EE 5143 – M3 Lab – Dr Rahman

This lab assignment is to give you a brief introduction to some features of the Quartus Prime Lite EDA package and to show you how to enter and compile code using this tool.

Quartus Prime (hereafter referred to as QP) is an integrated suite of many utilities that help in almost all aspects of digital system development. We'll encounter some but by no means all of these utilities in this course. By digital system we mean either a simple digital circuit (as simple as a single logic gate) or a much larger system composed of the interconnection of many sub-circuits each of which may also themselves be composed of lower level circuits. Hierarchical design is, in fact, a powerful idea that allows one to build systems of amazing complexity by designing simpler systems and then interconnecting them. Keep this in mind.

QP allows you to enter the description of a digital system in two principal ways: using a hardware description language (HDL) or using schematics. We will try out both but our main focus will be on HDL design entry. QP can be used with both VHDL and Verilog – the two main HDLs in use at this time. We'll make exclusive use of VHDL in this course.

Before you start using QP make a home for all of your development code. Go to the folder on your computer where the QP installation exists. On my computer this is a folder called 16.0 on the main C drive. It is called 16.0 because that is the version of QP I am using. Your version will most likely be different but that is fine. The name 16.0 was given by the QP installer when it originally installed QP Lite on my machine. Inside this folder you can create a folder for your code. Mine is called AlteraVHDLProjects but you can use any name of your choice. Then go inside this new folder and create some other folders to keep different types of code in, such as a folder for EE5143 and others for any other purposes (you can create the other folders later as the need arises). This way QP will not have to struggle as it compiles your code because code will be in a sub-directory of the main QP folder. This is important. You can also keep a folder called something like 'documentation' inside your projects folder to keep things like instruction manuals and user guides in one place for easy reference. Very often there is need to refer to these guides in the midst of writing code.

To get started for this lab, invoke QP and then select New Project Wizard – we use this every time we start a brand new project. For project directory name use M3Lab (suggested). The directory location would then appear something like:
C:/altera_lite/16.0/AlteraVHDLProjects/EE5143/M3Lab
You can do this either by first creating an EE5143 folder inside the AlteraVHDLProjects folder and then an M3Lab folder inside the EE5143 folder (before you start QP) or you can create those folders while you are using the wizard, either way works. All of your M3 lab files will then be created in the M3Lab folder (this includes code text file, compiled code file and other files).

Name the project as M3Lab in the wizard. The top level design entity then gets automatically named as M3Lab – remember this for later naming your VHDL entity.

Select next on the wizard and select empty project there.

Skip through the add files dialog box – we do not have any files to add to our project here as we are starting from scratch. In larger and especially hierarchical projects we may need to add some files at this step.

In the Family, Device & Board settings dialog you don't have to do anything with the board but make sure that the Family is set as Cyclone IV E. It doesn't matter for this lab but is a good practice if you will be using the Cyclone IV E based DE0-Nano board later on. Leave everything else as it is there and scroll through the device list (very long) to select EP4CE22F17C6 device. That line will get highlighted and you can see the different resources available with that device, such as a total of 154 I/O pins. Select Next then Next again and finally select Finish to come out of the wizard. The general structure of your project is now in place and you are returned to the main QP screen. Note that in the designation EP4CE22F17C6, F refers to the device being packaged as a flat-pack, C stands for commercial grade and the last 6 is a speed grade. The smaller the speed grade number the faster the device can work. 6 is actually a very high speed.

In the Project Navigator on the left you will see the device name at the top (this is the targeted device for the code and the compiler will generate a net –list for placement inside this particular device). Here is the good part – because QP now knows which device to go for it will automatically generate the best circuitry possible to carry out whatever you describe in your VHDL code. Thus, if you write, say, A <= B + C, then QP will generate the best possible adder to do this job with the three signals called A, B and C. You do not need to know how to design an adder circuit using gates etc. This is called inferring a circuit and is one of the most powerful facilities available with all HDLs. QP is smart so if the targeted device has a built-in (hard) adder then it will use it otherwise it will interconnect logic elements (much like interconnecting gates) to build you a (soft) adder inside the device. It is all automatic and the only thing you need to ensure is to write code which is syntactically (grammatically) and semantically (functionally) correct. Underneath the device name in the navigation column you will see the name of your top level entity M3Lab (or whatever else you chose to call it).

If you double click on the project/entity name at this time (try it) then you will be informed: Can't find design entity "M3Lab". This is because no code files yet exist which declare an entity of this name. We need to do something about this now.

To start a code text file go to files menu and select New. From the selection box called New select VHDL file. You will select Verilog HDL file if you were coding in Verilog. VHDL code text files are stored with extension .vhd whereas Verilog code text files are stored with extension .v. By default, the new file you create is called vhdl1.vhd. You can give it a name of your own choice when you save it. You can now save the (empty) file as M3Lab. Henceforth it will remain as the VHDL code file with the name M3Lab.vhd. This is essentially just a simple text file.

Now time to type in some code. Remove the navigation pane on the left by clicking on the x sign, if it interferes with typing the code (you can always get it back, if needed, by using Alt+ 0 or through the View menu). Type the following simple code declaring an entity and defining its associated architecture:

```
---------------------------------
-- Entity Declaration --
---------------------------------
-- here we describe the I/O of the design

entity M3Lab is
	port(
		a, b : in bit;
		c : out bit
	);
end entity;

----------------------------------
--	Architecture Definition  --
----------------------------------
--	here we put the description code of the design

architecture MyArchitecture of M3Lab is

begin

c <= a AND b;

end MyArchitecture;
```

Note that two dashes at the beginning of any line indicate a VHDL comment. VHDL reserved words (keywords) are colored blue by the text editor. Built-in data types and operators appear in red whereas user defined identifiers appear in black. Keep in mind that VHDL is (mostly) case insensitive and is also insensitive to white space so you can format your code text any way you want for better legibility. Also, note that there is a little icon with two lines of numbers just above your code which allows you to display or not display line numbers with your code.

You should already know what entity and architecture are and about their proper construction from M3 instructional material and reading from your textbook. This code takes two one bit signals 'a' and 'b' (these could come from two FPGA pins) ANDs them and the ANDed result is 'c' which could be routed to another FPGA pin. Of course, you will never use an FPGA to just function as a simple AND gate but this was just to get started.

Now you can compile your simple code by either pressing the blue triangle symbol in the tool bar or by going to 'Processing' menu and selecting 'Start Compilation'. It will take a while to compile your case, depending on the speed of your computer. Progress and various messages are displayed at the bottom in the messages area. Completion of various tools in the tool chain is displayed at the left. At the end of the process you should get a message saying 0 errors. Do not worry about warnings – quite a few may come your way; just ignore them for now. The code has been compiled. The last compilation window gives you a 'compilation report' which lists usage of resource utilization inside the FPGA. Even for the most complex project we'll build in this course, only a tiny fraction of the resources inside the FPGA will be used.

Now go to Assignments menu and select Pin Planner. This will open a new window to show all the FPGA pins. Note that the pins appear in rows and columns and are identified as G3, M6, T2 etc. In the horizontal window at the bottom you will see your signals a, b and c listed as nodes. A column will show their fitter locations. These are the FPGA pins that have been assigned automatically to your input and output signals by QP. In almost all cases you will need to change the signals to your own pin locations so as to connect to hardware that is already externally connected to the FPGA. Changing the pin binding is easy. Just click on any node name and while keeping the left mouse button pressed move that node and drop it on a selected pin on the chip view at the top. That pin location should now appear in the location column. Note that not all pins are selectable but most are. Once this has been done you need to compile the design again to physically bind the new pins to signal nodes. To go back from the Pin Planner window to the editor window just select QP from the Windows menu.

You can also go to Tools > Netlist Viewer > RTL Viewer to see what you have created in the FPGA. Generally it shows interconnected logic elements (LEs) but in this simple case it will just show an AND gate.

Make a copy of your VHDL code text window, the pin assignment window and the RTL Viewer window to submit as your lab assignment.

Next, go and make some arbitrary changes to your VHDL code so as to introduce errors and see what kind of error messages are generated on compilation.

Next week we'll use the DE0-Nano board for module 4 lab. To prepare for it, make sure you read the first three chapters of the DE0-Nano user manual this week (whether you have the board or not).