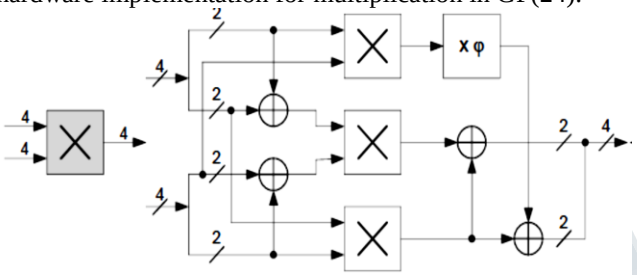

Fig 5: Hardware logic for multiplication in GF(22)

E. Multiplication in GF(22):

Let k = qw, where k = {k1 k0}2, q = {q1 q0}2 and w = {w1 w0}2 are elements of GF (22).

k= (q1w1) x2 + (q1w0+q0w1) x+q0w0 (13)

Modulo reduction to x2 term is done using the irreducible polynomial x2= x+1 in (13) to obtain (14)

k= (q1w1+q1w0+q0w1) x+ (q1w1+q0w0)ϵ GF(2) (14)

The equation above can now be implemented in hardware as multiplication in GF(2) which involves only the usage of AND gates. The formula for computing multiplication in GF(2) is as follows.

$$k_1 = q_1 w_1 \oplus q_0 w_1 \oplus q_1 w_0$$
$$k_0 = q_1 w_1 \oplus q_0 w_0 \quad (15)$$

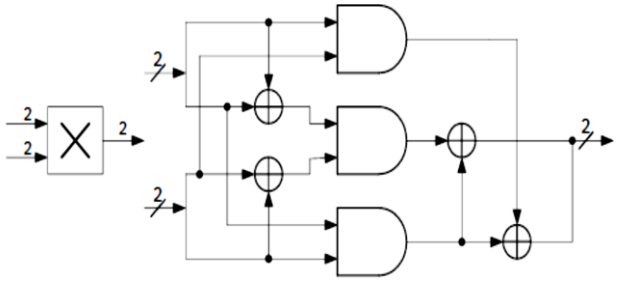

Fig 6: Representation of multiplication in GF (22)

F. Multiplication with constant φ in GF(22):

Let k = qφ, where k = {k1 k0}2, q = {q1 q0}2 and φ = {10}2 are elements of GF (22).

k= (q1 x +q0) x= q1x2+q0 (16)

Here the x2 term is substituted with x2 = x + 1, to yield the expression below

k=q1(x+1) + q0x=(q1+q0)x+q1 (17)

The logic obtained to design the multiplications with constant φ operation in GF(22) is

$$k_1 = q_1 \oplus q_0$$
$$k_0 = q_1 \quad (18)$$

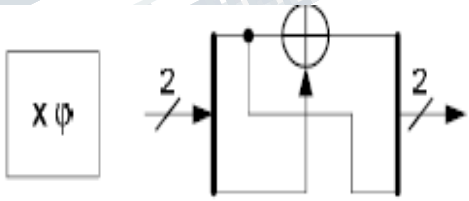

Fig 7: multiplication with constant φ

G. Multiplication inversion in GF(24):

The composite field decomposition approach is used to compute the multiplicative inverse of q (where q is an element of GF (24)) such that q-1 = {q3 -1,q2 -1,q1-1,q0-1}. Hence reduces the gate count and shortest path delay. The inverses of the individual bits can be computed from the equation below.

$$q_3^{-1} = q_3 \oplus q_3q_2q_1 \oplus q_3q_0 \oplus q_2$$
$$q_2^{-1} = q_3q_2q_1 \oplus q_3q_2q_0 \oplus q_3q_0 \oplus q_2 \oplus q_2q_1$$
$$q_1^{-1} = q_3 \oplus q_3q_2q_1 \oplus q_3q_1q_0 \oplus q_2 \oplus q_2q_0 \oplus q_1$$
$$q_0^{-1} = q_3q_2q_1 \oplus q_3q_2q_0 \oplus q_3q_1 \oplus q_3q_1q_0 \oplus q_3q_0 \oplus q_2 \oplus q_2q_1 \oplus q_2q_1q_0 \oplus q_1 \oplus q_0 \quad (19)$$

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 5, Issue 5, May 2018

IV. HARDWARE DESIGN IMPLEMENTATION AND RESULTS

The analytical validation of the combined S-box and InvS-box for AES is accoutrement and verified using the Spartan 6 (xc6slx2tqg144) board using HDL in Xilinx 14.6 tool. The proposed module is initiated and executed in the main module as combined implementation of S-box and InvS-box by using an enable pin to select SubByte/InvSubByte transformation for AES algorithm. The architecture is appliance using two 2:1 multiplexer and the design consists of implementing modules such as isomorphic functions and Invs-isomorphic functions, squaring unit, inversion unit and affine transformation.

Thus, the architecture utilizes 77 slice of LUT's and the reduction in area by 50% and decrease in gate count when compared with previous LUT methods for S-box and lower power consumption. The number of gates and mux used are tabulated below.

Table1: Synthesis Report

2:1 Multiplexer	2
Number of XOR gates	116
No of Slice LUTs	77
Path Delay (ns)	19.889

The simulation outcomes of the suggested architecture using Xilinx ISE14.6 is shown below in fig1,2,3. The SubByte and InvSubByte transformations are formed using the multiplicative inverse module, mux and affine transformations by using an enable pin to select either encryption or decryption based on the selection for the AES algorithm.

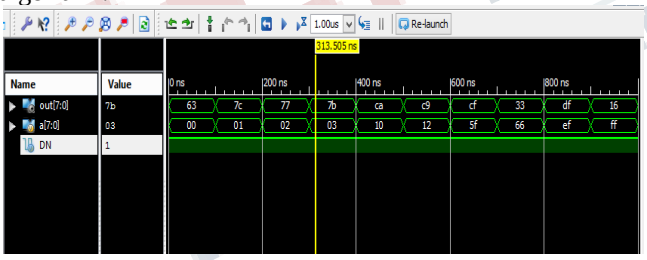

Fig8: Simulation result of SubByte transformations when enable pin EN=1

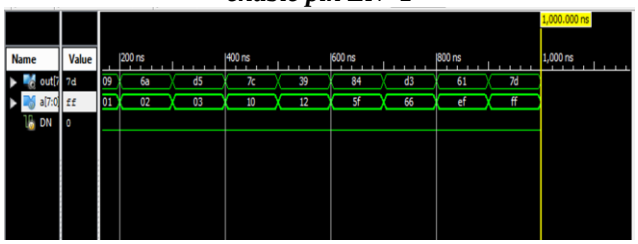

Fig9: Simulation result of InvSubByte transformations when enable pin EN=0

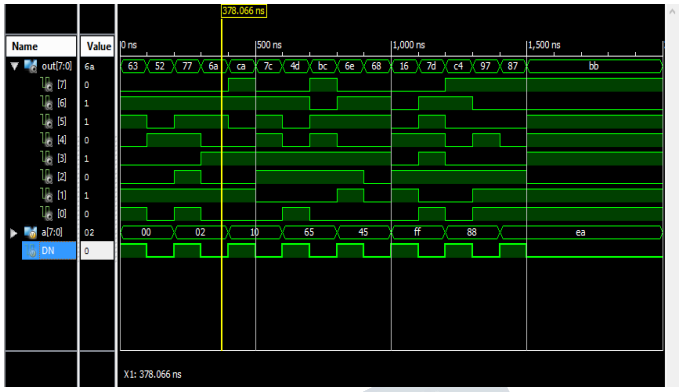

Fig10: Simulation result of combined SubByte &InvSubByte transformations

The power consumption of this proposed architecture of combined s-box and InvS-box for SubByte and InvSubByte of AES algorithm is 0.014w for an input of 128 bits and the frequency of operation is about 60MHz.

V. CONCLUSION

For the efficient implementation of proposed architecture of the SubBytes/InvSubByte is realized by combinational logic to overcome the unbreakable delays of LUTs in the analytical designs. Further, composite field arithmetic and finite fields is used to reduce the hardware complexity and also uses different approaches to implement inversion in subfield GF(24) are compared. The architecture is implemented on Spartan6 FPGA board using Verilog HDL code by making use of enable pin to select s-box/ Invs-box during the operation. The overall path delay initiated is 19.8ns and consumes very less power of 14mW and occupies very less area and memory for the reason that resource allocation is done using multiplicative inversion module.

VI. ACKNOWLEDGMENT

We acknowledge to department of electronics and communication for giving the laboratory resources.

REFERENCES

[1]Edwin NC Mui, "Practical Implementation of Rijndael S-Box Using combinational Logic", Custom R&D Engineer Texco Enterprise

[2] Xinmiao Zhang and Keshab K. Parhi, "High-Speed VLSI Architectures for the AES Algorithm.", IEEE

Transactions on Very Large Scale Integration(VLSI) Systems, Vol.12, No. 9, September 2004.

[3] P.V.S.ShastI, Anuja Agnihotri, Divya Kachhwaha, Jayasmita Singh and Dr.M.S.Sutaone, "A Combinational Logic implementation of S-box of AES", 54th International Midwest Symposium on Circuit and Systems, 2011.

[4] Bhoopal Rao Gangadari and Shaik Rafi Ahamed, "FPGA Implementation of Compact S-Box for AES algorithm using Composite field arithmetic".

[5] Vincent Rijmen, "Efficient Implementation of the Rijndael S-Box.", Katholieke Universities Leuven, Dept. ESAT. Belgium.

[6] Akashi Satoh, Sumio Morioka, Kohji Takano and Seiji Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization." Springer-Verilog Berlin Heidelberg, 2001.

[7] S.SrideviSathya Priya, N.M.SivaMangai "Multiplexer based High Throughput S-box for AES Application" Karunya University, ICECS 2015

[8] "Advanced Encryption Standard (AES)" Federal Information Processing Standards Publication 197, 26th November 2001.