EECS 355 – Assignment 3
Sequential Divider

Objective:
In this lab, you will design a sequential divider that operates on *unsigned* numbers. The divider will operate under the control of a start button.

Background Information:
The theory behind sequential operation is that common resources can be reused and given different computational tasks to perform during each clock cycle. A sequential divider reuses one comparator unit to perform all division steps through multiple successive clock cycles.

In this lab, you will instantiate only one comparator, and you will change the values being fed in as operands during each cycle.

Lab Description:

I. Create new sequential architecture:

1) Define a new architecture called "behavioral_sequential" inside the divider entity. You can add an architecture directly to the end of the entity after the previous architecture. The clk signal is use to synchronize the computation. At each rising edge of the clock, the divider should calculate one stage of the long division. The result is then fed back to the inputs of the comparator for the computation of the next stage. Please reuse the comparator component from the previous lab in your sequential architecture design (you may need to make changes to your comparator to suit your design, but the core functionality should remain the same). As before, the start button is used to signal that the inputs are ready and that computation should begin.

II. Testing your design:

1) Simulate your design using both the 16 and 32-bit versions of the test data from lab 3A. One of the advantages to having the divider defined with two architectures is that you do not need to rewrite the testbench in ModelSim. You do need to add the line:

```
for all : divider use entity WORK.divider (behavioral_sequential);
```

Add this line right after you declare the divider component in the testbench. This will instruct ModelSim to use the architecture labeled behavioral_sequential when instantiating any dividers. If you have more than one architecture in your design, it is best to explicitly state which architecture is to be used.

Additionally, you will need to modify the testbench to generate and handle the new clock signal that must be fed into the sequential design.

III. Top-level design:

1) Compile the project in Quartus II. Assign the pins using the assignment editor. Refer to the table at http://www.terasic.com.tw/attachment/archive/30/DE2_Pin_Table.pdf for pin locations. **Only begin assigning pins to your ports after you have successfully synthesized your design.** Note that in this lab, the top-level entity should remain the same as in the previous lab, except for the clk signal, which must now be assigned to the clock pin on the DE2.

2) Once your design has been compiled, it can be loaded onto the FPGA using the programming utility in Quartus II. To open the programmer, click on the  button in the toolbar, or double click "Program Device (Open Programmer) in the "Tasks" pane. Make sure that the "USB-Blaster" is selected next to the "Hardware Setup" button in the top left of the programming window. Click on "Start" to program the device.