Lab 2: MSI Combinatorial Components and VHDL

EEL 4712 – Fall 2014

Objectives:

The objective of this lab is to use VHDL to specify the designs and testbenches for various MSI combinatorial components. You will also explore the new features in Quartus, ModelSim, and the Altera DE0 board.

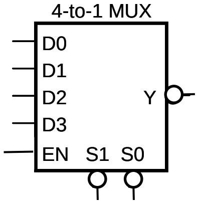
Required tools and parts:

Quartus II software package, ModelSim-Altera Starter Edition, Altera DE0 board.

Important: For this and the subsequent labs, put your name and section number as a comment at the beginning all VHDL files and other materials that you submit.

Pre-lab requirements:

1. Specify in VHDL the following 4-to-1 MUX:



- Note that the MUX has two **active low** inputs (S1, S0) and an **active low** output Y. So, you must specify your VHDL code accordingly.
- Use a **SELECT signal assign statement** or a **conditional signal assignment** statement.
- Modify the provided testbench (mux4to1WithAssert_tb) and perform a timing simulation.

Turn in on e-Learning: all VHDL files; a screenshot of the ModelSim transcript window that show "Simulation Finished" without any assertion errors.

Printout for your TA: the design of the MUX (.vhd file) and the timing simulation results. Note that you do not have to print out the simulation result of all possible combinations of the inputs, only some representative samples.

2. Design and simulate a display "decoder" for using the 7-segment LED displays on the Altera DE0 board. The ***logic*** table for the display "decoder" and the corresponding display are shown below:

- You can use any VHDL statements to implement the display "decoder", including **simple signal assignment statements**.
- Note that unlike the discrete LEDs, the 7-segment LEDs on the Altera DE0 board are **active low**. In other words, signals must be asserted "low" to illuminate the desired segments. So, you must specify your VHDL code accordingly.
- Write a testbench and perform a timing simulation. You don't have to use ASSERT statements for the display-decoder outputs.

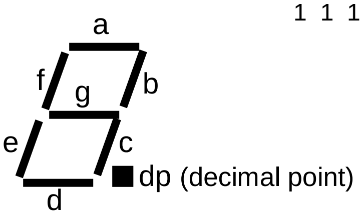
Turn in on e-Learning: all VHDL files; a screenshot of the ModelSim transcript window that show "Simulation Finished" without any assertion errors.

Printout for your TA: the design of the display "decoder" (.vhd file) and the timing simulation results.

Logic Table

(1 = true, 0 = false)

i3 i2 i1 i0	a b c d e f g
0 0 0 0	1 1 1 1 1 1 0
0 0 0 1	0 1 1 0 0 0 0
0 0 1 0	1 1 0 1 1 0 1
0 0 1 1	1 1 1 1 0 0 1
0 1 0 0	0 1 1 0 0 1 1
0 1 0 1	1 0 1 1 0 1 1
0 1 1 0	1 0 1 1 1 1 1
0 1 1 1	1 1 1 0 0 0 0
1 0 0 0	1 1 1 1 1 1 1
1 0 0 1	1 1 1 0 0 1 1
1 0 1 0	1 1 1 0 1 1 1
1 0 1 1	0 0 1 1 1 1 1
1 1 0 0	1 0 0 1 1 1 0
1 1 0 1	0 1 1 1 1 0 1
1 1 1 0	1 0 0 1 1 1 1
1 1 1 1	1 0 0 0 1 1 1



3. Design and simulate an 8-bit ripple-carry adder using the 4-bit adders from Lab 1.
   - The 4-bit ripple-carry adder from Lab 1 (VHDL version) will be used as a component in this design.
   - Use PORT MAP statements to specify the 8-bit adder.
   - Modify the provided testbench (adder_tb from Lab 1) and perform a timing simulation.

Turn in on e-Learning: all VHDL files; a screenshot of the ModelSim transcript window that show "Simulation Finished" without any assertion errors.

Printout for your TA: the design of the adder (.vhd file) and a representative printout of the timing simulation results.

4. Using PORT MAP statements, integrates the 8-bit adder (from Pre-lab 3) with **two** instances of the 7-segment LED display "decoder" (from Pre-lab 2).
   - The sum[7..0] outputs from the 8-bit adder should be connected to two instances of the display "decoder". The most significant 4 bits of the adder connect to one display "decoder" and the least significant 4 bits connect to the second display "decoder".
   - Perform a timing simulation using the same testbench as in Part 3, but modify it to show both the sum[7..0] outputs from the adder and the outputs from the two display "decoders". However, you don't have to use ASSERT statements for the new display-decoder outputs.
   - Note that during the in-lab Part 2 (below), the outputs of the two display decoders will be connected to two 7-segment LED displays on the Altera DE0 board.

Printout for your TA: the design of the design of the full circuit (.vhd) and a representative printout of the simulation results. (Nothing is turned into e-Learning)

***** Important: For full credit,** all prelab materials must be submitted to e-Learning by the beginning of your scheduled lab time.*

In-lab procedure:

1. Stand-alone testing of the 7-segment LED display "decoder" designed in Part 2 of the pre-lab.

- Select one of the four 7-segment LED displays on the Altera DE0 board:
  - Determine the FPGA pins that are connected to the selected 7-segment LED from the DE0 User Manual.
  - Use Pin Planner to assign the appropriate pins to the "decoder" outputs.
- Select 4 slide switches on the Altera DE0 board:
  - Determine the FPGA pins that are connected to the selected slide switches from the DE0 User Manual.
  - Use Pin Planner to assign the appropriate pins to the "decoder" inputs.
- Test the display "decoder" using a variety of test vectors. Your TA will ask you to display the results for one or more random test vectors.

2. Test the integrated circuit designed in Part 4 of pre-lab.
   - Similar to the steps outlined above in Part 1 of in-lab, determine the FPGA pins assigned to the outputs of **both** display "decoders".
   - Assign the appropriate FPGA pins on the Altera DE0 board to the 7-segment LED displays (See DE0 User Manual for details).

- Assign the carry-out Cout from the 8-bit adder to the decimal point (DP) most significant 7-segment LED display.
- Determine the FPGA pin assignments and connect the slide switches to the appropriate FPGA pins for use as X[7..0)] inputs.
- Build 8 switch circuits using a DIP switch bank and a SIP resistor pack on a bread board. Connect them to 8 GPIO pins on the Altera DE0 board. Use 3.3V and Ground from the GPIO headers to power your switch circuits.
- Determine the FPGA pin assignments and connect the selected GPIO header pins to the appropriate FPGA pins for use as Y[7..0)] inputs.
- Test the 8-bit adder. As before, your TA will ask you to display the results for one or more random test vectors.

3. Your TA will assign a task for you to design a MSI combinatorial component using VHDL, simulate it, and download it to the Altera DE0 board.