2020 Digital IC Design

Homework 4: RC4 Encrypt

1. Introduction

RC4 (Rivest Cipher 4) is a stream encryption algorithm with variable key length in cryptography. The same key is used for encryption and decryption, as it is a symmetric encryption algorithm. RC4 is the encryption algorithm used in Wired Equivalent Encryption (WEP) and was previously one of the algorithms used by TLS. But there are potential vulnerabilities in the RC4 algorithm. By February 2015, RFC 7465 prohibited the use of the RC4 encryption algorithm in TLS.

The RC4 algorithm consists of a pseudo-random number generator and XOR operations. The key length of RC4 is variable and up to 255 bytes. By given a key, the pseudo-random number generator generates an S box that will change during the encryption process and used to encrypt data. The RC4 encryption and decryption use the same set of algorithms as for the coincidence of XOR operation.

For this assignment, please implement an RC4 encryption and decryption circuit. The length of the key is fixed at 32 bytes, the length of the plain text is variable and up to 2048 characters, and the size of the S box is 64 bytes.

Use the input key to encrypt the plain text, and then output the encrypted characters, and then input the output encrypted characters to decrypt, and restore the original plain text.

2. Design Specifications

2.1 Block overview

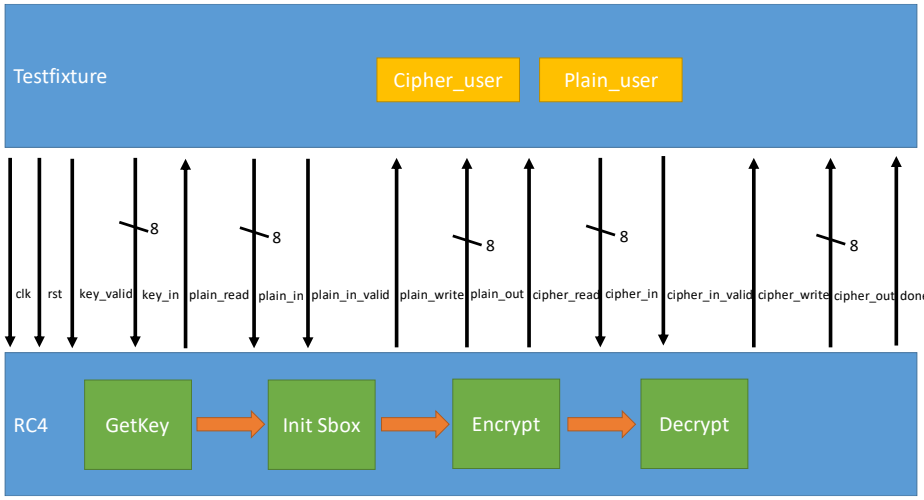

Figure 1 – System diagram

2.2 I/O Interface

Name	I/O	Width	Description
clk	I	1	System clock signal. This system is synchronized with the negative edge of the clock.
rst	I	1	High level NON-synchronous system reset signal
key_valid	I	1	When the key is ready, it will first set key_valid to high and then output the key value on the next negative edge.
key_in	I	8	The key data input signal line, the input data size is 8 bits, and will output at the next cycle that key_valid is set to high.
plain_read	O	1	When requesting plaintext, set plain_read to high and plain_write to low, and value will output at the next cycle.
plain_in	I	8	Input plain text data signal, which is composed of 8 bits unsigned integer.
plain_in_valid	I	1	The length of the plaintext is not fixed, plain_in_valid is set to high when the input plaintext is valid.
plain_write	O	1	To output the decrypted plaintext to TB memory, set plain_write to high. And set it to low when not writing.
plain_out	O	8	The decrypted result memory writing signal, which is composed of 8 bits unsigned integer (MSB), which is an unnumbered number.
cipher_read	O	1	When requesting ciphertext, set cipher_read to high and cipher_write to low, and value will output at the next cycle.
cipher_in	I	8	The input signal of ciphertext, which is composed of 8 bits unsigned integer.
cipher_in_valid	I	1	The length of the ciphertext is not fixed, cipher_in_valid is set to high when the input plaintext is valid.
cipher_write	O	1	To output the encrypted ciphertext to TB memory, set cipher_write to high. And set it to low when not writing.
cipher_out	O	8	The encrypted result memory writing signal, which is composed of 8 bits unsigned integer (MSB), which is an unnumbered number.
done	O	1	Set to high if all the works done. Make sure set to low when operating

2.3 Function Description

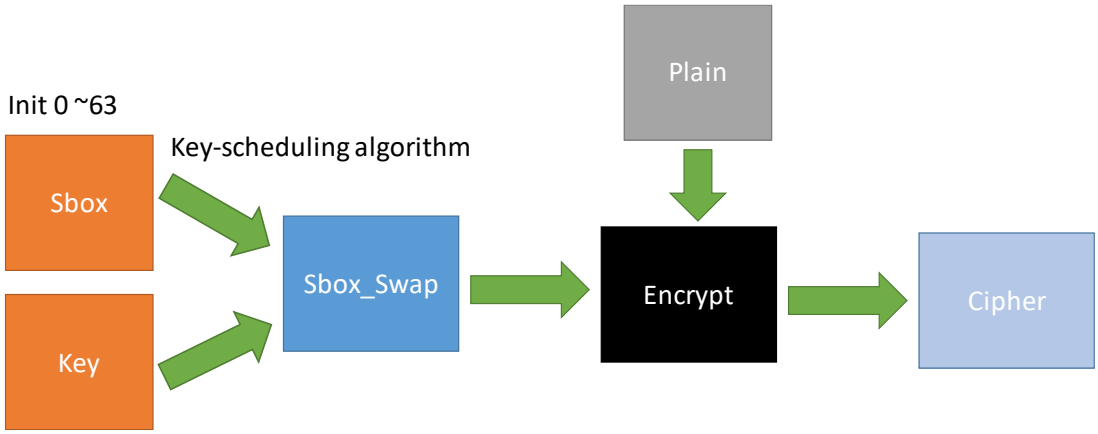

Figure 2 – Encryption flow

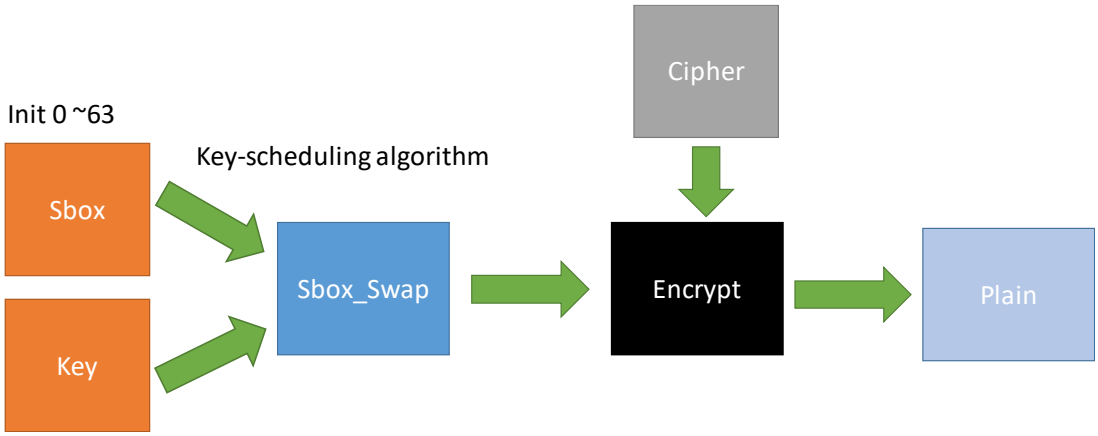

Figure 3 – Decryption flow

The key length of this system is 32, and the length of the plain text is up to 2048 characters. The key data is stored in the testfixture. After the system is reset, the next cycle will first set key_valid to high and then output the value of the key after another cycle. The key value is valid when key_valid is high but except for the first cycle. After the key value is input, shuffle the key and S box. S box is 0 ~ 63 at the beginning, use the following Pseudo codes shows in figure 4 to shuffle, and then the shuffled S box is used for encryption. The pseudo-code of the encryption algorithm is as shown in Figure 5. When the encryption is finished, use cipher_write and cipher_out to output the results to the memory in the testfixture. The input plaintext is valid when plain_in_valid is set to high.

And when input is completed, plain_in_valid is set to low. After the plaintext input is finished, cipher_read can be used to control the input of the ciphertext. (Note that when the encrypted plaintext is wrong, the input ciphertext is also wrong) The ciphertext is valid input when cipher_in_valid is set to high and set to low when the ciphertext is finished. The decryption algorithm flow is the same as encryption. If the system has completed the encryption and decryption, please set done to high to verify the encryption and decryption are correct.

```
for i from 0 to 63
    S[i] := i
endfor
j := 0
for i from 0 to 63
    j := (j + S[i] + key[i mod 32]) % 64
    swap values of S[i] and S[j]
endfor
```

Figure 4 – Key-scheduling algorithm (KSA)

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 64   //a
    j := (j + S[i]) mod 64 //b
    swap values of S[i] and S[j]  //c
    k := inputByte ^ S[(S[i] + S[j]) % 64]
    output K
endwhile
```

Figure 5 –Flow chart of encryption and decryption algorithm

Figure 6 shows that the time sequence of key_in when input. The input will start after the key_valid is set to high.

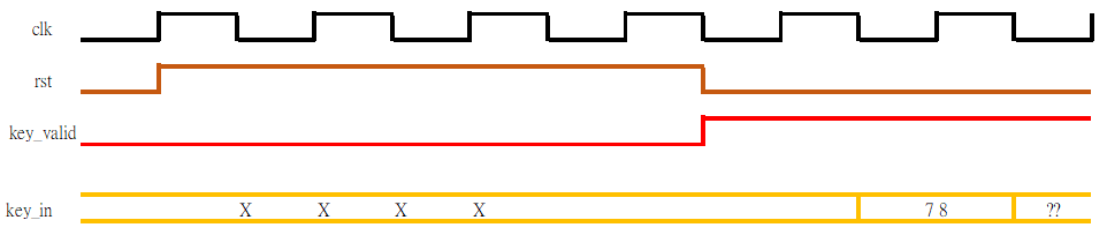

Figure 6 - Time sequence of key_in signal

Figure 7 shows that the time sequence of cipher data and plain data when reading. The input will start after the read is set to high.

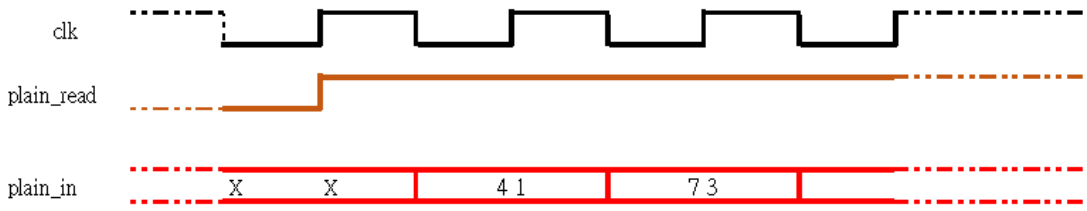

Figure 7 - Time sequence of plain data and cipher data signals

Figure 8 shows that the time sequence of cipher data and plain data when finishing. The data input is finished when plain_in_valid or cipher_in_valid is turn to low.

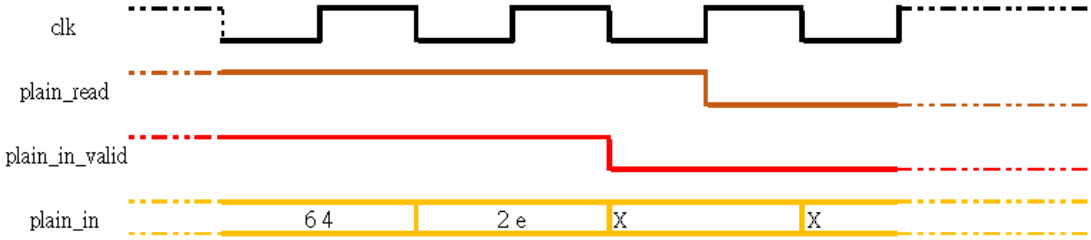

Figure 8 - Time sequence of plain data and cipher data signals when finishing

※NOTE: Both synthesis and post-synthesis simulation require a relatively long time.

3. Scoring

3.1 Functional Part **[60%, 30% of each TB]**

```
--------------------- T B 1 - S U M M A R Y -----------------

Congratulations! Cipher data have been generated successfully! The result is PASS!!

Congratulations! Plain data have been generated successfully! The result is PASS!!

** Note: $finish    : C:/Users/dic/Desktop/IC/0420/testfixture.v(244)
   Time: 44445 ns  Iteration: 2  Instance: /testfixture

--------------------- T B 2 - S U M M A R Y -----------------

Congratulations! Cipher data have been generated successfully! The result is PASS!!

Congratulations! Plain data have been generated successfully! The result is PASS!!

** Note: $finish    : C:/Users/dic/Desktop/IC/0420/testfixture2.v(244)
   Time: 39505 ns  Iteration: 2  Instance: /testfixture2
```

3.2 Gate Level Part **[20%, 10% of each TB]**

3.2.1 Synthesis

Your code should be synthesizable. After synthesizing in Quartus, the file named *RC4.vo* and RC4.sdo will be obtained.

Device : Cyclone II EP2C70F896C8

3.2.2 Simulation

All of the result should be generated correctly using **RC4.vo** and **RC4.sdo**, and you will get the following message in ModleSim simulation. (There should be no setup or hold time violations.)

```
--------------------- T B 1 - S U M M A R Y -----------------

Congratulations! Cipher data have been generated successfully! The result is PASS!!

Congratulations! Plain data have been generated successfully! The result is PASS!!

** Note: $finish    : C:/Users/dic/Desktop/IC/0421/testfixture.v(244)
   Time: 157348706 ps  Iteration: 0  Instance: /testfixture

--------------------- T B 2 - S U M M A R Y -----------------

Congratulations! Cipher data have been generated successfully! The result is PASS!!

Congratulations! Plain data have been generated successfully! The result is PASS!!

** Note: $finish    : C:/Users/dic/Desktop/IC/0421/testfixture2.v(244)
   Time: 171983706 ps  Iteration: 0  Instance: /testfixture2
```

3.3 Performance Part **[20%]**

Finish the requirements in 3.2 and record the logic elements count, minimum cycle time that pass the simulation and total simulation time.

Scoring with:

(Total logic elements + total memory bit + 9*embedded multiplier 9-bit element) × (longest gate-level simulation time in ns)

Less is better

```
`timescale 1ns/10ps
`define CYCLE      30.0
`define End_CYCLE  100000
```

Change the cycle time by your design with CYCLE macro.

Flow Summary

Flow Status	Successful - Thu Apr 23 16:33:12 2020
Quartus II Version	10.0 Build 262 08/18/2010 SP 1 SJ Full Version
Revision Name	RC4
Top-level Entity Name	RC4
Family	Cyclone II
Device	EP2C70F896C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	3,246 / 68,416 (5 %)
Total combinational functions	3,244 / 68,416 (5 %)
Dedicated logic registers	512 / 68,416 (< 1 %)
Total registers	512
Total pins	50 / 622 (8 %)
Total virtual pins	0
Total memory bits	192 / 1,152,000 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

```
--------------------- T B 1 - S U M M A R Y -----------------

Congratulations! Cipher data have been generated successfully! The result is PASS!!

Congratulations! Plain data have been generated successfully! The result is PASS!!

** Note: $finish    : C:/Users/dic/Desktop/IC/0423/testfixture.v(244)
   Time: 91675293 ps  Iteration: 0  Instance: /testfixture

--------------------- T B 2 - S U M M A R Y -----------------

Congratulations! Cipher data have been generated successfully! The result is PASS!!

Congratulations! Plain data have been generated successfully! The result is PASS!!

** Note: $finish    : C:/Users/dic/Desktop/IC/0423/testfixture2.v(244)
   Time: 49435293 ps  Iteration: 0  Instance: /testfixture2
```

4. Submission

You should classify your files into three directories and compressed to .zip format. The naming rule is HW4_studentID_name.zip.

	RTL category
*.v	All of your Verilog RTL code
	Gate-Level category
*.vo	Gate-Level netlist generated by Quartus
*.sdo	SDF timing information generated by Quartus
	Documentary category
*.pdf	The report file of your design (in pdf).

4.1 Report file

Please follow the spec of report. You are asked to describe how the circuit is designed as detailed as possible, and the summary results and minimum CYCLE in post-sim are necessary.

4.2 Please submit your .zip file to folder HW4 in the moodle.

Deadline: 2020-05-26 23:55

5. If you have any problem, please contact by the TA by email

weiting84610@gmail.com 陳威廷

f74044088@gs.ncku.edu.tw 倪祺婷

Appendix

The Sbox value in ***testbench1***:

[56, 31, 12, 3, 20, 0, 28, 11, 30, 40, 52, 46, 37, 57, 50, 43, 16, 48, 26, 14, 62, 29, 41, 17, 47, 9, 23, 4, 45, 36, 33, 19, 55, 10, 35, 24, 53, 34, 7, 44, 1, 27, 13, 60, 21, 8, 18, 6, 58, 49, 54, 2, 15, 42, 22, 63, 38, 51, 61, 59, 39, 25, 5, 32]

The Sbox value in ***testbench2***:

[7, 13, 8, 12, 30, 44, 18, 37, 24, 29, 26, 51, 60, 20, 4, 34, 14, 32, 46, 2, 17, 53, 63, 3, 62, 6, 25, 5, 56, 21, 52, 1, 43, 16, 47, 35, 11, 61, 58, 57, 33, 27, 10, 42, 28, 19, 15, 36, 23, 59, 54, 40, 41, 45, 39, 22, 38, 48, 9, 50, 55, 0, 31, 49]