Masked Advanced Encryption Standard for Area Optimization

M. Mano
P.G Scholar, Bannari Amman Institute of Technology

K. Rekha Swathi Sri
P.G Scholar, Bannari Amman Institute of Technology

G. Selva Priya
P.G Scholar, Bannari Amman Institute of Technology

ABSTRACT
The Advanced Encryption Standard (AES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST)[1]. In order to protect data, a high throughput masked Advanced Encryption Standard (AES) engine is used. The masked AES engine uses the unrolling technique which requires extremely large field programmable gate array (FPGA) resources. The area for a masked AES with an unrolled structure is optimized. The mapping of operations 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)] operations of the masked SubBytes step from ten to one. In order to be 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), throughput, Galios Field (GF), Masked AES

1. INTRODUCTION
With the development of information technology, protecting sensitive information via encryption is more and more important. The Advanced Encryption Standard (AES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST) in December 2001 [1]. The criteria defined by NIST for selecting AES fall 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 on 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 a variety of applications, such as secure communication systems, high performance database, RFID tags and smart cards.

2. ADVANCE ENCRYPTION STANDARD
AES is a symmetric encryption algorithm; it takes 128-bit data block as input and performs several rounds of transformations to generate ciphertext as output. Each 128-bit data block are processed in a 4 X 4 array of bytes, called the 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 Sub Bytes
The subBytes operation is a nonlinear byte substitution. Each byte from input state is replaced by another byte according to the substitution box(S-box). S-box is generated based upon a multiplicative inverse in the finite field GF(2^8) and a bitwise affine transformation. The affine transformation is the sum of multiple rotations of the byte as a vector, where addition 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, two and three bytes to the left simultaneously. Row n is shifted left circular by n-1 bytes. Each column of the output state of the ShiftRows step is composed of bytes from each column of the input state [1].

2.3 Mixcolumns
Each column of the state array is considered as a polynomial over GF(2^8). After multiplying modulo $X^4 + 1$ with a fixed polynomial a(x), given by $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x^1 + \{02\}$ the result is the corresponding column of the output state.

2.4 Addroundkey
Each byte of the array is added (respect to GF (2) to a byte of the corresponding array of round subkeys. Excluding the first and the last round, the AES with 128 bit round key proceeds for nine iterations[3]. First round of the encryption performs XOR with the original key and the last round skips MixColumn transform [5]. Round keys are generated by a procedure called round key expansion or key scheduling [6]. Those sub-keys are derived from the original key by XOR of two previous columns. For columns that are in multiples of four, the process involves round constants addition, S-Box and shift operations [7].

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

These results in the 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.

3. 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.

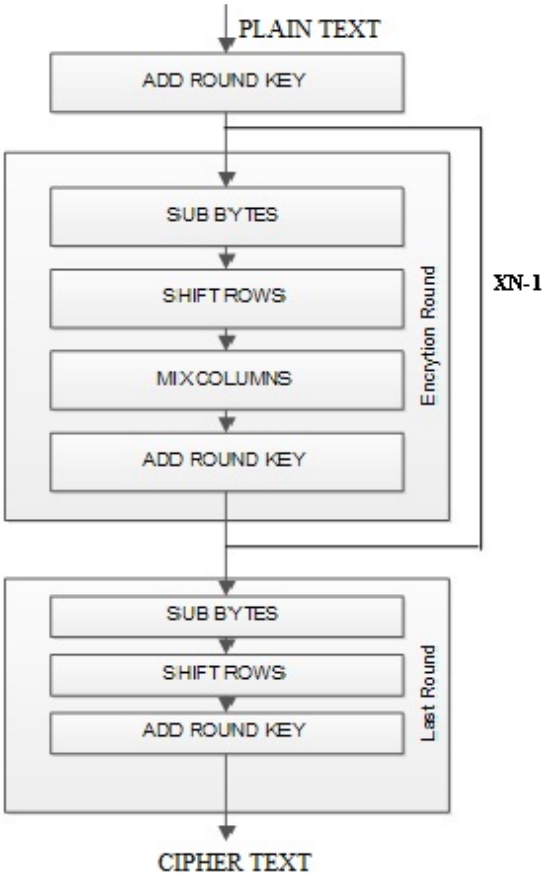

Fig 1 Encryption

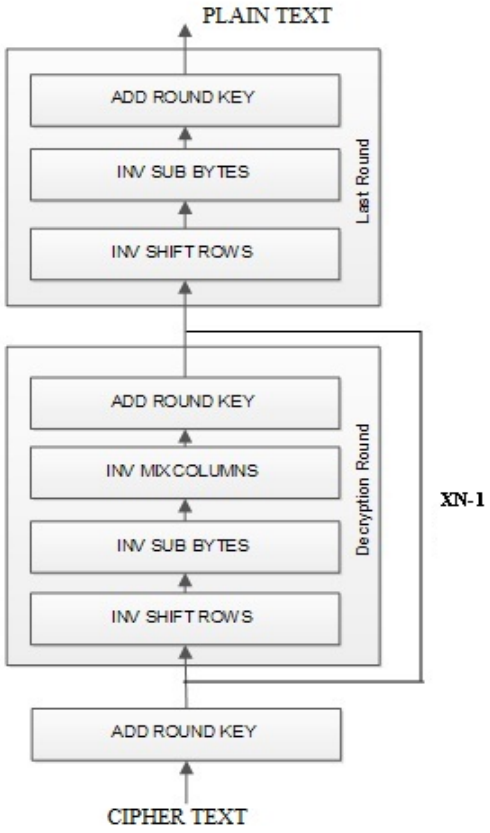

Fig 2 Decryption

4. 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

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

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

$$A(z + m + m) + b = Amap^{-1}(z + m + m)' + b \quad (2)$$

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 remain 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.

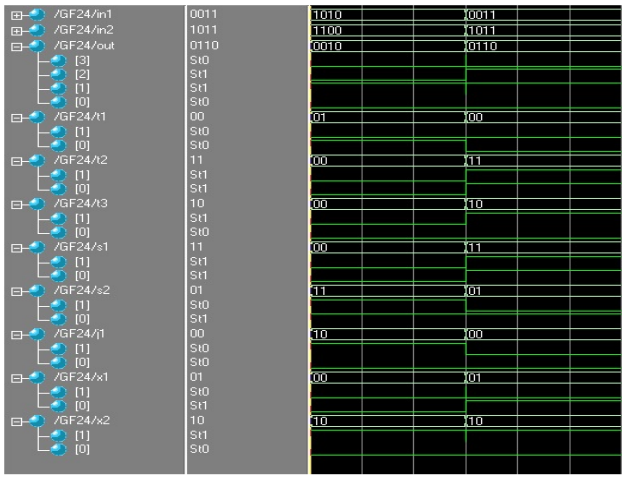

Fig 3 Masked Sbox GF(2^4)

5. 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.

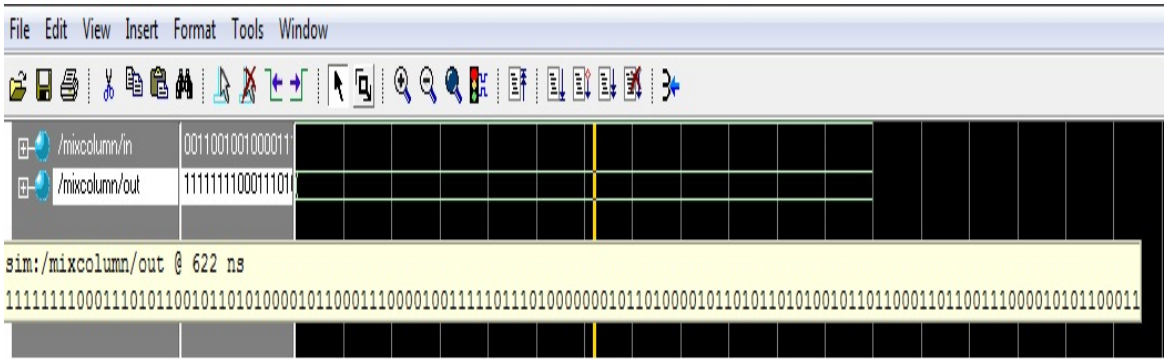

Fig 4 Masked Sbox

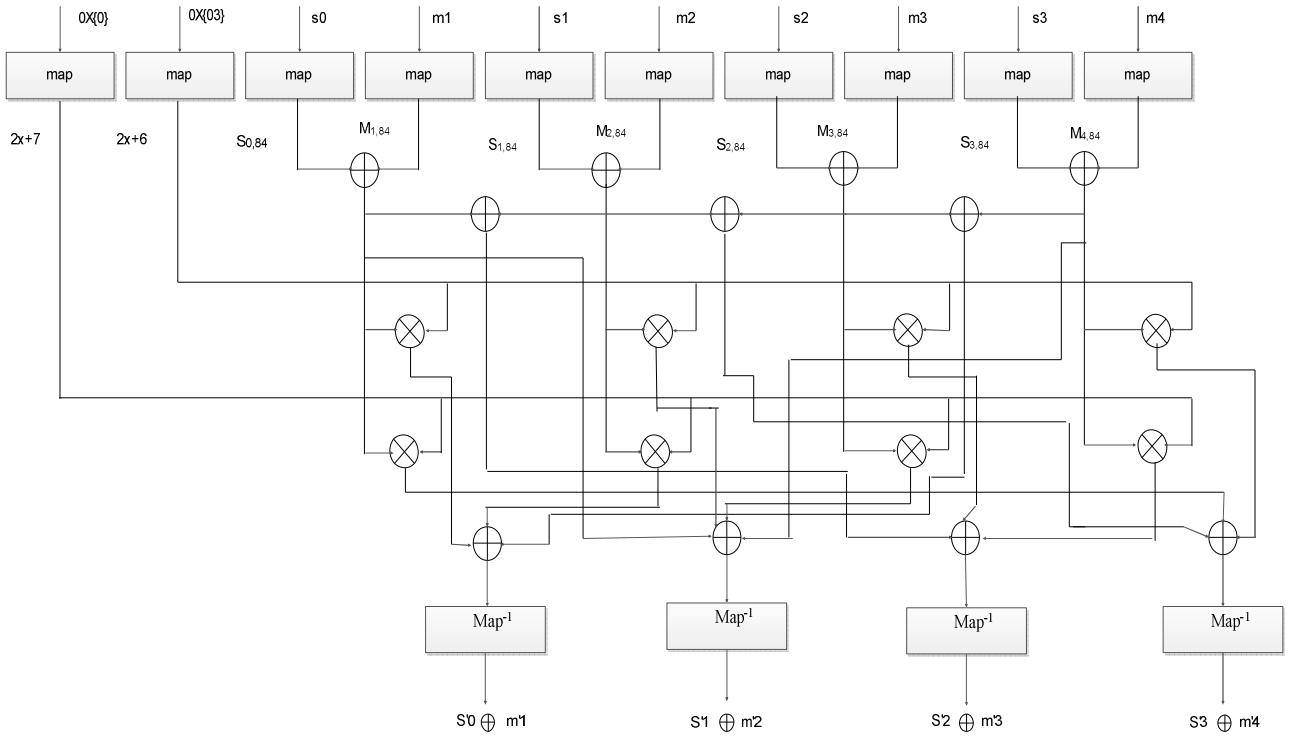

Fig 5 Computation of Masked Mixcolumn

Table 1 Comparison of SBOX

Designs	Area (No. of slice Reg)	Delay (ns)
Sbox	**77**	**8.291**
Proposed Sbox	**23**	**3.019**

6. 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 output of each transformation's are inserted with pipeline registers. Three pipelines to the masked S-box, called inner three pipelines are inserted. Note that the maximum pipelined stages for the design are 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 3. Table 1 shows the comparison of Sbox with the proposed Sbox. Synthesis is done using Xilinx tool, the proposed Sbox can be used for reducing the area, when implemented in AES. Hence the complete architecture of the AES can be implemented using the proposed Sbox and Mix Column block thus optimizing area for AES.

7. 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 AESS-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] J. Nechvatal et. al., Report on the development of Advanced Encryption Standard, NIST publication, October 2, 2000.

[8] http://csrc.nist.gov/CryptoToolkit/aes/

[9] 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.

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

[11] 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.

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

[13] 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