ECSE 323 Digital Systems Design Lab5 report

Group number: 32
Member name: Ziqi Wang 260569849 Yan Ren 260580535
System name: Enigma Machine

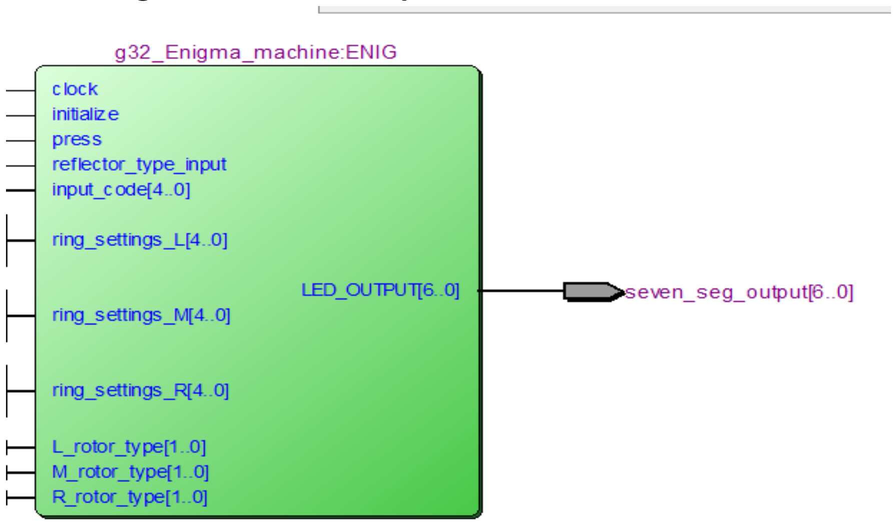
System features:

The Enigma machine we designed is basically a digital version of German military Enigma machine, which basically cipher a plaintext message into scrambled code. Compared to other encryption, our enigma machine is portable, reliable, easy-to-use, and upgradable. Portable: compared to the physical rotor mechanical machine, the digital one take advantage of the size and weight. We can use it on an Altera Board.

Reliable: The Enigma machine take the multiple cipher substitution to encrypt a message to the scrambled code. The encryption process is complicated and even two consecutive same letter will become different two letters in random. Besides, our machine provides 3 right setting choice from 26 and 4 rotor types to 3 rotors. And even more, we allow the users to preset the initial position of the rotors. Therefore, the machine originally provides $26^3*4^3*26^3$ = 19,770,609,664 possible initial combination encryption choice and it is even hard to crack even with the same machine we provide.

Easy-to-use: the enigma machine is basically encrypt and decrypt in the same manner. First you reset it machine to predefined initial condition as you like, and then put in the message or the scrambled code into the machine. Once you give message it yields the scrambled code, or you give scrambled code, the output should be the readable message. And to reset, you do not need to manually rotate the rotors to the initial condition. Instead, just press reset key. Upgradable: the machine is easy to upgrade, you can add more rotor types and the stecker type to make the machine more reliable or even change the rotor rotation direction

Block diagram of the entire system



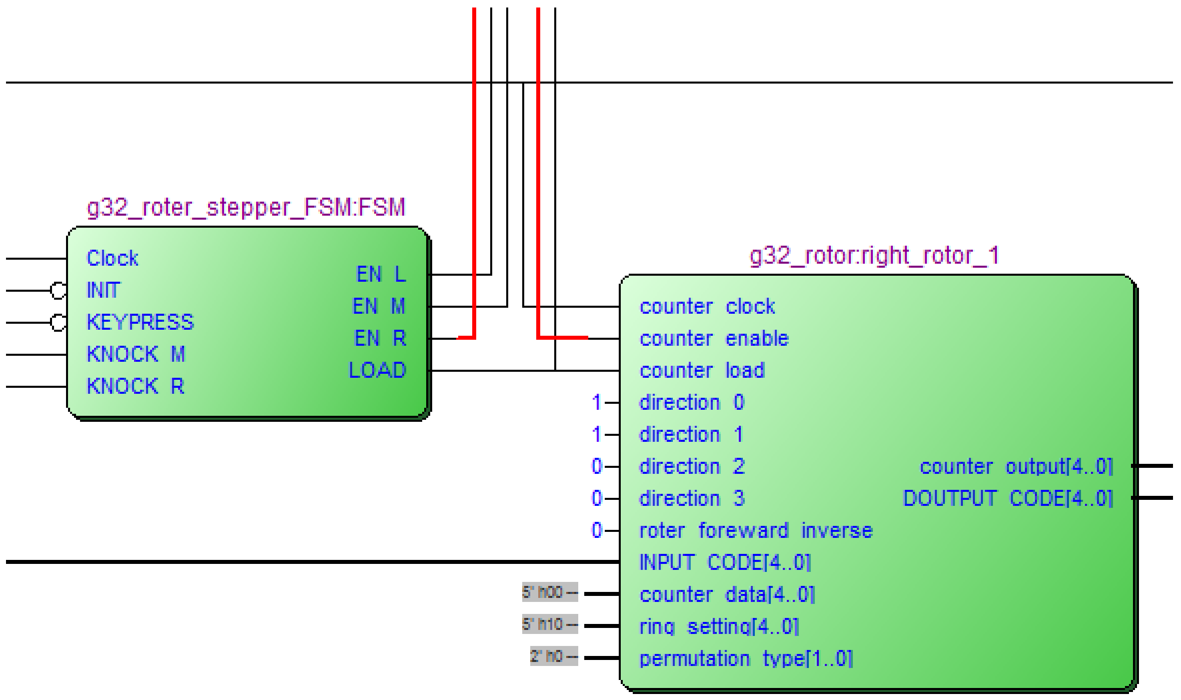
Diagram[1]: Enigma Machine block diagram

Detailed descriptions of the system

System overview

The whole system contains 3 rotor instances. Each one consists of two half rotor with different types, one is forward rotor for data coming into the Enigma Machine, the other one is the inverse rotor for data going out of the Enigma Machine. In other words, the letter code with go through each rotor and get scrambled twice. For three forward half rotors and three inverse half rotors, they all arrange in series as Right Rotor, Middle Rotor and Left Rotor. Combining with one reflector and two stecker, the complete data path for Enigma Machine is input data -> Stecker1 -> forward Right Rotor -> forward Middle Rotor -> forward Left Rotor -> Reflector -> inverse Left Rotor -> inverse Middle Rotor -> inverse Right Rotor -> Stecker2 -> output data.

To coordinate all six half rotors, one finite state machine is present, see diagram[2]. In this lab we, use the finite state machine from lab4. The output of EN_L connects to forward Left Rotor and inverse Left Rotor, output of EN_M connects to forward Middle Rotor and inverse Middle rotor and the output of EN_R connects to forward Right Rotor and inverse Right Rotor such that each pair of half rotors will rotate simultaneously in the same direction. The input code would be scrambled a total of 9 times. 6 times from the rotors, 1 from the reflector and 2 from the steckers.



Diagram[2]: FSM ---- rotor connection

Component details

Reflector

Reflector swaps pairs of codes. Thus, the reflector is behaved like a fixed rotor, which scrambles the code once more and passes the code back through the 3 rotors in reverse order. Because reflector is highly symmetric, thus the inverse output of the reflector will be exactly identically to the forwards one. Ant this will simplify the decryption process.

Reflection pairs refer to following table

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)

This table was taken from Tony Sale's web site: http://www.codesandciphers.org.uk/enigma/rotorspec.htm

Stecker

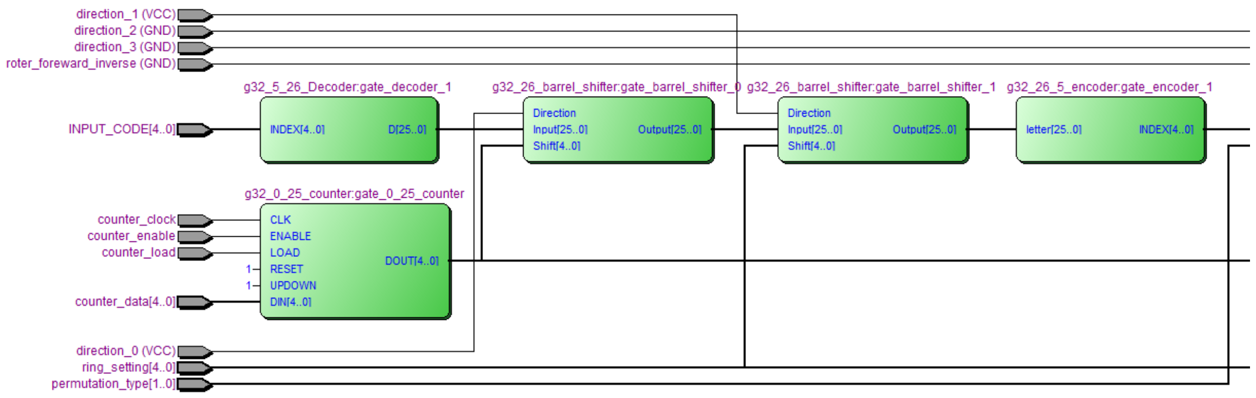
Stecker provides an additional scrambling step. User can define swap pairs of codes. In our design we use following pairs:

a <->k n <->l h <-> p g<->f x<->w i <-> o j <-> y

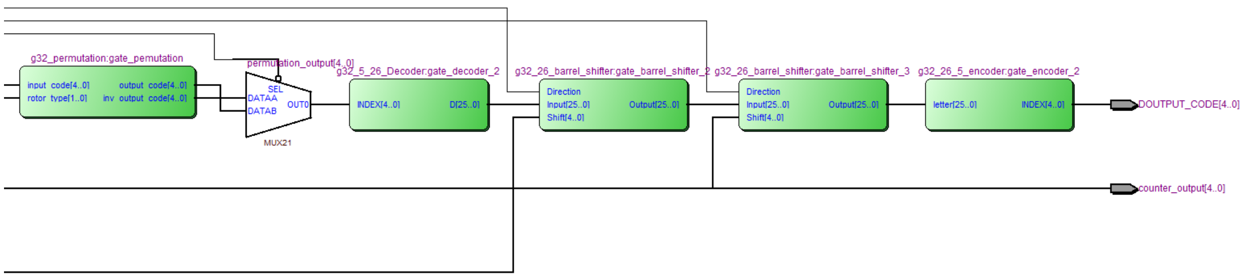
Rotor (one half)

A rotor consists of an inner core that the permutation, basically performance as a cipher map for each letter. There are four types of permutation, so there are also four types of rotor. The

difference of forward and inverse rotor is the permutation type inside, in which forward rotor use forward permutation and inverse rotor use inverse permutation. For example, if forward permutation maps ‘a’ to ‘e’, the corresponding inverse one maps ‘e’ to ‘a’. Inside one rotor, we have 2 pairs of barrel shifter. One pair performs ring setting function. They shift the same amount controlled by an input signal called “ring setting”. Another pair barrel shifter behaves like outer rotate ring. Their rotation is controlled by the output of 0-25 counter.
Following diagram[3],[4] is the components diagram for the rotor(half)



Diagram[3]: rotor part1

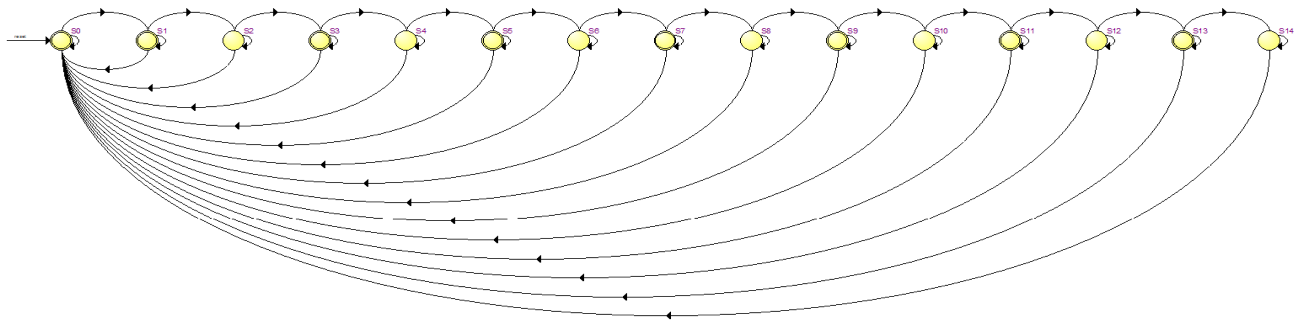


Diagram[4]: rotor part2

Descriptions of the user interface

The number of switches and buttons for an ALTERA board is limited and the basic configuration numbers of enigma inputs are large. Therefore, it is literally impossible to assign each input digit (std_logic) to an unique input pin. So, we decided to create an UI to wrap the enigma machine (as a component of UI) and provide users with better using experience.
The idea for the UI to reduce the necessary input pins is to make several pins used several times to configure several different setting. Therefore, we assign five toggle switches as “configuration” and a push button as “confirm” key. The FSM has a initial state “S0” where it resets configuration to default value (all rotor types: 1; all ring setting: A; reflector type: B). This is the state where the FSM start and whenever the user presses the “init”, the FSM will go back to S0 at the next clock edge. In the case where “init” is not high, each time user presses or releases the “confirm” key, the FSM will step into next state. The odd number of states (S1, S3, S5, S7, S9,

S11, S13) are the setting states where the FSM reads the “configuration” and loads it to corresponding input of enigma machine while the even number of states (S2, S4, S6, S8, S10, S12, S14 excluding S0) are the waiting state where FSM waits the users to change the toggle switches. In other words, the user presses ”confirm”, the FSM reads the configuration and once he/she releases it, the FSM waits he/she to change “configuration”. The loading order is: left rotor type(S1), mid rotor type(S3), right rotor type(S5), left ring setting(S7), mid ring setting(S9), right ring setting(S11) and reflector type(S13). Another thing needs to be emphasized is that some configuration may require less than 5 bits width signal, so we only take the proper number of digits from the least significant bit. After all configuration is settled, the FSM will stay in S14 and if the user pushes the “confirm” again, the FSM will go to initial state S0 and reset everything.
Following diagram[5] illustrates the user interface state diagram



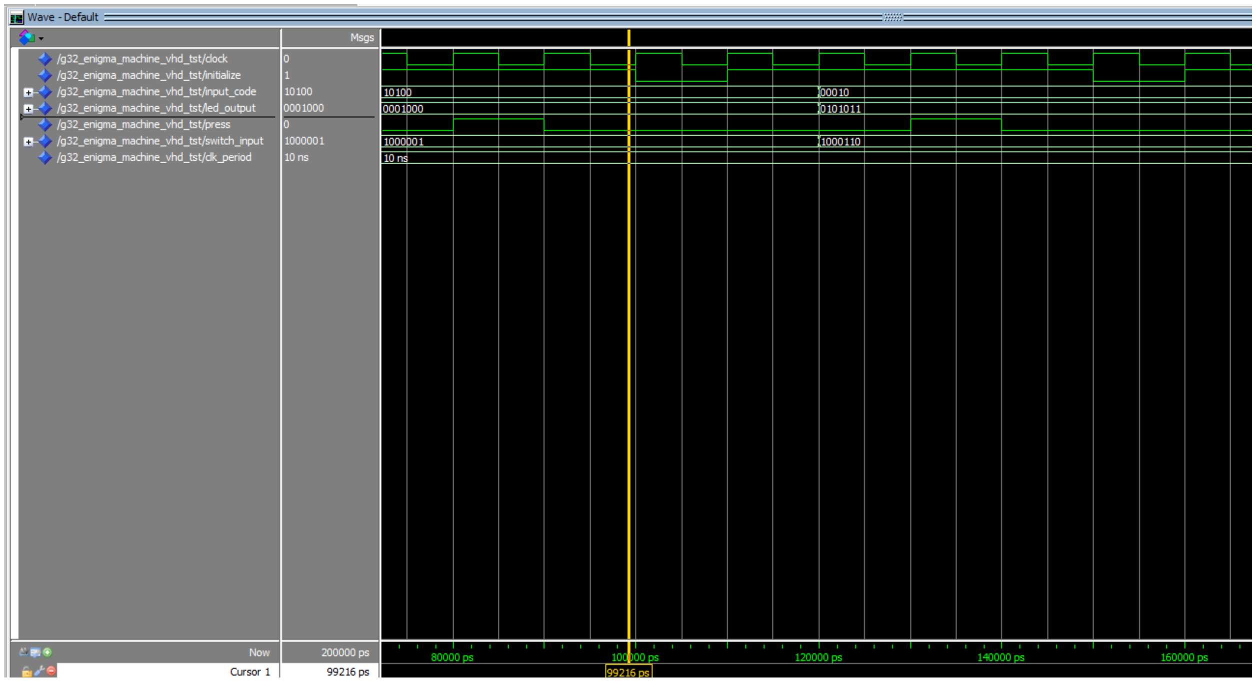
Diagram[5]: user interface FSM state diagram

UI user instruction:

1. Always initialize the enigma machine by pressing “init” before usage. Note you can step back to this step by press the same key.
2. If you want to keep the default configuration of the enigma machine (may reduce the confidence level), you can skip the configuration step (3 - 5)
3. Use last two toggle switches to select left rotor type and click (press and release) the “confirm” button to confirm. Repeat to select mid and right rotor type
4. Use last five toggle switches to select left rotor ring setting by letter’s 5 bit representation and click (press and release) the “confirm” button to confirm. Repeat to select mid and right rotor ring setting
5. Use last one toggle switch to select reflector type and click (press and release) the “confirm” button to confirm.
6. Press “confirm” one more time to set back to default value (same as press “init”)

System test

First we test the Enigma machine without user interface in ModelSim using test bench. Since user interface is removed. We need to preset the ring setting for each rotor pairs and rotor type for each rotor pairs in vhdl code. We set an input and take the record of the output. And then we initialize the system, keep the same setting. This time we pass the output we recorded previously. Check the output, if the system works correctly, we will get the same input as we pass the recorded output. Keep repeating this process, test all 26 characters. Then change the ring setting and rotor types to perform the same test under new settings. A typical simulation result is as following:



```
/g32_enigma_machine_vhd_tst/clock         0
/g32_enigma_machine_vhd_tst/initialize    1
/g32_enigma_machine_vhd_tst/input_code    10100
/g32_enigma_machine_vhd_tst/led_output    0001000
/g32_enigma_machine_vhd_tst/press         0
/g32_enigma_machine_vhd_tst/switch_input  1000001
/g32_enigma_machine_vhd_tst/clk_period    10 ns
Now                                       200000 ps
Cursor 1                                  99216 ps
```

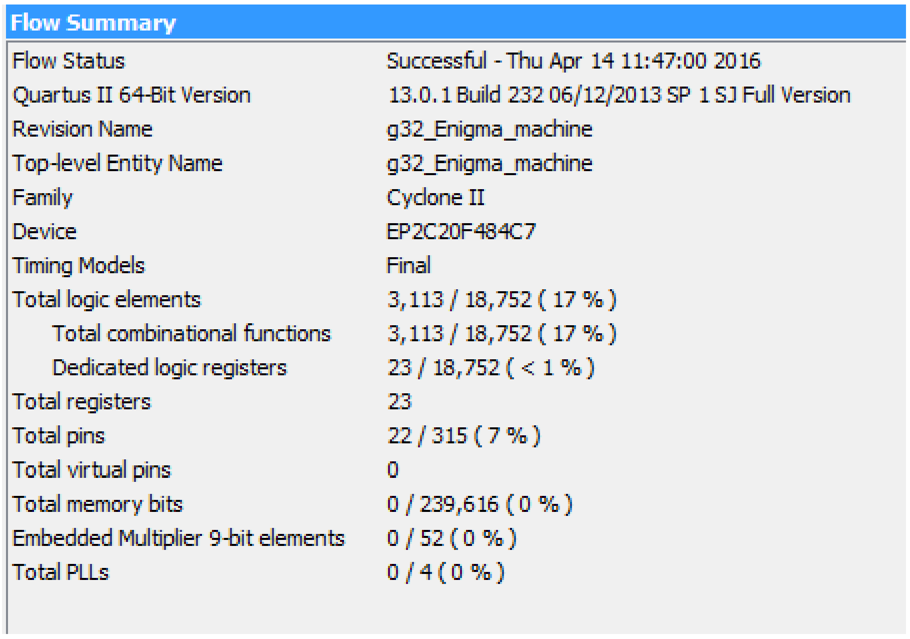
Diagram[6]: simulation wave diagram

Debugging suggestion:

When we find encoding and decoding results do not match, we usually check the outputs of each rotors. Therefor for debugging purpose, in VHDL code, we set output signals for each rotors' output. For most of the cases, the error exists in the Permutation part.

FPGA resource utilization and timing

Following diagram[7] illustrates the FPGA resource utilization



Flow Summary

Flow Status	Successful - Thu Apr 14 11:47:00 2016
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	g32_Enigma_machine
Top-level Entity Name	g32_Enigma_machine
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	3,113 / 18,752 (17 %)
Total combinational functions	3,113 / 18,752 (17 %)
Dedicated logic registers	23 / 18,752 (< 1 %)
Total registers	23
Total pins	22 / 315 (7 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Diagram[7]: FPGA compilation results

Our design uses 3113 total logic elements, 23 registers and 22 pins.

Timing analysis:

Create a “synopsys design constrains file” to specify the clock period: 20ns, frequency 50.0 MHz. From the timing analyzer result, observe in Slow Model, Fmax is 223.86 MHz, Setup Slack amount is 15.533, and Hold Slack is 0.445. In Fast Model, Setup Slack is 18.219 and Hold Slack is 0.215.

Conclusion

The main problem we encountered in lab5 is to debug the system in simulation. Likely most projects in industrial area, our enigma machine firstly failed to encrypt and decrypt the message properly. We initialized the configuration of the entire system and input a piece of message. After that, we collected the output letters and do the similar process to decrypt it. Unfortunately, we do not retrieve the origin message we encrypted. Therefore, we reviewed all component and figured out that the permutation in rotor has some errors. In the enigma machine, the permutation will be performed 6 times and even some tiny errors may be amplified largely. After fixing the problem, the system starts to work well.

Another problem we met is in design phase. Comparing to the mechanical rotor, the permutation and rotation in rotors works more complicatedly. Basically, before permutation, the input is already shifted while after permutation, the output is not shifted in the same manner, therefore, we add another two shifter to shift the output in each half rotor. However, the direction of these rotors is hard to decide. Then we use trial-and-error method to test out the correct direction. Similarly, the direction of the forwards and inverse rotors’ shifter directions.

The most promising feature could be added on for our enigma machine is to make full use of 6 seven-segment display LEDs. Our machine currently only apply the last one to show the letter input and letter output. Instead, we can fully use of board by making the first three LED sets only demonstrate the last 3 input letters and the last three LED sets only show the corresponding output letters. And once we input another new letter, all the previous letters shift to the left. Besides, currently, our enigma machine can only load “00000” - “A” for the rotors’ initial position, we may add more possible choices for users in the future. Additionally, we may allow the user to configure custom stecker in the future. Lastly, in the design, we have for each half of a rotor, we assign a counter. However, the counters for each rotor behaves the same. Therefore, we can get rid of three redundant counters in the rotors and make the counter unique to a single rotor.