
Synthesizing a RTL Design

Introduction

This lab shows you the synthesis process and effect of changing of synthesis settings. You will analyze the design and the generated reports.

Objectives

After completing this lab, you will be able to:

- Use the provided Xilinx Design Constraint (XDC) file to constrain the timing of the circuit
- Elaborate the design and understand the output
- Synthesize the design with the provided basic timing constraints
- Analyze the output of the synthesized design
- Change the synthesis settings and see their effects on the generated output
- Write a checkpoint after the synthesis so the results can be analyzed after re-loading it

Procedure

This lab is broken into steps that consist of general overview statements providing information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

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

Design Description

The design consists of a uart receiver receiving the input typed on a keyboard and displaying the binary equivalent of the typed character on the 8 LEDs. When a push button is pressed, the lower and upper nibbles are swapped. The block diagram is as shown in **Figure 1**.

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

The provided design places the UART (RX) pin of the PS (Processing System) on the Cortex-A9 in a simple GPIO mode to allow the UART to be connected (passed through) to the Programmable Logic. The processor samples the RX signal and sends it to the EMIO channel 0 which is connected to Rx input of the HDL module provided in the Static directory. This is done through a software application provided in the lab3.sdk folder hierarchy.

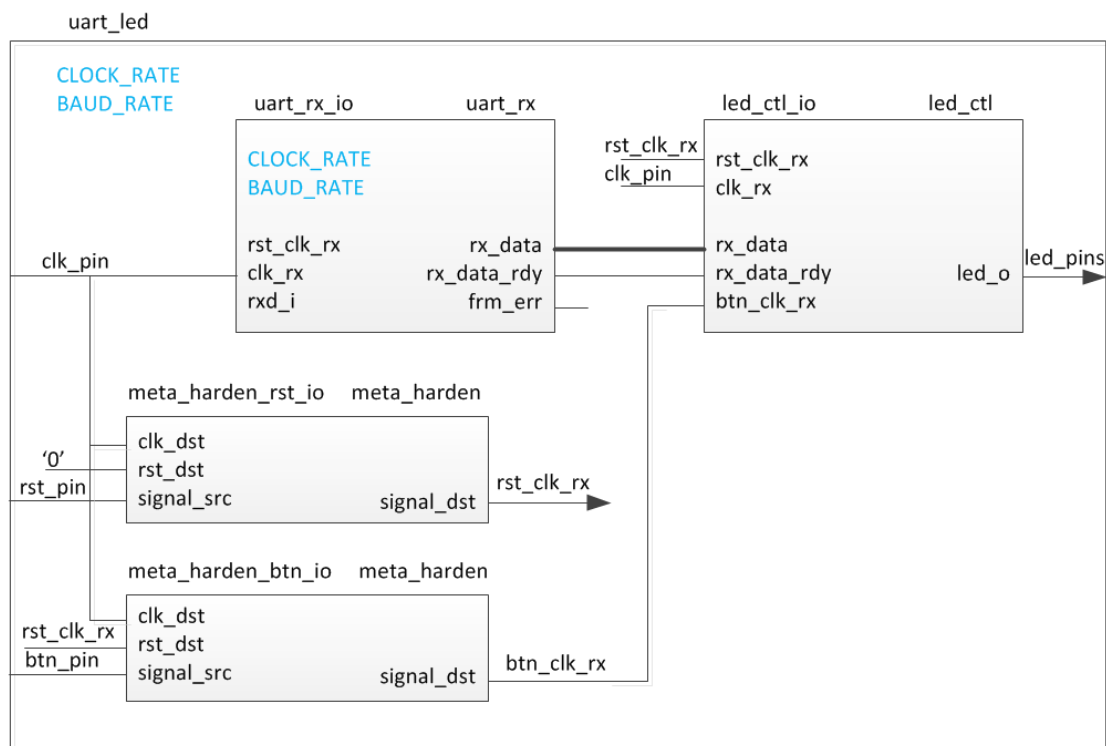


Figure 1. The Complete Design on PL

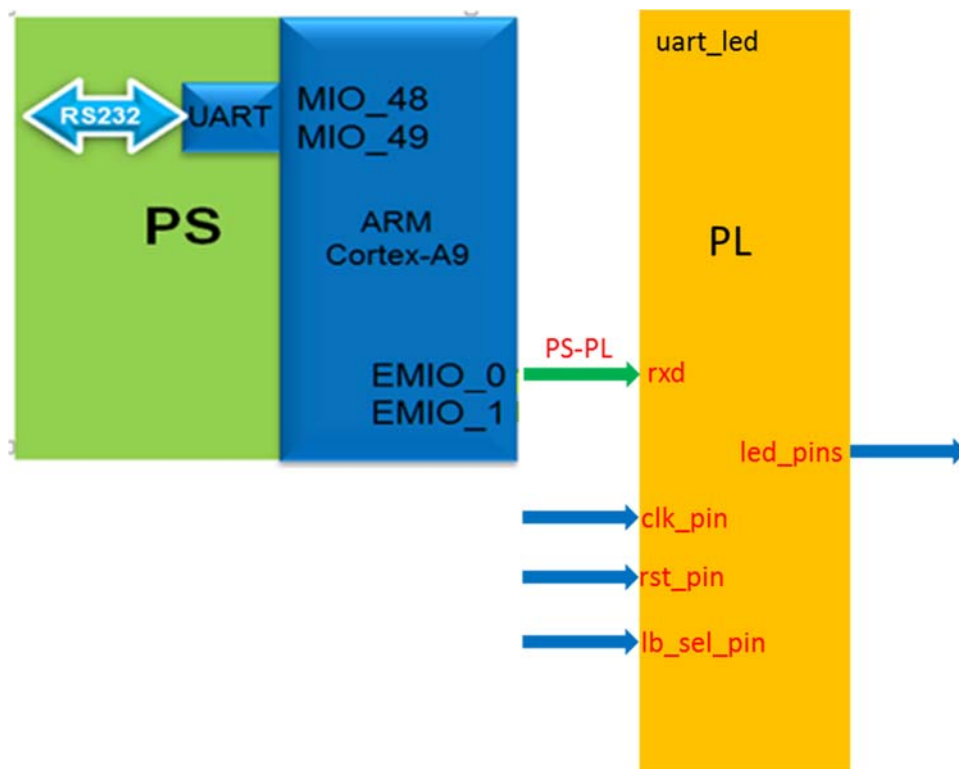
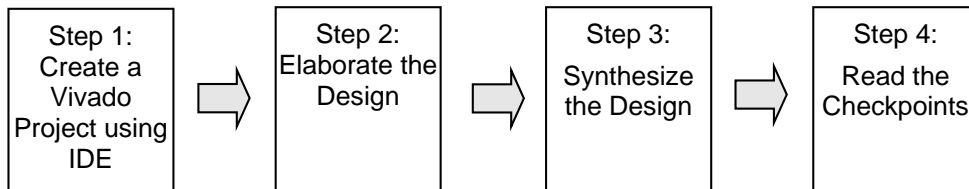


Figure 2. The Complete System

General Flow



Create a Vivado Project using IDE

Step 1

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

- 1-1. Launch Vivado and create a project targeting the XC7Z020clg484-1 device (ZedBoard), or the XC7Z010clg400-1 (Zybo), and use provided the tcl scripts (ps7_create_<board>.tcl) to generate the block design for the PS subsystem. Also, add the Verilog HDL files, uart_led_pins_<board>.xdc and uart_led_timing.xdc files from the <2016_2_ZYNQ_sources>\lab2 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 **lab2** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.
- 1-1-5. Select **RTL Project** option in the *Project Type* form, and click **Next**.
- 1-1-6. Using the drop-down buttons, select **Verilog** as the *Target Language* and *Simulator Language* in the *Add Sources* form.

1-1-7. Click on the **Green Plus** button, then the **Add Files...** button and browse to the **<2016_2_ZYNQ_sources>\lab2** directory, select all the Verilog files (*led_ctl.v, meta_harden.v, uart_baud_gen.v, uart_led.v, uart_rx.v, uart_rx_ctl.v and uart_top.v*), click **OK**, and then click **Next** to get to the *Add Existing IP* form.

1-1-8. Since we do not have any IP to add, click **Next** to get to the *Add Constraints* form.

1-1-9. Click on the **Green Plus** button, then **Add Files...** and browse to the **c:\xup\fpga_flow\2016_2_ZYNQ\lab2** directory (if necessary), select *uart_led_timing_<board>.xdc* and click **Open**.

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

This Xilinx Design Constraints file assigns the basic timing constraints (period, input delay, and output delay) to the design.

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

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

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

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

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

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

```
cd c:/xup/fpga_flow/2016_2_ZYNQ_sources/lab2
```

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

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

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

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

1-2. Analyze the design source files hierarchy.

1-2-1. In the *Sources* pane, expand the **uart_led** entry and notice hierarchy of the lower-level modules.

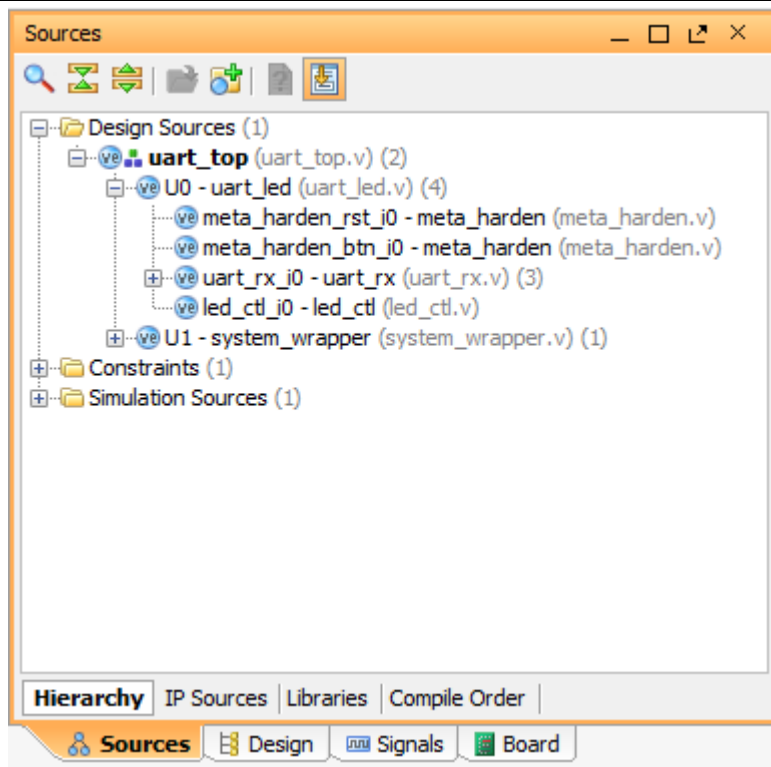


Figure 3. Opening the source file

1-2-2. 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 as shown in the design diagram (Figure 1). Also notice that the lower level modules are instantiated. The meta_harden modules are used to synchronize the asynchronous reset and push-button inputs.

Note: Notice that the 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

```

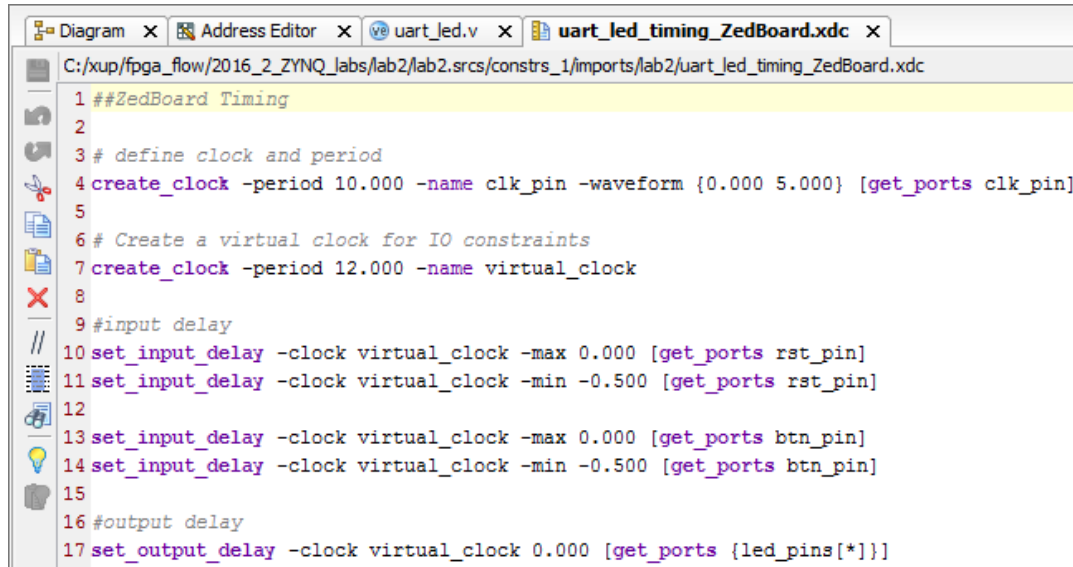
Figure 4. CLOCK_RATE parameter of uart_led for Zybo board

1-2-3. Expand **U0** and **uart_rx_i0** instance to see its hierarchy.

This module uses the baud rate generator and a finite state machine. The rxd_pin is sampled at a x16 the baud rate.

1-3. Open the **uart_led_timing_<board>.xdc** source and analyze the content.

1-3-1. In the *Sources* pane, expand the *Constraints* folder and double-click the **uart_led_timing_<board>.xdc** entry to open the file in text mode.



```

1 ##ZedBoard Timing
2
3 # define clock and period
4 create_clock -period 10.000 -name clk_pin -waveform {0.000 5.000} [get_ports clk_pin]
5
6 # Create a virtual clock for IO constraints
7 create_clock -period 12.000 -name virtual_clock
8
9 #input delay
10 set_input_delay -clock virtual_clock -max 0.000 [get_ports rst_pin]
11 set_input_delay -clock virtual_clock -min -0.500 [get_ports rst_pin]
12
13 set_input_delay -clock virtual_clock -max 0.000 [get_ports btn_pin]
14 set_input_delay -clock virtual_clock -min -0.500 [get_ports btn_pin]
15
16 #output delay
17 set_output_delay -clock virtual_clock 0.000 [get_ports {led_pins[*]}]

```

Figure 5. Timing constraints

Line 4 creates the period constraint of 10 ns with a duty cycle of 50%. Line 7 creates a virtual clock of 12 ns. This clock can be viewed as the upstream device is generating its output with respect to its clock and outputs data with respect to it. The btn_pin is constrained with respect to the upstream clock (lines 9, 10). The led_pins are constrained with respect to the upstream clock as the downstream device may be using it.

Note: Notice that the Zybo has an on-board clock of 125 Mhz and hence Line 4 creates the period constraint of 8 ns with a duty cycle of 50%.

Elaborate the Design

Step 2

2-1. Elaborate and perform the RTL analysis on the source file.

2-1-1. Expand the *Open Elaborated Design* entry under the *RTL Analysis* tasks of the *Flow Navigator* pane and click on **Schematic**. Click **OK**.

The model (design) will be elaborated and a logical view of the design is displayed.

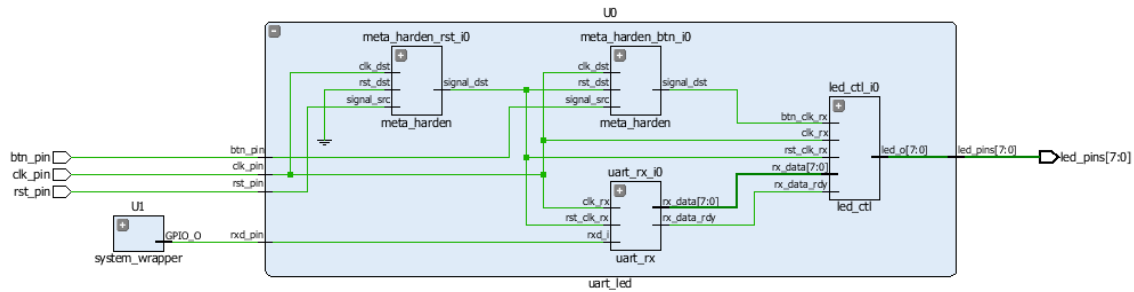


Figure 6. A logic view of the design one-level down from the top in component U0

You will see two components at the top-level, going down one level in component U0 shows 2 instances of meta_harden, one instance of uart_rx, and one instance of led_ctl.

- 2-1-2.** To see where the uart_rx_i0 gets generated, right-click on the uart_rx_i0 instance and select *Go To Source* and see that line 84 in the source code is generating it.
- 2-1-3.** Double-click on the uart_rx_i0 instance in the schematic diagram to see the underlying components.

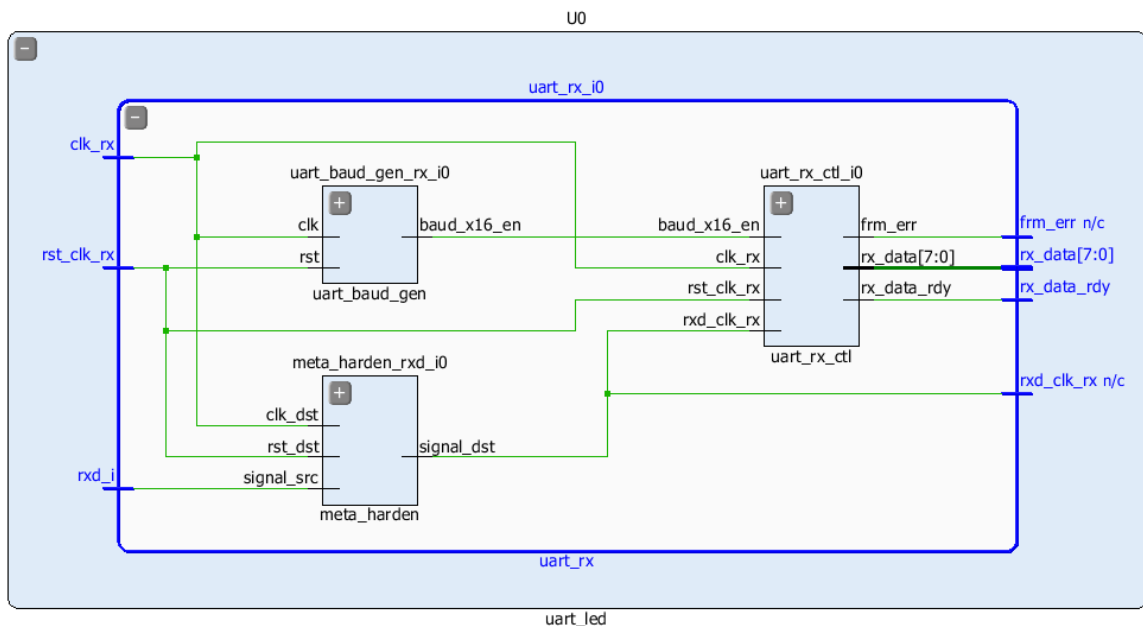


Figure 7. Lower level components of the uart_rx_i0 module

- 2-1-4.** Click on **Report Noise** under the *Open Elaborated Design* entry of the *RTL Analysis* tasks of the *Flow Navigator* pane.
- 2-1-5.** Click **OK** to generate the report named **ssn_1**.
- 2-1-6.** View the ssn_1 report and observe the unplaced ports, Summary, and I/O Bank Details are highlighted in red because the pin assignments were not done. Note that only output pins are reported as the noise analysis is done on the output pins.

Name	Port	I/O Std	Vcco	Slew	Drive Strength (mA)	Off-Chip Termination	Remaining Margin (%)	Notes
I/O Bank 0 (0)								
I/O Bank 13 (0)								
I/O Bank 33 (0)								
I/O Bank 34 (0)								
I/O Bank 35 (0)								
Unplaced Ports (8)								
led_pins[0]	led_pins[0]	LVCMOS18		1.80 SLOW	12 FP_VTT_50			CRITICAL WARNING - Unplaced port
led_pins[1]	led_pins[1]	LVCMOS18		1.80 SLOW	12 FP_VTT_50			CRITICAL WARNING - Unplaced port
led_pins[2]	led_pins[2]	LVCMOS18		1.80 SLOW	12 FP_VTT_50			CRITICAL WARNING - Unplaced port
led_pins[3]	led_pins[3]	LVCMOS18		1.80 SLOW	12 FP_VTT_50			CRITICAL WARNING - Unplaced port
led_pins[4]	led_pins[4]	LVCMOS18		1.80 SLOW	12 FP_VTT_50			CRITICAL WARNING - Unplaced port
led_pins[5]	led_pins[5]	LVCMOS18		1.80 SLOW	12 FP_VTT_50			CRITICAL WARNING - Unplaced port
led_pins[6]	led_pins[6]	LVCMOS18		1.80 SLOW	12 FP_VTT_50			CRITICAL WARNING - Unplaced port
led_pins[7]	led_pins[7]	LVCMOS18		1.80 SLOW	12 FP_VTT_50			CRITICAL WARNING - Unplaced port

Figure 8. Noise report

2-1-7. Click on **Add Sources** under the *Project Navigator*, select *Add or Create Constraints* option and click **Next**.

2-1-8. Click on the **Green Plus** button, then the **Add Files...** button and browse to the **<2016_2_ZYNQ_sources>\lab2** directory, select the **uart_led_pins_<board>.xdc** file (depending on the target board), click **OK**, and then click **Finish** to add the pins location constraints.

Notice that the sources are modified and the tools detect it, showing a warning status bar to reload the design.

Elaborated Design is out-of-date. Constraints were modified. [more info](#) [Reload](#)

2-1-9. Click on the **Reload** link. The constraints will be processed.

2-1-10. Click on **Report Noise** and click **OK** to generate the report named **ssn_1**. Observe that this time it does not show any errors (no red).

Synthesize the Design

Step 3

3-1. Synthesize the design with the Vivado synthesis tool and analyze the Project Summary output.

3-1-1. Click on **Run Synthesis** under the *Synthesis* tasks of the *Flow Navigator* pane.

Click **Save** if the Save Project dialog box is displayed.

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

3-1-2. Select the *Open Synthesized Design* option and click **OK** as we want to look at the synthesis output.

Click **Yes** to close the elaborated design if the dialog box is displayed.

3-1-3. Select the **Project Summary** tab

If you don't see the Project Summary tab then select **Layout > Default Layout**, or click the

Project Summary icon .

3-1-4. Click on the **Table** tab in the **Project Summary** tab and fill out the following information.

Question 1

Look through the table and find the number used of each of the following:

FF: _____
 LUT: _____
 I/O: _____
 BUFG: _____

3-1-5. Click on **Schematic** under the *Open Synthesized Design* tasks of *Synthesis* tasks of the *Flow Navigator* pane to view the synthesized design in a schematic view.

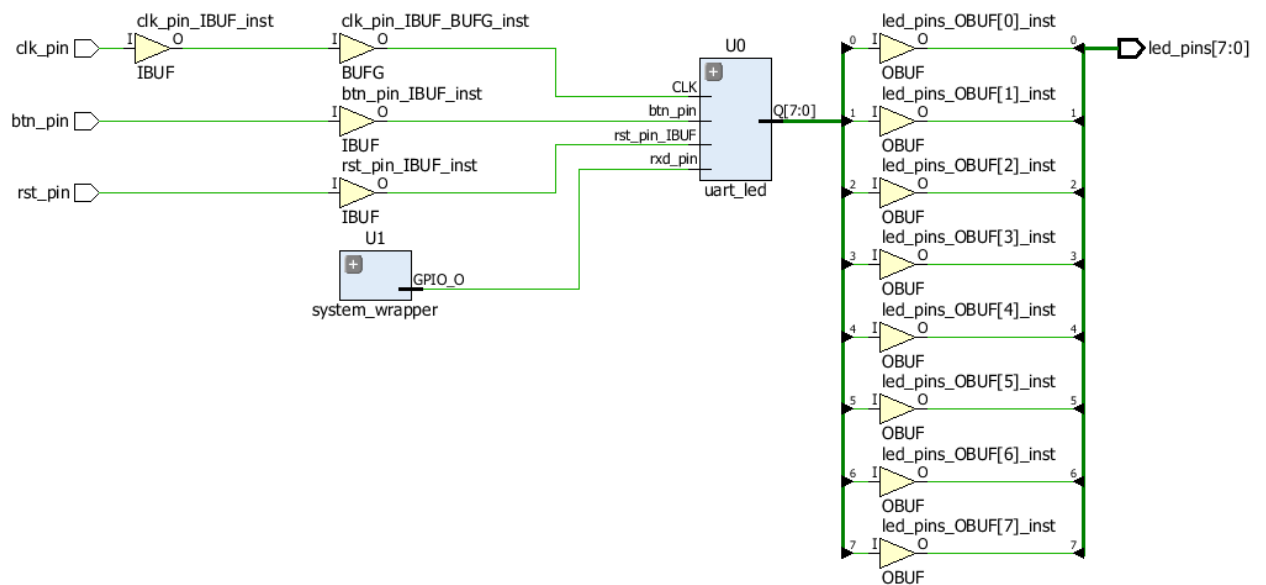


Figure 9. Synthesized design's schematic view

Notice that IBUF and OBUF are automatically instantiated (added) to the design as the input and output are buffered. There are still four lower level modules instantiated.

3-1-6. Double-click on **U0** and **uart_rx_i0** instances in the schematic view to see the underlying instances.

3-1-7. Select the **uart_baud_gen_rx_i0** instance, right-click, and select *Go To Source*.

Notice that line 86 is highlighted. Also notice that the **CLOCK_RATE** and **BAUD_RATE** parameters are passed to the module being called.

3-1-8. Double-click on the **meta_harden_rxd_i0** instance to see how the synchronization circuit is being implemented using two FFs. This synchronization is necessary to reduce the likelihood of meta-stability.

3-1-9. Click on the () in the schematic view to go back to its parent block.

3-2. Analyze the timing report.

3-2-1. Click on **Report Timing Summary** under the *Synthesized Design* tasks of the *Flow Navigator* pane.

3-2-2. Click **OK** to generate the Timing_1 report.

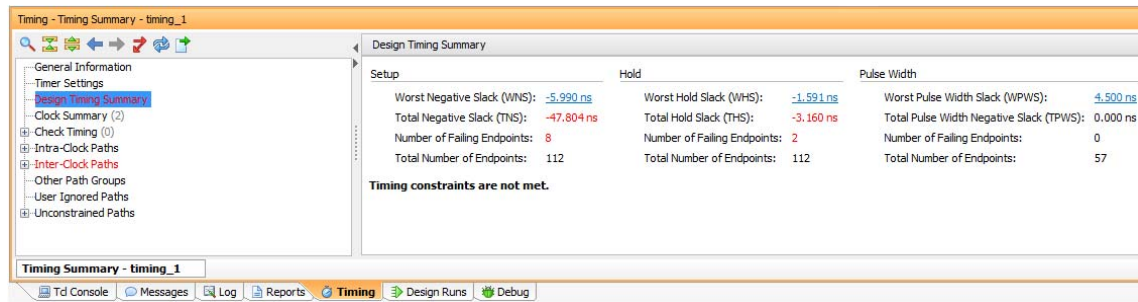


Figure 10. Timing report for the ZedBoard

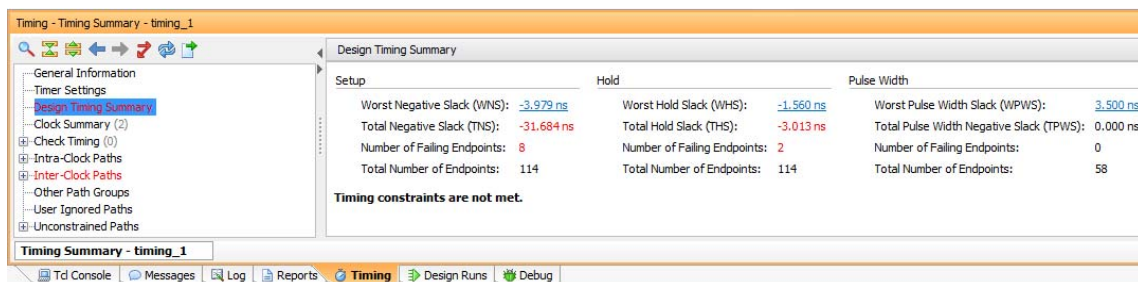


Figure 10. Timing report for the Zybo

Notice that the Design Timing Summary and Inter-Clock Paths entry in the left pane is highlighted in red indicating timing violations. In the right pane, the information is grouped in Setup, Hold, and Width columns.

Under the Setup column Worst Negative Slack (WNS) is linked indicating that clicking on it can give us insight on how the failing path has formed. The Total Negative Slack (TNS) is highlighted in red indicating the total amount of violations in the design and the Number of Failing Endpoints indicate total number of failing paths.

3-2-3. Click on the WNS link and see the 8 failing paths.

Inter-Clock Paths - clk_pin to virtual_clock - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 25	-5.990	1	1	U0/led_cti_j0/led_o_reg[3]/C	led_pins[3]	4.990	4.190	0.800	2.000	clk_pin	virtual_clock
Path 26	-5.984	1	1	U0/led_cti_j0/led_o_reg[5]/C	led_pins[5]	4.984	4.184	0.800	2.000	clk_pin	virtual_clock
Path 27	-5.984	1	1	U0/led_cti_j0/led_o_reg[2]/C	led_pins[2]	4.984	4.184	0.800	2.000	clk_pin	virtual_clock
Path 28	-5.983	1	1	U0/led_cti_j0/led_o_reg[4]/C	led_pins[4]	4.983	4.183	0.800	2.000	clk_pin	virtual_clock
Path 29	-5.975	1	1	U0/led_cti_j0/led_o_reg[7]/C	led_pins[7]	4.975	4.175	0.800	2.000	clk_pin	virtual_clock
Path 30	-5.967	1	1	U0/led_cti_j0/led_o_reg[6]/C	led_pins[6]	4.967	4.167	0.800	2.000	clk_pin	virtual_clock
Path 31	-5.965	1	1	U0/led_cti_j0/led_o_reg[1]/C	led_pins[1]	4.965	4.165	0.800	2.000	clk_pin	virtual_clock
Path 32	-5.955	1	1	U0/led_cti_j0/led_o_reg[0]/C	led_pins[0]	4.955	4.155	0.800	2.000	clk_pin	virtual_clock

Log Reports Timing Design Runs Debug

Figure 11. The 8 failing paths for the ZedBoard

Inter-Clock Paths - clk_pin to virtual_clock - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 25	-3.979	1		U0/led_ctl_i0/led_o_reg[7]/C	led_pins[7]	4.977	4.178	0.800	4.000	clk_pin	virtual_clock
Path 26	-3.977	1		U0/led_ctl_i0/led_o_reg[6]/C	led_pins[6]	4.975	4.176	0.800	4.000	clk_pin	virtual_clock
Path 27	-3.971	1		U0/led_ctl_i0/led_o_reg[1]/C	led_pins[1]	4.970	4.170	0.800	4.000	clk_pin	virtual_clock
Path 28	-3.969	1		U0/led_ctl_i0/led_o_reg[4]/C	led_pins[4]	4.968	4.168	0.800	4.000	clk_pin	virtual_clock
Path 29	-3.965	1		U0/led_ctl_i0/led_o_reg[5]/C	led_pins[5]	4.964	4.164	0.800	4.000	clk_pin	virtual_clock
Path 30	-3.963	1		U0/led_ctl_i0/led_o_reg[0]/C	led_pins[0]	4.962	4.162	0.800	4.000	clk_pin	virtual_clock
Path 31	-3.942	1		U0/led_ctl_i0/led_o_reg[3]/C	led_pins[3]	4.941	4.141	0.800	4.000	clk_pin	virtual_clock
Path 32	-3.918	1		U0/led_ctl_i0/led_o_reg[2]/C	led_pins[2]	4.916	4.116	0.800	4.000	clk_pin	virtual_clock

Log Reports **Timing** Design Runs Debug

Figure 11. The 8 failing paths for the Zybo

3-2-4. Double-click on the **Path 25** to see how the path is made.

Summary

Name: Path 25

Slack: -5.990ns

Source: U0/led_ctl_i0/led_o_reg[3]/C (rising edge-triggered cell FDRE clocked by clk_pin {rise@0.000ns fall@5.000ns period=10.000ns})

Destination: led_pins[3] (output port clocked by virtual_clock {rise@0.000ns fall@6.000ns period=12.000ns})

Path Group: virtual_clock

Path Type: Max at Slow Process Corner

Requirement: 2.000ns (virtual_clock rise@12.000ns - clk_pin rise@10.000ns)

Data Path Delay: 4.990ns (logic 4.190ns (83.970%) route 0.800ns (16.030%))

Logic Levels: 1 (OBUF=1)

Output Delay: 0.000ns

Clock Path Skew: -2.975ns

Clock Uncertainty: 0.025ns

Clock Domain Crossing: Inter clock paths are considered valid unless explicitly excluded by timing constraints such as set_clock_groups or set_false_path.

Source Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 10.000	10.000		
	(r) 0.000	10.000	Site: Y9	clk_pin
net (fo=0)	0.000	10.000		clk_pin
			Site: Y9	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.490	11.490	Site: Y9	clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.800	12.290		clk_pin_IBUF
				clk_pin_IBUF_BUF_inst/I
BUF (Prop_bufg_I_O)	(r) 0.101	12.391		clk_pin_IBUF_BUF_inst/O
net (fo=56, unplaced)	0.584	12.975		U0/led_ctl_i0/CLK
FDRE				U0/led_ctl_i0/led_o_reg[3]/C

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_O)	(r) 0.478	13.453		U0/led_ctl_i0/led_o_reg[3]/Q
net (fo=1, unplaced)	0.800	14.253		led_pins_OBUF[3]
			Site: V22	led_pins_OBUF[3]_inst/I
OBUF (Prop_obuf_I_O)	(r) 3.712	17.965	Site: V22	led_pins_OBUF[3]_inst/O
net (fo=0)	0.000	17.965		led_pins[3]
			Site: V22	led_pins[3]
Arrival Time		17.965		

Figure 12. Worst failing path for the ZedBoard

Summary

Name

Path 25

Slack

-3.979ns

Source

U0/led_ctl_i0/led_o_reg[7]/C (rising edge-triggered cell FDRE clocked by clk_pin {rise@0.000ns fall@4.000ns period=8.000ns})

Destination

<led_pins[7] (output port clocked by virtual_clock {rise@0.000ns fall@6.000ns period=12.000ns})

Path Group

virtual_clock

Path Type

Max at Slow Process Corner

Requirement

4.000ns (virtual_clock rise@12.000ns - clk_pin rise@8.000ns)

Data Path Delay

4.977ns (logic 4.178ns (83.931%) route 0.800ns (16.069%))

Logic Levels

1 (OBUF=1)

Output Delay

0.000ns

Clock Path Skew

-2.976ns

Clock Uncertainty

0.025ns

Clock Domain Crossing

Inter clock paths are considered valid unless explicitly excluded by timing constraints such as set_clock_groups or set_false_path.

Source Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 8.000	8.000		
	(r) 0.000	8.000	Site: L16	clk_pin
net (fo=0)	0.000	8.000		clk_pin
			Site: L16	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.491	9.491	Site: L16	clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.800	10.291		clk_pin_IBUF
				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.101	10.392		clk_pin_IBUF_BUFG_inst/O
net (fo=57, unplaced)	0.584	10.976		U0/led_ctl_i0/CLK
FDRE				U0/led_ctl_i0/led_o_reg[7]/C

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_O)	(r) 0.456	11.432		U0/led_ctl_i0/led_o_reg[7]/Q
net (fo=1, unplaced)	0.800	12.232		led_pins_OBUF[7]
			Site: W20	led_pins_OBUF[7]_inst/I
OBUF (Prop_obuf_I_O)	(r) 3.722	15.954	Site: W20	led_pins_OBUF[7]_inst/O
net (fo=0)	0.000	15.954		led_pins[7]
			Site: W20	<led_pins[7]
Arrival Time		15.954		

Figure 12. Worst failing path for the Zybo

Note that this is an estimate only. The nets are specified as unplaced and have all been allocated default values (0.584 ns). No actual routing delays are considered.

3-3. Generate the utilization and power reports.

- 3-3-1. Click **Report Utilization** under the Synthesized Design, and click **OK** to generate the utilization report. Click on **Summary** in the left pane.

Resource	Utilization	Available	Utilization %
LUT	88	53200	0.17
FF	56	106400	0.05
IO	11	200	5.50
BUFG	1	32	3.13

Figure 13. Utilization report for the ZedBoard

Resource	Utilization	Available	Utilization %
LUT	87	17600	0.49
FF	57	35200	0.16
IO	11	100	11.00
BUFG	1	32	3.13

Figure 13. Utilization report for the Zybo

Question 2

Look through the report and find the number used of each of the following:

FF: _____
 LUT: _____
 I/O: _____
 BUFG: _____

- 3-3-2.** Select Slice LUTs entry in the left pane and see the utilization by lower-level instances. You can expand the instances in the right pane to see the complete hierarchy utilization.

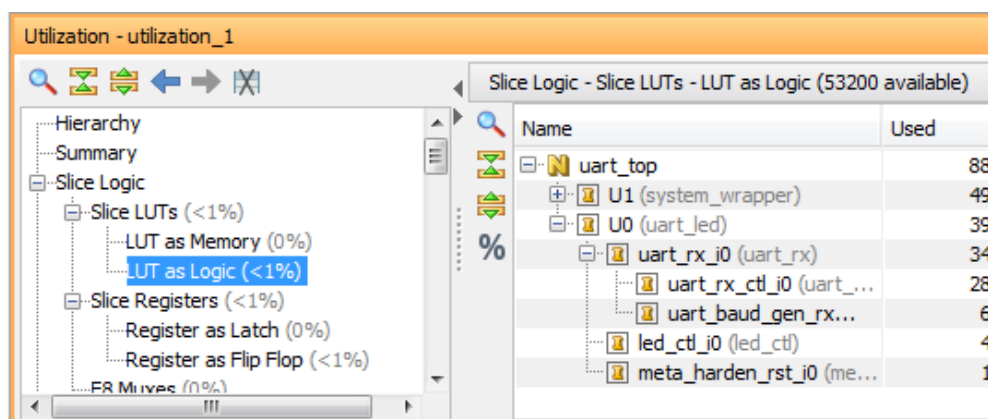


Figure 14. Utilization of lower-level modules for the ZedBoard

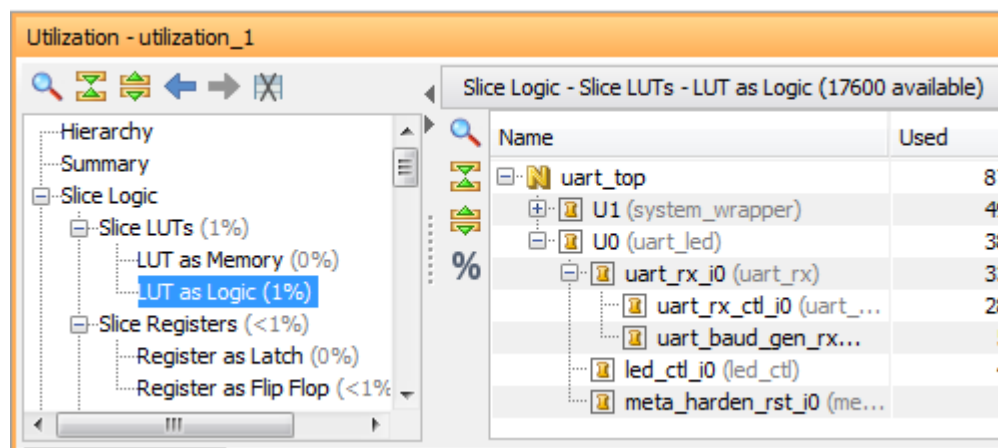


Figure 14. Utilization of lower-level modules for the Zybo

- 3-3-3.** Click **Report Power** under the Synthesized Design, and click **OK** to generate the estimated power consumption report using default values.

Note that this is just an estimate as no simulation run data was provided and no accurate activity rate, or environment information was entered.

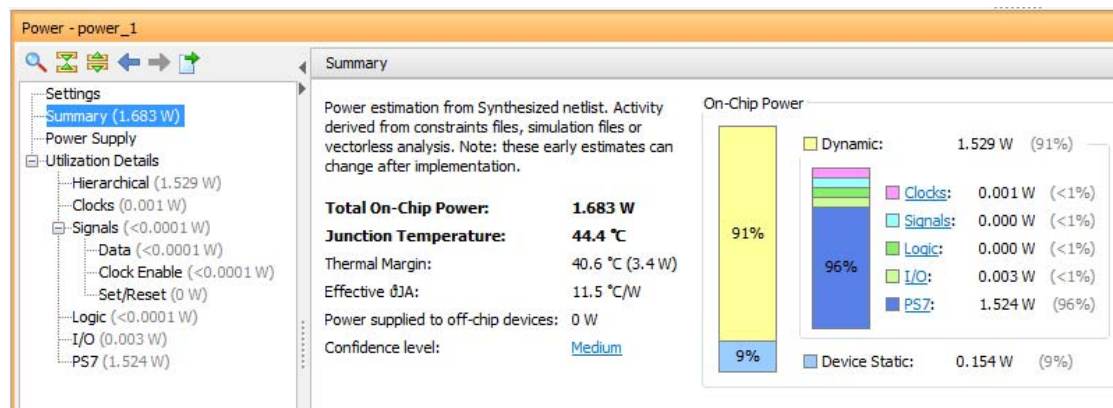


Figure 15. Power consumption estimation for the ZedBoard

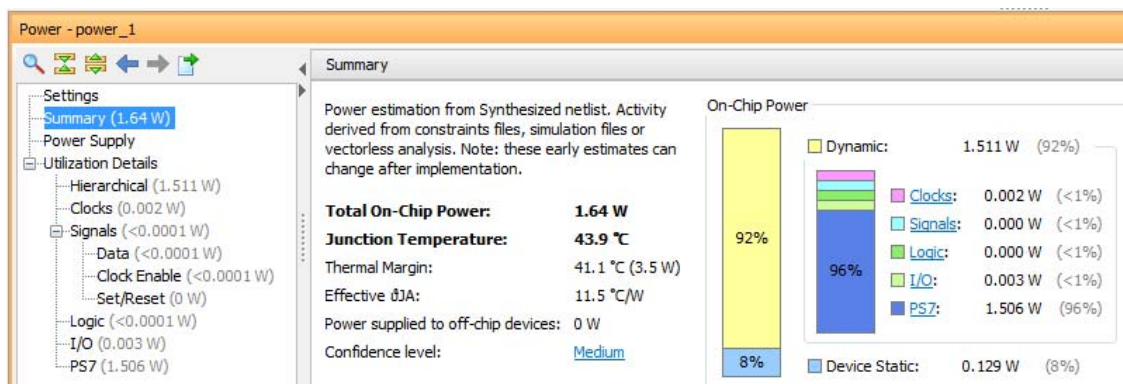


Figure 15. Power consumption estimation for the Zybo

Question 3

From the power report, find the % power consumption used by each of the following:

Clocks:	_____ %
Signals:	_____ %
Logic:	_____ %
I/O:	_____ %
PS7:	_____ %

You can move the mouse on the boxes which do not show the percentage to see the consumption.

3-4. Write the checkpoint in order to analyze the results without going through the actual synthesis process.

3-4-1. Select **File > Write Checkpoint...** to save the processed design so it can be opened later for further analysis.

3-4-2. A dialog box will appear showing the default name of the file in the current project directory.

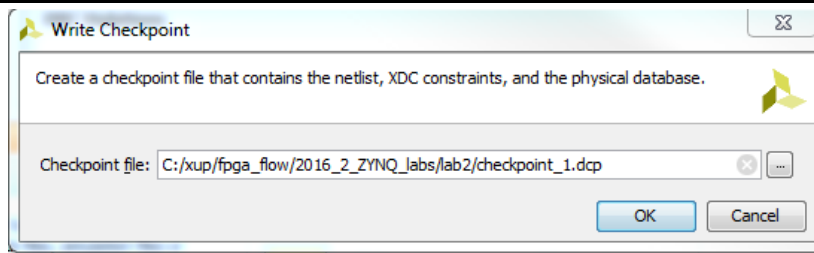


Figure 16. Writing checkpoint

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

3-5. Change the synthesis settings to flatten the design. Re-synthesize the design and analyze the results.

3-5-1. Click on the **Project Settings** under the *Project Manager*, and select **Synthesis**.

3-5-2. Click on the **flatten_hierarchy** drop-down button and select **full** to flatten the design.

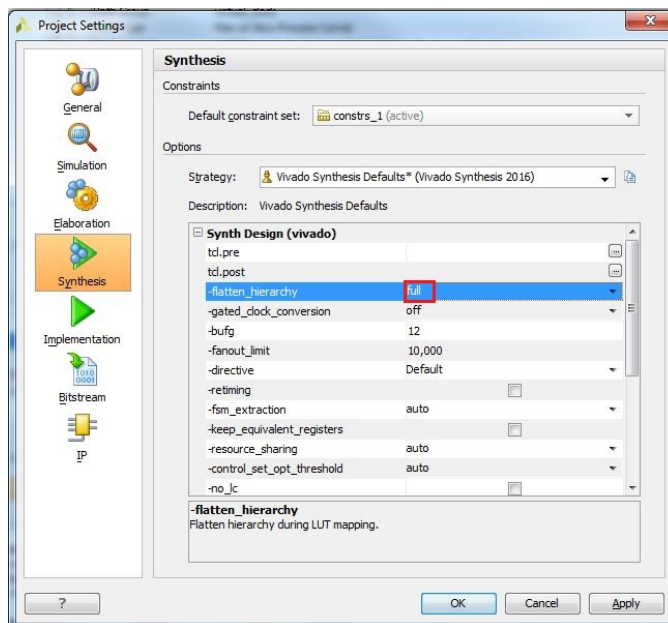


Figure 17. Selecting flatten hierarchy option

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

3-5-4. A Create New Run dialog box will appear asking you whether you want to create a new run since the settings have been changed.

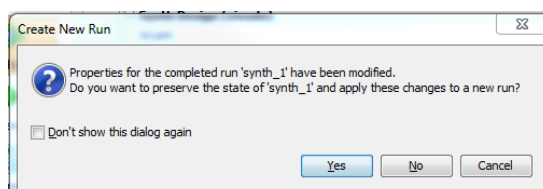


Figure 18. Create New Run dialog box

- 3-5-5.** Click **Yes**.
- 3-5-6.** Change the name from **synth_2** to **synth_flatten** and click **OK**.
- 3-5-7.** Click **Run Synthesis** to synthesize the design.
- 3-5-8.** Click **Save**, **OK**, and again **OK** to save the synthesized design and save the constraints.
The Reload Design dialog box may re-appear. Click **Cancel**.
- 3-5-9.** Click **OK** to open the synthesized design when synthesis process is completed.
- 3-5-10.** Click on **Schematic** under the *Open Synthesized Design* tasks of *Synthesis* tasks of the *Flow Navigator* pane to view the synthesized design in a schematic view.

Notice that the design is completely flattened.

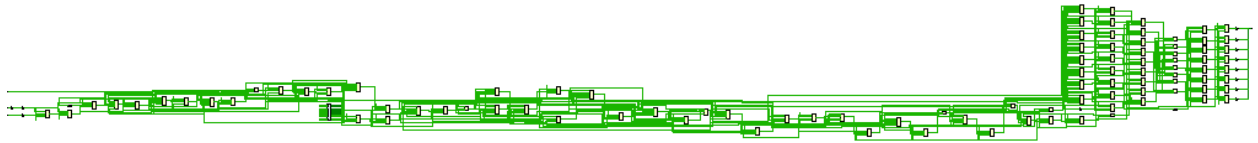


Figure 19. Flattened design

- 3-5-11.** Click on **Report Utilization** and observe that the hierarchical utilization is no longer available. Also note that the number of **Slice Registers** is **56**.

Note: Number of Slice Registers is **57** for Zybo.

3-6. Write the checkpoint in order to analyze the results without going through the actual synthesis process.

- 3-6-1.** Select **File > Write Checkpoint...** to save the processed design so it can be opened later for further analysis.
- 3-6-2.** A dialog box will appear showing the default name of the file (checkpoint_2.dcp) in the current project directory.
- 3-6-3.** Click **OK**.
- 3-6-4.** Close the project by selecting **File > Close Project**.

Read the Checkpoints

Step 4

5-1. Read the previously saved checkpoint (checkpoint_1) in order to analyze the results without going through the actual synthesis process.

5-1-1. Select **File > Open Checkpoint...** at the *Getting Started* screen.

5-1-2. Browse to <2016_2_ZYNQ_labs>\lab2 and select **checkpoint_1**.

5-1-3. Click **OK**.

5-1-4. If the schematic isn't open by default, in the netlist tab, select the **U0(uart_led)**, right-click and select **Schematic**.

You will see the hierarchical blocks. You can double-click on any of the first-level block and see the underlying blocks. You can also select any lower-level block in the netlist tab, right-click and select Schematic to see the corresponding level design.

5-1-5. In the netlist tab, select the top-level instance, **U0(uart_led)**, right-click and select **Show Hierarchy**.

You will see how the blocks are hierarchically connected.

5-1-6. Select **Tools > Timing > Report Timing Summary** and click **OK** to see the report you saw previously.

5-1-7. Select **Tools > Report > Report Utilization...** and click **OK** to see the utilization report you saw previously.

5-1-8. Select **File > Open Checkpoint**, browse to <2016_2_ZYNQ_labs>\lab2 and select **checkpoint_2**.

5-1-9. Click **No** to keep the Checkpoint_1 open.

This will invoke second Vivado GUI.

5-1-10. If the schematic isn't open by default, in the netlist tab, select the top-level instance, **uart_top**, right-click and select **Schematic**.

You will see the flattened design.

5-1-11. You can generate the desired reports on this checkpoint as you wish.

5-1-12. Close the **Vivado** program by selecting **File > Exit** and click **OK**.

Conclusion

In this lab you applied the timing constraints and synthesized the design. You viewed various post-synthesis reports. You wrote checkpoints and read it back to perform the analysis you were doing during the design flow. You saw the effect of changing synthesis settings.

Answers

1. Look through the table and find the number used of each of the following:

FF:	<u>88(87 for zybo)</u>
LUT:	<u>56(57 for zybo)</u>
I/O:	<u>11</u>
BUFG:	<u>1</u>

2. Look through the report and find the number used of each of the following:

FF:	<u>88(87 for zybo)</u>
LUT:	<u>56(57 for zybo)</u>
I/O:	<u>11</u>
BUFG:	<u>1</u>

3. From the power report, find the % power consumption used by each of the following (ZedBoard):

Clocks:	<u>1%</u>
Signals:	<u>1%</u>
Logic:	<u>1%</u>
I/O:	<u>1%</u>
PS7:	<u>96%</u>

For the Zybo:

Clocks:	<u>1%</u>
Signals:	<u>1%</u>
Logic:	<u>1%</u>
I/O:	<u>1%</u>
PS7:	<u>96%</u>