

Creating a DE2 Project Template

Introduction

The remaining labs in Digital II all involve increasingly complicated designs which you will implement and test on the Altera DE2 board. You should find it easier to do these labs if you learn how to set up a hierarchical project in Quartus II, and always start with a common method (i.e. a template) for developing each project.

On page 20 of the Quartus tutorial there is a description of how you can import a file to perform pin assignments rather than doing it manually with the Pin Assignment Editor. As you probably can better appreciate now, using the Assignment Editor is not only tedious but the process is highly susceptible to errors, some of which can be very difficult to find.

If you look carefully at the DE2 board, you will see that all the I/O devices have identifying labels printed on the board directly next to them. One of the virtues of importing the *DE2_pin_assignments* file is that all these names as printed on the board itself are the names that are mapped to the pins*.

*With one minor difference – all identical elements that are labeled on the board with numerical suffixes (i.e. LEDR7 – LEDR0, KEY1, etc) are named as array elements in Quartus (LEDR[7..0], KEY[1]).

This tutorial will show you how you can:

1. Never need to use the Pin Assignment Editor again.
2. Still use functionally descriptive names for I/O to your blocks (D, Q, clk, etc).
3. Easily keep track of which DE2 devices you've connected to I/O pins in your blocks (Clk is connected to KEY[0], an 8-bit output called SUM is displayed by LEDR[7..0], etc).

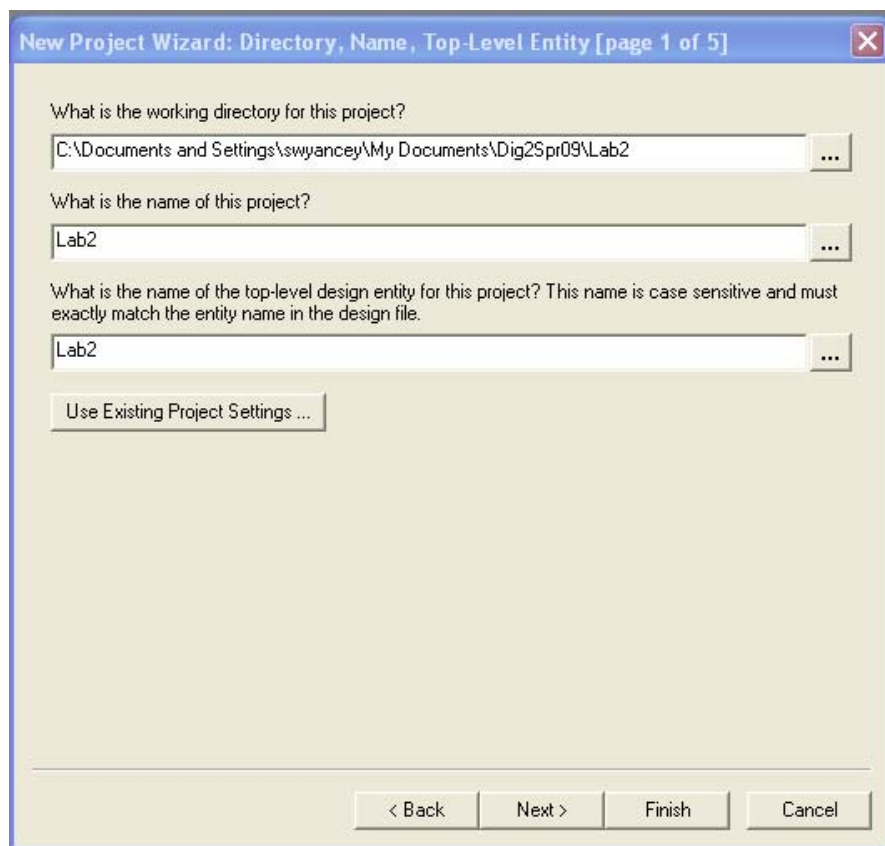
Step 1: Create a project with the New Project Wizard

You should read through the Quartus tutorial for a better understanding of the whole process, but these are the only steps you need to follow for the labs in this course:

- 1) Open Quartus and go to “File->New Project Wizard...”
- 2) On page one where the default project directory is “C:\altera\quartus60”, always change the project directory to one you create in your own “My Documents” folder – Click “...”, then click “My Documents”, then drill down to the correct subfolder (and if needed click on the “Create New Folder” icon and make a new subdirectory then click on “Save” without entering a filename).

One reason to not use the “C:\altera...” directory is so that you don’t get files from your project mixed up with someone else’s files. Another is to make it easier for you to find the directory you need to copy when you backup or move your project from one computer to another.

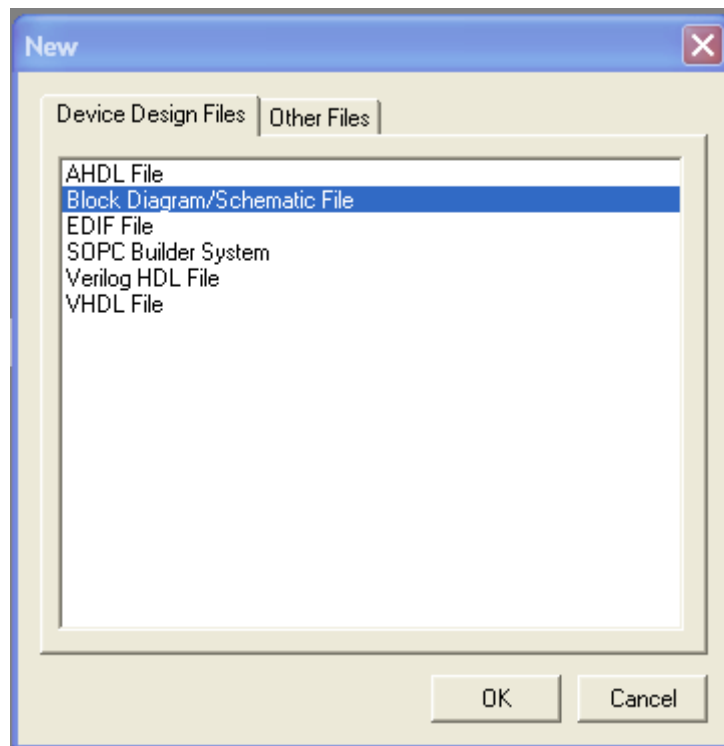
- 3) Name the project “LabXX” (or something more descriptive like “PulseDetector”).
- 4) In most cases you should let the top-level design entity name default to the project name you just filled in.
- 5) This shows creating a project for Lab 2:



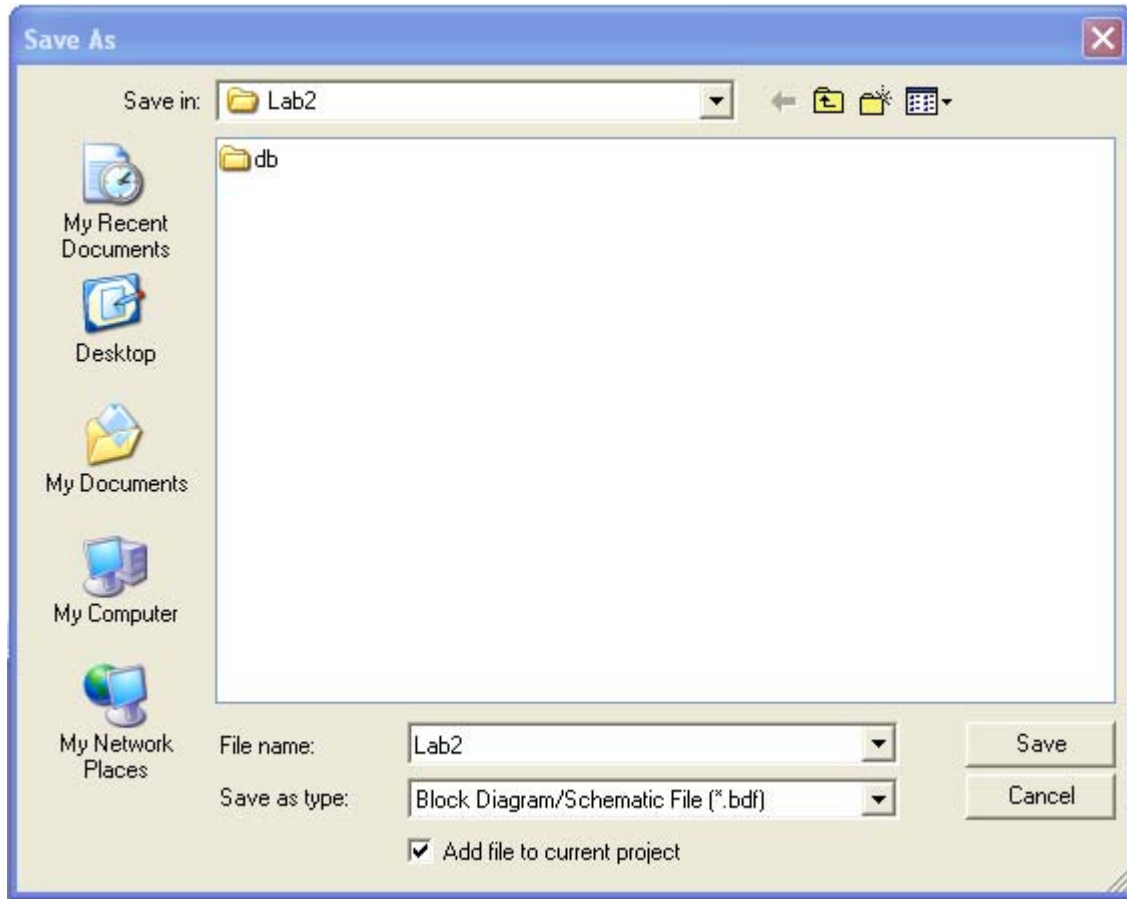
- 6) On page 2 you probably won't have any .vhd or .bdf files already in place to add to the project just yet, especially if you just created a new working directory, so skip this step. [This tool supports creating a whole complicated web of different project directories where part or all of one project can be incorporated by reference in another project, but for now you should probably stick to a simpler plan – where all files you use in a given project are kept in that project directory – this is much easier to keep track of when you frequently need to move your project from one machine to another.]
- 7) On page 3 the correct device settings should already be set by default, but check it to make sure (Cyclone II – EP2C35F672C6).
- 8) Skip page 4 and hit “Finish” on page 5.

Step 2: Create a blank top-level Schematic file

1. Go to “File->New...” and select a new Block Diagram/Schematic File. Click “OK”:



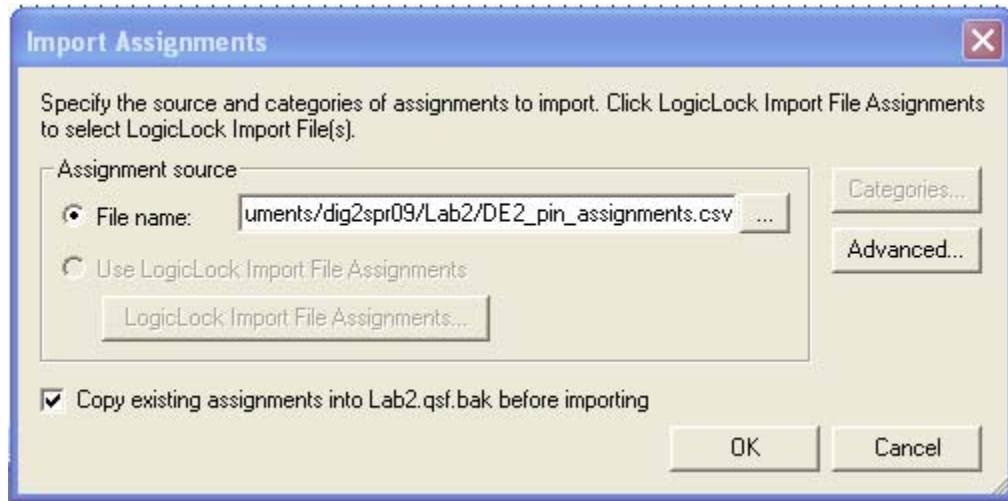
2. An editing window for the file "Block1.bdf" will pop up. Before you do anything else, go to "File->Save As..." and click on the "Save" button. By default the file name should be the same as your project name, and the check-box for "Add file to current project" should be checked:



Step 3: Import the Standard DE2 Pin Assignments

1. Go out to Windows if necessary and get a copy of the file *DE2_pin_assignments.csv* and put it somewhere Windows can find it. For this example I have copied it into my project directory, but anywhere will do.
2. You don't really need to keep *DE2_pin_assignments.csv* in your project directory because once you have performed this step to import the assignments, they are saved as part of the project database.
3. Go to "Assignments->Import Assignments..."

4. Click on “...” and find your copy of *DE2_pin_assignments.csv*:



5. Click on “OK” to import the assignments. In the messages frame you should see a green message “Info: Import completed...” if it worked properly.

Step 4: Think about your project hierarchy

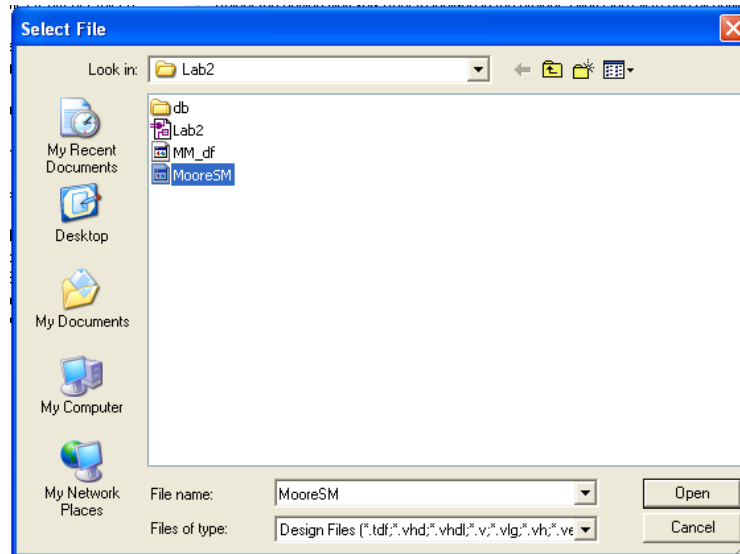
At this point what you do varies depending on the project you are developing. For the remaining labs in this course you are doing all your development and simulation in VHDL using the Mentor tools. Thus you will have one or more VHDL files that completely describe your design, and the only reason you are using Quartus is to map the I/O of the top-level block of your VHDL design to the physical devices of the DE2 board and then to download the design into the DE2’s FPGA.

The remaining steps assume your Quartus project is only used to download and physically verify a design that you have implemented at the top level as a single VHDL entity (and simulated elsewhere),

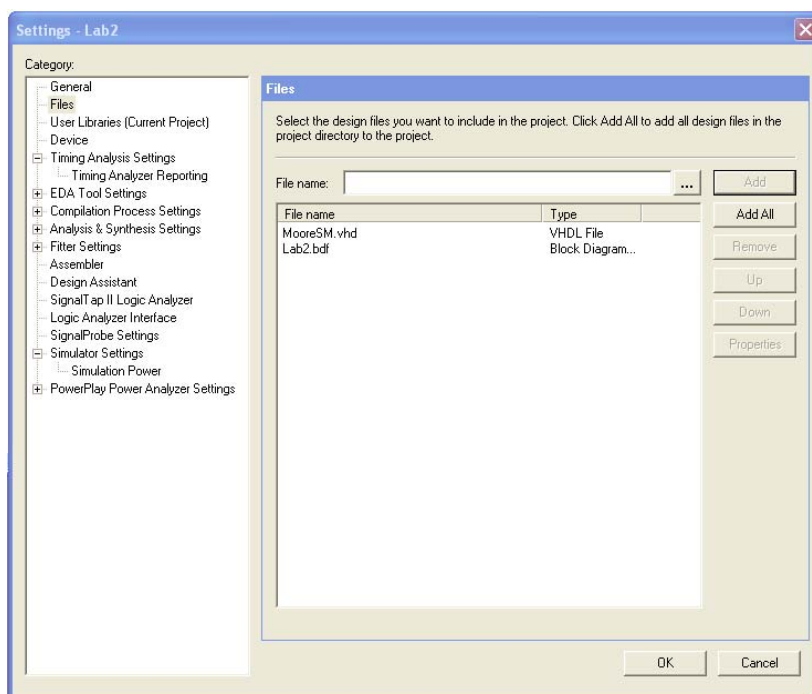
Step 5: Add your VHDL file(s) to the project

1. Copy your VHDL file(s) to your project directory.
2. Go to “Project->Add/Remove Files In Project...” and click on “...”.
3. This should open up a file dialog in your project directory.

- Click on a VHDL file you want to add to the project. If there is more than one, you can Ctrl-click on those after the first or drag-select them all. Once you have the file name(s) you want to add, click on “Open”.
- This shows a file named “MooreSM.vhdl” being added to the project:

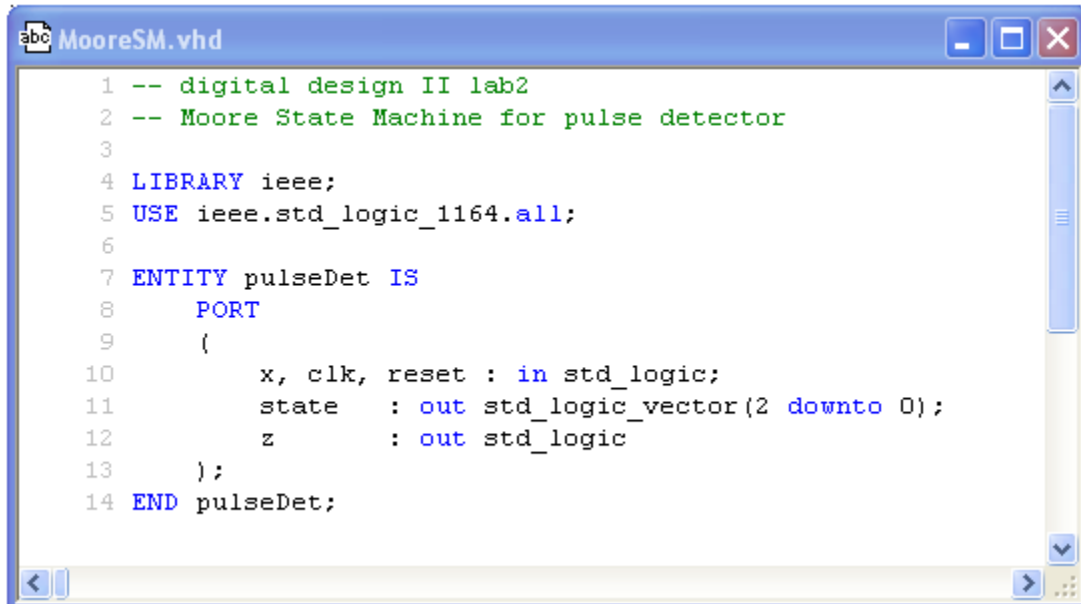


- When you click on “Open” it puts you back into the Settings dialog with the selected filename(s) filled in. You then have to click on “Add” to actually add the files to the project. When the files appear in the list in the lower right pane, click “OK”:



Step 6: Open your top-level VHDL file and generate a symbol

1. Go to “File->Open” and open the VHDL file that contains your top-level entity.
2. In this example, the VHDL file isn’t complete – it only has an entity defined and the architecture is missing. This was done intentionally to show that you don’t need anything except the definition of a top-level entity to proceed:



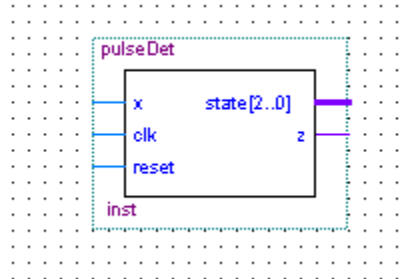
```
1 -- digital design II lab2
2 -- Moore State Machine for pulse detector
3
4 LIBRARY ieee;
5 USE ieee.std_logic_1164.all;
6
7 ENTITY pulseDet IS
8     PORT
9     (
10         x, clk, reset : in std_logic;
11         state      : out std_logic_vector(2 downto 0);
12         z          : out std_logic
13     );
14 END pulseDet;
```

3. Go to “File->Create/Update->Create Symbol Files for Current File”. This creates block symbols for the VHDL entities defined in the current file. These symbols can then be used in the block schematic editor. This step may take a few seconds and you will get a dialog box telling you when the operation completes successfully.

Step 7: Place an instance of your new entity in your schematic

1. Go back to your editing window for the block schematic file you created for the project.
2. Click on the Symbol Tool icon to place an instance. Now instead of just seeing a tree for the default libraries, you should also see a tree for your project. If you open up that tree you should see the name of your VHDL entity. When you click on it, you will see a block symbol for your VHDL entity. Click “OK” and you will have a symbol on your cursor that you can place in your schematic, just like you’ve done for entities from the libraries.

- This is the block symbol created from the VHDL entity shown in the last section (note the difference in vector/array syntax used for bus connections in the block schematic compared to the VHDL syntax):

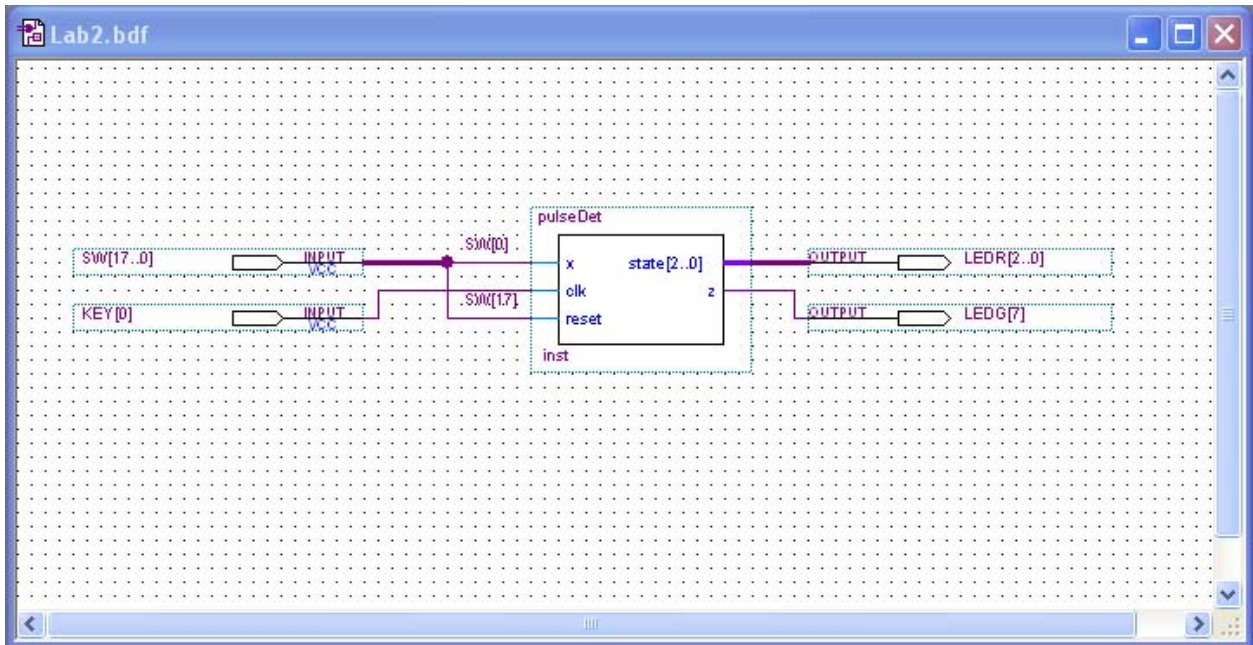


- Note that defining only the top-level entity in VHDL was sufficient to be able to generate a symbol. This allows you to do this part of the project “top-down” – you can define the I/O of your highest level block(s) and wire them up to your DE2 devices in your top-level schematic before you worry about exactly how you are going to implement the blocks. Note also though that you will get a compiler error if you attempt to compile the project without providing at least a stub for the architecture statement.
- Later on, you can reorganize and redefine your VHDL underneath this top-level entity as much as you want without needing to change your top-level block schematic so long as you don’t change the I/O definitions in the top-level entity statement.

Step 8: Place and name your I/O pins and wire them up

- Use the Symbol tool to place your I/O pins on the schematic. Double-click on a pin to give it a name. Name the pins using the standard names for the DE2 devices you wish to use.
- Note that you can use array syntax to refer to a range of DE2 devices, which – along with learning to use the bus tool and array names for your block I/O – can make drawing and understanding schematics a lot simpler.

3. This shows the I/O wired up for the pulse detector in Lab 2:



4. **IMPORTANT!** There is a bug/quirk in the QuartusII compiler that can cause problems if you use this method – if you make use of a discontinuous range of named pins (ex. use SW[17] and SW[0], but don't include SW[16..1]) then the project will compile successfully with only a warning, but the pins in question won't actually be connected.

This seems to have something to do with how the compiler forms pin “groups”. If you look in the pin assignment editor there is a “group” pane in the upper left quadrant. If pins sharing the same arrayed name cannot be placed in a single group because there are gaps in the range describing the set, then the proper group will not be formed and the names involved will be converted to non-array versions (ex. “SW[17]” will be converted to “SW17”). These names will no longer match up with the names assigned to pins by the DE2 import file and thus won't connect to anything.

It is OK to use just a single pin from a range (ex. use KEY[0], but no other KEY) and it is OK to use a range that doesn't start from zero (ex. LEDG[7..5]). Of course the easy way to avoid this problem altogether is to just avoid leaving gaps in the range of DE2 objects that you use (ex. choose to use KEY[1] and KEY[0] in your design rather than KEY[3] and KEY[0]). If you choose to use a discontinuous range of objects though, there are a couple ways you can satisfy the compiler. In the example above for Lab2 it shows how you can use an array name on one of your pins to cover any unused pins in the group, and only rip the wires you need from the resultant bus. The other option is to add extra pins (remembering to tie any unused outputs to vcc or gnd as needed).

Step 8: Compile the project and download to the DE2

After you've added an architecture statement to your VHDL so that it can do something, save your project, then compile it and download it to the DE2 board as usual. When you're testing your design, keep the top-level block schematic open to remind you how you mapped the DE2's devices to your entity's I/O.