

Xilinx Design Constraints

Introduction

In this lab you will use the `uart_led` design that was introduced in the previous labs. You will start the project with I/O Planning type, enter pin locations, and export it to the rtl. You will then create the timing constraints and perform the timing analysis.

Objectives

After completing this lab, you will be able to:

- Create a I/O Planning project
- Enter the pin locations and IO standards via Device view, Package Pins tab, and Tcl commands
- Create Period, Input Setup, and Output Setup delays
- Perform timing analysis

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.

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 `lab5.sdk` folder hierarchy.

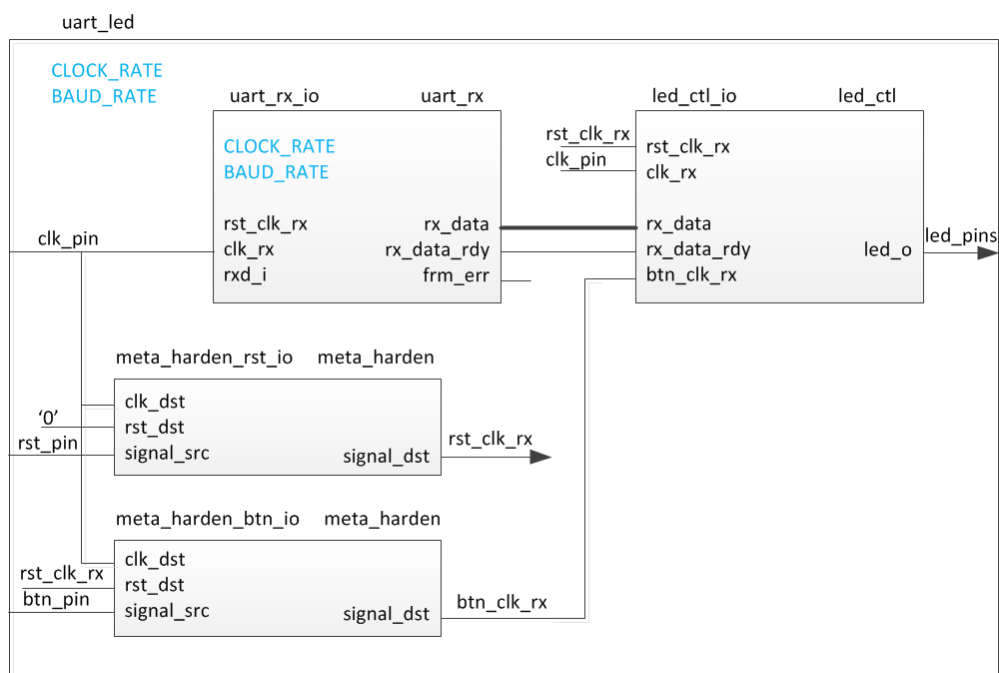


Figure 1. The Complete Design on PL

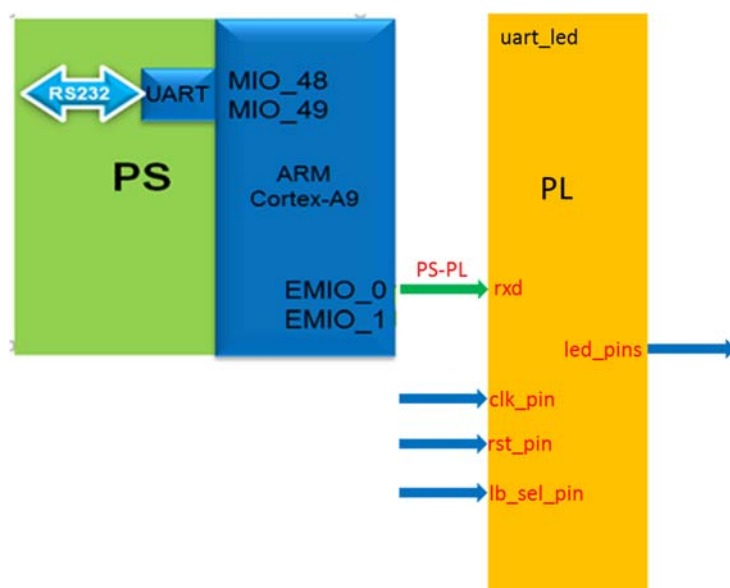
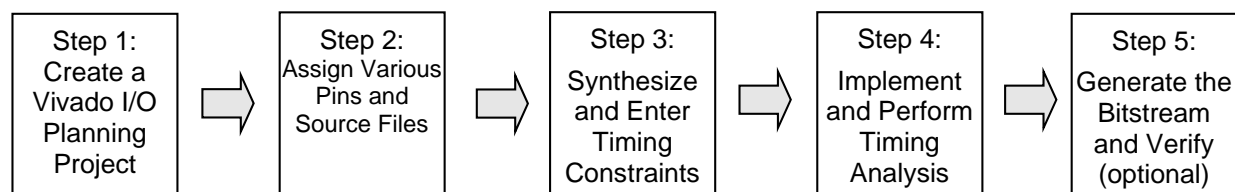


Figure 2. The Complete System

General Flow



Create a Vivado I/O Planning Project

Step 1

- 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>\lab5 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 **lab5** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.
- 1-1-5. Select **I/O Planning Project** option in the *Project Type* form, and click **Next**.
- 1-1-6. Select **Do not import I/O ports at this time**, and click **Next**.
- 1-1-7. 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.

- 1-1-8. Click **Next**.
- 1-1-9. Click **Finish** to create the Vivado project.

The device view window and package pins tab will be displayed.

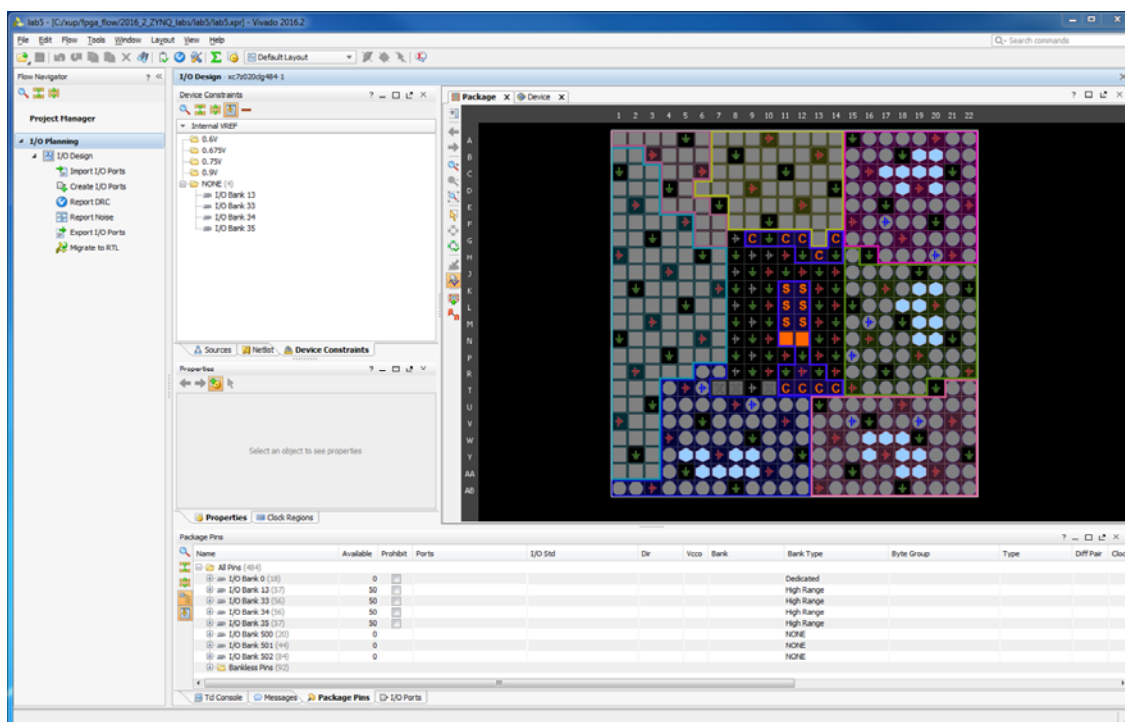


Figure 2. I/O Planning project's default windows and views for the ZedBoard

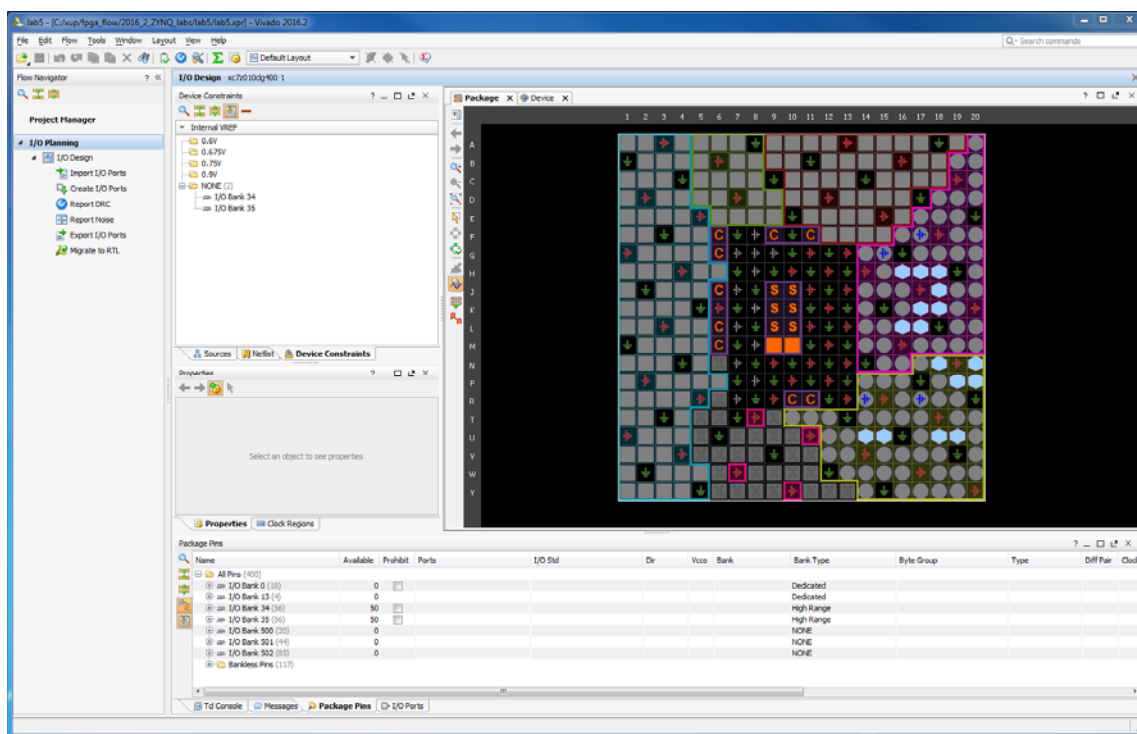


Figure 2. I/O Planning project's default windows and views for the Zybo

Create I/O Ports, Assign Various Pins and Add Source Files Step 2

2-1. Create input ports `clk_pin`, `btn_pin` and `rst_pin`.

- 2-1-1. Expand the *I/O Design* entry under the *I/O Planning* task of the *Flow Manager* and click on **Create I/O Ports**.

The Create I/O Ports form will be displayed.

- 2-1-2. Type `clk_pin` in the *Name* field, select **Input** for the *Direction* and select **LVC MOS33** as the *I/O Standard*, and click **OK**.

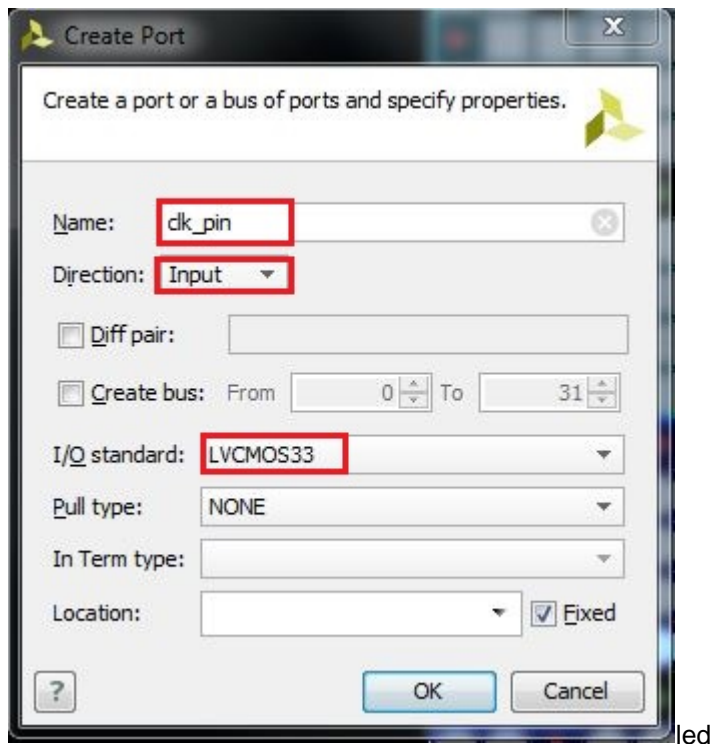


Figure 3. Creating I/O Port for `clk_pin` input

- 2-1-3. Similarly, create the `btn_pin` and `rst_pin` input ports.

2-2. For the ZedBoard, assign input pins `clk_pin`, `btn_pin` and `rst_pin` to Y9, T18, and P16 locations using the Device view and package pins.

For the Zybo, assign input pins `clk_pin`, `btn_pin` and `rst_pin` to L16, R18 and Y16 locations using the Device view and package pins.

For the ZedBoard, hover the mouse over **Y9** in the Device view window.

For the Zybo, hover the mouse over **L16** in the Device view window.



Figure 4. Locating Y9 pin in the Device view for the ZedBoard

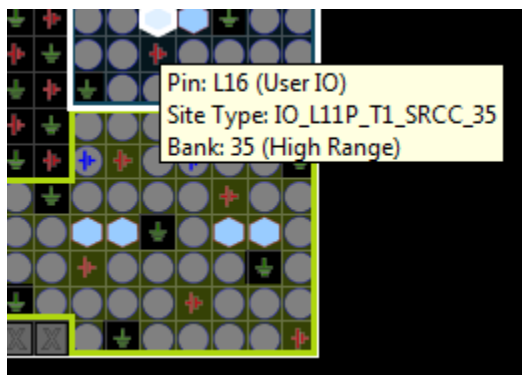


Figure 4. Locating L16 pin in the Device view for the Zybo

2-2-1. When located, click on it.

The pin entry will be highlighted and displayed in the Package Pins tab.

2-2-2. In the *Package Pins* pane, click in the Ports column of **Y9** (ZedBoard), or **L16** (Zybo) pin's row, and select **clk_pin**.

2-2-3. Similarly, add the **btn_pin** input port at **T18** (ZedBoard) or **R18** (Zybo).

2-2-4. Select **Edit > Find** or Ctrl-F to open the Find form. Select **Package Pins** in the *Find* drop-down field, type ***P16** (for the ZedBoard) or ***Y16** (for the Zybo) in the match criteria field, and click on **OK**.

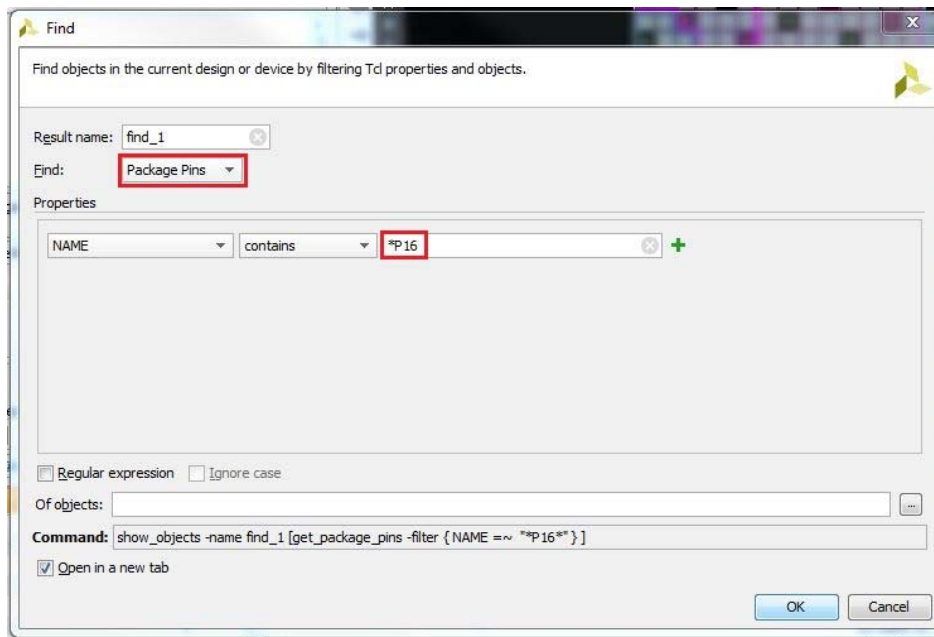


Figure 5. Finding a package pin for the ZedBoard (use Y16 for the Zybo)

Notice that the Find Results tab is opened, and the corresponding entry is shown in the tab.

2-2-5. Assign the `rst_pin` input to the pin.

2-3. For the ZedBoard, assign output pins `led_pins[0]` to `led_pins[7]` to locations U14, U19, W22, V22, U21, U22, T21, T22. Create them as a vector and assign them using the Tcl command `set_property`. They all will be LVCMOS33.

For the Zybo, assign output pins `led_pins[0]` to `led_pins[7]` to locations M14, M15, G14, D18, T20, U20, V20 and W20. Create them as a vector and assign them using the Tcl command `set_property`. All the pins will be LVCMOS33.

Note: Notice that Zybo has four LEDs hence we assign `led_pins[3:0]` to LEDs and `led_pins[7:4]` are assigned to PMOD JB.

2-3-1. In the I/O Ports tab, click on the create I/O port button on the left vertical ribbon.

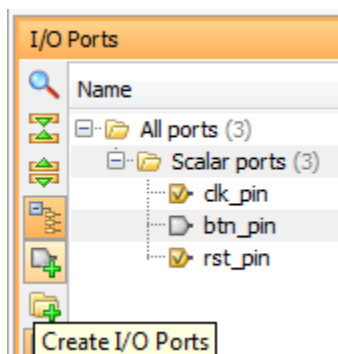


Figure 6. Create I/O Ports button

The Create I/O Ports form will be displayed.

- 2-3-2.** Type **led_pins** in the *Name* field, select *Output* direction, click on the check-box of **Create bus**, set the msb to **7**, and select **LVC MOS33** I/O standard and click **OK**.

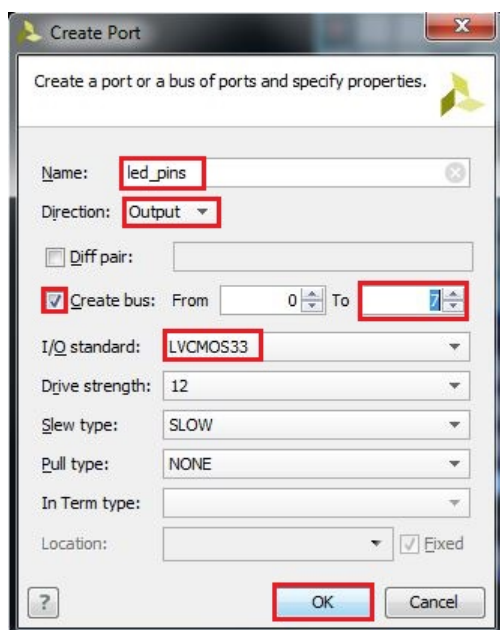


Figure 7. Creating I/O ports for the led_pins output

The led_pins entries will be created and displayed in the I/O Ports tab. Notice that the I/O standard and directions are already set, leaving only the pin locations to be assigned.

- 2-3-3.** Type the following commands in the console to assign the pin locations.

For the ZedBoard:

```
set_property -dict { PACKAGE_PIN U14 IOSTANDARD LVCMOS33 } [get_ports { led_pins[0] }];
set_property -dict { PACKAGE_PIN U19 IOSTANDARD LVCMOS33 } [get_ports { led_pins[1] }];
set_property -dict { PACKAGE_PIN W22 IOSTANDARD LVCMOS33 } [get_ports { led_pins[2] }];
set_property -dict { PACKAGE_PIN V22 IOSTANDARD LVCMOS33 } [get_ports { led_pins[3] }];
set_property -dict { PACKAGE_PIN U21 IOSTANDARD LVCMOS33 } [get_ports { led_pins[4] }];
set_property -dict { PACKAGE_PIN U22 IOSTANDARD LVCMOS33 } [get_ports { led_pins[5] }];
set_property -dict { PACKAGE_PIN T21 IOSTANDARD LVCMOS33 } [get_ports { led_pins[6] }];
set_property -dict { PACKAGE_PIN T22 IOSTANDARD LVCMOS33 } [get_ports { led_pins[7] }];
```

For the Zybo:

```
set_property -dict { PACKAGE_PIN M14 IOSTANDARD LVCMOS33 } [get_ports { led_pins[0] }];
set_property -dict { PACKAGE_PIN M15 IOSTANDARD LVCMOS33 } [get_ports { led_pins[1] }];
set_property -dict { PACKAGE_PIN G14 IOSTANDARD LVCMOS33 } [get_ports { led_pins[2] }];
set_property -dict { PACKAGE_PIN D18 IOSTANDARD LVCMOS33 } [get_ports { led_pins[3] }];
set_property -dict { PACKAGE_PIN T20 IOSTANDARD LVCMOS33 } [get_ports { led_pins[4] }];
set_property -dict { PACKAGE_PIN U20 IOSTANDARD LVCMOS33 } [get_ports { led_pins[5] }];
set_property -dict { PACKAGE_PIN V20 IOSTANDARD LVCMOS33 } [get_ports { led_pins[6] }];
set_property -dict { PACKAGE_PIN W20 IOSTANDARD LVCMOS33 } [get_ports { led_pins[7] }];
```

Select **File > Save Constraints**.

The Save Constraints form will be displayed.

- 2-3-4.** Enter **uart_led_<Board>** in the *File name* field, and click **OK**.

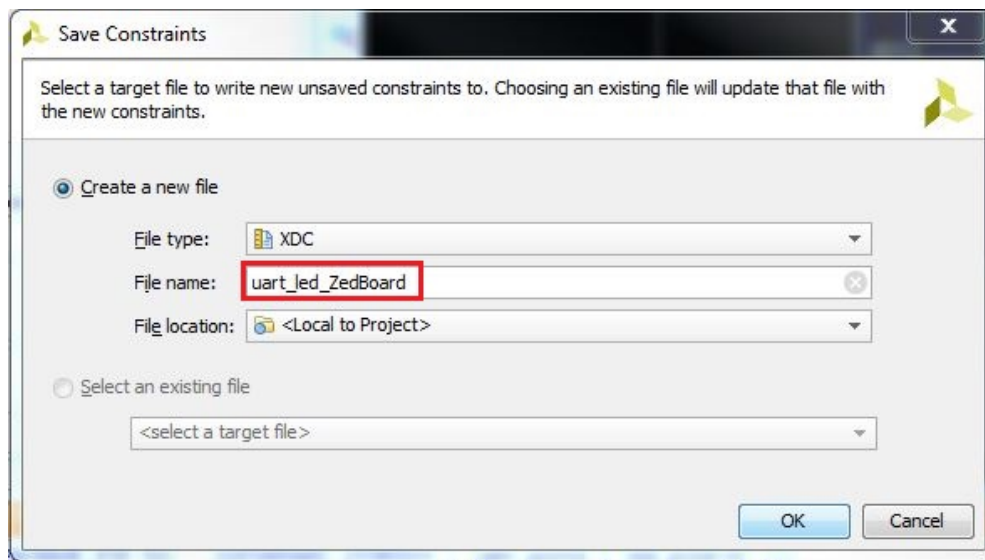


Figure 8. Saving constraints

The `uart_led_<Board>.xdc` file will be created and added to the Sources tab.

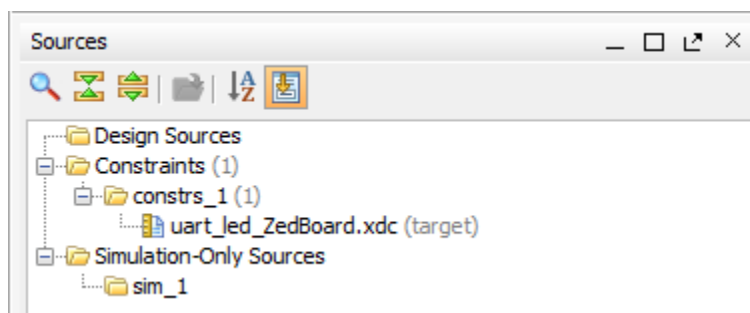


Figure 9. The `uart_led_<Board>.xdc` file added to the source tree

- 2-3-5.** Expand the **I/O Planning > I/O Design** in the *Flow Navigator* pane.
- 2-3-6.** Click on **Report DRC** and click **OK**. Notice the design rules checker is run warnings is reported. Ignore the warnings.
- 2-3-7.** Click on **Report Noise** and click **OK**. Notice the noise analysis is done on the output pins only (led_pins) and the results are displayed.
- 2-3-8.** Click on **Migrate to RTL**.

The Migrate to RTL form will be displayed with Top RTL file field showing `c:/xup/fpga_flow/2016_2_ZYNQ_labs/lab5/io_1.v` entry.
- 2-3-9.** Change `io_1.v` to **uart_top.v**, and click **OK**

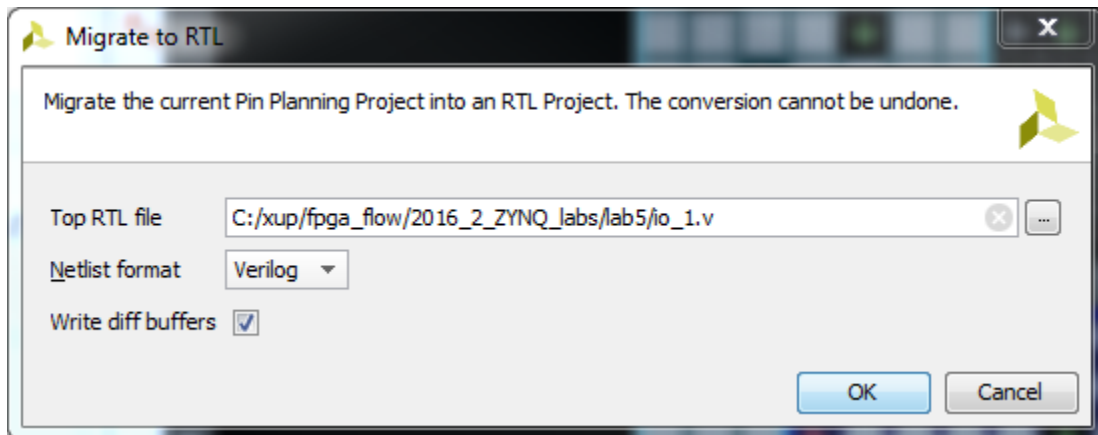


Figure 10. Assigning top-level file name

2-3-10. Select the **Hierarchy** tab and notice that the *uart_top.v* file has been added to the project with top-level module name as **ios**. If you double-click the entry, you will see the module name with the ports listing.

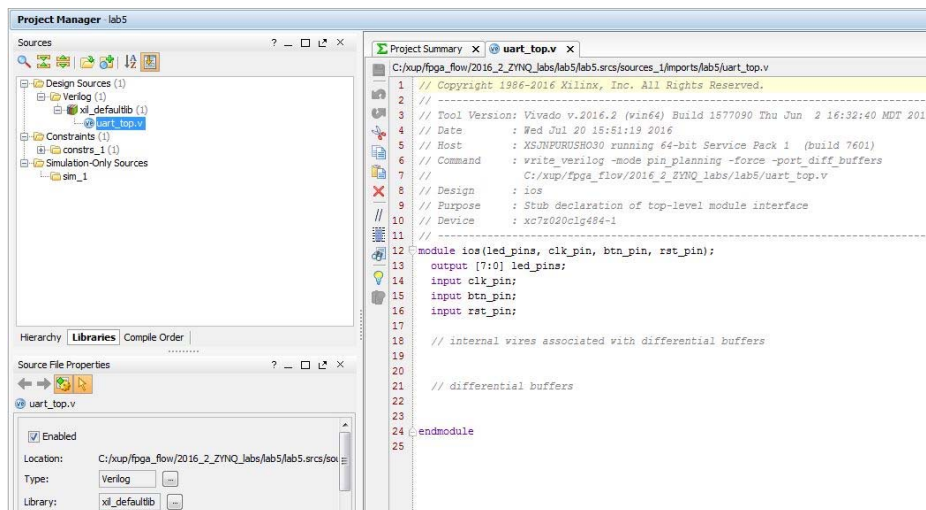


Figure 11. The top-level module content and the design hierarchy after migrating to RTL

2-4. Add the provided source files (from <2016_2_ZYNQ_sources>\lab5) to the project. Copy the *uart_top.txt* (located in the <2016_2_ZYNQ_sources>\lab5) content into the top-level source file.

2-4-1. Click on **Add Sources** in the *Flow Navigator*.

2-4-2. In the *Add Sources* form, select *Add or Create Design Sources*, and click **Next**.

2-4-3. Click on the **Green Plus** button, then the **Add Files...**

2-4-4. Browse to <2016_2_ZYNQ_sources>\lab5 and select all *.v* (*led_ctl.v*, *meta_harden.v*, *uart_baud_gen.v*, *uart_led.v*, *uart_rx.v*, *uart_rx_ctl.v*) files and click **OK**.

2-4-5. Click **Finish**.

2-4-6. Using Windows Explorer, browse to **<2016_2_ZYNQ_sources>\lab5** and open `uart_top.txt` using any text editor. Copy the content of it and paste it in `uart_top.v` (around line 22) in the Vivado project.

2-4-7. 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/lab5
```

2-4-8. 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.

2-4-9. 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 12. `CLOCK_RATE` parameter of `uart_led` for Zybo board

Synthesize and Enter Timing Constraints

Step 3


3-1. Synthesize the design. Use the Constraints Wizard to specify a clock frequency, and input and output delay constraints.

3-1-1. Click on the **Run Synthesis** in the *Flow Navigator* pane.

Click on the **Save** if Save project window appears.

When synthesis is completed a form with three options will be displayed.

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

3-1-3. In the *Flow Navigator* pane (under Synthesized Design), click on the **Constraints Wizard**  button. This will open up the Constraints Wizard.

3-1-4. Read the *Identify and Recommend Missing Timing Constraints* screen of the wizard to understand what the wizard does and click **Next**.

3-1-5. Specify the frequency of the object “clk_pin” to be **100 MHz**, notice the Period, Rise At and Fall At are automatically populated. Also notice the Tcl command that can be previewed at the bottom of the wizard. Click **Next** to proceed.

Note: Notice that the Zybo has an on-board clock of 125 Mhz and hence specify the frequency of the object “clk_pin” to be 125 MHz.

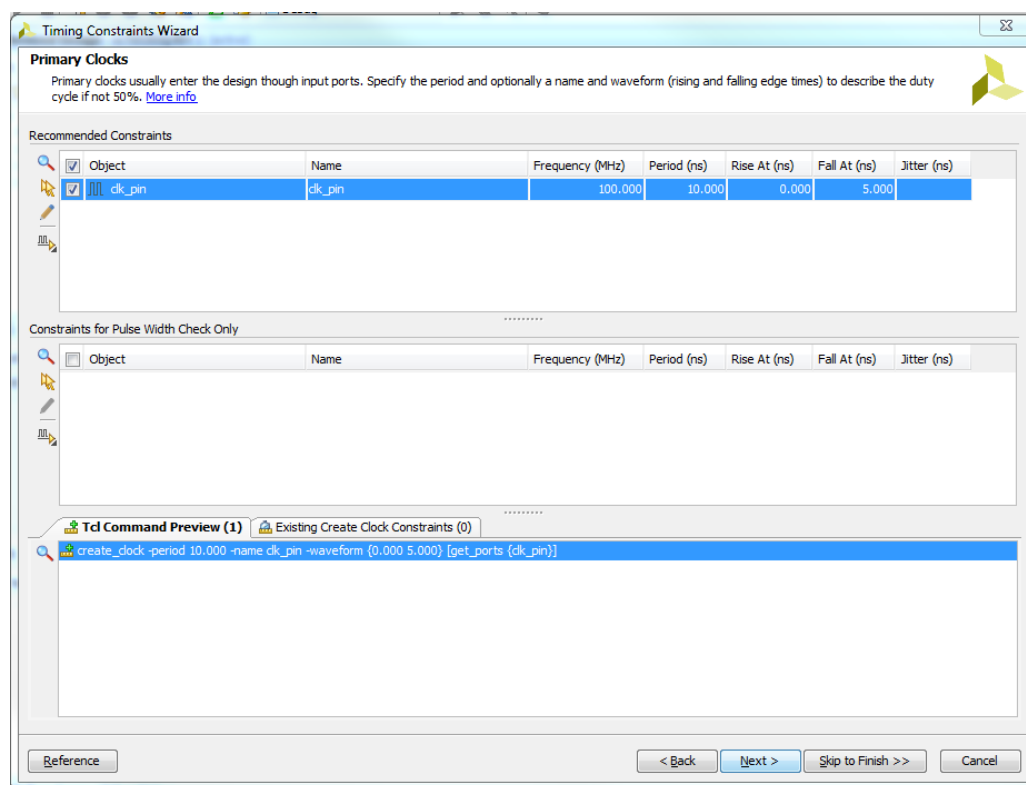


Figure 13. Constraints Wizard *clk_pin* parameters and Tcl command

3-1-6. There are no missing Generated Clocks, click **Next** to proceed.

3-1-7. There are no missing Forwarded Clocks, click **Next** to proceed.

3-1-8. There are no missing External Feedback Delays, click **Next** to proceed.

3-1-9. The wizard identifies Input Delays needed for the pins: btn_pin and rst_pin. Do the following:

- (1) Press Ctrl and select the two rows.
- (2) Enter the **tco_min** value to be **-0.5 ns** and everything else as **0 ns**. Click **Apply**.
- (3) Notice that under the Tcl Command Preview tab, 4 Tcl commands have been generated.
- (4) Click **Next**.

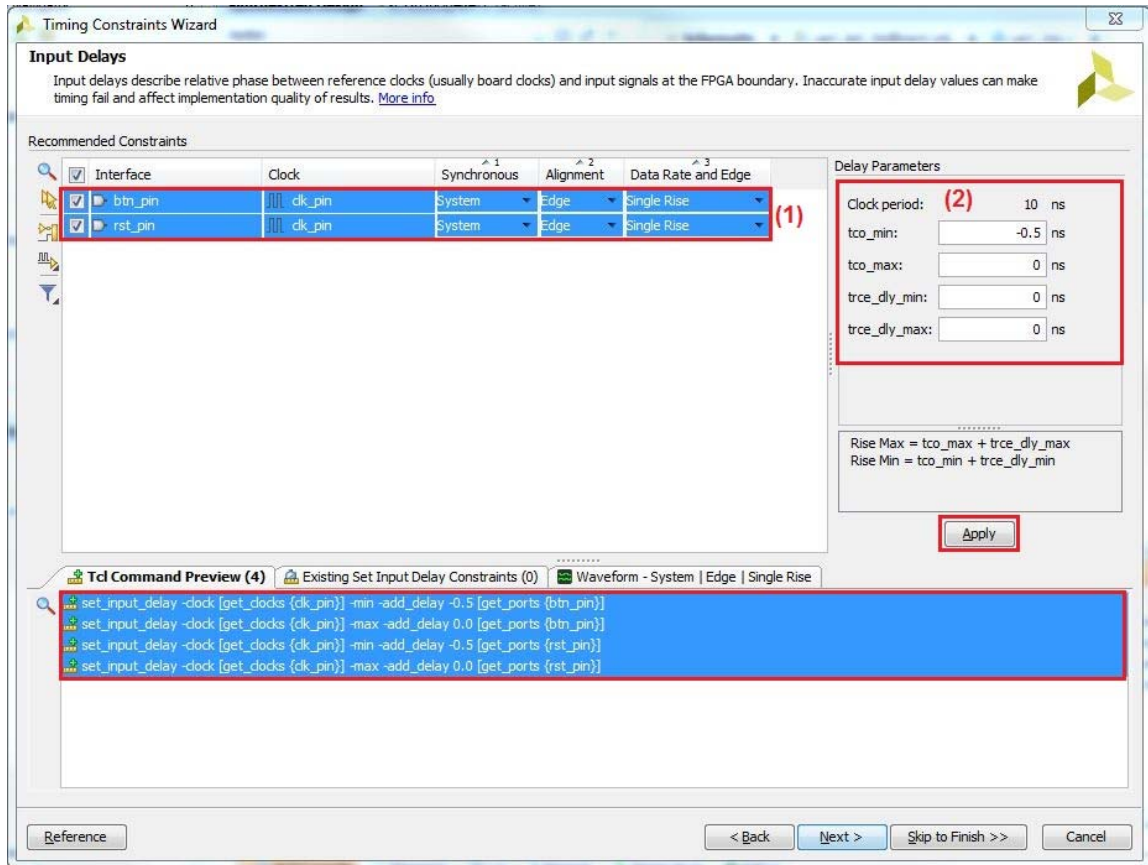


Figure 14. Specifying Input Delays for btn_pin and rst_pin

3-1-10. For ZedBoard Output Delays, specify all values (tsu, trce_dly_max, thd, trce_dly_min) to be **0 ns**. Click **Apply** and then click **Next**.
For Zybo, Enter the tsu and thd as **0 ns** and Enter the trce_dly_max and trce_dly_min as **-2 ns**. Click **Apply** and then click **Next**.

Note: Zybo has an on-board clock of 125 Mhz and requires output delay of -2ns for timing closure.

3-1-11. There are no Combinatorial Delays identified, click **Next** to proceed.

3-1-12. Click **Skip to Finish** to skip to the final Constraints Summary page. Read the description of each page.

3-1-13. Check On Finish – View Timing Constraints and click **Finish** to close the wizard. The option will open the Timing Constraints Editor to show you the generated timing constraint.

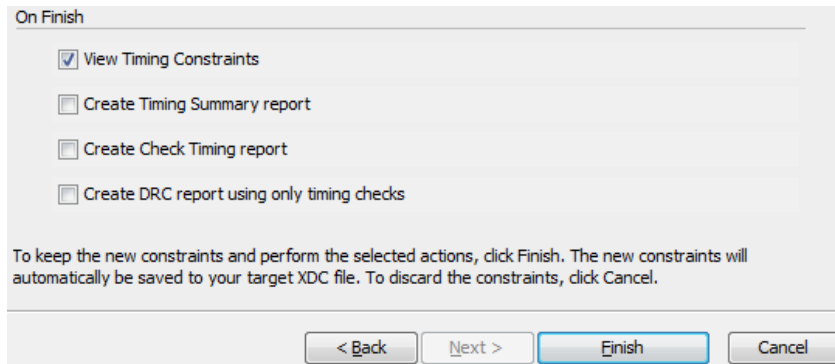


Figure 15. Selecting View Timing constraints

3-1-14. Note the wizard generated the clk_pin constraint for a 10 ns period (or 100 MHz). Notice in the All Constraints window, 7 constraints will be created.

Note: Notice that the Zybo has an on-board clock of 125 Mhz and hence wizard generated the clk_pin constraint for a 8 ns period (or 125 MHz).

There is no need to click Apply since the constraints have already been applied in the Constraints Wizard.

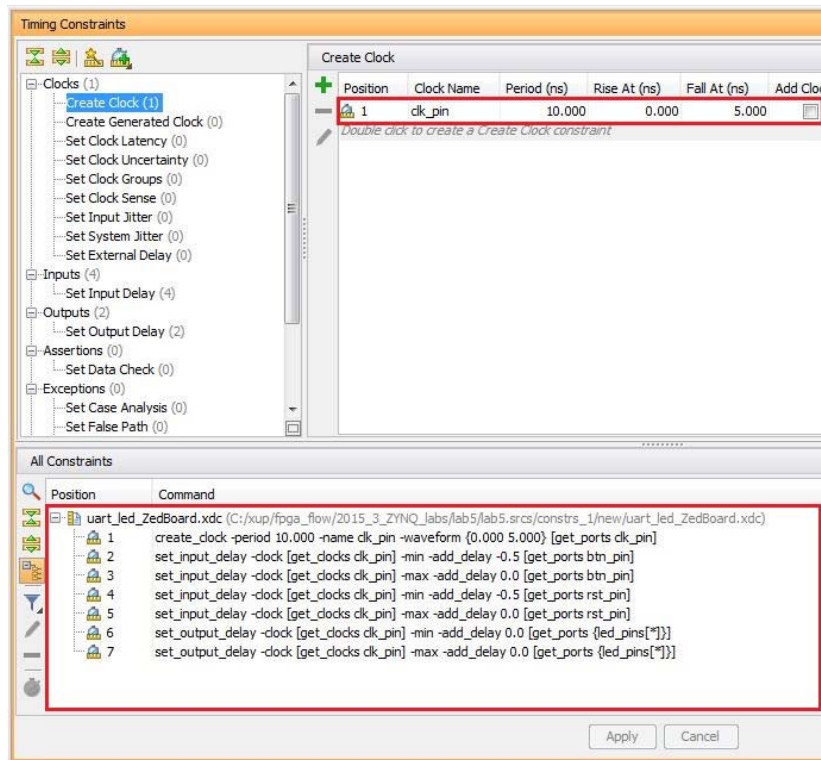


Figure 16. The constraints added after using the Constraints Wizard

3-1-15. Open uart_led_<board>.xdc (if it was already opened, click Reload in the yellow status bar) and notice additional constraints were added to the last line of the file.

3-2. Generate an estimated Timing Report showing both the setup and hold paths in the design.

3-2-1. In the Flow Navigator, select **Synthesized Design** > **Report Timing Summary**.

3-2-2. In the Options tab, select **min_max** from the Path delay type drop-down list.

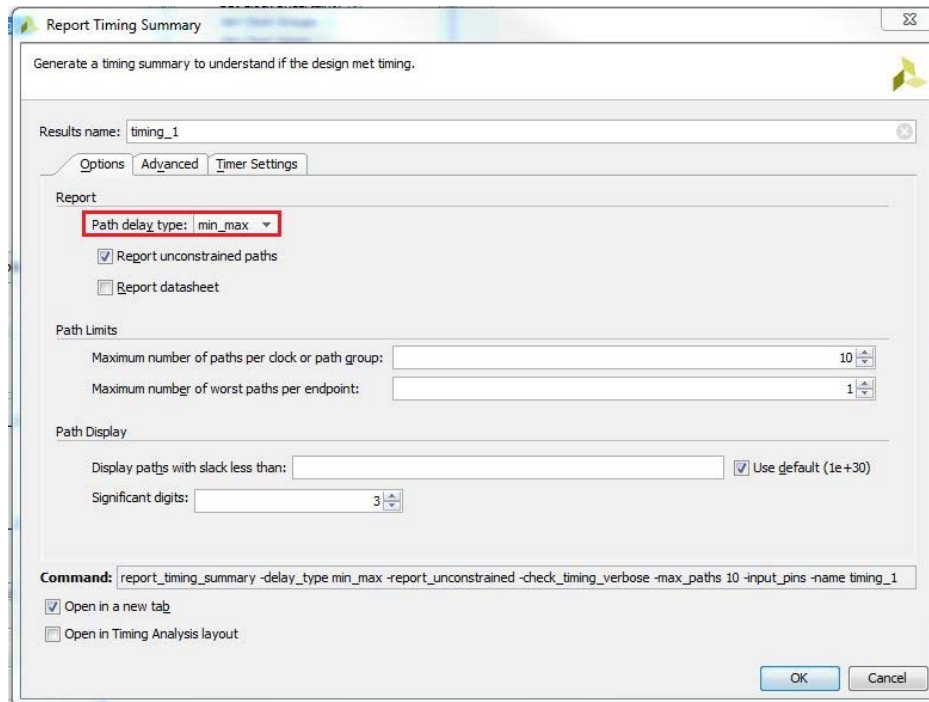


Figure 17. Performing timing analysis

3-2-3. Click **OK** to run the analysis.

The Timing Results view opens at the bottom of the Vivado IDE.

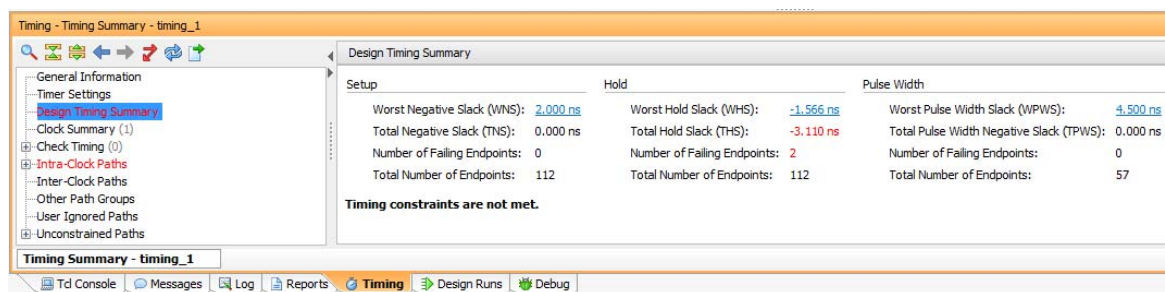


Figure 18. Timing summary for the ZedBoard

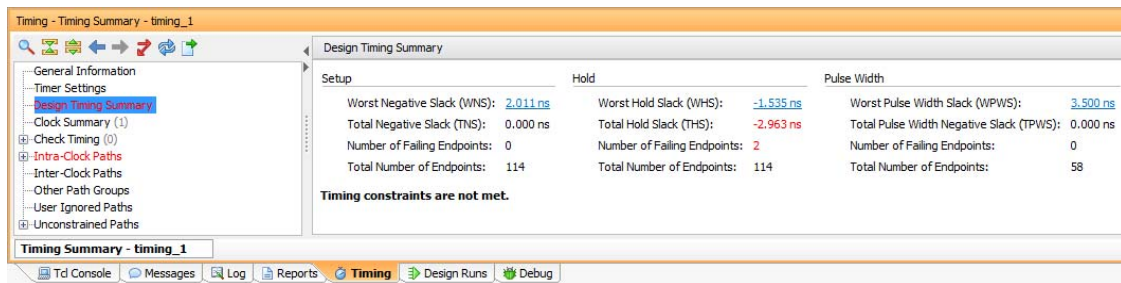


Figure 18. Timing summary for the Zybo

The *Design Timing Summary* report provides a brief worst Setup and Hold slack information and Number of failing endpoints to indicate whether the design has met timing or not.

Note that there are three timing failures under the hold check.

3-2-4. Click on the link next to *Worst Hold Slack* (WHS) to see the list of failing paths.

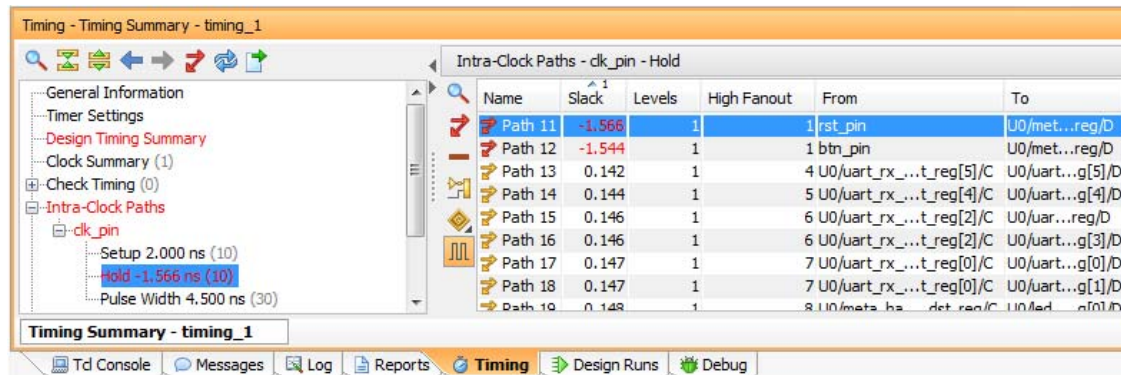


Figure 19. The list of paths showing hold violations for the ZedBoard

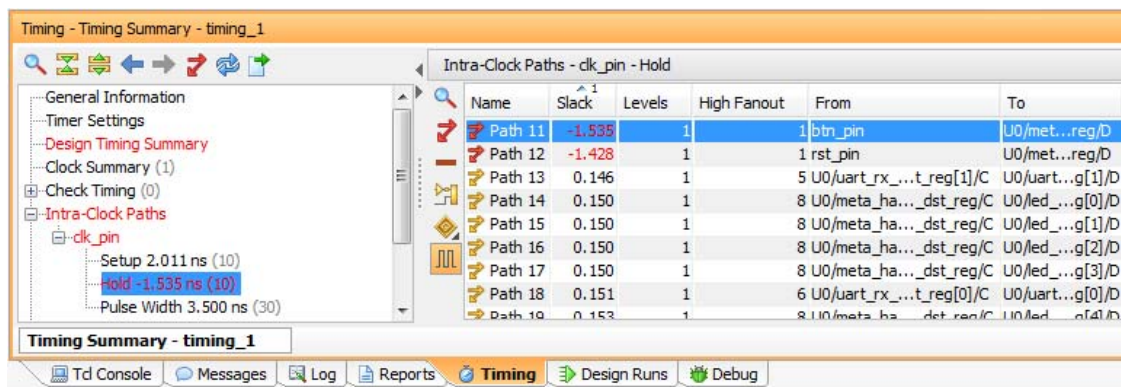


Figure 19. The list of paths showing hold violations for the Zybo

3-2-5. Double-click on the Path 11 to see the actual path detail.

Summary

Name	Path 11		
Slack (Hold)	-1.566ns		
Source	rst_pin (input port clocked by clk_pin {rise@0.000ns fall@5.000ns period=10.000ns})		
Destination	U0/meta_harden_rst_i0/signal_meta_reg/D (rising edge-triggered cell FDRE clocked by clk_		
Path Group	clk_pin		
Path Type	Hold (Min at Slow Process Corner)		
Requirement	0.000ns (clk_pin rise@0.000ns - clk_pin rise@0.000ns)		
Data Path Delay	2.138ns (logic 1.378ns (64.454%) route 0.760ns (35.546%))		
Logic Levels	1 (IBUF=1)		
Input Delay	-0.500ns		
Clock Path Skew	2.975ns		

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 0.000	0.000		
input delay	-0.500	-0.500		
net (fo=0)	(r) 0.000	-0.500	Site: P16	rst_pin rst_pin
			Site: P16	rst_pin_IBUF_inst/I
IBUF (Prop_ibuf_i_o)	(r) 1.378	0.878	Site: P16	rst_pin_IBUF_inst/O
net (fo=1, unplaced)	0.760	1.638		U0/meta_harden_rst_i0/rst_pin_IBUF
FDRE				U0/meta_harden_rst_i0/signal_meta_reg/D
Arrival Time		1.638		

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 0.000	0.000		
net (fo=0)	(r) 0.000	0.000	Site: Y9	clk_pin clk_pin
IBUF			Site: Y9	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf_i_o)	(r) 1.490	1.490	Site: Y9	clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.800	2.290		clk_pin_IBUF
BUFG				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_i_o)	(r) 0.101	2.391		clk_pin_IBUF_BUFG_inst/O
net (fo=56, unplaced)	0.584	2.975		U0/meta_harden_rst_i0/CLK
FDRE				U0/meta_harden_rst_i0/signal_meta_reg/C
clock pessimism	0.000	2.975		
FDRE (Hold_fdre_c_d)	0.228	3.203		U0/meta_harden_rst_i0/signal_meta_reg
Required Time		3.203		

Figure 20. Failing hold path for the ZedBoard

Summary

Name	Path 11			
Slack (Hold)	-1.535ns			
Source	btn_pin (input port clocked by clk_pin {rise@0.000ns fall@4.000ns period=8.000ns})			
Destination	U0/meta_harden_btn_i0/signal_meta_reg/D (rising edge-triggered cell FDRE clocked by clk_			
Path Group	clk_pin			
Path Type	Hold (Min at Slow Process Corner)			
Requirement	0.000ns (clk_pin rise@0.000ns - clk_pin rise@0.000ns)			
Data Path Delay	2.148ns (logic 1.389ns (64.631%) route 0.760ns (35.369%))			
Logic Levels	1 (IBUF=1)			
Input Delay	-0.500ns			
Clock Path Skew	2.976ns			

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 0.000	0.000		
input delay	-0.500	-0.500		
net (fo=0)	0.000	-0.500	Site: R18	btn_pin
				btn_pin
IBUF (Prop_ibuf_i_o)	(r) 1.389	0.889	Site: R18	btn_pin_IBUF_inst/I
net (fo=1, unplaced)	0.760	1.648		btn_pin_IBUF_inst/O
FDRE				U0/meta_harden_btn_i0/btn_pin
				U0/meta_harden_btn_i0/signal_meta_reg/D
Arrival Time		1.648		

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 0.000	0.000		
net (fo=0)	0.000	0.000	Site: L16	clk_pin
				clk_pin
IBUF			Site: L16	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf_i_o)	(r) 1.491	1.491	Site: L16	clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.800	2.291		clk_pin_IBUF
BUFG				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_i_o)	(r) 0.101	2.392		clk_pin_IBUF_BUFG_inst/O
net (fo=57, unplaced)	0.584	2.976		U0/meta_harden_btn_i0/CLK
FDRE				U0/meta_harden_btn_i0/signal_meta_reg/C
clock pessimism	0.000	2.976		
FDRE (Hold_fdre_c_d)	0.207	3.183		U0/meta_harden_btn_i0/signal_meta_reg
Required Time		3.183		

Figure 20. Failing hold path for the Zybo

3-2-6. Select Path11, right-click and select Schematic.

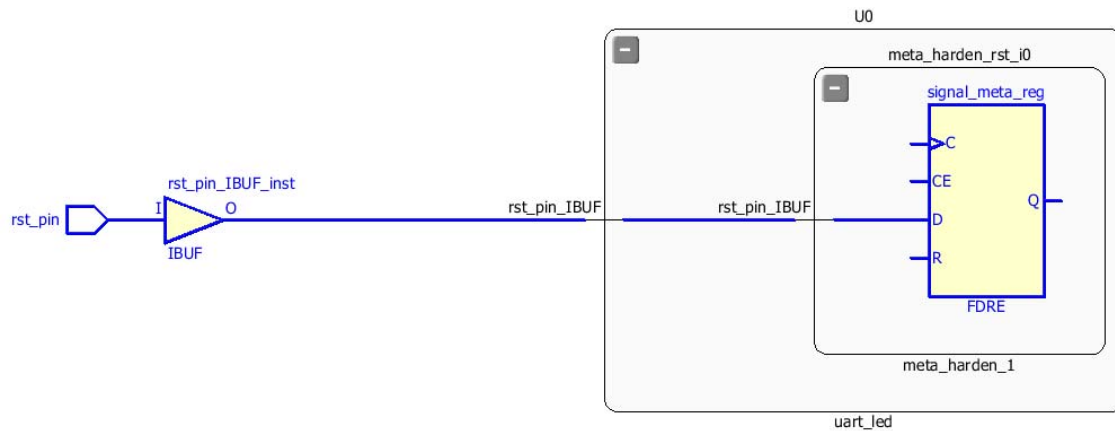


Figure 21. The schematic of the failing path for ZedBoard

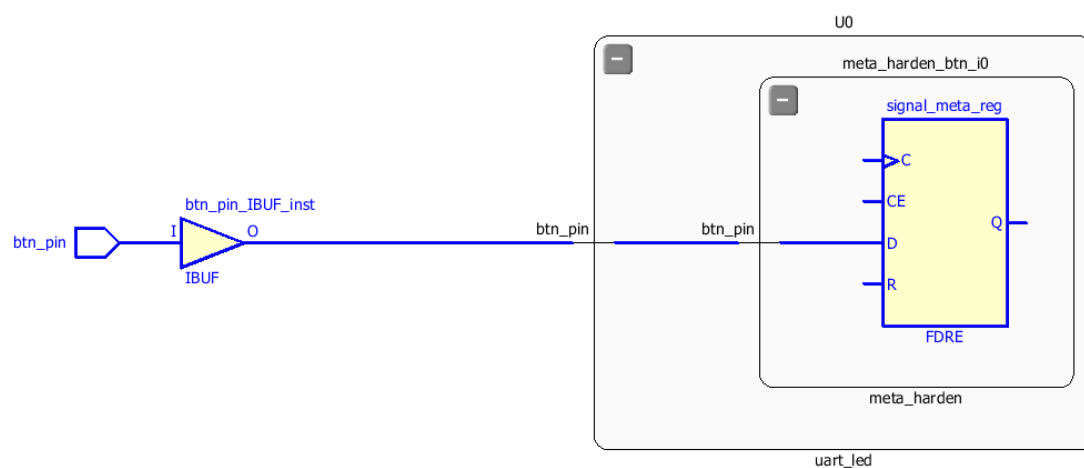


Figure 21. The schematic of the failing path for Zybo

The two failing paths are of the btn_pin and rst_pin.

Implement and Analyze Timing Summary

Step 4

4-1. Implement the design.

4-1-1. Click on the **Run Implementation** in the *Flow Navigator* pane.

4-1-2. Click **Yes** to run the synthesis first before running the implementation process.

When the implementation is completed, a dialog box will appear with three options.

4-1-3. Select the *Open Implemented Design* option and click **OK**.

4-1-4. Click **Yes** if you are prompted to close the synthesized design.

4-2. Generate a timing summary report.

4-2-1. In the Flow Navigator, under Implementation > Implemented Design, click **Report Timing Summary**.

4-2-2. Click **OK** to generate the report using the default settings.

The Design Timing Summary window opens at the bottom in the Timing tab.

Note that failing timing paths are indicated in red.

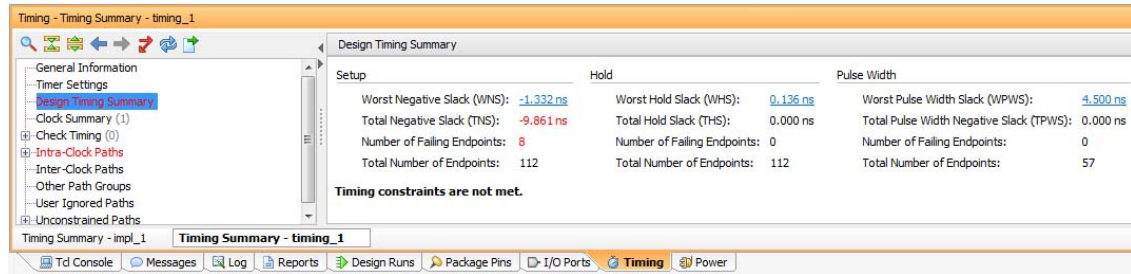


Figure 22. Failing setup paths for the ZedBoard

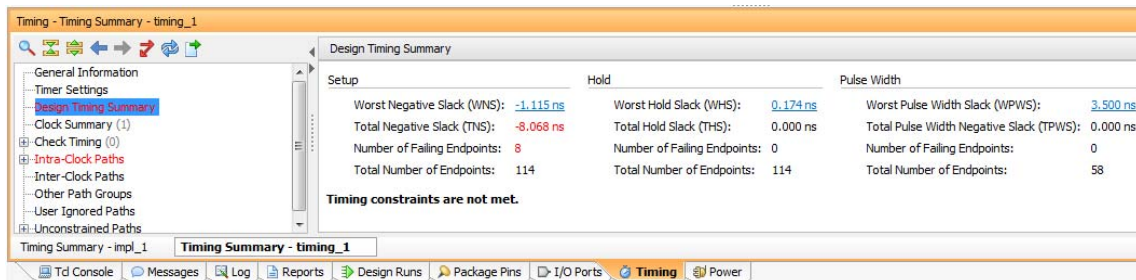


Figure 22. Failing setup paths for the Zybo

4-2-3. Click on the WNS to see the failing paths.

4-2-4. Double-click on the first failing path from the top and see the detailed analysis.

The output path delay can be reduced by placing the register in IOB.

4-2-5. Apply the constraint by typing the following command in the Tcl console.

```
set_property IOB TRUE [get_ports led_pins[*]]
```

4-2-6. Select **File > Save Constraints**. Click **OK** at the warning message. Click **Yes** to save the project.

4-2-7. Click on **Run Implementation**.

4-2-8. Click **Yes** to reset the synthesis run, perform the synthesis, and run implementation.

4-2-9. Open the implemented design and observe that the number of failing paths in the Design Runs tab reported is 0.

4-2-10. Click **Report Timing Summary**, and observe that there are no failing paths.

Generate the Bitstream and Verify the Functionality (Optional) Step 5

5-1. Generate the bitstream.

- 5-1-1. In the Flow Navigator, under *Program and Debug*, click **Generate Bitstream**.
- 5-1-2. The `write_bitstream` command will be executed (you can verify it by looking in the Tcl console).
- 5-1-3. Click **Cancel** when the bitstream generation is completed.

5-2. Connect the board and power it ON. Open a hardware session, and program the FPGA.

- 5-2-1. 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).
- 5-2-2. Turn ON the power.
- 5-2-3. Select the *Open Hardware Manager* option.
The Hardware Manager window will open indicating “unconnected” status.
- 5-2-4. 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.
- 5-2-5. 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.
- 5-2-6. Select the device and verify that the **ios.bit** is selected as the programming file in the General tab.

5-3. 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.

- 5-3-1. Start a terminal emulator program such as TeraTerm or HyperTerminal.
- 5-3-2. Select an appropriate COM port (you can find the correct COM number using the Control Panel).
- 5-3-3. Set the COM port for 115200 baud rate communication.
- 5-3-4. Right-click on the FPGA entry in the Hardware window and select Programming Device...
- 5-3-5. Click on the **Program** button.

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

5-4. Start a SDK session, point it to the c:/xup/fpga_flow/2016_2_ZYNQ_Sources /lab5/<board>/lab5.sdk workspace.

5-4-1. Open **SDK** by selecting **Start > All Programs > Xilinx Design Tools > SDK 2016.2 > Xilinx SDK 2016.2**

5-4-2. In the **Select a workspace** window, click on the browse button, browse to c:/xup/fpga_flow/2016_2_ZYNQ_Sources /lab5/ directory and select either c:/xup/fpga_flow/2016_2_ZYNQ_Sources /lab5/Zybo/lab5.sdk or c:/xup/fpga_flow/2016_2_ZYNQ_Sources /lab5/ ZedBoard/lab5.sdk and click **OK**.

5-4-3. Click **OK**.

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

5-4-4. Verify the functionality as you did in the previous lab, by typing some characters into the terminal, and watching the corresponding values appear on the LEDs.

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

5-4-6. Select **File > Close Hardware Manager**. Click **OK** to close it.

5-4-7. When done, close the **Vivado** program by selecting **File > Exit** and click **OK**.

5-4-8. Close the **SDK** program by selecting **File > Exit** and click **OK**.

Conclusion

In this lab, you learned how to create an I/O Planning project and assign the pins via the Device view, Package Pins tab, and the Tcl commands. You then exported to the rtl project where you added the provided source files. Next you created timing constraints and performed post-synthesis and post-implementation timing analysis.