Ijesm
Consulting, help, relaxation
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & MANAGEMENT

# A VHDL Implementation of Low Area Advance Encryption Standard Processor

**Nimmi Gupta**
Lakshmi Narain College of Technology, Bhopal(M.P), India

**ABSTRACT**

**In this work our aim to achieve a high through put compact. AES S-Box with minimum area consumption. To improve architectures are proposed for implementation of S-Box and Inverse S-box needed in the Advanced Encryption Standard (AES). Unlike previous work which rely on look-up table to implement the Subbytes and Invsubbytes transformations of the AES algorithm the proposed design employs Combinational logic only for implementing Subbytes (S-Box) and InvsubBytes (Inverse S-Box). The resulting hardware requirements are presented for proposed design and compared by ROM- based and Pre-Computation technique and improve with this two technique a new technique is Galois field arithmetic.**

*Keywords*- **Advanced Encryption Standard, VLSI architectures, Data Encryption, S-Box, Sub-byte Encryption**

## INTRODUCTION

The AES algorithm is a symmetric block cipher that processes data blocks of 128 bits using a cipher key of length 128, 192, or 256 bits. Each data block consists of a 4 × 4 array of bytes called the *state*, on which the basic operations of the AES algorithm are performed. The AES encryp- tion/decryption procedure is shown in Fig. 1. After an initial round key addition, a round func- tion consisting of four different transformations—*SubBytes()*, *ShiftRows()*, *MixColumns()*, and *AddRoundKey()* — is applied to the data block (i.e., the state array).

**Corresponding Author***
*Email- nimmi.gupta877@gmail.com*

The round function is per- formed iteratively 10, 12, or 14 times, depending on the key length. Note that in the last round MixColumns() is not applied. The four transfor- mations are described briefly as follows [1]:

- SubBytes():a nonlinear byte substitution that operates independently on each byte of the state using a substitution table (the S- Box)
- ShiftRows():a circular shifting operation on the rows of the state with different numbers of bytes (offsets)
- MixColumns(): the operation that mixes the bytes in each column by the

multiplication of the state with a fixed polynomial modulo $x^4 + 1$.

- AddRoundKey(): an XOR operation that adds a round key to the state in each itera- tion, where the round keys are generated during the key expansion phase smoothly changing the other properties. The major properties of concern as far as a speech signal is concerned are its pitch and envelope information.

The decryption procedure of the AES is basi- cally the inverse of each transformation (*InvSub- Bytes()*, *InvShiftRows()*, *InvMixColumns()*, and *AddRoundKey()*) in reverse order. However, the order of *InvSubBytes()* and *InvShiftRows()* is indifferent. The decryption procedure thus can be rearranged as shown in Fig. 1, where the InvRoundKeyis obtained by applying *InvMix- Columns()* to the respective original RoundKey [1]. Such a structural similarity in both the encryption and decryption procedures makes hardware implementation easier.

The SubBytes() transformation (S-Box operation) which consists of amultiplicative inverse over $GF(2^8)$ and an affine transform, is the most critical part of the AES algorithm in terms of computational complexity. However, the S-Box operation is required for both encryption and key expansion. Conventionally, the coefficients of the S-Box and inverse S-Box are stored in the LUTs, or a hard-wired multiplicative inverter over $GF(2^8)$ can be used, together with an affine transform circuit. A dedicated inverter, however, has a high area overhead. We propose an effi- cient implementation by a transformation of the S-Box over the finite field.

## METHODOLOGY

*Composite Field Arithmetic*

The non-LUT-based implementations of the AES algorithm are able to exploit the advantage of subpipelining further. Never- theless, these approaches may have high hardware complexities. Although two *Galois Fields* of the same order are isomorphic, the complexity of the field operations may heavily depend on the representations of the field elements. Composite field arith- metic can be employed to reduce the hardware complexity. We call two pairs

$$\{GF(2^n), Q(y) = y^n + \Sigma_{i=0}^{n-1} q_i y^i, q_i \in GF(2)\}$$
$$\{GF((2^n)^m), P(x) = x^m + \Sigma_{i=0}^{m-1} p_i x^i, p_i \quad GF(2^n)\}$$

and a composite field [12] if

- GF($2^n$) is constructed from by GF(2) Q(y)
- GF (($2^n$)$^m$) is constructed from by GF($2^n$) P(x).

Composite fields will be denoted by , GF (($2^n$)$^m$) and a composite field GF (($2^n$)$^m$) is isomorphic to the field GF($2^k$) for $k$=nm. Additionally, composite fields can be built iteratively from lower order fields. For example, the composite field GF($2^8$ ) of can be built iteratively from GF(2) using the following irreducible polynomials [7]:

GF $(2^2) \rightarrow$ GF$(2)$ $: x^2 + x + 1$

GF$((2^2)^2) \rightarrow$ GF $(2^2)$ $: x^2 + x + \varphi$

GF$(((2^2)^2)^2) \rightarrow$ GF$((2^2)^2)$ $: x^2 + x + \lambda$

Where $\varphi = \{10\}_2$ and $\lambda = \{1100\}_2$

Meanwhile, an isomorphic mapping function $f(x)$ and its inverse need to be applied to map the representation of an element in GF($2^8$ ) to its composite field and *vice versa*. The 8 × 8 binary matrix are decided by the field polynomial GF($2^8$ ) of and its

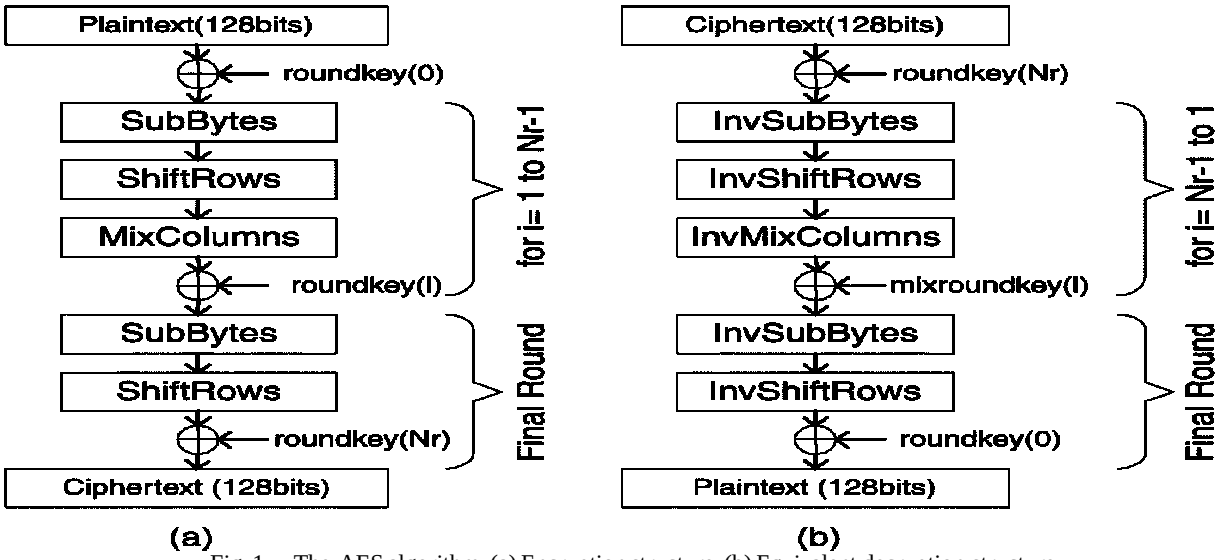


Fig. 1. The AES algorithm. (a) Encryption structure. (b) Equivalent decryption structure.

composite fields. Such a matrix can be found by the exhaustive-search- based algorithm in [12].The $\delta$ matrix corresponding to $p(x)= x^8+x^4+x^3+x+1$ and the field polynomials in (7) can be found as below:

$$\delta = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Taking the isomorphic mapping into consideration, not all the transformations in the AES algorithm are suitable to be implemented in the composite field. In order to fa- cilitate substructure sharing, the constant multiplications in the MixColumns/InvMixColumns transformation are im- plemented by first computing $\{02\}_{16}$ $S_{i,j}$ $\{04\}_{16,}$ $S_{i,j}$ and $\{08\}_{16,}$ $S_{i,j}$ then adding those terms corresponding to the nonzero bits in the constants. For example, the constant mul- triplication of $\{0b\}_{16}$ $=\{00001011\}_2$ can be computed by adding $S_{i,j,}$ $\{02\}$ $S_{i,j}$ and $\{08\}_{16}$ $S_{i,j}$. In this approach, the $\{02\}_{16}$ $\{02\}_{16}$ $S_{i,j}$ $\{04\}_{16,}$ $S_{i,j}$ and $\{08\}_{16,}$ $S_{i,j}$ can be computed by adding.In this approach, the and can be computed once and shared by all the constant multiplications. Meanwhile, the number of terms, which need to be added is determined by the number of nonzero bits in the constants. Using the δ matrix defined in (8), constant multiplications of $\{02\}_{16}$ and $\{03\}_{16}$ in $GF(2^8)$ in the MixColumns are mapped to constant multiplicatios of $\{5f\}_{16}$ and $\{5e\}_{16}$ in the composite field, respectively. Although the hardware overhead of the mapping of constants can be eliminated by computing the mapping beforehand, the composite field representations of $\{02\}_{16}$ and $\{03\}_{16}$ have more nonzero bits, which makes the constant multiplications more expensive. The same argument also holds for the constant multiplications used in the InvMix-Columns transformation, where $\{09\}_{16,}$ $\{0b\}_{16,}$ and $\{0e\}_{16}$ are mapped to $\{75\}_{16,}$ $\{2\alpha\}_{16,}$ and $\{57\}_{16}$ in the composite field, respectively. The only exception is that the composite field representation of $\{0d\}_{16,}$ which is $\{09\}_{16}$ has one less nonzero bit, but

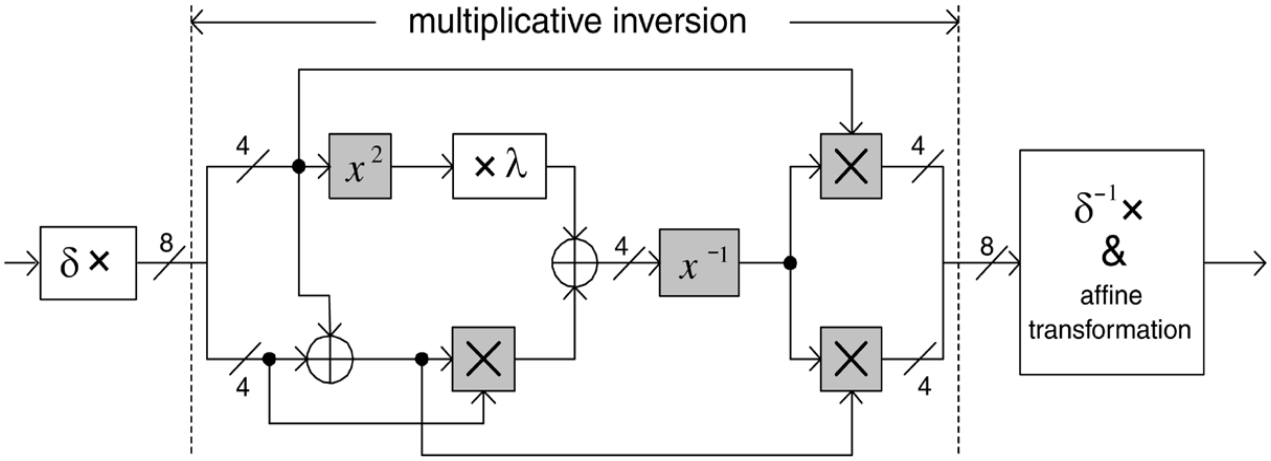


Fig. 3. Implementation of the SubBytes Transformation

this is offset by the larger number of nonzero bits in the composite field representations of the other three constants. Furthermore $\{10\}_{16}\, S_{i,j}$ $\{20\}_{16,}\, S_{i,j}$ and $\{10\}_{16}$ $S_{i,j}$ also need to be computed as a result of the higher weight nonzero bits in $\{75\}_{16,}$ $\{2\alpha\}_{16,}$ and $\{57\}_{16}$ which adds more complexity to the hardware implementations. Therefore, it is more efficient to implement the MixColumns/InvMixColumns in the original field GF($2^8$). The ShiftRows/InvShiftRows is a trivial transformation, only cyclical shifting is involved, and thus its implementation does not depend on the representa- tion of *Galois Field* elements. Meanwhile, the field addition, which is simply XOR operation, has the same complexity in the composite field and the original field. Additionally, the affine/inverse affine transformation can be combined with the inverse isomorphic/isomorphic mapping. Based on the above observations, it is more efficient to carry out only the multiplicative inversion in the SubBytes/InvSubBytes in the composite field, while keep the rest of the transformations in the original field GF($2^8$).

## SIMULATION AND RESULTS

In this Section, FPGA implementation and results of given architectures for S-box implemented on Xilinx XC3S400-5 device are listed. Xilinx ISE 9.2 is used to synthesize the design and provide post placement timing results. Table below showing the area consumed by various s-box architectures:

| SBOX | REGISTER | XOR |
|---|---|---|
| PCT-BASED | 133 | 954 |
| GF- BASED | 92 | 100 |

Table 1: Comparison of Various S-box architecture

.

## CONCLUSION

In this paper, the AES algorithm are presented. In order to explore the SubBytes/InvSubBytes is imple- mented by combinational logic to avoid the unbreakable delay of LUTs in the traditional designs. Additionally, composite field arithmetic is used to reduce the hardware complexity and different approaches for the implementation of inversion in subfield are compared. As an example of our

proposed architecture, fully subpipelined encryptors using 128-bit key are implemented on FPGA devices. Decryptors can be easily incorporated by using the encryptor/decryptor round unit architecture presented in this paper, and we expect the throughput will be slightly lower than the encryptor-only implementations. Meanwhile, using other key lengths can be implemented by adding more copies of round units and modifying the key expansion unit slightly. Furthermore, the number of round units in a loop can be reduced to meet the requirements of small area applications.

**REFERENCES**

[1] National Bureau of Standards, Data Encryption Standard (DES), US Department of Commerce. Federal Information Processing Standards Publication 46 (FIPS PUB 46), 15 January 1977.

[2] RSA Security. RSA's DES Challenge III is solved in record time. Available at http://www.rsa.com/rsalabs/node.asp?id=2108 , 18 January 1999.

[3] National Institute of Standards and Technology, US Department of Commerce. Commerce Department announces winner of Global Information Security Competition. Available at October 2000 .http://www.nist.gov/public_affairs/releases/g00-176.cfm , 2

[4] National Institute of Standards and Technology, US Department of Commerce. Federal Information Processing Standards Publication 197 (FIPS PUB 197). Available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf , 26 November 2001.

[5] IBM Corp., IBM 4764 product and PCIXCC feature overview. Available at: http://www-03.ibm.com/security/cryptocards/pcixcc/overview.shtml, 2008.

[6] NXP Corp., NXP takes lead on security for contactless smart cards [Online]. Available at: http://www.nxp.com/news/content/file_1273.html, 2008

[7] Altera Corp., Stratix IV device handbook. Availableat: http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf , November, 2008.

[8] Altera Corp. Quartus II development software literature: Power-Play power analysis. Available at: http://www.altera.com/literature/hb/qts/qts_qii5v3_03.pdf, November, 2008.

[9] P. Hamalainen, M. Hannikainen and T. Hamalainen, "Review of Hardware Architectures for Advanced Encryption Standard Implementations Considering Wireless Sensor Networks," Embedded Computer Systems: Architectures, Modeling, and Simulation, pp. 443-453, 2007.

[10] D. Raths., Energy hogs on the server farm [Online]. Available at: http://www.govtech.com/pcio/102970?id=102970&full=1&story_pg=1 , December 19, 2006.