International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 13
October 2017

A High-Secure Vlsi Architecture For Advanced Encryption Standard (Aes) Algorithm

V. SPANDANA
M.Tech (VLSI)
Dept of E.C.E
Chaitanya Institute of Technology and Science, Warangal, Telangana,India
Email: chinnispandu@yahoo.com

Dr. K. SEETHARAM
Associate Professor
Dept of E.C.E
Chaitanya Institute of Technology and Science, Warangal, Telangana,India
Email: seetha_ram2002@yahoo.com

Abstract
In this paper we present a high-performance, high throughput, and area efficient architecture for the VLSI implementation of the AES algorithm. The sub keys, required for each round of the Rijndael algorithm, are generated in real-time by the key-scheduler module by expanding the initial secret key, thus reducing the amount of storage for buffering. Moreover, pipelining is used after each standard round to enhance the throughput. A prototype chip implemented using 0:35^{1} CMOS technology resulted in a throughput of 232*Mbps* for iterative architecture and 1:83*Gbps* for pipelining architecture.
Keywords-AES; S box; FPGA

I. INTRODUCTION

Several techniques, such as cryptography, steganography, watermarking, and scrambling, have been developed to keep data secure, private, and copyright protected. 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.

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. 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.

II. RIJNDAEL ALGORITHM

Rijndael algorithm is an iterated block cipher supporting a variable data block and a variable key length of 128, 192 or 256 bits. The algorithm consists of three distinct phases: (i) an initial data/key addition, (ii) nine (128-bits), eleven (192-bits) or thirteen (256-bits) standard rounds, (iii) a final round which is a variation of a standard round. The number of standard rounds depends on the data block and key length. If the maximum length of the data block or key is 128, 192 or 256, then the number of rounds is 10, 12 or 14, respectively.

The initial key is expanded to generate the round keys, each of size equal to block length. Each round of the algorithm receives a new round key from the key schedule module. Each standard round includes four fundamental algebraic function transformations on arrays of bytes. These transformations are: byte substitution, shift row, mix column, and round key addition.

The final round of the algorithm is similar to the standard round, except that it does not have *Mix Column* operation. Decryption is performed by the application of the inverse transformations of the round functions. The sequence of operations for the standard round function differs from encryption. The computational performance differs between encryption and decryption because the inverse transformations in the round function is more complex than the corresponding transformation for encryption.

III. 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, *Nr ¡* 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 sub key and reuses it for calculating all other sub keys in real-time.

1) Byte Sub: 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*(28). In order to overcome the performance bottleneck.

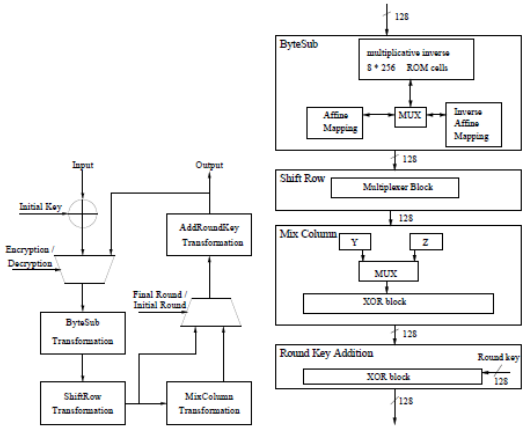


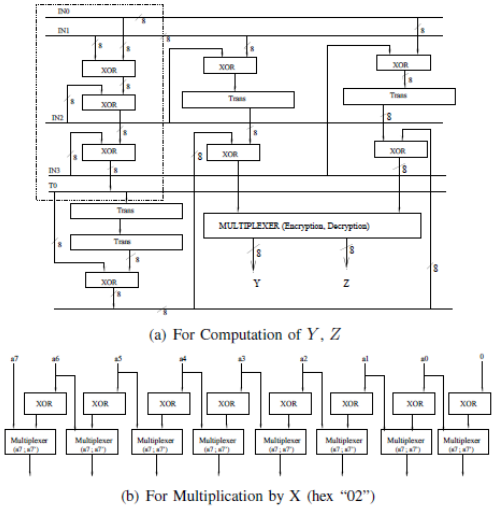
Fig 1(a) Rijndael Algorithm Data and Control Flow. (b) Architecture for the Standard Round in the Data Unit

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 13
October 2017

The implementation of multiplicative inverses is carried out using look-up tables. The implementation includes the affine mapping of the input in both encryption and decryption processes.

2) Shift Row: 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) Mix Column: In this transformation each column of the block state is considered as a polynomial over *GF*(28). 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.



(a) For Computation of Y, Z

(b) For Multiplication by X (hex "02")

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

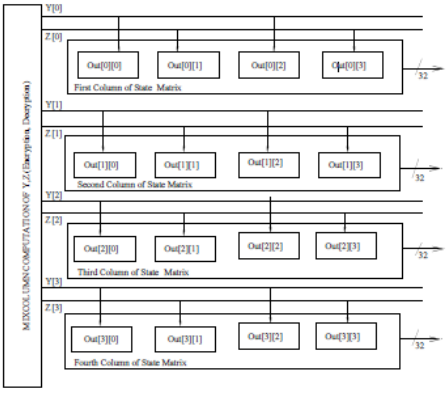


Fig. 3. Architecture for Mix Column Transformation for 128 bits

4) Add Round Key: In this transformation the round key obtained from the key scheduler is XORed with the block state obtained from the *Mix Column* transformation or *Shift Row* transformation based on the type of round being implemented. In the standard round, the round key is XORed with the output obtained from the *Mix Column* transformation. In the final round the round key is XORed with the output obtained from the *Shift Row* transformation. In the initial round, bitwise XOR operation is performed between the initial round key and the initial state block.

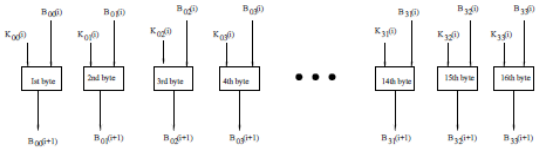


Fig. 4. Architecture for Round Key Addition Transformation

B. Architecture for Key Scheduling

In the key scheduling module, the initial key is expanded and the generated round keys are stored in four 32- bit registers. Both the forward and reverse key scheduling are done in the same device. The

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 13
October 2017

Byte Sub required in the key expansion unit is implemented using the S-Boxes.

Four S-Boxes are needed for a 128-bit key and 128-bit data block implemented using 8£256 ROM cells. Multiplexers are used as a control signal to distinguish between the initial key and the round key (obtained from the initial key using a "key expansion unit").

The least significant 32 bits of the 128-bit key is cyclically shifted to the left by a byte, implemented using combinational logic. The resulting word after the left shift operation is sent through the S-boxes and the affine mapping operation, in order to perform *Byte Sub*. The key resulting from the *Byte Sub* is XORed with the Round Constant (RCON). In this architecture, the round constant is generated using the combinational logic. The round constant should be symmetric with the round key being generated.

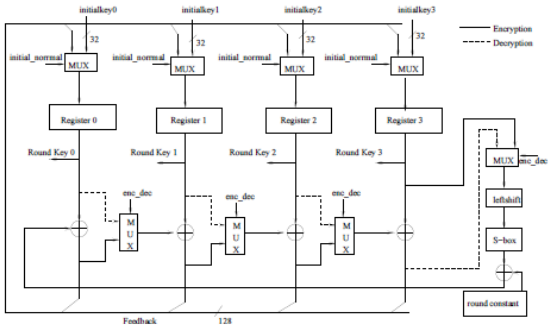


Fig. 5. Architecture for Key Scheduling Unit

The total number of round constants that need to be generated is equal to the number of rounds. The round constant is obtained in real-time by multiplying the previous round constant by X. This is amenable for implementation in the hardware using XOR operations. For the reverse key scheduling, the last round key should be generated with forward key scheduling for the first time. The last round key is expanded to generate the reverse round keys.

Decryption requires more cycles than encryption because it needs pre-scheduling to generate the last key value. Since the Rijndael algorithm allows different key lengths and block lengths, each round key is carefully set to have the same length as the data block. In the case where key length and the block length are not equal, previous, current and also the next round keys are needed in order to generate the appropriate set of round keys that are fed into the encryption module, which is performed by a "key alignment unit".

IV.RESULTS

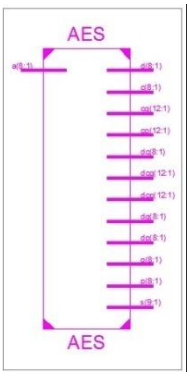


Fig. 6. RTL SCHEMATIC

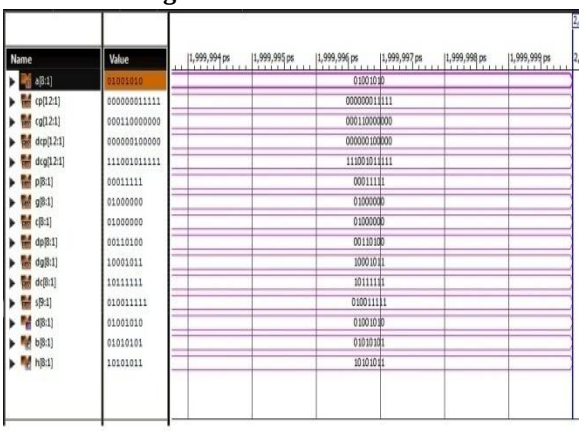


Fig. 7. OUTPUT

V. CONCLUSION

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.

VI. 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 Christof Paar, "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.

Authors:

V. SPANDANA studying M.Tech (VLSI) from Chaitanya Institute of Technology and Science, Warangal, Telangana,India.

Dr.K.SEETHARAM working as Associate Professor in Dept of E.C.E from Chaitanya Institute of Technology and Science, Warangal, Telangana, India.