

Lab #9

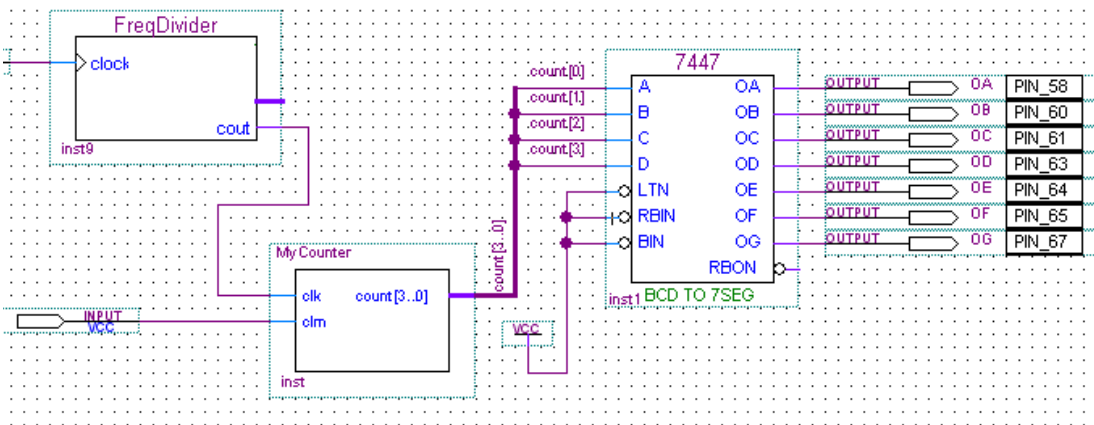
PRE-LAB

Using VHDL, design and simulate a synchronous counter that counts 0 to 9 like the counter you built for a previous lab (do not include the output z, however). Your counter should include an asynchronous clear, and produce 4 output bits (of std_logic_vector type) which are the number in BCD format. Alternatively, you can have your VHDL output the 7-bit SSD format.

Hint: Look at example 6.7 in Pedroni for ideas. The conv_std_logic_vector (integer, # bits) function contained in the ieee.std_logic_arith.all library may be useful for you.

LAB

Your goal is now to use your new counter in a new project that incorporates a slower clock (with a period of a second) and outputs the digit to the seven segment display. The general block diagram for the project you will build is shown below.

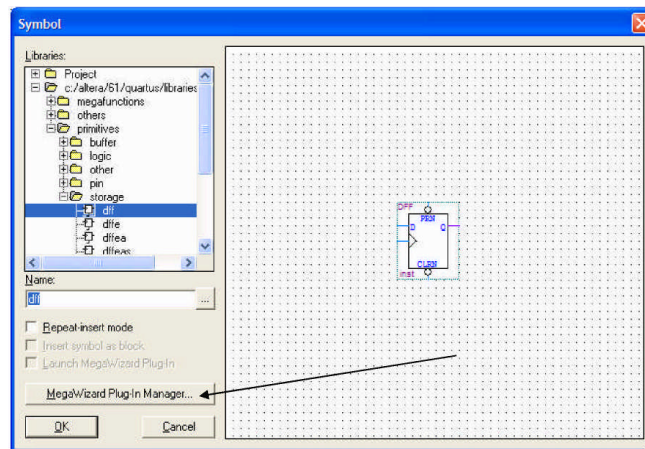


To build this system, you need to create a symbol for your VHDL counter so that you can incorporate it into the new project (which you will, of course, create in a new folder). You also need to implement the “FreqDivider” block which will slow the clock down from 50 MHz to 1 Hz. See the instructions that follow under “*Slowing down the clock*”. You will also need to use bus lines with this project, as is shown with the thick lines between the “MyCounter” block and “7447” block in the example above. More information on using bus lines is also included below. Once you have it all working on your Altera board, demonstrate it to the instructor.

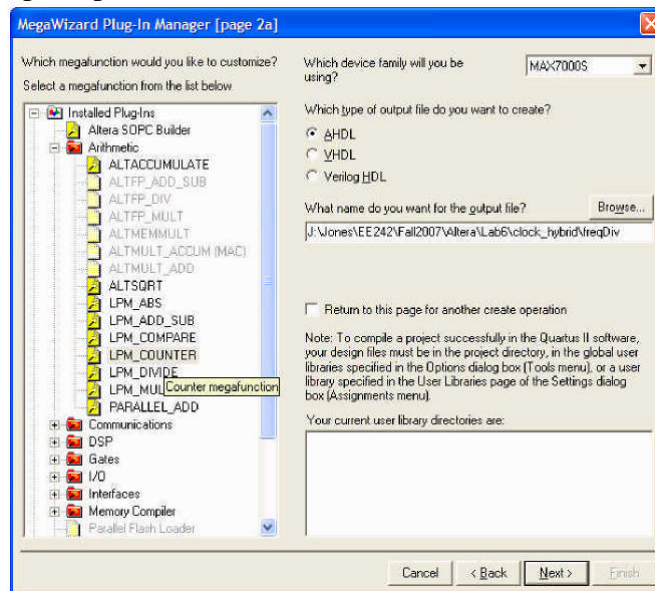
Slowing down the clock

The global clock (pin N2 on your proto board) is around 50 MHz. If you were to use this clock directly to drive the CLK in the VHDL design, then the state would change 50 million times per second. So you'll need to create a frequency divider. There are many ways to do this (feel free to do this in VHDL if you are so inclined). What we are going to do is feed the global clock into an intermediate counter that will count from 0 to a really big number (you figure it out). As this intermediate counter then returns to 0, it will set its carry out to "1". You can use this carry-out to then trigger your 0 to 9 counter.

To create the frequency divider, while in a schematic, double-click the background, and this familiar dialog box appears:



Click on the "MegaWizard Plug-in Manager". From there, "Create a new ...". Choose a VHDL output file. From there, open up the "Arithmetic" area from the menu on the left:



Select the "LPM_COUNTER", and fill in an output file-name like 'FreqDivider' in the associated box.

Next you'll have to answer some questions. Specifically, you should use an up counter that has enough bits for you to count to. You'll also have to choose a modulus count with a number for it to count to that will cause the last bit (add a carry out) to change state at whatever rate you want. When you're finished, Quartus will create your FreqDivider box, which will look something like the one in the example above.

Using a Bus

Those thicker lines coming out of your VHDL counter and coming out of the FreqDivider block are buses. A bus, in our case, is a collection of wires. Note that the buses are named, and that the convention of the [] square brackets indicates bit order. For example count[3..0] represents 4 wires where count[3] is the MSB. You can drag a bus out using the Orthogonal Bus Tool (it looks like the regular wire tool, but with a thicker line). When doing this, you should name the bus (highlight the bus, right click on “properties” and you can give it a name). You can use the same name as the name you are pulling it from.

You need to pull off individual (small) wires of a bus. Name them according to the wire you want to access. To do this, again, highlight the wire, right click on “properties” and give it a name. For example, count[0] is the LSB. Here is a close up view of the bus portion of the example:

