International Journal of Research
Available at https://edupediapublications.org/journa
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 13
September 2016

Design an AES Algorithm Using S.R & M.C Technique

KARICHETI SINGAIAH
ELECTRONICS & COMMUNICATION ENGINEERING
BVSR Engineering College.

M.PRIYANKA, Assistant professor
ELECTRONICS & COMMUNICATION ENGINEERING
BVSR Engineering College

Abstract—Now a days Advanced Encryption Standard (AES) is the most efficient public key encryption system based on Rijndael Algorithm that can be used to create faster and efficient cryptographic keys. AES generates keys through the properties of the Rijndael Algorithm instead of conventional method of the key generation. Althoughmany encryption algorithms can be relatively efficiently implemented in Xilinx software, there is still a need for special purpose cryptographic processors. First of all, high throughput applications, such as the encryption of the physical layer of Internet traffic, require an ASIC that does not affect the data throughput. By using proposed technique we get high security than existed technique.

Keywords: Cryptography, Rijndael, Encryption, Decryption, Cypher, Inverse cypher.

INTRODUCTION

Several techniques, such as cryptography, steganography, watermarking, and scrambling, have been developed to keep data secure, private, and copyright protected [1], [2]. Cryptography is an essential tool underlying virtually all networking and computer protection, traditionally used for military and espionage. However, the need for secure transactions in ecommerce, private networks, and secure messaging has moved encryption into the commercial realm [3].

Advanced encryption standard (AES) was issued as Federal Information Processing Standards (FIPS) by National Institute of Standards and Technology (NIST) as a successor to data encryption standard (DES) algorithms. In recent literature, a number of architectures for the VLSI implementation of AES Rijndael algorithm are reported [4], [5], [6], [7], [8]. It can be observed that some of these architectures are of low performance and some provide low throughput. Further, many of the architectures are not area efficient and can result in higher cost when implemented in silicon.

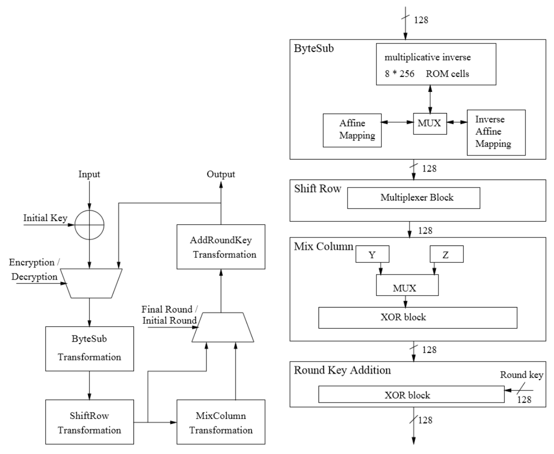
In this paper, we propose a high performance, high throughput and area efficient VLSI architecture for Rijndeal algorithm that is suitable for low cost silicon implementation. The proposed architecture is optimized for high throughput in terms of the encryption and decryption data rates using pipelining. Polynomial multiplication is implemented using XOR operation instead of using multipliers to decrease the hardware complexity. In the proposed architecture both the encryption and decryption modes use common hardware resources, thus making the design area efficient. Selective use of look-up tables and combinational logic further enhances the architecture's memory optimization, area, and performance. An important feature of our proposed architecture is an effective solution of online (real-time) round key generation needing significantly less storage for buffering.

THE PROPOSED VLSI ARCHITECTURE FOR RIJNDAEL

The proposed architecture showing the order of operation and control between the transformations is shown in Fig. 1(a).

A. Architecture of the Data Unit

The data unit consists of: the initial round of key addition, $N_r - 1$ standard rounds, and a final round. The architecture for a standard round composed of four basic blocks is shown in Fig. 1(b). For each block, both the transformation and the inverse transformation needed for encryption and decryption, respectively are performed using the same hardware resources. This implementation generates one set of subkey and reuses it for calculating all other subkeys in real-time.



(a) Rijndael Algorithm Data and Con- (b) Architecture for the Standard

Fig. 1. Top Level View of the Rijndael

1) ByteSub: In this architecture each block is replaced by its substitution in an S-Box table consisting of the multiplicative inverse of each byte of the block state in the finite field $GF(2^8)$. In order to overcome the performance bottleneck,control Flow Round in the Data Unitthe implementation of multiplicative inverses is carried out using look-up tables (stored in a table of 8 × 256). The implementation includes the affine mapping of the input in both encryption and decryption processes as follows:

2) ShiftRow: In this transformation the rows of the block state are shifted over different offsets. The amount of shifts is determined by the block length. The proposed architecture implements the shift row operation using combinational logic considering the offset by which a row should be shifted.

3) MixColumn: In this transformation each column of the block state is considered as a polynomial over $GF(2^8)$. It is multiplied with a constant polynomial $C(x)$ or $D(x)$ over a finite field in encryption or decryption, respectively. In hardware, the multiplication by the corresponding polynomial is done by XOR operations and multiplication of a block by X. This is implemented using a multiplexer, the control being the MSB is 1 or 0. The equations implemented in hardware for *MixColumn* in encryption and decryption are as follows.

In encryption process,

$In0$ is the least significant 8 bits of a column of a matrix. Architecture of different units are shown in Fig. 2 and the architecture of *MixColumn* transformation is shown in the Fig. 3.

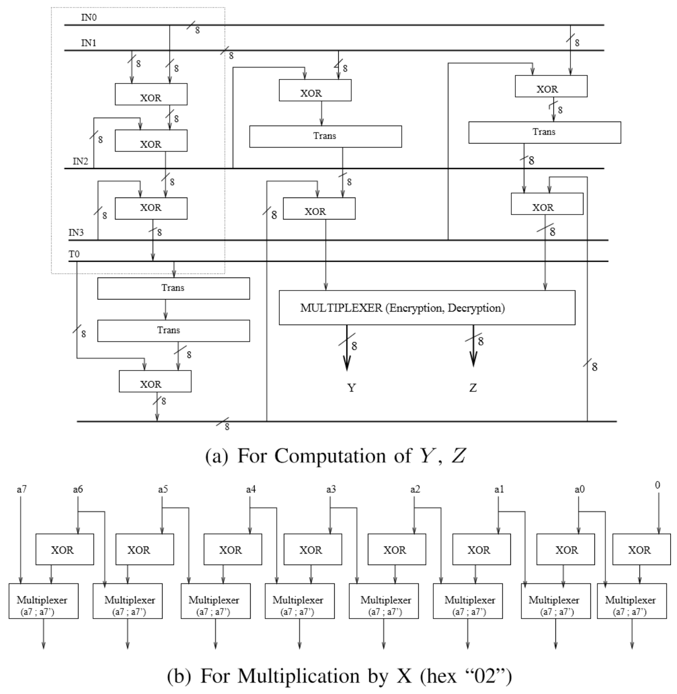

IN0
IN1
IN2
IN3
T0
XOR
Trans
MULTIPLEXER (Encryption, Decryption)
Y
Z
8
(a) For Computation of Y, Z
a7
a6
a5
a4
a3
a2
a1
a0
0
Multiplexer (a7 ; a7')
(b) For Multiplication by X (hex "02")

Fig. 2. Architecture for Units used in Mix Column Transformation

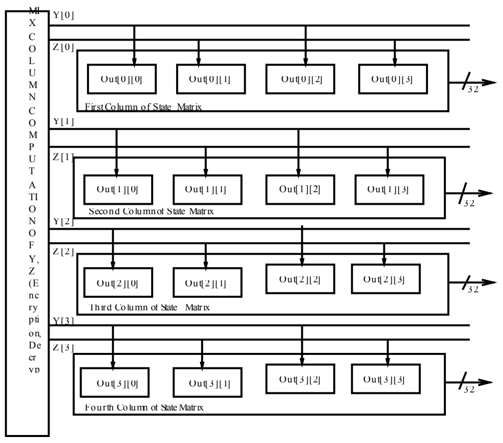


Fig. 3. Architecture for Mix Column Transformation for 128 bits

4) AddRoundKey: In this transformation (architecture represented in Fig. 4), the round key obtained from the key scheduler is XORed with the block state obtained from the *MixColumn* transformation or *ShiftRow* transformation based on the type of round being implemented. In the standard round, the round key is XORed with the output obtained from the *MixColumn* transformation. In the final round the round key is XORed with the output obtained from the *ShiftRow* transformation. In the initial round, bitwise XOR operation is performed between the initial round key and the initial state block.

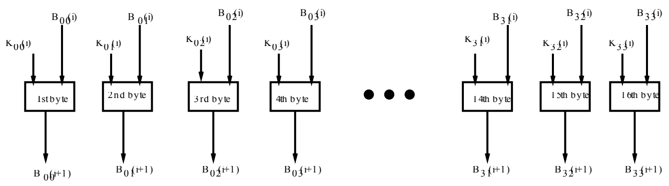


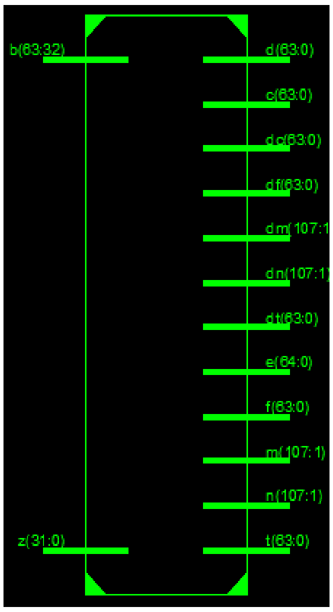
Fig. 4. Architecture for Round Key Addition Transformation

B. Memory Optimization

Since the design is based on one clock cycle for each encryption round, the memory modules had to be duplicated. For example, in the *ByteSub*, the S-boxes need to be duplicated 16 times. Consequently, the choice of memory architecture is very critical. Since all the table entries are fixed and defined in the standard, the usage of ROM is preferred. Specifically, the architecture requires several small ROM modules instead of one large module, since each lookup will only be based on a maximum of 8-bit address, which translates to 256 entries. We implemented the multiplicative inverse function using the look-up table of size 8×256. We have a total of 20 copies of the S-boxes in our design; 16 of them in encryption module and 4 in the key scheduling module.

D. Performance Evaluation

An AES-128 encryption / decryption of a 128-bit block was done in 11 clock cycles using the feedback logic. In each clock cycle, one transformation is executed and, at the same time, the appropriate key for the next round is calculated. The whole process concludes after 10 rounds of transformations. The outputs is shown in below figure 5 RTL and waveform.



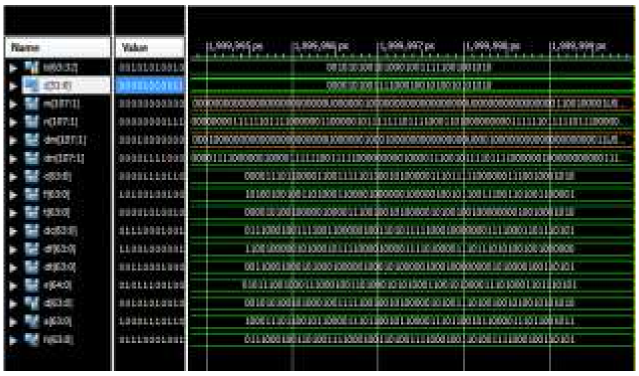


Fig.5 RTL, output wave form

V. Discussions and Conclusions

We have presented a VLSI architecture for the Rijndael AES algorithm that performs both the encryption and decryption. S-boxes are used for the implementation of the multiplicative inverses and shared between encryption and decryption. The round keys needed for each round of the implementation are generated in real-time. The forward and reverse key scheduling is implemented on the same device, thus allowing efficient area minimization. Although the algorithm is symmetrical, the hardware required is not, with the encryption algorithm being less complex than the decryption algorithm. The implementation of the key unit in the proposed architecture, can be scaled for the keys of length 192 and 256 bits easily.

References

[1] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli, “A DCT Domain Visible Watermarking Technique for Images,” in *Proc of the IEEE International Conf on Multimedia and Expo*, 2000, pp. 1029–1032.

[2] M. S. Kankanhalli and T. T. Guan, “Compressed-Domain Scrambler / Descrambler for Digital Video,” *IEEE Transactions on Consumer Electronics*, vol. 48, no. 2, pp. 356–365, May 2002.

[3] B. M. Macq and J. J. Quisquater, “Cryptography for Digital TV Broadcasting,” *Proceedings of the IEEE*, vol. 83, no. 6, pp. 944–957 , Jun 1995.

[4] H. Kuo and I. Verbauwhede, “Architectural Optimization for a 1.82 Gbits/sec VLSI Implementation of the AES Rijndael Algorithm,” in *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems*, 2001, vol. 2162, pp. 51–64.

[5] M. McLoone and J. V. McCanny, “Rijndael FPGA Implementation Utilizing Look-up Tables,” in *Proceedings of the IEEE Workshop on Signal Processing Systems*, 2001, pp. 349–360.

[6] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, “A Compact Rijndael Hardware Architecture with S-Box Optimization,” in *Proceedings of Advances in Cryptology - ASIACRYPT 2001*, 2001, pp. 171–184.

[7] S. Mangard, M. Aigner, and S. Dominikus, “A Highly Regular and Scalable AES Hardware Architecture,” *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 483–491, April 2003.

[8] T. Sodon O. J. Hernandez and M. Adel, “Low-Cost Advanced Encryption Standard (AES) VLSI Architecture: A Minimalist Bit-Serial Approach,” in *Proc of IEEE Southeast Conference*, 2005, pp. 121–125.

[9] J. Daemen and V. Rijmen, *The Design of Rijndael*, Springer-Verlag, 2002.

[10] A. J. Elbirt, W. Yip, B. Chetwynd, and ChristofPaar, “An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists,” in *Proceedings of the Third Advanced Encryption Standard (AES) Candidate Conference*, 2000, pp. 13–27.