Group 58

Yanhao Gong 260546543
Yufei Liu 260561054

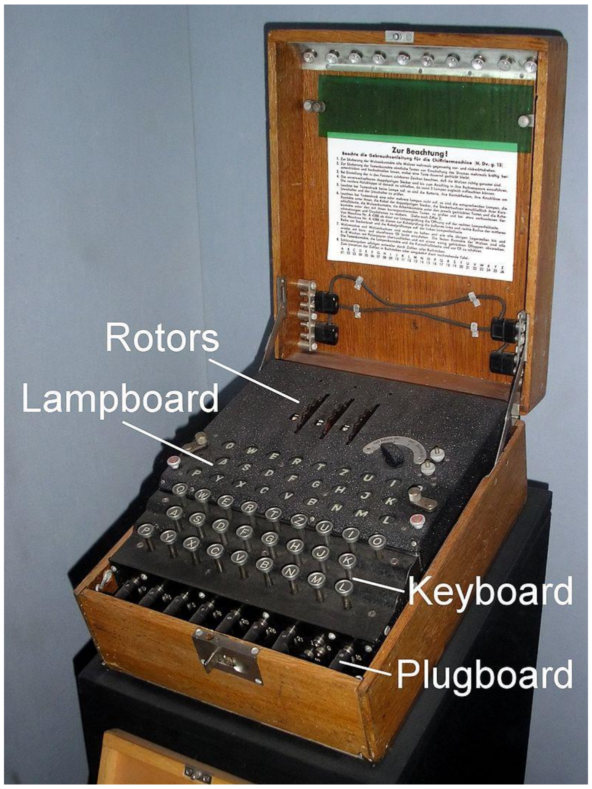
Lab5 Report

Enigma Machine

1. System Features

The Enigma machine is a cryptosystem that takes sets of letters as inputs by the senders, these inputs will be encrypted by the Enigma and output to the receivers based on the pre-set encryption settings. To decipher the encrypted outputs, the receivers just need to set the Enigma into the same encryption settings and then type those outputs back into the Enigma, the original inputs will be shown on the display.

The Enigma contains a stecker, a reflector and three rotors, which contribute to the complexity of encryption. Every input will be scrambled 9 times based on the encryption settings, which can also be customized by users. More details of how the system works will be discussed later. And a detailed user guide will also be given.



Military Enigma Machine (Wikipedia:https://en.wikipedia.org/wiki/Enigma_machine#/media/File:EnigmaMachineLabeled.jpg)

Yanhao Gong 260546543
Yufei Liu 260561054

2. Block Diagram

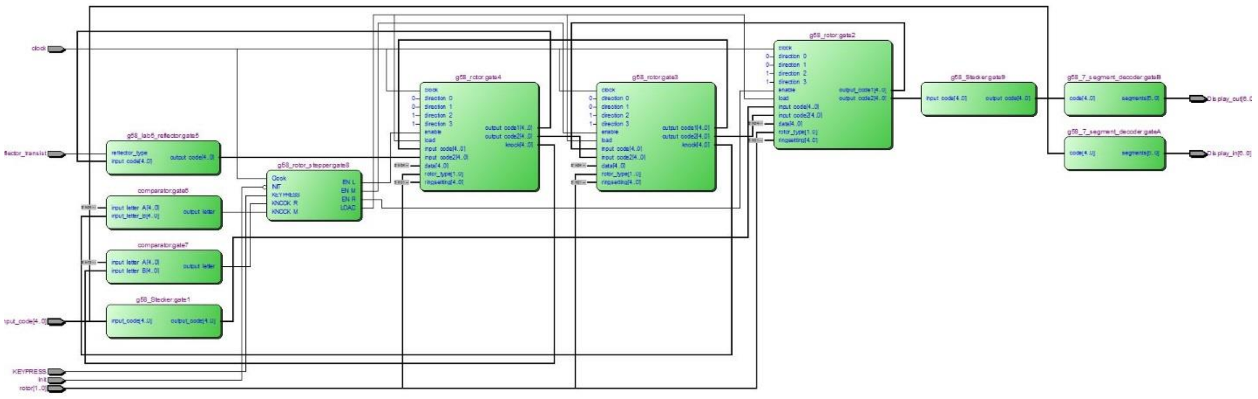


Figure 2.1 (Entire system)

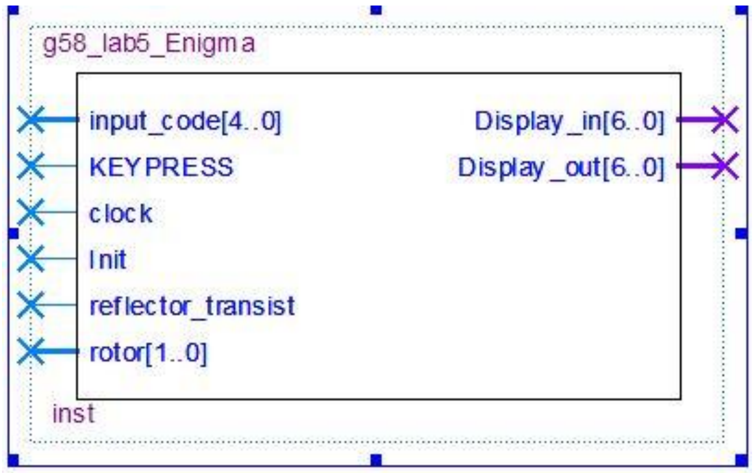
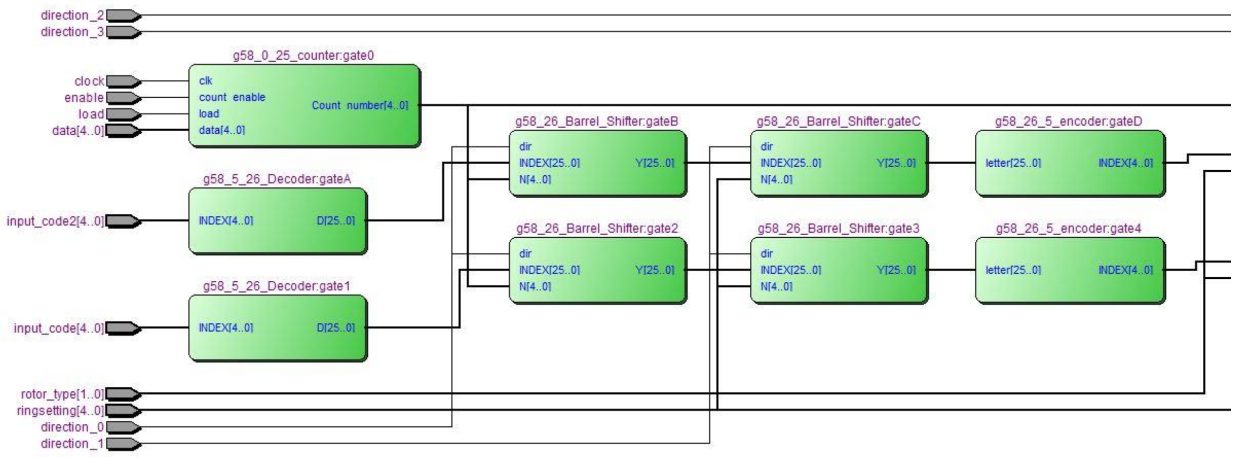


Figure 2.2 Symbol Diagram for Enigma Machine



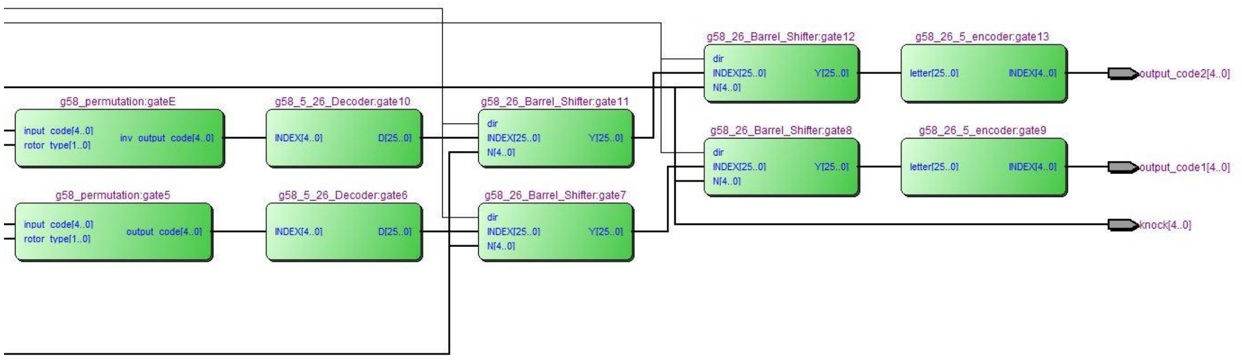


Figure 2.3 Block Diagram for rotor

3. Working Mechanism of the Enigma

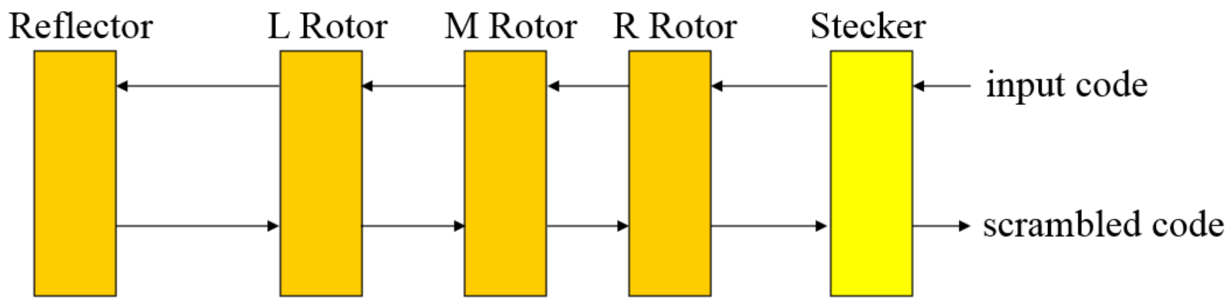


Figure 3.1

The complete Enigma System is shown as figure 2.1 and figure 3.1. The input goes into a Stecker and get scrambled by three rotors, then it feed into a reflector and reverse its direction. Now the input has been scrambled four times, it will go back into the 3 rotors and a Stecker and then output, the final output will be encrypted nine times.

The input first enters the stecker, and will be swapped one time after stecker. The swapped code will then enter the first rotor which is the right rotor. In the right rotor, the code will be scrambled as figure 3.2 shown

Yanhao Gong 260546543
Yufei Liu 260561054

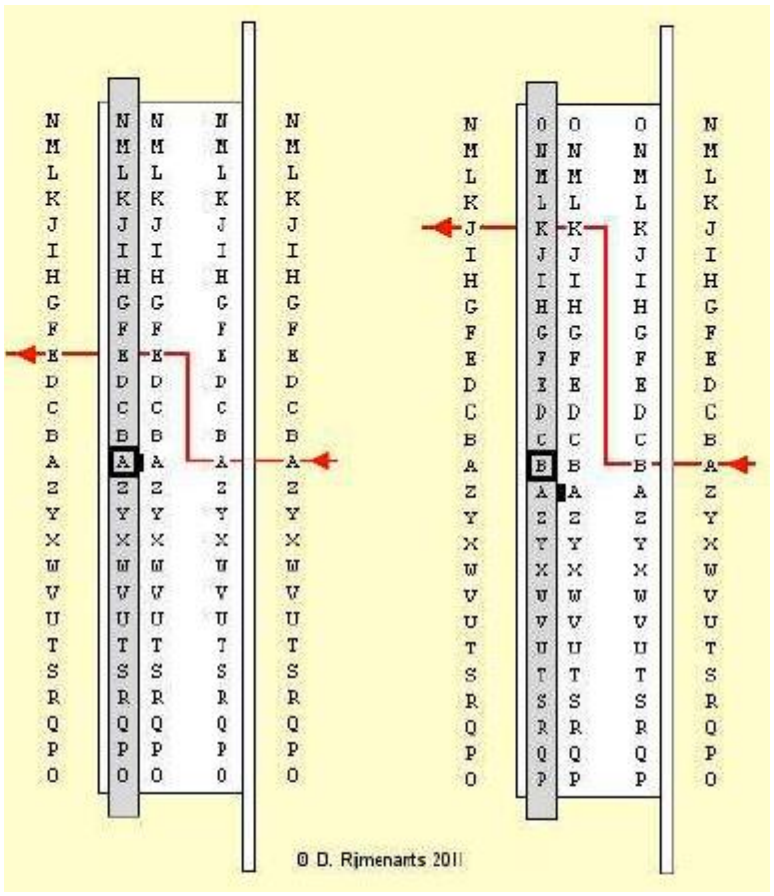


Figure 3.2

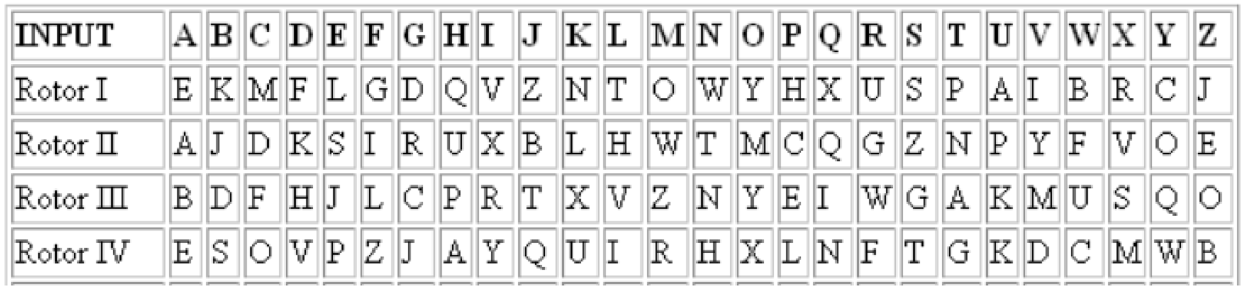
The code will then enter the middle rotor and left rotor. In each rotor, it will repeat the scrambling process descreibed above. Whther the rotor should rotate is determined by an enable signal and a pre set notch point. After the code get scrambled in each rotor, the code will be sent to comparator in which it will be compared with a notch point, if it matches, the result will be sent to the rotor stepper and the rotor stepper will generate that enable signal for rotors. According to the default rule, each time the key is pressed, the right rotor will rotate one position, rotation of the middle and left rotor is determined by the notch point. Once the code comes out the left rotor, it will enter the reflector which is like a fixed rotor, it will scramble the code one more time and reverse its direction. In the opposite direction, the same process will be repeated and the reciever will get the scrambled code.

1. Stecker

The Stecker is just the Plugboard shown at the first page, which swaps codes for a small number of code pairs. In our design, we defined 13 code pairs, which are:
(AB)(CD)(EF)(GH)(IJ)(KL)(MN)(OP)(QR)(ST)(UV)(WX)(YZ)
So if the input is A, it will be swapped to B through the Stecker.

2. Rotor

There are three rotors which are right, middle and left rotor. Each rotor contains 5-26 decoder, Barrel shifter, 26-5 encoder, permutation, and 0-25 counter, which are from previous labs. (Figure 3.3) The input goes into the 5-26 decoder first and will be shifted by two barrel shifters, after it get encoded by an encoder, it will be permutated by the permutation. The output of the permutation goes into the decoder and get shifted again, then it passing through an encoder and output. The permutation is an operation that maps a letter onto another letter in a one-to-one fashion. Different mapping methods will increase the complexity of the whole system, which we defined as rotor type. We used 4 rotor types which are as follow



INPUT	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Rotor I	E	K	M	F	L	G	D	Q	V	Z	N	T	O	W	Y	H	X	U	S	P	A	I	B	R	C	J
Rotor II	A	J	D	K	S	I	R	U	X	B	L	H	W	T	M	C	Q	G	Z	N	P	Y	F	V	O	E
Rotor III	B	D	F	H	J	L	C	P	R	T	X	V	Z	N	Y	E	I	W	G	A	K	M	U	S	Q	O
Rotor IV	E	S	O	V	P	Z	J	A	Y	Q	U	I	R	H	X	L	N	F	T	G	K	D	C	M	W	B

Figure 3.3

As we notice that each rotor has two inputs and two outputs, so in our design, we duplicated the path between input code and permutation (figure 3.4) and the path between permutation and output (figure 3.4). So that we could have two inputs that go through two permutations in different direction, which generate two outputs, one provides an output to the left, one provides an inverse output to the right. See figure 3.3, we combined the top and bottom rotor into one rotor.

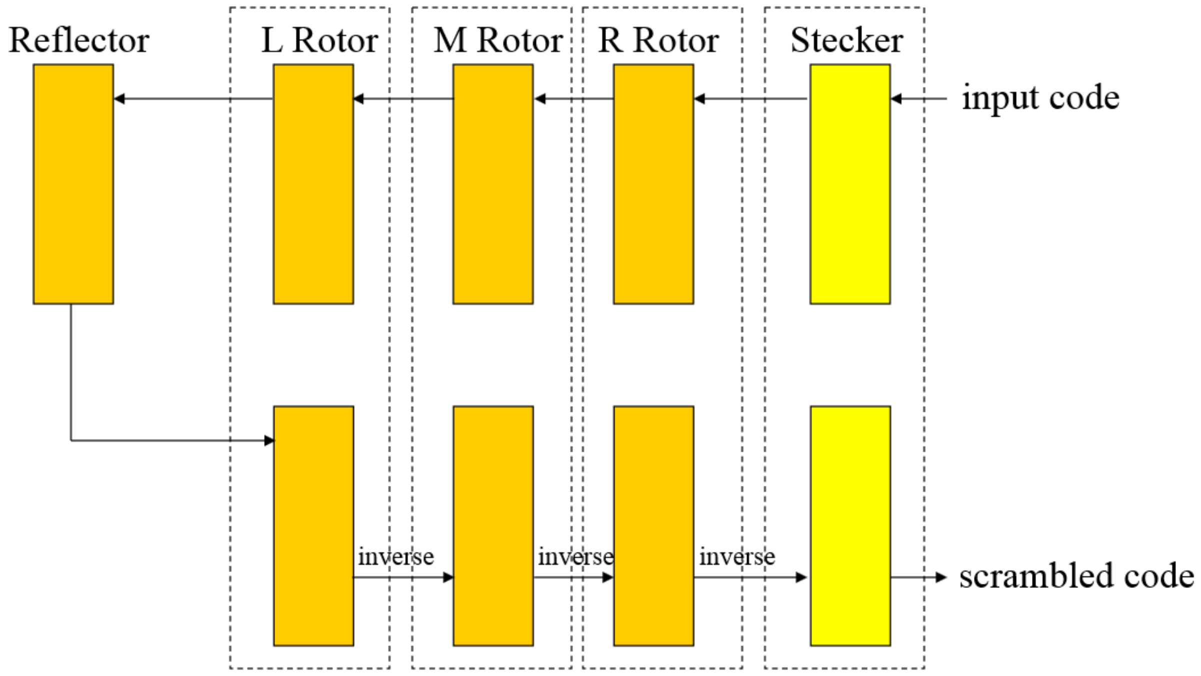


Figure 3.4

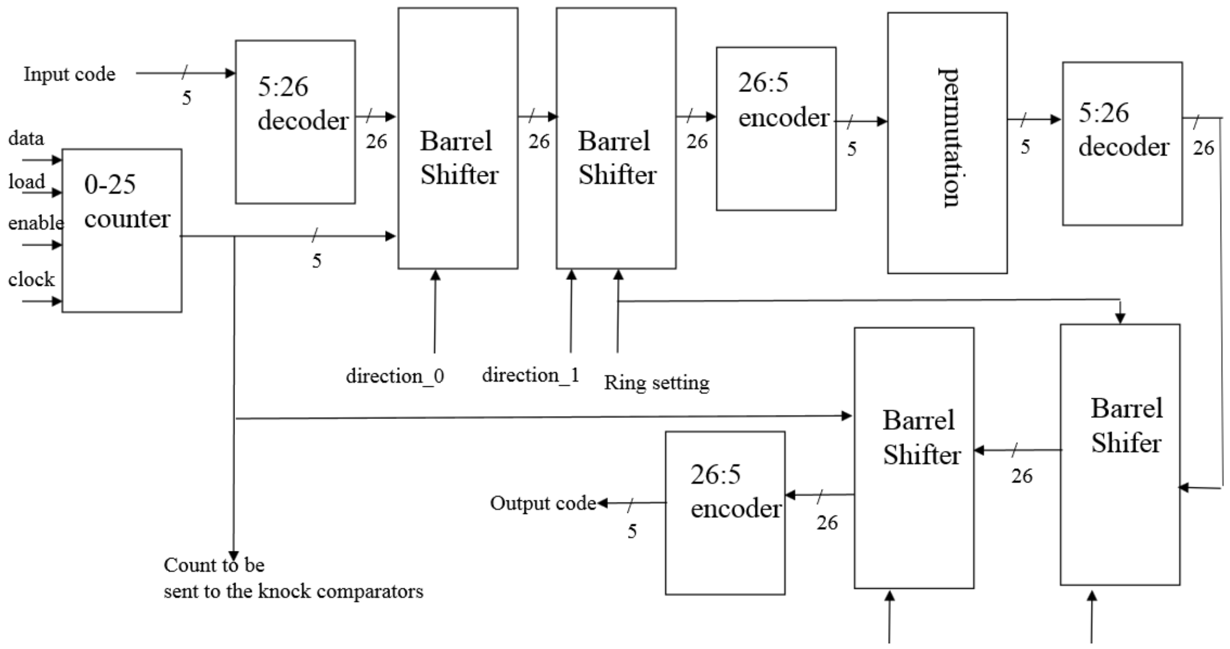


Figure 3.5

From figure 3.4, we can see that the output of the counter is sent to the comparator to compare. Since only middle and right rotor are related to the comparator, each output of the middle and right rotors' counter will connect to a comparator, in which the codes are compared and the comparison results will be sent to the Rotor Stepper which we designed at lab 4. The three enable outputs of the Rotor Stepper will feed back into the enable input of the rotor's counters. This loop ensures that every building blocks are mutual connected and affected.

There is also an additional complication called Ring Setting, which is like a movable outer ring, change the Ring Setting would change the position of the alphabet ring relative to the inner core. The inner core is consist of the two barrel shifters in figure 3.4.

Now, we have explained the working mechanism of one rotor, the output of this rotor will pass to the next rotor, and repeat the whole process described above.

3. Reflector

The reflector is like a fixed rotor, which scramble the code once more and passes the code back through the 3 rotors in reverse order. (Refer to figure 3.1) We defined two scramble rules as Reflector B and Reflector C, the rules are as follow:

reflector B	(AY) (BR) (CU) (DH) (EQ) (FS) (GL) (IP) (JX) (KN) (MO) (TZ) (VW)
reflector C	(AF) (BV) (CP) (DJ) (EI) (GO) (HY) (KR) (LZ) (MX) (NW) (TQ) (SU)

Figure 3.6

It is the reflector that makes the encryption process exactly as same as the deciphering process.

In conclusion, to ensure the massages are sent secretly and safely, the senders and receivers have to follow the same encryption setting, that is, the same rotor type, reflector type, direction of the Barrel shifter and the Ring Setting. And one can change these settings frequently to make the massages difficult to be deciphered by the enemies.

4. Users Guide

(1) Turn on the FPGA board by pressing [SW11](Red) button.
(2) Use the [SW9] button (leftmost) to switch the reflector type.
    SW9 down = reflector A (“0”)
    SW9 up = reflector B (“1”)

(3) You can choose the rotor type by using the [SW8] and [SW7] button which stands for “00”, “01”, “10” and “11”.
    SW8 down, SW7 down = rotor type I (“00”);
    SW8 down, SW7 up = rotor type II (“01”);
    SW8 up, SW7 down = rotor type III (“10”);
    SW8 up, SW7 up = rotor type IV (“11”);
(4) Once the reflector and rotor type is selected, press the third pushbutton from left to initial the entire Enigma Machine.
(5) Once the system is initialized, you can use [SW0] to [SW4] to input your code. You can input one letter every single time and the input will be displayed on the second right segment display. Then you need to press the last pushbutton to make the rotor turn and the output will be shown on the rightest segment display.
(6) Keep the same operation for your rest input letters.

Alphabet letters to binary code

Alphabet	Order	Switch Combination(1 = UP, 0 = DOWN) [SW4] to [SW0]
A	0	00000
B	1	00001
C	2	00010
D	3	00011
E	4	00100
F	5	00101
G	6	00110
H	7	00111
I	8	01000
J	9	01001
K	10	01010
L	11	01011
M	12	01100
N	13	01101

O	14	01110
P	15	01111
Q	16	10000
R	17	10001
S	18	10010
T	19	10011
U	20	10100
V	21	10101
W	22	10110
X	23	10111
Y	24	11000
Z	25	11001

Table 4.1

5. Simulation

Simulation of Stecker

As for our stecker, we defined 13 code pairs, which are:

(AB)(CD)(EF)(GH)(IJ)(KL)(MN)(OP)(QR)(ST)(UV)(WX)(YZ)

We gave an initial value "000000" to the input and used a for loop to increment it to "11001" which is Z in letter. The output should be swapped and also incremented with timeline. The result is shown below,

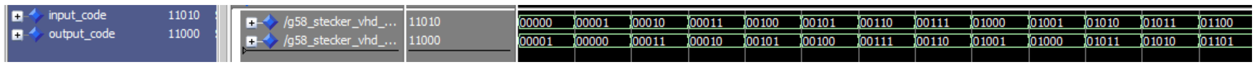


Figure 5.1

From figure 5.1, we can see that the result meet our expectation.

Simulation for Reflector

We have two type of reflector, which are shown at figure 3.6, we used two for loops in the test bench. The first loop is for reflector B, incrementing the input to 25 from 0, and then wait for 300 ns, switching to reflector C and repeat the for loop. So in this two loop period, we should have two sets of output. The result is shown below,

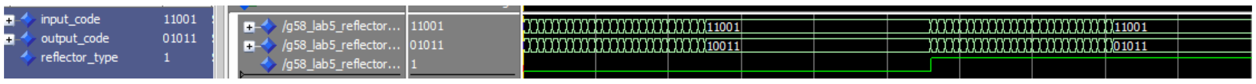


Figure 5.2

The detailed simulation is shown below, we changed he radix to unsigned, so that it is easier to see for grader.

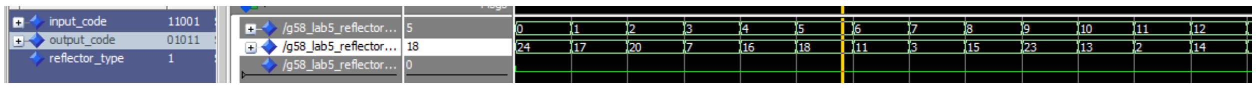


Figure 5.3

Simulation of the Enigma Machine

First of all, we give a clock which changes value every 20 ns. Then, we starts to test our circuit with default value "00011", "10010", "00011" "10010", and "00011" which refer to "DSDSD" respectively.

In our test-bench, we first set INIT to 1 which gives a value of '1' to load into the 0-25 counter. After 100 ns, change the value of INIT to 0(Figure 5.4). This operation will make the counter starting to count from the loaded number.

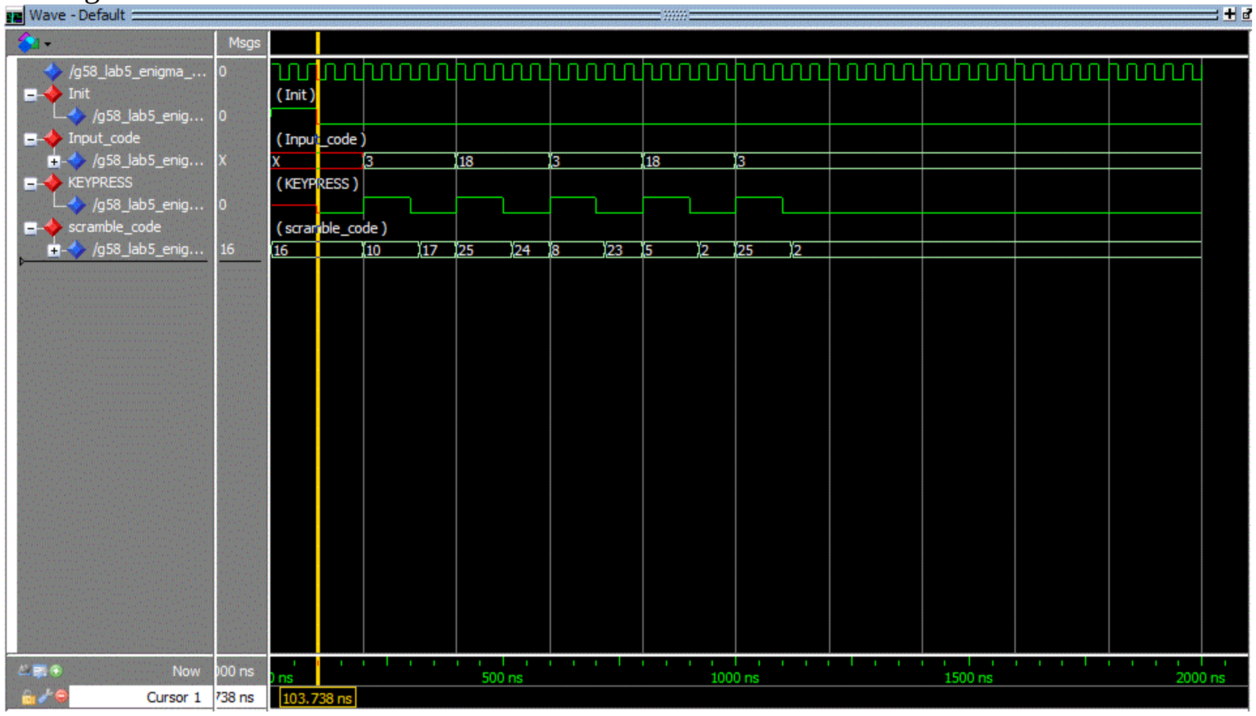


Figure 5.4

After setting the value of INIT, we set keypress to '0' and wait 100 ns and then gives our first input "00011". We must set keypress to '1' after giving the input to make sure the right rotor turns and give keypress back to '0' after 100 ns. Using the same vht description for next four inputs to finish our test bench. After simulating the circuit in Model-Sim, the result is shown below in Figure 5.5.

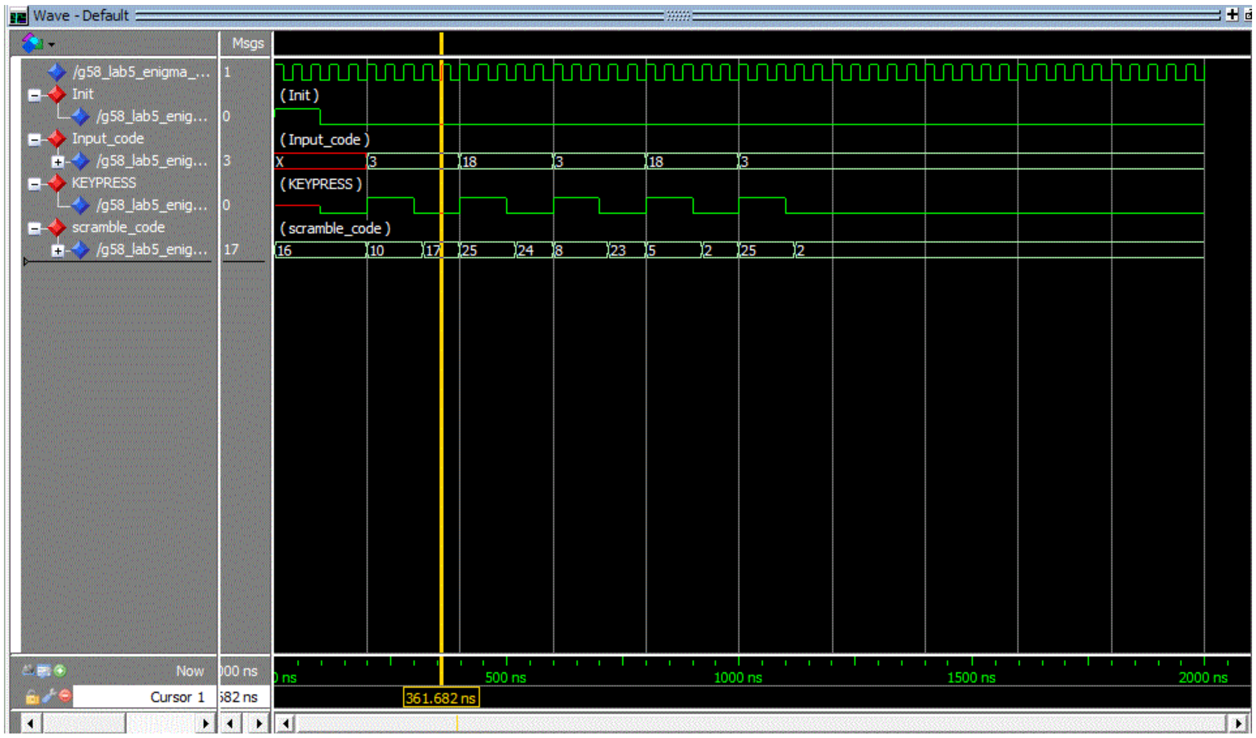


Figure 5.5

The inputs "DSDSD" maps onto '17', '24', '23', '2' and '2' which are "RYXCC". Then, we change the inputs in our test-bench to "RYXBB" and simulate it again in Model-Sim. The result is shown below in Figure 5.6.

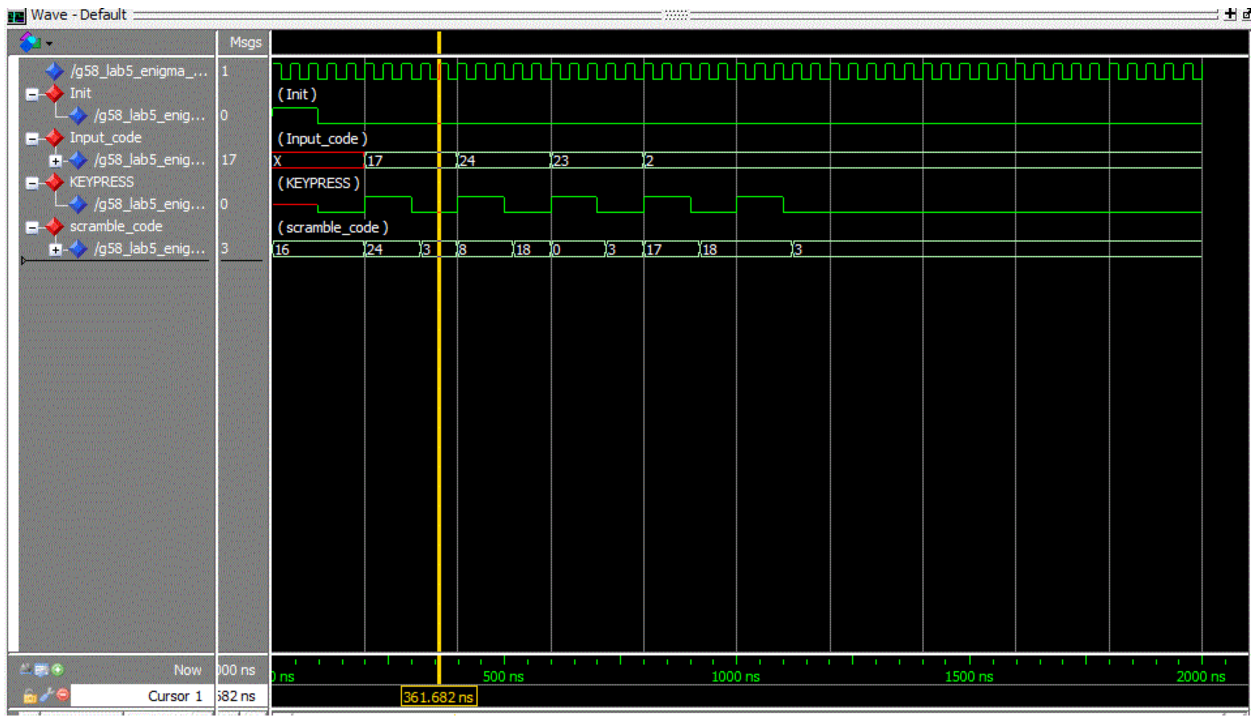
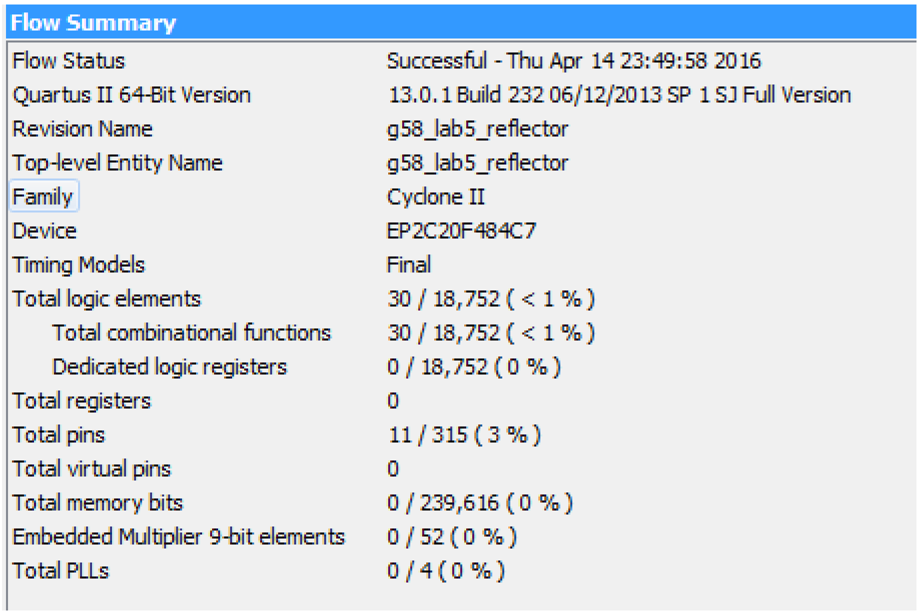


Figure 5.6

As we can see from Figure 5.6, after each time keypress is ‘1’, “RYXCC” is decoded back to “3-18-3-18-3” respectively which are “DSDSD”. As a result, the entire circuit is tested successfully.

6. FPGA Resource Utilization

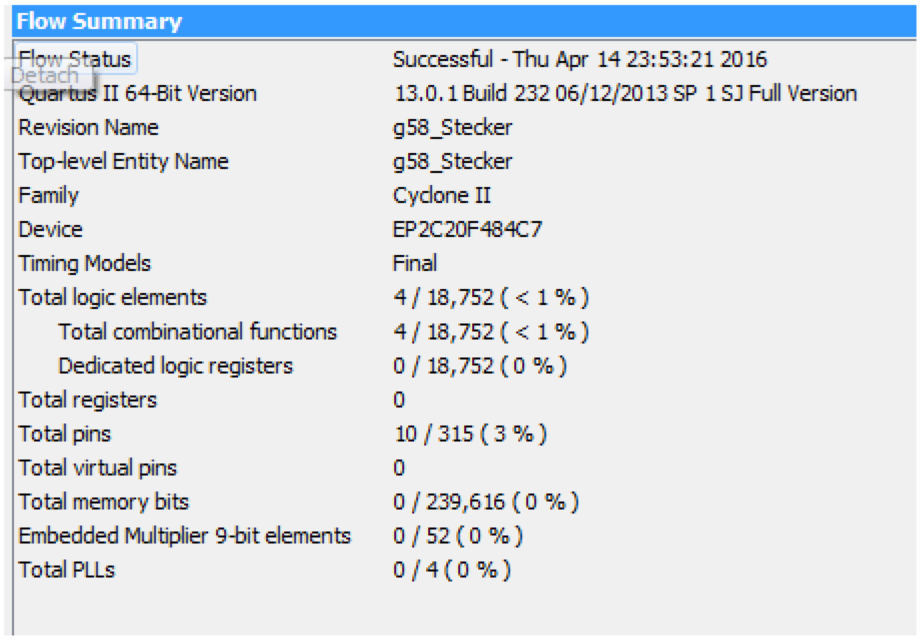
Reflector



Flow Summary	
Flow Status	Successful - Thu Apr 14 23:49:58 2016
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	g58_lab5_reflector
Top-level Entity Name	g58_lab5_reflector
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	30 / 18,752 (< 1 %)
Total combinational functions	30 / 18,752 (< 1 %)
Dedicated logic registers	0 / 18,752 (0 %)
Total registers	0
Total pins	11 / 315 (3 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 6.1

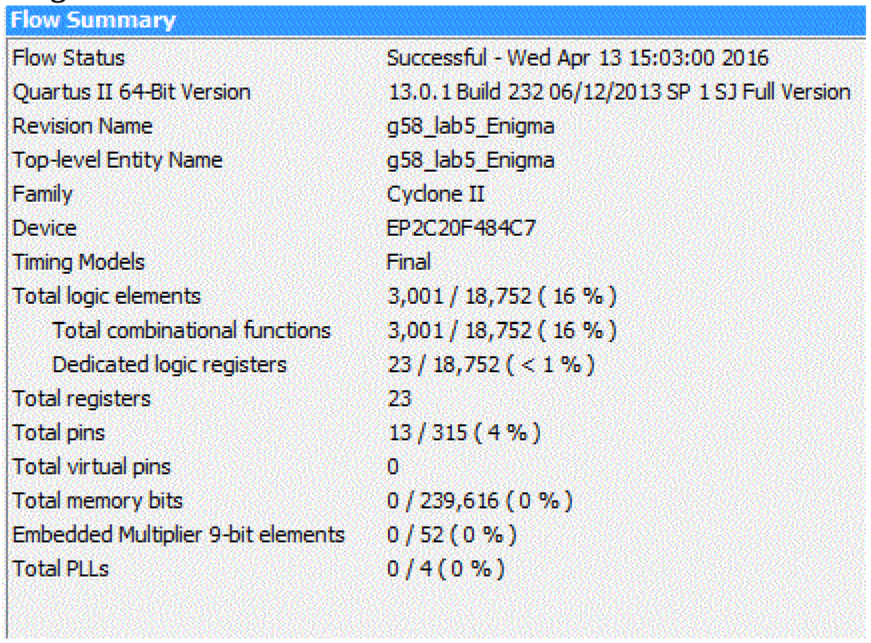
Stecker



Flow Summary	
Flow Status	Successful - Thu Apr 14 23:53:21 2016
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	g58_Stecker
Top-level Entity Name	g58_Stecker
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	4 / 18,752 (< 1 %)
Total combinational functions	4 / 18,752 (< 1 %)
Dedicated logic registers	0 / 18,752 (0 %)
Total registers	0
Total pins	10 / 315 (3 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 6.2

Enigma Machine



Flow Summary

Flow Status	Successful - Wed Apr 13 15:03:00 2016
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	g58_lab5_Enigma
Top-level Entity Name	g58_lab5_Enigma
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	3,001 / 18,752 (16 %)
Total combinational functions	3,001 / 18,752 (16 %)
Dedicated logic registers	23 / 18,752 (< 1 %)
Total registers	23
Total pins	13 / 315 (4 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 6.3

7. Timing

Enigma Machine

Fast model Hold Slack Value = 0.215

Slow Model Setup Slack Value = 1.657

Slow Model Fmax = 299.13 MHz

8. Overall conclusion

Learning outcome:

- How to write a basic VHDL code to realize specific functions.(lab1)
- How to write a test-bench for VHDL code.(lab1)
- How to simulate the VHDL code by using ModelSim.(lab1,lab2)
- How to put several building blocks together and make them work together to generate a new system and function.(lab3)
- How to design a testbed to ensure your design could work properly with no bugs.(lab3)
- How to make the timing analysis. (lab3)
- How to assign the outputs of VHDL code to the pin of the Altera board.(lab3)
- How to implement the circuits on Altera board.(lab3)
- How to design Finite State Machines using VHDL and use the SignalTap II logic analyzer.(lab4)
- How to get a complete digital system working on the Altera Board.(lab5)
- How to design the user interface and implement it on the Altera Board.(lab5)

Problems Arose:

The most frequently problems we met at the beginning are the VHDL grammar mistakes and the logic problems in the test bench. In the following labs, we paid enough attention on each part of the design so that there were few bugs.

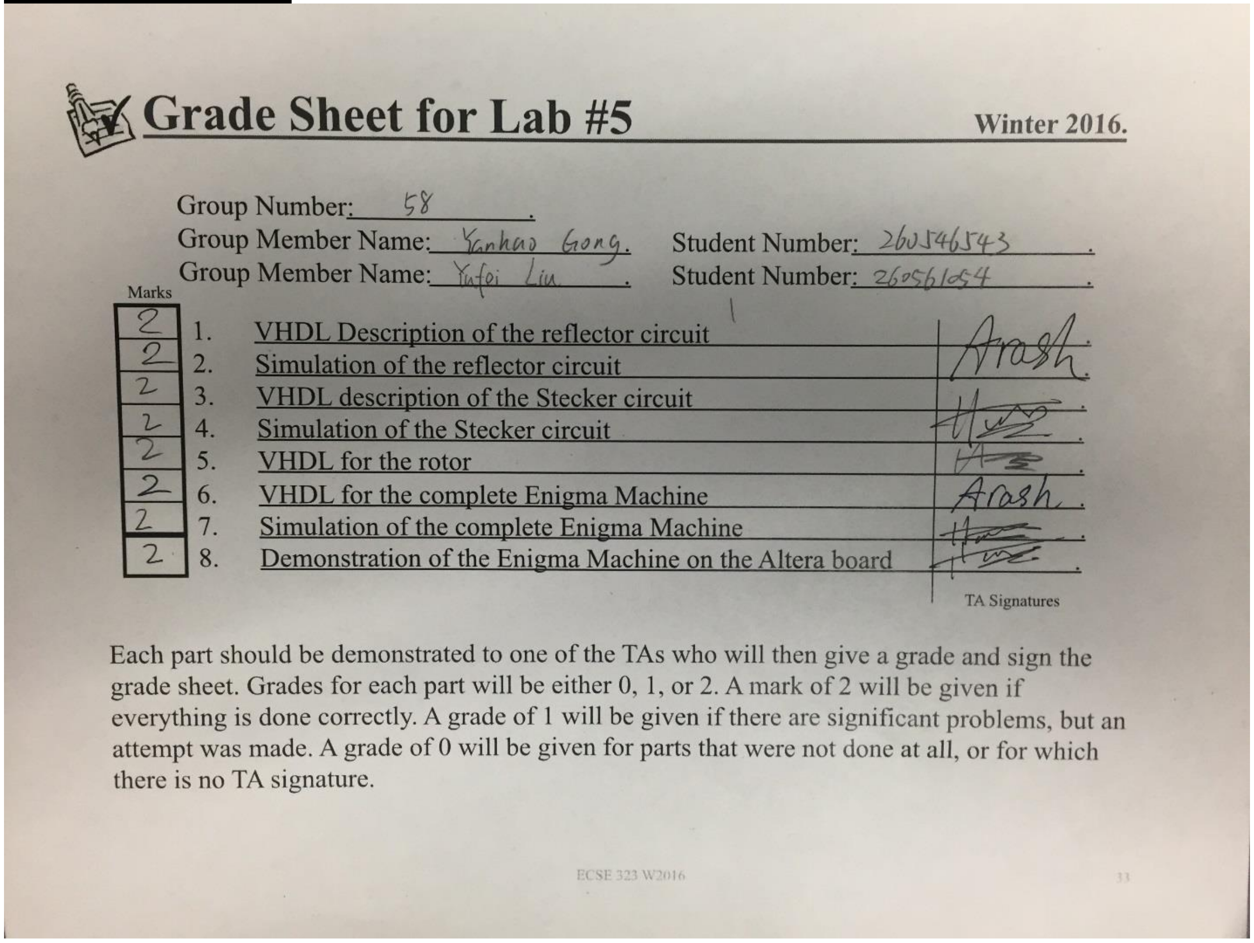
However, we did encounter a really tricky and tedious bug during lab 3, the implementation of the testbed on the Altera board. We did the simulation first and everything was working perfectly. But when we implemented it on the board, the letters on the 7-segments display did not count from A to Z, it totally messed up. We were so confused since the simulation was perfect, and there was no mistake on pin planner. We had been debugged it for several hours but got nothing. Then we asked the TA and friends in the lab for help, no one could fix it. Two days after the professor Gross came to the lab and we told him our situation, he told us to take apart the testbed and test it separately. After bunch of tests, finally we found that there was a tiny logic problem in the counter. In the process block of the counter, we should wrote "Rising_edge(clock)" instead of " clock = '1' ", or it will overlap infinitely with the next statement. But somehow it does not appear at the simulation.

After this issue, we realized that it is really important to think as an engineer, that is, think calmly and analyze calmly. If there are bugs, think about why there are bugs and what the sources of the bugs are, and then trace back to its sources.

9. Further Enhancements

1. Add more rotors to the Enigma Machine so that the complexity and safety of the system can be increased.
2. Make the notch points customizable, user can change the notch point by the interface.
3. Make the Ring Setting customizable, because of the switch constraint, we cannot assign another 5 inputs to the Altera board. But we can create an algorithm so that when we press a specific button, the system will jump into another state. In original state, the right 5 switches are assigned to the input with 5 bits, the left 5 switches are assigned to reflector type with 1 bit and rotor type with 2 bits. Now, if we press that specific button, we could enter a new state and the left 5 switches will be assigned to Ring setting with 5 bits.

10.Grading Sheet



Grade Sheet for Lab #5 **Winter 2016.**

Group Number: 58.
Group Member Name: Yanhao Gong. Student Number: 260546543.
Group Member Name: Yufei Liu. Student Number: 260561054.

Marks			TA Signatures
2	1.	VHDL Description of the reflector circuit	Arash
2	2.	Simulation of the reflector circuit	
2	3.	VHDL description of the Stecker circuit	[illegible]
2	4.	Simulation of the Stecker circuit	[illegible]
2	5.	VHDL for the rotor	[illegible]
2	6.	VHDL for the complete Enigma Machine	Arash
2	7.	Simulation of the complete Enigma Machine	[illegible]
2	8.	Demonstration of the Enigma Machine on the Altera board	[illegible]

Each part should be demonstrated to one of the TAs who will then give a grade and sign the grade sheet. Grades for each part will be either 0, 1, or 2. A mark of 2 will be given if everything is done correctly. A grade of 1 will be given if there are significant problems, but an attempt was made. A grade of 0 will be given for parts that were not done at all, or for which there is no TA signature.

ECSE 323 W2016 33