IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 04, Issue 06 (June. 2014), ||V1|| PP 25-29

Area Optimization in Masked Advanced Encryption Standard

R.Vijayabhasker, K.Mohankumar[1], S.R.Barkunan,
Assistant Professor Depatrment of ECE Anna University Regional Centre,Coimbatore
P.G Scholar, Anna University Regional Centre, Coimbatore
Assistant Professor Depatrment of ECE Anna University Regional Centre,Coimbatore

Abstract: - The Advanced Encryption Standard (AES) a symmetric-key block ciphertext published by National Institute of Standards and Technology (NIST)[1]. Order to protect a data high throughput masked Advanced Encryption Standard (AES) is used. The masked AES engine uses the unrolling techniques which required extreme level in large field programmable gate array (FPGA). The area optimization in masked AES with an unrolled structure. The mapping of operation from GF(2^8) to GF(2^4) as much as possible in order to optimize area. The number of mapping is reduced [GF(2^8) to GF(2^4)] and inverse mapping [GF(2^4) to GF(2^8)] operation of the SubBytes by step from zero to nine. In order to compatible, the masked MixColumns, masked AddRoundKey, and masked ShiftRows including the redundant masking values are carried over GF(2^4). By moving, mapping and inverse mapping outside the masked AES's round function, area can be reduced by 20%.

Keyword: - *Advanced Encryption Standard(AES), Field Programmable Gate Array(FPGA), throughput, Galios Field(GF).*

I. INTRODUCTION

The development of information technology protect sensitive information's via encryption is more and more important. The Advanced Encryption Standard (AES) is a symmetric-key block ciphertext published by the National Institute of Standards and Technology (NIST) in December 2001 [1]. The criteria is defined by NIST for selecting AES drop into three areas i) Security ii) Cost and iii) Implementation. Based on the criteria NIST selected Rijndale algorithm as Advanced Encryption Standard(AES) [1]. AES is a non-Feistel cipher that encrypts and decrypts a data block of 128 bits. It uses 10, 12, or 14 rounds. The key size, which can be 128, 192, or 256 bits, depends upon the number of rounds. The standard AES key size is 128 bits and 10 rounds. To provide security for the data, AES uses four types of transformations: substitution, permutation, mixing, and key-adding. AES has been widely used in various applications, such as secured communication systems. In high performance database, Radio Frequency Identification (RFID) tags and smart cards.

II. ADVANCE ENCRYPTION STANDARD

AES is a symmetric algorithm, it takes 128-bit data blocks input performs several rounds of transformations to generates the ciphertext output. Each 128-bit data block are processed in 4 X 4 array of bytes, This process is denoted as state [3]. The round key size can be 128, 192 or 256 bits. The number of rounds can be 10, 12 or 14 depending upon the length of the round key respectively. There are four basic transformations applied for encrypting the data.

2.1 SUBBYTES

This subBytes operations is a nonlinear bytes substitution. Each byte from input state is replace by another byte according to the substitution of S-box. S-box is generated based upon a multiplication inverse in the finite field GF(2^8) and a bitwise affine transformation. The affine transformation is the sum of multiple rotation of the byte as a vector, where additions is performed using the XOR operation.
SubByte → Multiplicative Inversion in GF(2^8) → Affine Transformation

2.2 SHIFT ROWS

Shift rows operation is used to shift the rows of the state. The bytes of data are cyclically shifted with a certain offset. The first row is left unchanged and the second, third and fourth row is shifted to one, three bytes to the left simultaneously. Shiftrow 'n' is shifts left circular by 'n-1' bytes. Each column output state of the ShiftRows step is composed on bytes from each column of the input state [1].

2.3 MIXCOLUMNS

Each column considered state over a polynomial GF(2^8), after multiplying modulo $X^4 + 1$ with fixed polynomials a(x), given by $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x^1 + \{02\}$ the results corresponding column of the output state.

2.4 ADD ROUNDKEY

Each byte array is added (respect to GF(2)) to a bytes corresponding of the array round subkeys. Excluding the first and the last round, the AES with 128 bits round key proceeds nine iterations. First round iteration of the encryption perform XOR with the original key at last round iteration skips MixColumn transform [5]. Round keys are generated by a procedure call round key scheduling or key expansion. Those sub-keys are derive from the original key by XOR two previous columns. Columns that are in multiples of four, the process involves round constants addition, S-Box and shiftrow operations [6].

All the four layers described above key scheduling have corresponding inverse operations. Deciphering is the process of converting the cipher text back to plain text, it is the inverse of cipher process. However, it should be noted that the MixColumn reverse operation required matrix elements that are quite complicated compared to {01}, {02} or {03} of the forward one.

This results are more complex deciphering hardware compared with the ciphering hardware. In the next section we demonstrate how the standard procedure for MixColumn transform is rewritten in order to ease its hardware implementation.

III. MASKED AES FOR UNROLLED STRUCTURE

The intermediate value X is concealed by exclusive-OR_{ing} it with the random mask m, in the Boolean masking implementation. The round function of the AES contains ShiftRows, Mix-Columns and AddRoundKey which are linear transformations, while SubBytes is the only nonlinear transformations of the AES. The linear transformations is defined as Operation; then, the masked Operation can be written as Operation(x $\oplus$ m) = Operation(x) $\oplus$ Operation(m). The masked nonlinear transformation SubBytes has the characteristic as S-box(x$\oplus$m) ≠ S-box(x) $\oplus$S-box(m). To mask the nonlinear transformation, a new S-box, denoted as S-box1, is recomputed as S-box1 (x $\oplus$ m) = S-box(x) $\oplus$ m' , where m and m' are the input and output masks of the SubBytes. In order to mask a 128-bit AES, it usually needs 6-byte random values. These 6-byte random values are defined as m, m', m1, m2, m3, and m4. m_{1234} = {m1,m2,m3,m4} is defined as the mask for one 32-bit MixColumns transformation, and it also holds that m'_{1234} = MixColumns(m_{1234}).

The Galois field GF(2^8) is an extension of the Galois field GF(2^4) over which to perform a modular reduction needs an irreducible polynomial of degree 2, $x^2 + \{1\}x + \{e\}$, and another irreducible polynomial of degree 4, $x^4 + x + 1$. In order to reduce the hardware resources, we calculate the masked AES engine mainly over GF(2^4). The plaintext and the masking values are mapped once from GF(2^8) to GF(2^4), and all the intermediate operations are computed over GF(2^4). Finally, the ciphertext is mapped back from GF(2^4) to the original field GF(2^8). All the masking values need to be mapped from GF(2^8) to GF(2^4), and we denote m_{84} = map(m), m'_{84} = map(m'), $m_{1234,84}$= map(m_{1234}), and $m_{1234,84}$ = map(m'_{1234}). The masked ShiftRows and masked AddRoundKey remain the same.

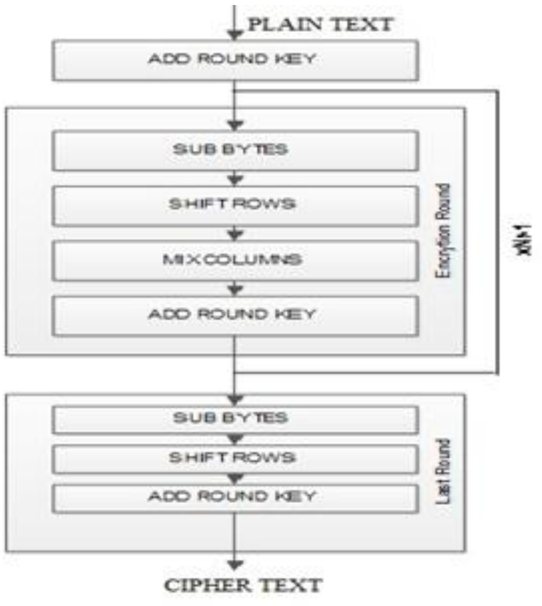


Fig 1.1 Encryption

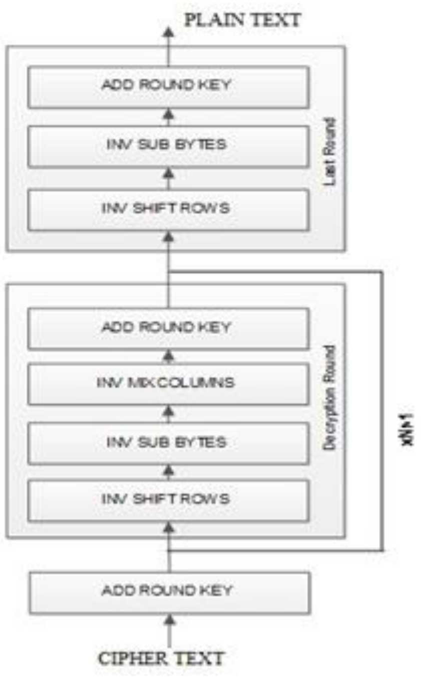


Fig 1.2 Decryption

IV. OPTIMIZED MASKED S-BOX OVER GF(2^4)

In order to move the mapping and inverse mapping outside AES's round operation, we exchange the computational sequence of masked affine and inverse mapping functions within masked S-box. The masked affine function needs to be adjusted with new scaling factors. The map operation is the mapping transformation of 8 × 8 matrix, and map^{-1} is constructed by the inverse map operation. We denote that the input values of the map function are (z + m) and m, and the output values of the map function are (z + m)' and m' , where {(z + m),m} € GF(2^8) and {(z + m)',m'} € GF(2^4)

It holds that

$(z + m + m)' = map(z + m + m)$ (1)

where $(z + m)' = \{a^* h + m_h, a^* l + m_l\} and\ m' = \{m_h, m_l\}$

As discussed before, m-affine and m-affine' are needed for scaling the output values and the output masking values. The following steps introduce the procedure to obtain the scaling values. The normal affine function (Ax + b) can be applied to the left and the right sides of (1) as

$A(z + m + m) + b = Amap^{-1}(z + m + m)' + b$

(2)

When mapping Equation (2) from GF(2^8) to GF(2^4), we

can get map (A(z + m + m) + b) = map' Amap^{-1}(z + m + m)'+ b' (3)

map(A(z + m) + b) + mapAm = mapAmap^{-1} (z + m)' + mapb + mapAmap^{-1} m' (4)

Therefore, we deduce that m-affine = mapAmap^{-1} + mapb and m-affine' = mapAmap^{-1}. The four tables in masked sbox remains the same in our previous work[11].

These four tables are the following:

1) T_{d1} : ((x + m),m) → x^2 × e + m;
2) T_{d2} : ((x + m), (y + m')) → ((x + m) + (y + m')) × (y + m');
3) T_{dm} : ((x + m), (y +m')) → (x + m) × (y + m'); and
4) T'_{inv} : ((x + m),m) → T_{inv}(x) + m.

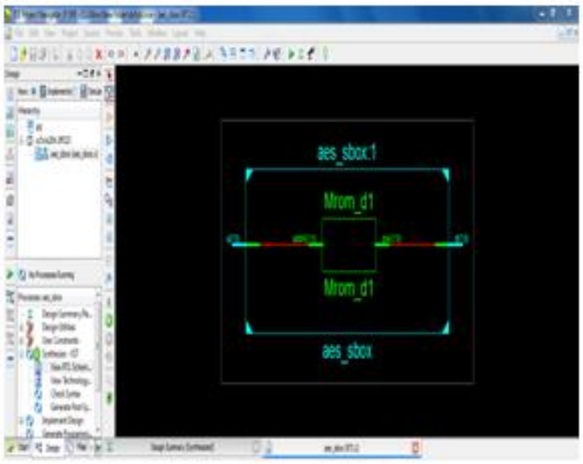
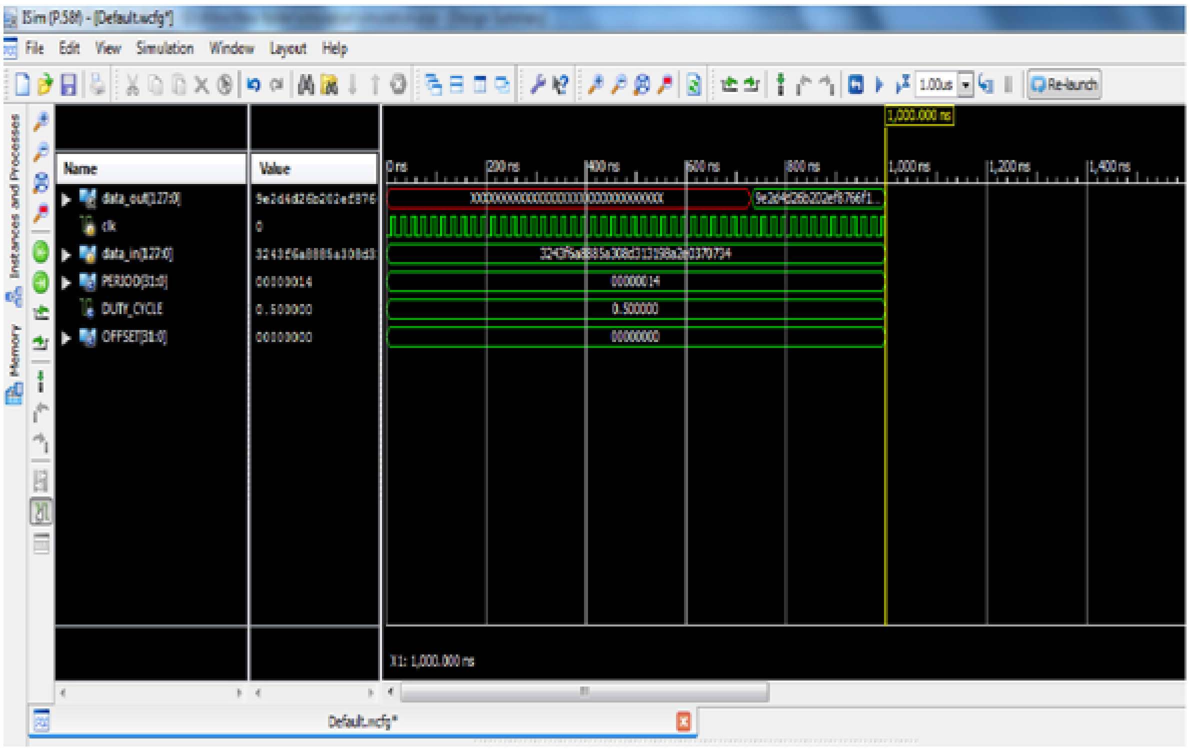


Fig 2 Masked Sbox GF(2^4)

V. MASKED MIXCOLUMNS OVER GF(2^4)

Scaling of Masked MixColumns is done by adjusting the operations over GF(2^4), and it needs to deduce the scaling factor of a modular multiplication with the fixed coefficients 0X02 and 0X03. If S is 1 byte of MixColumns, it holds that S = map(S_h, S_l) ~Shx + Sl, where S € GF(2^8) and S_h, S_l € GF(2^4). Therefore, scaling factors 2x + 6 and 2x + 7 of S = (4Sh + 2Sl)x + (fS_h + 6S_l) and (5S_h + 2S_l)x + (fS_h + 7S_l). Fig. 4 shows the scaling computation for the masked MixColumns.



Name	Value
data_out[127:0]	9e2d4d26b202ef876
clk	0
data_in[127:0]	3243f6a8885a308d3
PERIOD[31:0]	00000014
DUTY_CYCLE	0.500000
OFFSET[31:0]	00000000

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   9e2d4d26b202ef8766f1...
3243f6a8885a308d313198a2e0370734
00000014
0.500000
00000000
```

Fig 3 Masked Sbox

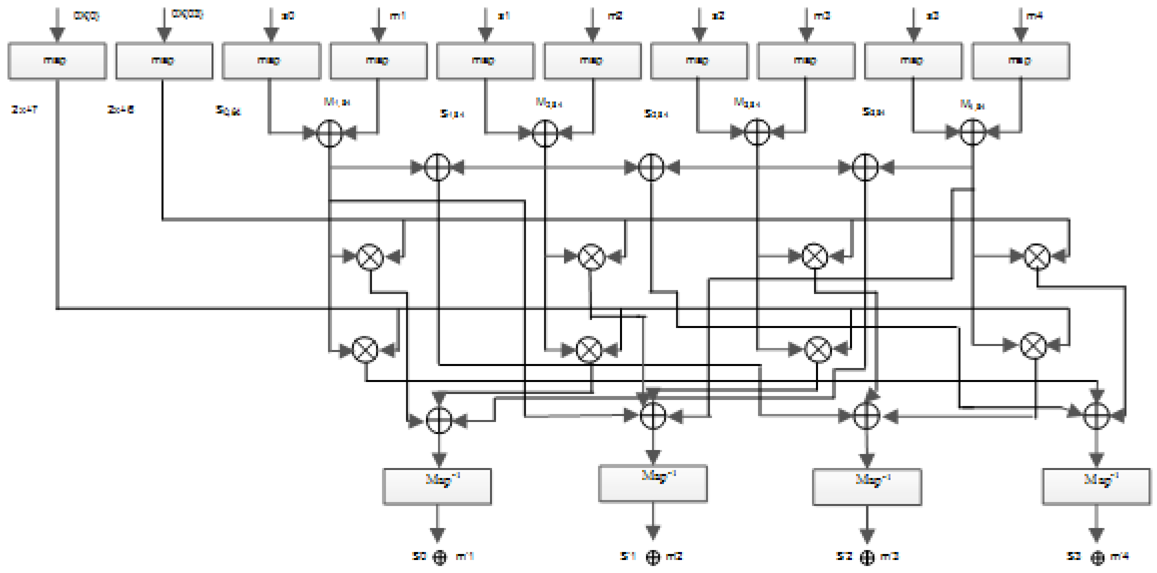


Fig 4 COMPUTATION OF MASKED MIXCOLUMN

VI. CONCLUSION

Throughputs can be enhanced by inserting pipeline registers for latency careless designs. In order to enhance the throughputs of each masked AES's round six-stage pipelines are inserted. Three pipelines to each round of the masked AES, called outer three pipelines are inserted. The pipeline registers are inserted at the output of each transformation. Three pipelines to the masked S-box, called inner three pipelines are inserted. Note that the maximum pipelined stages for the design is six. In order to be compatible with the encryption procedure, we also insert six-stage pipelines to the key expansion in order not to affect the critical path of the main encryption. High throughput is an important factor for large data transformation systems. Masked AES only needs to map the plaintext and masking values from GF(2^8) to GF(2^4) once at the beginning of the operation and map the ciphertext back from GF(2^4) to GF(2^8) once at the end of the operation. By moving, mapping and inverse mapping outside the masked AES's round function, area can be reduced by 20%. The output for Masked Sbox GF(2^4) is shown in fig 2 and the

REFERENCES

[1] NIST, "Advanced Encryption Standard (AES)," http://csrc.nist.gov/publications/fips/fips-197.pdf, Nov-2001.

[2] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in Proc. CHES LNCS, 2005, vol. 3659, pp. 157–171.

[3] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen, "A side-channel analysis resistant description of the AES S-box," in Proc. FSE LNCS, Setubal, Potugal, 2005, vol. 3557, pp. 413–423.

[4] L. Goubin and J. Patarin, "DES and differential power analysis (the 'duplication' method)," in Proc. CHES LNCS, 1999, vol. 1717, pp. 158–172.

[5] S. Messerges, "Securing the AES finalists against power analysis attacks," in Proc. FSE LNCS, 2000, vol. 1978, pp. 150–164.

[6] K. Gaj and P. Chodowiec, "Fast implementation and fair comparison of the final candidates for advanced encryption standard using field programmable gate arrays," in Proc. CT-RSA LNCS, 2001, vol. 2020, pp. 84–99.

[7] A. Hodjat and I. Verbauwhede, "A 21.54 Gbits/s fully pipelined processor on FPGA," in Proc. IEEE 12th Annu. Symp. Field-Programm. Custom Comput. Mach., 2004, pp. 308–309.

[8] NIST, "Data Encryption Standard (DES)," http://csrc.nist.gov/ publications/fips/fips46-3/fips46-3.pdf, Oct. 1999.

[9] I. Verbauwhede, P. Schaumont, and H. Kuo, "Design and Performance Testing of a 2.29 gb/s Rijndael Processor," IEEE J. Solid-State Circuits, vol. 38, no. 3, pp. 569-572, Mar. 2003.

[10] J. Daemen and V. Rijmen, The Design of Rijndael. Springer-Verlag, 2002.

[11] Z. Yuan, Y. Wang, J. Li, R. Li, and W. Zhao, "FPGA based optimization for masked AES implementation," in Proc.IEEE 54th Int. MWSCAS, Seoul, Korea, 2011, pp.1–4