Design of S-box and INV S-box using Composite Field Arithmetic for AES Algorithm

Sushma D K
Department of Electronics and Communication
The Oxford College of Engineering
Bangalore, India

Dr. Manju Devi
Department of Electronics and Communication
The Oxford College of Engineering
Bangalore, India

***Abstract*— The 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 (2^8) hence this approach is more advantages than the conventional LUT method that incurs unbreakable delay, large 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 large amount. The power consumption is reduced by resource sharing of multiplicative inverse module of proposed system. The proposed architecture is implemented on Spatan6 FPGA board using Verilog HDL in Xilinx ISE 14.6.**

Keywords*— 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) was adopted as the standard for encryption and decryption by National Institute of Standards. AES uses larger key sizes (128, 192 and 256bits) hence provides higher security than any other encryption technique. Encryption algorithms are mainly of two types 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 shared for the encryption and decryption processes.. Computation of mathematical inversion in finite field arithmetic by Sub-Byte transformation consumes the most of the resource. The AES algorithm is used in different application fields like Radio Frequency Identification (RFID) tags, World Wide Web (WWW) servers, Automated Teller Machines (ATMs), smart cards, cellular phones, digital video and sensor nodes. AES can be implemented in both hardware and software. The four important operations in AES algorithm are S-Box, inverse S-Box, MixColumn and InverseMixColumn steps in these are computationally more involved than addroundkey and shift row operations. The designs, which do not use ROMs or big lookup tables, implementations for S-Box and inverse S-Box have been popular of late for in VLSI or FPGA implementations.
Byte Substitution and Inverse Byte Substitution transformation is non-linear transformation that maps each byte of the state that is 128 bits to different value using the substitution tables for S-box and InvS-box. It can be implemented by using memory method and memory-less method. In memory method, ROM based LUT (Look-up table) is used to compute the S-box that utilizes more memory, which increases area, power of AES and thus disadvantage of this is unbreakable delay and latency because of finite time of the architecture. In memory-less method, implementation of S-Box using LUT and SOP approach is fast but effective in cost.
The structure of this paper is as follows. The construction of ByteSub and InvByteSub transformations is explained in section II. The Composite arithmetic operations is explained in section III. Hardware implementation of the proposed architecture is described in section IV

II. THE CONSTRUCTION OF BYTESUB AND INVBYTESUB TRANSFORMATION FOR AES

The ByteSub& InvByteSub transformation are calculated by the application of the multiplicative inverse to the plain text in GF (2^8) and then affine transformation is applied to it. For decryption, the InvByteSub transformation is calculated by the application of the inverse affine transformation to cipher text before applying the multiplicative inverse [6]. The multiplicative inverse operation is involved in both the ByteSub and its inverse transformations.

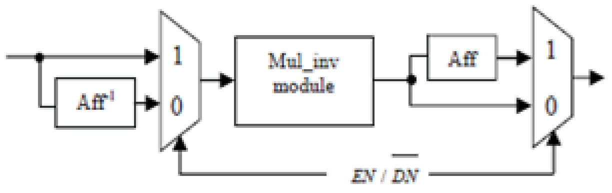


Fig 1: Combined ByteSub and invByteSub transformation

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(2^8) .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 may 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 set to 0 for SubByte while 1 is set when InvSubByte operation is desired.

The SubBytes is a nonlinear transformation, which computes the multiplicative inverse of each byte of the State in followed by an affine transformation. The SubBytes can be described by (1)

$$S^{1}_{i,j} = M \cdot S^{-1}_{i,j} + C \qquad (1)$$

Were $S_{i,j}(0<i,j<4)$ is considered a element of GF(2^8) M is 8x8 binary matrix and c is a 8 bit binary vector with only 4 nonzero bits. The transformations in the decryption process perform the inverse of the corresponding transformations in the encryption process. Specifically, the InvSubByte performs the following operation on each byte of the State by (2)

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

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

B. *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 (2^8). So, to reduce the gate count in large amount, composite field arithmetic is used.

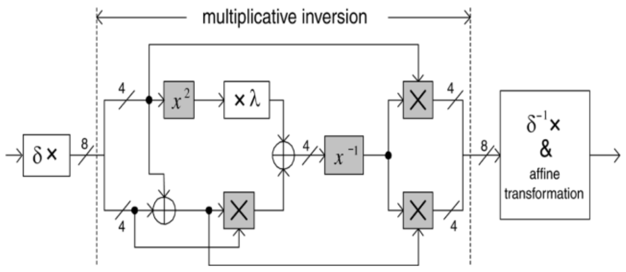


Fig 2: Multiplicative inverse module for AES algorithm

i. *Isomorphic mapping function and its inverse*

Composite field is denoted as GF($(2^n)^m$) , this is Isomorphic to the finite field GF(2^k), for k = nm. The composite field GF(2^8) can be formed iteratively from lower order fields like GF(2) using the irreducible polynomials which are mentioned in (5)

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

Where φ = $\{10\}_2$ & δ = $\{1100\}_2$. To represent an element of finite field GF(2^8) 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(2^8), 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 $p(x) = x^8 + x^4 + x^3 + x + 1$ of the finite field GF (2^8) 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 (2^8), 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).

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

ii. *Multiplicative inversion in GF(2^8):*

In the composite field GF(2^8) , an element can be expressed as bx + c, where b, c in GF(2^4) are first and second nibbles of the byte and x is a root of irreducible polynomial $P_2(x)$ in eq(3). The multiplicative inverse of bx + c modulo $P_2(x)$ can be computed by using Extended Euclidean algorithm [2] [5] as shown in (6).

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

From the above equation implies that there are multiply, addition, squaring and multiplication inversion in GF(2^4) operations in Galois Field.

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 q_H x + q_L for instance, if q = $\{1011\}_2$, it can be represented as $\{10\}_2$x + $\{11\}_2$, where q_H is $\{10\}_2$ and q_L is $\{11\}_2$. 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(2^4):*

Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation between the 2 elements.

B. *Squaring in GF(2^4):*

Let 'q' is an element in GF(2^4) which can written as q_H x + q_L and this can be split, let 'k' is another element in GF(2^4) which is equal to square of q as given in equation.

$$k_H x + k_L = (q_H x + q_L)^2 = q_H^2 x^2 + q_L^2 \qquad (7)$$

The x^2 term can be modulo reduced using the irreducible polynomial from (3). By setting $x^2 = x + \varphi$, doing so yields the new expressions below.

$$\begin{aligned} k_3 &= q_3 \\ k_2 &= q_3 \oplus q_2 \\ k_1 &= q_2 \oplus q_1 \\ k_0 &= q_3 \oplus q_1 \oplus q_0 \end{aligned} \qquad (8)$$

The hardware logic diagram to the above equations is shown below in fig:3.

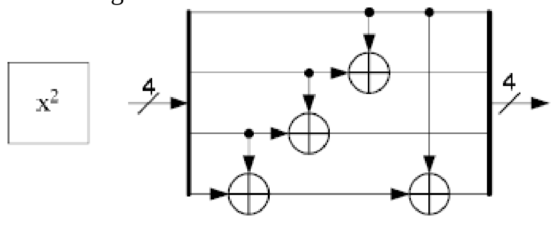


Fig3: Hardware logic diagram of square in GF(24)

C. *Multiplication with constant λ*

Let q and k are the 4bit elements of GF(2^4) and let k=qλ, where λ={1100}$_2$ hence neglect lower λ_L and the equation given by

$k = q_H \lambda_H x^2 + q_L \lambda_H x$ (9)

Modulo reduction can be performed by substituting $x^2 = x + \varphi$ using the irreducible polynomial in (3) to yield the expression below

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

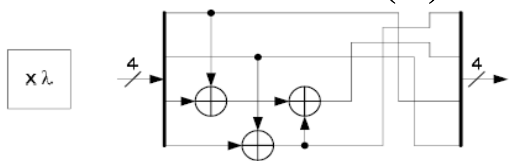


Fig 4: Hardware logic diagram of multiplication with constant λ

D. *Multiplication with constant φ in GF(2^2):*

Let k = qφ, where k = {k_1 k0}2, q = {q_1q_0}2 and φ = {10}2 are elements of GF (2^2).

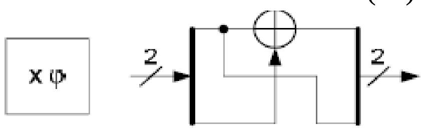
k= (q1 x +q0) x= q1x2+q0 (11)

Substitute the x2 term with x2 = x + 1, yield the expression below

$k=q_1(x+1) + q_0x= (q_1+q_0)x+q_1$ (12)

The formula obtained to compute its multiplications with constant φ operation in GF(2^2) is

$$k_1 = q_1 \oplus q_0$$
$$k_0 = q_1 \quad (13)$$



(14)

Fig 5: Hardware logic diagram of multiplication with constant φ

E. *Multiplication inversion in GF(2^4):*

The composite field decomposition approach is used to compute the multiplicative inverse of q (where q is an element of GF (2^4)) such that q-1 = {q_3^{-1},q_2^{-1},q_1^{-1},q_0^{-1}}. Hence reduces the gate count and shortest path delay. The inverses of the individual bits can be computed from the equation below

$$q_3^{-1} = q_3 \oplus q_3q_2q_1 \oplus q_3q_0 \oplus q_2$$
$$q_2^{-1} = q_3q_2q_1 \oplus q_3q_2q_0 \oplus q_3q_0 \oplus q_2 \oplus q_2q_1$$
$$q_1^{-1} = q_3 \oplus q_3q_2q_1 \oplus q_3q_1q_0 \oplus q_2 \oplus q_2q_0 \oplus q_1$$
$$q_0^{-1} = q_3q_2q_1 \oplus q_3q_2q_0 \oplus q_3q_1 \oplus q_3q_1q_0 \oplus q_3q_0 \oplus q_2 \oplus q_2q_1 \oplus q_2q_1q_0 \oplus q_1 \oplus q_0 \quad (13)$$

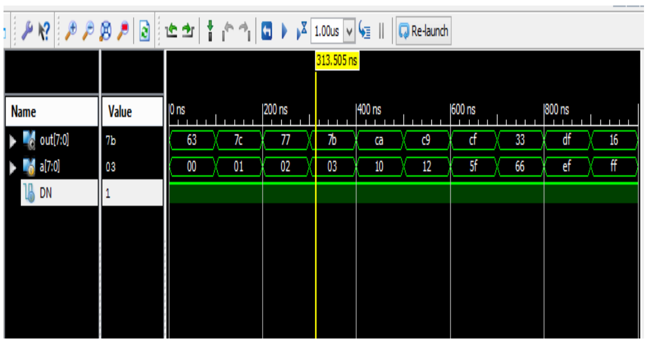
IV. HARDWARE DESIGN IMPELEMENTATION AND RESULTS

The analytical validation of the combined S-box and InvS-box for AES is accoutrement and verified using the Spartan 6 (xc6slx2tqg144) FPGA board using Verilog HDL in Xilinx 14.6 tool. The proposed module is initiated and implemented in the main module as combined implementation of s-box and InvS-box by using an enable pin to select SubByte and InvSubByte transformation for AES algorithm. The architecture is appliance using two 2:1 multiplexer as shown in fig.1and the design consists of implementing modules such as isomorphic function and Invs-isomorphic function, 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 low power consumption. The number of gates and mux used are tabulated below in table 1.

Table1: synthesis report

HDL SYSNTHSIS REPORT	
2:1 Multiplexer	2
Number of XOR gates	116
Slice LUTs	77
Delay (ns)	19.889

The simulation results of the proposed architecture using Xilinx ISE14.6 is shown below figures. 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.



Name	Value										
out[7:0]	7b	63	7c	77	7b	ca	c9	cf	33	df	16
a[7:0]	03	00	01	02	03	10	12	5f	66	ef	ff
DN	1										

Fig6: Simulation result of SubByte transformations when enable pin EN=1

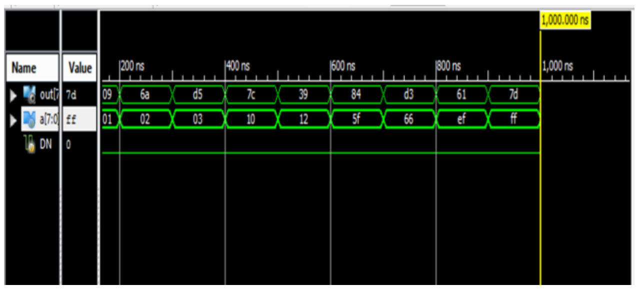


Fig7: Simulation result of InvSubByte transformations when enable pin EN=0

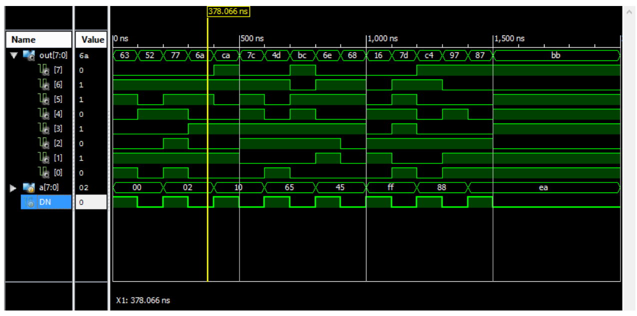


Fig8: Simulation result of combined SubByte &InvSubByte transformations

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.

CONCLUSION

For the efficient implementation of proposed architecture of the SubBytes/InvSubByte is implemented by combinational logic to avoid the unbreakable delay 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(2^4) 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 delay caused by the logic is 19.8ns and consumes very less power of 14mW and occupies very less area and memory because of resource sharing in multiplicative inversion module.

ACKNOWLEDGMENT

We acknowledge to department of electronics and communication for providing the lab resources.

REFERENCE

1. Edwin NC Mui, "Practical Implementation of Rijndael S-Box Usingcombinational 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. K. Jarvinen, M. Tommiska, and J. Skytta, “Comparative survey of high performance cryptographic algorithm implementations on FPGAs, ” IEEE Proceedings Information Security, vol. 152, no. 1, pp. 3–12, Oct 2005.
5. K. Shesha Shayee, J. Park, and P. Diniz, “Performance and area modeling of complete FPGA designs in the presence of loop transformations,” in 11th Annual IEEE Symposium on field-Programmable Custom Computing Machines, 2003, FCCM 2003, April 2003.
6. Bhoopal Rao Gangadari and Shaik Rafi Ahamed, “FPGA Implementation of Compact S-Box for AES algorithm using Composite field arithmetic”.
7. X. Zhang and K. Parhi, “High-speed VLSI architectures for the AES algorithm,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 9, pp. 957–967, Sept 2004.
8. L. Ali, I. Aris, F. S. Hossain, and N. Roy, “Design of an ultra-high speed {AES} processor for next generation {IT} security ,” Computers & Electrical Engineering, vol. 37, no. 6, pp. 1160 – 1170, 2011.
9. J. M. Granado-Criado and M. A. Vegaz, “A new methodology to Implement the AES algorithm using partial and dynamic reconfiguration,” *The {VLSI}* Journal *on Integration*, vol. 43, no. 1, pp. 72 – 80, 2010.
10. Vincent Rijmen, “Efficient Implementation of the Rijndael S-Box.”, Katholieke Universities Leuven, Dept. ESAT. Belgium.
11. Akashi Satoh, Sumio Morioka, Kohji Takano and Seiji Munetoh, “A Compact Rijndael Hardware Architecture with S-Box Optimization.” Springer-Verilog Berlin Heidelberg, 2001.
12. S.SrideviSathya Priya, N.M.SivaMangai “Multiplexer based High Throughput S-box for AES Application” Karunya University, ICECS 2015
13. “Advanced Encryption Standard (AES)” Federal Information Processing Standards Publication 197, 26th November 2001.