**Digital Logic II with Verilog**

**CST 133**

**Lab 6 - Registers & Synchronous Counter design**

**Objective:**

The purpose of this laboratory is to introduce the student to synchronous sequential counters. You will design a synchronous shift register, a binary counter, and a modulus ten BCD counter.

# **Part 1 -** Synchronous Shift Register

This bidirectional synchronous shift register is designed to incorporate many features a system designer may want in a shift register; the features include parallel inputs, parallel outputs, right and left shift serial inputs, control inputs, and an asynchronous clear line. A block diagram is shown below:



The register has four distinct modes of operation, namely, shift right, shift left and load. The function table for the inputs is shown below:

|  |  |  |
| --- | --- | --- |
| load | l\_r | Behaviour |
| 0 | 0 | Shift right (in the direction Q3 toward Q0) |
| 0 | 1 | Shift left (in the direction Q0 toward Q3) |
| 1 | 0 | Load shift register - Parallel load |
| 1 | 1 | Load shift register - Parallel load |

The reset signal is an asynchronous input signal that will clear the shift register. The data is loaded into the associated flip-flops and appear at the outputs after the positive transition of the clock input and the **“load”** signal high (‘1’).

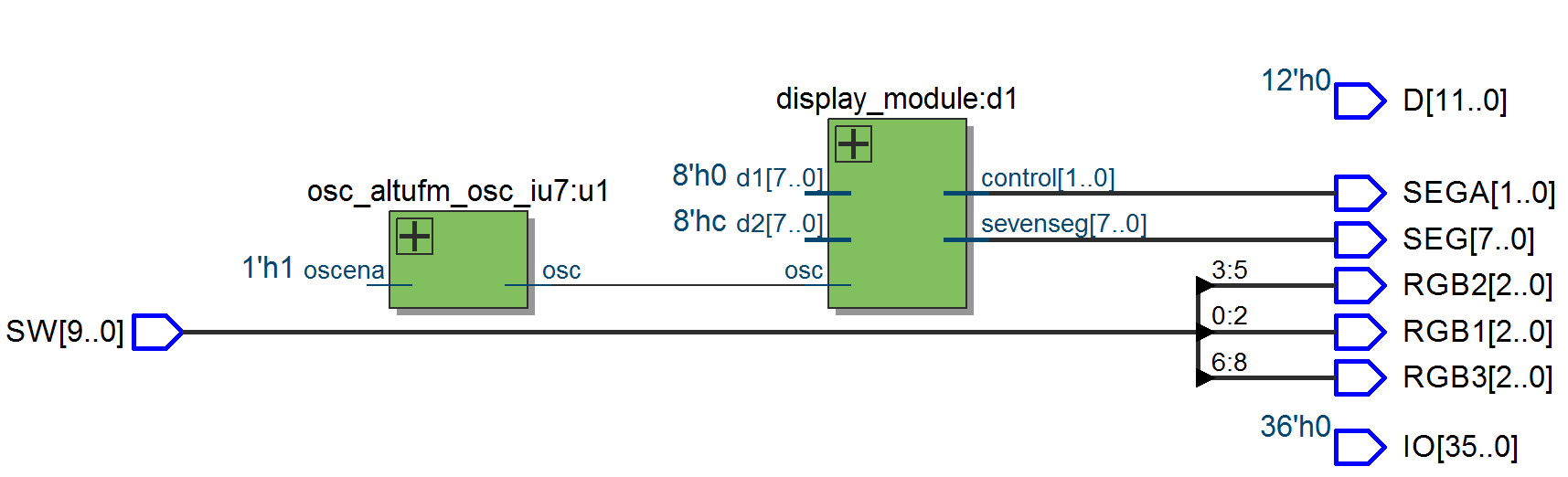
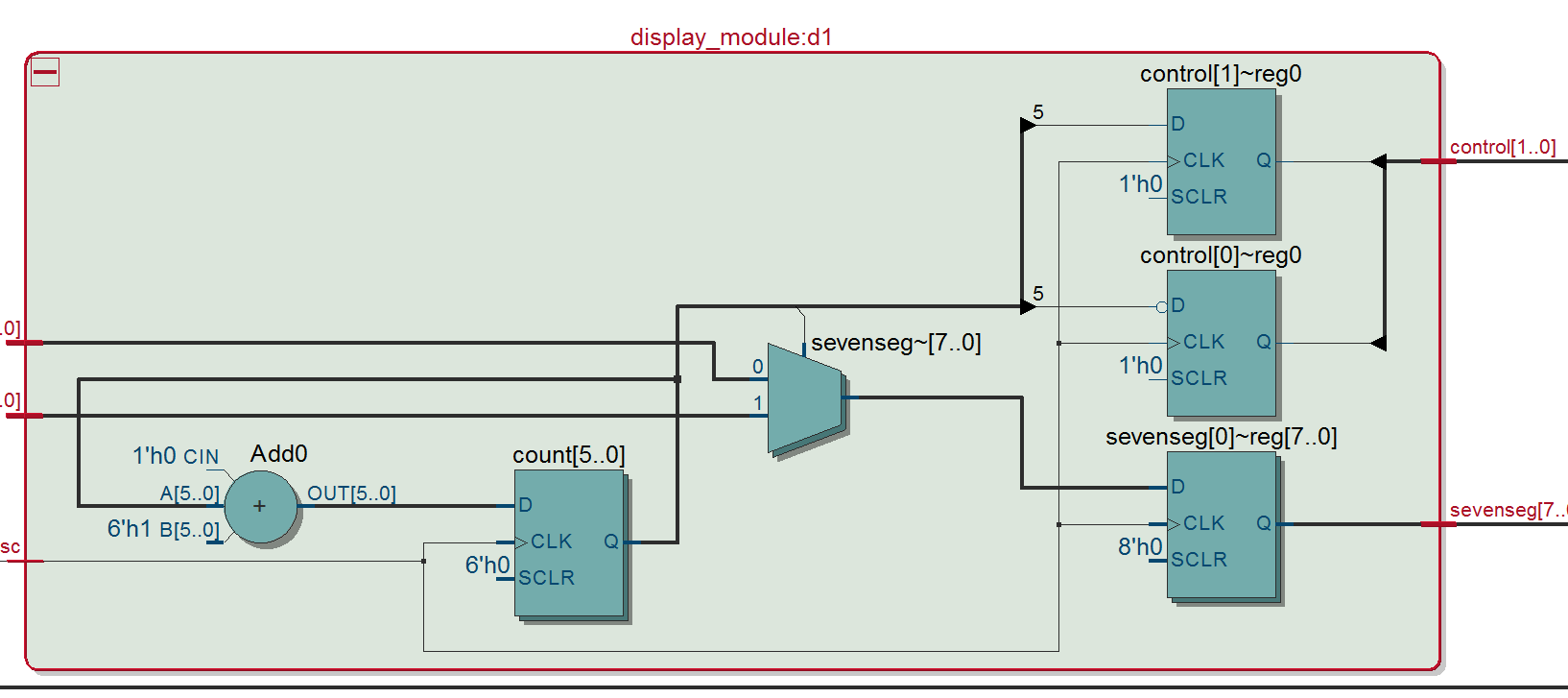
Shift right is accomplished synchronously with the rising edge of the clock pulse **“clk”** when **“load”** signal is LOW (‘0’). Serial data for this mode is entered at the “**sin”** (shift input signal). Shift left is similar with the only difference being the direction the data is shifted. The “**sout”** signal is attached to Q0 for a right shift and Q3 for a left shift.

**Implement and simulate this circuit using Quartus software as follows:**

1. Use the template project on the course website to start your design. Copy it over and call it lab6 using the Project > Copy Project menu option in Quartus.

1. Write a Verilog file that instantiates bidirectional synchronous shift register called shift\_reg.v . For this part you should use a behavioral style of Verilog code that specifies the bidirectional synchronous shift register as shown in the block above.
2. Instantiate a copy of shift\_reg in your top level file by connecting the inputs and outputs as specified in the table below.

1. Compile your code and use the RTL Viewer to examine the implemented circuit. Verify that the bidirectional synchronous shift register is implemented correctly.

1. Do a simulation to verify proper operation of the bidirectional synchronous shift register using a simulator tool of your choice (ModelSim or University Waveform editor).
   1. Exercise all input control signals with data that indicates that the design is operating correctly. Make sure to expand all inputs and outputs that are busses since the simulator may not show it correctly.
   2. Provide a screenshot of the simulation in your submission that notates all the test conditions. For instance, if in the simuation at 5 ns you are trying to test whether your add instruction works, circle or arrow that area with an explanation.
2. Provide a screenshot of the RTL Viewer for the **internals of the shift\_reg module**. Make sure that the screenshot is cropped to the RTL Viewer and simulation windows (Use the Windows snipping tool.)
   1. Go to TOOLS | NETLIST VIEWERS | RTL VIEWER after compilation
   2. The top level will open, but to get to the shift\_reg module, you will need to click on the + symbol.
   3. **As an example**, the below figure is from the top level of the RTL viewer. Note that this is from another design and only illustrates what the RTL diagrams should look like. 
   4. In order to see the internals of a particular block, you will need to click on the + symbol.
   5. After clicking on the +, a window showing the internals of the selected block will appear like the one shown below. 

7. Download the design to the DE10-Lite development board and then perform a physical test on your design.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Switches | Signal name | LED’s | Signal name | | SW[3] | D3 | LEDR[5] | Q3 | | SW[2] | D2 | LEDR[4] | Q2 | | SW[1] | D1 | LEDR[3] | Q1 | | SW[0] | D0 | LEDR[2] | Q0 | |  |  | LEDR[1] | sout | | SW[4] | load |  |  | | SW[5] | l\_r |  |  | | SW[6] | sin |  |  | | SW[7] | clock |  |  | | SW[8] | reset |  |  | | |  | | --- | | **Demonstrate the design to the instructor.** | |

# **Part 2 -** A binary counter

In this part of the lab you are to develop an eight bit synchronous binary up-down counter with a synchronous load and synchronous reset. The count advances one count on the positive transition of each clock pulse. The counter is synchronously reset to the zero state by a logical “1” on the **“reset"** signal and the positive level. A block diagram is shown below:



The counters function table for the inputs is shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| reset | load | up\_dn |  |
| 0 | 0 | 0 | Count up |
| 0 | 0 | 1 | Count down |
| 0 | 1 | 0 | Load the counter with D7-D0 |
| 0 | 1 | 1 | Load the counter with D7-D0 |
| 1 | X | X | Reset the counter to “00000000” |

**Implement and simulate this circuit using Quartus II software as follows:**

1. Continue using the previous project from part 1, noting that you will have to comment out the shift\_reg instance in your top level file.

1. Write a Verilog file that instantiates synchronous binary counter called sync\_counter.v For this part you should use a behavioral style of Verilog code that specifies the binary counter as shown in the block above.
2. Create an instance of the sync\_counter module in the top level file, connecting all the inputs and outputs to switches as indicated in the table below.

1. Compile your code and use the RTL Viewer to examine the implemented circuit. Verify that the binary counter is implemented correctly.

1. Do a timing simulation to verify proper operation of the bidirectional synchronous shift register using a simulator tool of your choice (ModelSim or University Waveform editor).
   1. Exercise all input control signals with data that indicates that the design is operating correctly. Make sure to expand all inputs and outputs that are busses since the simulator may not show it correctly.
   2. Provide a screenshot of the simulation in your submission that notates all the test conditions. For instance, if in the simuation at 5 ns you are trying to test whether your add instruction works, circle or arrow that area with an explanation.
2. Provide a screenshot of the RTL Viewer for the **internals of the sync counter module**. Make sure that the screenshot is cropped to the RTL Viewer and simulation windows (Use the Windows snipping tool.)

|  |  |  |  |
| --- | --- | --- | --- |
| Switches | Signal name | LED’s | Signal Name |
| SW7 | D7 | **D7** | Q7 |
| SW6 | D6 | **D6** | Q6 |
| SW5 | D5 | **D5** | Q5 |
| SW4 | D4 | **D4** | Q4 |
| SW3 | D3 | **D3** | Q3 |
| SW2 | D2 | **D2** | Q2 |
| SW1 | D1 | **D1** | Q1 |
| SW0 | D0 | **D0** | Q0 |
| BTN0 | clock |  |  |
| BTN1 | reset |  |  |
| SW8 | load |  |  |
| SW9 | up\_dn |  |  |

1. Download the design to the development board and then perform a physical test on your design.

|  |
| --- |
| **Demonstrate the design to the instructor.** |

**Part 3 -** *A modulus ten* BCD *counter*

In this part of the lab you are to develop a four bit synchronous up-down BCD counter with an asynchronous reset. The counter is a positive edge-triggered synchronous up/down BCD counter with a clock input (clk), an up/down count control input (up\_dn), an active LOW count enable input (ce) that allows the counter to count synchronously when the signal is a logical ‘0’. The output of the counter will be decoded to drive the seven segment display. A block diagram is shown below:



BCD counts from 0 to 9. Note that this is similar to your LAB4 adder setup, except instead of an adder we will wire the sync\_count module to the BCD decoder.

The counters function table for the inputs is shown below:

|  |  |  |
| --- | --- | --- |
| cen | up\_dn |  |
| 0 | 0 | BCD Count down |
| 0 | 1 | BCD Count up |
| 1 | 0 | The counter holds current value |
| 1 | 1 | The counter holds current value |

**Implement and simulate this circuit using Quartus software as follows:**

1. Continue using the previous project from part 2, noting that you will have to comment out the instances of shift\_reg and sync\_counter from parts 1 and 2 in your top level file.

1. Write a Verilog file that instantiates synchronous up-down BCD counter called bcd\_count. This module will count up in BCD.
2. Add a copy of your bcd\_decoder from Lab4 to the project.
3. Instantiate a copy of bcd\_count and bcd\_decoder in the top level file (This should be your de10\_generic.v file). Connect bcd\_count to bcd\_decoder together using a wire (4 bits as seen in block diagram above)
4. From the bcd\_decoder module, connect the outputs of the bcd\_decoder module directly to HEX0.
5. Compile your code and use the RTL Viewer to examine the implemented circuit. Verify that the synchronous up-down BCD counter is implemented correctly. Make sure you take a screenshot of the following:
   1. Screenshot of BCD counter internals (double click to get down to the lowest level)
   2. Screenshot of the top level of the RTL diagram
6. Do a simulation to verify proper operation of the bidirectional synchronous shift register using ModelSim.
   1. Exercise all input control signals with data that indicates that the design is operating correctly. Make sure to expand all inputs and outputs that are busses since the simulator may not show it correctly.
   2. Provide a screenshot of the simulation in your submission that notates all the test conditions. For instance, if in the simulation at 5 ns you are trying to test whether your add instruction works, circle or arrow that area with an explanation.
7. Configure your design and download it to the DE10-Lite development board and then perform a physical test on your design.

|  |
| --- |
| **Demonstrate the design to the instructor.** |

|  |  |  |  |
| --- | --- | --- | --- |
| Switch | Signal | Segment | Signal |
| SW[0] | cen | HEX0[0] | BCD decoder output |
| SW[1] | up\_dn | HEX0[1] | BCD decoder output |
|  |  | HEX0[2] | BCD decoder output |
|  |  | HEX0[3] | BCD decoder output |
| SW[9] | clk | HEX0[4] | BCD decoder output |
| SW[8] | reset | HEX0[5] | BCD decoder output |
|  |  | HEX0[6] | BCD decoder output |

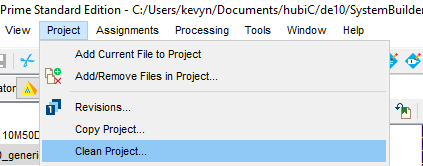
Submissions

The instructor does not guarantee last second availability. It is your responsibility to submit the lab in a timely manner. Note that the instructor’s mailbox may not be open after 5 PM.

A penalty of 20% per day up to 4 days will apply. After 4 days, no submissions will be accepted and you will receive a score of 0 for the lab.

For this lab you must submit the following to the course website

1. Demo the assignment to instructor in person before the start of the next lab
   1. Demo synchronous shift register
   2. Demo up-down counter
   3. Demo modulus 10 counter
2. Cleaned and zipped project folder. If you do not clean the project you will be subject to a 10% deduction in grade for this assignment.



1. RTL viewer screenshot for synchronous shift register
2. RTL viewer screenshot for up-down counter
3. RTL viewer screenshot for modulus 10 counter
4. Simulation screenshot for synchronous shift register (Show parallel load, and shift left and right functions.) Annotate.
5. Simulation screenshot for up-down counter
6. Simulation screenshot for modulus 10 counter (Show all ten values)

Note that screenshots must only capture the area of interest and must be readable to instructor.