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

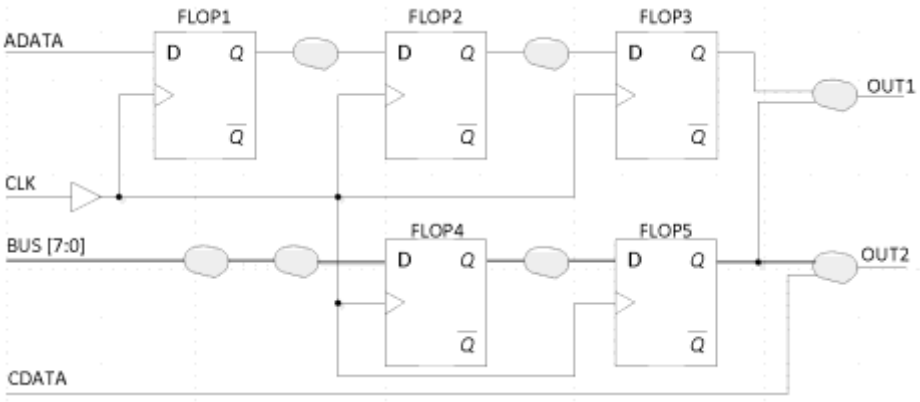
After completing this lab, you will be able to:

- 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

Timing Constraints

In combinatorial logic design, delays through the circuits will depend on the number of logic levels, the fan-out (number of gate inputs a net drives) on each net, and the capacitive loading on the output nets. When such circuits are placed between flip-flops or registers, they affect the clock speeds at which sequential designs can be operated. The synthesis and implementation tools will pack the design in LUT, flip-flops, and registers, as well as place them appropriately if the expected performance is communicated to them via timing constraints. Timing constraints can be categorized into global timing or path specific constraints. The path specific constraints have higher priority over global timing constraints, and the components which are used in those specific paths are placed and routed first.

The global timing constraints cover most of the design with very few lines of instructions. In any pure combinatorial design, the path-to-path constraint is used to describe the delay the circuit can tolerate. In sequential circuits, period, input delay, and output delay constraints are used. All four kinds of the timing constraints are shown in the figure below.



In the above figure, the paths which are covered between ADATA input and D port of FLOP1, BUS input and D port of FLOP4 can be constrained by a constraint called SET_INPUT_DELAY command. The set_input_delay command indicates how much time is spent between the Q output of a FF in the upstream device, the routing delay in the upstream device as well as the board delay. The tools will subtract that delay from the clock period of the clock signal listed in the command and will use the resulting delay to place and route the path between the input and the D input of FF. It will also consider delay experienced by the clock arriving to the clock port of the destination FF (e.g. FLOP1 in the above diagram). The max and min qualifiers are used for the setup and hold checks.

The paths between the port Q of FLOP3 and output OUT1, Q port of FLOP5 and OUT1, Q port of FLOP5 and OUT2 can be constrained by SET_OUTPUT_DELAY command. Again, the delay mentioned indicates how much delay is spent in the board delay, routing delay and the setup delay of the FF in the downstream device.

The paths between CDATA and OUT2 can be constrained by the SET_MAX_DELAY constraint.

The paths between Q port of FLOP1 and D port of FLOP2, Q port of FLOP2 and D port of FLOP3, Q port of FLOP4 and D port of FLOP5 can be constrained by the period constraint. The period constraint is created using the create_clock command. The create_clock command may refer ot a pin of the FPGA design or may not refer any pins. When the clock pin is not referred, a virtual clock will be created. When the pin is referred, the period parameter indicates rising to rising edge delay and waveform option indicates when the rising edge occurs and the second number indicates when the falling edge occurs. The waveform option can be used to create clocks of non-50% duty cycle and/or phase delayed clock signal.

```
create_clock -name CLK -period 10.0 -waveform (0 5.0) [get_ports CLK]
set_input_delay -clock CLK -max 3.0 [all_inputs]
set_input_delay -clock CLK -min 1.0 [all_inputs]
set_output_delay -clock CLK 2.0 [all_outputs]
set_max_delay 5.0 -from [get_ports CDATA] -to [get_ports OUT2]
```

Note that the clock period is defined at 10 ns. This is applied throughout the example for consistency. Further details on the syntax of each constraint type can be found in UG903, the Vivado Using Constraints Guide.

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**.

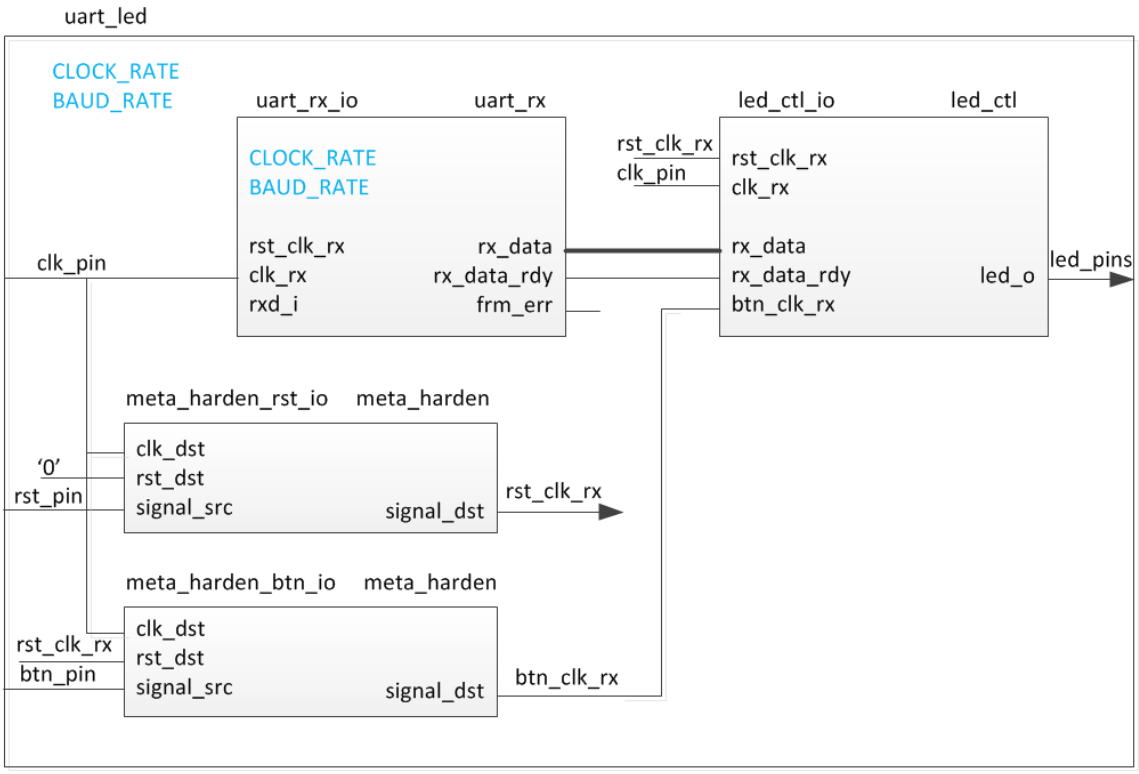
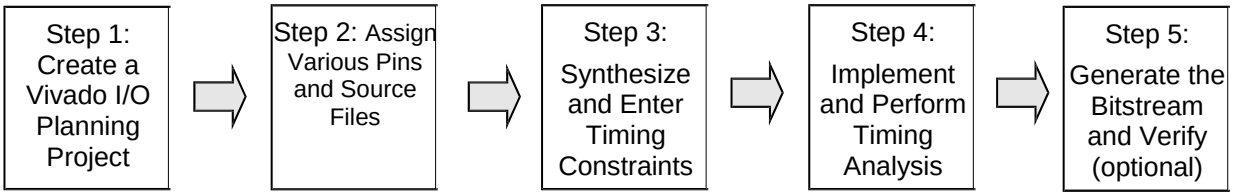


Figure 1. The design

General Flow



In the instructions below;
{***sources***} refers to: …\2016_2_artix7_sources
{***labs***} refers to : …\2016_2_artix7_labs

Board support for the Basys3, Nexys4 DDR, Nexys Video is not included in Vivado 2016.2 by default. The relevant zip files need to be extracted and saved to: {Vivado installation}\data\boards\board_files\.

These files can be downloaded either from the Digilent, Inc. webpage (https://reference.digilentinc.com/vivado/boardfiles2015) or the XUP webpage (http://www.xilinx.com/support/university/vivado/vivado-workshops/Vivado-fpga-design-flow.html) where this material is also hosted.

Create a Vivado I/O Planning Project — Step 1

1-1. Launch Vivado and create an I/O Planning project targeting XC7A35TCPG236-1 (Basys3), XC7A100TCSG324-1 (Nexys4 DDR), or XC7A200TSBG484-1 (Nexys Video).

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 **{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 **XC7A100TCSG324-1** (for the Nexys4 DDR), the **XC7A35TCPG236-1** (for the Basys3), or **XC7A200tsbg484-1** (for the Nexys Video).

Using the **Boards** option, you may also select the **Nexys4 DDR**, the **Basys3**, or the **Nexys Video** depending on your board.

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.

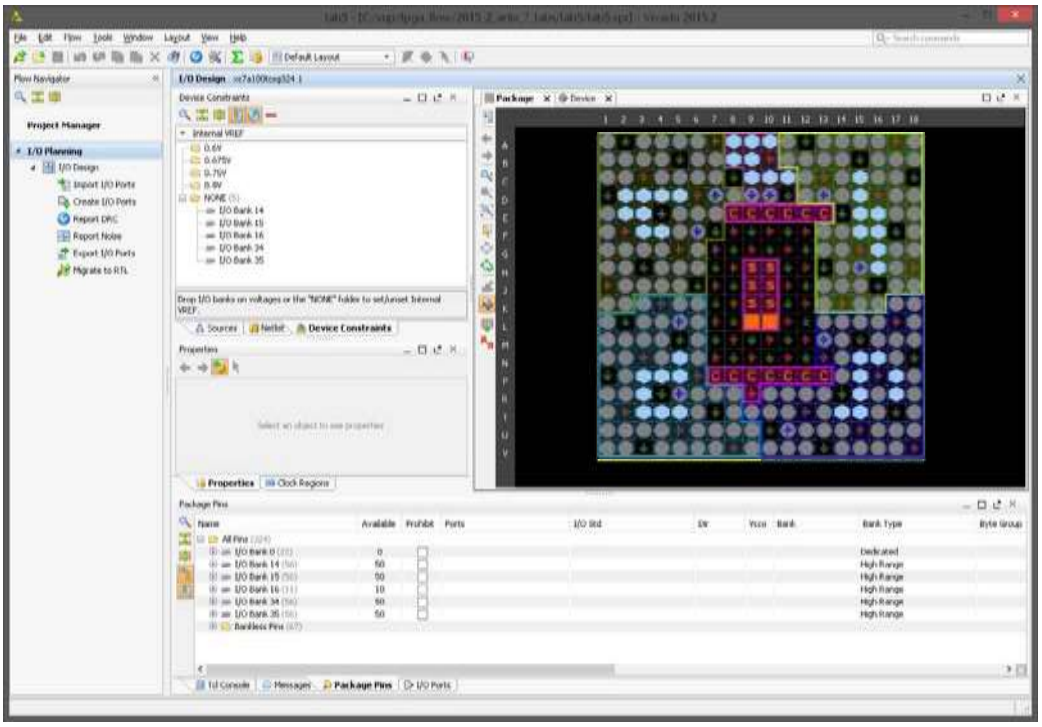


Figure 2. I/O Planning project's default windows and views for the Nexys4 DDR

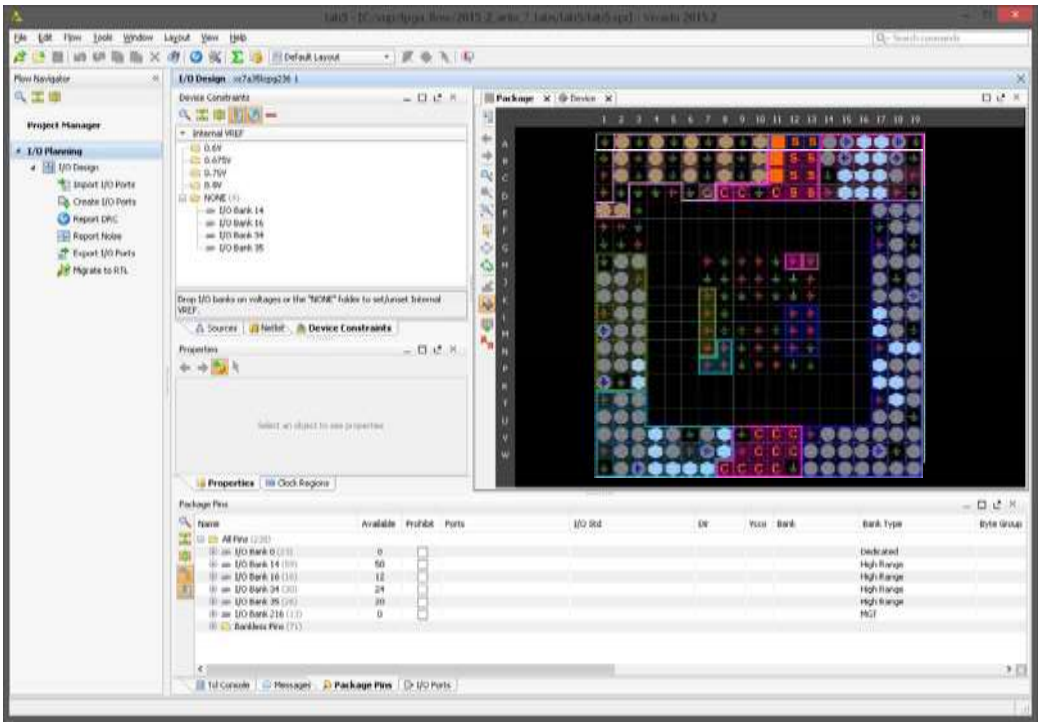


Figure 2. I/O Planning project's default windows and views for the Basys3

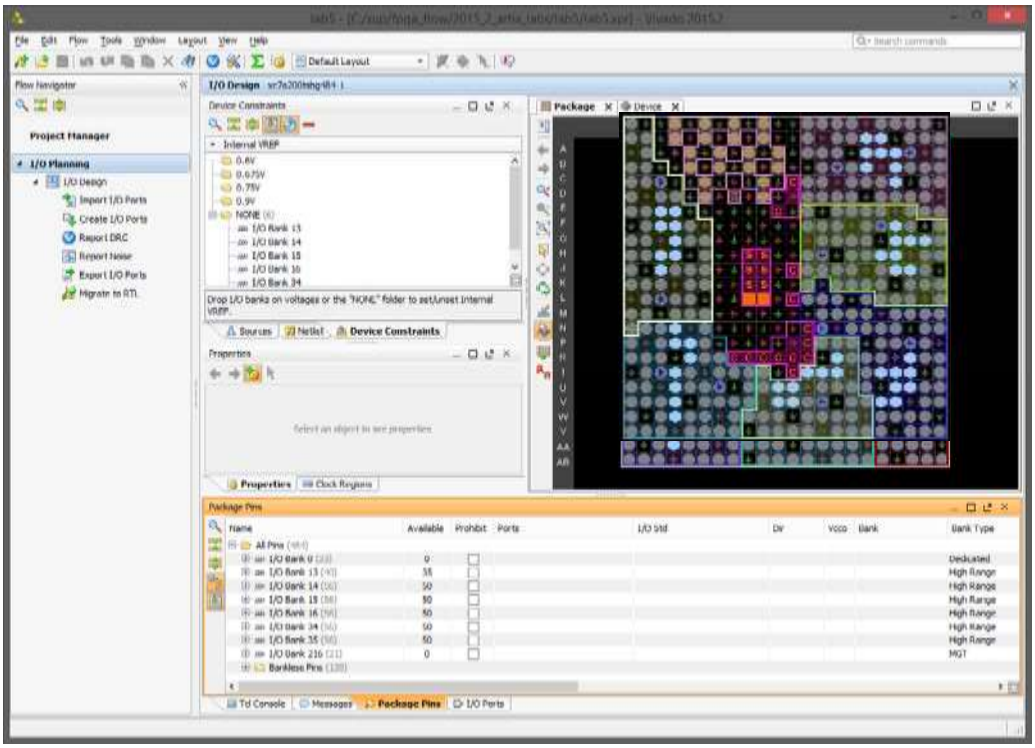


Figure 2. I/O Planning project's default windows and views for the Nexys Video

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

2-1. Create input ports clk_pin, btn_pin, rxd_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 a Port* form will be displayed.

2-1-2. Type **clk_pin** in the *Name* field, select **Input** for the *Direction* and select **LVCMOS33** 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**, **rxd_pin** and **rst_pin** input ports**.**

If using the Nexys Video, *btn_pin* must have *I/O STANDARD* **LVCMOS12**, *rxd_pin* **LVCMOS33**, and *rst_pin* **LVCMOS12**.

2-2. For the Nexys4 DDR, assign input pins clk_pin, btn_pin, rxd_pin, and rst_pin to E3, M18, C4, and N17 locations using the Device view and package pins.

For the Basys3, assign input pins clk_pin, btn_pin, rxd_pin, and rst_pin to W5, T18, B18, and U18 locations using the Device view and package pins.

For the Nexys Video, assign input pins clk_pin, btn_pin, rxd_pin, and rst_pin to R4, F15, V18, and B22 locations using the Device view and package pins.

2-2-1. For the Nexys4 DDR, move the mouse over the Device view window and hover over it on the **E3** location.

For the Basys3, hover the mouse over **W5** in the Device view window.

For the Nexys Video, hover the mouse over **R4** in the Device view window.

Figure 4. Locating E3 pin in the Device view for the Nexys4 DDR

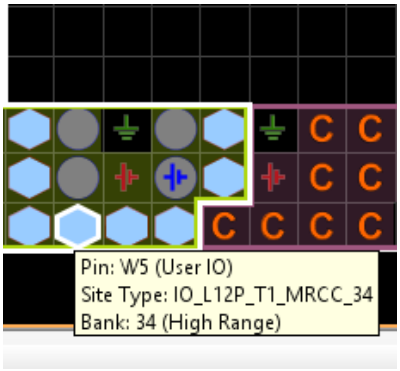


Figure 4. Locating W5 pin in the Device view for the Basys3

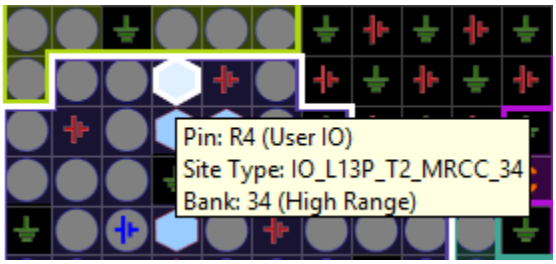


Figure 4. Locating R4 pin in the Device view for the Nexys Video

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

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

2-2-3. In the *Package* Pins pane, click in the Ports column of **E3** (Nexys4 DDR), **W5** (Basys3), or **R4** (Nexys Video) pin's row, and select **clk_pin.**

2-2-4. Similarly, add the **btn_pin** input port at **M18** (Nexys4 DDR), **T18** (Basys3), or **F15** (Nexys Video).

2-2-5. Select **Edit > Find** or Ctrl-F to open the Find form. Select **Package Pins** in the *Find* drop-down field, type ***C4** (for the Nexys4 DDR), ***B18** (for the Basys3), or ***V18** (for the Nexys Video) in the match criteria field, and click on **OK**.

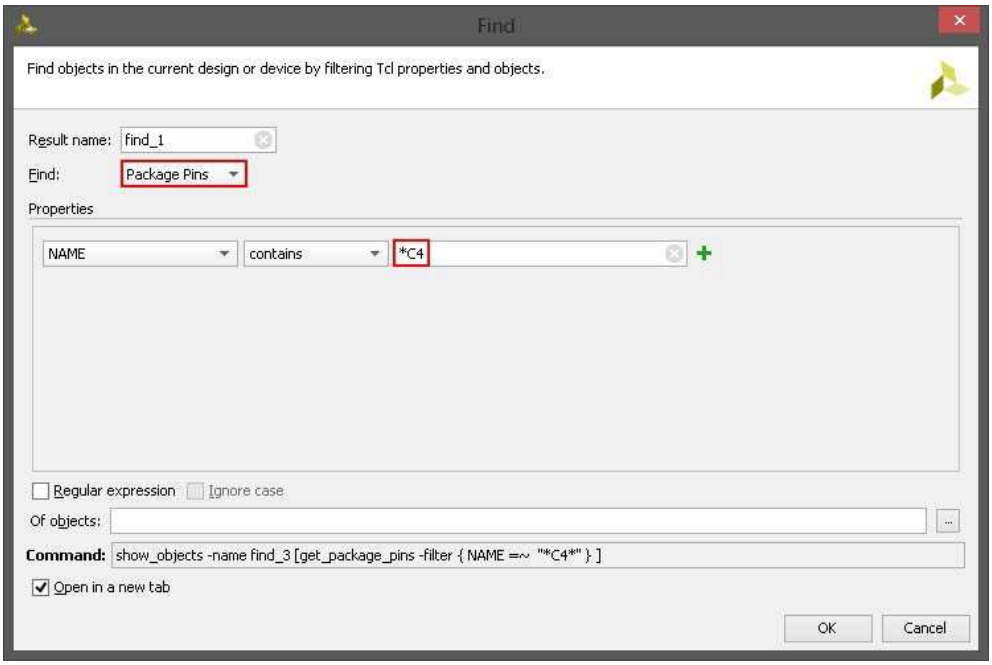


Figure 5. Finding a package pin for the Nexys4 DDR (use B18 for the Basys3 or V18 for the Nexys Video)

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

2-2-6. Assign the **rxd_pin** input to the pin.

2-2-7. Similarly add the **rst_pin** input by assigning it to the **N17** location for the Nexys4 DDR. For the Basys3, assign it to **U18** and **B22** for the Nexys Video.

2-3. For the Nexys4 DDR, assign output pins led_pins[7] to led_pins[0] to locations U16, U17, V17, R18, N14, J13, K15, and H17. Create them as a vector and assign them using the Tcl command set_property. They all will be LVCMOS33.

For the Basys3, assign output pins led_pins[7] to led_pins[0] to locations V14, U14, U15, W18, V19, U19, E19 and U16. Create them as a vector and assign them using the Tcl command set_property. All the pins will be LVCMOS33.

For the Nexys Video, assign output pins led_pins[7] to led_pins[0] to locations T14, T15, T16, U16, V15, W16, W15, Y13. Create them as a vector and assign them using the Tcl command set_property. All the pins will be LVCMOS25.

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

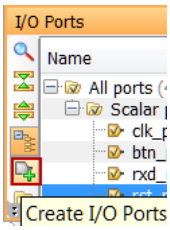


Figure 6. Create I/O Ports button

The *Create a Port* 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 **LVCMOS33** I/O standard if you are using a Nexys4 DDR or Basys3, or **LVCMOS25** if you are using the Nexys Video and click **OK**.

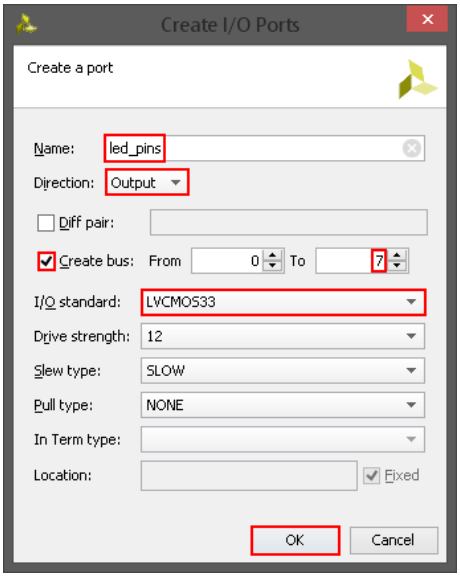


Figure 8. Creating I/O ports for the led_pins output

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 Nexys4 DDR:

```
set_property PACKAGE_PIN U16 [get_ports led_pins[7]]
set_property PACKAGE_PIN U17 [get_ports led_pins[6]]
set_property PACKAGE_PIN V17 [get_ports led_pins[5]]
set_property PACKAGE_PIN R18 [get_ports led_pins[4]]
set_property PACKAGE_PIN N14 [get_ports led_pins[3]]
set_property PACKAGE_PIN J13 [get_ports led_pins[2]]
set_property PACKAGE_PIN K15 [get_ports led_pins[1]]
set_property PACKAGE_PIN H17 [get_ports led_pins[0]]
```

For the Basys3:

```
set_property PACKAGE_PIN V14 [get_ports led_pins[7]]
set_property PACKAGE_PIN U14 [get_ports led_pins[6]]
set_property PACKAGE_PIN U15 [get_ports led_pins[5]]
set_property PACKAGE_PIN W18 [get_ports led_pins[4]]
```

```
set_property PACKAGE_PIN V19 [get_ports led_pins[3]]
set_property PACKAGE_PIN U19 [get_ports led_pins[2]]
set_property PACKAGE_PIN E19 [get_ports led_pins[1]]
set_property PACKAGE_PIN U16 [get_ports led_pins[0]]
```

For the Nexys Video:

```
set_property PACKAGE_PIN T14 [get_ports led_pins[7]]
set_property PACKAGE_PIN T15 [get_ports led_pins[6]]
set_property PACKAGE_PIN U15 [get_ports led_pins[5]]
set_property PACKAGE_PIN T16 [get_ports led_pins[4]]
set_property PACKAGE_PIN V15 [get_ports led_pins[3]]
set_property PACKAGE_PIN W16 [get_ports led_pins[2]]
set_property PACKAGE_PIN W15 [get_ports led_pins[1]]
set_property PACKAGE_PIN Y13 [get_ports led_pins[0]]
```

2-3-4. Select **File > Save Constraints**.

The Save Constraints form will be displayed.

2-3-5. Enter **uart_led** in the *File name* field, and click **OK**.

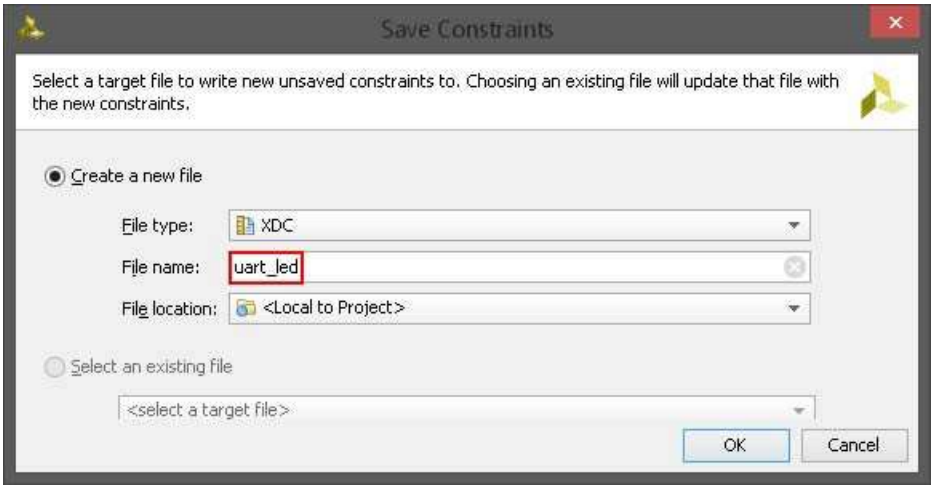


Figure 8. Saving constraints

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

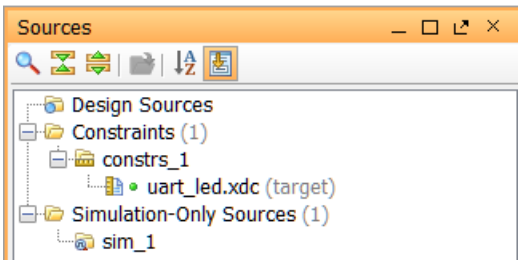


Figure 9. The uart_led.xdc file added to the source tree

2-3-6. Expand the **I/O Planning > I/O Design** in the *Flow Navigator* pane.

2-3-7. Click on **Report DRC** and click **OK**. Notice the design rules checker is run and one warning is reported. Ignore the warning.

2-3-8. 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.

2-3-9. Click on **Migrate to RTL.**

The Migrate to RTL form will be displayed with Top RTL file field showing *{labs}/lab5/io_1.v* entry.

2-3-10. Change *io_1.v* to **uart_led.v**, and click **OK**

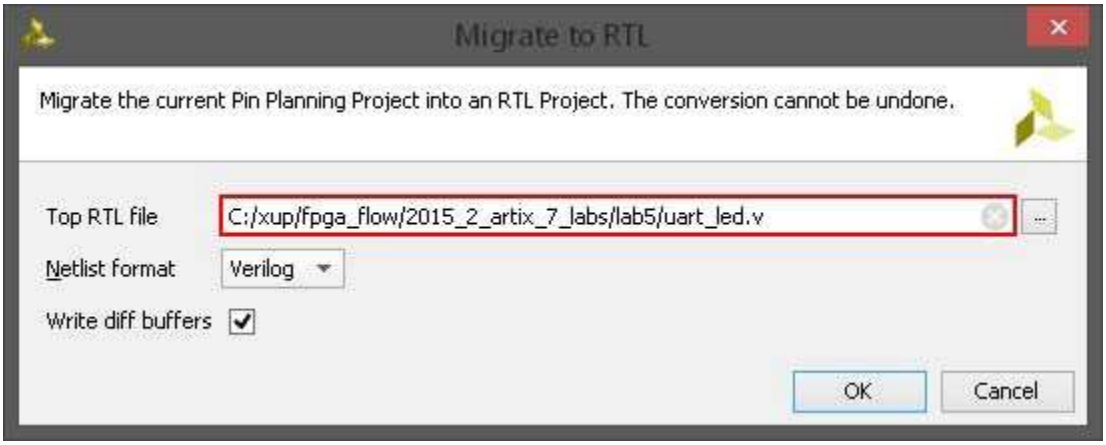


Figure 10. Assigning top-level file name

2-3-11. Select the **Hierarchy** tab 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.

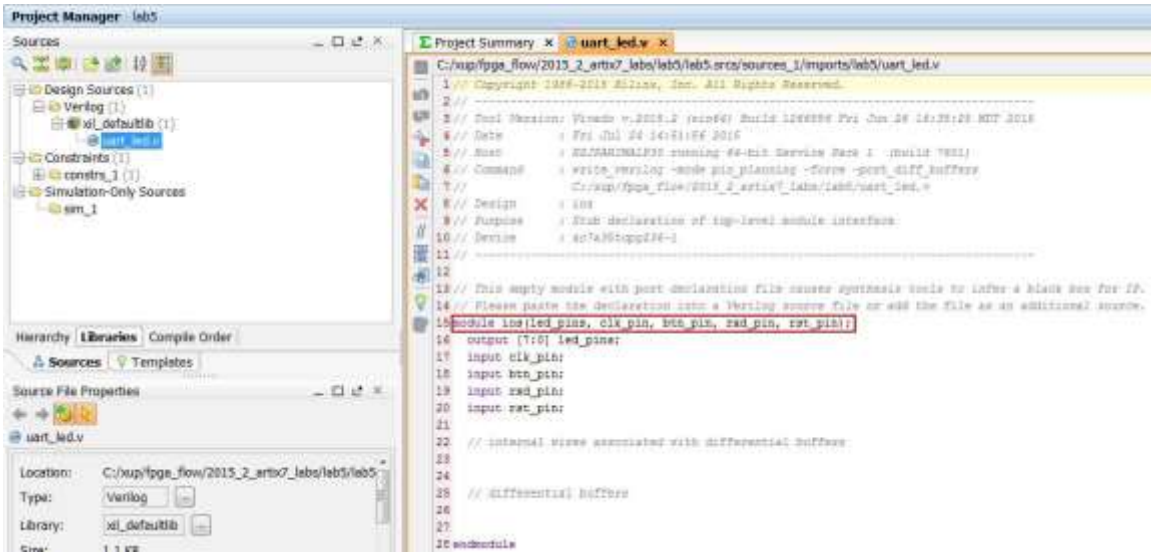


Figure 11. The top-level module content and the design hierarchy after migrating to RTL

2-4. Add the provided source files (from {sources}\lab5) to the project. Copy the uart_led.txt (located in the {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 **{sources}\lab5** and select all *.v* files (led_ctrl, uart_rx, meta_harden, uart_baud_gen, uart_rx_ctl), and click **OK**.

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

2-4-6. Using Windows Explorer, browse to **{sources}\lab5** and open uart_led.txt using any text editor. Copy the content of it and paste it in uart_led.v (around line 22) in the Vivado project.

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.

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.

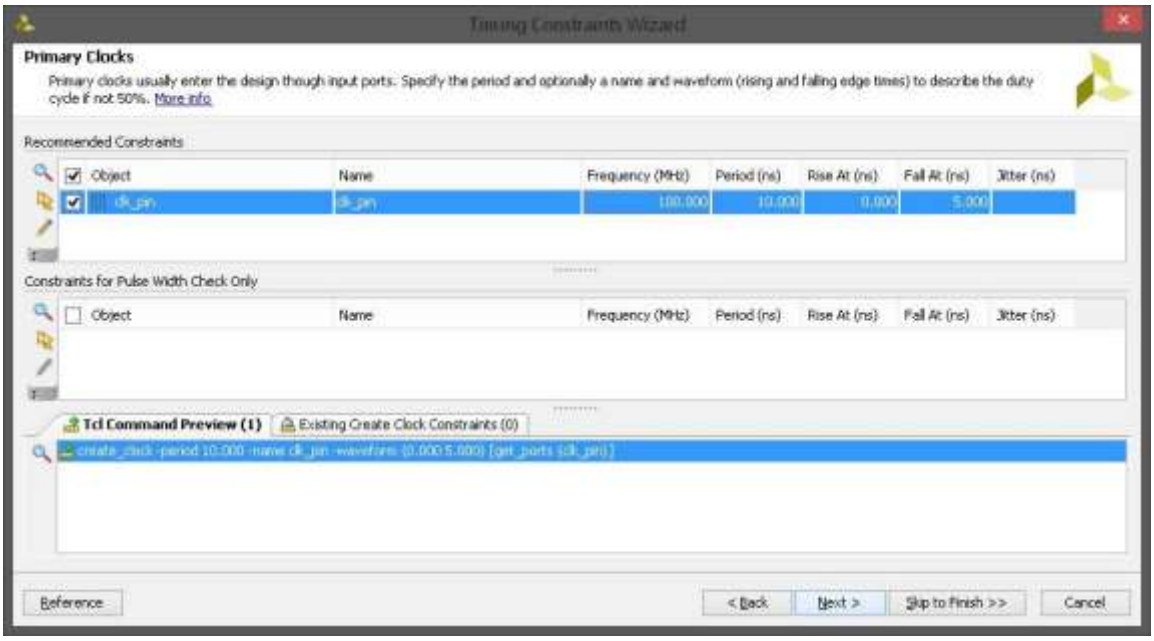


Figure 12. Constraints Wizard *clk_pin* parameters and Tcl command

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, 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**.

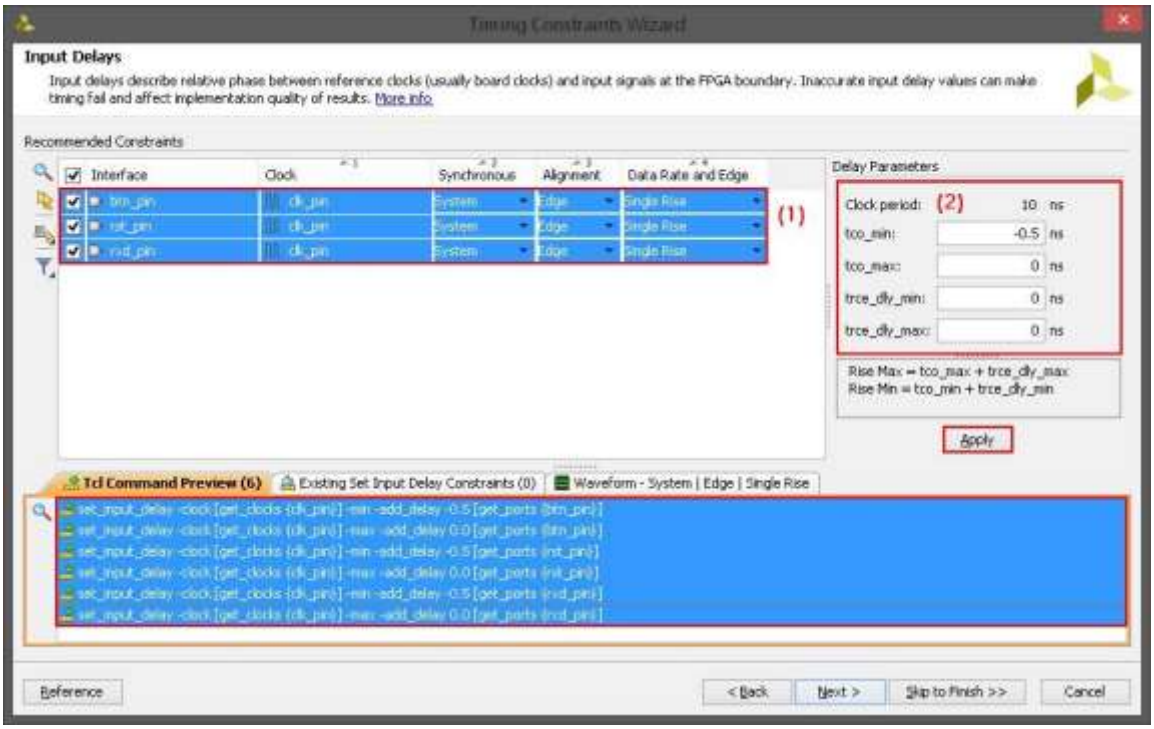


Figure 13. Specifying Input Delays for btn_pin, rst_pin and rxt_pin

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 **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.

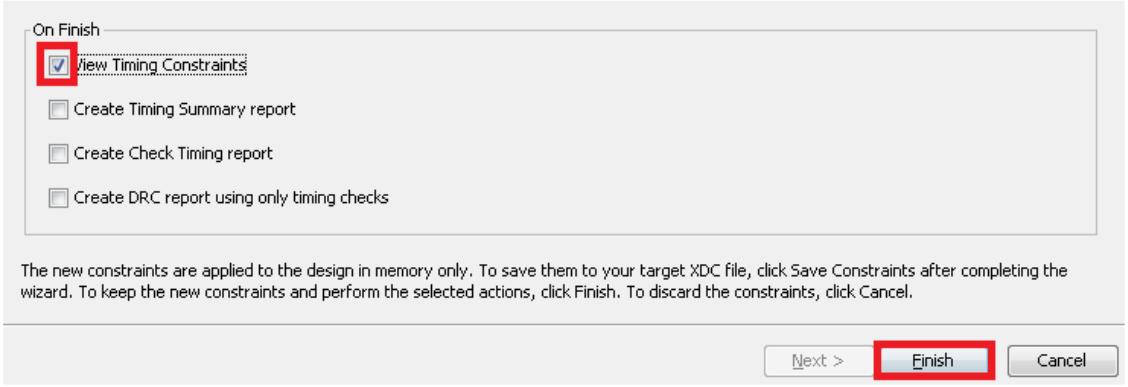


Figure 14. Selecting View Timing constraints

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, 9 constraints will be created.

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

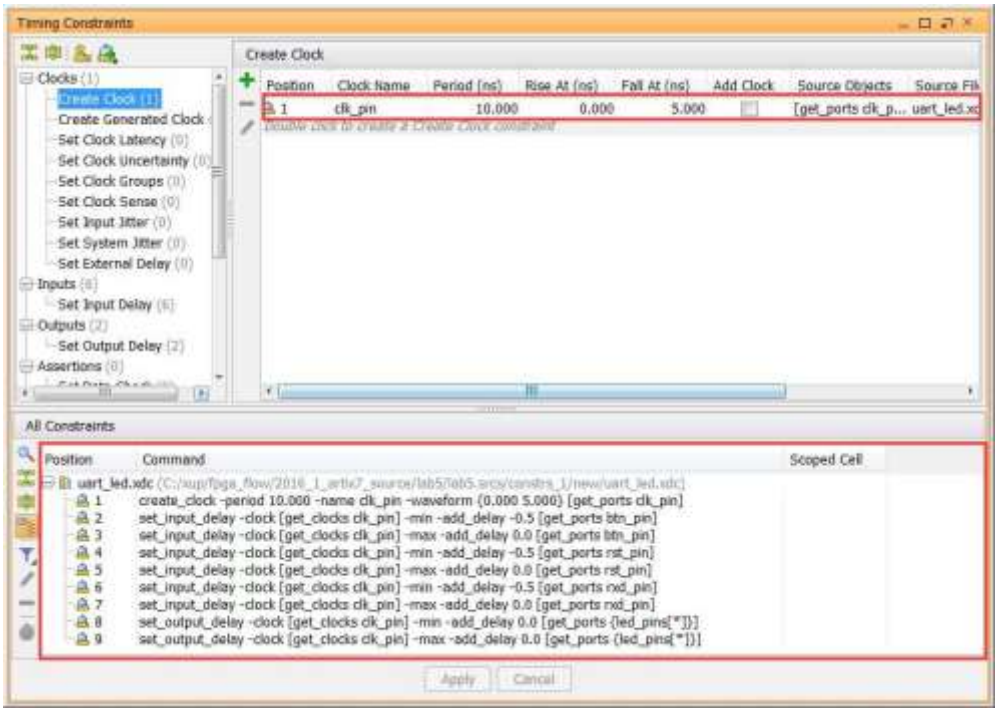


Figure 15. The constraints added after using the Constraints Wizard

3-1-15. 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.

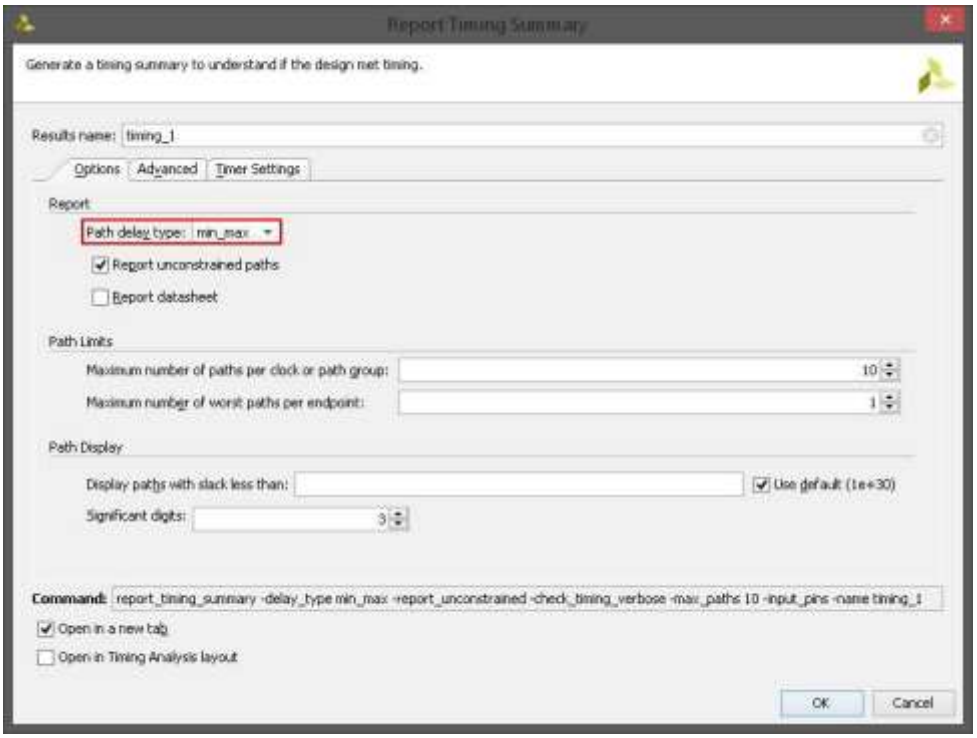


Figure 16. Performing timing analysis

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

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

Figure 17. Timing summary for the Nexys4 DDR

Figure 17. Timing summary for the Basys3

Figure 17. Timing summary for the Nexys Video

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.

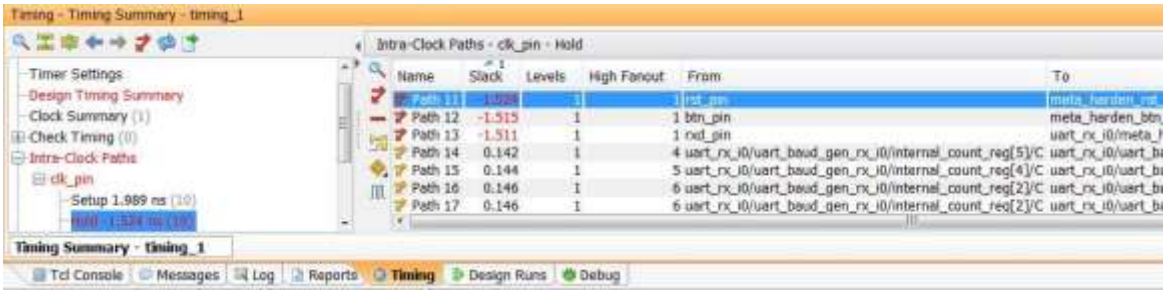


Figure 18. The list of paths showing hold violations for the Nexys4 DDR

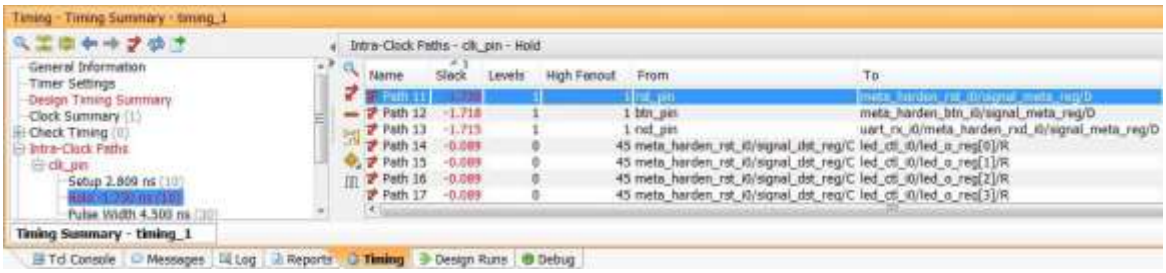


Figure 18. The list of paths showing hold violations for the Basys3

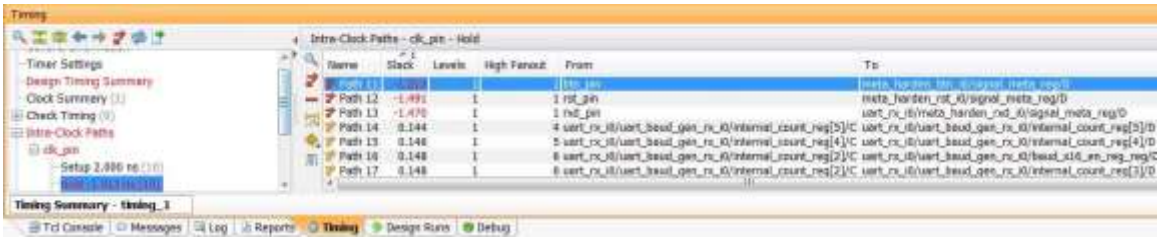
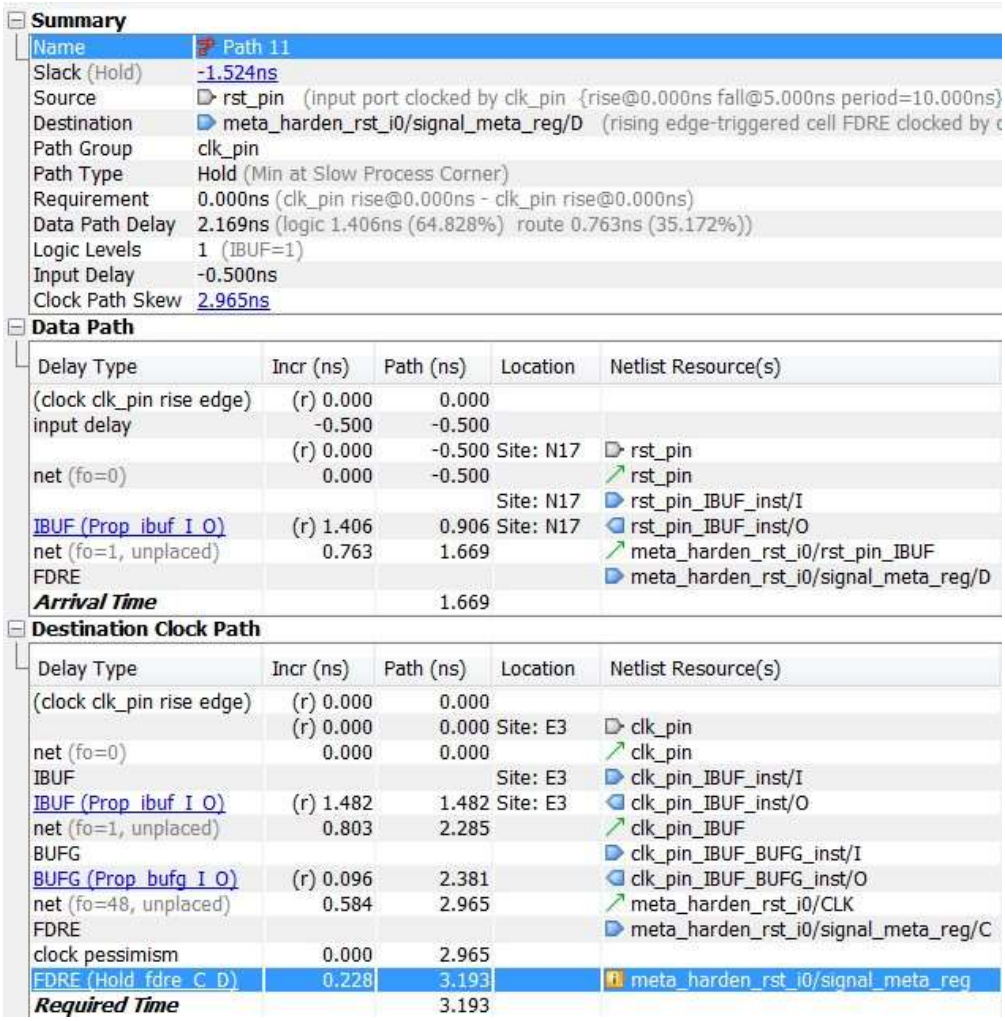


Figure 18. The list of paths showing hold violations for the Nexys Video

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



Summary

Name	Path 11
Slack (Hold)	-1.524ns
Source	rst_pin (input port clocked by clk_pin {rise@0.000ns fall@5.000ns period=10.000ns}
Destination	meta_harden_rst_i0/signal_meta_reg/D (rising edge-triggered cell FDRE clocked by c
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	2.169ns (logic 1.406ns (64.828%) route 0.763ns (35.172%))
Logic Levels	1 (IBUF=1)
Input Delay	-0.500ns
Clock Path Skew	2.965ns

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		
	(r) 0.000	-0.500	Site: N17	rst_pin
net (fo=0)	0.000	-0.500		rst_pin
			Site: N17	rst_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.406	0.906	Site: N17	rst_pin_IBUF_inst/O
net (fo=1, unplaced)	0.763	1.669		meta_harden_rst_i0/rst_pin_IBUF
FDRE				meta_harden_rst_i0/signal_meta_reg/D
Arrival Time		1.669		

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: E3	clk_pin
net (fo=0)	0.000	0.000		clk_pin
IBUF			Site: E3	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.482	1.482	Site: E3	clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.803	2.285		clk_pin_IBUF
BUFG				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.096	2.381		clk_pin_IBUF_BUFG_inst/O
net (fo=48, unplaced)	0.584	2.965		meta_harden_rst_i0/CLK
FDRE				meta_harden_rst_i0/signal_meta_reg/C
clock pessimism	0.000	2.965		
FDRE (Hold_fdre_C_D)	0.228	3.193		meta_harden_rst_i0/signal_meta_reg
Required Time		3.193		

Figure 19. Failing hold path for the Nexys4 DDR

Summary

Name	Path 11
Slack (Hold)	-1.730ns
Source	rst_pin (input port clocked by clk_pin {rise@0.000ns fall@5.000ns period=10.000ns}
Destination	meta_harden_rst_i0/signal_meta_reg/D (rising edge-triggered cell FDRE clocked by
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	2.131ns (logic 1.371ns (64.342%) route 0.760ns (35.658%))
Logic Levels	1 (IBUF=1)
Input Delay	-0.500ns
Clock Path Skew	3.154ns

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		
	(r) 0.000	-0.500	Site: U18	rst_pin
net (fo=0)	0.000	-0.500		rst_pin
			Site: U18	rst_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.371	0.871	Site: U18	rst_pin_IBUF_inst/O
net (fo=1, unplaced)	0.760	1.631		meta_harden_rst_i0/rst_pin_IBUF
FDRE				meta_harden_rst_i0/signal_meta_reg/D
Arrival Time		1.631		

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: W5	clk_pin
net (fo=0)	0.000	0.000		clk_pin
IBUF			Site: W5	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.458	1.458	Site: W5	clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.800	2.258		clk_pin_IBUF
BUFG				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.096	2.354		clk_pin_IBUF_BUFG_inst/O
net (fo=48, unplaced)	0.800	3.154		meta_harden_rst_i0/CLK
FDRE				meta_harden_rst_i0/signal_meta_reg/C
clock pessimism	0.000	3.154		
FDRE (Hold_fdre_C_D)	0.207	3.361		meta_harden_rst_i0/signal_meta_reg
Required Time		3.361		

Figure 19. Failing hold path for the Basys3

Summary

Name	Path 11
Slack (Hold)	-1.513ns
Source	btn_pin (input port clocked by clk_pin {rise@0.000ns fall@5.000ns period=10.000ns}
Destination	meta_harden_btn_i0/signal_meta_reg/D (rising edge-triggered cell FDRE clocked by
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	2.149ns (logic 1.389ns (64.640%) route 0.760ns (35.360%))
Logic Levels	1 (IBUF=1)
Input Delay	-0.500ns
Clock Path Skew	2.955ns

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		
	(r) 0.000	-0.500	Site: F15	btn_pin
net (fo=0)	0.000	-0.500		btn_pin
			Site: F15	btn_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.389	0.889	Site: F15	btn_pin_IBUF_inst/O
net (fo=1, unplaced)	0.760	1.649		meta_harden_btn_i0/btn_pin_IBUF
FDRE				meta_harden_btn_i0/signal_meta_reg/D
Arrival Time		1.649		

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: R4	clk_pin
net (fo=0)	0.000	0.000		clk_pin
IBUF			Site: R4	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.475	1.475	Site: R4	clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.800	2.275		clk_pin_IBUF
BUFG				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.096	2.371		clk_pin_IBUF_BUFG_inst/O
net (fo=48, unplaced)	0.584	2.955		meta_harden_btn_i0/CLK
FDRE				meta_harden_btn_i0/signal_meta_reg/C
clock pessimism	0.000	2.955		
FDRE (Hold_fdre_C_D)	0.207	3.162		meta_harden_btn_i0/signal_meta_reg
Required Time		3.162		

Figure 19. Failing hold path for the Nexys Video

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

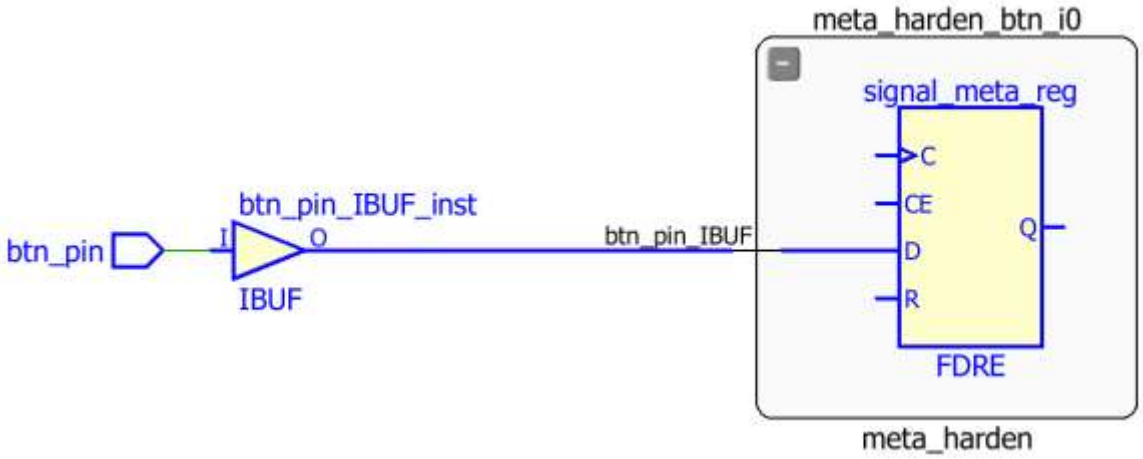


Figure 20. The schematic of the failing path

The three failing paths are of the btn_pin, rxd_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 21. Failing setup paths for the Nexys4 DDR

Figure 21. Failing setup paths for the Basys3

Figure 21. Failing setup paths for the Nexys Video

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.

4-2-11. **Demonstrate the passing timing report to the TA.**

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.**

The `write_bitstream` command will be executed (you can verify it by looking in the Tcl console).

5-1-2. 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). Make sure that is the board set to select USB power (JP3 on the Nexys4 DDR and JP2 on the Basys3).

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 **uart_led.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.

5-3-6. 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.

5-3-7. **Demonstrate the working circuit to the TA.**

5-3-8. When finished, close the terminal emulator program and power OFF the board.

5-3-9. Select **File > Close Hardware Manager**. Click **OK** to close it.

5-3-10. 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.