ECSE-323
Digital System Design

Lab #3 – *Sequential Circuit Design* ***Fall*** 2018

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.

You will create a add/subtract calculator on the FPGA board with display and user inputs.

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 8-bit full adder and subtractor.
2. Design of 8-bit add/subtract calculator circuit with a 32-bit register file.
3. Pin mapping for inputs and outputs, such as switches, push button and LEDs.
4. Simulation and testing on FPGA board.

Design of adder and subtractor.

Implement a 8-bit full adder and subtractor.

The 8-bit full adder VHDL description is found in the vhdl_1 basics lecture notes on the course website.

The full subtractor can be implemented by modifying the full adder or use 1-bit subtractors from lab 1. The operator is changed from “+” to “-”.

Implement the code and insure that it compiles.

Design of Regsiter

A register is a data structure that consists of a set of memory cells which have common control lines and a shared address.
An 8-bit register would have an input (8 bits wide) where each of the input lines would drive a particular memory cell. The 8-bit register would have an output (8 bits wide) where each output would be driven by the contents of a particular memory cell.
The register memory cells have a common clock and are typically all constructed using the same type of flip-flop.
The next slides will provide the VHDL code for a 4-bit register using DFF.

The Entity of 4-bit register

```
1  -- Entity Description 4-Bit register
2  library IEEE;
3  use IEEE.std_logic_1164.all;
4
5  entity register_ex_1 is
6  port (d_out: out std_logic_vector(3 downto 0);
7        d_in : in std_logic_vector(3 downto 0);
8        clk, reset, enable   : in std_logic);
9  end register_ex_1;
10
11
```

The Architecture of 4-bit register

```
architecture example of register_ex_1 is
signal internal: std_logic_vector(3 downto 0);
begin
-- creat process with a sensitivity list
    process(clk, reset, enable)
    begin
        if reset ='1' then
            --Asynch reset for data
            internal <= (others=>'0');
        elsif rising_edge(clk) then
            if enable='1' then
                internal <= d_in;
            end if;
        end if;
    end process;
--update the output ports
d_out <= internal;
end example;
```

A 2 4-bit register file

```
4  entity register_ex_2 is
5  port (d_out: out std_logic_vector(7 downto 0);
6        d_in : in std_logic_vector(3 downto 0);
7        clk, reset, enable, sel_r   : in std_logic);
8  end register_ex_2;
9
10 architecture example of register_ex_2 is
11 signal internal : std_logic_vector(7 downto 0);
12 begin
13     -- creat process with a sensitivity list
14     process(clk, reset, enable, sel_r)
15     begin
16         if reset ='1' then  --Asynch reset for data
17             internal <= (others=>'0');
18         elsif rising_edge(clk) then
19             if enable='1' then
20                 if sel_r ='1' then -- Choose lower four bits when select right = 1
21                     internal(3 downto 0) <= d_in;
22                 else -- Choose upper four bits when select right = 0
23                     internal(7 downto 4) <= d_in;
24                 end if;
25             end if;
26         end if;
27     end process;
28 d_out <= internal; --update the output ports
29 end example;
```

You can modify previous code to create a VHDL circuit that contains 4 8-bit wide register file (32 bit in total).

R0 and R1 registers will be used to store user inputs.
R2 and R3 registers will be used to store output from adder and subtractor.

You will have to design the control logic of when and how each register is accessed (read/write).

Your circuit should have following list ports to allow user input, selection of mode, and display of results.

```
4  entity lab3_top is
5  port ( clk, reset, enable: in std_logic;
6           wr_reg : in std_logic; -- wr_reg='0' to write R0, wr_reg='1' to write R1
7           user_input: in std_logic_vector(7 downto 0); -- user input
8           add_mode: in std_logic; -- '1' for add operation, '0' for subtract operation
9           data_out: out std_logic_vector(7 downto 0);-- sum or diff to be displayed on LEDs
10          bit_out: out std_logic -- cout or bout to be displayed on a LED or segment
11          );
12 end entity;
13
14 architecture example of lab3_top is
15 -- the 4 8-bit register memory
16 signal internal: std_logic_vector(31 downto 0);
```

Integrating adder and subtractor

The 8-bit adder and 8-bit subtractor will be integrated into the circuit to calculate the sum or difference.

In add mode, the sum of R0 and R1 will be stored in R2, and the carry out will be store in the LSB of R3. The carry out and sum should also be shown on LEDs(on as ‘1’ and off as ‘0’).

In subtractor mode, the difference and borrow out of R0 and R1 will be stored and displayed in the same manner.

Integrating adder and subtractor

To import other VHDL circuits in the same project directory, you need to instantiate components in current VHDL file.

The following example show the instantiation of a 8-bit adder and 8-bit subtractor.

```
14  architecture example of lab3_template is
15  -- the 4 8-bit register memory
16  signal internal: std_logic_vector(31 downto 0);
17  -- the adder
18  component adder is
19  port(a, b: in std_logic_vector(7 downto 0);
20       sum: out std_logic_vector(7 downto 0);
21       cout: out std_logic);
22  end component;
23  -- the subtractor
24  component subtractor is
25  port(a, b: in std_logic_vector(7 downto 0);
26       diff: out std_logic_vector(7 downto 0);
27       bout: out std_logic);
28  end component;
29  -- the internal signal
30  signal sum, diff: std_logic_vector(7 downto 0);
31  signal cout, bout: std_logic;
32  begin
33      add: adder port map(a=>internal(7 downto 0),b=>internal(15 downto 8),sum=>sum,cout=>cout);
34
35      sub: subtractor port map(a=>internal(7 downto 0),b=>internal(15 downto 8),diff=>diff,bout=>bout);
```

Control Logics

Next, you need to implement the control logic for mode of operation (add/subtract), content to display(sum/diff) , and user input to registers. This needs to be done in process, where control logics such as “if”, “when”, and “case” are allowed.

```
33      add: adder port map(a=>internal(7 downto 0),b=>internal(15 downto 8),sum=>sum,cout=>cout);
34
35      sub: subtractor port map(a=>internal(7 downto 0),b=>internal(15 downto 8),diff=>diff,bout=>bout);
36
37      -- control logics
38      process(clk, reset, enable, wr_reg, add_mode)
39      begin
40          if reset ='1' then  --Asynch reset for data
41              -- asynchronous reset
42          elsif rising_edge(clk) then
43              if enable='1' then
44                  if wr_reg='0' then
45                      -- write data to register R0
46                  else
47                      -- write data to register R1
48                  end if;
49              end if;
50
51              if add_mode='1' then
52                  -- save sum to R2 and cout to R3(0), update display output as wells
53              else
54                  -- save diff to R2 and bout to R3(0), update display output as wells
55              end if;
56          end if;
57      end process;
58  end architecture;
```

Pin Planning

Once your circuit is complete and compiles. You can simulate in VWF to see if it operates correctly and start pin planning to configure the connections to LEDs, switches and push buttons.

The configuration is done in Pin Planner (Quartus=> Assignments => Pin Planner)

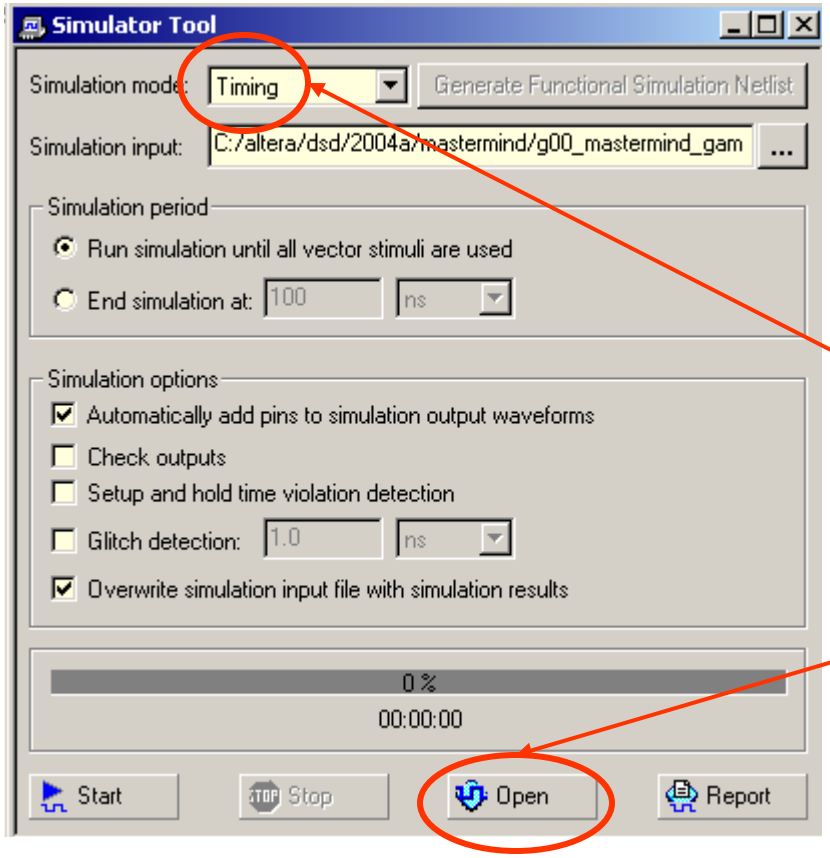
Go to section 3.6 of the DE1-SoC board user manual to find Pin No. of LEDs, switches and pushbutton.

There are 9 bits of output to display: 8 bits for sum/diff and 1 bit for cout/bout. You should display the 8-bit result in 8 consecutive LEDs on the board, with MSB on the left side. For the single bit, choose another LED.

You should also allow user to set R0 and R1 (operands for add/subtract) and set mode of operation. This can be done using switches on the FPGA board.

There should also be a button or switch for ‘enable’ signal to enable user input.

2. Timing Simulation of the 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.

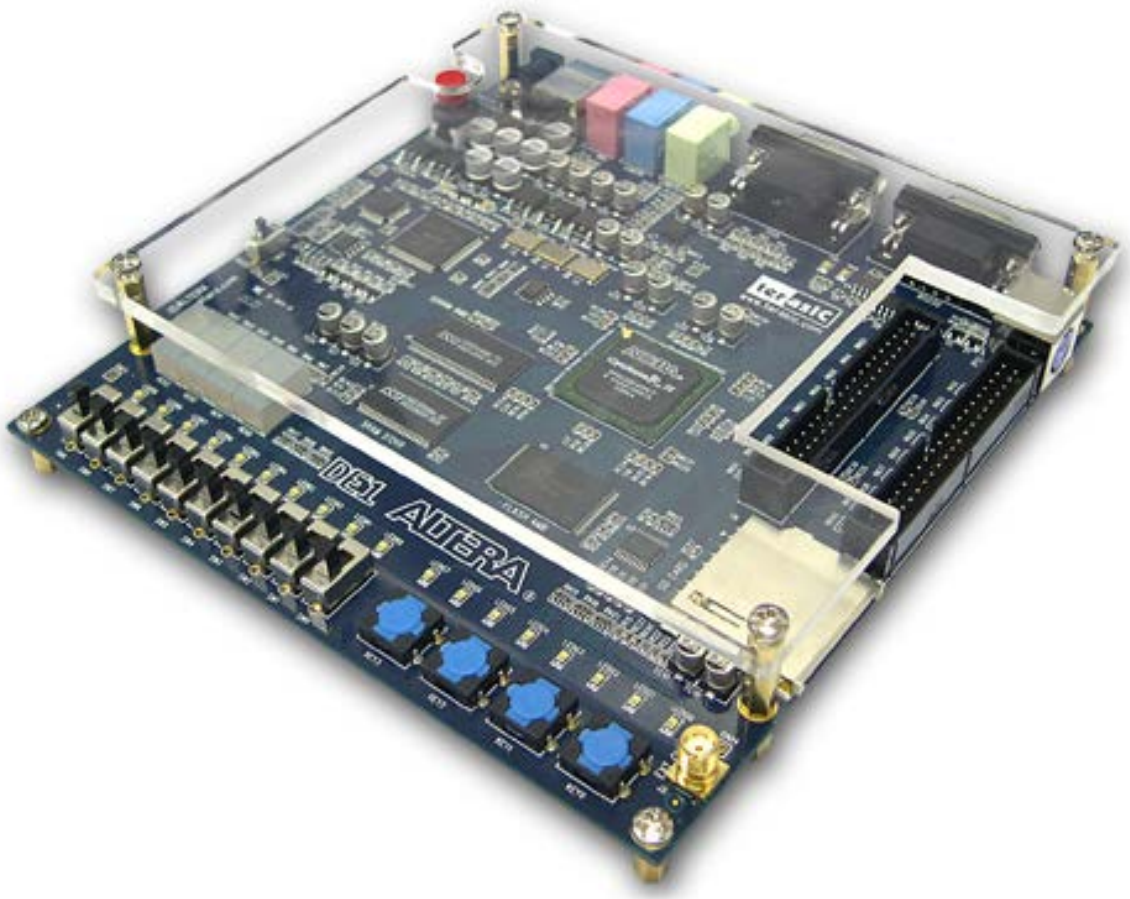
Show the results of the simulation to your TA. You should demonstrate the 4 different register being loaded (one at a time) the Reset operation, the operation of the adder and subtractor Be sure to display the register contents as output to show that it is changing appropriately.

3. 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 $125.

4. 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.

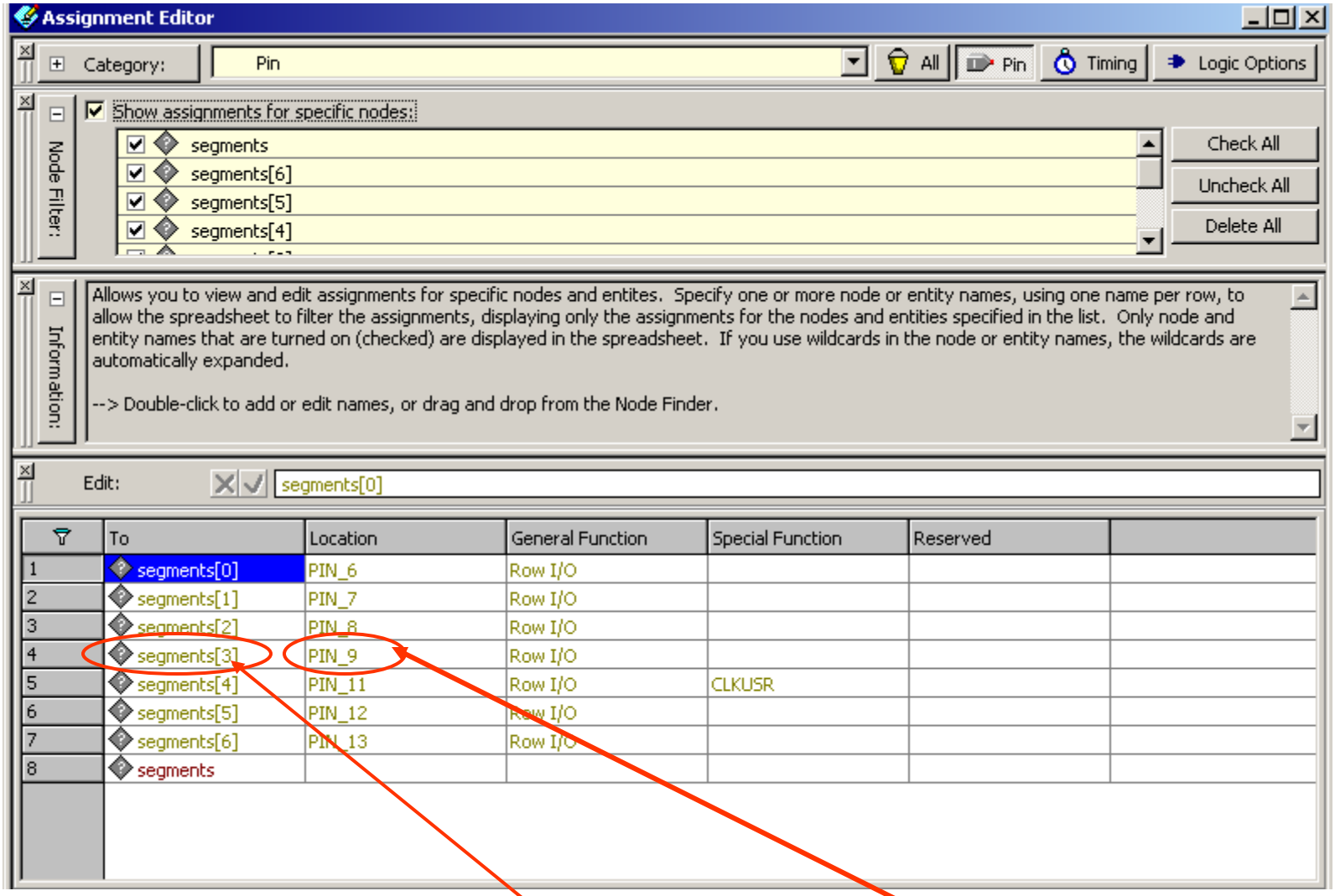
The mapping of the Altera Board's individual LED display segments to the pins on the Cyclone II device is listed in Table 4.3 on page 29 of the DE1 Development and Education Board Users Manual. The pin mappings for the other board components can also be found in section 4 of the manual.

You will also want to connect, for testing purposes, 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

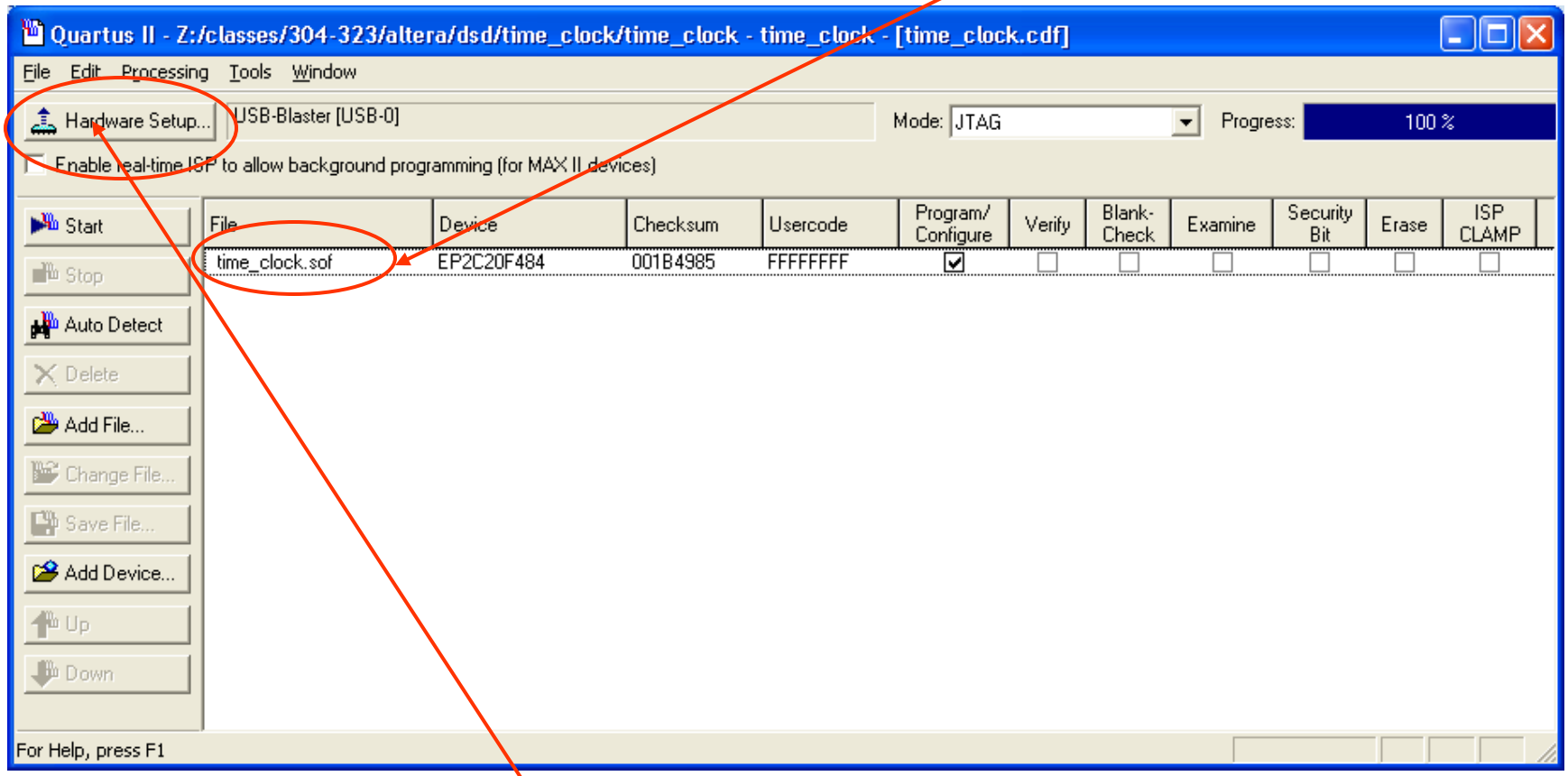
Once you have assigned all of the inputs and outputs of your circuit to appropriate device pins, ***re-compile*** your design (and remember to reset the division ratio used in your pulse generator circuit back to its correct value from the value you set for simulation!)

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.

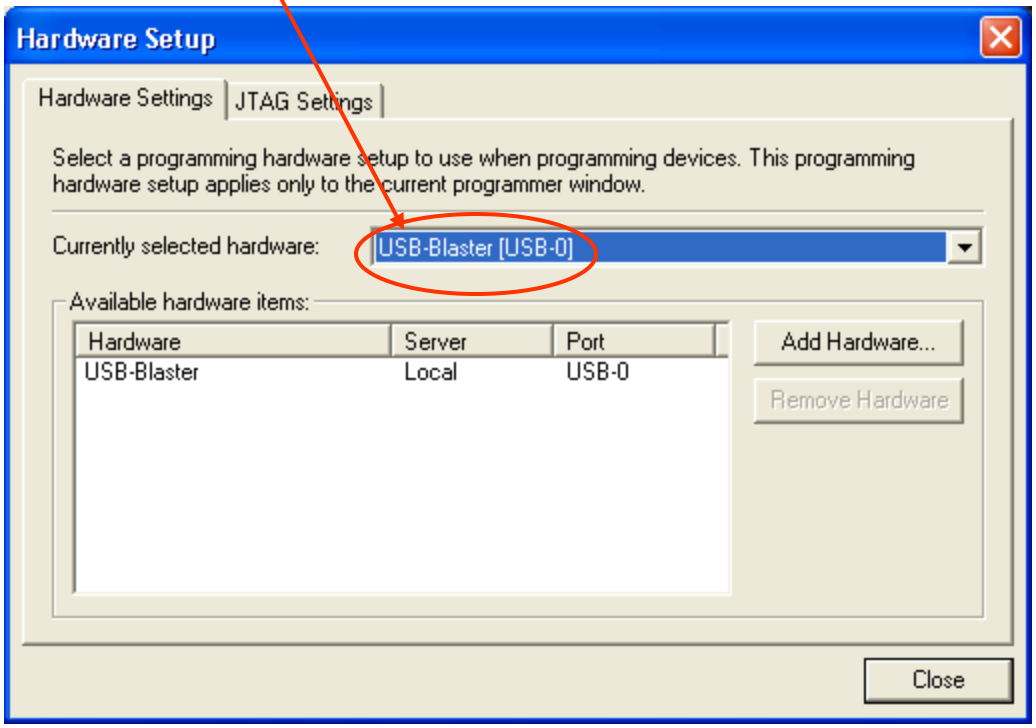
Next select the ***Programmer*** item from the ***Tools*** menu. You should see a window like the one shown below. There should be a file listed. If not, click "Add File".



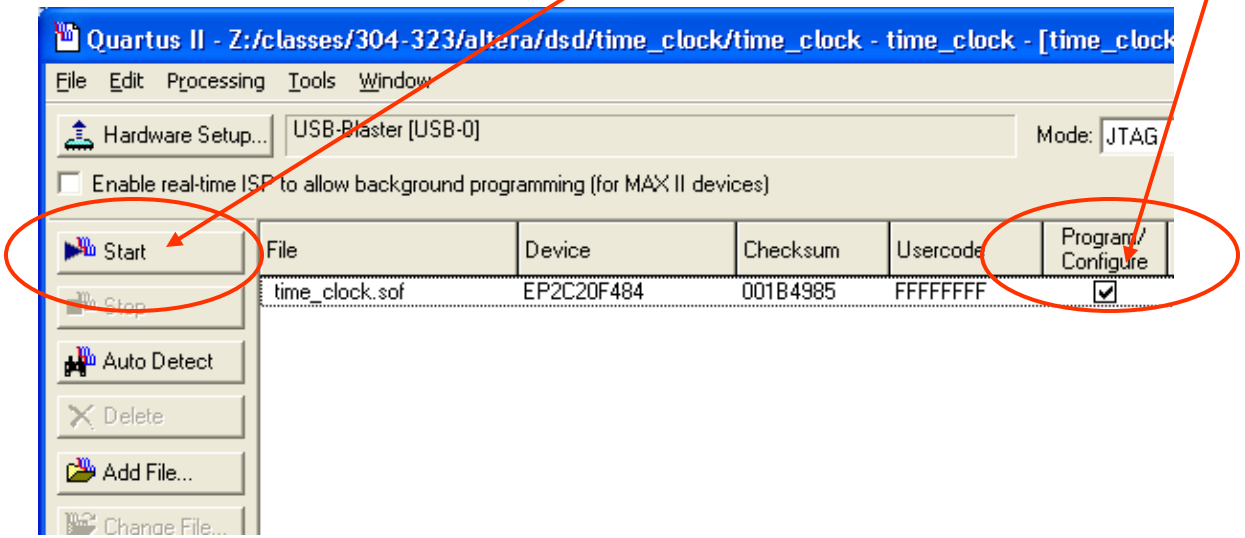
If there is no device visible, click on ***Hardware Setup***

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.

Load values into the registers using the dipswitches. There are 10 switches, two will be used for addressing, eight for data. The contents of the registers can be displayed using either the diodes or the 7-segment display.

Demonstrate the reset functionality of the registers. The reset can be connected to a push button.

Demonstrate the adder and subtractor circuits function properly by performing an addition or subtraction. You may use values stored in a register as input for the adder and subtractor. The output can be displayed using ether the diodes or the 7-segment display.

You must clearly indicate the choice of inputs and outputs.

5. 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.

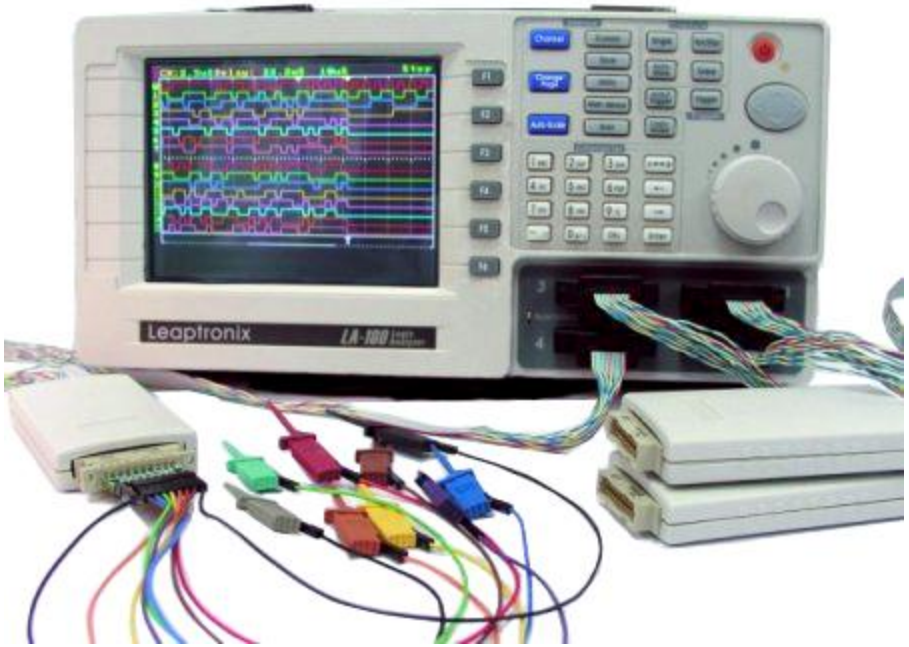
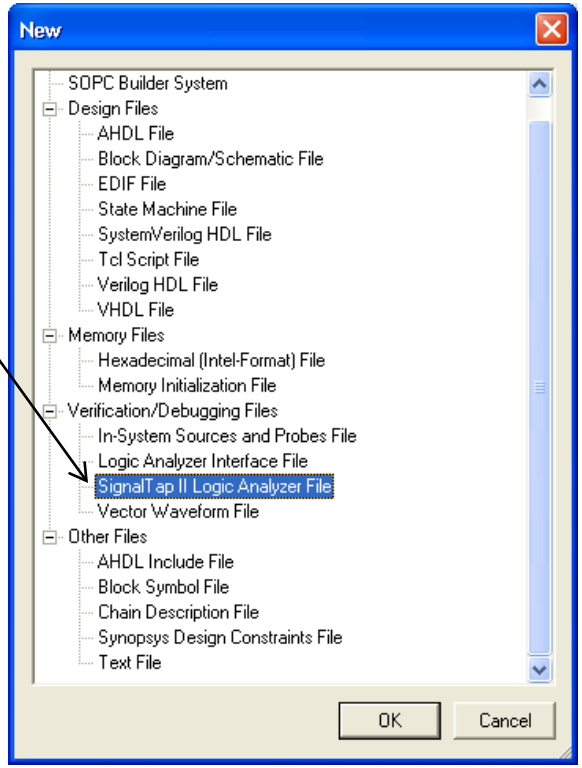


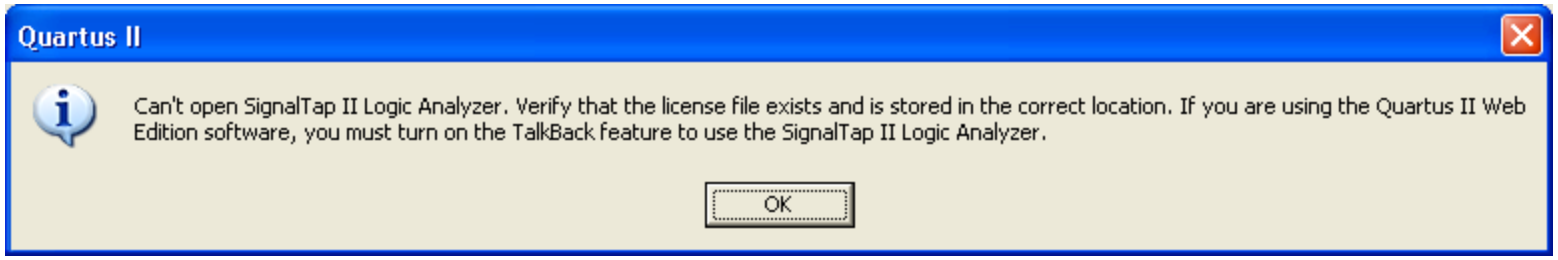
Image from http://www.elexp.com/tst_p100.htm

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.

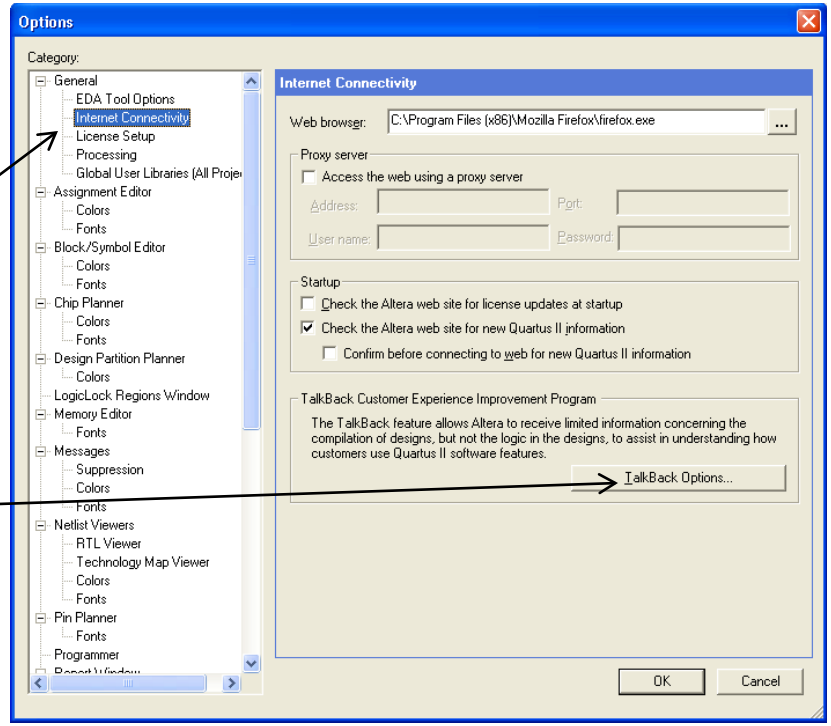


You may get the following error message popup (especially if you are using the web version of Quartus on your own computer:

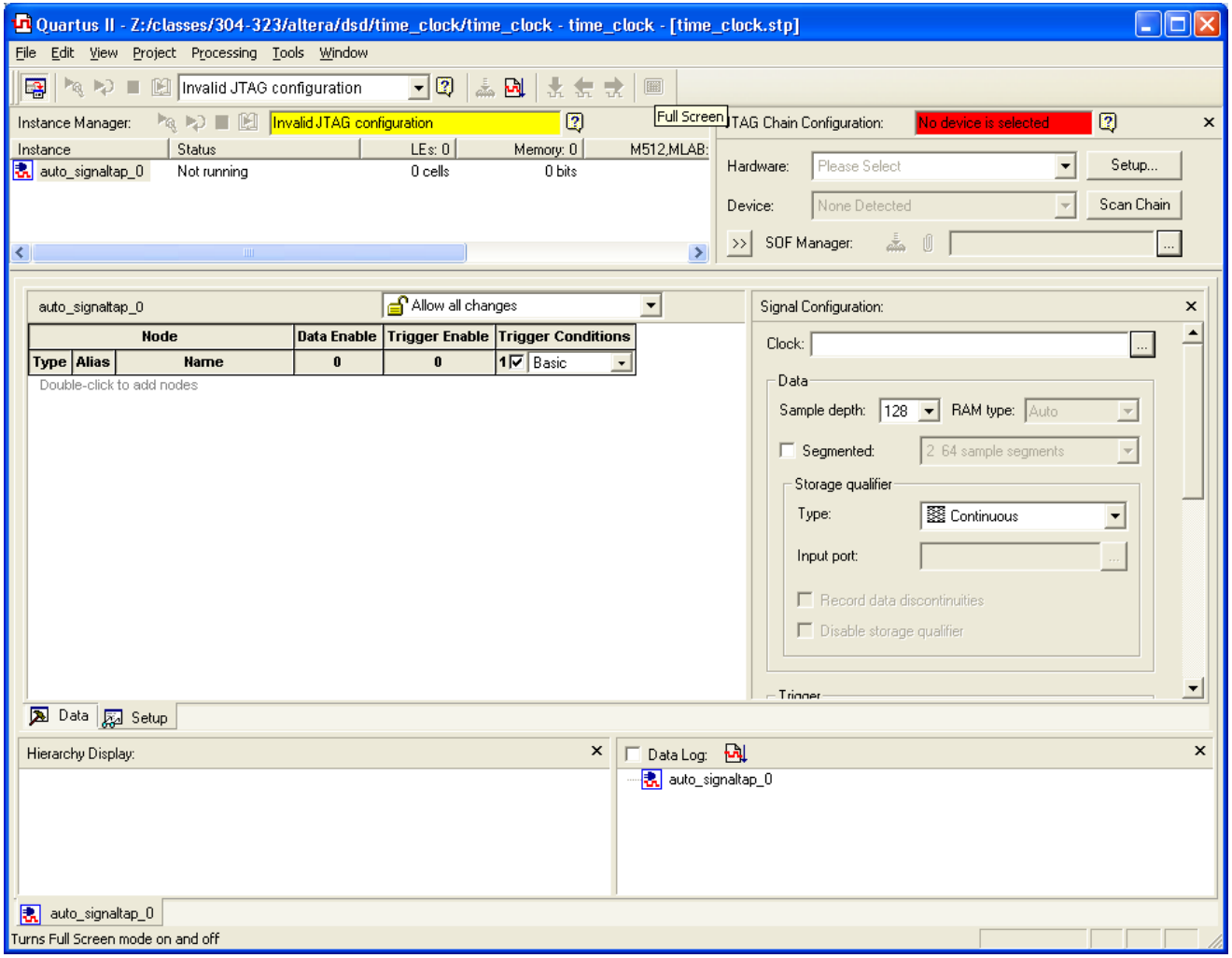


If so, turn on TalkBack by going to Tools/Options and selecting the Internet Connectivity tab. Then click on TalkBack Options and turn on the TalkBack Feature in the window that pops up.

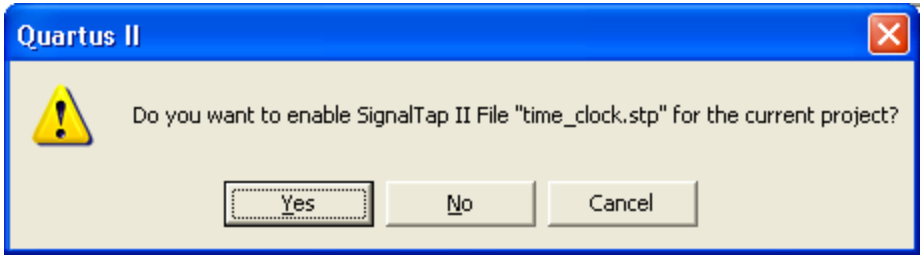
You will have to restart Quartus.



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

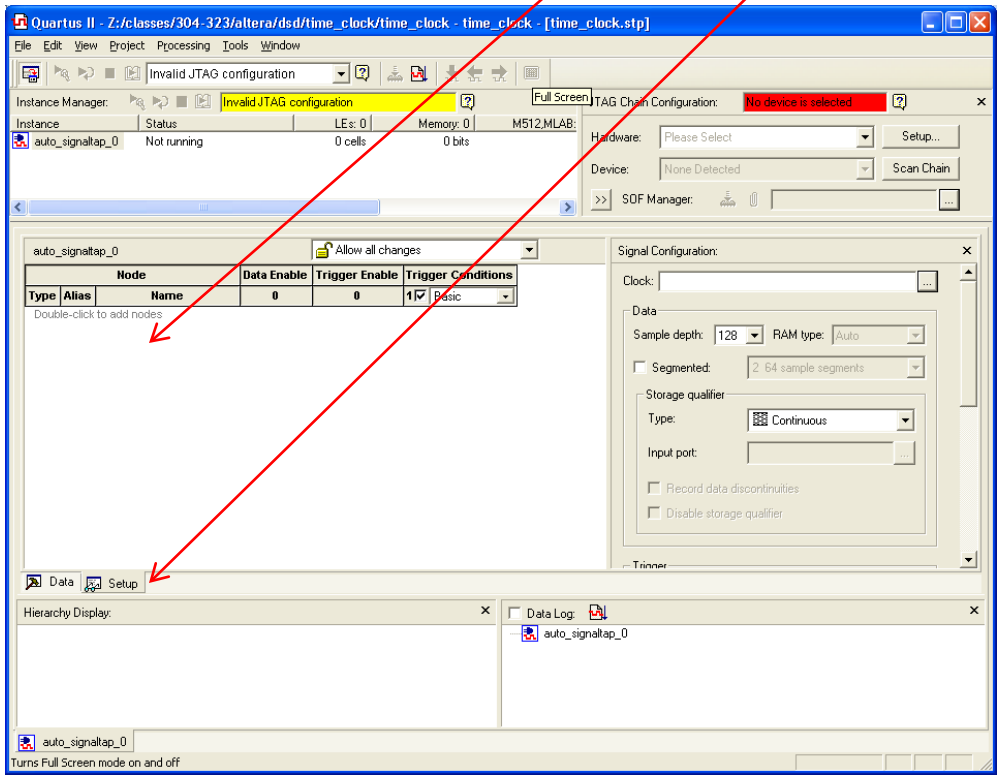


Before doing anything else, save the file (File/Save) giving the name “gNN_stack52_testbed.stp”. In the window that pops up, say “Yes”. This enables the SignalTap II file to be active for the current project.



The next thing to do is to specify the circuit nodes to be analyzed. For this tutorial, we will look at the values in the registers R0, R1, R2, R3,.

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 regsiters.

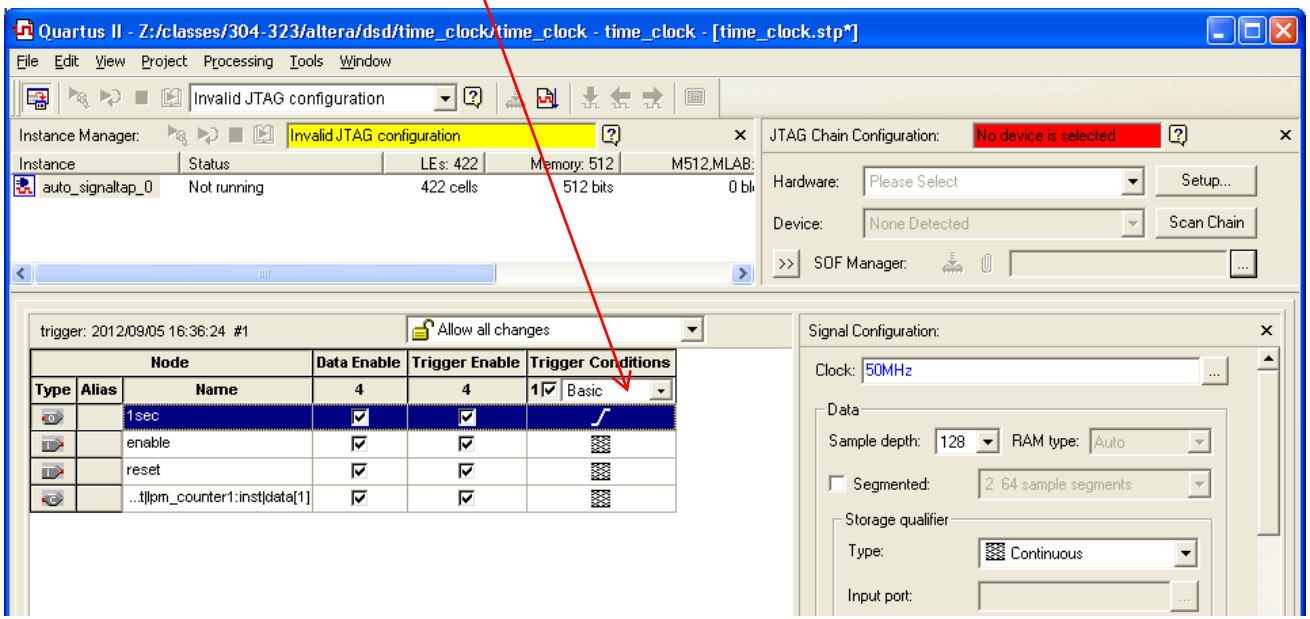
Make sure that the “Filter:” pane at the top has “SignalTap II: pre-synthesis” selected.

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.

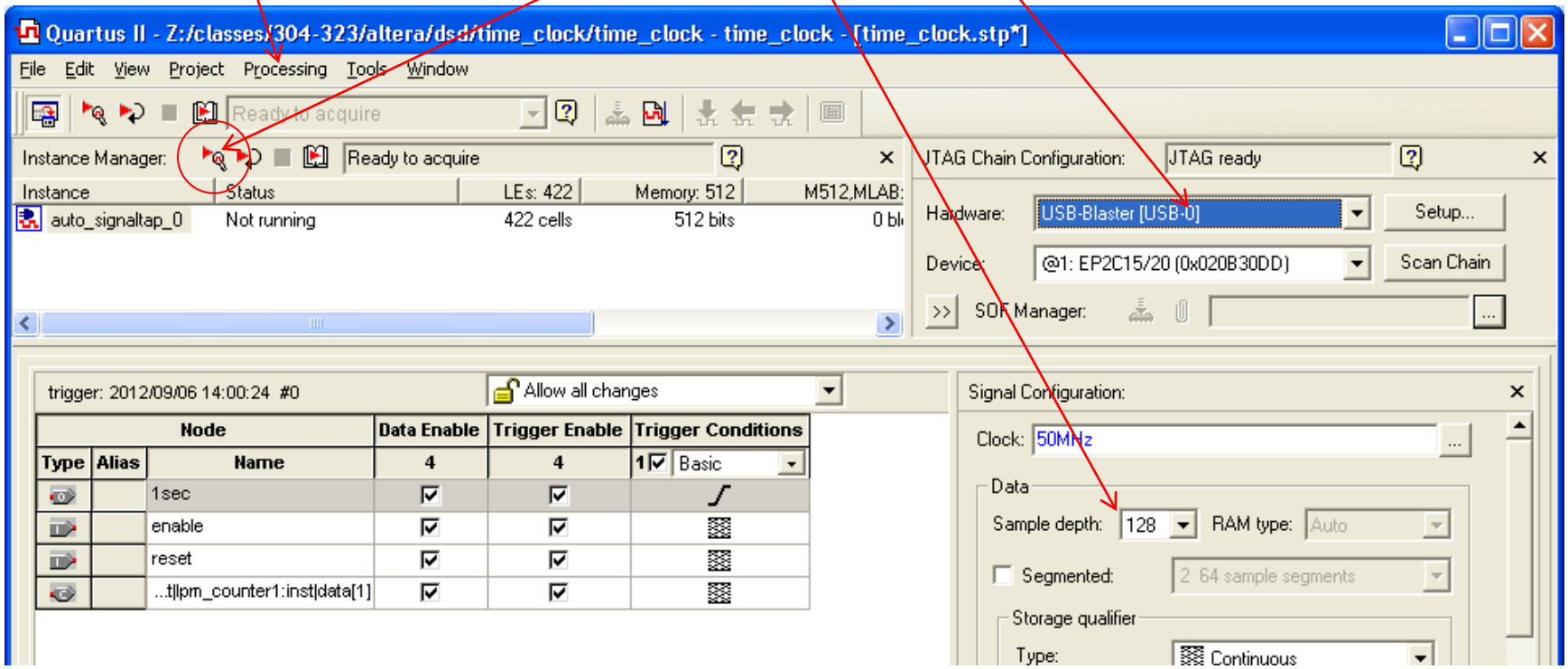
Once the design has been recompiled you can setup the triggering conditions, which will tell SignalTap II when it should start collecting measurements. This is done with the Setup tab selected.

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”:

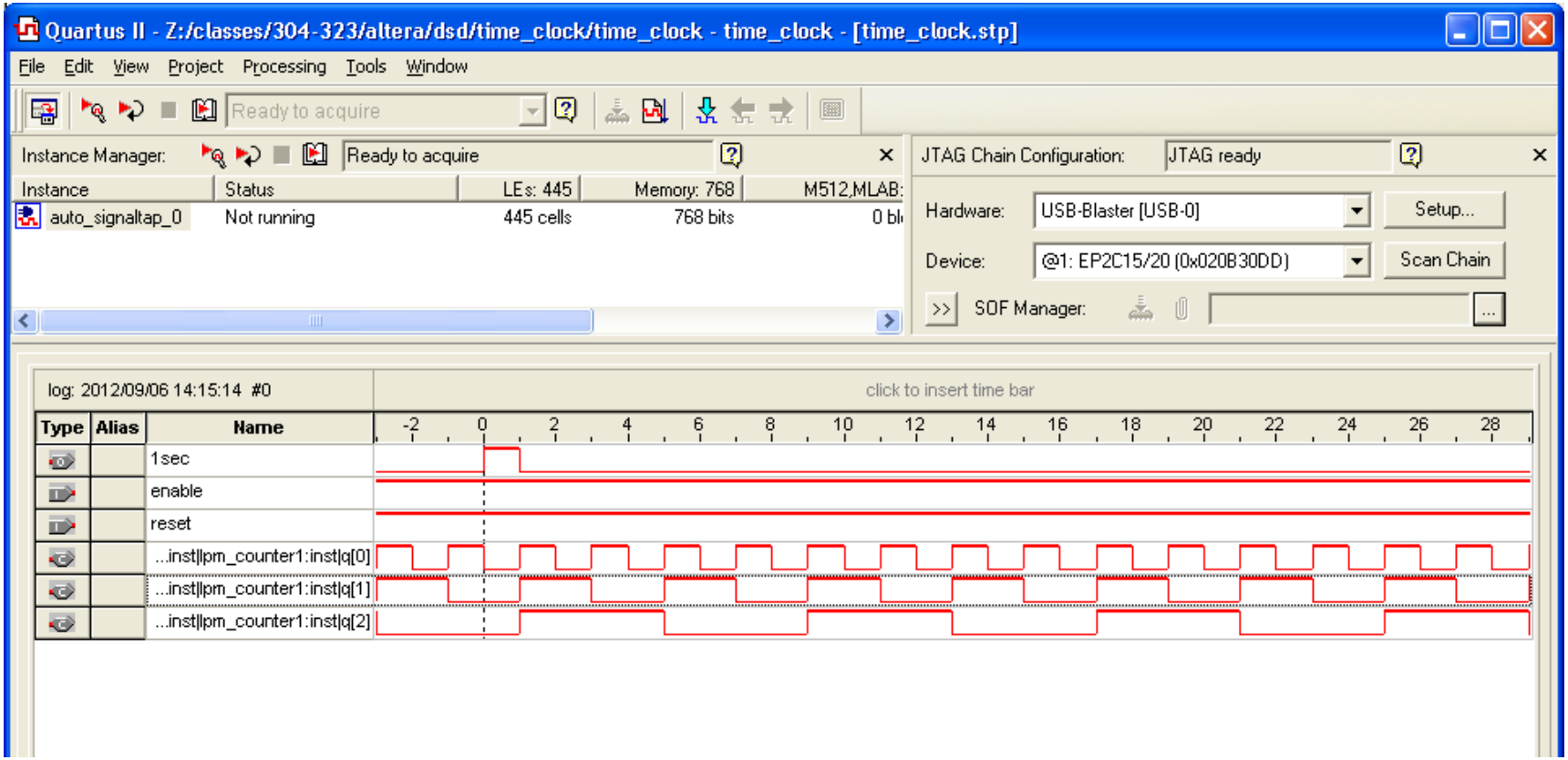


Next, set the Hardware to USB-Blaster. A connection will be made between the Quartus SignalTap II window and the on-chip logic analyzer circuitry and the analyzer will be ready to acquire data.

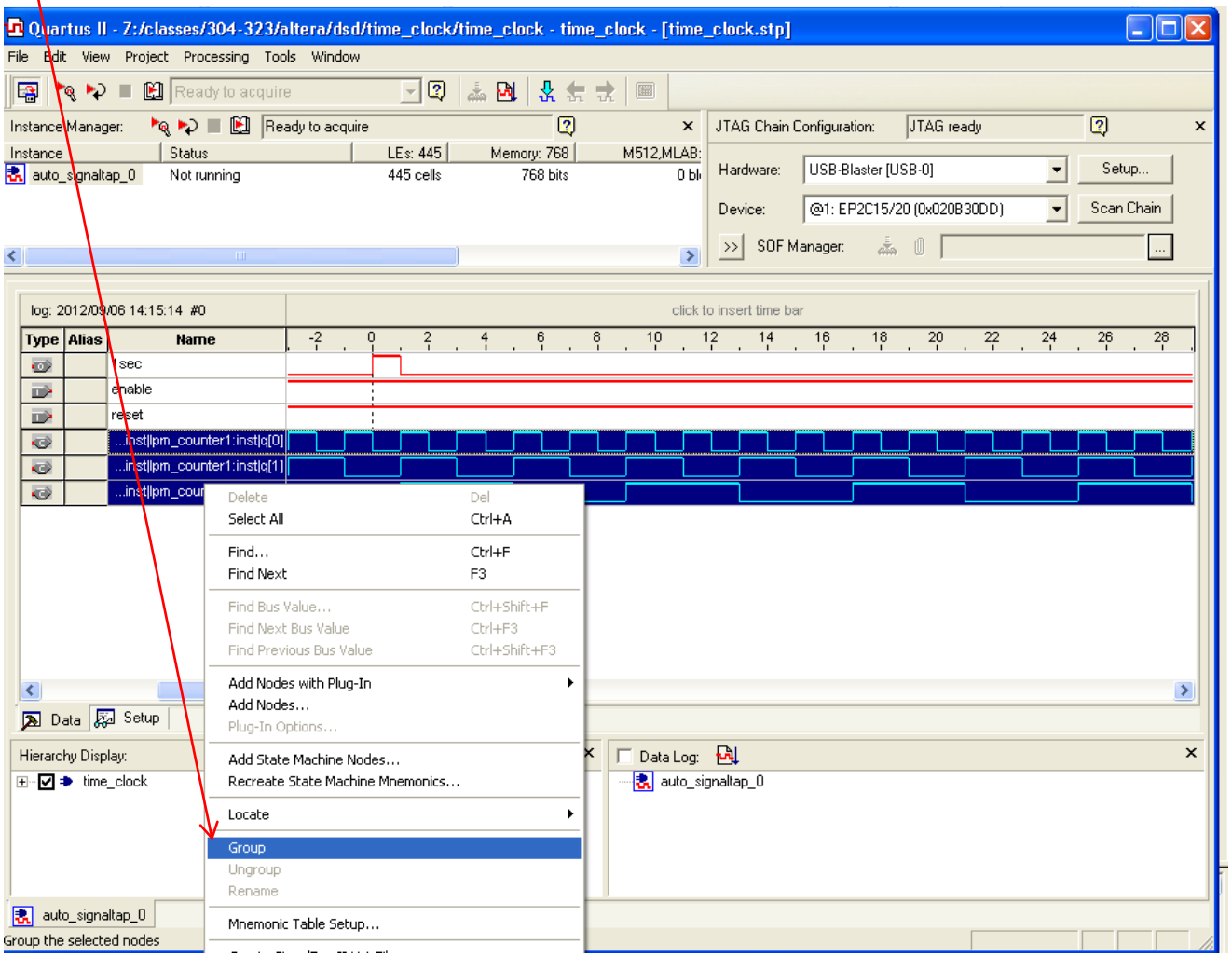
To actually acquire the data, press on the Run Analysis button, or select Processing/Run Analysis from the menu bar. The analyzer will capture 128 samples, set by the value in the Sample Depth box.



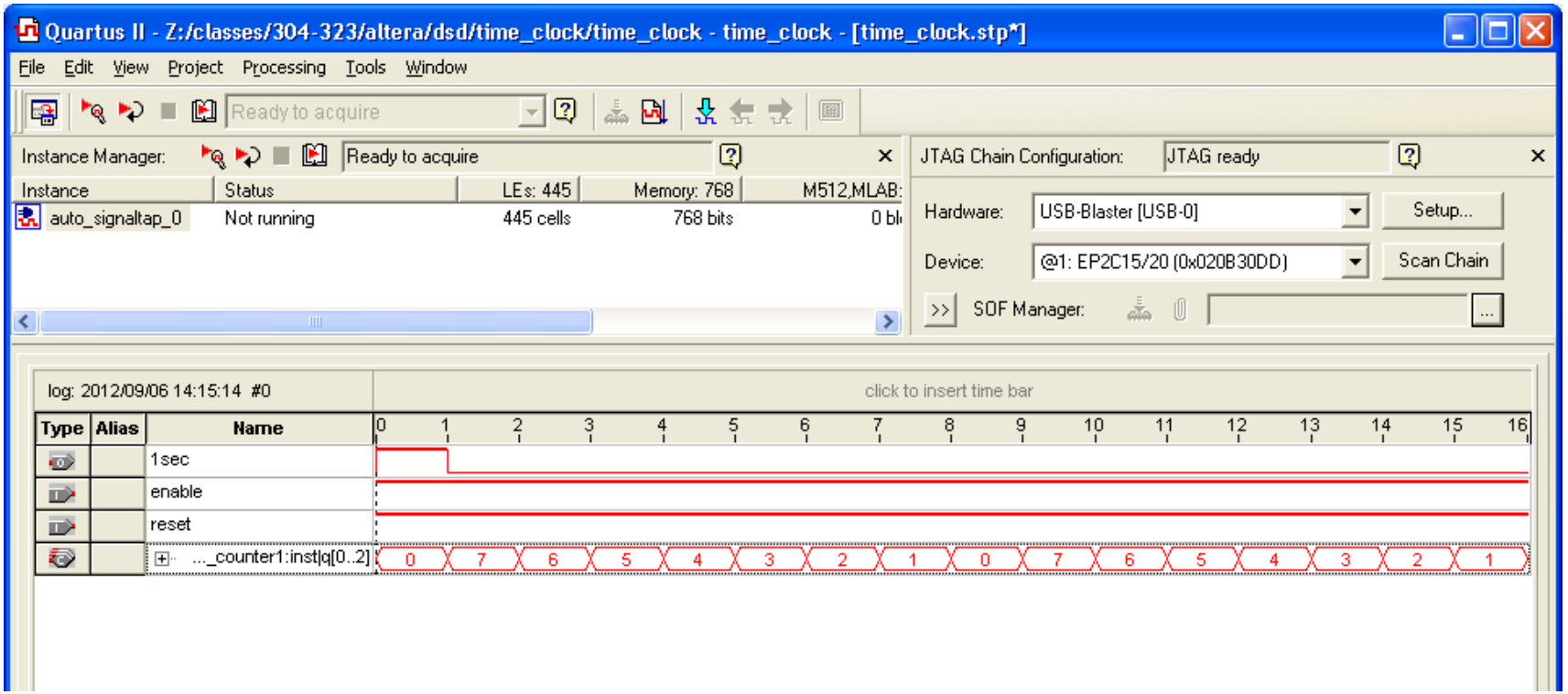
You should get a display like the one below.
You can zoom in and out by placing the cursor over the signal traces and left- or right-clicking (left zooms in, right zooms out).



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 Registers circuit changes in the various operation modes and the proper operation of the adder and subtractor.

6. Writeup of the Lab Report .

Write up a short report, for circuits (adder, subtractor and top level) 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.

For each circuit, include:

- 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.
- The schematic diagram of the top level 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 analysis or functional analysis of the circuit.

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).

Grade Sheet for Lab #3

Winter 2018.

Group Number:_____.

Group Member Name:_____. Student Number:_____.

Group Member Name:_____. Student Number:_____.

Marks			TA Signatures
	1.	VHDL and schematic for the 8-bit adder and subtractor	.
	2.	VHDL and Schematic for the top level design	.
	3.	Simulation of the 8-bit adder and subtractor	.
	4.	Simulation of the top-level circuit	.
	5.	Demonstration of the calculator in 2 different mode	.
	6.	Demonstration of SignalTap II analysis for top-level circuit	.

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.