

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

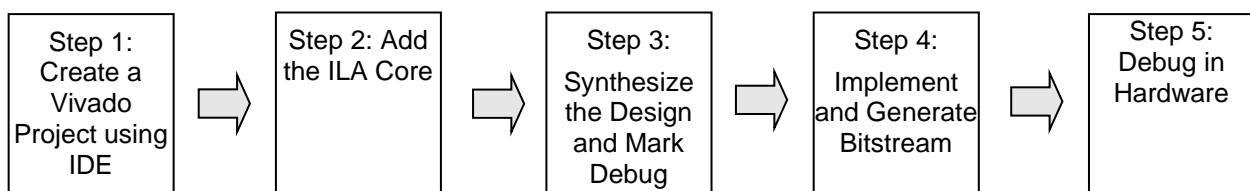
After completing this lab, you will be able to:

- 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 lab.

General Flow



Add the ILA Core

Step 2

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

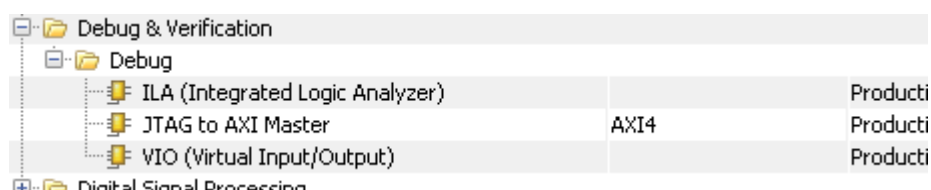


Figure 2. ILA in IP Catalog

We will be connecting the ILA core/component to the LED port.

- 1-1-4. Change the component name to **ila_led**.
- 1-1-5. Change the *Number of Probes* to **2**.

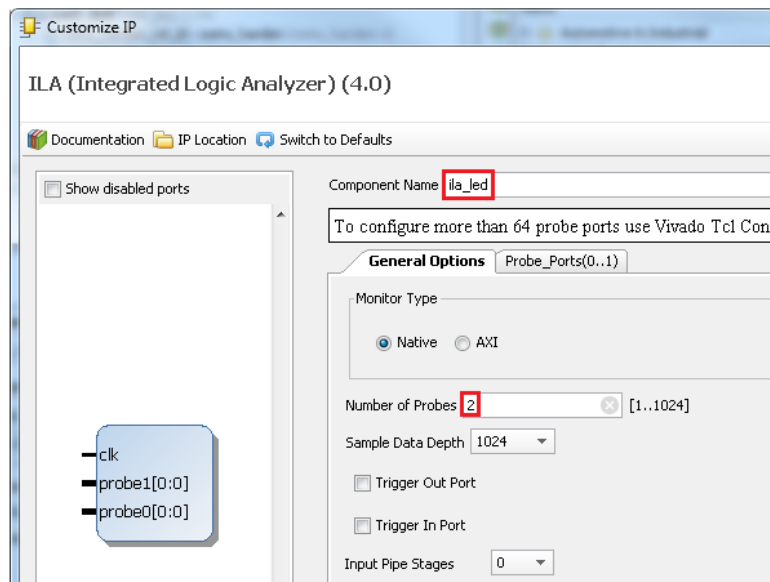


Figure 3. Setting the component name and the number of probes field

- 1-1-6.** Select the *Probe Ports* tab and change the PROBE1 port width to **8** for the ZedBoard and **4** for the Zybo, leaving the PROBE0 width to **1**. Leave the Number of Comparators as default for both probes.

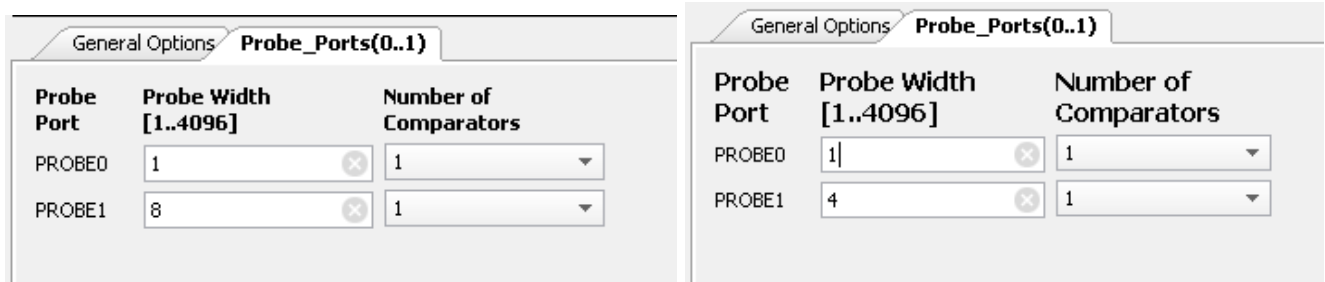


Figure 4. Setting the probes width for the ZedBoard

Figure 4. Setting the probes width for the Zybo

- 1-1-7.** Click **OK**.

The Generate Output Products dialog box will appear.

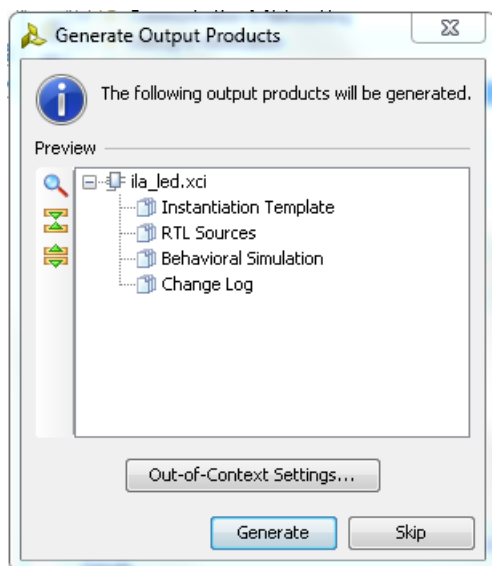


Figure 5. The Generate Output Products

- 1-1-8.** Click the **Generate** button to generate the core including the instantiation template. Notice the core is added to the *Design Sources* view.

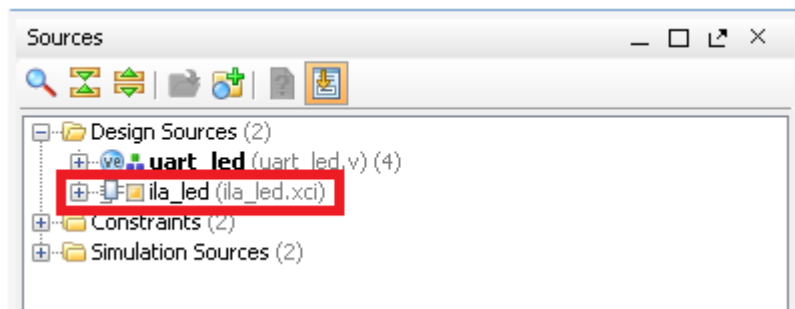


Figure 6. Newly generate ila core added in the design source

- 1-1-9.** Select the **IP Sources** tab, expand **IP > ila_led > Instantiation Template** and double-click the **ila_led.veo** entry to see the instantiation template.
- 1-1-10.** Instantiate the ila_led in design by copying lines 56 – 62 and pasting around line 104 (before “endmodule” on the last line) in the uart_led.v file.
- 1-1-11.** Change *your_instance_name* to **ila_led_i0**.
- 1-1-12.** 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)

```

105 ila_led ila_led_i0 (
106     .clk(clk_pin),           // input wire clk
107     .probel(led_pins),       // input wire [7 : 0] probe1
108     .probe0(rx_data_rdy)     // input wire [0 : 0] probe0
109 );
110
111 endmodule

```

Figure 7. Instantiating the ILA Core in the uart_led.v

- 1-1-13.** Select **File > Text Editor > Save File**.

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

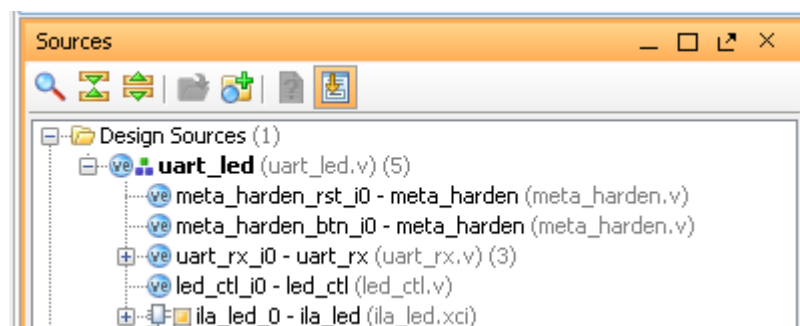


Figure 8. ILA Core added to the design

Synthesize the Design and Mark Debug

Step 3

2-1. 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.

2-1-1. 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.

2-1-2. Select the *Open Synthesized Design* option and click **OK** as we want to look at the synthesis output.

2-1-3. 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.

2-1-4. Select the **rx_data** bus between the *uart_rx_i0* and the *led_ctl_i0* instances, right-click, and select **Mark Debug**.

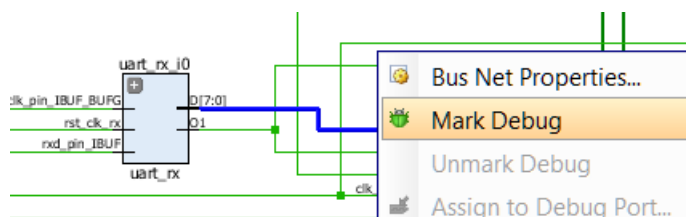


Figure 9. Marking a bus to debug

2-1-5. Confirm the debug nets selection. Click **OK**.

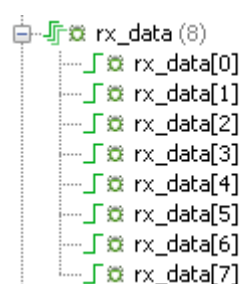
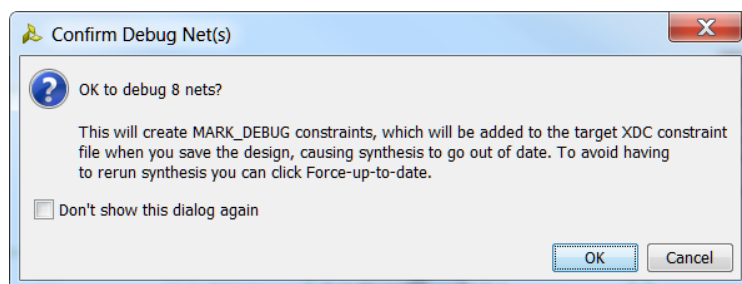


Figure 10. Confirm Debug Net(s) dialog box Figure 11. Nets with debug icons


2-1-6. Select the **Netlist** tab and notice that the nets which are marked/assigned for debugging have a debug icon next to them.

2-1-7. If not already in the layout, 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.

Name	Driver Cell
dbg_hub (labtools_xsdbmasterlib_v3)	
ila_led_i0 (labtools_ila_v4)	
clk (1)	
probe0 (1)	
probe1 (8)	
Unassigned Debug Nets (8)	
rx_data (8)	FDRE
rx_data[0]	FDRE
rx_data[1]	FDRE
rx_data[2]	FDRE
rx_data[3]	FDRE
rx_data[4]	FDRE
rx_data[5]	FDRE
rx_data[6]	FDRE
rx_data[7]	FDRE

Figure 12. Debug tab showing assigned and unassigned nets

- 2-1-8.** 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.
- 2-1-9.** In the Set Up Debug wizard click **Next** three times, then **Finish**.
- 2-1-10.** 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.
- 2-1-11.** Select **File > Constraints > Save**. There may be a warning about synthesis going out of date from a saved constraints file, acknowledge it by clicking **OK**.
- 2-1-12.** Open `uart_led_pins.xdc` and notice the debug nets have been appended to the bottom of the file.
- 2-1-13.** 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.

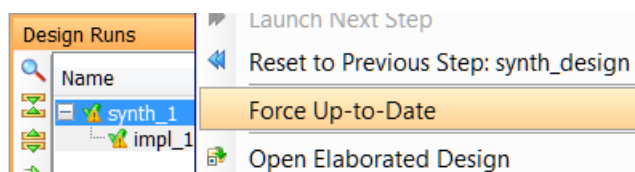


Figure 15. Forcing the design to be up-to-date

Implement and Generate Bitstream

Step 4

3-1. Generate the bitstream.

- 3-1-1.** Click on the **Generate Bitstream** to run the implementation and bit generation processes.
- 3-1-2.** Click **Yes** to run the processes.
- 3-1-3.** When the bitstream generation process has completed successfully, a box with four options will appear. Select the **Open Hardware Manager** option and click **OK**.

Debug in Hardware

Step 5

4-1. Plug-in the PmodUSB UART module into the top-row of the JA PMOD connector on the ZedBoard or the JE PMOD connector on the Zybo. Connect the module to the host machine using a micro-USB cable. Connect the board and power it ON. Open a Hardware Manager, and program the FPGA.

- 4-1-1.** Plug-in the PmodUSB UART module into the top-row of the JA PMOD connector of the ZedBoard. If using the Zybo, plug the PmodUSB UART module to the top-row of the JE PMOD connector.
- 4-1-2.** Connect the micro-USB cable between the PmodUSB UART module and the host USB port.
- 4-1-3.** Make sure that the Micro-USB cable is connected to the JTAG PROG connector (next to the power supply switch). Connect the power jack and turn ON the power.
- 4-1-4.** Click on the **Open New Hardware Target** link.
- You can also click on the Open Recent Hardware Target link if the board was already targeted before.
- 4-1-5.** Click **Next** to see the Hardware Server Settings form.
- 4-1-6.** Click **Next** with the JTAG Clock Frequency of **15000000** selected.

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

4-1-7. Click **Finish**.

4-1-8. The Hardware Manager status changes from Unconnected to the server name and the device is highlighted. Also notice that the Status indicates that it is not programmed.

4-1-9. Select the device and verify that the **uart_led.bit** is selected as the programming file in the General tab.

4-2. **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.**

4-2-1. Start a terminal emulator program such as TeraTerm or HyperTerminal.

4-2-2. Select an appropriate COM port (you can find the correct COM number using the Control Panel).

4-2-3. Set the COM port for 115200 baud rate communication.

4-2-4. Right click on the target FPGA and select **Program Device...** . Click **Program**.

The Hardware Manager will open showing the **Debug Probes** window (if not shown use *Window > Debug Probes*). Notice that there are two debug cores.

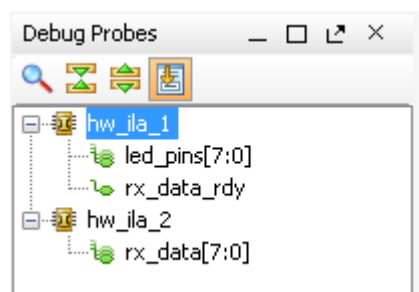


Figure 16. Debug probes

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

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210248470364 (2)	Open
arm_dap_0 (0)	Not programmed
xc7z020_1 (3)	Programmed
XADC (System Monitor)	
hw_ila_1 (ILA)	Idle
hw_ila_2 (ILA)	Idle

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/21027954024...	Open
arm_dap_0 (0)	Not programmed
xc7z010_1 (3)	Programmed
hw_ila_1 (ILA)	Idle
hw_ila_2 (ILA)	Idle
XADC (System Monitor)	

Figure 17. Hardware Manager status for the ZedBoard - Hardware Manager status for the Zybo

4-2-5. Select the XC7Z020 or XC7Z010, and click on the **Run Trigger Immediate** button to see the signals in the waveform window.

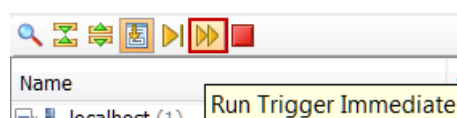



Figure 18. Opening the waveform window

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.

4-3. **Setup trigger conditions to trigger on a write to led port (rx_data_rdy=1) and the trigger position to 512. Arm the trigger.**

4-3-1. Click on the *hw_ila_1* tab on the top. Below the waveform window, select the *Trigger Setup – hw_ila_1* tab and click on the “+” sign in the centre of the window to add probes.

4-3-2. Select **rx_data_rdy** and click **OK**.

- 4-3-3.** Click on the **rx_data_rdy** compare value ($== [B] X$) and change the value from x to 1.
- 4-3-4.** Click the *Settings - hw_ila_1* tab and make sure the trigger position of the *hw_ila_1* is **512**.
- 4-3-5.** Similarly, click on the *hw_ila_2* tab on the top, add *rx_data* as a trigger probe and make sure the trigger position of the *hw_ila_2* is 512.
- 4-3-6.** Select the *hw_ila_1* tab and click on the Run Trigger button () button, Observe that the *hw_ila_1* core is armed and showing the status as **Waiting For Trigger**. Also note that the *hw_ila_2* status is Idle.

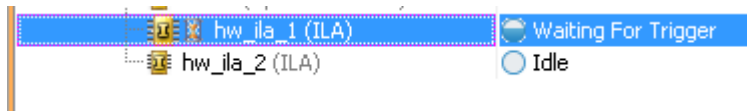


Figure 22. Hardware analyzer running and in capture mode

- 4-3-7.** In the terminal emulator window, type a character, and observe that the *hw_ila_1* status changes from Waiting For Trigger to Idle as the *rx_data_rdy* became 1.
- 4-3-8.** See the waveform for *hw_ila_1*. Notice that the *rx_data_rdy* goes from 0 to 1 at 512th sample and the *led_pis* [7:0] bits also changes from 0 to the bit pattern corresponding to the character you typed.

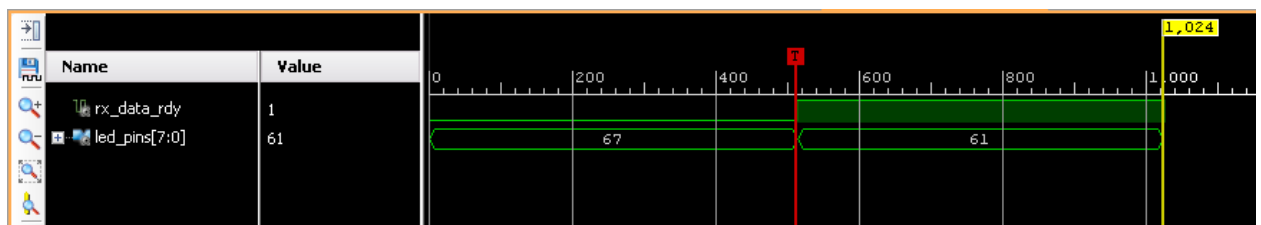


Figure 23. Zoomed waveform view for the ZedBoard

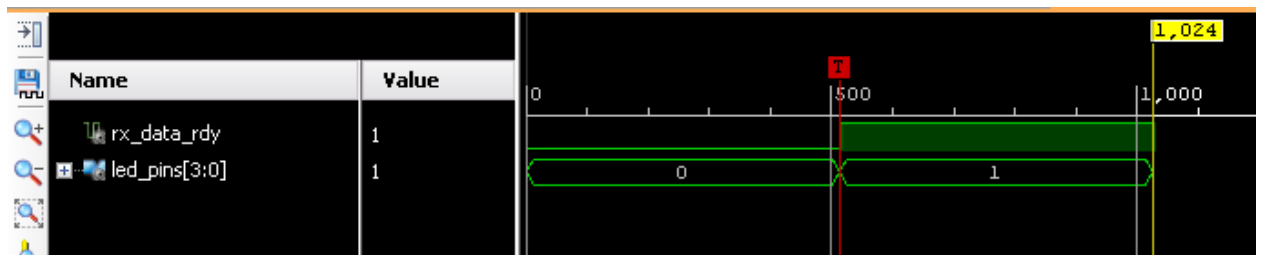


Figure 23. Zoomed waveform view for the Zybo

- 4-3-9.** In the trigger window of the *hw_ila_2*, change the trigger condition for *rx_data*[7:0]'s Radix from Hexadecimal to Binary. Change XXXX_XXXX to **0101_0101** (for the ASCII equivalent of U). Click **OK**.

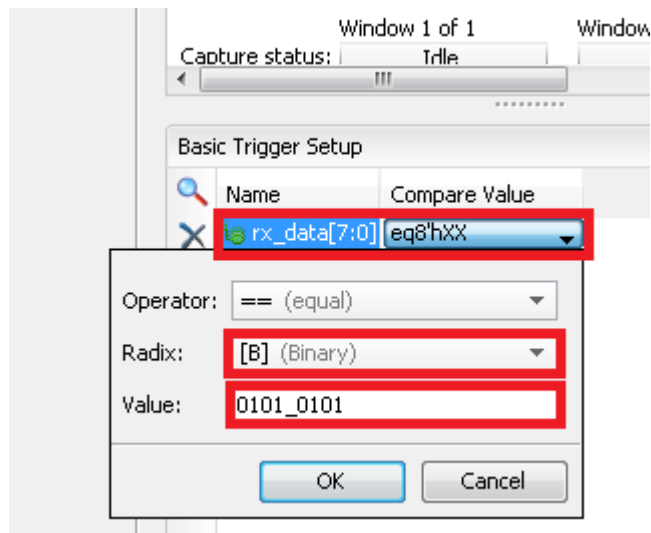


Figure 24. Setting up trigger condition for a particular input pattern

- 4-3-10. 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.
- 4-3-11. Switch to the terminal emulator window and type **U** (capital letter; on keyboard “shift+u”) to trigger the core.
- 4-3-12. Select the corresponding waveform window and verify that it shows 55.

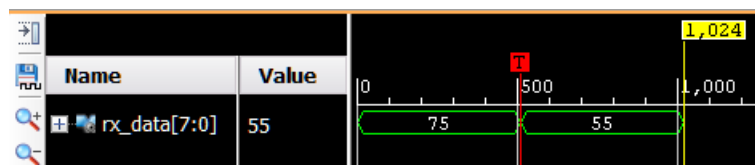


Figure 25. Second ila core triggered

- 4-3-13. When satisfied, close the terminal emulator program and power OFF the board
- 4-3-14. Select **File > Close Hardware Manager**. Click **OK** to close it.
- 4-3-15. 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.