ECSE-323
Digital System Design

Lab #5 – *System Integration for the Enigma Machine* Winter 2016

Prof. W. Gross and Prof. J. Clark

Introduction

In this lab you will put together all of the parts for the complete Engima Machine system and implement it on the Altera board.

Learning Outcomes .

After completing this lab you should know how to:

• Get a complete digital system working on the Altera board, and will have gained experience in implementing user interfaces.

Table of Contents .

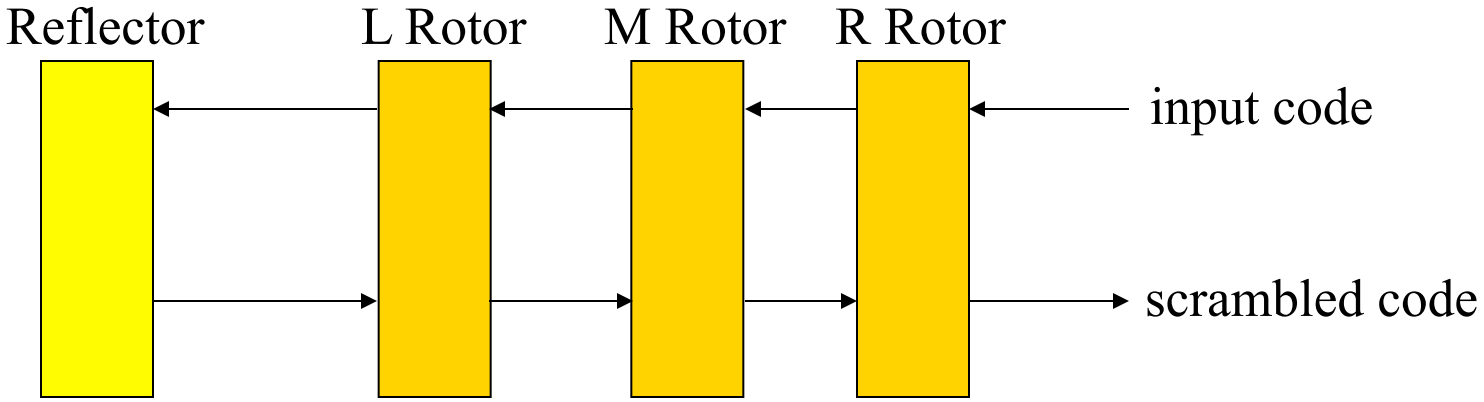
This lab consists of the following stages:

1. Design of a reflector circuit
2. Simulation of the reflector circuit
3. Design of a “Stecker” circuit
4. Simulation of the Stecker circuit
5. Design of the complete Enigma Machine system
6. Simulation of the Enigma Machine
7. Design of the Enigma Machine user interface
8. Testing of the Enigma Machine on the Altera board
9. Writeup of the lab report

1. The Reflector Circuit .

The Enigma machine consists of 3 rotors, which successively scramble an input character code, as well as a *reflector*. The reflector is like a fixed rotor, which scrambles the code once more and passes the code back through the 3 rotors in reverse order.

Thus there are a total of ***7*** scramblings that are done to an input code.



The reflector swaps pairs of codes. Thus, the action of the different reflector types in the Enigma Machine can be specified in the cycle representation:

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)
reflector B Dünn	(AE) (BN) (CK) (DQ) (FU) (GY) (HW) (IJ) (LO) (MP) (RX) (SZ) (TV)
reflector C Dünn	(AR) (BD) (CO) (EJ) (FN) (GT) (HK) (IV) (LM) (PW) (QZ) (SX) (UY)

For example, with the *B reflector*, **A** is reflected to **Y**, and **Y** is reflected to **A**.

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

Write a VHDL description for a circuit that implements the reflector operation. It should be able to specify the action of the ***first two*** reflector types (types B and C in the table on the previous page).

The circuit can be implemented using the approach taken in Lab 4 to make the permutation circuit (except that the inverse permutation is not needed).

Use the following entity declaration, and employ a process block, with nested case statements (the outer for the reflector type, the inner for the input_code).

```
entity gNN_reflector is
 port ( input_code          : in std_logic_vector(4 downto 0);
        reflector_type      : in std_logic;
        output_code         : out std_logic_vector(4 downto 0));
end gNN_reflector;
```

2. Simulation of the Reflector Circuit .

Once you have finished your VHDL description of the reflector circuit, show it to the TA and explain its design.

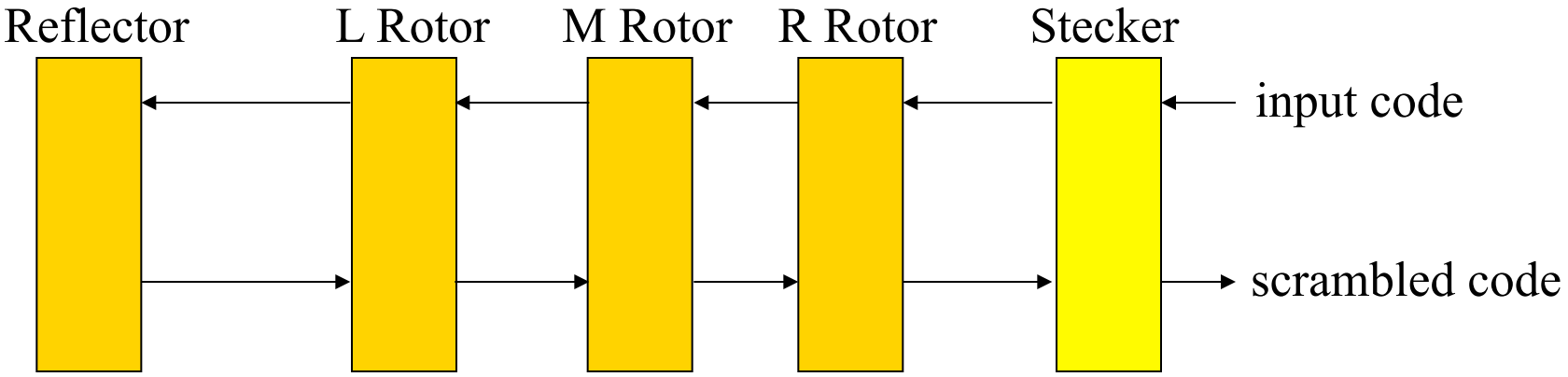
Next, perform a functional simulation of the circuit with Modelsim.

Test all 26 valid input codes, for each of the 2 reflector types. Show your simulations to the TA.

3. Design of the Stecker Circuit .

Some versions of the Enigma machine had an additional scrambling step, through the use of a plugboard, or ***Stecker*** in German. This plugboard acted like a simplified version of a fixed rotor, in that it was able to swap codes for a small number of code pairs (up to 13). For example, if a cable was plugged between plugs **A** and **W**, then codes **A** and **W** would be swapped in both directions.

In this type of system an input code would be scrambled a total of ***9*** times.



Write a VHDL description for a circuit that implements the Stecker operation.

The choice of codes to swap should be ***hardwired***, up to a maximum of 13. The particular choice is up to you. If you have time, you could think of how to make the choice of plugs programmable, but it will be a headache!

The circuit can be implemented using the same approach taken in Lab 4 to make the permutation circuit (except that the inverse permutation is not needed, since the swapping is symmetric).

Use the following entity declaration, and employ a process block, with a single case statement.

```
entity gNN_Stecker is
 port ( input_code         : in std_logic_vector(4 downto 0);
        output_code        : out std_logic_vector(4 downto 0));
end gNN_Stecker;
```

4. Simulation of the Stecker Circuit .

Once you have finished your VHDL description of the Stecker circuit, show it to the TA and explain its design.

Next, perform a *functional* simulation of the circuit.

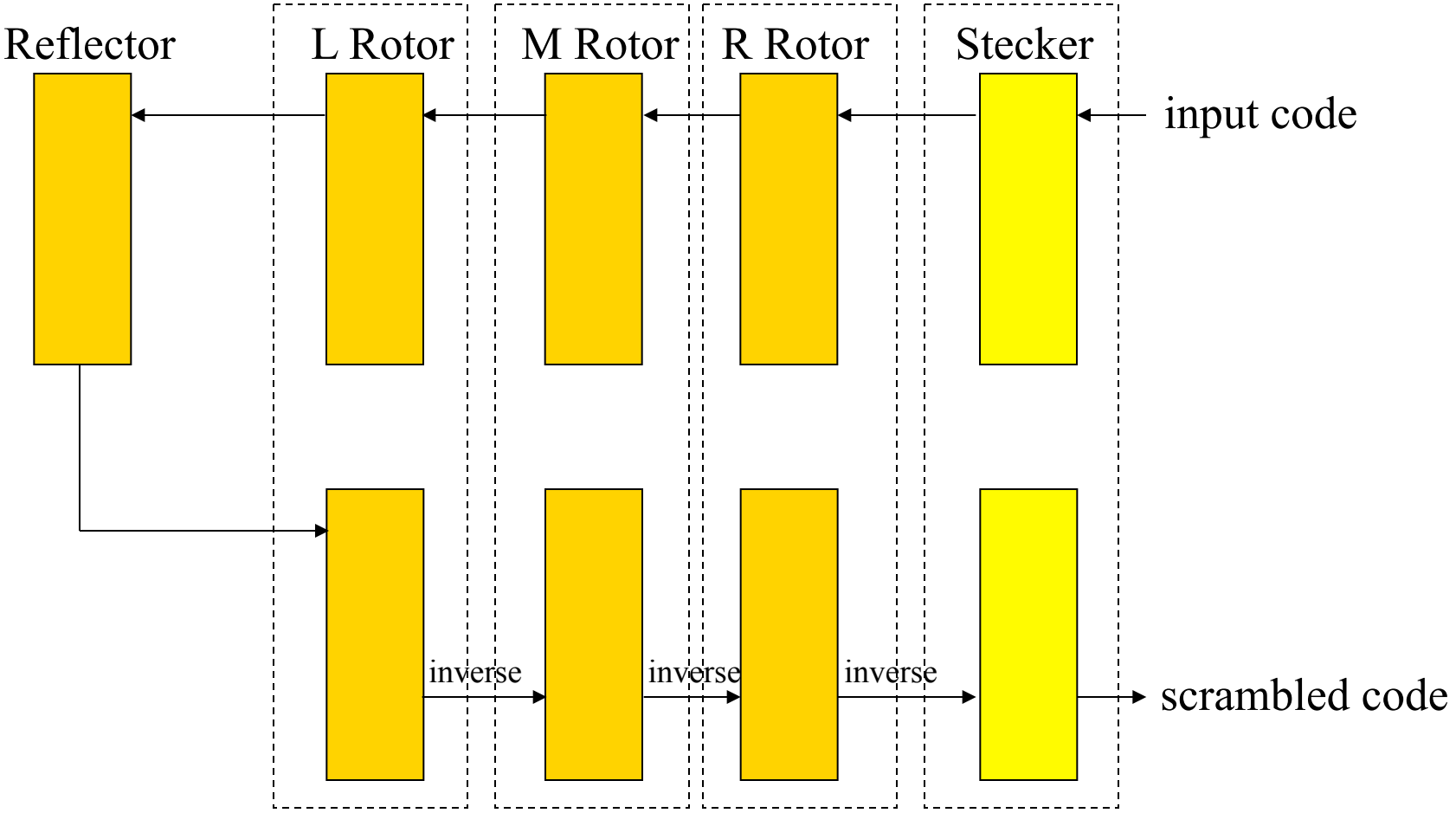
Test all 26 valid input codes and show your simulations to the TA.

TIME CHECK

You should be this far at the end of your *first* 2-hour lab period!

5. Design of the Complete Enigma Machine .

The overall structure of the complete Enigma Machine is as shown:

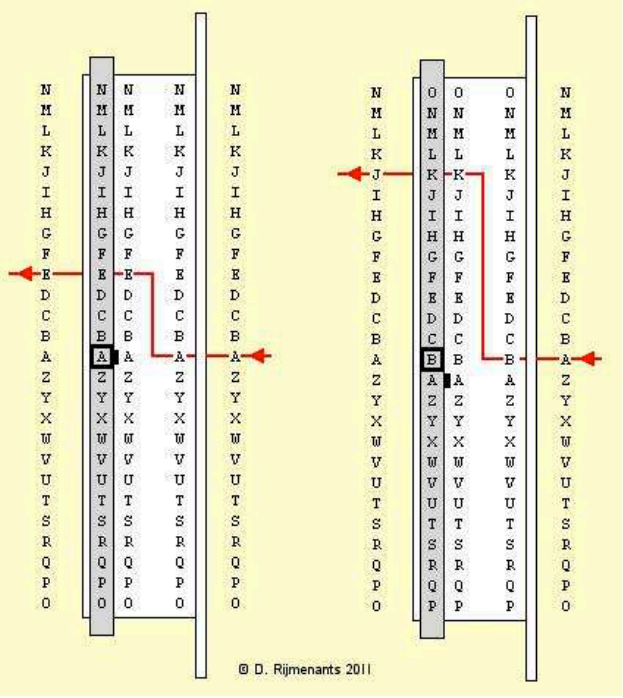


The Rotors

A rotor consists of an inner core that implements the permutation. A signal enters at the right at a particular position of the inner core, and exits at another position on the left of the inner core, determined by the permutation implemented by the rotor type.

There is an additional complication called the Ring Setting. There is a moveable outer ring. Rotating the outer ring changes the Ring Setting and offsets the position of the alphabet ring relative to the inner core.

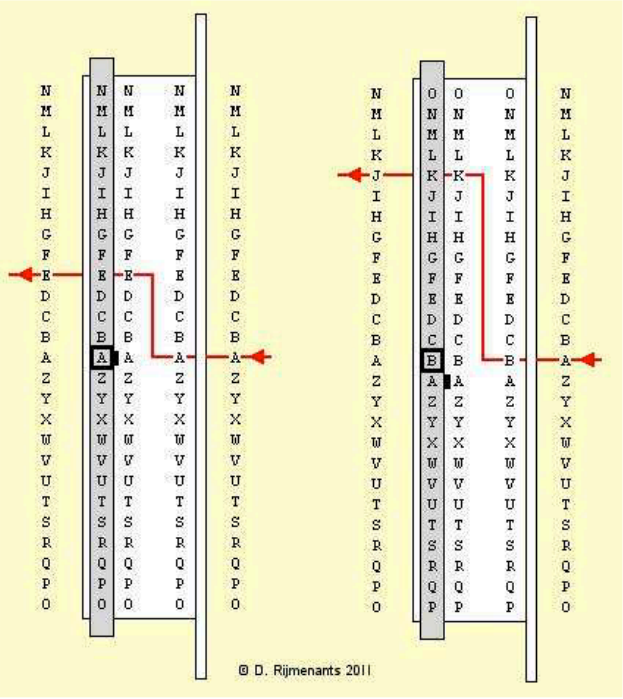
Let's look at an example of a Type I rotor, with Ring Setting A and in Rotor Position A. Look at the figure on the left. The rotor position is indicated by the letter in the window on the alphabet ring, and the ring setting is indicated by the black bar to the right of the position window.



Say a letter "A" is to be encoded. The signal enters at the "A" position on the right, enters the inner core at position A and is encrypted according to the type I permutation, and so exits the inner core at position E. Since the ring is not rotated relative to the core, the signal exits the rotor at position E. A has been encrypted to E.

This figure is from Dirk Rijmenants web site, http://users.telenet.be/d.rijmenants/en/enigmatech.htm

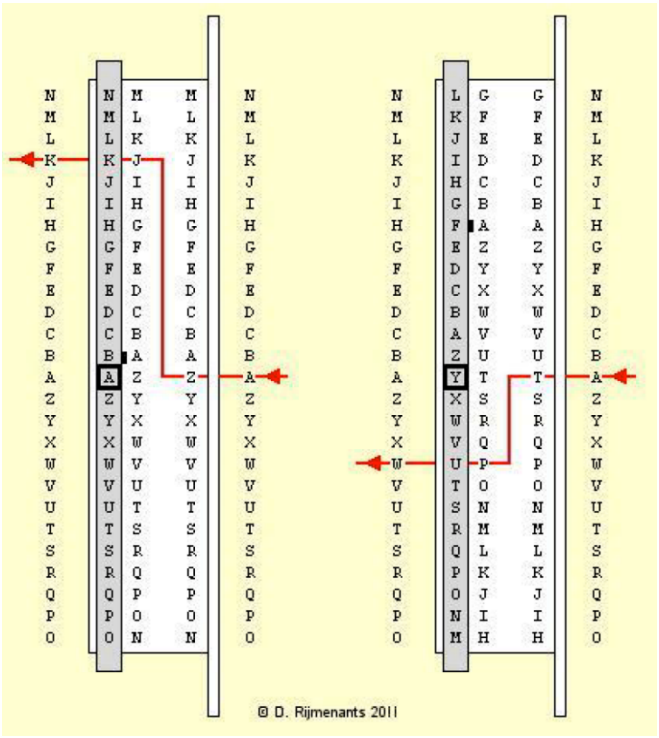
Now let's look at what happens next. Say this is the right rotor, so the rotor rotates by one position to position B (which can now be seen in the window).



Say another letter "A" is to be encoded. The signal is applied to the A position on the right, but enters the inner core at position B since the core is rotated by one position. The permutation gives us K. Notice that K on the inner core is now lined up with position J. A has been encrypted to J.

This figure is from Dirk Rijmenants web site, http://users.telenet.be/d.rijmenants/en/enigmatech.htm

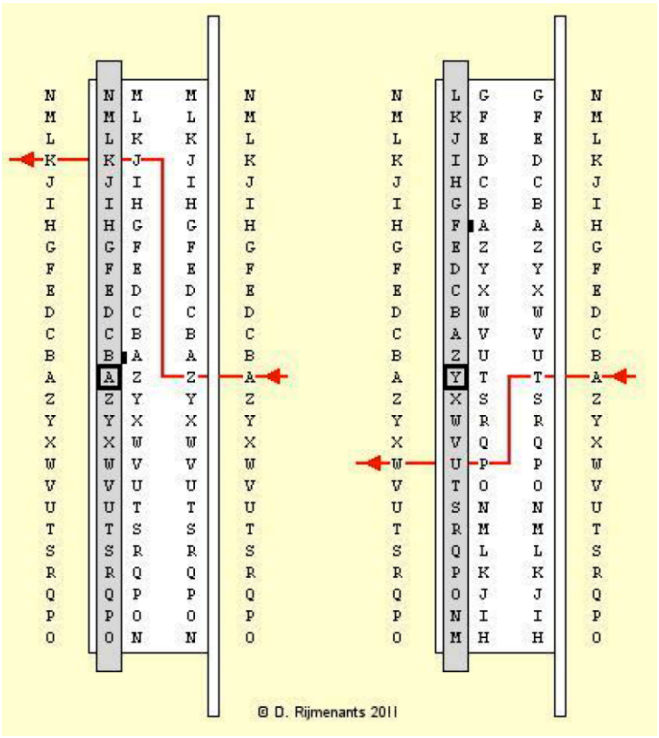
Now lets add a ring setting of B. The ring setting indicates which letter on the ring is aligned with the first contact (A) of the inner core.



In the example on the left, the rotor is in position A and has a ring setting of B. Again, we want to encrypt the letter A. The signal enters on the right at position A, but because of the ring setting, A lines up with Z on the inner core. The permutation Z → J is applied, but J now aligns with position K. A has been encoded to K.

This figure is from Dirk Rijmenants web site, http://users.telenet.be/d.rijmenants/en/enigmatech.htm

Finally, let's look at the rotor in position Y, with ring setting F. Again, we want to encrypt the letter A.



The signal enters on the right at position A, but because of the combined effect of the position and the ring setting, A lines up with T on the inner core. First, consider the effect of rotor position, which lines up A with Y, and then the ring setting, which shifts the position by another 5 places, to end up at T. T is permuted to P. The ring setting shifts this 5 places to U, which is lined up to W (U shifted 24 places gives W). A has been encrypted to W.

A technical detail is that when a key is pressed, the rotors advance *before* the signal runs through them. i.e. if the right rotor is in position Z, then the key is pressed, the right rotor is advanced to position A, and the scrambling is done in position A. The middle and left rotors rotate less often, but when they do, the same principle is applied.

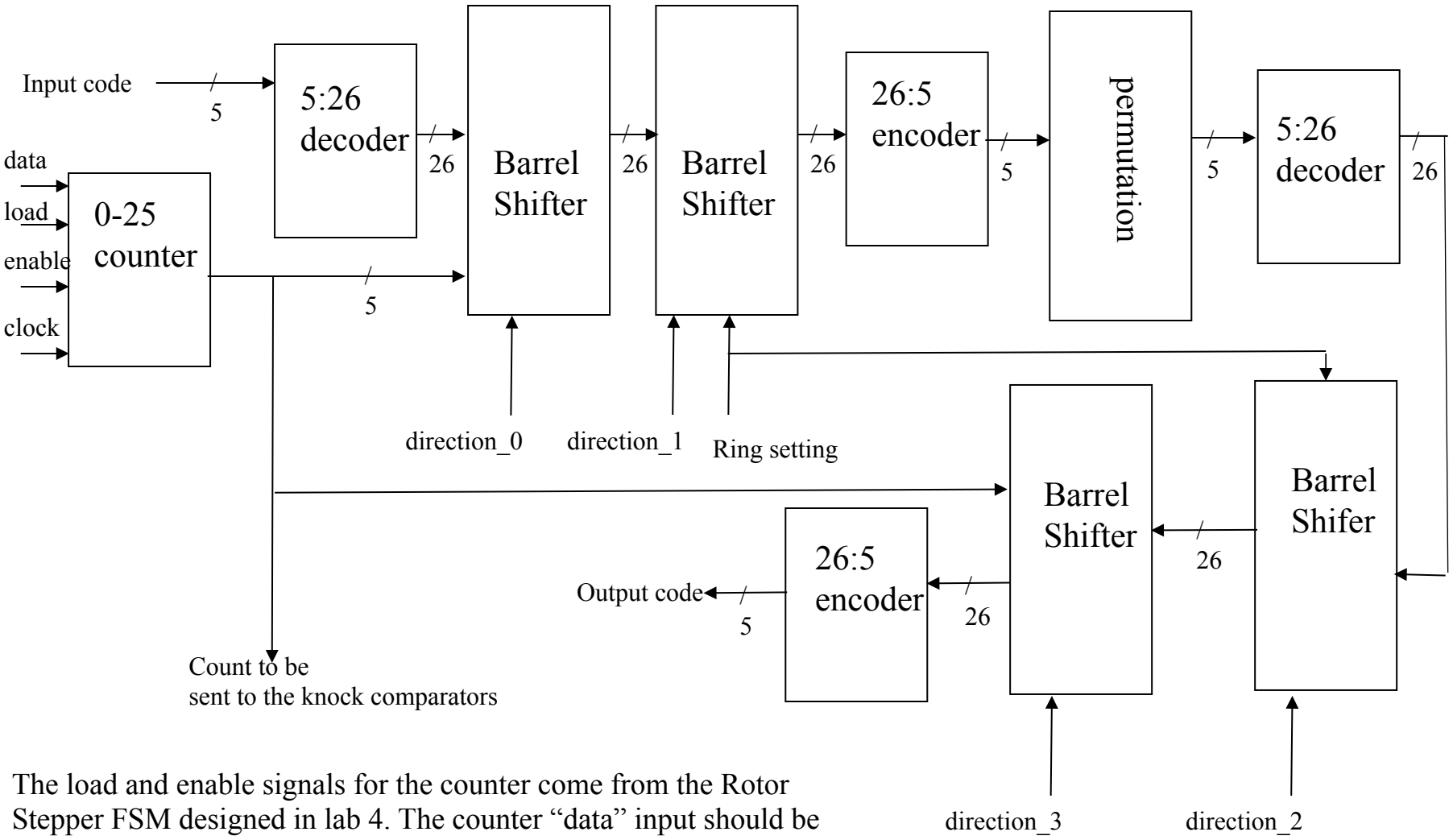
This figure is from Dirk Rijmenants web site, http://users.telenet.be/d.rijmenants/en/enigmatech.htm

The position and ring settings can be implemented by applying shifts (in both right and left directions). You have already implemented a barrel shifter that does a right circular shift. You will need to implement left shifts also to implement the operation of the rotor.

Modify your barrel shifter to implement both left and right shifts, which you can select by a direction signal.

Now, we can design the rotor. The position of the rotor is indicated by the output of the 0-25 counter. The counter output should also be used as input to the knock comparator.

Details of the rotor (one half)



The load and enable signals for the counter come from the Rotor Stepper FSM designed in lab 4. The counter "data" input should be set to the desired initial positon for the rotor. You should determine the values of the direction signal for each barrel shifter.

The design on the previous page is for one half of the rotor (that encrypts the signal from right to left). The signal will also pass through each rotor from left to right on its return path from the reflector. Complete the design of the rotor by implementing the reverse path. Remember to use the inverse permutation on the reverse path.

The complete rotor should take in a signal on the right (from the stecker, or the rotor to the right), scramble it and pass the output to the next rotor to the left (or the reflector). It should also take an input from the left, scramble it with the inverse permutation and pass it to the rotor to the right (or the stecker).

Create a new VHDL module that encapsulates the entire rotor mechanism. Call it *gNN_rotor*. Show your VHDL to the TA and explain the operation of your complete rotor design.

6. Simulation of the Complete Enigma Machine.

Now that you have designed the rotor, write the VHDL for the complete Enigma Machine.

Once you have finished your VHDL of the Enigma Machine circuit, show it to the TA and explain its design.

Then, perform a functional simulation of the circuit.

Test all 26 valid input codes for a few initial settings (rotor types, reflector types, rotor initial positions, and ring settings) and show your simulations to the TA. You can use just one fixed Stecker setup.

TIME CHECK

You should be this far at the end of your *second* 2-hour lab period!

7. Design of the Enigma Machine User Interface.

Once you have debugged your system via simulation, you can adapt it so that it can be implemented on the Altera board. This essentially means adding ***user interface*** circuitry. The user interface will make use of the switches, buttons, and LED displays on the Altera board to permit a user to setup the Enigma machine (e.g. set the initial rotor position, ring settings, rotor types and reflector type) and enter codes and view the (un)scrambled codes.

It is up to you to determine how to implement the user interface and how to make use of the limited user interface resources on the Altera board. You may wish to multiplex the switches so they can be used for more than one purpose (e.g. for specifying initial settings).

TIME CHECK

You should be this far at the end of your *third* 2-hour lab period!

8. Testing of the Enigma Machine on the Altera Board.

Download your complete system with user interface to the Altera board. Try to send a short message (10 characters or so), following the procedure given in the next two pages. Write down the encoded message (and put this in your report, along with the initial settings that you assumed).

Then follow the procedures for decoding the encoded message, and see if you get back the original message. If not, it is time to do some debugging.

Try sending a message to your friends!

Demonstrate the functioning of your system to the TA.

The Cryptographic Key for the Enigma Machine is determined by the initial settings of the following items:

- **Rotor Types (including reflector type).**
- **Initial position of the rotors.**
- **Ring settings.**
- **Stecker Plug settings.**

These settings are typically valid for one day, and would be given in a code-book (very top-secret!) that both the sender and receiver would have. As noted in the procedures outlined on the next page, the initial rotor positions would be modified by the sender for each message sent.

Note: The procedures outlined on the next page are fairly simple and were superseded by more complex procedures during World War II. You can use these more complex procedures if you wish, but you will have to learn about these on your own.

Encoding Operating Procedure:

• Set up the Machine according to the day's initial settings.
• Choose a new arbitrary starting position for the current message to be encoded. For example, an operator might select ***DSD***, this became the *message setting*.
• Enter ***DSD*** into the machine, twice (to compensate for possible transmission errors). The resulting code sequence, e.g. ***XZQLTE,*** is an encrypted indicator which is then transmitted.
• Next, rotate the rotors to the message settings, (e.g. ***DSD*** in this example)
• Enter the text of the desired message, and transmit its encoding.

Decoding Operating Procedure:

• Set up the Machine according to the day's initial settings.
• Enter the first 6 letters of the received message (e.g. ***XZQLTE***). Recall that this is the encoding of the sender's message setting sent twice. So, the decoding by the Enigma Machine should yield the message setting, in this case, ***DSDDSD***.
• Move the rotors to the message setting (***DSD*** in this example).
• Enter the remainder of the received transmission into the Enigma Machine, which will decipher the message.

TIME CHECK

You should be this far at the end of your *fourth* 2-hour lab period!

9. Writeup of the Lab Report .

Write up a report for the complete ***Enigma Machine*** system that you designed. You do not need to give details of modules that you designed in labs 1 through 4. Just make reference to their reports as needed.

The report must include the following items:

- A header listing the group number and the names and student numbers of each group member.
- A title, giving the name of your system.
- A description of the system's features (like an advertising brochure describing what the system does).
- A block diagram of the entire system.
- A *detailed* description of how your system works, referring to the block diagram.
- A description of the user interface (i.e. a users guide to operation).
- A discussion of how the complete system was tested.
- A summary of the FPGA resource utilization and timing.
- A conclusion section discussing problems or significant issues that arose during the design process, as well as a discussion of possible enhancements or extensions that could be made to your system.

Some items to remember when preparing the report for lab #5:

- The document must have page numbers.
- All diagrams, tables, and figures should have captions describing the contents of the figure, and be given a figure number.
- All such figures must be referred to in the text of the report. Do not just throw a figure into the report without referring to it. Figures should be used to augment the textual matter.
- If you include figures or text taken from other sources, be sure to cite these properly. **Avoid plagiarism!** All VHDL descriptions should contain the name of the designers and the date written.
- Hand-drawn figures and diagrams are not acceptable!
- Turn off the grid in the Quartus schematics when making screenshots!
- Break large block diagrams into multiple smaller ones when inserting them into the report.
- Do not include long vhdl descriptions into the report. Instead, include the .vhd files in your report submission zip file.

Don't wait until the last day to start writing the report. Start writing it early on in the lab 5 period. *The report is due on the last day of classes*, and not one week later!

The report should be done in html or pdf (preferred), or in Microsoft Word, and uploaded to the *myCourses* site.

The presentation of the report (grammar, spelling, and clarity of the diagrams) will also be graded.

Make sure that you have uploaded ***all*** of the design files (e.g. .bdf and .vhd files) used in your project. Also include the device programming file (.sof or .pof file) so that the grader can download your design to his/her Altera board.

The report is due at 11:59 PM on *Friday, April 15*.
Late submissions will be accepted, but will be assessed a penalty of *1 mark per day*, (out of a possible maximum of 10).

Grade Sheet for Lab #5

Winter 2016.

Group Number: _____.

Group Member Name: _____. Student Number: _____.

Group Member Name: _____. Student Number: _____.

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

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.