Lab #3: Introducing ModelSim and VHDL

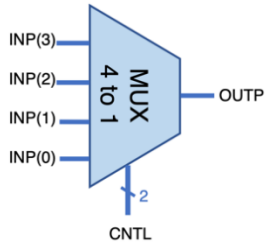
3. Introduction

In the previous labs you have designed logic circuits using **schematic** diagrams, and you have implemented them on a DE10 board to check if they work. In this lab, you will model your design using **VHDL,** and will check if it works in **simulation** using **ModelSim,** before also implementing on your DE10 board.

- In part 3.1 you will use a provided VHDL design for a **Multiplexer** to learn the basic **ModelSim** simulation process.
- In part 3.2 you will re-design your **7-segment decoder** in **VHDL** (much easier!) and simulate it using **ModelSim**
- In part 3.3 you will develop VHDL code for a simple traffic lights controller and test it in the lab.

3.1 Simulating and Implementing a Multiplexer using VHDL

A Multiplexer (shown below) is like a multi-way switch where the binary number applied to the control (**CNTL**) input specifies which of the ***n*** possible inputs (**INP**) is connected to the output (**OUTP**). In this section you will simulate, implement, and test a 4 to 1 multiplexer using VHDL and ModelSim.



Truth Table

CNTL	OUTP
00	INP(0)
01	INP(1)
10	INP(2)
11	INP(3)

1. Download **Lab3P1.zip** from Blackboard and unzip it to a folder called **Lab3P1**. The unzipped files include **Lab3P1.vhd**, **Lab3P1.do** and **Lab3P1-pins.qsf**.
2. Then open Quartus and create a new **Project** called **Lab3P1** in this folder, remembering to set the device to be **Cyclone V**, **5CSXFC6D6F31C6**, and setting the **Assignments > Device > Pin Options** > **Voltage** to **3.3-V LVTTL**.
3. Select **Project > Add files in Project** to add your downloaded **Lab3P1.vhd** file to your Project. This contains the VHDL code for implementing a 4 to 1 multiplexer. **Open** the file and see if you can figure out how it works.
4. Select **Assignments>Import Assignments** and browse to your downloaded **Lab3P1-pins.qsf** file to import the pin assignments. This maps the **CNTL** control inputs of your Mux4 entity to **SW[9..8]**, and the four **INP** inputs to **SW[3..0]**. The selected output **OUTP** is assigned to **LEDR[0]**. After importing, select **Assignments -> Pin Planner** to double check if all your input and output signals have a pin name in the **Location** column.
5. In the **Project Navigator>Files** tab, select your VHDL file and **R-click > Set As Top Level Entity**. Then **Compile** your Project as usual and fix any errors.

We now want to Simulate our design using ModelSim to check if it works before implementing it.

6. Select **Tools > Run Simulation Tool -> RTL Simulation** to bring up **ModelSim**.

- If you got a message saying "You did not specify an EDA simulation tool.", go to **Tools > Options > EDA Tool Options**, and set the lowermost **ModelSim-Altera** field to:
  C:\intelFPGA_lite\20.1\modelsim_ase\win32aloem
- If you are asked to select a simulation language, choose "VHDL" from the list.

7. Wait for ModelSim to open, then within ModelSim, click: **Simulate >Start Simulation** to prepare to simulate.
   - A pop-up window will appear to enable you to specify the entity to be simulated.
   - Expand the **work** library by clicking the **+** to the left of it. Select the Entity **Lab3P1** and click **ok**.
   - The simulator will configure itself for simulation
8. If necessary, click **View > Wave** and **View > Objects** to open the Waveform and Objects windows
9. In the Objects window, select the signals you want to watch, and **R-click > Add Wave** or drag them into the Wave window. In this case Add the **INP**, **CNTL**, and **OUTP** signals to the Wave window.
10. In any simulation you need to specify the **inputs** that are to be applied whose response you want to observe. These inputs (and how they change with time) are specified in your downloaded **Lab3P1.do** file. **File>Open** it and note particularly:
    - Lines starting with a **#** are **comments**
    - The **INP** inputs applied to the Mux are 1010 at t=0 ns, changing to 0101 at t=100 ns
      - `force INP 1010 0 ns 0101 100 ns`
    - The **CNTL** control inputs change every 20 ns starting at 00, then 01, then 10, then 11, then repeating
      - `force CNTL 00 0 ns, 01 20 ns, 10 40 ns, 11 60 ns, -repeat 80 ns`
    - [Values can be specified in simple binary (as above) or multibit binary (e.g. 2#0101) or hex (e.g. 16#3F)]
11. Given the **INP** and **CNTL** inputs specified above, and your knowledge of what a multiplexer does, <u>in your notebook</u>, fill in the following underlined blank places for CNTL, and predict the value of the **OUTP** output:
    - At t=10 ns, (INP=1010; CNTL=00) ⇒ OUTP= __
    - At t=30 ns, (INP=1010; CNTL=__) ⇒ OUTP= __
    - At t=50 ns, (INP=1010; CNTL=__) ⇒ OUTP= __
    - At t=70 ns, (INP=1010; CNTL=__) ⇒ OUTP= __
12. Now Run the simulation, applying the inputs specified above by typing **do ../../Lab3P1.do** in the Transcript/Command window (the lower pane of the ModelSim tool). [Note: ModelSim runs in a subdirectory, hence the ../../ prefix is needed to specify the parent directory location of the Lab3P1.do file]
    - It is assumed Lab3P1.do is in your Quartus project folder.
    - ModelSim assumes the current working folder is Quartus_project_folder/simulation/modelsim.
    - Type "pwd" in the Transcript/Command window to find out the current working folder.
13. To view the results on an appropriate (approx. 80 ns) timescale, in the Wave window, **R-click > Zoom Full** or click on the zoom In / zoom Out button to show your waveforms nicely.
    - Do the results match those you predicted in Step #11 above? Yes/No: _____
    - Explain what is happening at t=150 ns: INP=_____, CNTL=__ ⇒ OUTP= __
    - If any signals are shown in **red** / unknown, it indicates you forgot to initialize them in your VHDL code.
14. Open the **Snipping Tool**, and capture a picture of the Wave window. Name it MUX41 and save it to the Lab3P1 folder. Leave Modelsim open for checkoff and switch back to Quartus window.
15. Having validated your design in Simulation, now return to **Quartus** and **Program** your design on the DE10 board using the **Lab3P1.sof** file that was generated during compilation in step #5.
    - If there was any error you fixed in ModelSim, you need to recompile your design again in Quartus.

16. Use the **SW[9..8]** (CNTL input) switches to select each input in turn, and in each case verify that the output then responds correctly as you flip the **appropriate** input switch from among the **SW[3..0]** multiplexer (INP) inputs.

Demonstrate this part to the instructor/TA.

3.2 Implement a 7-Segment decoder using VHDL

The aim of this section is to implement the same 7-segment decoder that you designed in Lab2, only this time using VHDL. You should find it much quicker and easier ☺!

1. Download **Lab3P2.zip** and upzip it to a new folder called **Lab3P2**.
2. Open Quartus and create a new **Project** called **Lab3P2** in this folder, remembering to set the device to be **Cyclone V**, **5CSXFC6D6F31C6**, and setting the **Assignments > Device > Pin Options** > **Voltage** to **3.3-V LVTTL**.
3. Select **Project > Add files in Project** to add your downloaded **Decode7seg.vhd** file to your Project. It contains incomplete VHDL code for implementing a 7-segment decoder. Your job is to complete it!
4. **File > Open** your **Decode7seg.vhd** file and edit it to define its behavior. Do not use the logic functions you derived in Lab2. Instead, use a **WITH SELECT signal assignment** to define *all* the outputs associated with each row of your Lab2 **truth table** in single statement. See the instructional comments in the file.
5. To set the Pin Assignments, select **Assignments>Import Assignments** and browse to **DE10-pins.qsf** and hit **ok**. This maps the inputs and outputs of your VHDL entity to the switches, LEDs and HEX displays on your board. After importing, select **Assignments -> Pin Planner** to double check if all your input and output signals have a pin name in the **Location** column.
6. In the **Project Navigator->Files** tab, select your VHDL file and **R-click > Set as Top Level Entity**. Then **Compile** your Project as usual and fix any errors in your VHDL code.
7. Select **Tools -> Run Simulation Tool -> RTL Simulation** to bring up **ModelSim**. Then select **Simulate >Start Simulation,** then **View** the **Objects** and **Wave** windows and **drag/add** the input / output signals to the Wave display.
8. Open the provided **.do** script file named **Lab3P2.do** (included in Lab3P2.zip) and observe its contents. Then run this script by entering "**do ../../ Lab3P2.do**" in the Transcript/Command window (lower pane).
9. **Right-click** in the **Wave** window and **Zoom** In/Out to see your results on an appropriate scale, and check if the output matches your Lab 2 truth table specification. If you need to modify your VHDL code, select **Simulate >End Simulation**, then switch to Quartus to make changes.
10. Having validated your design in the Wave window, open the Snipping Tool, and capture a picture of the Wave window. Save it in your Lab3P2 folder. Leave Modelsim open for checkoff and switch back to Quartus window.
11. Now **Program** your design on the DE10 board using the **Lab3P2.sof** file that was generated during compilation in step #6.
12. Flip the **SW[3..0]** switches starting at **0000** and then counting up to **1111**, and check the display shows the correct hex value in each case. If there are any **errors**, note which segment is incorrect and check the relevant bit in your VHDL code.

Demonstrate this part to the instructor/TA.

3.3 Designing a Traffic Light Controller in VHDL

The aim of this section is to design a traffic light decoder in which the count input comes from a 4-bit counter that increments by one on every rising clock edge, and the lights output is a bit pattern to control the six traffic lights (i.e., the LEFT light Red, Yellow, Green, and the RIGHT light Red, Yellow Green) in the lab room. The sequence of traffic lights should be RG (Red Green) – RY – RR – GR – YR – RR – RG - …

1. Complete the table below (copy to your notebook) to define the desired behavior of your traffic lights controller. The 4-bit counter counts from **0000** to **1111, which** gives you **16** clock cycles before it wraps around back to 0000.
   - Decide how many of the total 16 cycles you want to assign to each traffic light pattern (e.g., give more cycles to RG and GR states to allow cars to pass).
   - Then work out what range of count value inputs must therefore correspond to each of these states
   - Finally specify what bit pattern should be outputted in each case to switch the correct lights on/off.

Traffic Lights (Left-Right)	**# cycles**	**count** values (from – to)	**lights –** LR LY LG RR RY RG (active high signals)
Red – Green (RG)	4	0000 - 0011	1 0 0 0 0 1
Red – Yellow (RY)	2		
Red – Red (RR)			
Green – Red (GR)			
Yellow – Red (YR)			
Red – Red (RR)			
repeat	Sum=16	- 1111	

2. Download **Lab3P3.zip** from the blackboard and unzip to a new folder called **Lab3P3.**
3. As before, create and configure a new Project (this time **Lab3P3**) and **add** the downloaded **Lab3P3.vhd** to the project.
4. Edit **Lab3P3.vhd** using a **case** statement to assign the input **count** values to corresponding **Lights** bit patterns (as defined by your table).
5. Import your Pin Assignments from **Lab3P3-pins.qsf**. This maps the **count** and **lights** signals in your VHDL code to the signals of the physical traffic lights board in the classroom.
6. Then **Compile** your Project as usual and fix any errors in your VHDL code.
7. Enter the **Simulate** mode, **View** the **Objects** and **Wave** windows and **drag/add** the input / output signals to the Wave display.
8. Open the provided **.do** script file named **Lab3P3.do** and observe its contents. Then run this script.
9. **Right-click** in the **Wave** window and **Zoom** In/Out to see your results on an appropriate scale, and check if the output matches your table design specification. Having validated your design in the Wave window, open the Snipping Tool, and capture a picture of the Wave window.
10. Go to the **traffic light station in the back of the room**, log into the computer connected to the lights, map your **H drive** if this is the first time you are using the computer, load Quartus then select **Tools** -> **Programmer** from the menu, and program the DE10 board using your Lab3P3.sof. Observe the traffic lights cycling through the sequence you defined in your table. Go back to your workstation if the traffic lights behavior is not what is expected and if you need to modify your VHDL code or Quartus project.

Demonstrate this part to the instructor/TA.

Submission for Lab 3:

After you have been checked off all the parts by the instructor/TA, zip all the Lab3 project folders into one zip file, and name the zip file Lab3_<your last name>.zip, e.g. Lab3_Wang.zip, and submit it to Lab3 assignment in Blackboard. **Make sure the project folders include the ModelSim simulation waveform pictures you captured.**