SignalTap II with Verilog Designs

For Quartus II 13.1

1 Introduction

This tutorial explains how to use the SignalTap II feature within Altera's Quartus ® II software. The SignalTap II Embedded Logic Analyzer is a system-level debugging tool that captures and displays signals in circuits designed for implementation in Altera's FPGAs.

Contents:

2 Background

Quartus® II software includes a system level debugging tool called SignalTap II that can be used to capture and display signals in real time in any FPGA design.

During this tutorial, the reader will learn about:

- Probing signals using the SignalTap software
- Setting up triggers to specify when data is to be captured

This tutorial is aimed at the reader who wishes to probe signals in circuits defined using the Verilog hardware description language. An equivalent tutorial is available for the reader who prefers the VHDL language.

The reader is expected to have access to a computer that has Quartus II software installed. The detailed examples in the tutorial were obtained using Quartus II version 13.1, but other versions of the software can also be used.

Note:

Please note that there are no red LEDs on a DE0 or DE0-Nano board. All procedures using red LEDs in this tutorial are to be completed on the DE0 or DE0-Nano boards using green LEDs instead. If you are doing this tutorial on a DE0 board, replace *LEDR* with *LEDG* in the Verilog modules below. If you are foing this tutorial on a DE0-Nano board, replace *LEDR* with *LED* below.

Additionally, the DE0-Nano is limited to 4 switches. If you are doing this tutorial on a DE0-Nano, replace all occurances of *[7:0]* with *[3:0]* below.

3 Example Circuit

As an example, we will use the switch circuit implemented in Verilog in Figure 1. This circuit simply connects the first 8 switches on the DE-series board to the first 8 red LEDs on the board. It does so at the positive edge of the clock (CLOCK_50) by loading the values of the switches into a register whose output is connected directly to the red LEDs.

```
// Top-level module
module  switches (SW, CLOCK_50, LEDR);
    input  [7:0] SW;
    input  CLOCK_50;
    output reg  [7:0]  LEDR;

    always @(posedge CLOCK_50)
        LEDR [7:0] <= SW [7:0];
endmodule
```

Figure 1. The switch circuit implemented in Verilog code

Implement this circuit as follows:

- Create a project *switches*.
- Include a file *switches.v*, which corresponds to Figure 1, in the project.
- Select the correct device that is associated with the DE-series board. A list of device names for DE-series boards can be found in Table 1.
- Import the relevant qsf file. For example, for a DE2-115 board, this file is called *DE2-115.qsf* and can be imported by clicking Assignments > Import Assignments. For convenience, this file is provided in the *design_files* subdirectory within the tutorials folder, which is included on the CD-ROM that accompanies the DE-series board and can also be found on Altera's DE-series web pages. The node names used in the sample circuit correspond to the names used in this file.
- Compile the design.

Board	Device Name
DE0	Cyclone III EP3C16F484C6
DE0-Nano	Cyclone IVE EP4CE22F17C6
DE1-SoC	Cyclone V SoC 5CSEMA5F31C6
DE2-115	Cyclone IVE EP4CE115F29C7

Table 1. DE-series FPGA device names

4 Enabling the Quartus II TalkBalk Feature

When using the Quartus II Web Edition, the TalkBack feature must be enabled in order to access the SignalTap II software. Enabling TalkBack allows Quartus II and other Altera programs to send a limited amount of data regarding

their usage to Altera. This data is primarily used to better understand how users interact with the software, and does not include any design files. The full list of what data is or is not sent can be found in the TalkBack License Agreement.

To enable the TalkBack feature in Quartus II, select Tools > Options. In the Options window, select Internet Connectivity from the menu, and click on TalkBack Options... to open the window shown in Figure 2. If you accept the TalkBack License Agreement, then check the box labelled Enable sending TalkBack data to Altera and click OK.

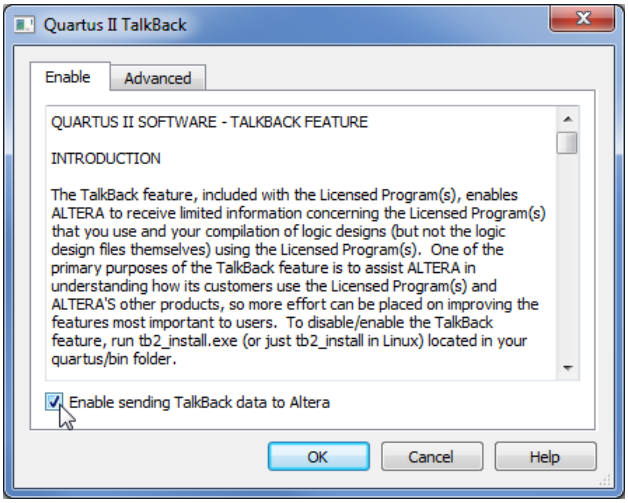

Figure 2. Enabling the TalkBack feature.

5 Using the SignalTap II software

In the first part of the tutorial, we are going to set up the SignalTap Logic Analyzer to probe the values of the 8 LED switches. We will also set up the circuit to trigger when the first switch (LED[0]) is high.

1. Open the SignalTap II window by selecting File > New, which gives the window shown in Figure 3. Choose SignalTap II Logic Analyzer File and click OK.

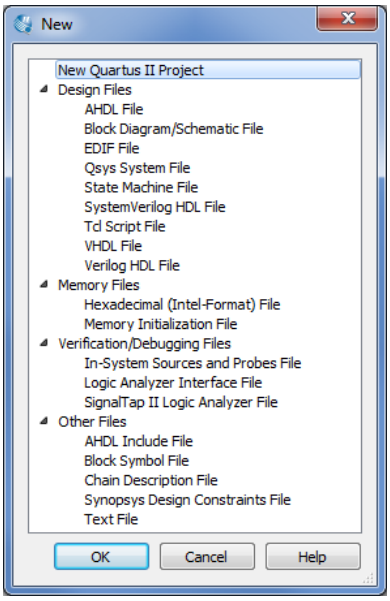

Figure 3. Need to prepare a new file.

2. The SignalTap II window with the Setup tab selected is depicted in Figure 4. Save the file under the name *switches.stp*. In the dialog box that follows (Figure 5), click OK. For the dialog "Do you want to enable SignalTap II file 'switches.stp' for the current project?" click Yes (Figure 6). The file *switches.stp* is now the SignalTap file associated with the project.

   Note: If you want to disable this file from the project, or to disable SignalTap from the project, go to Assignments > Settings. In the category list, select SignalTap II Logic Analyzer, bringing up the window in Figure 7. To turn off the analyzer, uncheck Enable SignalTap II Logic Analyzer. Also, it is possible to have multiple SignalTap files for a given project, but only one of them can be enabled at a time. Having multiple SignalTap files might be useful if the project is very large and different sections of the project need to be probed. To create a new SignalTap file for a project, simply follow Steps 1 and 2 again and give the new file a different name. To change the SignalTap file associated with the project, in the SignalTap II File name box browse for the file wanted, click Open, and then click OK. For this tutorial we want to leave SignalTap enabled and we want the SignalTap II File name to be *switches.stp*. Make sure this is the case and click OK to leave the settings window.

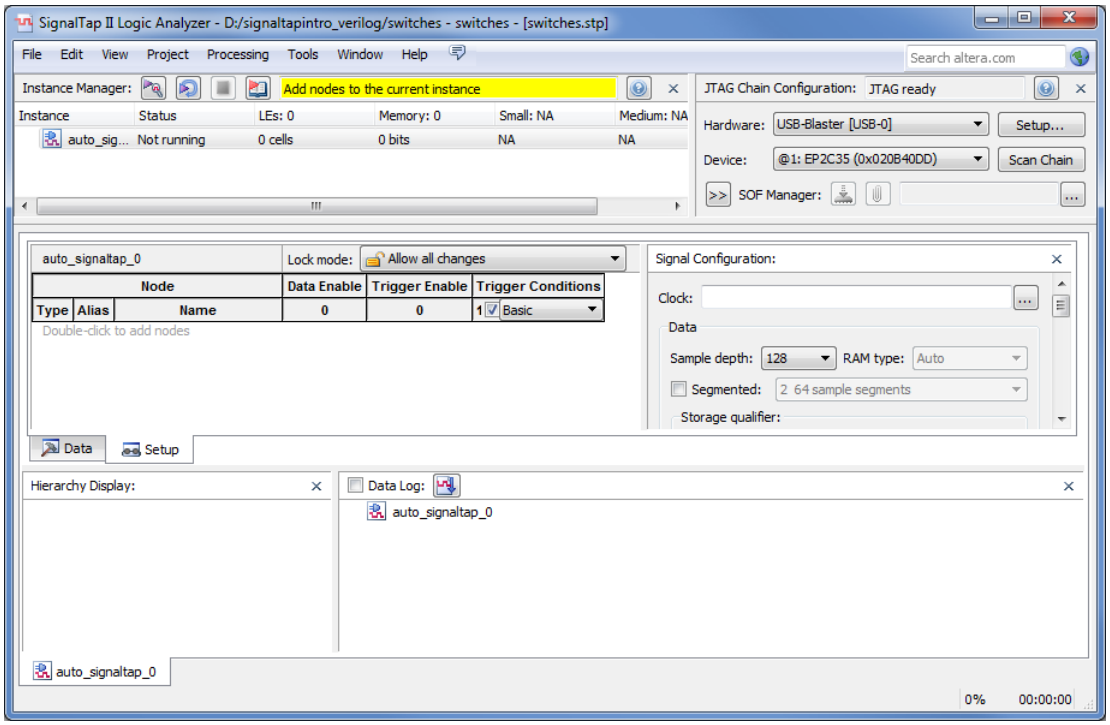

Figure 4. The SignalTap II window.

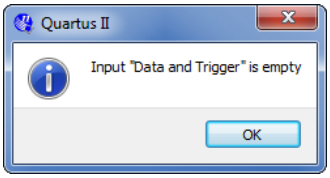

Figure 5. Click OK to this dialog.

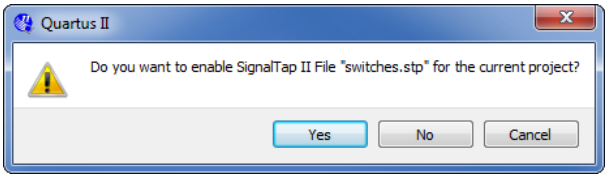

Figure 6. Click Yes to this dialog.

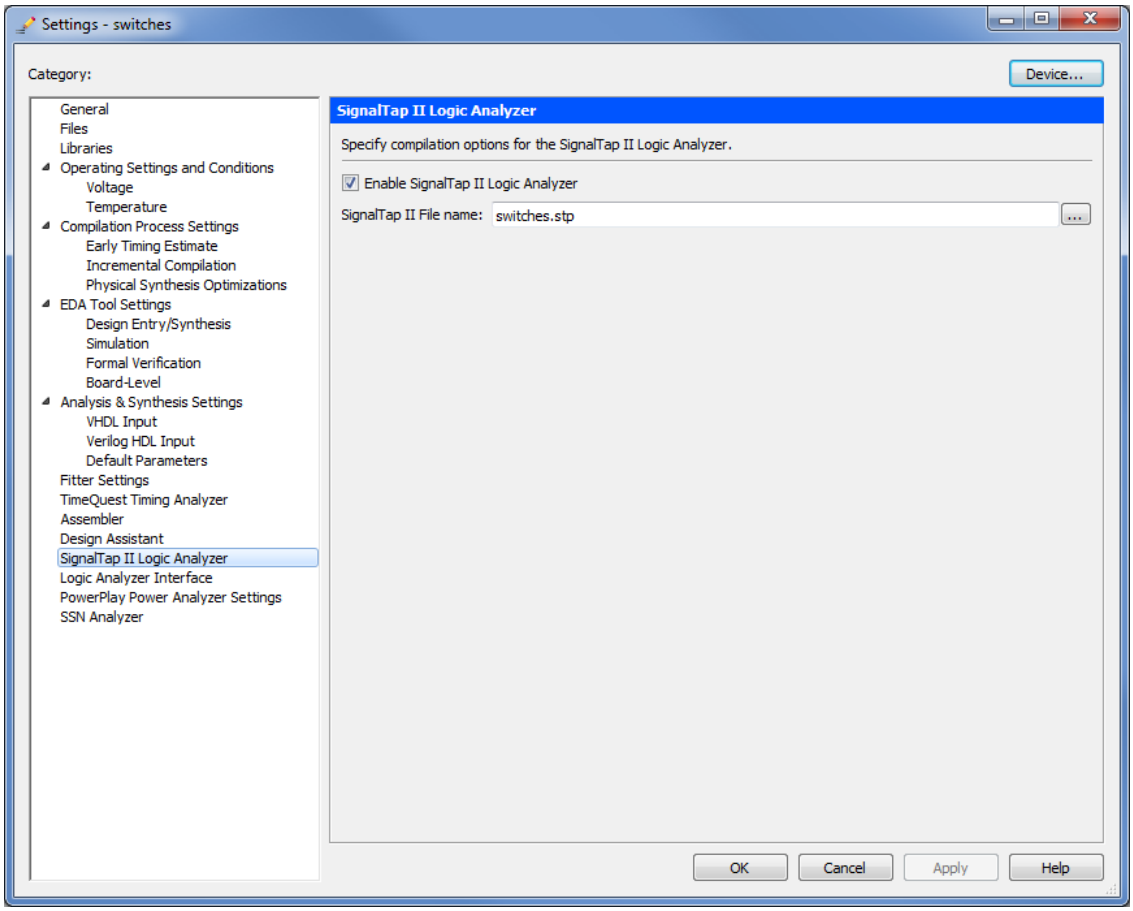

Figure 7. The SignalTap II Settings window.

3. We now need to add the nodes in the project that we wish to probe. In the Setup tab of the SignalTap II window, double-click in the area labeled Double-click to add nodes, bringing up the Node Finder window, as shown in Figure 8. Click on ⌄⌄ to show more search options. For the Filter field, select SignalTap II: pre-synthesis, and for the Look in field select |switches|. Click List. This will now display all the nodes that can be probed in the project. Highlight SW[0] to SW[7], and then click the > button to add the switches to be probed. Then click OK.

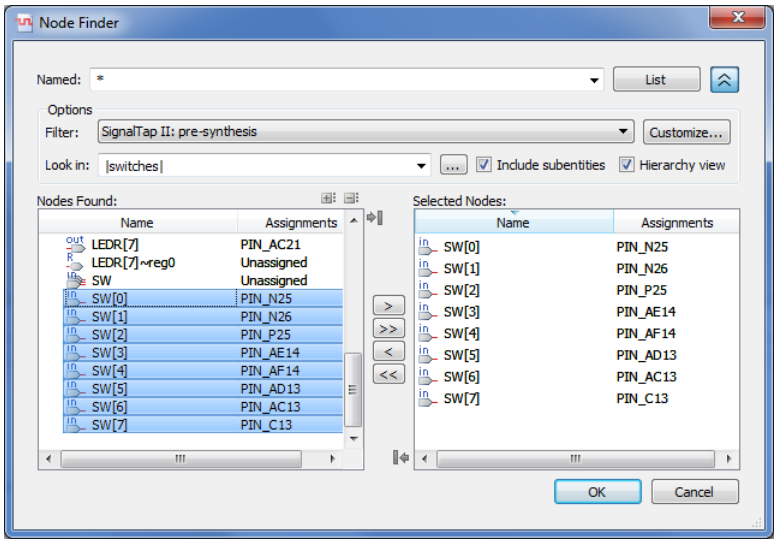

Figure 8. Adding nodes in the Node Finder window on a DE-series board.

4. Before the SignalTap analyzer can work, we need to specify what clock is going to run the SignalTap module that will be instantiated within our design. To do this, in the Clock box of the Signal Configuration pane of the SignalTap window, click ..., which will again bring up the Node Finder window. Select List to display all the nodes that can be added as the clock, and then double-click CLOCK_50, which results in the image shown in Figure 9. Click OK.

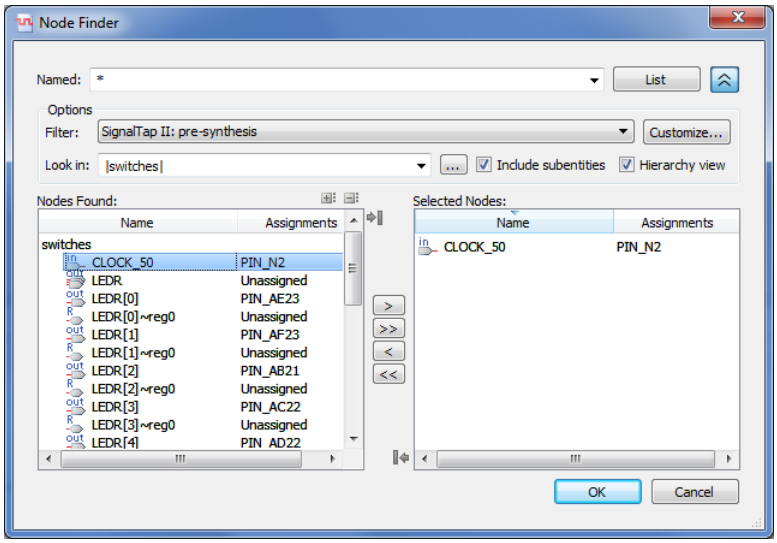

Figure 9. Setting CLOCK_50 as the clock for the SignalTap instance on a DE-series board.

5. With the Setup tab of the SignalTap window selected, select the checkbox in the Trigger Conditions column. In the dropdown menu at the top of this column, select Basic AND. Right-click on the Trigger Conditions

cell corresponding to the node SW[0] and select High. Now, the trigger for running the Logic Analyzer will be when the first switch on the DE-series board is set to high, as shown in Figure 10. Note that you can right click on the Trigger Conditions cell of any of the nodes being probed and select the trigger condition from a number of choices. The actual trigger condition will be true when the logical AND of all these conditions is satisfied. For now, just keep the trigger condition as SW[0] set to high and the others set to their default value, Don't Care.

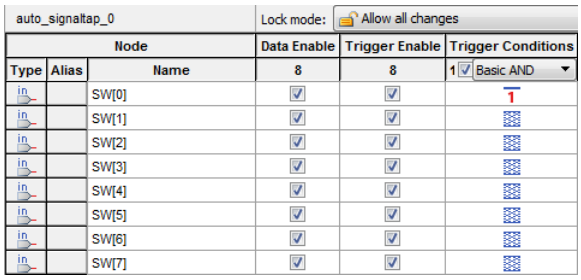

Figure 10. Setting the trigger conditions.

6. For SignalTap II to work, we need to properly set up the hardware. First, make sure the DE-series board is plugged in and turned on. In the Hardware section of the SignalTap II window, located in the top right corner, click Setup..., bringing up the window in Figure 11. Double click USB-Blaster in the Available Hardware Items menu, then click Close.

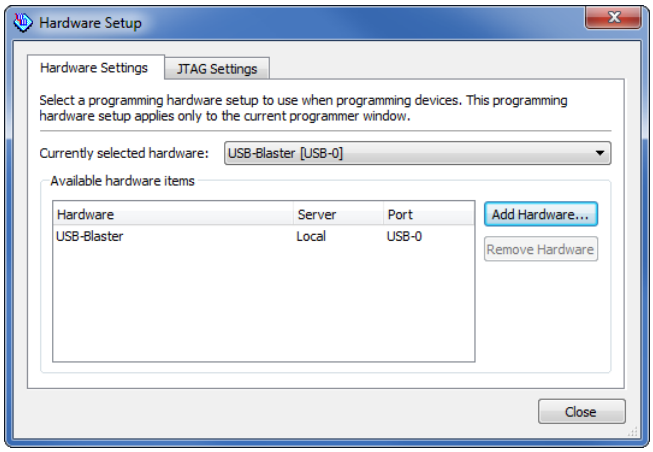

Figure 11. Setting up hardware.

7. The last step in instantiating SignalTap in your design is to compile the design. In the main Quartus II window, select Processing > Start Compilation and indicate that you want to save the changes to the file by clicking Yes. After compilation, go to Tools > Programmer and load the project onto the DE-series board.

6 Probing the Design Using SignalTap II

Now that the project with SignalTap II instantiated has been loaded onto the DE-series board, we can probe the nodes as we would with an external logic analyzer.

1. On the DE-series board, first set all of the switches (0-7) to low. We will try to probe the values of these switches once switch 0 becomes high.
2. In the SignalTap window, select Processing > Run Analysis or click the icon . Then, click on the Data tab of the SignalTap II Window. You should get a screen similar to Figure 12. Note that the status column of the SignalTap II Instance Manager pane says "Waiting for trigger." This is because the trigger condition (Switch 0 being high) has not yet been met. (This is of course if Switch 0 is actually low as instructed in the previous step. If it is not, set it to low and then click Run Analysis again).

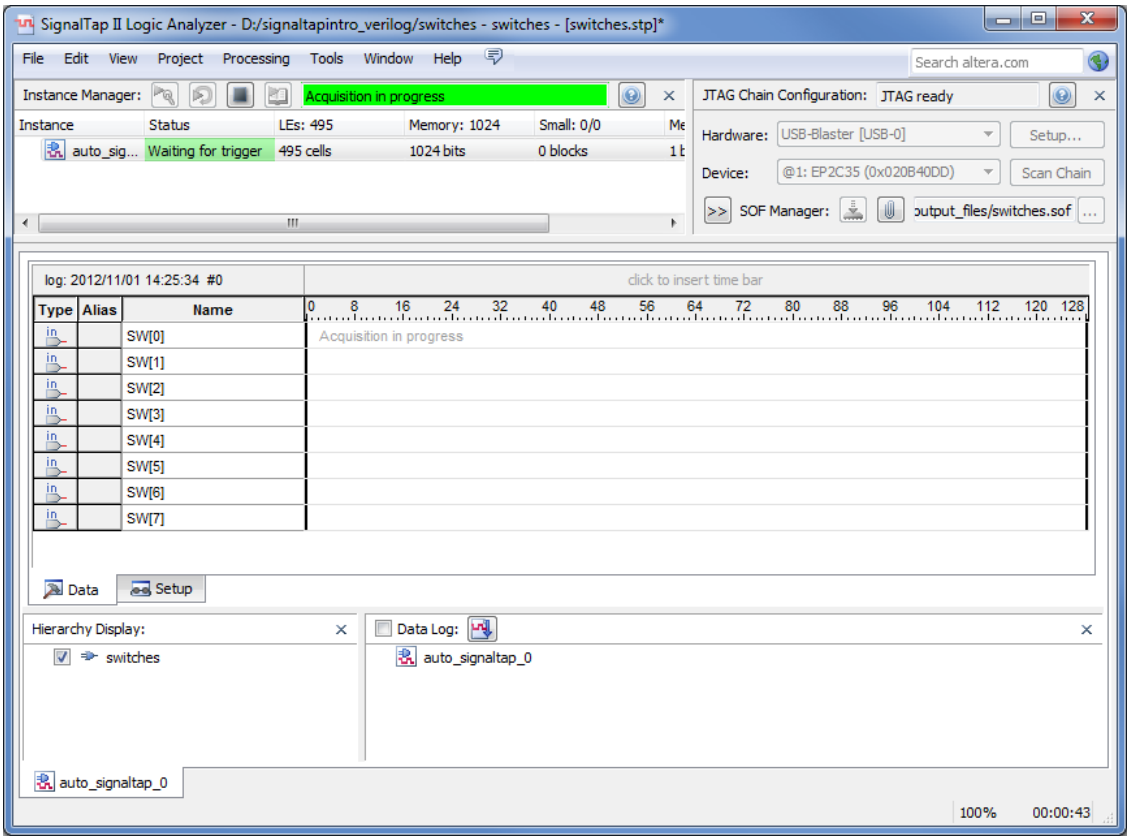

Figure 12. SignalTap II window on a DE-series board after Run Analysis has been clicked.

3. Now, to observe the trigger feature of the Logic Analyzer, set Switch 0 on the DE-series board to high. The data window of the SignalTap II window should display the image in Figure 13. Note that this window shows the data levels of the 8 nodes being tapped before the trigger condition was met and also after. To see this, flip on any of the switches from 0-7 and then click Run Analysis again. When switch 0 is set to high again, you will see the values of the switches displayed on the SignalTap II Logic Analyzer.

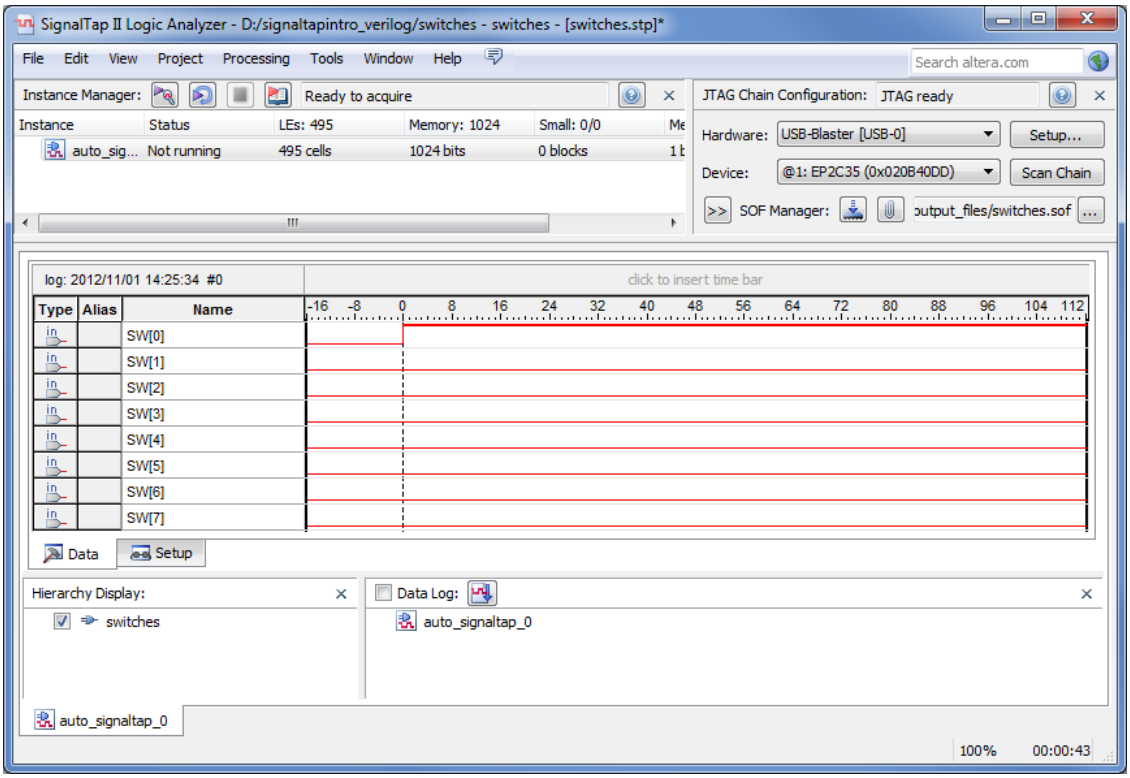

Figure 13. Graphical display of values after trigger condition is met.

7 Advanced Trigger Options

Sometimes in a design you may want to have a more complicated triggering condition than SignalTap's basic triggering controls allow. The following section describes how to have multiple trigger levels.

7.1 Multiple Trigger Levels

In this section, we will set up the analyzer to trigger when there is a positive edge from switch 0, switch 1, switch 2, and then switch 3, in that order.

1. Click the Setup tab of the SignalTap II window.
2. In the Signal Configuration pane, select 4 from Trigger Conditions dropdown menu as in Figure 14 (you may have to scroll down in the Signal Configuration pane to see this menu). This modifies the node list window by creating three new Trigger Conditions columns.

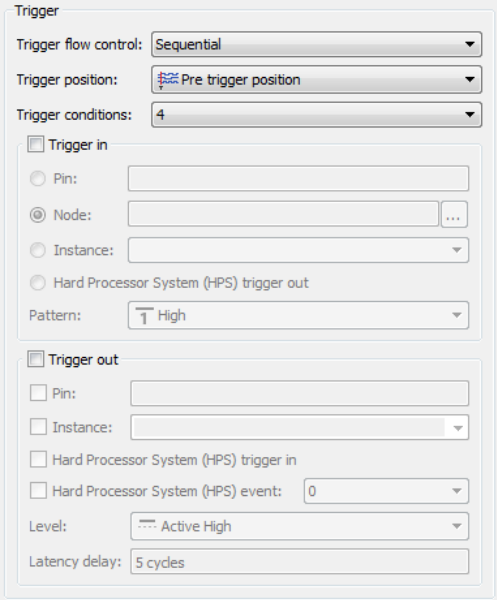

Figure 14. Set trigger levels to 4.

3. Right click the Trigger Condition 1 cell for SW[0], and select Rising Edge. Do the same for the Trigger Condition 2 cell for SW[1], Trigger Condition 3 for SW[2], and Trigger Condition 4 for SW[3]. You should end up with a window that looks like Figure 15.

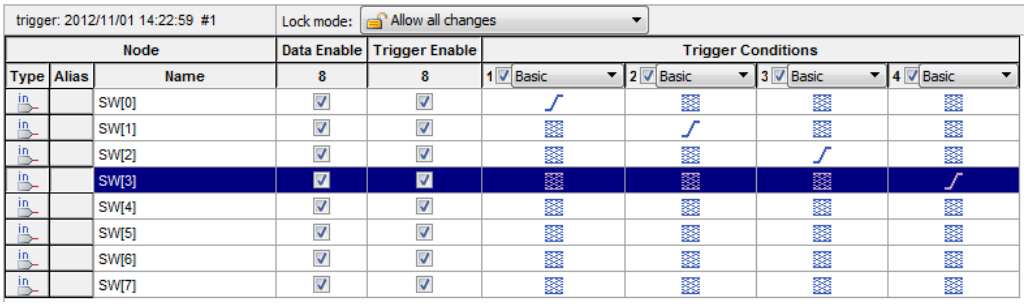

Figure 15. Multiple trigger levels set.

4. Now, recompile the design and load it onto the DE-series board again.

5. Go back to the SignalTap II window, click on the Data tab, and then click Processing > Run Analysis. Note that the window will say "Waiting for trigger" until the appropriate trigger condition is met. Then, in sequence, flip to high switches 0, 1, 2, and then 3.

   After this has been done, you will see the values of all the switches displayed as in Figure 16. Experiment by following the procedure outlined in this section to set up other trigger conditions and use the DE-series board to test these trigger conditions.

If you want to continuously probe the analyzer, instead of clicking "Run Analysis," click "Autorun Analysis" which is the icon right next to the "Run Analysis" icon. If you do this, every time the trigger condition is met the value in the display will be updated. You do not have to re-select "Run Analysis." To stop the "Autorun Analysis" function, click the icon.

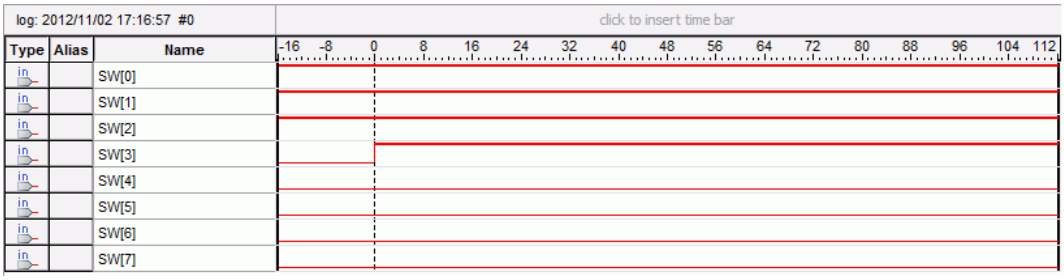

Figure 16. Logic Analyzer display when all four trigger conditions have been met.

7.2 Advanced Trigger Conditions

In this section we will learn how to create advanced trigger conditions. Our trigger condition will be whenever any one of the first 3 LED displays have a positive or negative edge. This means that the Logic Analyzer will update its display every time one of these inputs changes. Note that we could have any logical function of the nodes being probed to trigger the analyzer. This is just an example. After you implement this in the next few steps, experiment with your own advanced triggers.

1. Have the *switches* project opened and compiled from the previous examples in this tutorial.
2. Open the SignalTap window and select the Setup tab. In the Signal Configuration pane make sure that the number of Trigger Conditions is set to 1.
3. In the Trigger Conditions column of the node list, make sure the box is checked and select Advanced from the dropdown menu as in Figure 17. This will immediately bring up the window in Figure 18. This window allows you to create a logic circuit using the various nodes that you are probing with SignalTap.

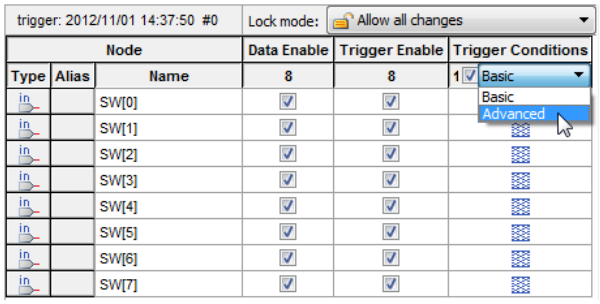

Figure 17. Select Advanced from the Trigger Level dropdown menu.

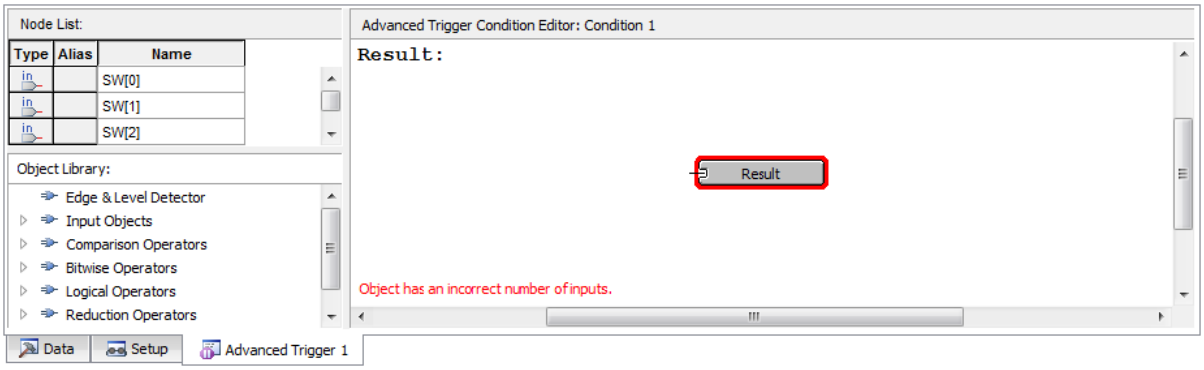

Figure 18. The Advanced Trigger editing window.

4. In the node list section of this window, highlight the 3 nodes SW[0] to SW[2], and click and drag them into the white space of the Advanced trigger window, resulting in Figure 19. Note that you can also drag and drop each node individually.

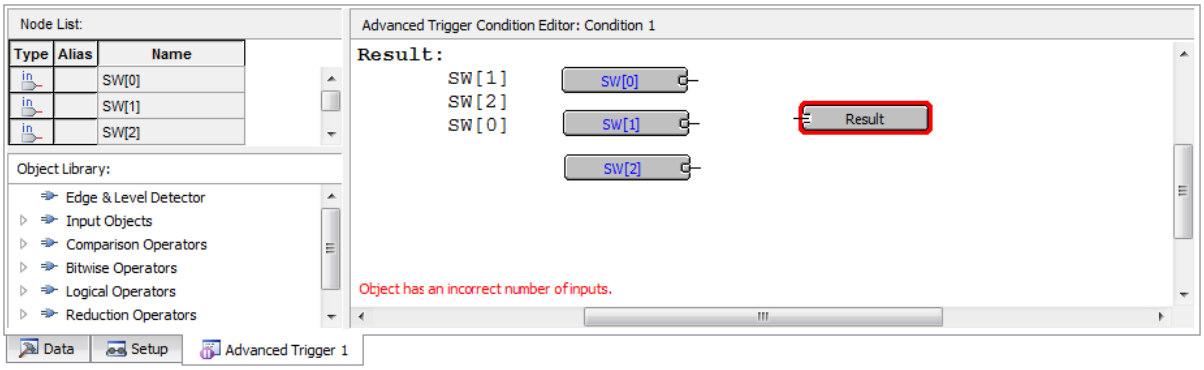

Figure 19. The three input nodes of interest dragged into the Advanced Trigger Editing Window.

5. We now need to add the necessary logical operators to our circuit. We will need an OR gate as well as three edge level detectors. To access the OR gate, click on the plus sign next to Logical Operators in the Object Library and select Logical Or, as in Figure 20. Then drag and drop the operator into the editing window.

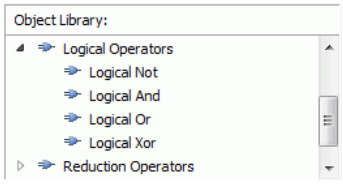

Figure 20. Select the Logical Or operator from the Object Library window and drag this into the editing window.

6. In the object library click Edge and Level Detector and drag this into the editing window. Do this three

times and then arrange the circuit as in Figure 21. The three inputs should each be connected to the input of an edge and level detector and the output of each of these detectors should be connected to the OR gate. The output of the OR gate should be connected to the output pin already in the editing window.

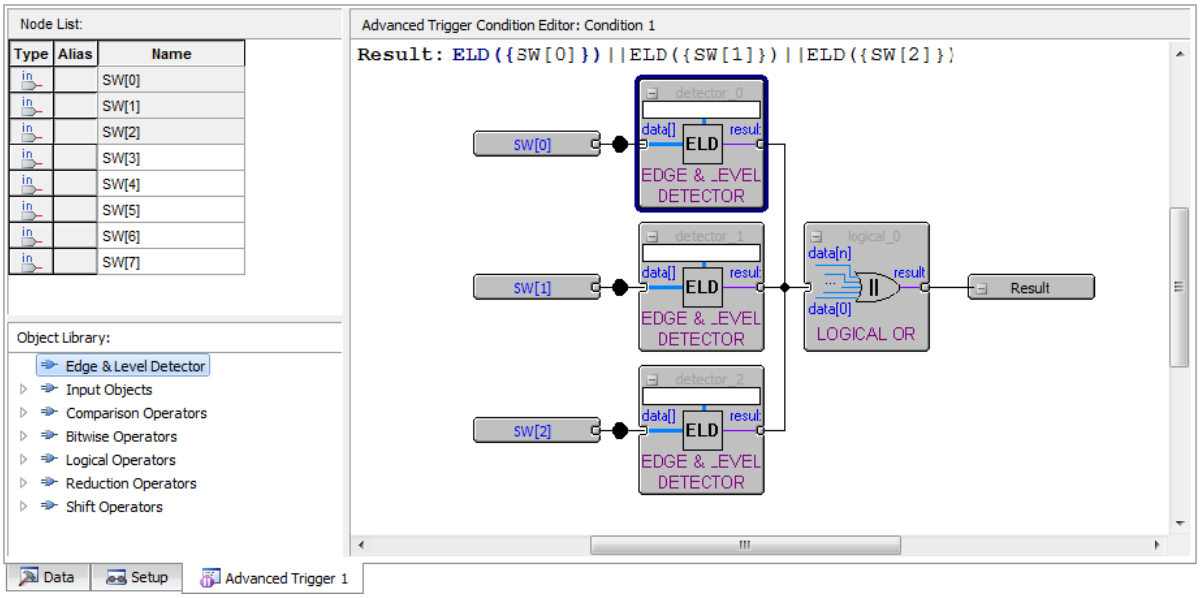

Figure 21. Arrange the elements to create a circuit that looks like this.

7. We now need to set each edge and level detector to sense either a falling edge or a rising edge. Double click one of the edge and level detectors, bringing up the window in Figure 22. Type E in the setting box and then click OK. This will mean that the detector will output 1 whenever there is either a falling edge or a rising edge of its input. Repeat this step for the two remaining edge and level detectors.

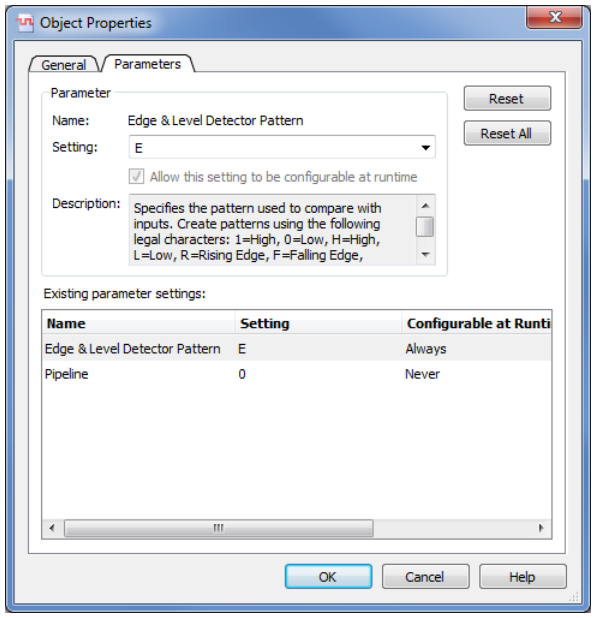

Figure 22. Type E in the setting box so that the function triggers on both rising and falling edges.

8. To test this Advanced trigger condition, compile the designed circuit again and load it onto the DE-series board. Then run Signal Tap as described in the previous section. You should note that the Analyzer should sense every time you change one of the first three switches on the board.

8 Sample Depth and Buffer Acquisition Modes

In this section, we will learn how to set the Sample Depth of our analyzer and about the two buffer acquisition modes. To do this, we will use the previous project and use segmented buffering. Segmented buffering allows us to divide the acquisition buffer into a number of separate, evenly sized segments. We will create a sample depth of 256 bits and divide this into eight 32-sample segments. This will allow us to capture 8 distinct events that occur around the time of our trigger.

1. Change the trigger condition back to Basic and have only one trigger condition. Make the trigger condition to be at either edge of SW[0].
2. In the Signal Configuration pane of the SignalTap II window, in the Sample depth dropdown menu of the Data pane select 256. This option allows you to specify how many samples will be taken around the triggers in your design. If you require many samples to debug your design, select a larger sample depth. Note, however, that if the sample depth selected is too large, there might not be enough room on the board to hold your design and the design will not compile. If this happens, try reducing the sample depth.
3. In the Signal Configuration pane of the SignalTap II window, in the Data section of the pane check Segmented. In the dropdown menu beside Segmented, select 8 32 sample segments. This will result in a pane that looks like Figure 23.

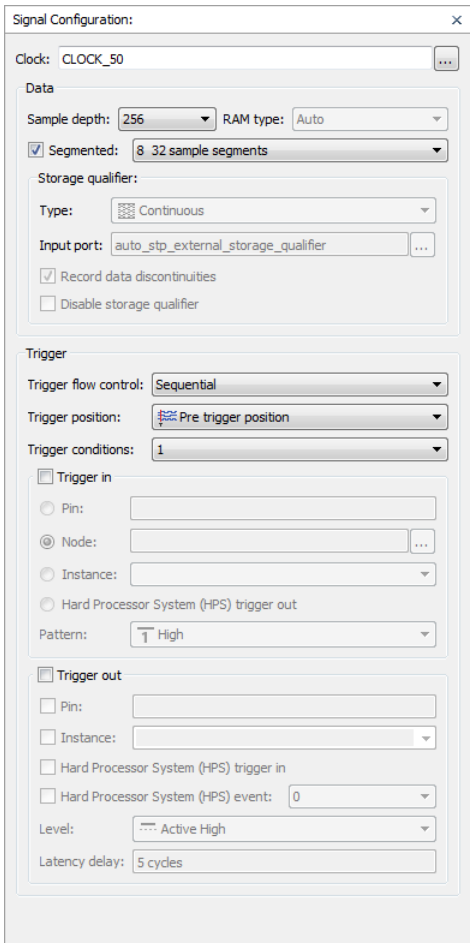

Figure 23. Select Segmented buffer acquisition mode with 8 32 sample segments.

4. Recompile and load the designed circuit onto the DE-series board. Now, we will be able to probe the design using the Segmented Acquisition mode.

5. Go back to the SignalTap II window and click Processing > Run Analysis. Now, flip SW[0] up and down, and in between flips change the values of the other 7 switches. After you have done this 8 times, the values in the buffer will be displayed in the data window, and this will display the values that the 8 switches were at around each trigger. A possible waveform is presented in Figure 24. This resulted from the user flipping up one more switch between each flip of SW[0].

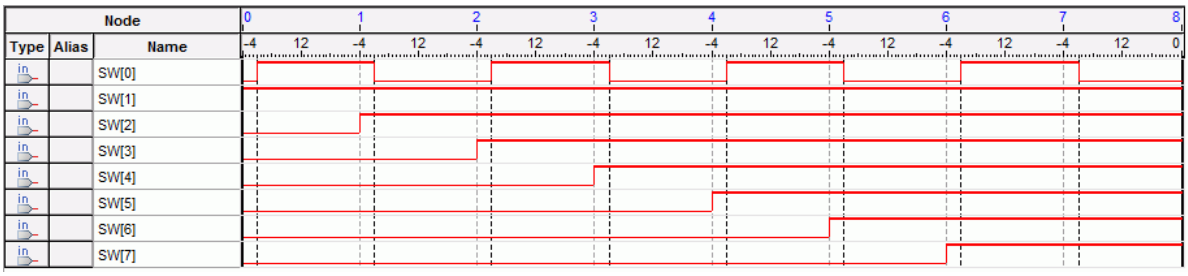

Figure 24. Possible waveforms that could result when using the Segmented Acquisition mode.

8.1 Use of Synthesis Keep Directive

Sometimes a design you create will have wires in it that the Quartus compiler will optimize away. A very simple example is the Verilog code below:

```
module threeInputAnd(SW, LEDR, CLOCK_50);
    input CLOCK_50;
    input [2:0] SW;
    output reg [0:0] LEDR;

    wire ab, abc /*synthesis keep*/;

    assign ab=SW[0]&SW[1];
    assign abc=ab&SW[2];

    always @ (posedge CLOCK_50)
    begin
        LEDR[0]<=abc;
    end
endmodule
```

Figure 25. Using the Synthesis Keep directive in Quartus II.

A diagram of this circuit is shown in Figure 26. The triangular symbols labeled **ab** and **abc** are buffers inserted by Quartus. They do not modify the signals passing through them.

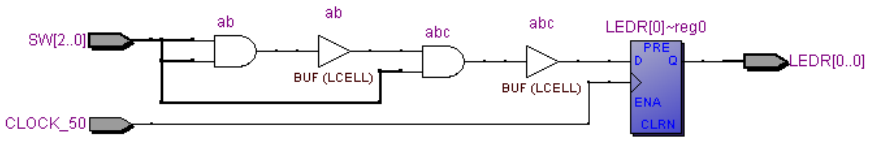

Figure 26. The circuit implemented by the code in Figure 25

We wish to instantiate a SignalTap II module that will probe the values of the inputs SW[2:0] and the outputs LEDR[2:0]. We also want to probe the internal wire **ab**. However, normally when this Verilog code is compiled (without the /*synthesis keep*/ directive), the wire **ab** is optimized away into one logic element, as in Figure 27.

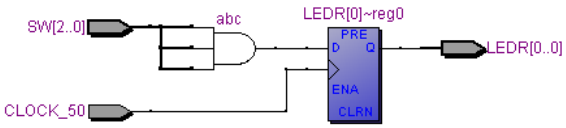

Figure 27. The same circuit without the Synthesis Keep directive.

If you wish to probe this internal wire, however, you will have to direct Quartus that you do not want this wire to be optimized away. To do so, place the text */*synthesis keep*/* on the line that declares the wire, right before the semicolon of the line. Figure 25 already contains this directive. We will now demonstrate how this wire can be probed:

1. Create a new Quartus project threeInputAnd and copy the Verilog code from Figure 25. Compile the project.
2. Go to Tools > SignalTap II Logic Analyzer, and then in the Setup pane of the SingalTap II window, right click and choose Add Nodes.
3. For the Filter field, select SignalTap II: pre-synthesis. Select |threeInputAnd| in the Look in drop-down menu and click the List button. Move the nodes **ab**, **SW[0]**, **SW[1]**, **SW[2]**, and **LEDR[0]** into the Selected Nodes list and then click OK.
4. In the Signal Configuration pane, select **CLOCK_50** as the clock signal.
5. Set a Trigger Condition to trigger when **ab** becomes high.
6. Import the relevant pin assignment file for the DE-series board (or assign the pins manually, as described in Section 7 of the Quartus II Introduction tutorials). For a DE2-115 board, this file is named *DE2-115.qsf*
7. Compile the project again.
8. Go to Tools > Programmer and load the circuit onto the DE-series board.

9. Open the SignalTap window again, and select the Data tab. Set all the switches on the DE-series board to the low position. Then, start the analysis by selecting Processing > Run Analysis.

10. Set the first two switches to the high position. The Trigger Condition should be satisfied.