SignalTap II with VHDL Designs

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:

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.

Doing 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 VHDL hardware description language. An equivalent tutorial is available for the reader who prefers the Verilog language.

PREREQUISITES
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 the Quartus II version 8.0, but other versions of the software can also be used.

1 Example Circuit

As an example, we will use the switch circuit implemented in VHDL in Figure 1. This circuit simply connects the first 8 switches on the DE2 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.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY switches IS
   PORT ( CLOCK_50    : IN     STD_LOGIC;
          SW          : IN     STD_LOGIC_VECTOR(7 DOWNTO 0);
          LEDR        : OUT    STD_LOGIC_VECTOR(7 DOWNTO 0)); - - red LEDs
END switches;

ARCHITECTURE Behavior OF switches IS
BEGIN
   PROCESS (CLOCK_50)
      BEGIN
      IF(RISING_EDGE(CLOCK_50)) THEN
         LEDR <= SW;
      END IF;
   END PROCESS;
END Behavior;
```

Figure 1. The switch circuit implemented in VHDL code.

Implement this circuit as follows:

- Create a project *switches*.

- Include a file *switches.vhd*, which corresponds to Figure 1, in the project. For convenience, this file is provided in the directory *DE2_tutorials\design_files*, which can be found on Altera's DE2 web pages.
- Choose the Cyclone II EP2C35F672C6 device, which is the FPGA chip on Altera's DE2 board.
- Import the csv file called *DE2_pin_assignments.csv* by clicking Assignments->Import Assignments. The node names used in the sample circuit correspond to the names used in this file.
- Compile the design.

2 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 2. Choose SignalTap II Logic Analyzer File and click OK.

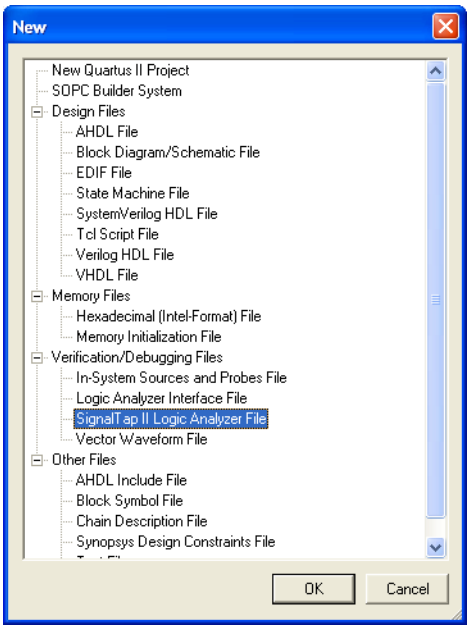

Figure 2. Need to prepare a new file.

2. The SignalTap II window with the Setup tab selected is depicted in Figure 3. Save the file under the name *switches.stp*. In the dialog box that follows (Figure 4), click OK. For the dialog "Do you want to enable SignalTap II file 'switches.stp' for the current project?" click Yes (Figure 5). 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 6. 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.

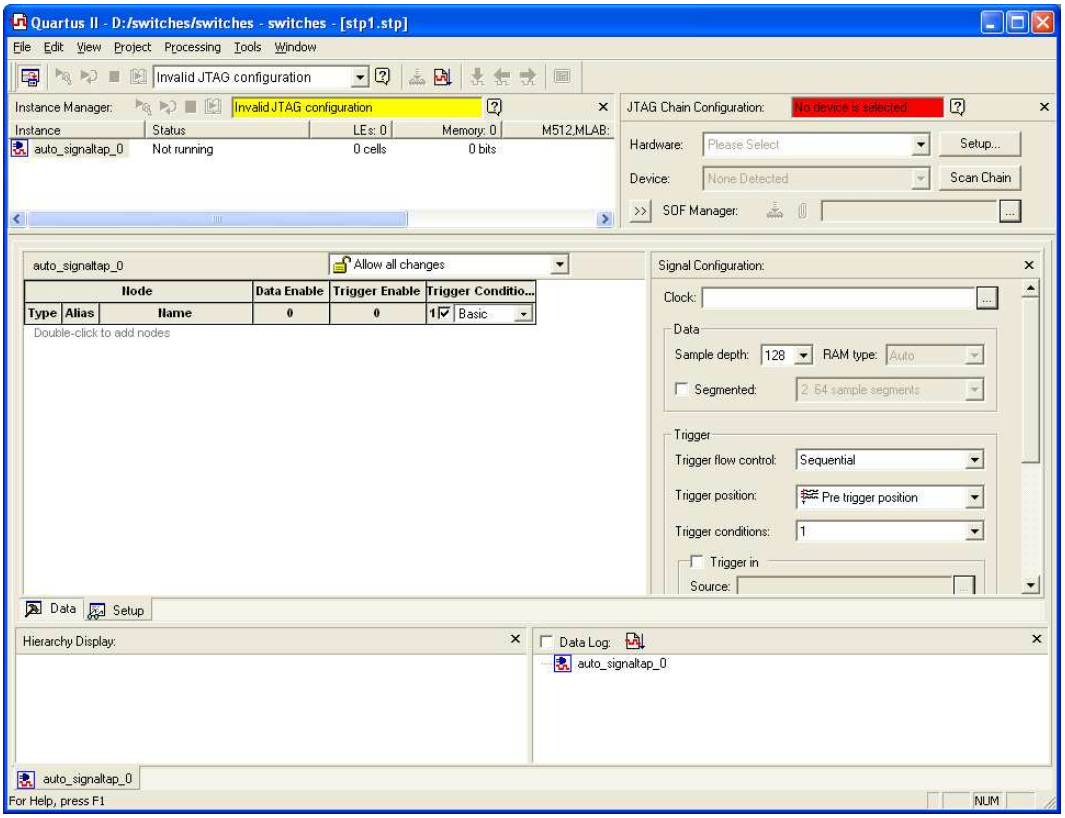

Figure 3. The SignalTap II window.

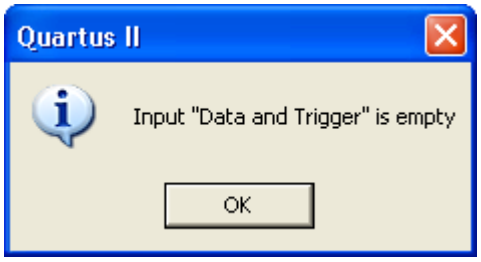

Figure 4. Click OK to this dialog.

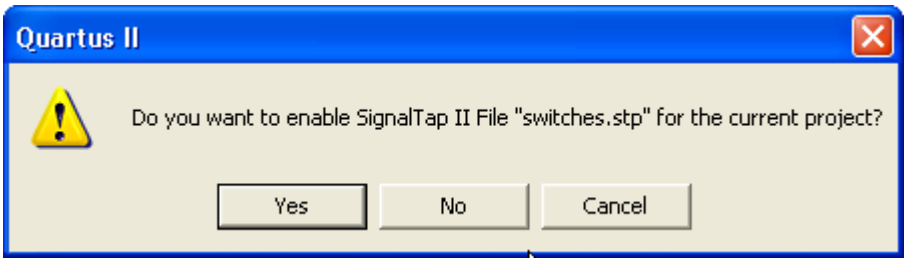

Figure 5. Click Yes to this dialog.

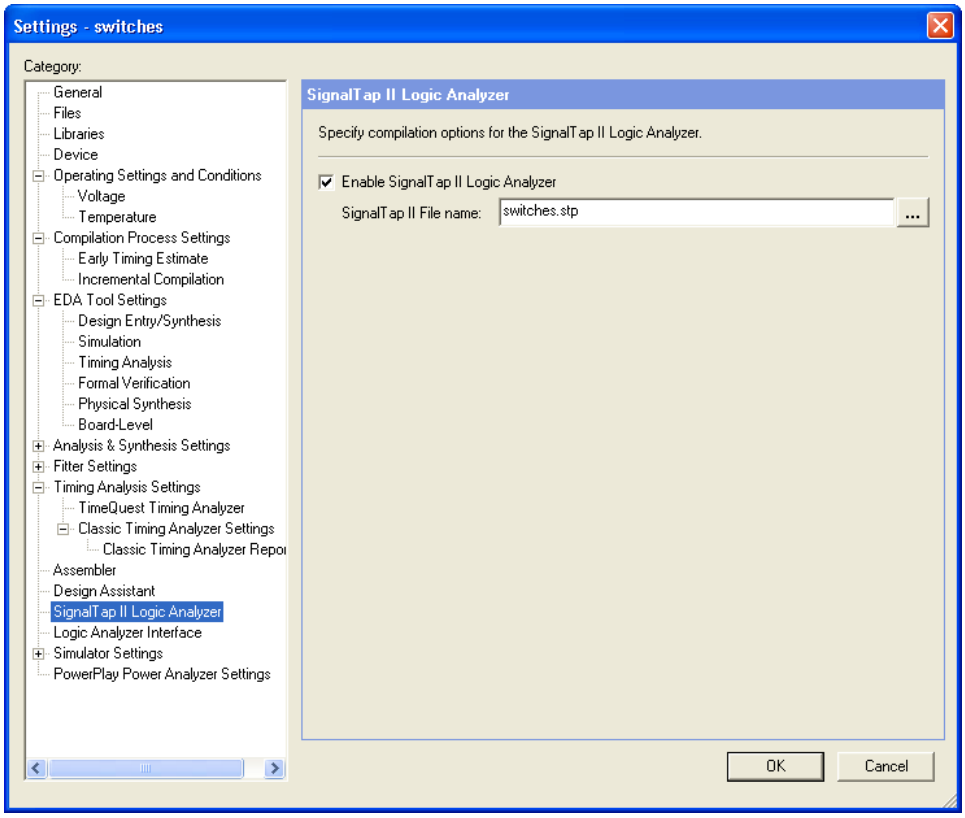

Figure 6. The SignalTap II Settings window.

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 7. For the Filter field, select SignalTap II: pre-synthesis. 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.

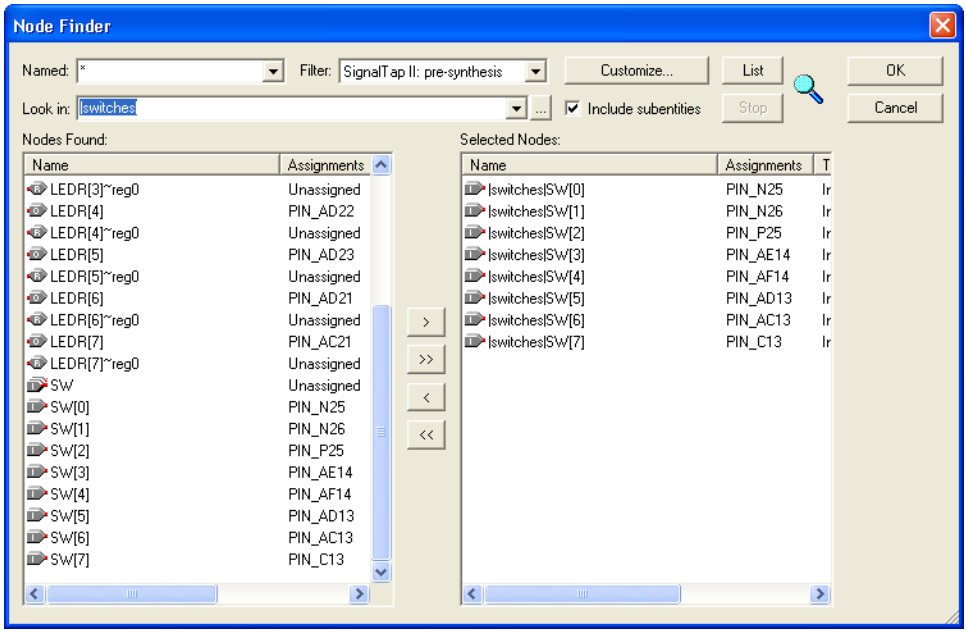

Figure 7. Add nodes in the Node Finder window.

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 11. Click OK.

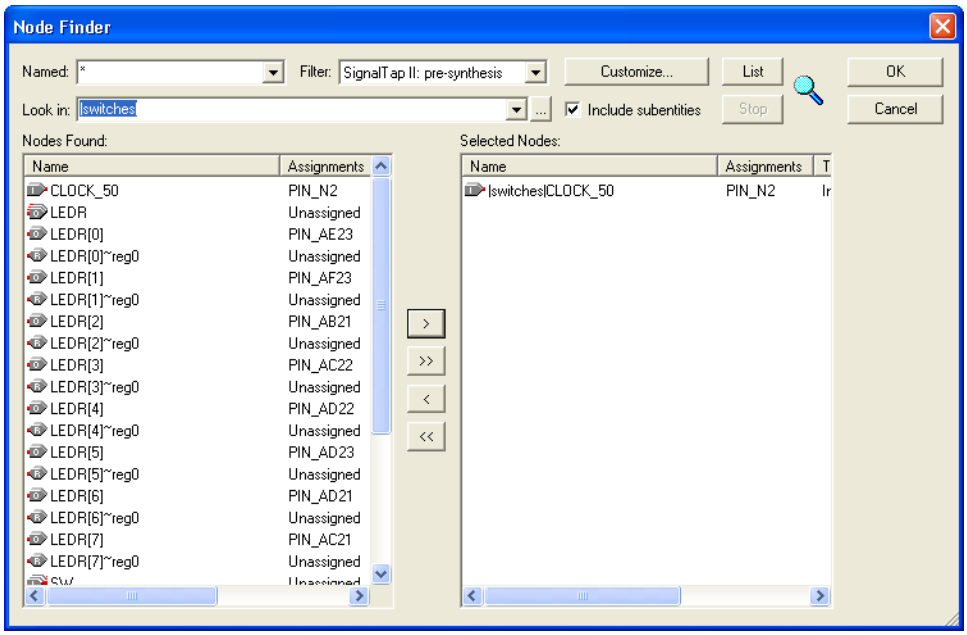

Figure 8. Set CLOCK_50 as the clock for this SignalTap instance.

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. 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 DE2 board is set to high, as shown in Figure 9. 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.

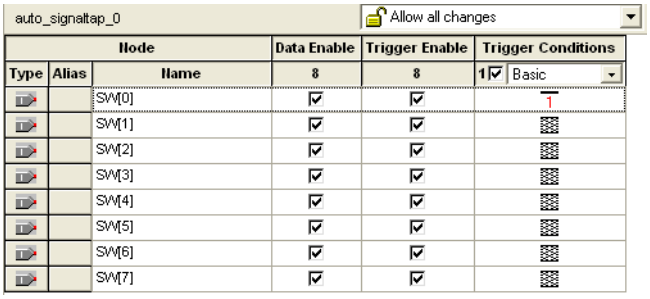

Figure 9. Setting the trigger conditions.

6. For SignalTap II to work, we need to properly set up the hardware. First, make sure the DE2 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 10. Double click USB-Blaster in the Available Hardware Items menu, then click Close.

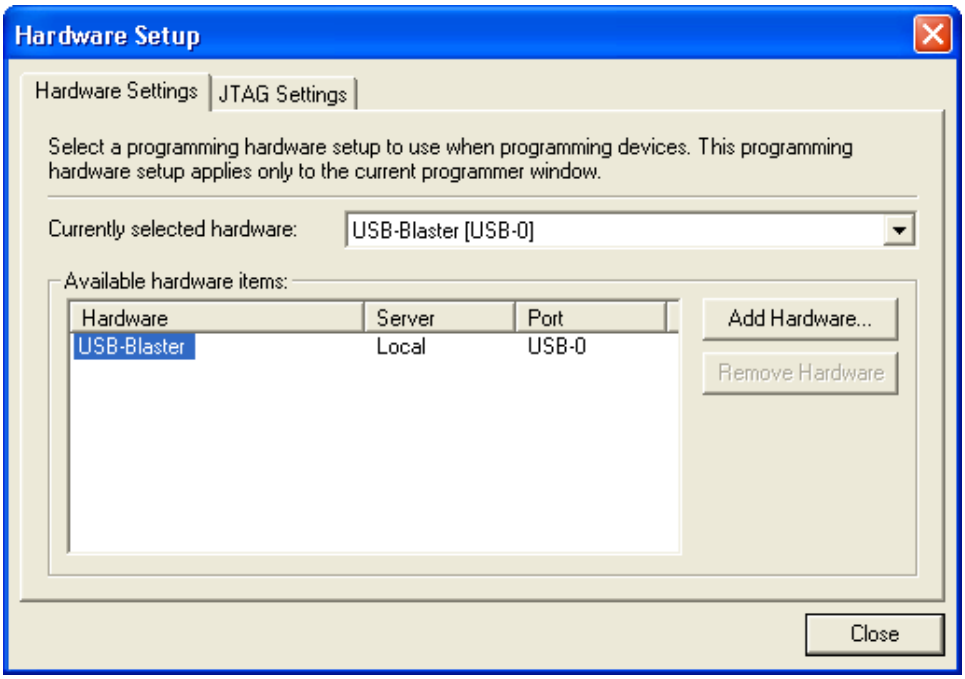

Figure 10. Setting up hardware.

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 DE2 board.

3 Probing the Design Using SignalTap II

Now that the project with SignalTap II instantiated has been loaded onto the DE2 board, we can probe the nodes as we would with an external logic analyzer.

1. On the DE2 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 11. 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).

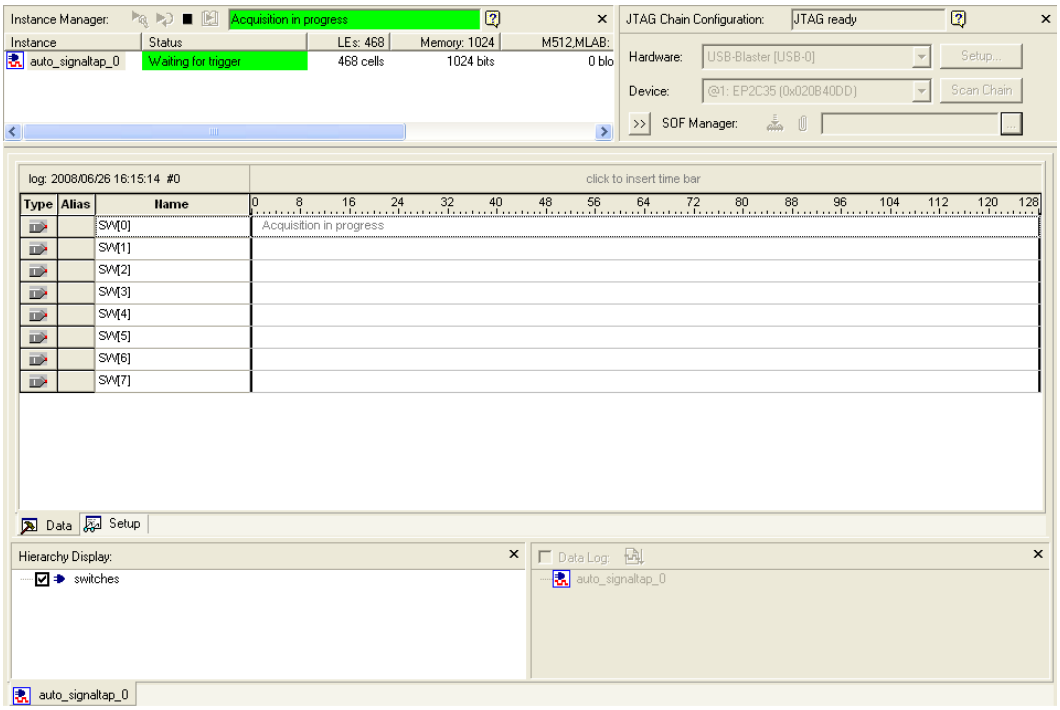

Figure 11. SignalTap II window after Run Analysis has been clicked.

3. Now, to observe the trigger feature of the Logic Analyzer, set Switch 0 on the DE2 board to high. The data window of the SignalTap II window should display the image in Figure 12. 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.

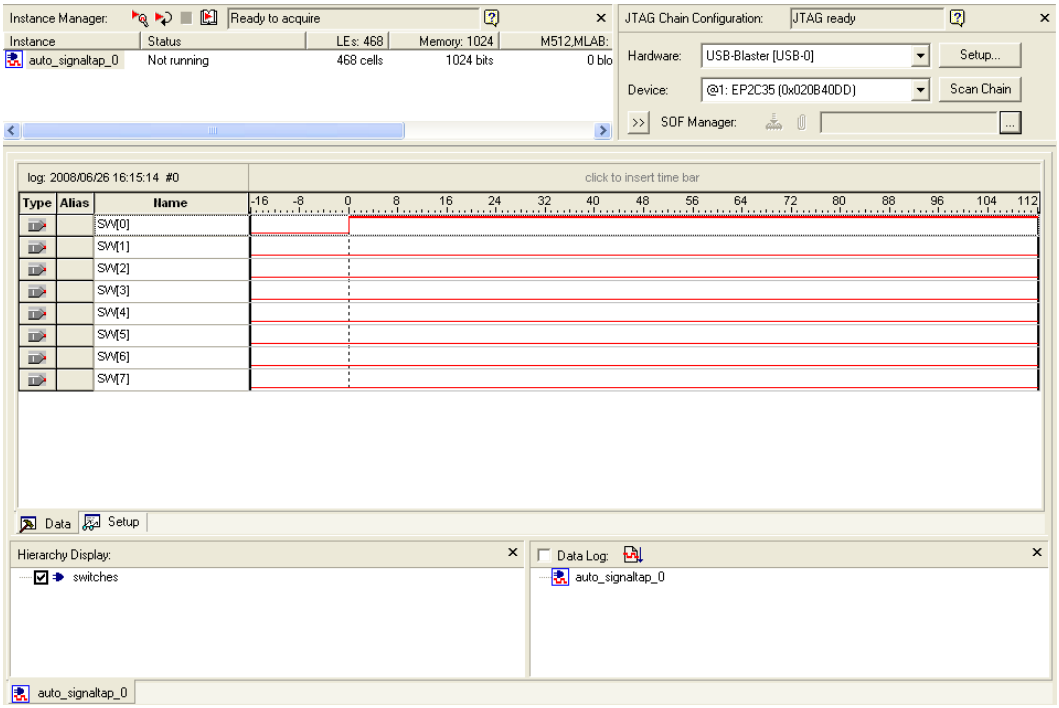

Figure 12. Graphical display of values after trigger condition is met.

4 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 as well as how to create advanced triggering options.

4.1 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 13. This modifies the node list window by creating three new Trigger Conditions columns.

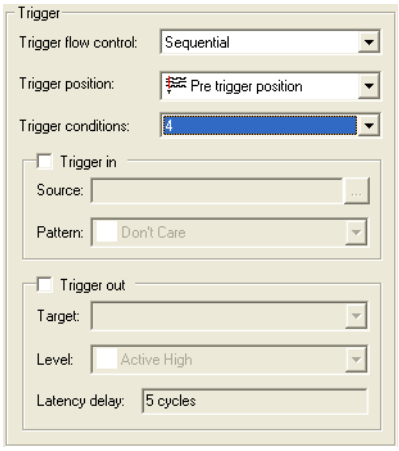

Figure 13. Set trigger levels to 4.

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 14.

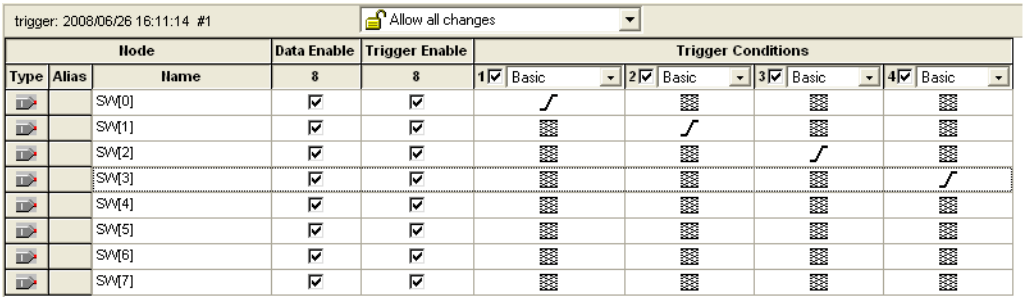

Figure 14. Multiple trigger levels set.

4. Now, recompile the design and load it onto the DE2 board again.

5. Go back to the SignalTap II window, click on the Data tab, and then click 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 15. Experiment by following the procedure outlined in this section to set up other trigger conditions and use the DE2 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.

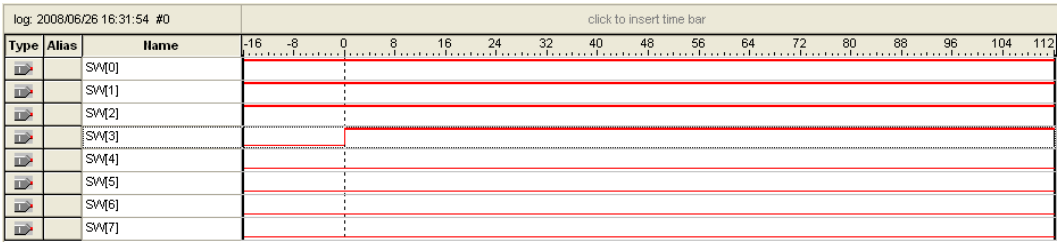

Figure 15. Logic Analyzer display when all four trigger conditions have been met.

4.2 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 everytime 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 16. This will immediately bring up the window in Figure 17. This window allows you to create a logic circuit using the various nodes that you are probing with SignalTap.

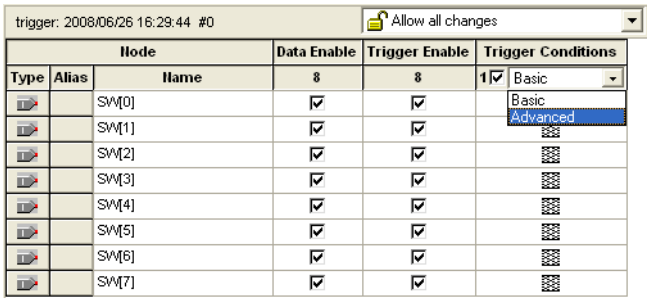

Figure 16. Select Advanced from the Trigger Level dropdown menu.

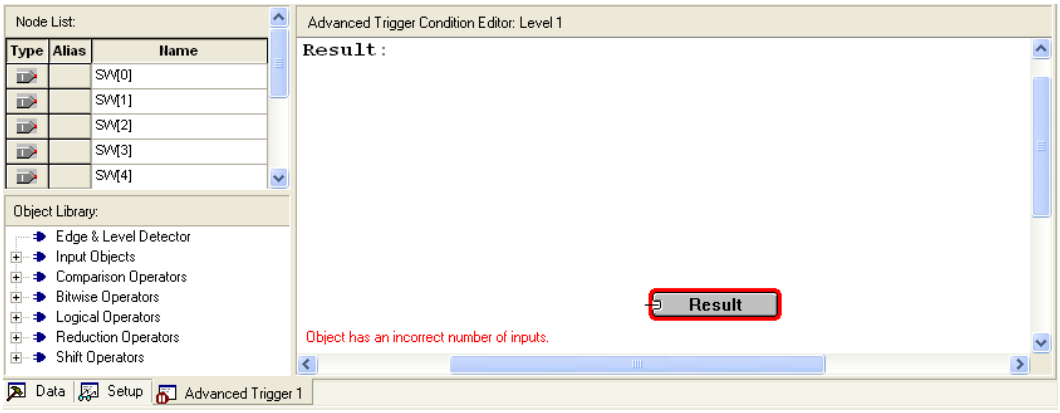

Figure 17. The Advanced Trigger editing window.

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 18. Note that you can also drag and drop each node individually.

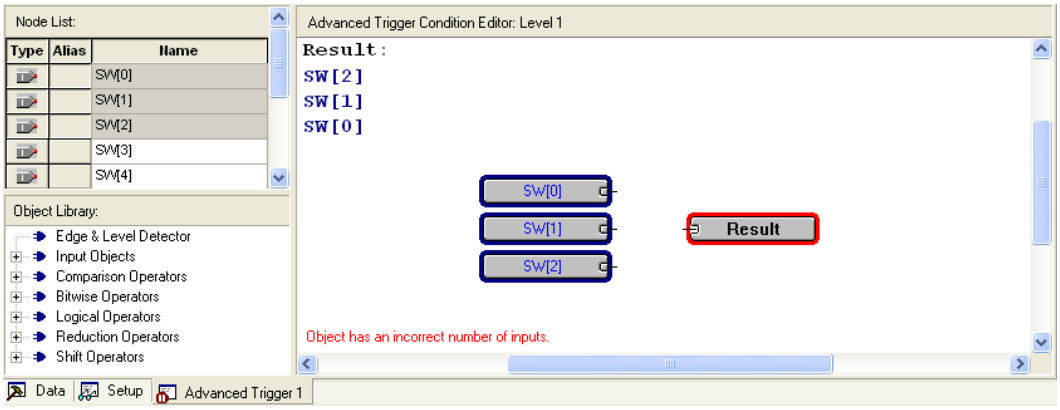

Figure 18. The three input nodes of interest dragged into the Advanced Trigger Editing Window.

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 and select Logical Or, as in Figure 19. Then drag and drop the operator into the editing window.

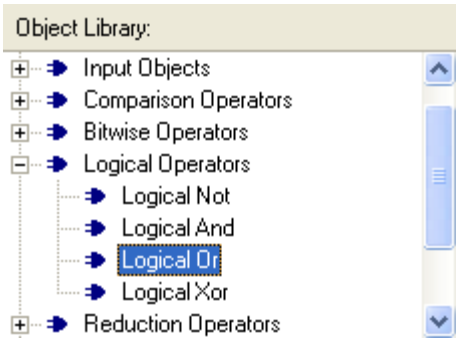

Figure 19. Select the Logical Or operator from the Object Library window and drag this into the editing window.

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 20. 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.

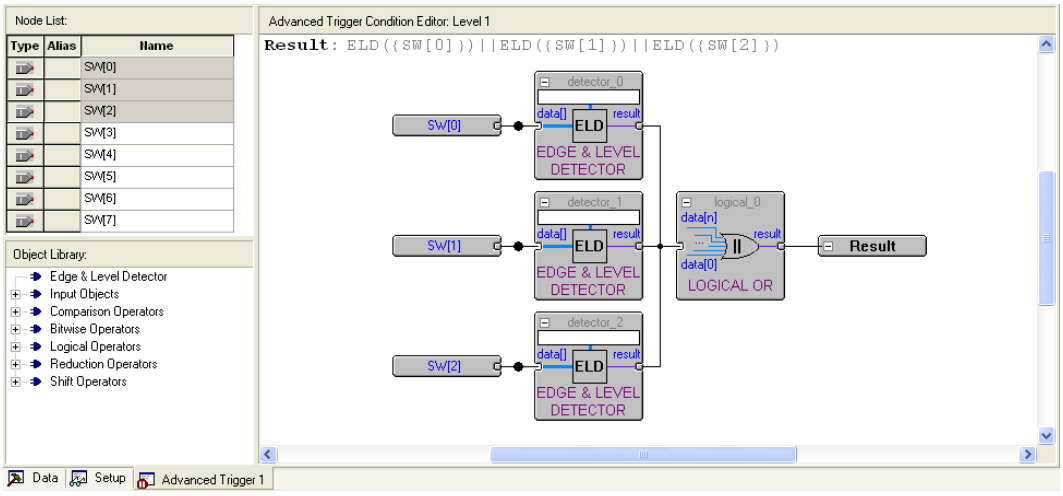

Figure 20. Arrange the elements to create a circuit that looks like this.

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 21. 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.

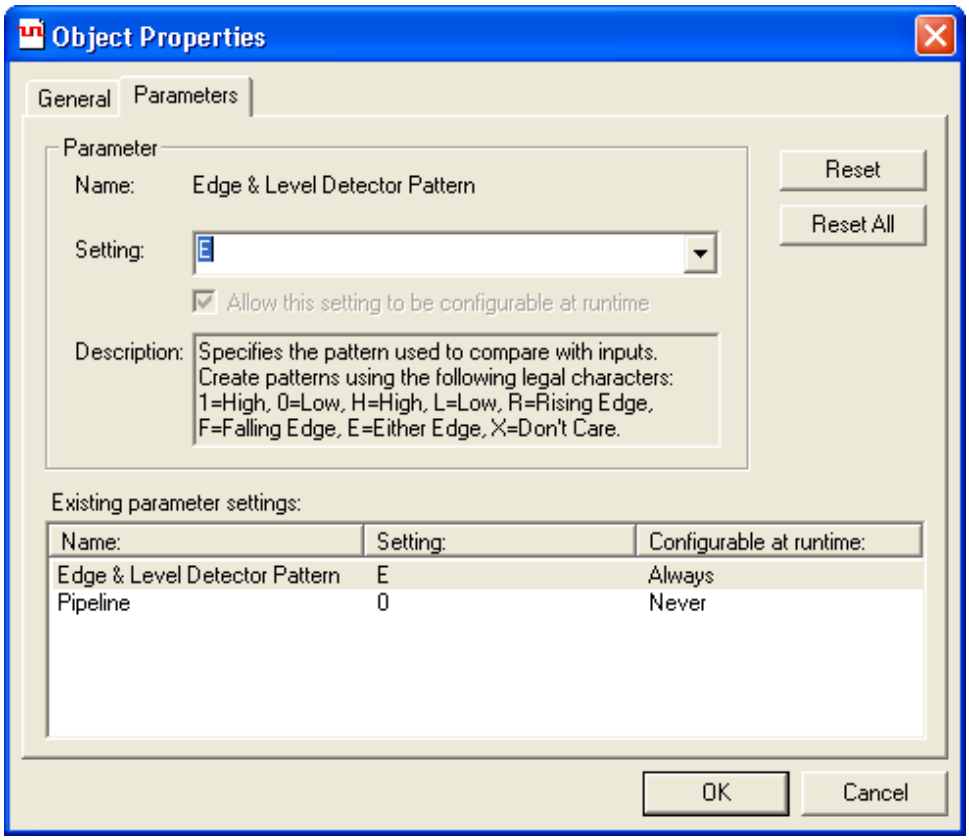

Figure 21. Type E in the setting box so that the function triggers on both rising and falling edges.

8. To test this Advanced trigger condition, compile the designed circuit again and load it onto the DE2 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.

5 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 level. 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. At a sample depth of 256 you should have no problems in compiling our example design.

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 22.

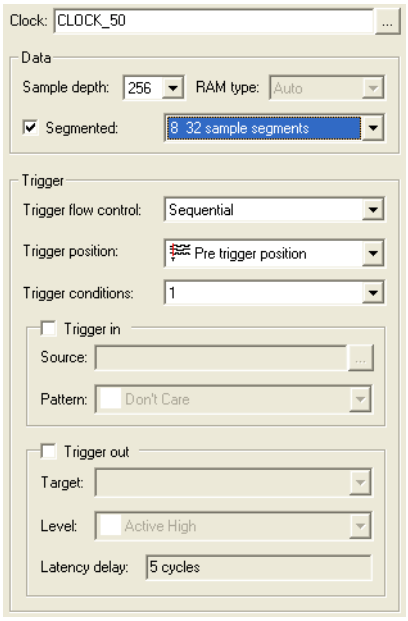

Figure 22. Select Segmented buffer acquisition mode with 8 32 sample segments

4. Recompile and load the designed circuit onto the DE2 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 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 23. This resulted from the user flipping up one more switch between each flip of SW[0].

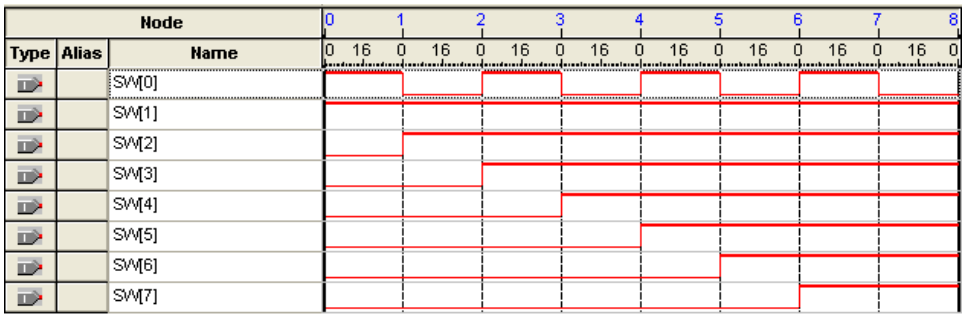

Figure 23. Possible waveforms that could result when using the Segmented Acquisition mode.

5.1 Use of 'Keep' Attribute

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

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY threeInputAnd IS
   PORT ( CLOCK_50     : IN     STD_LOGIC;
          SW           : IN     STD_LOGIC_VECTOR(2 DOWNTO 0);
          LEDR         : OUT    STD_LOGIC_VECTOR(0 DOWNTO 0));
END threeInputAnd;

ARCHITECTURE Behavior OF threeInputAnd IS
   SIGNAL ab, abc : STD_LOGIC;
   ATTRIBUTE keep : BOOLEAN;
   ATTRIBUTE keep OF ab, abc : SIGNAL IS true;
   BEGIN
      ab <= SW(0) AND SW(1);
      abc <= ab AND SW(2);

      PROCESS (CLOCK_50)
      BEGIN
         IF (RISING_EDGE(CLOCK_50)) THEN
            LEDR(0) <= abc;
         END IF;
      END PROCESS;
END Behavior;
```

Figure 24. Using the 'keep' attribute in Quartus II.

A diagram of this circuit is shown in Figure 25. The triangular symbols labeled **ab** and **abc** are buffers inserted by Quartus. They do not modify the signals passing through them.

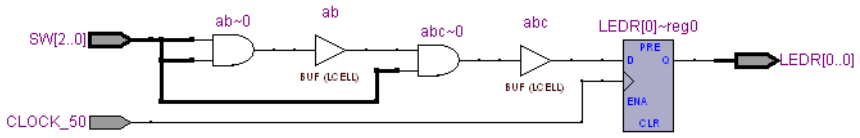

Figure 25. The circuit implemented by the code in Figure 24.

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 VHDL code is compiled (without the two ATTRIBUTE lines), the wire **ab** is optimized away into one logic element, as in Figure 26.

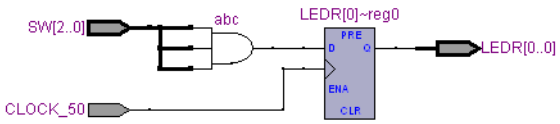

Figure 26. The same circuit without the 'keep' attribute.

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, first an attribute called 'keep' of type BOOLEAN needs to be declared. This is what the first line (*ATTRIBUTE keep : BOOLEAN;*) is for. Then, the attribute needs to be applied to the desired signals (in this case, signals **ab** and **abc**). This is achieved with the second line (*ATTRIBUTE keep OF ab, abc : SIGNAL IS true;*). Figure 24 already contains these lines. We will now demonstrate how this wire can be probed:

1. Create a new Quartus project threeInputAnd and copy the VHDL code from Figure 24. 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. 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 pin assignment file *DE2_pin_assignments.csv* (or assign the pins manually, as described in Section 5 of the Quartus II Introduction tutorials).
7. Compile the project again.
8. Go to Tools > Programmer and load the circuit onto the DE2 board.
9. Open the SignalTap window again, and select the Data tab. Set all the switches on the DE2 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.