ECSE-323
Digital System Design

Lab #3 – *Sequential Circuit Design* Winter 2017

Introduction

In this lab you will learn how to use the Altera Quartus II FPGA design software to implement sequential logic circuits. You will also learn how to do timing simulations and circuit synthesis.

This lab will introduce the use of the Altera DE1 Development and Education Board, which is a hardware *Field Programmable Gate Array* (FPGA) prototyping system.

Learning Outcomes .

After completing this lab you should know how to:

- Synthesize hardware from schematic descriptions for FPGA target hardware
- Program the FPGA on the Altera DE1 board
- Use the SignalTap II embedded logic analyzer
- Perform timing simulations of circuits mapped to hardware

Table of Contents .

This lab consists of the following stages:

1. Design of a 52-element stack circuit
2. Timing simulation of the 52-element stack circuit
3. Obtaining the Altera DE1 board kit
4. Design of a testbed for the stack circuit
5. Programming (configuration) of the FPGA
6. Testing of the stack circuit using the testbed
7. Testing of the stack circuit using the SignalTap II logic analyzer
8. Writeup of the lab report

1. Design of a 52-element Stack .

We will use a structure known as a ***stack*** to hold the deck of cards, the play pile, and the hands of the user and computer players.

A stack is a data structure which consists of a number of storage elements (in this case 52 slots, each holding a 6-bit value). Data is typically entered into the stack only from its top end, and pushes the current data down. This is called a ***PUSH*** operation. Similarly, data is usually read out from the stack by shifting the top value out, and moving the current data up. This is called a ***POP*** operation.

We will modify the stack operation slightly by allowing the POP operation to delete an element from the interior of the stack, at a location determined by an 6-bit input address value, rather than from the top of the stack as in a normal stack POP. (In this case the behavior is much like RAM).

The stack circuit should have the following inputs and outputs:

- **DATA** – 6 bit input, contains the data to be “PUSH”ed onto the stack
- **MODE** – 2 bit control input, determines what action is to be taken on the next rising clock transition (NOP, INIT, POP, PUSH)
- **ADDR** – 6 bit address input, sets the location to be viewed, and the location of the element to be removed during a POP operation
- **ENABLE** – 1 bit control input, enables the operation specified by the MODE input. If ENABLE is low, no operation takes place.
- **RST** – 1 bit asynchronous reset input, this should clear (set to zero) all stack locations, and set NUM to be zero
- **CLK** – 1 bit clock input
- **VALUE** – 6 bit output, indicating the value of the stack location pointed to by the **ADDR** input.
- **EMPTY** – 1 bit status output, indicating that the stack is empty (no elements)
- **FULL** – 1 bit status output, indicating that the stack is full (52 elements)
- **NUM** – 6 bit status output, indicating the number of stack elements

The stack operations are:

• **POP** – If NUM=0 (empty=1) then nothing happens. Otherwise, stack slots ADDR+1 through stack slot 52 are shifted up by one place, while stack slots 0 through ADDR-1 are left unchanged. The element in stack slot ADDR is lost (overwritten by the value from stack slot ADDR+1). The value of NUM is decremented by one. Stack slot 52 is the "bottom" of the stack.

• **PUSH** – If NUM=52 (full=1) then nothing happens. Otherwise, the value of NUM is incremented by one. The value at the DATA input is loaded into stack slot 0, and all of the other stack slots are shifted down (i.e. stack slot 0 gets shifted to stack slot 1, stack slot 1 to stack slot 2, and so on with stack slot 50 being shifted to stack slot 51). The contents of the last slot (slot 51) will be lost.

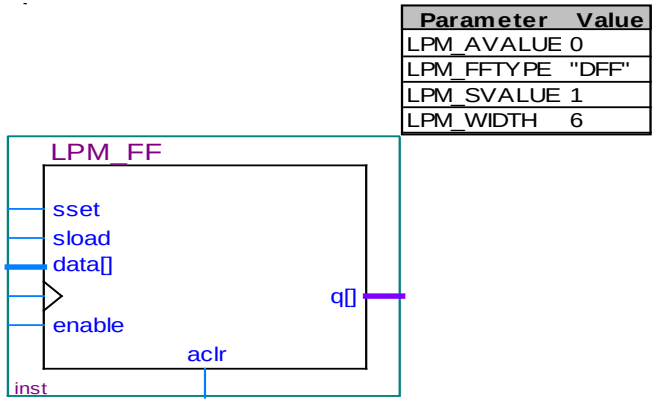
• **INIT** – The value of NUM is set to 52, the top entry is set to 0, the next entry is set to 1, the next to 2, and so on, with the final entry (stack bottom) set to 51.

• **NOP** – (No Operation). Nothing happens in this operation.

• **RESET** – The value of NUM is set to 0, and the entries in each stack location are all set to zero.

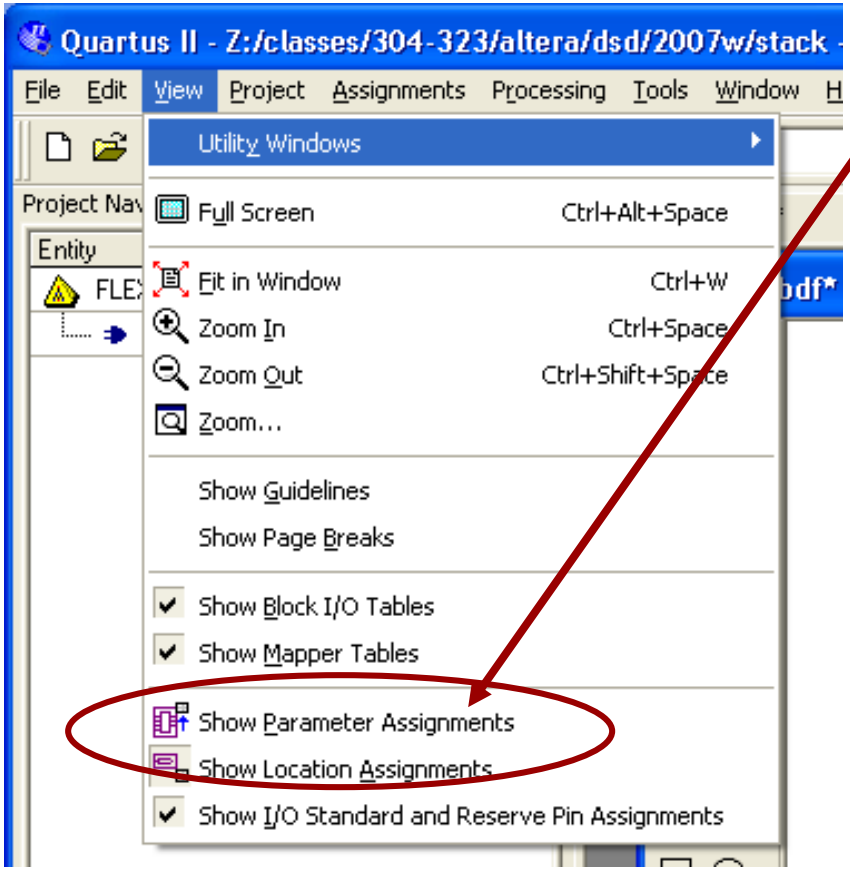
The design of the stack can be done using schematic capture in Quartus, or directly in VHDL (this section will present schematic capture if you haven't learned about describing sequential systems in VHDL yet).

Construct the stack by connecting together ***52*** LPM_FF modules into a bi-directional shift register (get used to using cut-and-paste!).

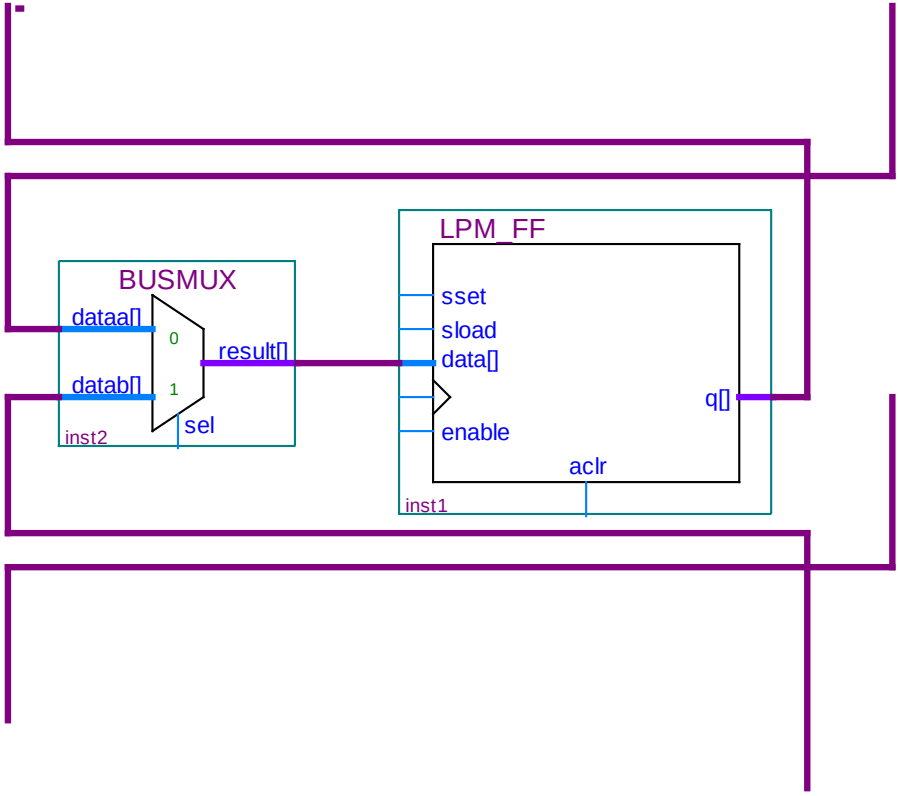


Each register element will hold 6 bits (representing a card number). The ***sset*** (synchronous set) control input will cause a synchronous loading of the parameter value of LPM_SVALUE. This will be used to initialize the contents of the stack. The ***aclr*** input asynchronously clears the register to 0. Note: you do not need to use the ***sload*** input, as the ***enable*** input will serve the same purpose.

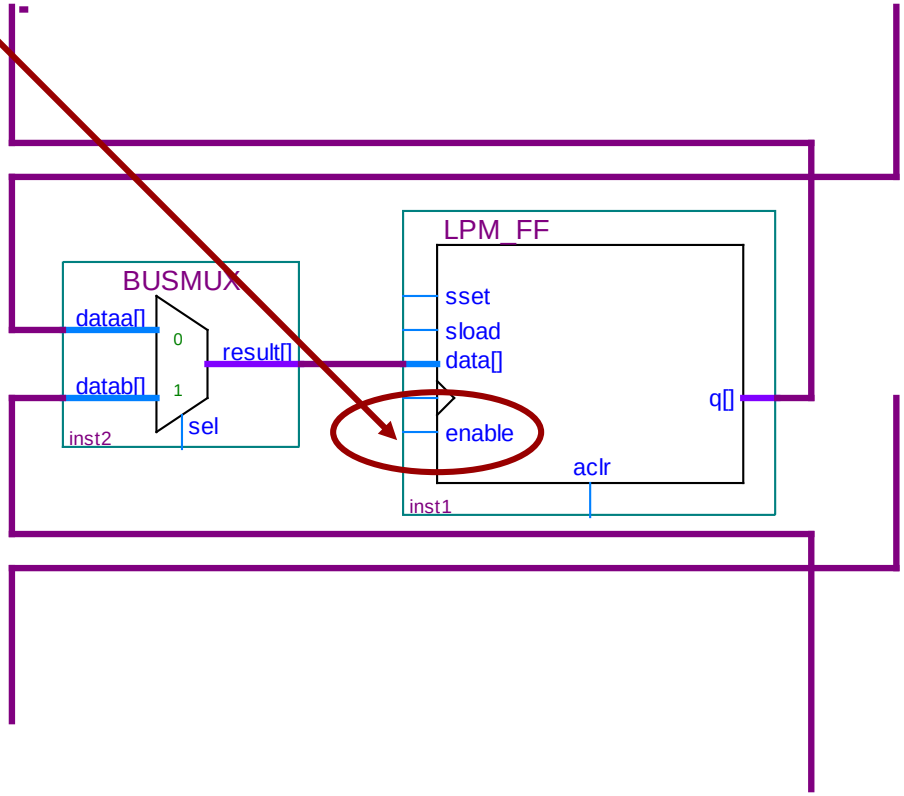
TIP: You can turn the module parameter information on or off using the "***Show Parameter Assignments***" item in the "***View***" menu:



To make the shift-register bi-directional (i.e. able to shift in either direction), connect the data input through a ***BUSMUX*** multiplexer module, which passes the output of either the previous or the next register in the chain. The setting of the BUSMUX ***sel*** input therefore determines the shift direction. The top input of the busmux for slot 0 should be connected to the DATA input of your stack module.

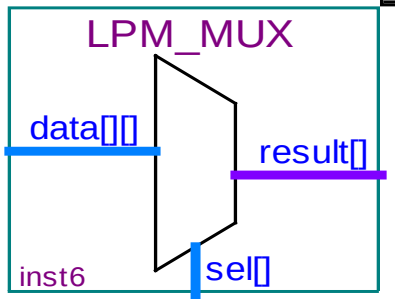


The POP function will be a little tricky to implement, since only part of the stack is being shifted. Control which parts are to be shifted by using the ***enable*** input of the stack flipflops. You may find the ***gNN_pop_enable*** circuit you designed in the last lab to be useful here.



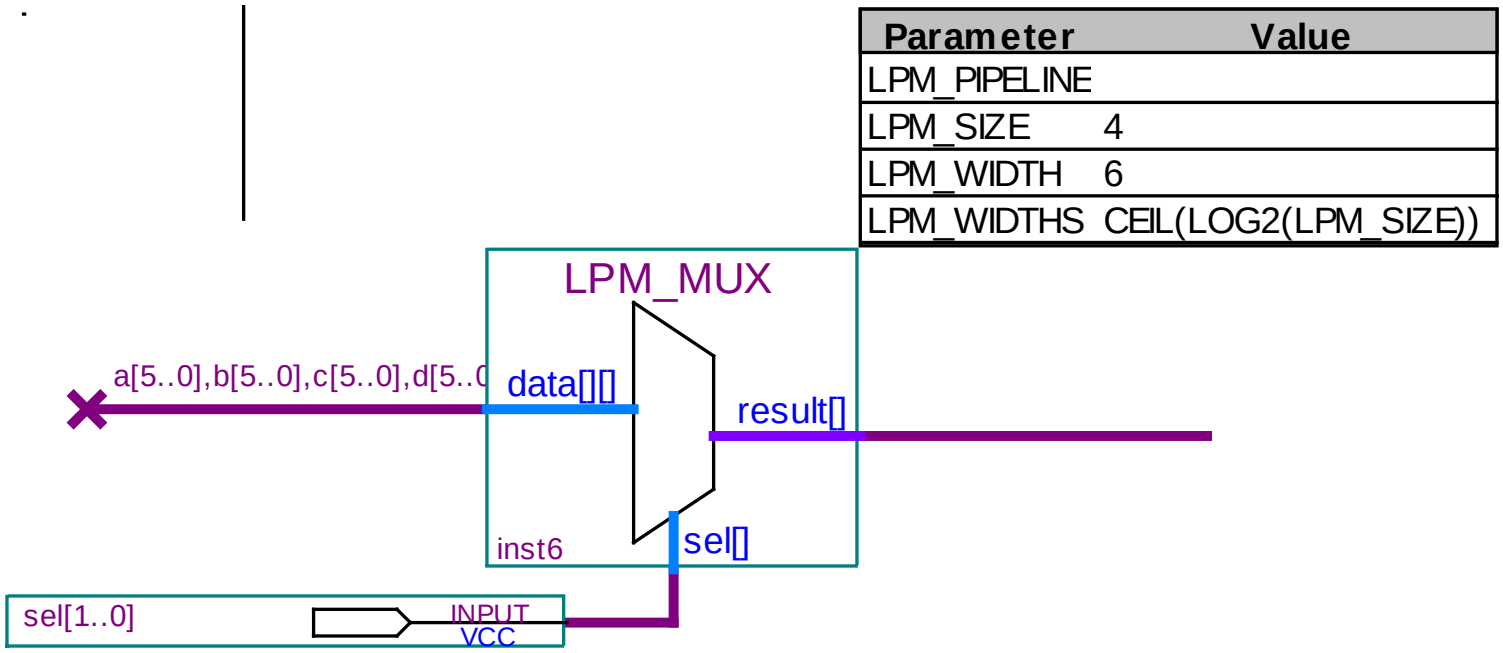
To obtain the ***VALUE*** output, use an ***lpm_mux*** module (not a busmux module). The ***ADDR*** input will be used to select the appropriate stack value to be passed to the ***VALUE*** output.

Parameter	Value
LPM_PIPELINE	
LPM_SIZE	52
LPM_WIDTH	6
LPM_WIDTHS	CEIL(LOG2(LPM_SIZE))

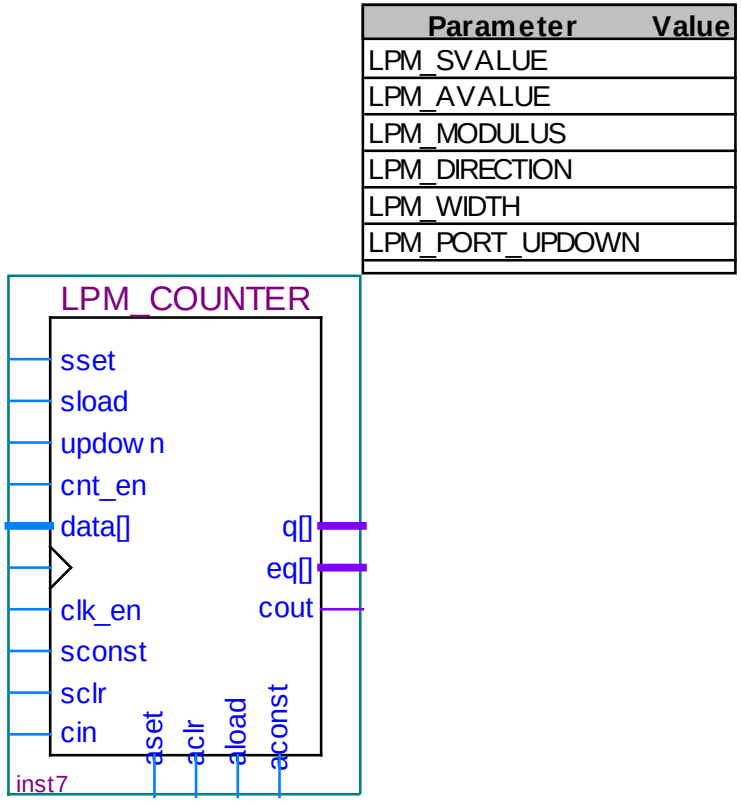


Make sure to set the ***aclr***, ***clken***, and ***clock*** ports of the lpm_mux module to “UNUSED” (right-click on the module, select “Properties”, then choose the “Ports” tab).

Note that the ***data*** input of the lpm_mux is written as a 2D array of signals. You can connect 1D arrays to this input by stringing their names together, as shown in the example below. The example is of an lpm_mux module having four 6-bit inputs. When ***sel=00*** the vector ***d[5..0]*** is passed to the output, when ***sel=01*** the vector ***c[5..0]*** is passed to the output, and so on.



Use an ***lpm_counter*** module to implement the computation of ***NUM***.
Set irrelevant module ports to UNUSED (e.g. aload, aset, cin, cout, …)



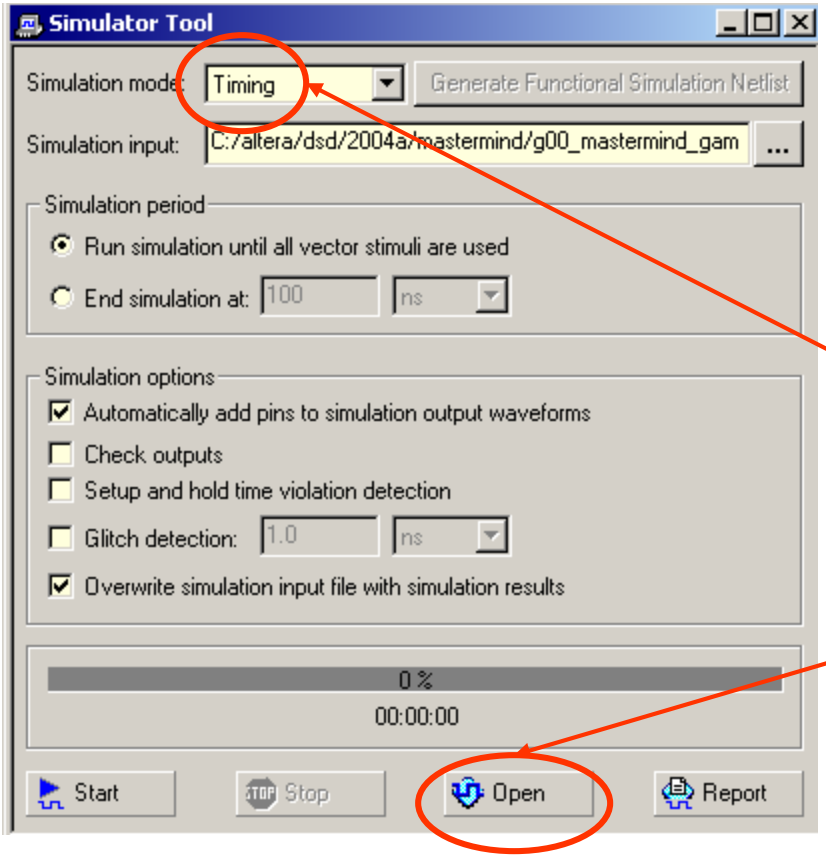
2. Layout of the 52-element Stack Circuit .

Design the complete stack circuit using *schematic capture* (or VHDL).

Name your circuit ***gNN_stack52*** (where NN is your group number).

When you have finished laying out the schematic diagram show it to the TA.

3. Timing Simulation of the 52-element Stack Circuit .



Next, compile the stack circuit and create a symbol for it, then insert the symbol into a new empty schematic. Compile the project using the ***Processing/Start Compilation*** menu item. Once that is done successfully you can 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 20 nsec. Set the END TIME (in the Edit menu) to 2 usec (which means 100 clock pulses).

Draw waveforms for the other input signals so as to fully test the functionality of the stack circuit. You may want to run more than one simulation, to test each operation individually.

Once all your input waveforms have been setup click on "Start" in the simulator window to run the simulation.

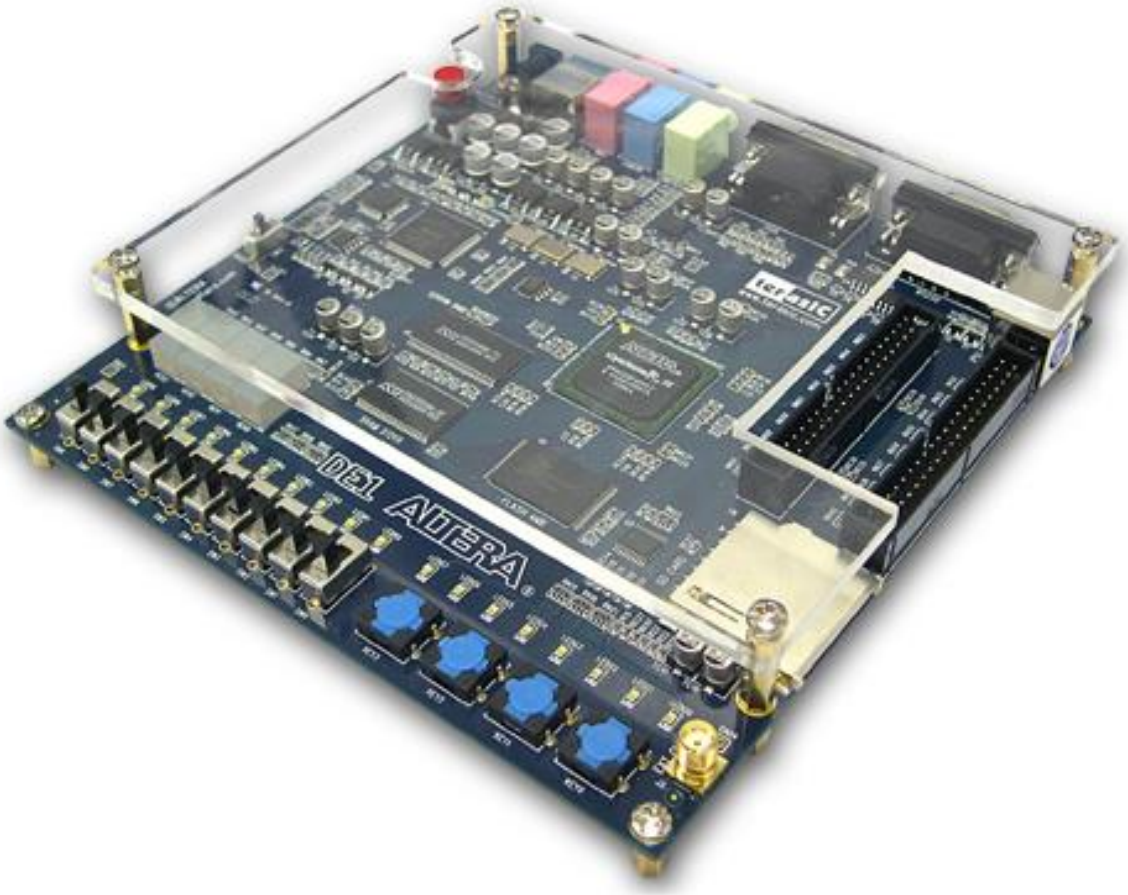
Show the results of the simulation to your TA. You should demonstrate the 4 different operation modes (No Op, Init, Push, Pop) of the stack, the Reset operation, as well as the proper functioning of the VALUE select multiplexer. Be sure to display the NUM output to show that it is changing appropriately.

4. 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



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. Ask for ***Mr. Charles Burtles.*** He 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).

All group members must be present and display their ID card!

Print out and sign the waiver form (from the WebCT Experiments page) 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.**

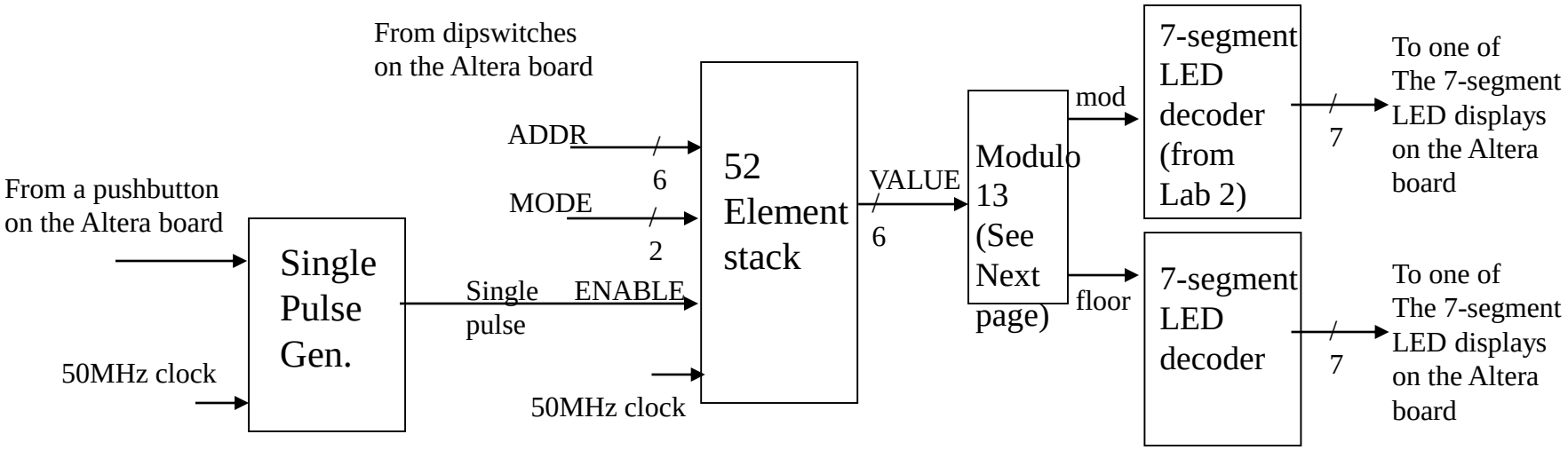
Please note that you are responsible for any loss of or damage to the kits. The academic price for the Altera DE1 kits is currently over $150.

5. Design of a Test-Bed for the Stack Circuit

You will now design a ***test-bed*** for your stack circuit. This will be expanded in future labs into the full card game system.

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:



(connect DATA to some non-zero constant inside your design)

Modulus-13 conversion

The MOD 13 of a number X can be found by finding the value X - floor shift_right (X*5,6)*13 and then using *13 as 1101 and converting multiply to shifting left and adding. More work since 13 is not power of 2. Shifting right by 6 is straight forward. Subtraction implies that the shifted value is in two's complement (signed integers used). You are ready to build this circuit now. You may use the RANDU as a starting point.

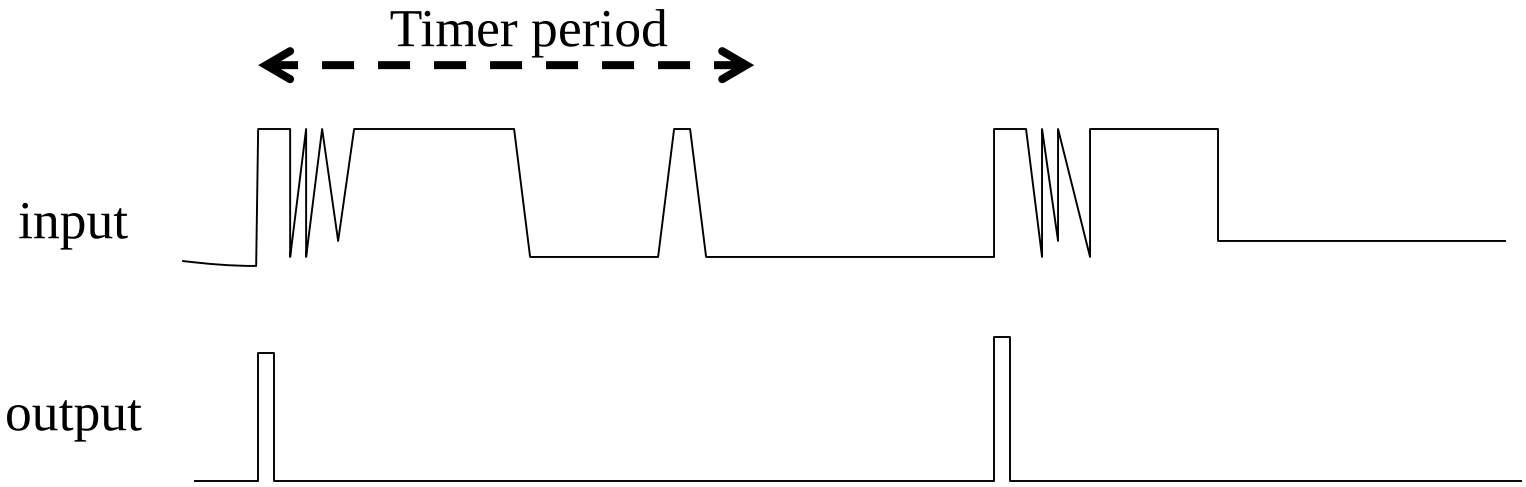
For example, if *A=37*, then *A/13=2.846…*, and *floor(A/13)=2*. Thus *Amod13(37) = 37-2*13 =11*

The first order of business in designing the test-bed is to design the single-pulse generator. The purpose of this circuit is to “debounce” the noisy pushbutton on the Altera board. When you press on one of these buttons, the contact bounces many times before making a solid contact. This causes the signal to swing many times between 1 and 0 values.

Your circuit should use a counter (and one or more ***SRFF*** modules) to act as a timer. This timer will begin counting when the button is first pressed and generate a ***single pulse that is one clock cycle wide***. The circuit should not generate any more pulses until the counter has reached its limit. The count limit should be chosen to give a count time which is longer than the expected time of the button bouncing (assume this is about 40 msec). At a clock rate of 50MHz, this implies a count limit of 2,000,000 or about 2^21.

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. ***Do not use any other signal to clock your sequential modules – avoid “ripple counting***”.

The diagram below illustrates the desired behaviour of the pulse generator circuit:



The single pulse output will be used to enable a single stack operation (e.g. one “POP” operation) every time the pushbutton is pressed. If the stack enable signal was high for more than one clock cycle, then more than one operation would be performed each time you pushed the button. That is not what you want!

6. Timing Simulation of the Stack Testbed .

When the schematic for the test-bed is complete, create a symbol for it, and insert an instance into a new Quartus schematic window.

Show your schematic to the TA and explain to him or her how your circuit works.

Before simulation, first compile the design using the ***Processing/Start Compilation*** menu item.

Viewing the Compilation Report

Look at the ***Flow Summary*** section of the Compilation Report and note the FPGA resource utilization (i.e. how many logic cells were used?).

Also take a look at the ***floorplan*** (under the ***Fitter/Floorplan View*** section of the compilation report, and observe how the circuit has been fitted into the FPGA. Read over the datasheet for the Cyclone II chip (available on myCourses) to understand the architecture of the device.

Finally, look at the ***Timing Analyzer Summary*** (under the ***Timing Analyzer*** section of the compilation report). Take note of the path with the largest propagation delay. 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.

Carry out the simulation of the testbed circuit following the same sequence of steps that you carried out for the simulation of the stack circuit.

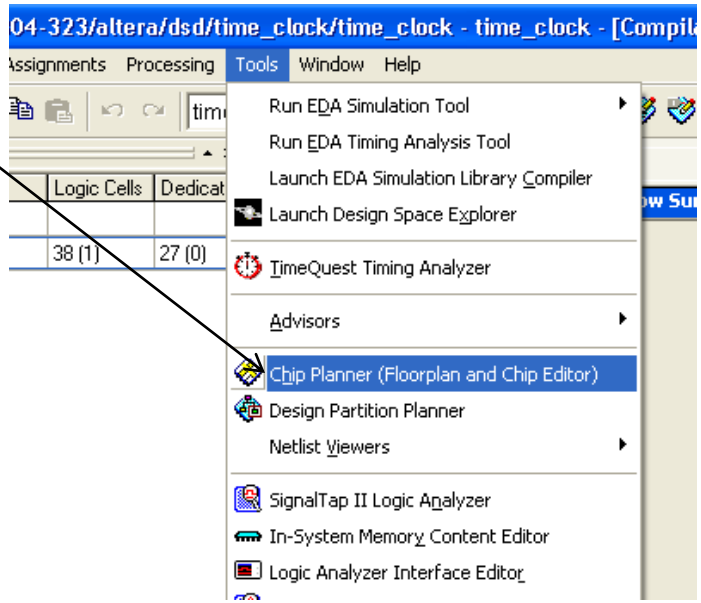
You should always be aware of how long your simulations will run. You can get a good handle on the computational load for the simulation by counting the number of clock cycles that will occur through the length of the simulation. If you use a 50 MHz clock (20 nsec period) and run the simulation for 200msec then you will run through 10 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, for simulation use a slower clock, (e.g. 50Khz) as you did in the previous simulation. You will need to compensate for this by changing the count factor in your pulse generator (and will need to change it back when you map to the hardware device).

Show the results of your simulation to the TA.

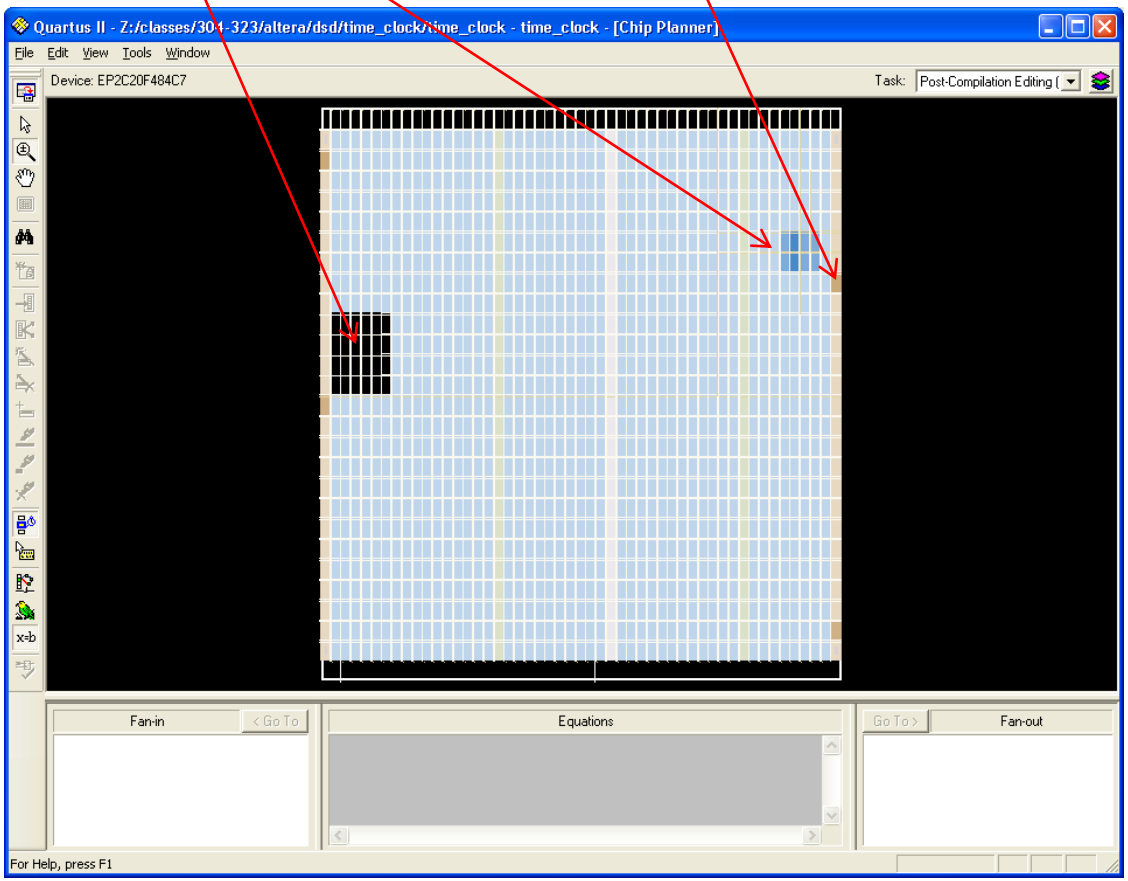
Viewing the Floorplan

Also take a look at the ***floorplan*** and observe how the circuit has been fit into the FPGA. Read over the datasheet for the CycloneII chip (available on myCourses) to understand the architecture of the device.

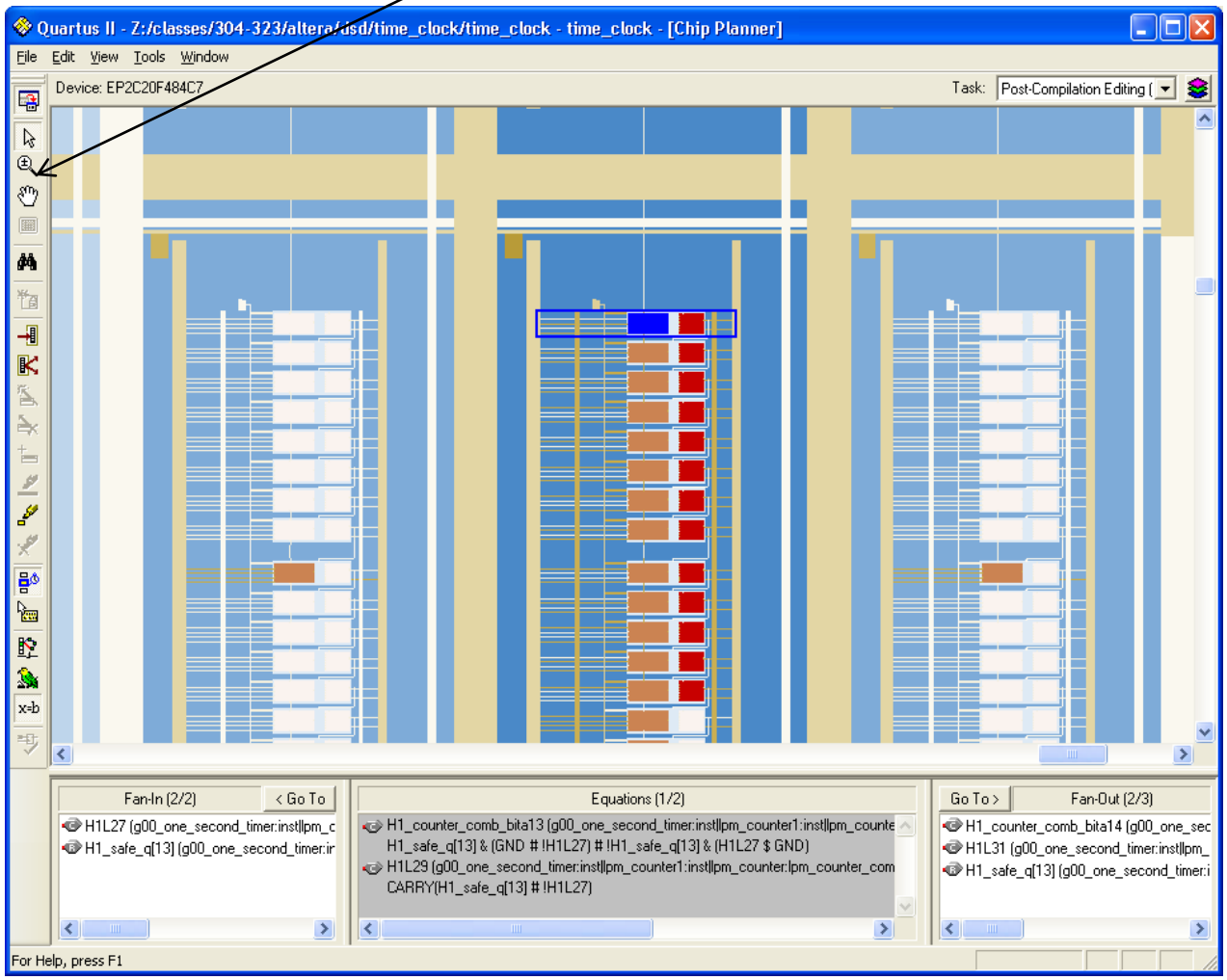
To view the floorplan, select "Chip Planner" from the "Tools" menu.



The floorplan shows the mapping of your circuit to the FPGA, i.e. which logic blocks are used for which circuit functions, which interconnects are used, and which FPGA pins are assigned to the circuit inputs and outputs.



Zoom in using the zoom tool to see details of the logic block usage



7. 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 II chip on the Altera DE1 board.

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 to.

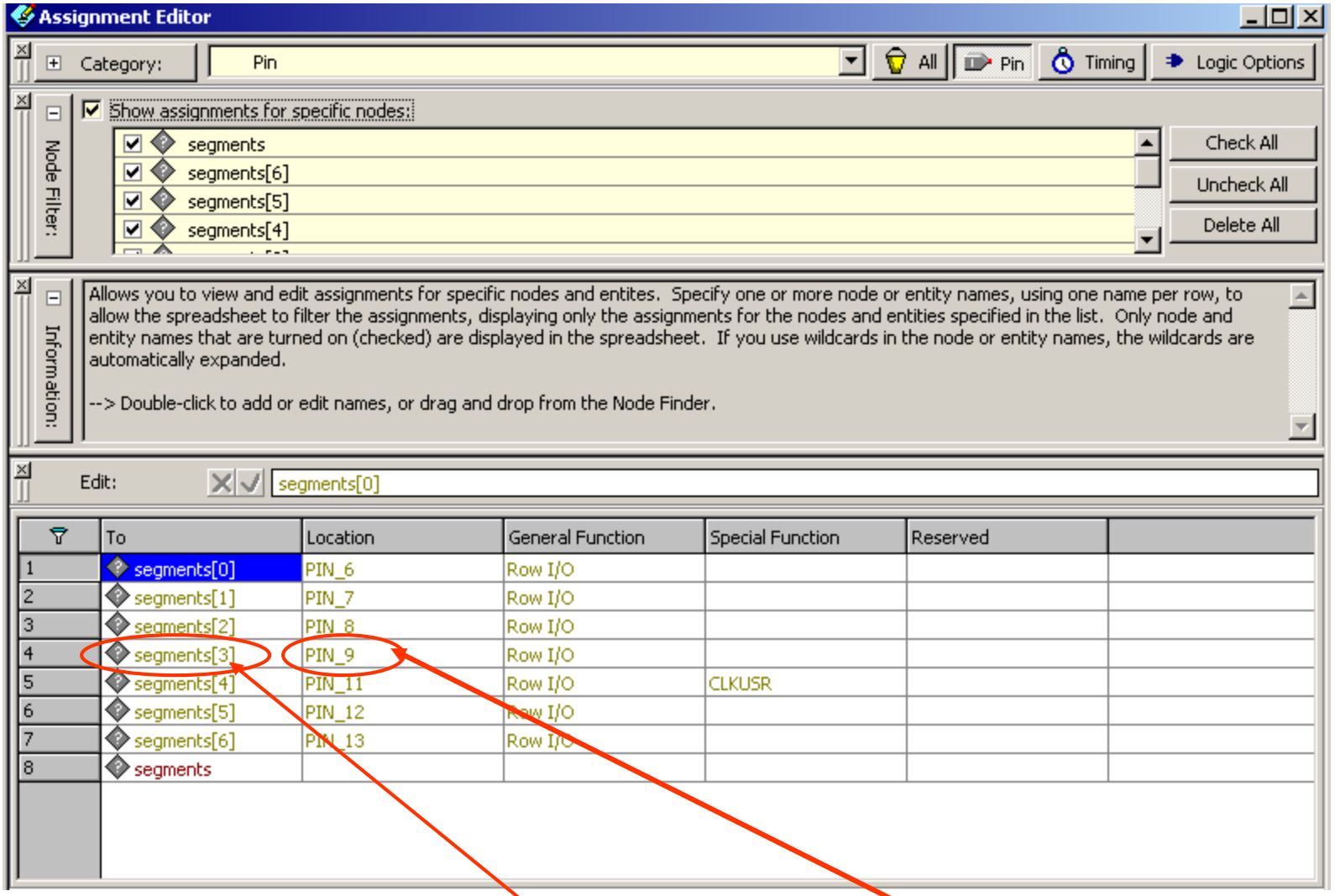
The mapping of the Altera Board's individual LED display segments to the pins on the Cyclone II device is listed in Table 4.3 on page 29 of the DE1 Development and Education Board Users Manual. A quick online search will allow you to download the manual for your own use.
The pin mappings for the other board components can also be found in section 4 of the manual. Note make sure that you use the manual for the correct board. DE1 with cyclone 2.

You will also want to connect, for testing purposes, the stack'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 II 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 II chip pin that is connected to the hardware clock. As stated in table 4.5 of the DE1 users manual, this is pin ***L1***.

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.



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

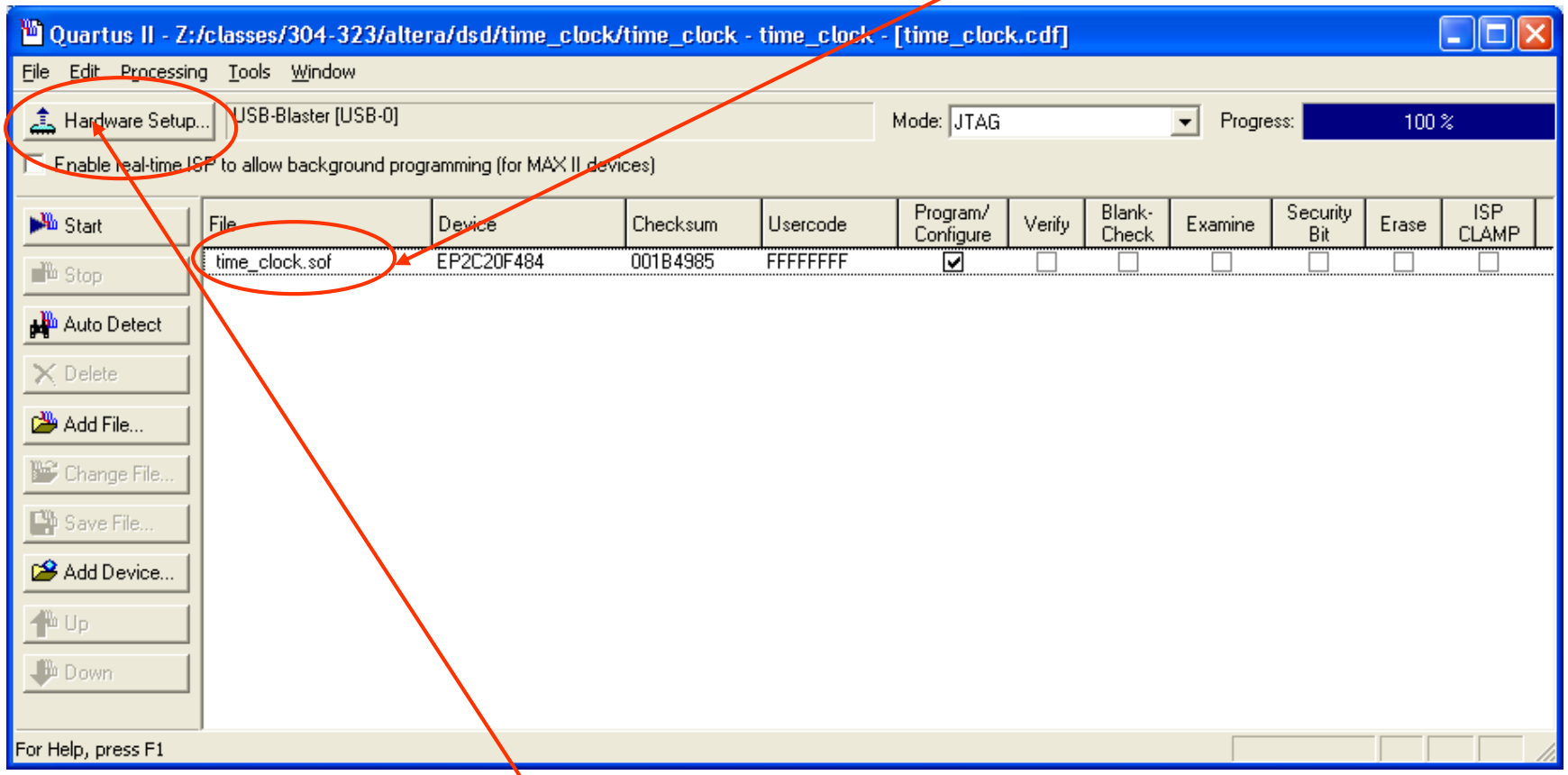
Once you have assigned all of the inputs and outputs of your circuit to appropriate device pins, ***re-compile*** your design (and remember to reset the division ratio used in your pulse generator circuit back to its correct value from the value you set for simulation!)

You can check that the pins have been assigned correctly by looking at the floorplan view (zoom in to see the pin labels), and verifying that the right pins have been used.

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.

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.

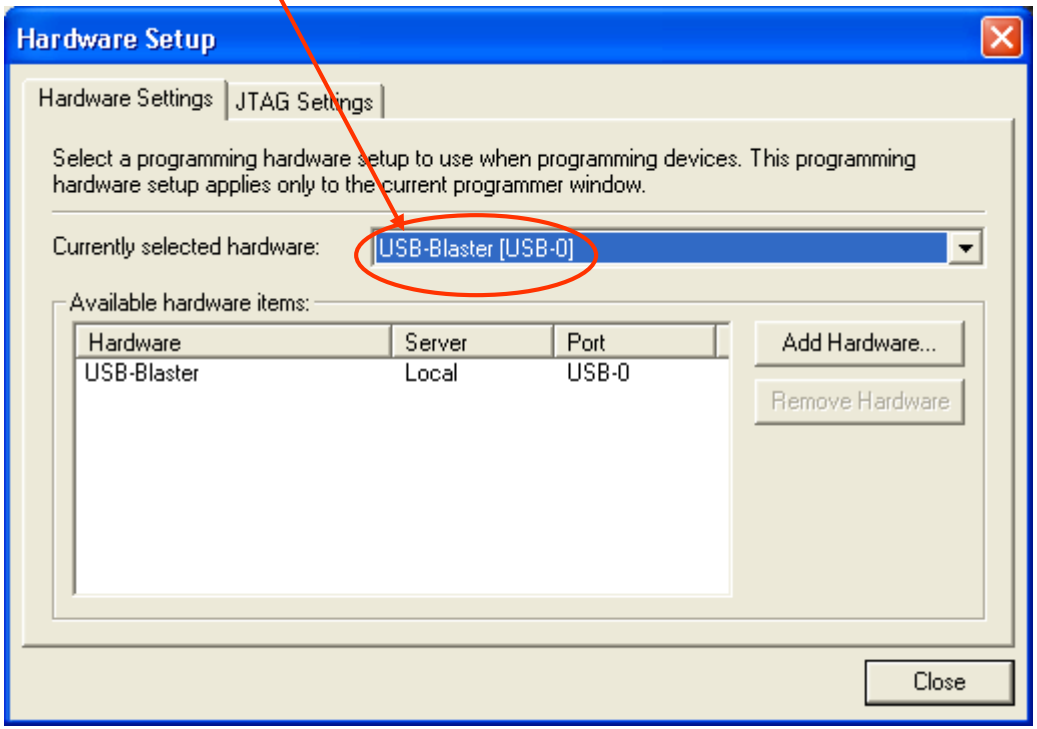
Next select the ***Programmer*** item from the ***Tools*** menu. You should see a window like the one shown below. There should be a file listed. If not, click "Add File".



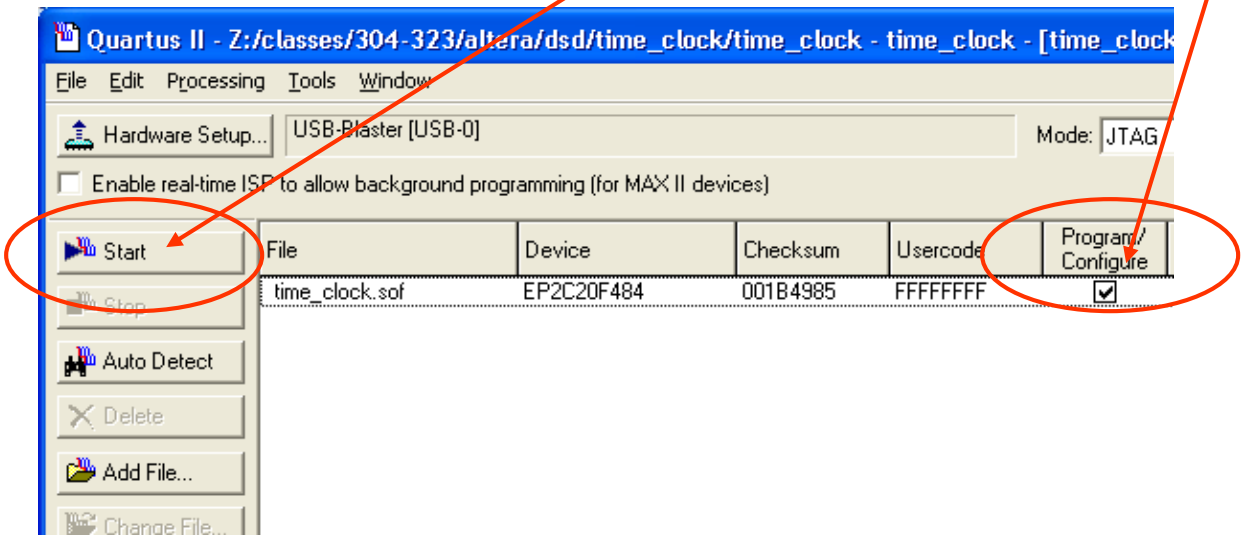
If there is no device visible, click on ***Hardware Setup***

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.



Demonstrate to the TA that your circuit is functioning properly.

Show the effect of pushing the **RESET** push button. This should initialize the stack.

Using 6 of the dipswitches, select different stack locations to view. The card value at the selected stack location should be displayed on the 7-segment LED.

Using one of the other pushbuttons, POP the stack element currently being viewed. Check to see that this element actually was removed. Did you have any difficulty with multiple actions (e.g. removal of more than one element per button press)?

Demonstrate the other operational modes (Push, Init).

8. On-Chip Testing using the SignalTap II Logic Analyzer .

The LEDs on the DE1 board are limited in number, and a human observer of these can only detect changes at a relatively low speed. Thus the LEDs provide only a limited means for probing the circuit behaviour when debugging a design. One could use an oscilloscope to measure signals brought out to the expansion connectors (the connectors on the right hand edge of the DE1 board), but there are no oscilloscopes in the lab, and even if there were, the signals brought out to the connectors can have only a relatively low maximum data rate due to the inductance of the output pins and connector.

It would be advantageous to be able to measure activity on the high speed data lines inside the chip.

One could use simulation to get an idea of the behaviour of the circuits inside the chip, but simulations do not always match the physical reality due to imperfect circuit modeling assumptions.

Fortunately, the Quartus II software provides a way to get at and measure the signals inside the chip. It does this with a tool called ***SignalTap II***.

SignalTap II is a logic analyzer whose circuitry is embedded into the FPGA fabric along with the circuit under test, and uses the FPGA's onchip memory blocks to store time samples of the data lines being analyzed.

The SignalTap II logic analyzer can only be used if there are enough free logic block and memory block resources in the FPGA to support it. For the projects being constructed in this course, there is plenty of free resources left over so that you will always be able to fit in a SignalTap II analyzer.

The functioning of the SignalTap II logic analyzer is patterned after hardware logic analyzer systems that have long been employed in digital system design and maintenance.

These systems include a set of high-speed probes (usually between 8 and 256) which connect to digital wires to be analyzed.

Software inside the analyzer samples the data on the probes at regular intervals, set by an internal or external clock. Typically the sampling is started, or triggered, by some logical condition on the lines being probed, and runs for a fixed time, or until a stop condition is sensed on the inputs.

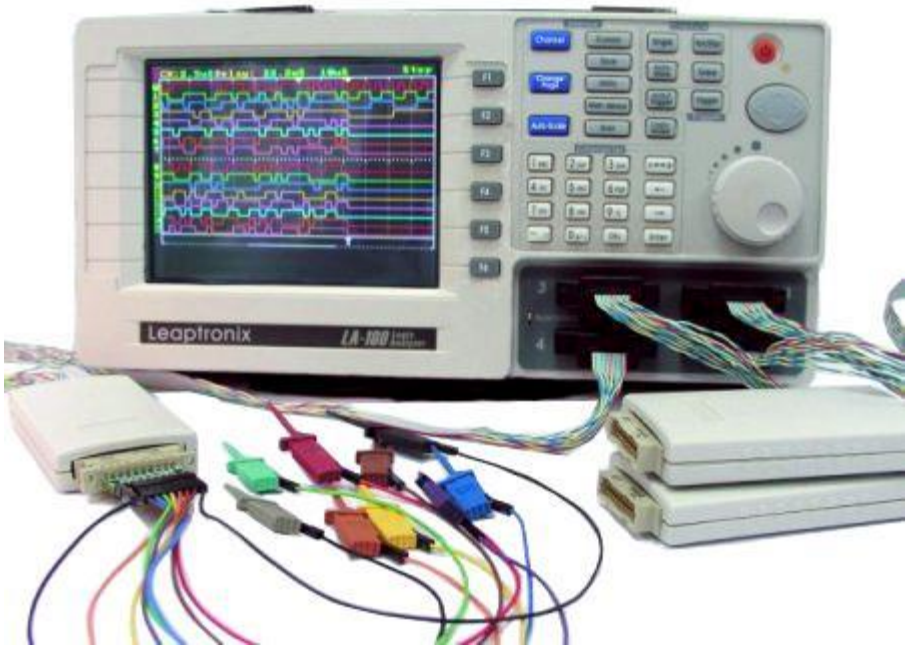
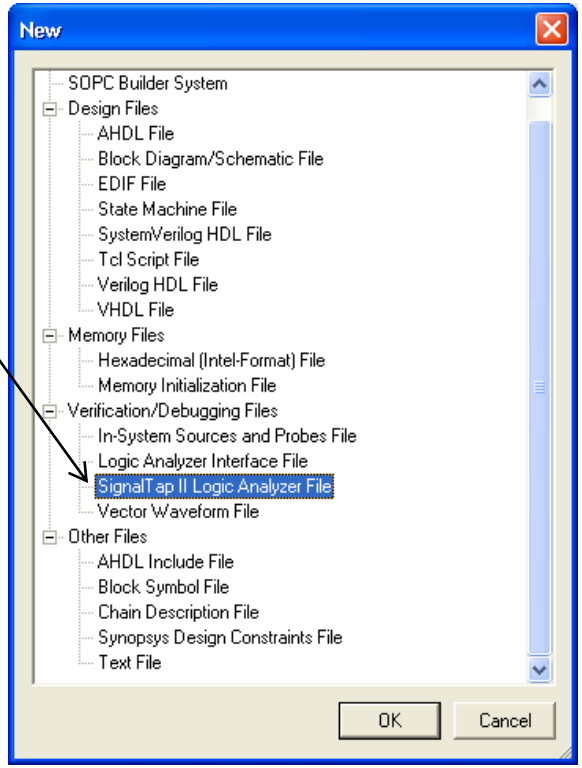


Image from http://www.elexp.com/tst_p100.htm

Likewise, the SignalTap II logic analyzer captures data on a set of data lines (nodes in the circuit that you specify), starting on a user-defined condition, and continuing for a set time period or number of samples. The data thus sampled is stored in on-chip memory, which is then read by the Quartus II software after the sampling period has finished.,

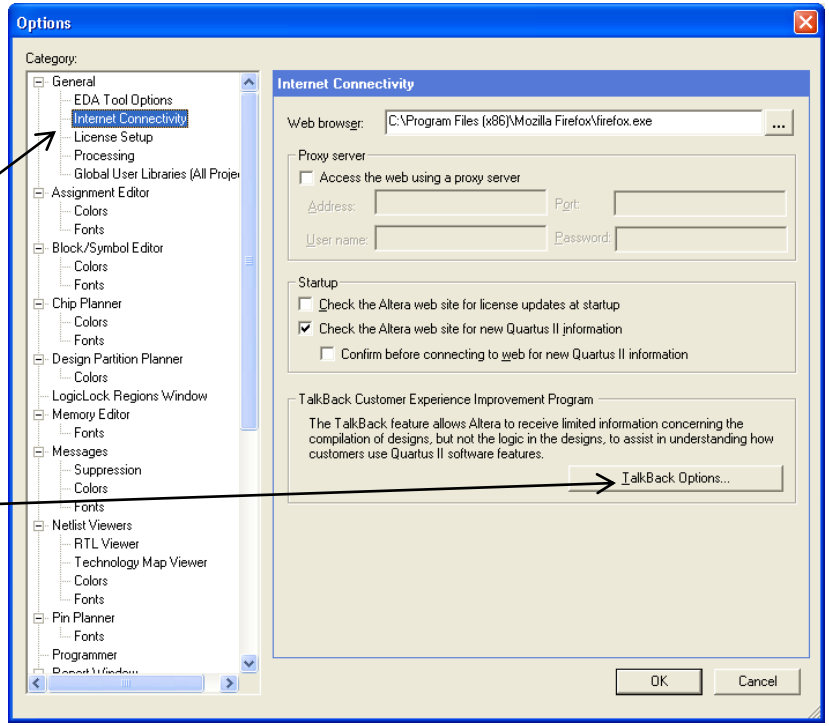
To setup the Quartus software so that it can use the SignalTap II tool, first create a new SignalTap II Logic Analyzer File by choosing the corresponding item from the File/New menu.



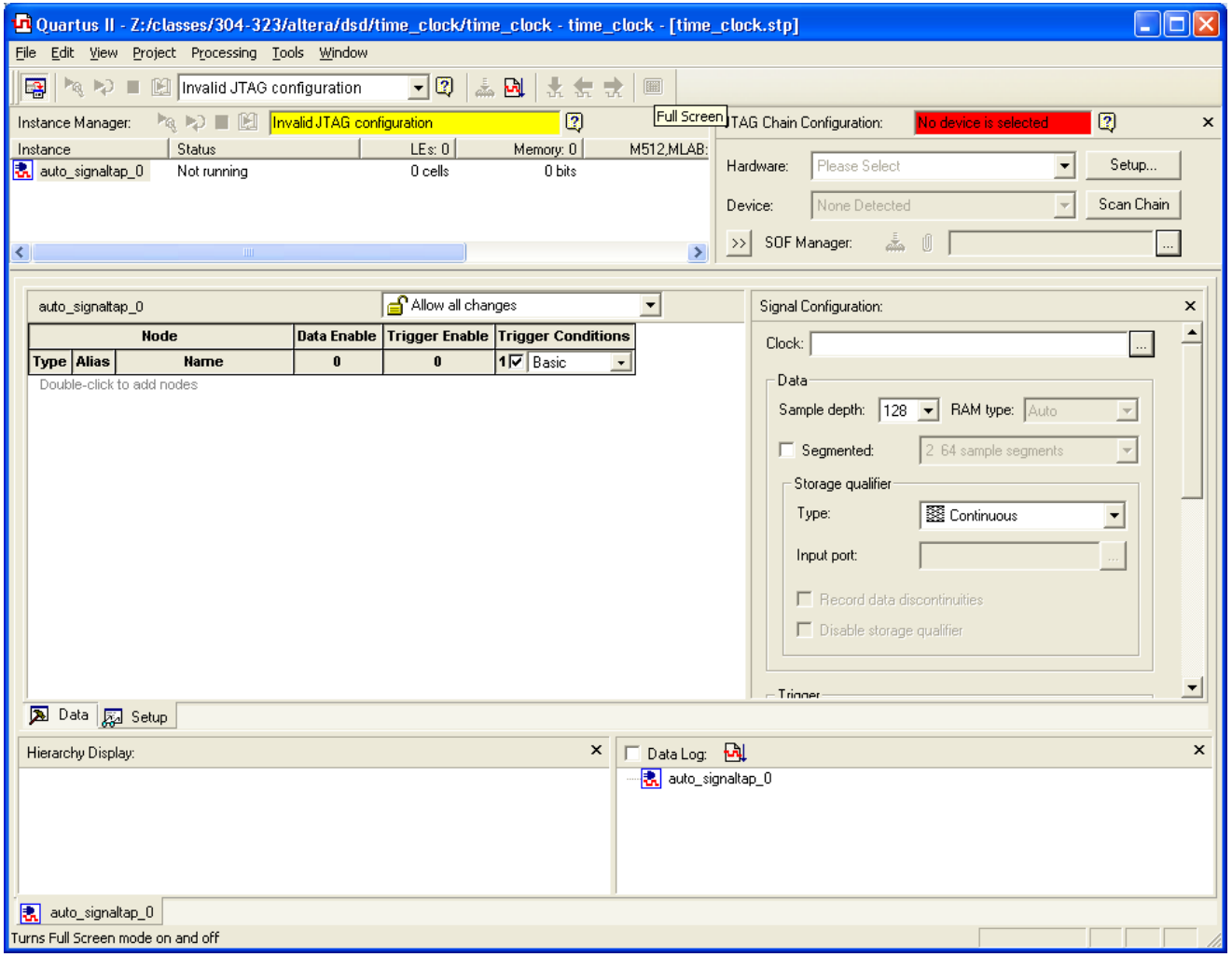
You may get the following error message popup (especially if you are using the web version of Quartus on your own computer:

If so, turn on TalkBack by going to Tools/Options and selecting the Internet Connectivity tab. Then click on TalkBack Options and turn on the TalkBack Feature in the window that pops up.

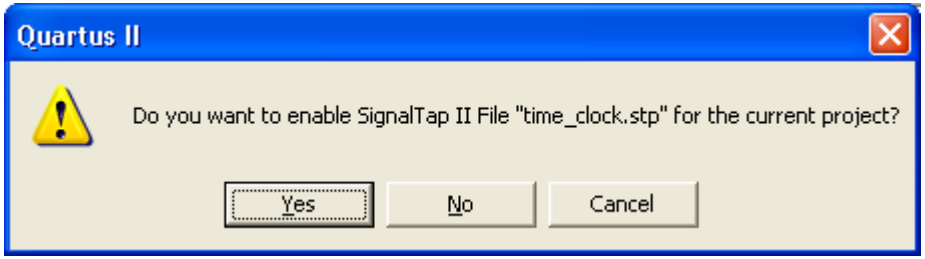
You will have to restart Quartus.



The SignalTap II window should now appear, and look like the screenshot below

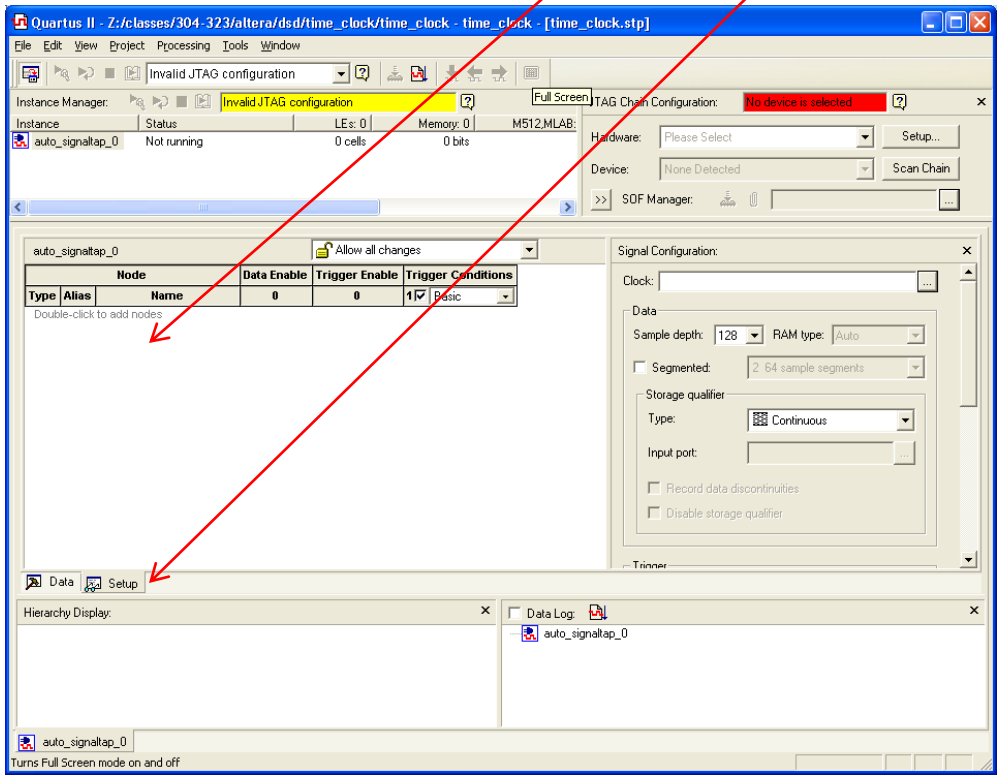


Before doing anything else, save the file (File/Save) giving the name “gNN_stack52_testbed.stp”. In the window that pops up, say “Yes”. This enables the SignalTap II file to be active for the current project.



The next thing to do is to specify the circuit nodes to be analyzed. For this tutorial, we will look at the VALUE, EMPTY, FULL, and NUM output signals of the stack52 module.

To specify the signals to be analyzed, make sure that the Setup tab is selected, and double-click in the area labeled “Double-click to add nodes”. This will bring up the node finder window, which should be familiar to you.



Add the nodes corresponding to the ADDRESS, MODE, enable and reset inputs to the stack52 module.

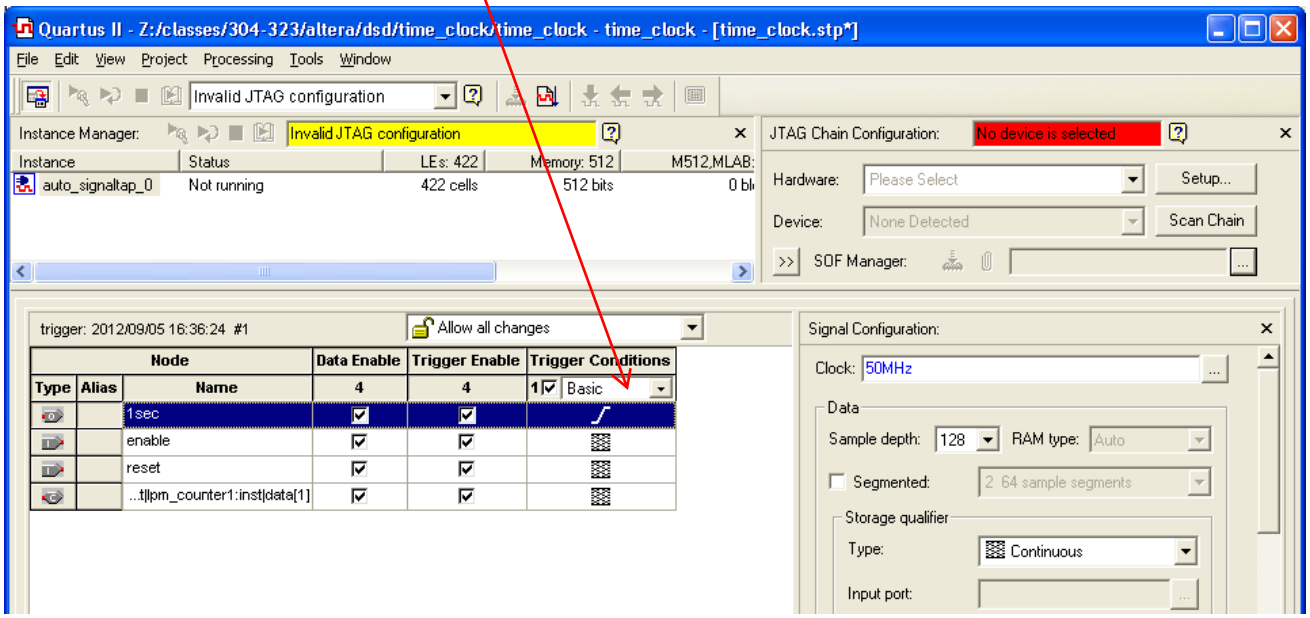
Make sure that the “Filter:” pane at the top has “SignalTap II: pre-synthesis” selected.

Do not add the clock input as this should be used as the sampling clock (specified by entering it into the “Clock” item in the “Signal Configuration” pane on the right hand side.

Once the nodes have been added, recompile the circuit. This will add in the logic analyzer circuitry to the design, alongside the timer circuit. You will have to recompile every time you make a change to the SignalTap II settings (e.g. to add in another signal to measure). Once compiled, download the design to the board using the Programmer.

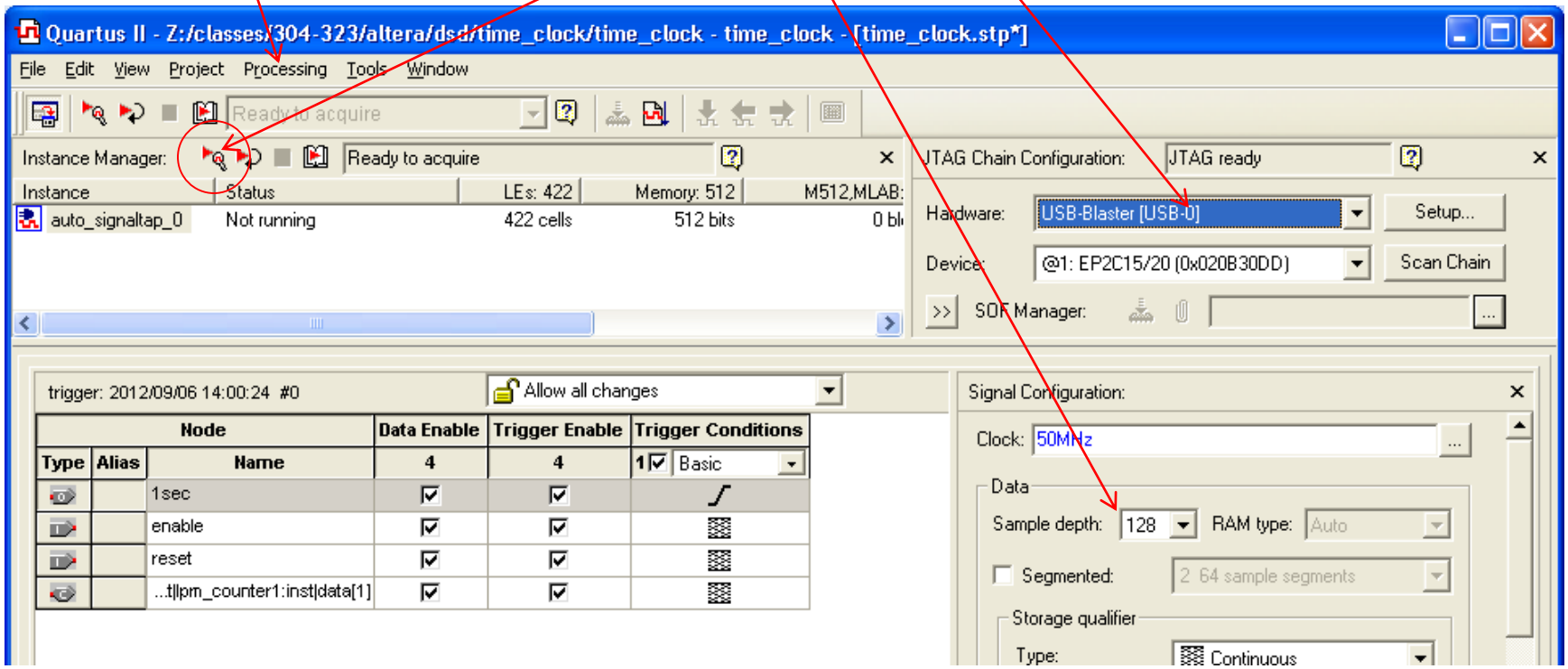
Once the design has been recompiled you can setup the triggering conditions, which will tell SignalTap II when it should start collecting measurements. This is done with the Setup tab selected.

Under “Trigger Conditions” select “Basic”. We will want to trigger the data capture when the single pulse enable goes high. Right-click on the trigger condition box in the row corresponding to the pulse output and select “rising edge”:

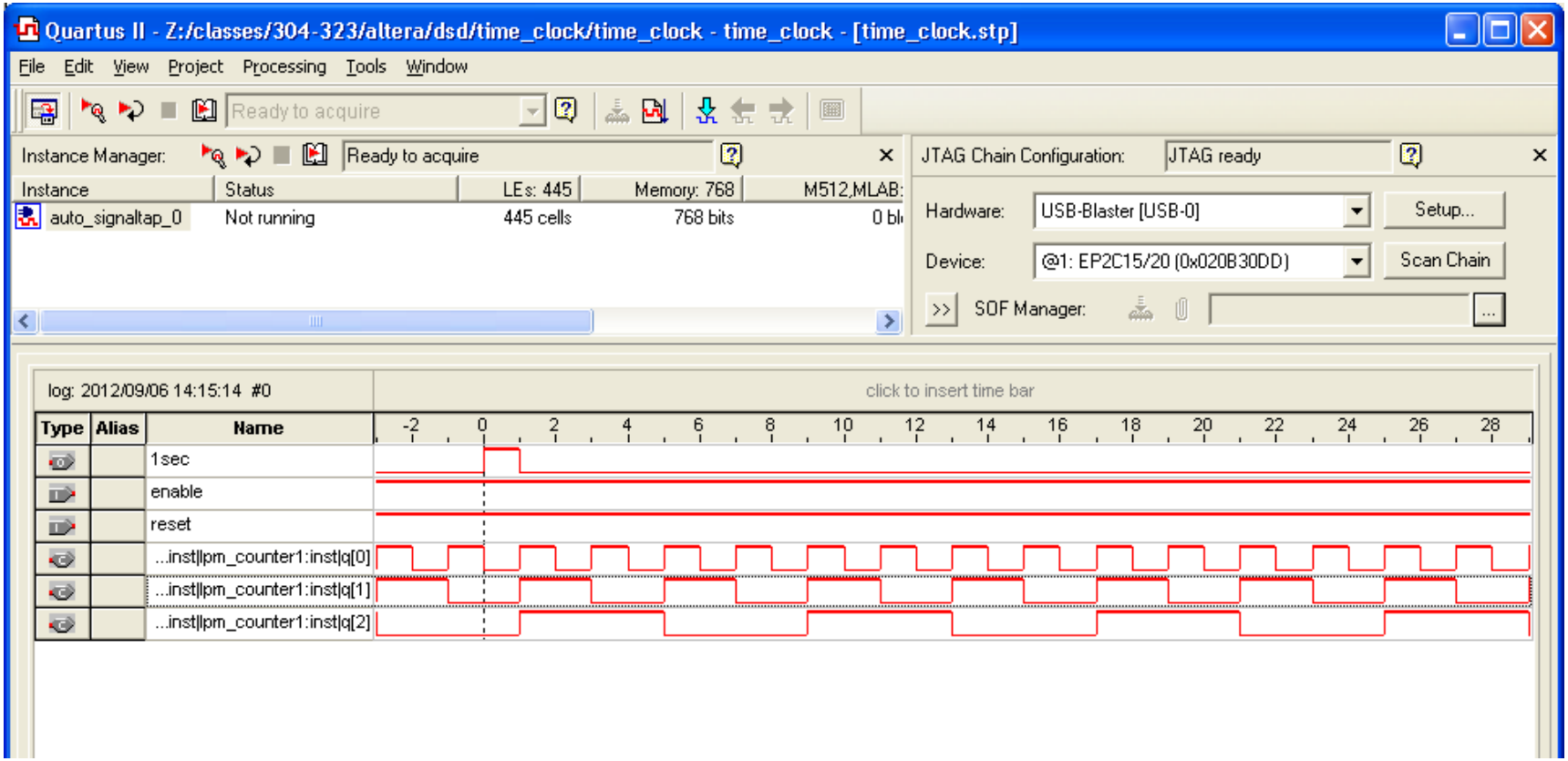


Next, set the Hardware to USB-Blaster. A connection will be made between the Quartus SignalTap II window and the on-chip logic analyzer circuitry and the analyzer will be ready to acquire data.

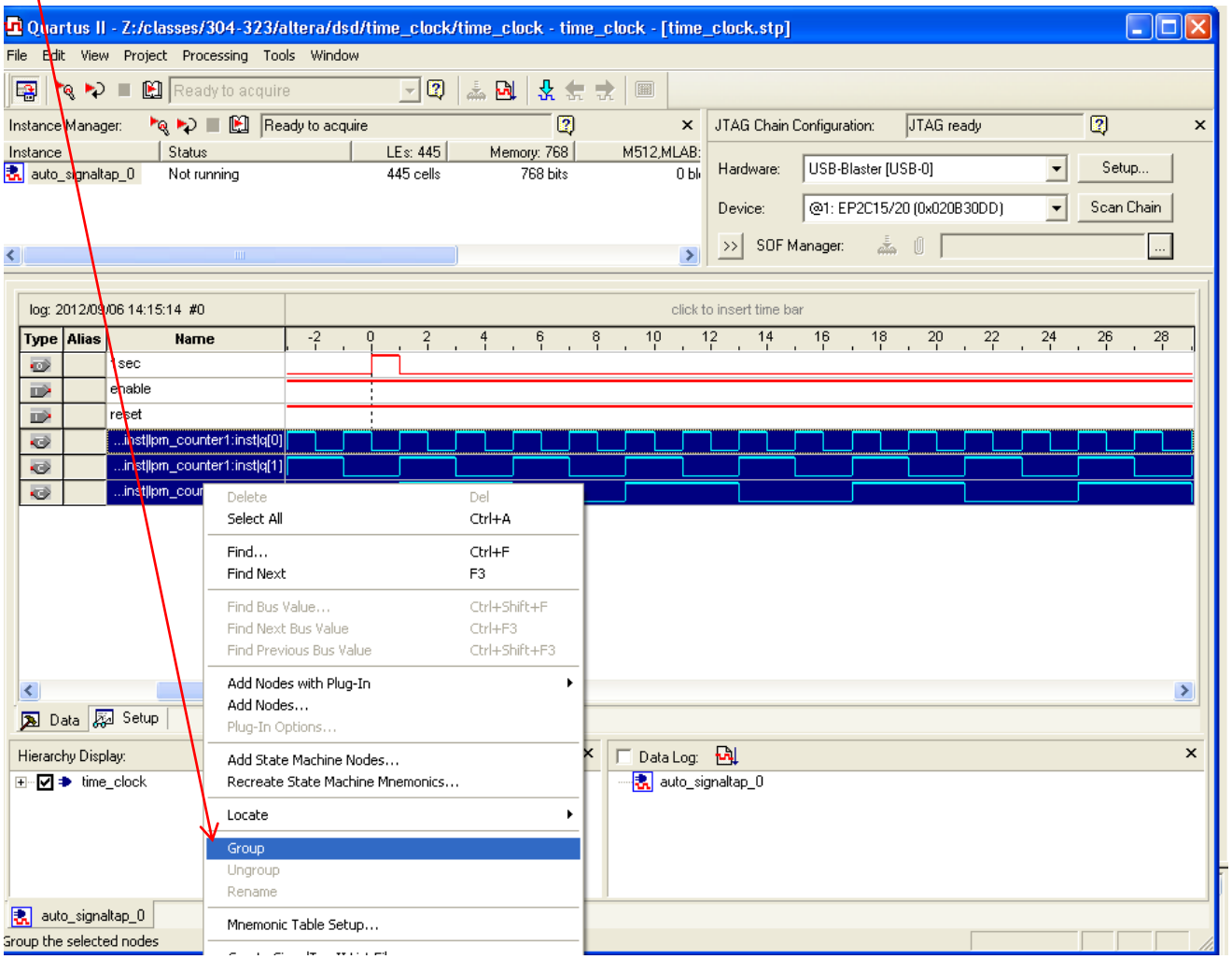
To actually acquire the data, press on the Run Analysis button, or select Processing/Run Analysis from the menu bar. The analyzer will capture 128 samples, set by the value in the Sample Depth box.



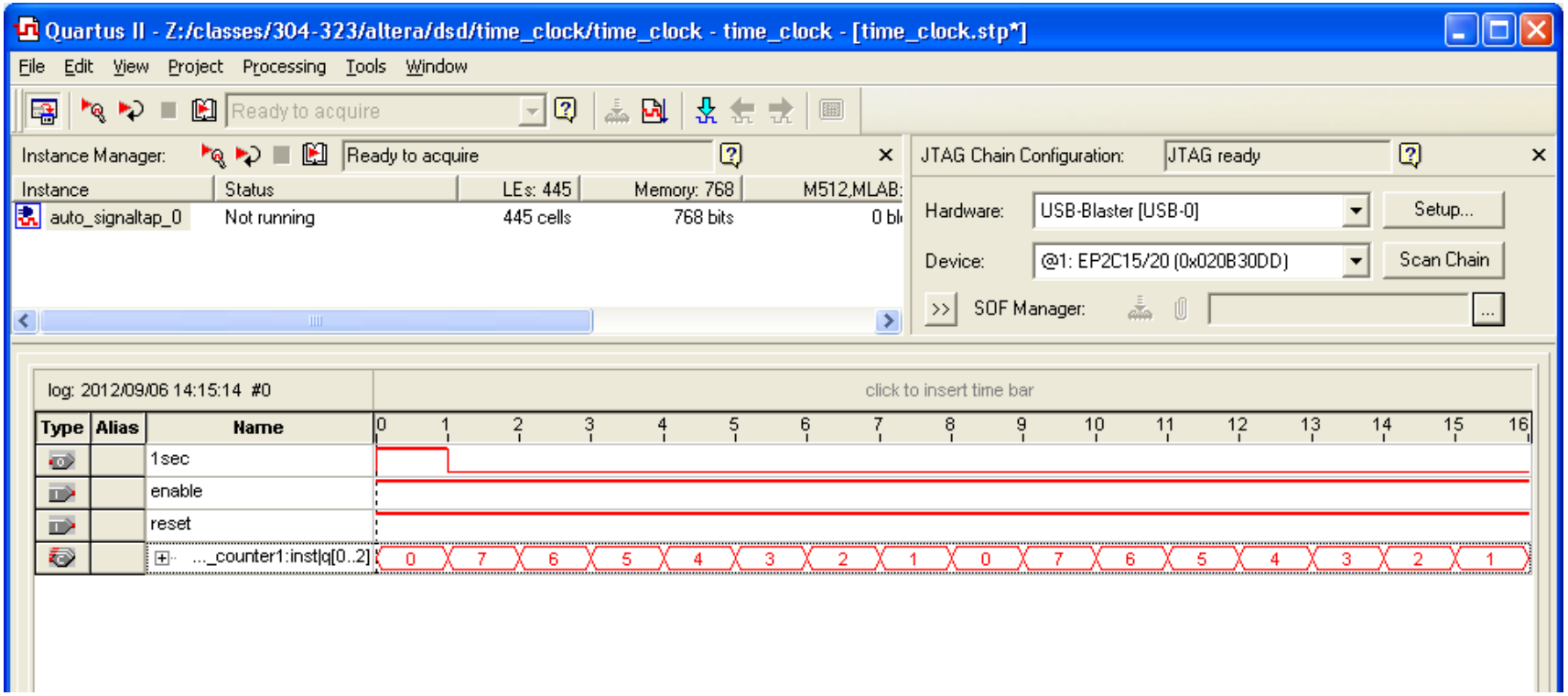
You should get a display like the one below.
You can zoom in and out by placing the cursor over the signal traces and left- or right-clicking (left zooms in, right zooms out).



If you have a set of signals that constitute a bus, you can group them together for convenience. To do this, select all the signals and then right-click. Select "Group" from the menu that pops up. Use this to view the VALUE output.



Once grouped, the bus values will be shown as below. You can set the format of the displayed values by right-clicking on the bus name and selecting the very last entry ("Bus Display Format"). Shown is the "Unsigned Decimal" format.



Show the resulting waveforms of your analyses to the TA showing how the VALUE output of the stack52 circuit changes in the various operation modes.

9. Writeup of the Lab Report .

Write up a short report, for the ***gNN_stack52*** circuit that you designed in this lab.

The report must include the following items:

- A header listing the group number (and company name if you gave it one), the names and student numbers of each group member.
- A title, giving the name (e.g. ***gNN_stack52***) and function of the circuit.
- A description of the circuit's function, listing its inputs and outputs. Provide a pinout or symbol diagram.
- The schematic diagram of the ***gNN_stack52*** circuit.
- A discussion of how the circuit was tested, giving details of the testbed and showing representative simulation plots, as well as any physical measurements you may have made.
- 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 report, and all associated design files 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 second laboratory session.

Grade Sheet for Lab #3

Winter 2017.

Group Number: _____.

Group Member Name: _____. Student Number: _____.

Group Member Name: _____. Student Number: _____.

Marks			TA Signatures
	1.	Schematic for the stack circuit	.
	2.	Simulation of the stack circuit	.
	3.	Schematic of the test-bed circuit	.
	4.	Show the timing analysis for the test-bed circuit	.
	5.	Simulation of the test-bed circuit	.
	6.	Floorplan of the test-bed circuit	.
	7.	Demonstration of the test-bed on the Altera board	.
	8.	Demonstration of SignalTap II analysis	.

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.