Implementation of Rijndael Block Cipher Algorithm

YunKyung Lee, YoungSoo Park

ETRI(Electronics and Telecommunications Research Institute)

161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350 KOREA

e-mail : neohappy@etri.re.kr, yspark@etri.re.kr

Abstract : This paper presents the design of Rijndael crypto-processor with 128 bits, 192 bits and 256 bits key size. In October 2000 Rijndael cryptographic algorithm is selected as AES(Advanced Encryption Standard) by NIST(National Institute of Standards and Technology). Rijndael algorithm is strong in any known attacks. And it can be efficiently implemented in both hardware and software. We implement Rijndael algorithm in hardware, because hardware implementation gives more fast encryption/decryption speed and more physically secure. We implemented Rijndael algorithm for 128 bits, 192 bits and 256 bits key size with VHDL, synthesized with Synopsys, and simulated with ModelSim.

This crypto-processor is implemented using on-the-fly key generation method and using lookup table for S-box/SI-box. And the order of Inverse Shift Row operation and Inverse Substitution operation is exchanged in decryption round operation of Rijndael algorithm. It brings about decrease of the total gate count.

Crypto-processor implemented in these methods is applied to mobile systems and smart cards, because it has moderate gate count and high speed.

1. Introduction

Rijndael algorithm is a block cipher performing encryption and decryption for 128 bits input data and private key cipher which uses same key in data encryption and data decryption. It also called AES-128, AES-192 and AES-256 for used cipher key size. Each is 128 bits, 192 bits and 256 bits key size. And iterated round number varies as used key size. Table 1 shows the number of rounds as a function of key size[1].

Table 1. Number of Rounds as a function of key size.

Key length	128 bits	192 bits	256 bits
Round number	10 rounds	12 rounds	14 rounds

Hardware implementation of Rijndael algorithm gives the faster data encryption and decryption time and the higher security than software implementation[2]. It is important in hardware implementation that each module of round operation is implemented faster, because data encryption and decryption is completed after specific number of round iteration in Rijndael algorithm.

This paper presents implemented Rijndael processor structure of moderate gate count and high speed. In the following clause, general Rijndael algorithm is described and then characteristic of implemented Rijndael processor is described. Finally, the synthesized results are presented.

2. Rijndael Algorithm

Rijndael algorithm is a block cipher executing data encryption and data decryption with 128 bits, 192 bits and 256 bits cipher key. Encrypted and decrypted data is generated after each 10 rounds, 12 rounds and 14 rounds operation for 128 bits, 192 bits, 256 bits key size.

Encryption round is made up of Substitution, Shift Rows, MixColumn and Add Round Key operation. Substitution operation is a non-linear byte mapping operation and S-box is used. Shift Rows operation is byte shift operation per row (state(basic operation unit of

Rijndael algorithm) is 4 rows matrix form, and column based operation is performed, usually.). MixColumn operation is matrix multiplication that columns are multiplied by specific constants. Add Round Key operation is XORing of MixColumn result data and round keys.

Decryption round is inverse of encryption round and is consist of Inverse Shift Rows, Inverse Substitution, Add Round Key and Inverse MixColumn. Inverse Shift Rows operation is byte shift operation per row reversly. Inverse Substitution operation is non-linear byte mapping operation using SI-box, which is inverse of S-box. Inverse MixColumn operation is matrix multiplication that columns are multiplied by specific constants which is inversion of constants used in MixColumn operation in GF(2^8). Figure 1 shows the flow chart of Rijndael algorithm.

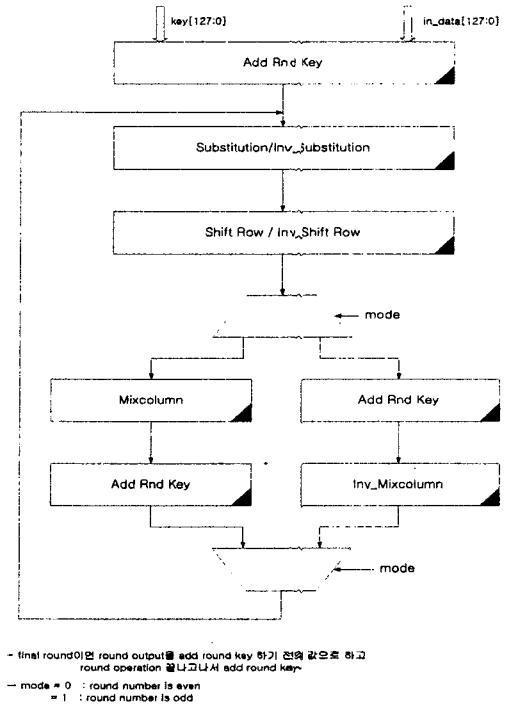

Figure 1. Flow Chart of Rijndael Algorithm

First round of Rijndael algorithm performs only Add Round Key operation. And Final round performs three operations except MixColumn(or Inverse MixColumn). Oter rounds perform all four operations.

S-box used in Substitution operation is gained as follows.

1. Take the multiplicative inverse in the finite field GF(2^8) ; Assume that multiplicative inverse of '0' is '0'.
2. Apply an Affine transform

SI-box used in Inverse Substitution operation is the inverse of S-box, which is gained by applying Inverse Affine transform followed by taking the multiplicative inverse in the finite field GF(2^8)[1]. Figure 2 shows the matrix form of Affine transform. bi($0 < i < 7$) is polynomial expression of 1 byte input value. And b7 is MSB(Most Significant Bit) and b0 is LSB(Least Significant Bit).

$$\begin{pmatrix} b0' \\ b1' \\ b2' \\ b3' \\ b4' \\ b5' \\ b6' \\ b7' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} b0 \\ b1 \\ b2 \\ b3 \\ b4 \\ b5 \\ b6 \\ b7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Figure 2. Affine Transform

Inverse Affine transform is inverse transformation of Affine transform. XORing of Input value and '63'(hexadecimal value) followed by multiplication with inverse matrix used in Affine transform. Figure 3 shows the Inverse Affine transform.

$$\begin{pmatrix} b0 \\ b1 \\ b2 \\ b3 \\ b4 \\ b5 \\ b6 \\ b7 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \left(\begin{pmatrix} b0' \\ b1' \\ b2' \\ b3' \\ b4' \\ b5' \\ b6' \\ b7' \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \right)$$

Figure 3. Inverse Affine Transform

3. Implemented Rijndael processor structure

3.1. Round Operation

Round operation is consist of Substitution, Shift Rows, MixColumn and Add Round Key as previous description.

Each Shift Rows and Inverse Shift Rows modules needs four 32 bits shift registers. First 32 bits shift register used in Shift Rows is simple register, second 32 bits shift register is 1 byte rotate shift left register, third shift left register is 2 byte rotate shift left register and final shift left register is 3 byte rotate shift left register. And in inverse Shift Rows module, first shift register is simple register too, second shift register is 3 byte rotate shift left register, third register is 2 byte shift left register, and final register is 1 byte shift left register. So first and third register can be shared in encryption mode and decryption mode.

S-box used in encryption mode of Rijndael algorithm is obtained by taking multiplicative inverse in finite field $GF(2^8)$ followed by affine transform. And SI-box used in decryption mode is obtained by inverse affine transform followed by taking multiplicative inverse in finite field $GF(2^8)$. If S-box and SI-box is obtained by using combinational logic, then it needs multiplicative inverse calculation module. It's hardware implementation is difficult and it decreases the effect of hardware implementation, i.e. high speed. So we implemented S-box and SI-box using lookup table. The synthesis result of S-box implemented using lookup table is about 500 gates and it is acceptable.

For reusable implementation of encryption and decrytion module, we implemented decryption round in exchanged order of Inverse Substitution and Inverse Shift Rows. These two operations perform transformations per byte, so it is not affect to the decryption result. Data processing unit of each round operation is 128 bits which is basic data processing unit of the Rijndael algorithm. It results in no supplementary register use, and no increasing of gate count and data processing time.

3.2. Round Key Generation

Different round key is needed per round operation. Two round key generation methods exist. The one is that round key is generated before encryption or decryption start and stored in registers. It needs supplementary register about 128 * (total round number + 1) bits , key protection method for external attack, and round key generation time. Another method is on-the-fly key generation method, which generates round key during other round operation. Round key is used only in Add Round Key operation which is last operation of round operations. And round key generation time is smaller than the execution time of the other round operations. So on-the-fly key generation is not an increasing factor of total encryption or decryption time.

In AES-192(192 bits key Rijndael algorithm) and AES-256(256 bits key Rijndael algorithm), data processing unit of round operation is 128 bits, too. The round operation structure is the same as it of AES-128. but key generation mechanism is different each other. Supplementary registers and additional key control are needed. Three types of round key control method applied to each round key generation in AES-192 and two types of round key control method applied to each round key generation in AES-256.

4. Implementation Results

We implemented Rijndael algorithm for 128 bits, 192 bits and 256 bits key size in previously described methods using VHDL. And we synthesized using Synopsys. The synthesized result is that the total gate count of AES-128 and AES-192 encryption and decryption processor is about 40,000 gates, and the total gate count of AES-256 encryption and decryption processor is about 45,000 gates. Data arrival time is about 600 ns. It is reasonable.

Some modified structure of implemented crypto processor applied to mobile system and smart card, etc.

5. Conclusions

This paper describes the Rijndael crypto processor implementation methods for 128 bits, 192 bits and 256 bits key size, and synthesizing results. Applied key generation method is on-the-fly key generation method. Unlike AES-128 implementation, AES-192 implementation and AES-256 implementation needs supplementary registers and more delicate key control. Lookup table method is applied

to S-box/SI-box implementation. So we obtained moderate gate counts and high speed Rijndael crypto processor.

References

[1] Joan Daemen, Vincent Rijmen, "AES Proposal : Rijndael"

[2] AJ Elbirt, W Yip, B Chetwynd, C Paar, " An FPGA-BASED Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists," IEEE Tran. on VLSI

[3] http://csrc.nist.gov/encryption/aes/