USING LPM MODULES AND VECTOR INPUTS

August 30th, 2007
CSC343
Fall 2007
Prepared by: Steven Medina

PURPOSE

Our purpose here is to learn how to use LPM modules. An LPM module is a "black box" representation of a larger circuit. In other words, you can only see the inputs and outputs of the circuit design. What is inside the box would represent *how* it works. However, when using an LPM module, *how* it works is not important. All that matters is that it *does* work. In this lab, you will learn how to utilize these LMP modules to create the same multiplexer you created in the previous lab. You will then create slightly more complex designs using the idea of LMP modules.

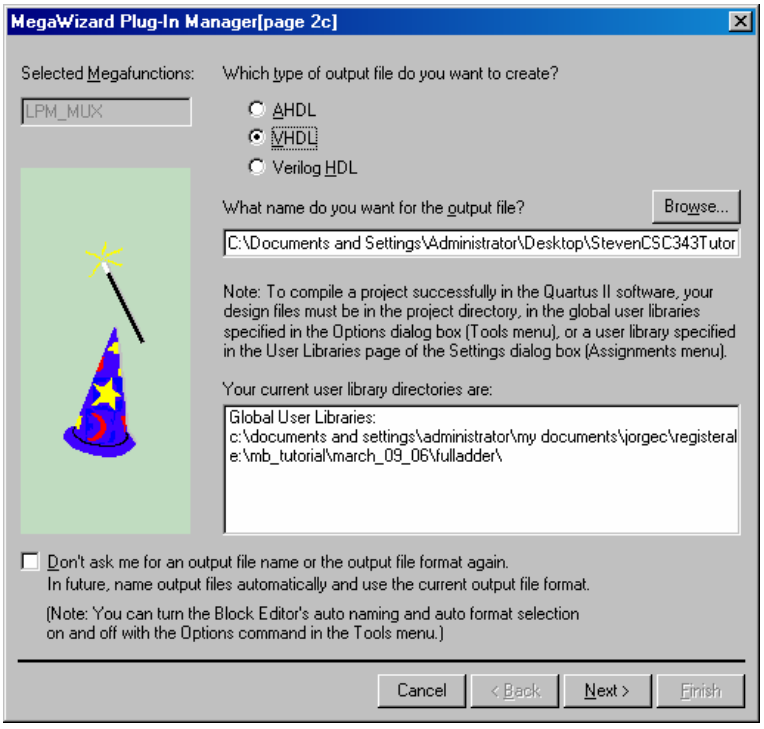
After learning LPM modules, you will learn how to create designs that use inputs and outputs that are larger than 1-bit long. This will include the use of a *counter.* A counter is simply a circuit block that counts. In our case, we will use a design that counts only upward and by 1's.

USING LPM MODULES

1) Create a new Quartus project file. If you forgot how, go back to "Project1Tutorial" to read how to do it. I recommend naming your project your name followed by "lpm_mux". For example, mine would be stevenlpm_mux. Open a new block diagram file (bdf).

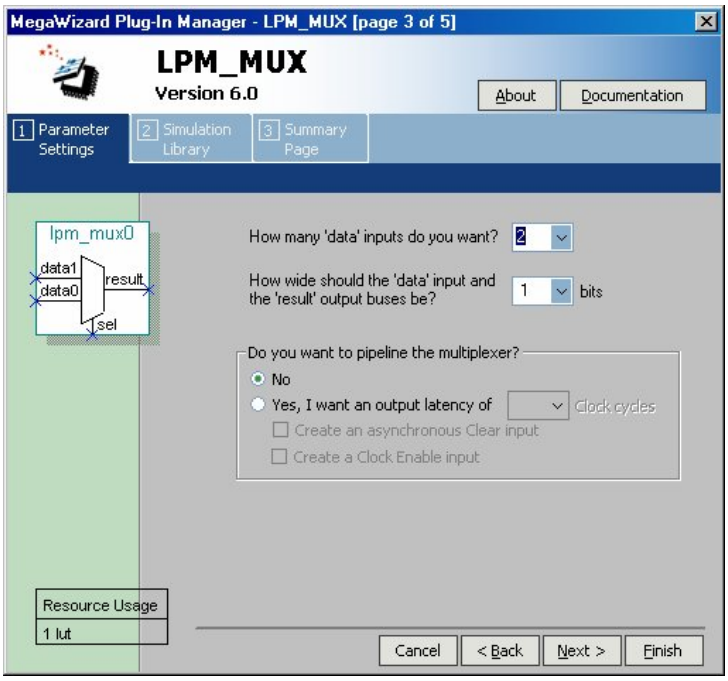
2) Click on "symbol tool". In the menu, click the plus sign next to the bottom of the two folders. Click on "megafunctions", then gates, then lpm_mux.

3) After clicking on lpm_mux, the following menu will appear.

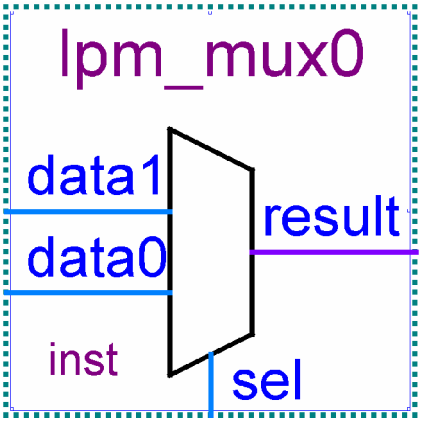


Where it asks “which type of output do you want to create?”, click “VHDL”. Then click next.

4) Another menu of options will appear which looks as follows:



Where it asks how many inputs you want, click 2. Where it asks how wide the data input and the result output buses should be, click 1. Do not pipeline the multiplexer. Click finish, and then finish again. You will now have a 2-to-1 multiplexer.



2-to-1 multiplexer created using an LPM module.

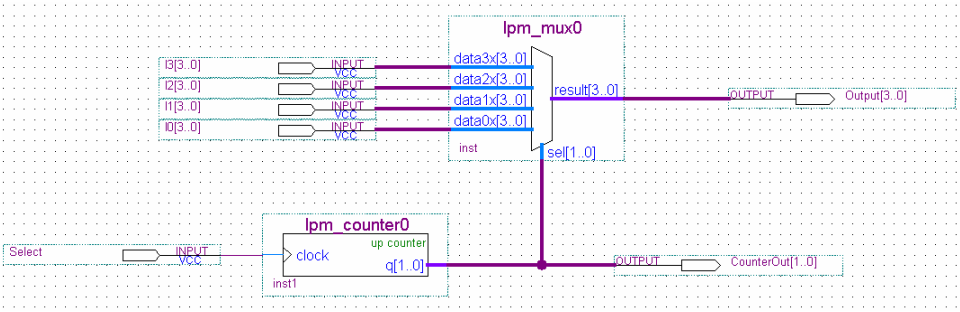
As you can see, we have a “black box” representation of a 2-to-1 multiplexer. This was much easier to create than connecting every gate that makes up a multiplexer.

5) After creating your multiplexer, add input and output pins to your design and compile. Open a vector waveform file.

Simulate your design just as you did in the “Project1Tutorial “. Compare your simulation to the simulation in “Project1Tutorial”. If done correctly, they should have the same logic values.

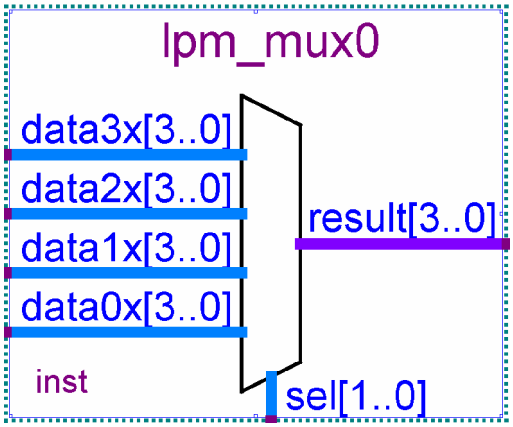
ORTHOGONAL BUS TOOL AND LPM COUNTER

1) Open a new project. Then open a new block diagram file. You are going to create the multiplexer design shown below.



This design includes a 4-to-1 multiplexer and a counter. Also, the inputs are more than one bit long. In fact, the inputs of the multiplexer are all four bits long.

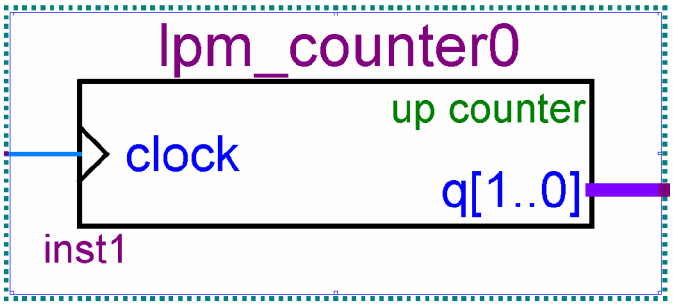
2) Click on the “symbol tool” box. Click on “lpm_mux”, just as you did in the first multiplexer in this tutorial. In the first menu, select VHDL like normal. Click “next”. In the next menu, set the number of data inputs to 4. Also, where it asks how wide the data input and result output should be, select 4. Click “finish” twice. You should have a multiplexer that looks as follows.



Place your multiplexer on your block diagram.

3) Next, we will create an LPM counter. Click on the “symbol tool” box. Click on megafunctions, then arithmetic. You should see an option for “lpm_counter”. Click on this option to bring up the LPM wizard for this counter.

4) In the first screen, click on VHDL. Then click “next” to advance to the next screen. Where it asks “How wide should the ‘q’ output be”, select 2 bits. Make sure the counter direction is set to “up only”. Click finish twice. Following the diagram on the beginning of page 3, place your new LPM counter on your diagram to make it look similar to the diagram on page 3. Your LPM counter should look like the image below.



5) Place an input pin going into the “clock” input of your counter. Connect the input pin to the counter.

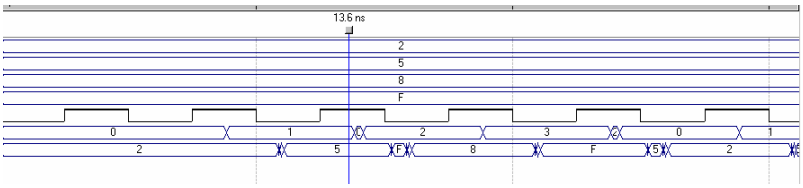
6) The output of your counter is 2 bits long, as well as the selector input for the multiplexer. Therefore, you cannot use the “Orthogonal Node Tool” to connect the counter to the multiplexer like we usually do. We will use the tool right under it, called the “Orthogonal Bus Tool. You will notice that the line connecting the output of the counter to the 2-bit input of the multiplexer is thicker than normal. This is the connection used anytime you are connecting two blocks together that have more than one bit in either their input or their output. The normal naming convention for an input that has more than one bit is a *vector*.

7) Connect input pins in each of the four inputs of the multiplexer. Since the inputs of the multiplexer are 4-bits long, you will use the orthogonal bus tool to make your connections. This also goes for the output pin that you connect from the output of the multiplexer.

8) When renaming the input and output pins to the multiplexer, you have to use a vector notation to do so. This is because they are 4-bits long. For example, I would normally name the input pins to this multiplexer I0, I1, I2, and I3. However, in this case, we must name them I0[3..0], I1[3..0], I2[3..0], and I3[3..0]. The numbers in parenthesis “[3..0]” represent a 4-bit vector whose bits are bit 3, bit 2, bit 1, and bit 0. This naming convention MUST be used for all input and output pins that are connected to vectors.

9) Change the name of your output pin to meet the naming convention used for vector outputs (For example, output[3..0]).

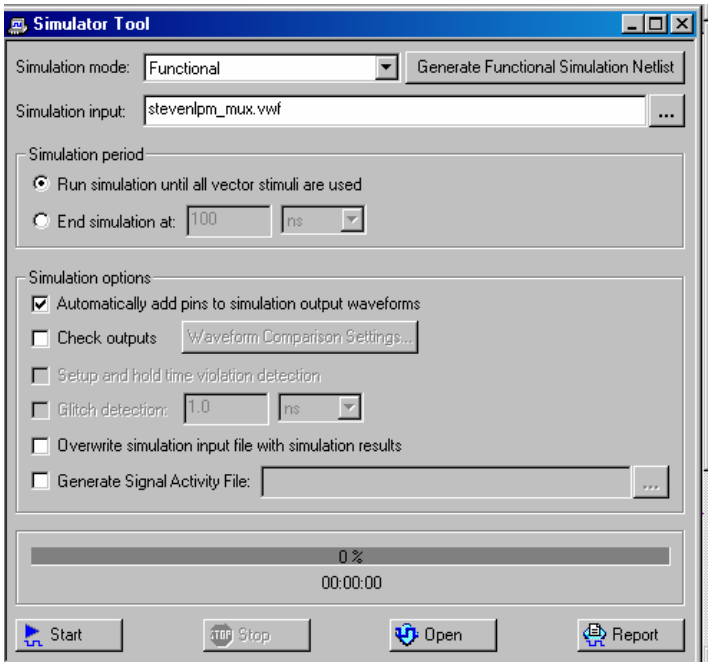
10) Simulate your design with a vector waveform file. Make sure to give initial values to your four inputs. Set the input into the selector as a clock (right click the selector input, click “value”, and then click “clock”. This was also done in the first tutorial). Your simulation should look similar to the image below



FUNCTIONAL SIMULATION

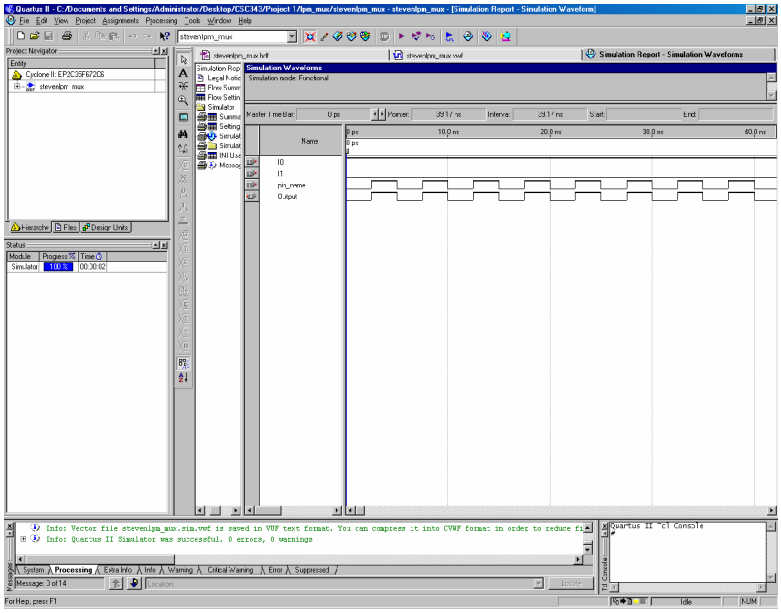
Functional simulations are simulations done in vector waveform files, but with NO delays. They are much easier to read when trying to determine how a circuit design works. Open

1) Open a vector waveform file. Click on Processing=>Simulator tool. You will see a box that looks as follows.



Under the “Simulation Mode” dropdown menu, change the mode from “timing” to “functional”. Then, click “Generate Functional Simulation Netlist”.

2) Click “Start” on the lower left-hand corner. Usually this will simulate your design. If the simulation does not show up after this, simply close the box, and click Processing=>Start Simulation. This will simulate your design with no timing delays. A picture of this is shown below.



As you can see, it is easier to analyze the logic when using a “functional” simulation. However, remember that in reality, there will always be some delay. Only the “timing” simulator mode will show the actual delays.