Reconfigurable architecture of Advanced Encryption Standard with Improved Performance

S.G.Itkalkar
Department of E&TC
GHRIET Wagholi.
Pune, India
s.itkalkar@raisoni.net

N.B.Hulle
Department of E&TC
GHRIET Wagholi.
Pune, India
nagnath.hulle@raisoni.net

V.A.Suryawanshi
Department of E&TC
GHRIET Wagholi
Pune, India
vishnu.suryawanshi@raisoni.net

Abstract—The Advanced Encryption Standard (AES) is a secret key cryptographic algorithm used for secure data transmission. AES has four major transformations, out of that Sub-byte transformation is the most expensive one in terms of area and Mix-column transformation in terms of speed. Proposed paper presents hardware implementation of Sub-bytes transformation using combinational logic and Read Only Memory (ROM) base. As compared to the typical ROM based lookup table, the Combinational Logic implementation is capable of higher speeds since it can be pipelined and small in terms of area occupancy.

Index Terms— AES,FPGA,VHDL,ROM,Look Up Table,Combnational Logic and S-box.

1 Introduction

The rapid growths of the Internet as a vehicle for secure communication and Electronic system has brought cryptographic processing performance to the forefront of high throughput and compact system design. The most popular and widely used symmetric key encryption algorithm is Advanced Encryption Standard.

National Institute of Standard and Technology (NIST) agreed on the block-based cipher system, data encryption standard (DES) in 1970. Since the DES cipher system was worn-out by violence attack methods, then NIST announced the new DES algorithm, which is called the Triple DES (3DES). The 3DES uses the same algorithm as the DES and increases the difficulty of illegal breaks. But the 3DES algorithm exists two disadvantages. First, the 3DES requires three times more execution cycles than the DES, so the execution efficiency of 3DES is not good enough. Second, both DES and 3DES only use a 64-bit length block data, the security and safety are not enough. Due to the two disadvantages of DES and 3DES, the NIST gave up the DES cipher systems and then asked for a new generation cipher system, which was called the advanced encryption standard(AES) in 1997.

On January 2, 1997, The National Institute of Standards and Technology (NIST) published a request for comments for the "Development of a Federal Information Processing Standard for Advanced Encryption Standard." NIST sought to "consider alternatives that offer a higher level of security than that offered by the Data Encryption Standard (DES), which grew vulnerable to brute-force attacks due to its 56-bit effective key length. Fifteen AES candidate algorithms were announced in August, 1998.

Five finalists were chosen on August 9, 1999 named as MARS, RC6, RIJNDAEL, SERPENT and TWO FISH. Advanced encryption standard discovered by two cryptography researchers Vincent Rijmen and Joan Daeman.

This cipher has been accepted by Federal Information Processing Standards (FIPS) in November 2001 by National Institute of Standards and Technology (NIST) [1].

The name "Rijndael" is based on the algorithm's authors' names, Joan Daemen and Vincent Rijmen. In English, it is pronounced "Rhine Dahl." Rijndael is based on the Square Cipher. By using hardware, a higher data rate for fast applications such as routers can be achieved compared to Software implementation. As well as the hardware implementation is also physically secure since tempering by an attacker is very difficult. The efficiency of AES hardware implementation in terms of area, throughput security and power consumption depends largely on the AES architecture [2].To save power and area feedback logic methods are used where as to increase the speed loop unrolling method is used.[3]

This paper is organized in following sections. Section II AES algorithm overview, section III is on Related Work of Sub-bytes on AES, section IV shows hardware implementation of Sub-bytes, V is results and discussion and section VI concludes the paper.

2 AES Algorithm Overview

The AES algorithm uses fixed length of data of 128 bits and a variable key length of 128,192 and 256 bits. The execution rounds for encryption and decryption are 10, 12 and 14 for 128 bit data and 128 bit key, 128 bit data and 192 bit key and 128 bit data and 256 bit key.

The architecture of AES consists of encryption, decryption and key expansion unit. In encryption the input data in terms of plain text can be encrypted in terms of cipher text. The decryption unit decrypts cipher text in terms of plain text. The AES algorithm is shown in fig.1

The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of it's counterpart in the encryption algorithm.

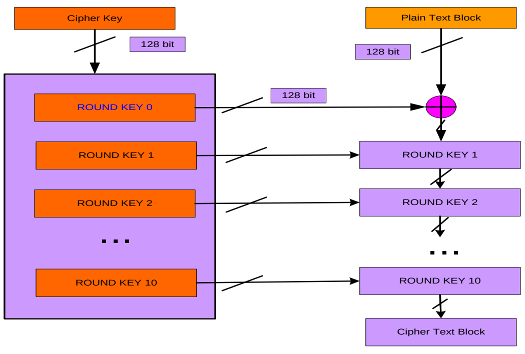

Fig 1 Basic Architecture of AES Encryption

The tenth round simply leaves out the Mix Columns stage. The first nine rounds of the decryption algorithm consist of the following: Inverse Shift rows, Inverse Substitute bytes, Inverse Add Round Key and Inverse Mix Columns. Again, the tenth round simply leaves out the Inverse Mix Columns stage. Each of these stages will now be considered in more detail. AES algorithm consists of: Add Round Key, Sub bytes, Shift rows and Mix column

Add Round Key xors the 128 bits of state are bit-wise with the 128 bits of the round key. The operation is viewed as a column wise operation between the 4 bytes of a state column and one word of the round key. This transformation is as simple as possible which helps in efficiency but it also affects every bit of state.

Sub Bytes stage (known as SubBytes) is simply a table lookup using a 16×16 matrix of byte values called an s-box. This matrix consists of all the possible combinations of an 8 bit sequence (28 = 16 × 16 = 256). However, the s-box is not just a random permutation of these values and there is a well defined method for creating the s-box tables. The designers of Rijndael showed how this was done unlike the s-boxes in DES for which no rationale was given.

In shift row the first row of state is *not* altered. The second row is shifted 1bytes to the left in a circular manner. The third row is shifted 2 bytes to the left in a circular manner. The fourth row is shifted 3 bytes to the left in a circular manner. The Inverse Shift Rows transformation (known as Inv-Shift-Rows) performs these circular shifts in the opposite direction for each of the last three rows (the first row was unaltered to begin with). The shift rows cyclic shift is shown in fig.2

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

$\Rightarrow$

1	2	3	4
6	7	8	5
11	12	9	10
16	13	14	15

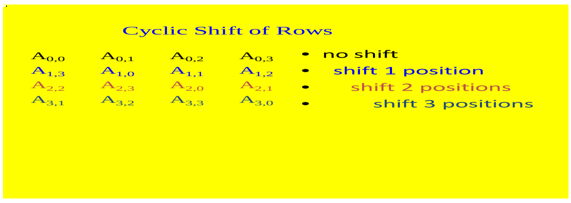
Cyclic Shift of Rows
A0,0 A0,1 A0,2 A0,3 • no shift
A1,3 A1,0 A1,1 A1,2 • shift 1 position
A2,2 A2,3 A2,0 A2,1 • shift 2 positions
A3,1 A3,2 A3,3 A3,0 • shift 3 positions

Fig 2 shows the cyclic shift rows operation.

Mix-column is basically a substitution but it makes use of arithmetic of GF(2^8). Each column is operated on individually. Each byte of a column is mapped into a new value that is a function of all four bytes in the column.

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{pmatrix} = \begin{pmatrix} r_0 & r_5 & r_9 & r_{13} \\ r_1 & r_6 & r_{10} & r_{14} \\ r_2 & r_7 & r_{11} & r_{15} \\ r_3 & r_8 & r_{12} & r_{16} \end{pmatrix}$$

Each element of the product matrix is the sum of products of elements of one row and one column. In this case the individual additions and multiplications are performed in GF(2^8). The Mix-Columns transformation of a single column j (0 <j< 3) of state can be expressed as:

$$r_0 = \{02.d4\} + \{03.bf\} + \{01.5d\} + \{01.30\}$$

$$r_1 = \{01.d4\} + \{02.bf\} + \{03.5d\} + \{01.30\}$$

$$r_2 = \{01.d4\} + \{01.bf\} + \{02.5d\} + \{03.30\}$$

$$r_3 = \{03.d4\} + \{01.bf\} + \{01.5d\} + \{02.30\}$$

where • denotes multiplication over the finite field GF(2^8). The Inv-Mix-Columns is defined by the following matrix multiplication:

Key Scheduling Unit

The key expansion unit is responsible for generation of keys for different rounds of encryption and decryption process. It consists of RCON (Round Constant), Sub-bytes and Rot word.

Initially the given key's last column is cyclically shifted upward by one bit then it is passed through the s-box operation. After passed through s-box it will be xored with round constant and finally it will xored with first column of given key which will be generated as first round keys first column. Then this result will be xored with second column of given key which will be produced as second column of first round key. The obtained result will be xored with third column of given key which will be act as third column of first round key. Finally this result will be xored with last column of given key which represents as final i.e. fourth column of first round key.

The first round key will act as a initial key for second round key thus the same procedure will be repeated for remaining rounds. Such keys generated and stored in BRAMS for encryption and decryption process for implementation. Key schedule is shown in fig 3 and 4.

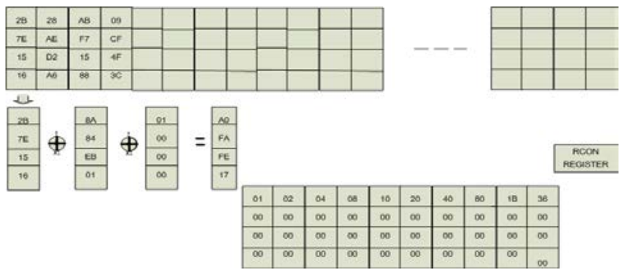

Fig 3 Key schedule for AES

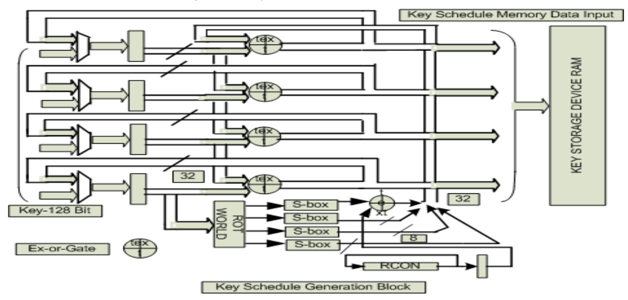

Fig.4 Key generation and memory storage block

3 Related Work on Sub-byte

Many researchers have presented their work on different hardware implementation Sub-byte.

In article [6] the author has presented his work on Application Specific Integrated Circuit (ASIC) Instead of using look-up tables for implementing the S-box, logic gate implementation based on a previously known low-complexity composite field using normal basis is utilized. They present improved formulations for the inversion in the sub-fields within the S-box to reduce the area complexity of the implementations. After analyzing the complexities of the new architecture, we compare the ASIC implementation of the proposed S-box using 0:18¹ CMOS technology.

The author of article [7] presented a concurrent fault detection scheme for the S-box and the inverse S-box as the only two nonlinear operations within the Advanced Encryption Standard. The proposed parity-based fault detection approach is based on the low-cost composite field implementations of the S-box and the inverse S-box. They divide the structures of these operations into three blocks and find the predicted parities of these blocks. Our simulations show that except for the redundant units approach which has the hardware and time overheads of close to 100 percent, the fault detection capabilities of the proposed scheme for the burst and random multiple faults are higher than the previously reported ones.

In article [8] the work presented by author is based on common sub expression elimination method. Design for the SB transformation, ISB transformation, as well as for the integrated SB and ISB (denoted by SB+ISB). In SB, ISB, or SB+ISB, MI over GF(2^8) is usually implemented in two ways: look-up table (LUT) and composite field. Some implementations use LUT for MI over subfields. It has been shown that composite field implementations outperform LUT-based methods in both area and CPD [9]. Many options for subfields and bases exist in the construction of composite fields. They can be divided into two categories that use polynomial or normal basis for the subfields, respectively.

The author of article [10] Most implementations conventionally make use of the memory intensive look up table approach for Substitute byte/Inverse Substitute Byte (SB/ISR) block implementations resulting in an unbreakable delay. The proposed work employs a memory-less combinatorial design for the implementation of SB/ISR as an alternative to achieve higher speeds by eliminating memory access delays while retaining or enhancing the over all area efficiency. The author categories all of the feasible constructions for the composite Galois field GF $(((2^2)^2)^2)$ (AES) S-box into four main architectures by their field representations and their algebraic properties. For each of the categories, a new optimization scheme which exploits algebraic normal form representation followed by a sub-structure sharing optimization is presented. This is performed by converting the subfield GF $((2^2)^2)$ inversion into several logical expressions, which will be in turn reduced using a common sub-expression elimination algorithm. The authors show that this technique can effectively reduce the total area gate count as well as the critical path gate count in composite field AES S-boxes. The resulting architecture that achieves maximum reduction in both total area coverage and critical path gate count is found [11].

In paper [12] the author proposed an architecture which employs a Boolean simplification of the truth table of the logic function with the aim of reducing the delay. The S-Box is designed using basic gates such as AND gate, NOT gate, OR gate and multiplexer. Theoretically, the design reduces the overall delay and efficiently for applications with high-speed performance. This approach is suitable for FPGA implementation in term of gate area.

4 Hardware Implementation for Sub-byte Using Look-Up Table and composite field

One of the most common and straight forward implementation of the S-Box for the SubByte operation which was done in previous work was to have the pre-computed values stored in a ROM based lookup table. In this implementation, all 256 values are stored in a ROM and the input byte would be wired to the ROM's address bus. However, this method suffers from an unbreakable delay since ROMs have a fixed access time for its read and write operation.[13]Furthermore, such implementation is expensive in terms of hardware.

The Sub Bytes transformation calculates non-linear bytes transformation under GF (2^8). The S-box can be implemented using combinational logic circuit access delays to achieve higher speeds by eliminating memory. The sub byte operation proposed in this architecture is implemented using look–up table method and using composite field. In the look-up table input data of 8 bit can be read from s-box table. The Sub Bytes process is divided into two steps. In the first step, 8-bit inverse multiplication in GF (2^8) is calculated, and then in the second step, the affine transform is used to get the output results.

Sub-Byte transformation is a nonlinear substitution that operates on individual bytes using a substitution table (S-box), which contains a permutation of all 256 possible 8-
bit values. S-Box is defined as the multiplicative inverse in the finite field GF(2^8) with the irreducible polynomial $m(x)=x^8+x^4+x^3+x+1$ followed by an affine transformation. The operation of the S-box can be expresses as

y = M * multiplicative_inverse(x) + c,

Where M is an 8x8 binary matrix and C is a 8-bit binary vector.

$$M = \begin{pmatrix} 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \\ 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \end{pmatrix}$$

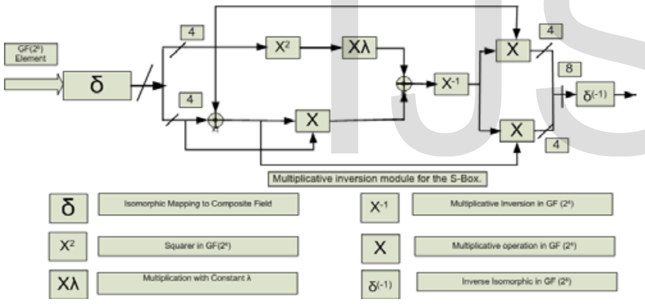
GF(2⁸) Element
δ
4
4
X²
Xλ
X⁻¹
X
X
X
4
8
4
δ(-1)
Multiplicative inversion module for the S-Box.
δ
Isomorphic Mapping to Composite Field
X²
Squarer in GF(2⁴)
Xλ
Multiplication with Constant λ
X⁻¹
Multiplicative Inversion in GF (2⁴)
X
Multiplicative operation in GF (2⁴)
δ(-1)
Inverse Isomorphic in GF (2⁸)

Fig 5. Implemented hardware architecture on the FPGA

5 Results and Discussion

The hardware implementation of S-box for ROM and Combinational logic is carried on Spartan 3E XC3S500E using Xilinx software. The VHDL code is used. Comparative analysis of hardware implementation of S-box for ROM and Combinational logic is shown in table 1.The RTL schematic of s-box implementation using ROM based and Combinational logic is shown in fig 6 and 7. Output waveforms for ROM based approach and Combinational is shown in fig 8 and 9.

Fig 6 RTL Schematic for S-box using combinational logic.

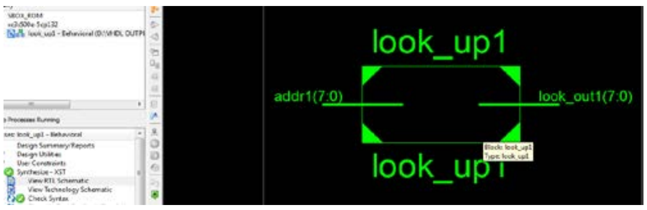

Fig 7 RTL Schematic for ROM based approach.

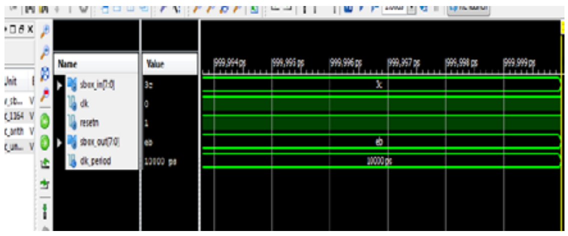

Fig.8.Output waveforms for S-box combinational logic

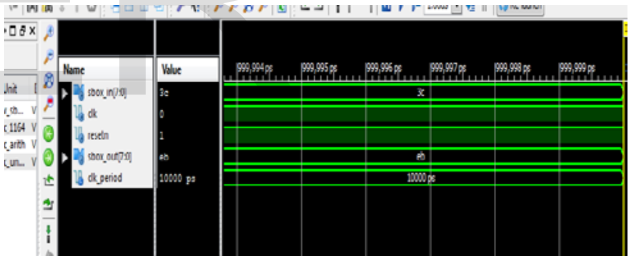

Fig.9.Output waveforms for S-box combinational logic

Table 1

Sr. No	Parameter	ROM	Combinational Logic
1	Number of Slices	64	38
2	Number of 4 input LUTs	128	75
3	Number of Bounded IOBs	16	18

Using Combinational logic:

Minimum period: 6.442ns (Maximum Frequency: 155.229MHz)
Minimum input arrival time before clock: 1.731ns
Maximum output required time after clock: 4.040ns.

6 Conclusion

A combinational logic based S-Box and ROM based approach for the Sub Byte transformation is discussed and its internal operations are explained. As compared to the typical ROM based lookup table, the Combinational Logic implementation is both capable of higher speeds since it can be pipelined and small in terms of area occupancy. The Combinational Logic compact and high speed architecture allows the S-Box to be used in both areas limited and demanding throughput AES chips for various applications, ranging from small smart cards to high speed servers

References:

[1]	National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES), 2001. FIPS-197.
[2]	A.Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-BoxOptimization," in Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology: Springer-Verlag, 2001.
[3]	N. Sklavos and O. Koufopavlou, "Architectures and VLSI implementations of the AES-Proposal Rijndael," Computers, IEEE Transactions on, vol. 51, pp. 1454-1459, 2002.
[4]	A J. Menezes, P. C. van Oorschot and S A. Vanstone Handbook of Applied Cryptographic Research, CRC Press, 1996
[5]	M. Dworkin, NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation – Methods and Techniques, National Institute of Standards and Technology, December 2001.
[6]	Mehran Mozaffari-Kermani and Arash Reyhani-Masoleh "A Low-Cost S-box for the Advanced Encryption Standard UsingNormal Basis" 978-1-4244-3355-1/09 2009 IEEE
[7]	Mehran Mozaffari-Kermani, and Arash Reyhani-Masoleh, "A Low-Power High-Performance Concurrent Fault Detection Approach for the Composite Field S-Box and Inverse S-Box" IEEE TRANSACTIONS ON COMPUTERS, VOL. 60, NO. 9, SEPTEMBER 2011
[8]	Ning Chen and Zhiyuan Yan "Compact Designs of MixColumns and SubBytes Using a Novel Common Subexpression Elimination Algorithm" 978-1-4244-1684-4/08 IEEE
[9]	S.-F. Hsiao, M.-C. Chen, M.-Y. Tsai, and C.-C. Lin, "System on-chip implementation of the whole advanced encryption standard processor using reduced XOR-based sum-of-product operations," *IEE Proc. Inf. Secur.*, vol. 152, no. 1, pp. 21–30, Oct. 2005.
[10	Nalini C,Dr. Anandmohan P.V, Poomaiah D.V, and V.D.kulkami "Compact Designs of SubBytes and MixColumn for AES" 2009 WEE International Advance Conputing Conference (IACC 2009) Patialae, India, 6-7 Ma-crch 2009
[11	M.M. Wong, M.L.D. Wong, A.K. Nandi, I. Hijazin "Composite field GF(((2^2)^2)^2) Advanced Encryption Standard (AES) S-box with algebraic normal form representation in the subfield inversion" Published in IET Circuits, Devices & Systems Revised on 18th May 2011 doi: 10.1049/iet-cds.2010.0435
[12	Nabihah Ahmad Rezaul Hasan Warsuzarina Mat Jubadi "Design of AES S-Box using combinational logic Optimization"2010 IEEE symposium on Industrial Electronics and applications (ISIEA-2010) Oct 3-5 2010 Penang Malaysia
[13	Edwin NC Mui "Practical Implementation of Rijndael S-Box Using Combinational Logic"