ECSE-323
Digital System Design

Lab #3 – *Sequential Circuit Design* Winter 2014

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. You will learn how to use the Altera SignalTapII in-circuit logic analyzer.

You will design a circuit to generate the basic timing pulses for the Earth time clock and the Mars time clock, as well as a test-bed for this circuit. Finally, you will design a circuit that can count Hours, Minutes, and Seconds.

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
- Perform timing simulations of circuits mapped to hardware
- Use the SignalTapII logic analyzer to do in-circuit testing

Table of Contents

This lab consists of the following stages:

1. Design of the Earth-Mars basic timing circuit
2. Simulation of the Earth-Mars basic timing circuit
3. Design of a test-bed for the basic timing circuit
4. Programming (configuration) of the University Board FPGA
5. Testing of the Earth-Mars timer circuits on the Altera board
6. Design of an Hours-Minutes-Seconds counter
7. Testing of the HMS counter using the SignalTapII logic analyzer
8. Writeup of the lab report

1. Design of the Earth-Mars Clock Timer Circuit .

For the course project you will design a system that can tell the time on both Earth and Mars, given longitudes on each planet.

http://www.glendalenewspress.com/news/tn-gnp-me-keeping-a-watch-on-mars-time-2013103-002,0,7111129.photo

Thus, for example, if it is 12 Noon at the summit of *Mont Royal*, the clock will be able to say what time it is at the summit of *Olympus Mons* on Mars.

In this lab you will design, using VHDL with Altera library modules, a sequential circuit that produces the basic one second timing pulses for the Earth and Mars time clocks. Later labs will build on this and develop the complete system.

Mars time clock app for Android by Requio Web Design (http://msl-curiosity.requio.com/)

The first order of business in developing the Mars Clock is to design a clock frequency divider which will provide one pulse each second.

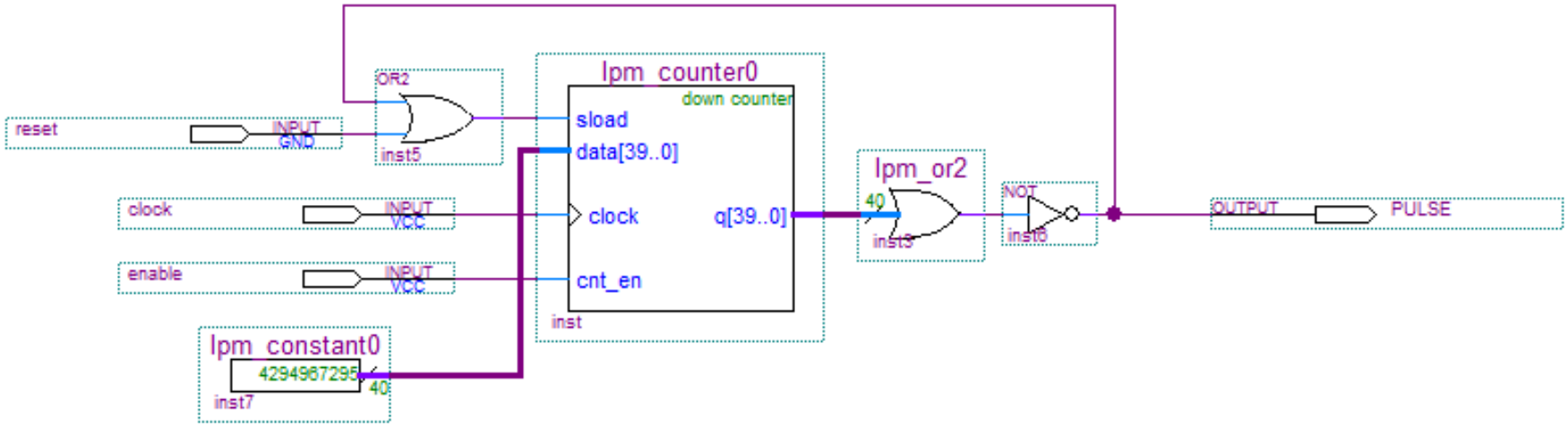
You will be using the Altera DE1 development board, which contains a clock generator which runs at 50 MHz (a period of 20 nsec). This is much faster than 1 cycle per second!

So you need to derive a time pulse which is much 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 DSD 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 <u>same</u> high speed clock. This produces a fully synchronous system.

Design of the synchronous frequency divider circuit



In this circuit, a counter (using the Altera *LPM_COUNTER* component) counts down to zero. Once it reaches zero, on the next positive clock edge it loads in a constant value, and then counts down from this value. The PULSE output will go high when the count value is equal to zero (the ***NOR*** gate detects this condition). ***The PULSE output will stay high for just one clock period.***

As is required in synchronous design, the PULSE 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, in VHDL, that will produce a repetitive pulse with a period of 1 second. This will produce the basic time interval for the Earth clock. Name the output of the timer ***EPULSE***. Use the Altera LPM_COUNTER module as a component to implement the count operations.

You will have to compute the appropriate constant value to load in to give the required amount of division and determine how many bits the counter should have. These will be different than in the example shown in the previous page. ***Don't be afraid to use a large number of bits - the Cyclone II FPGA on the DE1 board has a lot of flipflops!***

Note that the divider will run off of a *50 MHz* clock. Use this frequency in computing the division factor for your timer.

Once you have the design of the Earth timer, design a circuit that will produce a repetitive pulse with a period of 1 *Martian* second. This will produce the basic time interval for the Mars clock. Name the output of this timer ***MPULSE***.

What is the length of a Martian second, you ask?

A second on Mars is NOT the same as a second on Earth! It is just a little bit longer!

NASA uses the following approach for the clocks on their Mars rovers - they divide the Martian day, which is 24 hours, 39 minutes and 35.244 seconds long using Earth time units, into 24 equal length Martian hours, each of which is divided evenly into 60 Mars minutes, which in turn are divided evenly into 60 Mars seconds.

Thus, a *Martian second* has a length of
(24*3600+39*60+35.244)/(24*3600) = **1.027491252** *Earth seconds*.
The constant to be loaded into the Mars time counter will therefore need to be greater than that for the Earth time counter by this factor.

Write the module using VHDL, calling the circuit ***gNN_Basic_Timer.***

Show your VHDL description to the TA.

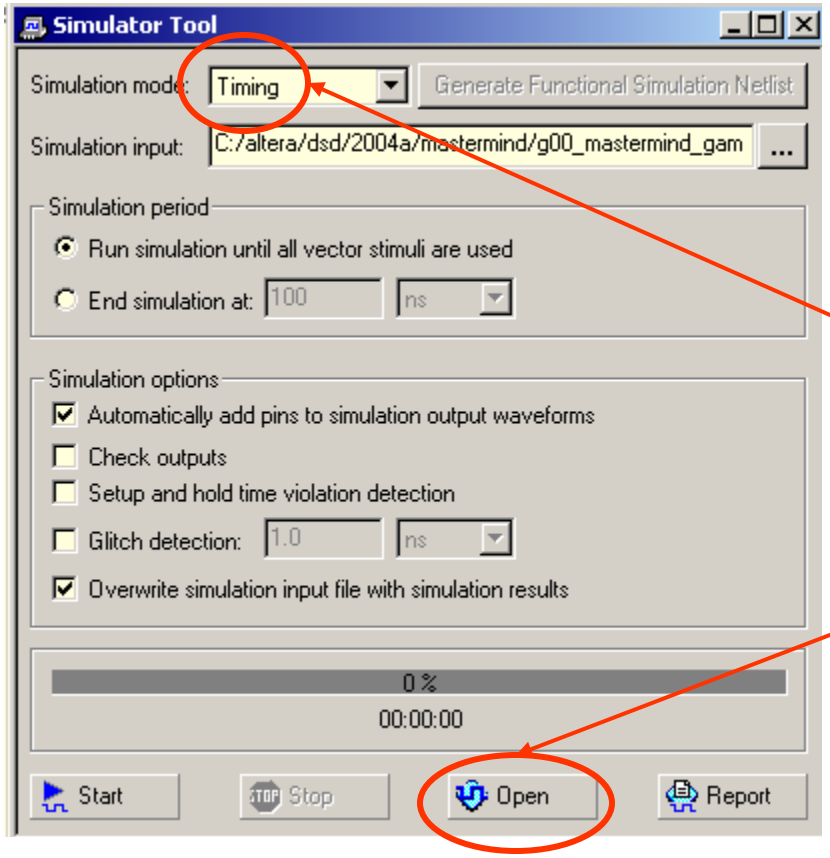
2. Simulation of the Earth-Mars Clock Timer Circuit .

The next step is to simulate your counter. First compile the circuit, and create a symbol for it. Then carry out the simulation.

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 at least 5 sec (so that you can see a few timer output pulses) then you will run through at least 250 million clock cycles! Running a simulation with this many clock cycles will take a long time.

So, just to make sure your timer is dividing the clock frequency properly, set the division ratio to something much smaller, say 1000, and run the simulation for only 100 usec. Your simulated circuit won't give the right division ratio in this case, but you will be able to see if your circuit properly acts as a divider.

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



Insert the symbol for your timer circuit 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 100 usec.

Then, click on "Start" in the simulator window to run the simulation.

Show the results of the simulation to your TA, showing the slow divergence of the two pulse trains.

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 is the propagation delay of the EPULSE and MPULSE outputs 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 a Timer Test-Bed .

You will now design a ***test-bed*** for your Earth-Mars timer 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.

You will test the Earth-Mars timer by using its outputs (EPULSE and MPULSE) as ***clock enable*** signals for two ***0-59*** up-counters, one enabled by the EPULSE signal, the other by the MPULSE signal.

The ***0-59*** counters should be implemented (in VHDL) as a cascade of a ***0-9*** up-counter (with a 4-bit count output) followed by a ***0-5*** up-counter (with a 3-bit count output). The counting of the 0-9 counter should be enabled by the MPULSE or EPULSE signals, while the counting of the 0-5 counter should be enabled by the ANDing of the MPULSE (or EPULSE) signals with a signal that goes high when the 0-9 counter reaches a value of 9.

Note that ALL counters should be driven by the 50 MHz master clock.

Design of a Timer Test-Bed (cont)

The Altera DE1 board has four 7-segment LEDs that can be used to display the 0-59 count values for both the Martian and Earth timing pulses.

The 0-5 and the 0-9 counter outputs should be input to four separate ***gNN_7_segment_decoder*** modules (designed in lab #2), which will generate the appropriate signals for driving the 7-segment LEDs on the Altera board.

Finally, all of the counters and the ***gNN_Basic_Timer*** circuit should have their RESET inputs tied together and connected to one of the pushbuttons.

Use VHDL to describe the circuit, using components to instantiate the timers, counters, and segment decoder modules.

Show the VHDL description of the testbed to the TA.

4. Testing the Timer.

As you did with the Timer circuit, do a timing simulation of the testbed circuit. To speed things up and save on simulation time, greatly decrease the division factors of the timers. Just remember to reset these to the proper values before downloading to the DE1 board.

Show the simulation results to the TA, demonstrating the effect of the reset and that your circuit correctly generates the proper LED signals.

Once you have simulated your test-bed and are satisfied that it functions properly, it is time to download the design onto the Altera DE1 board.

Demonstrate to the TA that your circuit is functioning properly on the DE1 board by showing counting up from 0 to 59, for both the Mars and the Earth timers.

Show the effect of pushing the RESET push button (which should synchronize the 2 counters).

The counters are running at slightly different rates, so their values will begin to diverge. Before beginning the test, make the following computations:

- Estimate how many Earth seconds it will take until the 2 count values are different.

- Estimate how many Earth seconds it will take before the counters are back in sync (showing the same value).

Test your predictions using the counter running on the DE1 board and show the results to the TA (along with how you made your predictions).

TIME CHECK

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

Take the next week off (the labs will be closed during the study break).

5. Design of an Hours-Minutes-Seconds Counter .

Using VHDL, modify the test-bed module to make a counter circuit that counts Hours, Minutes, and Seconds. The seconds and minutes counts should range from 0 to 59, and the Hours count should range from 0-23.

The Seconds counter should increment whenever an enable input (call it *sec_clock*) is high (the enable signal should only be high for one clock cycle). The Minutes counter should increment whenever the Seconds count goes from 59 back to 0. The Hours counter should increment whenever the Minutes count goes from 59 back to 0.

You have already done most of the work in designing the testbed for the timer module (i.e. the 0-59 counters can be re-used to serve as the Seconds and Minutes counters). You just need to add in the Hours counter.

The VHDL design entity should be named ***gNN_HMS_Counter*** (where NN is the number of your group) and should have the following inputs and outputs:

- **clock** (1-bit asynchronous input, should be connected to the master 50MHz clock)
- **reset** (1-bit asynchronous input, when high the counts are all set to zero)
- **sec_clock** (1-bit synchronous input, a pulse with a period of 1 second, the counter should be incremented when this input goes high)
- **count_enable** (1-bit synchronous input, the incrementing of the counter will only happen if this is high)
- **load_enable** (1-bit synchronous input, if high the count values will be set to the values at the H_Set, M_Set, and S_Set inputs. Loading should have priority over incrementing)
- **H_Set, M_Set, S_Set** (5-bit, 6-bit, 6-bit synchronous inputs, these values are copied to the count outputs when load_enable is high)
- **Hours, Minutes, Seconds** (5-bit, 6-bit, 6-bit synchronous outputs, corresponding to the count values)
- **end_of_day** (1-bit synchronous output, which should go high for one clock cycle when the Hours count goes from 23 to 0)

The *sec_clock* signal should not be used as an asynchronous clock signal! Use the *clock* signal for this, since you should adhere to a strict synchronous design methodology, which means that all of your sequential elements must be clocked with the same clock signal (i.e. the 50MHz clock). It should be used to enable counting, just as was done in the timer testbed circuit.

Show the VHDL description of the HMS counter to the TA.

6. Simulation of the HMS Counter .

Compile your HMS counter circuit, and do a functional simulation. To save on simulation time, you can use a fast pulse signal for the *sec_clock* input, say with a period of one microsecond (no need to wait for a whole second). Run the simulation for at least one simulated day.

Show the simulation results to your TA.

For the hardware testing on the DE1 board, you won't make a testbed with the LED displays at this time. Instead, you will make use of a *logic analyzer* tool called SignalTap II to inspect the operation of the hardware.

TIME CHECK

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

7. 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 will be plenty of free resources left over so that you should 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.

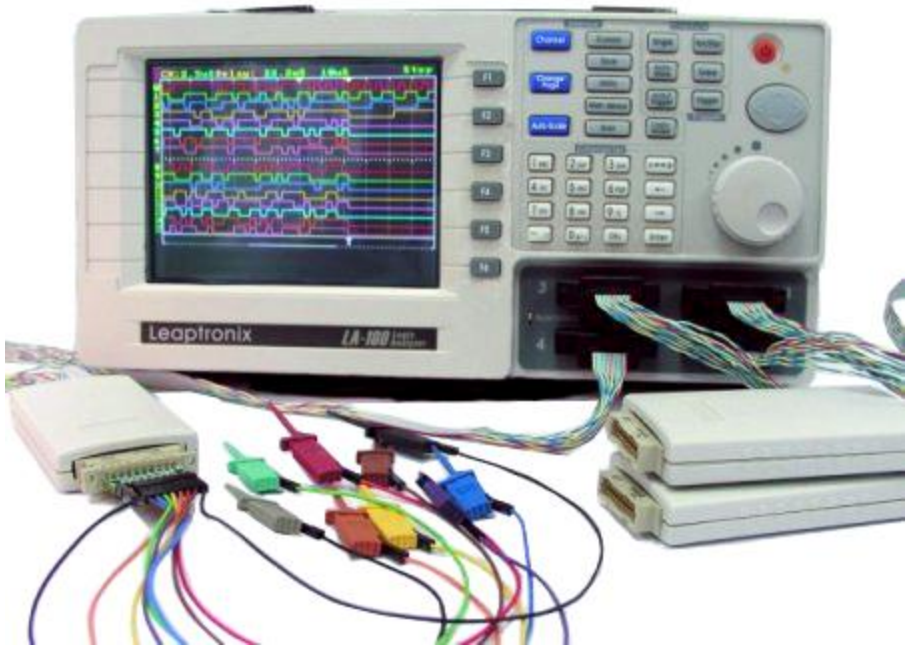
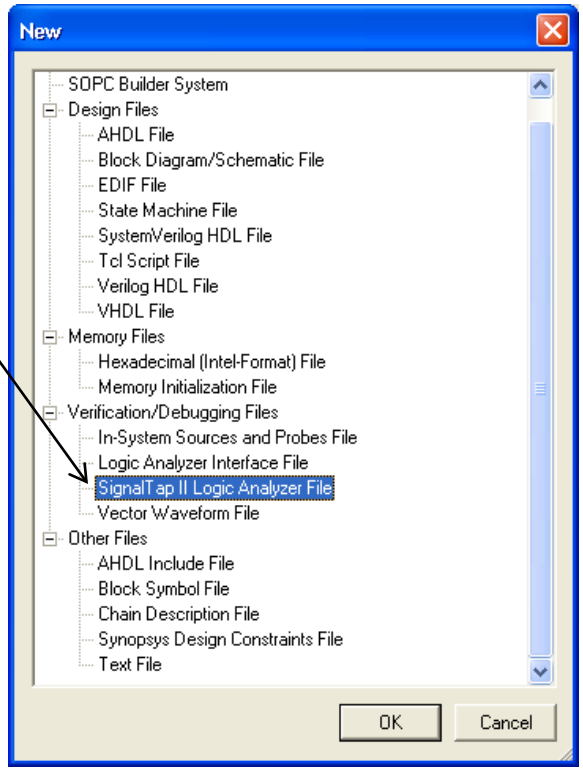


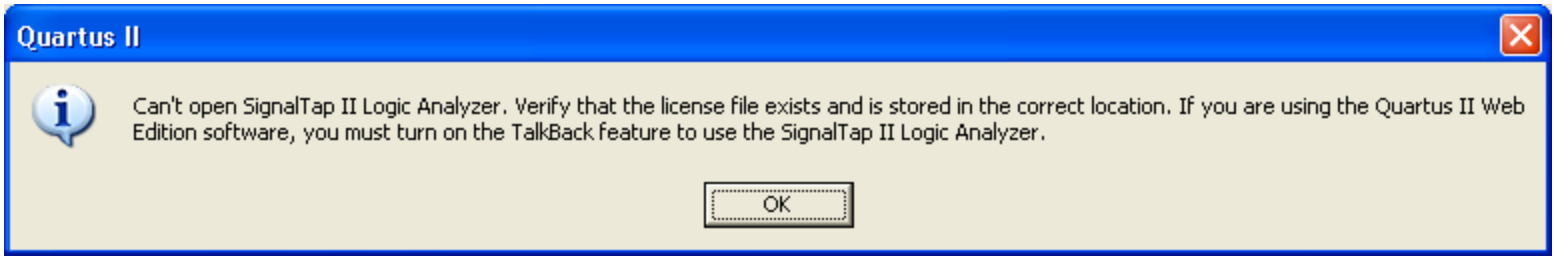
Image from http://www.elexp.com/tst_p100.htm

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.

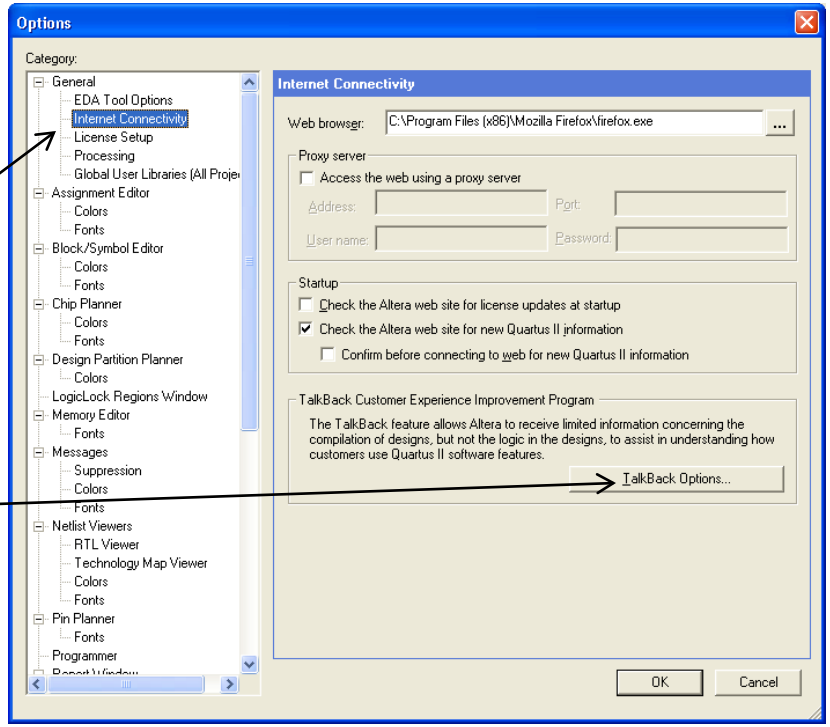


You may get the following error message popup (especially if you are using the web version of Quartus on your own computer:

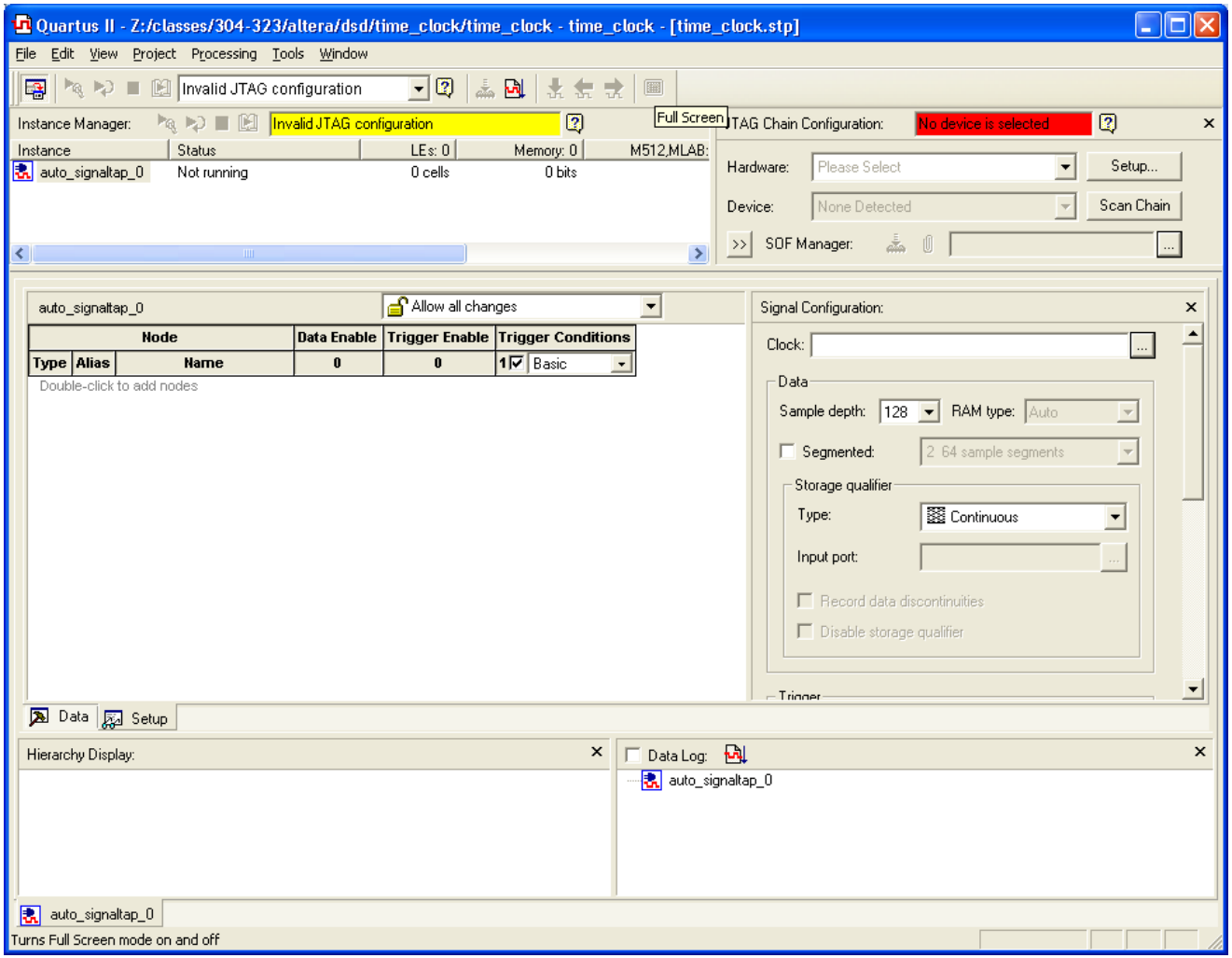


If so, turn on TalkBack by going to Tools/Options and selecting the Internet Connectivity tab. Then click on TalkBack Options and turn on the TalkBack Feature in the window that pops up.

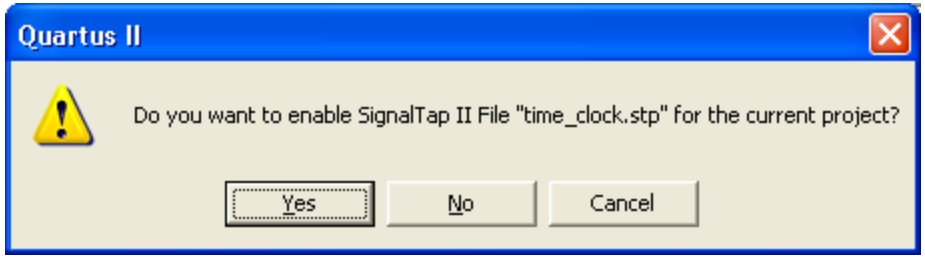
You will have to restart Quartus.



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

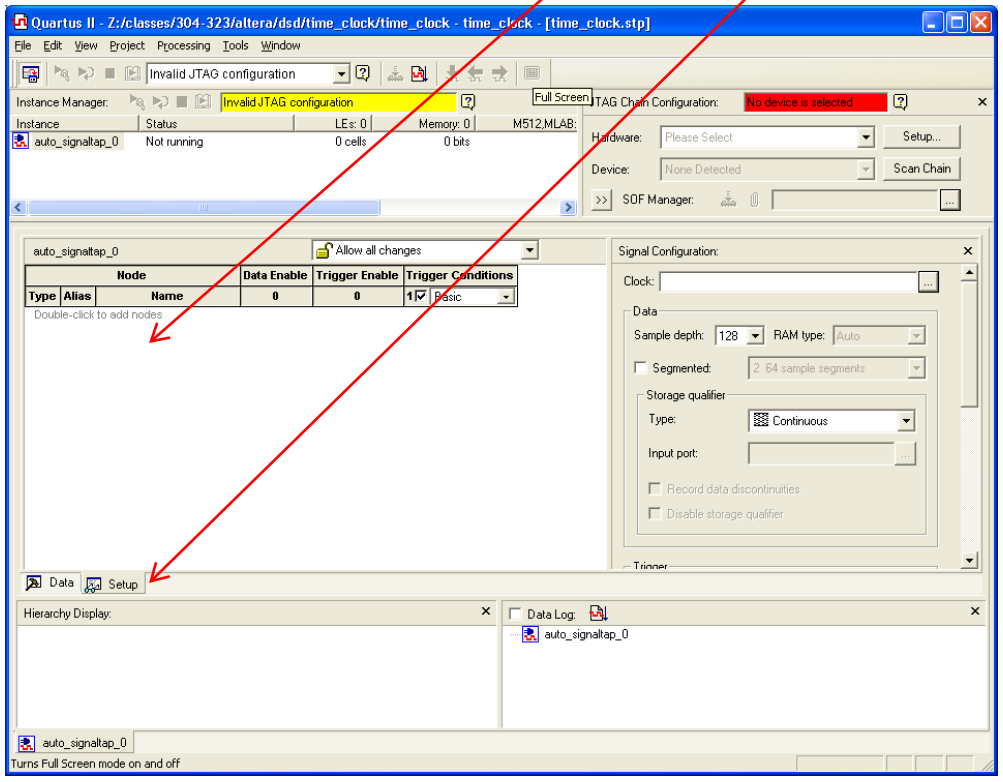


Before doing anything else, save the file (File/Save) giving the name "gNN_HMS_counter.stp". In the window that pops up, say "Yes". This enables the SignalTap II file to be active for the current project.



The next thing to do is to specify the circuit nodes to be analyzed. For this tutorial, we will look at the output signal of the envelope 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 inputs of the gNN_HMS_counter module.

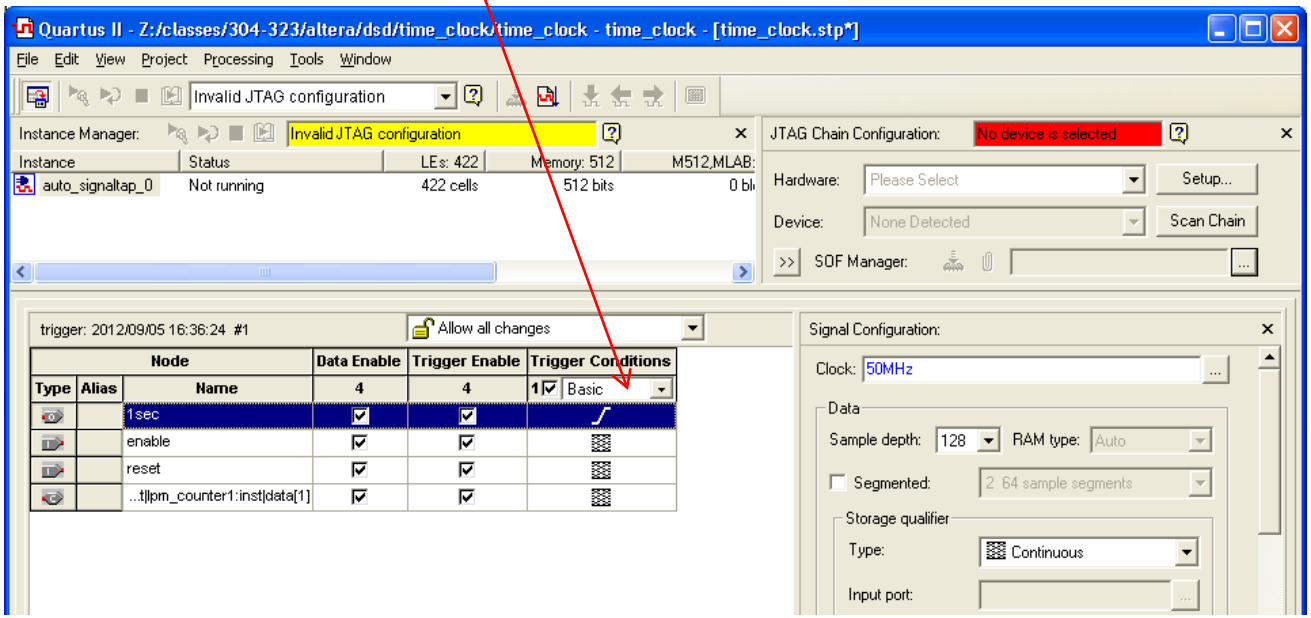
Make sure that the “Filter:” pane at the top has “SignalTap II: pre-synthesis” selected.

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.

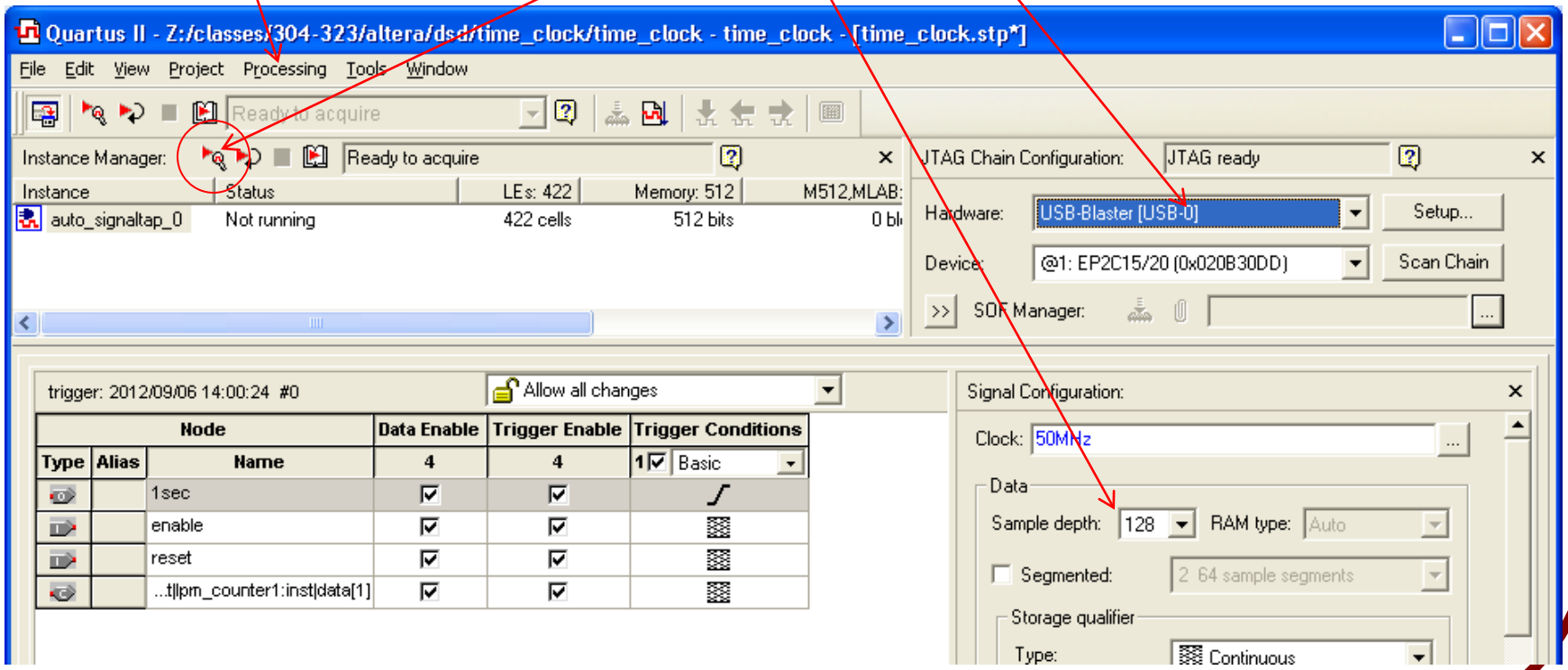
Once the design has been recompiled you can setup the triggering conditions, which will tell SignalTap II when it should start collecting measurements. This is done with the Setup tab selected.

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

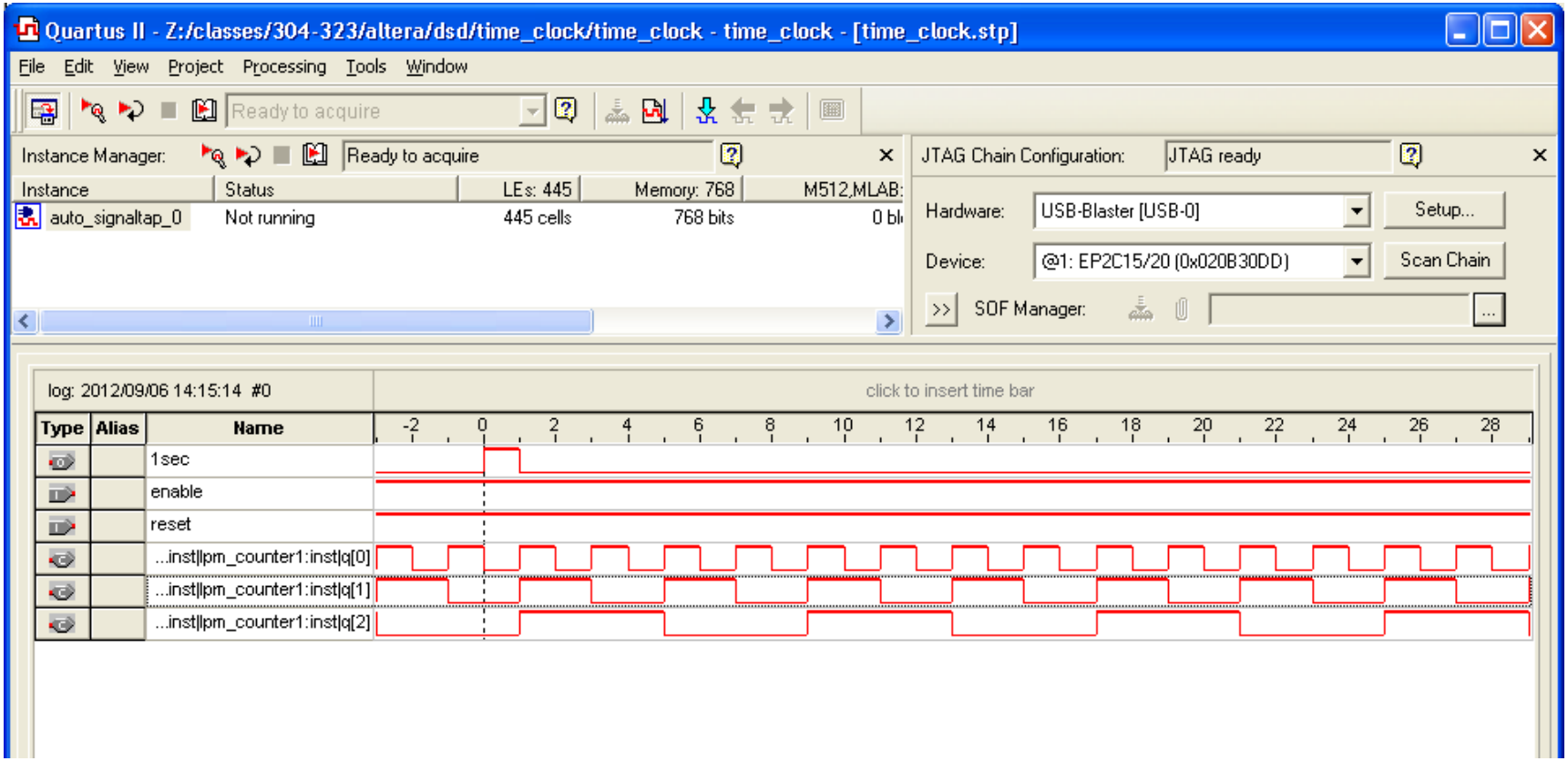


Next, set the Hardware to USB-Blaster. A connection will be made between the Quartus SignalTap II window and the on-chip logic analyzer circuitry and the analyzer will be ready to acquire data.

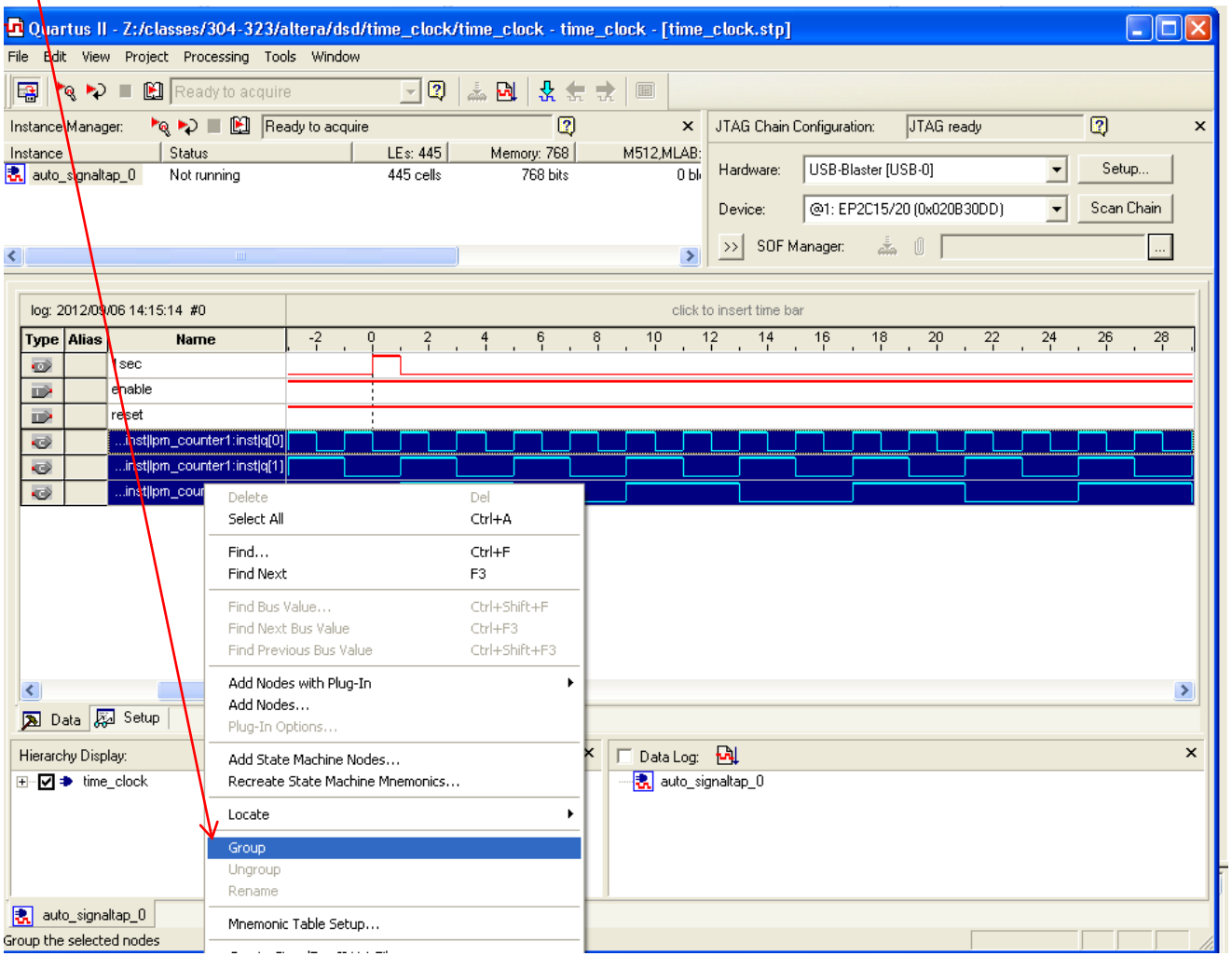
To actually acquire the data, press on the Run Analysis button, or select Processing/Run Analysis from the menu bar. The analyzer will capture 128 samples, set by the value in the Sample Depth box.



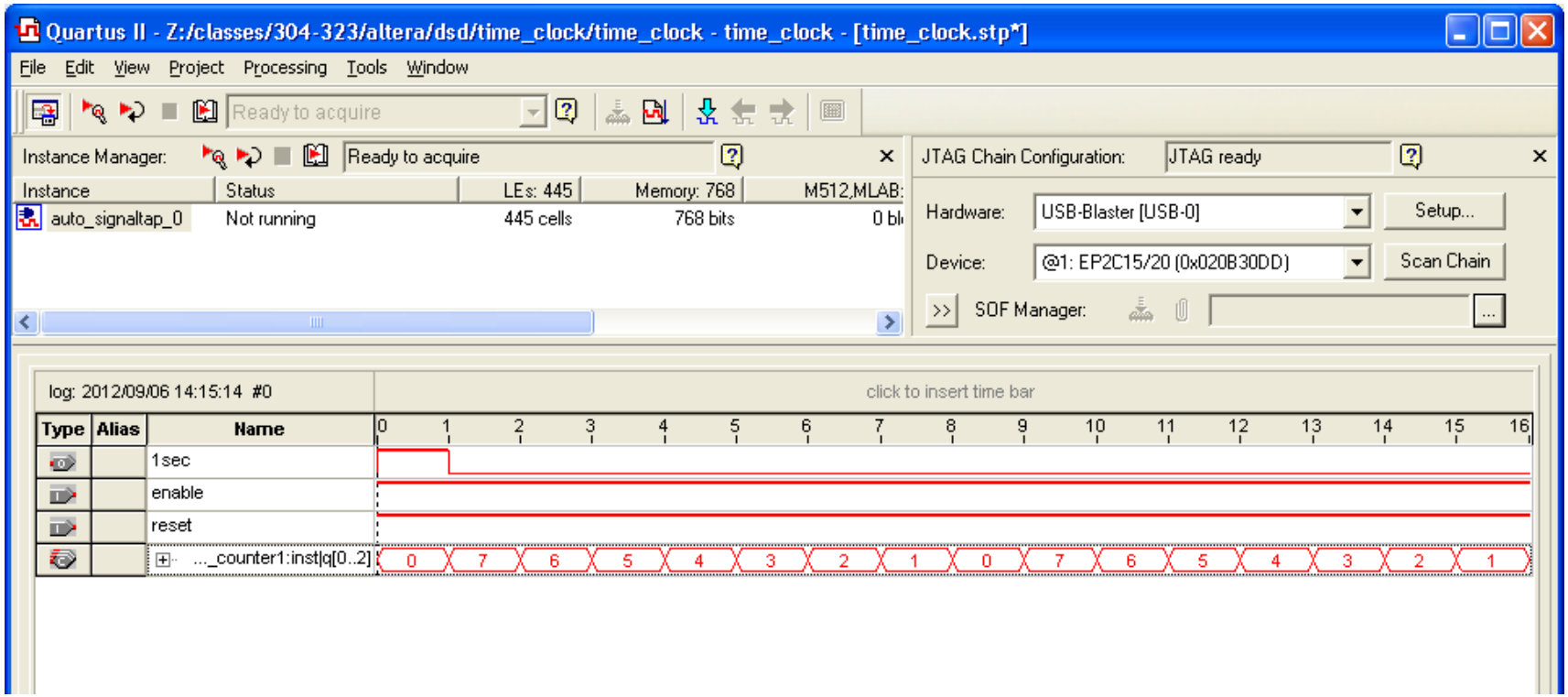
You should get a display like the one below.
You can zoom in and out by placing the cursor over the signal traces and left- or right-clicking (left zooms in, right zooms out).



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

TIME CHECK

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

8. Writeup of the Lab Report .

Write up a short report for the complete ***gNN_Basic_Timer*** circuit that you designed in this lab (you do not need to write a report for the gNN_HMS_counter at this time).

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_Basic_Timer***) and function of the circuit.
- A description of the circuit's function, listing the inputs and outputs. Provide a pinout or symbol diagram.
- The schematic diagram of the ***gNN_Basic_Timer*** circuit.
- A discussion of how the timer 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 report should be done in pdf and uploaded to the myCourses site using the assignment 3 submission.

Make sure that you have uploaded the report and ***all*** of the design files (e.g. .bdf and .vhd files) used in your project packaged together in a single zip file.

The report is due one week after the last day of the lab period, or Friday, March 21.

Grade Sheet for Lab #3

Winter 2014.

Group Number: _____.

Group Member Name: _____. Student Number: _____.

Group Member Name: _____. Student Number: _____.

Marks			TA Signatures
	1.	VHDL description for the Earth-Mars timer circuit	
	2.	Simulation of the Earth-Mars timer circuit	
	3.	VHDL description of the timer test-bed circuit	
	4.	Simulation of the timer test-bed circuit	
	5.	Demonstration of the timer test-bed on the Altera board	
	6.	VHDL description for the Earth-Mars timer circuit	
	7.	Simulation of the Earth-Mars timer circuit	
	8.	SignalTapII analyzer waveforms	

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.