ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

FPGA IMPLEMENTATION OF A COMPACT AES ALGORITHM WITH S-BOX OPTIMIZATION

Ms.PatilSarika B., Prof. Padma Lohiya
Electronics and Telecommunication Engg,D.Y Patil College Of Engg, Akurdi, Pune ,India

Abstract

This paper proposes a compact AES algorithm to achieve less slice consumption of FPGA. Proposed design is based on iterative round looping architecture. S-box is implemented using composite field arithmetic which requires less area than lookup table.We used same S-box for key expansion block. This design supports 128-bits key size. It uses 8-bit data path to decrease the parallelism of operations and therefore reduces the hardware utilization.Synthesis of our complete design is done using Xilinx ISE 14.5 and implemented on Spartan 3 FPGA using VHDL language. GUI is developed in visual basics 6.0. This GUI is used to send a plain text and key for encryption. Decrypted data is also displayed on the same. The results from the Place and Route report indicate that area occupied by this architecture is 680 slices. This design is very well suited for small embedded applications.

KEYWORDS: AES, LUT, Sbox, Composite field arithmetic, Galois field, GUI.

INTRODUCTION

AES is a FIPS approved cryptographic algorithmwhich maintains safty and is used to provide security to electronic data[1]. It is useful to transmit and receive data through insecure networks. In 2001 National Institute of Standard and Technology (NIST)replaced previous encryption standards like DESand triple DES with AES algorithm because of its efficiency ,implementation and flexibility. As AES is widely adapted for various applications from high end computers to loe power portable devices like RFID tags , Bluetooth ,smart cards etc. No of hardware architecures toimplement AES were proposed to meet different requirements. Two most important are high throughput design and lowareadesing . First one is used to achieve highest operating frequency and second is used to minimize the size of design and lower the power comsumption.

This paper focus on second requirement. A compact AES algorithm with combinational logic based S-box is implemented on Spartan 3 FPGA . Proposed design occupies 680 slices .

AES ALGORITHM

The Advanced Encryption Standard is a subcategory of much larger encryption algorithm known as Rijndael. The AES algorithm is a symmetric cipher in which a common secret key is used for both encryption and decryption. AES is a block cipher as it operates on fixed-length group of bits.AES has a block size of 128 bits and a key size of 128,192 and 256 bits which terns in 10, 12 and 14 rounds of operation respectively. After these rounds of operation, eachbit of cipher text depends on each bit of plain text. Contrasting DES, the decryption algorithm differs from encryption in AES.Although same steps are used in encryption and decryption,the order in which these steps are applied are different.But for the last round all other rounds are identical and the last round doesn't have mix column operation. AES calculations are done in a Galois field of GF (2^8).

The AES carry out four different transformations:Substitute-Bytes,ShiftRows, mixColumns and AddRoundKey.Similarly, the inversed versions of these conversions are used in decryption but the AddRoundKeyoperation (bit-wise xor) is same. By using a separate key schedule algorithm,different round keys are generated for each round of operation[5]. The block diagram of the AES algorithm is shown in Fig.1

A. **SubBytes**

SubByte transformation is a highly non-linear byte substitution where each byte inthe state array is replaced with another from a lookup table called an

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

S-Box. This S-boxis invertible,is constructed by composing two transformations:

1. Takethe multiplicative inverse in the finite fieldGF(2^8),

2.Apply the affine transformation (over GF(2)).

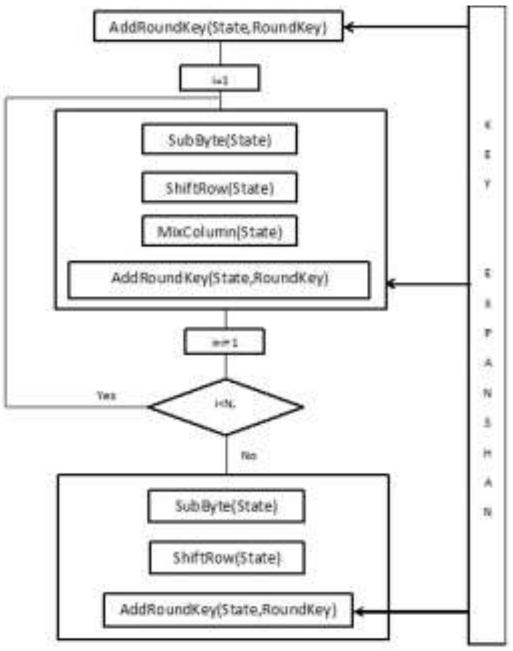


Figure 1 Encryption Process

B.ShiftRows

In ShiftRowstransformation ,the bytes in the last three rows of state are cyclically shifted over different numbers of bytes .In ShiftRows operation it is shifted left.There is noshift in the first row whereas the second,third and fourth rows are shifted by 1, 2 and 3 times respectively which is shown in fig3.

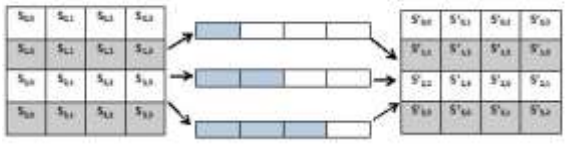


Figure 2 ShiftRow Example

C. MixColumns

The mixcolomn transformation operates on the state column by column considering each column as a four term polynomial over GF (2^8). Each polynomial is multiplied by the fix polynomial c(x) modulo the polynomialk(x) as shown below [2].

$$c(x) = \{03\}x^3 + \{02\}x^2 + \{01\}x + \{01\} \qquad (1)$$

$$k(x) = x^4 + 1 \qquad (2)$$

For the 128 bit key, each column is multiplied by the known matrix) which is given by

$$C = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \qquad (3)$$

In this matrix, multiplication by one means remain unchanged, multiplication by two means shitting byte to theleft and multiplication by three means shifting to the left and then EX-ORing the original value[5].

D. AddRoundKey

In addroundkeybitwise XOR operation is performed between the RoundKey and the output from Mixcolumn.The key isobtained from the Rijndael's Key Schedule.

E. Key Expansion Logic

The initial RoundKey will be the same as the first key in encryption where as in decryption it will be the last RoundKey. The RoundKey for all other rounds are generated from the Key Expansion routine.

Round keys are generatedearlier and stored in memory or generated on the fly in the prior, the round keys can be read out frommemory by using corresponding addresses for high speed applications. In the latter, the round keys are generated onthe fly which requires space only for single roundkey[3].

COMPACT SBOX IMPLEMENTATION

This section illustrates the steps involved in constructing the multiplicative inverse module for the S-Box using Composite Field Arithmetic. . The multiplicative inversecomputation will first be covered and the affine transformation will then follow to complete the methodology involved for constructing the S-Box for the SubByteoperation[4].

The individual bits in a byte representing GF(2^8) element can be viewed as coefficients to each power term in GF(2^8) polynomial. For instance, 100010112 is representing the polynomial q7 + q3 + q + 1 in GF(2^8). The multiplicative inverse circuitinGF(2^8) can be produced as shown in Figure 3.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

The Substitute Byte computes the multiplicative inverse of the individual byte ofthe state matrix followed by affine transformation. The multiplicative inverse computation will be done by dividing the more complex GF(2^8) to lower order fieldsof GF(2), GF(2^2) and GF($(2^2)^2$).

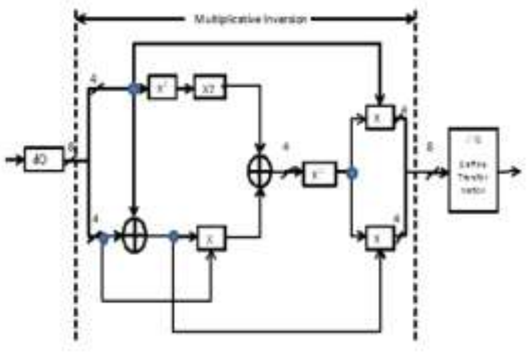


Figure 3 Block diagram of S-box implementation

The figure 2 shows the propagation of the input data of into a composite field based S-Box. The input data will first undergo the multiplicative inversion. The values at which the high and low nibbles are transformed to are indicated by the 4 bit numbers outside of the logical blocks. The example can be worked by hand since the tables containing the results for GF (2^4) multiplication and multiplicative inverses are provided. After theinverse isomorphic mapping operation of the multiplicative inversion module, the Affine Transformation is applied to the multiplicative inverse to yield the S-Box substituted value for the given input of 0xCB. Doing so yields an output of 0xF2 which agrees with the S-Box table provided in [1].

The legends for the blocks within the multiplicative inversion module from above are illustrated in the table 1 below.

Table 1: Legends for the blocks within the multiplicative inversion module

Symbol	Description
δ	Isomorphic Mapping to Composite
X^{-1}	Multiplicative inversion in GF(2^4)
X^2	Squarer in GF(2^4)
X	Multiplication in GF(2^4)
δ^{-1}	Inverse Isomorphic Mapping to

The multiplicative inverse computation will be done by decomposing the morecomplex GF(2^8) to lower order fields of GF(2^1), GF(2^2) and GF($(2^2)^2$) [3] In order to complete the above, the following irreducible polynomials are used.

GF (2^2) GF (2) $\Longrightarrow$:X^2+X+1
GF (($2^2)^2$) GF (2^2) :$\Longrightarrow$ $X^2+X+\varphi$
GF (((${2^2})^2)^2$) GF (($2^2)^2$):$\Longrightarrow$ $X^2+X+\lambda$

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

In matrix form, the affine transformation element of the S-box can be expressed [4] as

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (4)$$

The sub operations involved in computing the multiplicative inverse is grouped to implement S-Box in sucha way to reduce both area and critical path that results inreduced power consumption.

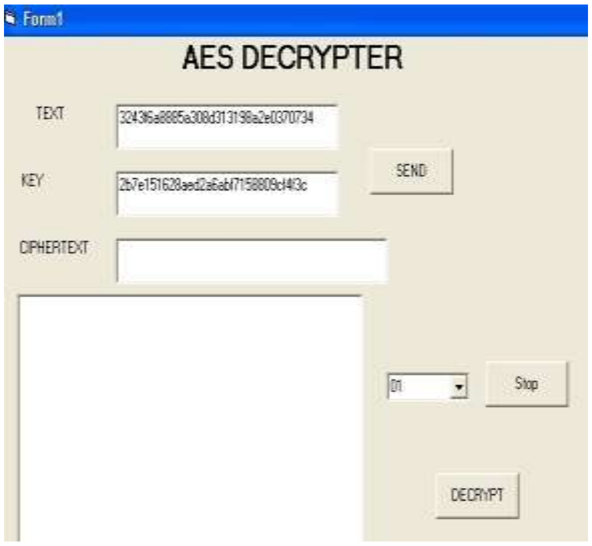
The input and output for Sbox block is 8 bit. Output of Sbox is 0xED for input value 0x53 which agrees with Sbox table. The area occupied by the Sbox is 41slices out of total 3,583 slices.

RESULTS AND DISCUSSION

The synthesis is done using Xilinx ISE 14.5 by using VHDL language on a Xilinx Spartan-III X3CS400-5PQ208 FPGA. GUI is implemented using visual basic 6.0.This GUI is used to accept a text and cipherkey entered by user for encryption as shown in fig 6 . It show encryptedcipher text , all the keys generated and decrypted data in each round as shown in fig 7. It shows final decrypted data which is same to plaintext. Setting of Baud rate and COM port can be done in the software.

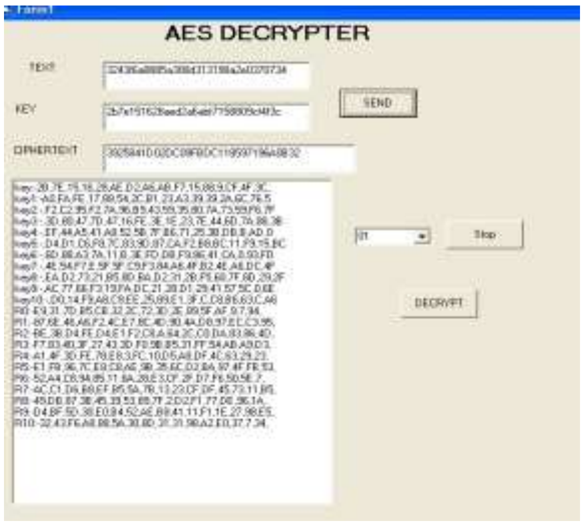
Resource utilization of AES Encryption core design is shown in Table 2, and resource utilization of AES Encryption core with Sbox using lookup table is shown in Table 3. Comparing the results it can be stated that the implemented design of AES Encryption core uses less area of the FPGA and is more efficient. The AES Encryption core with lookup table uses 1.5 times the number of slices used in this AES design.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114



TEXT: 32436a8885a308d313198a2e0370734
KEY: 2b7e151628aed2a6abf7158809cf4f3c
CIPHERTEXT:
01

Figure 4 Plain text and key entered by User



TEXT: 32436a8885a308d313198a2e0370734
KEY: 2b7e151628aed2a6abf7158809cf4f3c
CIPHERTEXT: 3925841D02DC09FBDC118597196A0B32
01

```
key : 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
key1 : A0 FA FE 17 88 54 2C B1 23 A3 39 39 2A 6C 76 5
key2 : F2 C2 95 F2 7A 96 B9 43 59 35 80 7A 73 59 F6 7F
key3 : 3D 80 47 7D 47 16 FE 3E 1E 23 7E 44 6D 7A 88 3B
key4 : EF 44 A5 41 A8 52 5B 7F B6 71 25 3B DB 0B AD 0
key5 : D4 D1 C6 F8 7C 83 9D 87 CA F2 B8 BC 11 F9 15 BC
key6 : 6D 88 A3 7A 11 0B 3E FD DB F9 86 41 CA 0 93 FD
key7 : 4E 54 F7 0E 5F 5F C9 F3 84 A6 4F B2 4E A6 DC 4F
key8 : EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F
key9 : AC 77 66 F3 19 FA DC 21 28 D1 29 41 57 5C 0 6E
key10 : D0 14 F9 A8 C9 EE 25 89 E1 3F C C8 B6 63 C A8
R0 E9 31 7D B5 CB 32 2C 72 3D 2E 89 5F AF 9 7 94
R1 87 6E 46 A6 F2 4C E7 8C 4D 90 4A D8 97 EC C3 95
R2 BE 3B D4 FE D4 E1 F2 C8 A 64 2C C0 DA 83 86 4D
R3 F7 83 40 3F 27 43 3D F0 9B B5 31 FF 54 AB A9 D3
R4 A1 4F 3D FE 78 E8 3 FC 10 D5 A8 DF 4C 63 29 23
R5 E1 FB 96 7C E8 C8 AE 9B 2B 6C D3 BA 97 4F FB 53
R6 52 A4 C8 94 85 11 6A 28 E3 CF 2F D7 F6 50 5E 7
R7 AC C1 D6 B8 EF B5 5A 7B 13 23 CF DF 45 73 11 B5
R8 49 D8 87 38 45 39 53 89 7F 2 D2 F1 77 DE 96 1A
R9 D4 BF 5D 30 E0 B4 52 AE B8 41 11 F1 1E 27 98 E5
R10 32 43 F6 A8 88 5A 30 8D 31 31 98 A2 E0 37 7 34
```

Figure 5 Cipher text , Keys and Decrypted Plain Text

Table 2 Device utilization-AES Encryption core.

Logic utilization	Used	Available	Utilization
No of slice flip-lops	680	7,168	9%
No of 4 inputLUT	2,458	7,168	34%
No of occupied slices	1431	3,584	39%
No of bonded IOB's	4	141	2%

Table 3 Device utilization-AES Encryption core with Lookup Table

Logic utilization	Used	Available	Utilization
No of slice flip-flops	922	7,168	12%
No of 4 input LUT's	2,932	7,168	40%
No of occupied slices	1,729	3,584	48%
No of bonded IOB's	4	141	2%

CONCLUSION

In this paper a solution for compact AES encryption on FPGA is presente. The SBox is designed by using composite field arithmetic andresultingimplementation fits with minimum number of slices.As compared to the typical ROM based lookup table, the presented implementation is capable of small area occupancy This make it suitable for security focused low resource applications.

REFERENCES

[1] National Institute of Standards and Technology(NIST),Advanced Encryption Standard (AES) Federal Information Processing Standards Publication 197 (FIPS PUB 197), Nov.2001

[2] T. Manoj Sharma and R. Thilagavathy, "Performance Analysis of Advanced Encryption Standard for Low power and Area Applications" in Proc. IEEE on ICT,2013, pp 967-972

[3] P. Hamalainen, T. Alho, M. Hannikainen, and T.D. Hamalainen, "Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core", in Proc. DSD, 2006, pp.577-583.

[4] E.-N. Mui "Practical Implementation of Rijndael S-Box Using combinational logic", 2007. [online] Available:http://www.xess.com/projects/Rijndael_SBox. pdf

[5] RajenderManteena, "A VHDL Implemetation of the Advanced Encryption Standard-Rijndael Algorithm", College of Engineering University of South Florida, 2004.

T. Good and M. Benaissa, "AES on FPGA from the Fastest to the Smallest," Lecture Notes in Computer Science, vol.3659, pp.427-440, Sep. 2005.