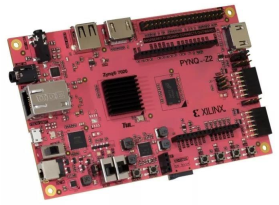
VLSI for DSP

Yuan-Ho Chen

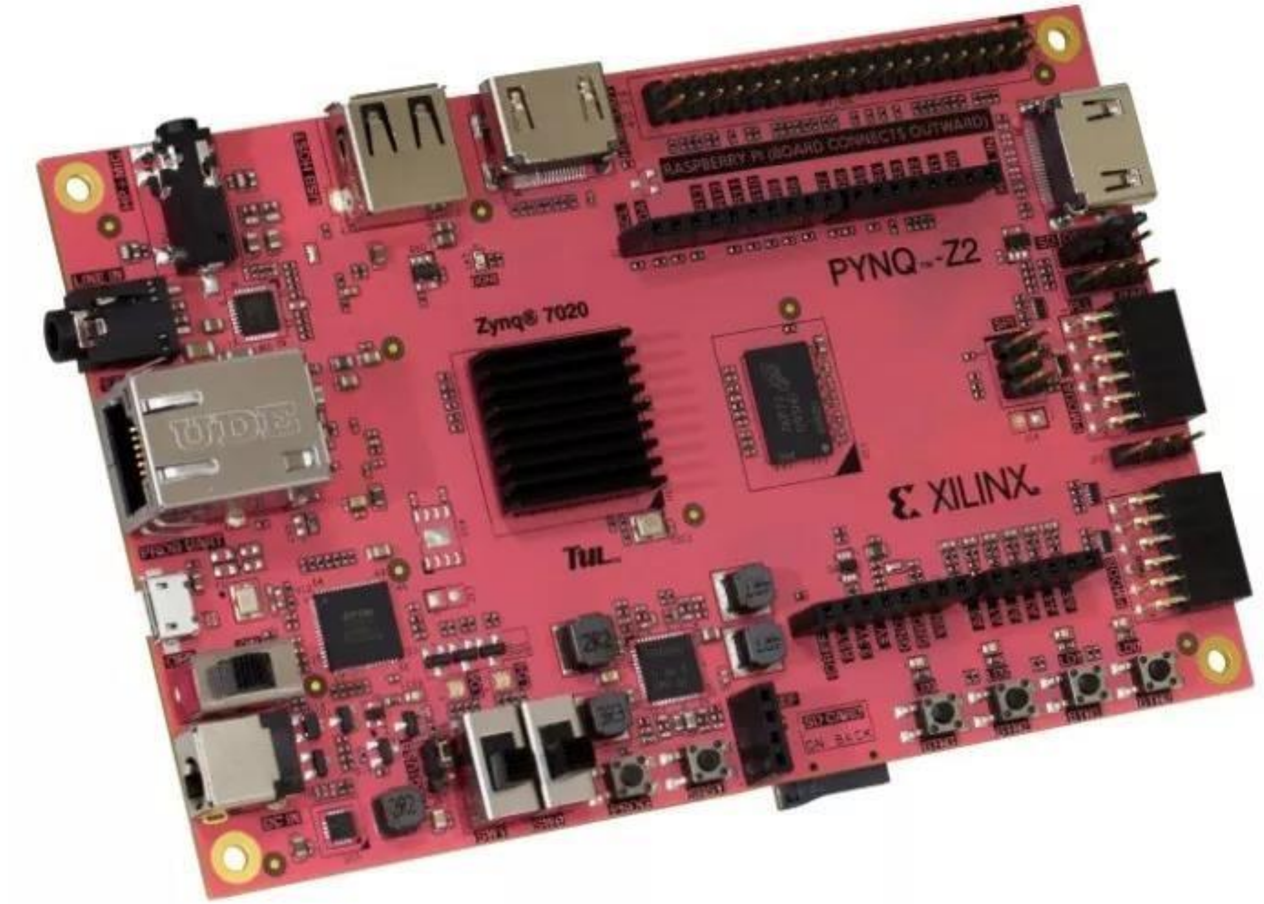
Department of Electronic Engineering

Chang Guan University

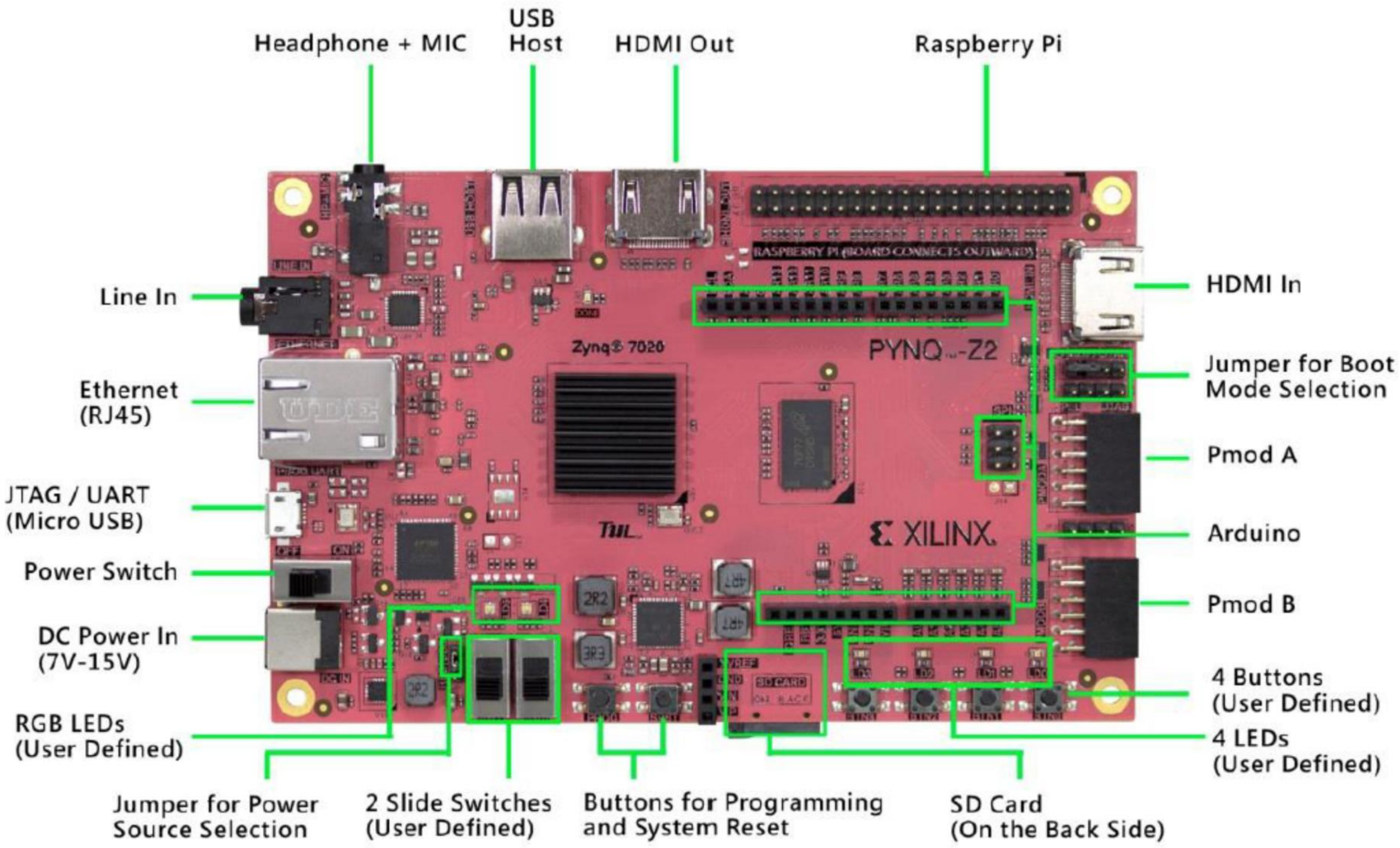
chenyh@mail.cgu.edu.tw



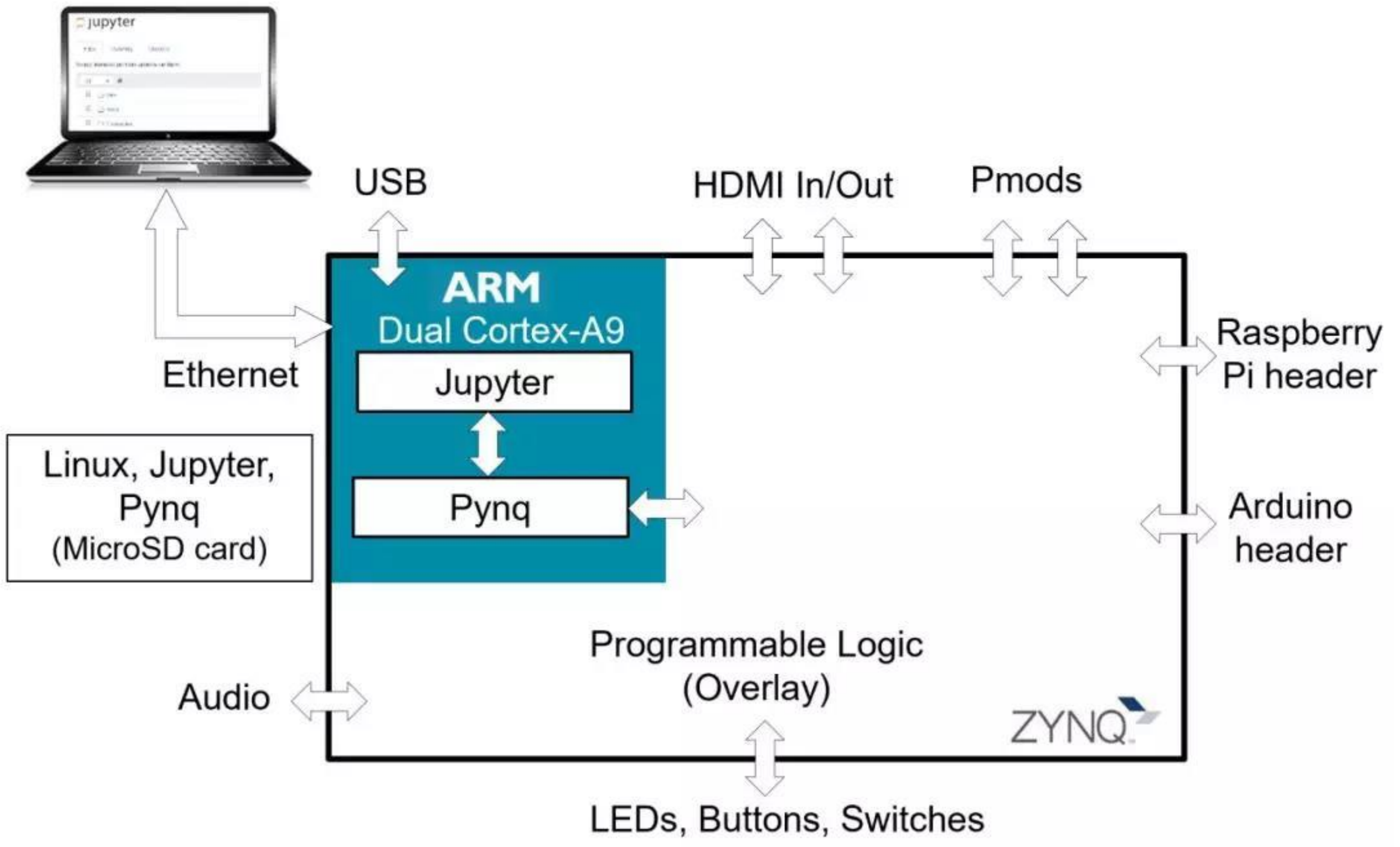
Xilinx PYNQ-Z2 FPGA



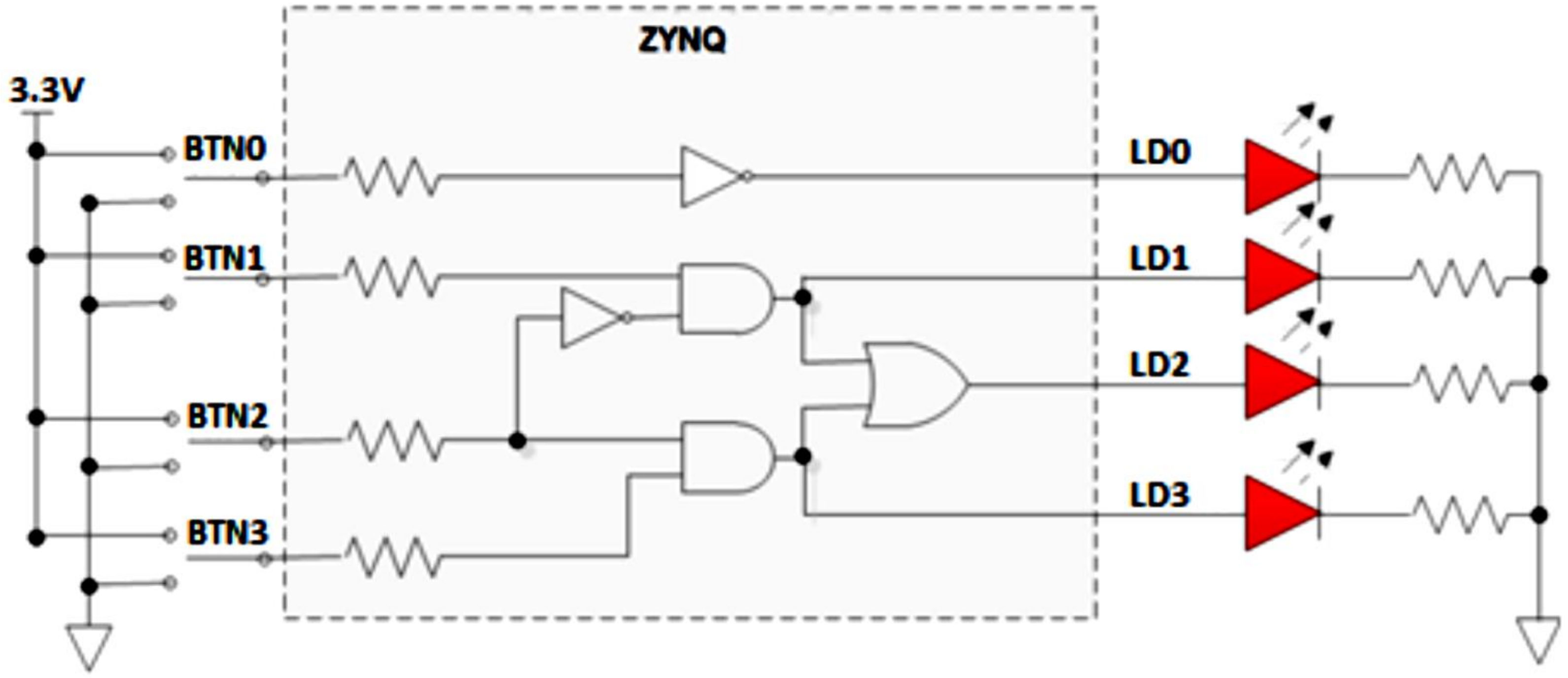
Xilinx PYNQ-Z2 FPGA



Xilinx PYNQ-Z2 FPGA



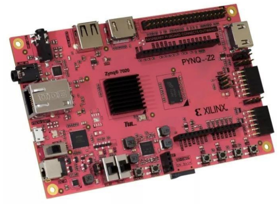
Lab1



VLSI for DSP

Vivado Design Flow
Getting Started

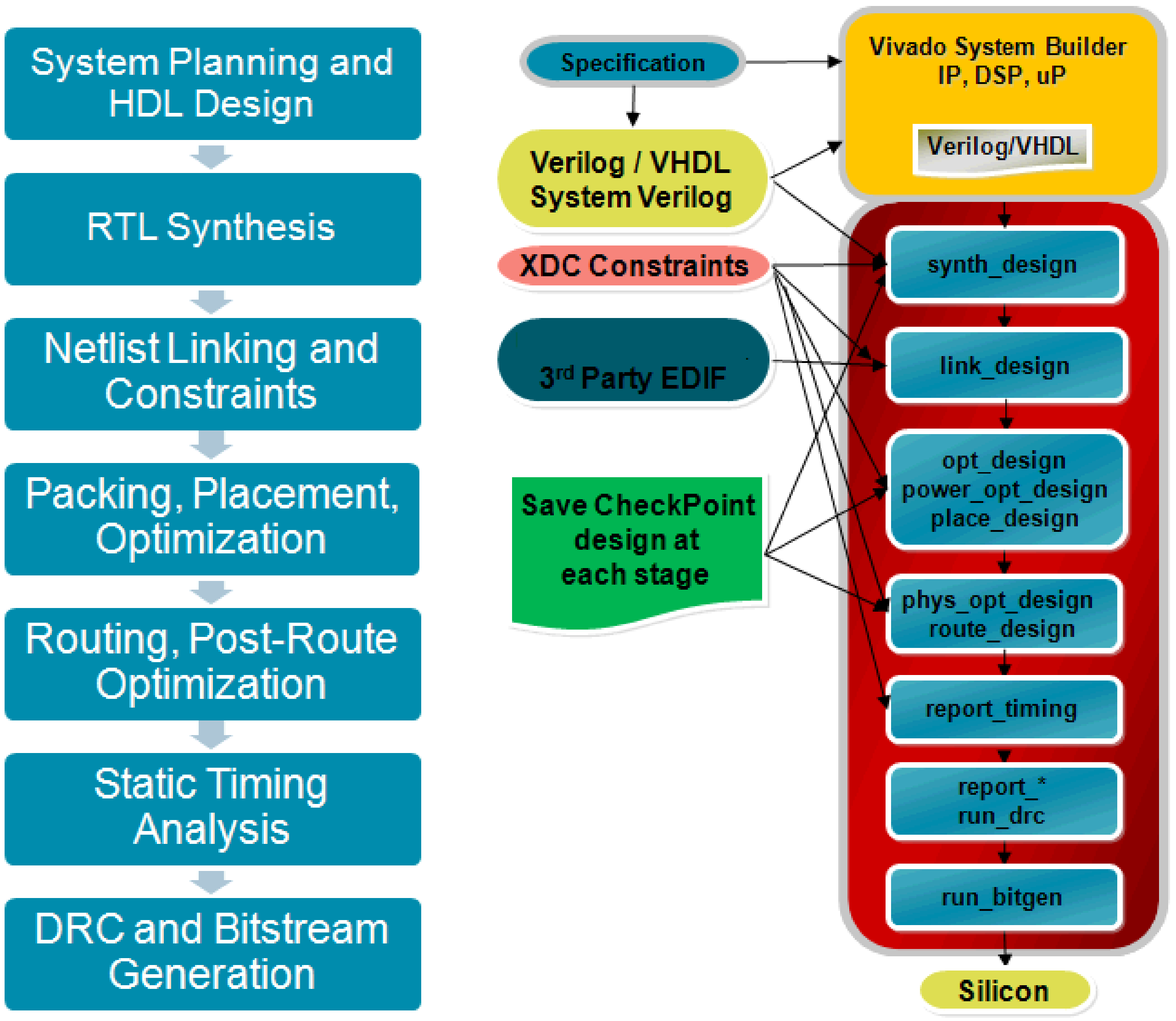
Department of Electronic Engineering
Chang Guan University
chenyh@mail.cgu.edu.tw



Objectives

- Create a Vivado project sourcing HDL model(s) and targeting the ZYNQ device located on the PYNQ-Z2
- Use the provided Xilinx Design Constraint (XDC) file to constrain the pin locations
- Simulate the design using the Vivado simulator
- Synthesize and implement the design
- Generate the bitstream
- Configure ZYNQ using the generated bitstream and verify the functionality

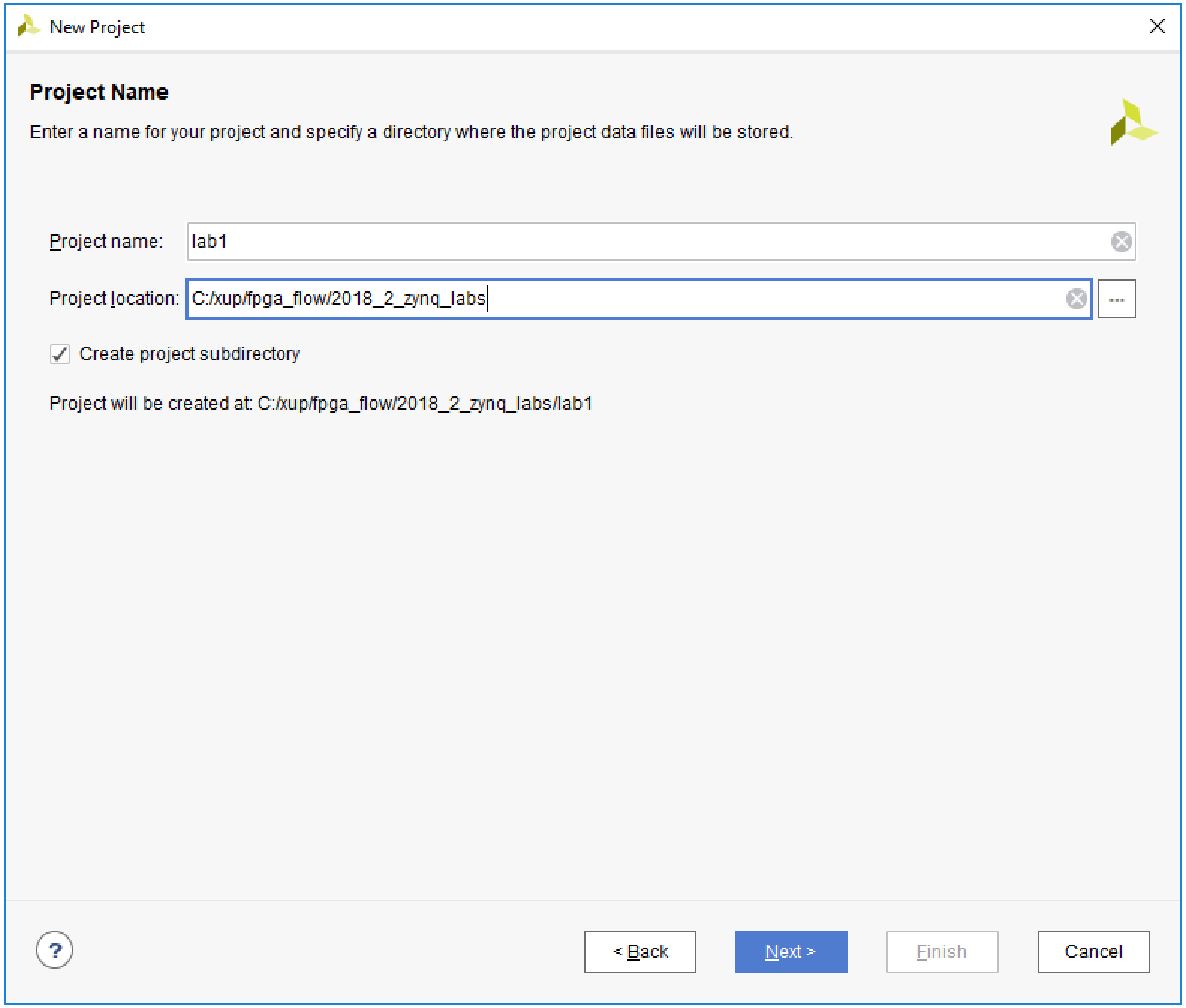
Vivado Design Flow


System Planning and HDL Design
RTL Synthesis
Netlist Linking and Constraints
Packing, Placement, Optimization
Routing, Post-Route Optimization
Static Timing Analysis
DRC and Bitstream Generation
Specification
Verilog / VHDL System Verilog
XDC Constraints
3rd Party EDIF
Save CheckPoint design at each stage
Vivado System Builder IP, DSP, uP
Verilog/VHDL
synth_design
link_design
opt_design power_opt_design place_design
phys_opt_design route_design
report_timing
report_* run_drc
run_bitgen
Silicon

Create a Vivado Project using IDE

- Launch Vivado and create an empty project targeting the PYNQ-Z2 board, selecting Verilog as a target language. Use the provided lab1.v and lab1_zynq.xdc files.
  1. Open Vivado by selecting **Start > Xilinx Design Tools > Vivado 2018.2**
  2. Click **Create New Project** to start the wizard. You will see *Create A New Vivado Project* dialog box. Click **Next**.
  3. Click the Browse button of the *Project location* field of the **New Project** form, browse to **C:\xilinx_works**, and click **Select**.
  4. Enter **lab1** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.

Create a Vivado Project using IDE


New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: lab1

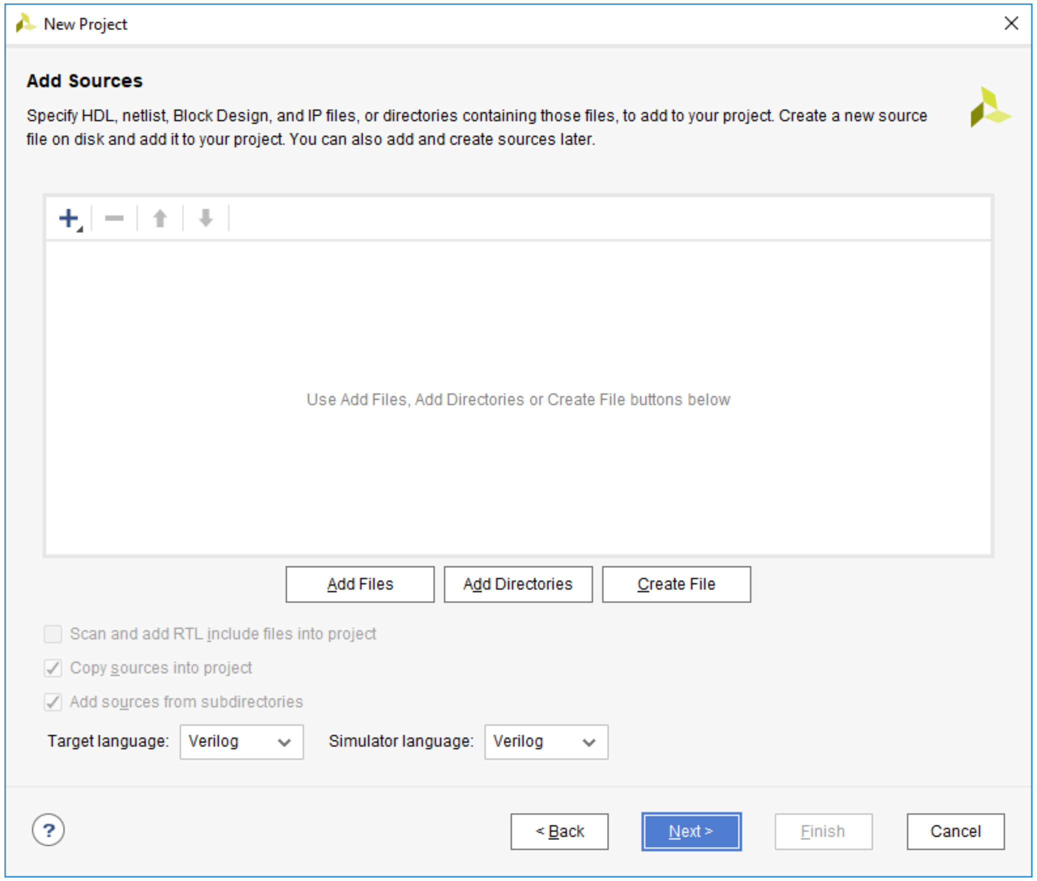
Project location: C:/xup/fpga_flow/2018_2_zynq_labs

✓ Create project subdirectory

Project will be created at: C:/xup/fpga_flow/2018_2_zynq_labs/lab1

Create a Vivado Project using IDE

5. Select **RTL Project** option in the *Project Type* form, and click **Next**.
6. Using the drop-down buttons, select **Verilog** as the *Target Language* and *Simulator Language* in the *Add Sources* form.



New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

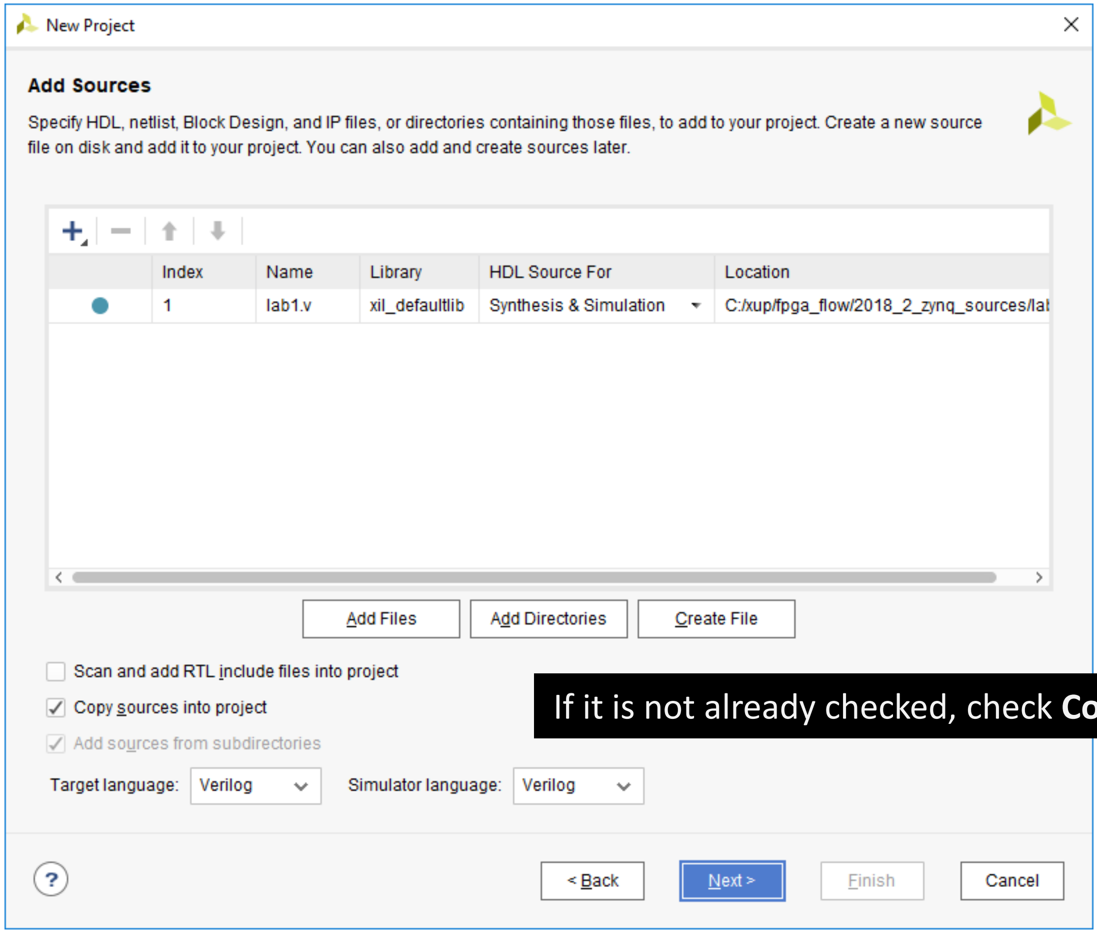
Use Add Files, Add Directories or Create File buttons below

- [ ] Scan and add RTL include files into project
- [✓] Copy sources into project
- [✓] Add sources from subdirectories

Target language: Verilog   Simulator language: Verilog

Create a Vivado Project

7. Click on the **Blue Plus** button, then **Add Files...** and browse to the **C:\xilinx_works\lab1** directory, select *lab1.v,* click **OK**.



New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

	Index	Name	Library	HDL Source For	Location
●	1	lab1.v	xil_defaultlib	Synthesis & Simulation	C:/xup/fpga_flow/2018_2_zynq_sources/lat

☐ Scan and add RTL include files into project
☑ Copy sources into project
☑ Add sources from subdirectories

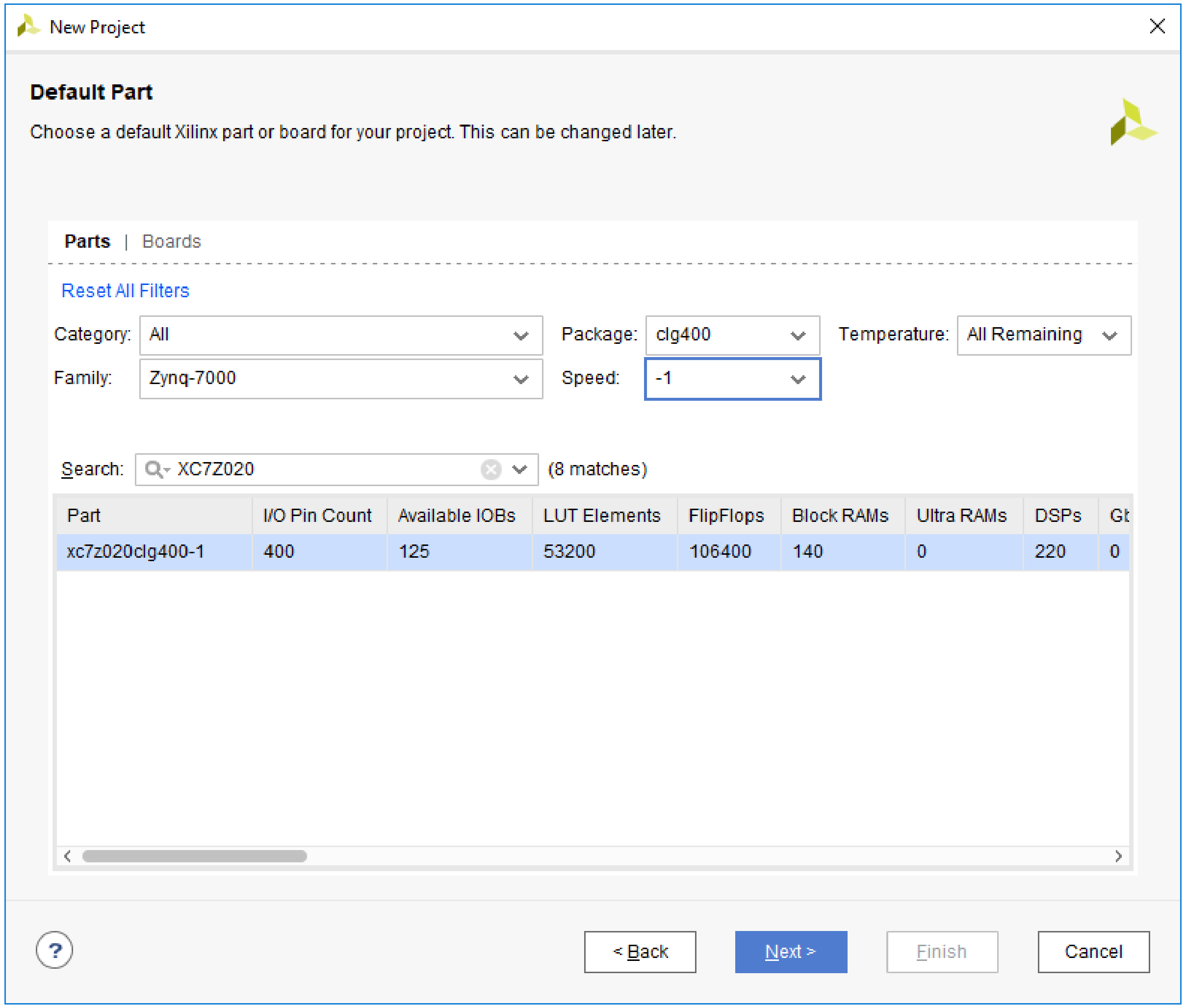
Target language: Verilog
Simulator language: Verilog

If it is not already checked, check **Copy sources into project**

Create a Vivado Project

8. Click **Next** to get to the *Add Constraints* form.
9. Click on the **Blue Plus** button, then **Add Files…** and browse to the **C:\xilinx_works\lab1** directory (if necessary), select *lab1_zynq.xdc* and click **OK** (if necessary), and then click **Next.**
10. The Xilinx Design Constraints file assigns the physical IO locations on FPGA to the switches and LEDs located on the board. This information can be obtained either through the board's schematic or the board's user guide.
11. In the *Default Part* form, use the **Parts** option and various drop-down fields of the **Filter** section. Select the **XC7Z020clg400-1**.
12. Click **Finish** to create the Vivado project.

Create a Vivado Project



New Project

Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

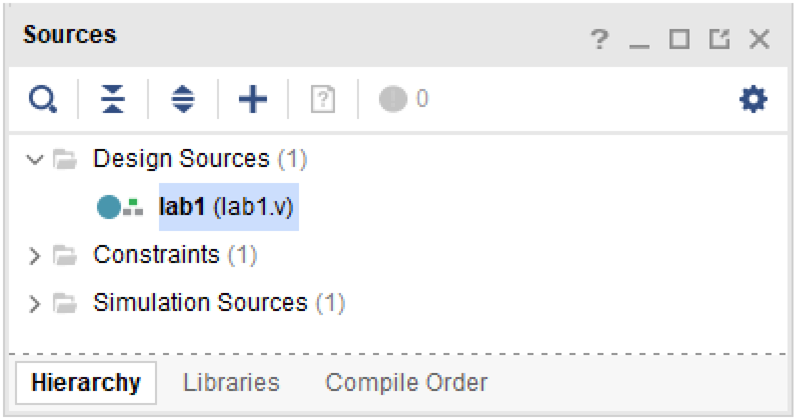
Category: All
Family: Zynq-7000
Package: clg400
Speed: -1
Temperature: All Remaining

Search: XC7Z020 (8 matches)

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gt
xc7z020clg400-1	400	125	53200	106400	140	0	220	0

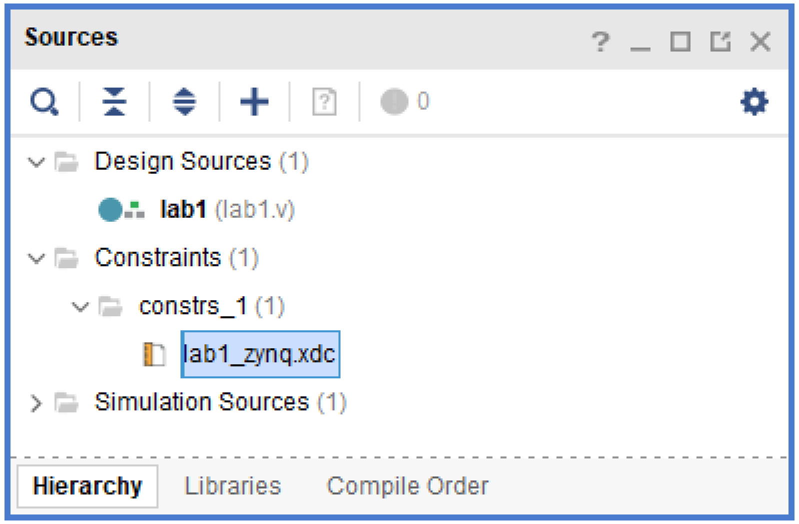
Analyze lab1.v

1. In the *Sources* pane, double-click the **lab1.v** entry to open the file in text mode.
2. Notice in the Verilog code that the first line defines the timescale directive for the simulator. Lines 2-4 are comment lines describing the module name and the purpose of the module.
3. Line 7 defines the beginning (marked with keyword **module**) and Line 17 defines the end of the module (marked with keyword **endmodule**).
4. Lines 8-9 defines the input and output ports whereas lines 12-15 defines the actual functionality.



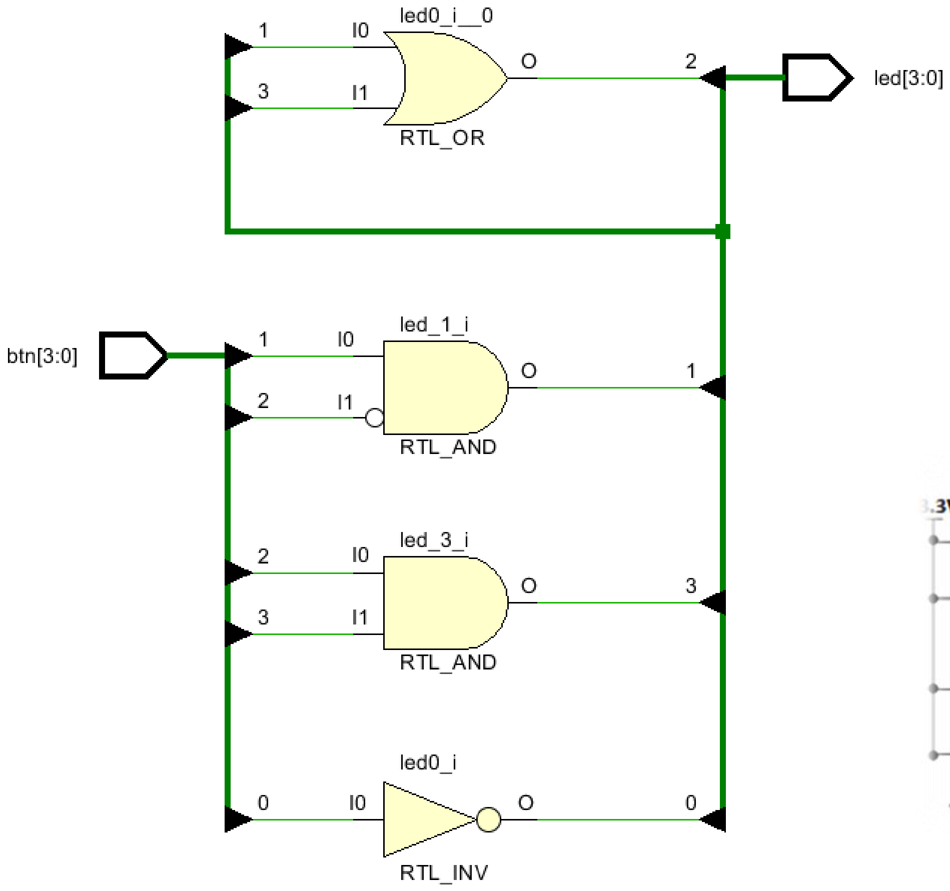
Analyze lab1_zynq.xdc

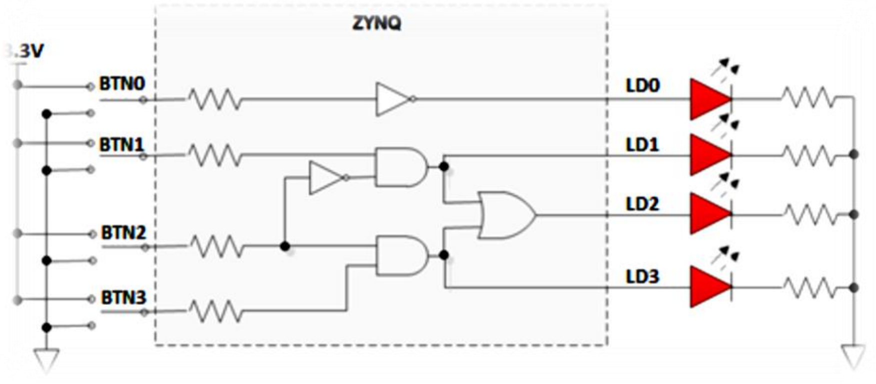
1. In the *Sources* pane, expand the *Constraints* folder and double-click the **lab1_zynq.xdc** entry to open the file in text mode.
2. Lines 5-8 define the pin locations for the input buttons and lines 13-16 define pin locations for output LEDs.



Perform RTL analysis on the source file

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

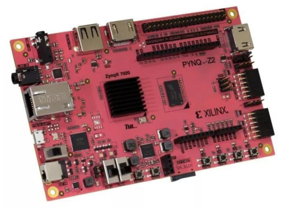




VLSI for DSP

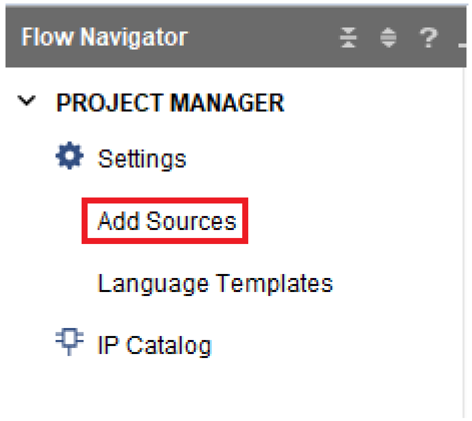
Simulate the Design using the Vivado Simulator

Department of Electronic Engineering
Chang Guan University
chenyh@mail.cgu.edu.tw



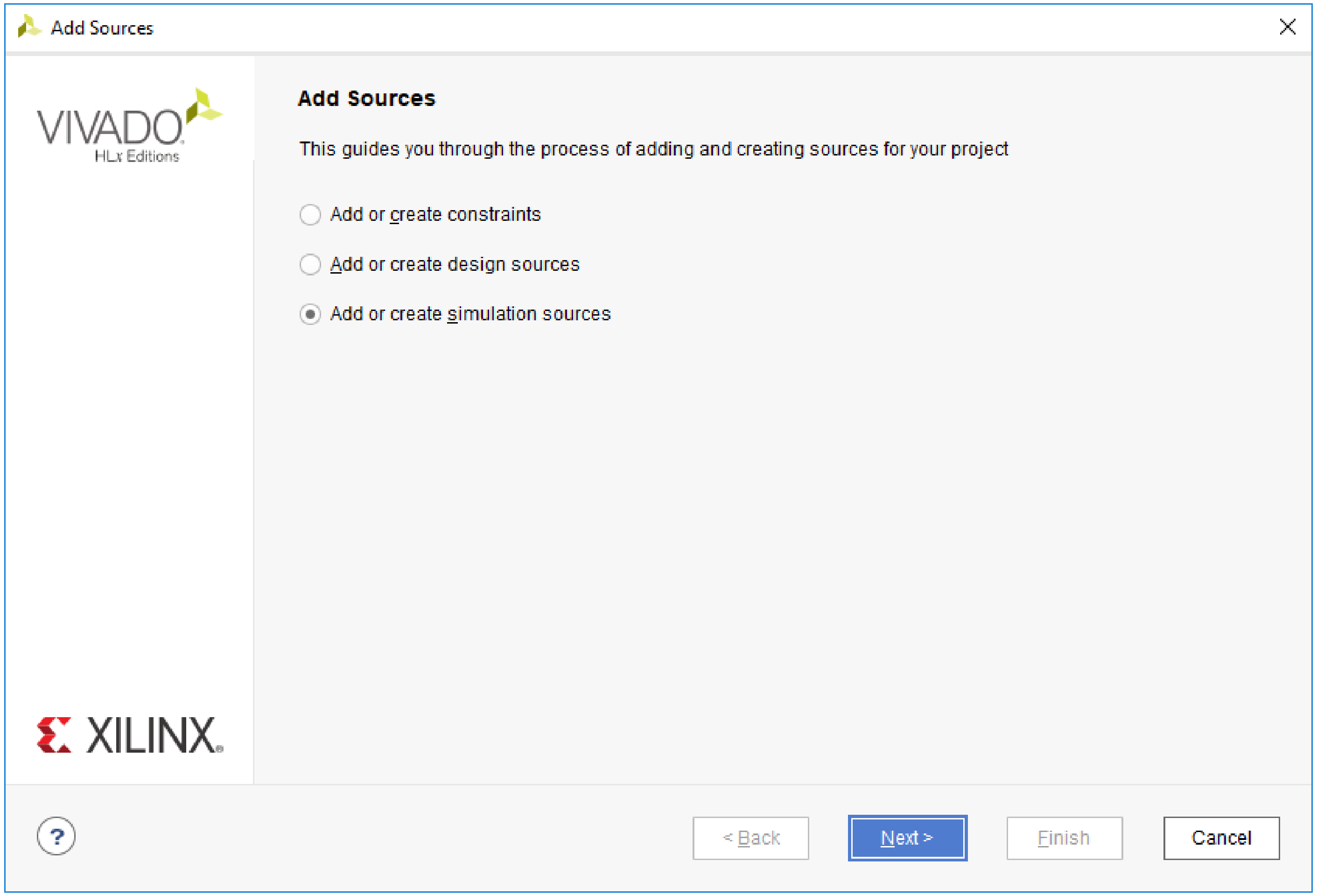
Add the lab1_tb.v testbench file

1. Click **Add Sources** under the *Project Manager* tasks of the *Flow Navigator* pane.



Add the testbench file

2. Select the *Add or Create Simulation Sources* option and click **Next**.


Add Sources

Add Sources

This guides you through the process of adding and creating sources for your project

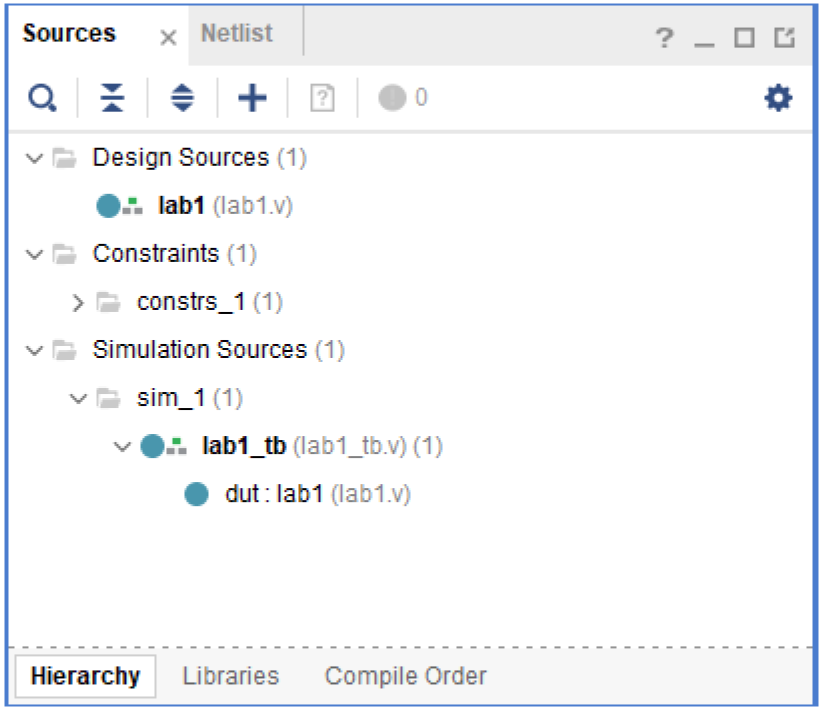
( ) Add or create constraints

( ) Add or create design sources

(●) Add or create simulation sources

Add the testbench file

3. In the *Add Sources Files* form, click the **Blue Plus** button and then **Add Files….**
4. Browse to the **C:\xilinx_works\lab1** folder and select *lab1_tb.v* and click **OK**.
5. Click **Finish**.
6. Select the *Sources* tab and expand the *Simulation Sources* group.

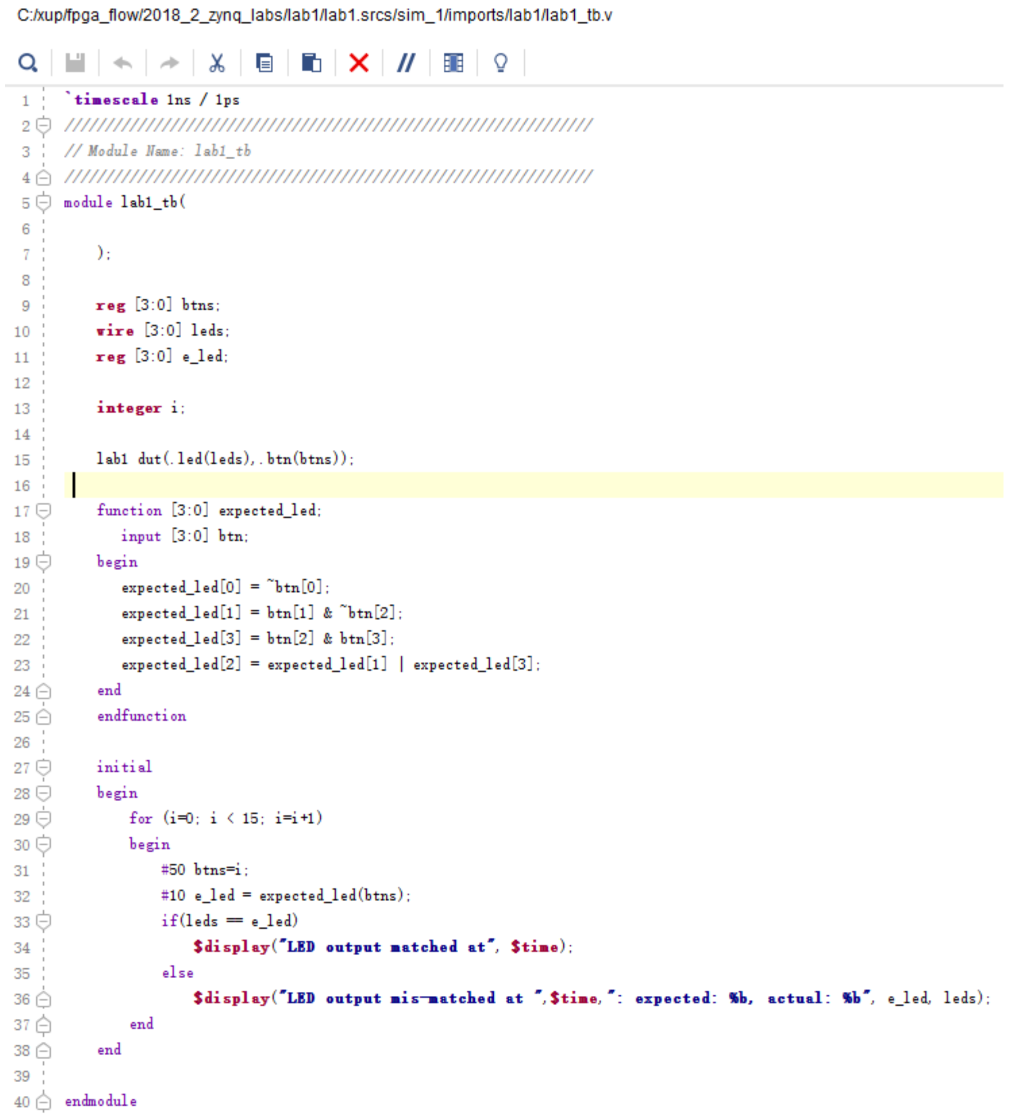


Design Sources (1)
lab1 (lab1.v)
Constraints (1)
constrs_1 (1)
Simulation Sources (1)
sim_1 (1)
lab1_tb (lab1_tb.v) (1)
dut : lab1 (lab1.v)

Add the testbench file

7. Using the Windows Explorer, verify that the **sim_1** directory is created at the same level as ***constrs_1*** and ***sources_1*** directories under the ***lab1.srcs*** directory, and that a copy of lab1_tb.v is placed under **lab1.srcs > sim_1 > imports > lab1**.
8. Double-click on the **lab1_tb** in the *Sources* pane to view its contents.

testbench file

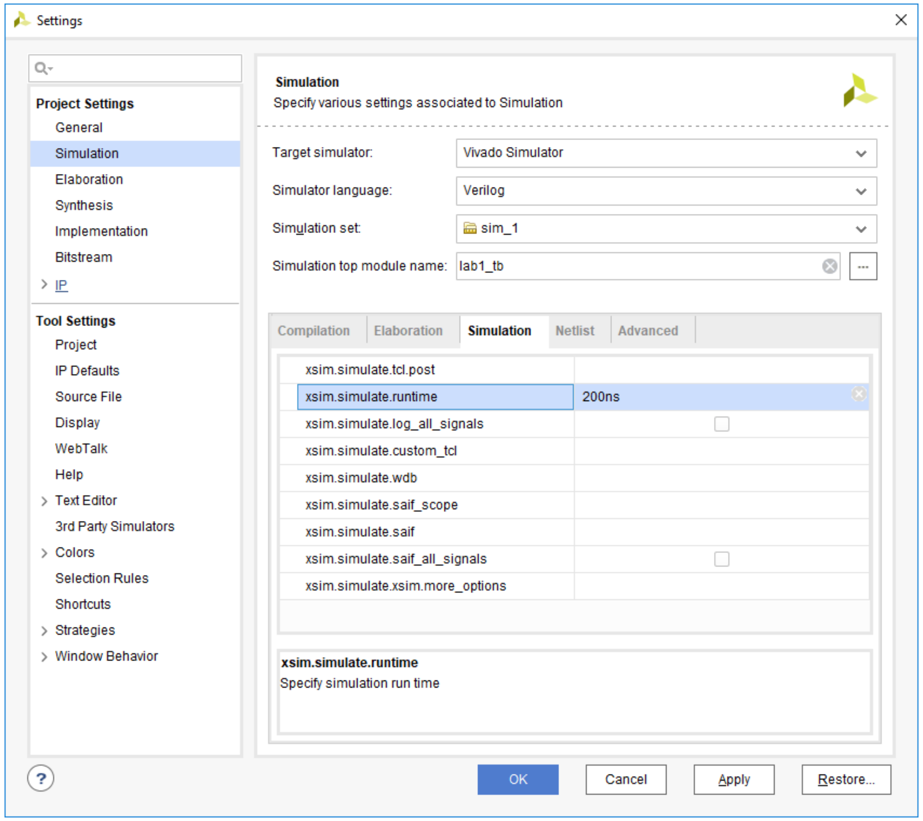


C:/xup/fpga_flow/2018_2_zynq_labs/lab1/lab1.srcs/sim_1/imports/lab1/lab1_tb.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Module Name: lab1_tb
//////////////////////////////////////////////////////////////////
module lab1_tb(

    );

    reg [3:0] btns;
    wire [3:0] leds;
    reg [3:0] e_led;

    integer i;

    lab1 dut(.led(leds),.btn(btns));

    function [3:0] expected_led;
       input [3:0] btn;
    begin
       expected_led[0] = ~btn[0];
       expected_led[1] = btn[1] & ~btn[2];
       expected_led[3] = btn[2] & btn[3];
       expected_led[2] = expected_led[1] | expected_led[3];
    end
    endfunction

    initial
    begin
        for (i=0; i < 15; i=i+1)
        begin
            #50 btns=i;
            #10 e_led = expected_led(btns);
            if(leds == e_led)
                $display("LED output matched at", $time);
            else
                $display("LED output mis-matched at ",$time,": expected: %b, actual: %b", e_led, leds);
        end
    end

endmodule
```

Simulate the design for 200ns

1. Select **Settings** under the *Project Manager* tasks of the *Flow Navigator* pane.
2. A **Settings** form will appear showing the **Simulation** properties form.
3. Select the **Simulation** tab, and set the **Simulation Run Time** value to 200 ns and click **OK**.



Simulation
Specify various settings associated to Simulation

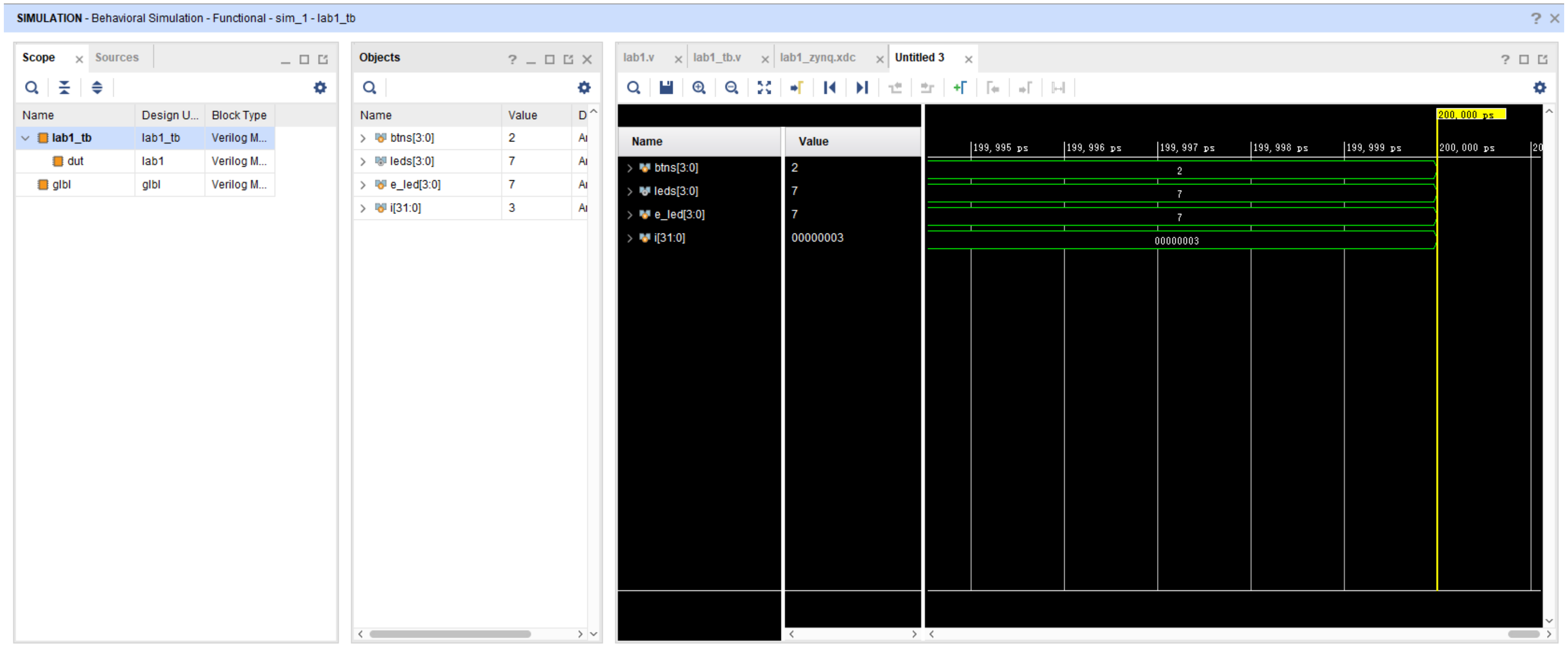
Target simulator:	Vivado Simulator
Simulator language:	Verilog
Simulation set:	sim_1
Simulation top module name:	lab1_tb

xsim.simulate.tcl.post	
xsim.simulate.runtime	200ns
xsim.simulate.log_all_signals	☐
xsim.simulate.custom_tcl	
xsim.simulate.wdb	
xsim.simulate.saif_scope	
xsim.simulate.saif	
xsim.simulate.saif_all_signals	☐
xsim.simulate.xsim.more_options	

xsim.simulate.runtime
Specify simulation run time

Simulate the design

4. Click on **Simulation > Run Simulation > Run Behavioral Simulation** under the *Project Manager* tasks of the *Flow Navigator* pane.



SIMULATION - Behavioral Simulation - Functional - sim_1 - lab1_tb

Scope

Name	Design U...	Block Type
lab1_tb	lab1_tb	Verilog M...
dut	lab1	Verilog M...
glbl	glbl	Verilog M...

Objects

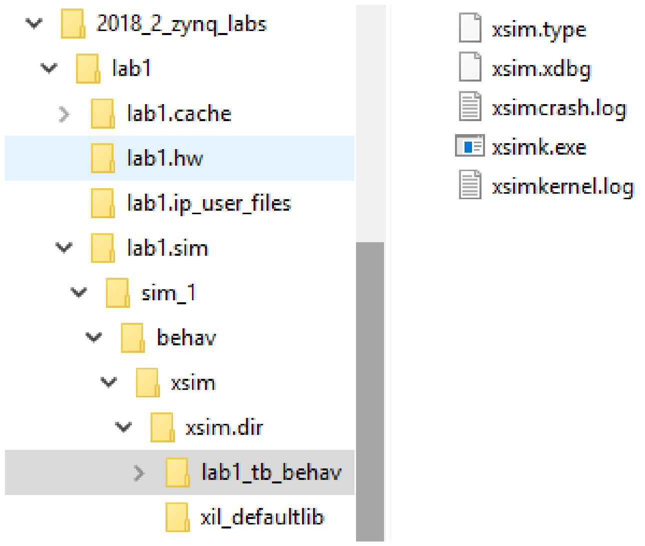
Name	Value	D
btns[3:0]	2	A
leds[3:0]	7	A
e_led[3:0]	7	A
i[31:0]	3	A

Untitled 3

Name	Value
btns[3:0]	2
leds[3:0]	7
e_led[3:0]	7
i[31:0]	00000003

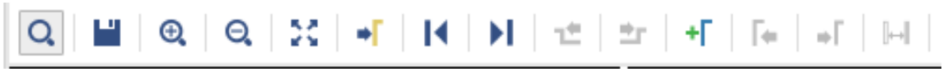
Simulate the design

- You will see four main views:
  - (i) *Scopes,* where the testbench hierarchy as well as glbl instances are displayed,
  - (ii) *Objects,* where top-level signals are displayed,
  - (iii) the waveform window, and
  - (iv) *Tcl Console* where the simulation activities are displayed.
- Notice that since the testbench used is self-checking, the results are displayed as the simulation is run.
- Notice that the **lab1.sim** directory is created under the **lab1** directory, along with several lower-level directories.



Simulate the design

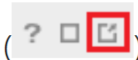
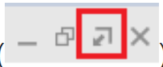
You will see several buttons next to the waveform window which can be used for the specific purpose as listed in the table below.



Various buttons available to view the waveform

4. Click on the *Zoom Fit* button ( ) to see the entire waveform.

   Notice that the output changes when the input changes.

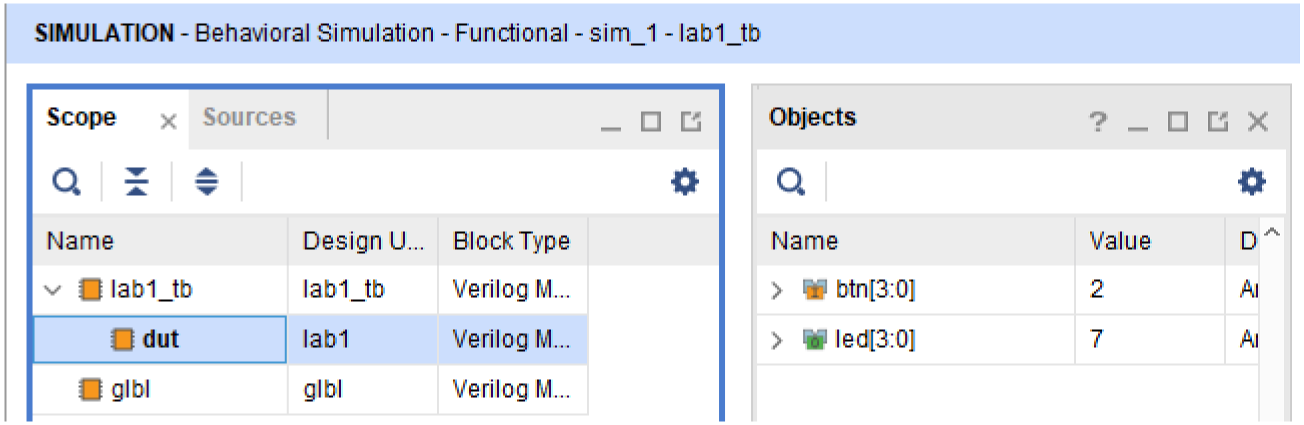
   You can also float the simulation waveform window by clicking on the Float button () on the upper right hand side of the view. This will allow you to have a wider window to view the simulation waveforms. To reintegrate the floating window back into the GUI, simply click on the Dock Window button ().

Change display format

- Select **i[31:0]** in the waveform window, right-click, select *Radix*, and then select *Unsigned Decimal* to view the for-loop index in an unsigned *integer* form.
- Similarly, change the radix of **btn[3:0]** to *Hexadecimal*.
- Leave the **leds[3:0]** and **e_led[3:0]** radix to *binary* as we want to see each output bit.

Add more signals

1. Expand the **lab1_tb** instance, if necessary, in the *Scopes* window and select the **dut** instance.
   – The btn[3:0] and led[3:0] signals will be displayed in the *Objects* window.



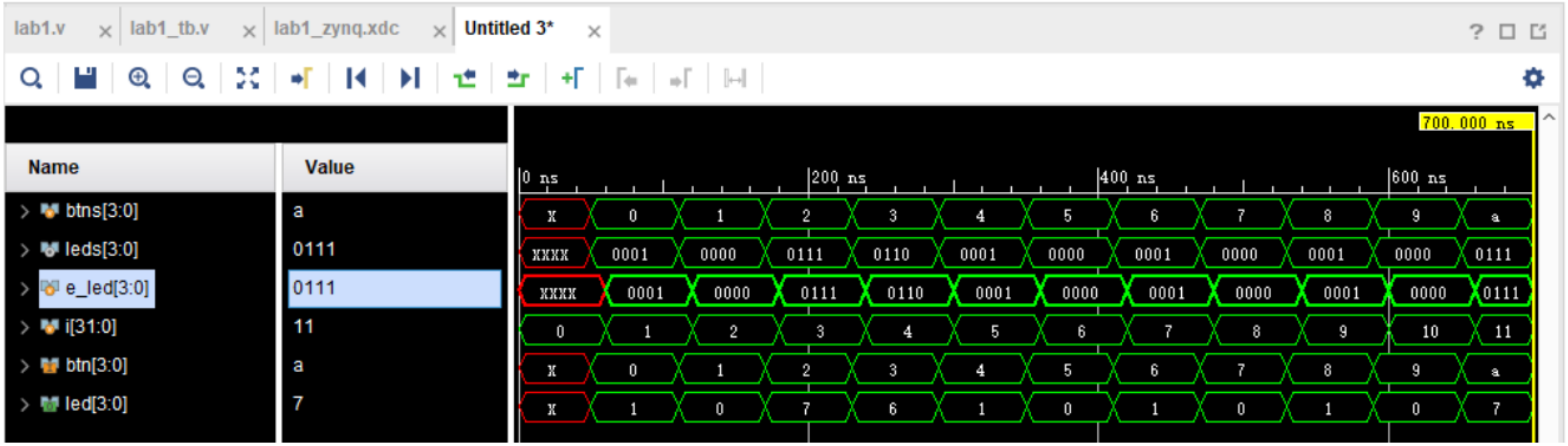
SIMULATION - Behavioral Simulation - Functional - sim_1 - lab1_tb

Name	Design U...	Block Type
lab1_tb	lab1_tb	Verilog M...
dut	lab1	Verilog M...
glbl	glbl	Verilog M...

Name	Value	D
btn[3:0]	2	A
led[3:0]	7	A

Run the simulation for 500 ns

2. Select **btn[3:0]** and **led[3:0]** and drag them into the waveform window to monitor those lower-level signals.
3. On the simulator tool buttons ribbon bar, type 500 over in the simulation run time field, click on the drop-down button of the units field and select ns since we want to run for 500 ns (total of 700 ns), and click on the ( ) button. The simulation will run for an additional 500 ns.
4. Click on the *Zoom Fit* button and observe the output.



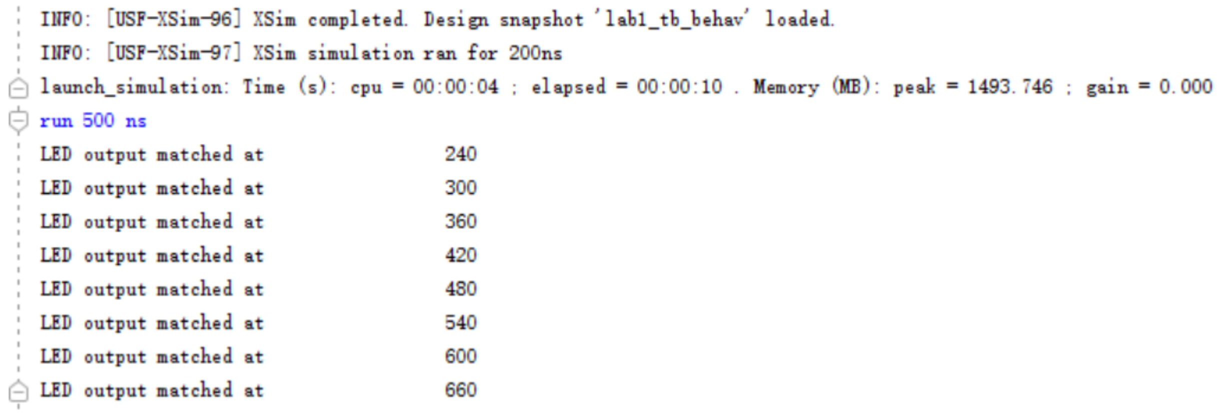
Name	Value
btns[3:0]	a
leds[3:0]	0111
e_led[3:0]	0111
i[31:0]	11
btn[3:0]	a
led[3:0]	7

```
btns[3:0]   X    0    1    2    3    4    5    6    7    8    9    a
leds[3:0]   XXXX 0001 0000 0111 0110 0001 0000 0001 0000 0001 0000 0111
e_led[3:0]  XXXX 0001 0000 0111 0110 0001 0000 0001 0000 0001 0000 0111
i[31:0]     0    1    2    3    4    5    6    7    8    9    10   11
btn[3:0]    X    0    1    2    3    4    5    6    7    8    9    a
led[3:0]    X    1    0    7    6    1    0    1    0    1    0    7
```

Running simulation for additional 500 ns

Function Simulation

Observe the Tcl Console window and see the output is being displayed as the testbench uses the $display task.



```
INFO: [USF-XSim-96] XSim completed. Design snapshot 'lab1_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 200ns
launch_simulation: Time (s): cpu = 00:00:04 ; elapsed = 00:00:10 . Memory (MB): peak = 1493.746 ; gain = 0.000
run 500 ns
LED output matched at                    240
LED output matched at                    300
LED output matched at                    360
LED output matched at                    420
LED output matched at                    480
LED output matched at                    540
LED output matched at                    600
LED output matched at                    660
```

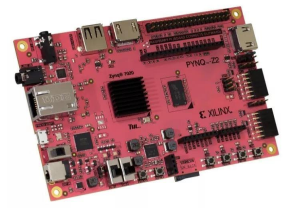
Tcl Console output after running the simulation for additional 500 ns

5. Close the simulator by selecting **File > Close Simulation**.
6. Click **OK** and then click **Discard** to close it without saving the waveform.

VLSI for DSP

Synthesize the Design

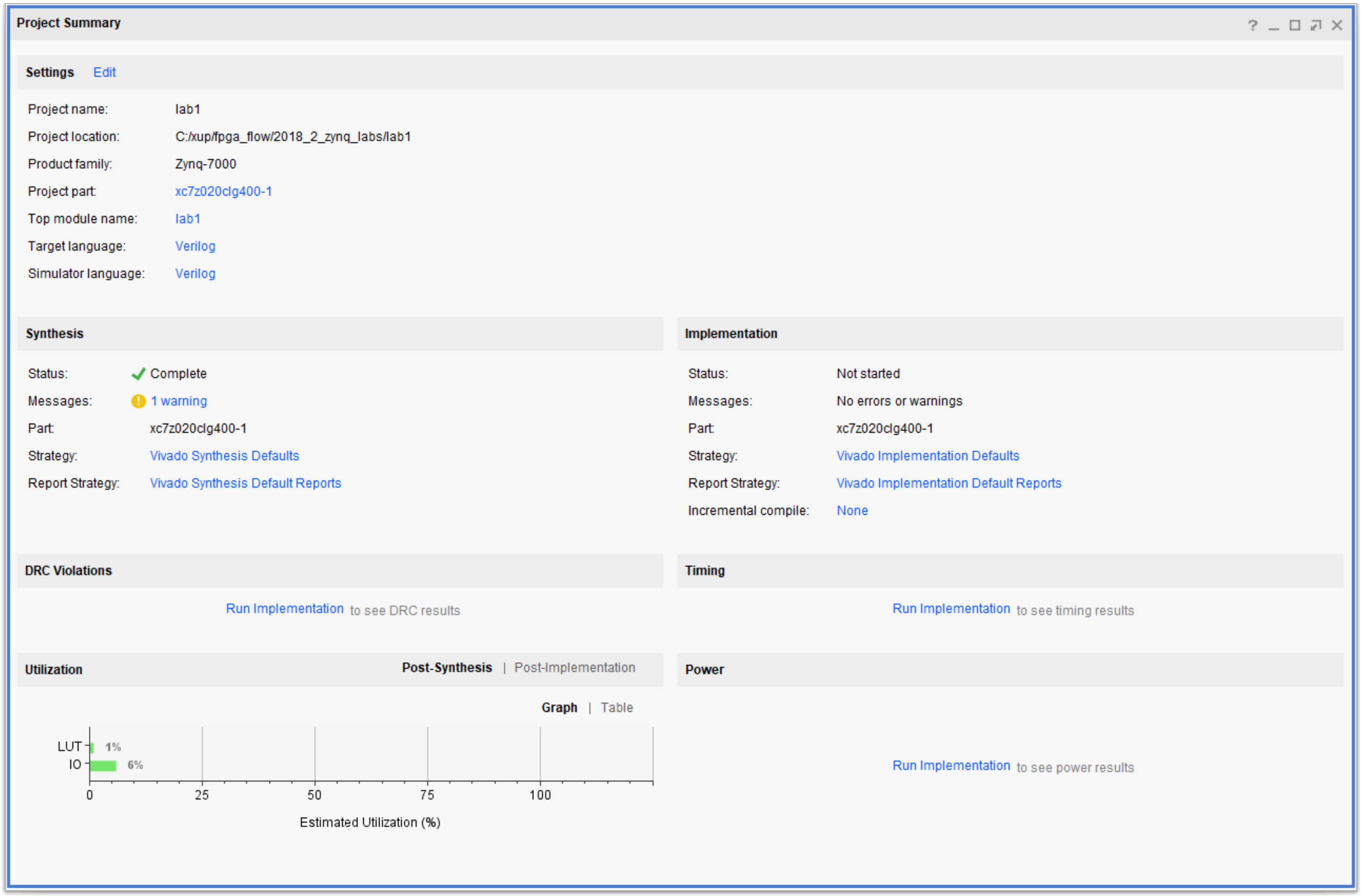
Department of Electronic Engineering
Chang Guan University
chenyh@mail.cgu.edu.tw



Synthesize the design

1. Click on **Run Synthesis** under the *SYNTHESIS* tasks of the *Flow Navigator* pane.
   - The synthesis process will be run on the lab1.v file (and all its hierarchical files if they exist). When the process is completed a *Synthesis Completed* dialog box with three options will be displayed.
2. Select the *Open Synthesized Design* option and click **OK** as we want to look at the synthesis output before progressing to the implementation stage.
   - Click **Yes** to close the elaborated design if the dialog box is displayed.
3. Select the **Project Summary** tab and understand the various windows.
   - If you don't see the Project Summary tab then select **Window > Project Summary** or click the **Project Summary** icon Σ

Synthesize the design



Project Summary

Settings Edit

Project name:	lab1
Project location:	C:/xup/fpga_flow/2018_2_zynq_labs/lab1
Product family:	Zynq-7000
Project part:	xc7z020clg400-1
Top module name:	lab1
Target language:	Verilog
Simulator language:	Verilog

Synthesis

Status:	✓ Complete
Messages:	1 warning
Part:	xc7z020clg400-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports

Implementation

Status:	Not started
Messages:	No errors or warnings
Part:	xc7z020clg400-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental compile:	None

DRC Violations

Run Implementation to see DRC results

Timing

Run Implementation to see timing results

Utilization Post-Synthesis | Post-Implementation

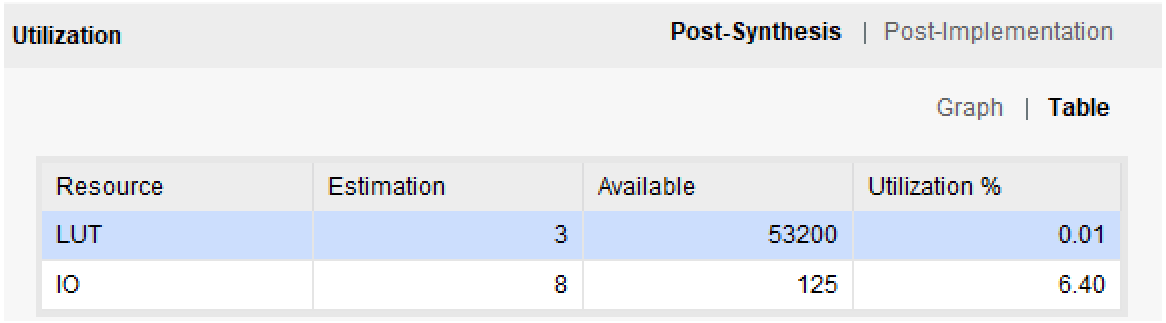
Graph | Table

Power

Run Implementation to see power results

Synthesize the design

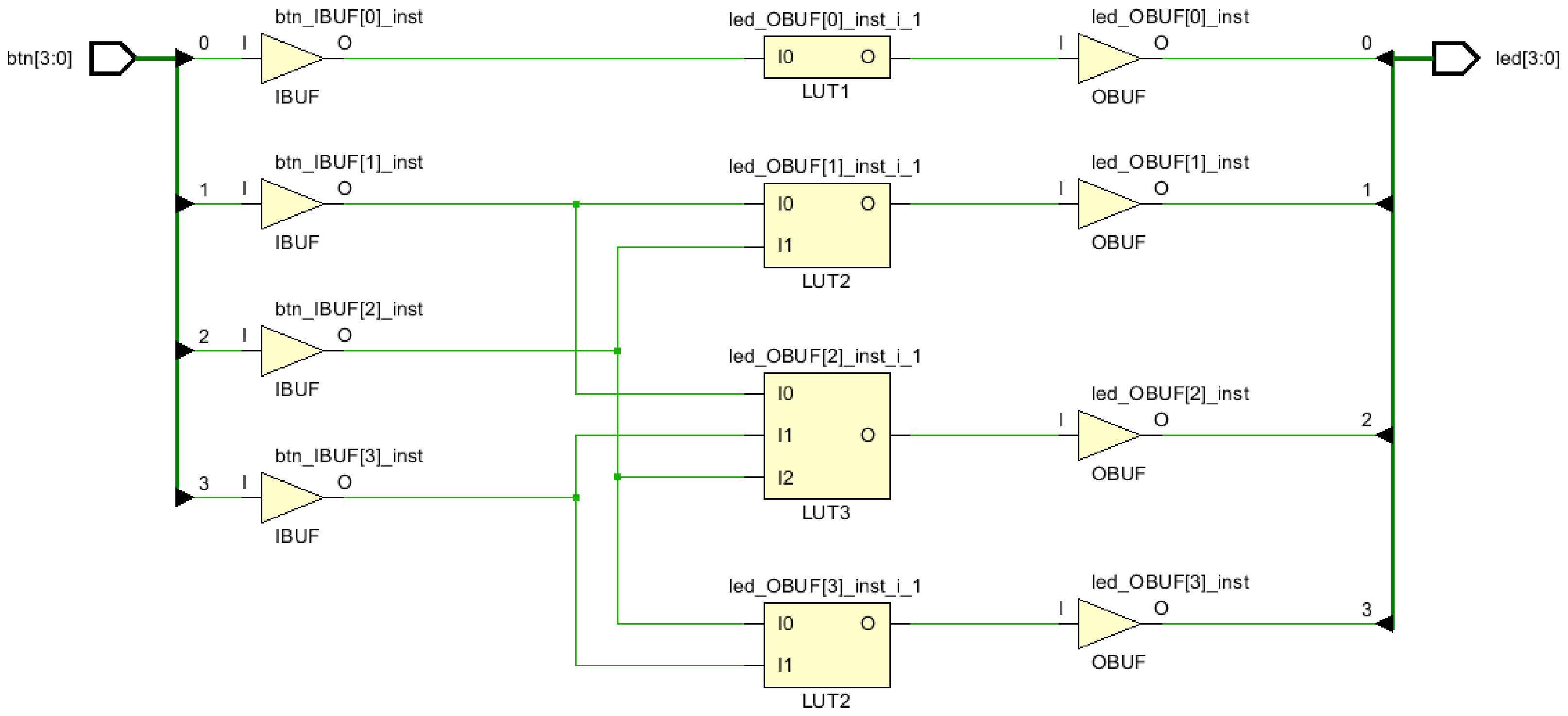
4. Click on the **Table** tab in the **Project Summary** tab.
   – Notice that there are an estimated 3 LUTs and 8 IOs (4 input and 4 output) that are used.



Resource	Estimation	Available	Utilization %
LUT	3	53200	0.01
IO	8	125	6.40

Synthesize the design

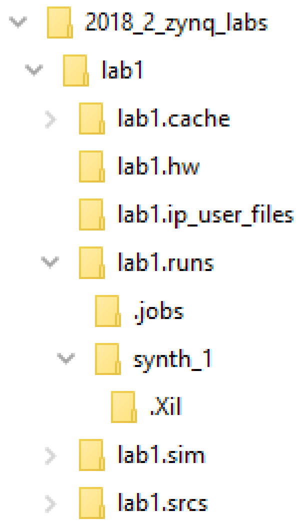
5. In The *Flow Navigator*, under *Synthesis* (expand *Open Synthesized Design* if necessary), click on **Schematic** to view the synthesized design in a schematic view.



Synthesized design's schematic view

Synthesize the design

- Notice that IBUFs and OBUFs are automatically instantiated (added) to the design as the input and output are buffered. The logical gates are implemented in LUTs (1 input is listed as LUT1, 2 input is listed as LUT2, and 3 input is listed as LUT3). Four gates in RTL analysis output are mapped onto four LUTs in the synthesized output.
- Using Windows Explorer, verify that **lab1.runs** directory is created under **lab1**. Under the **runs** directory, **synth_1** directory is created which holds several files related to synthesis.

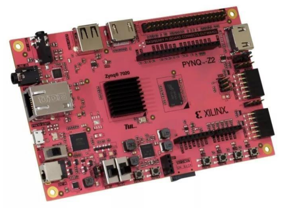


Directory structure after synthesizing the design

VLSI for DSP

Implement the Design

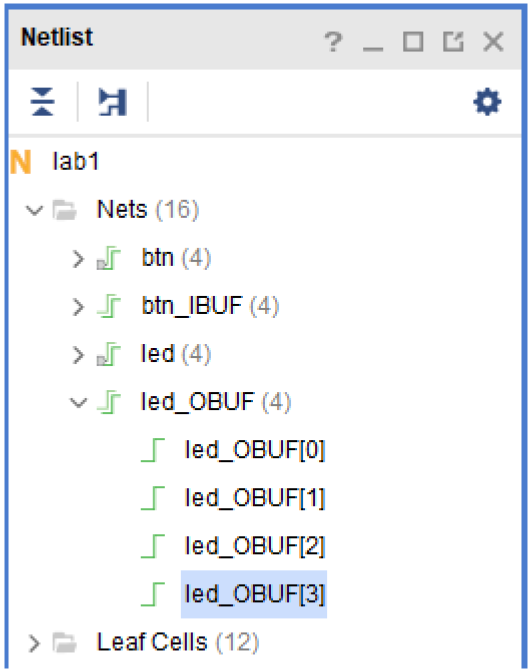
Department of Electronic Engineering
Chang Guan University
chenyh@mail.cgu.edu.tw



Implement the design

1. Click on **Run Implementation** under the *Implementation* tasks of the *Flow Navigator* pane.
   - The implementation process will be run on the synthesized design. When the process is completed an *Implementation Completed* dialog box with three options will be displayed.
2. Select **Open implemented design** and click **OK** as we want to look at the implemented design in a Device view tab.
3. Click **Yes,** if prompted, to close the synthesized design. The implemented design will be opened.
4. In the *Netlist* pane, select one of the nets (e.g. led_OBUF[3]) and notice that the net displayed in the X1Y2 clock region in the Device view tab (you may have to zoom in to see it).
5. If it is not selected, click the *Routing Resources* icon to show routing resources.

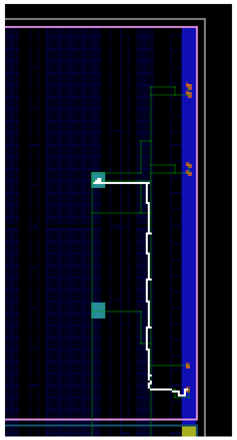
Implement the design



Netlist

N lab1
- Nets (16)
  - btn (4)
  - btn_IBUF (4)
  - led (4)
  - led_OBUF (4)
    - led_OBUF[0]
    - led_OBUF[1]
    - led_OBUF[2]
    - led_OBUF[3]
- Leaf Cells (12)

Viewing implemented design

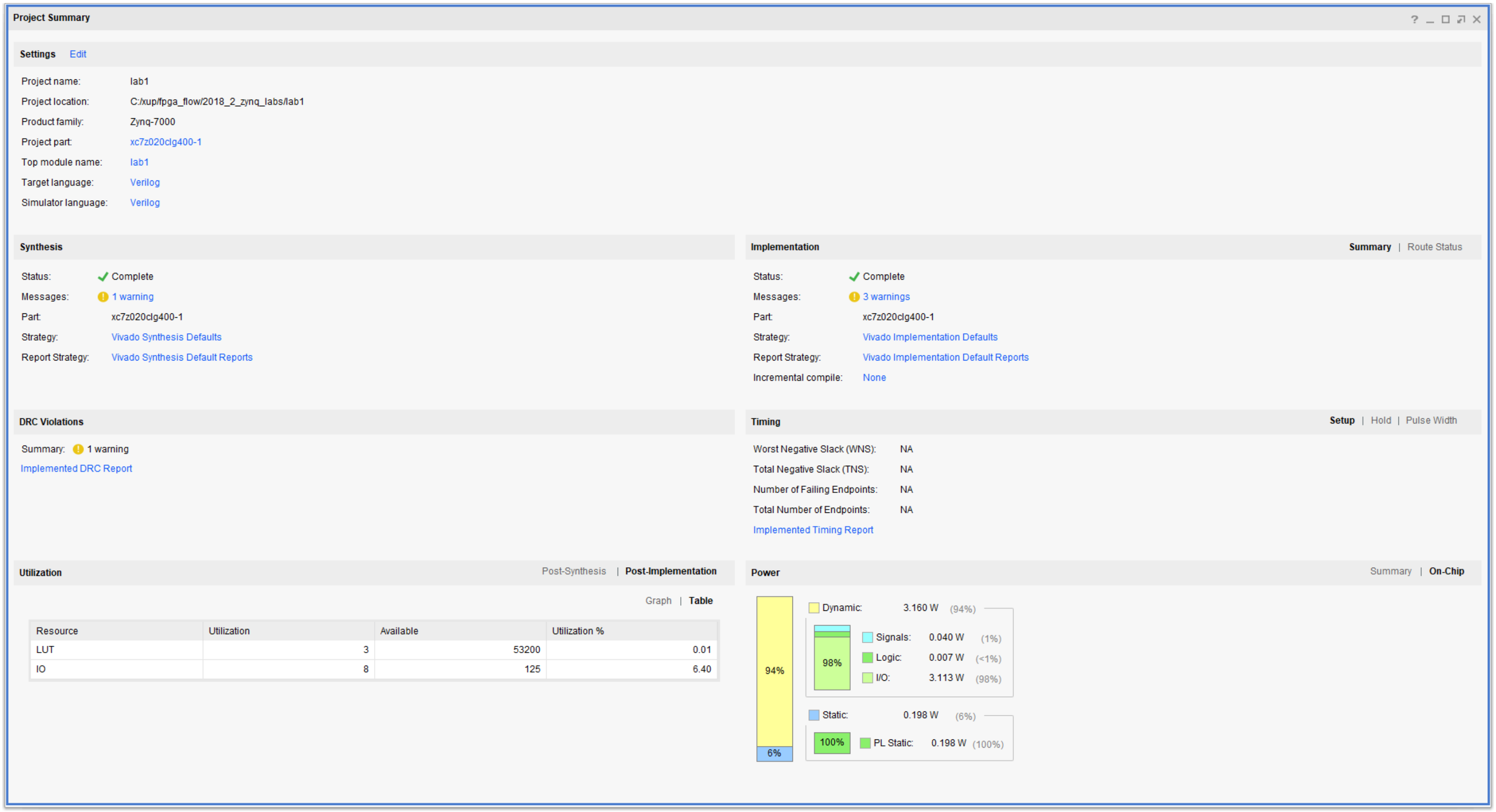


Selecting a net

Implement the design

6. Close the implemented design view by selecting **File > Close Implemented Design**, and select the **Project Summary** tab (you may have to change to the Default Layout view) and observe the results.
7. Select the Post-Implementation tab.
   – **Notice** that the actual resource utilization is 3 LUTs and 8 IOs. Also, it indicates that no timing constraints were defined for this design (since the design is combinatorial).

Implementation results



Project Summary

Settings Edit

Project name:	lab1
Project location:	C:/xup/fpga_flow/2018_2_zynq_labs/lab1
Product family:	Zynq-7000
Project part:	xc7z020clg400-1
Top module name:	lab1
Target language:	Verilog
Simulator language:	Verilog

Synthesis

Status:	✓ Complete
Messages:	1 warning
Part:	xc7z020clg400-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports

Implementation — **Summary** | Route Status

Status:	✓ Complete
Messages:	3 warnings
Part:	xc7z020clg400-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental compile:	None

DRC Violations

Summary: 1 warning

Implemented DRC Report

Timing — **Setup** | Hold | Pulse Width

Worst Negative Slack (WNS):	NA
Total Negative Slack (TNS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA

Implemented Timing Report

Utilization — Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	3	53200	0.01
IO	8	125	6.40

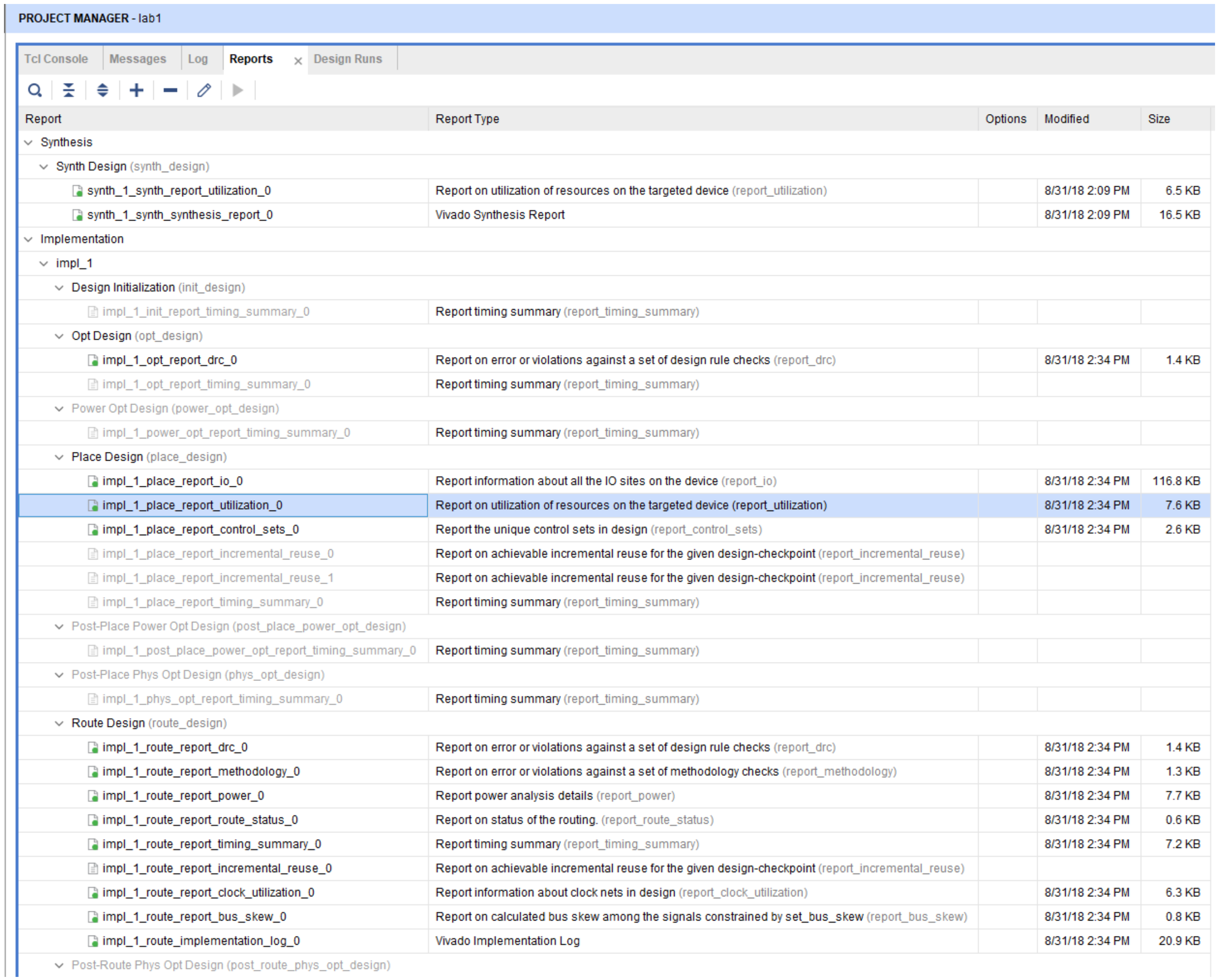
Power — Summary | **On-Chip**

Dynamic:	3.160 W	(94%)
Signals:	0.040 W	(1%)
Logic:	0.007 W	(<1%)
I/O:	3.113 W	(98%)
Static:	0.198 W	(6%)
PL Static:	0.198 W	(100%)

Reports

8. In Vivado, select the **Reports** tab in the bottom panel (if not visible, click *Window* in the menu bar and select **Reports**), and double-click on the *Utilization Report* entry under the *Place Design* section.

Reports



PROJECT MANAGER - lab1

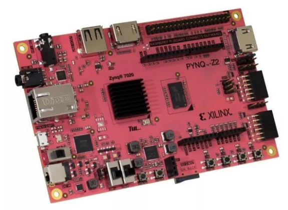
Tcl Console | Messages | Log | Reports | Design Runs

Report	Report Type	Options	Modified	Size
Synthesis				
Synth Design (synth_design)				
synth_1_synth_report_utilization_0	Report on utilization of resources on the targeted device (report_utilization)		8/31/18 2:09 PM	6.5 KB
synth_1_synth_synthesis_report_0	Vivado Synthesis Report		8/31/18 2:09 PM	16.5 KB
Implementation				
impl_1				
Design Initialization (init_design)				
impl_1_init_report_timing_summary_0	Report timing summary (report_timing_summary)			
Opt Design (opt_design)				
impl_1_opt_report_drc_0	Report on error or violations against a set of design rule checks (report_drc)		8/31/18 2:34 PM	1.4 KB
impl_1_opt_report_timing_summary_0	Report timing summary (report_timing_summary)			
Power Opt Design (power_opt_design)				
impl_1_power_opt_report_timing_summary_0	Report timing summary (report_timing_summary)			
Place Design (place_design)				
impl_1_place_report_io_0	Report information about all the IO sites on the device (report_io)		8/31/18 2:34 PM	116.8 KB
impl_1_place_report_utilization_0	Report on utilization of resources on the targeted device (report_utilization)		8/31/18 2:34 PM	7.6 KB
impl_1_place_report_control_sets_0	Report the unique control sets in design (report_control_sets)		8/31/18 2:34 PM	2.6 KB
impl_1_place_report_incremental_reuse_0	Report on achievable incremental reuse for the given design-checkpoint (report_incremental_reuse)			
impl_1_place_report_incremental_reuse_1	Report on achievable incremental reuse for the given design-checkpoint (report_incremental_reuse)			
impl_1_place_report_timing_summary_0	Report timing summary (report_timing_summary)			
Post-Place Power Opt Design (post_place_power_opt_design)				
impl_1_post_place_power_opt_report_timing_summary_0	Report timing summary (report_timing_summary)			
Post-Place Phys Opt Design (phys_opt_design)				
impl_1_phys_opt_report_timing_summary_0	Report timing summary (report_timing_summary)			
Route Design (route_design)				
impl_1_route_report_drc_0	Report on error or violations against a set of design rule checks (report_drc)		8/31/18 2:34 PM	1.4 KB
impl_1_route_report_methodology_0	Report on error or violations against a set of methodology checks (report_methodology)		8/31/18 2:34 PM	1.3 KB
impl_1_route_report_power_0	Report power analysis details (report_power)		8/31/18 2:34 PM	7.7 KB
impl_1_route_report_route_status_0	Report on status of the routing. (report_route_status)		8/31/18 2:34 PM	0.6 KB
impl_1_route_report_timing_summary_0	Report timing summary (report_timing_summary)		8/31/18 2:34 PM	7.2 KB
impl_1_route_report_incremental_reuse_0	Report on achievable incremental reuse for the given design-checkpoint (report_incremental_reuse)			
impl_1_route_report_clock_utilization_0	Report information about clock nets in design (report_clock_utilization)		8/31/18 2:34 PM	6.3 KB
impl_1_route_report_bus_skew_0	Report on calculated bus skew among the signals constrained by set_bus_skew (report_bus_skew)		8/31/18 2:34 PM	0.8 KB
impl_1_route_implementation_log_0	Vivado Implementation Log		8/31/18 2:34 PM	20.9 KB
Post-Route Phys Opt Design (post_route_phys_opt_design)				

VLSI for DSP

Perform Timing Simulation

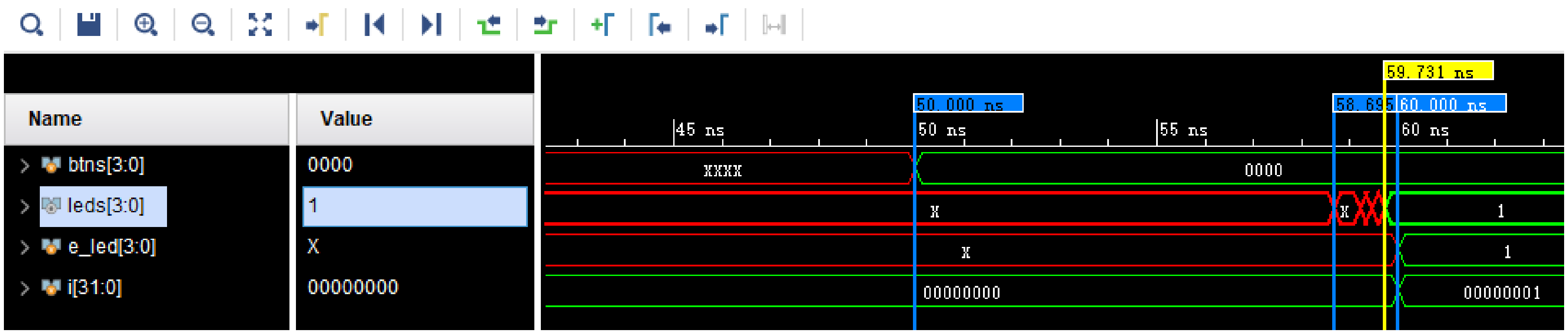
Department of Electronic Engineering
Chang Guan University
chenyh@mail.cgu.edu.tw



Run a timing simulation

1. Select **Run Simulation > Run Post-Implementation Timing Simulation** process under the *Simulation* tasks of the *Flow Navigator* pane.
   - The Vivado simulator will be launched using the implemented design and **lab1_tb** as the top-level module.
   - Using the Windows Explorer, verify that **timing** directory is created under the **lab1.sim > sim_1 > impl** directory. The **timing** directory contains generated files to run the timing simulation.
2. Click on the **Zoom Fit** button to see the waveform window from 0 to 200 ns.
3. Right-click at 50 ns (where the btns input is set to 0000b) and select **Markers > Add Marker**.
4. Similarly, right-click and add a marker at around 58.000 ns where the **leds** changes.
5. You can also add a marker by clicking on the Add Marker button ( ). Click on the **Add Marker** button and left-click at around 60 ns where **e_led** changes.

Timing Simulation



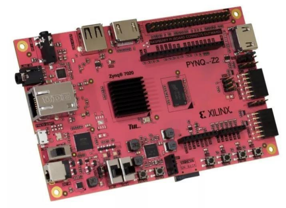
– Notice that we monitored the expected led output at 10 ns after the input is changed (see the testbench) whereas the actual delay is about 8 to 9.7 ns (depending on the board).

6. Close the simulator by selecting **File > Close Simulation** without saving any changes.

VLSI for DSP

Generate the Bitstream and Verify Functionality

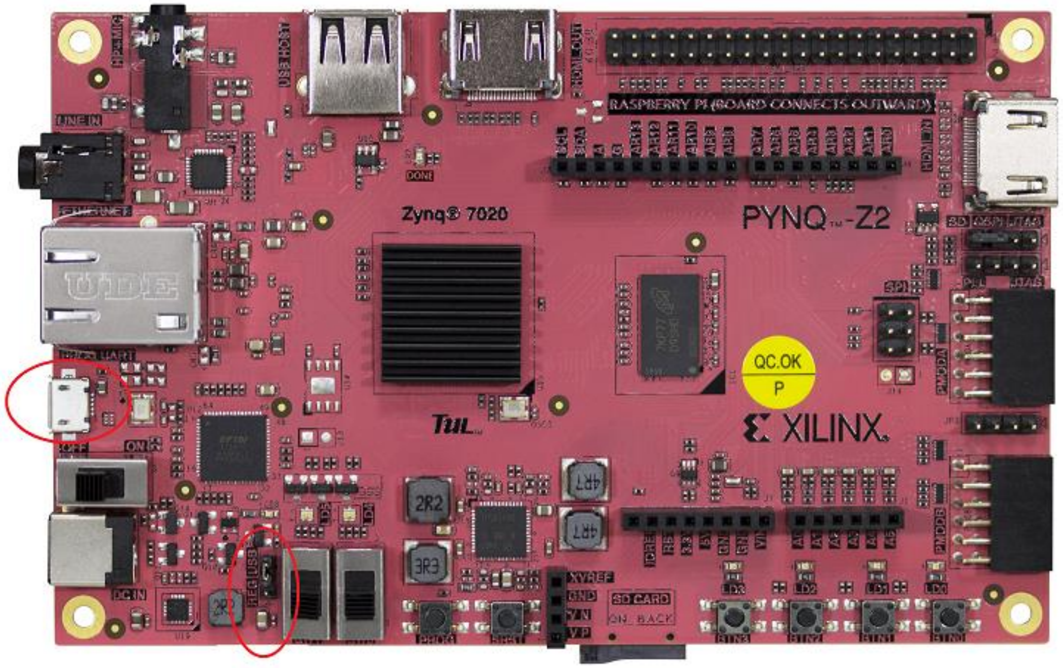
Department of Electronic Engineering
Chang Guan University
chenyh@mail.cgu.edu.tw



Generate the Bitstream

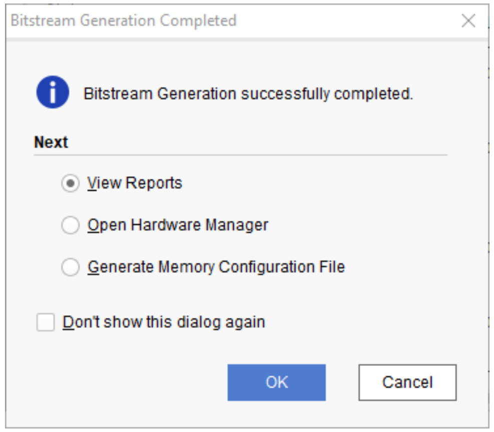
- **Connect the board and power it ON. Generate the bitstream, open a hardware session, and program the FPGA.**

1. Make sure that the Micro-USB cable is connected to the JTAG PROG connector.
2. The PYNQ-Z2 can be powered through USB power via the JTAG PROG.
   – Make sure that the board is set to use USB power.



Generate the Bitstream

3. Power **ON** the board.
4. Click on the **Generate Bitstream** entry under the *PROGRAM AND DEBUG* tasks of the *Flow Navigator* pane.
   - The bitstream generation process will be run on the implemented design. When the process is completed a *Bitstream Generation Completed* dialog box with three options will be displayed.



Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

(●) View Reports
( ) Open Hardware Manager
( ) Generate Memory Configuration File

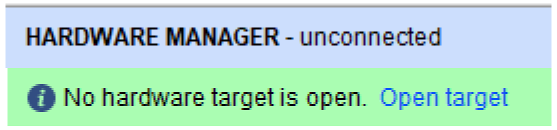
[ ] Don't show this dialog again

OK | Cancel

   - This process will have generated a **lab1.bit** file under the **impl_1** directory in the **lab1.runs** directory.

Generate the Bitstream

5. Select the *Open Hardware Manager* option and click **OK**.
   – The Hardware Manager window will open indicating "unconnected" status.
6. Click on the **Open target** link.

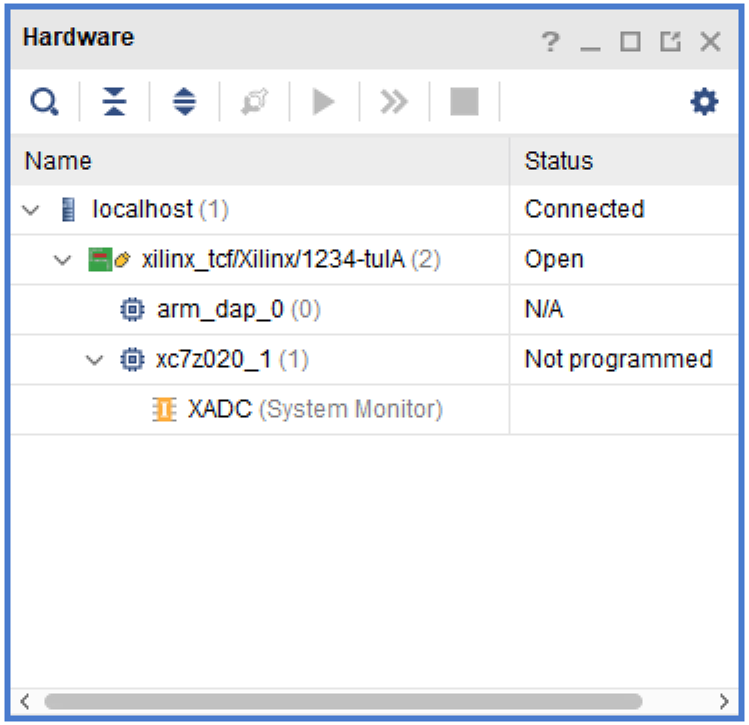


HARDWARE MANAGER - unconnected

No hardware target is open. Open target

Download

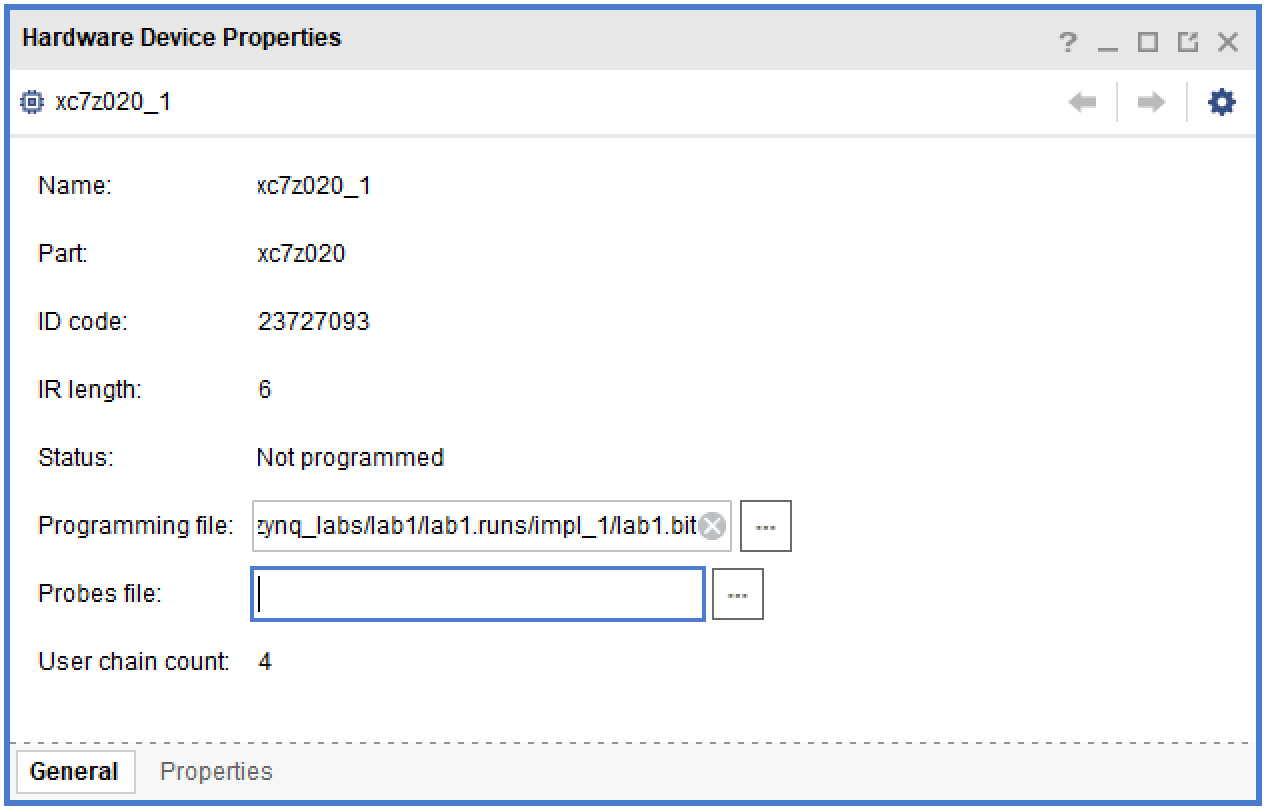
7. From the dropdown menu, click **Auto Connect.**
   - The Hardware Session status changes from Unconnected to the server name and the device is highlighted. Also notice that the Status indicates that it is not programmed.



Name	Status
localhost (1)	Connected
xilinx_tcf/Xilinx/1234-tulA (2)	Open
arm_dap_0 (0)	N/A
xc7z020_1 (1)	Not programmed
XADC (System Monitor)	

Download

- Select the device and verify that the lab1.bit is selected as the programming file in the General tab.



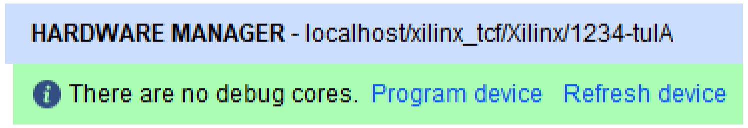
Hardware Device Properties

xc7z020_1

Name:	xc7z020_1
Part:	xc7z020
ID code:	23727093
IR length:	6
Status:	Not programmed
Programming file:	zynq_labs/lab1/lab1.runs/impl_1/lab1.bit
Probes file:	
User chain count:	4

Download

8. Click on the *Program device* link in the green information bar to program the target FPGA device. Another way is to right click on the device and select *Program Device.*


HARDWARE MANAGER - localhost/xilinx_tcf/Xilinx/1234-tulA

There are no debug cores. Program device Refresh device

Selecting to program the FPGA

Download

9. Click **Program** to program the FPGA.
   - The DONE LED will lit when the device is programmed. You may see some other LEDs lit depending on switch positions.
10. Verify the functionality by flipping the switches and observing the output on the LEDs (Refer to the earlier logic diagram).
11. When satisfied, power **OFF** the board.
12. Close the hardware session by selecting **File > Close Hardware Manager.**
13. Click **OK** to close the session.
14. Close the **Vivado** program by selecting **File > Exit** and click **OK**.