COE3DQ5 Lab #1
Introduction to Computer-Aided Design using Verilog

Objective

To gain experience with the Quartus design environment, understand distinct SystemVerilog-2005 coding techniques used for combinational and sequential logic, use design synthesis and simulation, and implement peripheral circuitry in the Altera DE2-115 board.

Preparation

- Revise the logic design material and the Quartus design environment
- Read this document and get familiarized with SystemVerilog-2005 language constructs

Experiment 1

The purpose of this exercise is to get you familiarized with the Quartus environment and the DE2-115 board. You will find in the ***experiment1*** directory a design file with a module whose ports are shown in Figure 1. There 18 input switches, 9 green light emitting diodes (LEDs) and 18 red LEDs.

Figure 1 – Design ports for experiment1

First, you have to perform the following:

- create a project with device EP4CE115F29C7 from the Cyclone IV family
- select SystemVerilog-2005 as the specification language
- compile the design (these first three tasks need to be done for all the experiments)
- create the pin assignments using the table shown below
- re-compile the design and program the field-programmable gate array (FPGA) device

Input Switches		Output Green LEDs		Output Red LEDs	
Physical Pin	**Design Port**	**Physical Pin**	**Design Port**	**Physical Pin**	**Design Port**
PIN_AB28	SWITCH_I[0]	PIN_E21	LED_GREEN_O[0]	PIN_G19	LED_RED_O[0]
PIN_AC28	SWITCH_I[1]	PIN_E22	LED_GREEN_O[1]	PIN_F19	LED_RED_O[1]
PIN_AC27	SWITCH_I[2]	PIN_E25	LED_GREEN_O[2]	PIN_E19	LED_RED_O[2]
PIN_AD27	SWITCH_I[3]	PIN_E24	LED_GREEN_O[3]	PIN_F21	LED_RED_O[3]
PIN_AB27	SWITCH_I[4]	PIN_H21	LED_GREEN_O[4]	PIN_F18	LED_RED_O[4]
PIN_AC26	SWITCH_I[5]	PIN_G20	LED_GREEN_O[5]	PIN_E18	LED_RED_O[5]
PIN_AD26	SWITCH_I[6]	PIN_G22	LED_GREEN_O[6]	PIN_J19	LED_RED_O[6]
PIN_AB26	SWITCH_I[7]	PIN_G21	LED_GREEN_O[7]	PIN_H19	LED_RED_O[7]
PIN_AC25	SWITCH_I[8]	PIN_F17	LED_GREEN_O[8]	PIN_J17	LED_RED_O[8]
PIN_AB25	SWITCH_I[9]			PIN_G17	LED_RED_O[9]
PIN_AC24	SWITCH_I[10]			PIN_J15	LED_RED_O[10]
PIN_AB24	SWITCH_I[11]			PIN_H16	LED_RED_O[11]
PIN_AB23	SWITCH_I[12]			PIN_J16	LED_RED_O[12]
PIN_AA24	SWITCH_I[13]			PIN_H17	LED_RED_O[13]
PIN_AA23	SWITCH_I[14]			PIN_F15	LED_RED_O[14]
PIN_AA22	SWITCH_I[15]			PIN_G15	LED_RED_O[15]
PIN_Y24	SWITCH_I[16]			PIN_G16	LED_RED_O[16]
PIN_Y23	SWITCH_I[17]			PIN_H15	LED_RED_O[17]

Table 1 – Pin assignments for experiment1

You have to perform the following tasks in the lab for this experiment:

- check if the logic functions for signals NOT, AND2, OR2, AND3 and OR3 work correctly
- create the logic functions for NAND4 and NOR4 between input switches 8 to 11
- create a logic function AND_OR that has both AND/OR operations using switches 4 to 7 (for the AND_OR function, the first two and the last two inputs are ANDed and then the results are ORed)
- create a logic function AND_XOR that has both AND/XOR operations using switches 0 to 3 (for the AND_XOR function, the first two and the last two inputs are ANDed and then the results are XORed)
- compare the estimated number of logic elements (LEs) against the ones reported by the compiler

Experiment 2

In this experiment you will get familiarized with the 7-segment-display and priority encoders.

In directory ***experiment2*** you will find two design files and one configuration file. The configuration file (QSF) contains all the pin assignments. The **convert_hex_to_seven_segment** Verilog file contains the logic for converting a four bit signal to the 7 signals required to control the LEDs of the 7-segment-display. Note, the LEDs of the 7-segment-display are active low. The truth tables of the 7 logic equations are not given in this document, however they can be found in the “case statement” in the design file. The logic implementation of this “case statement” takes only 7 look-up tables (LUTs) and it is shown below.

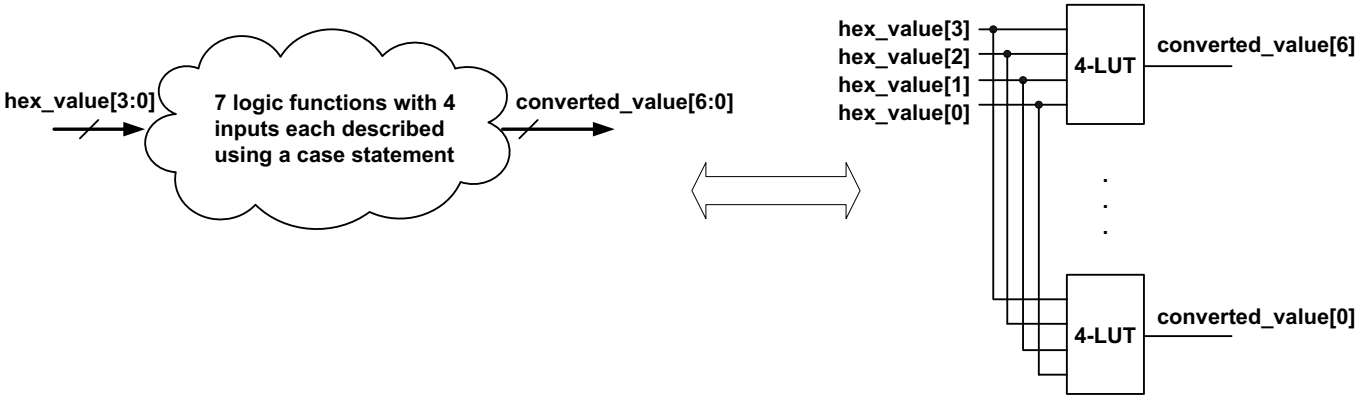

Figure 2 – Logic implementation of convert_hex_to_seven_segment

The priority encoder described in design file **experiment2** reads switches 0 to 9 and it encodes the position of the most significant switch (that is turned on) to a 4-bit binary-coded-decimal (BCD) signal called “value”. This value will be passed to the design unit that converts a 4-bit value and passes it to the 7-segment-display. The logic implementation of the priority encoder is shown below. Note, if no switch is turned on then the displayed value will be 4’hF.

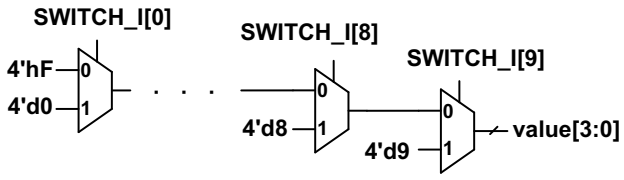

Figure 3 – Logic implementation of the priority encoder

You have to perform the following tasks in the lab for this experiment:

- understand the source code and verify if the priority encoder works correctly
- extend the priority encoder to read all the 18 switches from the DE2-115 board and display the position of the most significant switch onto the two rightmost 7-segment-displays in 2-digit BCD format (if all the switches are turned off then the displayed value should be 8’hFF).

Experiment 3 (optional)

The purpose of this experiment is to better your understanding of counters and to get you familiarized with simulation. There are no implementations on the DE2-115 board for the two “sub-experiments”, given in directories ***experiment3a*** and ***experiment3b***.

In **part a** you are given the code for the 1-digit BCD up-counter. Whenever the count value reaches 9 the 4-bit **BCD_count** register will be reset. Otherwise it counts up. The implementation is shown below.

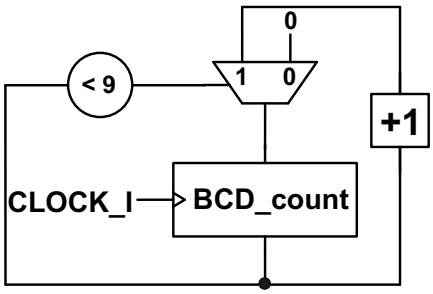

Figure 4 – Logic implementation of the 1-digit BCD up-counter

In **part b** you are given the code for the 2-digit BCD up-counter with parallel load capability. The implementation is shown below. BCD_count[0] is used as an enable for BCD_count[1].

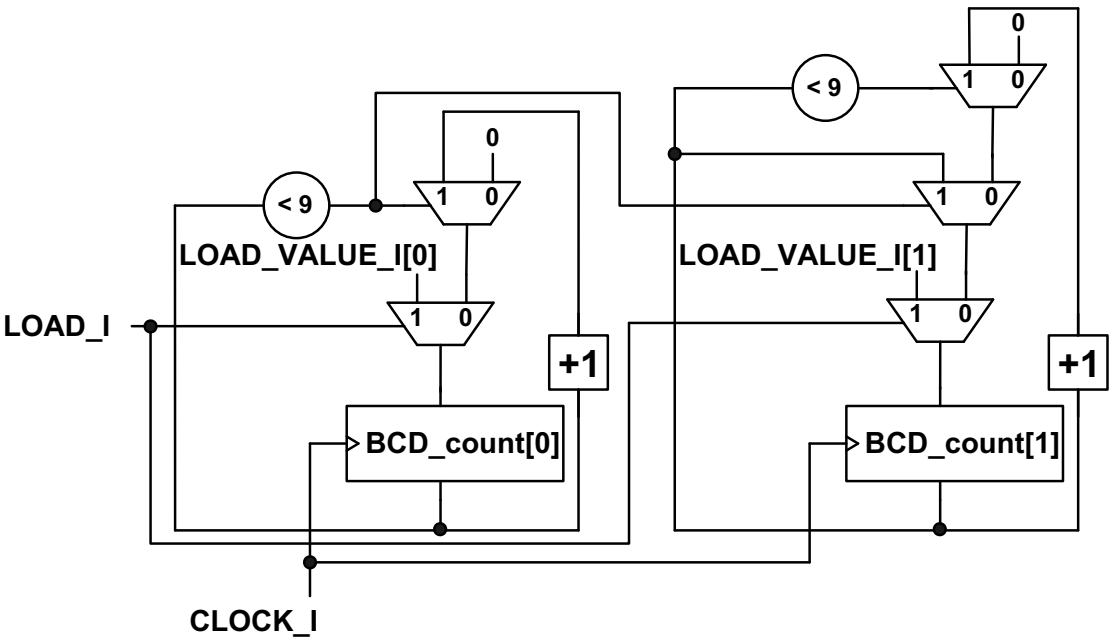

Figure 5 – Logic implementation of the 2-digit BCD up-counter with parallel load

You have to perform the following tasks in the lab for this experiment:

- simulate the two designs from **parts a** and **b** to understand how they work

Experiment 4

The purpose of this experiment is to introduce the concepts of clock division and edge detection, and to display the counters clocked at 1 Hz on the 7-segment-displays.

If the reference clock is 50 MHz, in order to display the counter values at 1Hz on the 7-segment-display LEDs, clock division is required. This can be done using a reference counter clocked at 50MHz. This counter is reset to zero every 25,000,000 clock cycles when it will also toggle the content of an 1-bit flip-flop (called one_sec_clock). The output of this flip-flop can be used as the 1Hz clock, as shown in the figure below. Nonetheless there is a limitation of this approach because (due to clock skew problems) it is not recommended that outputs from some flip-flops to be used as clock inputs to any other flip-flops.

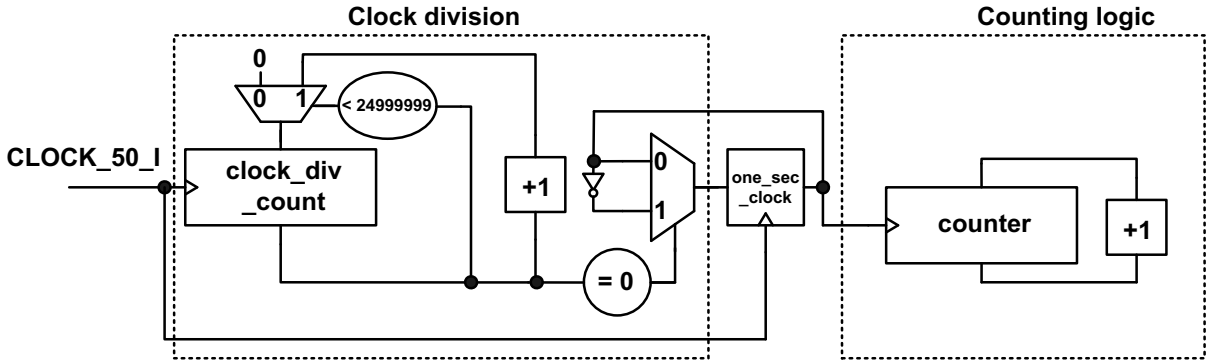

Figure 6 – Clock division with low-speed clock signal driven from the output of a flip-flop

This problem can be fixed if edge detection circuitry is used. At the output of the one_sec_clock flip-flop we add an additional buffer that can be used for detecting when there is one positive edge on the output of the one_sec_clock signal. Both flip-flops are synchronized at 50MHz and they are used to generate the count_enable signal that is used as a synchronous enable for the low-speed counter (as shown in the figure below). Note, in the reference design from the directory ***experiment4*** the counter value is displayed on the 7-segment-display in the hex format and the most significant switch is used as an active-low reset (for all the experiments with sequential circuitry).

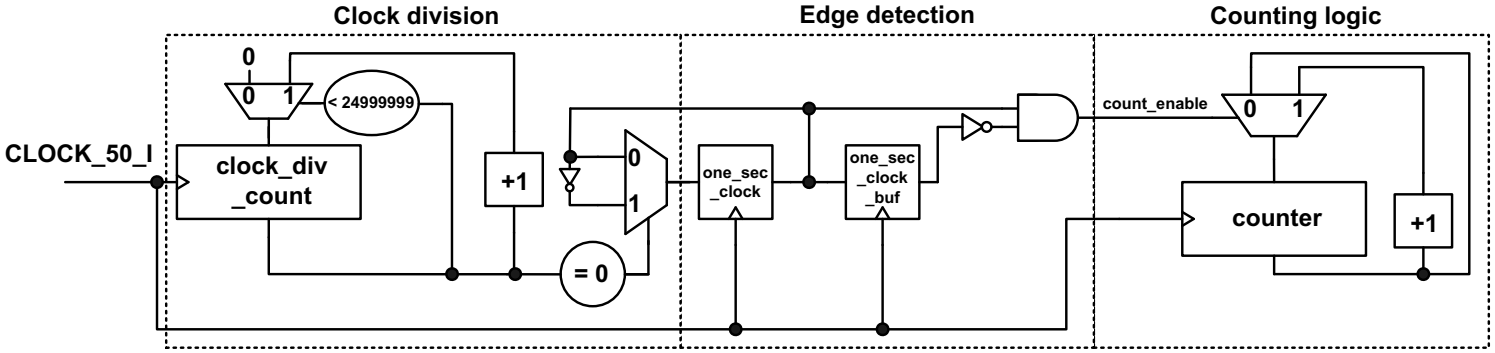

Figure 7 – Updating the counter every second using only one reference clock

You have to perform the following tasks in the lab for this experiment:

- display on the 7-segment displays a 2-digit BCD up-counter (from 00 to 59) that updates every second
- repeat the previous task, however update the counter every half a second

Experiment 5

The purpose of this experiment is to introduce the concept of debouncing and controlled counters.

After pressing a mechanical switch (such as a push-button on the DE2-115 board) there is a transient period during which the switch "bounces". The signal from the push-button (active low when pressed on this DE2-115 board) is monitored at 1KHz and its value is dumped in a left-shift register. The enable signal for the shift register is generated in a similar fashion as the count_enable signal in the previous example and it is not detailed in the figure below.

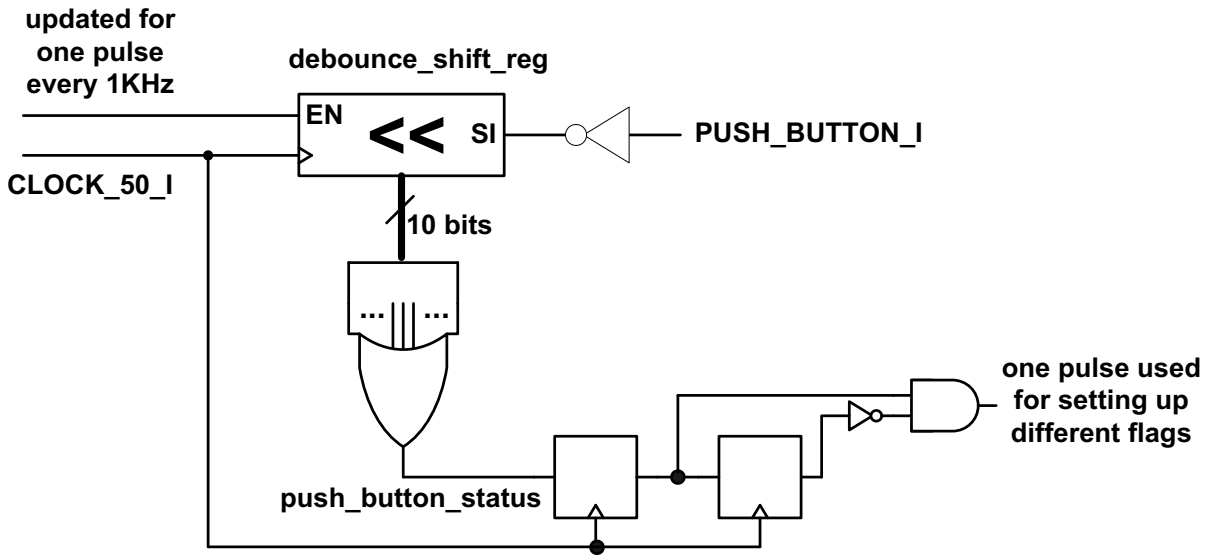

Figure 8 – Debouncing circuitry

When the push-button is not pressed the content of the shift register is zero. As soon as the button is pressed the content will have at least one "1" and the push-button status will be set. So long as the value of the shift register has at least one "1", any transients will be masked and the status of the push-button will not be updated. After the push-button is released, only after the transients have passed the value of the push-button status will be reset. Since only one action should take place each time a push-button is pressed, an edge detection circuitry on the push-button status signal is provided (it is based on the same principles as in the previous experiment). The one pulse generated by this circuitry can be used for providing the enable signals for starting/stopping counters, as shown in the example from the directory ***experiment5***.

You have to perform the following tasks in the lab for this experiment:

- in the reference design push-button 0 is used to start/stop the counter shown on the 7-segment-displays; modify the reference code to use push-button 1 for setting the direction of counting to *up* and push-button 2 for setting the direction of counting to *down*; the counting direction can be changed irrespective whether the counter is active or not; note also, while the counter is not active if push-buttons 1 or 2 are pressed multiple times then only the *last* button that is pressed takes effect;

Write-up Template

COE3DQ5 – Lab #1 Report
Group number
Student names
McMaster email addresses
Date

Your evaluation is based on one take-home exercise. In addition to your report, you should submit ONLY the source files, i.e., Verilog and QSF files.

Exercise (3 marks) – Based on the knowledge from the experiments from this lab, perform the following:

- As for the in-lab experiment 5, switch 17 is used as an asynchronous reset for all the flip-flops. While asserted, the circuit is considered to be "asleep" the green LEDs, as well as the 7-segment-displays are not lightened, i.e. as if the board is not powered. While switch 17 is deasserted, the circuit is considered to be "awake" and its behavior is given below.

- The two *rightmost* 7-segment-displays show the state of a 2-digit BCD counter, which counts both up and down and it is controlled by the four push-buttons on the DE2-115 board. While active the counter updates once a second. On power-up, or immediately after switch 17 has been deasserted, the 2-digit BCD counter starts counting up from state 00. Push-buttons 0, 1 and 2 have the same behavior as for the in-lab experiment 5; push-button 3 is used to restart the 2-digit BCD counter after it freezes, which is defined as follows: while counting up, the counter freezes when it reaches state 59; while counting down, the counter freezes when it reaches state 00. If the counter is frozen, it maintains its state and any activity on push-buttons 0, 1 and 2 is ignored. While "awake" and "frozen", the counter will restart only after push-button 3 has been pressed and it will resume counting in the opposite direction from the direction it was counting before freezing. Note, if the counter is not frozen, any activity on push-button 3 is ignored.

- The *leftmost* 7-segment-display shows in hex format the position of the *least* significant switch (from switches 15 down to 0) which is *low*. If none of the switches 15 down to 0 are *low* then the *leftmost* 7-segment-display should not be lightened. The 7-segment-display *next* to the *leftmost* 7-segment-display shows the index of the last push-button which has been ***released*** (note, for this particular functionality the push-buttons are monitored while the circuit is "awake" regardless of the state of the BCD counter). After power-up, or immediately after switch 17 is deasserted, the 7-segment-display *next* to the *leftmost* 7-segment-display should not be lightened until at least one of the push-buttons has been pressed and released. You should assume that two or more push-buttons can be kept pressed at the same time, however they are not released simultaneously. Note also, the 7-segment-displays that have not been discussed above should not be lightened.

- The green LEDs are partitioned into three groups and they are driven as follows:

    - Green LED 8 is lightened only if at least one of the switches 14 down to 10 is high
    - Green LED 7 is lightened only if all of the switches 14 down to 10 are high
    - Green LED 6 is lightened only if the number of switches from 14 down to 10 that are high is an odd number

    - Green LED 5 is lightened only if at least one of the switches 9 down to 5 is high
    - Green LED 4 is lightened only if all of the switches 9 down to 5 are high
    - Green LED 3 is lightened only if the number of switches from 9 down to 5 that are high is an odd number

    - Green LED 2 is lightened only if at least one of the switches 4 down to 0 is high
    - Green LED 1 is lightened only if all of the switches 4 down to 0 are high
    - Green LED 0 is lightened only if the number of switches from 4 down to 0 that are high is an odd number

- It should not escape your attention the similarity in behavior for the three groups of green LEDs. Therefore, it is ***mandatory*** to describe the functionality for a group within a module that gets ***instantiated*** three separate times: once with switches 14 down to 10 as inputs and green LEDs 8 down to 6 as outputs, once with switches 9 down to 5 as inputs and green LEDs 5 down to 3 as outputs and once with switches 4 down to 0 as inputs and green LEDs 2 down to 0 as outputs.

Submit your sources and in report describe your reasoning. Your report is expected to be half-a-page and under no circumstances it should exceed a full page.

VERY IMPORTANT NOTE:

This lab has a weight of 3% of your final grade. The report has no value without the source files. Your report must be in "pdf" format and together with the requested source files it should be included in a directory called "coe3dq5_group_xx_takehome1" (where xx is your group number). Archive this directory (in "zip" format) and upload it through Avenue to Learn before noon on the day you are scheduled for lab 2. Late submissions will be penalized.