# ECSE-323
# Digital System Design

**Lab #2 – *Combinational Circuit Design with VHDL*** Winter 2016

Prof. W. Gross and Prof. J. Clark

# Introduction

In this lab you will learn how to use the Altera Quartus II FPGA design software to implement combinational logic circuits described in VHDL.

You will design a number of combinational circuits needed for implementing the Enigma machine.

# Learning Outcomes .

***After completing this lab you should know how to:***

• Describe a circuit with VHDL, using component statements as well as conditional and selected signal assignment statements

# Table of Contents .

***This lab consists of the following stages:***

1. Design of a 26:5 Encoder circuit
2. Design of a 26-bit Barrel Shifter Circuit
3. Design of a 7-Segment LED decoder/driver
4. Writeup of the Lab Reports

# 1. Design of a 26:5 Encoder circuit.

In this part of the lab you will create a circuit that does the inverse of the operation performed by the 5:26 decoder circuit designed in Lab 1. The 26:5 encoder circuit takes as input a 26-bit vector (corresponding to a letter of the alphabet) and produces a 5-bit output named INDEX which indicates the *highest* bit position over all input bits that have a high value. If no inputs bits are high, then assert an ERROR signal.

For example, if inputs bits 7, 19, and 23 are all high, with the remaining inputs low, the output should be INDEX=23, ERROR=0.

# VHDL Description of the encoder circuit.

The entity declaration should have the following form (remember to replace the header with your own information)

```
-- This circuit takes as input a 26-bit vector and produces a 5-bit output named
-- INDEX which indicates the highest bit position over all input bits that have
-- a high value. If no inputs bits are high, then assert an ERROR signal.
--
-- entity name: g00_26_5_encoder
--
-- Copyright (C) 2016 Warren Gross
-- Version 1.0
-- Author: Warren Gross; warren.gross@mcgill.ca
-- Date: January 1, 2016

library ieee; -- allows use of the std_logic_vector type
use ieee.std_logic_1164.all;

entity g00_26_5_encoder is
 port ( letter      : in std_logic_vector(25 downto 0);
        INDEX       : out std_logic_vector(4 downto 0);
        ERROR       : out std_logic);
end g00_26_5_encoder;
```

The architecture body will contain the functionality of the circuit. You should use a single ***conditional assignment statement*** to implement the bulk of the circuit.

Conditional assignment statements are tailor-made for this sort of job, as they can easily describe priority logic.

The priority arises in this case, because we want to find the index of the highest bit position that has a high value. Thus, bit 25 should be checked first, and if it is high, then the INDEX value is set to 25. If bit 25 is low, then bit 24 is checked, and so forth. If we go through all of the bits, and none of them are high, then we set ERROR to be high (and the INDEX value is a don't care and can be set to any value you want).

# CREATE THE VHDL DESIGN FILE

Create an empty VHDL file window (using the "New" command from the "File" menu in Quartus).

Save the file under the name *gNN_26_5_encoder.vhd*. Include it into your project, by checking the "***Add File to Current Project***" box.

First enter the entity declaration from the previous slide. Then fill in the file with the VHDL architecture body for your *gNN_26_5_encoder* circuit.

Save the file and show your VHDL description to the TA.


## TIME CHECK

You should be this far (i.e. have completed the lab) at the end of your *first* 2-hour lab period!

# SIMULATE THE DESIGN

Using the techniques learned in lab #1, do a ***functional*** simulation of the *gNN_26_5_encoder* circuit. This simulation should test all 26 single bit high cases, as well as a few other (multi-bit high or no-bit high cases) input patterns.

Write the VHDL testbench to generate the circuit inputs and run the simulation using ModelSim.

Show the TA the results of your simulation.

# 2. Design of a 26-bit Barrel Shifter Circuit .

The Enigma machine encrypts messages by passing each letter through a scrambling process consisting of 3 successive code wheels. Each of these code wheels maps a letter onto another letter in a one-to-one fashion.

For example, the so-called type 1 Rotor has the following mapping:

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**

**EKMFLGDQVZNTOWYHXUSPAIBRCJ**

(e.g. *A* maps to E, *F* maps to *G*, and so forth…).

The wheels can also be *rotated*, so that the mapping is permuted. For example, with a rotation of 2 places the Rotor 1 mapping becomes:

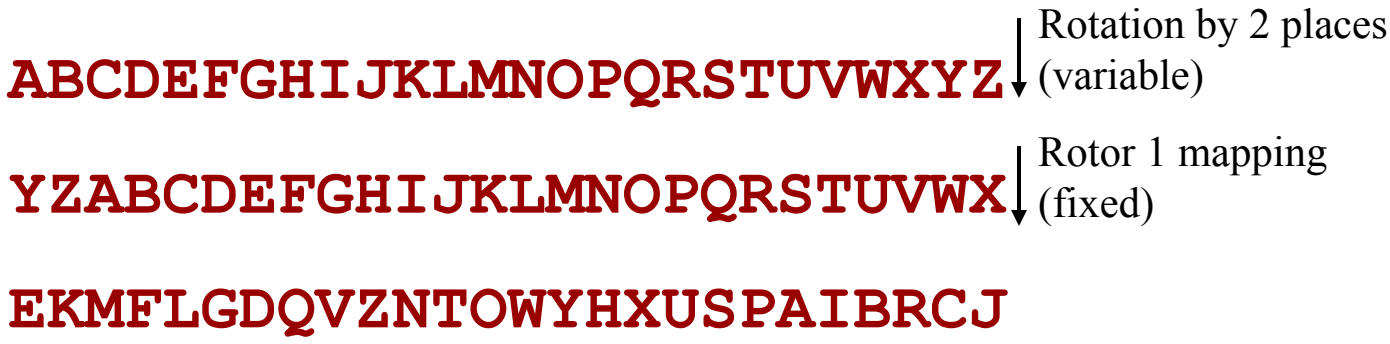
**ABCDEFGHIJKLMNOPQRSTUVWXYZ**

**MFLGDQVZNTOWYHXUSPAIBRCJEK**

In the next lab you will design the circuit to perform the mapping. In *this* lab, you will design the circuit that rotates, or permutes, the mapping.

We will implement the rotation of the mapping by keeping the mapping fixed and rotating the *input* to the mapping instead.

That is, we keep the mapping the same, but precede it with a simple rotation mapping. Take the example of rotating two places from the previous slide:




So, we see that *A* maps to *M, B* maps to *F*, etc., just as on the previous slide.

So what we need is a circuit that can take a 26-bit index signal and rotate it by a given number of places, *N*. This could be done with a 26-bit shift-register, by loading the register with the input bits and then shifting *N* times, feeding the MSB back into the LSB of the register.

The rotation operation can also be done with a purely combinational circuit. Such a circuit is often called a ***Barrel Shifter***.

Barrel shifters are often used in Arithmetic Logic Units in microprocessors, and find use in various applications. They are especially useful in compression and encryption programs, much for the same reason that they are useful for simulating the Enigma machine.

Hardware implementations of Barrel Shifters typically are done using collections of multiplexers. You could design the barrel shifter this way, but it would be very tedious to lay out the schematic. In this lab you will write a relatively simple VHDL description of a barrel shifter, and let the Quartus compiler do most of the work in figuring out the hardware implementation.

Perhaps the most straightforward way to describe a barrel shifter function in VHDL is to use a Selected Signal Assignment statement, employing the VHDL concatenation operator ***&***. The & operator takes two vector signals and concatenates them to make a third, which has a length equal to the sum of the lengths of the two operand signals.

e.g.

```
Y <= X(5 downto 3) & Z(12 downto 4);
```

The left hand side vector must have a length of 12 bits (3 + 9).

To implement a 26-bit barrel shifter using a Selected Signal Assignment statement you would have 26 cases, one for each possible shift amount.

The general outline of the description would be as follows:

```
with N select
Y <=
…
    X(21 downto 0) & X(25 downto 22) WHEN "00100",
…
    X WHEN others;
```

**You fill in the rest of the cases!**

# CREATE THE DESIGN FILE

Once you have written the VHDL description, show your completed VHDL description to your TA.


## TIME CHECK

You should be this far (i.e. have completed the lab) at the end of your *second* 2-hour lab period!

# SIMULATE THE DESIGN

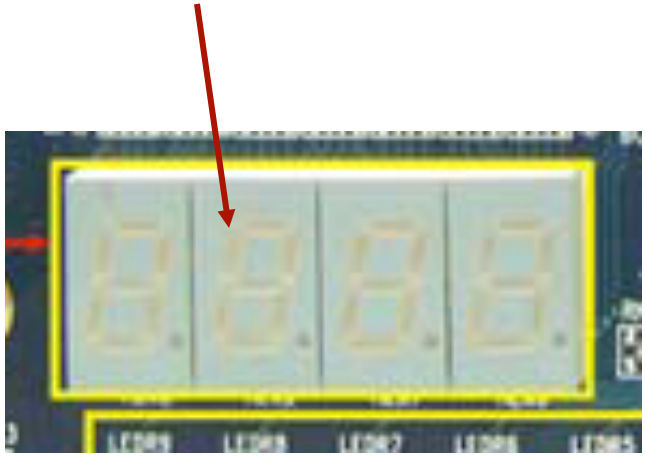
Compile the *gNN_barrel_shifter* circuit and do a functional simulation. This simulation should test ***a representative set*** of input patterns of the input code (at least 16).

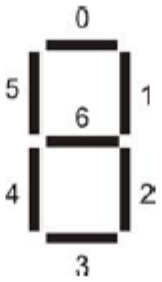
Write the VHDL testbench to generate the circuit inputs and run the simulation using ModelSim.

Show the TA the results of your simulation.

# 3. Design of a 7-Segment LED decoder/driver .

A 7-segment LED display has 7 individual light-emitting segments, as shown in the picture below. By turning on different segments at any one time we can obtain different characters or numbers. There are four of these on the Altera board, which you will use later in your full implementation of the Enigma machine.



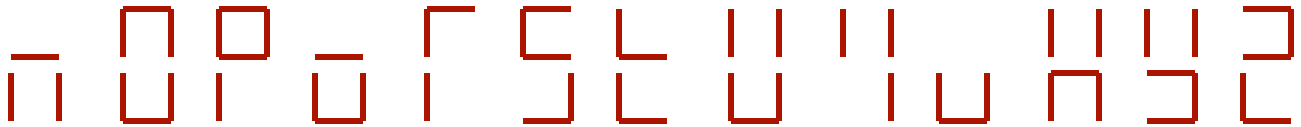



numbering of the LED segments (from the Altera DE1 board manual)

In this part of the lab you will design a circuit that will be used to drive the 7-segment LEDs on the Altera board. It takes in a 5-bit code representing the 26 letters of the alphabet and generates the 7-segment display associated with the input code, as shown below.

***The outputs should be made active-low. This is convenient, as many LED displays, including the ones on the Altera board, turn on when their segment inputs are driven low.***

CODE: 0 1 2 3 4 5 6 7 8 9 10 11 12



CODE: 13 14 15 16 17 18 19 20 21 22 23 24 25

Codes 26 through 31 should result in a blank display (all segments off)

To implement the 7-segment LED decoder, write a VHDL description using ***a single selected signal assignment statement***.

Use the following *entity declaration*, replacing the gNN in gNN_7_segment_decoder with your group's number (e.g. g08). You will have to supply the architecture body…

```
entity gNN_7_segment_decoder is
 port ( code : in std_logic_vector(4 downto 0);
        segments : out std_logic_vector(6 downto 0));
end gNN_7_segment_decoder;
```

# CREATE THE DESIGN FILE

Once you have written the VHDL description, show the completed VHDL description to your TA.


## TIME CHECK

You should be this far (i.e. have completed the lab) at the end of your *third* 2-hour lab period!

# SIMULATE THE DESIGN

Compile the *gNN_7_segment_decoder* circuit and do a functional simulation. This simulation should test ***all 32 possible*** input patterns of the input code.

Write the VHDL testbench to generate the circuit inputs and run the simulation using ModelSim.

Show the TA the results of your simulation.


## TIME CHECK

You should be this far (i.e. have completed the lab) at the end of your *fourth* 2-hour lab period!

# 4. Writeup of the Lab Reports .

Write up three (3) short reports, describing each of the ***gNN_26-5_encoder, gNN_barrel_shifter*** and ***gNN_7_segment_decoder*** circuits.

The reports must include the following items:

- A header listing the group number, the names and student numbers of each group member.
- A title, giving the name (e.g. ***gNN_barrel_shifter***) and function of the circuit.
- A description of the circuit's function, listing the inputs and outputs. Provide a pinout or symbol diagram.
- The VHDL description of the circuit including the testbench (don't embed these in the text of the report, instead include them as a separate file in the assignment submission zip file).
- A complete discussion of how the circuit was tested, showing representative simulation plots, and detailing what test cases were used.

The lab report, and all associated design files must be submitted, as an assignment to the myCourses site. Only one submission need be made per group (both students will receive the same grade!).

**Combine all of the files that you are submitting into one *zip* file, and name the zip file *gNN_LAB_2.zip* (where NN is your group number).**

**The reports are due at 11:59 PM, Friday, February 26.**

# Grade Sheet for Lab #2

**Winter 2016.**

Group Number: _____.

Group Member Name: _____. Student Number: _____.

Group Member Name: _____. Student Number: _____.

| Marks | | | TA Signatures |
|---|---|---|---|
| | 1. | VHDL code for the *5:26 encoder* circuit | _____. |
| | 2. | Simulation of the *5:26 encoder* circuit | _____. |
| | 3. | VHDL code for the *barrel shifter* circuit | _____. |
| | 4. | Simulation of the *barrel shifter* circuit | _____. |
| | 5. | VHDL code for the *7_segment_decoder* circuit | _____. |
| | 6. | Simulation of the *7_segment_decoder* circuit | _____. |

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.