ECSE-323
Digital System Design

Lab #3 – *Sequential Circuit Design* Fall 2008

Introduction

In this lab you will learn how to use the Altera Quartus II FPGA design software to implement sequential logic circuits.

Table of Contents

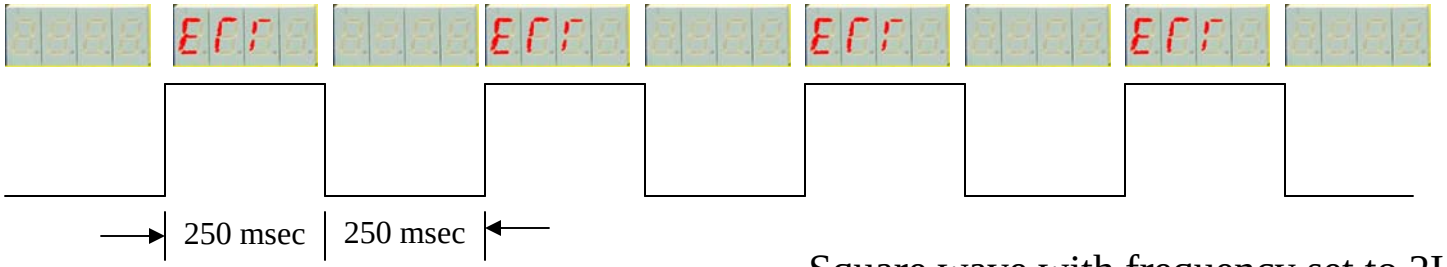
This lab consists of the following stages:

1. VHDL design of an “Error” flasher circuit
2. Timing simulation of the flasher circuit
3. VHDL design of a testbed for the flasher circuit
4. Testing of the flasher circuit using the testbed on the DE1 board
5. VHDL design of an “Error” scroller circuit
6. Timing simulation of the scroller circuit
7. VHDL design of a testbed for the scroller circuit
8. Testing of the scroller circuit using the testbed on the DE1 board
9. Writeup of the lab report

1. Design of an "Error" Flasher Circuit .

For the course project we would like to build a circuit that will flash the characters "Err " on the LED display. Flashing should occur twice per second. The duty cycle (percentage of time the display is on) should be 50%.

To begin the design of the flasher circuit, you will need to design a timer circuit that can generate a square-wave waveform with frequency selectable among 1,2,4,8 Hz.



Square wave with frequency set to 2Hz.

The first order of business in designing the Timer is to design a clock frequency divider.

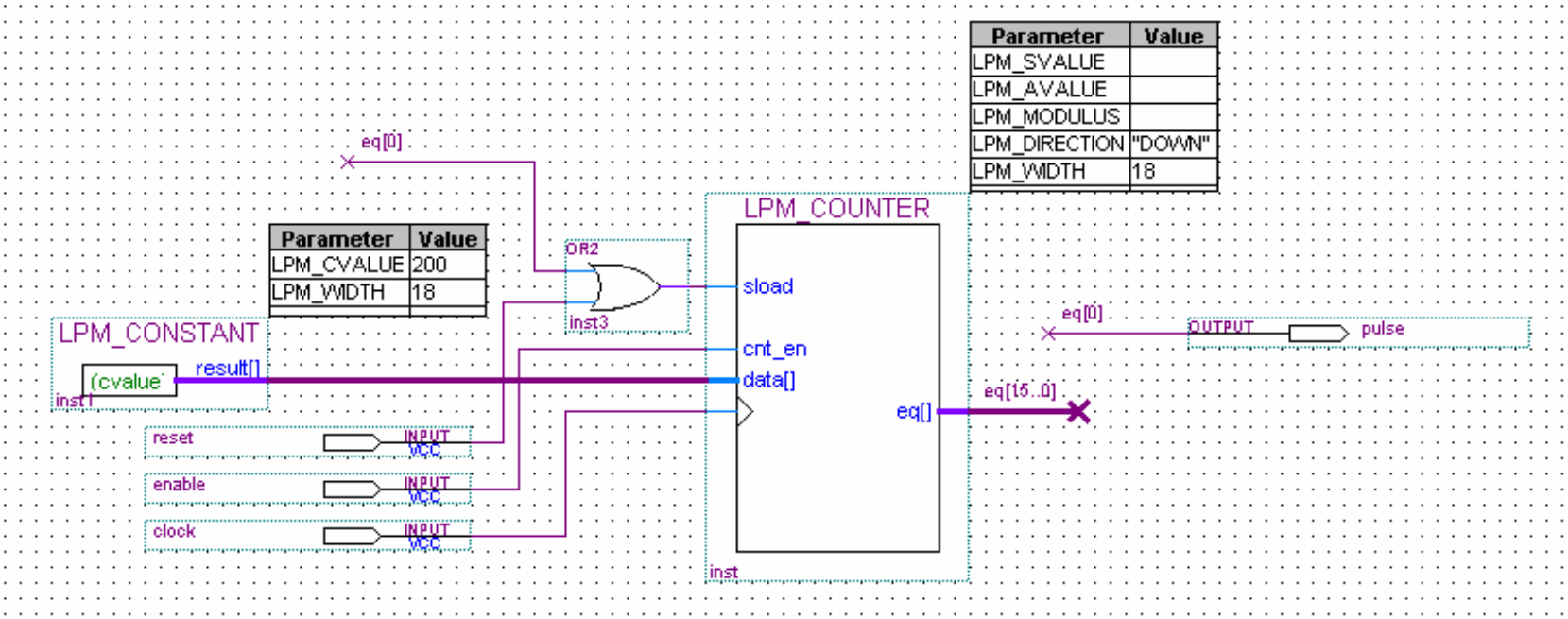
The Altera DE1 development board contains two different clock generators which run at 27 and 50 MHz. These are both much faster than 2Hz!

So you need to derive a time pulse which is slower than the Altera board's clock. This is done with a ***Frequency Divider*** circuit.

There is a right way, and a wrong way, to make a Frequency Divider. Before taking the DSD course your approach might be to make a ripple-counter, in which the high-speed clock is used to clock a counter whose LSB output is fed to the clock input of another counter, and so on in a chain until you are left with a low enough frequency. But this is a bad approach.

You want to run ALL of your sequential units with the same high speed clock. This produces a fully synchronous system.

The RIGHT way to make a frequency divider…



In this circuit, a counter counts down to zero. Once it reaches zero, it loads in a constant value, and then counts down from this value. The output of the counter is a pulse that is high when the count value is equal to zero (the ***eq*** output of the ***lpm_counter*** module is provided for such a purpose).

This will be high for just one clock cycle at a time.

The output should be used as an ***enable*** control for other counters, and **not** used as clock inputs for these counters!

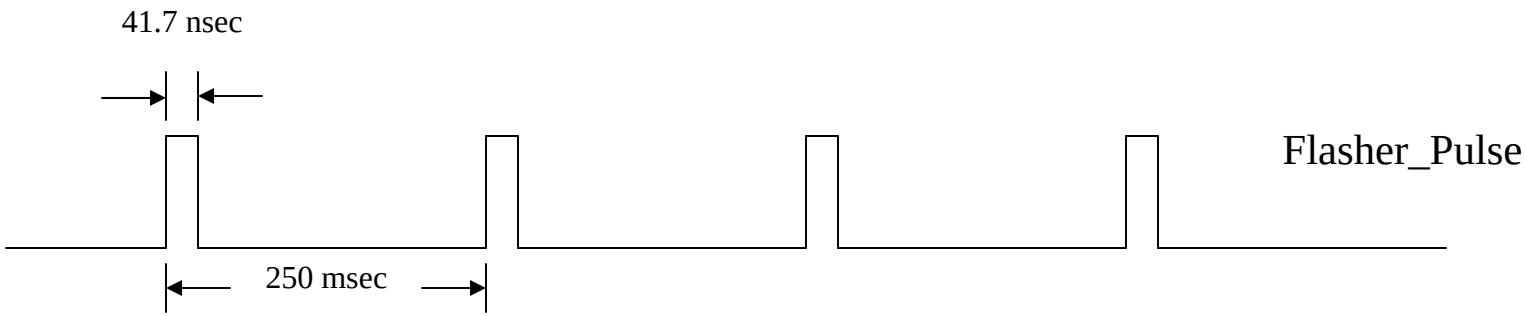
The period of the timer is equal to (N+1)*Tc, where N is the value loaded and Tc is the clock period.

Using the approach depicted on the previous page, design a circuit that will produce a repetitive pulse with a period of either **1/16, 1/8, ¼, or ½ second** as set by the frequency select input. This will produce the basic time interval for the flasher.

*Name the output of the timer **Flasher_Pulse**. The circuit should have 3 inputs – a clock input, a reset input, and a 2-bit frequency select input.*

Use VHDL to describe the circuit. Call the circuit g***NN_Flasher_Timer*.**

(not to scale!)



You will have to compute the appropriate constant value to load in to give the required amount of frequency division *(hint: it is quite large!)* and determine how many bits the counter should have.
Don't be afraid of large numbers of bits - the Cyclone chips on the Altera board have a lot of flipflops!

Assume that you will use the ***27 MHz*** clock on the Altera board, and take this frequency into account when computing the division factor for your timer.

The g***NN_Flasher_Timer*** circuit just produces a one-clock-cycle wide (i.e. 37nsec wide) pulse every 250msec. But we want a 50% duty cycle for our flashed message.

The way to do this is to implement a ***toggle*** flipflop which is ***enabled*** by the ***Flasher_Pulse*** signal, and is ***clocked by the 27MHz clock***. Each time the toggle flipflop is enabled, a clock transition will cause the output of the flipflop to change state. Since the Flasher_Pulse is only high for one clock cycle there will only be one clock transition during each flasher pulse.

Thus, when the frequency select input is set to 2Hz, the Flasher_Pulse will be running at a rate of 4Hz and the flipflop will change state once every 250msec, in spite of the fact it is being clocked 27 million times a second. The result is that the flipflop output will be high for 250msec then low for 250msec and so on, giving a 2Hz 50% duty cycle waveform.

Next, add to the toggle flipflop two 2:1 7-bit wide ***bus multiplexers***, whose select lines are each connected to the output of the flipflop. The "0" inputs of both multiplexers should be connected to a constant "0000000" code, while the "1" input of one multiplexer should be connected to the code for the "E" symbol (i.e. code=14="0001110") and the one input of the other multiplexer should be connected to the code for the "r" symbol (i.e. code=29="0011101").

Write a VHDL description of the toggle flipflop/multiplexer combination circuit. Look for and use an lpm module to implement the toggle flipflop. The VHDL design entity should have three inputs – the clock input, the enable input, and the 2-bit frequency select. It should have two outputs, which are the outputs of the multiplexers, representing the two flashing codes. Call the circuit ***gNN_Flasher_Mux***.

Compile the circuit.

Next, write a VHDL description of a circuit that combines the ***gNN_Flasher_Timer*** and ***gNN_Flasher_Mux*** circuits into one – called ***gNN_Flasher***.

The VHDL design entity should have three inputs – the clock input, the reset input, and the frequency select input. It should have two outputs, which are the outputs of the multiplexers, representing the two flashing codes.

Show your VHDL description of the circuit to the TA. Compile the circuit.

2. Timing Simulation of the Flasher Circuit .

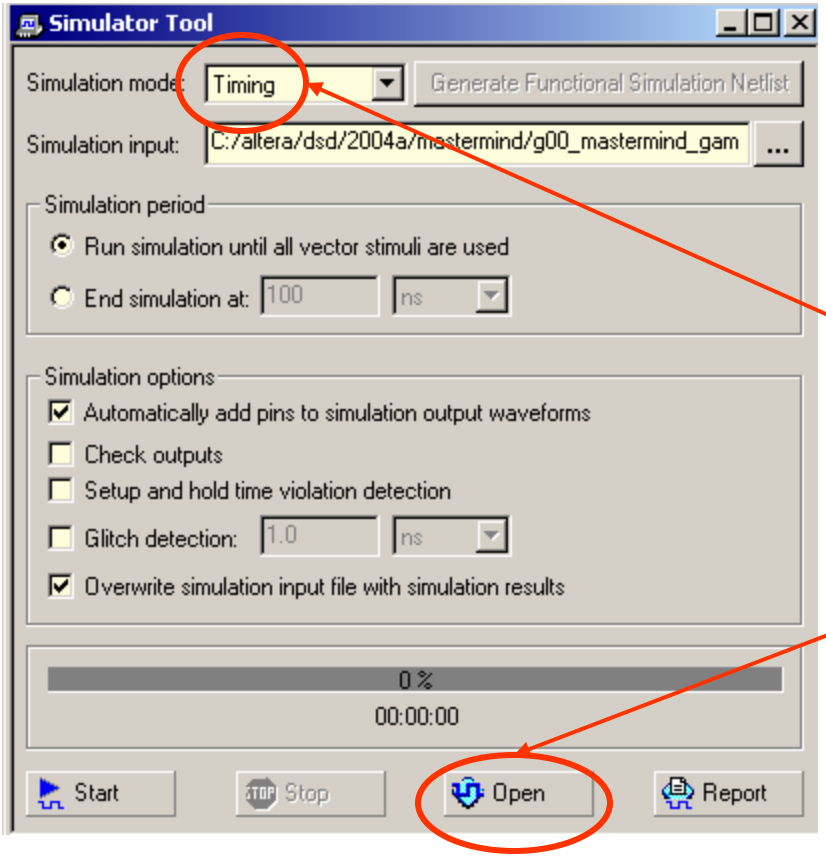
The next step is to simulate the ***gNN_Flasher*** circuit.

You should always be aware of how long your simulations will run.

You can get a good handle on the computational load for the simulation by counting the number of clock cycles that will occur through the length of the simulation. If you use a 27 MHz clock (37 nsec period) and run the simulation for at least 2 microseconds (so that you can see a few timer output pulses) then you will run through at least 54 million clock cycles. Running a simulation with this many clock cycles will take a long time!

So, ***for simulation purposes only***, you can reduce the division factor of your pulse generator, say to produce a period of 250microseconds rather than 250msec. This will reduce the simulation time by a factor of 1000.

You will need to change this back before downloading to the hardware!



Recompile the project to incorporate the change to the frequency division factor. Then proceed to simulation.

Select the Simulator Tool item from the Tools menu. A window like the one at the left will appear.

Select "Timing" as the simulation mode.

Click on "Open" to bring up the Waveform Editor.

Click on "Open" to edit the waveforms. Draw the clock waveform (using the Clock tool in the menu along the left hand side of the waveform editor), and set its period to 37 nsec. Set the END TIME (in the Edit menu) to 2000 usec.

Then, click on "Start" in the simulator window to run the simulation. Run the simulation for each of the 4 different frequency values.

Show the results of the simulation to your TA.

Note the presence of any *glitches* (pulses that shouldn't be there). These glitches are only a problem if they persist across clock transitions. Any glitches that occur at times other than the clock transitions are harmless. There might not be any glitches for this circuit.

What are the propagation delays of the output from the positive edge of the clock signal? Show how you found this value from the simulation to your TA.

TIME CHECK

You should be this far at the end of your *first* 2-hour lab period!

3. Design of the Flasher Testbed .

You will now combine the Flasher circuit and the 7-segment decoder circuit designed in the previous lab into a ***test-bed*** for your flasher circuit.

A test-bed for a module is a circuit that present inputs to a module and displays outputs of the module in a way that lets the tester evaluate the performance of the module.

Combine the circuits in VHDL using component instantiation statements into a new design entity which you can call ***gNN_Flasher_Testbed***. You should include four instances of the LED decoder circuit from the previous lab, and connect the outputs of these to the appropriate pins for the 7-segment LEDs on the DE1 board (as you did in the previous lab, only now you will be connecting to all 4 LED displays). Connect the left-most 7-segment decoder input to the "E" output of the flasher and the inputs of the middle two 7-segment decoders to the "r" output of the flasher. The rightmost 7-segment decoder should have its input hardwired to the code "127" (which should produce a blank). You should use the ripple blanking feature of the 7-segment decoders.

Connect the two frequency select bits to two of the switches on the DE1 board.

Viewing the Compilation Report

Compile the testbed circuit. Look at the ***Flow Summary*** section of the Compilation Report and note the FPGA resource utilization (i.e. how many logic cells were used?). You will include this information in your report.

Also take a look at the ***floorplan*** (under the ***Tools/Chip Planner***) and observe how the circuit has been fit into the FPGA. Your circuit will only take up a very small part of the chip so you will have to zoom in. Find a used logic cell and double-click on it. A window will popup showing how that logic cell is being used. Read over the datasheet for the Cyclone chip (available on WebCT) to understand the architecture of the device.

Finally, look at the ***Summary*** entry in the ***Timing Analyzer*** section of the compilation report. Take note of the path with the largest propagation delay (worst case tco). Include this in your report. It is an important number, as it determines the maximum speed of any circuit that uses your design.

Show the compilation report and Timing Analyzer summary to the TA.

4. Testing the Error Flasher on the Altera DE1 Board .

Once you compiled the testbed, it is time to map it onto the target hardware, in this case the Cyclone chip on the Altera DE1 board.

As you did in lab #2, make sure you have correctly mapped the output ports of your gNN_Flasher module to the appropriate pins on the Cyclone chip (those that are hardwired to the segments of the four 7-segment LED displays on the DE1 board) and the frequency select inputs to the switches on the DE1 board.

After you do the pin mappings recompile the circuit.

Once the circuit is recompiled, transfer the design to the DE1 board using the programming procedure you learned in lab #2. If everything is working properly you should see a flashing “Err” on the LED display. The flashing rate should be selectable by the two switches connected to the frequency select inputs.

Proudly show your flashing display to the TA.

TIME CHECK

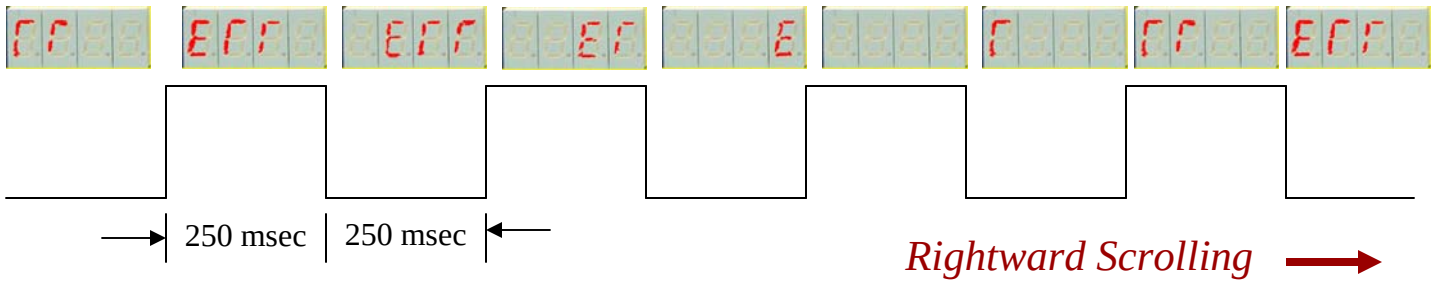
You should be this far at the end of your *second* 2-hour lab period!

5. Design of an "Error" Scroller Circuit .

In the second part of the lab you will make an scrolling error message.

You should use the same approach as you did for the error flasher adding whatever circuits or lpm modules you deem necessary. Use VHDL with component instantiation statements throughout your implementation.

The scrolling behaviour should be as shown below. The direction should be adjustable with a 1-bit mode input (which you will connect to a switch on the DE1 board) to be either leftward scroll or rightward scroll. The scroll rate should be set by your *gNN_Flasher_Timer* circuit.



5. Design of an "Error" Scroller Circuit (continued) .

Call your scrolling circuit ***gNN_Scroller***.

This circuit should have 4 inputs – a clock, a reset, a 1-bit scroll direction input, and a 2-bit frequency input. It should have four 7-bit outputs, one for each of the destination 7-segment LEDs. These should output the display codes, not the 7-segment data. You will use the 7-segment decoder module separately to map the display codes to the 7-segment data.

Do the following steps:

1. Write the VHDL description for the ***gNN_Scroller*** circuit (including any new modules that you may have to design). You should try to re-use previously designed modules as much as possible. Show your VHDL description to the TA.

2. Do a timing simulation of the ***gNN_Scroller*** circuit. Show your simulation results to the TA.

TIME CHECK

You should be this far at the end of the *third* 2-hour lab period!

6. Testing the Error Scroller on the Altera DE1 Board .

As you did for the Error Flasher, you should make a testbed for the Error Scroller circuit. Do the following steps:

1. Write a VHDL description for the Error Scroller Testbed.
2. Compile the circuit then assign the pins to the appropriate locations
3. Recompile the design and then download to the DE1 board.
4. Proudly show your scrolling display to the TA, demonstrating scrolling leftward and rightward, and the four different scrolling rates it can do.

TIME CHECK

You should be this far (i.e. have completed the lab) at the end of your *final* 2-hour lab period!

9. Writeup of the Lab Report .

Write up two short reports, for the ***gNN_Flasher*** and ***gNN_Scroller*** circuits that you designed in this lab (you do not have to write a report for the test bed circuits, but refer to these in the testing section of the reports). This report should also describe the design and testing of the new components used in the Flasher and Scroller circuits.

Each report must include the following items:

- A header with the group number, the names and student numbers of each group member.
- A title, giving the name and function of the circuit.
- A description of the circuit's function, listing its inputs and outputs. Provide a pinout or symbol diagram.
- A block diagram of the circuit (including details of its component parts).
- A discussion of how the circuit was tested, giving details of the testbed and showing representative simulation plots.
- A summary of the timing performance of the circuit, giving the timing analysis and the simulated propagation delays.
- A summary of the FPGA resource utilization (from the Compilation Report's Flow Summary).

The ***lab reports***, and all associated ***design files*** (e.g. .vhd, .bdf, .wvf) must be submitted, as an assignment to the WebCT site. Only one submission need be made per group (both students will receive the same grade!).

Combine all of the files that you are submitting into one *zip* file, and name the zip file gNN_LAB_3.zip (where NN is your group number).

The reports are due one week after the last day of the lab period, or Friday November 7.

Grade Sheet for Lab #3

Fall 2008.

Group Number: _____.

Group Member Name: _____. Student Number: _____.

Group Member Name: _____. Student Number: _____.

Marks			
	1.	VHDL for the Error Flasher circuit	_____.
	2.	Simulation of the Error Flasher circuit	_____.
	3.	Compilation report and Timing Analysis of the Error Flasher	_____.
	4.	Demonstration of the Error Flasher on the DE1 board	_____.
	5.	VHDL for the Error Scroller circuit	_____.
	6.	Simulation of the Error Scroller circuit	_____.
	7.	Demonstration of the Error Scroller on the DE1 board	_____.
			TA Signatures

Each part should be demonstrated to one of the TAs who will then give a grade and sign the grade sheet. Grades for each part will be either 0, 1, or 2. A mark of 2 will be given if everything is done correctly. A grade of 1 will be given if there are significant problems, but an attempt was made. A grade of 0 will be given for parts that were not done at all, or for which there is no TA signature.