**CprE 381 – Computer Organization and**

**Assembly Level Programming**

**Lab #5**

*In this lab you will build up a design of an 8-bit adder/subtractor circuit. An adder/subtractor is a circuit that takes two inputs and a one-bit control value to select between addition or subtraction of the two inputs. You will be using both VHDL and Quartus Prime schematics to build the circuit. You will use ModelSim to simulate and test the circuits. This will be a helpful introduction for the following labs.*

*The book Free Range VHDL is provided on the class website and is a good tool to learn VHDL.*

**0)** Folder structure setup.

1. Create a new folder for lab 5 <user home folder>/cpre381/lab5. Use this directory to save VHDL and Quartus files.
2. Unzip the provided lab5.zip in the lab5 folder.
3. Note: Any mentioned of simulation in this document refers to using the program ModelSim to simulate VHDL entities. ECpE computers have ModelSim installed on Linux and Windows.

**1)** A **One’s Complimenter** unit is a combinational functional unit that takes a single input and negates each individual bit. Using VHDL create an 8-bit one’s complimenter and provide your solution to this problem (VHDL code, simulation waveforms using ModelSim) in your submission. You will find the VHDL ‘not’ operator useful for this problem. The ‘not’ operator works on both std\_logic and std\_logic\_vectors.

**2)** A **Two-Input Multiplexer** (or 2:1 mux) takes two inputs, and forwards one of them to its output based on the value of a control signal. Using VHDL create 2:1 mux with 8-bit inputs and 1-bit control signal. Provide your solution to this problem (VHDL code, simulation waveforms) in your submission. You can choose either of the two methods to implement the 2:1 mux:

1. use the ‘with-select’ operator to implement the 2:1 mux. You can find the details about the ‘with-select’ operator in the Free Range VHDL book provided on the class website.
2. Draw a truth table and K-map and implement the multiplexor function using primitive gate operators like ‘and’, ‘or’, ‘not’, etc.

**3)** A **Full Adder** takes three bits as input (two operands and one carry-in bit) and calculates both a sum and a carry-out value. A 1-bit full adder VHDL implementation and a Block-Symbol File (full\_adder.bsf) has been provided for you in the lab5.zip. You can use the block symbol file to create an 8-bit full adder in Quartus using the schematic tool. Using Quartus create an 8-bit full adder schematic and export it to a block-symbol file (full\_adder\_8\_bit.bsf).

1. Open Quartus Prime 16.0. Select File 🡺 New Project Wizard. Click Next. Choose the working directory for the project to be <user home folder>/cpre381/lab5 and name the project and top-level design entity as “lab5”.
2. Browse for the lab5 folder and select to add full\_adder.bsf and full\_adder.vhd to the project. Then select Finish.
3. Select File 🡺 New… 🡺 Block Diagram/Schematic File. A new, blank schematic will appear on your screen. To insert components, you can double click on the empty space in the schematic. Instantiate a 1-bit full adder by selecting Project 🡺 full\_adder in the Symbol dialog. Click to place the instance. Create an 8-bit full adder using the 1-bit full adder block and save it as full\_adder\_8\_bit.bdf:
   1. Instantiate 8 full\_adder blocks in the schematic file.
   2. Create three input pins and two output pins via the Inputs button
   3. Right click on the input pins and rename one to i\_A[7..0], one to i\_B[7..0], and one to i\_Cin. The [7..0] tells Quartus that these inputs are 8-bit vectors.
   4. Right click on the output pins and rename one to o\_Sum[7..0] and the other to o\_Cout. The carry in and carry out are 1-bit.
   5. Wire the i\_Cin to the first full\_adder’s input i\_Carry\_in.
   6. Wire the o\_Cout to the last full\_adder’s output o\_Carry\_out.
   7. Using the Orthogonal Bus Tool create a bus for i\_A[7..0], i\_B[7..0], and o\_Sum[7..0] and give each bus the same name and the pin that it’s connected to. To rename a bus right click on the bus and select Properties.
   8. Using the Orthogonal Node Tool wire the remaining ports of each 1-bit full adder to the corresponding buses to create an 8-bit full adder. Name each wire with the matching index of the 1-bit full adder it’s connected to. For example:
      1. on the first 1-bit full adder, i=0:
         1. Wire port i\_A to the i\_A[7..0] bus and name the wire i\_A[0]
         2. Wire port i\_B to the i\_B[7..0] bus and name the wire i\_B[0]
         3. Wire port o\_Sum to the o\_Sum [7..0] bus and name the wire o\_Sum[0]
      2. On the i*th* 1-bit full adder:
         1. Wire port i\_A to the i\_A[7..0] bus and name the wire i\_A[i]
         2. Wire port i\_B to the i\_B[7..0] bus and name the wire i\_B[i]
         3. Wire port o\_Sum to the o\_Sum [7..0] bus and name the wire o\_Sum[i]
   9. Double check that all ports and wires are connected properly. If there is an ‘x’ on the end of a wire, then it is not connected.
4. Simulate the 8-bit full adder provide screenshots of the simulation waveform with multiple test cases in your submission.
   1. To simulate the 8-bit full adder we first need to convert the schematic into VHDL. With the 8-bit full adder schematic tab selected go to File 🡺 Create/Update 🡺 Create HDL Design File for Current File. Select File type as “VHDL” and click OK.
   2. Open ModelSim and compile both full adder VHDL files (full\_adder.vhd, full\_adder\_8\_bit.vhd) and simulate your design.

**4)** An **Adder/Subtractor with Control** takes two values (A, B) as input, plus a control bit (nAdd\_Sub), and calculates A+B when nAdd\_Sub = ‘0’, and A-B when nAdd\_Sub = ‘1’. Using Quartus create an 8-bit Adder/Subtractor schematic and simulate it.

1. Draw a schematic showing an 8-bit adder/subtractor with control. How is the ‘nAdd\_Sub’ bit used? Include your answer in your report. The only components you can use:
   1. 1, 8-bit one’s complimenter
   2. 1, 2:1 mux (8-bit)
   3. 1, 8-bit full adder
   4. Inputs, Outputs, and Buses
2. You can import your one’s complimenter, 2:1 mux, and 8-bit full-adder VHDL files as schematic blocks. To do this, add your VHDL files to your project via Project 🡺 Add/Remove files in Project… Then open the VHDL file in Quartus and go to File 🡺 Create/Update 🡺 Create Symbol Files for Current File. Create a new block diagram file in your Quartus project and implement the adder/subtractor with control using only these three components.

1. Thoroughly test this design for correctness in ModelSim. Follow 3d) if you forgot how to go from block diagram to simulation. Don’t forget to Provide multiple waveform screenshots in your write-up to confirm that this component is working correctly. What test-cases did you include and why?

**Submission:**

* Create a zip file *Lab-5-submit.zip*, including the completed code, block diagram files, and screenshots from the four lab parts.
* The lab report which answers all questions from this document. You can include your screenshots in the report if you’d like.
* The file names in your zip file should be self-explained.
* Submit the zip file on BlackBoard Learn under “Lab 5” assignment.