Laboratory Assignment 2 – Synthesizing and Implementing a RTL Design

Introduction

This lab shows you the synthesis and implementation process and the effect of changing settings. You will analyze the design and the generated reports; perform static timing analysis; implement the design with the default settings; and generate a bitstream. Then you will open a hardware session and program the FPGA. You will use on-board UART of the Basys3 to validate your design.

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 and analyze the output
- Change the synthesis settings and see their effects on the generated output
- Implement the design, generate various reports and analyze the results
- Generate bitstream and verify the functionality in hardware

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**.

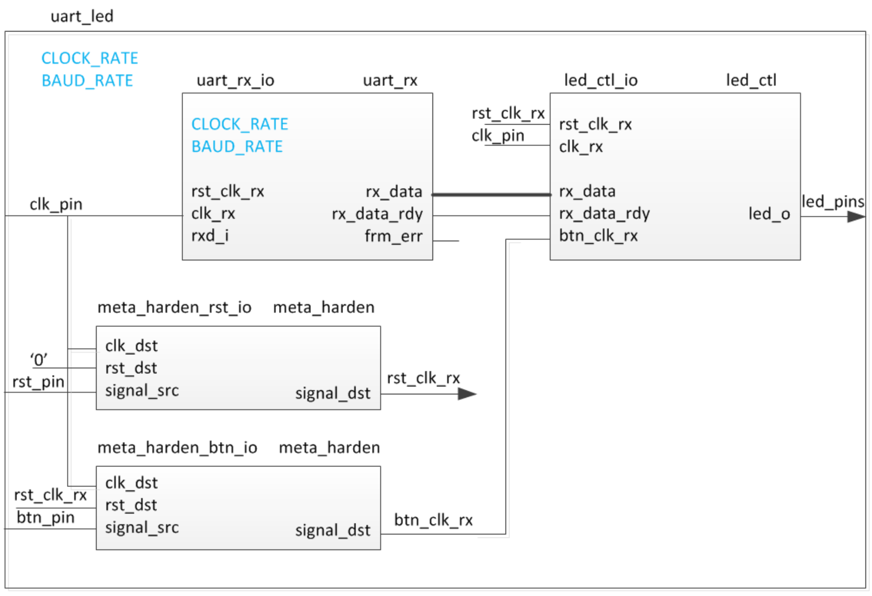
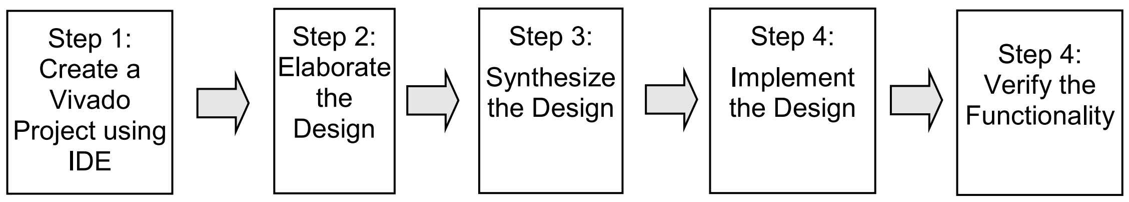


Figure 1. The Completed Design

General Flow


Step 1: Create a Vivado Project using IDE
Step 2: Elaborate the Design
Step 3: Synthesize the Design
Step 4: Implement the Design
Step 4: Verify the Functionality

In the instructions below;
{***sources***} refers to: C:\Users\MEE_CE\sources
{***labs***} refers to : C:\ Users\MEE_CE\labs

Create a Vivado Project using IDE — Step 1

1-1. Launch Vivado and create a project targeting XC7A35TCPG236-1 (Basys3) and using the Verilog HDL.

1-1-1. Start downloading the source files provided (Moodle) and save them in {sources}/lab2

1-1-2. Open Vivado and 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 **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 **{sources}\lab2** directory, select all the Verilog files *(led_ctl.v, meta_harden.v, uart_baud_gen.v, uart_led.v, uart_rx.v, and uart_rx_ctl.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 Cons*traints form.

1-1-9. Click on the **Green Plus** button, then **Add Files…** and browse to the **{sources}\lab2** directory (if necessary), select *uart_led_timing.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.

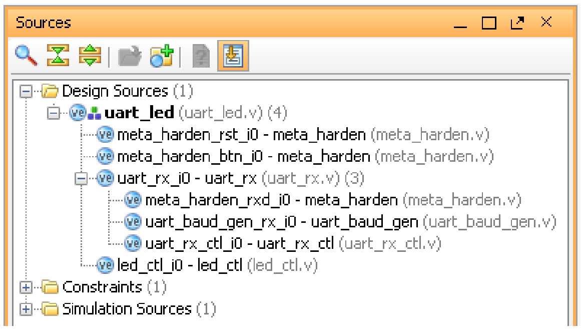
1-1-11. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section, select **XC7A35TCPG236-1** (for the Basys3) part. Using the **Boards** option, you may select the **Basys3**.

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

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

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.



```
Sources
Design Sources (1)
  uart_led (uart_led.v) (4)
    meta_harden_rst_i0 - meta_harden (meta_harden.v)
    meta_harden_btn_i0 - meta_harden (meta_harden.v)
    uart_rx_i0 - uart_rx (uart_rx.v) (3)
      meta_harden_rxd_i0 - meta_harden (meta_harden.v)
      uart_baud_gen_rx_i0 - uart_baud_gen (uart_baud_gen.v)
      uart_rx_ctl_i0 - uart_rx_ctl (uart_rx_ctl.v)
    led_ctl_i0 - led_ctl (led_ctl.v)
Constraints (1)
Simulation Sources (1)
```

Figure 2. Opening the source file

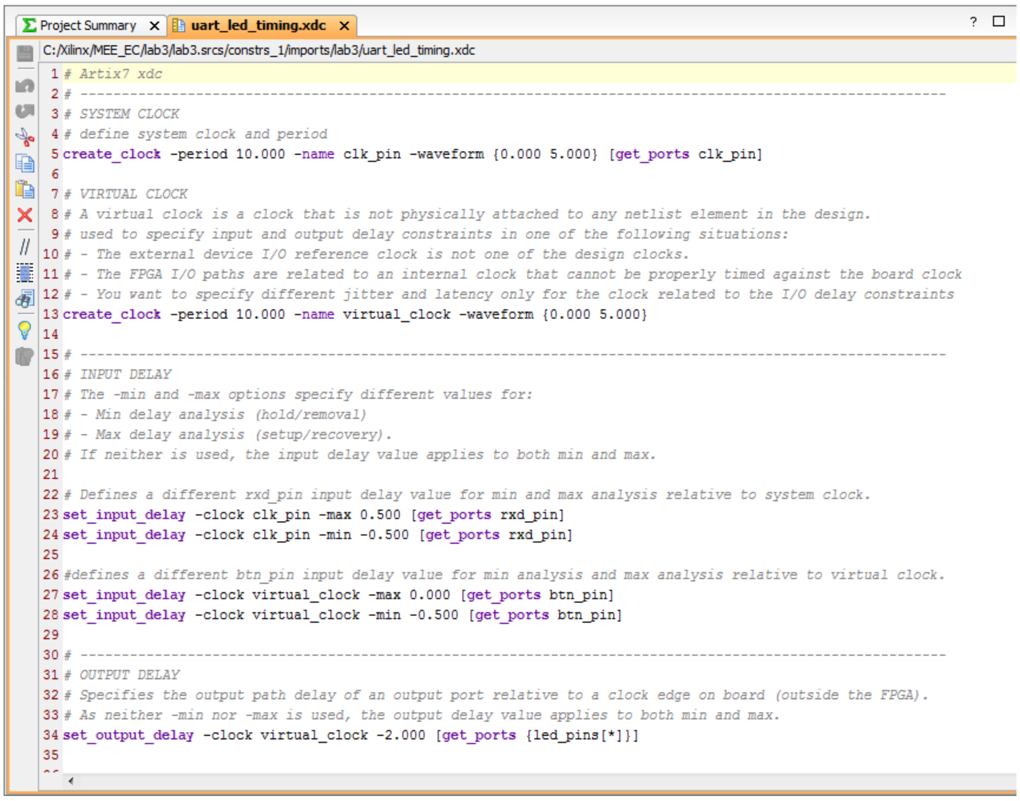
1-2-2. Double-click on the **uart_led** entry to view its content.

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.

1-2-3. Expand **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.xdc source and analyze the content.

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



C:/Xilinx/MEE_EC/lab3/lab3.srcs/constrs_1/imports/lab3/uart_led_timing.xdc

```
# Artix7 xdc
# ----------------------------------------------------------------------------------------------------------
# SYSTEM CLOCK
# define system clock and period
create_clock -period 10.000 -name clk_pin -waveform {0.000 5.000} [get_ports clk_pin]

# VIRTUAL CLOCK
# A virtual clock is a clock that is not physically attached to any netlist element in the design.
# used to specify input and output delay constraints in one of the following situations:
# - The external device I/O reference clock is not one of the design clocks.
# - The FPGA I/O paths are related to an internal clock that cannot be properly timed against the board clock
# - You want to specify different jitter and latency only for the clock related to the I/O delay constraints
create_clock -period 10.000 -name virtual_clock -waveform {0.000 5.000}

# ----------------------------------------------------------------------------------------------------------
# INPUT DELAY
# The -min and -max options specify different values for:
# - Min delay analysis (hold/removal)
# - Max delay analysis (setup/recovery).
# If neither is used, the input delay value applies to both min and max.

# Defines a different rxd_pin input delay value for min and max analysis relative to system clock.
set_input_delay -clock clk_pin -max 0.500 [get_ports rxd_pin]
set_input_delay -clock clk_pin -min -0.500 [get_ports rxd_pin]

#defines a different btn_pin input delay value for min analysis and max analysis relative to virtual clock.
set_input_delay -clock virtual_clock -max 0.000 [get_ports btn_pin]
set_input_delay -clock virtual_clock -min -0.500 [get_ports btn_pin]

# ----------------------------------------------------------------------------------------------------------
# OUTPUT DELAY
# Specifies the output path delay of an output port relative to a clock edge on board (outside the FPGA).
# As neither -min nor -max is used, the output delay value applies to both min and max.
set_output_delay -clock virtual_clock -2.000 [get_ports {led_pins[*]}]

```

Figure 3. Timing constraints

Line 5 creates the period constraint of 10 ns with a duty cycle of 50%. Line13 creates a virtual clock of 12 ns. This clock can be viewed as the upstream device board clock (the upstream device generates its output with respect to the board clock). The rxd_pin is constrained with respect to the system clock (lines 23, and 24) whereas the btn_pin is constrained with respect to the upstream clock (lines 27, 28). The led_pins are constrained with respect to the upstream clock as the downstream device may be using it.

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**.

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

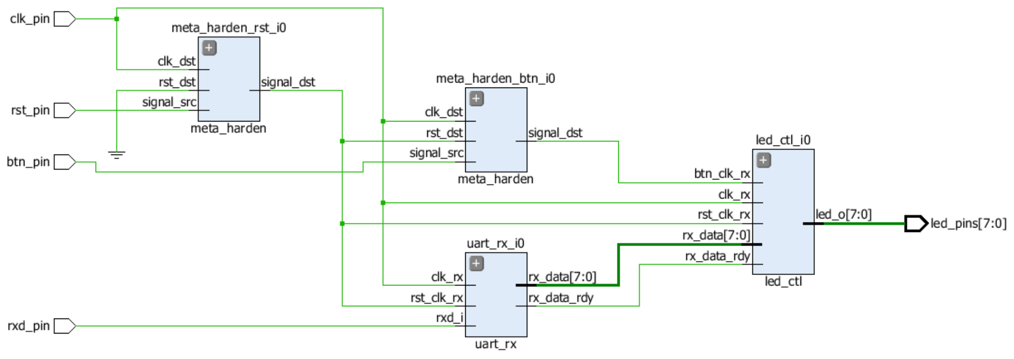


Figure 4. A logic view of the design

You will see four components at the top-level, 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.

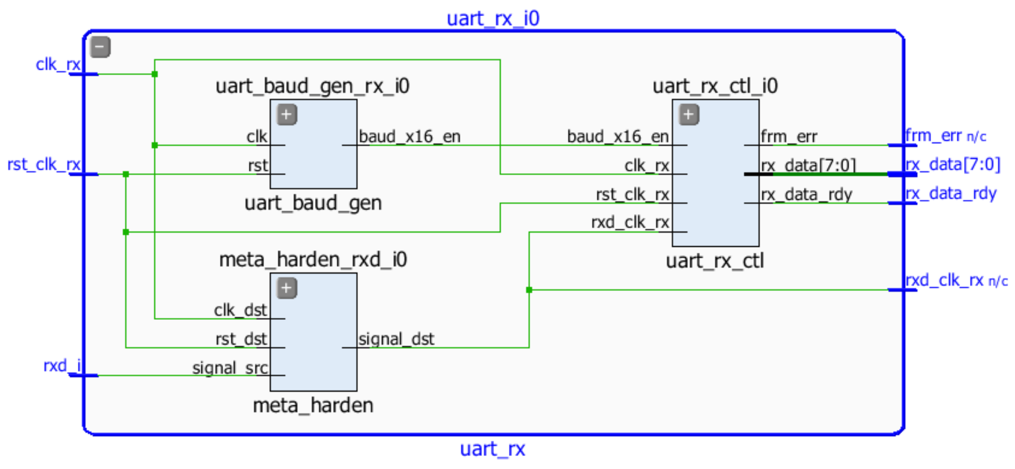
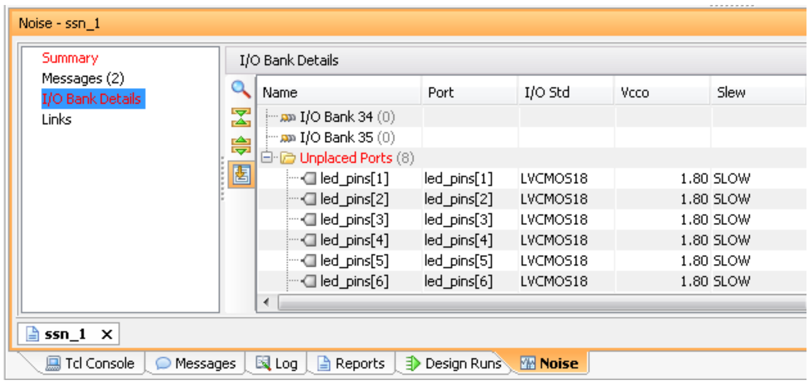


Figure 5. Lower level components of the uart_rx_i0 module

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.



Noise - ssn_1

Summary
Messages (2)
I/O Bank Details
Links

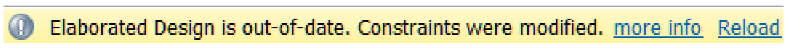
I/O Bank Details

Name	Port	I/O Std	Vcco	Slew
I/O Bank 34 (0)				
I/O Bank 35 (0)				
Unplaced Ports (8)				
led_pins[1]	led_pins[1]	LVCMOS18	1.80	SLOW
led_pins[2]	led_pins[2]	LVCMOS18	1.80	SLOW
led_pins[3]	led_pins[3]	LVCMOS18	1.80	SLOW
led_pins[4]	led_pins[4]	LVCMOS18	1.80	SLOW
led_pins[5]	led_pins[5]	LVCMOS18	1.80	SLOW
led_pins[6]	led_pins[6]	LVCMOS18	1.80	SLOW

Figure 6. 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 **{sources}\lab2** directory, select the **uart_led_pins_basys3. xdc** file, 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 re-load 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.

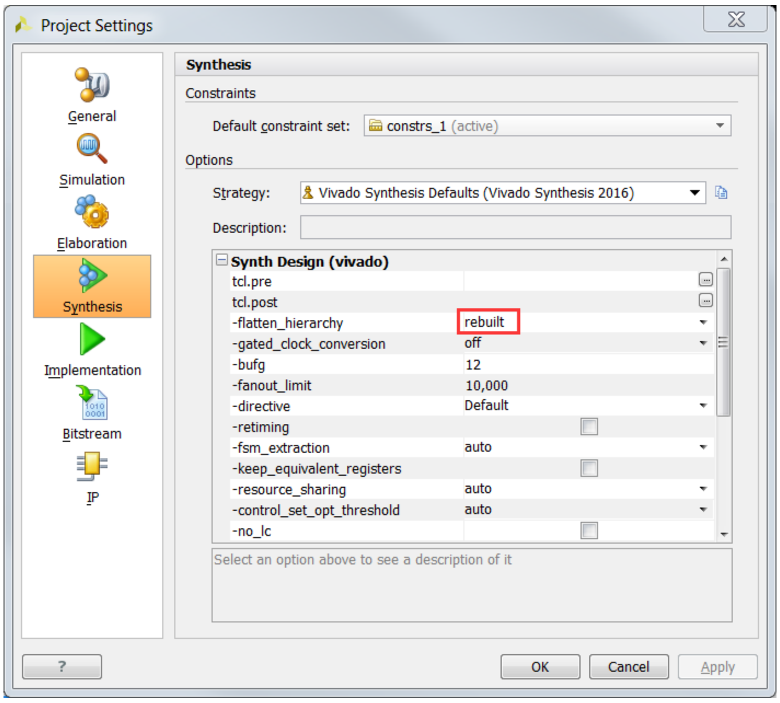
3-1-1. Click on the **Synthesis Settings** in the *Flow Navigator* pane.

3-1-2. Make sure that the *flatten_hierarchy* is set to **rebuilt (Figure 7)**, which allows the design hierarchy to be flattened for synthesis and then rebuilt, which is more useful for design analysis because many logical references will be maintained.

3-1-3. Click **OK**. A Create New Run dialog box will appear asking you if a new run should be created. Click **Yes** and then **OK** to create the new run with **synth_2** name.

3-1-4. Click on **Run Synthesis** under the *Synthesis* tasks of the *Flow Navigator* pane.

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



Synthesis

Constraints

Default constraint set: constrs_1 (active)

Options

Strategy: Vivado Synthesis Defaults (Vivado Synthesis 2016)

Description:

Synth Design (vivado)	
tcl.pre	
tcl.post	
-flatten_hierarchy	rebuilt
-gated_clock_conversion	off
-bufg	12
-fanout_limit	10,000
-directive	Default
-retiming	☐
-fsm_extraction	auto
-keep_equivalent_registers	☐
-resource_sharing	auto
-control_set_opt_threshold	auto
-no_lc	☐

Select an option above to see a description of it

Figure 7. Setting hierarchy to rebuilt

3-1-5. 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.

3-1-6. 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 Σ.

3-1-7. 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:	______________________

3-1-8. 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.

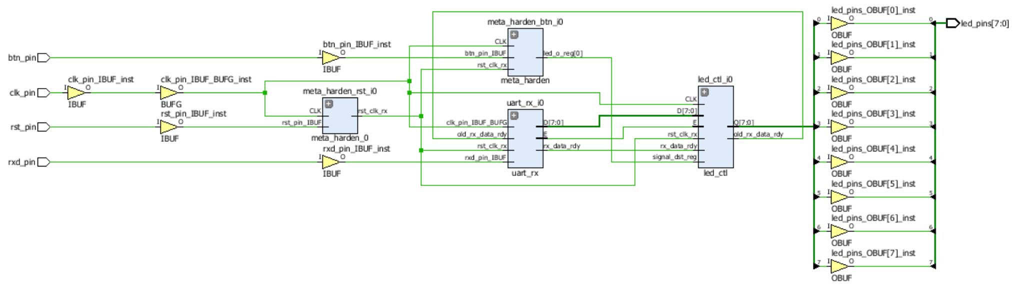


Figure 8. Synthesized design's schematic view

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.

3-1-9. Double-click on the **uart_rx_i0** instance in the schematic view to see the underlying instances.

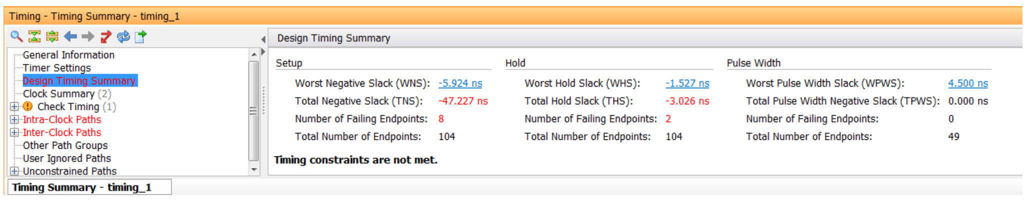
3-1-10. Select the **uart_baud_gen_rx_i0** instance, right-click, and select *Go To Source*. Notice that uart_baud_gen_rx_i0 is highlighted. Also notice that the CLOCK_RATE and BAUD_RATE parameters are passed to the module being called.

3-1-11. Double-click on the **meta_harden_rxd_io** instance to see how the synchronization circuit is being implemented using two FFs. This synchronization is necessary to reduce the likelihood of meta-stability.

3-2. 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. Leave all the settings unchanged, and click **OK** to generate a default timing report, *timing_1.*


Timing - Timing Summary - timing_1

General Information
Timer Settings
Design Timing Summary
Clock Summary (2)
Check Timing (1)
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):	-5.924 ns	Worst Hold Slack (WHS):	-1.527 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	-47.227 ns	Total Hold Slack (THS):	-3.026 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	8	Number of Failing Endpoints:	2	Number of Failing Endpoints:	0
Total Number of Endpoints:	104	Total Number of Endpoints:	104	Total Number of Endpoints:	49

Timing constraints are not met.

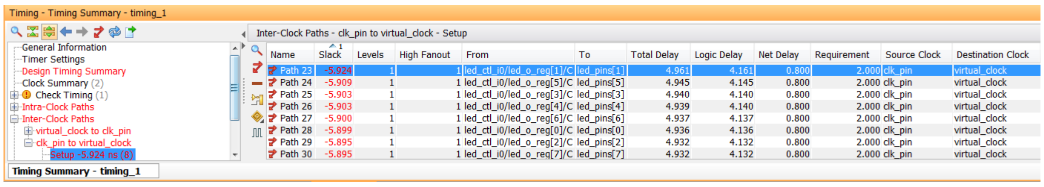
Timing Summary - timing_1

Figure 9. Timing report for the Basys3

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 **Worst Negative Slack** (WNS) link and see the 8 failing paths.


Timing - Timing Summary - timing_1

General Information
Timer Settings
Design Timing Summary
Clock Summary (2)
Check Timing (1)
Intra-Clock Paths
Inter-Clock Paths
virtual_clock to clk_pin
clk_pin to virtual_clock
Setup -5.924 ns (8)

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 23	-5.924	1	1	led_ctl_i0/led_o_reg[1]/C	led_pins[1]	4.961	4.161	0.800	2.000	clk_pin	virtual_clock
Path 24	-5.909	1	1	led_ctl_i0/led_o_reg[5]/C	led_pins[5]	4.945	4.145	0.800	2.000	clk_pin	virtual_clock
Path 25	-5.903	1	1	led_ctl_i0/led_o_reg[3]/C	led_pins[3]	4.940	4.140	0.800	2.000	clk_pin	virtual_clock
Path 26	-5.903	1	1	led_ctl_i0/led_o_reg[4]/C	led_pins[4]	4.939	4.140	0.800	2.000	clk_pin	virtual_clock
Path 27	-5.900	1	1	led_ctl_i0/led_o_reg[6]/C	led_pins[6]	4.937	4.137	0.800	2.000	clk_pin	virtual_clock
Path 28	-5.899	1	1	led_ctl_i0/led_o_reg[0]/C	led_pins[0]	4.936	4.136	0.800	2.000	clk_pin	virtual_clock
Path 29	-5.895	1	1	led_ctl_i0/led_o_reg[2]/C	led_pins[2]	4.932	4.132	0.800	2.000	clk_pin	virtual_clock
Path 30	-5.895	1	1	led_ctl_i0/led_o_reg[7]/C	led_pins[7]	4.932	4.132	0.800	2.000	clk_pin	virtual_clock

Timing Summary - timing_1

Figure 10. The 8 failing paths for the Basys3

3-2-4. Double-click on the **Path 23** to see how the path is made. The path report shows four sections: (i) Summary, (ii) Source Clock Path, (iii) Data Path, and (iv) Destination Clock Path.

3-2-5. Select Path 23 in the timing summary panel, or the Path summary view, right-click, and select **Schematic**. The schematic for the output data path will be displayed.

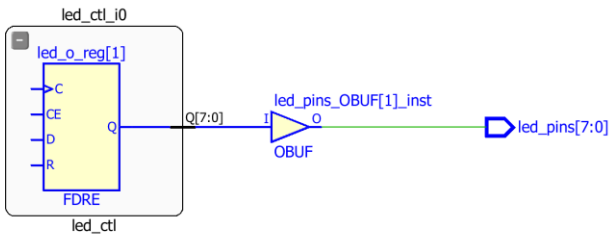


Figure 11. The output data path

3-2-6. In order to see how the Source Clock Path is made up in schematic form, double-click on left end of the C pin of the FDRE in the schematic. This will show the net between the BUFG and C port of the FDRE.

3-2-7. Similarly, double-click on the left end of the BUFG to see the path between IBUF and BUFG.

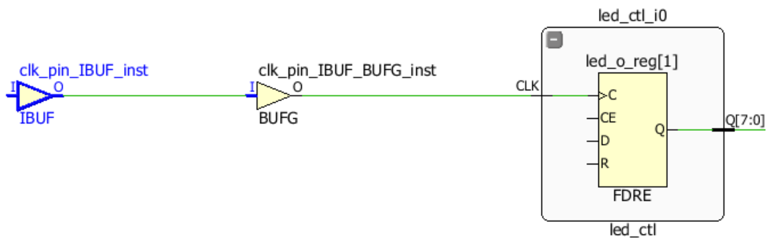


Figure 12. Source to clock port of the FDRE

3-2-8. Finally, double-click on the input pin of IBUF to see the path between the clock input pin and the IBUF.

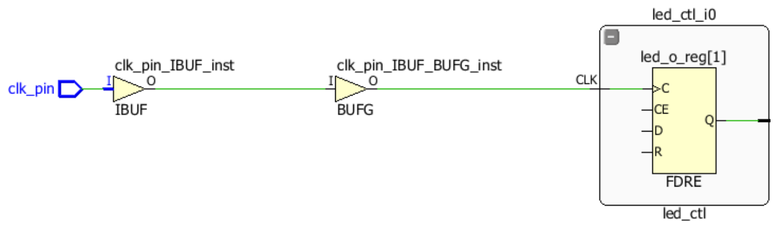
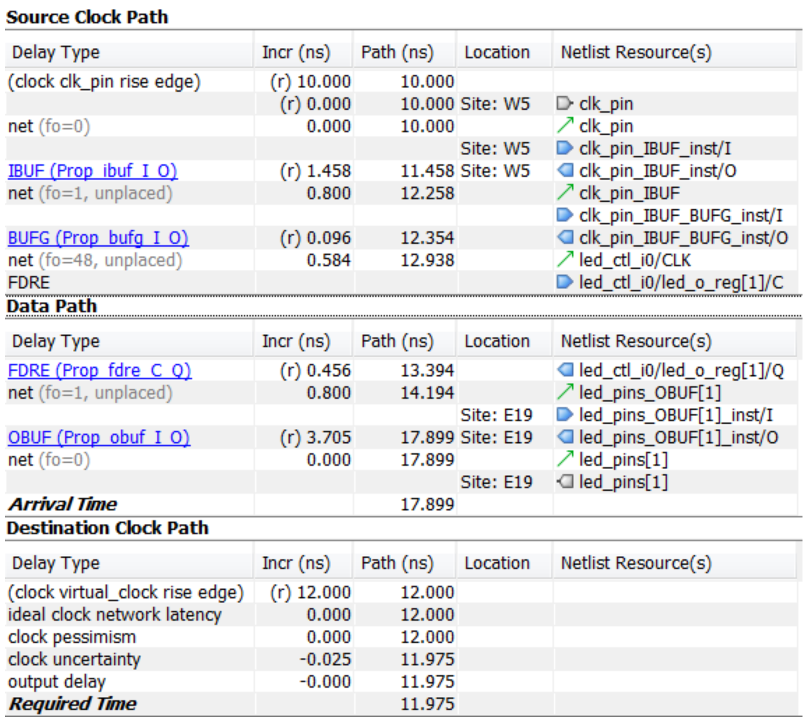


Figure 13. The schematic view of the source clock path

This corresponds to the Source Clock Path in the timing report.



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: W5	clk_pin
net (fo=0)	0.000	10.000		clk_pin
			Site: W5	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.458	11.458	Site: W5	clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.800	12.258		clk_pin_IBUF
				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.096	12.354		clk_pin_IBUF_BUFG_inst/O
net (fo=48, unplaced)	0.584	12.938		led_ctl_i0/CLK
FDRE				led_ctl_i0/led_o_reg[1]/C

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_Q)	(r) 0.456	13.394		led_ctl_i0/led_o_reg[1]/Q
net (fo=1, unplaced)	0.800	14.194		led_pins_OBUF[1]
			Site: E19	led_pins_OBUF[1]_inst/I
OBUF (Prop_obuf_I_O)	(r) 3.705	17.899	Site: E19	led_pins_OBUF[1]_inst/O
net (fo=0)	0.000	17.899		led_pins[1]
			Site: E19	led_pins[1]
Arrival Time		17.899		

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock virtual_clock rise edge)	(r) 12.000	12.000		
ideal clock network latency	0.000	12.000		
clock pessimism	0.000	12.000		
clock uncertainty	-0.025	11.975		
output delay	-0.000	11.975		
Required Time		11.975		

Figure 14. Worst failing path for the Basys3

The rise time instant for the input clock is 10ns. Considering the delay introduced by the input buffer, routing nets and global buffer is (1.458ns+0.8ns+0.096ns+0.584ns), the rising edge arrives the flip-flop (FDRE) at 12.938ns. So the clock path skew is 2.938ns.

The time delay from data (D) to output (Q) in the flip-flop is 0.456ns. So the data arrives at the FDRE output pin at 13.394ns. Adding the net delay and the OBUF operation delay (0.8ns+3.705ns) data arrives at the output pin (led_pin) at 17.899ns.

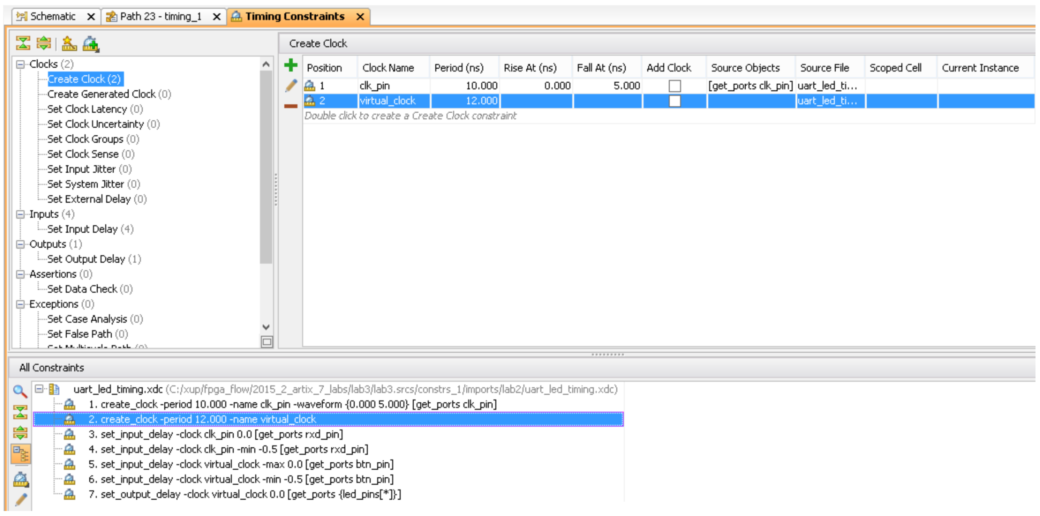
We are assuming that this led_pin will be clocked externally by a board clock with 12ns period. Considering that this clock period may have an uncertainty of 0.025ns, data should be available at time 11.975ns, so that it can be captured by the rising edge of this virtual clock. However, it will arrive only at 17.899ns, which corresponds to a slack of -5.924ns (11.975ns-17.899ns).

Note that this is an estimate only. The nets are specified as unplaced and have all been allocated default values (0.800 ns). No actual routing delays are considered.

3-3. Change the design constraint to constrain the virtual clock period to 10ns. Re-synthesize the design and analyze the results.

3-3-1. Click **Edit Timing Constraints** under the Synthesized Design.

The Timing Constraints GUI will appear, showing the design has two create clocks, four inputs, and one output constraints. It also shows the constraints in the text form in the All Constraints section.



Position	Clock Name	Period (ns)	Rise At (ns)	Fall At (ns)	Add Clock	Source Objects	Source File	Scoped Cell	Current Instance
1	clk_pin	10.000	0.000	5.000	☐	[get_ports clk_pin]	uart_led_ti...		
2	virtual_clock	12.000			☐		uart_led_ti...		

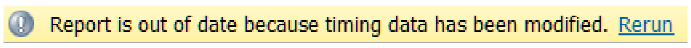
Double click to create a Create Clock constraint

```
uart_led_timing.xdc (C:/xup/fpga_flow/2015_2_artix_7_labs/lab3/lab3.srcs/constrs_1/imports/lab2/uart_led_timing.xdc)
    1. create_clock -period 10.000 -name clk_pin -waveform {0.000 5.000} [get_ports clk_pin]
    2. create_clock -period 12.000 -name virtual_clock
    3. set_input_delay -clock clk_pin 0.0 [get_ports rxd_pin]
    4. set_input_delay -clock clk_pin -min -0.5 [get_ports rxd_pin]
    5. set_input_delay -clock virtual_clock -max 0.0 [get_ports btn_pin]
    6. set_input_delay -clock virtual_clock -min -0.5 [get_ports btn_pin]
    7. set_output_delay -clock virtual_clock 0.0 [get_ports {led_pins[*]}]
```

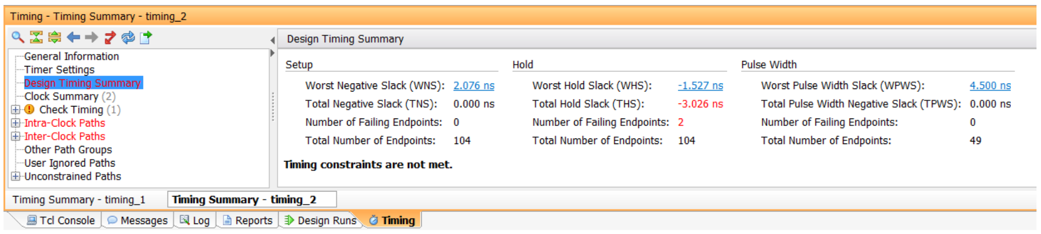
Figure 15. Timing Constraints showing 12 ns Virtual Clock period defined

3-3-2. Click in the Period cell of the virtual_clock and change the period from 12 to 10

3-3-3. Click **Apply**. Note that since the timing constraint has changed, a warning message in the console pane is displayed to rerun the report.


Report is out of date because timing data has been modified. Rerun

3-3-4. Click on **Rerun**. Notice that setup timing violations are gone. However, there are still 2 failing paths for the Hold.



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.076 ns	Worst Hold Slack (WHS):	-1.527 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	-3.026 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	2	Number of Failing Endpoints:	0
Total Number of Endpoints:	104	Total Number of Endpoints:	104	Total Number of Endpoints:	49

Timing constraints are not met.

Figure 16. Setup timing met for the Basys3

3-3-5. Click on the **WHS** link to see the paths.

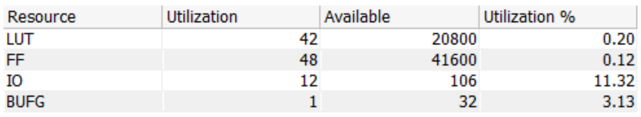
3-3-6. Double-click on the first path to see the timing compositions. Notice that the clock path delay does not include the entire clock period.

3-3-7. Select **File > Save Constraints…**

3-3-8. Click **OK** and then click **Yes** to save the synthesized design. Notice that the Synthesis Out-of-Date status is displayed on the top-right corner.

3-4. Generate the utilization and power reports.

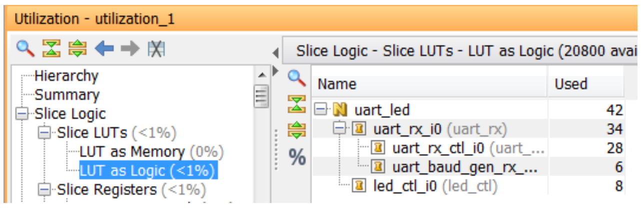
3-4-1. Click **Report Utilization** under the Synthesized Design, and click **OK** to generate the utilization report. Note how many resources are being used in this design.



Resource	Utilization	Available	Utilization %
LUT	42	20800	0.20
FF	48	41600	0.12
IO	12	106	11.32
BUFG	1	32	3.13

Figure 17. Utilization report for the Basys3

3-4-2. 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.



Utilization - utilization_1

- Hierarchy
- Summary
- Slice Logic
  - Slice LUTs (<1%)
    - LUT as Memory (0%)
    - LUT as Logic (<1%)
  - Slice Registers (<1%)

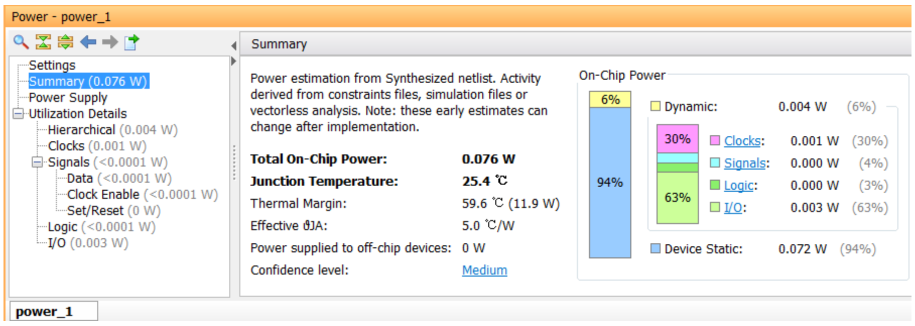
Slice Logic - Slice LUTs - LUT as Logic (20800 avai

Name	Used
uart_led	42
uart_rx_i0 (uart_rx)	34
uart_rx_ctl_i0 (uart_…	28
uart_baud_gen_rx_…	6
led_ctl_i0 (led_ctl)	8

Figure 18. Utilization of lower-level modules for the Basys3

3-4-3. Click **Report Power** under the Synthesized Design, and click **OK** to generate the estimated power consumption report using default values. From the power report, find the % power consumption used by each of the following: clocks, signals, logic, I/O.

Note that this is just an estimate as no simulation run data was provided and no accurate activity rate, or environment information was entered. You can move the mouse on the boxes which do not show the percentage to see the consumption.



Power - power_1

- Settings
- Summary (0.076 W)
- Power Supply
- Utilization Details
  - Hierarchical (0.004 W)
  - Clocks (0.001 W)
  - Signals (<0.0001 W)
    - Data (<0.0001 W)
    - Clock Enable (<0.0001 W)
    - Set/Reset (0 W)
  - Logic (<0.0001 W)
  - I/O (0.003 W)

power_1

Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.076 W**
Junction Temperature:	**25.4 ℃**
Thermal Margin:	59.6 ℃ (11.9 W)
Effective ϑJA:	5.0 ℃/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

On-Chip Power

Dynamic:	0.004 W	(6%)
Clocks:	0.001 W	(30%)
Signals:	0.000 W	(4%)
Logic:	0.000 W	(3%)
I/O:	0.003 W	(63%)
Device Static:	0.072 W	(94%)

Figure 19. Power consumption estimation for the Basys3

3-5. Write the checkpoint in order to analyze the results without going through the actual synthesis process.

3-5-1. Select **File > Write Checkpoint…** to save the processed design so it can be opened later for further analysis.

3-5-2. A dialog box will appear showing the default file name in the current project directory. Click **OK**.

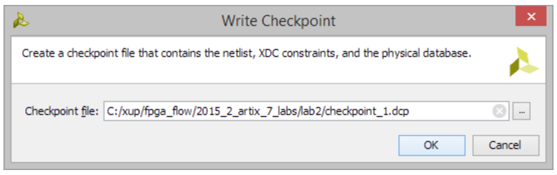

Write Checkpoint
Create a checkpoint file that contains the netlist, XDC constraints, and the physical database.
Checkpoint file: C:/xup/fpga_flow/2015_2_artix_7_labs/lab2/checkpoint_1.dcp
OK
Cancel

Figure 20. Writing checkpoint

3-6. Change the synthesis settings to flatten the design. Re-synthesize the design and analyze the results.

3-6-1. Click on the **Project Settings** under the *Project Manager*, and select **Synthesis**.

3-6-2. Click on the **flatten_hierarchy** drop-down button and select **full** to flatten the design.

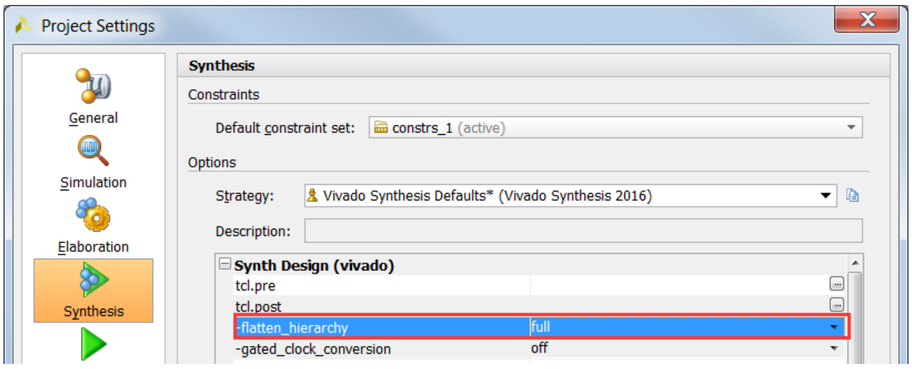


Figure 21. Selecting flatten hierarchy option

3-6-3. Click **OK**.

3-6-4. A Create New Run dialog box will appear asking you whether you want to create a new run since the settings have been changed. Click **Yes**.

3-6-5. Change the name from **synth_2** to **synth_flatten** and click **OK**.

3-6-6. Click **Run Synthesis** to synthesize the design.

3-6-7. 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**.

3-6-8. Click **OK** to open the synthesized design when synthesis process is completed.

3-6-9. 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 22). Notice that the design is completely flattened.

3-6-10. Click on **Report Utilization** and observe that the hierarchical utilization is no longer available. Also note that the number of elements is the same (there was no opportunity for optimization).

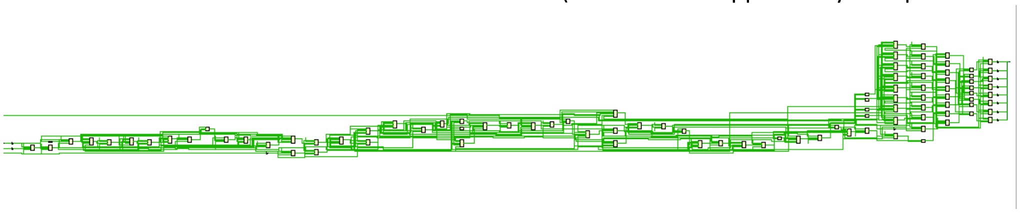


Figure 22. Flattened design

3-7. Write the checkpoint in order to analyze the results without going through the actual synthesis process.

3-7-1. Select **File > Write Checkpoint…** to save the processed design so it can be opened later for further analysis.

3-7-2. A dialog box will appear showing the default name of the file (checkpoint_2.dcp) in the current project directory. Click **OK**.

3-7-3. Close the project by selecting **File > Close Project**.

Read the Checkpoints Step 4

4-1. Read the previously saved checkpoint (checkpoint_1) in order to analyze the results without going through the actual synthesis process.

4-1-1. Select **File > Open Checkpoint…** at the *Getting Started* screen.

4-1-2. Browse to **{labs}\lab2** and select **checkpoint_1.**

4-1-3. Click **OK**.

4-1-4. If the schematic isn't open by default, in the netlist tab, select the top-level instance, **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.

4-1-5. In the netlist tab, select the top-level instance, **uart_led**, right-click and select **Show Hierarchy.** You will see how the blocks are hierarchically connected.

4-1-6. Select **Tools > Timing > Report Timing Summary** and click **OK** to see the report you saw previously.

4-1-7. Select **Tools > Report > Report Utilization…** and click **OK** to see the utilization report you saw previously

4-1-8. Select **File > Open Checkpoint**, browse to **{labs}\lab2** and select **checkpoint_2.**

4-1-9. Click **No** to keep the Checkpoint_1 open. This will invoke second Vivado GUI.

4-1-10. If the schematic isn't open by default, in the netlist tab, select the top-level instance, **uart_led**, right-click and select **Schematic**. You will see the flattened design.

4-1-11. You can generate the desired reports on this checkpoint as you wish.

4-1-12. Close the **Vivado** program by selecting **File > Exit** and click **OK**.

Implement the Design Step 5

5-1. Run the implementation after saving the synthesis run. Perform the timing analysis.

5-1-1. In the Design Runs tab, right-click on the synth_2 and select **Reset Runs**. Make sure the generated files are deleted. Click **Reset**.

5-1-2. Click the **Close Design** link in the status bar. If prompted, do not save anything.

5-1-3. Click on the **Run Implementation** in the *Flow Navigator* pane.

5-1-4. Click **OK** when prompted to run the synthesis first before running the implementation process.

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

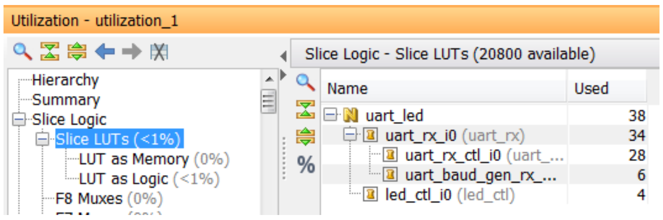
5-1-5. Select the *Open Implemented Design* option and click **OK**.

5-2. View the amount of FPGA resources consumed by the design using Report Utilization.

5-2-1. In the *Flow Navigator* pane, select **Open Implemented Design** > **Report Utilization**. The Report Utilization dialog box opens.

5-2-2. Click **OK**. The utilization report is displayed at the bottom of the Vivado IDE. You can select any of the resources on the left to view its corresponding utilization.

5-2-3. Select Slice LUTs to view how much and which module consumes the resource. Note that the utilization report for LUTs in lower than the estimate given after synthesis.


Utilization - utilization_1

- Hierarchy
- Summary
- Slice Logic
  - Slice LUTs (<1%)
    - LUT as Memory (0%)
    - LUT as Logic (<1%)
  - F8 Muxes (0%)

Slice Logic - Slice LUTs (20800 available)

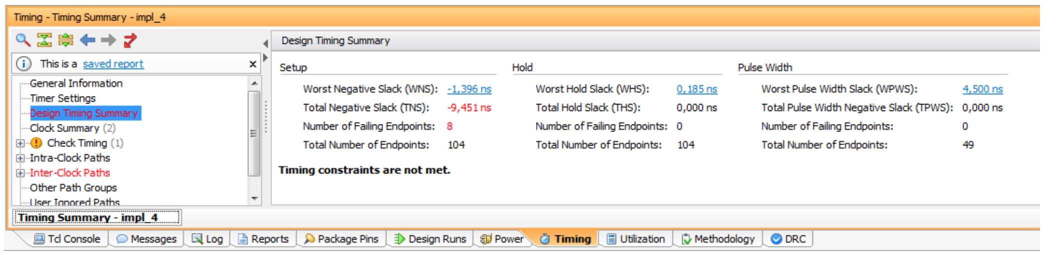
Name	Used
uart_led	38
uart_rx_i0 (uart_rx)	34
uart_rx_ctl_i0 (uart_...	28
uart_baud_gen_rx_...	6
led_ctl_i0 (led_ctl)	4

Figure 23. Resource utilization for the Basys3

5-3. Generate a timing summary report.

5-3-1. In the Flow Navigator, under Implementation > Implemented Design, click **Report Timing Summary**. The Report Timing Summary dialog box opens.

5-3-2. Leave all the settings unchanged and click **OK** to generate the report.


Timing - Timing Summary - impl_4

This is a saved report

- General Information
- Timer Settings
- Design Timing Summary
- Clock Summary (2)
- Check Timing (1)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths

Timing Summary - impl_4

Design Timing Summary

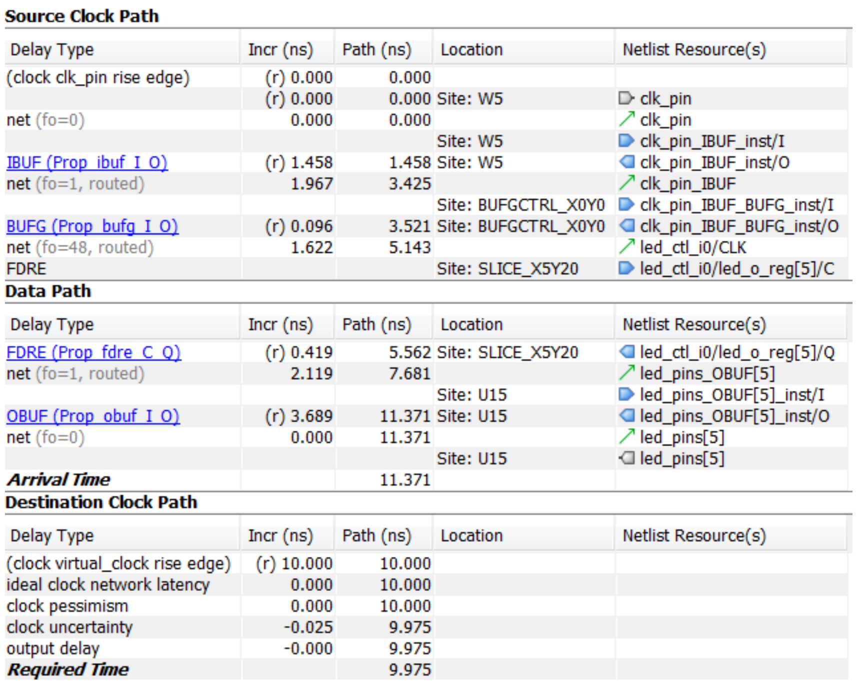
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-1,396 ns	Worst Hold Slack (WHS):	0,185 ns	Worst Pulse Width Slack (WPWS):	4,500 ns
Total Negative Slack (TNS):	-9,451 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	8	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	104	Total Number of Endpoints:	104	Total Number of Endpoints:	49

Timing constraints are not met.

Figure 24. The timing summary report showing timing violations for the Basys3

5-3-3. Click on the WNS link to see a detailed report to determine the failing path entries.

5-3-4. Double-click on the first failing path to see why it is failing.



Source 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
			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, routed)	1.967	3.425		clk_pin_IBUF
			Site: BUFGCTRL_X0Y0	clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.096	3.521	Site: BUFGCTRL_X0Y0	clk_pin_IBUF_BUFG_inst/O
net (fo=48, routed)	1.622	5.143		led_ctl_i0/CLK
FDRE			Site: SLICE_X5Y20	led_ctl_i0/led_o_reg[5]/C

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_Q)	(r) 0.419	5.562	Site: SLICE_X5Y20	led_ctl_i0/led_o_reg[5]/Q
net (fo=1, routed)	2.119	7.681		led_pins_OBUF[5]
			Site: U15	led_pins_OBUF[5]_inst/I
OBUF (Prop_obuf_I_O)	(r) 3.689	11.371	Site: U15	led_pins_OBUF[5]_inst/O
net (fo=0)	0.000	11.371		led_pins[5]
			Site: U15	led_pins[5]
Arrival Time		11.371		

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock virtual_clock rise edge)	(r) 10.000	10.000		
ideal clock network latency	0.000	10.000		
clock pessimism	0.000	10.000		
clock uncertainty	-0.025	9.975		
output delay	-0.000	9.975		
Required Time		9.975		

Figure 25. First failing path delays for the Basys3

Compared to delays from the synthesis report, the net delays are actual delays (rather than an estimated figure). The data path delay is longer than the destination clock path delay giving a negative slack (violation). The data path delay is 11.371 ns, the destination clock path is 9.975 ns and the negative slack is -1.396 ns.

At this point we can ignore this violation as the LED display change by a few nanoseconds won't be observable by human eyes. We can also change the output delay by -2 ns and make the timings meet.

5-3-5. Select **Implemented Design > Edit Timing Constraints** the *Flow Navigator* pane.

5-3-6. Select the *Set Output Delay* entry in the left pane, and change the Delay Value to **-2.000** ns.

5-3-7. Click **Apply**.

5-3-8. Click **Rerun** link to re-run the timing report. Observe that the timing violations of the Intra-clock paths are gone.

5-3-9. Expand the **Intra-Clock Paths** folder on the left, expand *clk_pin*, and select the Setup group to see the list of 10 worst case delays on the right side.

5-3-10. Double-click on the any path to see how that is made up of. Also right-click on it and select **Schematic.** Click on the **Device** tab and see the highlighted path in the view.

5-3-11. Select **Implemented Design > Report Clock Networks**.

5-3-12. Click **OK**. The Clock Networks report will be displayed in the Console pane showing two clock net entries.

5-3-13. Select *clk_pin* entry and observe the selected nets in the Device view. The clock nets are spread across multiple clock regions.

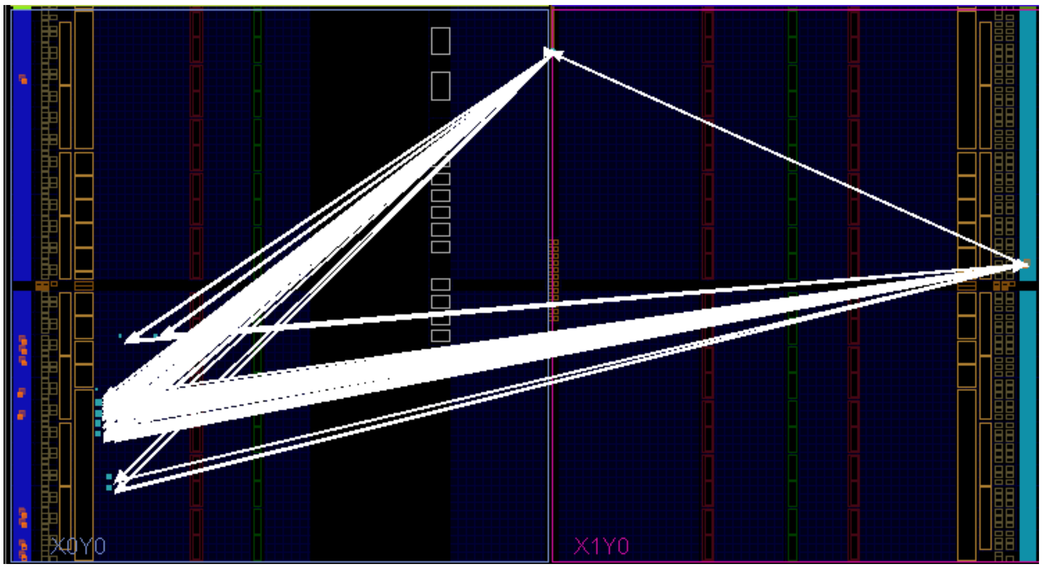


Figure 26. Clock nets for the Basys3

Generate the Bitstream — Step 6

6-1. Generate the bitstream.

6-1-1. In the Flow Navigator, under Program and Debug, click **Generate Bitstream.**

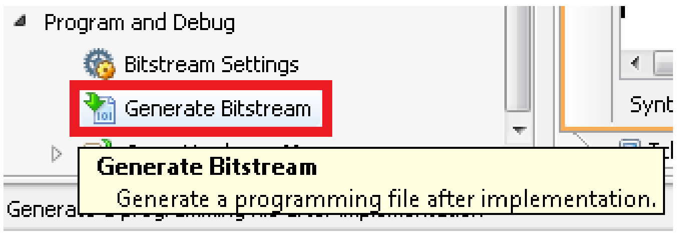


Figure 27. Generating the bitstream

6-1-2. Click **Save** to save the constraints since the timing constraints had been changed, click **OK**, and then **Yes** to reset the runs and re-run all the processes.

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

6-1-3. Click **Cancel** when the bitstream generation is completed.

Verify the Functionality — Step 7

7-1. Connect the board and power it ON. Open a hardware session, and program the FPGA.

7-1-1. Make sure that the micro-USB cable is connected to the JTAG PROG connector (next to the power supply connector). Make sure that the jumper on the board is set to select USB power (JP3 for the Nexys4 DDR and JP2 for the Basys3).

7-1-2. Select the *Open Hardware Manager* option and click **OK**.

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

7-1-3. Click on the **Open target** link, then **Auto Connect** from the dropdown menu.

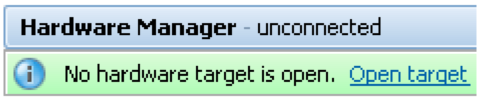

Hardware Manager - unconnected
No hardware target is open. Open target

Figure 28. Opening new hardware target

7-1-4. The Hardware Session status changes from Unconnected to the server name and the device is highlighted. Also notice that the Status indicates that it is not programmed.

7-1-5. Select the device in the *Hardware Device Properties,* and verify that the **uart_led.bit** is selected as the programming file in the General tab.

7-2. Start a terminal emulator program such as TeraTerm or HyperTerminal. Select an appropriate COM port (you can find the correct COM number using the Control Panel). Set the COM port for 115200 baud rate communication. Program the FPGA and verify the functionality.

7-2-1. Start a terminal emulator program such as TeraTerm or HyperTerminal.

7-2-2. Select an appropriate COM port (you can find the correct COM number using the Control Panel).

7-2-3. Set the COM port for 115200 baud rate communication.

7-2-4. Right-click on the FPGA entry in the Hardware window and select **Program Device…**

7-2-5. Click on the **Program** button.

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

7-2-6. Type in some characters in the terminal emulator window and see the corresponding ASCII equivalent bit pattern displayed on the LEDs.

7-2-7. Press and hold BTNU and see the the upper four bits are swapped with the lower four bits on the LEDs.

7-2-8. When satisfied, close the terminal emulator program and power OFF the board.

7-2-9. Select **File > Close Hardware Manager**. Click **OK**.

7-2-10. 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.

Also, you learned about many of the reports available to designers in the Vivado IDE. You had the opportunity to learn basic design analysis tools including the Schematic viewer, delay path properties and reports viewer, Device viewer, and selecting primitive parents. You also learned about the basic timing report options that are at your disposal. You verified the functionality in hardware by typing characters on the host machine and seeing the LED pattern changes.