

# ECSE-323

# Digital System Design

**Lab #3** – *FPGAs and Sequential Circuit Design*

Prof. W. Gross and Prof. J. Clark

Winter 2016

# Introduction

---

In this lab you will learn how to use the Altera Quartus II FPGA design software to implement sequential logic circuits described in VHDL. You will also learn how to do circuit synthesis and perform static timing analysis of the synthesized circuit using the TimeQuest timing analyzer.

You will design circuits needed to implement the Enigma machine.

This lab will introduce the use of the Altera DE1 University Board, which is a hardware prototyping system.

***NOTE: This is a 3-week lab.***

# Learning Outcomes

*After completing this lab you should know how to:*

- Design synchronous sequential circuits using VHDL process blocks
- Learn how to use Altera LPM modules
- Synthesize hardware from VHDL descriptions for FPGA target hardware
- Perform static timing analysis of circuits mapped to hardware
- Program and run circuits on the Altera DE1 FPGA board

# Table of Contents

---

*This lab consists of the following stages:*

1. Design and simulation of a 0-to-25 Counter Circuit
2. Obtain the Altera Design Laboratory Kit
3. Testing the 7-Segment LED Decoder on the Altera Board
4. Design of a Test-Bed
5. Using the TimeQuest Timing Analyzer
6. Running the Testbed on the Altera Board
7. Writeup of the Lab Report

# 1. Design and simulation of a 0 to 25 Counter Circuit

The Engima machine consists of 4 rotating wheels, with 26 positions each. So, the obvious approach is to implement the rotation of a single wheel with a 5-bit counter that counts up from 0 to 25 and repeats.

Your job in this lab is to design such a counter. You will need to use a VHDL process block to design this sequential circuit.

Make sure to include *count\_enable* and *reset* inputs in your design, as they will come in handy later.

**When you have finished the VHDL show it to the TA.**



Write a VHDL testbench to test your 0-to-25 counter circuit. You should define a periodic signal with a period of 100 ns to serve as the clock.

Simulate the counter in Modelsim for at least 5  $\mu$ s.

In addition to testing the counter sequence, make sure that your testbench tests the reset function, as well as the count\_enable function.

**Show the results of the simulation to the TA.**

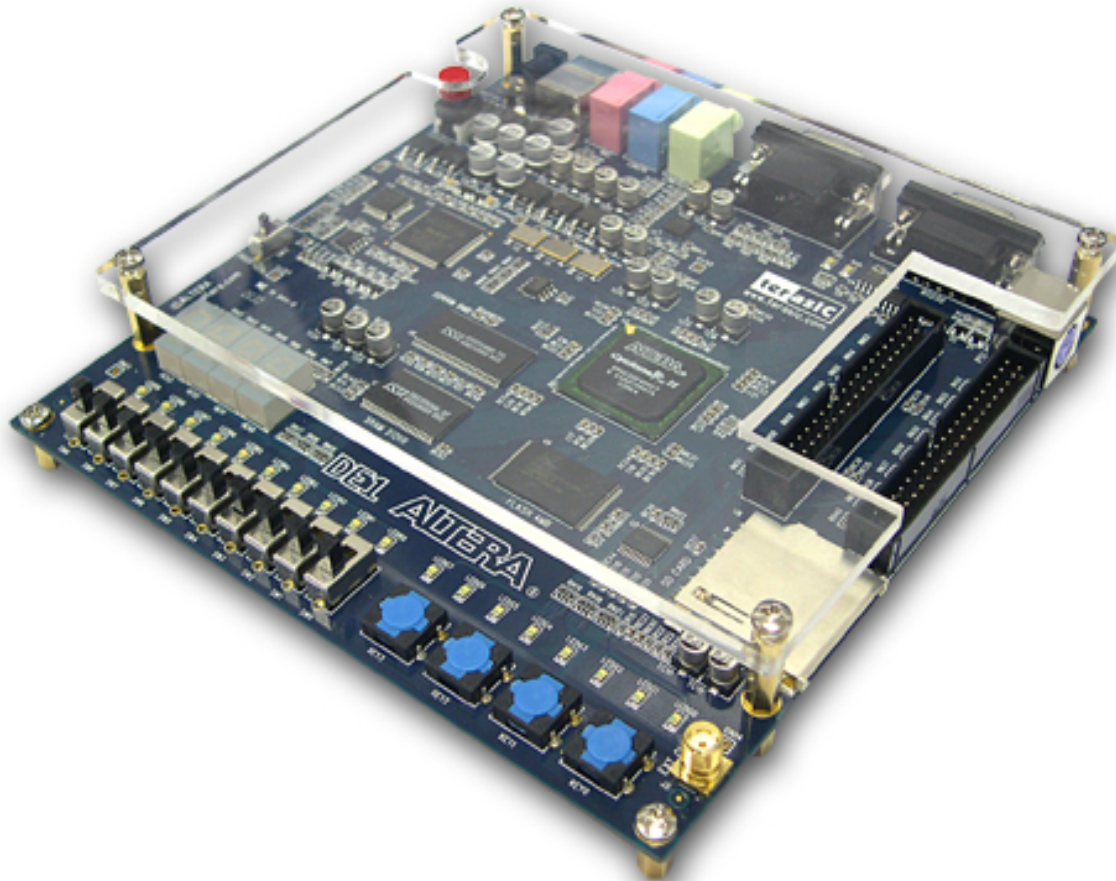


## **2. Obtain the Altera Design Laboratory Kit**

For the remainder of the lab experiments, groups will be using the Altera DE1 Development and Education Board. This package includes:

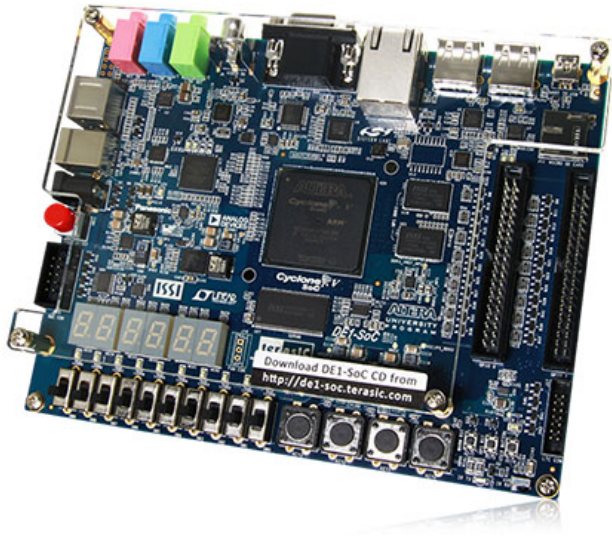
- Altera DE1 Board with a *Cyclone II EP2C20F484C7* FPGA
- Altera DE1 CD-ROM - Version 0.5 with documentation
- Altera Quartus II DVD - Version 7.0
- 1 Power Supply Adapter DC 7.5V/0.8A (US wall plug)
- 1 USB Cable
- 6 Silicon Footstands
- 2 Cables (black- and red-colored)
- 2 PIN Headers, 1P1N

# The Altera DE1 Development and Education Board



Some of the lab groups will be given a newer version of the Altera DE1 board, called the DE1-SoC board. This board is essentially the same as the DE1 board, but instead of a Cyclone II FPGA, it has a newer Cyclone V FPGA.

The lab instructions will refer to the DE1 board only, but if you have a DE1-SoC board then there are some differences you will have to take into account.



1. The FPGA device on the DE1-SoC board is the Cyclone V 5CSEMA5F31C6N
2. You must use the Altera Quartus Prime V15.0 (or V15.1) software, which also comes with a newer version of Modelsim
3. Some of the details (such as pin numbers, programming procedure) are different, but you can find the correct values and procedures in the DE1-SoC users manual.

You MUST use the correct software for your board, i.e. **V13sp0** for DE1 and **V15.x** for DE1-SoC



Each group will have their own package, which they can keep with them until the end of the course. To obtain the lab kit, ***all*** of the group members should go to the ECE department Technician's office, located in room 4140 of the Trottier building. They will have a list of the lab groups for this course, and will give you one of the Altera lab kits after you present appropriate identification (McGill student IDs).

***Print out and sign the waiver form accepting responsibility for the kit. Bring this waiver with you when you go to pick up the lab kit.***

***All members of the group must be present in order to receive the kits.***

***Please note that you are responsible for any loss of or damage to the kits. The list price for the Altera DE1 kits is currently USD\$125. The list price for the Altera DE1-SoC kits is currently USD\$175.***



nvtech.com

## TIME CHECK

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

### **3. Testing the 7-Segment LED Decoder on the Altera Board** .

You will now test the 7-segment decoder circuit you designed in Lab 2. Compile the decoder in the Quartus software. Once you have compiled the LED decoder circuit, it is time to map it onto the target hardware, in this case the Cyclone II chip on the Altera DE1 board. Please begin by reading over the DE1 user's manual, which can be found on the myCourses lab experiments page.

Since you will now be working with an actual device, you have to be concerned with which device package pins the various inputs and outputs of the project are connected.

In particular, you will want to connect the LED segment outputs from the instances of the *gNN\_7\_segment\_decoder* circuit to the corresponding segments of one of the four 7-segment LED displays on the Altera board.

**The mapping of the Altera Board's 7-segment LEDs' segments to the pins on the Cyclone FPGA device is listed in Table 4.4 on page 31 of the DE1 Development and Education Board Users Manual. The pin mappings for the other board components can also be found in section 4 of the manual.**

You will also want to connect, for testing purposes, 5 of the *slide switches* on the DE1 board to the inputs of the *gNN\_7\_segment\_decoder* circuit.

The mapping of the slide switches to the FPGA pins is given in Table 4.1 on pages 28 and 29 of the DE1 user's manual.

You can tell the compiler of your choices for pin assignments for your inputs and outputs by opening the *Assignment Editor*, which can be done by choosing the *Pins* item in the *Assignments* menu, as shown in the screenshot on the next page.

If you are using the DE1-SoC, remember to check for the correct pin assignments, which will be different than for the DE1 board.

**Assignment Editor**

Category: Pin

☒ Show assignments for specific nodes:

- ☒ segments
- ☒ segments[6]
- ☒ segments[5]
- ☒ segments[4]

Check All  
Uncheck All  
Delete All

Information:  
Allows you to view and edit assignments for specific nodes and entites. Specify one or more node or entity names, using one name per row, to allow the spreadsheet to filter the assignments, displaying only the assignments for the nodes and entites specified in the list. Only node and entity names that are turned on (checked) are displayed in the spreadsheet. If you use wildcards in the node or entity names, the wildcards are automatically expanded.  
--> Double-click to add or edit names, or drag and drop from the Node Finder.

Edit: ☒ segments[0]

	To	Location	General Function	Special Function	Reserved
1	segments[0]	PIN_6	Row I/O		
2	segments[1]	PIN_7	Row I/O		
3	segments[2]	PIN_8	Row I/O		
4	segments[3]	PIN_9	Row I/O		
5	segments[4]	PIN_11	Row I/O	CLKUSR	
6	segments[5]	PIN_12	Row I/O		
7	segments[6]	PIN_13	Row I/O		
8	segments				

Enter the pin number for a given circuit node into the Location boxes

Once you have assigned all of the inputs and outputs of your circuit to appropriate device pins, *re-compile* your design.

**You can check that the pins have been assigned correctly by looking at the floorplan on the pin planner (zoom in), and verifying that the right pins have been used.**

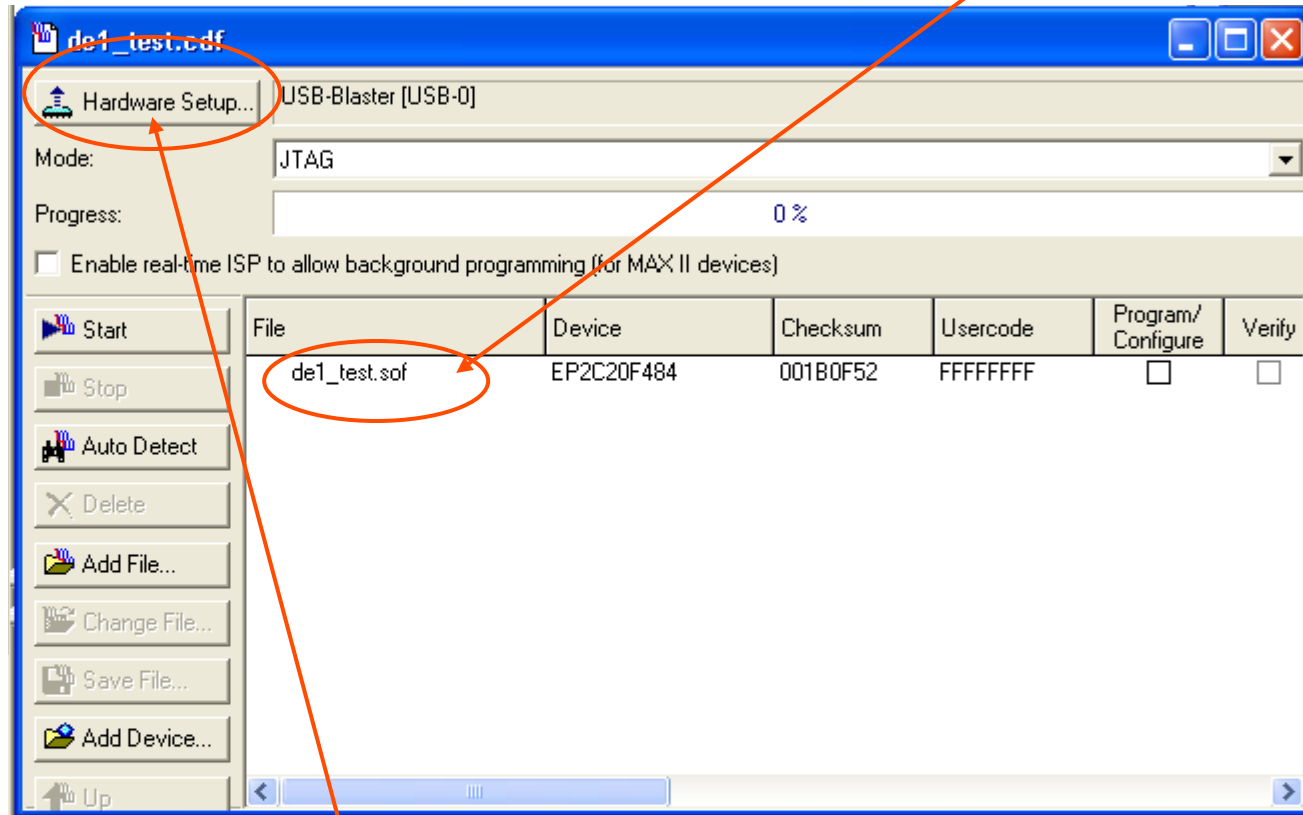


**Show your floorplan to the TA.**

Your design is now ready to be downloaded to the target hardware. Read section 4.1 of the DE1 user's manual for information on configuring (programming) the Cyclone II FPGA on the board. You will be using the *JTAG mode* to configure the device. The programming procedures are a little different than what is pictured in these slides for the DE1-SoC board, but you can find the procedure in the DE1-SoC user's manual.

**Take the Altera board out of the kit box, and connect the USB cable to the computer's USB port and to the USB connector on the Altera board.**

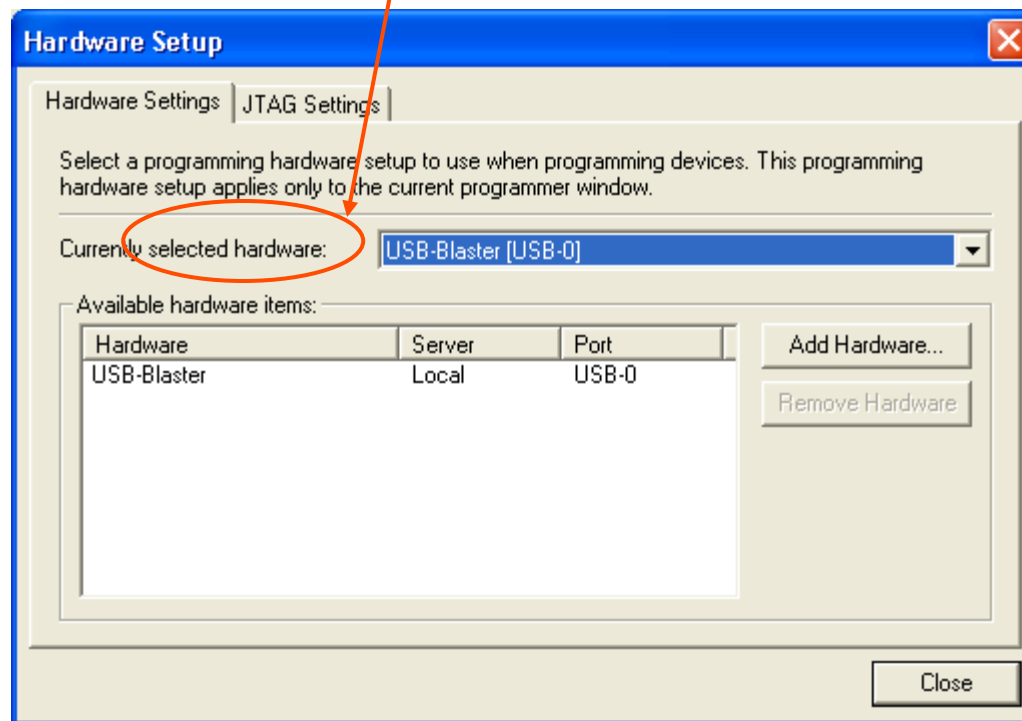
Next select the ***Programmer*** item from the ***Tools*** menu. You should see a window like the one shown below. There should be a .sof (SRAM Output File) file listed. If not, click “Add File”.



If there is no device visible, click on ***Hardware Setup***

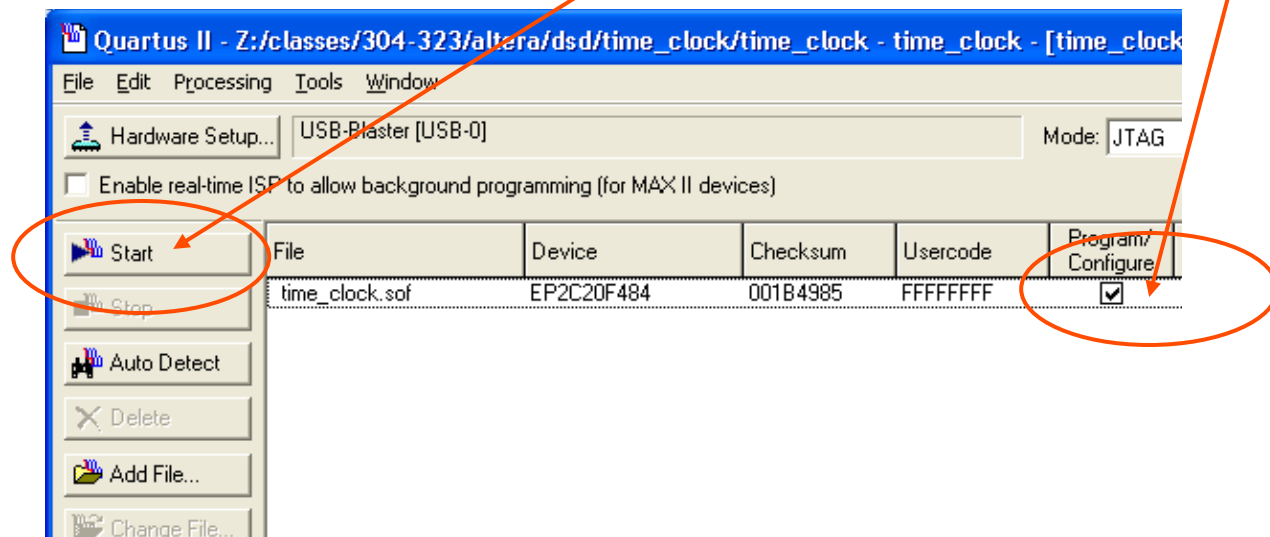
In the Hardware Setup window, select the correct communication hardware. Select “**USB-Blaster**”.

Click on Close to return to the Programmer window. If you still do not see a device, check the jumper settings on the board, and make sure that the USB cable is connected.





If everything seems in order (i.e. a file and a device are shown) you can carry out the FPGA programming. To do this, click on **Start**. Make sure that the Program/Configure checkbox is checked.



Make sure that the displayed symbols correspond correctly to the selected input code.

Demonstrate to the TA that your circuit is functioning properly by going through all of the 32 different switch settings.





nvtech.com

## TIME CHECK

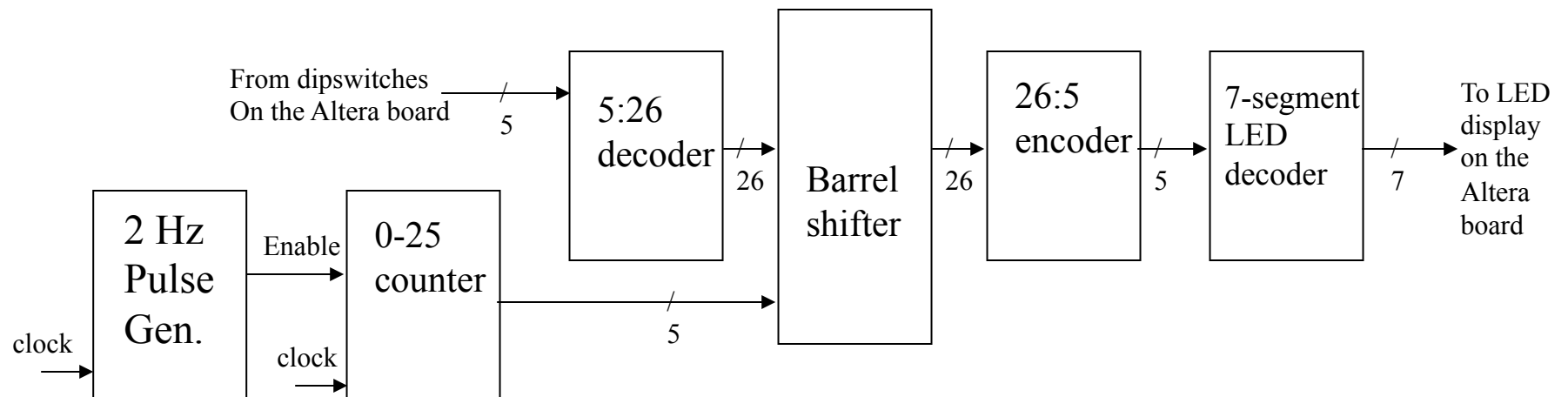
You should be this far at the end of your *second* 2-hour lab period!

## 4. Design of a Test-Bed.

You will now design a FPGA **test-bed** (not to be confused with a simulation testbench!) for your counter and other circuits, which will be expanded in future labs, and will ultimately form the basis for the final version of the Enigma Machine.

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.

The test-bed to be constructed in this lab should be as shown in the diagram below:



The first order of business in designing the test-bed is to design the pulse generator. You have designed all of the other building blocks for the testbed in previous lab experiments.

The Altera DE1 development board contains a clock generator which runs at 50 MHz (a period of 20 ns). This is much faster than the human eye can see, so we will want to run our counter at a much slower rate, say 2 cycles per second.

***However, following proper synchronous digital system design methodology, you should run ALL of your sequential units with the same high speed clock.*** This produces a fully synchronous system. Thus, we will still run the 0-25 counter off of the 50 MHz clock, but enable the counter only when the output of the pulse generator is high. This means that our pulse generator should output pulses that are only high for the width of one clock period.

## Design of a pulse generator using the Altera lpm\_counter module

The design of the pulse generator will be based around a counter that 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 circuit is a pulse that is high for one clock cycle when the count value is equal to zero.

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 pulse is equal to  $(N+1) \cdot T_c$ , where  $N$  is the value loaded and  $T_c$  is the clock period.

Design such a circuit in VHDL that produces a repetitive pulse with a period of ***0.5 second***, and a width of 20 *nsec*. We will use this as the count\_enable for the 0-25 counter. Name the output of the timer ***EPULSE***.

You will have to compute the constant value to load in to give the required amount of division and determine how many bits the counter should have.

To implement your counter, you are to use the Altera-supplied counter module as a component. This component is called ***lpm\_counter***.

Altera includes a library of LPM (**L**ibrary of **P**arametrized **M**odules) modules that include common circuit building blocks. You include LPMs as components in your VHDL description.

To use the *lpm\_counter* module in your design, you must include the following two lines at the beginning of your design entity:

```
LIBRARY lpm;  
USE lpm.lpm_components.all;
```

Note: For all of the Altera LPM modules, you do not have to include a COMPONENT statement in the architecture declarations area, as this is included when you invoke the lpm library. If you want to include one of your own modules as a component, you need to include the COMPONENT statement to declare it.

The Integer Arithmetic IP Cores User Guide gives details about the *lpm\_counter* module. Read the manual and learn how to use the *lpm\_counter* module in your VHDL design.

The *lpm\_counter* module needs a *port map* statement when you do the component instantiation, as with any component. However, you also will also need to use the generic map statement in your component instantiation. The *generic map* statement goes before the *port map* statement, and is used to declare parameters to the module, which can implement many different versions of the counter.

Show your VHDL design of the testbed circuit to the TA.





Carry out the simulation of the testbed circuit with Modelsim. Run the simulation long enough to have two full cycles of counts from 0 through 25.

**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 50 MHz clock (20 ns period) and run the simulation for at least 2 sec (so that you can see a few timer output pulses) then you will run through at least 100 million clock cycles! Running a simulation with this many clock cycles will probably take a long time, perhaps 10 minutes to half an hour.

So, just to make sure your pulse generator is working properly, set its division ratio to something much smaller, say 1000, and run the simulation for only 100  $\mu$ s. Your simulated circuit won't give the right division ratio in this case, but you will be able to see if your circuit generates pulses.

**When it comes to implementing the circuit in hardware, don't forget to put the division ratio back to the correct value!**

Show the results of your simulation to the TA.





nvtech.com

## TIME CHECK

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

## 5. Using the TimeQuest Timing Analyzer .

To ensure a properly working circuit, the designer must take into consideration various timing constraints. In class we saw that for a register to correctly store an input value, the input must be held stable for a period (called the *setup time*) before the clock edge, and also for a period (called the *hold time*) after the clock edge.

Whether a circuit meets these timing constraints can only be known after the circuit is *synthesized*. After synthesis is done one can analyze the circuit to see if the timing constraints (setup and hold times) are satisfied.

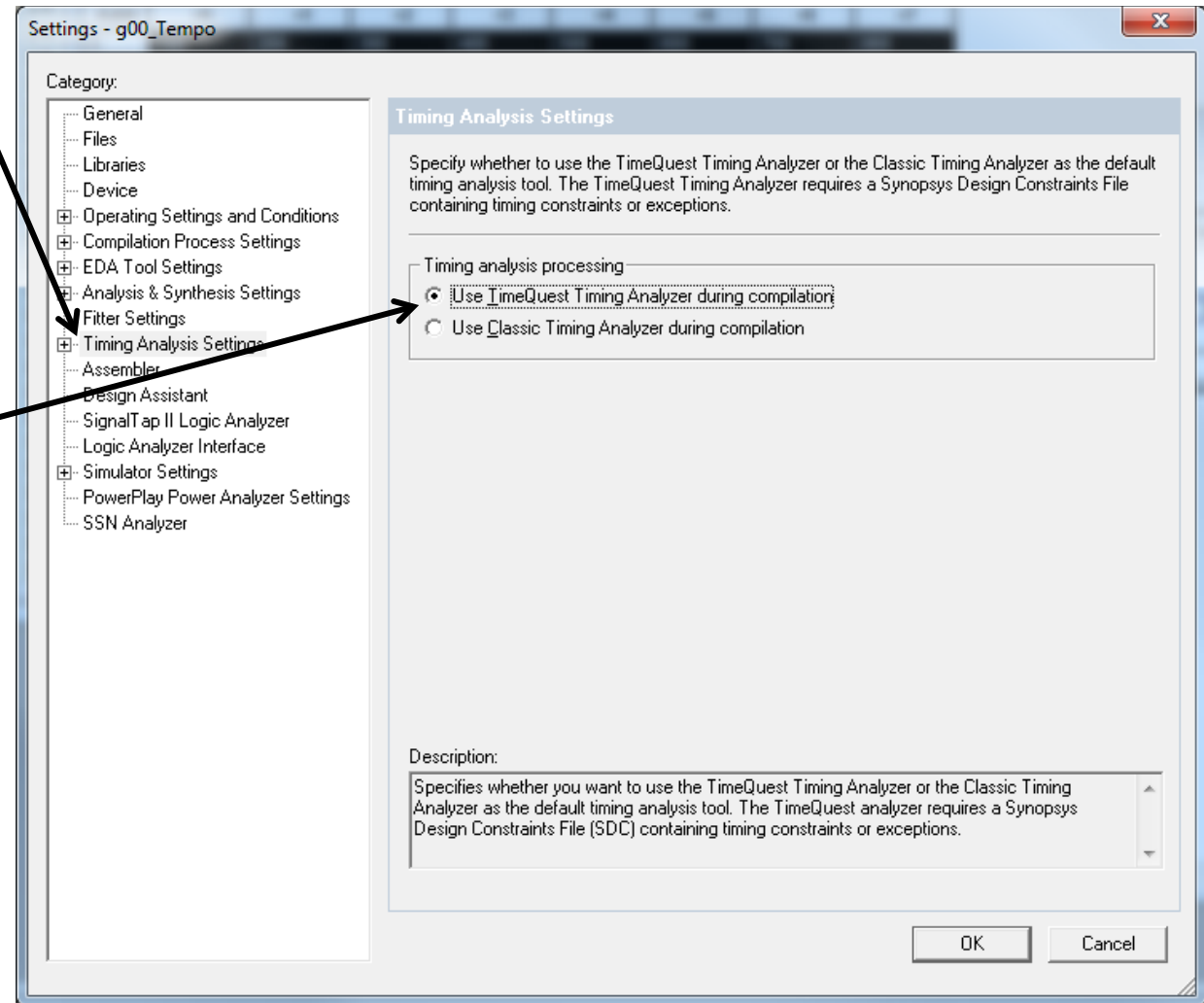
In Quartus II, timing analysis can be done using the *TimeQuest Timing Analyzer*. Please read pages 7-6 through 7-12 of the document “*Altera TimeQuest Timing Analyzer.pdf*” before proceeding to the next part of the lab.

From this reading you should gain an understanding of the term “*Slack*”, which refers to the margin by which a timing constraint is met (or not met).

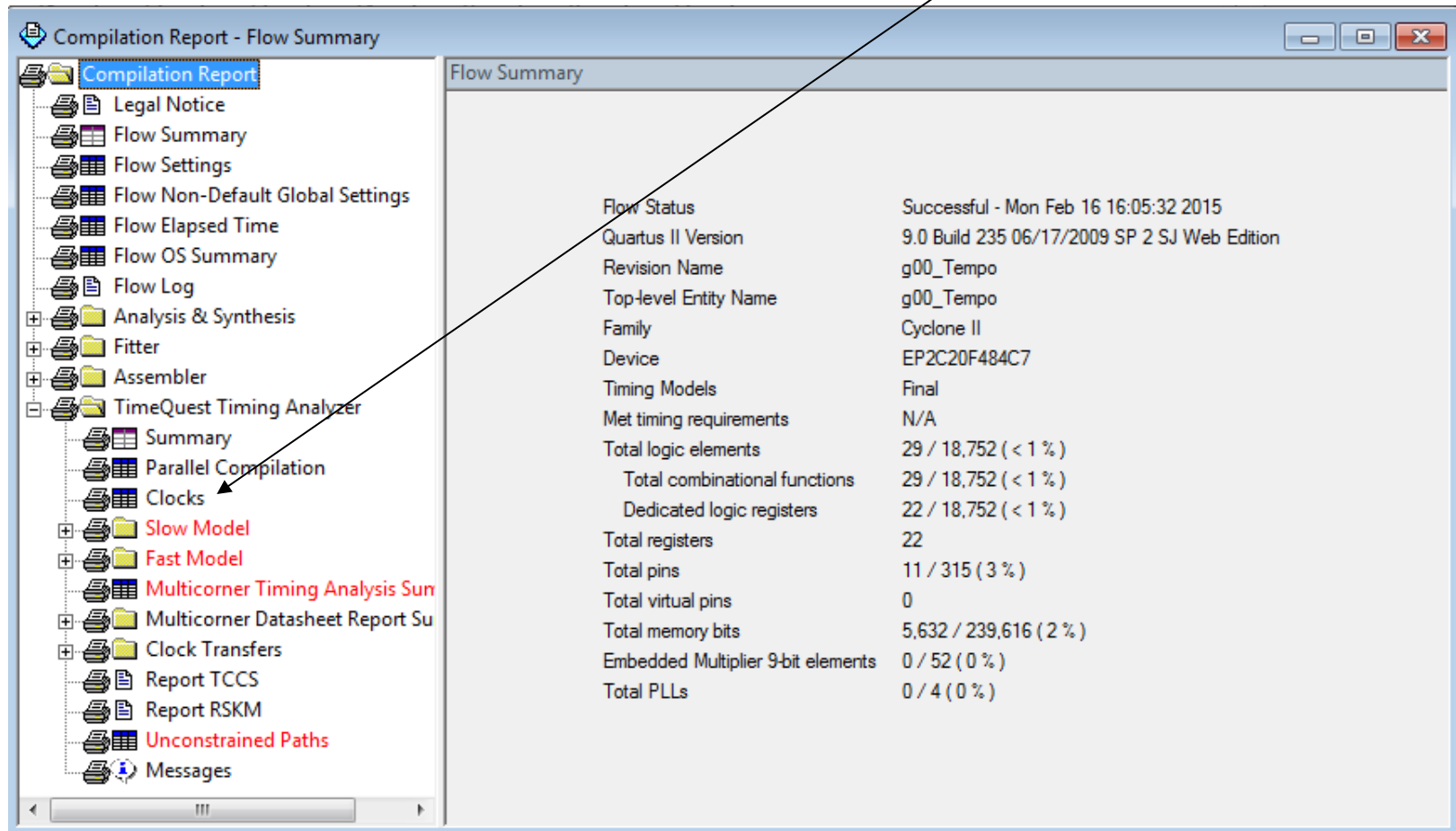
To make use of the TimeQuest Timing Analyzer, go to the “Settings” item under the “Assignments” menu item.

Click on “Timing Analysis Settings”

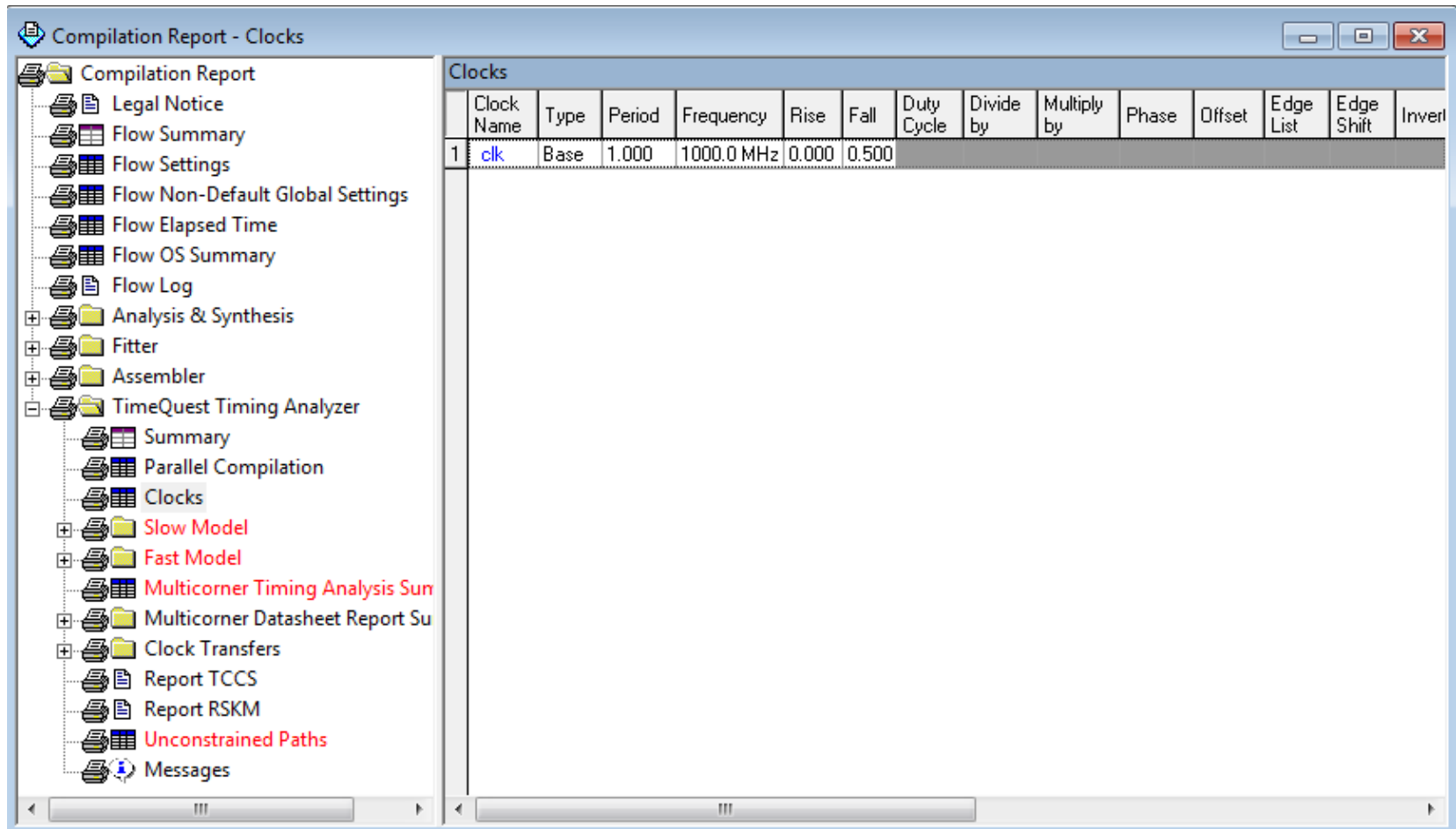
Then select “Use TimeQuest Timing Analyzer during compilation”



After compiling your design, there will be a “TimeQuest Timing Analyzer” section in the Compilation Report. Click on the “Clocks” item.



You will see your circuit's clock listed, but with an incorrect frequency.  
*This is a default value, which needs to be changed.*



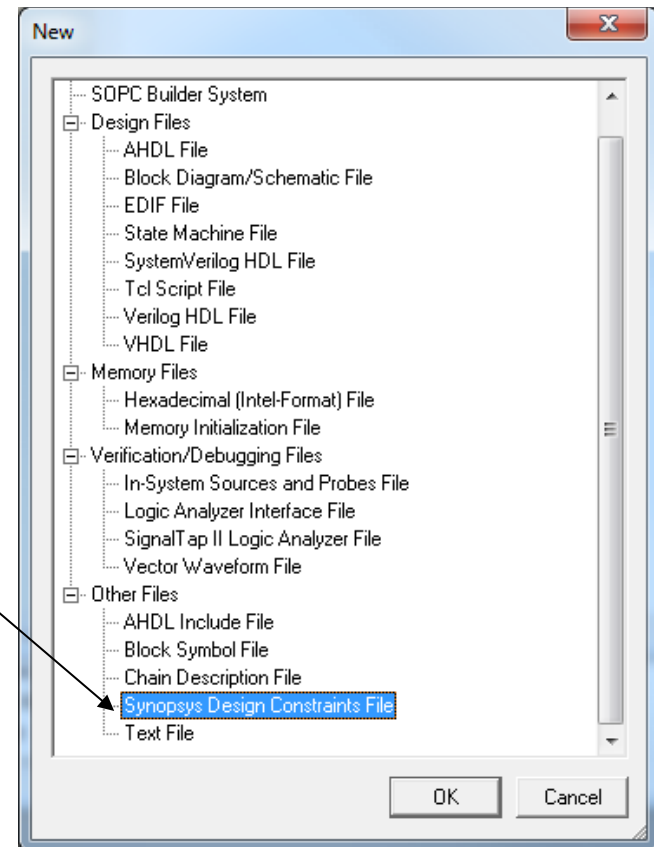
The screenshot shows the 'Compilation Report - Clocks' window. The left pane contains a tree view of the compilation report, with 'Clocks' selected under 'TimeQuest Timing Analyzer'. The right pane displays a table of clock parameters.

	Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle	Divide by	Multiply by	Phase	Offset	Edge List	Edge Shift	Invert
1	clk	Base	1.000	1000.0 MHz	0.000	0.500								

From the “File/New” menu item, select  
“Synopsys Design Constraints File”.

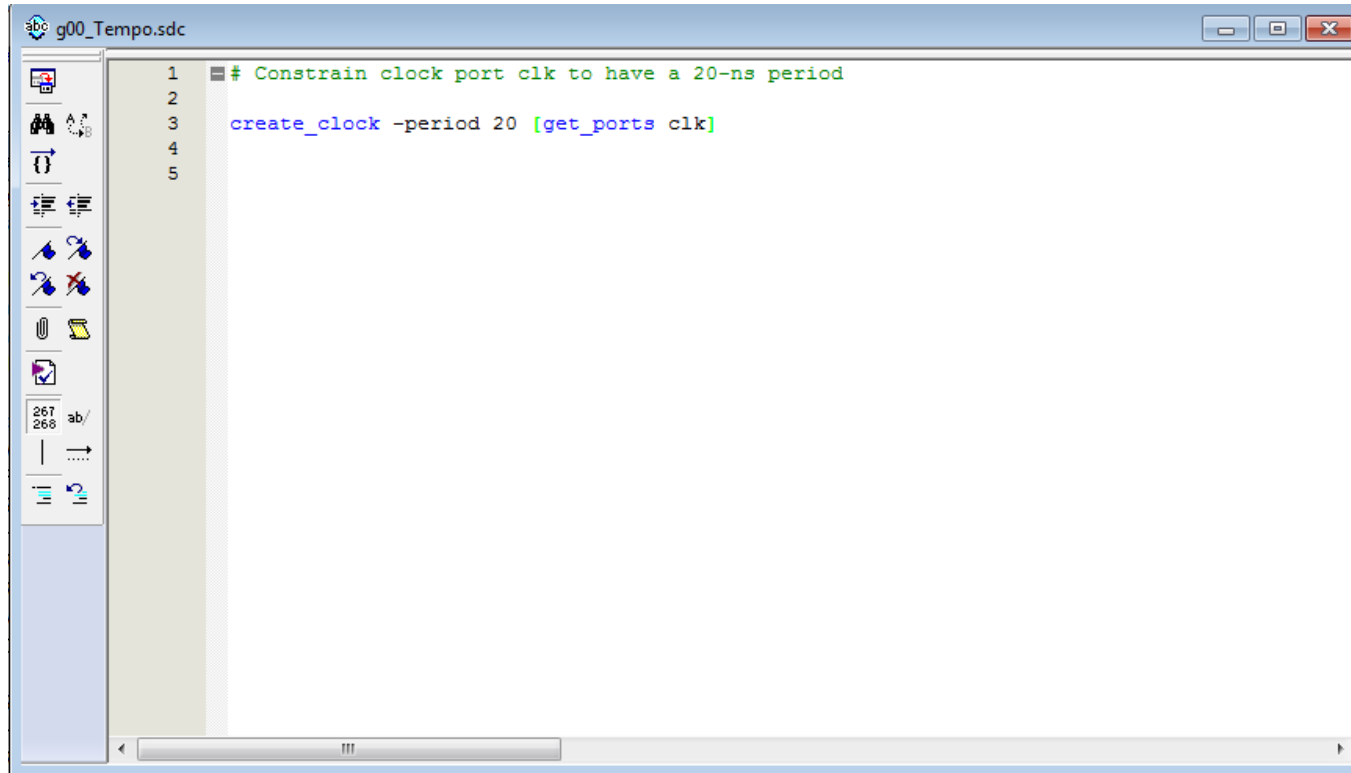
This “.sdc” file is where we can specify  
various timing constraints for our design.

We will add a single constraint specifying  
the clock period.



Type the following text into the window, and save the file with the name “gNN\_testbed.sdc” where NN is your group number. This will tell the timing analyzer that your clock period is 20ns.

Recompile your design. The Timing Analyzer will read the .sdc file and use the constraint information when doing its analysis.





After the compilation is finished, look at the timing summaries for the ***Slow Model*** and the ***Fast Model***. Pay attention to the ***Fmax Summary*** (which gives the maximum clock speed) and the ***Setup*** and ***Hold*** summaries (which give the slack amounts for the setup and hold constraints).

Compilation Report - Slow Model Fmax Summary

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	179.73 MHz	179.73 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such that the duty cycle (in terms of a percentage) is maintained. Altera

Mark down the following values, which you should show to the TA and include in your report:

Fast Model *Hold* Slack Value = \_\_\_\_\_

Slow Model *Setup* Slack Value = \_\_\_\_\_

Slow Model *Fmax* = \_\_\_\_\_



(Make sure you understand why the Fmax value is only shown in the timing report for the slow model, and why we only care about the Hold Slack in the fast model, and the Setup Slack in the slow model.)

**Are the Slack values all positive?**

**Is the Fmax value greater than 50MHz?**

**Here is how to make your boss happy, and get a nice bonus on your next paycheck:**

The FPGA device used in the DE1 board is the Altera EP2C20F484C7N. For the DE1-SoC it is the Altera 5CSEMA5F31C6.

The number 7 in the second-to-last position of the part number indicates the *speed grade* of the device. Smaller speed-grade numbers indicate a device with lower propagation delays. Faster chips are more expensive.

The cost per chip of the EP2C20F484C7N is currently **\$93.25** (as of Sept. 15, 2015) from Digikey.ca in single quantities). The cost per chip for the next slower speed grade device, the EP2C20F484C8N, is currently **\$77.68** (from Digikey.ca in single quantities).

Change the device used in your design (using the Assignments/Device menu) to the EP2C20F484C8N and recompile. For the DE1-SoC board, choose the Cyclone V 5CSEMA5F31C7 device. Look at the timing analysis. ***Does the circuit still meet the timing constraints?*** If so, tell your boss (or the TA) how much money you can save the company (assuming that your company will make 100,000 units of your design).



Remember to set the device back to the correct speed grade before proceeding to the next section.



nvtech.com

## TIME CHECK

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

## 6. Running the Testbed on the Altera Board

Once you have simulated your testbed and are satisfied that it functions properly, it is time to map it onto the target hardware, in this case the Cyclone chip on the Altera DE1 board.

You will also want to connect, for testing purposes, the counter's RESET signal to one of the Push Buttons on the Altera board.

Note that the output of a push button is high when the button is not being pushed, and is low when the button is being pushed.

It is useful to connect the Altera board clock oscillator signal to the global clock input of the Cyclone chip. This is already done for you on the board. What you need to do is to assign the clock input pin in your schematic to the Cyclone chip pin that is connected to the hardware clock. As stated in the DE1 users manual, this is pin #**91**. For the DE1-SoC board this is pin # PIN\_AF14.

Once you have assigned all of the inputs and outputs of your circuit to appropriate device pins, ***re-compile*** your design (and remember to reset the division ratio used in your pulse generator circuit back to its correct value from the value you set for simulation).

Also remember to set the device back to the correct speed grade before recompiling.

**You can check that the pins have been assigned correctly by looking at the floorplan view, and verifying that the right pins have been used. *Show the floorplan to your TA.***



Your design is now ready to be downloaded to the target hardware.

Download your design to the Altera board and demonstrate to the TA that your circuit is functioning properly by showing the effect of the barrel shifter in changing the decoded output.

Show the effect of pushing the RESET push button. This should reset the counter to zero, therefore setting the amount of barrel shifting to zero.

Using the dipswitches, input different settings of the input code. This code should be the one that is displayed when the system is reset (barrel shift set to zero at that point in time).





## TIME CHECK

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



## 7. Writeup of the Lab Report

Write up 2 short reports, one for the *0-to-25 counter* and one for the complete *test\_bed* circuit that you designed in this lab.

The report must include the following items:

- A header listing 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 the inputs and outputs. Provide a pinout or symbol diagram.
- The VHDL description of the circuit.
- A discussion of how the circuit was tested, showing representative simulation plots.
- A summary of the timing performance of the circuit, giving the timing analysis.
- A summary of the FPGA resource utilization.

The lab report, and all associated design files must be submitted, as an assignment to the myCourses 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 at 11:59 PM, Friday, March 18.**



# Grade Sheet for Lab #3

Winter 2016.

Group Number:\_\_\_\_\_.

Group Member Name:\_\_\_\_\_.

Student Number:\_\_\_\_\_.

Group Member Name:\_\_\_\_\_.

Student Number:\_\_\_\_\_.

Marks

	1.	<u>VHDL for the 0-25 counter circuit</u>	_____.
	2.	<u>Simulation of the 0-25 counter circuit</u>	_____.
	3.	<u>Floorplan of the 7-segment decoder circuit</u>	_____.
	4.	<u>Demonstration of the 7-segment decoder on the FPGA</u>	_____.
	5.	<u>VHDL for the test-bed circuit</u>	_____.
	6.	<u>Simulation of the test-bed circuit</u>	_____.
	7.	<u>Show the timing analysis for the test-bed circuit</u>	_____.
	8.	<u>Show your boss how much money you saved</u>	_____.
	9.	<u>Floorplan of the test-bed circuit</u>	_____.
	10.	<u>Demonstration of the test-bed on the Altera board</u>	_____.

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.