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 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 LEDs. Depending on the target board, when a push button is pressed, the lower and upper nibbles are displayed on the lower-half and/or upper-half of the LED array. The block diagram is as shown in **Figure 1**.

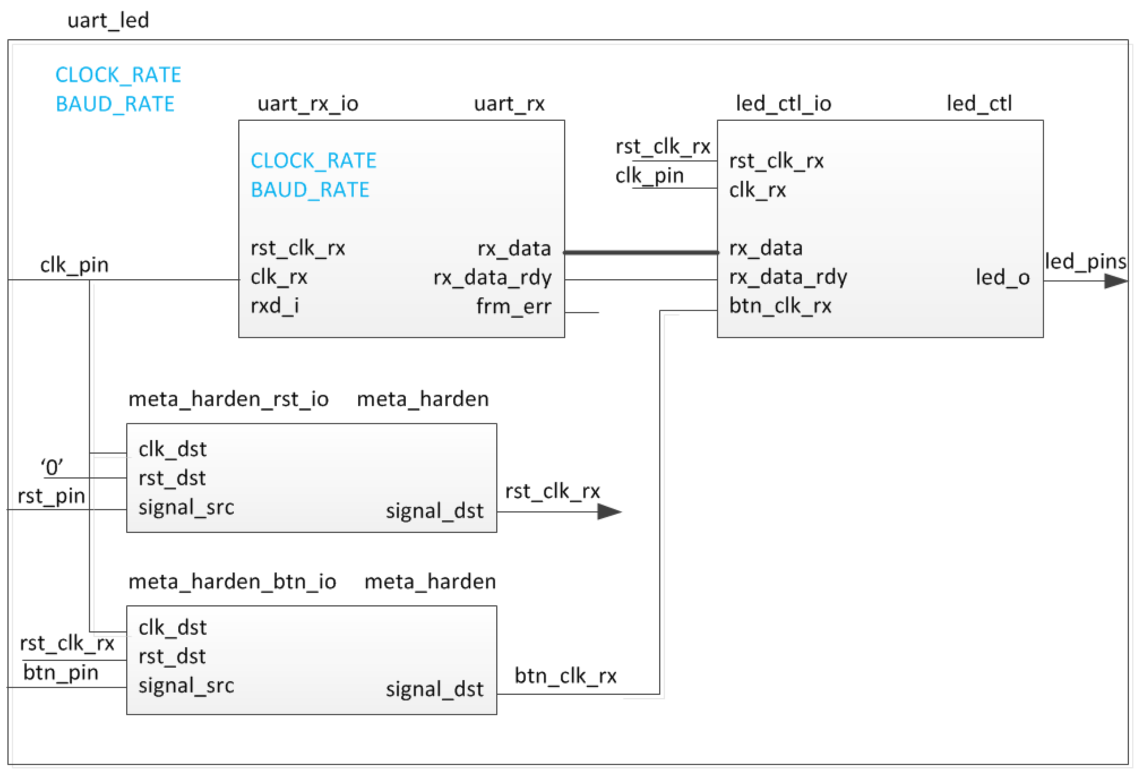
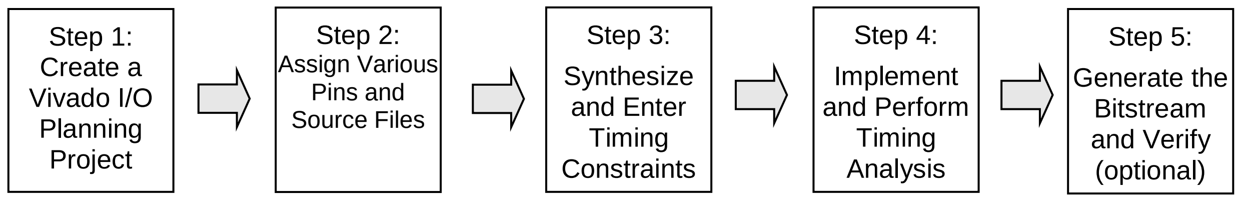


Figure 1. The design

General Flow


Step 1: Create a Vivado I/O Planning Project
Step 2: Assign Various Pins and Source Files
Step 3: Synthesize and Enter Timing Constraints
Step 4: Implement and Perform Timing Analysis
Step 5: Generate the Bitstream and Verify (optional)

Create a Vivado I/O Planning Project Step 1

1-1. Launch Vivado and create a I/O Planning project targeting either the XC7Z010CLG400-1 or XC7Z020CLG484-1 device, depending on the target board.

References to **<2014_2_zynq_labs>** is a placeholder for the **c:\xup\fpga_flow\2014_2_zynq_labs** directory and **<2014_2_zynq_sources>** is a placeholder for the **c:\xup\fpga_flow\2014_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 2014.2 > Vivado 2014.2**

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 **<2014_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**.

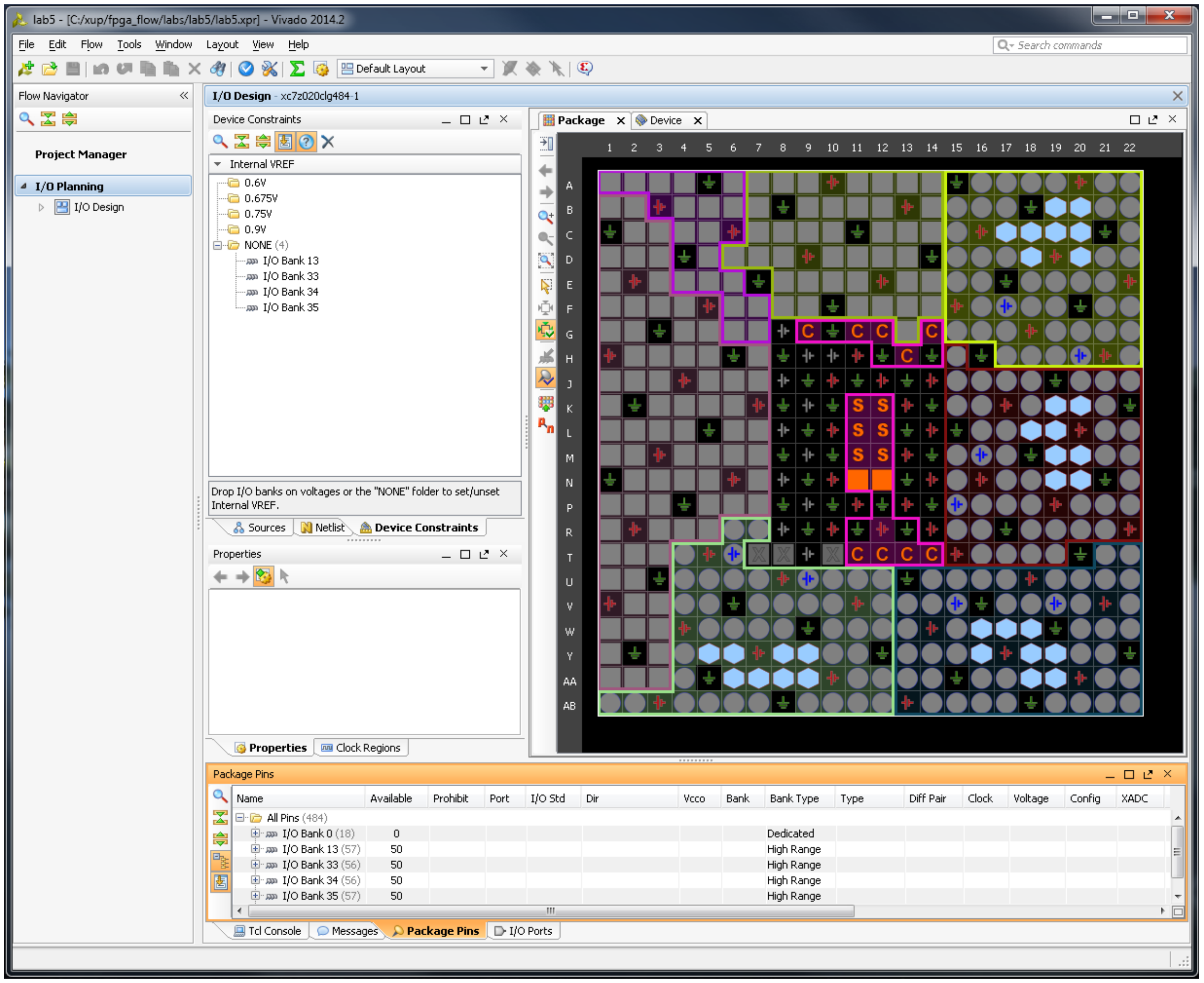
1-1-5. Select **I/O Planning Project** option in the *Project Type* form, and click **Next**.

1-1-6. Select **Do not import I/O ports at this time**, and click **Next**.

1-1-7. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section, select the **XC7Z020CLG484-1** part for the ZedBoard or the **ZC7Z010CLG400-1** part for the Zybo. Click **Next**.

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

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

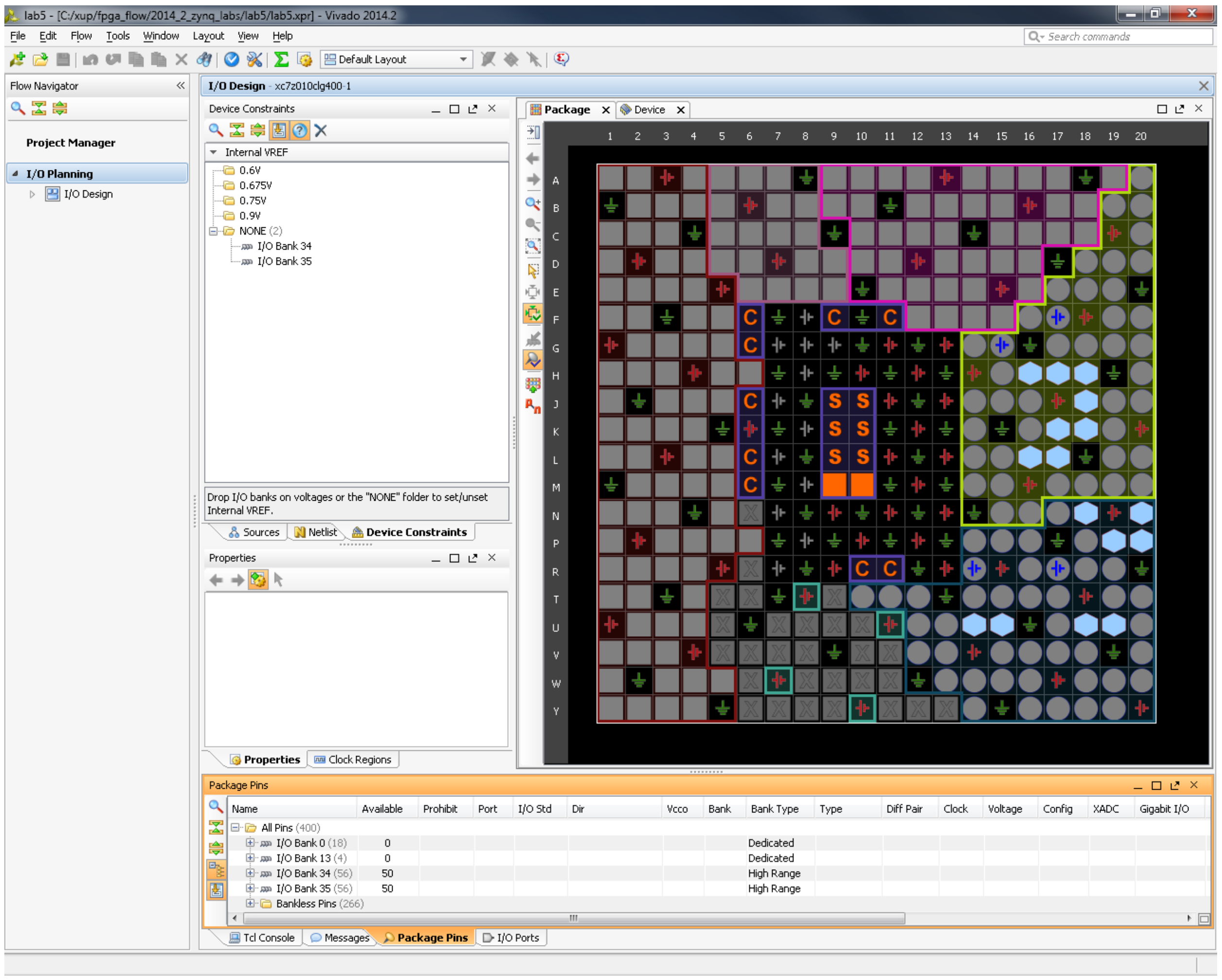


Internal VREF
- 0.6V
- 0.675V
- 0.75V
- 0.9V
- NONE (4)
  - I/O Bank 13
  - I/O Bank 33
  - I/O Bank 34
  - I/O Bank 35

Drop I/O banks on voltages or the "NONE" folder to set/unset Internal VREF.

Name	Available	Prohibit	Port	I/O Std	Dir	Vcco	Bank	Bank Type	Type	Diff Pair	Clock	Voltage	Config	XADC
All Pins (484)														
I/O Bank 0 (18)	0							Dedicated						
I/O Bank 13 (57)	50							High Range						
I/O Bank 33 (56)	50							High Range						
I/O Bank 34 (56)	50							High Range						
I/O Bank 35 (57)	50							High Range						

Figure 2. I/O Planning project's default windows for the ZedBoard's XC7Z020CLG484-1



Drop I/O banks on voltages or the "NONE" folder to set/unset Internal VREF.

Package Pins

Name	Available	Prohibit	Port	I/O Std	Dir	Vcco	Bank	Bank Type	Type	Diff Pair	Clock	Voltage	Config	XADC	Gigabit I/O
All Pins (400)															
I/O Bank 0 (18)	0							Dedicated							
I/O Bank 13 (4)	0							Dedicated							
I/O Bank 34 (56)	50							High Range							
I/O Bank 35 (56)	50							High Range							
Bankless Pins (266)															

Figure 2. I/O Planning project's default windows for the Zybo's XC7Z010CLG400-1

Assign Various Pins and Add Source Files Step 2

2-1. Assign input pin locations using the Device view and package pins. For the ZedBoard, pins clk_pin, btn_pin, rxd_pin, and rst_pin to Y9, T18, Y10, and P16.

For the Zybo, pins clk_pin, btn_pin, rxd_pin, and rst_pin to L16, R18, J15, and P16.

2-1-1. Move mouse over the Device view window and hover over it on the **Y9** location for the ZedBoard. For the Zybo, place the cursor over location **L16**.

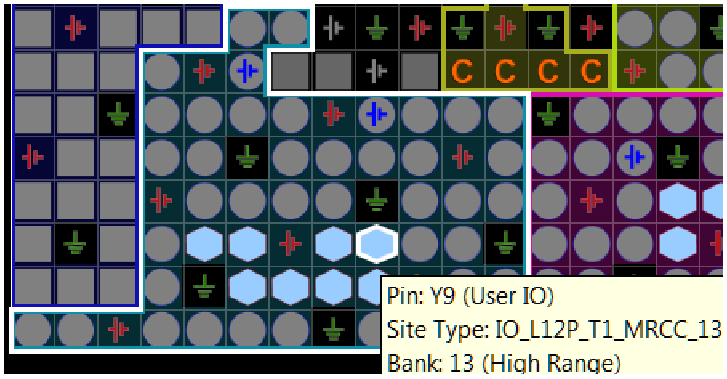

Pin: Y9 (User IO)
Site Type: IO_L12P_T1_MRCC_13
Bank: 13 (High Range)

Figure 3. Locating a pin in the Device view for the XC7Z020CLG484-1

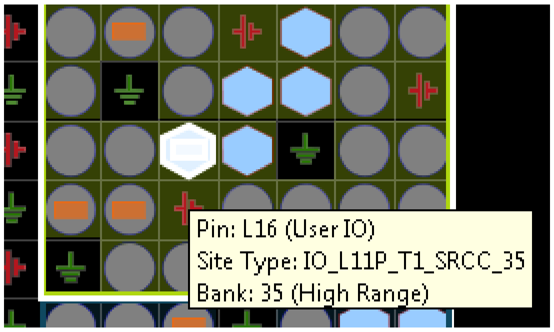

Pin: L16 (User IO)
Site Type: IO_L11P_T1_SRCC_35
Bank: 35 (High Range)

Figure 3. Locating a pin in the Device view for the XC7Z010CLG400-1

2-1-2. When located, click on it. The Y9 (or L16) entry will be displayed in the Package Pins tab.

2-1-3. In the *Package* Pins pane, click in the Port column of Y9 pin's row, enter **clk_pin.** For the Zybo, do the same except for the Port column in L16's row.

2-1-4. Click in the next column, *I/O Std*.

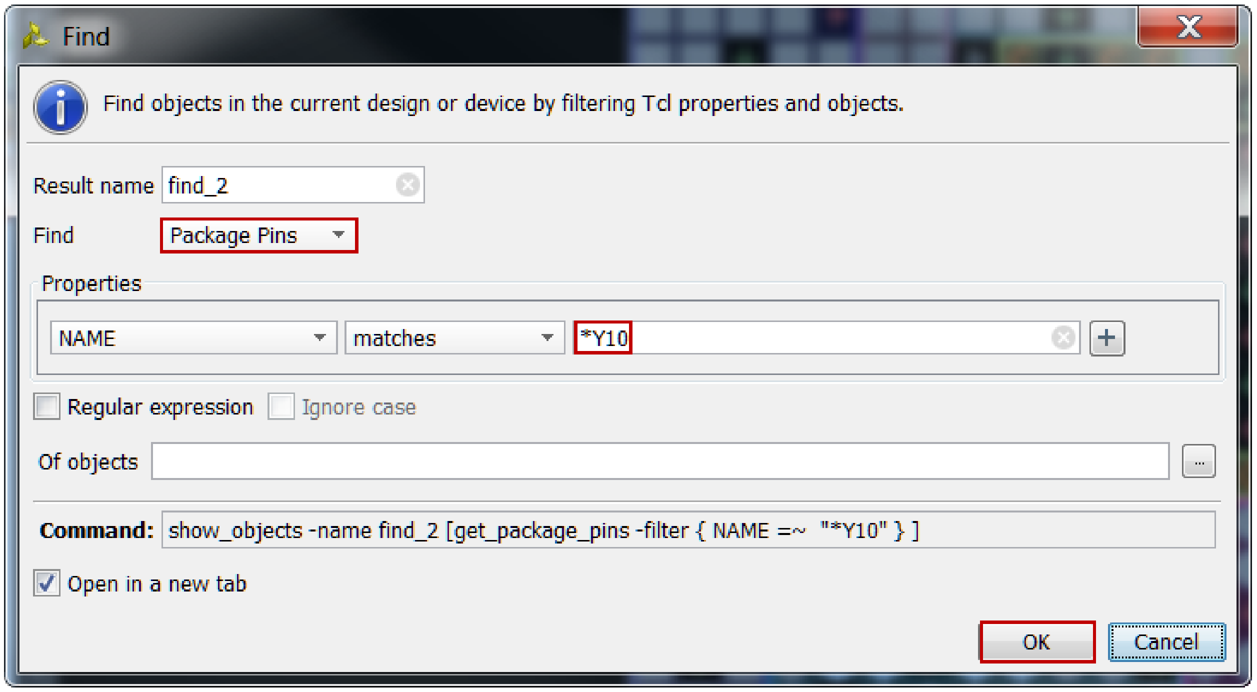
Notice that LVCMOS18 may be displayed in Red due to an IO standard mismatch.

2-1-5. Click on the *LVCMOS18* entry to see a pop-up window. Scroll down and select **LVCMOS33** and then hit **Enter**. The LVCMOS33 is displayed in black. Keep the direction as input since clk_pin is an input port.

2-1-6. Similarly, add the **btn_pin** input port at **T18** with *LVCMOS33* standard. For the Zybo, the port is at **R18**.

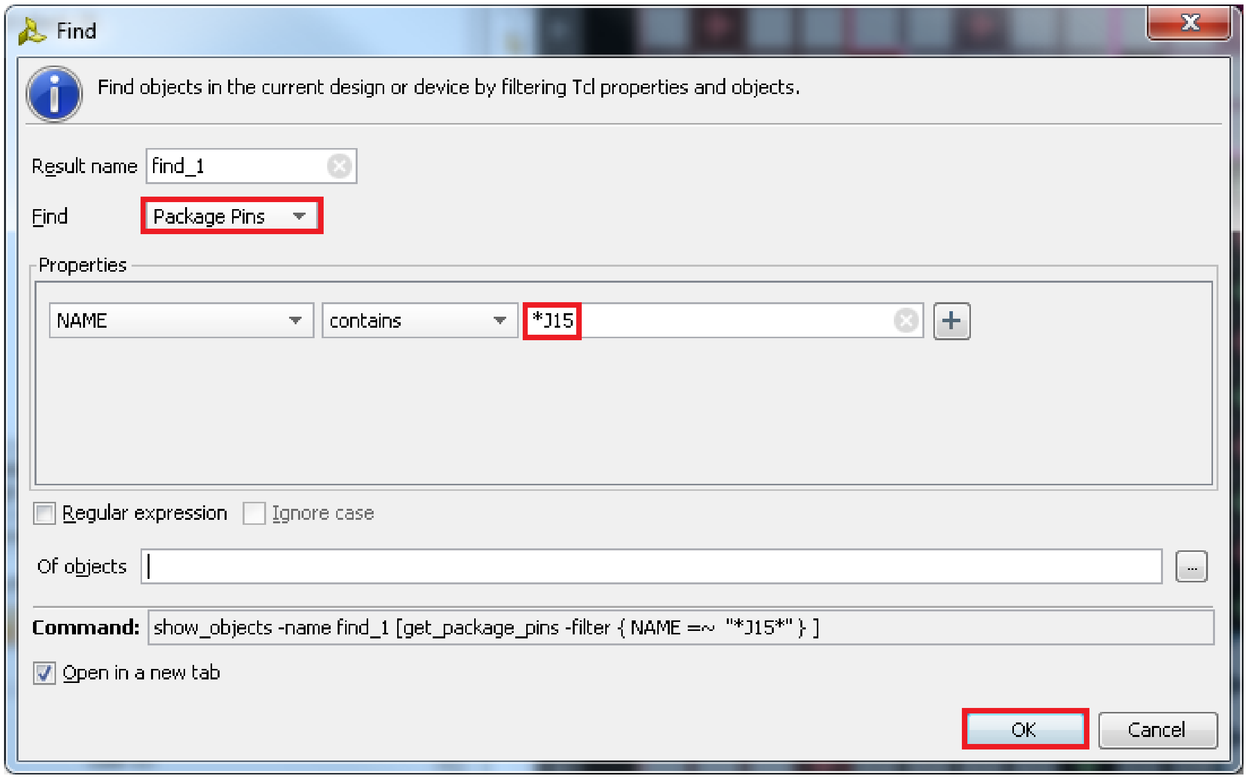
2-1-7. Select **Edit > Find** or Ctrl-F to open the Find form. Select **Package Pins** in the *Find* drop-down field, type ***Y10** in the match criteria field. For the Zybo, search for "**J15".**

Click **OK**.



Command: `show_objects -name find_2 [get_package_pins -filter { NAME =~ "*Y10" } ]`

Figure 4. Finding a package pin on the ZedBoard's XC7Z020CLG484-1



Command: `show_objects -name find_1 [get_package_pins -filter { NAME =~ "*J15*" } ]`

Figure 4. Finding a package pin on the Zybo's XC7Z010CLG400-1

Notice that the pin is highlighted in the Device view and the corresponding entry is displayed in the Package Pins tab.

2-1-8. Assign **rxd_pin** to Y10 or J15 with *LVCMOS33* standard.

2-1-9. Similarly add the **rst_pin** input assigning it to **P16** location with *LVCMOS33* standard. For the Zybo, search for the same pin **P16**.

2-1-10. Add the **zed_pin** input assigning it to **M15** location with *LVCMOS33* standard. For the Zybo, search for the same pin **Y18**.

2-2. For the ZedBoard, ssign output pins led_pins[7] to led_pins[0] to U14, U19, W22, V22, U21, U22, T21, and T22 locations.

For the Zybo, assign output pins led_pins[3] to led_pins[0] to D18, G14, M15, and M14 locations.

Create them as a vector and assign them using Tcl command set_propoerty. They all will be LVCMOS33.

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

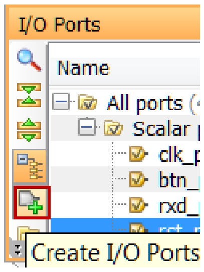
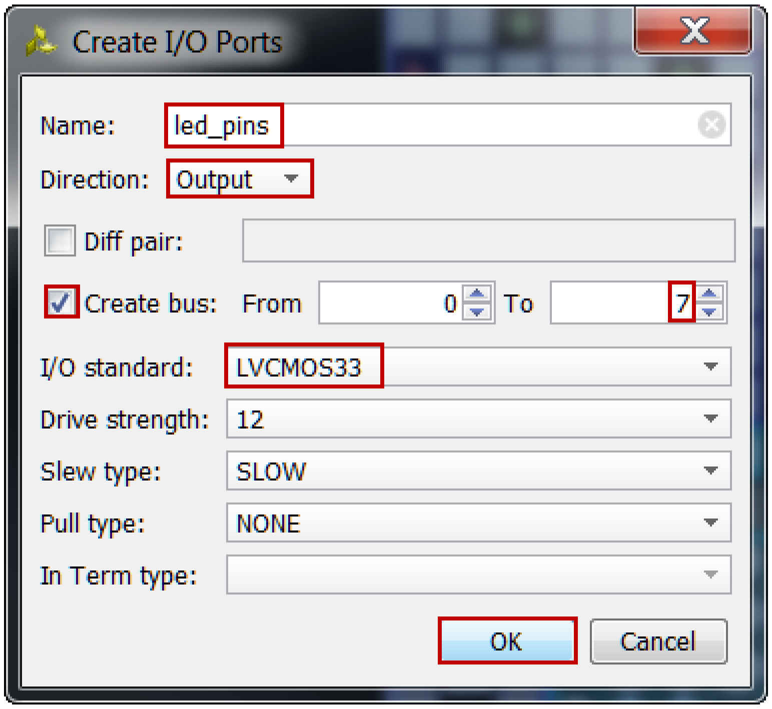


Figure 5. Create I/O Ports button

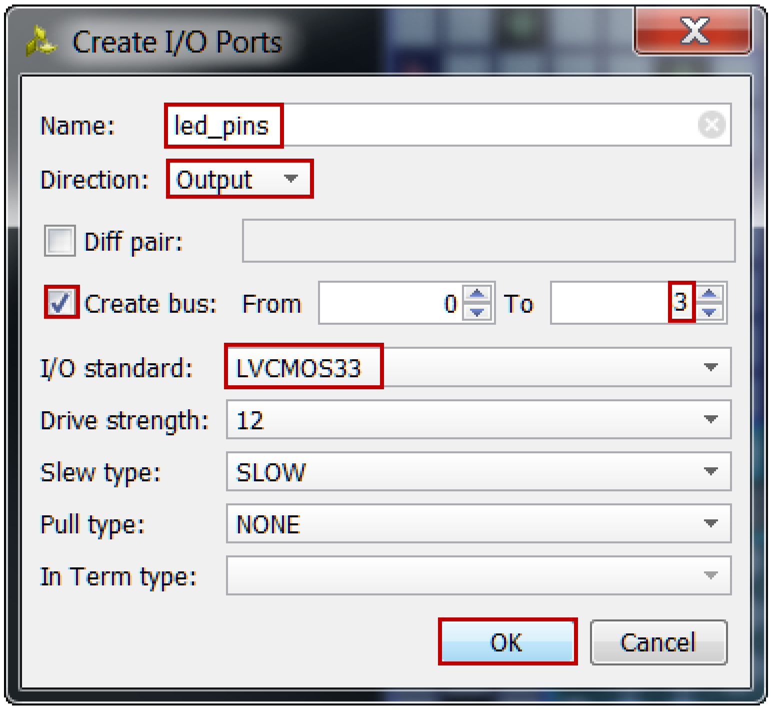
The Create I/O Ports form will be displayed.

2-2-2. Type **led_pins** in the *Name* field, select *Output* direction, click on the check-box of **Create bus**, set the msb to **7** for the ZedBoard, and select **LVCMOS33** I/O standard. For the Zybo, set the msb to **3** and everything else stays the same. Click **OK**.



Name: led_pins
Direction: Output
Diff pair: ☐
Create bus: ✓ From 0 To 7
I/O standard: LVCMOS33
Drive strength: 12
Slew type: SLOW
Pull type: NONE
In Term type:

Figure 6. Creating I/O ports for the led_pins output for the ZedBoard



Name: led_pins
Direction: Output
Diff pair: ☐
Create bus: ✓ From 0 To 3
I/O standard: LVCMOS33
Drive strength: 12
Slew type: SLOW
Pull type: NONE
In Term type:

Figure 6. Creating I/O ports for the led_pins output for the Zybo

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.

2-2-3. In the Tcl console, type the following commands to assign the pin locations for the ZedBoard:

```
set_property PACKAGE_PIN T22 [get_ports led_pins[0]]
set_property PACKAGE_PIN T21 [get_ports led_pins[1]]
set_property PACKAGE_PIN U22 [get_ports led_pins[2]]
set_property PACKAGE_PIN U21 [get_ports led_pins[3]]
set_property PACKAGE_PIN V22 [get_ports led_pins[4]]
set_property PACKAGE_PIN W22 [get_ports led_pins[5]]
set_property PACKAGE_PIN U19 [get_ports led_pins[6]]
set_property PACKAGE_PIN U14 [get_ports led_pins[7]]
```

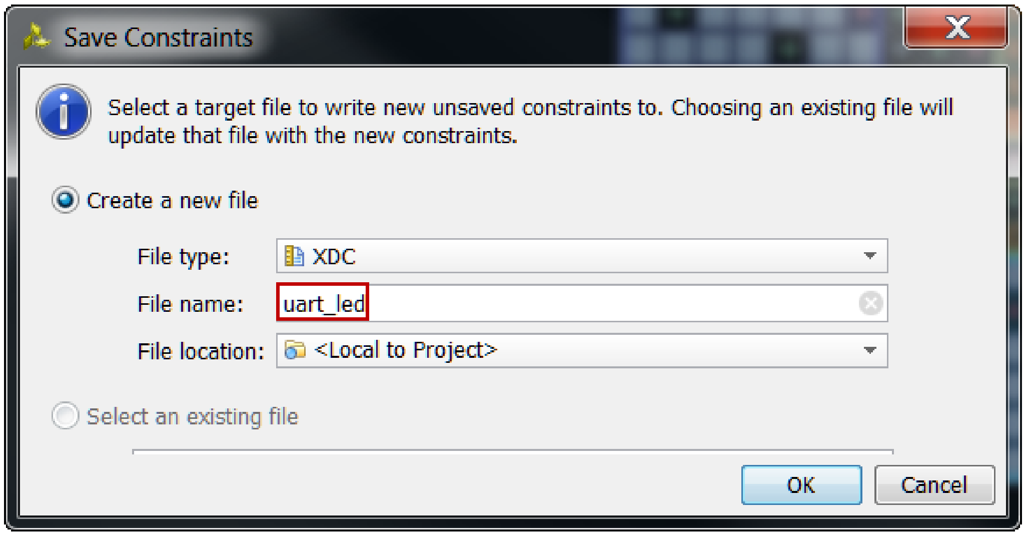
For the Zybo type the following commands:

```
set_property PACKAGE_PIN M14 [get_ports led_pins[0]]
set_property PACKAGE_PIN M15 [get_ports led_pins[1]]
set_property PACKAGE_PIN G14 [get_ports led_pins[2]]
set_property PACKAGE_PIN D18 [get_ports led_pins[3]]
```

2-2-4. Select **File > Save Constraints**.

The Save Constraints form will be displayed.

2-2-5. Enter **uart_led** in the *File name* field, and click **OK**.


Save Constraints

Select a target file to write new unsaved constraints to. Choosing an existing file will update that file with the new constraints.

(●) Create a new file

File type: XDC

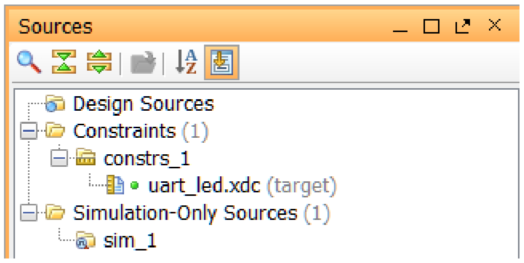
File name: uart_led

File location: <Local to Project>

( ) Select an existing file

Figure 7. Accessing clocking wizard

The uart_led.xdc file will be created and added to the Sources tab.


Sources

Design Sources
Constraints (1)
constrs_1
uart_led.xdc (target)
Simulation-Only Sources (1)
sim_1

Figure 8. The uart_led.xdc file added to the source tree

2-2-6. Expand the **I/O Planning > I/O Design** in the *Flow Navigator* pane.

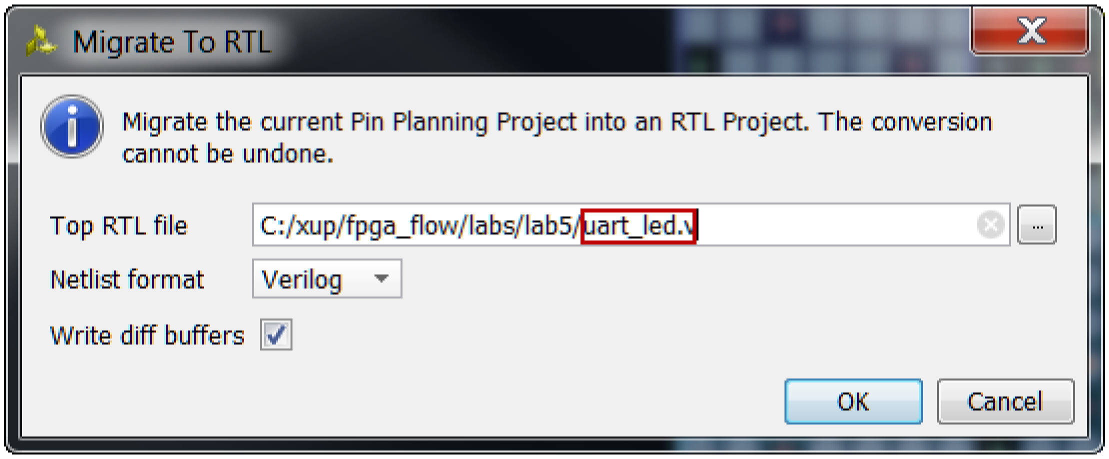
2-2-7. Click on **Report DRC** and click **OK**. Notice the design rules checker is run and some warnings are reported. Ignore these warnings.

2-2-8. 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.

2-2-9. Click on **Migrate to RTL.**

The Migrate to RTL form will be displayed with Top RTL file field showing c:/xup/fpga_flow/labs/lab5/io_1.v entry.

2-2-10. Change *io_1.v* to **uart_led.v**. Select the target language as Verilog and click **OK**


Migrate To RTL

Migrate the current Pin Planning Project into an RTL Project. The conversion cannot be undone.

Top RTL file: C:/xup/fpga_flow/labs/lab5/uart_led.v
Netlist format: Verilog
Write diff buffers: ✓

Figure 9. Assigning top-level file name

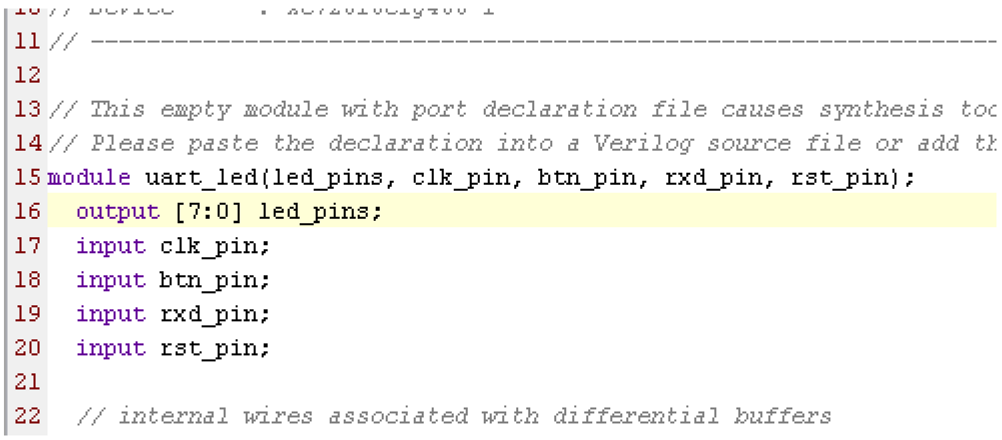
2-2-11. Select the **Hierarchy** tab in the Sources pane and notice that the *uart_led.v* file has been added to the project with top-level module name as **ios**. If you double-click the entry, you will see the module name with the ports listing. Rename the module to **uart_led**.

2-2-12. Open the XDC file under constrains and comment out these lines with a hash mark #:

For the ZedBoard:
```
set_property direction IN [get_ports {clk_pin}]
set_property direction IN [get_ports {btn_pin}]
set_property direction IN [get_ports {rxd_pin}]
set_property direction IN [get_ports {rst_pin}]
set_property direction OUT [get_ports {led_pins[0]}]
set_property direction OUT [get_ports {led_pins[1]}]
set_property direction OUT [get_ports {led_pins[2]}]
set_property direction OUT [get_ports {led_pins[3]}]
set_property direction OUT [get_ports {led_pins[4]}]
set_property direction OUT [get_ports {led_pins[5]}]
set_property direction OUT [get_ports {led_pins[6]}]
set_property direction OUT [get_ports {led_pins[7]}]
```

For the Zybo:
```
set_property direction IN [get_ports {clk_pin}]
set_property direction IN [get_ports {btn_pin}]
set_property direction IN [get_ports {rxd_pin}]
set_property direction IN [get_ports {rst_pin}]
```

set_property direction OUT [get_ports {led_pins[0]}]
set_property direction OUT [get_ports {led_pins[1]}]
set_property direction OUT [get_ports {led_pins[2]}]
set_property direction OUT [get_ports {led_pins[3]}]

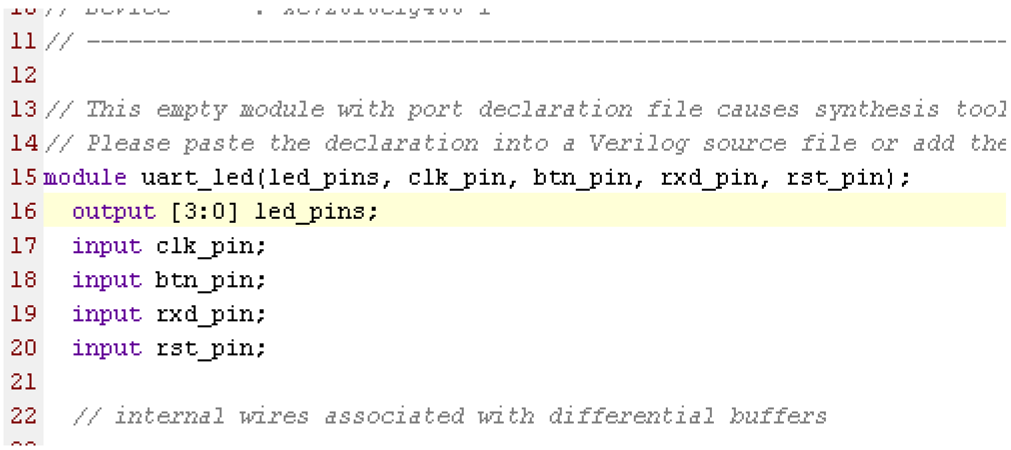


```
// ----------------------------------------------------------------------

// This empty module with port declaration file causes synthesis toc
// Please paste the declaration into a Verilog source file or add tl
module uart_led(led_pins, clk_pin, btn_pin, rxd_pin, rst_pin);
  output [7:0] led_pins;
  input clk_pin;
  input btn_pin;
  input rxd_pin;
  input rst_pin;

  // internal wires associated with differential buffers
```

Figure 10. The top-level module pin declaration after migrating to RTL for the ZedBoard



```
// ----------------------------------------------------------------------

// This empty module with port declaration file causes synthesis tool
// Please paste the declaration into a Verilog source file or add the
module uart_led(led_pins, clk_pin, btn_pin, rxd_pin, rst_pin);
  output [3:0] led_pins;
  input clk_pin;
  input btn_pin;
  input rxd_pin;
  input rst_pin;

  // internal wires associated with differential buffers
```

Figure 10. The top-level module pin declaration after migrating to RTL for the Zybo

2-3. Add the provided source files (from <2014_2_zynq_sources>\<board>\lab5) to the project. Copy the uart_led.txt (located in the <2014_2_zynq_sources>\<board>\lab5) content into the top-level source file.

2-3-1. Click on **Add Sources** in the *Flow Navigator*.

2-3-2. In the *Add Sources* form, select *Add or Create Design Sources*, and click **Next**.

2-3-3. Click **Add Files…**

2-3-4. Browse to **<2014_2_zynq_sources>\<board>\lab5** and select all *.v* files (led_ctrl, uart_rx, meta_harden, uart_baud_gen, uart_rx_ctl), and click **OK**.

2-3-5. Make sure the source files are copied into the project with the correct check-box. Click **Finish**.

2-3-6. Using Windows Explorer, browse to **<2014_2_zynq_sources>\<board>\lab5** and open uart_led.txt using any text editor. Copy the content of it and paste it in uart_led.v (around line 21) in the Vivado project.Select **File > Save File.**

Synthesize and Enter Timing Constraints — Step 3

3-1. Synthesize the design.

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

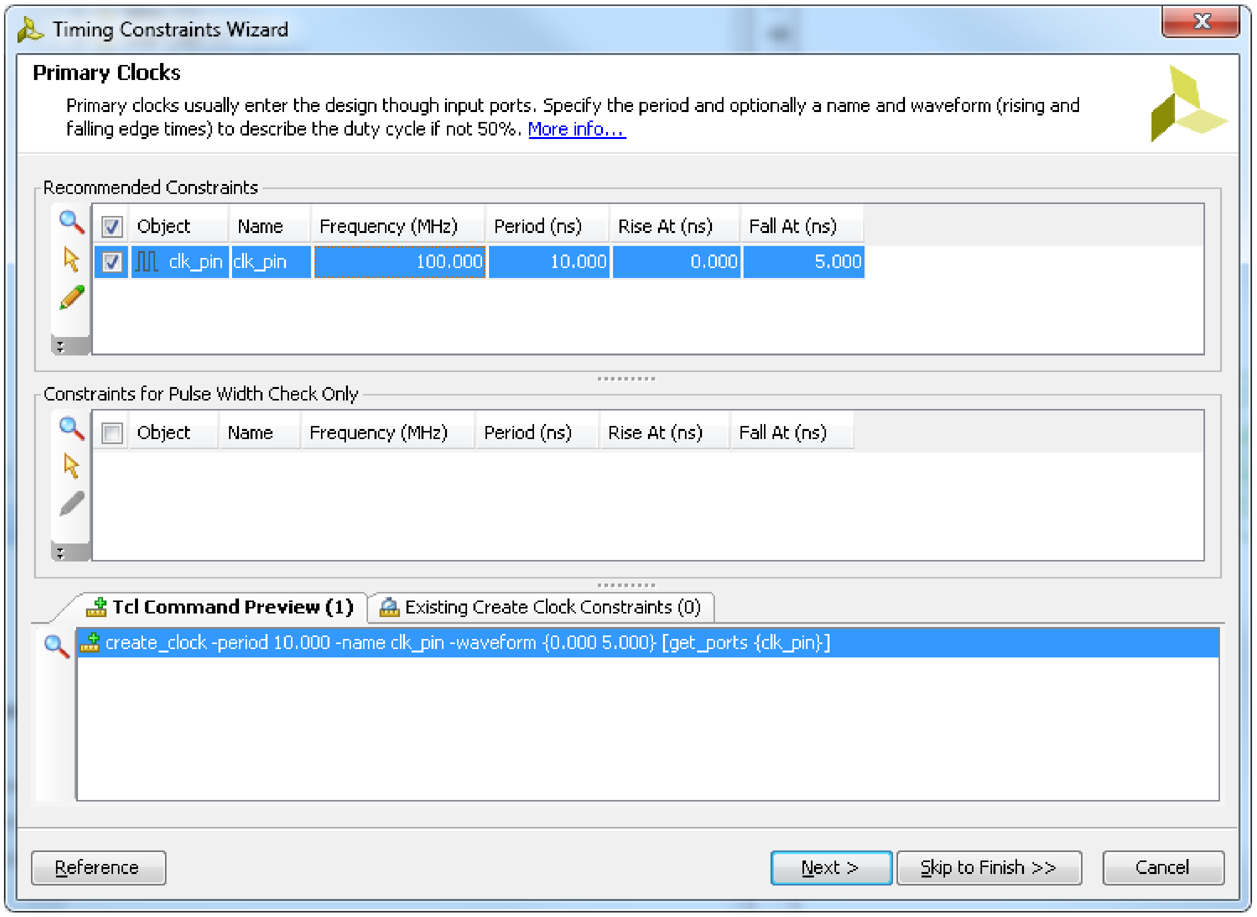
When synthesis is completed a form with three options will be displayed.

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** for the ZedBoard or **125 MHz** for the Zybo. 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.



Timing Constraints Wizard

Primary Clocks

Primary clocks usually enter the design though input ports. Specify the period and optionally a name and waveform (rising and falling edge times) to describe the duty cycle if not 50%. More info...

Recommended Constraints

☑	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)
☑	clk_pin	clk_pin	100.000	10.000	0.000	5.000

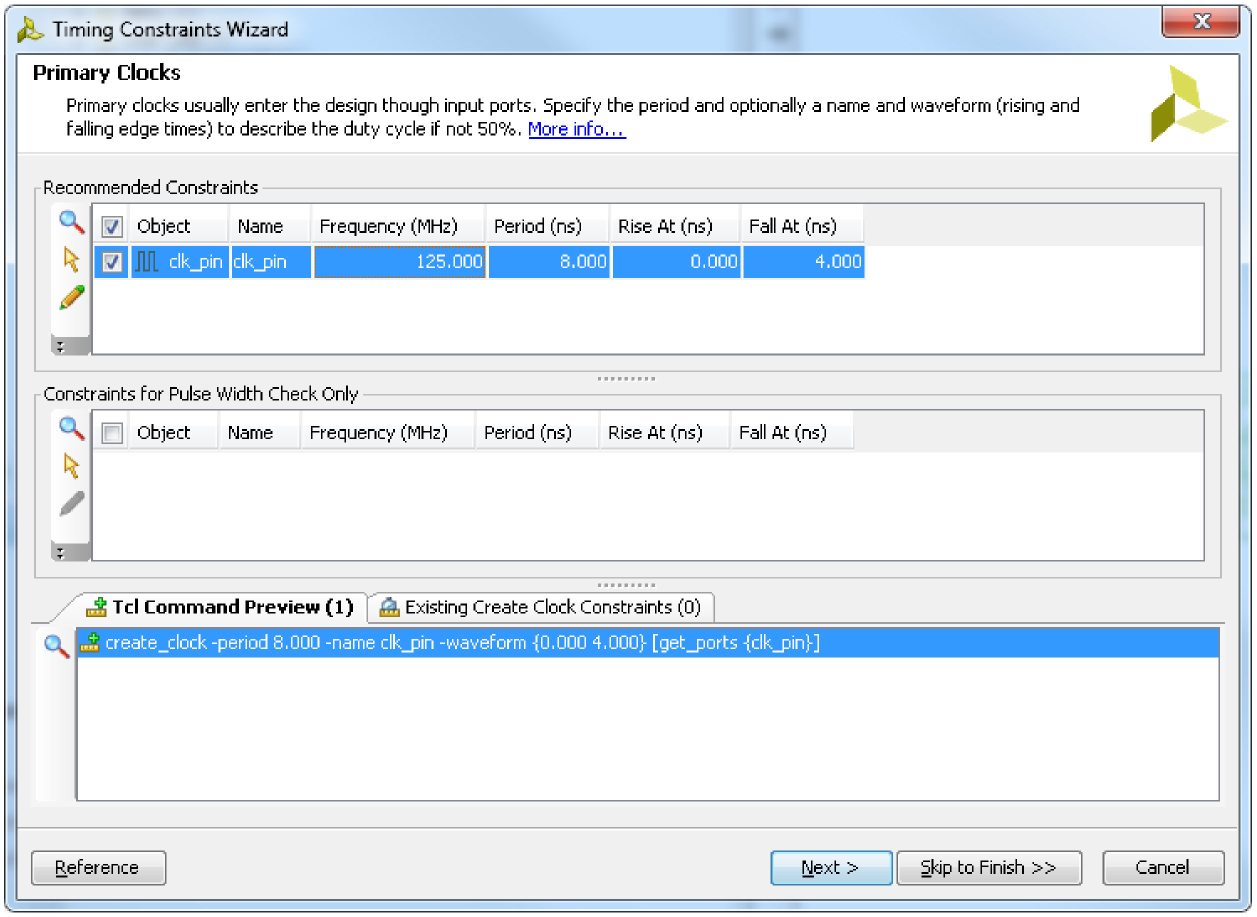
Constraints for Pulse Width Check Only

☐	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)

Tcl Command Preview (1) | Existing Create Clock Constraints (0)

```
create_clock -period 10.000 -name clk_pin -waveform {0.000 5.000} [get_ports {clk_pin}]
```

Figure 11. Constraints Wizard *clk_pin* parameters and Tcl command for the ZedBoard (100 MHz)



Primary Clocks

Primary clocks usually enter the design though input ports. Specify the period and optionally a name and waveform (rising and falling edge times) to describe the duty cycle if not 50%. More info...

Recommended Constraints

✓	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)
✓	clk_pin	clk_pin	125.000	8.000	0.000	4.000

Constraints for Pulse Width Check Only

	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)

Tcl Command Preview (1) | Existing Create Clock Constraints (0)

```
create_clock -period 8.000 -name clk_pin -waveform {0.000 4.000} [get_ports {clk_pin}]
```

Figure 11. Constraints Wizard *clk_pin* parameters and Tcl command for the Zybo (125 MHz)

3-1-6. The wizard informs you 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.

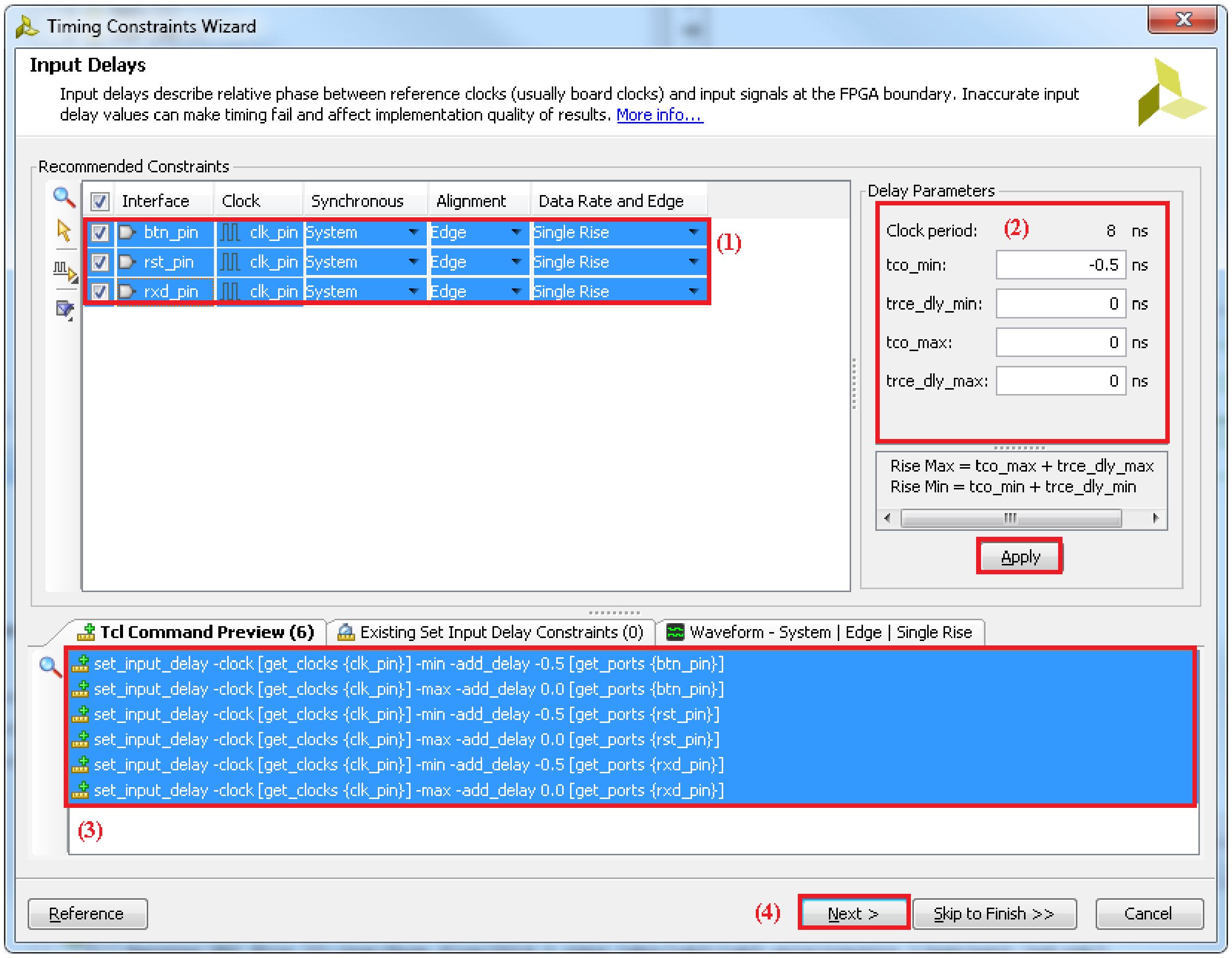
3-1-9. The wizard identifies Input Delays needed for the pins: btn_pin, rst_pin and rxd_pin. Do the following:

(1) Press Ctrl and select all three 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, 6 Tcl commands have been generated.

(4) Click **Next**.


Timing Constraints Wizard
Input Delays
Input delays describe relative phase between reference clocks (usually board clocks) and input signals at the FPGA boundary. Inaccurate input delay values can make timing fail and affect implementation quality of results. More info...
Recommended Constraints
Interface
Clock
Synchronous
Alignment
Data Rate and Edge
(1)
Delay Parameters
(2)
Rise Max = tco_max + trce_dly_max
Rise Min = tco_min + trce_dly_min
Apply
Tcl Command Preview (6)
Existing Set Input Delay Constraints (0)
Waveform - System | Edge | Single Rise
(3)
Reference
(4)
Next >
Skip to Finish >>
Cancel

Interface	Clock	Synchronous	Alignment	Data Rate and Edge
btn_pin	clk_pin	System	Edge	Single Rise
rst_pin	clk_pin	System	Edge	Single Rise
rxd_pin	clk_pin	System	Edge	Single Rise

Clock period:	8	ns
tco_min:	-0.5	ns
trce_dly_min:	0	ns
tco_max:	0	ns
trce_dly_max:	0	ns

```
set_input_delay -clock [get_clocks {clk_pin}] -min -add_delay -0.5 [get_ports {btn_pin}]
set_input_delay -clock [get_clocks {clk_pin}] -max -add_delay 0.0 [get_ports {btn_pin}]
set_input_delay -clock [get_clocks {clk_pin}] -min -add_delay -0.5 [get_ports {rst_pin}]
set_input_delay -clock [get_clocks {clk_pin}] -max -add_delay 0.0 [get_ports {rst_pin}]
set_input_delay -clock [get_clocks {clk_pin}] -min -add_delay -0.5 [get_ports {rxd_pin}]
set_input_delay -clock [get_clocks {clk_pin}] -max -add_delay 0.0 [get_ports {rxd_pin}]
```

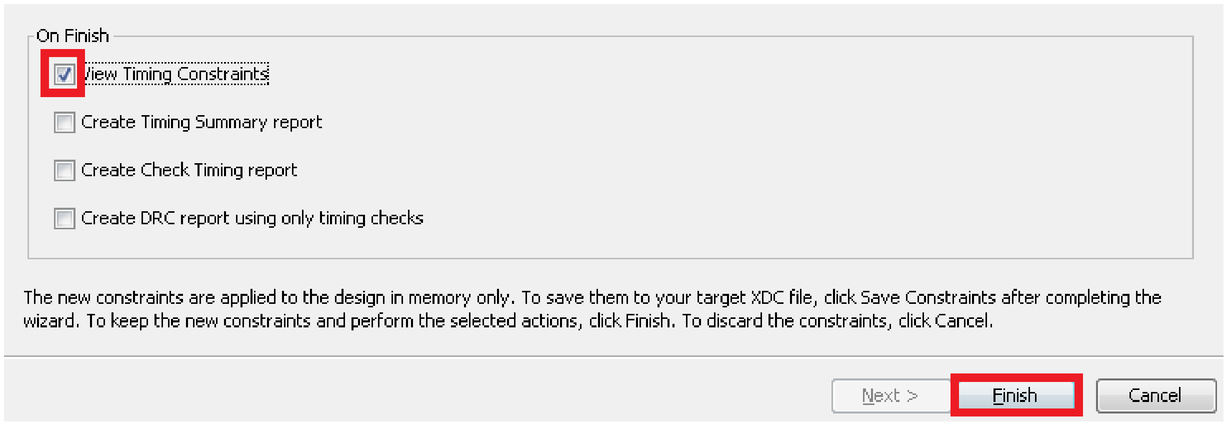
Figure 12. Specifying Input Delays for btn_pin, rst_pin and rxt_pin

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

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

3-1-12. Click **Next** for the next few screens until 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 XDC file to show you the generated timing constraint.


On Finish

[✓] View Timing Constraints

[ ] Create Timing Summary report

[ ] Create Check Timing report

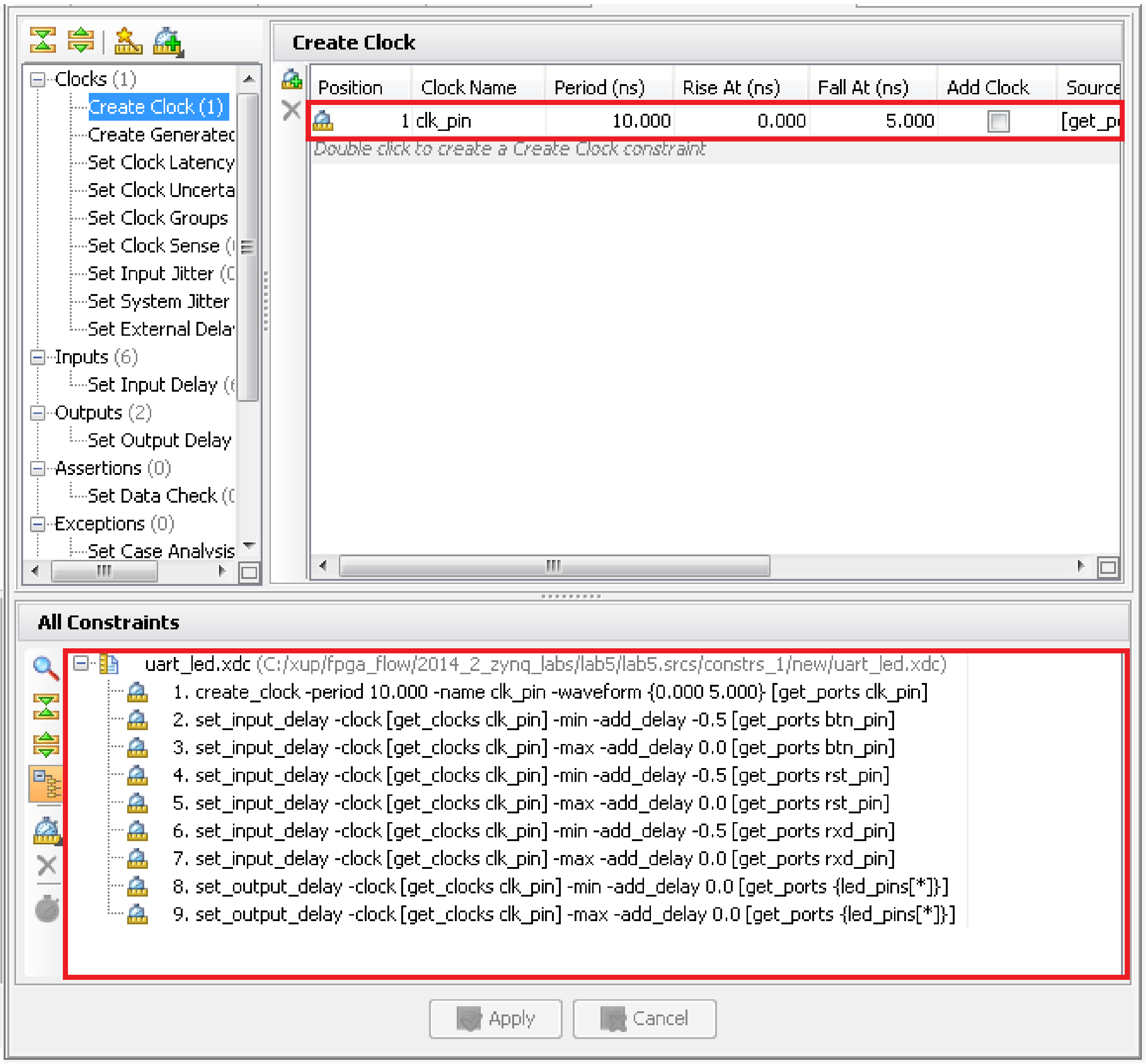
[ ] Create DRC report using only timing checks

The new constraints are applied to the design in memory only. To save them to your target XDC file, click Save Constraints after completing the wizard. To keep the new constraints and perform the selected actions, click Finish. To discard the constraints, click Cancel.

Figure 13, selecting View Timing constraints

3-1-14. Note the wizard generated the clk_pin constraint for a 10 ns period (or 100 MHz) for the ZedBoard or 8 ns (125 MHz) for the Zybo. Notice in the All Constraints window, 9 constraints will be created.

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

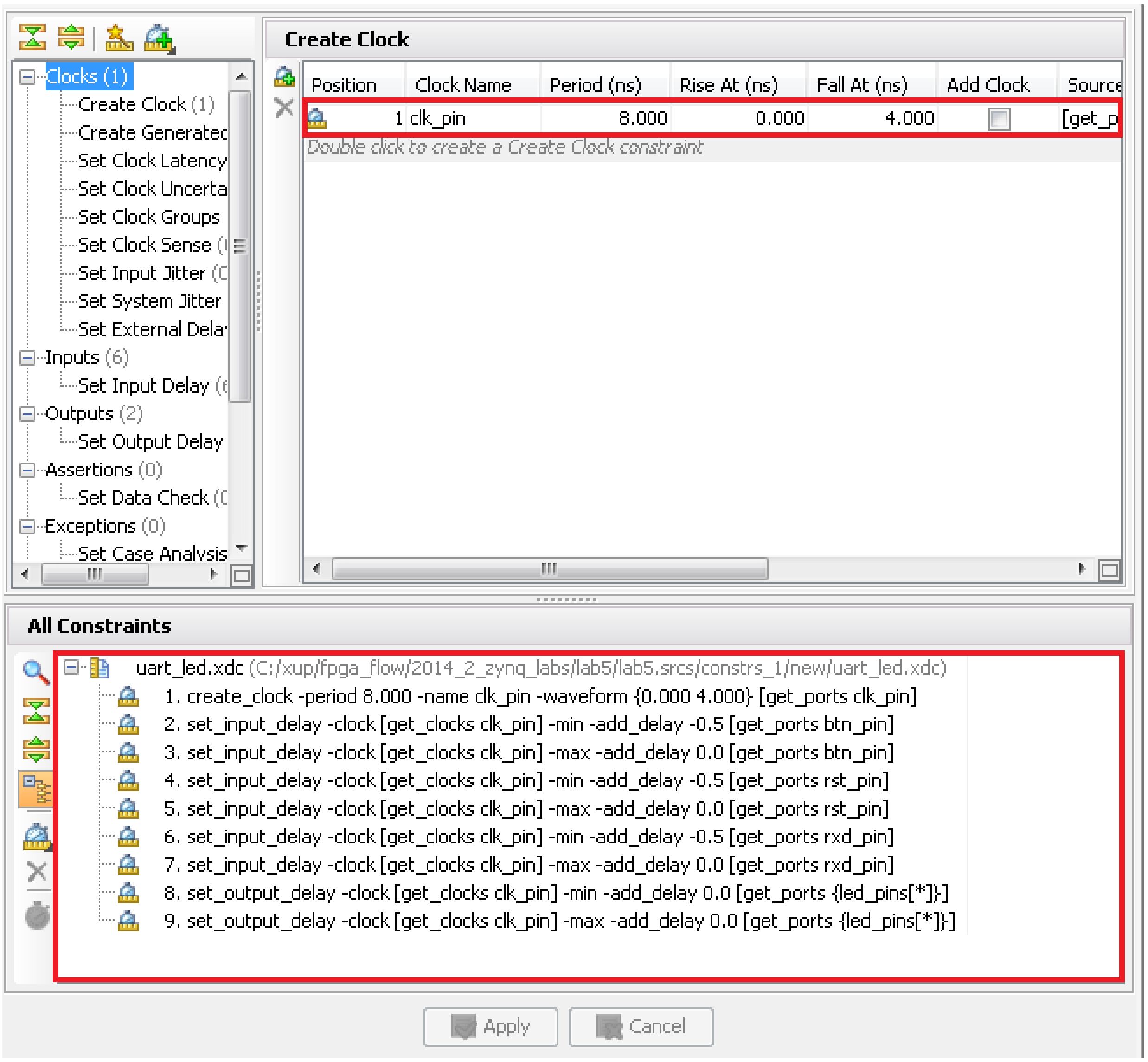


Position	Clock Name	Period (ns)	Rise At (ns)	Fall At (ns)	Add Clock	Source
1	clk_pin	10.000	0.000	5.000	☐	[get_p

uart_led.xdc (C:/xup/fpga_flow/2014_2_zynq_labs/lab5/lab5.srcs/constrs_1/new/uart_led.xdc)

```
1. create_clock -period 10.000 -name clk_pin -waveform {0.000 5.000} [get_ports clk_pin]
2. set_input_delay -clock [get_clocks clk_pin] -min -add_delay -0.5 [get_ports btn_pin]
3. set_input_delay -clock [get_clocks clk_pin] -max -add_delay 0.0 [get_ports btn_pin]
4. set_input_delay -clock [get_clocks clk_pin] -min -add_delay -0.5 [get_ports rst_pin]
5. set_input_delay -clock [get_clocks clk_pin] -max -add_delay 0.0 [get_ports rst_pin]
6. set_input_delay -clock [get_clocks clk_pin] -min -add_delay -0.5 [get_ports rxd_pin]
7. set_input_delay -clock [get_clocks clk_pin] -max -add_delay 0.0 [get_ports rxd_pin]
8. set_output_delay -clock [get_clocks clk_pin] -min -add_delay 0.0 [get_ports {led_pins[*]}]
9. set_output_delay -clock [get_clocks clk_pin] -max -add_delay 0.0 [get_ports {led_pins[*]}]
```

Figure 14. The constraints added for the ZedBoard



Position	Clock Name	Period (ns)	Rise At (ns)	Fall At (ns)	Add Clock	Source
1	clk_pin	8.000	0.000	4.000	☐	[get_p

```
uart_led.xdc (C:/xup/fpga_flow/2014_2_zynq_labs/lab5/lab5.srcs/constrs_1/new/uart_led.xdc)
1. create_clock -period 8.000 -name clk_pin -waveform {0.000 4.000} [get_ports clk_pin]
2. set_input_delay -clock [get_clocks clk_pin] -min -add_delay -0.5 [get_ports btn_pin]
3. set_input_delay -clock [get_clocks clk_pin] -max -add_delay 0.0 [get_ports btn_pin]
4. set_input_delay -clock [get_clocks clk_pin] -min -add_delay -0.5 [get_ports rst_pin]
5. set_input_delay -clock [get_clocks clk_pin] -max -add_delay 0.0 [get_ports rst_pin]
6. set_input_delay -clock [get_clocks clk_pin] -min -add_delay -0.5 [get_ports rxd_pin]
7. set_input_delay -clock [get_clocks clk_pin] -max -add_delay 0.0 [get_ports rxd_pin]
8. set_output_delay -clock [get_clocks clk_pin] -min -add_delay 0.0 [get_ports {led_pins[*]}]
9. set_output_delay -clock [get_clocks clk_pin] -max -add_delay 0.0 [get_ports {led_pins[*]}]
```

Figure 14. The constraints added for the Zybo

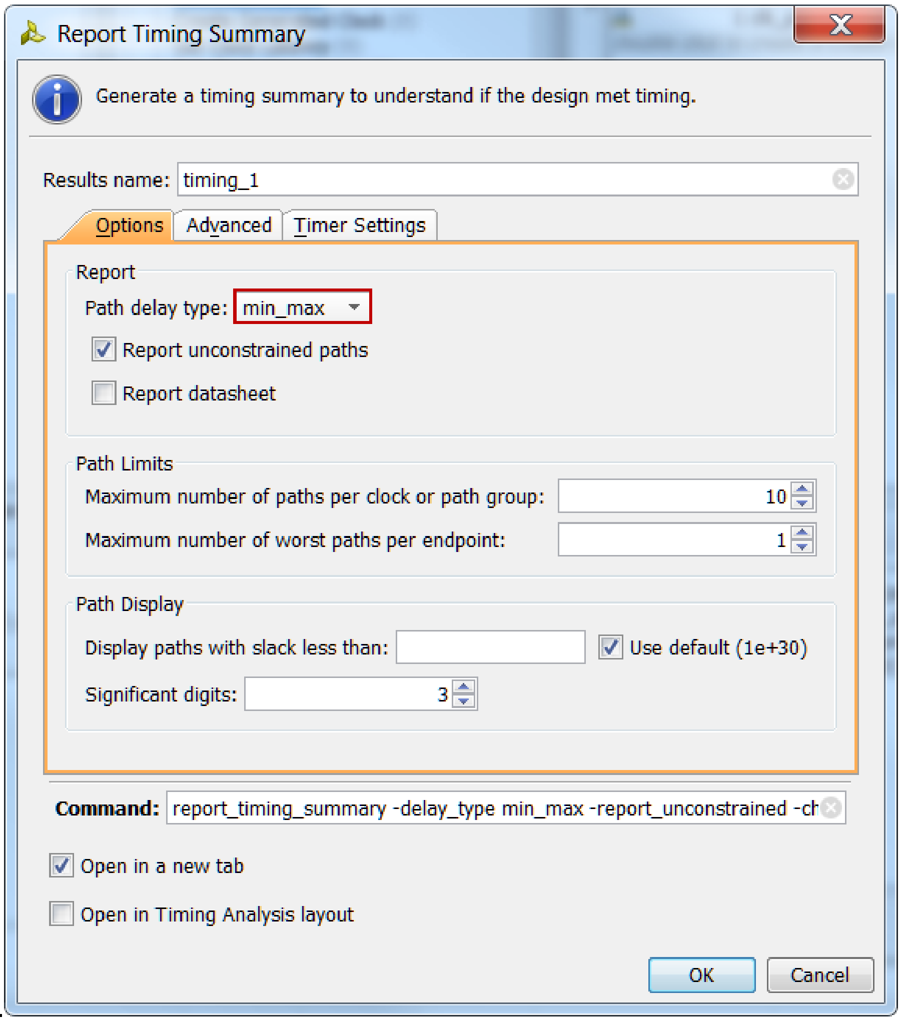
3-1-15. Select **File > Save Constraints** to save the changes to the target XDC. Click **Ok** at the warning about synthesis going out of date and then click **Yes** to apply the constraints.

3-1-16. Open uart_led.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.



Report Timing Summary

Generate a timing summary to understand if the design met timing.

Results name: timing_1

Options | Advanced | Timer Settings

Report

Path delay type: min_max

[✓] Report unconstrained paths

[ ] Report datasheet

Path Limits

Maximum number of paths per clock or path group: 10

Maximum number of worst paths per endpoint: 1

Path Display

Display paths with slack less than: [✓] Use default (1e+30)

Significant digits: 3

Command: report_timing_summary -delay_type min_max -report_unconstrained -ch

[✓] Open in a new tab

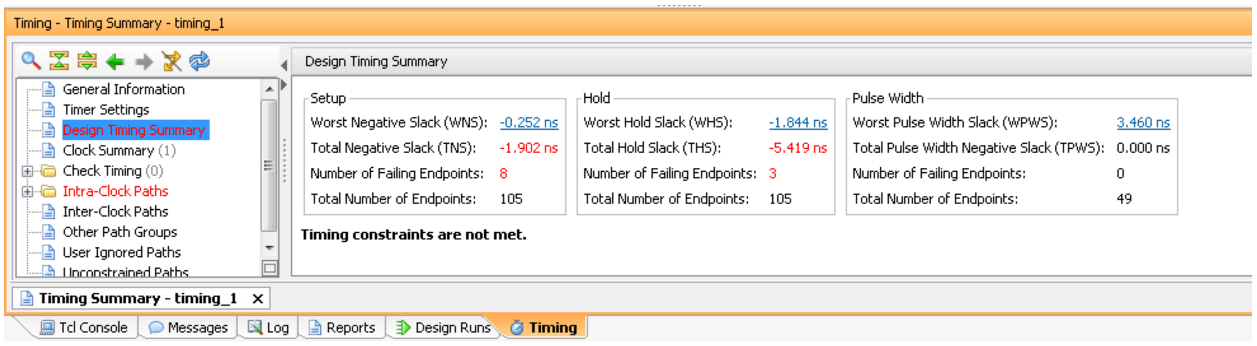
[ ] Open in Timing Analysis layout

OK Cancel

Figure 15. Performing timing analysis

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

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



Timing - Timing Summary - timing_1

- General Information
- Timer Settings
- Design Timing Summary
- Clock Summary (1)
- Check Timing (0)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths

Design Timing Summary

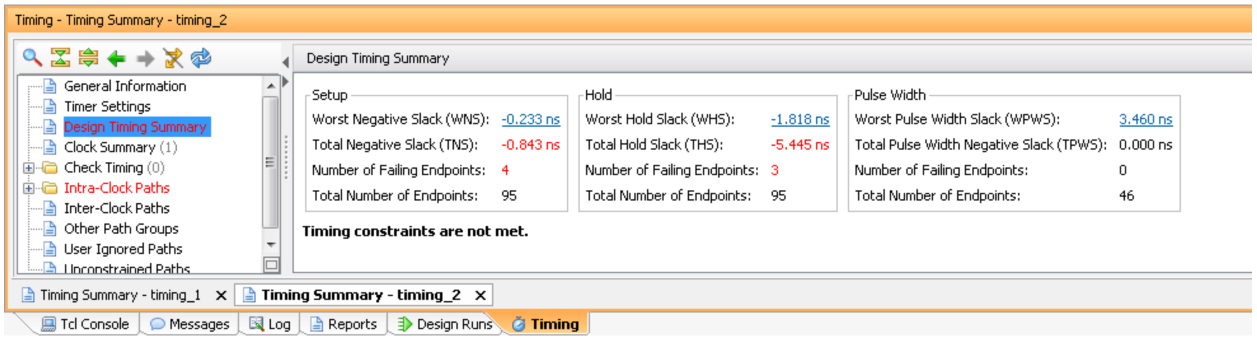
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-0.252 ns	Worst Hold Slack (WHS):	-1.844 ns	Worst Pulse Width Slack (WPWS):	3.460 ns
Total Negative Slack (TNS):	-1.902 ns	Total Hold Slack (THS):	-5.419 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	8	Number of Failing Endpoints:	3	Number of Failing Endpoints:	0
Total Number of Endpoints:	105	Total Number of Endpoints:	105	Total Number of Endpoints:	49

Timing constraints are not met.

Timing Summary - timing_1

Tcl Console | Messages | Log | Reports | Design Runs | Timing

Figure 16. Timing summary for the ZedBoard



Timing - Timing Summary - timing_2

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (0)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-0.233 ns	Worst Hold Slack (WHS):	-1.818 ns	Worst Pulse Width Slack (WPWS):	3.460 ns
Total Negative Slack (TNS):	-0.843 ns	Total Hold Slack (THS):	-5.445 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	4	Number of Failing Endpoints:	3	Number of Failing Endpoints:	0
Total Number of Endpoints:	95	Total Number of Endpoints:	95	Total Number of Endpoints:	46

Timing constraints are not met.

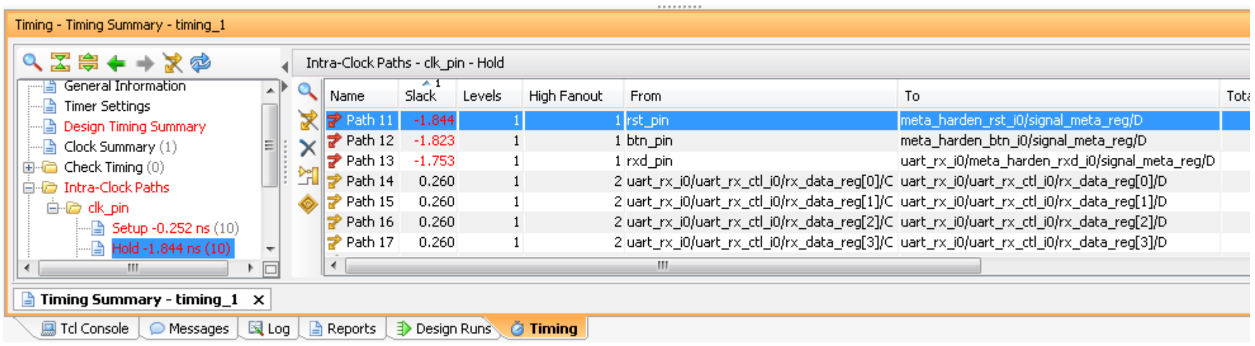
Timing Summary - timing_1 ✕ | Timing Summary - timing_2 ✕

Figure 16. 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.

For the ZedBoard that there are 8 failures under the setup check and 3 timing failures under the hold check. For the Zybo, there are 4 failures under the setup check and 3 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.



Timing - Timing Summary - timing_1

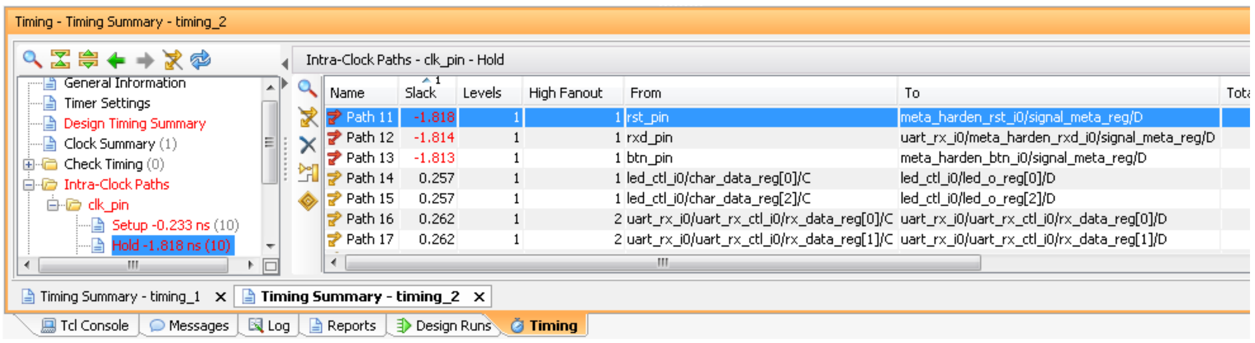
General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (0)
Intra-Clock Paths
clk_pin
Setup -0.252 ns (10)
Hold -1.844 ns (10)

Intra-Clock Paths - clk_pin - Hold

Name	Slack	Levels	High Fanout	From	To	Tota
Path 11	-1.844	1	1	rst_pin	meta_harden_rst_i0/signal_meta_reg/D	
Path 12	-1.823	1	1	btn_pin	meta_harden_btn_i0/signal_meta_reg/D	
Path 13	-1.753	1	1	rxd_pin	uart_rx_i0/meta_harden_rxd_i0/signal_meta_reg/D	
Path 14	0.260	1	2	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[0]/C	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[0]/D	
Path 15	0.260	1	2	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[1]/C	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[1]/D	
Path 16	0.260	1	2	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[2]/C	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[2]/D	
Path 17	0.260	1	2	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[3]/C	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[3]/D	

Timing Summary - timing_1 ✕

Figure 17. The list of paths showing hold violations for the ZedBoard



Timing - Timing Summary - timing_2

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (0)
Intra-Clock Paths
clk_pin
Setup -0.233 ns (10)
Hold -1.818 ns (10)

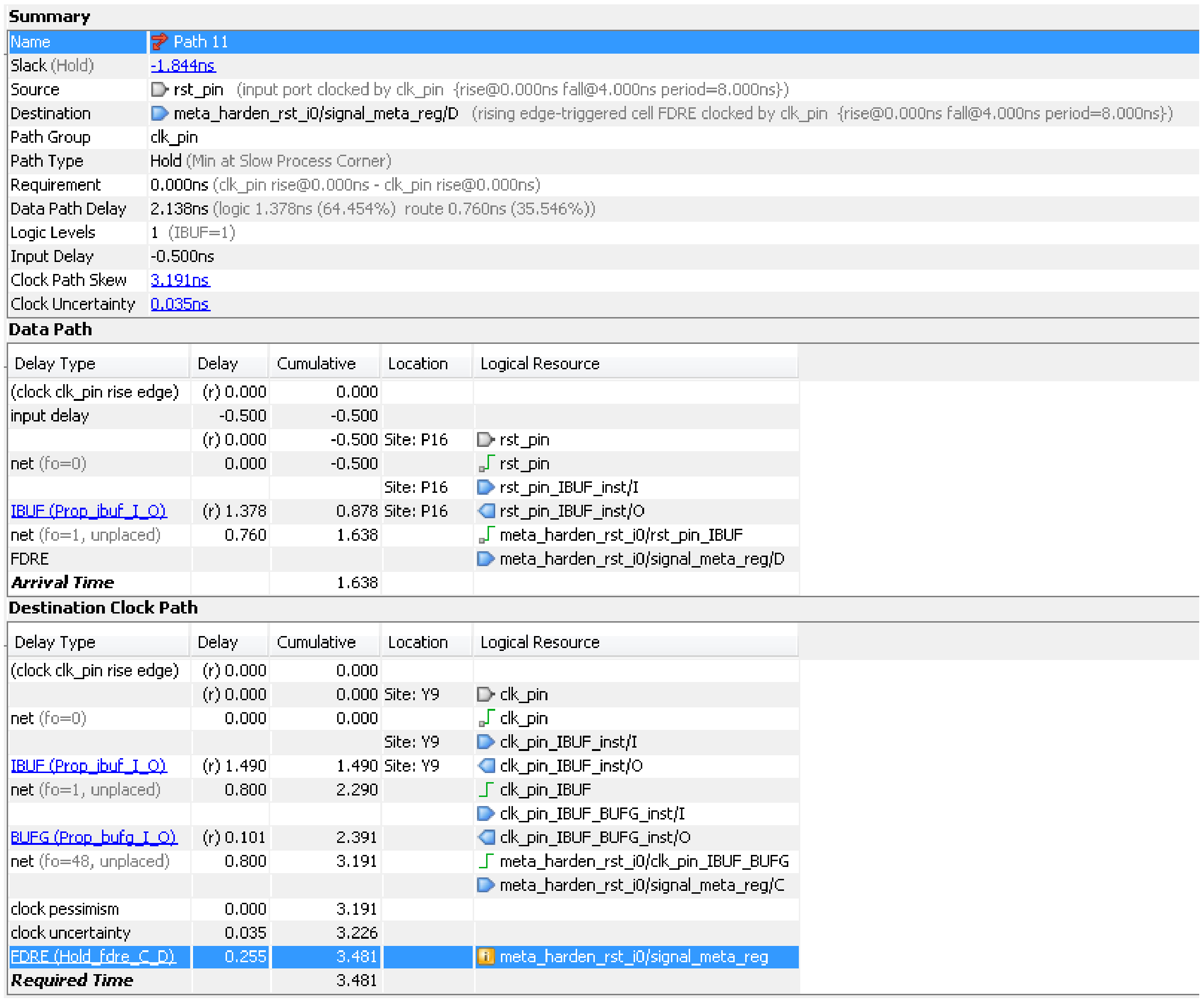
Intra-Clock Paths - clk_pin - Hold

Name	Slack	Levels	High Fanout	From	To	Tota
Path 11	-1.818	1	1	rst_pin	meta_harden_rst_i0/signal_meta_reg/D	
Path 12	-1.814	1	1	rxd_pin	uart_rx_i0/meta_harden_rxd_i0/signal_meta_reg/D	
Path 13	-1.813	1	1	btn_pin	meta_harden_btn_i0/signal_meta_reg/D	
Path 14	0.257	1	1	led_ctl_i0/char_data_reg[0]/C	led_ctl_i0/led_o_reg[0]/D	
Path 15	0.257	1	1	led_ctl_i0/char_data_reg[2]/C	led_ctl_i0/led_o_reg[2]/D	
Path 16	0.262	1	2	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[0]/C	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[0]/D	
Path 17	0.262	1	2	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[1]/C	uart_rx_i0/uart_rx_ctl_i0/rx_data_reg[1]/D	

Timing Summary - timing_1 ✕ | Timing Summary - timing_2 ✕

Figure 17. 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)	-1.844ns
Source	rst_pin (input port clocked by clk_pin {rise@0.000ns fall@4.000ns period=8.000ns})
Destination	meta_harden_rst_i0/signal_meta_reg/D (rising edge-triggered cell FDRE clocked by clk_pin {rise@0.000ns fall@4.000ns period=8.000ns})
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	3.191ns
Clock Uncertainty	0.035ns

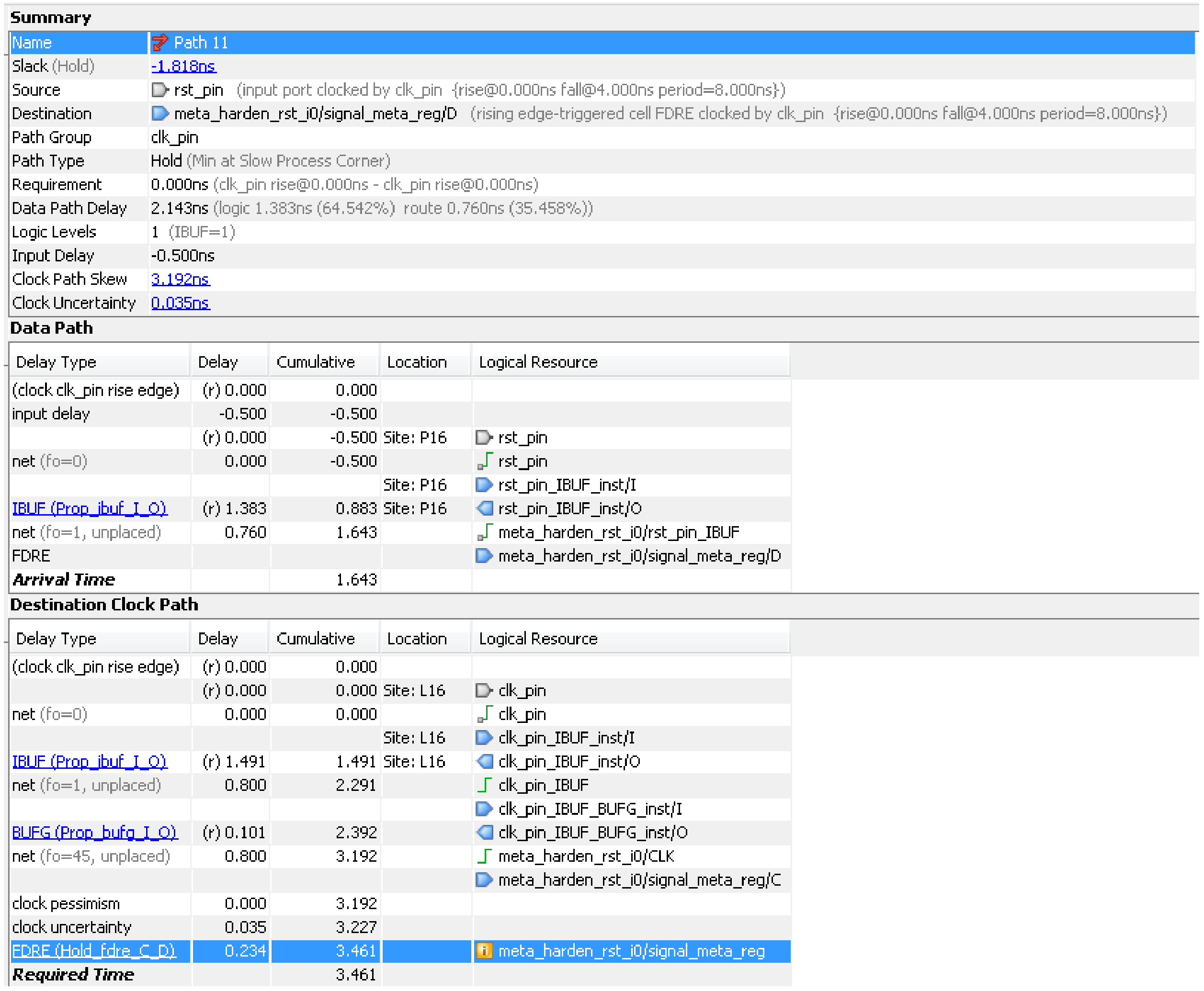
Data Path

Delay Type	Delay	Cumulative	Location	Logical Resource
(clock clk_pin rise edge)	(r) 0.000	0.000		
input delay	-0.500	-0.500		
	(r) 0.000	-0.500	Site: P16	rst_pin
net (fo=0)	0.000	-0.500		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		meta_harden_rst_i0/rst_pin_IBUF
FDRE				meta_harden_rst_i0/signal_meta_reg/D
Arrival Time		1.638		

Destination Clock Path

Delay Type	Delay	Cumulative	Location	Logical Resource
(clock clk_pin rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: Y9	clk_pin
net (fo=0)	0.000	0.000		clk_pin
			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
				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.101	2.391		clk_pin_IBUF_BUFG_inst/O
net (fo=48, unplaced)	0.800	3.191		meta_harden_rst_i0/clk_pin_IBUF_BUFG
				meta_harden_rst_i0/signal_meta_reg/C
clock pessimism	0.000	3.191		
clock uncertainty	0.035	3.226		
FDRE (Hold_fdre_C_D)	0.255	3.481		meta_harden_rst_i0/signal_meta_reg
Required Time		3.481		

Figure 18. Failing hold path 11 for the ZedBoard



Summary

Name	Path 11
Slack (Hold)	-1.818ns
Source	rst_pin (input port clocked by clk_pin {rise@0.000ns fall@4.000ns period=8.000ns})
Destination	meta_harden_rst_i0/signal_meta_reg/D (rising edge-triggered cell FDRE clocked by clk_pin {rise@0.000ns fall@4.000ns period=8.000ns})
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	2.143ns (logic 1.383ns (64.542%) route 0.760ns (35.458%))
Logic Levels	1 (IBUF=1)
Input Delay	-0.500ns
Clock Path Skew	3.192ns
Clock Uncertainty	0.035ns

Data Path

Delay Type	Delay	Cumulative	Location	Logical Resource
(clock clk_pin rise edge)	(r) 0.000	0.000		
input delay	-0.500	-0.500		
	(r) 0.000	-0.500	Site: P16	rst_pin
net (fo=0)	0.000	-0.500		rst_pin
			Site: P16	rst_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.383	0.883	Site: P16	rst_pin_IBUF_inst/O
net (fo=1, unplaced)	0.760	1.643		meta_harden_rst_i0/rst_pin_IBUF
FDRE				meta_harden_rst_i0/signal_meta_reg/D
Arrival Time		1.643		

Destination Clock Path

Delay Type	Delay	Cumulative	Location	Logical Resource
(clock clk_pin rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: L16	clk_pin
net (fo=0)	0.000	0.000		clk_pin
			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
				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.101	2.392		clk_pin_IBUF_BUFG_inst/O
net (fo=45, unplaced)	0.800	3.192		meta_harden_rst_i0/CLK
				meta_harden_rst_i0/signal_meta_reg/C
clock pessimism	0.000	3.192		
clock uncertainty	0.035	3.227		
FDRE (Hold_fdre_C_D)	0.234	3.461		meta_harden_rst_i0/signal_meta_reg
Required Time		3.461		

Figure 18. Failing hold path 11 for the Zybo

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

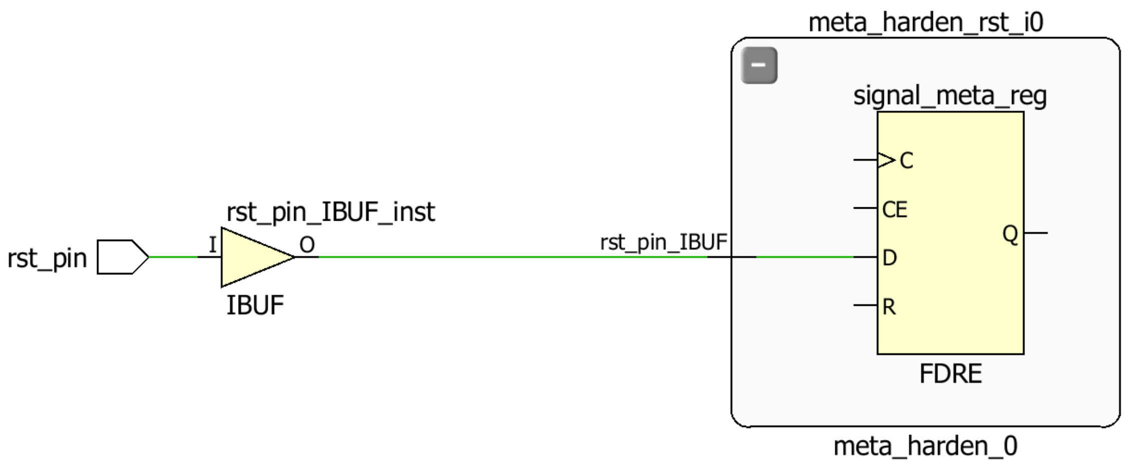


Figure 19. The schematic of the failing path

You can see that the failing path is at the input of rst_pin. Similarly, Path 12 and Path 13 are of btn_pin and rxd_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** and 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.

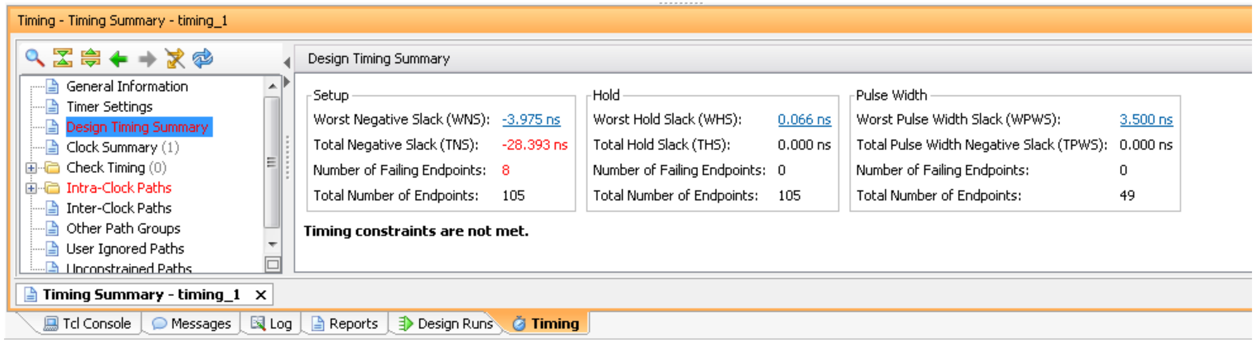
4-2. Generate a timing summary report.

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.

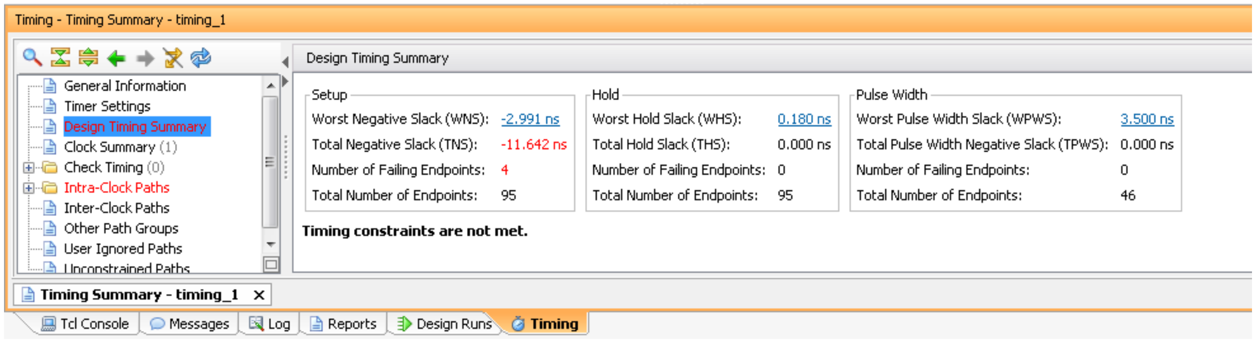


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-3.975 ns	Worst Hold Slack (WHS):	0.066 ns	Worst Pulse Width Slack (WPWS):	3.500 ns
Total Negative Slack (TNS):	-28.393 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	8	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	105	Total Number of Endpoints:	105	Total Number of Endpoints:	49

Timing constraints are not met.

Figure 20. Failing setup paths for the ZedBoard


Timing - Timing Summary - timing_1

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (0)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths

Design Timing Summary

Setup
Worst Negative Slack (WNS): -2.991 ns
Total Negative Slack (TNS): -11.642 ns
Number of Failing Endpoints: 4
Total Number of Endpoints: 95

Hold
Worst Hold Slack (WHS): 0.180 ns
Total Hold Slack (THS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 95

Pulse Width
Worst Pulse Width Slack (WPWS): 3.500 ns
Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 46

Timing constraints are not met.

Figure 20. Failing setup paths for the Zybo

4-2-3. Click on the WNS to see the failing paths. The output path delay can be reduced by placing the register in IOB.

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

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

4-2-5. Select **File > Save Constraints**. Click **OK** at the warning.

4-2-6. Click on **Run Implementation**.

4-2-7. Click **Yes** to reset the synthesis run, perform synthesis first and then run implementation.

4-2-8. Open the implemented design. Open the Timing Summary.

For the ZedBoard, 8 failing WNS endpoints are reported. For the Zybo, 4 failing WNS endpoints are still reported. However, these paths are related to the LEDs and will not impact the functionality of the design. As such, they can be ignored.

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. Plug-in the PmodUSBUart module into the top-row of the JA PMOD connector for the ZedBoard or the JE PMOD connector for 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.

5-2-1. Connect the micro-USB cable between the PmodUSBUart module and the host USB port.

5-2-2. Plug-in the PmodUSBUart module into the top-row of the JA PMOD connector. For the Zybo, plug the PmodUSBUart module into the top-row of the JE PMOD connector.

5-2-3. Make sure that the Micro-USB cable is connected to the JTAG PROG connector (next to the power supply connector). If using the ZedBoard, connect the power jack.

Turn **ON** the power.

5-2-4. Select the *Open Hardware Manager* option and click **OK**.

The Hardware Manager window will open indicating "unconnected" status.

5-2-5. 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. In this case skip to step 5-2-9.

5-2-6. Click **Next** to see the Vivado Hardware Server Settings form.

5-2-7. Click **Next** with the frequency of 15000000 selected.

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

5-2-8. Click **Next** and then **Finish**.

5-2-9. 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.

5-2-10. Select the device and verify that the **uart_led.bit** is selected as the programming file in the General tab. It may also appear as ios.bit in case you forgot to change the module name to uart_led earlier.

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 **Program 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.

5-3-6. Verify the functionality as you did in the previous lab, by typing some characters into the terminal. For the ZedBoard, all eight bits of will be displayed on the LEDs, press BTNU to swap the position of the four bits on the LEDs.

For the Zybo, only the lower four bits will be displayed, press BTN0 to display the upper four bits.

5-3-7. When satisfied, close the terminal emulator program and power OFF the board.

5-3-8. Select **File > Close Hardware Manager**. Click **OK** to close it.

5-3-9. When done, close the **Vivado** 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.