

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.

Note: You will notice certain procedures have different variations depending on development board being ZedBoard or Zybo. It will be explicitly mentioned in notes when such variation is encountered

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**.

In this design we will use board's USB-UART which is controlled by the Zynq's ARM Cortex-A9 processor. Our PL design needs access to this USB-UART. So first thing we will do is to create a Processing System design which will put the USB-UART connections in a simple GPIO-style and make it available to the PL section. The complete system is shown in **Figure 2**.

The provided design places the UART (RX) pin of the PS (Processing System) on the Cortex-A9 in a simple GPIO mode to allow the UART to be connected (passed through) to the Programmable Logic. The processor samples the RX signal and sends it to the EMIO channel 0 which is connected to Rx input of the HDL module provided in the Static directory. This is done through a software application provided in the `lab6.sdk` folder hierarchy.

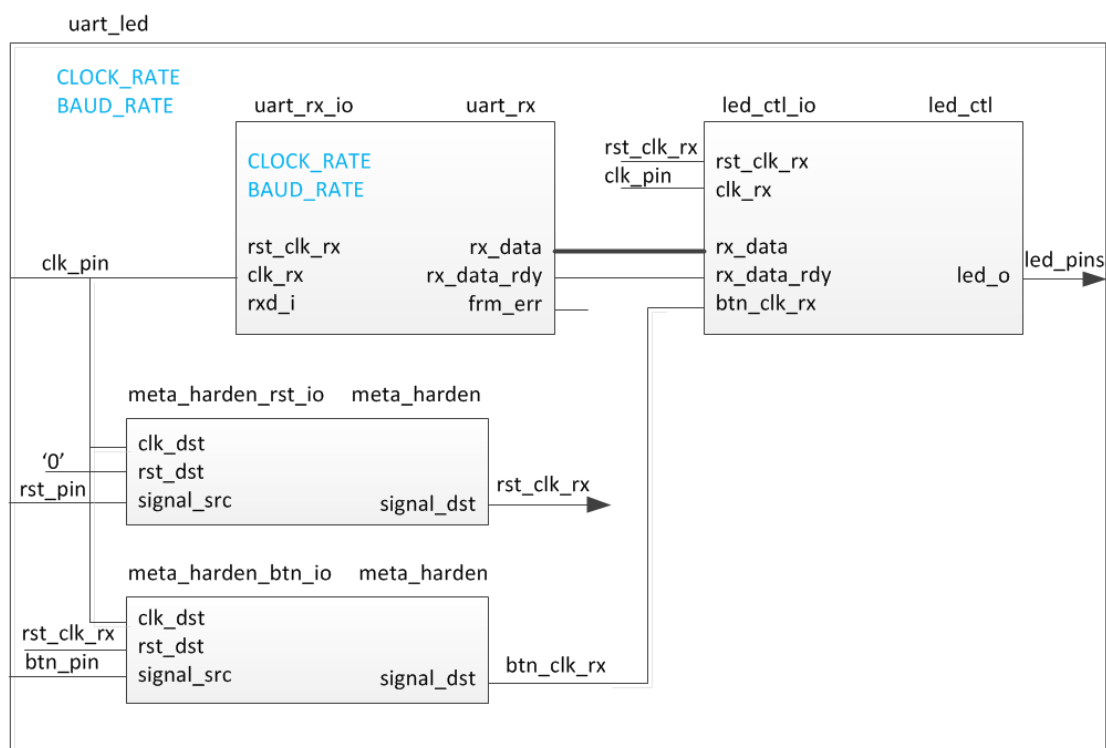


Figure 1. The Complete Design on PL

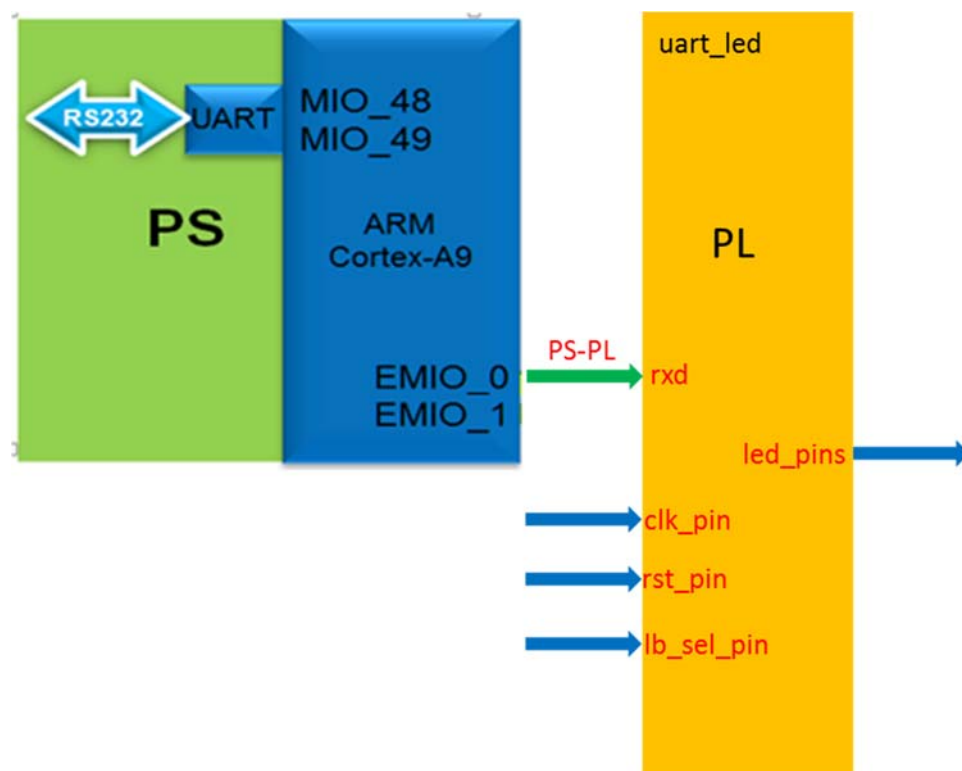
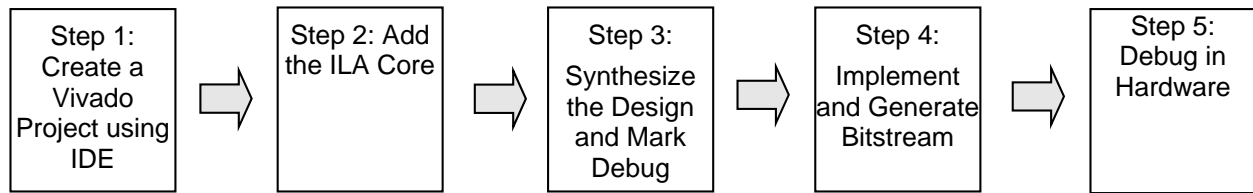


Figure 2. The Complete System

General Flow



Create a Vivado Project using IDE

Step 1

In this design we will use board's USB-UART which is controlled by the Zynq's ARM Cortex-A9 processor. Our PL design needs access to this USB-UART. So first thing we will do is to create a Processing System design which will put the USB-UART connections in a simple GPIO-style and make it available to the PL section.

- 1-1. Launch Vivado and create a project targeting the XC7Z020clg484-1 device (ZedBoard), or the XC7Z010clg400-1 (Zybo), and use provided the tcl scripts (ps7_create_<board>.tcl) to generate the block design for the PS subsystem. Also, add the Verilog HDL files, uart_led_pins_<board>.xdc and uart_led_timing.xdc files from the <2016_2_ZYNQ_sources>\lab6 directory.

References to <2016_2_ZYNQ_labs> is a placeholder for the c:\xup\fpga_flow\2016_2_ZYNQ_labs directory and <2016_2_ZYNQ_sources> is a placeholder for the c:\xup\fpga_flow\2016_2_ZYNQ_sources directory.

Reference to <board> means either the **ZedBoard** or the **Zybo**.

- 1-1-1. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2016.2 > Vivado 2016.2**
- 1-1-2. Click **Create New Project** to start the wizard. You will see *Create A New Vivado Project* dialog box. Click **Next**.
- 1-1-3. Click the Browse button of the *Project location* field of the **New Project** form, browse to <2016_2_ZYNQ_labs>, and click **Select**.
- 1-1-4. Enter **lab6** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.
- 1-1-5. Select **RTL Project** option in the *Project Type* form, and click **Next**.
- 1-1-6. Using the drop-down buttons, select **Verilog** as the *Target Language* and *Simulator Language* in the *Add Sources* form.
- 1-1-7. Click on the **Green Plus** button, then the **Add Files...** button and browse to the <2016_2_ZYNQ_sources>\lab6 directory, select all the Verilog files (*led_ctl.v*, *meta_harden.v*,

uart_baud_gen.v, uart_led.v, uart_rx.v, uart_rx_ctl.v and uart_top.v), click **OK**, and then click **Next**.

1-1-8. Click **Next** to get to the *Add Constraints* form.

1-1-9. Click on the **Green Plus** button, then **Add Files...** and browse to the `c:\xup\fpga_flow\2016_2_ZYNQ_sources\lab6` directory (if necessary), select *uart_led_timing_<board>..xdc* and the appropriate *uart_led_pins_<board>.xdc* and click **Open**.

1-1-10. Click **Next**.

1-1-11. In the *Default Part* form, Use the **Boards** option, you may select the **Zedboard** or the **Zybo** depending on your board from the Display Name drop down field.

You may also use the **Parts** option and various drop-down fields of the **Filter** section. If using the ZedBoard, select the **XC7Z020clg484-1** part. If using the Zybo, select the **XC7Z010clg400-1** part.

Note: Notice that Zedboard and Zybo may not be listed under **Boards** menu as they are not in the tools database. If not listed then you can download the board files for the desired boards either from Digilent Inc website or from the XUP website's workshop material pages.

1-1-12. Click **Next**.

1-1-13. Click **Finish** to create the Vivado project.

1-1-14. In the Tcl Shell window enter the following command to change to the lab directory and hit **Enter**.

```
cd c:/xup/fpga_flow/2016_2_ZYNQ_sources/lab6
```

1-1-15. Generate the PS design by executing the provided Tcl script.

```
source ps7_create_zed.tcl (for ZedBoard) or
```

```
source ps7_create_zybo.tcl (for Zybo)
```

This script will create a block design called *system*, instantiate ZYNQ PS with one GPIO channel 49 and one EMIO channel. It will then create a top-level wrapper file called *system_wrapper.v* which will instantiate the *system.bd* (the block design). You can check the contents of the tcl files to confirm the commands that are being run.

1-1-16. Double-click on the **uart_led** entry to view its content.

Notice in the Verilog code, the **BAUD_RATE** and **CLOCK_RATE** parameters are defined to be 115200 and 100 MHz respectively.

Note: Zybo has an on-board clock of 125 Mhz and hence the *uart_led.v* has to be modified on line 38. **CLOCK_RATE** parameter will be modified to match the on-board clock rate of 125 Mhz.

```

23
24 `timescale 1ns/1ps
25 module uart_led (
26     // Write side inputs
27     input          clk_pin,      // Clock input (from pin)
28     input          rst_pin,      // Active HIGH reset (from pin)
29     input          btn_pin,      // Button to swap high and low bits
30     input          rxd_pin,      // RS232 RXD pin - directly from pin
31     output [7:0] led_pins       // 8 LED outputs
32 );
33
34 //*****
35 // Parameter definitions
36 //*****
37 parameter BAUD_RATE      = 115_200;
38 parameter CLOCK_RATE     = 125_000_000;
39

```

Figure 3. CLOCK_RATE parameter of uart_led for Zybo board

Add the ILA Core

Step 2

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

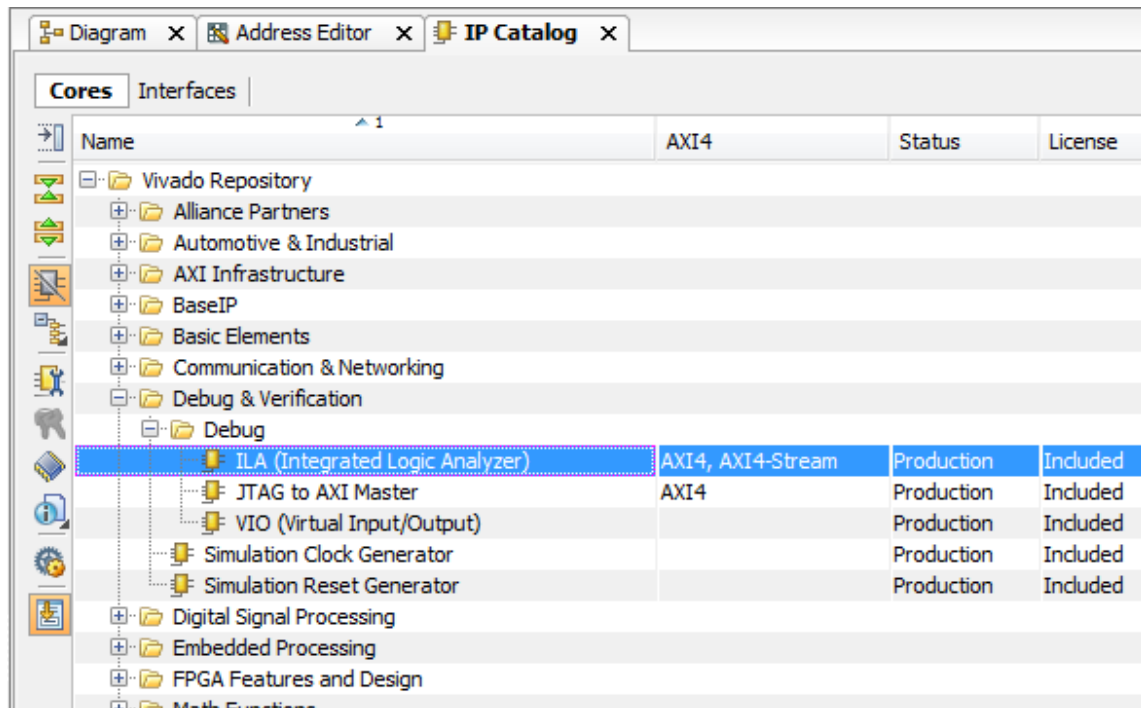


Figure 4. ILA in IP Catalog

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

2-1-4. Click **Customize IP** on the following Add IP window. The ILA IP will open.

2-1-5. Change the component name to **ila_led**.

2-1-6. Change the *Number of Probes* to **2**.

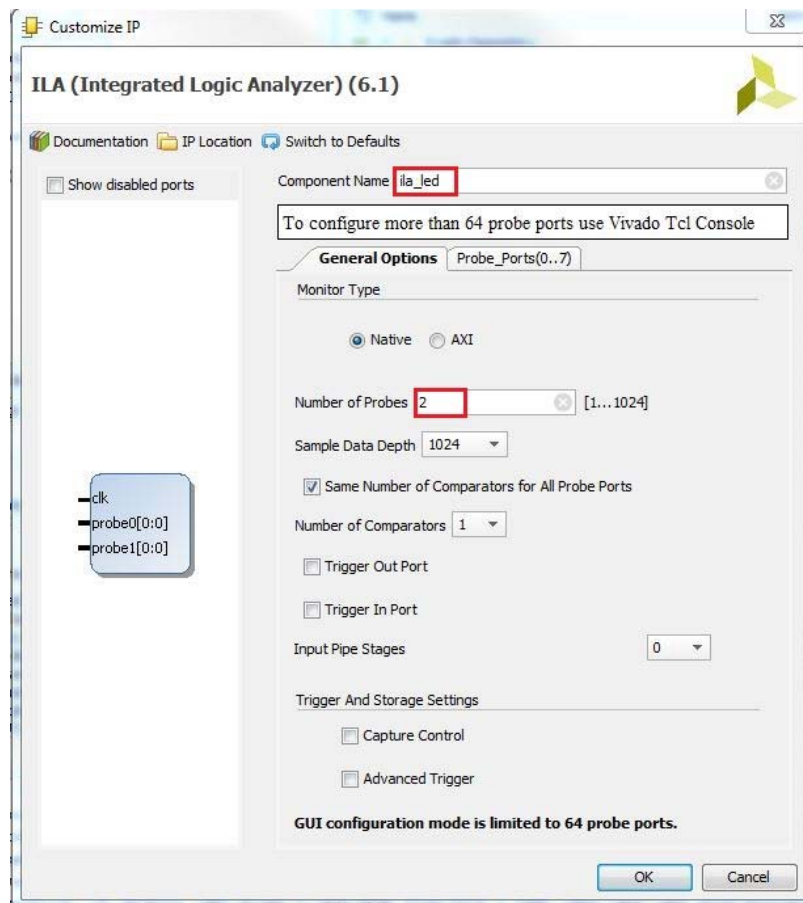


Figure 5. Setting the component name and the number of probes field

2-1-7. Select the *Probe Ports* tab and change the PROBE1 port width to **8**, leaving the PROBE0 width to **1**.

General Options Probe_Ports(0..7)			
Probe Port	Probe Width [1..4096]	Number of Comparators	Probe Trigger or Data
PROBE0	1	1	DATA AND TRIGGER
PROBE1	8	1	DATA AND TRIGGER

Figure 6. Setting the probes widths

2-1-8. Click **OK**.

The Generate Output Products dialog box will appear.

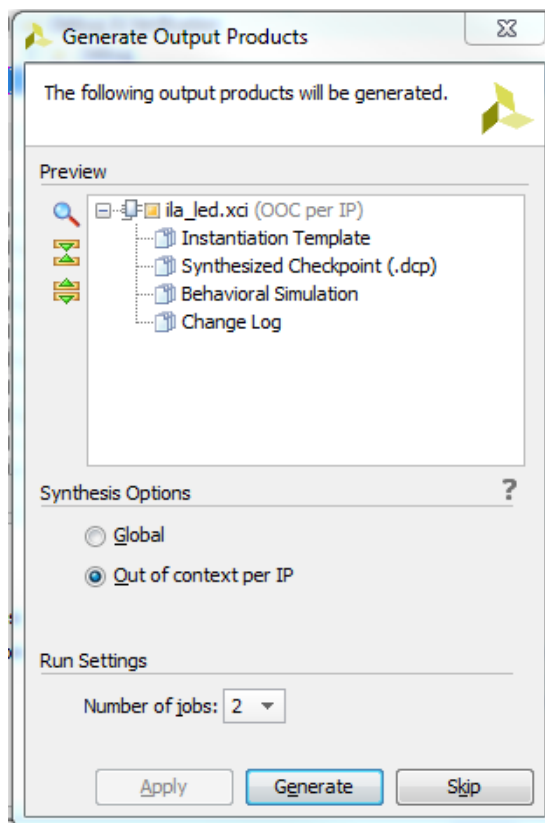


Figure 7. The Generate Output Products

- 2-1-9.** Click the **Generate** button to generate the core including the instantiation template. Click **OK** at the warning box. Notice the core is added to the *Design Sources* view.

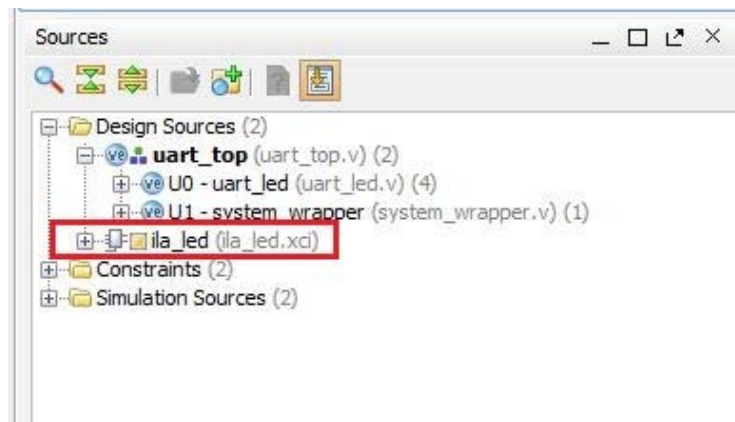


Figure 8. Newly generate ila core added in the design source

- 2-1-10.** 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.

- 2-1-11.** Instantiate the `ila_led` in design by copying lines 56 – 62 and pasting to ~line 69 (before “endmodule” on the last line) in the `uart_top.v` file.

- 2-1-12.** Change `your_instance_name` to `ila_led_i0`.

2-1-13. 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_out)
.probe1(PROBE1)	.probe1(led_pins)

```

70 ila_led ila_led_i0 (
71     .clk(clk_pin), // input wire clk
72     .probe0(rx_data_rdy_out), // input wire [0:0] probe0
73     .probe1(led_pins) // input wire [7:0] probe1
74 );
75
76 endmodule

```

Figure 9. Instantiating the ILA Core in the uart_top.v

2-1-14. Select **File > Save File**.

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

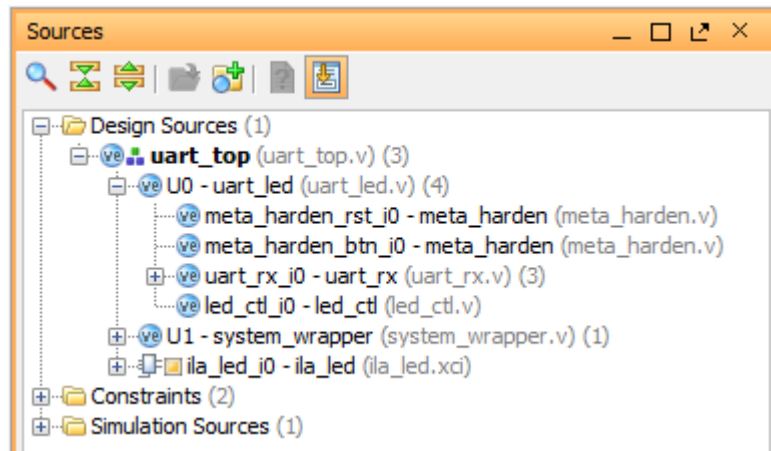


Figure 10. ILA Core added to the design

Synthesize the Design and Mark Debug

Step 3

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.

3-1-1. Click on **Run Synthesis** under the *Synthesis* tasks of the *Flow Navigator* pane. Click **Save** to Save Project if prompted.

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

3-1-2. Select the *Open Synthesized Design* option and click **OK**.

3-1-3. 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.

- 3-1-4.** Expand component **U0** and Select the **rx_data** bus between the *uart_rx_i0* and the *led_ctl_i0* instances, right-click, and select **Mark Debug**.

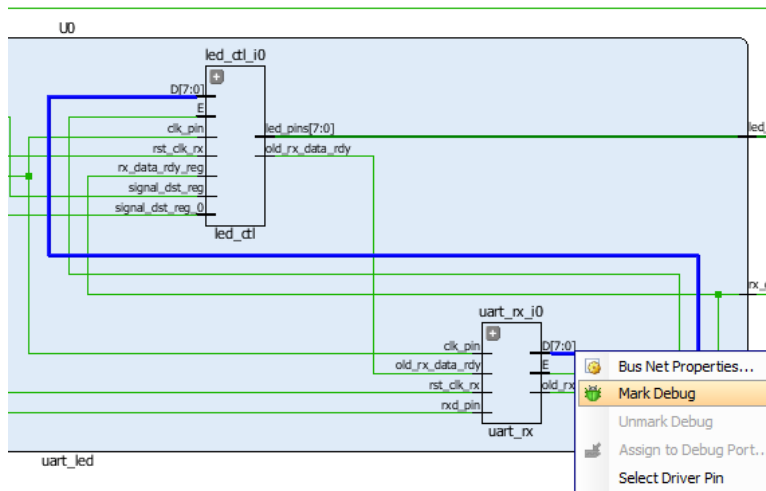


Figure 11. Marking a bus to debug

- 3-1-5.** Select **File > Save Constraints**.

- 3-1-6.** Click **OK**, and then again **OK** to use **uart_led_timing_<board>.xdc** as the target.

- 3-1-7.** Select the **Netlist** tab and notice that the nets which are marked/assigned for debugging have a debug icon next to them.

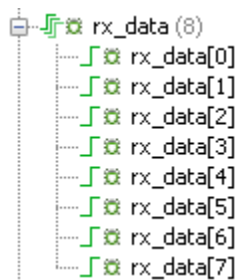


Figure 12. Nets with debug icons

- 3-1-8.** 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.

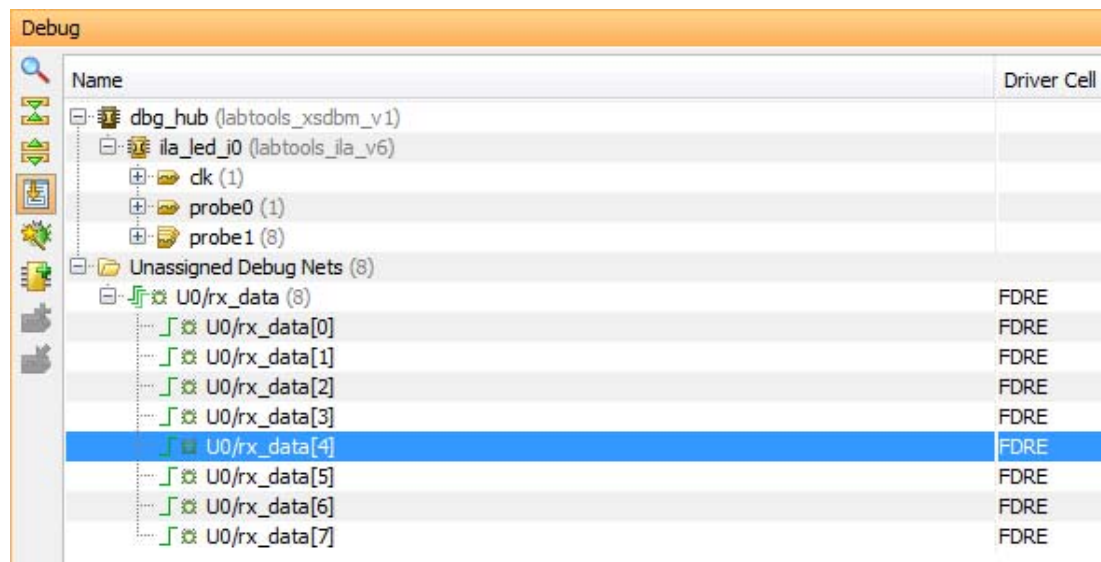


Figure 13. Debug tab showing assigned and unassigned nets

3-1-9. 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.

3-1-10. In the Set Up Debug wizard click **Next**.

Note that rx_data is listed, with the Clock Domain as clk_pin_IBUF_BUFG.

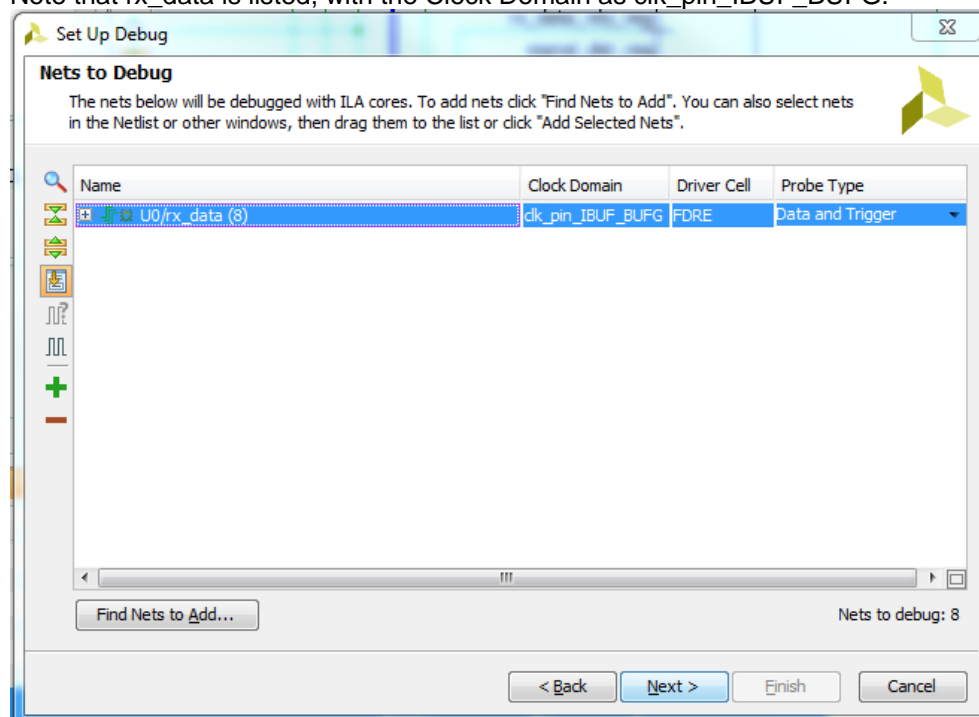


Figure 14. The remaining nets after removing already assigned nets in the Set Up Debug wizard

3-1-11. Click **Next** and again **Next** (leaving everything as defaults) then **Finish**.

3-1-12. In the Synthesized Design Schematic, click on the net on the output side of the BUFG for the input pin named `clk_pin`. Hover over the now highlighted net and notice the name is `clk_pin_IBUF_BUFG`. This is the clock net selected for the debug nets earlier.

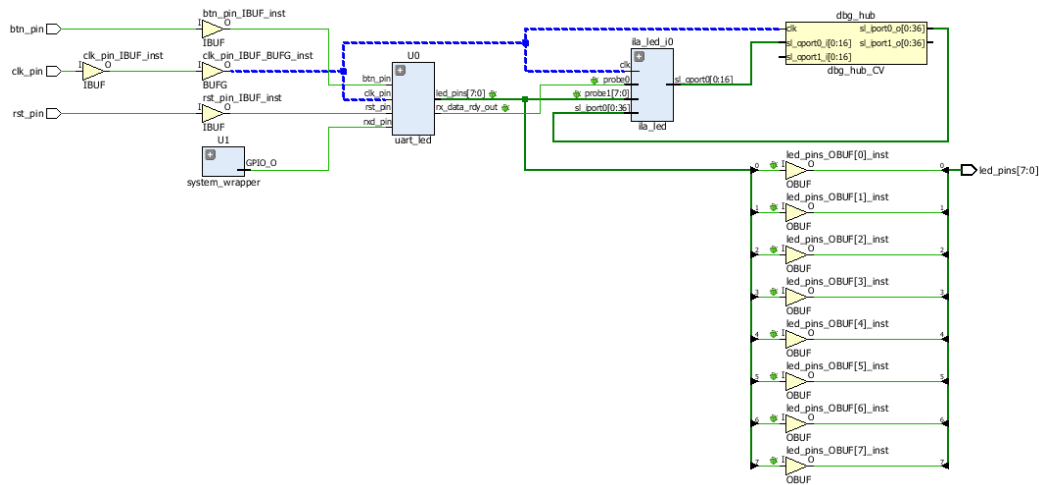


Figure 15. Locating `clk_pin_IBUF_BUFG` in the design

3-1-13. Right click on `uart_led_pins_<board>.xdc` in the sources pane and select **Set as Target Constraint File**. This will save the changes to the file

3-1-14. Select **File > Save Constraints** and click **OK** and Click **Yes**.

3-1-15. Open `uart_led_pins_<board>.xdc` and notice the debug nets have been appended to the bottom of the file.

3-1-16. 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.

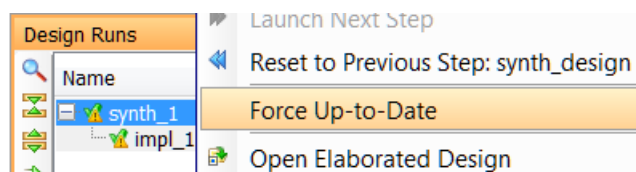


Figure 16. Forcing the design to be up-to-date

Implement and Generate Bitstream

Step 4

4-1. Generate the bitstream.

4-1-1. Click on the **Generate Bitstream** to run the implementation and bit generation processes.

4-1-2. Click **Yes** to run the implementation processes.

4-1-3. When the bitstream generation process has completed successfully, a box with three options will appear. Select the **Open Hardware Manager** option and click **OK**.

Debug in Hardware

Step 5

5-1. Connect the board and power it ON. Open a hardware session, and program the FPGA.

5-1-1. Make sure that the Micro-USB cable is connected to the JTAG PROG connector next to the power supply connector for the Zedboard. The Zybo JTAG PROG connector is located next to the power supply switch).

5-1-2. Turn ON the power.

5-1-3. Select the *Open Hardware Manager* option and click **OK**.

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

5-1-4. Click on the **Open target** link, then **Auto Connect** from the dropdown menu.

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

5-1-5. 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.

5-1-6. Select the device and verify that the **uart_top.bit** is selected as the programming file in the General tab. Also notice that there is an entry in the *Debug probes file* field.

5-2. 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.

5-2-1. Start a terminal emulator program such as TeraTerm or HyperTerminal.

5-2-2. Select an appropriate COM port (you can find the correct COM number using the Control Panel).

5-2-3. Set the COM port for 115200 baud rate communication.

5-2-4. 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. Debug Probes window will also be opened if not, then select **Window > Debug Probes**.

5-3. Start a SDK session, point it to the `c:/xup/fpga_flow/2016_2_ZYNQ_Sources/lab6/<board>/lab6.sdk` workspace.

5-3-1. Open **SDK** by selecting **Start > All Programs > Xilinx Design Tools > SDK 2016.2 > Xilinx SDK 2016.2**

5-3-2. In the **Select a workspace** window, click on the browse button, browse to `c:/xup/fpga_flow/2016_2_ZYNQ_Sources/lab6/` directory and select either

c:/xup/fpga_flow/2016_2_ZYNQ_Sources/lab6/Zybo/lab6.sdk or
 c:/xup/fpga_flow/2016_2_ZYNQ_Sources/lab6/ZedBoard/lab6.sdk and click **OK**.

5-3-3. Click **OK**.

In the *Project Explorer*, right-click on the `uart_led_zynq`, select *Run As*, and then **Launch on Hardware (System Debugger)**

In the Debug Probes window in Vivado notice that there are two debug cores, `hw_ila_1` and `hw_ila_2`.

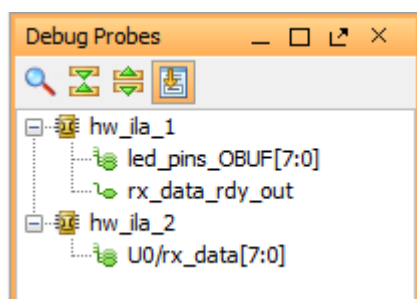


Figure 17. Debug probes

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

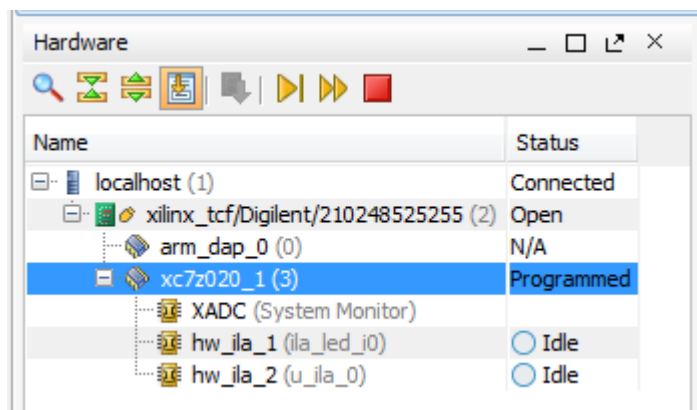


Figure 18. Hardware session status for the ZedBoard

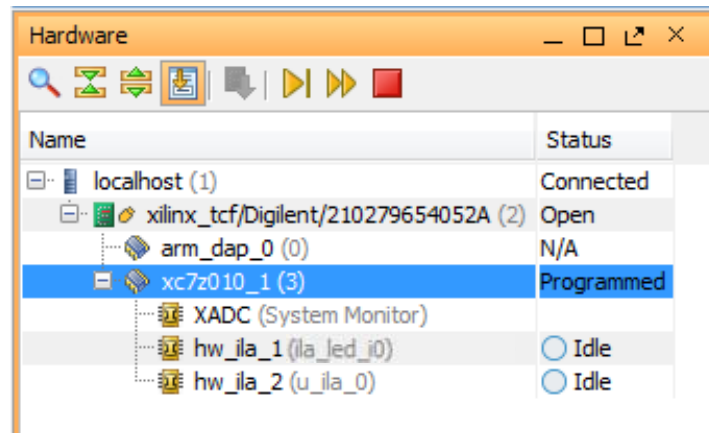


Figure 18. Hardware session status for the Zybo

- 5-3-4.** Select the target FPGA (XC7A100T, XC7A35T, or XC7A200T), and click on the **Run Trigger Immediate** button to see the signals in the waveform window.

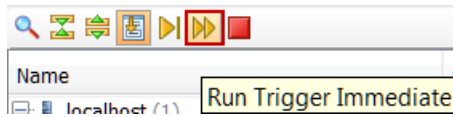


Figure 19. Opening the waveform window

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.

- 5-4. Setup trigger conditions to trigger on a write to led port (rx_data_rdy=1) and the trigger position to 512. Arm the trigger.**

- 5-4-1.** In the **Debug Probes** window, right-click on the **rx_data_rdy** and select **Add Probes to Basic Trigger Setup**.

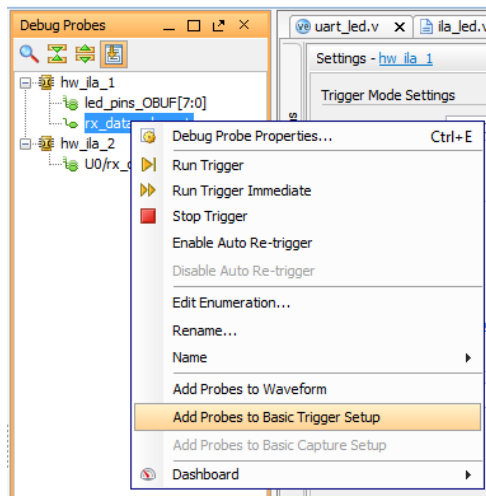


Figure 20. Adding a signal to trigger setup

- 5-4-2.** 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**.

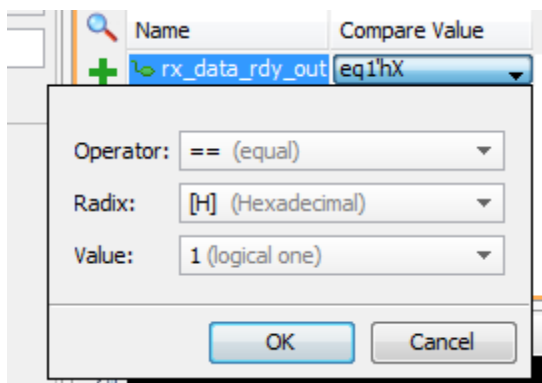


Figure 21. Setting the trigger condition

5-4-3. Set the trigger position of the *hw_ila_1* to **512**.

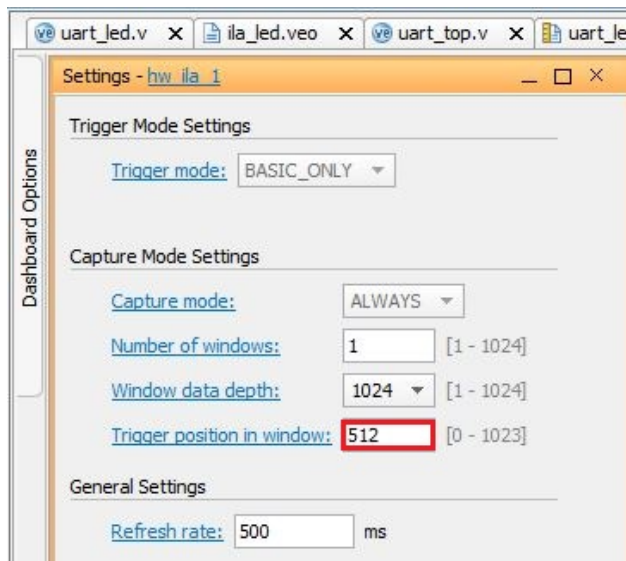



Figure 22. Setting up the trigger position

5-4-4. Similarly, set the trigger position of the *hw_ila_2* to **512**.

5-4-5. 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**.

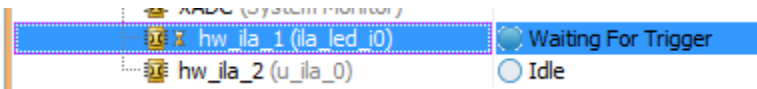


Figure 23. Hardware analyzer running in capture mode

5-4-6. 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.

5-4-7. 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.

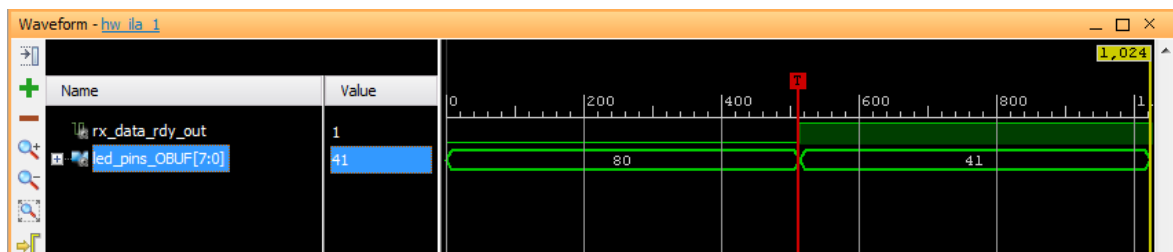


Figure 24. Zoomed waveform view

5-4-8. Add the *hw_ila_2* probes to the trigger window of the *hw_ila_2* and 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).

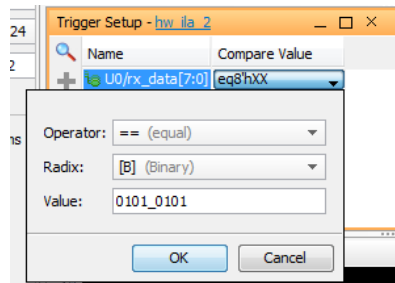


Figure 25. Setting up trigger condition for a particular input pattern

5-4-9. 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.

5-4-10. Switch to the terminal emulator window and type **U** (shift+u) to trigger the core.

5-4-11. Select the corresponding waveform window and verify that it shows 55 after the trigger.

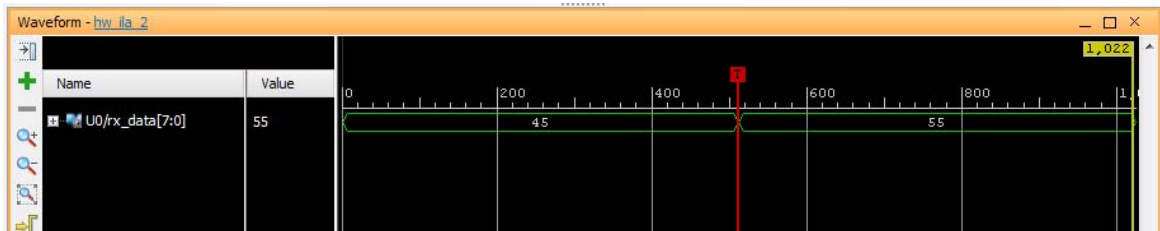


Figure 26. Second ila core triggered

5-4-12. When satisfied, close the terminal emulator program and power OFF the board

5-4-13. Select **File > Close Hardware Manager**. Click **OK** to close it.

5-4-14. Close the **Vivado** program by selecting **File > Exit** and click **OK**.

5-4-15. Close the **SDK** 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.