Design of High Speed and Low Area Masked AES Using Complexity Reduced Mix-Column Architecture

J.Balamurugan[1], Dr.E.Logashanmugam[2]
Research Scholar[1], Professor and Head[2],
St.Peter's University[1], Sathyabama University[2], TN, India

Abstract - In this paper, presents optimized mix column architecture to get less area and delay than the existing mix column. Because the proposed mix column architecture is reduced using Xtime unit. In the existing architecture, we are used complex mix-column in masked AES architecture. It consists of 8 multipliers and 11 adders. It consumes more area and power. In the proposed architecture, a novel mix-column is introduced to reduce the area and power than the existing method. The proposed mix-column consists of only 12 adders and 4 Xtime circuit. Instead of normal multiplier in the MixColumns, we are using Xtime unit to reduce the circuit complexity. The proposed 128bits AES Scheme provides high secure and less area. It is used for ATM, Net banking, Satellite Communication, Military applications.
.

Keywords: Advanced Encryption Standard, ATM, Mix-Column Transformation, Multipliers, X-time circuit.

I. INTRODUCTION

This AES standard specifies the Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 196, and 256 bits. Rijndael was designed to handle additional block sizes and key lengths; however they are not adopted usually. Throughout the remainder of this standard, the algorithm specified here in will be referred to as "the AES algorithm" [1]. The algorithm may be used with the three different key lengths indicated above, and therefore these different "flavours" may be referred to as "AES-128", "AES-192", and "AES-256". This standard may be used by Federal departments and agencies when an agency determines that sensitive (unclassified) information requires cryptographic protection. Other FIPS-approved cryptographic algorithms may be used in addition to, or in lieu of, this standard. Federal agencies or departments that use cryptographic devices for protecting classified information can use those devices for protecting sensitive (unclassified) information in lieu of this standard [2]. In addition, this standard may be adopted and used by non-Federal Government organizations.

ICGPC 2014
St.Peter's University, TN, India.

Such use is encouraged when it provides the desired security for commercial and private organizations. The algorithm specified in this standard may be implemented in software, firmware, hardware, or any combination thereof. The specific implementation may depend on several factors such as the application, the environment, the technology used, etc [4]. In advanced encryption standard a plain text is converted into cipher text. Cipher stands for a Series of transformations that converts a plaintext to cipher text using the Cipher Key. The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The (AES) algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called cipher text; decrypting the cipher text converts the data back into its original form, called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits [3].

Table.1: AES Algorithm

AES keys	Key Length	Block size	No of rounds
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

The AES algorithm takes the Cipher Key, K, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of Nb (Nr + 1) The algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data [5].

Key Expansion includes the following functions:

(1)Root Word (2) Sub Word (3) Rcon [i/NK]

The input and output for the AES algorithm each consist of sequences of 128 bits (digits with values of 0 or 1). These sequences will sometimes be referred to as blocks and the number of bits they contain will be referred to as their length. The Cipher Key for the AES algorithm is a sequence of 128, 192 or 256 bits.

Other input, output and Cipher Key lengths are not permitted by this standard [6]. The bits within such sequences will be numbered starting at zero and ending at one less than the sequence length (block length or key length). The number i attached to a bit is known as its index and will be in one of the ranges 0 <i<128,0<i<192 or 0 < i < 256 depending on the block length and key length.

II. FUNDAMENTALS OF AES

The AES has four steps through which it accomplishes data security while transferring any data from sender to receiver. The steps are illustrated as follows [7]

2.1 Sub Bytes Transformation:

The Sub Bytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box which is invertible is constructed by composing two transformations:
1. Take the multiplicative inverse in the finite field GF (2^8) and the element {00} is mapped to itself.
2. Apply the following affine transformation the bit, bi is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte c with the value {63} or {01100011}. Here and elsewhere, a prime on a variable indicates that the variable is to be updated with the value on the right [10].

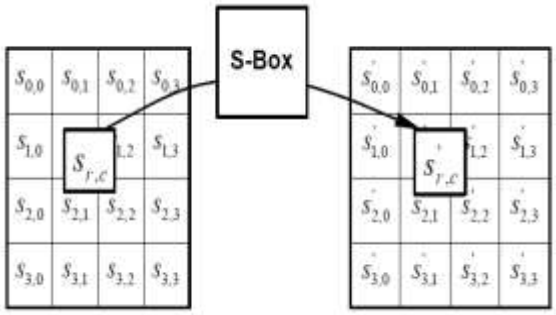

Fig .1 Sub Bytes applies the s-box to each Byte of the State

2.2 Shift Row Transformation:

In the Shift Rows () transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, r = 0, is not shifted. the shift value shift(r, Nb) depends on the row number, r, as follows (Nb = 4): shift(1,4); shift(2,4); shift(3,4) shift(5.4) This has the effect of moving bytes to "lower" positions in the row (i.e., lower values of c in a given row), while the "lowest" bytes wrap around into the "top" of the row (i.e., higher values of c in a given row) [8].

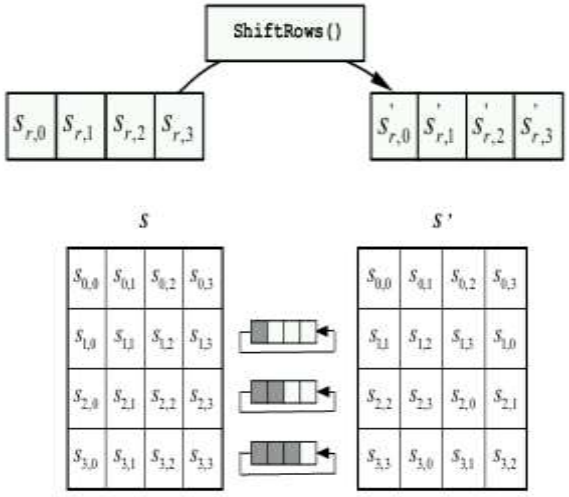

Fig.2 Shift Rows Cyclically Shifts The Last Three Rows In The State.

2.3 Mix Column Transformation:

The Mix Columns transformation operates on the State column-by-column, treating each column as a four-term polynomial [12]. The columns are considered as polynomials over GF (28) and multiplied modulo x4 + 1 with a fixed polynomial a(x), given by a(x)={03}x3 + {01}x2 + {01}x + {02}

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb.$$

Fig. 3 Add round Key Transformation.

2.4 Add Round Key:

In the Add Round Key transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of Nb words from the key schedule Those Nb words are each added into the columns of the State, such that In the Cipher, the initial Round Key addition occurs when round = 0, prior to the first application of the round function. The application of the Add Round Key transformation to the Nr rounds of the Cipher occurs [9].

2.5 Key Expansion:

The AES algorithm takes the Cipher Key, K, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of Nb (Nr + 1) words: the algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [wi], with i in the range 0 < i < Nb (Nr + 1) [11].

The expansion of the input key into the key schedule proceeds according to the pseudo code. Sub Word is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function Rot Word takes a word [a0, a1, a2, a3] as input, performs a cyclic permutation, and returns the word [a1, a2, a3, a0]. The round constant word array, Rcon[i], contains the values given by [xi-1,{00},{00},{00}], with x i-1 being powers of x (x is denoted as {02}) in the field GF(2^8), (note that i starts at 1, not 0).It can be seen that the first Nk words of the expanded key are filled with the Cipher Key [13]. Every following word, wi, is equal to the XOR of the previous word, wi-1, and the word Nk positions earlier, wi-Nk. For words in positions that are a multiple of Nk, a transformation is applied to wi-1 prior to the XOR, followed by an XOR with a round constant, Rcon[i]. This transformation consists of a cyclic shift of the bytes in a word (Rot Word), followed by the application of a table lookup to all four bytes of the word (Sub Word). It is important to note that the Key Expansion routine for 256-bit Cipher Keys (Nk = 8) is slightly different than for 128- and 192-bit Cipher Keys. If Nk = 8 and i-4 is a multiple of Nk, then Sub Word is applied to wi-1 prior to the XOR [12].

Similarly while obtaining data at the receiving end, AES performs four steps to decrypt data. They are mentioned in the following steps,

2.6 Inv Sub Bytes Transformation:

Inv Sub Bytes is the inverse of the byte substitution transformation, in which the inverse S box is applied to each byte of the State [14].

2.7 Inv Shift Rows Transformation

Inv Shift Rows is the inverse of the Shift Rows transformation. The bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, r = 0, is not shifted. The bottom three rows are cyclically shifted by Nb shift(r, Nb) bytes, where the shift value shift(r, Nb) depends on the row number [15],

2.8 Inv Mix Coloumn Transformations:

Inv Mix Columns is the inverse of the Mix Column transformation. Inv Mix Columns operates on the State column-by-column, treating each column as a four term polynomial. The columns are considered as polynomials over GF(28) and multiplied modulo x4 + 1 with a fixed polynomial a-1(x), given by a-1(x) = {0b}x3 + {0d}x2 + {09}x + {0e} [16].

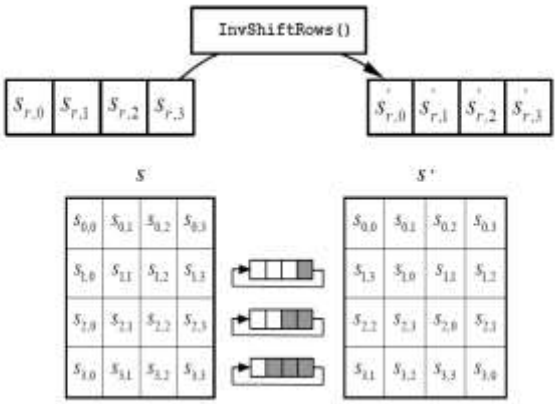

Fig. 4 Inv Shift Rows Transformation.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb.$$

Fig. 5 Inv-Mix Columns Transformation

2.9 Inverse Add Round Key Transformation:

It only involves an application of the XOR operation in cipher text. As the XOR operation is performed to the output of mix column transformation using, a round Key, the original input being send is obtained from the cipher text which is a encrypted protected form of the plain input text or data [17].

III. PREVIOUS WORK

In the existing architecture, complex mix-column in masked AES architecture is used. It consists of 8 multipliers and 11 adder operators. More number of adders and multipliers makes the system more complex and induces delay in the system. Complexity in system area too induces delay. In the Boolean masking implementation, the intermediate value x is concealed by exclusive-O Ring it with the random mask m. In the round function of the AES, Shift Rows, Mix- Columns, and Add Round Key are linear transformations, while Sub Bytes is the only nonlinear transformation of the AES. We define the linear transformations as Operation; then, the masked Operation can be written as Oper(x ⊕ m) = Oper(x) ⊕ Oper(m). However, the masked nonlinear transformation Sub Bytes has the characteristic as S-box(x ⊕ m) _= S-box(x) ⊕ S-box (m). In order to mask the nonlinear transformation, a new S-box, denoted as S-box_, is recomputed as S-box_ (x

⊕ m) = S-box(x) ⊕ m_, where mandm_ are the input and output masks of Sub Bytes [18].

The existing mix column architecture is given as follows. The diagram shows the complexity of the mix column architecture.

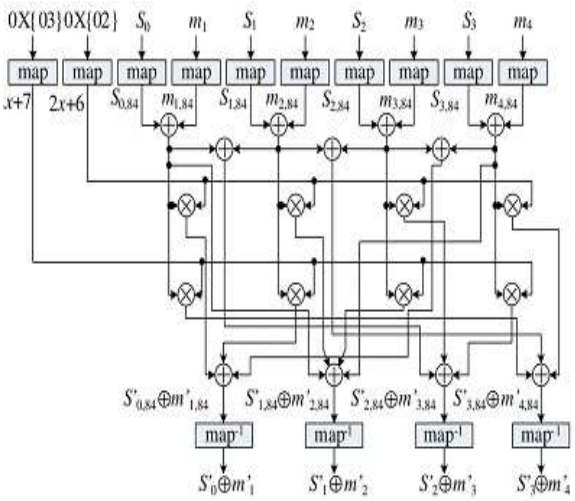

Fig. 6 Existing Mix-Column Architecture

Masked Mix Columns can be scaled to adjust the operations over GF(24), 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 Mix Columns, it holds that S = map (Sh Sl) ∼= Shx + Sl, where S ∈ GF (28) and Sh, Sl ∈ GF (24). Therefore, scaling factors 2x + 6 and 2x + 7 of S equal to (4Sh + 2Sl x + (fSh + 6Sl) and (5Sh + 2Sl)x + (fSh + 7Sl).The most optimized design was the work of Standaert et al., in which they achieved the best throughput/slices(4.2) on FPGA among the existing designs [6]. To our best knowledge, there have been no other works on the high-throughput masked AES on FPGA platforms.

In order to have a fair comparison, we also implement the masked AES design using Oswald's masked S-box [12] with non-pipelined and six stage pipelined unrolled structures. The area-optimized design with the pipelined structure proposed in this brief achieves 36.2% area reduction compared with Oswald's pipelined design. It occupies just 53% larger area than that of the unprotected baseline design with the pipelined structure. Furthermore, in order to compare the capability of defending DPA attacks by the unprotected AES and the masked AES. The correlation curve of the first Byte has the highest correlation than other values (not equalling to (0x13). Consequently, we can obtain all guessed 16 bytes of the last round's key for the unprotected AES and the masked AES [19].

Optimization for Proposed Architecture Usually, throughputs can be significantly improved by inserting pipeline registers for latency careless designs. For each Masked AES's round, we insert six-stage pipelines to enhance the throughputs. We insert three pipelines to each round of the masked AES, called outer three pipelines. The pipeline registers are inserted at the output of each transformation. We insert three pipelines to the masked S-box, called inner three pipelines. That the maximum pipelined stages for our proposed 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 [20].

IV. PROPOSED MASKED AES FOR UNROLLED STRUCTURE

The proposed AES mix column is changed to get less area and delay than the existing mix column. The proposed mix column architecture is reduced using X-time circuit. In the existing architecture, we used complex mix-column applied in masked (AES) architecture [21]. It consists of 8 multipliers and 11 adders. It consume more area and power. In the proposed method, a novel mix-column is introduced to reduce the area and power than the existing method. The proposed mix-column consists of only 12 adders and 4 X-time Circuit. Instead of normal multiplier in the mix-column, we are using X-time unit to reduce the circuit complexity.

The AES specifies FIPS-(Federal information processing standard) approved cryptographic algorithm that can be used to protect electronic data. The (AES) algorithm is a cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an intangible form called cipher text; decrypting the cipher text converts the data back into its original form, called plaintext. Using multipliers (8nos) and adder operators (11nos), will make the AES mix-column system more complex, leading to the consumption of more area and delay in execution. Instead, adders (12nos) and X-time circuit (4nos) are used.

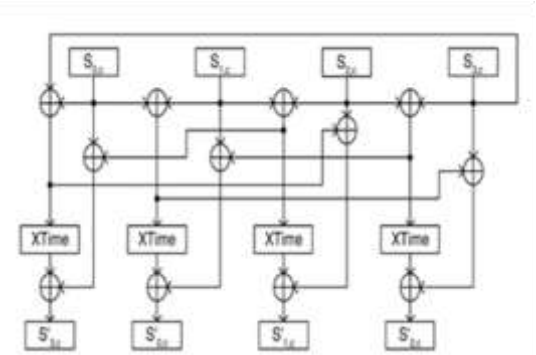

Fig. 7 Proposed complexity reduced mix Column Architecture.

V. RESULTS AND DISCUSSIONS

The proposed AES with optimized MixColumns is designed using Verilog HDL. Simulation is carried out by ModelSim6.3c and Synthesize is carried out by Xilinx10.1. The results are tabulated as shown in the below table2. Proposed Encryption and Decryption offers low area, less

delay and low power utilization than the conventional AES encryption and Decryption.

Table 2. Comparison of different AES techniques.

Different AES techniques	Slices	Delay (ns)	Power (w)
Encryption with complex MixColumns	584	3.557	21.345
Encryption with optimized MixColumns	474	3.509	18.450
Decryption with complex MixColumns	959	4.729	43.683
Decryption with optimized MixColumns	865	4.640	12.565

VI. CONCLUSION

This paper proposes an area efficient, high speed and low power modified AES with Optimized MixColumns. The proposed AES provide low area and less power by use of Optimized MixColumns. Both the design is implemented on Xilinx Virtex4 XC4VLX25 FPGA device. Comparative study of the conventional AES with complex Mix Columns and proposed AES with optimized MixColumns was done. The AES with optimized MixColumns as compared to AES with complex Mix Columns shows much more improvement in device utilization by way of reducing the area, delay and power. The proposed method offers 25% area and 10% power reduction than the existing architecture. Hence it is concluded that an efficient method for reducing the power dissipation and area is achieved by using a AES with optimized Mix Columns.

REFERENCES

[1] *Advanced Encryption Standard (AES)*, FIPS-197, Nat. Inst. of Standards and Technol., 2001.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. CRYPTO*, 1999, vol. LNCS 1666, pp. 388–397.

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

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

[5] 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.

[6] F. X. Standaert, G. Rouvroy, J. J. Quisquater, and J. D. Legat, "Efficient implementation of Rijndael encryption in reconfigurable hardware: Improvements and design tradeoffs," (in German), in *Proc. CHES LNCS*, 2003, vol. 2779, pp. 334–350.

[7] G. P. Saggese, A.Mazzeo, N. Mazzocca, and A. G.M. Strollo, "An FPGAbased performance analysis of the unrolling, tiling, pipelining of the AES algorithm," in *Proc. FPL LNCS*, Aveiro, Portugal, 2003, vol. 2778, pp. 292–302.

[8] M. McLoone and J. V. McCanny, "Rijndael FPGA implementations utilizing look-up tables," in *Proc. IEEE Workshop Signal Process. Syst.*, Antwerp, Belgium, 2001, pp. 349–360.

[9] V. Rijmen, "Efficient Implementation of the Rijndael S-Box," Dept. ESAT., Katholieke Universiteit Leuven, Leuven, Belgium, 2006. [Online].

Available: http://www.networkdls.com/Articles/sbox.pdf

[10] 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.

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

[12] 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.

[13] H. Kim, S. Hong, and J. Lim, "A fast and provably secure higher-order masking of AES S-box," in *Proc. CHES LNCS*, Nara, Japan, 2011, vol. 6917, pp. 95–107.

[14] C. Carlet, L. Goubin, E. Prouff, M. Quisquater, and M. Rivain, "Higherorder masking schemes for S-boxes," in *Proc. FSE LNCS*, 2012, vol. 7549, pp. 366–384.

[15] J. D. Goli´c, "Techniques for random masking in hardware," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 2, pp. 291–300, Feb. 2007.

[16] D. Canright and L. Batina, "A very compact 'perfectly masked' S-box for AES," in *Proc. ACNS LNCS*, 2008, vol. 5037, pp. 446–459.

[17] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. New York: Spinger-Verlag, 2007.

[18] 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.

[19] M. Alam, S. Ghosh, M. J. Mohan, D. Mukhopadhyay, D. R. Chowdhury, and I. S. Gupta, "Effect of glitches against masked AES S-box implementation and countermeasure," *IET Inf. Security*, vol. 3, no. 1, pp. 34–44, Feb. 2009.

[20] E. Trichina, T. Korkishko, and K. H. Lee, "Small size, low power, side channel-immune AES coprocessor: Design and synthesis results," in *Proc. AES LNCS*, 2005, vol. 3373, pp. 113–127.

[21] S. K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S. K. Hsu, H. Kaul, M. A. Anders, and R. K. Krishnamurthy, "53 Gbps native *GF*(24)2 composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance m