IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 4, November 2012)

Area optimized S-BOX Architecture for Advance Encryption Standards

Nimmi Gupta Tarun Verma

nimmi.gupta877@gmail.com *tarunindia@rediff.com*

***Abstract*— This paper presents an area optimized for composite field arithmetic based SubBytes transformation (Sbox) used in Advanced Encryption Standard (AES) encryption. The proposed architecture is based on pre-computation technique. Implementation is proposed on FPGA using Xilinx ISE on XC3S 400-5 and results are shown in the paper.**

***Keywords*—AES, S-BOX, Sub-Byte, Encryption.**

I. INTRODUCTION

On 2nd January 1997, the National Institute of Standards and Technology (NIST) invited proposals for new algorithms for the new Advanced Encryption Standard (AES). [1] The goal was to replace the older Data Encryption Standard (DES) which was introduced in November 1976 when DES was no longer secure. After going through 2 rounds of evaluation, Rijndael was selected and named the Advanced Encryption Standard algorithm on 26th November 2001. The AES algorithm has a fixed block size of 128 bits and a key length of 128, 192 or 256 bits. It generates its key from an input key using the Key Expansion function. The AES operates on a 4x4 array of bytes which is called a *state*. The state undergoes 4 transformations which are namely the AddRoundKey, SubByte, ShiftRow and MixColumn transformation. [4] The AddRoundKey transformation involves a bitwise XOR operation between the state array and the resulting Round Key that is output from the Key Expansion function. SubByte transformation is a highly non-linear byte substitution where each byte in the state array is replaced with another from a lookup table called an S-Box. ShiftRow transformation is done by cyclically shifting the rows in the array with different offsets. Finally, MixColumn transformation is a column mixing operation, where the bytes in the new column are a function of the 4 bytes of a column in the state array. Of all the transformation above, the SubByte transformation is the most computationally heavy. [3]

Figure below showing the block diagram for AES:

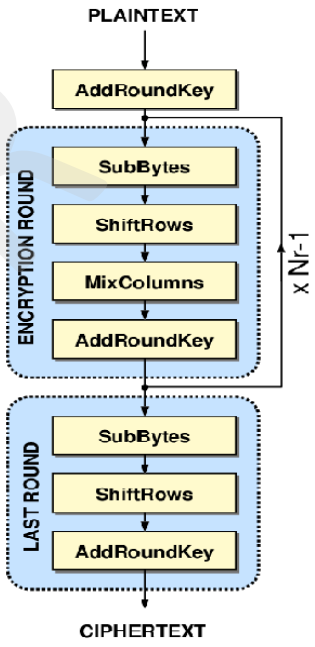

Figure 1: AES Block Diagram

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. [3] Furthermore, such implementation is expensive in terms of hardware. A more refined way of implementing the S-Box is to use combinational logic. Such examples of work that implements the S-Box using this method were [1], [3] and [5]. This S-Box has the advantage of having small area occupancy, in addition to be capable of being pipelined for increased performance in clock frequency.

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 4, November 2012)

The delay of the composite field arithmetic based S-box used by Zhang and Parhi [3] much less than the ROM based architecture. The S-box architecture therein is realized as a cascade of three blocks by combining certain operations in the Zhang and Parhi implementation. This paper is an FPGA implementation of S-BOX based on composite field arithmetic I GF(2^2) field to get optimized speed and area.

II. S-BOX ARCHITECTURE

SubByte transformation is a highly non-linear byte substitution where each byte in the state array is replaced with another from a lookup table called an S-Box. ShiftRow transformation is done by cyclically shifting the rows in the array with different offsets. Finally, MixColumn transformation is a column mixing operation, where the bytes in the new column are a function of the 4 bytes of a column in the state array Of all the transformation above, the SubByte transformation is the most computationally heavy. The SubByte transformation is computed by taking the multiplicative inverse in GF (2^8) followed by an affine transformation. For its reverse, the InvSubByte transformation, the inverse affine transformation is applied first prior to computing the multiplicative inverse. This section illustrates the steps involved in constructing the multiplicative inverse module for the S-Box using composite field arithmetic. Since both the SubByte and InvSubByte transformation are similar other than their operations which involve the Affine Transformation and its inverse, therefore only the implementation of the SubByte operation will be discussed in this paper. The multiplicative inverse computation will first be covered and the affine transformation will then follow to complete the methodology involved for constructing the S-Box for the SubByte operation.

The multiplicative inverse computation will be done by decomposing the more complex GF(2^8) to lower order fields of GF(2^1), GF(2^2) and GF((2^2)^2). Computation of the multiplicative inverse in composite fields cannot be directly applied to an element which is based on GF (2^8). That element has to be mapped to its composite field representation via an isomorphic function, δ. Likewise, after performing the multiplicative inversion, the result will also have to be mapped back from its composite field representation to its equivalent in GF(28) via the inverse isomorphic function, δ-1. Both δ and δ-1 can be represented as an 8x8 matrix. Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation between the 2 elements. Other functions as shown in figure below can be also be realized in field of GF (2^2) and GF (2^4).

Figure below showing the s-box architecture:

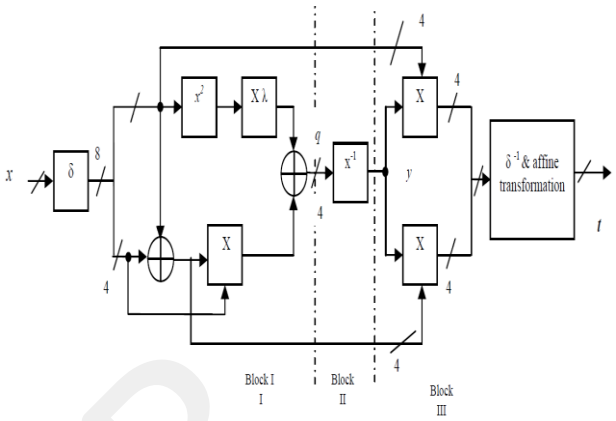

Figure 1. SBOX Architecture

The work flow of the this architecture is as that very first the 8 bit input is fed for the isomorphic mapping, the morphed output from this block is divided into two groups of 4 bit. Upper block is first squared and multiplied by constant and the lower block is multiplied by the addition of upper and lower block. Both the output are added and then its inverse is calculated. The inversed 4 bit is multiplied by lower block and upper block separately. Both the output of 4 bit is now combined to make 8. Now the affine transform of this 8 bit calculated. The output is now sub byte transform of the input.

III. 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	SLICES	GATE COUNT
ROM-based	0	65539
PCT-BASED	169	2542
GF- BASED	50	1250

Table 1: Comparison of Various S-box architectures

IV. CONCLUSIONS

This paper presented area optimized architecture for S-box used in AES encryption. The architecture is based on Galois Field technique applied to three block design of S-box. The proposed design is efficient in terms of area as well as speed compared to the fastest known implementation of S-box. The technique can also be

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 4, November 2012)

applied to Inverse S-box so as to increase the speed of decryption unit.

REFERENCES

[1] Akashi Satoh, Sumio Morioka, Kohji Takano and Seiji Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization.", Springer-Verlag Berlin Heidelberg, 2001.

[2] Vincent Rijmen, "Efficient Implementation of the Rijndael S-Box.", Katholieke Universities Leuven, Dept. ESAT. Belgium.

[3] Xinmiao Zhang and Keshab K. Parhi, "High-Speed VLSI Architectures for the AES Algorithm.", IEEE Transactions on Very Large Scale Integration(VLSI) Systems, Vol. 12, No. 9, Septemper 2004.

[4] "Advanced Encryption Standard (AES)" Federal Information Processing Standards Publication 197, 26th November 2001.

[5] Tim Good and Mohammed Benaissa, "Very Small FPGA Application-Specific Instruction Processor for AES.", IEEE Transactions on Circuits and Systems – I: Regular Papers, Vol. 53, No. 7, July 2006.

[6] The Advanced Encryption Standard http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[7] R. Liu, K.K.Parhi "Fast composite field architectures for Advanced Encryption tandard" Proceedings GLSVLSI'08, Orlando, Florida, USA, pp 65-70, May 4–6, 2008.

[8] T Good, M. Benaissa, 'AES on FPGA from the Fastest to the Smallest',LNCS 3659, pp. 427-440, 2004.

[9] J Zambreno, D. Nguyen, Alok Choudhary, ' Exploring Area/Delay Trade-offs in an AES FPGA Implementation, Leture Notes in Computer Scienc 3203, Proc.FPL, Antwerp, Belgium, pp 575-585,2004.

[10] M. M . Wong , M.L.D. Wong, A high throughput Low power compact AES S-box implementation using composite field arithmetic and Algebraic form representation, Proc.IEEE 2nd Asea Symposium on Quality Electronic Design, pp 318-323, 2010 .

[11] Rashmi Ramesh Rachh, P.V. Ananda Mohan and B.S. Anami, "Efficient Implementations of AES S box and Inverse S- box", Proc. IEEE ENCON, Singapore,pp 1-6, 2009

[12] M. Fayed, M. El-Kharashi and F. Watheq Gebali, 'A High-Speed, Fully-Pipelined VLSI Architecture for Real-Time AES', 4th International

[13] Conference on Information & Communications Technology, 2006.