Hardware Debugging

Introduction

In this lab you will use the uart\_led design that was introduced in the previous labs. You will use Mark Debug feature and also the available Integrated Logic Analyzer (ILA) core (in IP Catalog) to debug the hardware.

Objectives

After completing this lab, you will be able to:

* Use the Integrated Logic Analyzer (ILA) core from the IP Catalog as a debugging tool
* Use Mark Debug feature of Vivado to debug a design
* Use hardware debugger to debug a design

Procedure

This lab is broken into steps that consist of general overview statements providing information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab1.

Design Description

The design consists of a uart receiver receiving the input typed on a keyboard and displaying the binary equivalent of the typed character on the 8 LEDs. When a push button is pressed, the lower and upper nibbles are swapped. The block diagram is as shown in **Figure 1**.

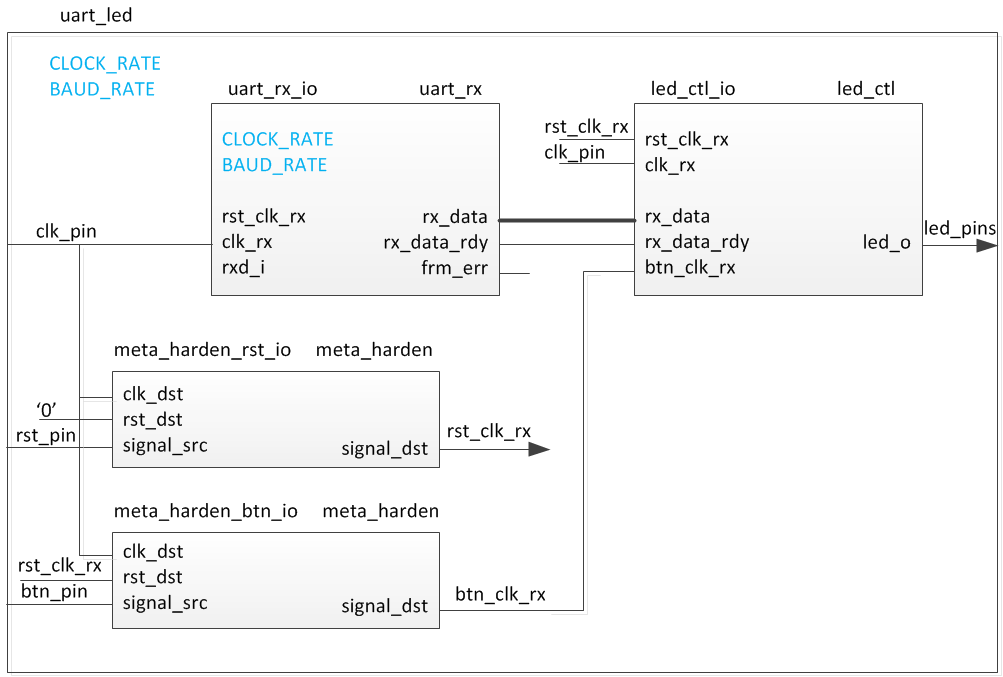


Figure 1. The Completed Design

General Flow

Step 5:

Debug in Hardware

Step 4:

Implement and Generate Bitstream

Step 3:

Synthesize the Design and Mark Debug

Step 2: Add the ILA Core

Step 1: Create a Vivado Project using IDE

1. Create a Vivado Project using IDE Step 1
   1. Launch Vivado and create a project targeting the XC7A100TCSG324-1 (Nexys4) or the XC7A35TCPG236-1 (Basys3) device and using the Verilog HDL. Use the provided Verilog and Xilinx Design Constraints source files from the <*2014\_2\_artix7\_sources>\lab6* directory.

|  |
| --- |
| References to **<2014\_2\_artix7\_labs>** is a placeholder for the **c:\xup\fpga\_flow\2014\_2\_artix7\_labs** directory and **<2014\_2\_artix7\_sources>** is a placeholder for the **c:\xup\fpga\_flow\2014\_2\_artix7\_sources** directory.  Reference to **<board>** means either the **Nexys4** or the **Basys3**. |

* + 1. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2014.2 > Vivado 2014.2**
    2. Click **Create New Project** to start the wizard. You will see *Create A New Vivado Project* dialog box. Click **Next**.
    3. Click the Browse button of the *Project location* field of the **New Project** form, browse to **<2014\_2\_artix7\_labs>**, and click **Select**.
    4. Enter **lab6** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.
    5. Select **RTL Project** option in the *Project Type* form, and click **Next**.
    6. Using the drop-down buttons, select **Verilog** as the *Target Language* and *Simulator Language* in the *Add Sources* form.
    7. Click on the **Add Files…** button, browse to the **<2014\_2\_artix7\_sources>\lab6** directory, select all the Verilog files *(led\_ctl.v, meta\_harden.v, uart\_baud\_gen.v, uart\_led.v, uart\_rx.v, and uart\_rx\_ctl.v),* click **OK**, and then click **Next**.
    8. Click **Next** to get to the *Add Cons*traints form.
    9. You will see uart\_led\_timing.xdc and both uart\_led\_pins\_<board>.xdc files have automatically been included. Delete the XDC file not targeting the board by highlighting on it and clicking “X” on the right hand side of the GUI.
    10. Click **Next.**
    11. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section, select the **XC7A100TCSG324-1** part for the Nexys4 or the **XC7A35TCPG236-1** for the Basys3. Click **Next**.
    12. Click **Finish** to create the Vivado project.

1. Add the ILA Core Step 2
   * 1. Click **IP Catalog** under the *Project Manager* tasks of the *Flow Navigator* pane.
     2. The catalog will be displayed in the Auxiliary pane.
     3. Expand the **Debug & Verification > Debug** folders and double-click the **ILA** entry.

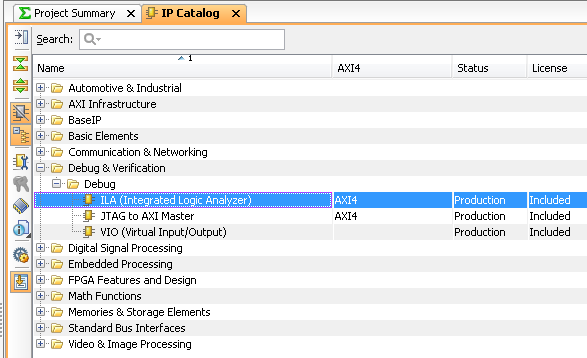


Figure 2. ILA in IP Catalog

This exercise will be connecting the ILA core/component to the LED port which is 8-bit wide.

* + 1. Change the component name to **ila\_led**.
    2. Change the *Number of Probes* to **2**.

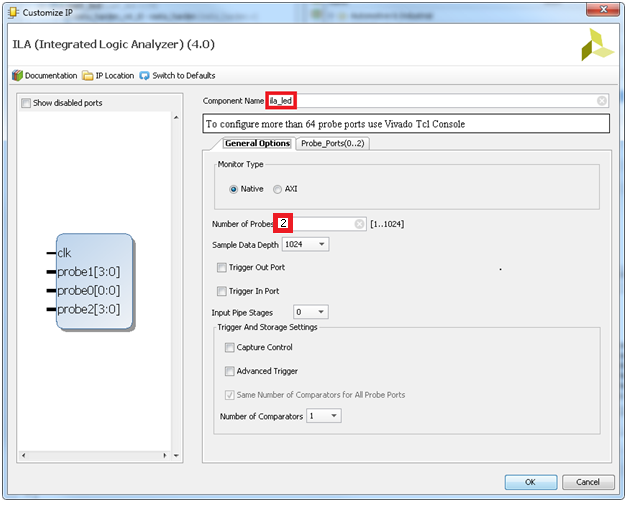


Figure 3. Setting the component name and the number of probes field

* + 1. Select the *Probe Ports* tab and change the PROBE1 port width to **8**, leaving the PROBE0 width to **1**.

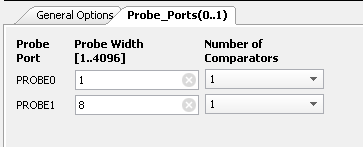


Figure 4. Setting the probes widths

* + 1. Click **OK**.

The Generate Output Products dialog box will appear.

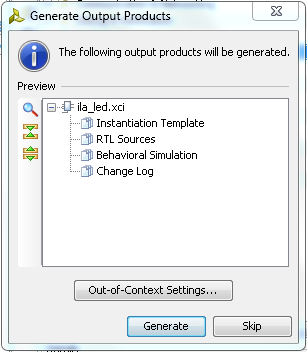


Figure 5. The Generate Output Products

* + 1. Click the **Generate** button to generate the core including the instantiation template. Notice the core is added to the *Design Sources* view.

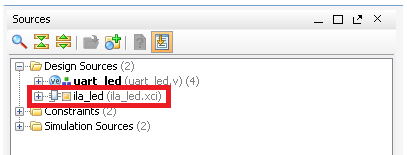


Figure 6. Newly generate ila core added in the design source

* + 1. Select the **IP Sources** tab, expand the **IP(1) > ila\_led > Instantiation Template**, and double-click the **ila\_led.veo** entry to see the instantiation template.
    2. Instantiate the ila\_led in design by copying lines 57 – 61 and pasting to ~line 104 (before “endmodule” on the last line) in the uart\_led.v file.
    3. Change *your\_instance\_name* to **ila\_led\_i0.**
    4. Change the following port names in the Verilog code to connect the ila to existing signals in the design:

.CLK(CLK) .CLK(clk\_pin)

.PROBE0(PROBE0) .PROBE0(rx\_data\_rdy)

.PROBE1(PROBE1) .PROBE1(led\_pins)

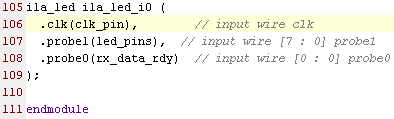


Figure 7. Instantiating the ILA Core in the uart\_led.v

* + 1. Select **File > Save File**.

Notice that the ILA Core instance is in the design hierarchy.

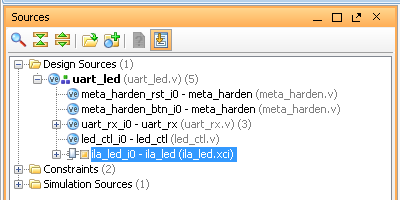


Figure 8. ILA Core added to the design

1. Synthesize the Design and Mark Debug Step 3
   1. Synthesize the design. Open the synthesized design. View the schematic. Add Mark Debug on the rx\_data bus between the uart\_rx\_i0 and led\_ctl\_i0 instances.
      1. Click on **Run Synthesis** under the *Synthesis* tasks of the *Flow Navigator* pane.

The synthesis process will be run on the uart\_led.v and all its hierarchical files. When the process is completed a *Synthesis Completed* dialog box with three options will be displayed.

* + 1. Select the *Open Synthesized Design* option and click **OK**.
    2. Click on **Schematic** under the *Synthesized Design* tasks of *Synthesis* tasks of the *Flow Navigator* pane to view the synthesized design in a schematic view.
    3. Select the **rx\_data** bus between the *uart\_rx\_i0* and the *led\_ctl\_i0* instances, right-click, and select **Mark Debug**.

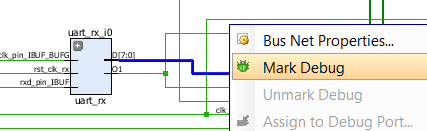


Figure 9. Marking a bus to debug

* + 1. A warning message indicating that the synthesis will be shown as out-of-date will appear. Click **OK**.

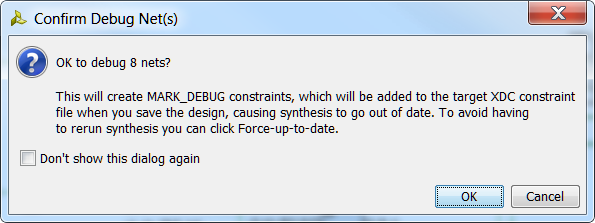


Figure 10. Confirm Debug Net(s) dialog box

* + 1. Select the **Netlist** tab and notice that the nets which are marked/assigned for debugging have a debug icon next to them.

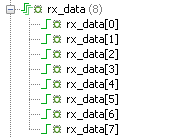


Figure 11. Nets with debug icons

* + 1. Select the **Debug** layout or **Layout > Debug**.

Notice that the **Debug** tab is visible in the Console pane showing Assigned and Unassigned Debug Nets groups.

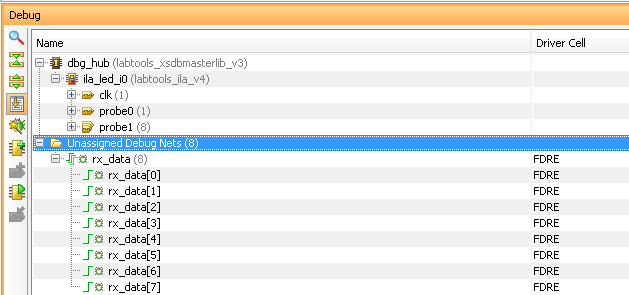


Figure 12. Debug tab showing assigned and unassigned nets

* + 1. Either click on the  button in the left vertical tool buttons of the Debug pane, or right-click on the *Unassigned Debug Nets* and select the **Set up Debug…** option.
    2. In the Set Up Debug wizard click **Next.**
    3. There is an error message that there is no identified clock domain for the nets. Press Ctrl and select/highlight all three nets. Click on the **Select Clock Domain** button  on the side bar to assign a clock domain to the signals.
    4. In the Select Clock Domain window, there is an info message that nothing was found for GLOBAL\_CLOCK. Click **OK** to close the box.
    5. In the Select Clock Domain window, select **ALL\_CLOCK** from the drop down menu  and **uncheck** Display unique nets. Notice a list of clocks found. Select **xlnx\_opt**\_ as the clock domain. Click **OK** to exit the window.
    6. The nets are now listed with a xlnx\_opt\_ selected as the clock domain. Since some of the nets are already assigned, we can remove them from this setup. Select and delete the led\_pins(8) and rx\_data\_rdy nets. Click **Remove Nets** on the side bar to remove the previously mentioned nets from the list.

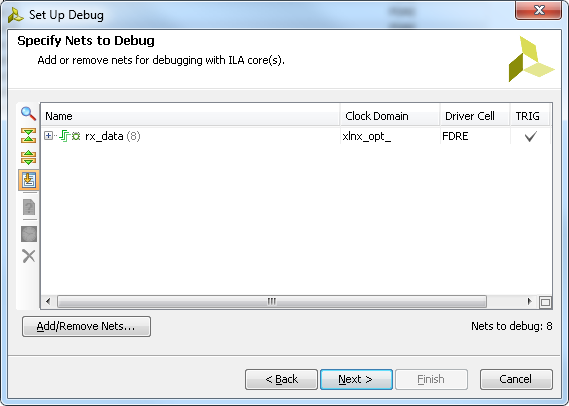


Figure 13. The remaining nets after removing already assigned nets in the Set Up Debug wizard

* + 1. Click **Next** and again **Next** (leaving everything as defaults) then **Finish**.
    2. In the Synthesized Design Schematic, click on the net on the output side of the IBUF for the input pin named clk\_pin. Hover over the now highlighted net and notice the name is xlxn\_opt\_. This is the clock net selected for the debug nets earlier.

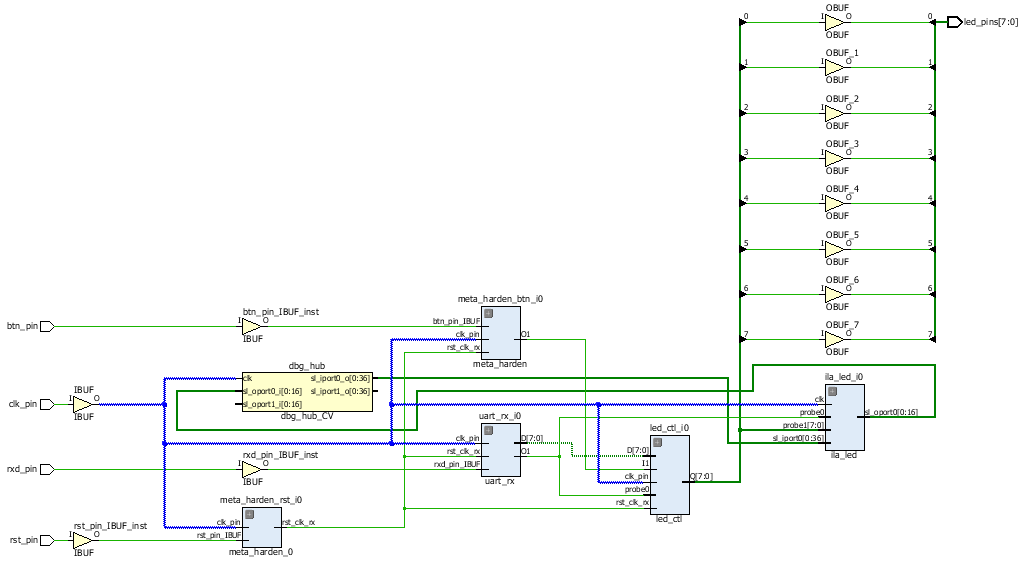


Figure 14. Locating xlnx\_opt\_ in the design

* + 1. Right click on **uart\_led\_pins.xdc** in the sources pane and select **Set as Target Constrain File**. This will save the changes to the file
    2. Select **File > Save Constraints** and click **OK.**
    3. Open uart\_led\_pins.xdc and notice the debug nets have been appended to the bottom of the file.
    4. Perform this step if synthesis is not already up-to-date, in the **Design Runs** tab, right-click on the *synth\_1* and select **Force Up-to-Date** to ensure that the synthesis process is not re-run, since the design was not changed.

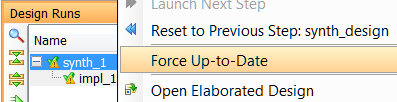


Figure 15. Forcing the design to be up-to-date

1. Implement and Generate Bitstream Step 4
   1. Generate the bitstream.
      1. Click on the **Generate Bitstream** to run the implementation and bit generation processes.
      2. Click **Yes** to run the implementation processes.
      3. When the bitstream generation process has completed successfully, a box with four options will appear. Select the **Open Hardware Manager** option and click **OK**.
2. Debug in Hardware Step 5
   1. Connect the board and power it ON. Open a hardware session, and program the FPGA.
      1. Make sure that the Micro-USB cable is connected to the JTAG PROG connector (next to the power supply connector). Make sure that the jumper is set to select USB power (JP3 on the Nexys4 and JP2 on the Basys3).
      2. Turn ON the power.
      3. Select the *Open Hardware Manager* option and click **OK**.

The Hardware Manager window will open indicating “unconnected” status.

* + 1. Click on the **Open a new hardware target** link.

You can also click on the **Open recent target** link if the board was already targeted before.

* + 1. Click **Next**  to see the Vivado Hardware Server Settings form.
    2. Click **Next** with the Target Hardware selected.

The JTAG cable which uses the xilinx\_tcf should be detected and identified as a hardware target. It will also show the hardware devices detected in the chain.

* + 1. Click **Next** and **Finish**.
    2. The Hardware Session status changes from Unconnected to the server name and the device is highlighted. Also notice that the Status indicates that it is not programmed.
    3. Select the device and verify that the **uart\_led.bit** is selected as the programming file in the General tab.
  1. Start a terminal emulator program such as TeraTerm or HyperTerminal. Select an appropriate COM port (you can find the correct COM number using the Control Panel). Set the COM port for 115200 baud rate communication. Program the FPGA and verify the functionality.
     1. Start a terminal emulator program such as TeraTerm or HyperTerminal.
     2. Select an appropriate COM port (you can find the correct COM number using the Control Panel).
     3. Set the COM port for 115200 baud rate communication.
     4. Right-click on the FPGA, and select **Program Device…** and click **Program**.

The programming bit file be downloaded and the DONE light will be turned ON indicating the FPGA has been programmed

The Debug Probes window will also open. Notice that there are two debug cores, hw\_ila\_1 and hw\_ila\_2.

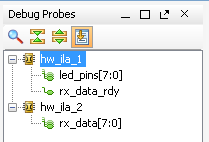


Figure 16. Debug probes

The hardware session status window also opens showing that the FPGA is programmed having two ila cores with the idle state.

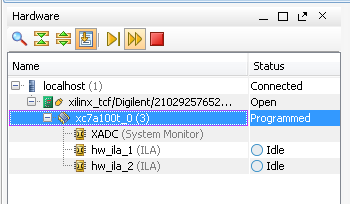


Figure 17. Hardware session status for the Nexys4

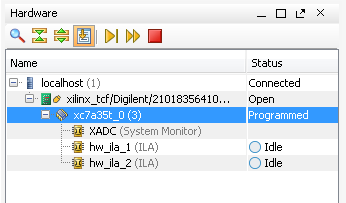


Figure 17. Hardware session status for the Basys3

* + 1. Select the target FPGA (XC7A100T or XC7A35T), and click on the **Run Trigger Immediate** button to see the signals in the waveform window.

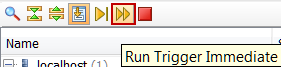


Figure 18. Opening the waveform window

Two waveform windows will be created, one for each ila; one ila is of the instantiated ILA core and another for the MARK DEBUG method.

* 1. Setup trigger conditions to trigger on a write to led port (rx\_data\_rdy=1) and the trigger position to 512. Arm the trigger.
     1. In the **Debug Probes** window, right-click on the **rx\_data\_rdy** and select **Add Probes to Basic Trigger Setup**.

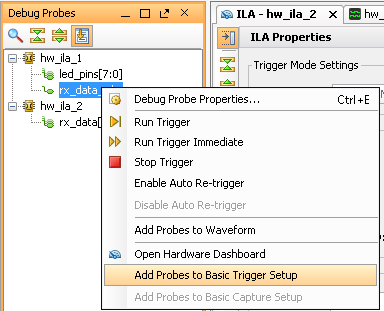


Figure 19. Adding a signal to trigger setup

* + 1. In the Basic Trigger Setup window, click on the drop-down button, set the compare value (== [B] X) and change the value from x to 1. Click **OK**.

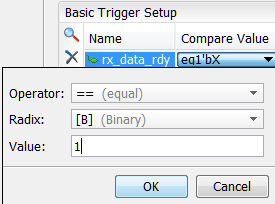


Figure 20. Setting the trigger condition

* + 1. Set the trigger position of the *hw\_ila\_1* to **512**.

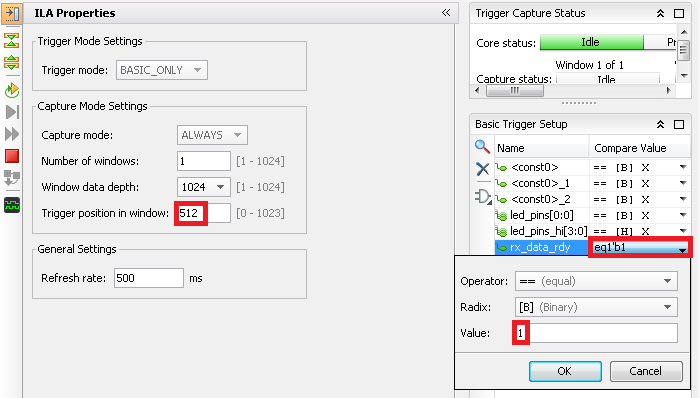


Figure 21. Setting up the trigger position

* + 1. Similarly, set the trigger position of the *hw\_ila\_2* to **512**.
    2. Select the hw\_ila\_1 in the Hardware window and then click on the Run Trigger () button. Observe that the hw\_ila\_1 core is armed and showing the status as **Waiting for Trigger**.

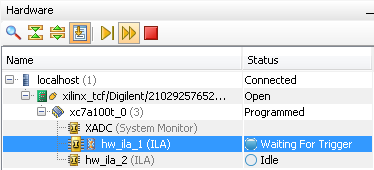


Figure 22. Hardware analyzer running in capture mode

* + 1. In the terminal emulator window, type a character, and observe that the hw\_ila\_1 status changes from capturing to idle as the rx\_data\_rdy became 1.
    2. Select the hw\_ila\_data\_1.wcfg window and see the waveform. Notice that the rx\_data\_rdy goes from 0 to 1 at 512th sample and the led\_ctl\_i0 [7:0] bits also changes from 0 to the bit pattern corresponding to the character you typed..



Figure 23. Zoomed waveform view

* + 1. Add the hw\_ila\_2 probes to the trigger window of the hw\_ila\_2 and change the trigger condtion for rx\_data[7:0]’s Radix from Hexadecimal to binary. Change XXXX\_XXXX to **0101\_0101** (for the ASCII equivalent of U).

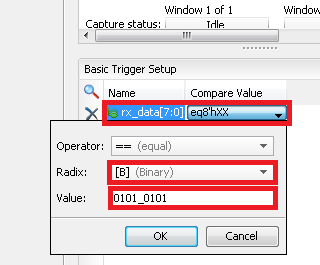


Figure 24. Setting up trigger condition for a particular input pattern

* + 1. In the Hardware window, right-click on the **hw\_ila\_2** and select **Run Trigger,** and notice that the status of the hw\_ila\_2 changes from *idle* to *Waiting for Trigger.* Also notice that the hw\_ila\_1 status does not change from idle as it is not armed.
    2. Switch to the terminal emulator window and type **U** (shift+u) to trigger the core.
    3. Select the corresponding waveform window and verify that it shows 55 after the trigger.

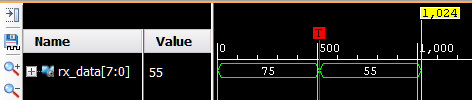


Figure 25. Second ila core triggered

* + 1. When satisfied, close the terminal emulator program and power OFF the board
    2. Select **File > Close Hardware Manager**. Click **OK** to close it.
    3. Close the **Vivado** program by selecting **File > Exit** and click **OK**.

Conclusion

You used ILA core from the IP Catalog and Mark Debug feature of Vivado to debug the hardware design.