A Low-Resource AES Encryption Circuit Using Dynamic Reconfiguration

Peera Thontirawong and Prabhas Chongstitvatana
Department of Computer Engineering, Chulalongkorn University
Bangkok, Thailand
Email: prabhas@chula.ac.th

Abstract

This paper presents an implementation of an Advanced Encryption Standard (AES) encryption unit using dynamic reconfiguration based on the Xilinx Spartan-3 FPGA platform. The proposed design reuses resource of FPGA by adapting dynamic reconfiguration to reduce the number of resource used in the circuit. By changing circuits at runtime, the size of the whole circuit is limited to the largest reconfigurable module. The implementation of the dynamic reconfigurable AES encryption unit on XC3S200-4FT256 requires only 359 slices, while achieving throughput about 18 Kbps, and 16 Mbps if assume that there is no reconfiguration delay.

Key Words: AES, FPGA, Dynamic Reconfiguration

1. Introduction

In 2001, the National Institute of Standards and Technology (NIST) accepted the Rijndael algorithm [1], designed by Joan Daemon and Vincent Rijmen, as the Advanced Encryption Standard (AES) [2]. This new AES has replaced the Data Encryption Standard (DES) [3]. The AES is more robust than DES but it is also more complex than DES. This complexity demands more computational effort from an AES device. The hardware implementation of the AES algorithm is attractive since the software implementation is unable to satisfy the higher throughput requirement.

The hardware designs and implementations for the AES algorithm have been invented by many researches. Many Field Programmable Gate Array (FPGA) implementations of the AES algorithm have been introduced [4-6]. Motivated by the need for higher throughput with limited resource, several hardware designs and implementations of the AES algorithm have proposed either for very high throughput [6] or for more limited resource needed [4,5].

Presently, the privacy of data is very important. As mobile communication devices are becoming smaller and more ubiquitous, the need for encryption capability in these devices arise. The AES algorithm implemented on these devices is necessary. Since the FPGA is the most adaptive device, the implementation of the AES algorithm on limited resource FPGA is interesting.

The dynamic reconfiguration is an idea to reuse the resource in FPGA, so the resource is used more efficiently. This paper proposes an alternative design to implement the AES encryption algorithm in limited resource FPGA by using 8 bits datapath and dynamic reconfiguration concept.

The remainder of this paper is organized as follows. In Section 2, the AES algorithm is briefly described. Section 3 presents the design and implementation. Section 4 shows the implementation result, followed by the conclusion in Section 5.

2. The AES Algorithm

Refer to FIPS 197 [2], the AES algorithm operates on a block of 128 bits input and transforms it into an encrypted block of 128 bits output by using a 128, 192 or 256 bits cipher key. The number of rounds to be performed during the execution of the algorithm depends on the key size (see Table 1).

Table 1 Key-Block-Round Combinations

Algorithm	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

The AES encryption algorithm is separated into two parts: Cipher and Key Expansion. They are described in following subsections.

2.1 Cipher

The Cipher is described in the pseudo code in Figure 1, and its transformation process is described in following subsections.

```
Cipher(byte in[4*Nb],    byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4, Nb]
    state = in
    AddRoundKey(state, w[0, Nb-1])
    for round = 1 step 1 to Nr-1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[Nr*Nb, (round+1)*Nb-1])
    end for
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr*Nb, (Nr-1)*Nb-1])
    out = state
end
```

Figure 1 Pseudo code for the Cipher. Note that the array *w*[ ] contains the key schedule. *Nb* is the block size. *Nr* is the number of round. *AddRoundKey*, *SubBytes*, *ShiftRows* and *MixColumns* are subfunctions.

SubBytes Transformation

The SubBytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box), see Figure 2.

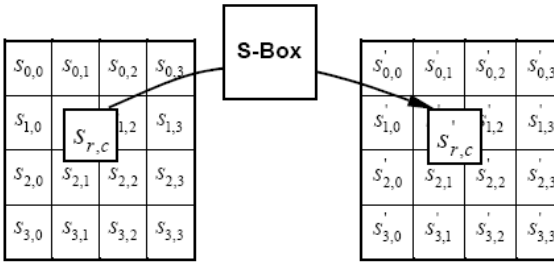

Figure 2 SubBytes applies the S-box to each byte of the state.

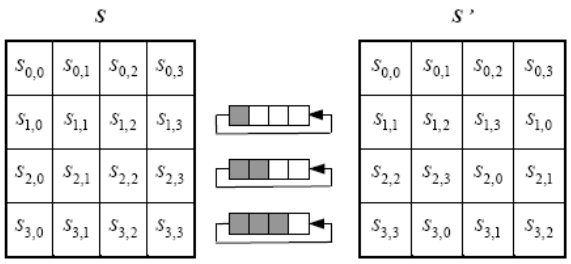

Figure 3 ShiftRows cyclically shifts the last three rows in the State.

ShiftRows Transformation

In the ShiftRows transformation, the bytes in the last three rows of the State are cyclically shifted over different number of bytes (offsets), see Figure 3.

MixColumns Transformation

In the MixColumns transformation, each column is treated as a four-term polynomial over GF(2^8) and multiplied modulo with a fixed polynomial *a(x)*, given by (1).

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

This can be written as a matrix multiplication (2).

$$Let \quad s'(x) = a(x) \otimes s(x)$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad for \quad 0 \le c \le Nb \quad (2)$$

AddRoundKey Transformation

In the AddRoundKey transformation, a Round Key is added to the State by a simple bitwise XOR operation.

2.2 Key Expansion

The expansion of the input key into the key schedule proceeds according to the pseudo code in Figure 4.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
    word temp
    i = 0
    while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
       i = i+1
    end while
    i = Nk
    while (i < Nb*(Nr+1))
      temp = w[i-1]
      if (i mod Nk = 0)
        temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and I mod Nk = 4)
        temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i+1
    end while
    out = state
end
```

Figure 4 Pseudo code for Key Expansion. Note that the array *w*[ ] contains the key schedule. *Nk* is the key size. *Nb* is the block size. *Nr* is the number of round. *SubWord* and *RotWord* are subfunctions.

SubWord is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function RotWord takes a word $[a_0, a_1, a_2, a_3]$ as an input, performs a cyclic permutation, and returns the word $[a_1, a_2, a_3, a_0]$. The round constant word array, Rcon[i], contains the values given by $[x^{i-1}, \{00\}, \{00\}, \{00\}]$, with x^{i-1} being powers of x (x is denoted as {02}) in the field $GF(2^8)$.

3. Design and Implementation

To implement a dynamic reconfigurable circuit on a FPGA device, the device should support partial reconfiguration. This allows some part of the circuit to be reconfigured while the remaining circuit is preserved. In this paper, the implementation is done on Xilinx FPGA Spartan-3 platform (XC3S200-4FT256) which supported partial reconfiguration.

For Spartan-3 FPGA, a reconfiguration data is loaded on a column-basis. This means that each reconfigurable module must occupied an entire column. Since Spartan-3 does not provide TBUFs, so the bus macro which is used to transfer data between fixed modules and reconfigurable modules have to be implemented by slices. The implementation of slices based bus macro is presented in [7]. The implemented bus macros are shown in Figure 5.

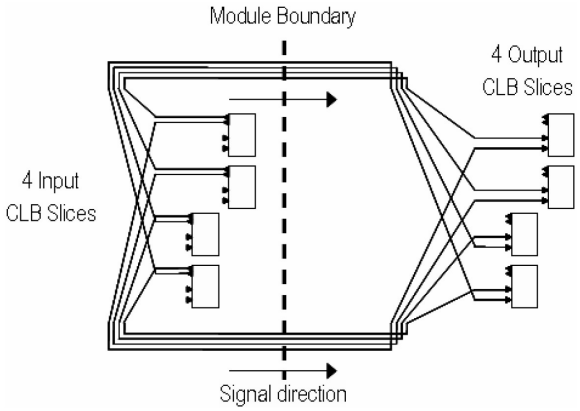

Figure 5 Basic 8-input, 8-output left-to-right bus macro

To make the simplest dynamic reconfigurable circuit, only one reconfigurable module is the best choice because the number of bus macro used in the circuit increased when the number of bit used by datapath and control is increasing. The 8-bit datapath is selected in order to minimize the number of bus macro used.

There are four main transformations of AES encryption algorithm so the AES encryption circuit can be separated into five modules. They are SBox module, MixColumn module, XOR module, Register module and Control Unit module. SBox module, MixColumn module and XOR module are reconfigurable modules because these three modules do not work concurrently. Two modules, Reconfiguration Control Unit module and Reconfiguration Program Memory module are necessary to make the circuit dynamically reconfigure itself. These modules are described in the following subsections.

3.1 SBox Module

SBox module is a reconfigurable module that performs SubBytes and SubWord transformations. This module is implemented by using a substitution table or SBox. SBox module requires 16 bus macros for datapath, 8 for 8-bit input and 8 for 8-bit output.

3.2 MixColumn Module

MixColumn module performs MixColumns transformation, and it is reconfigurable. Because the MixColumns transformation operated on 32-bit data, MixColumn module requires 4 clock cycles to produce first 8-bit output and two additional control signals to generate correct output. The design of MixColumn module is shown in Figure 6.

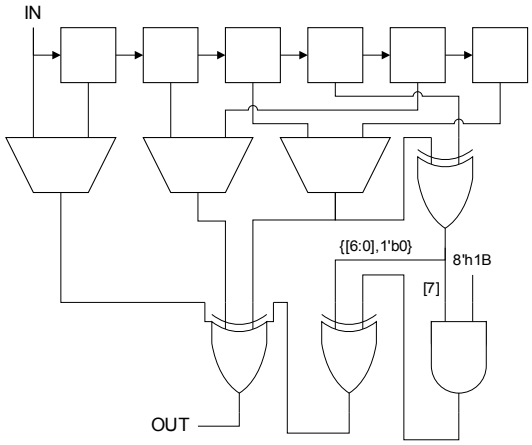

Figure 6 Design of MixColumn module

MixColumn module requires 16 bus macros for datapath, 8 for 8-bit input and 8 for 8-bit output, and it requires additional 2 bus macros for two control signals.

3.3 XOR Module

XOR module performs AddRoundKey and AddRcon transformation, and it is reconfigurable. This module is implemented by eight 2-bit XOR gates. XOR module requires 24 bus macros for datapath, 16 for two 8-bit input and 8 for 8-bit output.

3.4 Register Module

Register module stores current State and Key of AES encryption algorithm, it performs ShiftRows and RotWord transformations, and it also generates Rcon. This module is implemented by three 16x8-bit registers, 4x8-bit register, 10x8-bit ROM, eight 1-to-2 decoders, eight 4-to-1 multiplexors and eight 2-to-1 multiplexors.

Two 16x8-bit registers store State. The input of these two registers is coming from a decoder. ShiftRows transformation is done by select the right data from one register and sends it to another one. Another 16x8-bit register stores Key. A 4x8-bit register store temporary Key while doing Key Expansion and a 10x8-bit ROM stores Rcon.

Two multiplexors direct the correct data to two 8-bit bus macros. First bus macro accepts State, Key and temporary Key. Another bus macro accepts Key and Rcon.

3.5 Control Unit Module

Control Unit module is a one-hot finite state machine that controls the process of 8-bit datapath AES encryption. This module has 134 states as shown in Figure 7

In Figure 7, Read State, Read Key and Write State are 16-states that read input State, read input Key and write output State consecutively. ReconfigSBox, ReconfigMixColumn, ReconfigXOR are wait states that waiting for a completion of reconfiguration process.

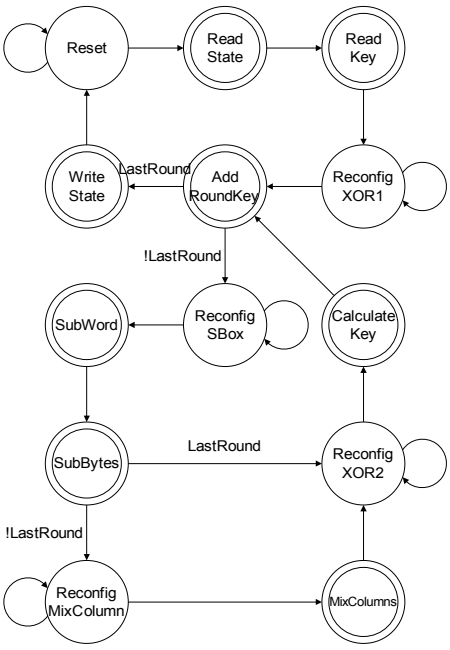

Figure 7 Control Unit module state diagram

3.6 Reconfiguration Control Unit Module

Reconfiguration Control Unit module is a finite state machine that controls the reconfiguration process. This module selects the correct reconfiguration program from Reconfiguration Program Memory and generates correct reconfiguration signals to reconfigure FPGA dynamically.

The dynamic reconfiguration of Spartan-3 is done by SelectMAP reconfiguration mode. This mode uses 8-bit wide data to reconfig FPGA, and requires some additional signals in order to perform reconfiguration. More information of Spartan-3 configuration can be found in 6.

3.7 Reconfiguration Program Memory Module

Reconfiguration Program Meomory module stores the programs of reconfigurable module. This module requires large amount of memory because the programs size is large. The best solution for this module is to implement it on an external RAM or ROM.

Since the bus macros of every reconfigurable module should be identical, the maximum number of bus macro is defined by XOR module and MixColumn module. In addition, three bus macros are added to ensure that the reconfigurable module is correct, and one bus macro is added to complete routing of unused signals. The total number of bus macro used in this circuit is 30. The final dynamic reconfigurable AES encryption's architecture is shown in Figure 8.

4. Implementation Result

After implementing the dynamic reconfigurable AES encryption circuit in XC3S200-4FT256 FPGA, the design occupies 359 slices. The maximum tested frequency is 50 MHz.

The throughput of the AES-128 encryption circuit can be approximately computed from the equation (3), while *cycle* is the number of clock cycles used to encrypt 128-bit data, and it is measured from the counter which is embedded in the circuit.

$$Throughput(bps) = 128 \times \frac{frequency}{cycle} \qquad (3)$$

The implementation result of the dynamic reconfigurable AES encryption is shown in Table 2. As shown in Table 2 the slowest process of dynamic reconfigurable AES encryption circuit is the reconfiguration, so the throughput is only 18 Kbps. If the reconfiguration delay is omitted, the throughput

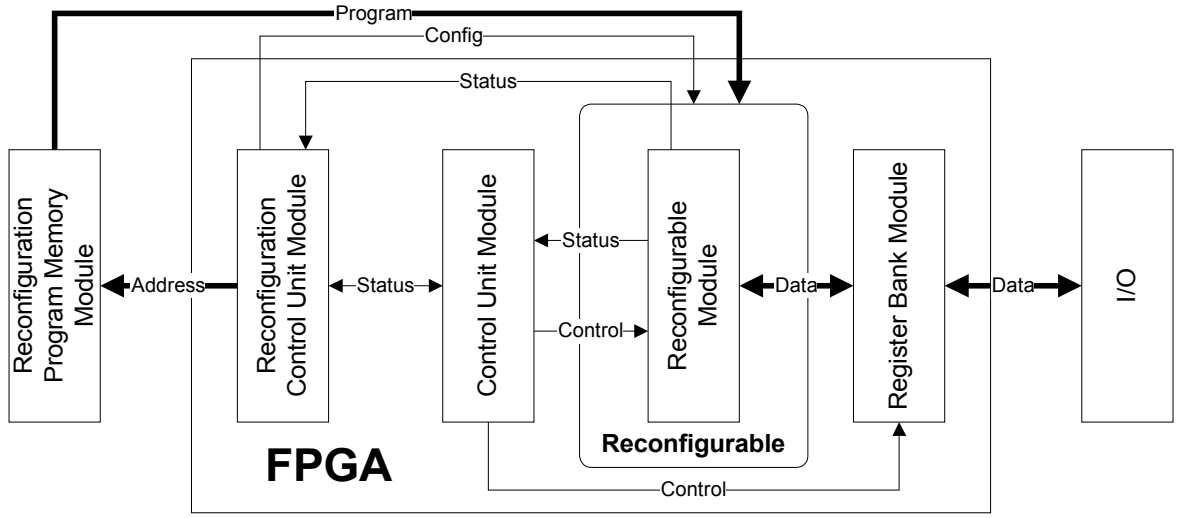

Figure 8 Dynamic reconfigurable AES encryption architecture

Table 2 Implementation result

Device	XC3S200-4FT256
Architecture	8-bit
Total No. of Slice used	359
Static module	232
SBox module	127
MixColumn module	103
XOR module	63
No. of BRAM used	0
No. of external RAM used	25872 bytes
Frequency	50 MHz
Throughput	17.895 Kbps
Total No. of Clock Cycle used	357,629
I/O	48
AES	717
Reconfiguration	356,864
Throughput (no reconfig delay)	16.121 Mbps

of 16 Mbps is achieved. These figures compared well to other works [4, 5, 6, 10, 11, 12].

5. Conclusion

This paper presented a dynamic reconfigurable AES encryption circuit which is suitable for a limited resource FPGA. The idea of dynamic reconfiguration can be adapted to reduce the resource used in very large circuit, but the speed of dynamic reconfigurable circuit is slowed by the reconfiguration process. Presently, the speed of configuration process is limited by the amount of configuration program bit and the configuration method. In this paper, SelectMAP, the fastest configuration method is used, and the reconfigure data is minimized by performing reconfiguration on only 4 columns of CLB which is the minimum number for partially reconfiguration of Spartan-3 FPGA device. In the future, if a faster dynamic reconfigurable FPGA device is produced, the performance of a dynamic reconfigurable circuit will be better than present.

This paper did not present a dynamic reconfigurable AES decryption circuit, but the decryption circuit can be implemented easily by a little modification to Control Unit and adding two reconfigurable modules to do InverseSubBytes and InverseMixColumns transformations.

6. References

[1] J. Daemon and V. Rijnmen, The design of Rijndael: AES-The Advanced Encryption Standard, New York, Sringer-Verlag, 2002.

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

[3] National Institute of Standards and Technology (NIST), Data Encryption Standard (DES), Federal Information Processing Standards (FIPS) Publication 46-3, 1999.

[4] P. Chodowiec and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm", Cryptographic Hardware and Embedded Systems 2003 (CHES 2003), LNCS vol. 2779, pp. 319-333, Springer-Verlag, Oct. 2003.

[5] T. Good and M. Benaissa, "AES on FPGA from the Fastest to the Smallest", Cryptographic Hardware and Embedded Systems 2005 (CHES 2005), LNCS vol. 3659, pp. 427-440, Springer-Verlag, 2005.

[6] Hodjat and I. Verbauwhede, “Area-Throughput Trade-offs for Fully Pipelined 30 to 70 Gbits/s AES Processors”, IEEE Transactions on Computer, vol. 55, pp. 366-372, Apr. 2006.
[7] P. Lysaght, B. Brodget, J. Mason, J. Young, and B. Bridgford, “Invited Paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAs”, International Conference on Field Programmable Logic and Applications, 2006
[8] Xilinx Inc., Spartan-3 Generation Configuration User Guide, v1.2, May. 2007.
[9] Guy Gogniat, Tilman Wolf, Wayne Burleson, “Reconfigurable Security Primitive for Embedded Systems”, SOCC 2005.
[10] Ming-Haw Jing, Zih-Heng Chen, Jian-Hong Chen, and Yan-Haw Chen, “Reconfigurable system for high-speed and diversified AES using FPGA”, Microprocessors and Microsystems Volume 31, Issue 2, 5 March 2007, Pages 94-102.
[11] O.Perez, Y.Berviller,C.Tanougast, and S.Weber, “The Use of Runtime Reconfiguration on FPGA Circuits to Increase the Performance of the AES Algorithm Implementation”, Journal of Universal Computer Science, vol. 13, no. 3 (2007), 349-362.