2020 Digital IC Design Homework 4: RC4 Encrypt

NAME	Tran Thi Ai
Student ID	P76087081

Simulation Result

Functional simulation	Pass	Gate-level simulation	Pass	Gate-level simulation time	TB1: 145020551 ps TB2: 77222051 ps

(your pre-sim result)

```
#  ---------------- Cipher is correct ! ----------------
#  ---------------- Plain is correct ! -----------------
# ------------------------------------------------------------
#
# --------------------- 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    : D:/13_Digital_IC_Design/HW4/Ai/model1/testfixture.v(244)
#    Time: 179025 ns  Iteration: 2  Instance: /testfixture

 ---------------- Cipher is correct ! ----------------
 ---------------- Plain is correct ! -----------------
------------------------------------------------------------

--------------------- 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    : D:/13_Digital_IC_Design/HW4/Ai/model2/testfixture2.v(244)
   Time: 31515 ns  Iteration: 2  Instance: /testfixture2
```

(your post-sim result)

```
---------------- Cipher is correct ! ----------------
---------------- Plain is correct ! -----------------
------------------------------------------------------------

--------------------- 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    : D:/13_Digital_IC_Design/HW4/Ai/gate1/testfixture.v(244)
   Time: 145020551 ps  Iteration: 0  Instance: /testfixture
 ---------------- Cipher is correct ! ----------------
 ---------------- Plain is correct ! -----------------
------------------------------------------------------------

--------------------- 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    : D:/13_Digital_IC_Design/HW4/Ai/gate2/testfixture2.v(244)
   Time: 77222051 ps  Iteration: 0  Instance: /testfixture2
```

Synthesis Result

Total logic elements	7,411 / 68,416 (11 %)
Total memory bit	192 / 1,152,000 (< 1 %)
Embedded multiplier 9-bit element	0 / 300 (0 %)

(your flow summary)

Flow Summary

Flow Status	Successful - Tue Jun 02 18:32:12 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	7,411 / 68,416 (11 %)
Total combinational functions	7,411 / 68,416 (11 %)
Dedicated logic registers	1,084 / 68,416 (2 %)
Total registers	1084
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 %)

Description of your design

RC4 is a stream cipher that is used for generating pseudorandom stream of bits (a keystream). This generated key is combined with the plain text using bit-wise xor to perform encryption or combined with the cipher text to perform decryption, therefore the same algorithm is used for both encryption and decryption. A control FSM is used to control the state. The steps are as follows:

Step 1: I implement check key_valid, if 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 (key_in) is input, we shuffle the key and the S box.

```
state_sbox: begin
    if (j == Size_Sbox) begin
        state   <= state_mix;
        k2    <= k2 + Sbox[k1] + data_key[k1[4:0]];
    end
    else begin
        Sbox[j] <= j;
        Sbox_pl[j] <= j;
        j       <= j+1;
end
end

state_mix: begin
    Sbox[k1]   <= Sbox[k2[5:0]];
    Sbox[k2[5:0]]   <= Sbox[k1];
    Sbox_pl[k1]   <= Sbox_pl[k2[5:0]];
    Sbox_pl[k2[5:0]]   <= Sbox_pl[k1];
    if (k1 == Size_Sbox - 1)begin
        state  <= state_cipher;
        k1          <= 8'b0;
        k2          <= 8'b0;
    end
    else begin
        k1          <=  k1 + 1;
        state  <=  state_sbox;

    end
end
```

Step 2: After finishing shuffle, we implemented the encryption. We used the pseudo_code of the encryption algorithm. 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.

```
state_cipher: begin
    kl      = kl + 1;
    cipher_write <= 1'b0;
    k2     = k2 + Sbox[kl[5:0]];
    Sbox[kl[5:0]]    <= Sbox[k2[5:0]];
    Sbox[k2[5:0]]    <= Sbox[kl[5:0]];
    state             <= state_cipher2;
    plain_read    <= 1'bl;
    end

state_cipher2: begin
    plain_read    <= 1'b0;
    cipher_write <= 1'bl;
    state    <= state_cipher;
    if (!plain_in_valid) begin
        state <= state_plain;
        kl <= 0;
        k2 <= 0;
        cipher_write <= 1'b0;
    end
end
```

Step 3: After finishing the encryption, The state transfer into the next state, this is the decryption. In here, it used the decryption algorithm flow is the same as encryption. Then, set done to high to verify the encryption and decryption are correct

```
state_plain: begin
    kl      = kl + 1;
    plain_write <= 1'b0;
    k2     = k2 + Sbox_pl[kl[5:0]];
    Sbox_pl[kl[5:0]]    <= Sbox_pl[k2[5:0]];
    Sbox_pl[k2[5:0]]    <= Sbox_pl[kl[5:0]];
    state             <= state_plain2;
    cipher_read    <= 1'bl;

end

state_plain2: begin
    cipher_read    <= 1'b0;
    plain_write <= 1'bl;
    state    <= state_plain;
    if (!plain_in_valid && !cipher_in_valid) begin
        done <= 1;
        state    <= state_sbox;
    end

end
```

Step 4: I implemented assign it.

```
assign final_data1  = Sbox_pl[k1[5:0]] + Sbox_pl[k2[5:0]];
assign final_data  = Sbox[k1[5:0]] + Sbox[k2[5:0]];
assign cipher_out       = plain_in ^ Sbox[final_data[5:0]];
assign plain_out       = cipher_in ^ Sbox_pl[final_data1[5:0]];
```

This is the clock cycle modified when post-sim.

```
`timescale 1ns/10ps
`define CYCLE      24.3
`define End_CYCLE  10000
`define KEY         "Key_1.dat"
```

```
`timescale 1ns/10ps
`define CYCLE      24.5
`define End_CYCLE  10000
`define KEY         "Key_2.dat"
```

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