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 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 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 EP2C35F672C6 from the Cyclone II 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_N25	SWITCH_I[0]	PIN_AE22	LED_GREEN_O[0]	PIN_AE23	LED_RED_O[0]
PIN_N26	SWITCH_I[1]	PIN_AF22	LED_GREEN_O[1]	PIN_AF23	LED_RED_O[1]
PIN_P25	SWITCH_I[2]	PIN_W19	LED_GREEN_O[2]	PIN_AB21	LED_RED_O[2]
PIN_AE14	SWITCH_I[3]	PIN_V18	LED_GREEN_O[3]	PIN_AC22	LED_RED_O[3]
PIN_AF14	SWITCH_I[4]	PIN_U18	LED_GREEN_O[4]	PIN_AD22	LED_RED_O[4]
PIN_AD13	SWITCH_I[5]	PIN_U17	LED_GREEN_O[5]	PIN_AD23	LED_RED_O[5]
PIN_AC13	SWITCH_I[6]	PIN_AA20	LED_GREEN_O[6]	PIN_AD21	LED_RED_O[6]
PIN_C13	SWITCH_I[7]	PIN_Y18	LED_GREEN_O[7]	PIN_AC21	LED_RED_O[7]
PIN_B13	SWITCH_I[8]	PIN_Y12	LED_GREEN_O[8]	PIN_AA14	LED_RED_O[8]
PIN_A13	SWITCH_I[9]			PIN_Y13	LED_RED_O[9]
PIN_N1	SWITCH_I[10]			PIN_AA13	LED_RED_O[10]
PIN_P1	SWITCH_I[11]			PIN_AC14	LED_RED_O[11]
PIN_P2	SWITCH_I[12]			PIN_AD15	LED_RED_O[12]
PIN_T7	SWITCH_I[13]			PIN_AE15	LED_RED_O[13]
PIN_U3	SWITCH_I[14]			PIN_AF13	LED_RED_O[14]
PIN_U4	SWITCH_I[15]			PIN_AE13	LED_RED_O[15]
PIN_V1	SWITCH_I[16]			PIN_AE12	LED_RED_O[16]
PIN_V2	SWITCH_I[17]			PIN_AD12	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.

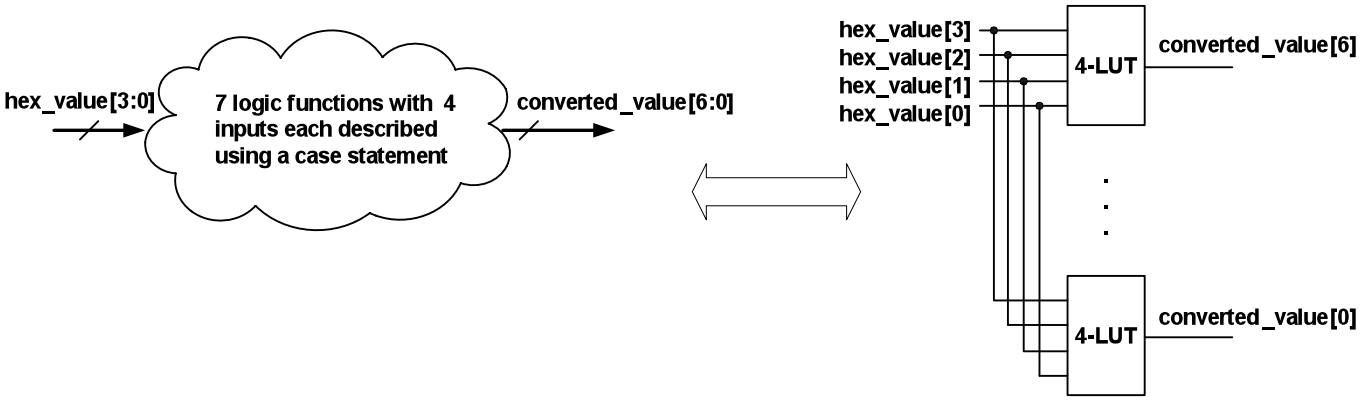

Figure 2 – Logic implementation of convert_hex_to_seven_segment

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.

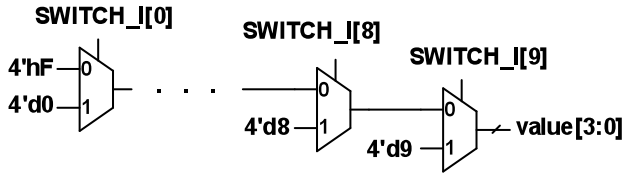

Figure 3 – Logic implementation of the priority encoder

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 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

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 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.

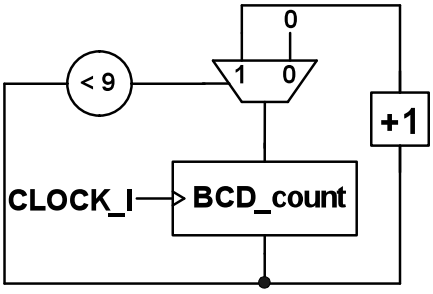

Figure 4 – Logic implementation of the 1-digit BCD up-counter

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].

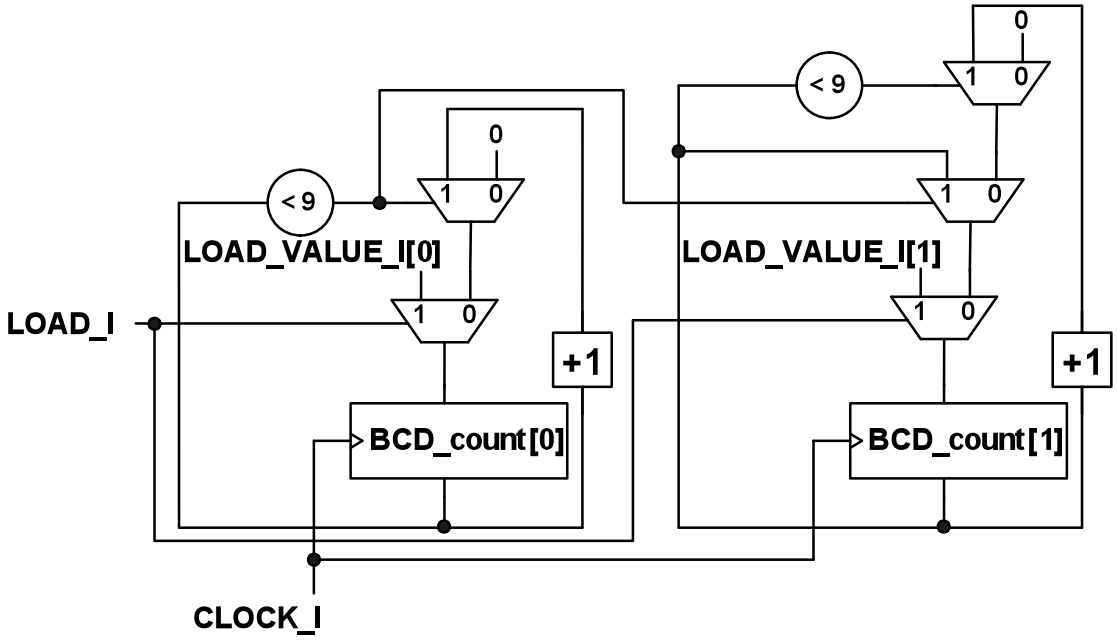

Figure 5 – Logic implementation of the 2-digit BCD up-counter with parallel load

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

- simulate the two designs from **parts a** and **b** to verify their correctness
- change the 2-digit BCD up-counter from **part b** to a 2-digit BCD down-counter with parallel load

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.

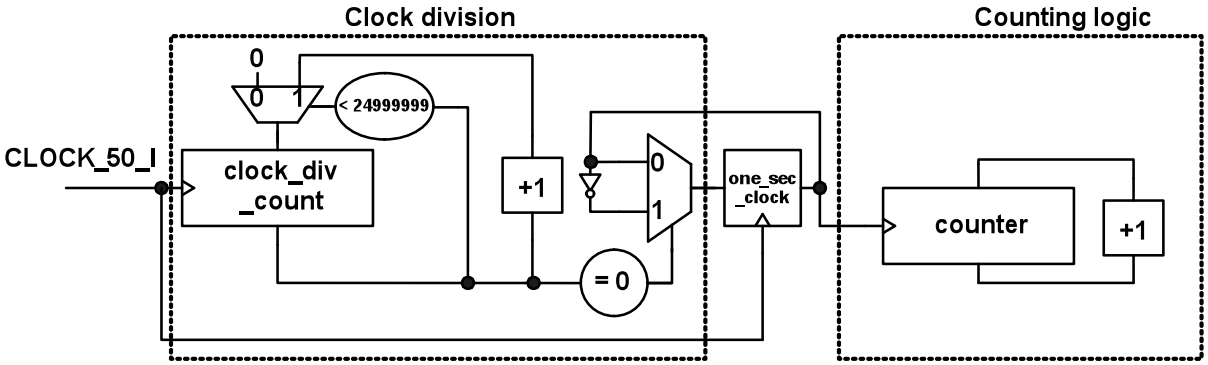

Figure 6 – Clock division with low-speed clock signal driven from the output of a flip-flop

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).

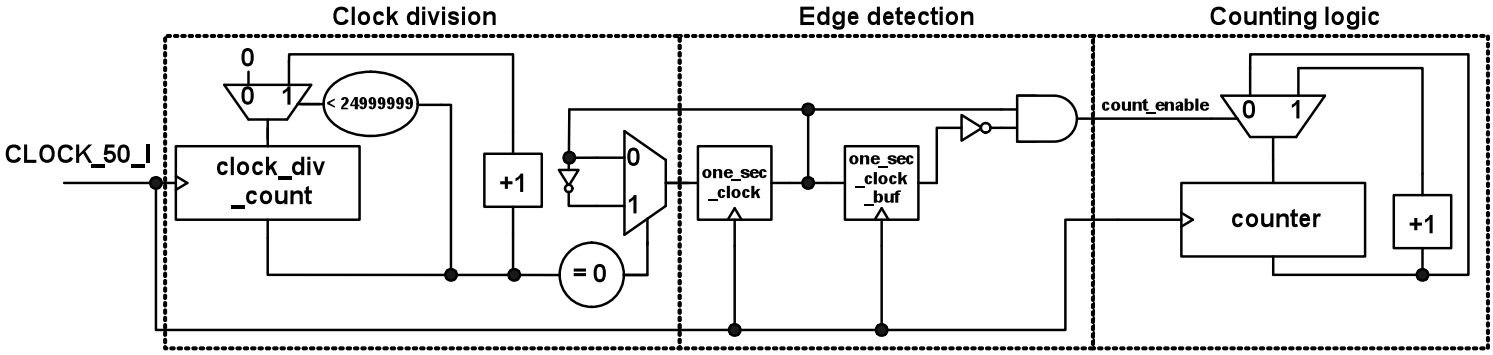

Figure 7 – Updating the counter every second using only one reference clock

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 board) there is a transient period during which the switch "bounces". The signal from the push-button (active low when pressed on this DE2 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.

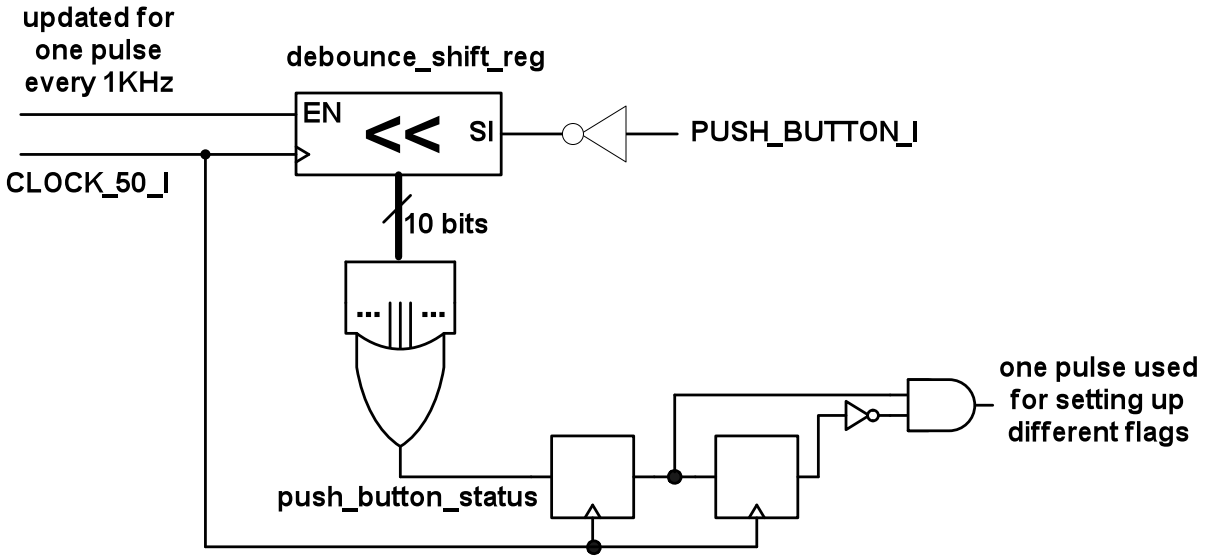

Figure 8 – Debouncing circuitry

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 and IDs
McMaster email addresses
Date

There are 2 take-home exercises that you have to complete within a week. When the source files are requested, submit ONLY the Verilog and "QSF" files. Label the top level modules as exercise1 and exercise2. If, for any particular reason, you will add/remove/change the signals in the port list from the port names used in the design files from the in-lab experiments, make sure that these changes are properly documented in the source code.

Exercise 1 (2 marks) – Change your in-lab contribution to ***experiment2*** as follows. All the 18 switches from the DE2 board should be read. The two leftmost switches (positions 17 and 16) will determine what is displayed onto the 7-segment-displays.

- If both switches 17 and 16 are low, then the position of the *most* significant switch (that is turned on) of all the remaining switches (positions 0 to 15) will be displayed onto the rightmost 7-segment-display in single-digit *hexadecimal* format; all the other 7-segment-displays should not be lightened (i.e., as if the DE2 board is not powered); note, if none of the switches are turned on, then none of the 7-segment-displays should be lightened;
- If switch 17 is low and switch 16 is high, then the position of the *most* significant switch of all the remaining switches will be displayed onto the four rightmost 7-segment-displays in *4-digit binary* format. As before, all the 7-segment-displays that are not used will not be lightened; also, if none of the switches 0 to 15 are turned on, then none of the 7-segment-displays should be lightened (the same applies to the following two combinations of switches 17 and 16);
- If switch 17 is high and switch 16 is low, then the position of the *least* significant switch (that is turned on) of all the remaining switches will be displayed onto the rightmost 7-segment-display in single-digit *hexadecimal* format;
- If both switches 17 and 16 are high, then the position of the *least* significant switch of all the remaining switches will be displayed onto the four rightmost 7-segment-displays in *4-digit binary* format.

Submit your sources and in your report write at most a half-a-page paragraph describing your reasoning.

Exercise 2 (2 marks) – Change your in-lab contribution to ***experiment5***, such that the counter updates every second in 2-digit BCD format within the range 00 to 59. While the counter is changing values within the 00 to 59 range, the functionality of push-buttons 0, 1 and 2 should be exactly the same as specified for the in the-lab ***experiment5***. However, when the end of the range is reached the following should occur. While counting up, when 59 is reached the counter should automatically stop. At this time, the activity on push-buttons 1 and 2 does not matter and the counter will be restarted only when push-button 0 is pressed again, at which time the counting direction will be automatically changed to count down. Following the same line of reasoning, while counting down when 00 is reached the counter should be frozen and it will be restarted only when pressing push-button 0 again, at which time the counting direction will be automatically changed to count up.

Submit your sources and in your report write at most a half-a-page paragraph describing your reasoning.

VERY IMPORTANT NOTE:
This lab has a weight of 4% of your final grade. The report has no value without the source files, where requested. 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). You must submit the electronic files before noon on the day you are scheduled for lab 2 (specific details of the electronic submission will be communicated in the lab and updated in this section of the document in due time). Late submissions will be penalized.