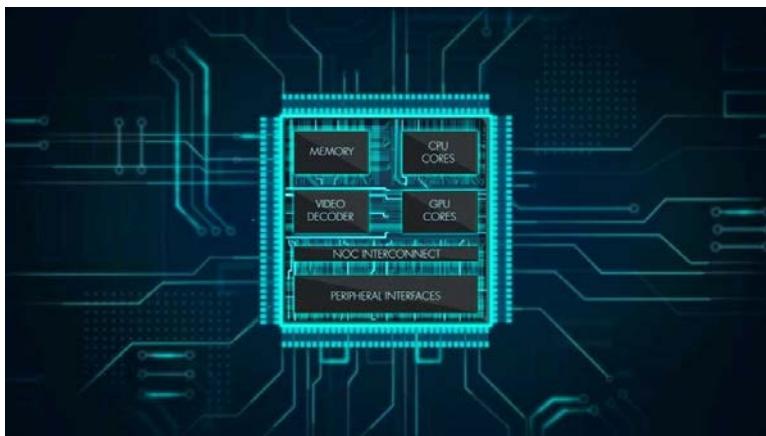


Digital System Design

Term-Project



Instructor : **Prof. Hyun Kim**

Lecture Note : eclass.seoultech.ac.kr

Office : Mirae Hall #526

Web : idsl.seoultech.ac.kr

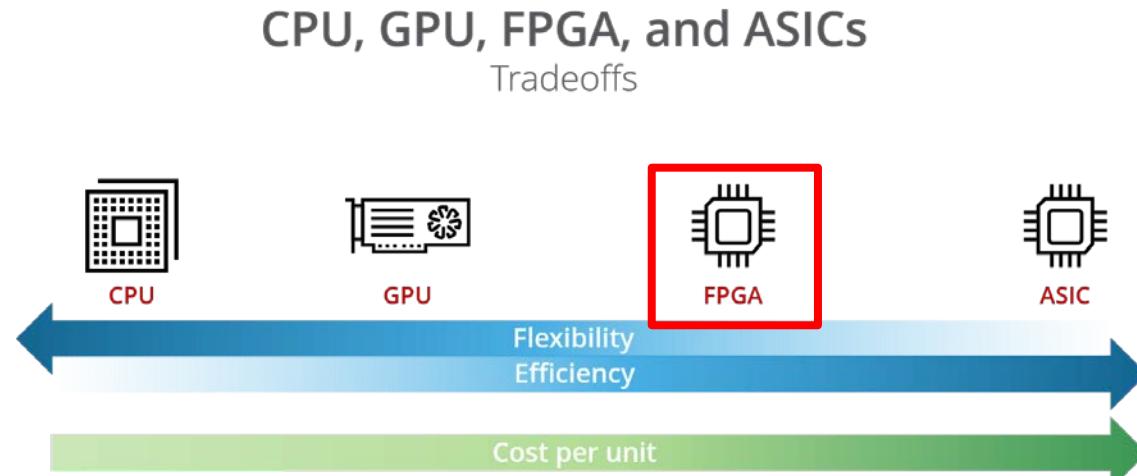
Contact : hyunkim@seoultech.ac.kr / 010-9600-5427

Contents

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ Vivado – Simulation
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

What is a FPGA?

- ◆ Field Programmable Gate Array: FPGA
- ◆ FPGA is a software-reconfigurable hardware substrate
 - Reconfigurable functions
 - Reconfigurable interconnection of functions
 - Reconfigurable input/output (IO)
 - ...
- ◆ When used well, FPGAs:
 - Provide higher performance than running software on a CPU
 - Provide more flexibility of implementation than dedicated hardware on a CPU



◆ Top 2 FPGA Vendors



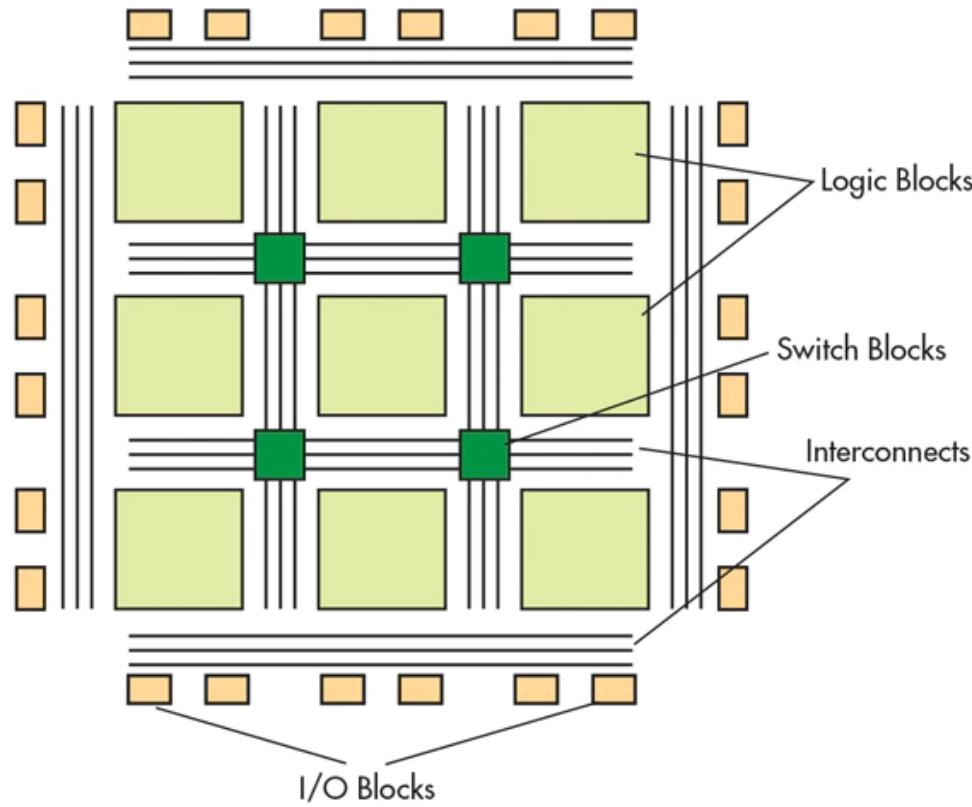
◆ Market Share

Top 5 Company's Sales of FPGA/PLD, in All Applications, Worldwide, (Millions of U.S. Dollars)

Rank 2017	Rank 2018	Company Name	2017	2018	Change	Share of Market
1	1	Xilinx	2,476	2,904	17.3%	51.1%
2	2	Intel	1,858	2,033	9.4%	35.8%
3	3	Microchip (formerly Microsemi)	321	376	17.1%	6.6%
4	4	Lattice Semiconductor	261	285	9.2%	5.0%
		Others	71	85	19.7%	1.5%
		Total Market	4,987	5,683	14.0%	100.0%

Source: Gartner (April 2019)

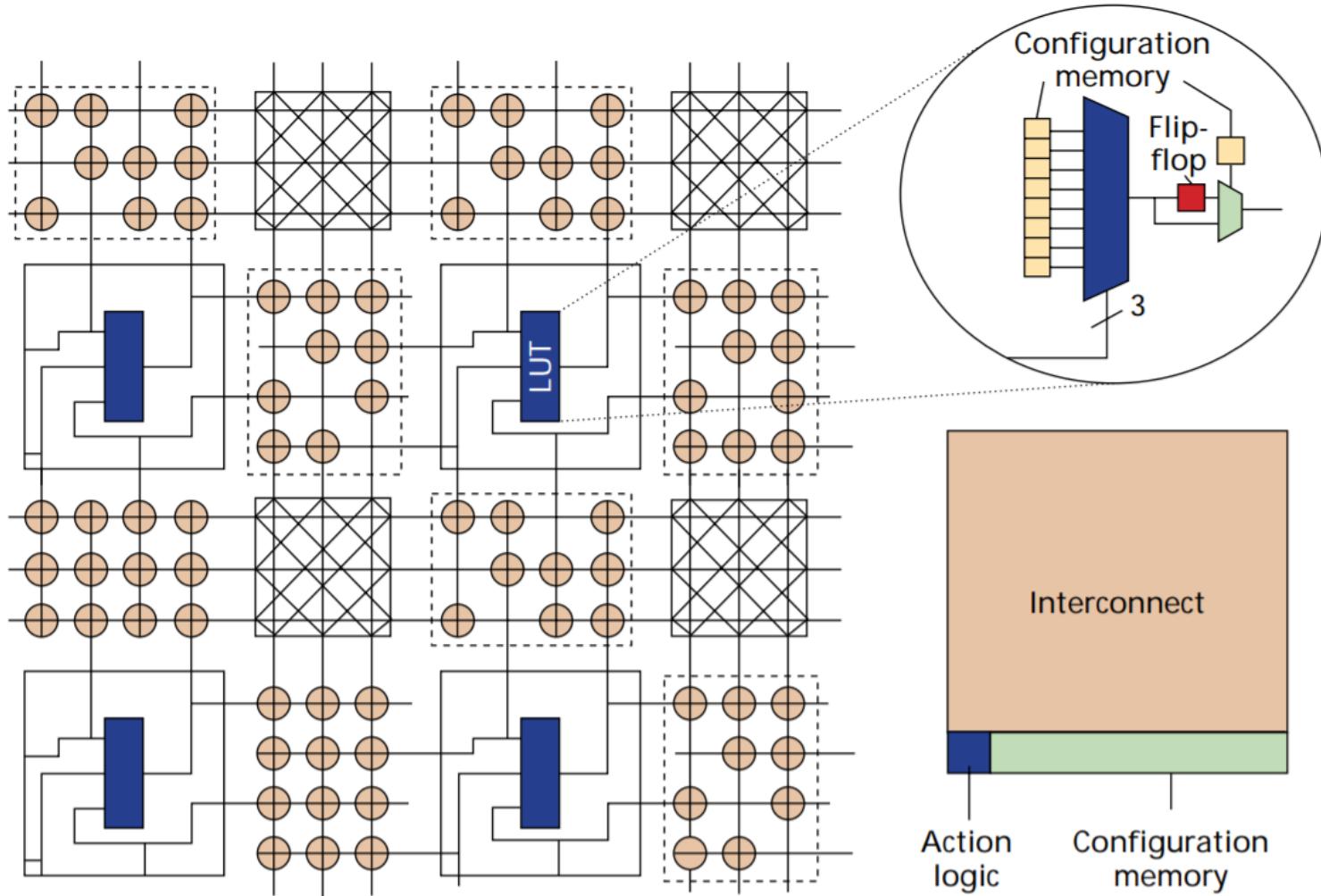
A High-Level Overview of FPGAs



- ◆ We configure logic blocks, their connections, and IO blocks to create hardware circuits and map programs onto those circuits

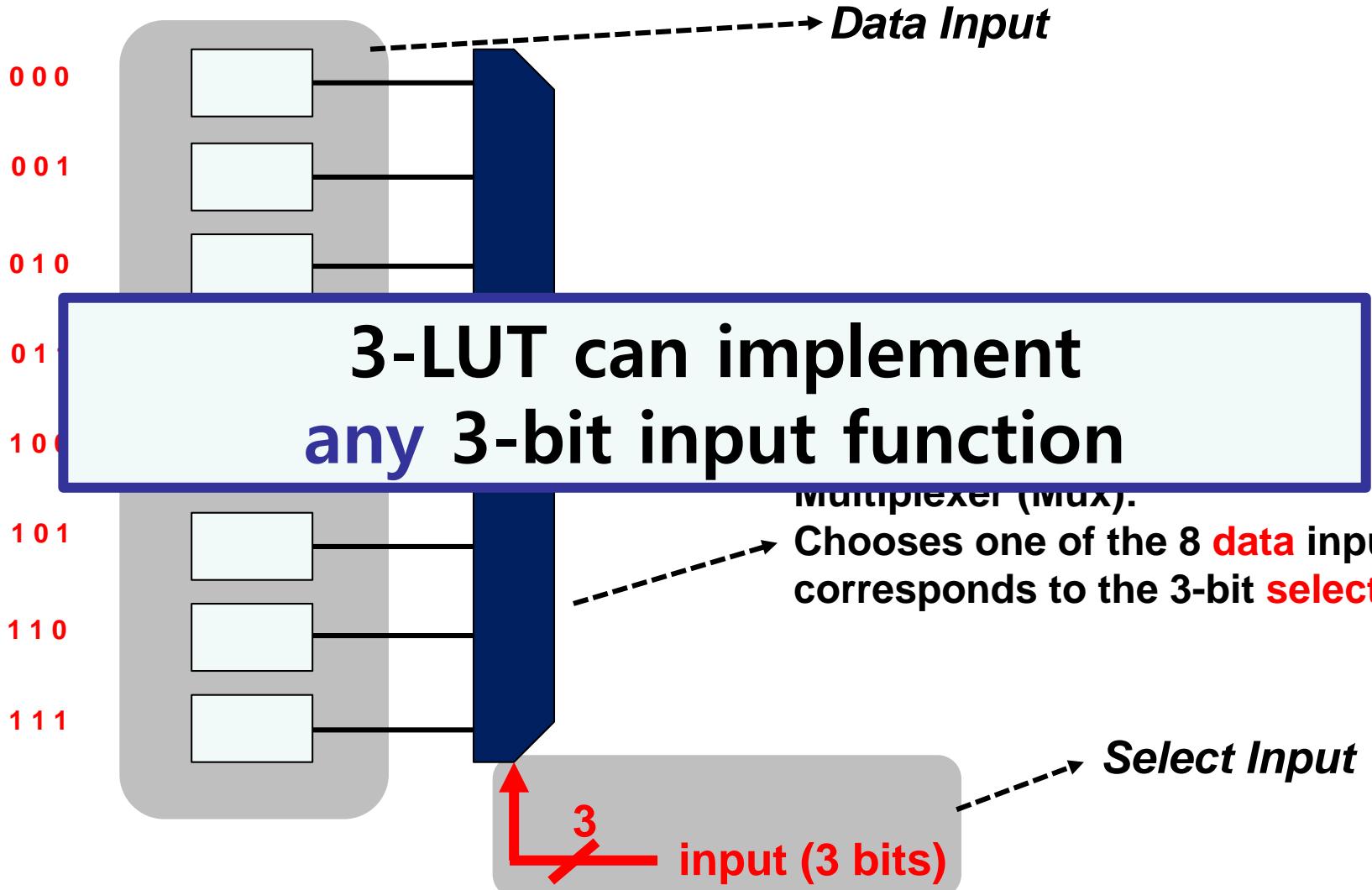
FPGA Architecture

◆ Lookup Tables(LUT) and Switches



Programming LUT

◆ 3-bit input LUT



An Example of Programming a LUT

- ◆ Implementing a function that outputs '1' when there are at least two '1's in a 3-bit input

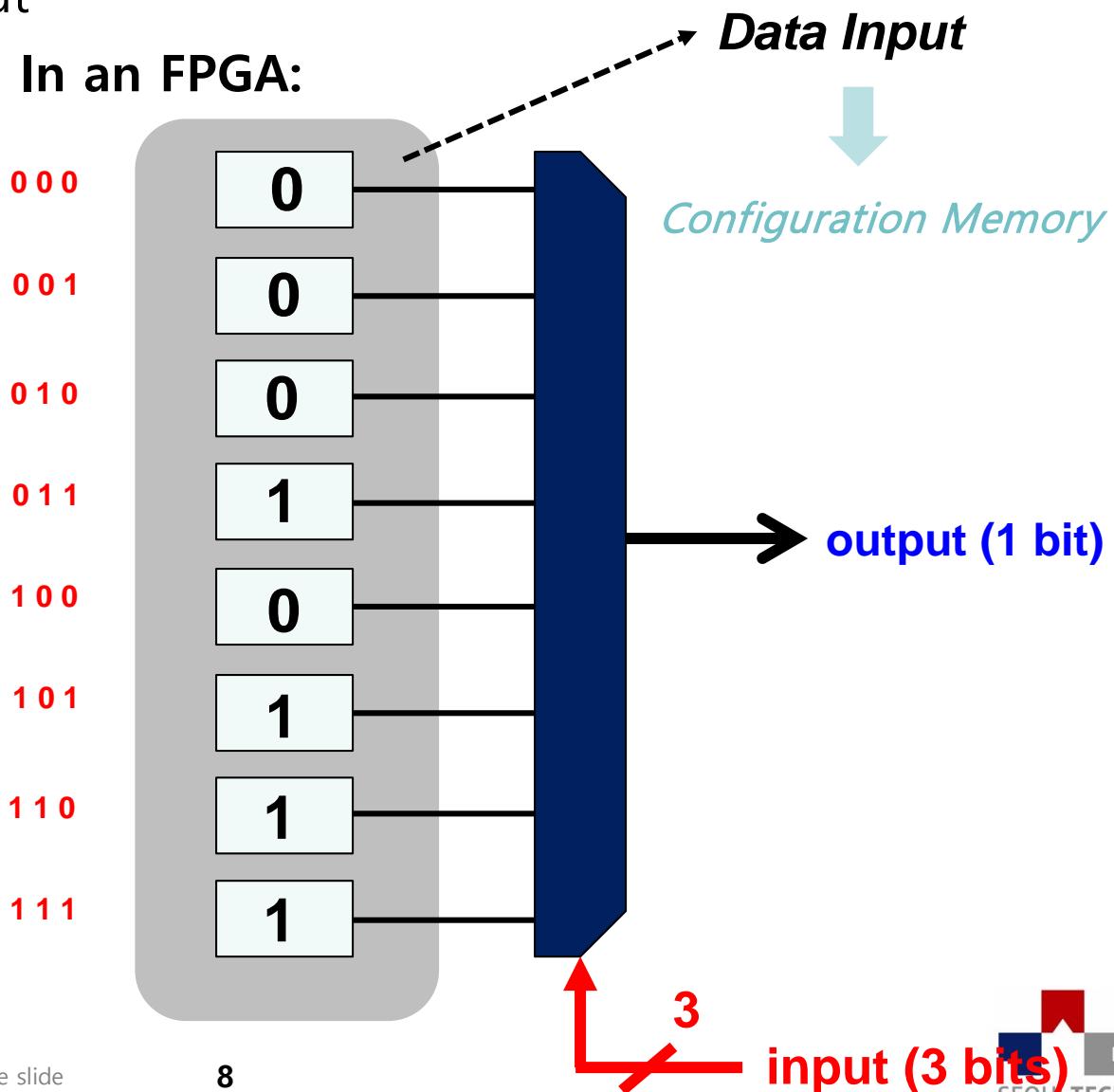
In C:

```
int count = 0;
for(int i = 0; i < 3; i++) {
    count += input & 1;
    input = input >> 1;
}

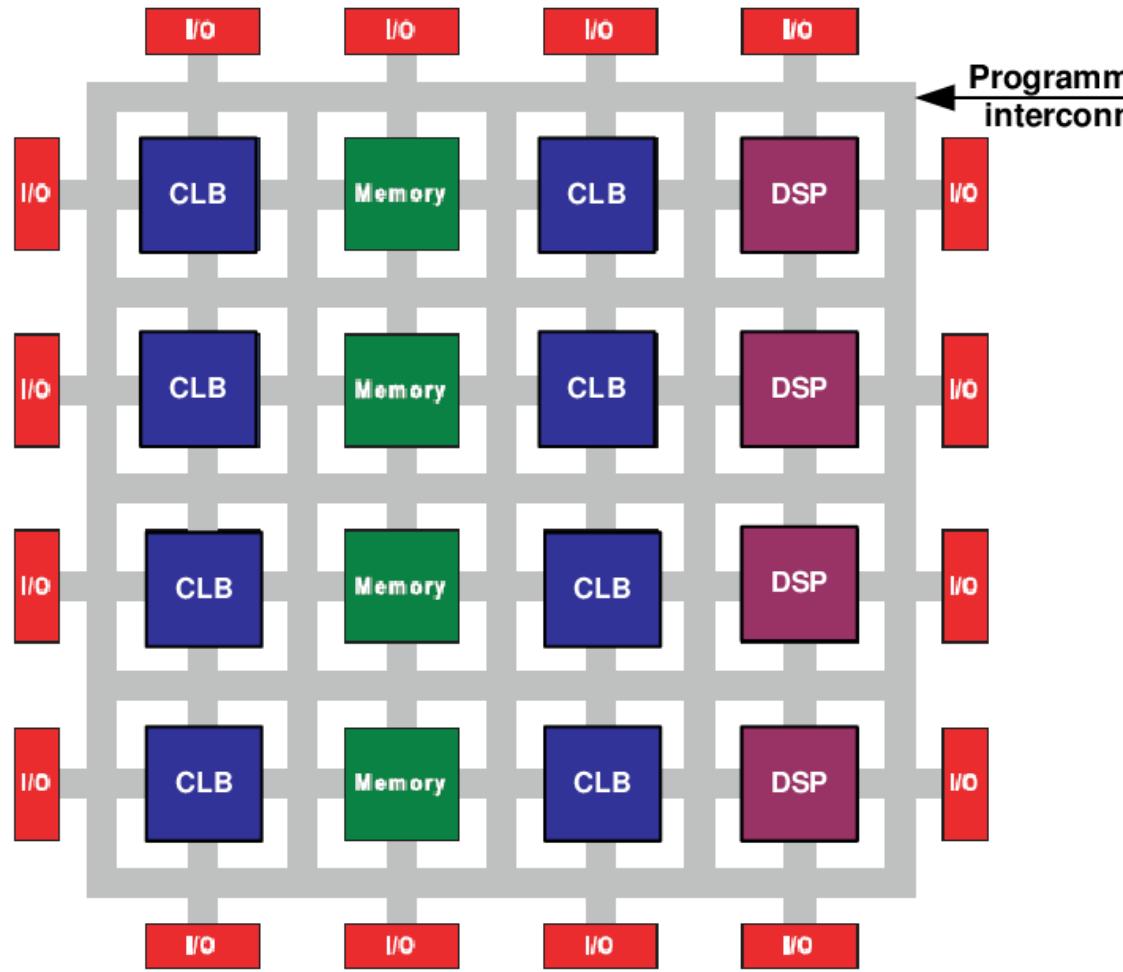
if(count > 1) return 1;

return 0;
```

```
switch(input){
    case 0:
    case 1:
    case 2:
    case 4:
        return 0;
    default:
        return 1;}
```



Xilinx's FPGA Architecture



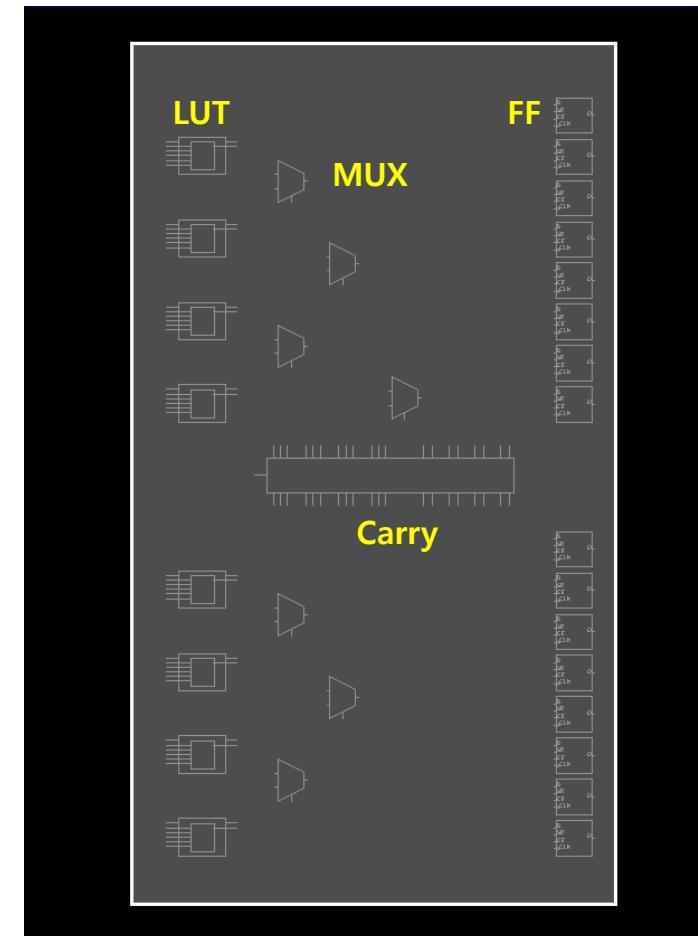
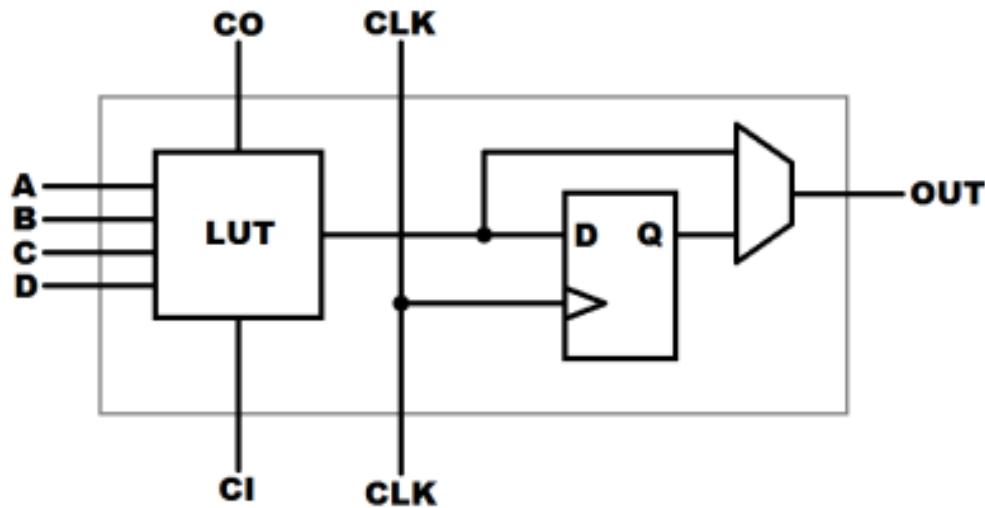
Configurable Logic Block (CLB)

Image source: In book: Digital and Analogue Electronics Circuits and Systems (pp.53)

Components of FPGAs

◆ CLB (Configurable Logic Block)

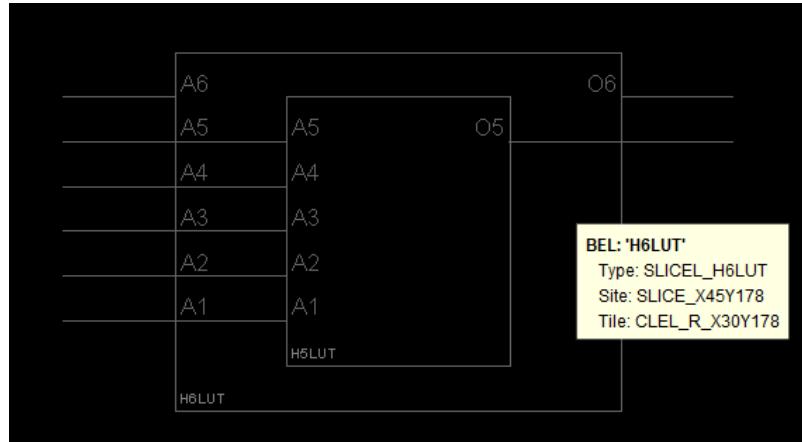
An individual CLB consists of a number of discrete logic components itself, such as look-up tables(LUTs) and flip-flops.



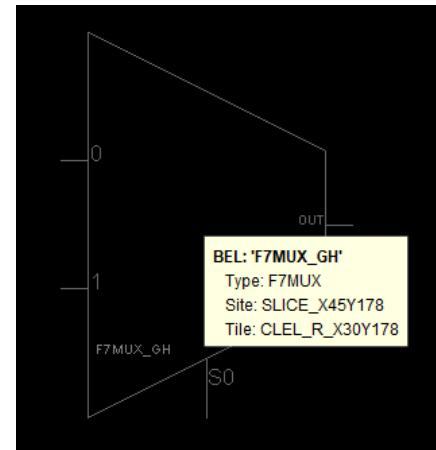
Components of FPGAs

◆ These components differ depending on FPGA Chips

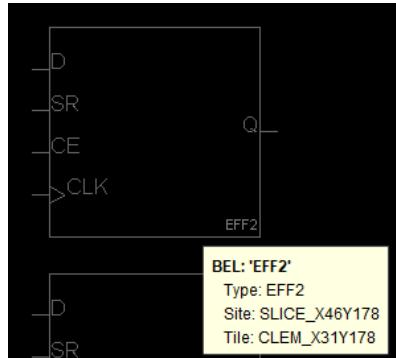
LUT



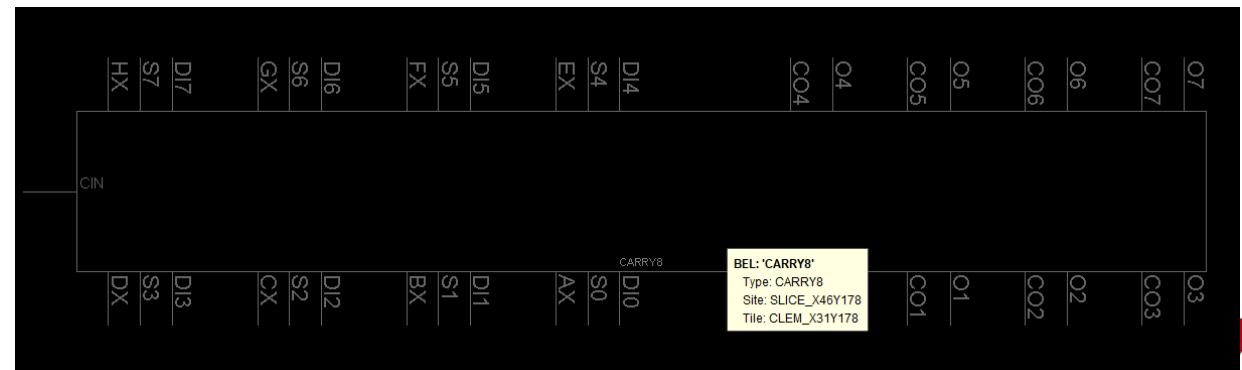
MUX



FF

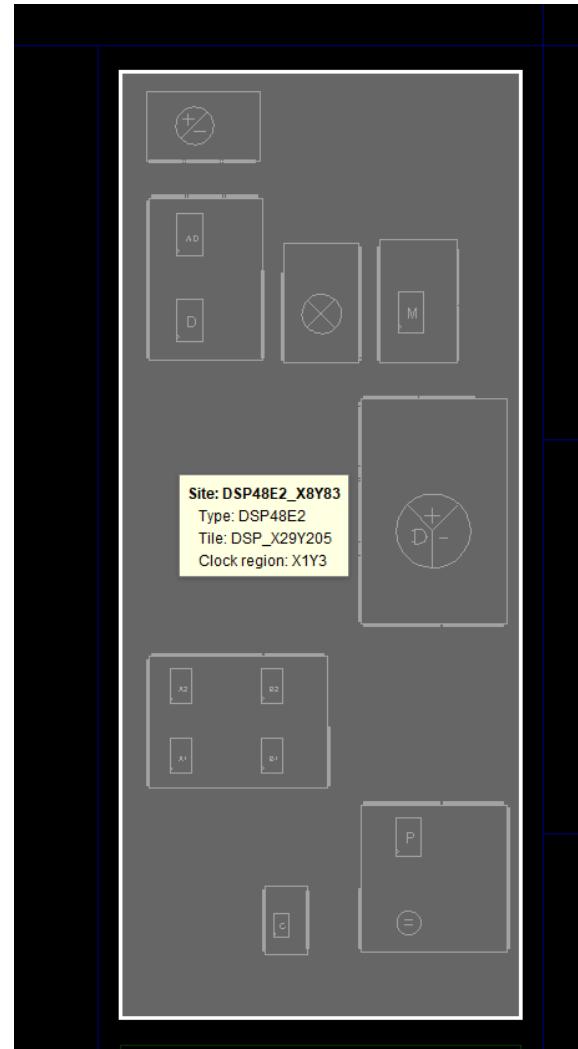
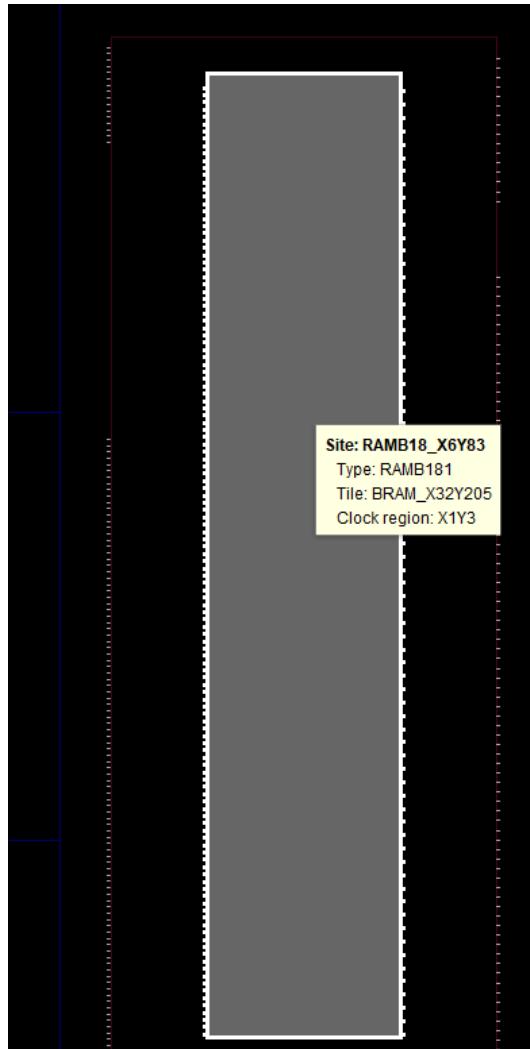


CARRY



Components of FPGAs

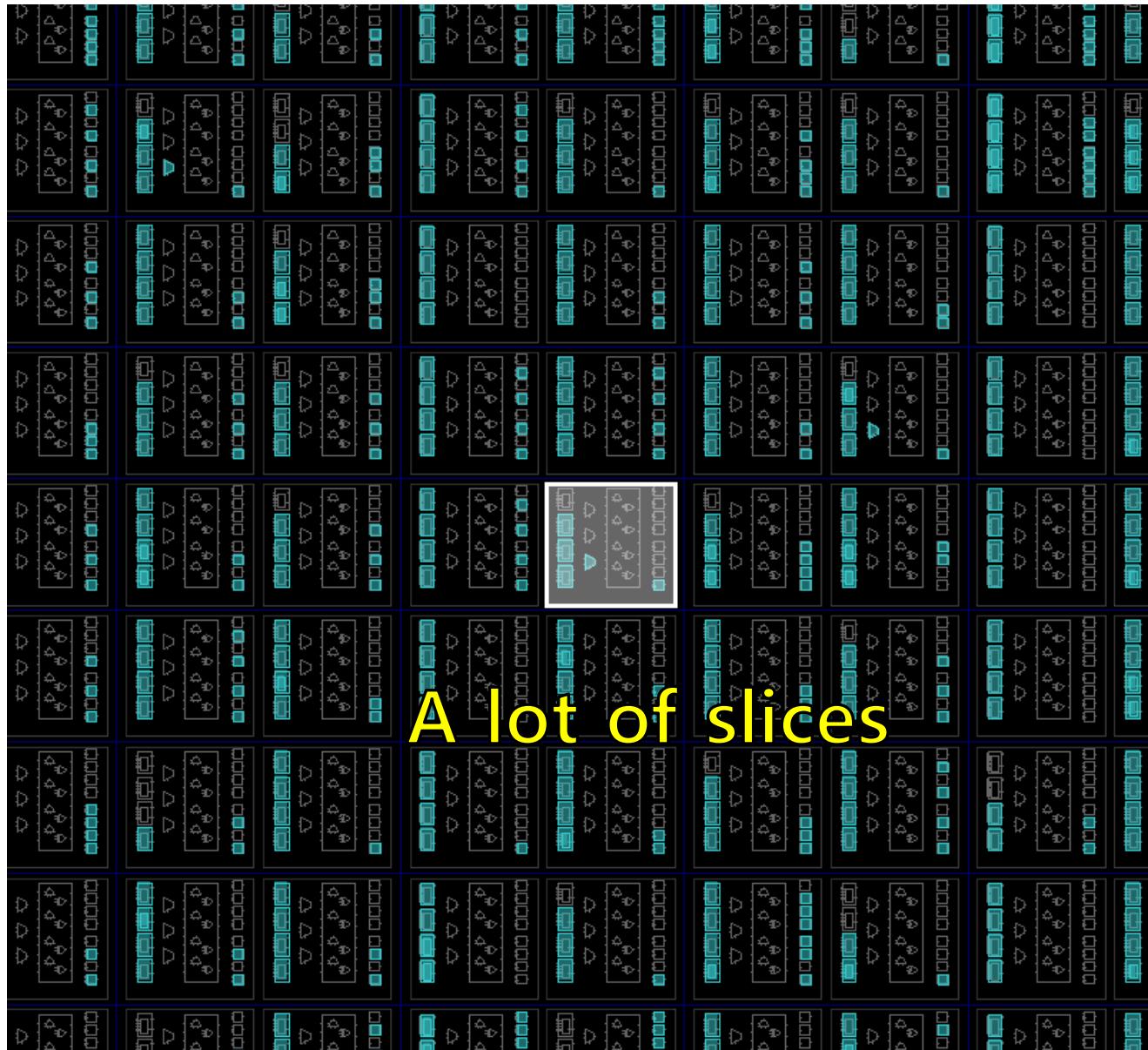
◆ MEMORY & DSP



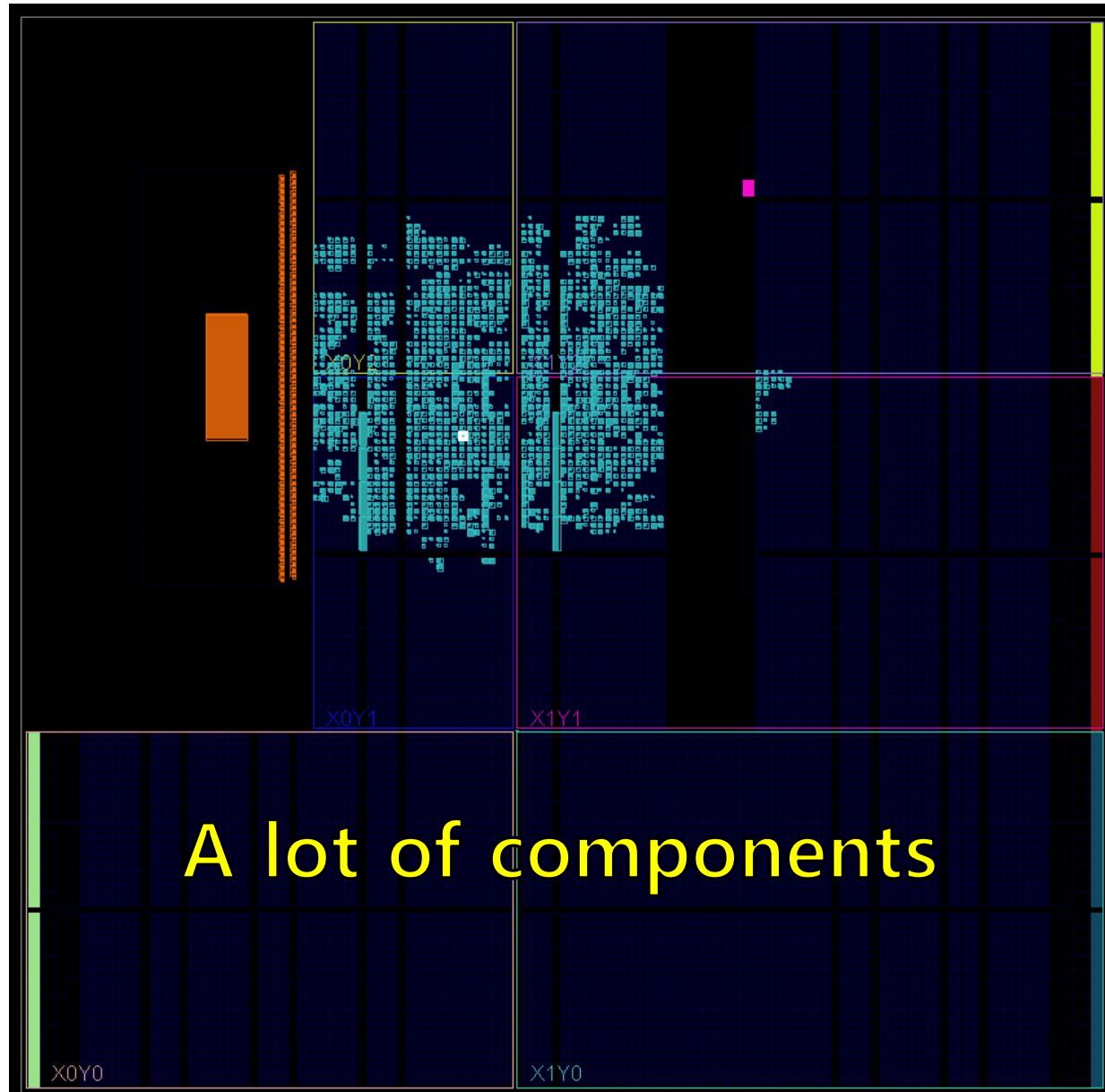
Components of FPGAs



Components of FPGAs

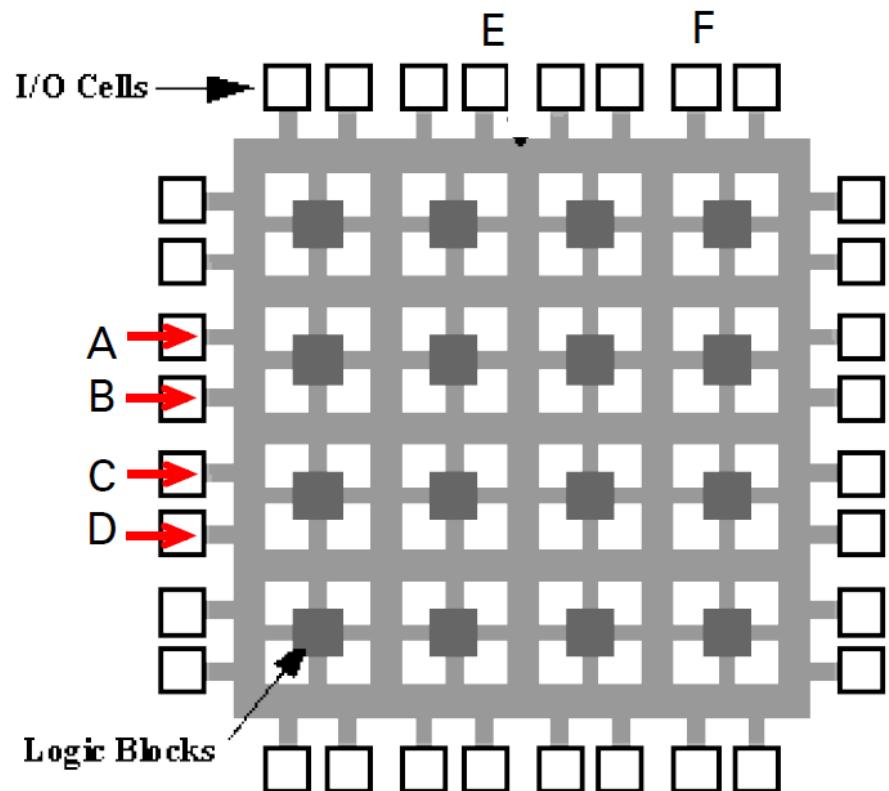


Components of FPGAs



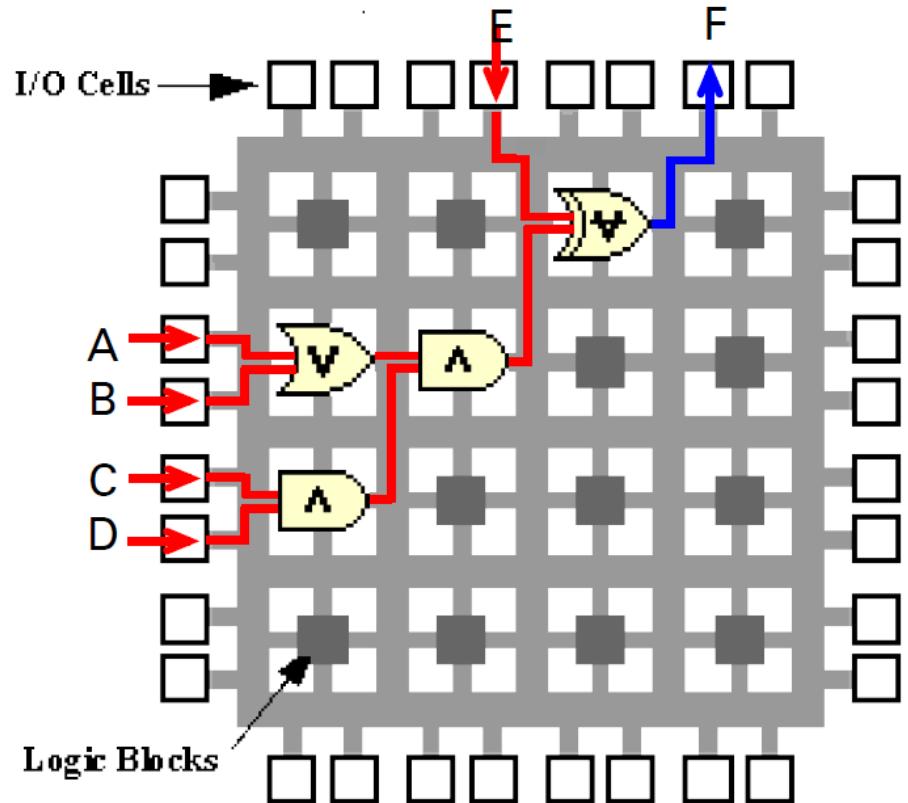
Example of Implementing Logic

Implementing
Logic on FPGA:
 $F = \{(A+B)CD\} \oplus E$



Example of Implementing Logic

Implementing
Logic on FPGA:
 $F = \{(A+B)CD\} \oplus E$



Basys 3: A low-end FPGA Board

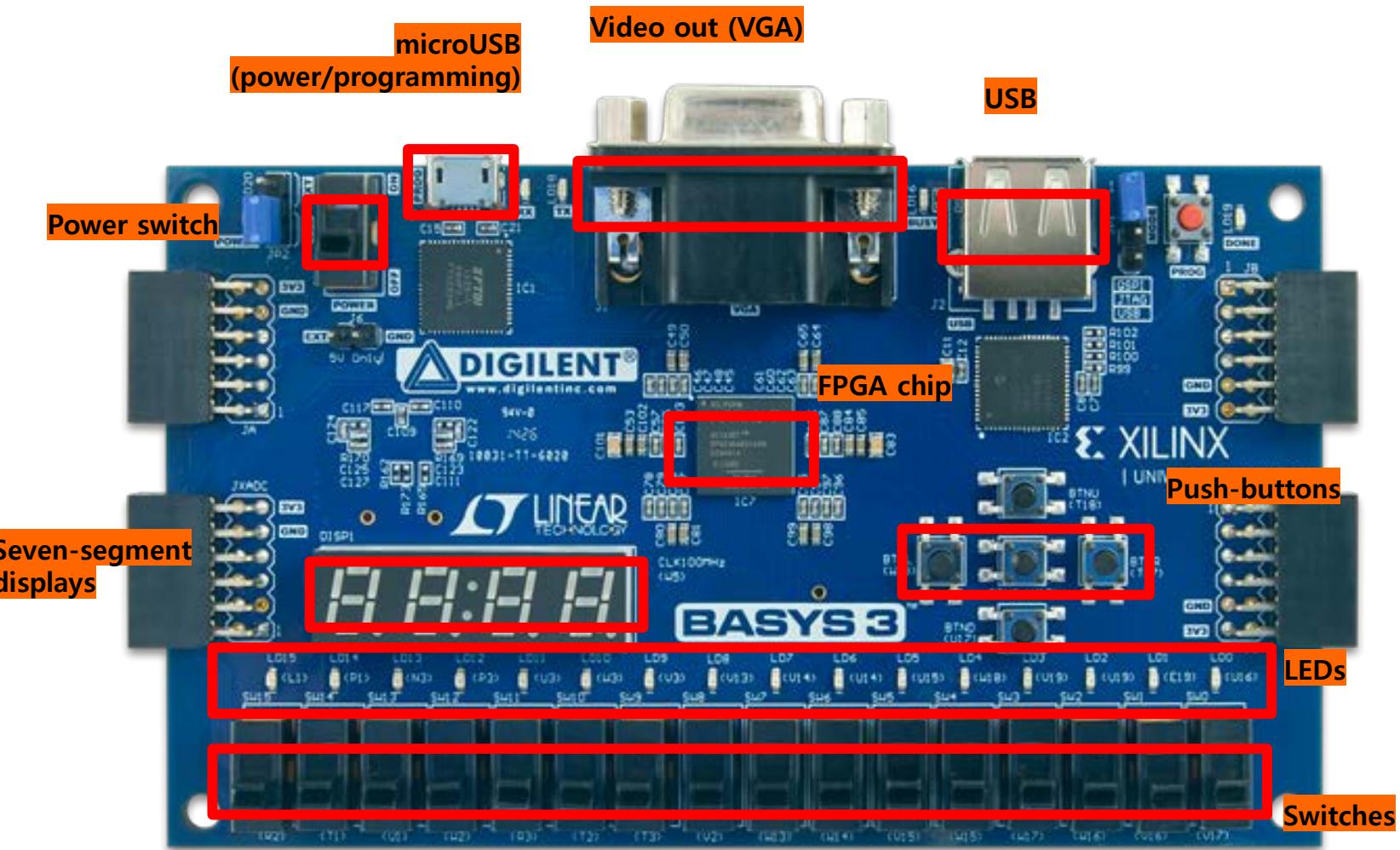
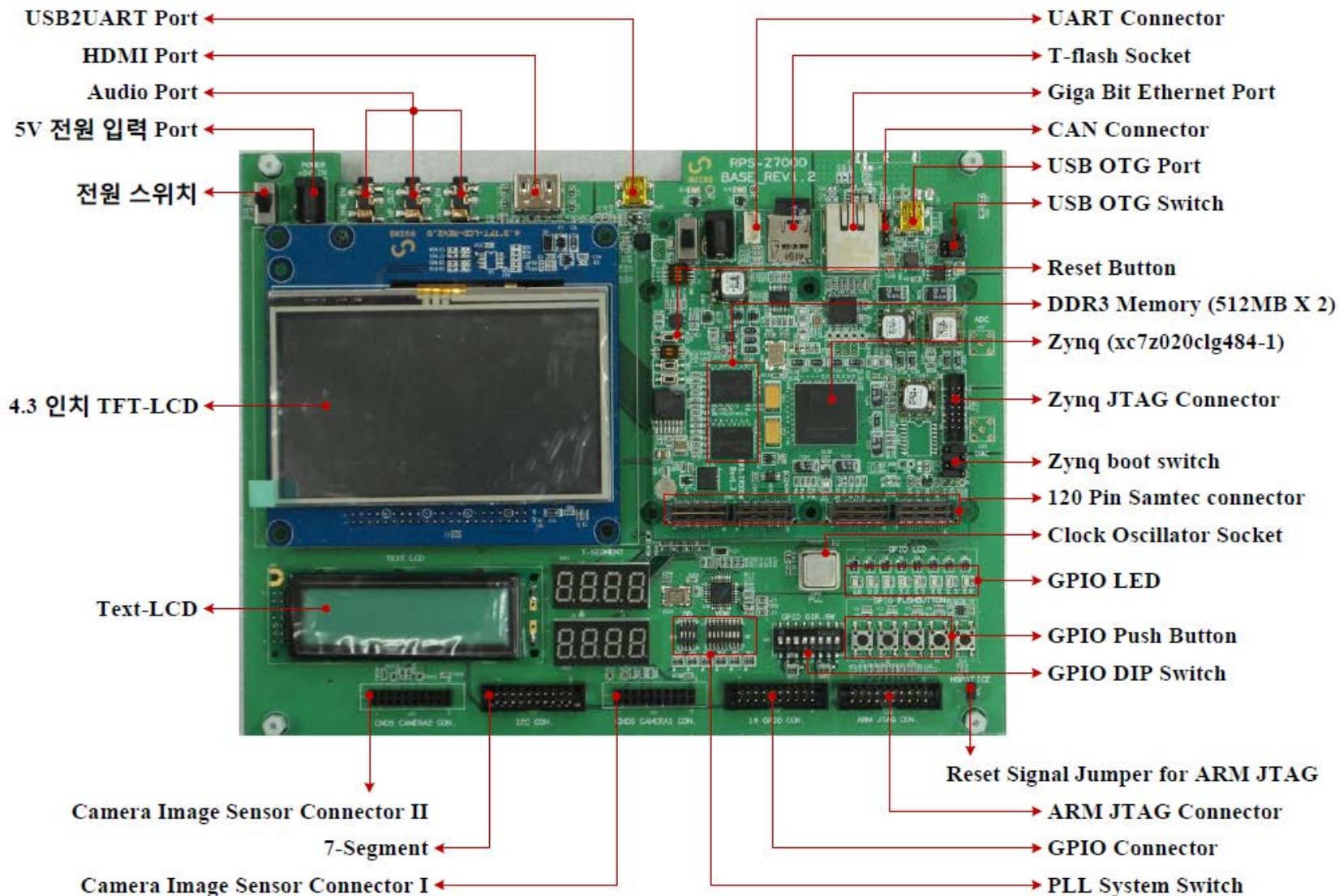


image source: onur-digitaldesign_comparach lecture slide

RPS-ZYNQ7000

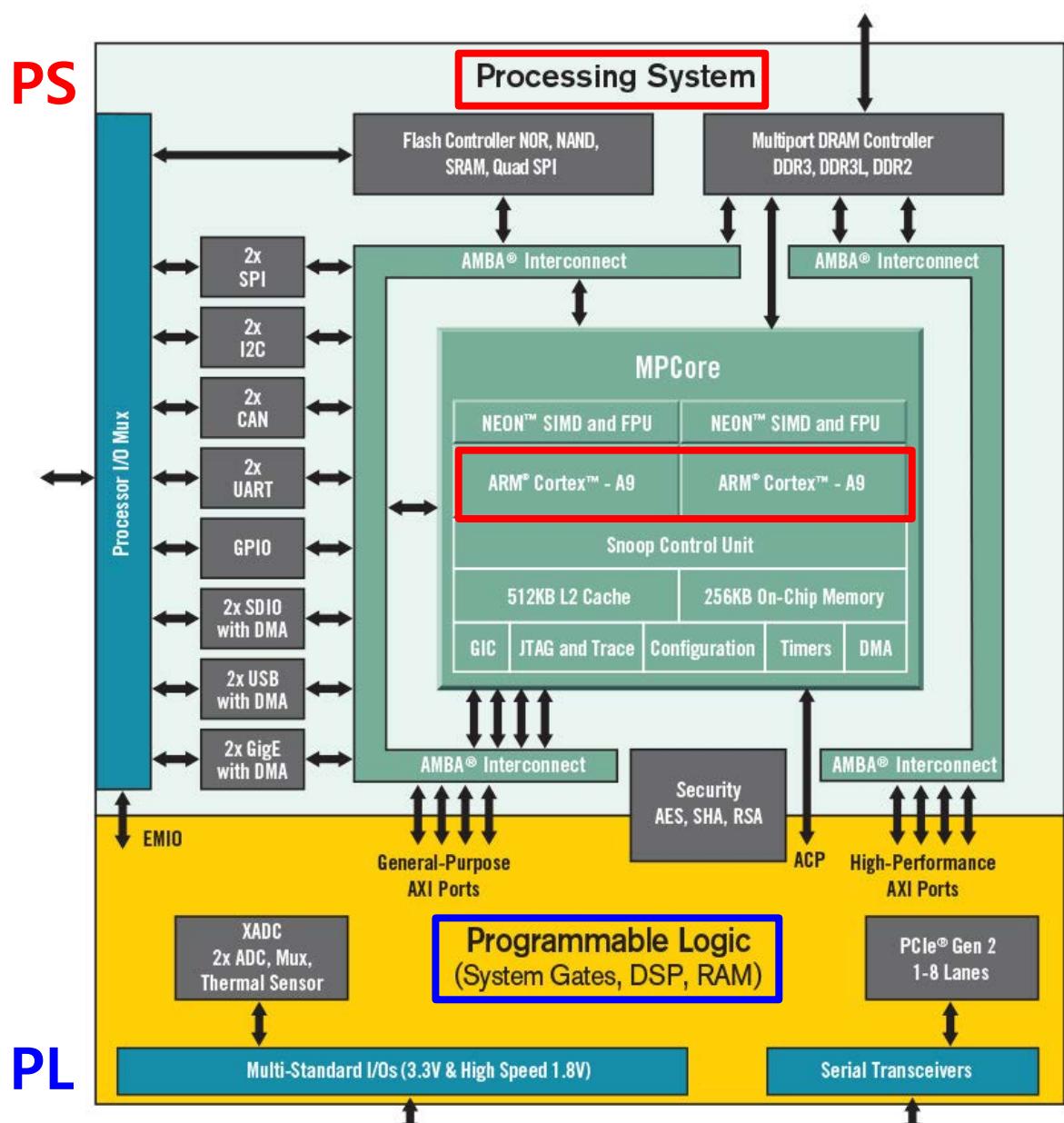


What is a ZYNQ?

- ◆ Xilinx SoCs are processor-centric platforms that offer software, hardware and I/O programmability in a single chip. The Zynq-7000 family is based on the SoC architecture. Zynq-7000 products incorporate a dual core ARM Cortex-A9 based Processing System (PS) and Xilinx Programmable Logic in a single device.

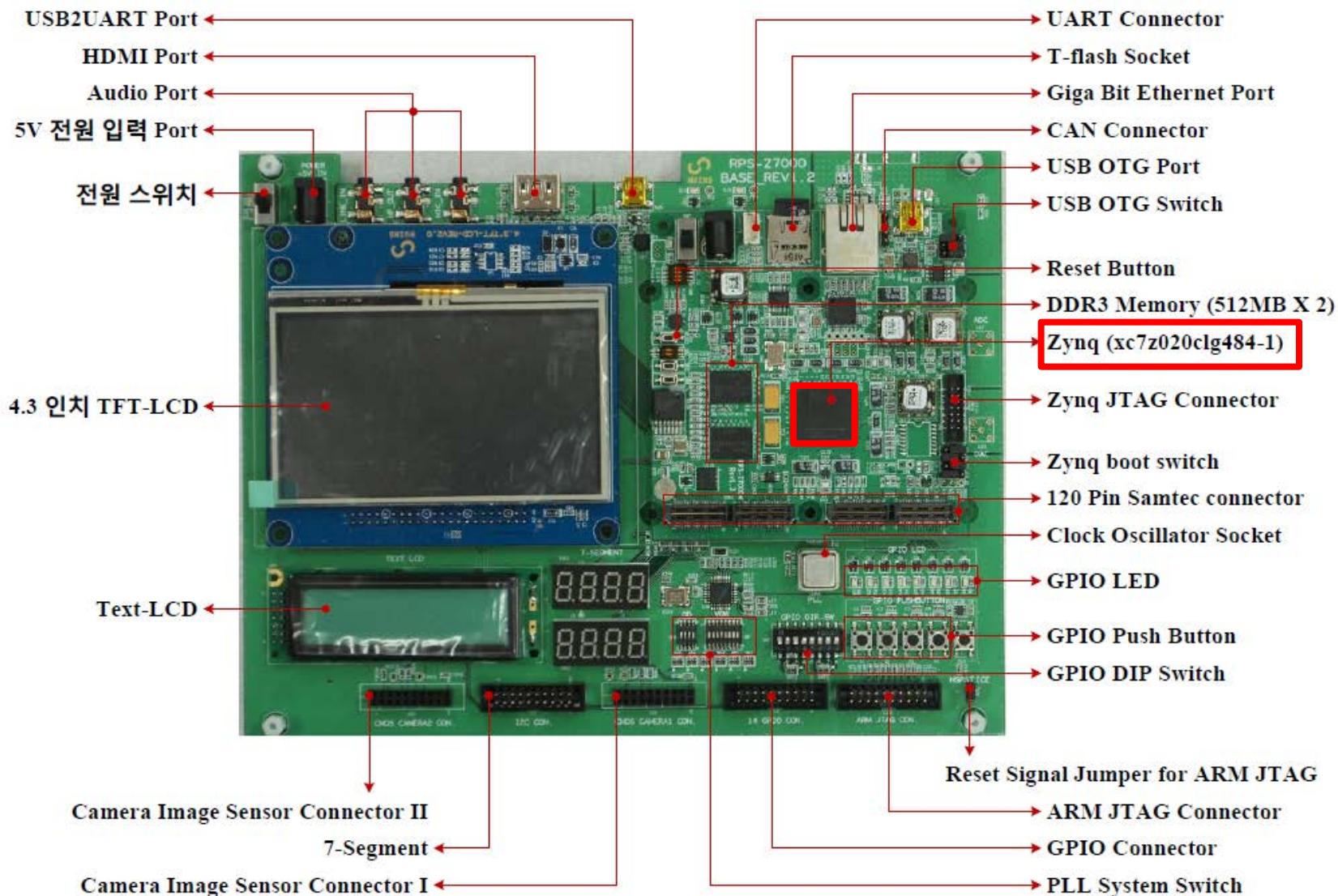


What is a ZYNQ?

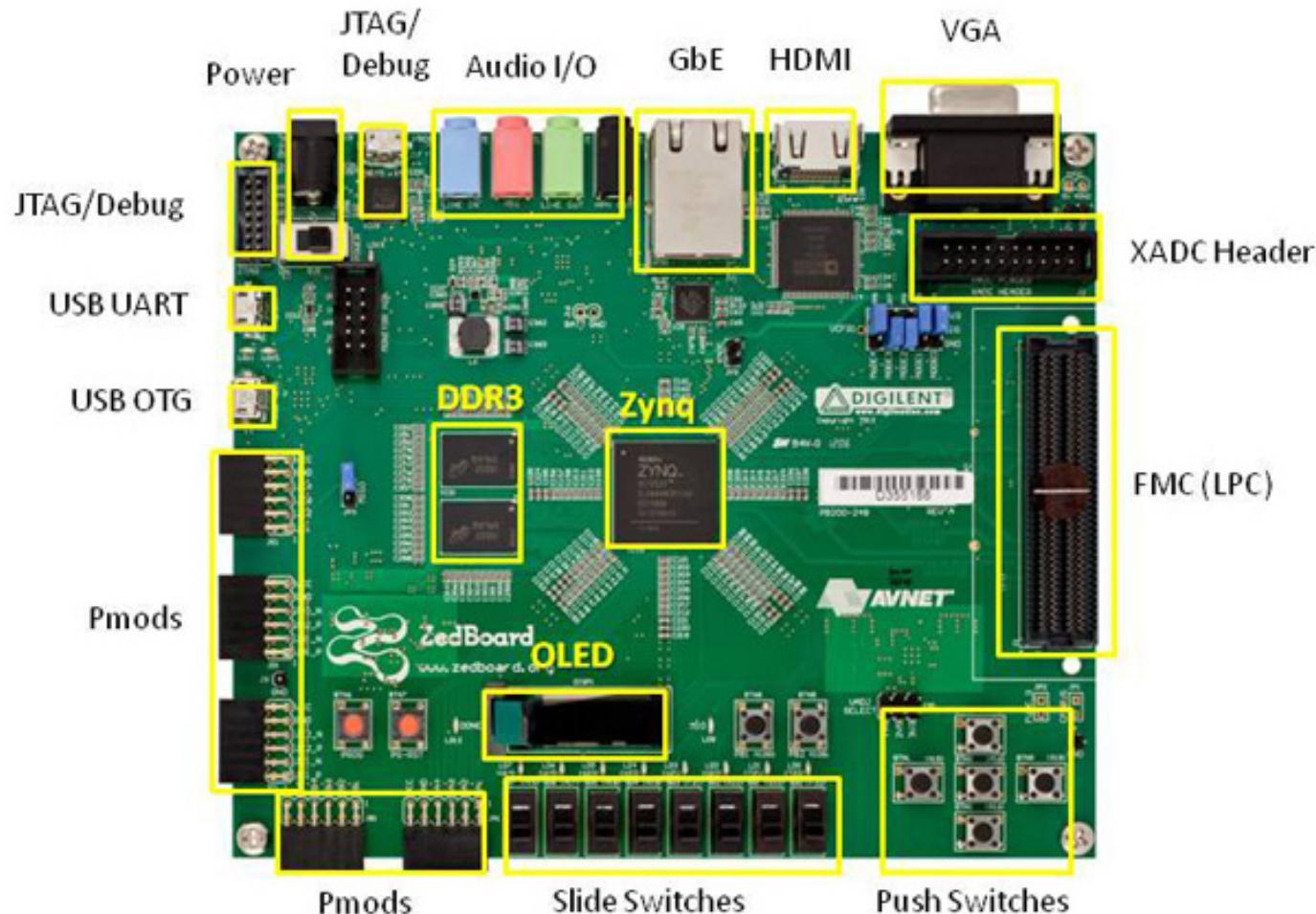


PL

RPS-ZYNQ7000



ZedBoard: Same FPGA chip that we use



* SD card cage and QSPI Flash reside on backside of board

ZedBoard: Same FPGA chip that we use

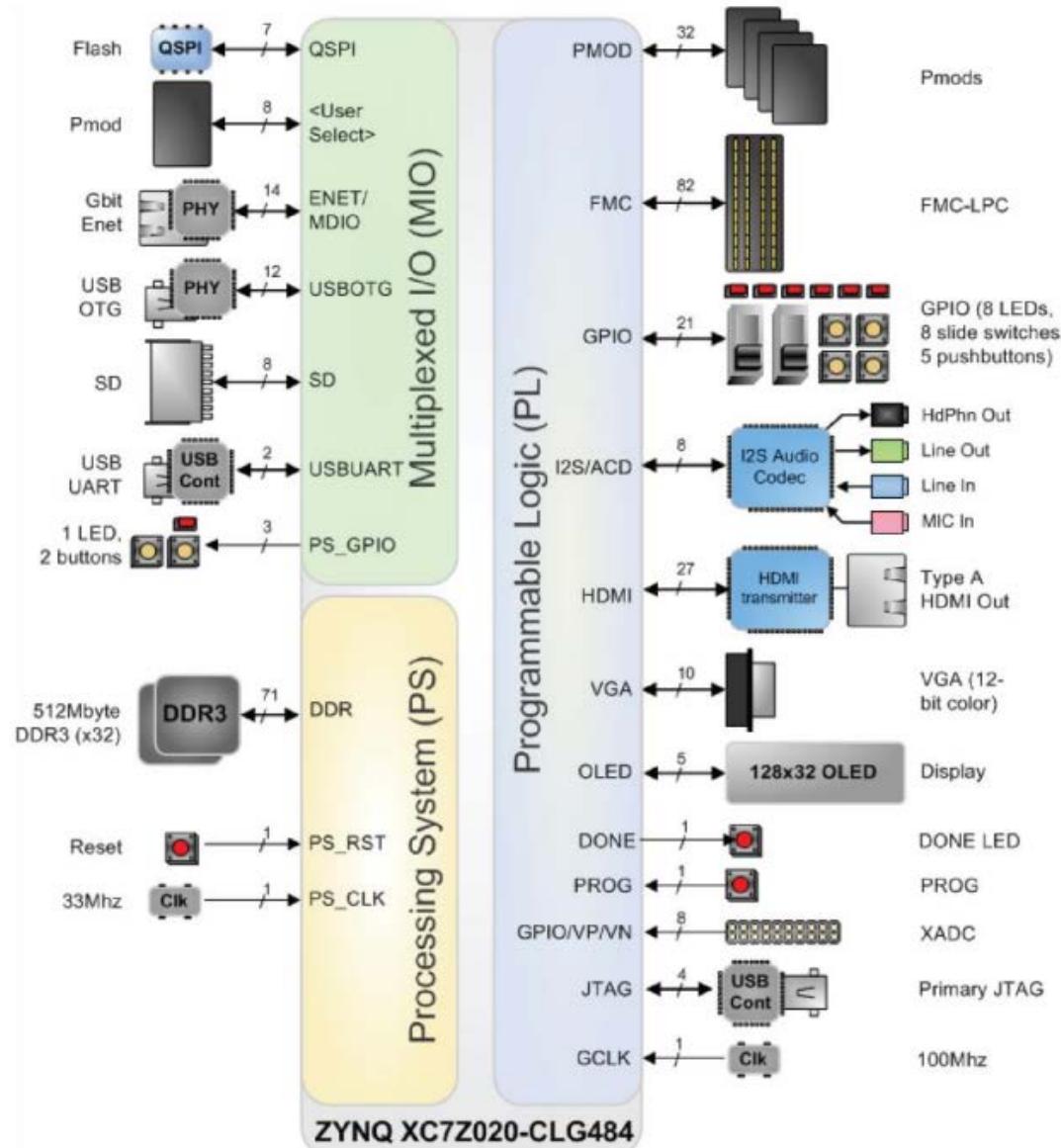


Figure 1 - ZedBoard Hardware Block Diagram

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ Vivado – Simulation
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

Design Flow

- ◆ We are going to use Vivado tool in order to model and verify design.
- ◆ After verifying design, we are going to program design(bit file) onto FPGAs.

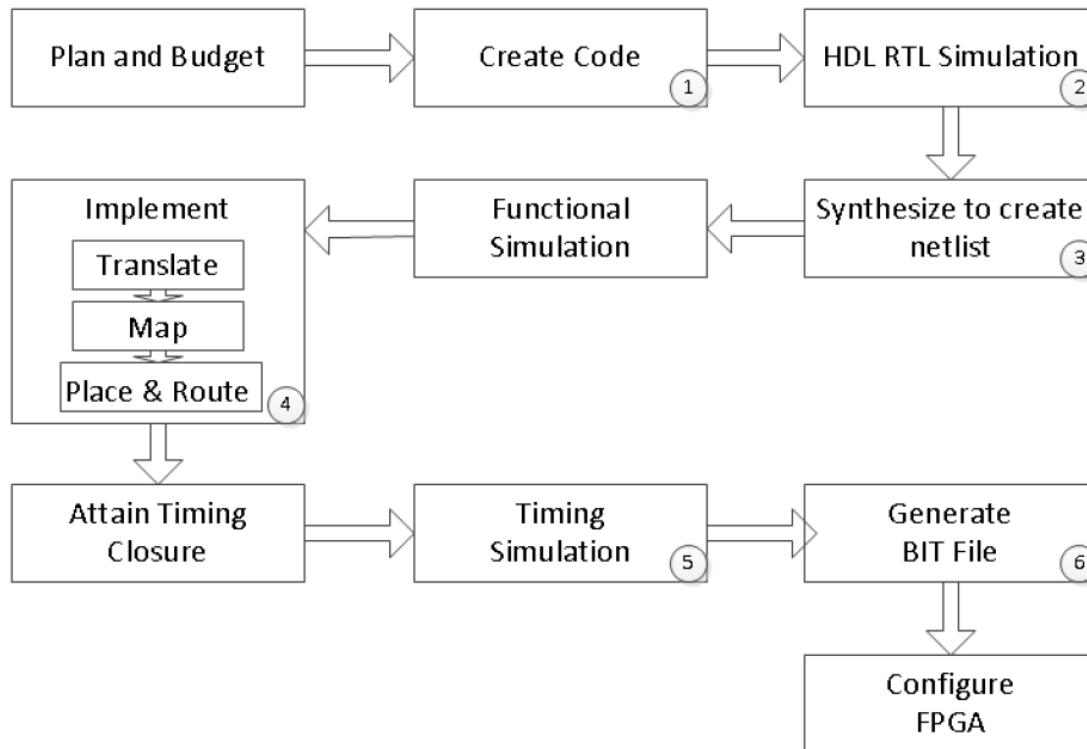


Figure 1. A typical design flow

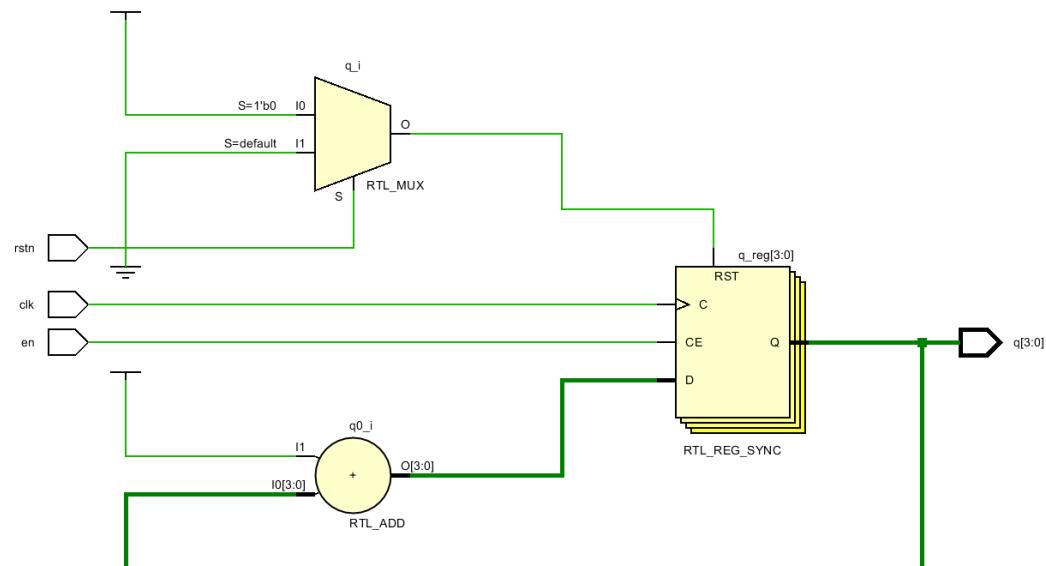
Design Flow

◆ Before Synthesis

HDL Code, RTL Modeling

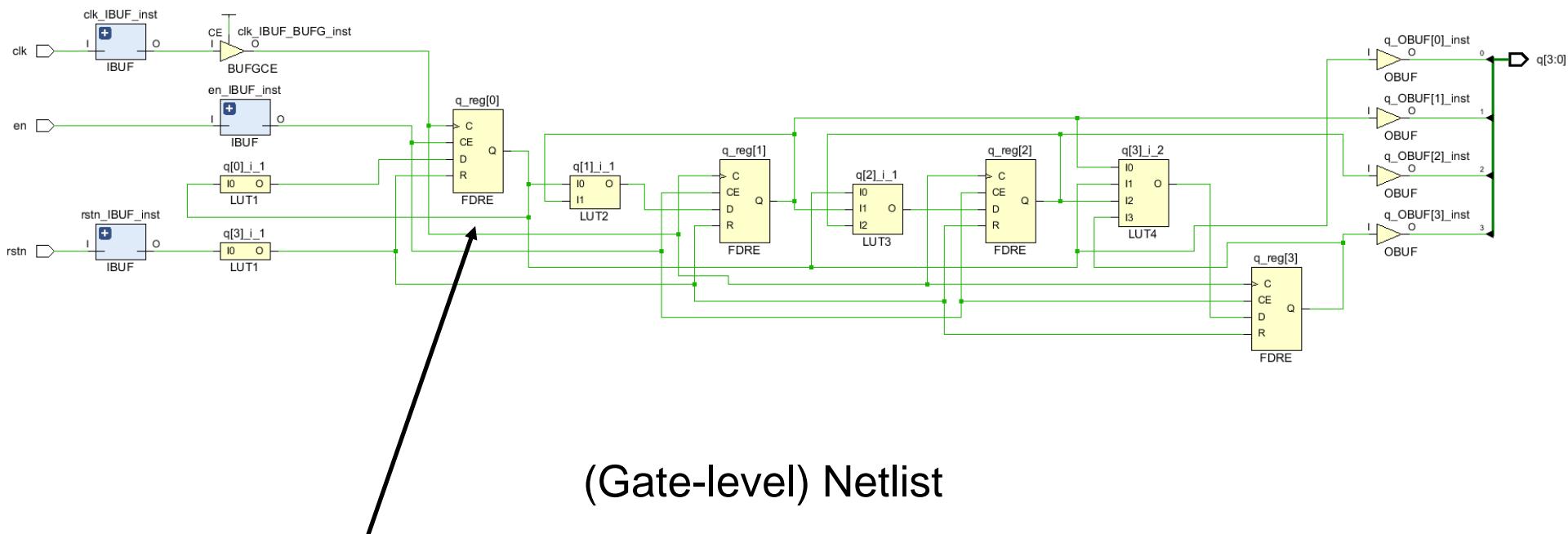
```
1 module test(
2     input wire clk,
3     input wire rstn,
4     input wire en,
5     output reg [3:0] q
6 );
7
8     always @ (posedge clk) begin
9         if (!rstn)
10            q <= 4'h0;
11        else if (en)
12            q <= q + 4'h1;
13    end
14
15 endmodule
```

(RTL Base) Elaborated Design Schematic



Design Flow

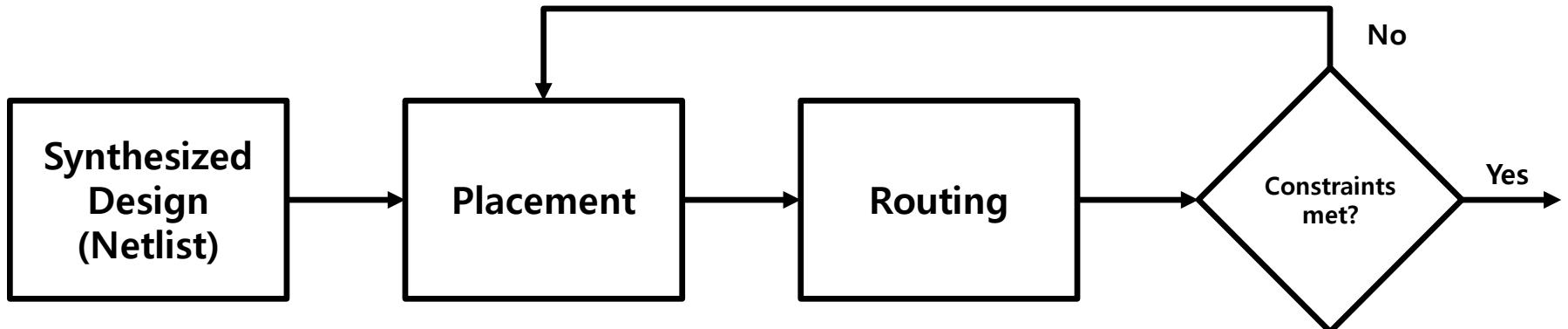
- ◆ After Synthesis, your RTL model is converted into a **netlist**, which describes the connections between different block available on the desired FPGA chip. The Synthesized model is a draft of the final design, made with the use of available resources.



FDRE : Single D-type flip-flop with data (D), clock enable (CE), and synchronous reset (R) inputs and data output (Q)

Design Flow

- ◆ Implementation(Place and Route) process do implement synthesized design into the target FPGA device. The implementation tool needs to have information about the physical composition of the device, routing paths between the different logical blocks and signal propagation timings.



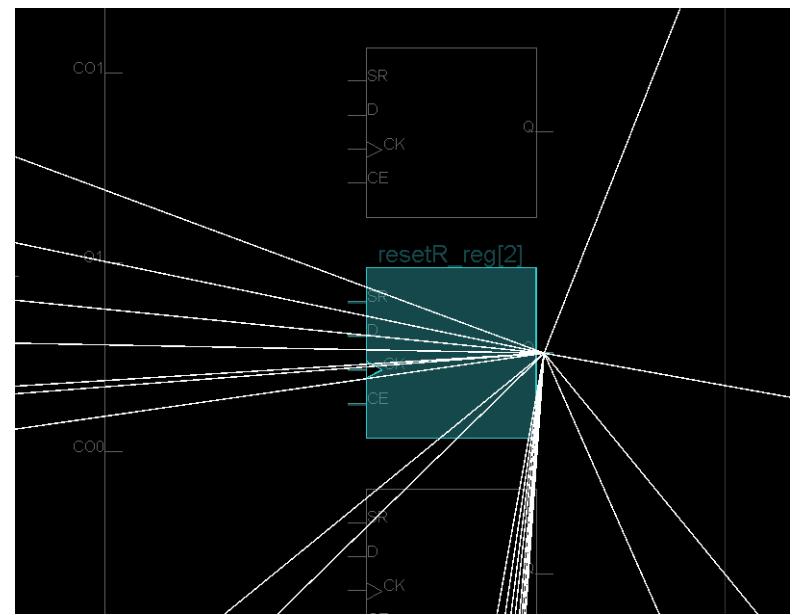
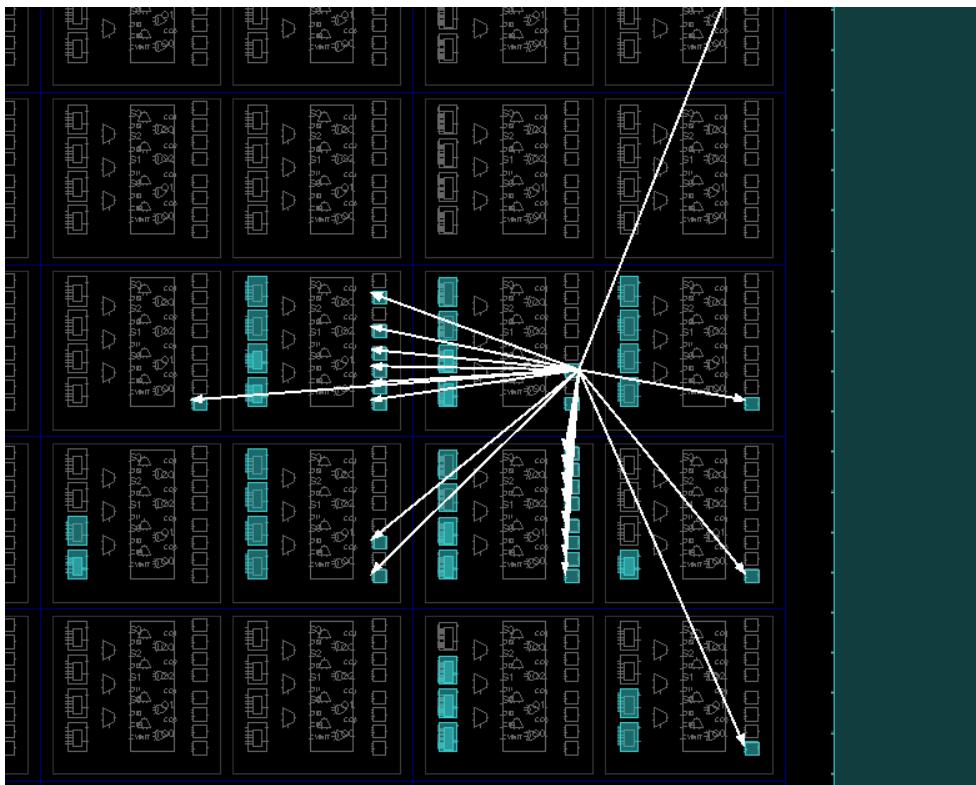
Design Flow

- ◆ Activated cells are indicated as follows



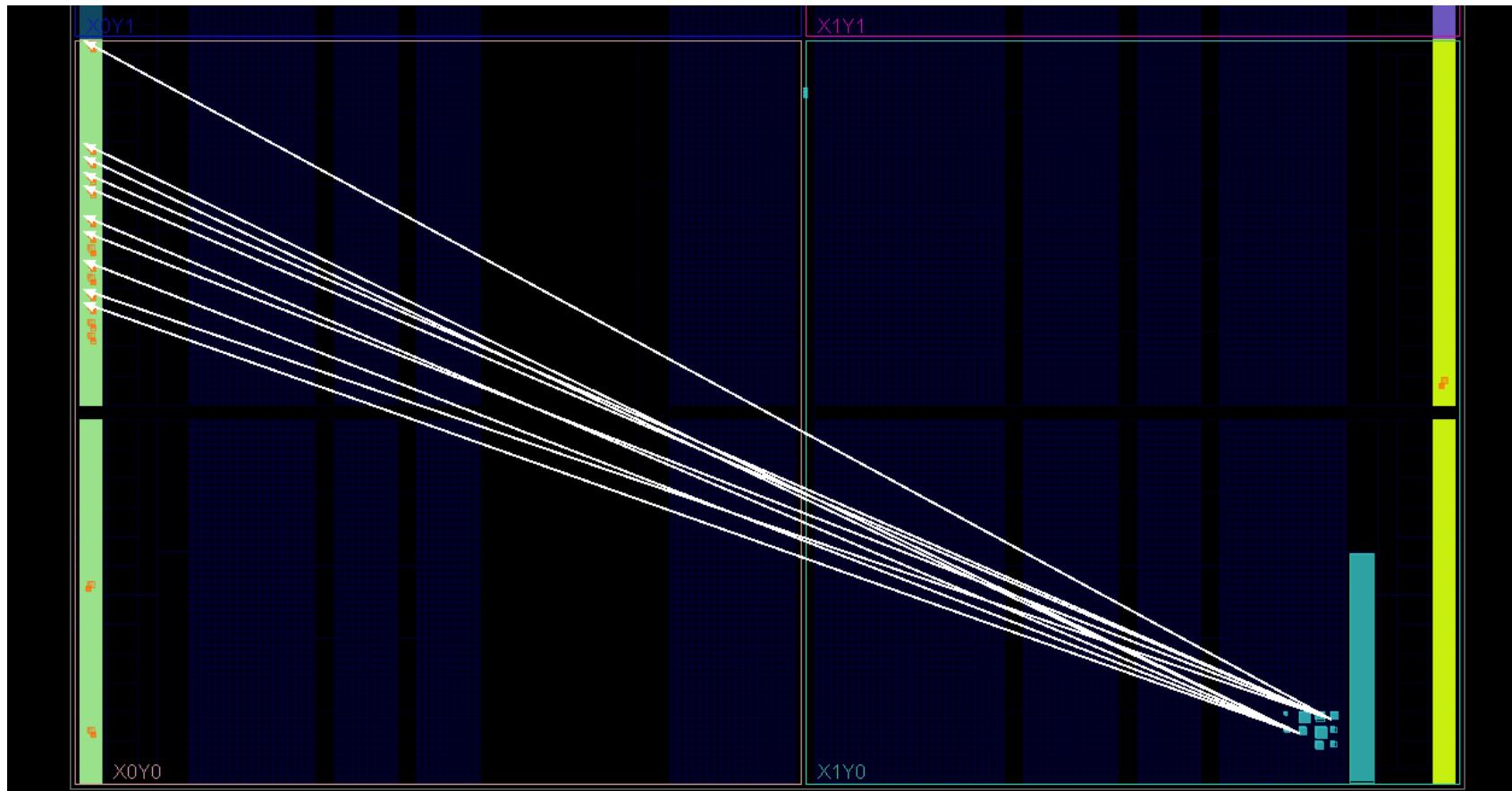
Design Flow

◆ You can see the connections between cells as follows



Design Flow

- ◆ Connections between cells and I/O ports.



Project explanation

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ Vivado – Simulation
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

Introduction

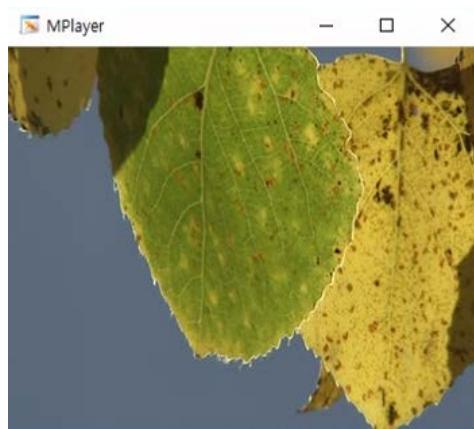
- ◆ Subject: Image down-sampling (DS) & up-sampling (US)

- ◆ Common Task (Necessary)

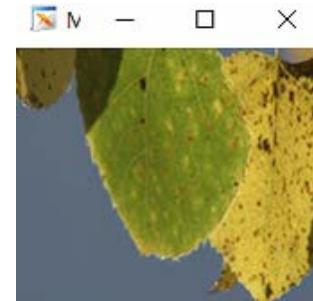
[256x256] -> [128x128] -> [256x256]

- ◆ Extra Task (Optional)

[256x256] -> [96x96] -> [256x256]



DS
→



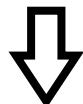
US
→



Explanation - DS & US Example

Down Sampling

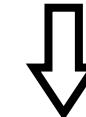
5	3	1	4
8	7	1	5
3	2	4	0
5	9	7	5



5	1
3	4

Up Sampling

5	1
3	4



5	5	1	1
5	5	1	1
3	3	4	4
3	3	4	4

- ◆ There are various implementation methods of DS, so please apply your own ideas → Depending on the method, the image quality changes

Explanation – RGB IMG



Explanation – ROM Data Structure

- ◆ 24 bits for expressing one pixel

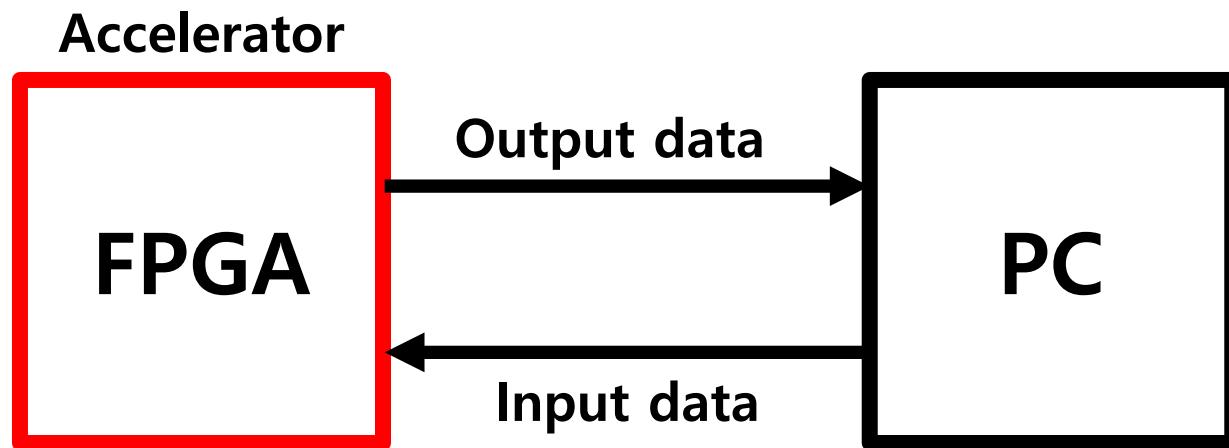


Example - Width : 24bit, Depth : 8

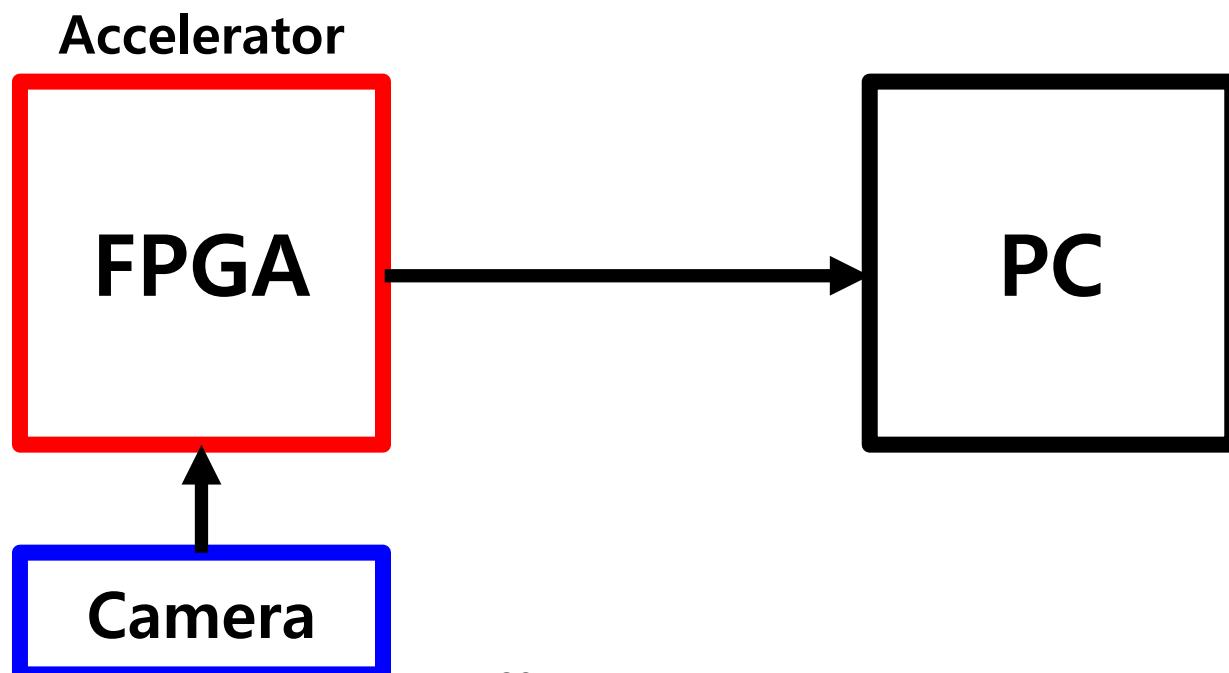
Addr	Data
0	24'b111000101000100101111110
1	24'b110111111000100110000010
2	24'b111000101000011001110111
3	24'b111000111000100001111011
4	24'b111000101000101001111100
5	24'b111000101000011001111001
6	24'b111000001000011101110011
7	24'b110111101000010101110101

Explanation – Use of FPGA as An Accelerator

Ex 1

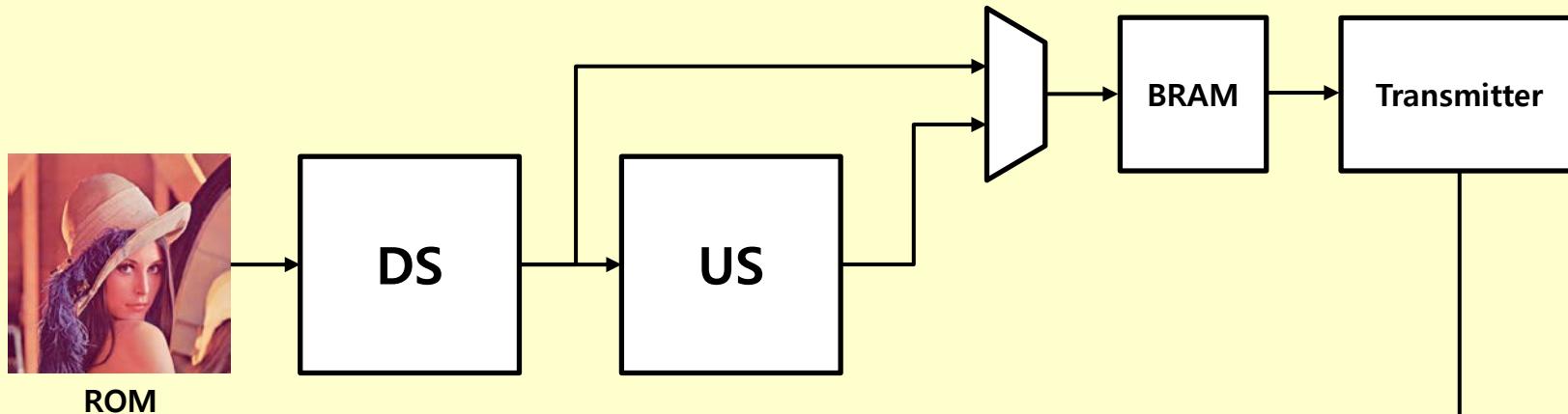


Ex 2

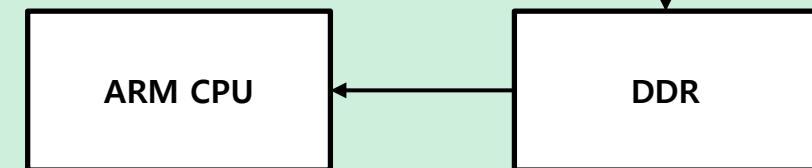


Configuration

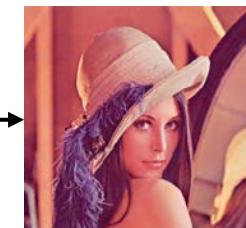
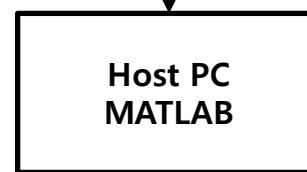
PL



PS



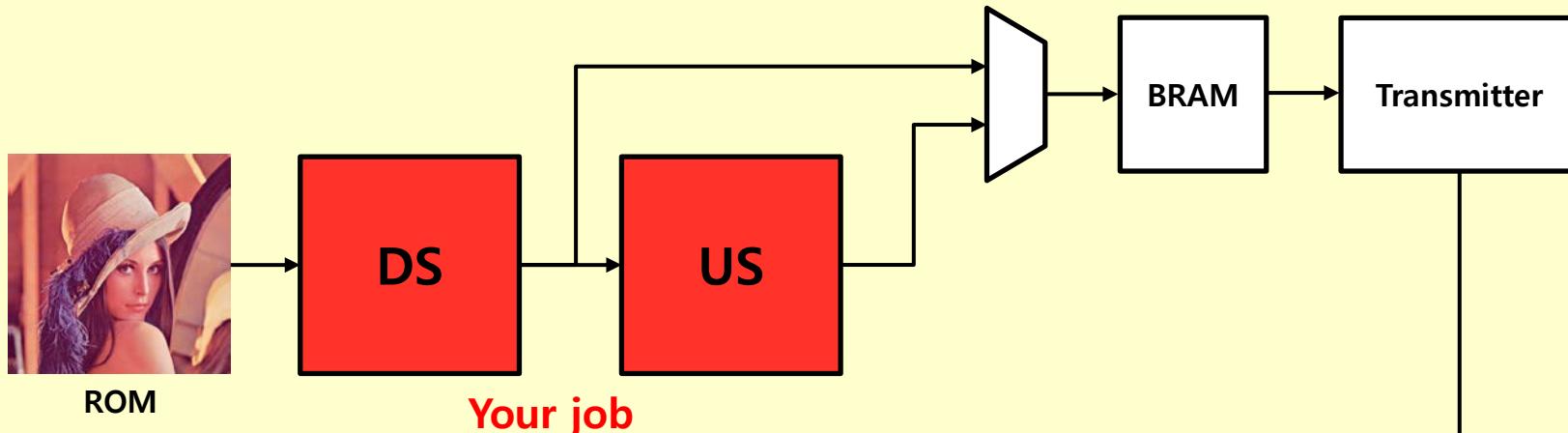
ZYNQ FPGA



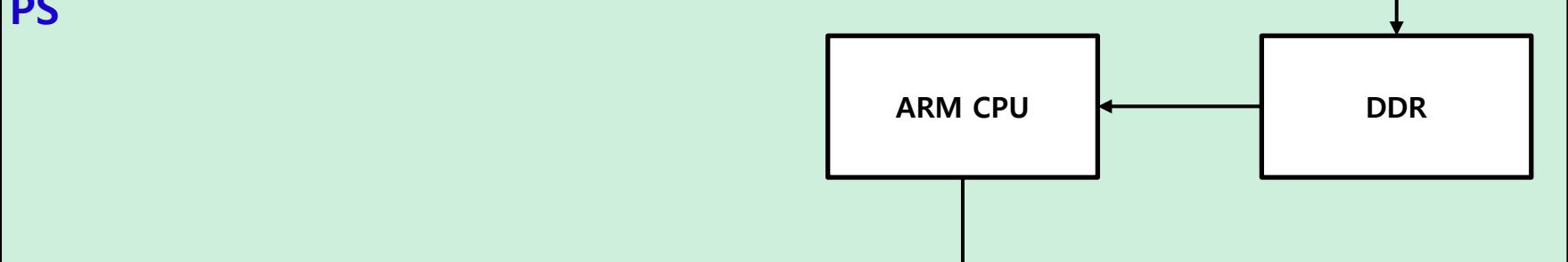
PSNR

Configuration

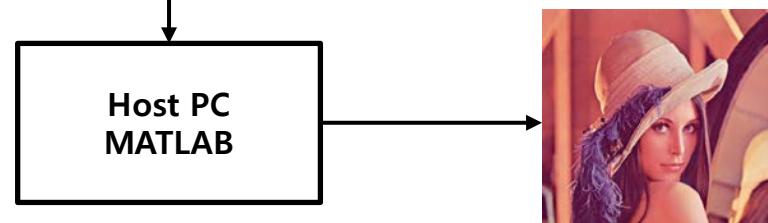
PL



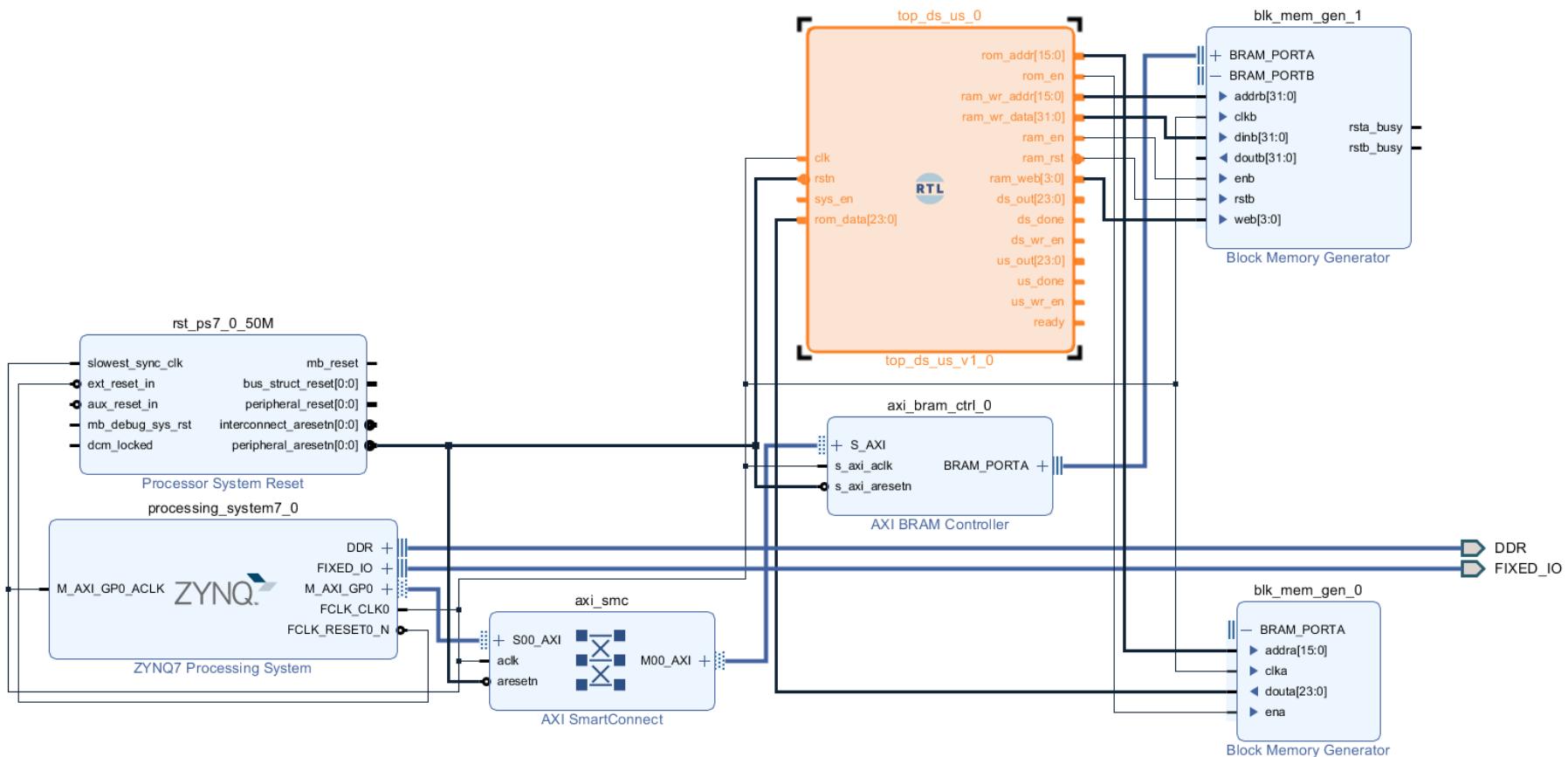
PS



ZYNQ FPGA



Configuration – Block Design



Design module hierarchy

tb_top_ds_us_test.v

top_ds_us_test.v

top_ds_us.v

ds.v

us.v



block_rom

**Dual port
block ram**

Design module hierarchy

tb_top_ds_us_test.v

top_ds_us_test.v

top_ds_us.v

ds.v

us.v

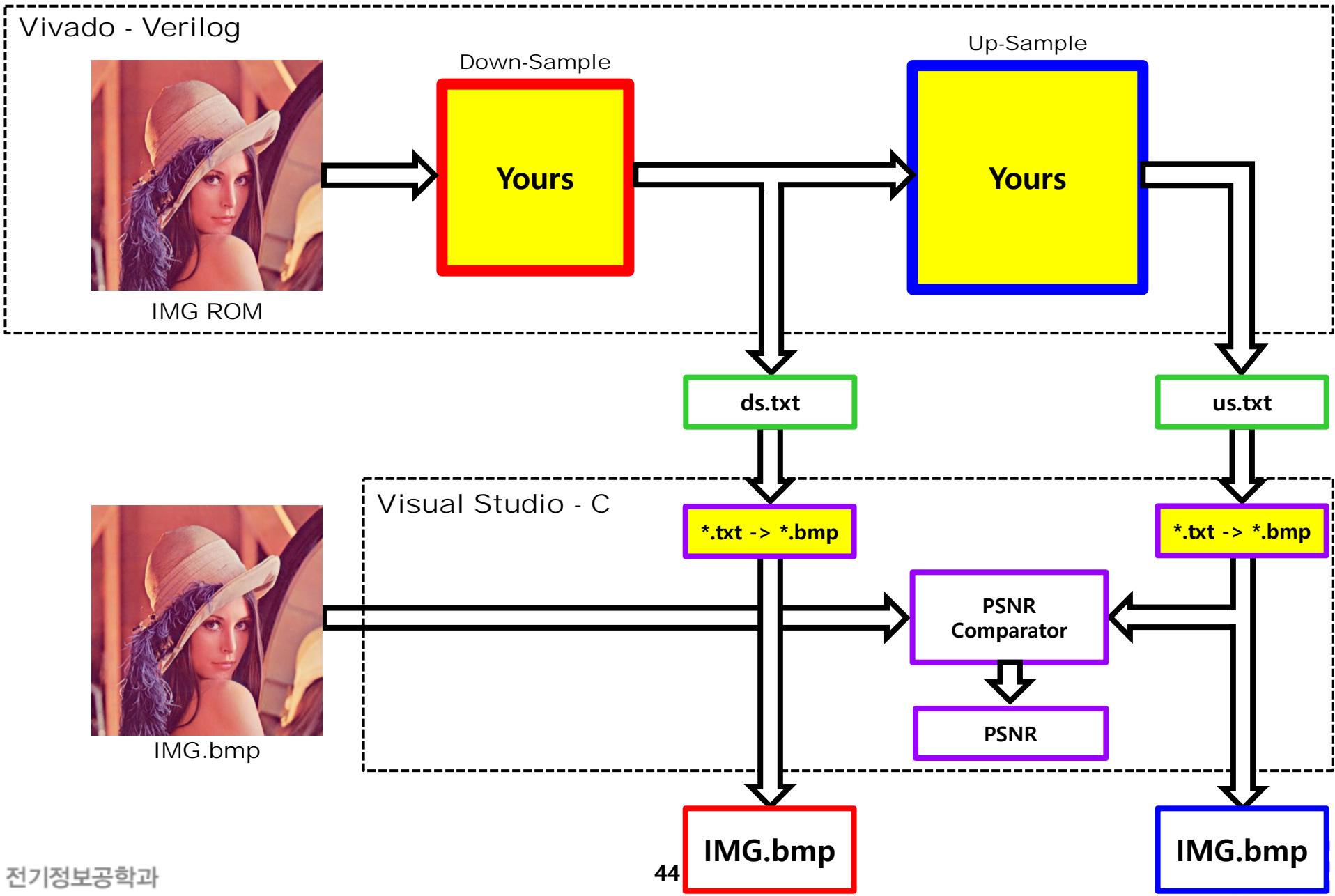
fsm.v



block_rom

**Dual port
block ram**

Simulation method example



Required Knowledge – bmp format (1)

I recommend you to try to see all the value yourself

Offset	Contents
0~53	Format Data
54~	Pixel Data

offset	size	description
0	2	signature, must be 4D42 hex
2	4	size of BMP file in bytes (unreliable)
6	2	reserved, must be zero
8	2	reserved, must be zero
10	4	offset to start of image data in bytes
14	4	size of BITMAPINFOHEADER structure, must be 40
18	4	image width in pixels
22	4	image height in pixels
26	2	number of planes in the image, must be 1
28	2	number of bits per pixel (1, 4, 8, or 24)
30	4	compression type (0=none, 1=RLE-8, 2=RLE-4)
34	4	size of image data in bytes (including padding)
38	4	horizontal resolution in pixels per meter (unreliable)
42	4	vertical resolution in pixels per meter (unreliable)
46	4	number of colors in image, or zero
50	4	number of important colors, or zero

Required Knowledge – bmp format (2)

I recommend you to try to see all the value yourself.

When you check this value, it returns 54, which means pixel data starts from 54th byte address

If the size of image is 128 by 64 (128x64), then 18th byte data contain the value of 128 and 22th byte data contain the value of 64

If you load a given image, 28th byte data would contain 24

offset	size	description
0	2	signature, must be 4D42 hex
2	4	size of BMP file in bytes (unreliable)
6	2	reserved, must be zero
8	2	reserved, must be zero
10	4	offset to start of image data in bytes
14	4	size of BITMAPINFOHEADER structure, must be 40
18	4	image width in pixels
22	4	image height in pixels
26	2	number of planes in the image, must be 1
28	2	number of bits per pixel (1, 4, 8, or 24)
30	4	compression type (0=none, 1=RLE-8, 2=RLE-4)
34	4	size of image data in bytes (including padding)
38	4	horizontal resolution in pixels per meter (unreliable)
42	4	vertical resolution in pixels per meter (unreliable)
46	4	number of colors in image, or zero
50	4	number of important colors, or zero

Required Knowledge – bmp format

Example Code

```
5     FILE *fp_read;
6     fopen_s(&fp_read, "D:\orinigal.bmp", "rb");
7
8     FILE *fp_write;
9     fopen_s(&fp_write, "D:\downsampled.bmp", "wb");
10
11    unsigned char* data = (unsigned char*)malloc(sizeof(unsigned char) * 256 * 256 * 3 + 54);
12    fread(data, sizeof(unsigned char), 256 * 256 * 3 + 54, fp1);
13
14    /*
15     * Your logic, Pixel data : data[54] ~ data[256*256*3+54-1]
16     */
17
18    fclose(fp_read);
19    fclose(fp_write);
20    free(data);
```

※ This is just an example code and therefore, you don't have to follow this style!

Required Knowledge – System Task

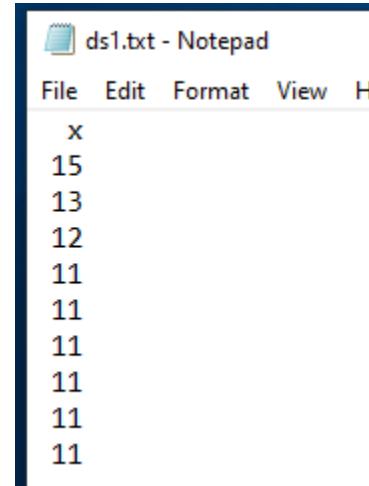
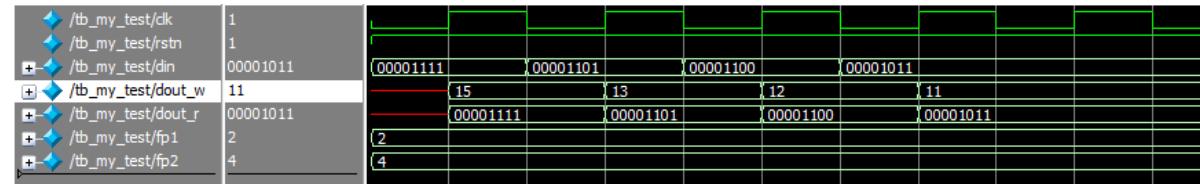
File Write in Verilog

```
// file pointer
integer fp;

// file open
initial begin
    fp = $fopen("ds1.txt");
end

// data write
always @ ( posedge clk ) begin
    $fdisplay(fp, "%d", dout_w);
end

// file close
initial begin
    #1000    $finish ;
    $fclose(fp);
end
```



This txt file is stored in your project location
You can add some conditions to write proper data

Required Knowledge – PSNR (1)

- ◆ Peak Signal to Noise Ratio (PSNR) is the standard way to measure fidelity

$$PSNR = 10 \log_{10} \left(\frac{m^2}{MSE} \right)$$

**where m is the maximum value of a pixel possible.
For gray scale images (8 bits per pixel) m = 255**

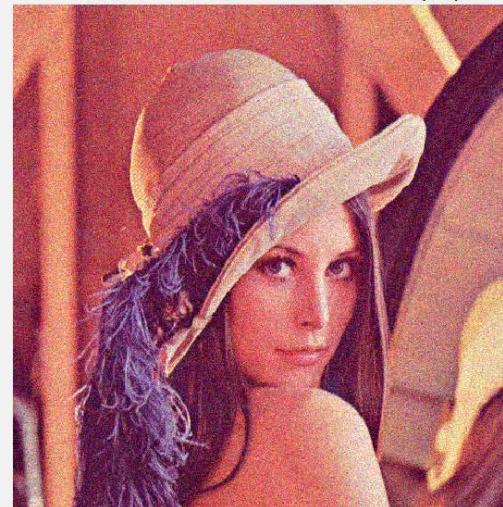
- ◆ PSNR is measured in decibel (dB)
- ◆ A high PSNR means less image quality damage (low MSE)

Required Knowledge – PSNR (2)

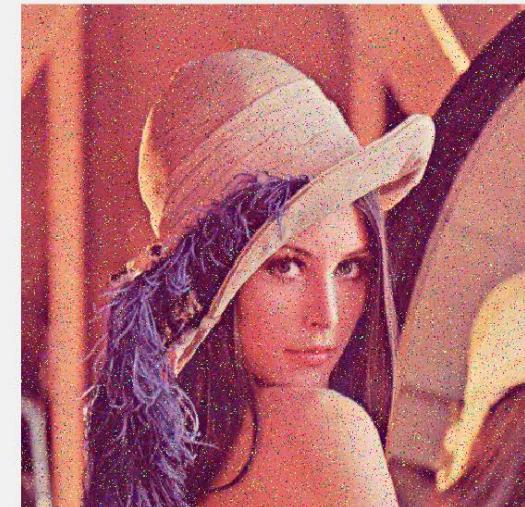
Original Image, PSNR = Inf



Gaussian Noise, PSNR = 20.213209(dB)



Salt&Pepper Noise, PSNR = 18.182128(dB)



- As the difference between two images as greater, PSNR gets low value!
 - .5 to 1 dB is said to be a perceptible difference
 - In general, PSNRs of at least 30 dB are required to be considered acceptable
-
- * We will provide a C code for calculating PSNRs

Required Knowledge – Cycle(Throughput)

- ◆ In HW design, cycle is one of the important factors. If there are enough resource, it's better to exploit resource as many as possible in order to increase throughput (which means shorter cycle, less number of clocks per task).
- ◆ I'll give you a skeleton code "top.v, tb_top.v". There is an example circuit how to measure your total clock.

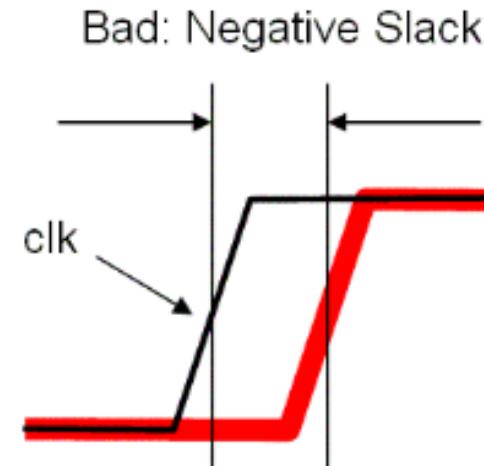
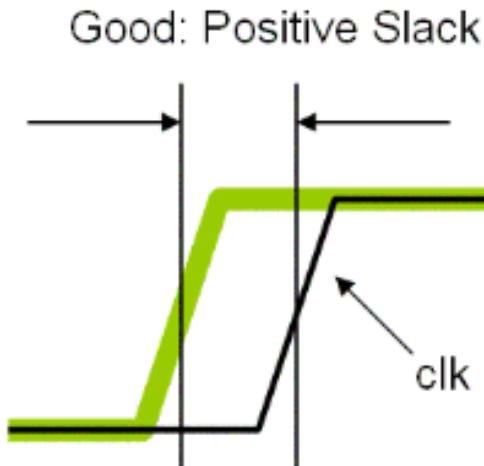
```
always @ (posedge clk or negedge rstn) begin
    if ( !rstn )
        total_clock <= 0;
    else begin
        if ( en_system )
            if ( tlast ) begin
                $display("Total Clock = %d", total_clock);
                total_clock <= 0;
            end
            else
                total_clock <= total_clock + 1;
        else
            total_clock <= 0;
    end
end
```

Required Knowledge – Timing

- ◆ After synthesis, you can see “estimated timing information”, which includes a few things about “slack”.
- ◆ We don’t have enough time to explain detail about slack, but simply speaking, if there is any negative slack in your logic, then your design would fail.

Positive slack : There is some spare time, so it's good.

Negative slack : There need some time to meet design constraints, so it's bad.



source from <<http://babyworm.net/archives/170>>

Required Knowledge – Timing

- ◆ Check your <Design Timing Summary> report, after synthesis.
Following values should be greater than or equal to 0.
- ◆ This example tell us there might be some paths which can cause problems.

Design Timing Summary

should be positive

Setup

Worst Negative Slack (WNS): 7.189 ns

Total Negative Slack (TNS): 0.000 ns

Number of Failing Endpoints: 0

Total Number of Endpoints: 20437

Worst Hold Slack (WHS): -0.107 ns

Total Hold Slack (THS): -483.437 ns

Number of Failing Endpoints: 9887

Total Number of Endpoints: 20437

Hold

Required Knowledge – Timing

◆ Post Synthesis

Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 7.189 ns	Worst Hold Slack (WHS): -0.107 ns	Worst Pulse Width Slack (WPWS): 3.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): -483.437 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 9887	Number of Failing Endpoints: 0
Total Number of Endpoints: 20437	Total Number of Endpoints: 20437	Total Number of Endpoints: 6604

◆ Post Implementation

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 4.094 ns	Worst Hold Slack (WHS): 0.010 ns	Worst Pulse Width Slack (WPWS): 3.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 17209	Total Number of Endpoints: 17209	Total Number of Endpoints: 5708

- ◆ However, post synthesis report is not exact. It is just an estimated value. In order to obtain almost exact value, you have to perform “Implementation” and see post implementation timing report.
- ◆ In this case, even though there were some problems according to post-synthesis report, it turned out that your design would work well.

Required Knowledge – Timing

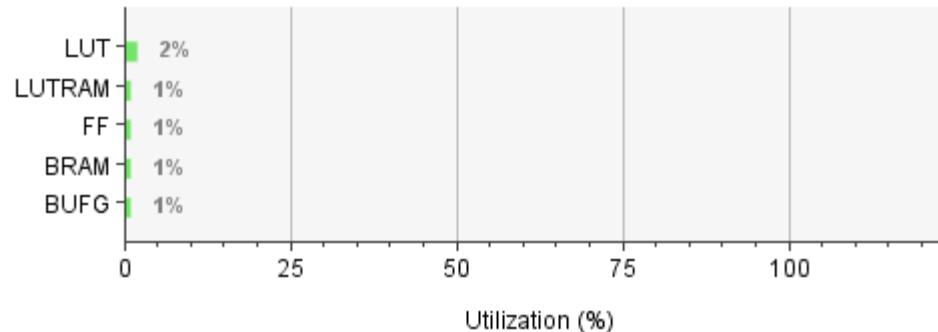
- ◆ Since our design is quite simple and there is no constraints, your timing result would be highly likely to be seen as follows.

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): inf	Worst Hold Slack (WHS): inf	Worst Pulse Width Slack (WPWS): NA
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): NA
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: NA
Total Number of Endpoints: 723	Total Number of Endpoints: 723	Total Number of Endpoints: NA

Required Knowledge – HW Utilization

Summary

Resource	Utilization	Available	Utilization %
LUT	4375	274080	1.60
LUTRAM	1045	144000	0.73
FF	6501	548160	1.19
BRAM	2	912	0.22
BUFG	1	404	0.25



After synthesis of the designed logic, you can see how many resources are used in your design by checking utilization report

I'll address detail of it another slide.

Practice 4: Down-sample

◆ Simple Down-sampler

- Function: select one pixel data and discard the rest pixel data.
- Main circuits
 - Address generator
 - Data flow controller
- Main signals
 - system - enable
 - memory(read) - enable, address, data
 - effective data – valid, output data

Data[0]
Data[1]
Data[2]
Data[3]
...
Data[13]
Data[14]
Data[15]

Data[0]	Data[1]	Data[2]	Data[3]
Data[4]	Data[5]	Data[6]	Data[7]
Data[8]	Data[9]	Data[10]	Data[11]
Data[12]	Data[13]	Data[14]	Data[15]



Sampled Data[0]	Sampled Data[1]
Sampled Data[4]	Sampled Data[5]

Practice 4: Up-sample

◆ Up-sample

- Function: Copy and generate same pixel data
- Main circuits
 - Address generator
 - Reproduction logic
 - Data flow controller
- Main signals
 - system – enable
 - memory(read) – enable, address, data
 - memory(write) – write enable, enable, address, data

Data[0]	Data[1]
Data[2]	Data[3]



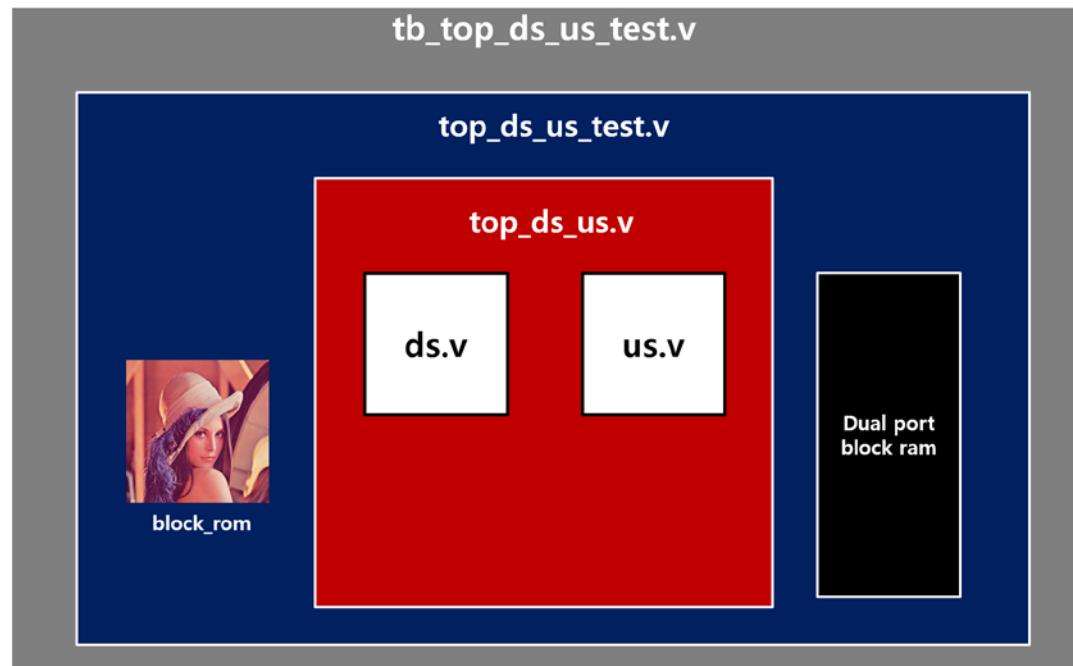
New Data[0]	New Data[1]	New Data[2]	New Data[3]
New Data[4]	New Data[5]	New Data[6]	New Data[7]
New Data[8]	New Data[9]	New Data[10]	New Data[11]
New Data[12]	New Data[13]	New Data[14]	New Data[15]

New Data[0]
New Data[1]
New Data[2]
New Data[3]
...
New Data[13]
New Data[14]
New Data[15]

Practice 4: (Down-up sample)

◆ Cascaded Down-Up Sample

- Function: Generate restored pixel data for the same period.
- Method: Wire down-sampler and up-sampler module
- Main circuits:
 - DS & US
 - Data flow controller
- Main signals:
 - system – enable
 - memory access
 - handshake signal



Practice 4 – Design flow

Down Sampling

Reference Code (C, Python, etc.)



RTL Implementation and Behavioral Simulation (It requires the same result with Ref SW Code)

Post Synthesis Functional Simulation (It requires the same result with Behavioral Simulation)

Synthesis Report

Up Sampling

Reference Code (C, Python, etc.)



RTL Implementation and Behavioral Simulation (It requires the same result with Ref SW Code)

Post Synthesis Functional Simulation (It requires the same result with Behavioral Simulation)



Synthesis Report

◆ Outputs to be submitted

- DS/US reference SW codes
- DS/US RTL codes
- Practice 4 report should include following results (PSNR, cycle, HW utilization)

Practice 4 – Files to submit

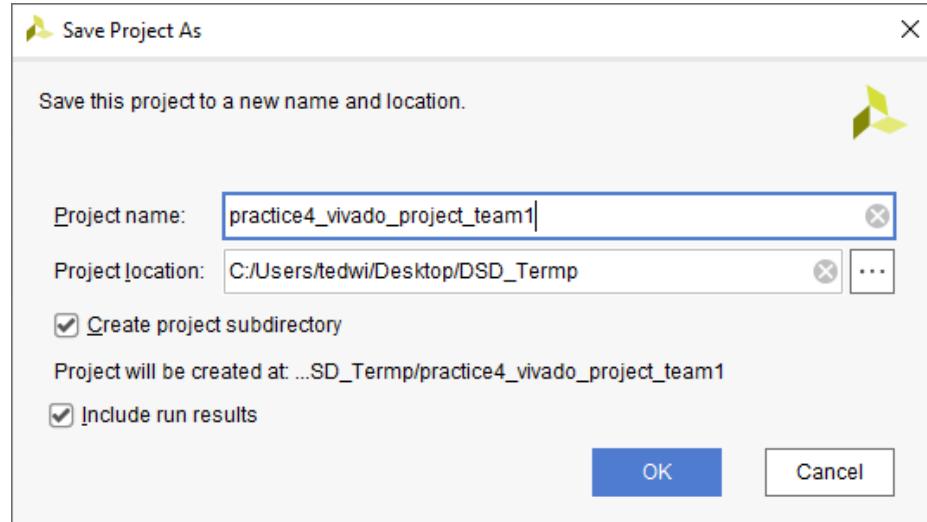
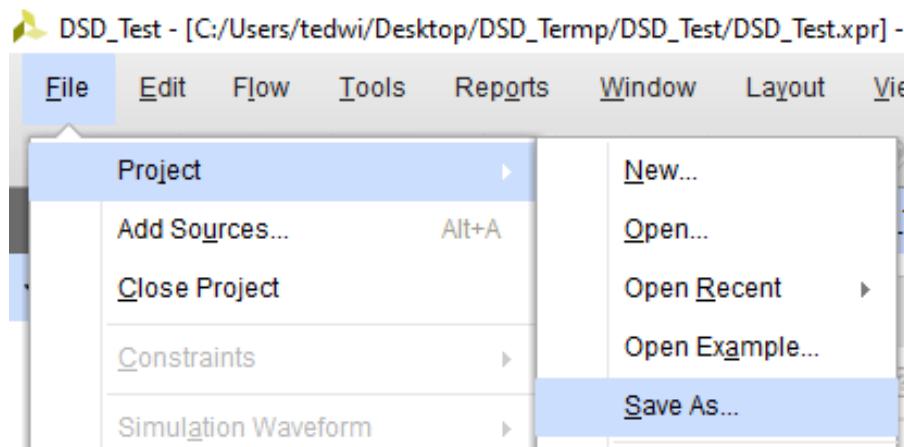
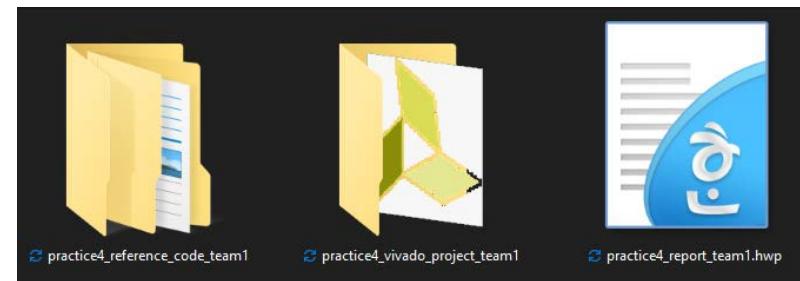
- ◆ Zip file
 - file name: "practice4_team#.zip"



- ◆ Practice 4 - Report
 - file name: "practice4_report_tema#.hwp"

- ◆ Reference code
 - file name: "practice4_reference_code_team#"

- ◆ Vivado Project File
 - file name: "practice4_vivado_project_team#"



Term-project Proposal

◆ You have to submit proposal of term-project.

- 1. Abstract

- (Summarize your project goal, features and performance.)

- 2. Objective

- (Organize your objective and background)

- 3. Preliminary Research

- (*Organize related knowledge or theories you searched to achieve your goal.*)

- 4. Design Contents

- (*Summarize features and functions : DS & US algorithm and issues with regard to RTL, synthesis and e.t.c.)*

Term-project Diary

- ◆ Summarize and submit weekly term-project progress.

Digital System Design – Project Diary

Date : 2021-xx-xx

Student ID		Name	
Describe what you did in this week, design issues, designed code, and future plans, etc.			

Evaluation Criterion

C		RTL		Diary	PSNR
DS	US	DS	US	3.3 * 3 (=10)	15
5	5	10	10		
Cycle	Timing	Utilization	Proposal	Demo & Presentation	Report
5	5	5	5	5	10

- ◆ C and Verilog codes are evaluated by the successful operation and the quality of the codes
- ◆ We will evaluate your results (especially, PSNRs) using both given images and new test image
- ◆ Relative evaluation of PSNR is applied only when C code is successful
- ◆ Relative evaluation of cycle is applied only when functional verification is successful
- ◆ Relative evaluation of timing and utilization is applied only when synthesis is successful
- ◆ If you succeed in an optional mission, we will give you additional 10 points

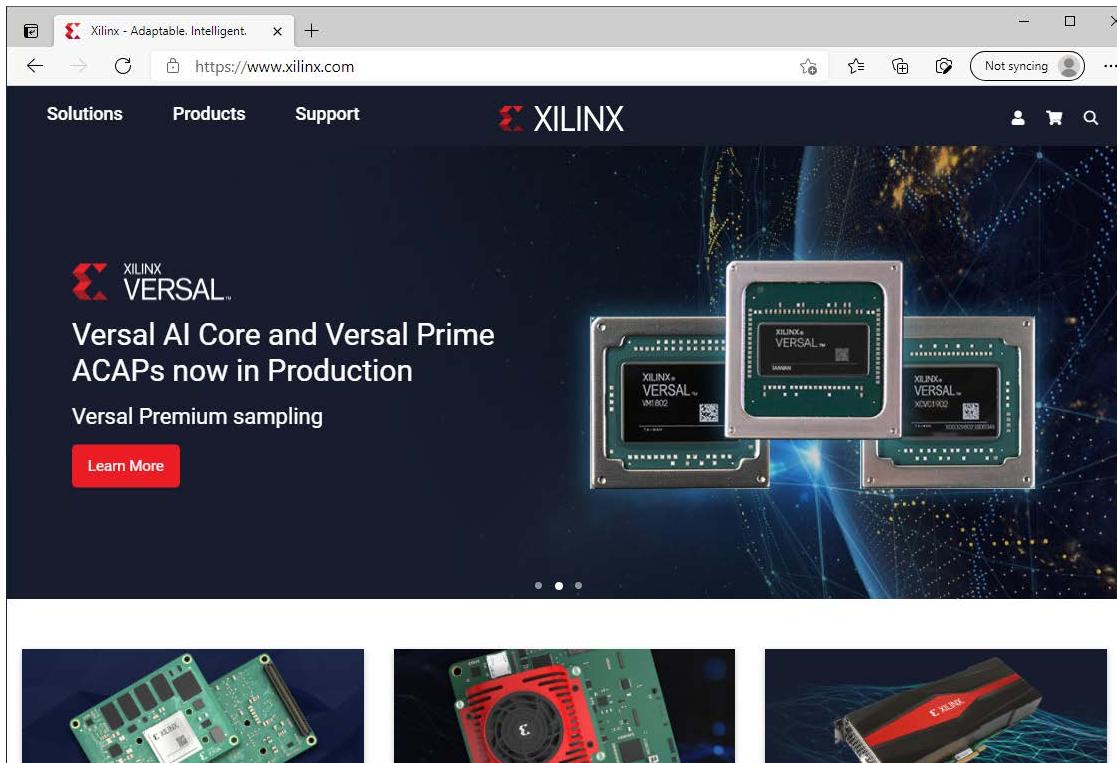
Schedule

- ◆ Diary #1
 - 5/20(Thu.) ~ 5/26(Wed.)
- ◆ Diary #2, Practice4, Proposal
 - 5/27(Thu.) ~ 6/2(Wed.)
- ◆ Final Exam
 - 6/3(Thu.)
- ◆ Diary #3
 - 6/3(Thu.) ~ 6/9(Wed.)
- ◆ Term-project report & files
 - 6/10(Thu.) ~ 6/16(Wed.)
- ◆ Term-project demo & presentation
 - 6/17(Thu.)

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ Vivado – Simulation
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

Download Vivado

- ◆ In order to download Vivado, you have to create a Xilinx account.
- ◆ <https://www.xilinx.com/>



Download Vivado

The screenshot shows the Xilinx website's main navigation bar. The 'Products' tab is highlighted with a red box. Below it, a sidebar lists various product categories: Devices, Accelerators, System-on-Modules (SOMs), Evaluation Boards & Kits, Ethernet Adapters, Software Development, Hardware Development, Embedded Development, Core Technologies, and App Store. The 'Hardware Development' category is also highlighted with a red box.

The screenshot shows the 'Hardware Development Tools' section of the Xilinx website. It features a heading 'Hardware Development Tools' and a list of tools. The 'Vivado® Design Suite' tool is highlighted with a red box. Other listed tools include Intellectual Property, System Generator, Add-On for MATLAB & Simulink, and E WEBINAR.

Download Vivado

The screenshot shows a web browser window displaying the Xilinx Vivado Design Suite HLx Editions page. The page has a dark background with yellow streaks. On the left, it says "Vivado Design Suite HLx Editions". In the center, there's a large "VIVADO" logo with a yellow triangle icon. Below the logo, the text "productivity multiplied" is visible. At the bottom, there are several navigation links: "Overview" (underlined), "Documentation", "Features / Buy", "Getting Started", and "Video". Two buttons are present: "What's New in Vivado" and "Download Vivado", with the "Download Vivado" button being highlighted with a red rectangle. A "Feedback" button is located on the right side. The browser address bar shows the URL: <https://www.xilinx.com/products/design-tools/vivado.html#overview>.

Download Vivado

- ◆ In this lab, we are going to use "Vivado HL WebPACK Edition".

Vivado Design Suite - HLx Editions

Vivado Design Suite - HLx Edition Features	Vivado HL Design Edition	Vivado HL System Edition	Vivado Lab Edition	Vivado HL WebPACK Edition (Device Limited)	Free 30-day Evaluation
Accelerating Implementation					
Synthesis and Place and Route	•	•		•	•
Dynamic Function eXchange	•	•		•	•
Accelerating Verification					
Vivado Simulator	•	•		•	•
Vivado Device Programmer	•	•	•	•	•
Vivado Logic Analyzer	•	•	•	•	•
Vivado Serial I/O Analyzer	•	•	•	•	•
Debug IP (ILA/VIO/IBERT)	•	•		•	•
Accelerating High Level Design					
Vivado High-Level Synthesis	•	•		•	•
Vivado IP Integrator	•	•		•	•
System Generator for DSP	*	•		*	•
Add-on for Matlab and Simulink	*	•		*	•

Download Vivado

The screenshot shows a web browser window displaying the Xilinx Downloads page. The URL in the address bar is <https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vi>. The page has a dark header with 'Solutions', 'Products', and 'Support' links, and the Xilinx logo. Below the header, there's a breadcrumb trail: Home / Support / Downloads. The main title is 'Downloads'. There are two dark buttons: 'Licensing Help' and 'Alveo Accelerator Card'. Below these are five links: 'Vivado (HW Developer)' (underlined), 'Vitis (SW Developer)', 'Vitis Embedded Platforms', 'Alveo Packages', and 'PetaLinux'. The 'Vivado (HW Developer)' link is underlined. The page is divided into sections by year. The '2020' section contains '2020.3', '2020.2', '2020.1', and '2019.2'. The '2019' section contains '2019'. The '2018' section contains '2019.1', '2018.3', '2018.2', and '2018.1'. A red box highlights the 'Vivado Archive' link under the 2019.2 section. Other visible links in the sidebar include 'ISE Archive' and 'CAE Vendor Libraries Archive'.

Version	Notes
We strongly recommend using the latest releases available.	
2020.3	2019
2020.2	
2020.1	
2019.2	
Vivado Archive	
ISE Archive	
CAE Vendor Libraries Archive	
2019.1	
2018.3	
2018.2	
2018.1	

Download Vivado

Vivado Design Suite - HLx Editions - 2019.1 Full Product Installation

Important

We strongly recommend to use the web installers as it reduces download time and saves significant disk space.

Please see [Installer Information](#) for details.

Note: Download verification is only supported with Google Chrome and Microsoft Edge web browsers.

 [Vivado HLx 2019.1: WebPACK and Editions - Windows Self Extracting Web Installer \(EXE - 64.62 MB\)](#)

MD5 SUM Value : 743003070fb77857ad098bd6873bdf0b

 [Vivado HLx 2019.1: WebPACK and Editions - Linux Self Extracting Web Installer \(BIN - 115.05 MB\)](#)

MD5 SUM Value : 533000dc5324be422915eb4e93f9ce59

Download Verification

Digests

Signature

Public Key

 [Vivado HLx 2019.1: All OS installer Single-File Download \(TAR/GZIP - 21.39 GB\)](#)

MD5 SUM Value : 47388a71dc5962a4b8d76e752928616e

Download Verification

Digests

Signature

Public Key

Download Includes

Vivado Design Suite
HLx Editions (All
Editions)

Download Type

Full Product
Installation

Last Updated

May 29, 2019

Answers

2019.x - Vivado
Known Issues

Documentation

Release Notes
What's New in Vivado

Support Forums

Installation and
Licensing

/app/xilinxinc_f5awsprod_1/exknv8ms950lm0Ldh0x7/sso/saml



Sign-In

E-mail Address

juntaepark@seoultech.ac.kr

Password

Sign In

Create Account

[Forgot / Reset Password?](#)

[Resend account activation e-mail?](#)

We're consistently improving our site security.
Account login now uses e-mail addresses.

[Learn More](#)

Download Vivado

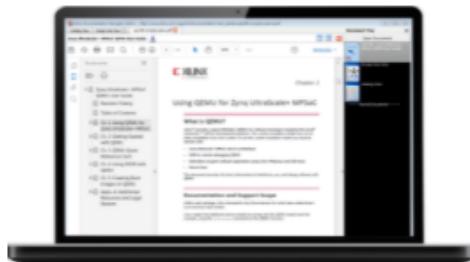
Job Function*

Student

For more information about how we process your personal information, please see our [privacy policy](#).

Download

New to Vivado HLx?



Visit the [Vivado HLx Getting Started page](#) to launch your design! These resources include:

- Installation and Licensing information
- Best Practices, Getting Started and Methodology Guides
- On Demand Videos
- Development Boards and Kits
- Support Resources

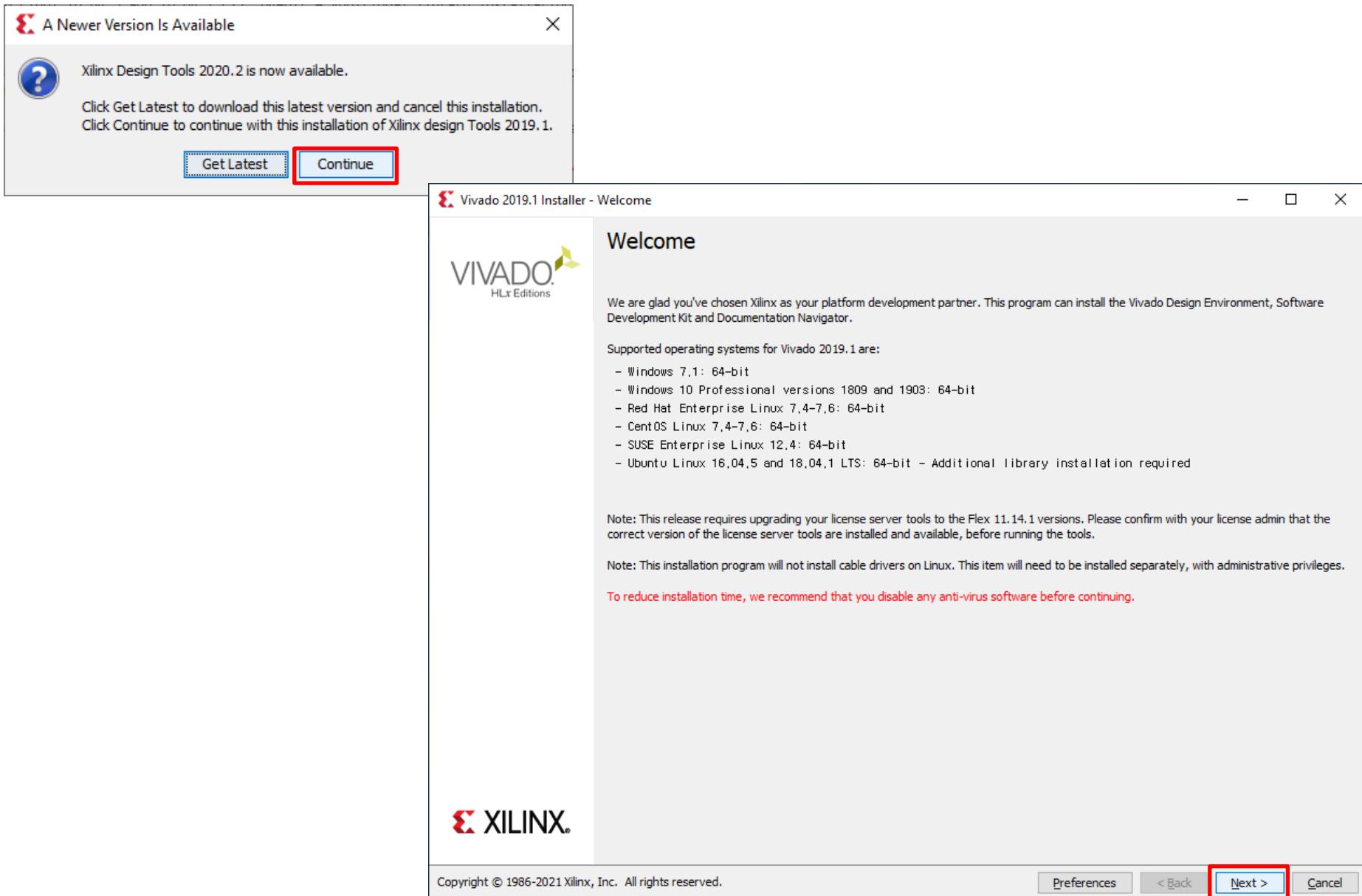
Thank you for download Vivado HLx!

Download Vivado

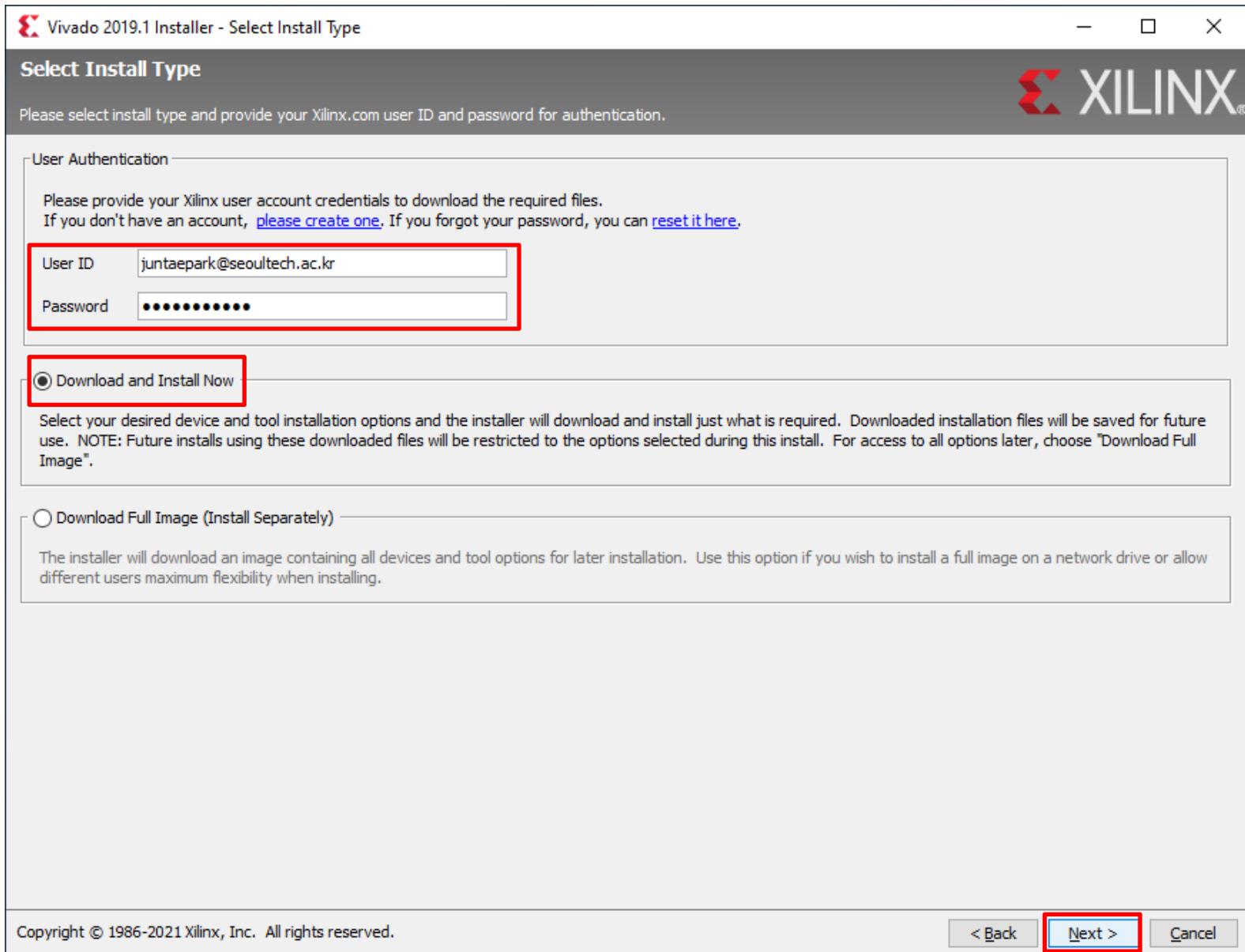
- ◆ Execute *.exe file to install Vivado



Download Vivado



Download Vivado



Download Vivado

Vivado 2019.1 Installer - Accept License Agreements

Accept License Agreements

Please read the following terms and conditions and indicate that you agree by checking the I Agree checkboxes.

Xilinx Inc. End User License Agreement

By checking "I Agree" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, I AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

I Agree

WebTalk Terms And Conditions

By checking "I Agree" below, I also confirm that I have read [Section 13 of the terms and conditions](#) above concerning WebTalk and have been afforded the opportunity to read the WebTalk FAQ posted at <https://www.xilinx.com/products/design-tools/webtalk.html>. I understand that I am able to disable WebTalk later if certain criteria described in Section 13(c) apply. If they don't apply, I can disable WebTalk by uninstalling the Software or using the Software on a machine not connected to the internet. If I fail to satisfy the applicable criteria or if I fail to take the applicable steps to prevent such transmission of information, I agree to allow Xilinx to collect the information described in Section 13(a) for the purposes described in Section 13(b).

I Agree

Third Party Software End User License Agreement

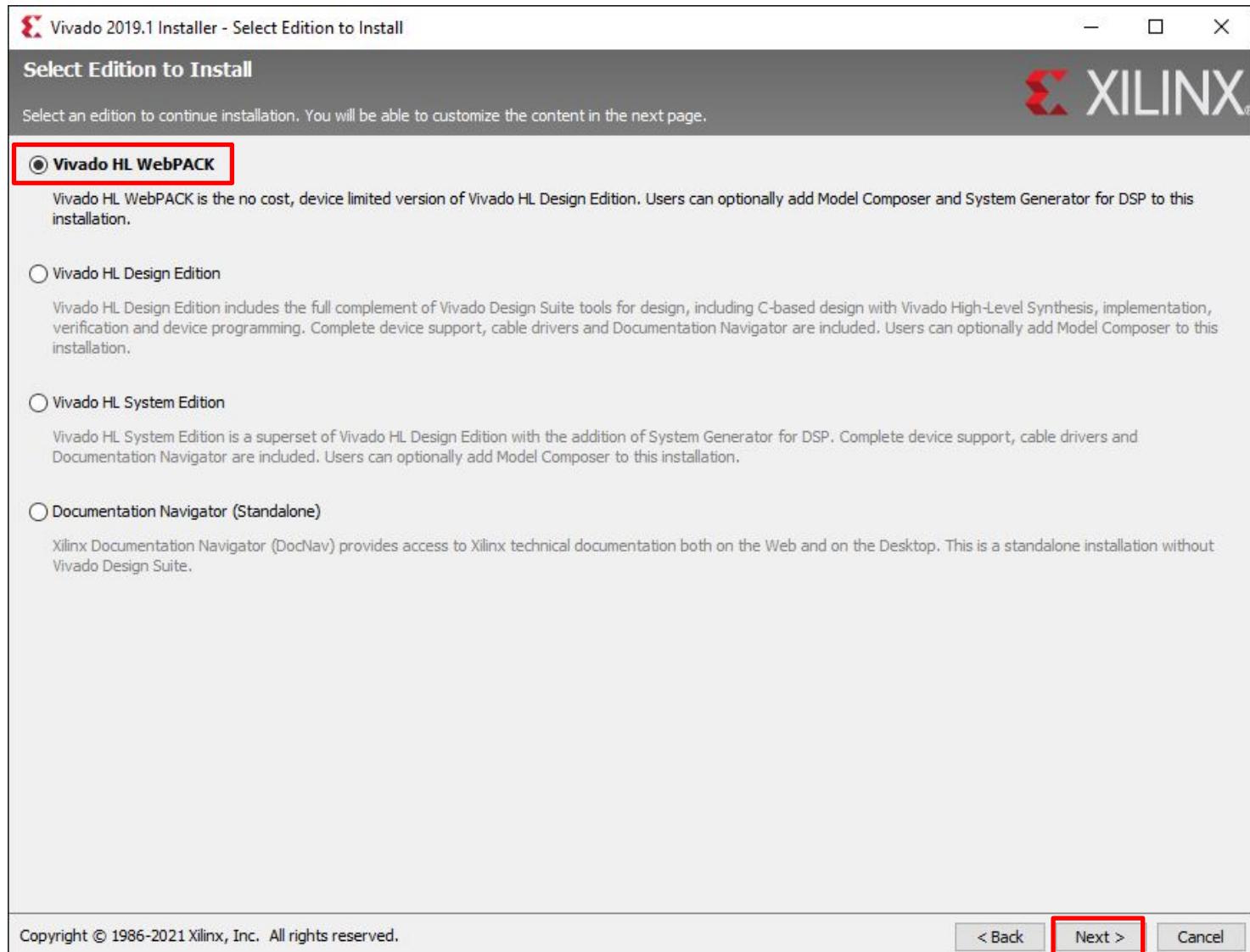
By checking "I Agree" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, I AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

I Agree

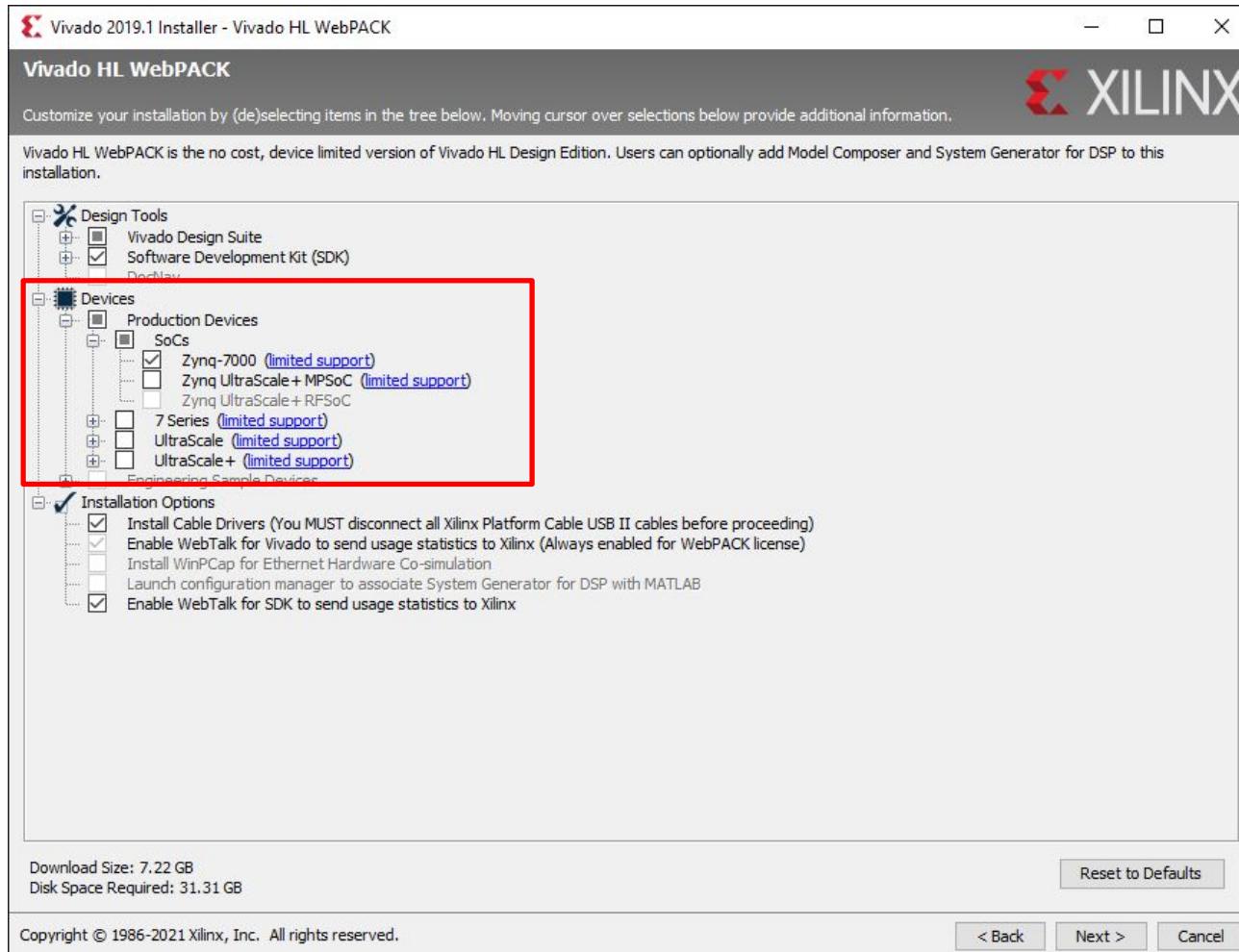
Copyright © 1986-2021 Xilinx, Inc. All rights reserved.

< Back Next > Cancel

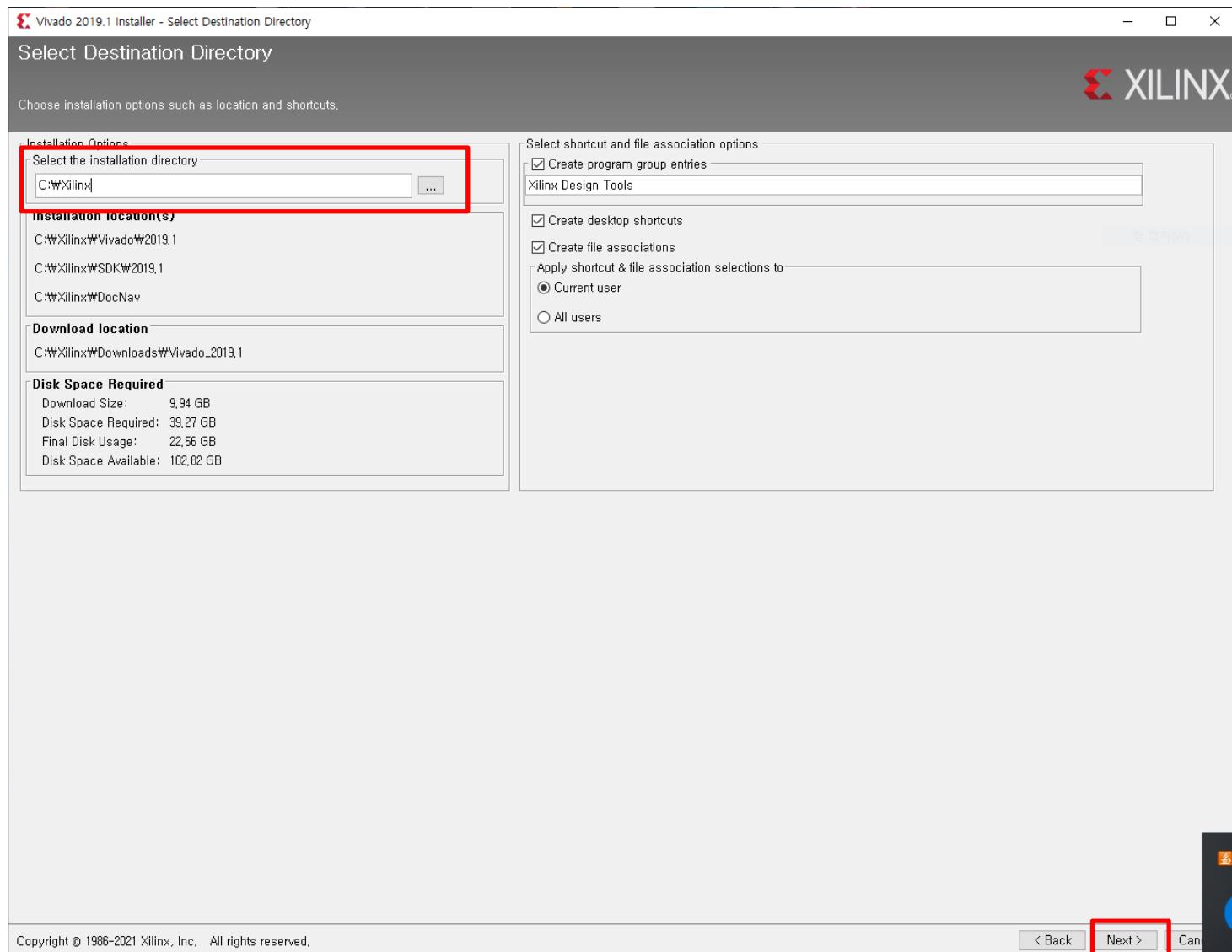
Download Vivado



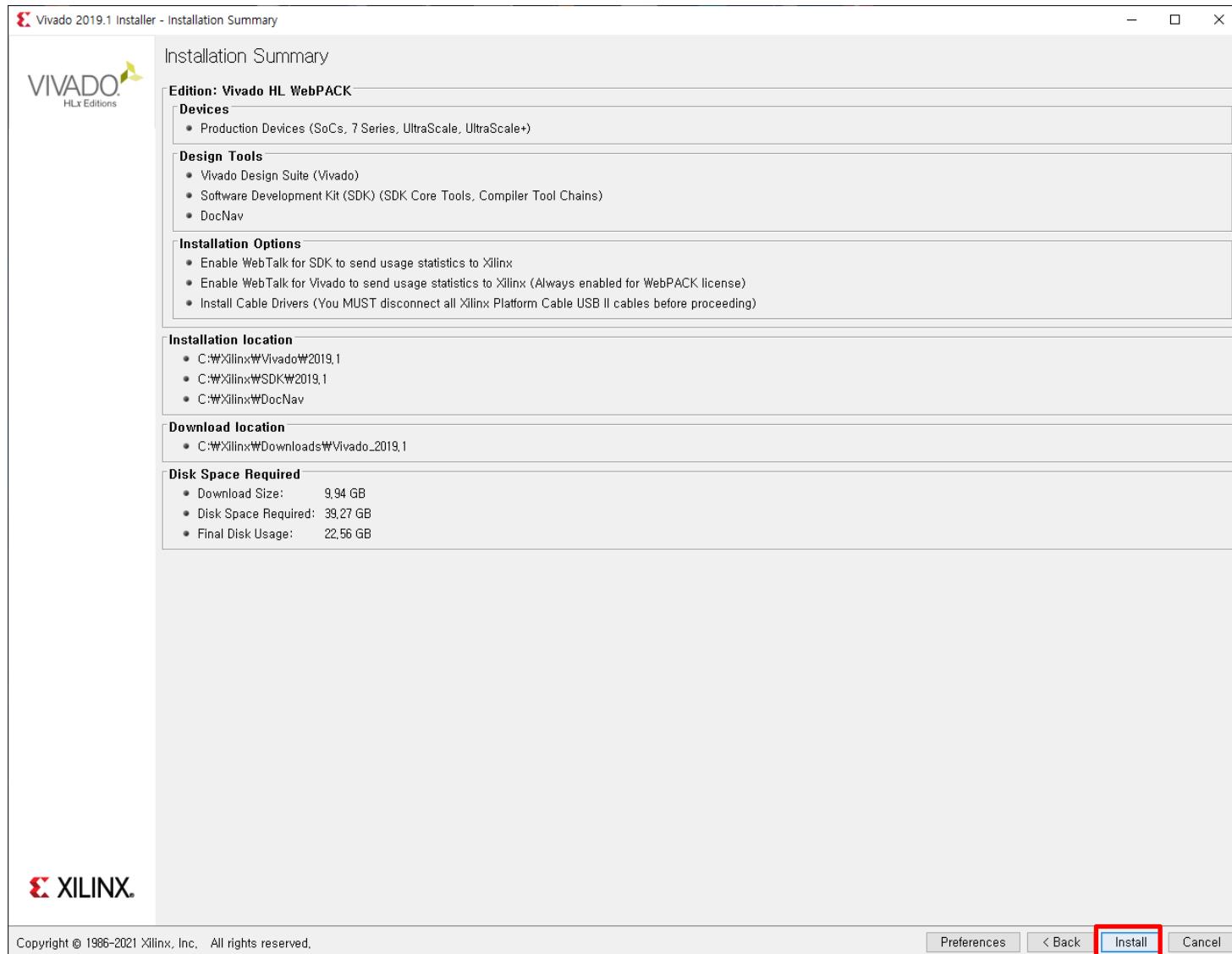
- ◆ In order to reduce required disk space, uncheck devices except for Zynq-7000 series which include our FPGA chip. Of course, it's not necessary if your computer has enough free disk space.



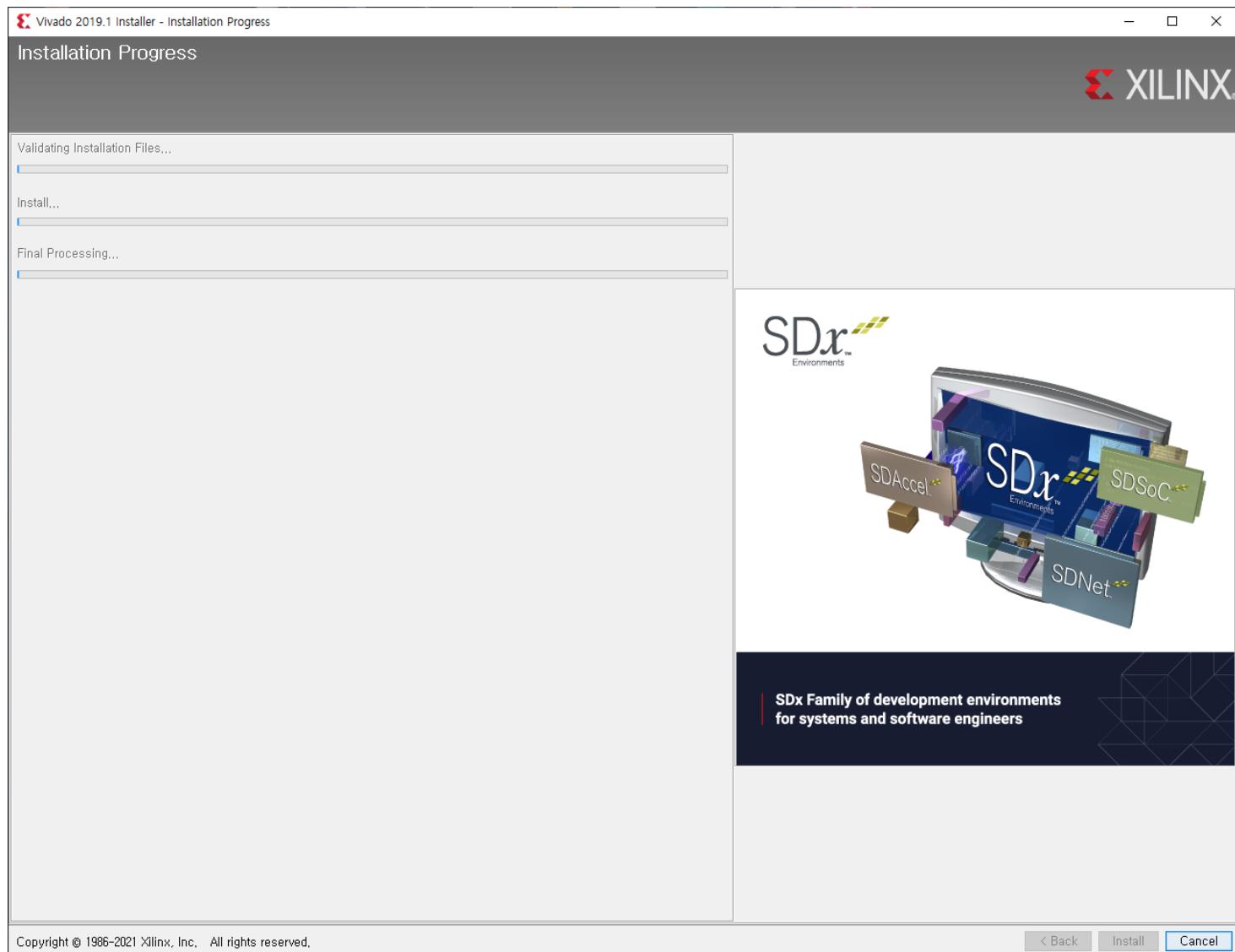
Download Vivado



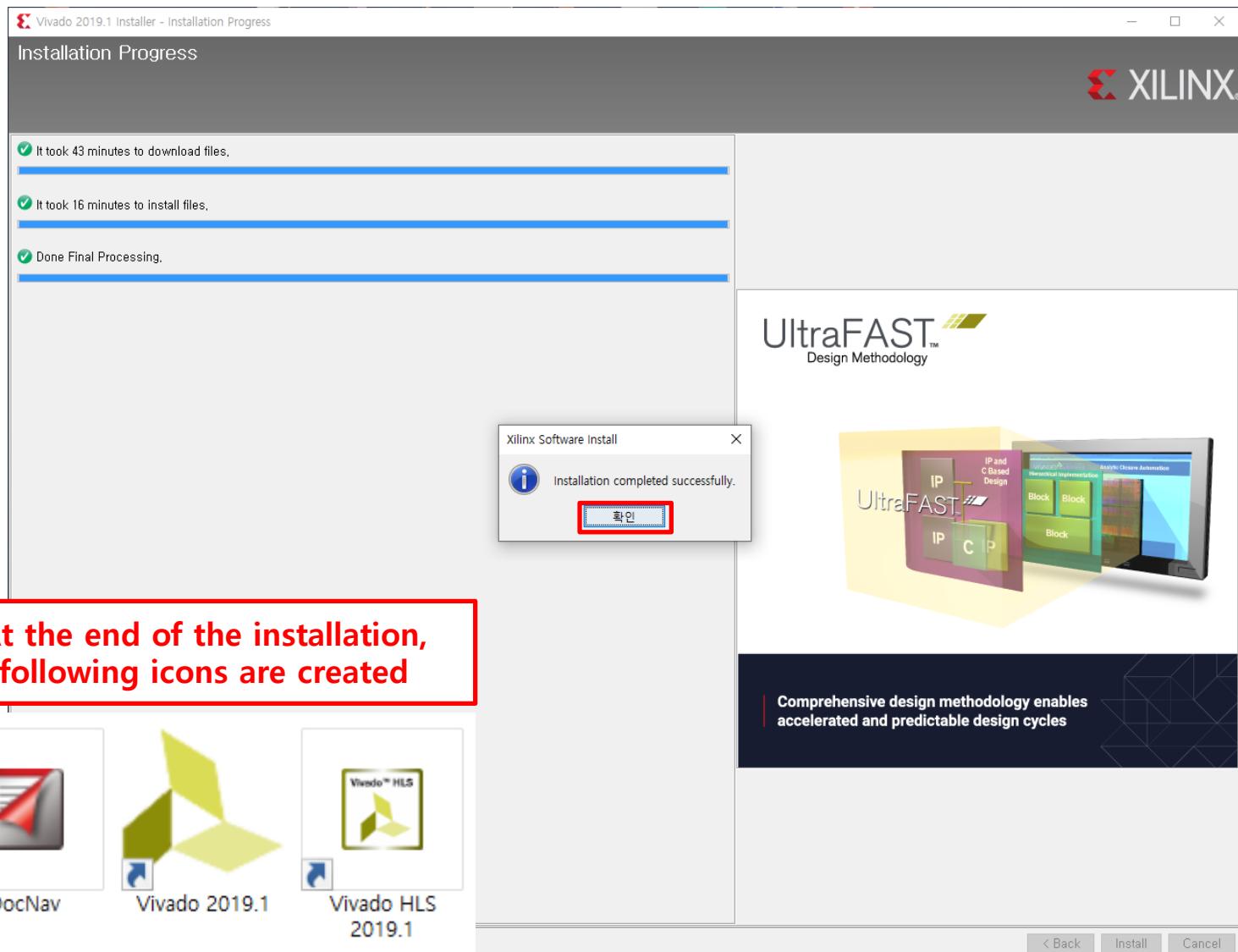
Download Vivado



Download Vivado



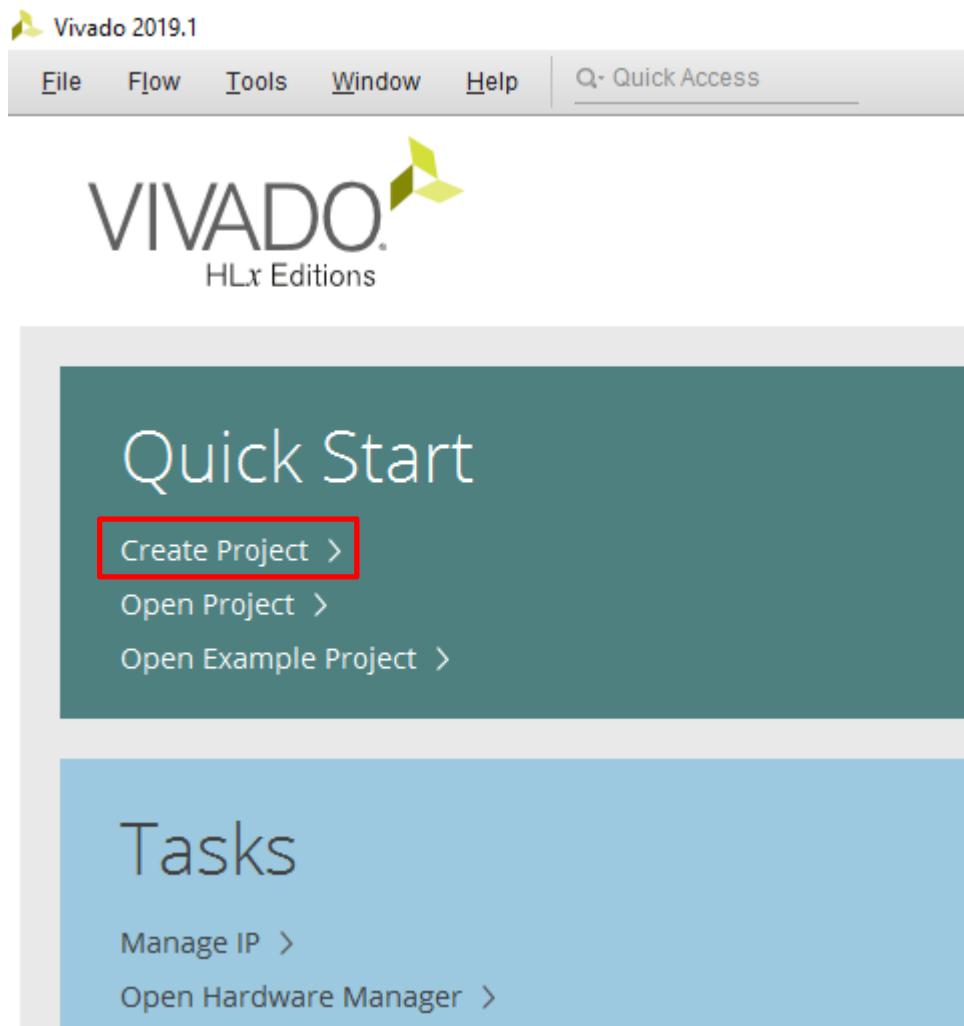
Download Vivado



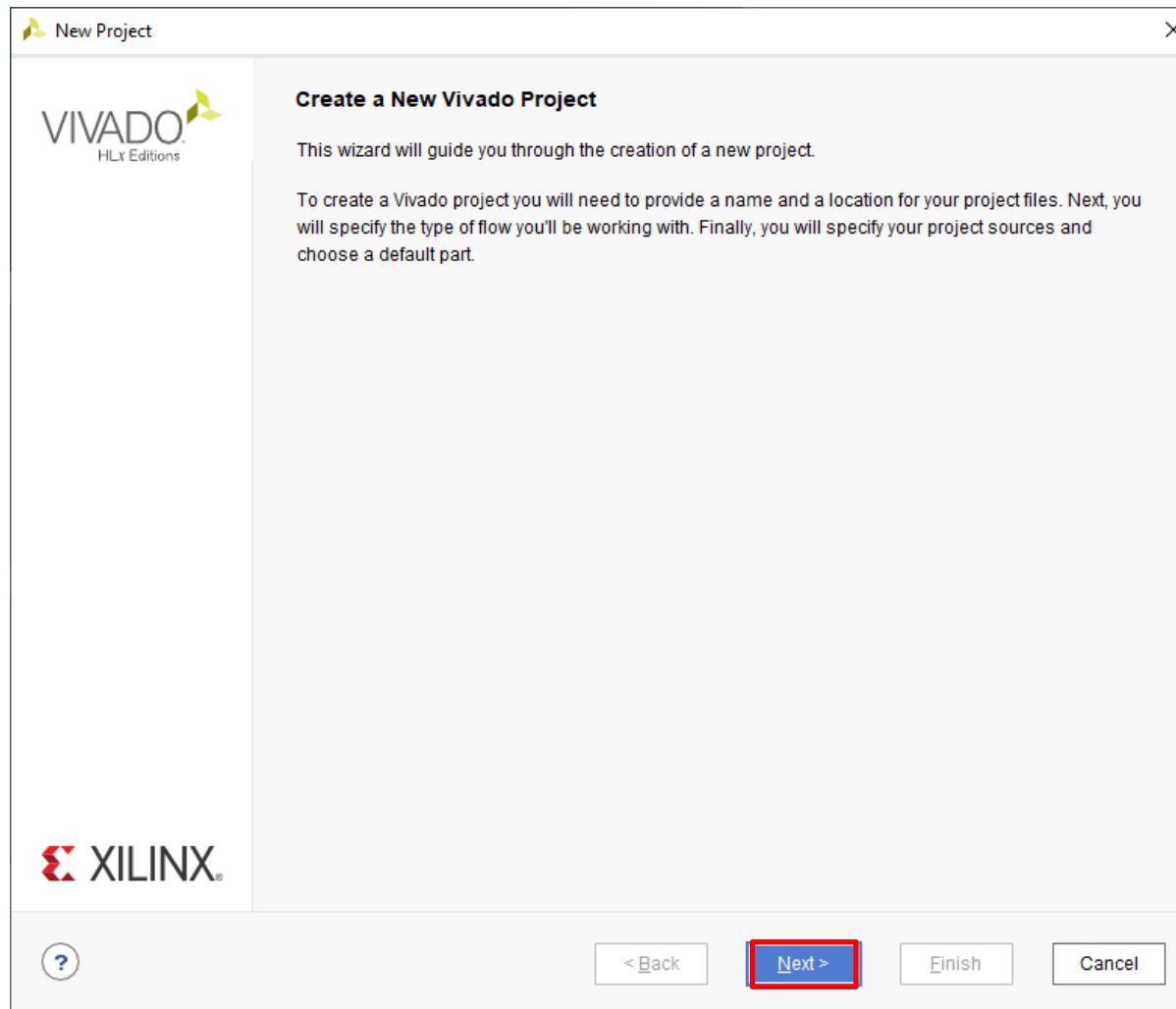
Vivado – Create project

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ Vivado – Simulation
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

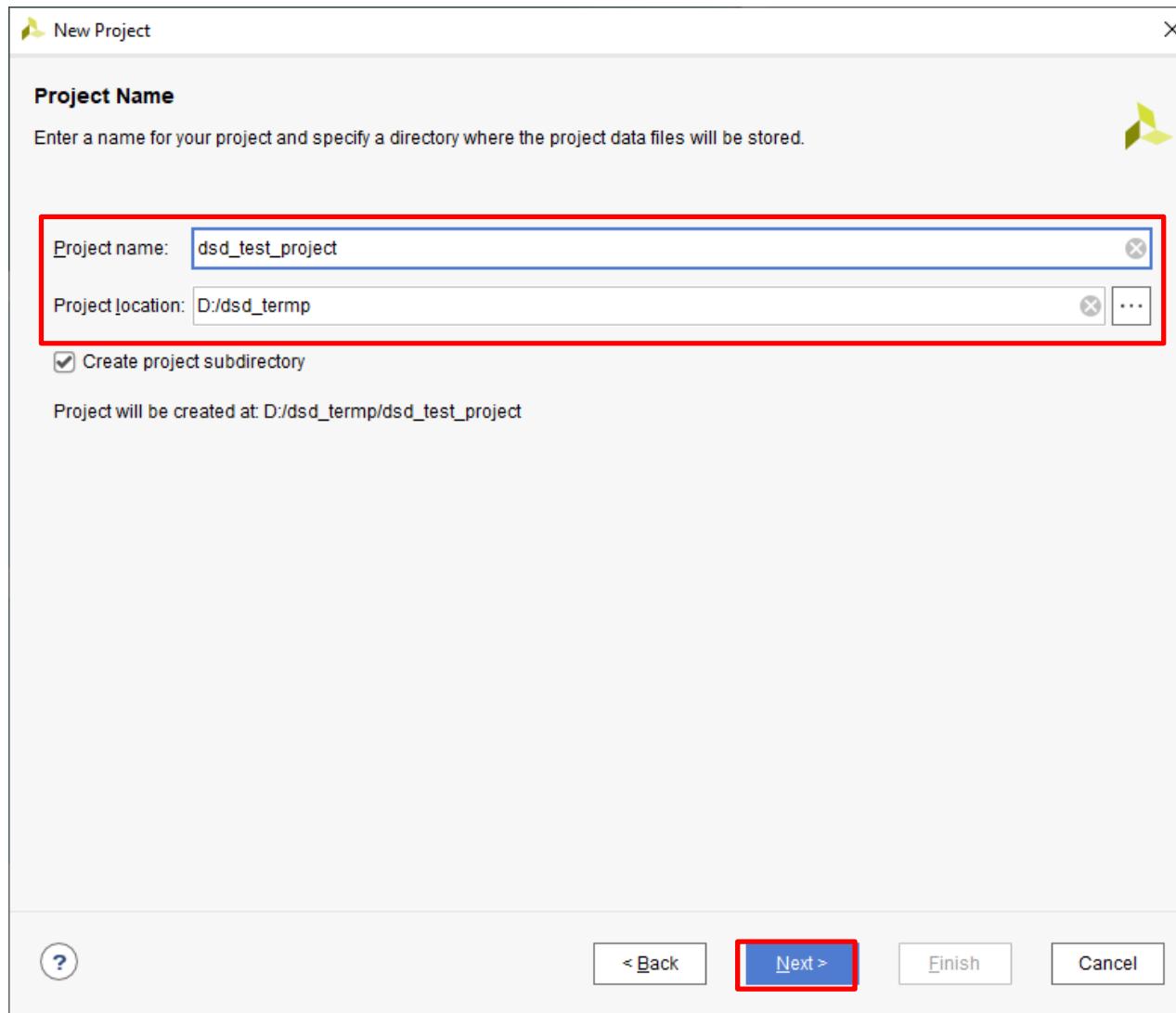
Vivado – Create Project



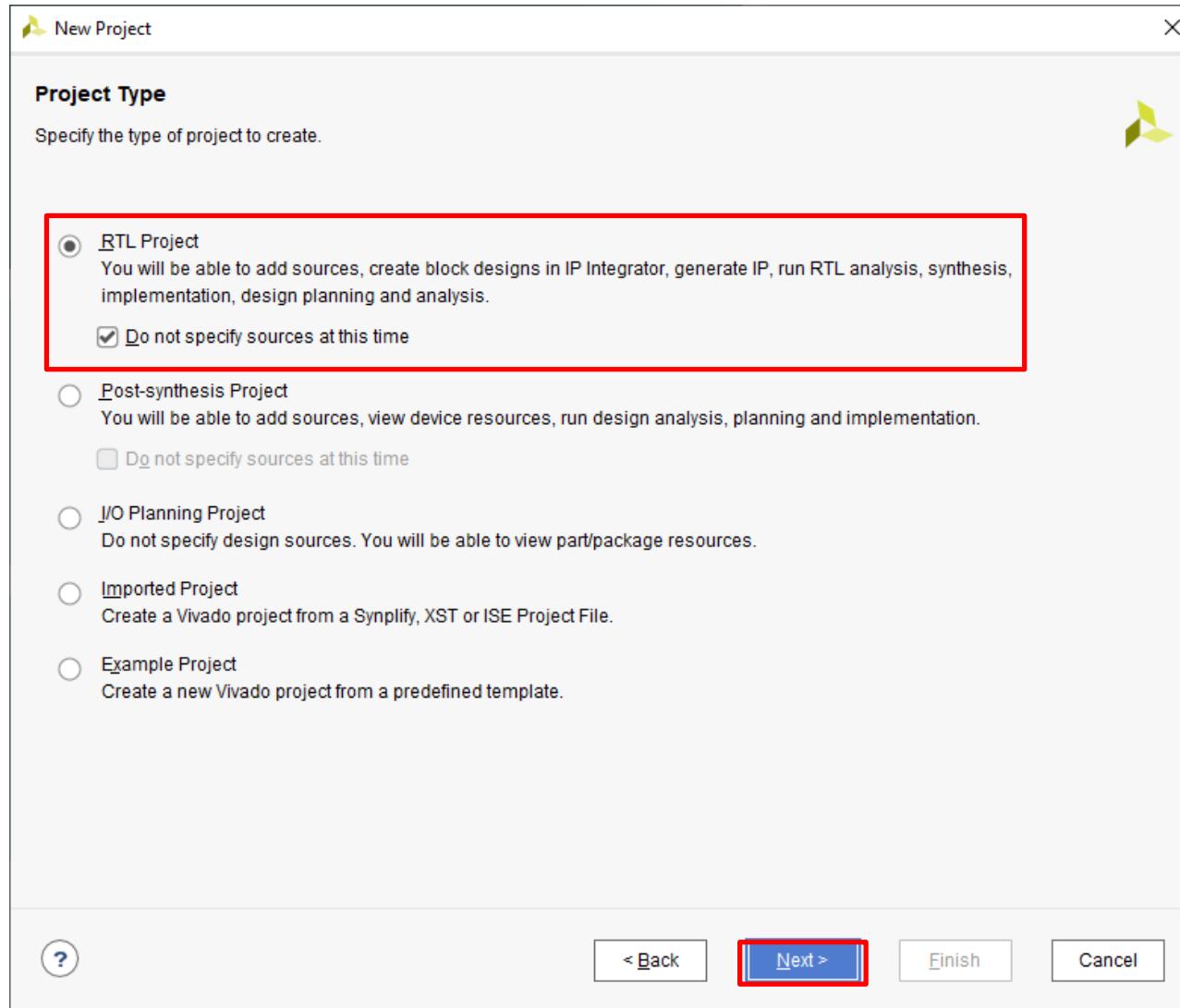
Vivado – Create Project



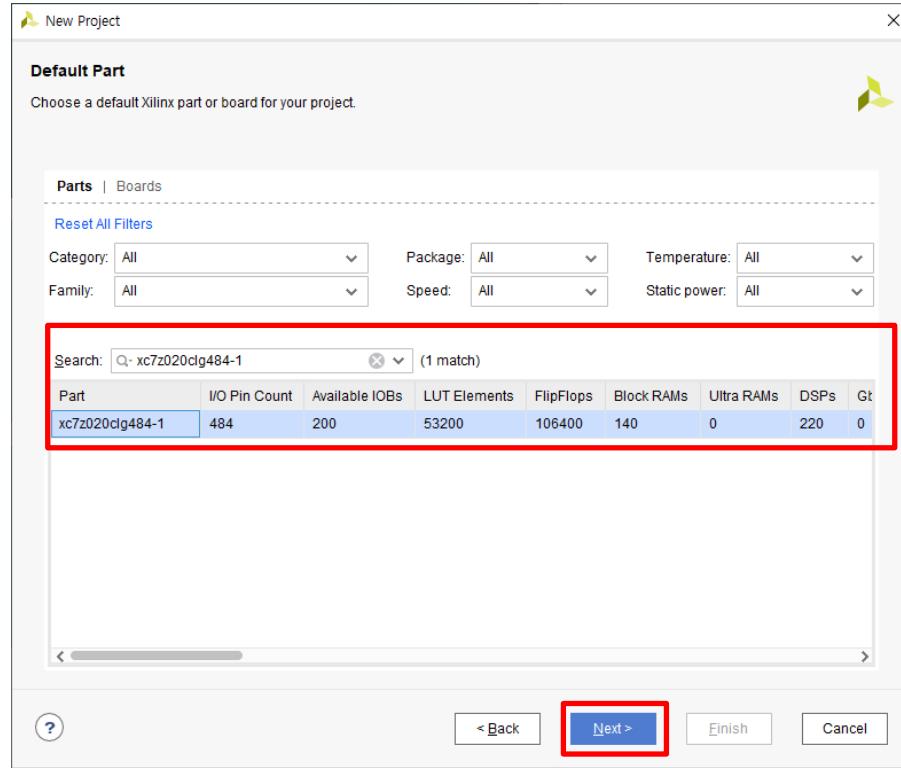
Vivado – Create Project



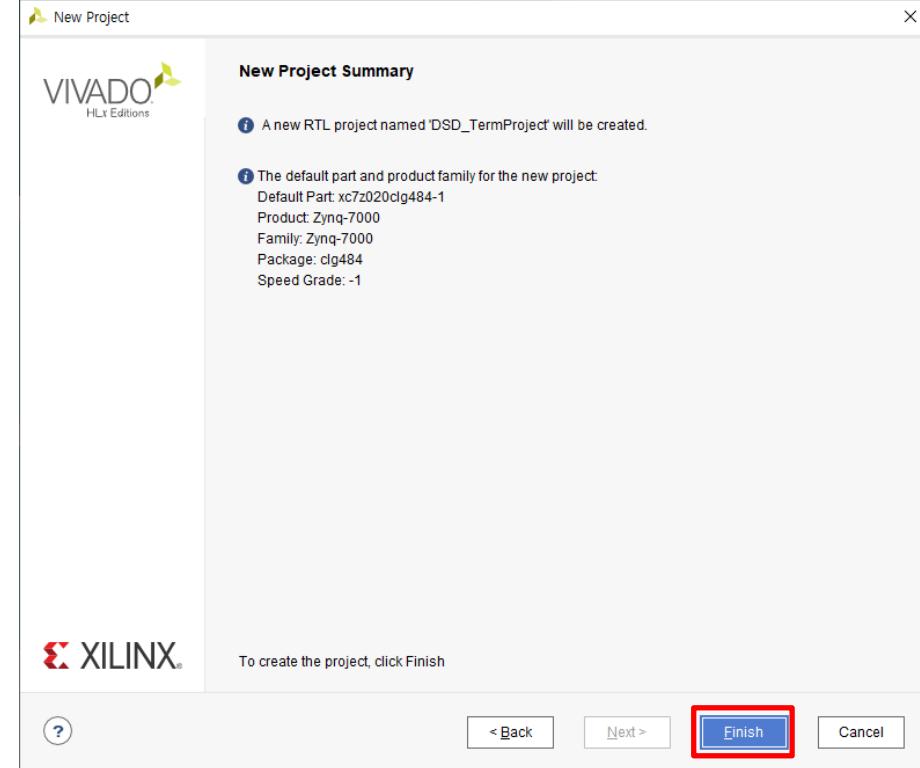
Vivado – Create Project



Vivado – Create Project



Parts name : xc7z020clg484-1



Vivado – Create Project

The screenshot shows the Vivado 2019.1 Project Manager interface. The left sidebar, titled "Flow Navigator", contains several sections: PROJECT MANAGER (Settings, Add Sources, Language Templates, IP Catalog), IP INTEGRATOR (Create Block Design, Open Block Design, Generate Block Design), SIMULATION (Run Simulation), RTL ANALYSIS (Open Elaborated Design), SYNTHESIS (Run Synthesis, Open Synthesized Design), IMPLEMENTATION (Run Implementation, Open Implemented Design), and PROGRAM AND DEBUG (Generate Bitstream, Open Hardware Manager). The central area is divided into three main sections: "Sources" (listing Design Sources, Constraints, Simulation Sources like sim_1, and Utility Sources), "Project Summary" (containing Overview and Dashboard tabs, Settings and Edit tabs, and detailed project information such as Project name: dsd_test_project, Project location: D:/dsd_temp/dsd_test_project, Product family: Zynq-7000, etc.), and "Design Runs" (listing Tcl Console, Messages, Log, Reports, and Design Runs tabs, showing a table of design runs for synth_1 and impl_1 with columns for Name, Constraints, Status, WNS, TNS, WHS, THS, TPWS, Total Power, Failed Routes, LUT, FF, BRAMs, URAM, DSP, Start, Elapsed, and Run Strategy). Red boxes highlight the Flow Navigator, Sources, and Project Summary sections.

PROJECT MANAGER - dsd_test_project

Sources

- Design Sources
- Constraints
- Simulation Sources
 - sim_1
 - Utility Sources

Project Summary

Overview | Dashboard

Settings Edit

Project name: dsd_test_project
Project location: D:/dsd_temp/dsd_test_project
Product family: Zynq-7000
Project part: xc7z020clg484-1
Top module name: Not defined
Target language: Verilog
Simulator language: Mixed

Synthesis

Status: Not started
Messages: No errors or warnings
Part: xc7z020clg484-1
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports
Incremental synthesis: None

Implementation

Status: Not started
Messages: No errors or warnings
Part: xc7z020clg484-1
Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Implementation Default Reports
Incremental implementation: None

DRC Violations

Timing

Run Implementation

Utilization

Run Synthesis to see utilization results

Power

Run Implementation to see power results

Tcl Console | Messages | Log | Reports | Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2019)
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Impleme)

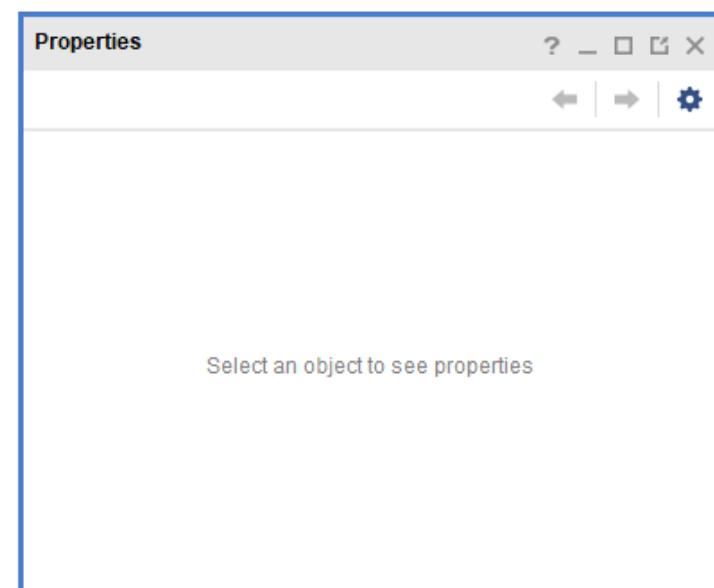
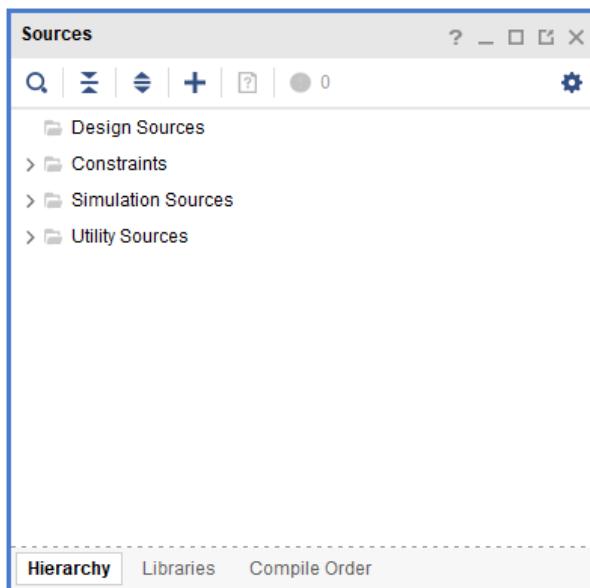
Vivado - Default Windows



- ◆ You can see the flow navigator on the left side of Vivado.
- ◆ Main tools are located from top to bottom. In order to generate bit file and program it onto FPGAs, you have to follow this flow.

Vivado - Default Windows

- ◆ In sources window, you can add or remove source files such as design sources, constraints and simulation sources.
- ◆ In properties window, you can see the properties of object.



Vivado - Default Windows

- ◆ In project summary window, you can see the major information of design.

The screenshot shows the Vivado Project Summary window with the following details:

Project name: dsd_test_project
Project location: D:/dsd_termpp/dsd_test_project
Product family: Zynq-7000
Project part: xc7z020clg484-1
Top module name: Not defined
Target language: Verilog
Simulator language: Mixed

Synthesis

Status:	Not started	Status:	Not started
Messages:	No errors or warnings	Messages:	No errors or warnings
Part:	xc7z020clg484-1	Part:	xc7z020clg484-1
Strategy:	Vivado Synthesis Defaults	Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Synthesis Default Reports	Report Strategy:	Vivado Implementation Default Reports
Incremental synthesis:	None	Incremental implementation:	None

Implementation

Status:	Not started	Status:	Not started
Messages:	No errors or warnings	Messages:	No errors or warnings
Part:	xc7z020clg484-1	Part:	xc7z020clg484-1
Strategy:	Vivado Implementation Defaults	Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports	Report Strategy:	Vivado Implementation Default Reports
Incremental implementation:	None	Incremental implementation:	None

DRC Violations

Run Implementation to see DRC results

Timing

Run Implementation to see timing results

Utilization

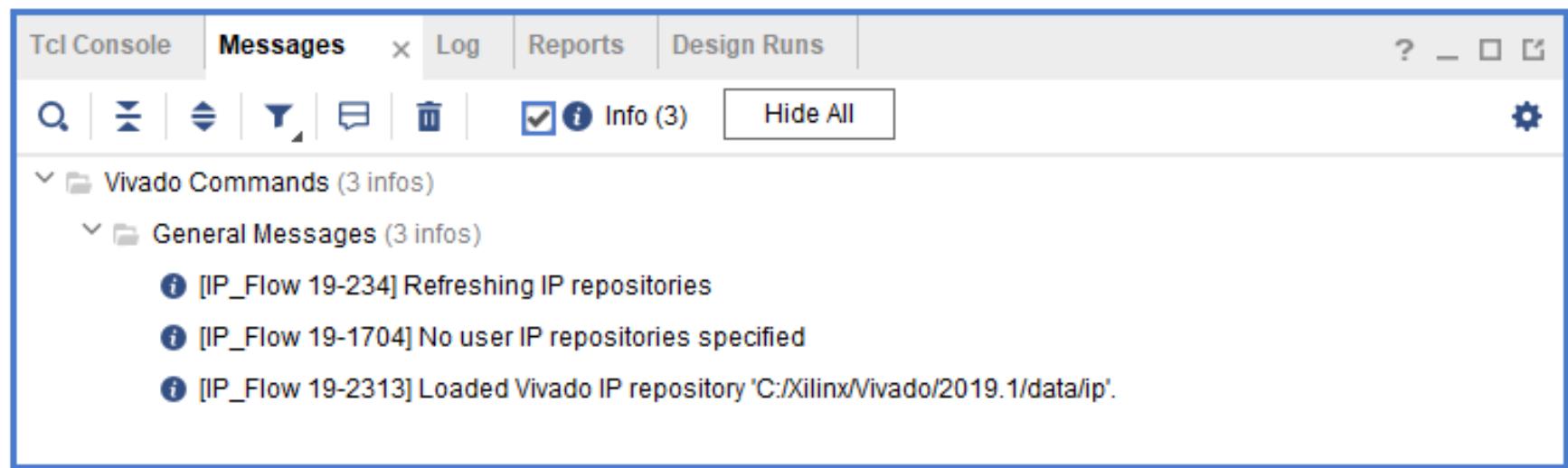
Run Synthesis to see utilization results

Power

Run Implementation to see power results

Vivado - Default Windows

- ◆ In the below window, there are 5 tabs.
- ◆ We will discuss it later.



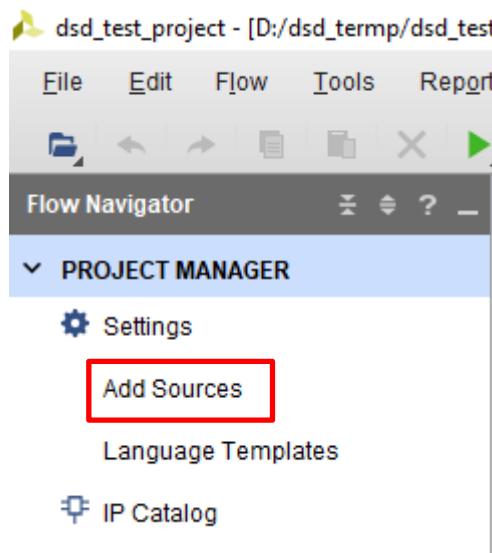
Vivado – Add/Create sources

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ **Vivado – Add/Create sources**
- ◆ Vivado – Simulation
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

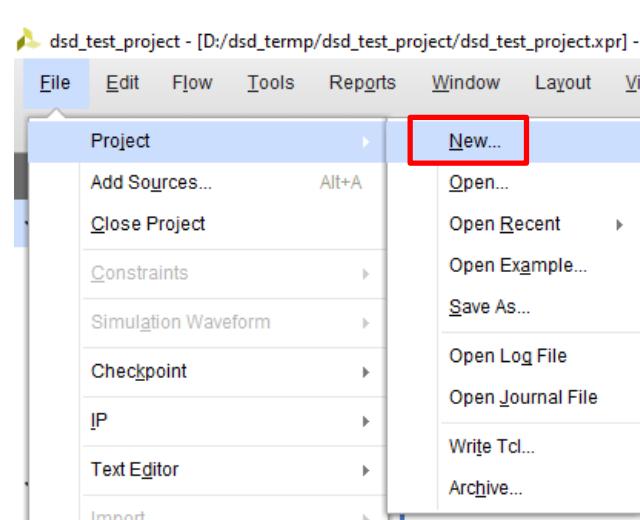
Vivado – Add sources

- ◆ You can add sources by following ways.

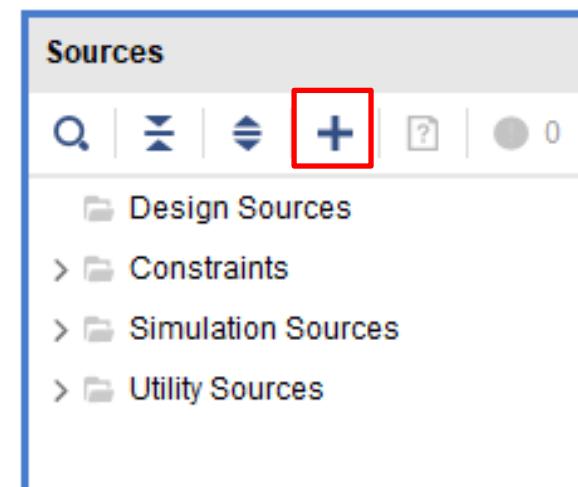
1



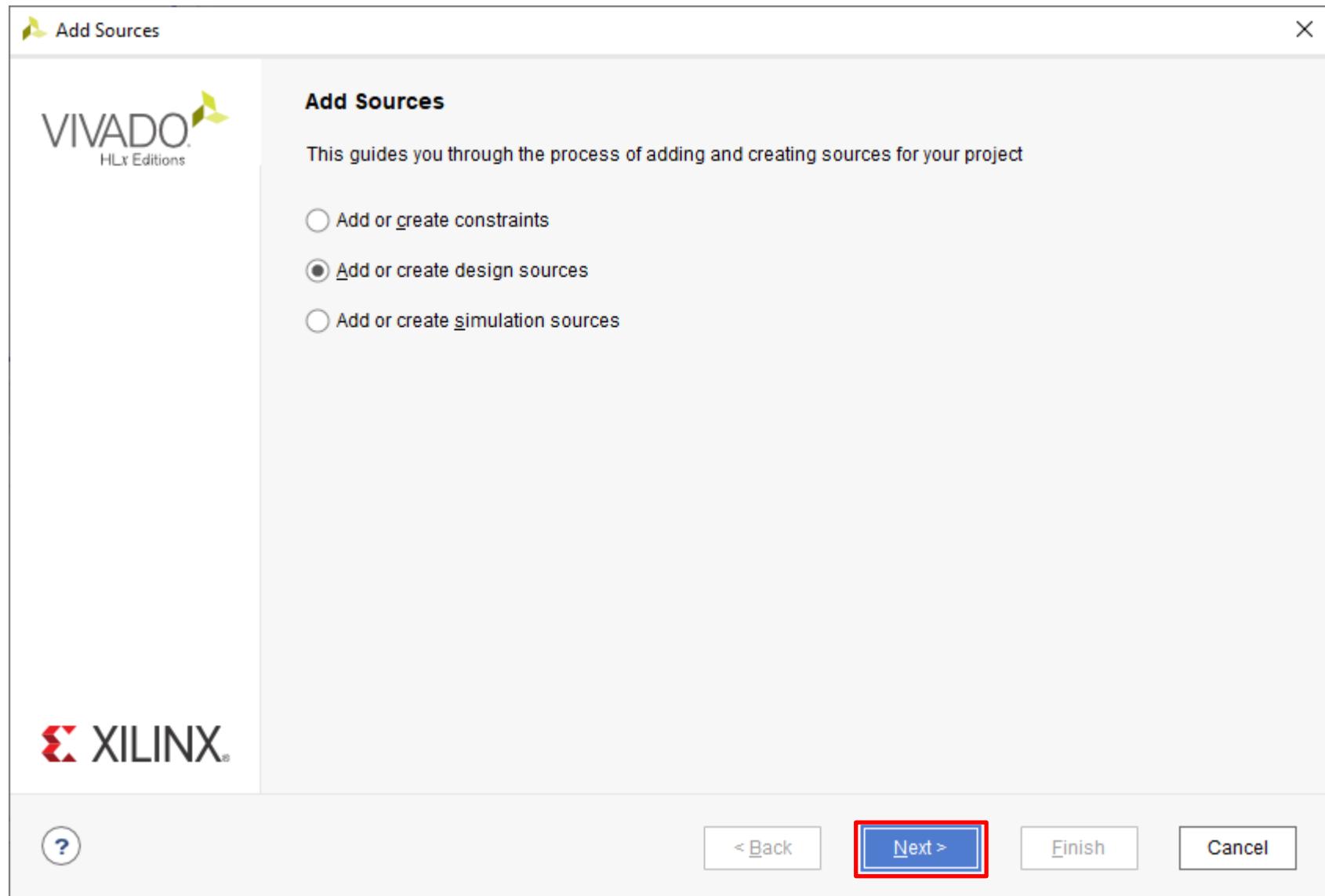
2



3

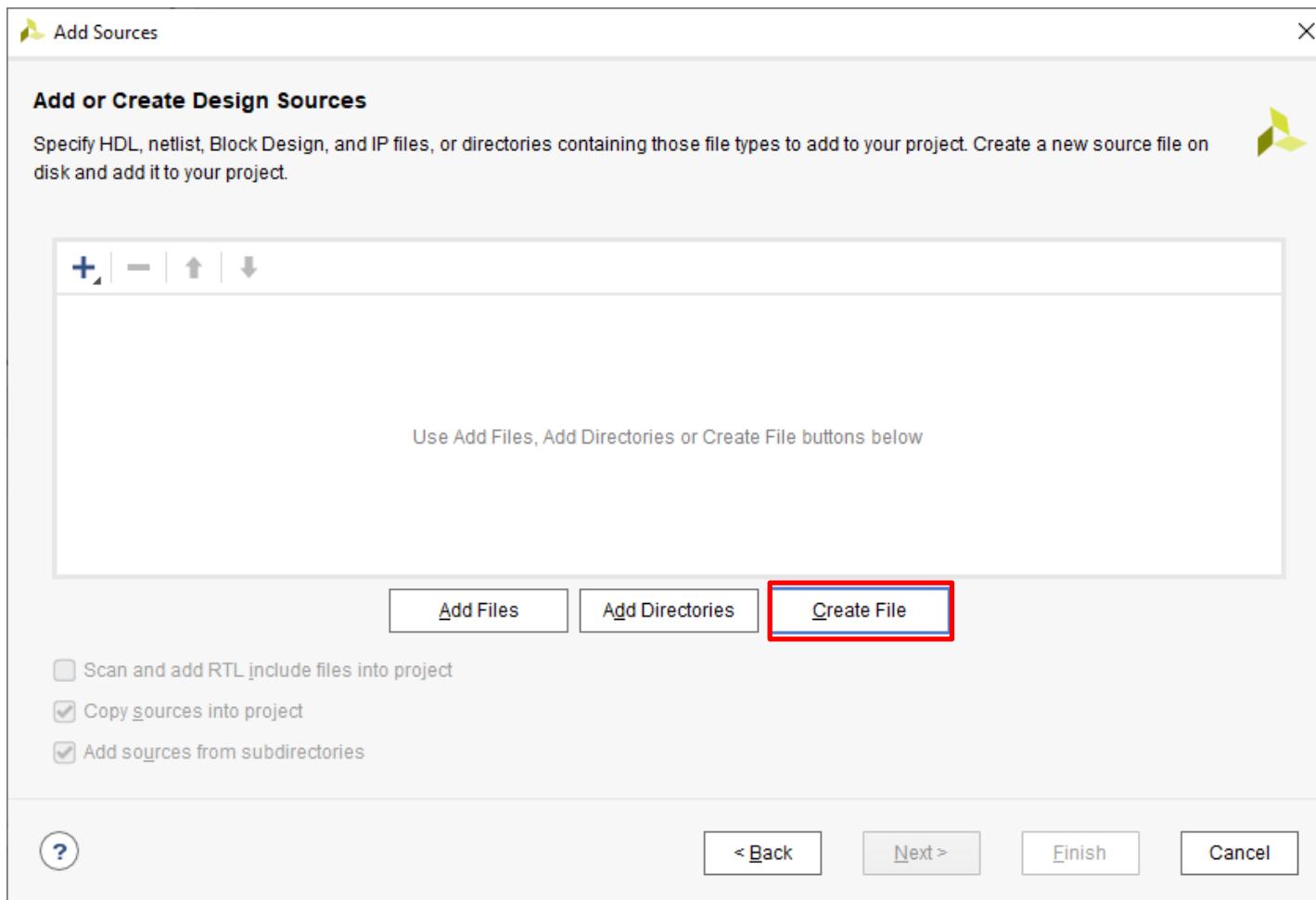


Vivado – Add sources



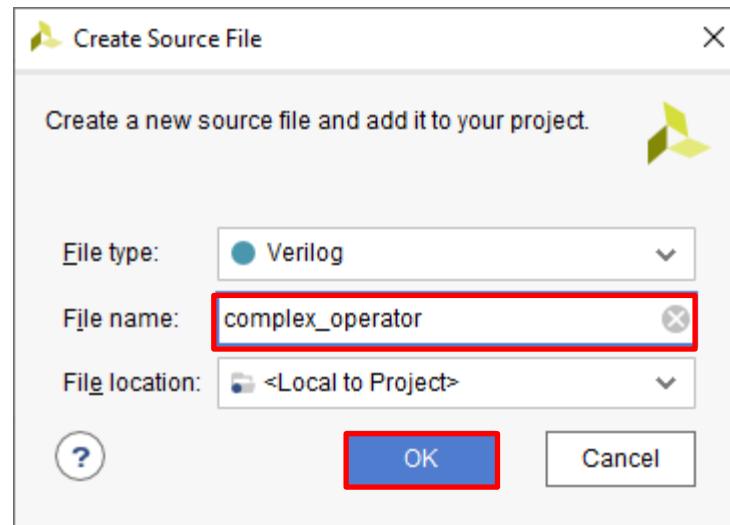
Vivado – Add sources

- ◆ Click “Create File”



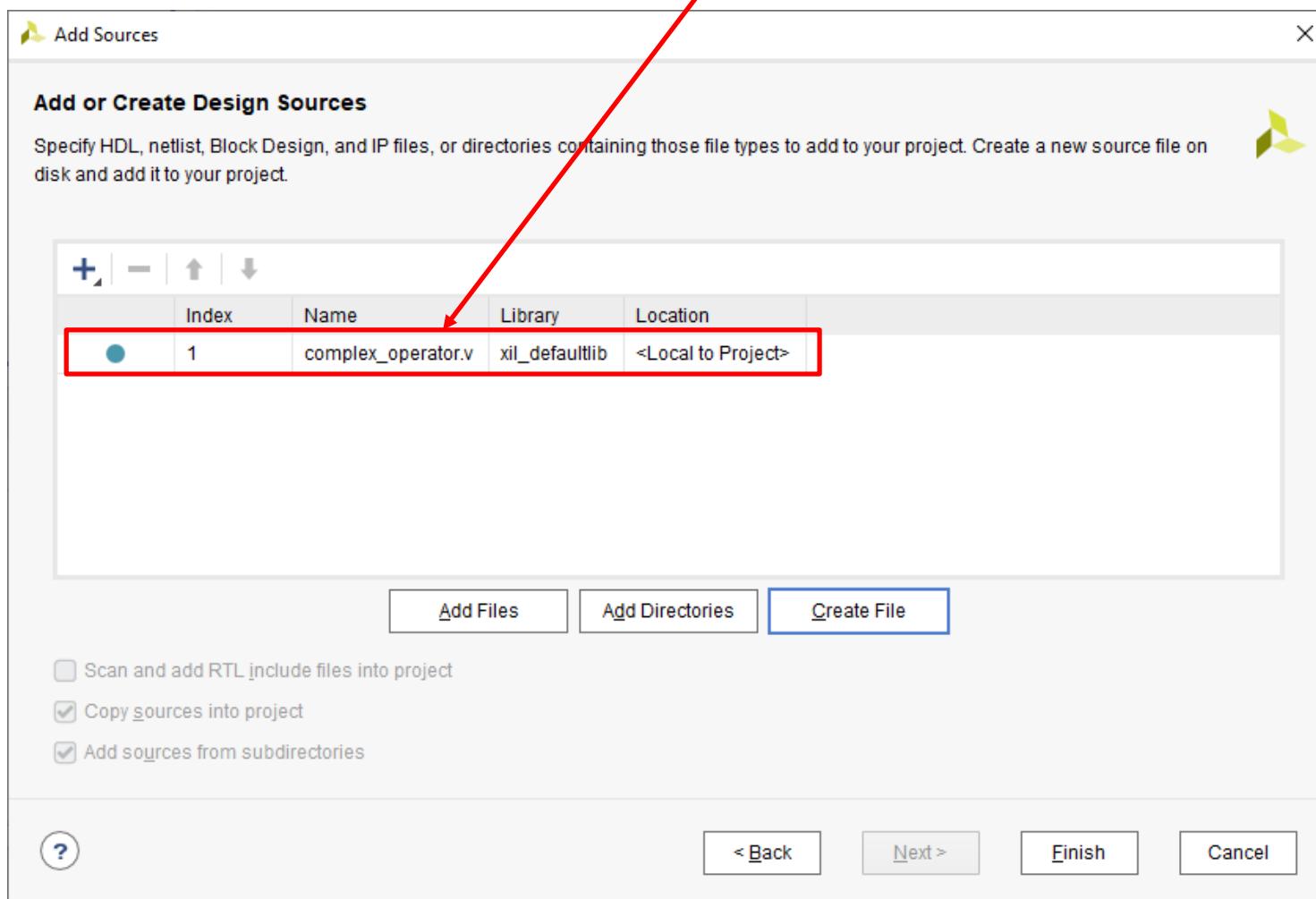
Vivado – Add sources

- ◆ Choose file type and write file name.



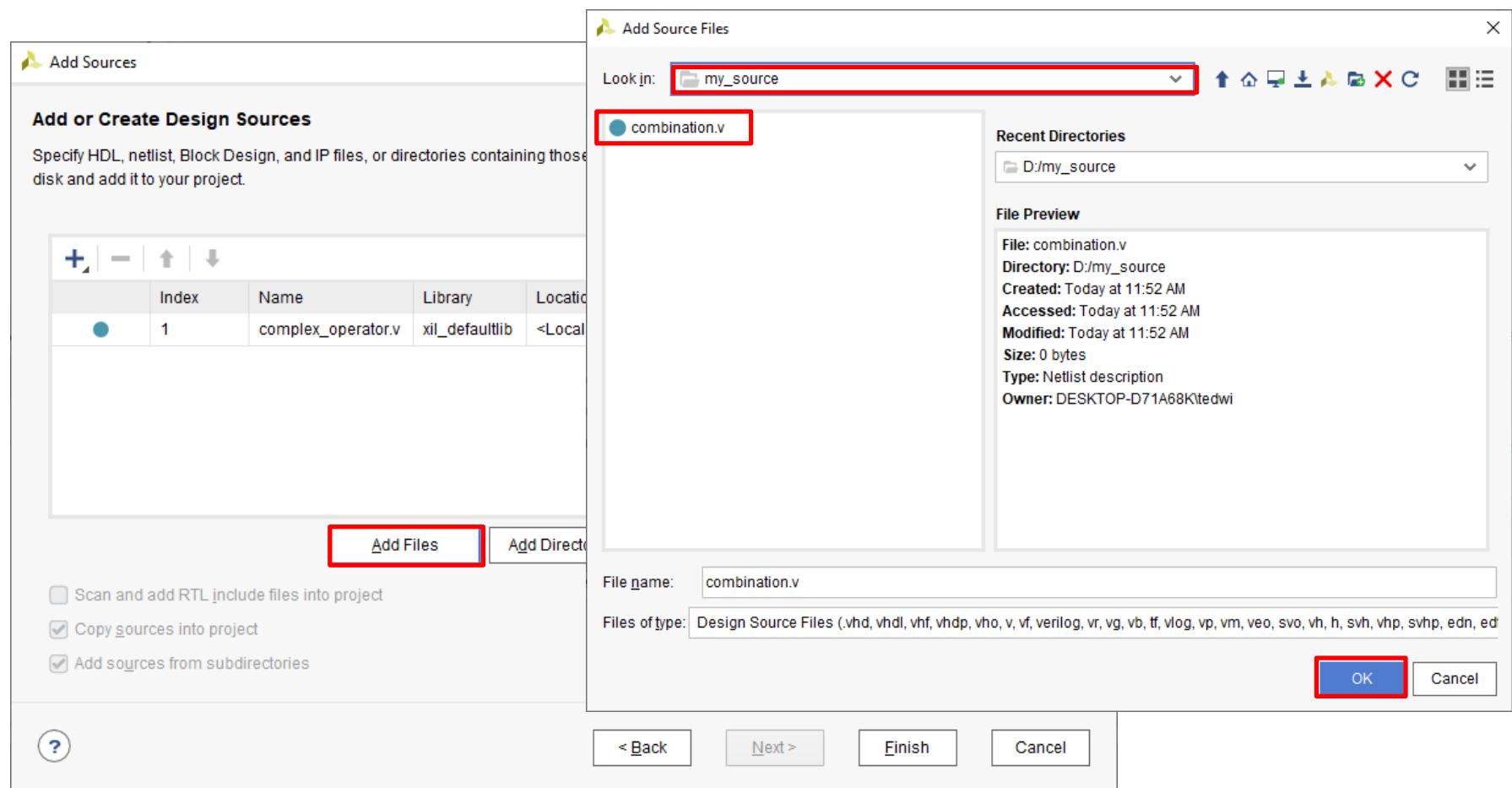
Vivado – Add sources

- ◆ Now you can see the new file is added.



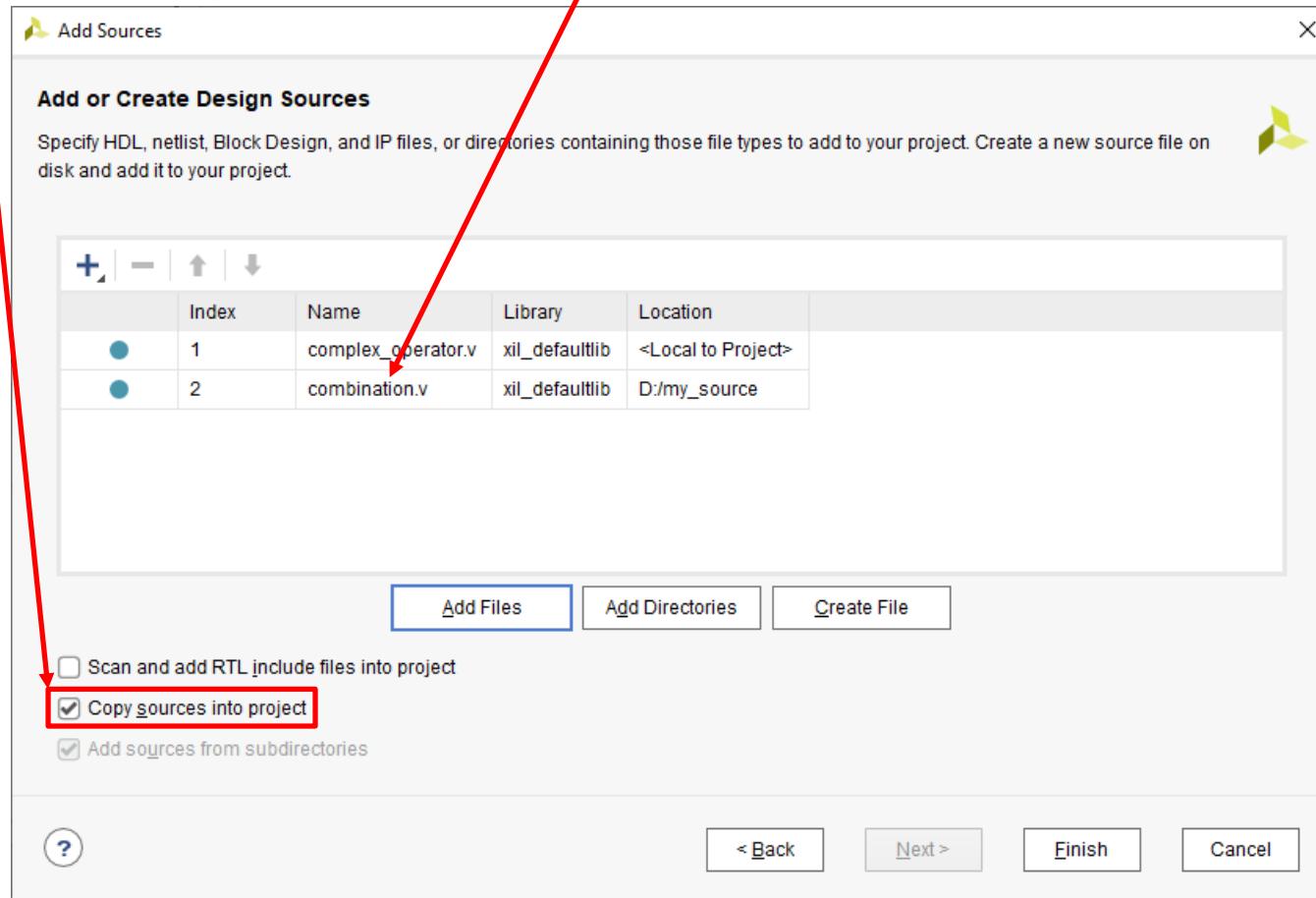
Vivado – Add sources

◆ You can also add existing sources.



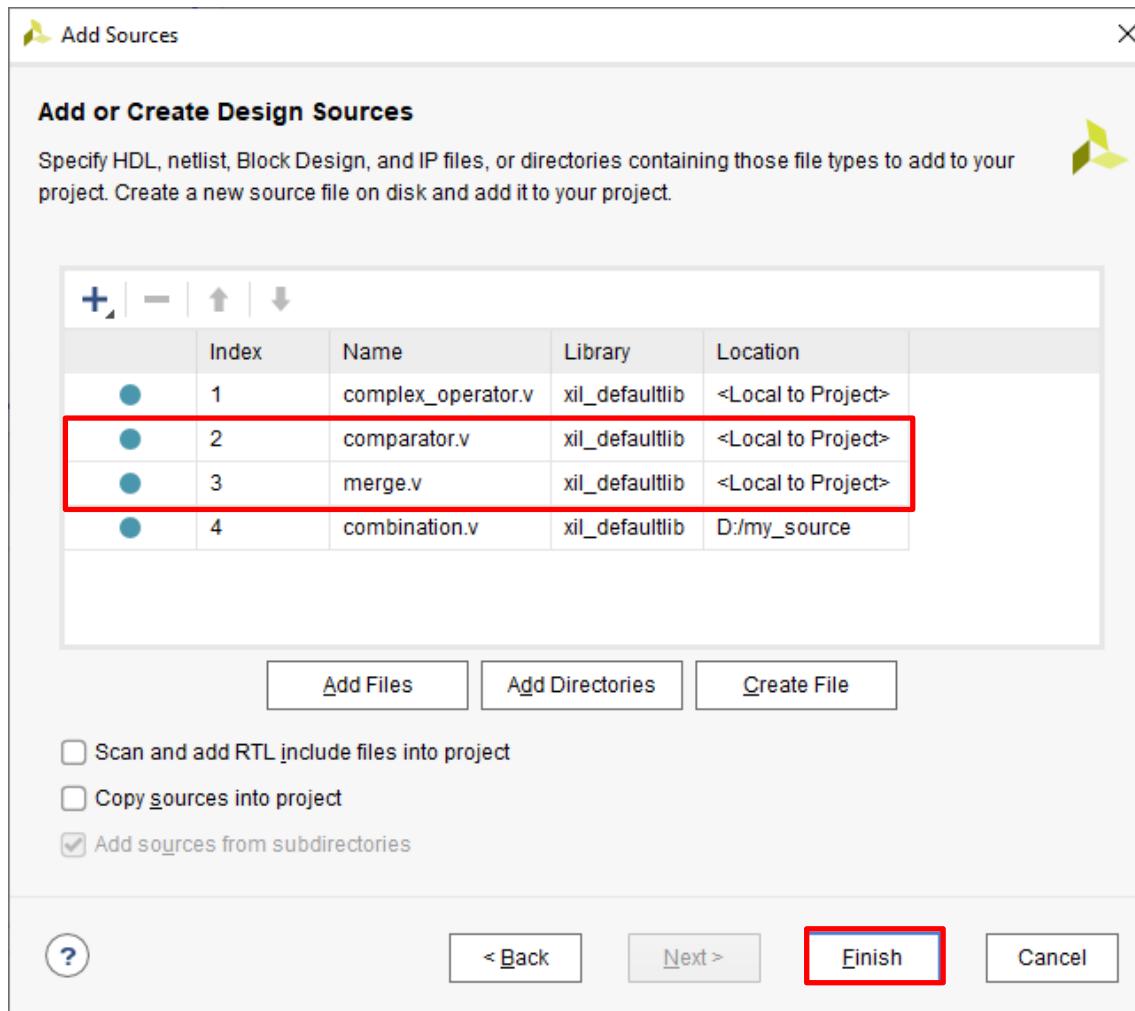
Vivado – Add sources

- ◆ Then, you can see your source file is added.
- ◆ If you don't want to change original file, then check "Copy sources into project" option.

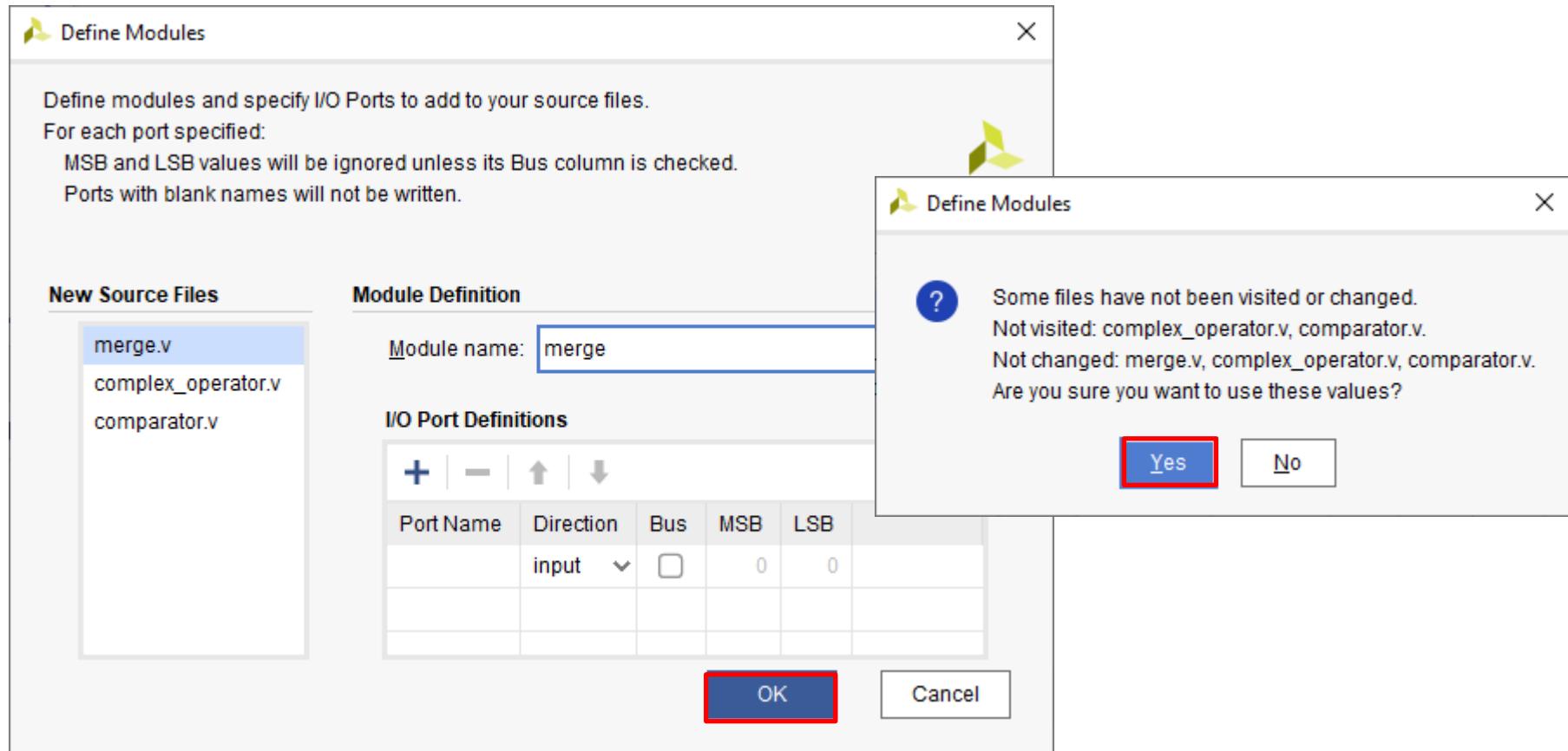


Vivado – Add sources

- ◆ Create 'comparator.v' and 'merge.v' file as you did before.

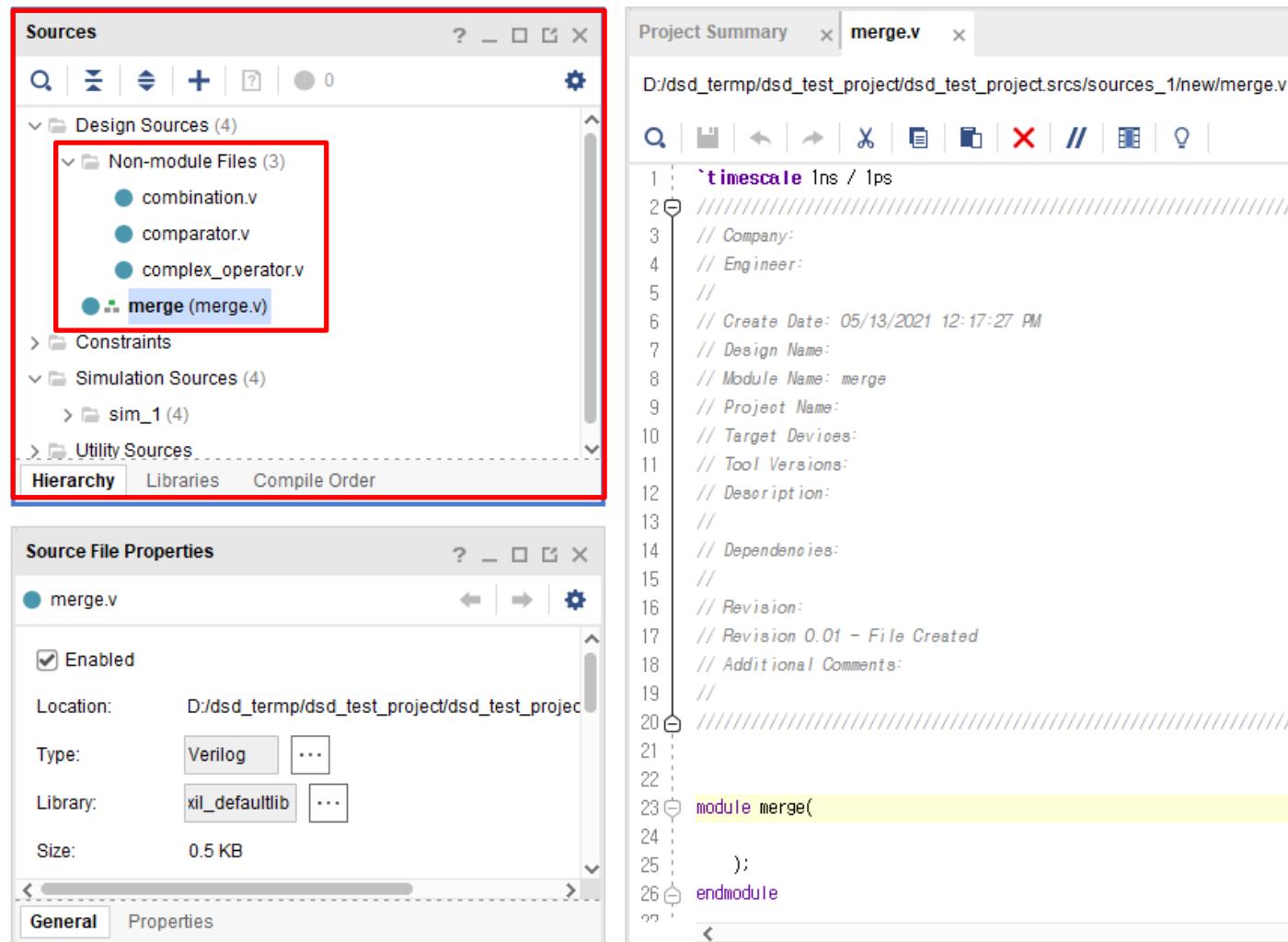


Vivado – Add sources



Vivado – Add sources

- ◆ You can see design sources through source tap
- ◆ You can see and edit source files by opening file(double click or right click -> 'open file').



Vivado – Add sources

- ◆ Sources are organized hierarchically

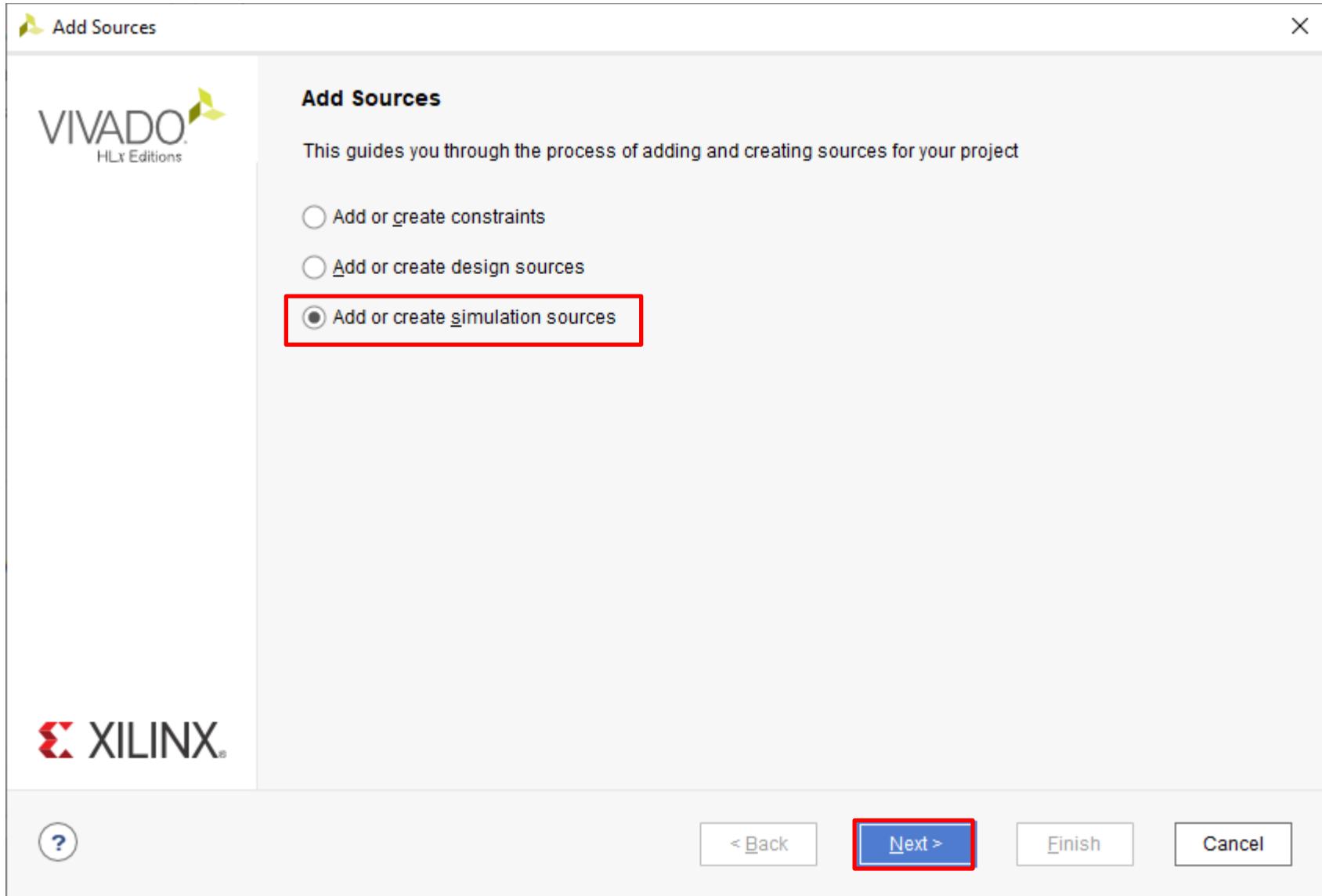
The screenshot shows the Vivado IDE interface with two main windows:

- Sources Browser:** On the left, it displays a tree view of project sources. A red box highlights the "Design Sources" section, which contains a folder named "merge" (merge.v). Inside "merge", there are three sub-items: "complex_operator_inst", "combination_inst", and "comparator_inst".
- Code Editor:** On the right, the "merge.v" file is open in the editor. The code defines a module "merge" with parameters, inputs, outputs, and wires, along with instantiations of "complex_operator", "combination", and "comparator" modules.

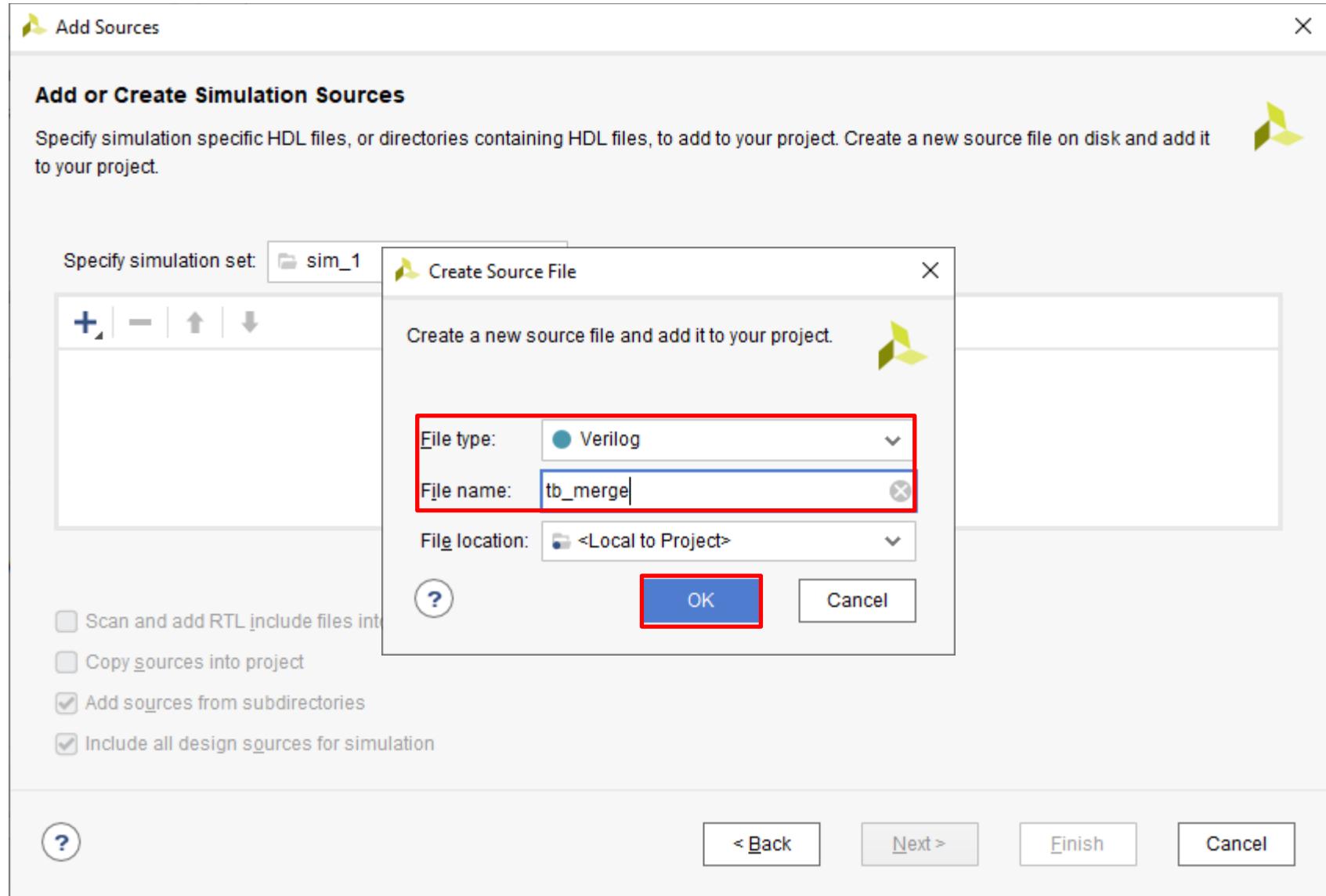
```
// Tool Versions:  
// Description:  
//  
// Dependencies:  
//  
// Revision:  
// Revision 0.01 - File Created  
// Additional Comments:  
//  
module merge #(W = 8);  
    parameter W = 8;  
    input [W-1:0] in1, in2;  
    input [2:0] sel;  
    output [1:0] dout;  
  
    wire [W-1:0] out_compute, out_combination;  
  
    complex_operator #(W) complex_operator_inst (.x(in1), .y(in2), .sel(sel), .z(out_compute));  
    combination #(W) combination_inst (.n1(in1), .n2(in2), .result(out_combination));  
    comparator #(W) comparator_inst (.din1(out_compute), .din2(out_combination), .dout(dout));  
endmodule
```

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ **Vivado – Simulation**
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

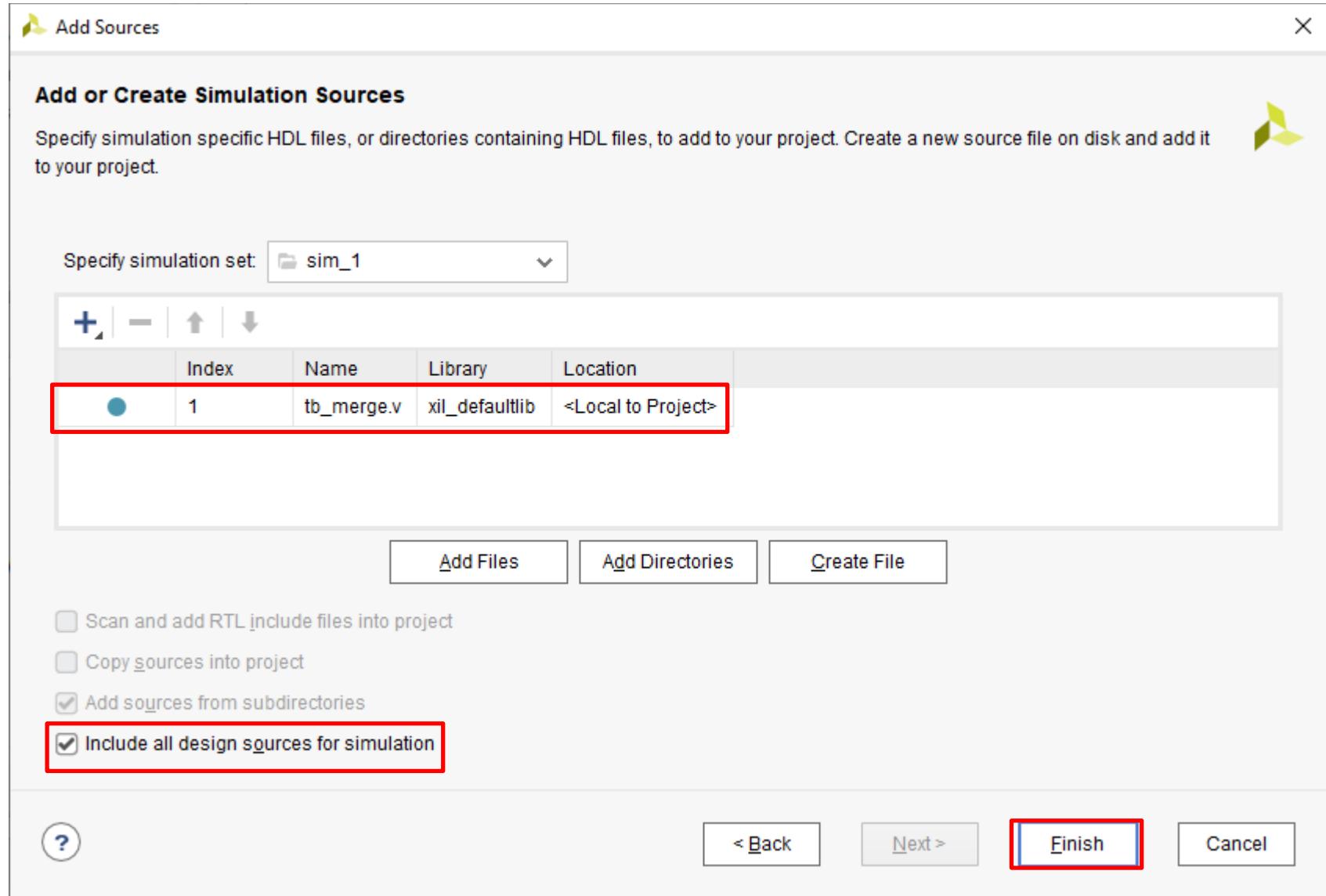
Vivado – Simulation



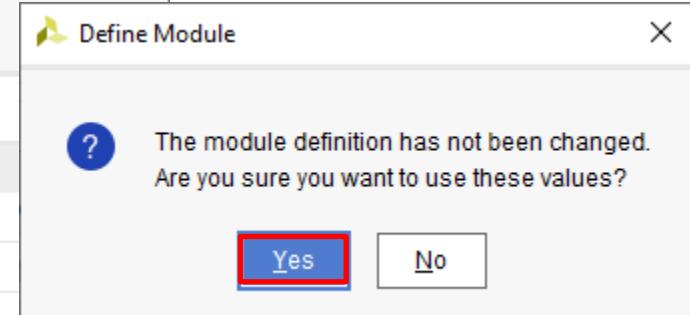
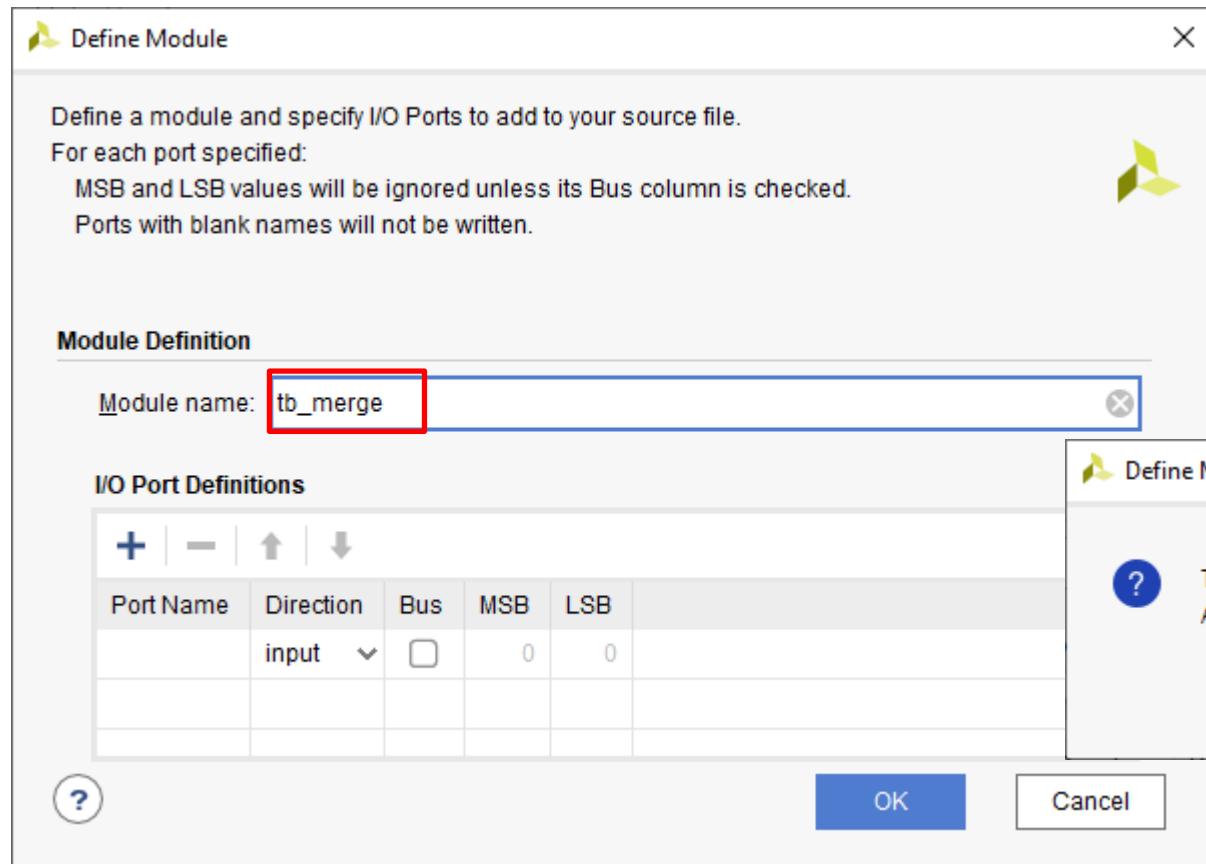
Vivado – Simulation



Vivado – Simulation



Vivado – Simulation



Vivado – Simulation

The screenshot shows the Vivado IDE interface with three main windows:

- Sources Window:** Shows the project structure. A red box highlights the "Simulation Sources" section under "Design Sources". Inside, "merge (merge.v)" and "tb_merge (tb_merge.v)" are listed.
- Project Summary Window:** Displays the current simulation file: tb_merge.v. The code content is as follows:

```
1 `timescale 1ns / 1ps
2 // Company:
3 // Engineer:
4 //
5 // Create Date: 05/13/2021 12:25:26 PM
6 // Design Name:
7 // Module Name: tb_merge
8 // Project Name:
9 // Target Devices:
10 // Tool Versions:
11 // Description:
12 //
13 // Dependencies:
14 //
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 module tb_merge(
21 );
22
23 endmodule
```

- Source File Properties Window:** For tb_merge.v. It shows the file is enabled, located at D:/dsd_termp/dsd_test_project/dsd_test_project, and is a Verilog file in the xil_defaultlib library.

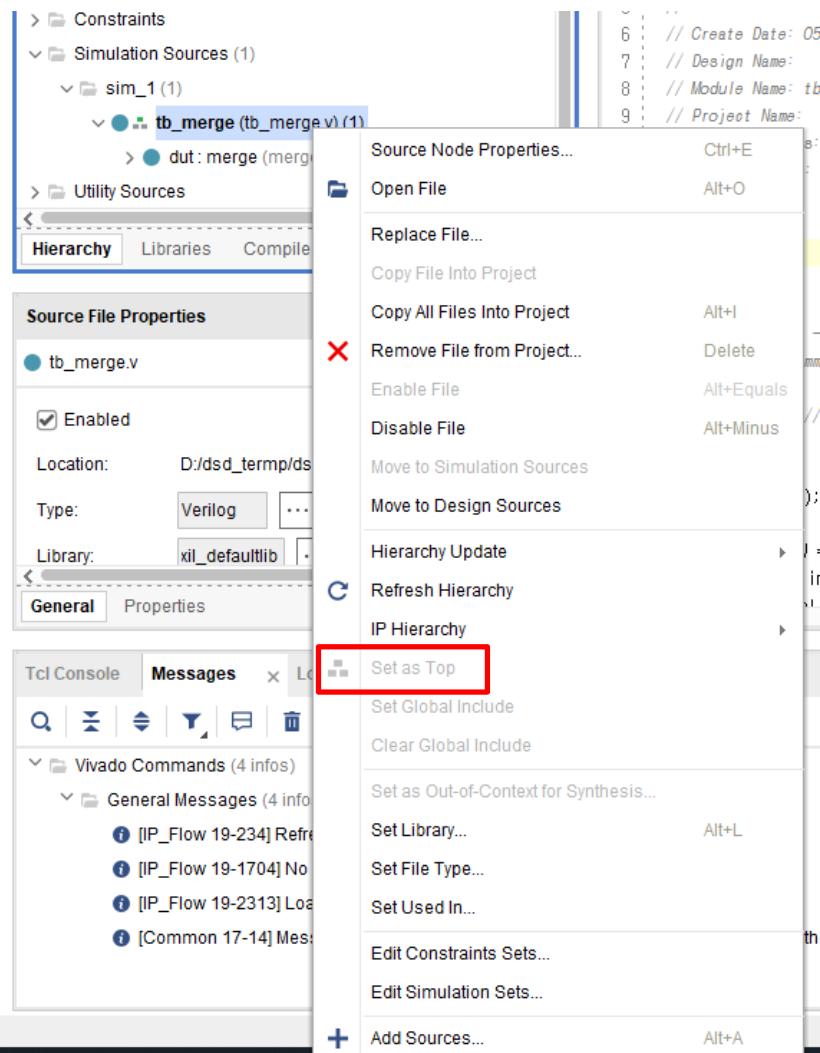
Vivado – Simulation

The screenshot shows the Vivado IDE interface with several open windows:

- Sources** pane (left): Shows the project structure with Design Sources (merge.v), Constraints, Simulation Sources (sim_1), and Utility Sources. The tb_merge (tb_merge.v) file under sim_1 is highlighted with a red box.
- Project Summary** pane (top right): Shows tabs for merge.v, combination.v, comparator.v, complex_operator.v, and tb_merge.v. tb_merge.v is the active tab.
- tb_merge.v** code editor (right): Displays the Verilog testbench code. The code includes a timescale declaration, company and engineer comments, and a module definition for tb_merge. A yellow highlight covers the dependencies section.
- Source File Properties** pane (bottom left): Shows properties for tb_merge.v, including location (D:/dsd_termp/dsd_test_project/dsd_test_project), type (Verilog), and library (xil_defaultlib).

Vivado – Simulation

- ◆ If there are multiple testbench sources, then you can select a source that will be simulated by setting it as top file.



Vivado – Simulation

The screenshot shows the Vivado Flow Navigator interface. On the left, there's a sidebar with various project management and simulation-related options. A red box highlights the 'SIMULATION' section, which contains a 'Run Simulation' option. Another red box highlights the 'Run Behavioral Simulation' option under the 'SIMULATION' menu, which is part of a larger context menu. This context menu also includes 'Run Post-Synthesis Functional Simulation' and 'Run Post-Synthesis Timing Simulation'. Below this, under 'SYNTHESIS', are 'Run Post-Implementation Functional Simulation' and 'Run Post-Implementation Timing Simulation'. At the bottom right of the interface, there's a progress dialog box titled 'Run Simulation' with the message 'Executing analysis and compilation step...'. A red box highlights the text 'wait a minute...' at the bottom of the slide.

Flow Navigator

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

PROGRAM AND DEBUG

- Generate Bitstream
- Open Hardware Manager

Utility Sources

Run Behavioral Simulation

Run Post-Synthesis Functional Simulation

Run Post-Synthesis Timing Simulation

Run Post-Implementation Functional Simulation

Run Post-Implementation Timing Simulation

Location: D:/dsd_te

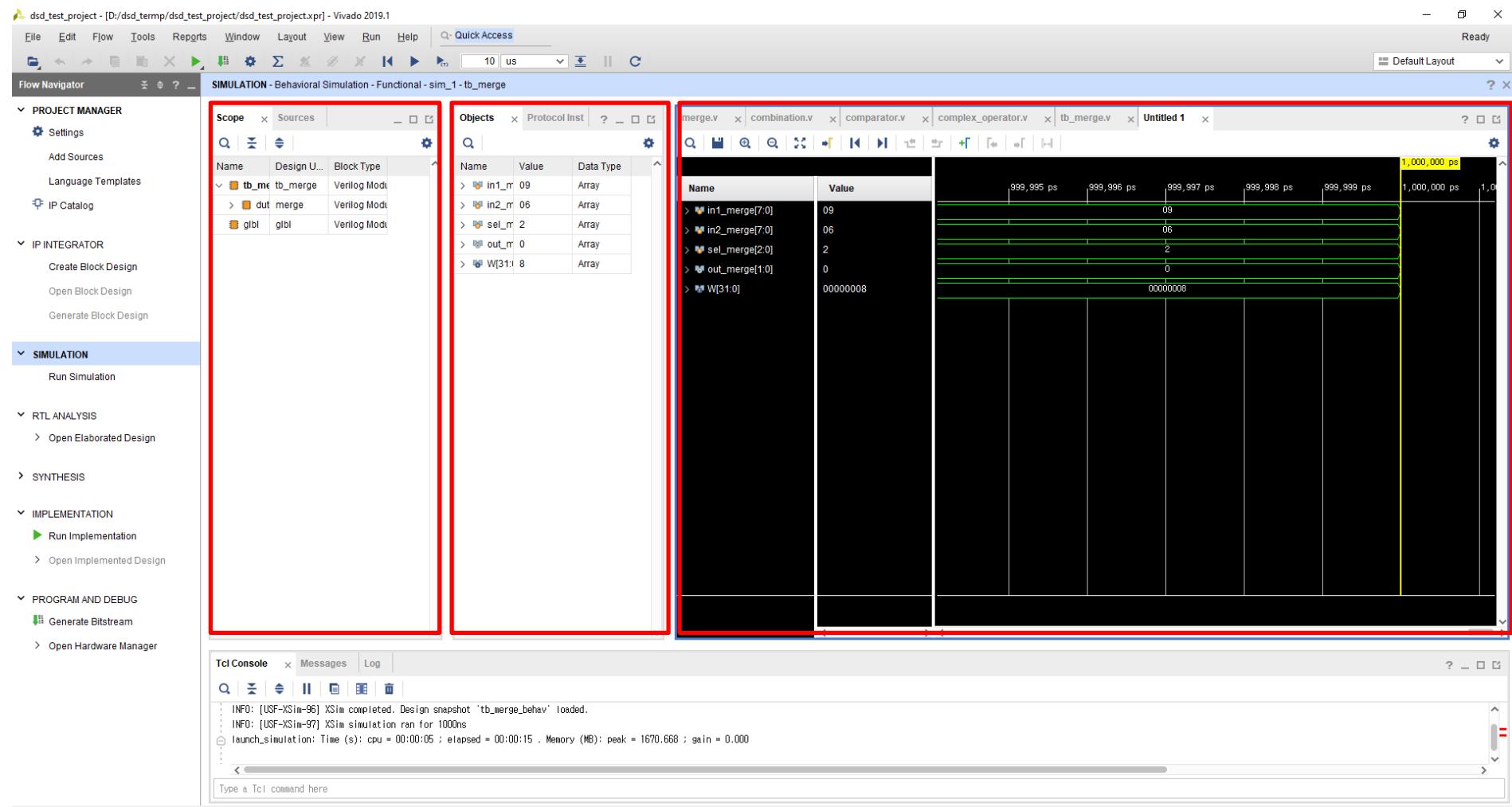
Run Simulation

Executing analysis and compilation step...

Background Cancel

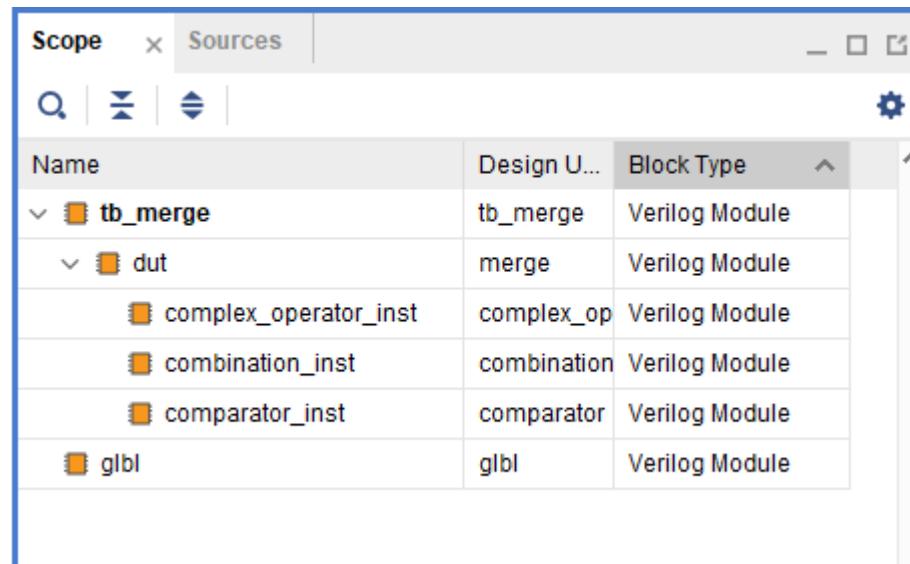
wait a minute...

Vivado – Simulation



Vivado – Simulation

- ◆ You can see the simulated module instances.



Vivado – Simulation

- ◆ You can see all the variables of selected instance through 'Objects' window.

The screenshot shows the Vivado Objects window. On the left, the tree view shows the hierarchy: tb_merge > dut > complex, combinational, comparison, and glbl. The 'dut' node is expanded, and an arrow points from the 'tb_merge' node to the expanded 'dut' node. On the right, a table lists variables for the 'dut' instance. A red box highlights the first five rows of the table. The columns are Name, Value, and Data Type.

Name	Value	Data Type
> in1_merge[7:0]	09	Array
> in2_merge[7:0]	06	Array
> sel_merge[2:0]	2	Array
> out_merge[1:0]	0	Array
> W[31:0]	8	Array

The screenshot shows the Vivado Objects window. On the left, the tree view shows the hierarchy: tb_merge > dut > complex, combinational, comparison, and glbl. The 'dut' node is expanded, and an arrow points from the 'tb_merge' node to the expanded 'dut' node. On the right, a table lists variables for the 'dut' instance. A red box highlights the first seven rows of the table. The columns are Name, Value, and Data Type.

Name	Value	Data Type
> in1[7:0]	09	Array
> in2[7:0]	06	Array
> sel[2:0]	2	Array
> dout[1:0]	0	Array
> out_compute[7:0]	0f	Array
> out_combination[7:0]	54	Array
> W[31:0]	8	Array

Vivado – Simulation

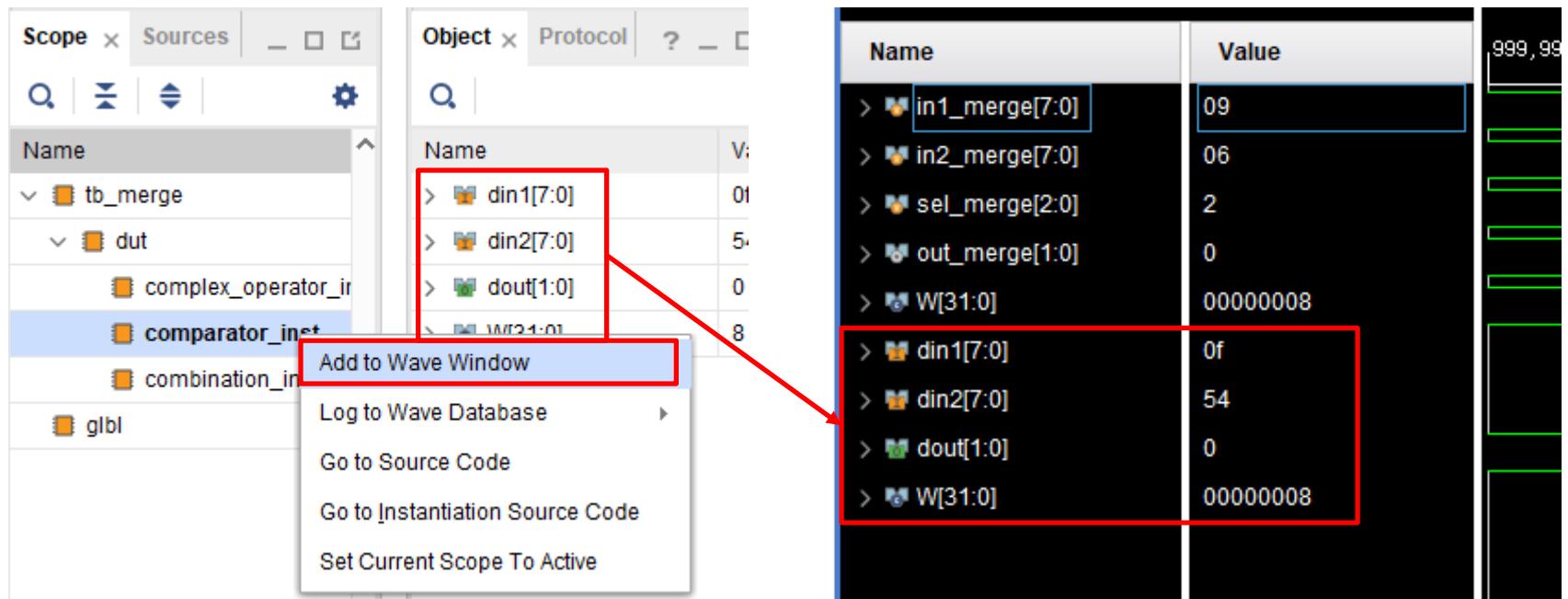
Name	Value	Data Type
> tb_merge		
dut		
complex_operator_inst		
comparator_inst		
combination_inst		
glbl		

Name	Value	Data Type
> tb_merge		
dut		
complex_operator_inst		
comparator_inst		
combination_inst		
glbl		

Name	Value	Data Type
> tb_merge		
dut		
complex_operator_inst		
comparator_inst		
combination_inst		
glbl		

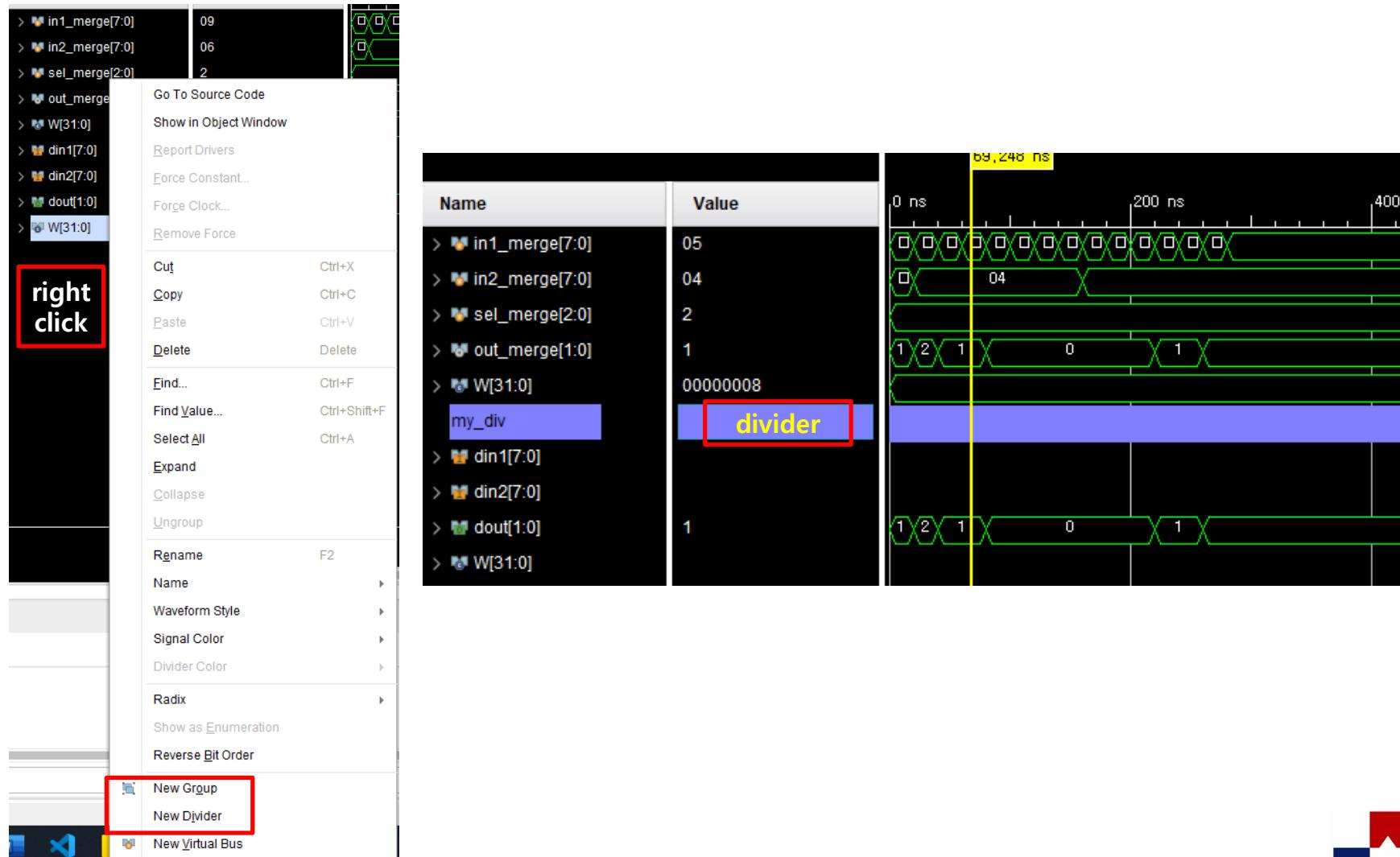
Vivado – Simulation

- ◆ You can add specific signals to wave window.

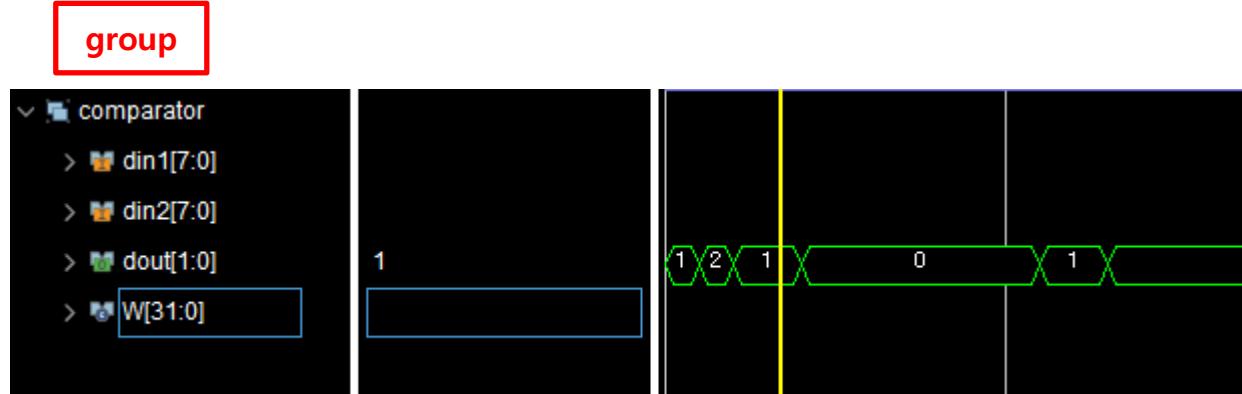


Vivado – Simulation

◆ You can add divider and group to the wave window.

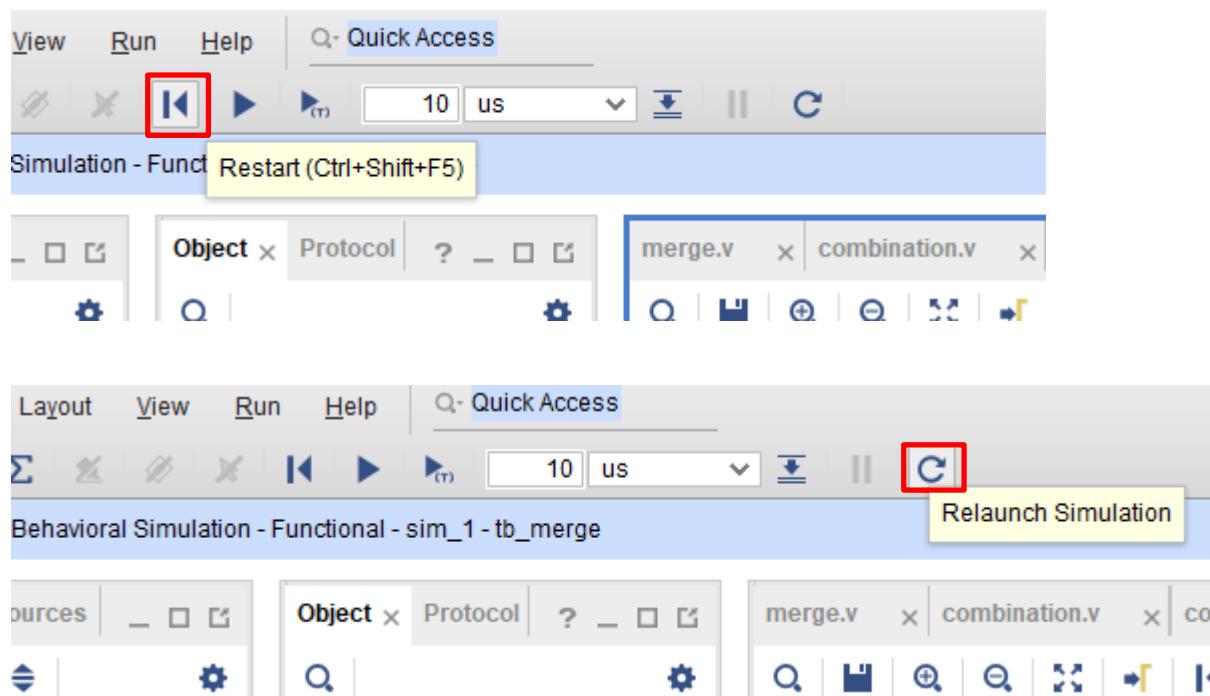


Vivado – Simulation



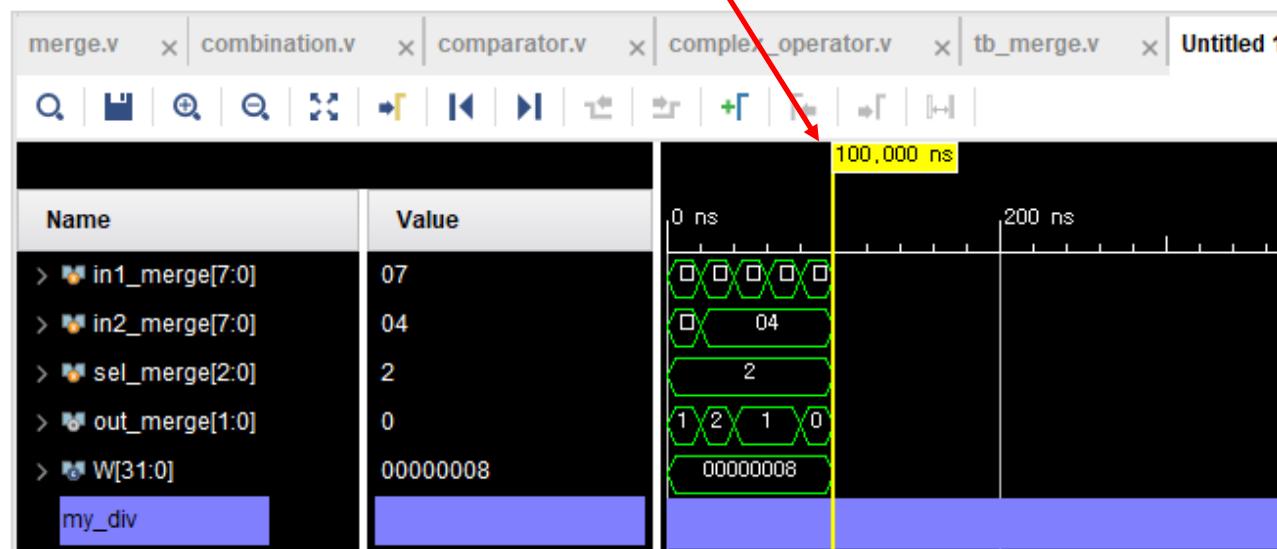
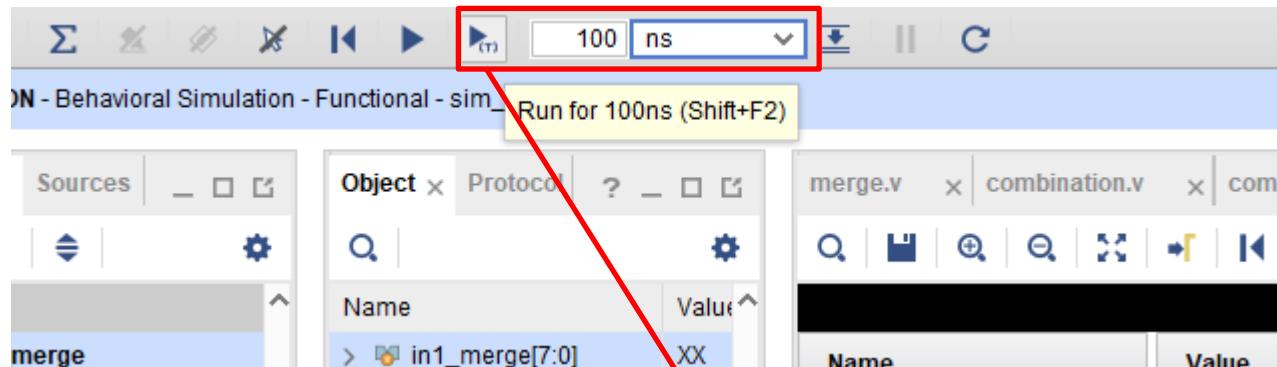
Vivado - Simulation

- ◆ Restart: restart simulation with the condition of last launched setting(changes in source files after launching simulation are not reflected)
- ◆ Relaunch Simulation: relaunch simulation(it takes more time than restarting simulation but changes in source files are reflected)

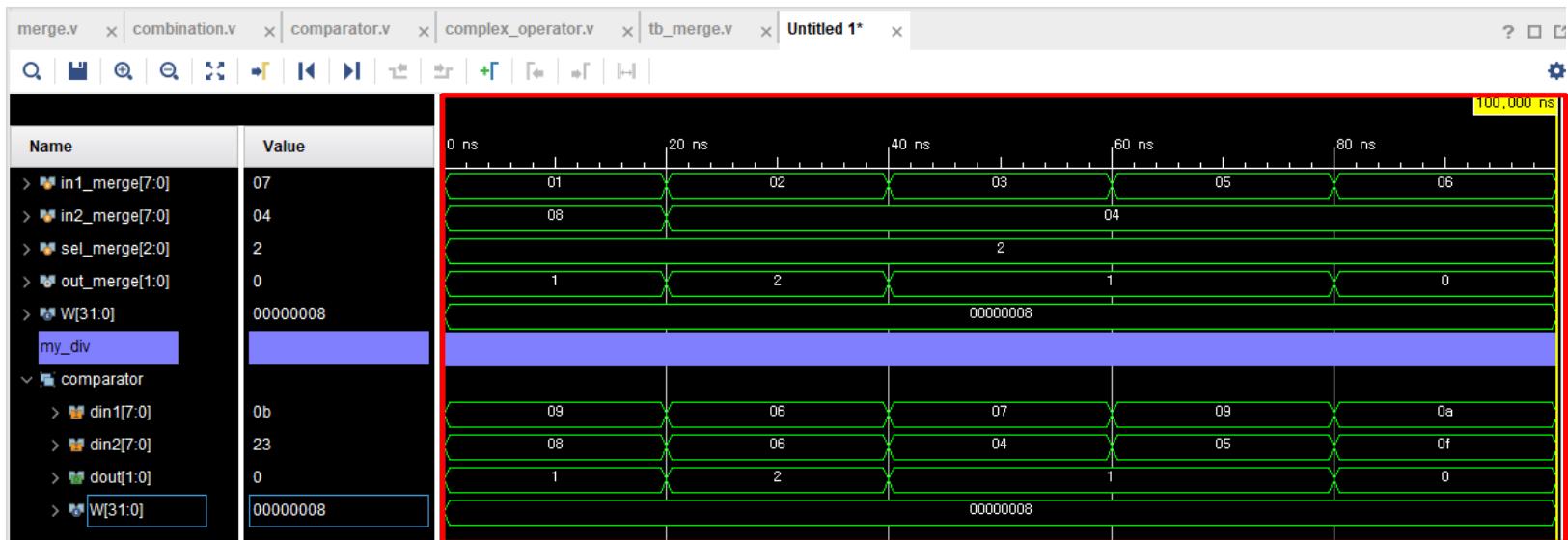
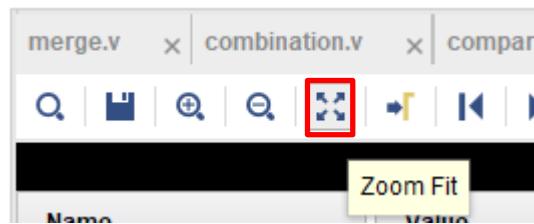


Vivado - Simulation

❖ df

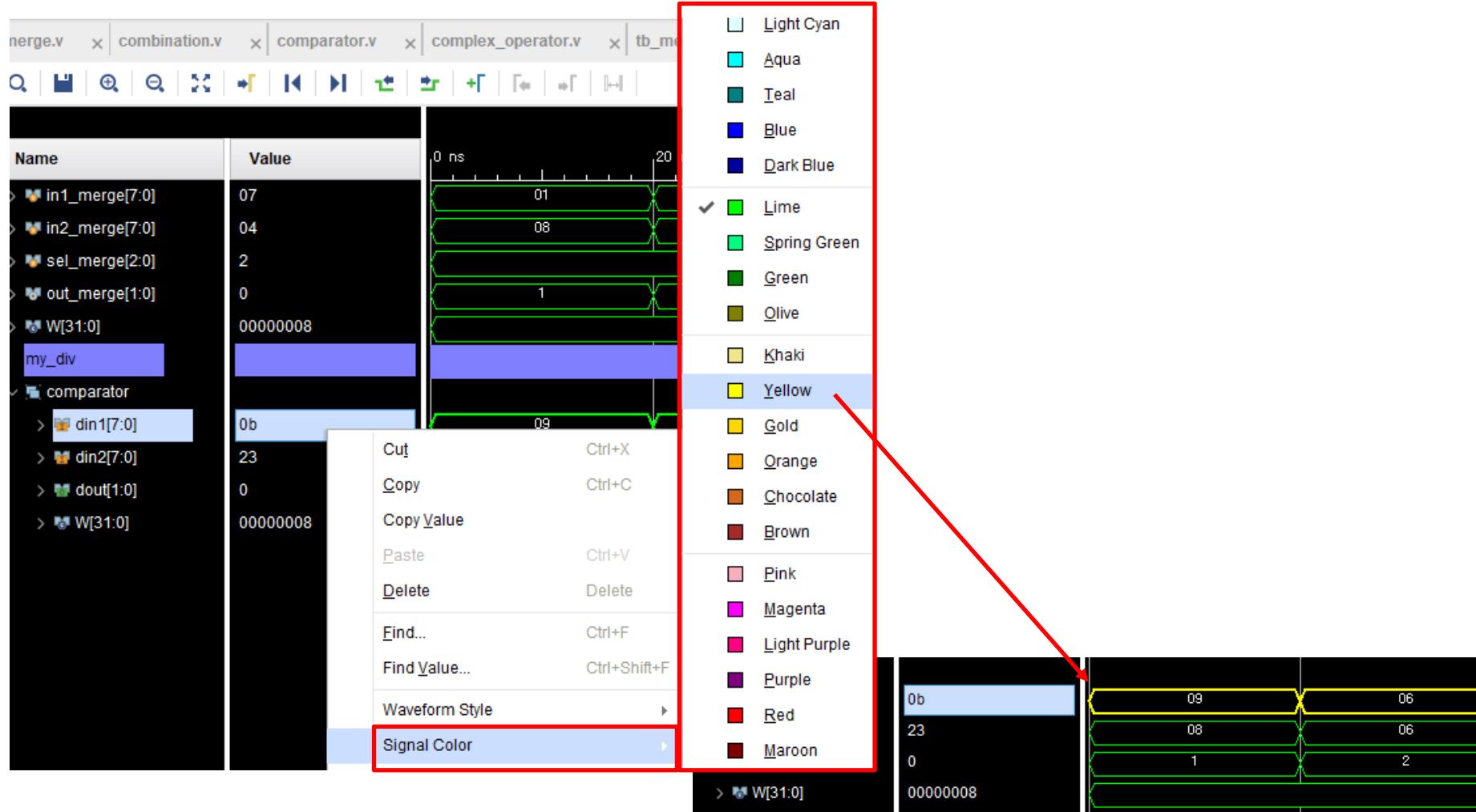


Vivado - Simulation



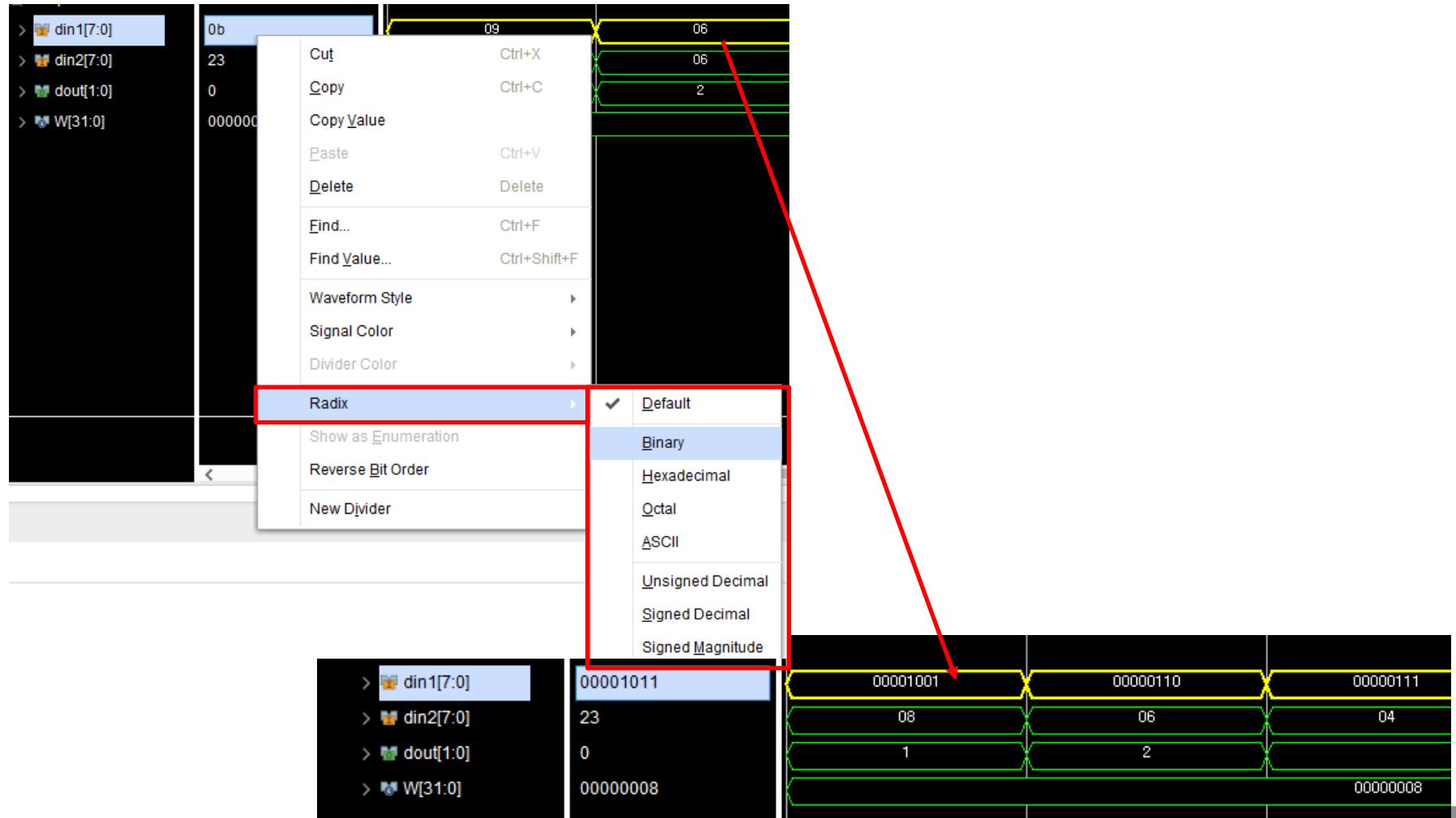
Vivado - Simulation

- ◆ You can change the color of signals



Vivado - Simulation

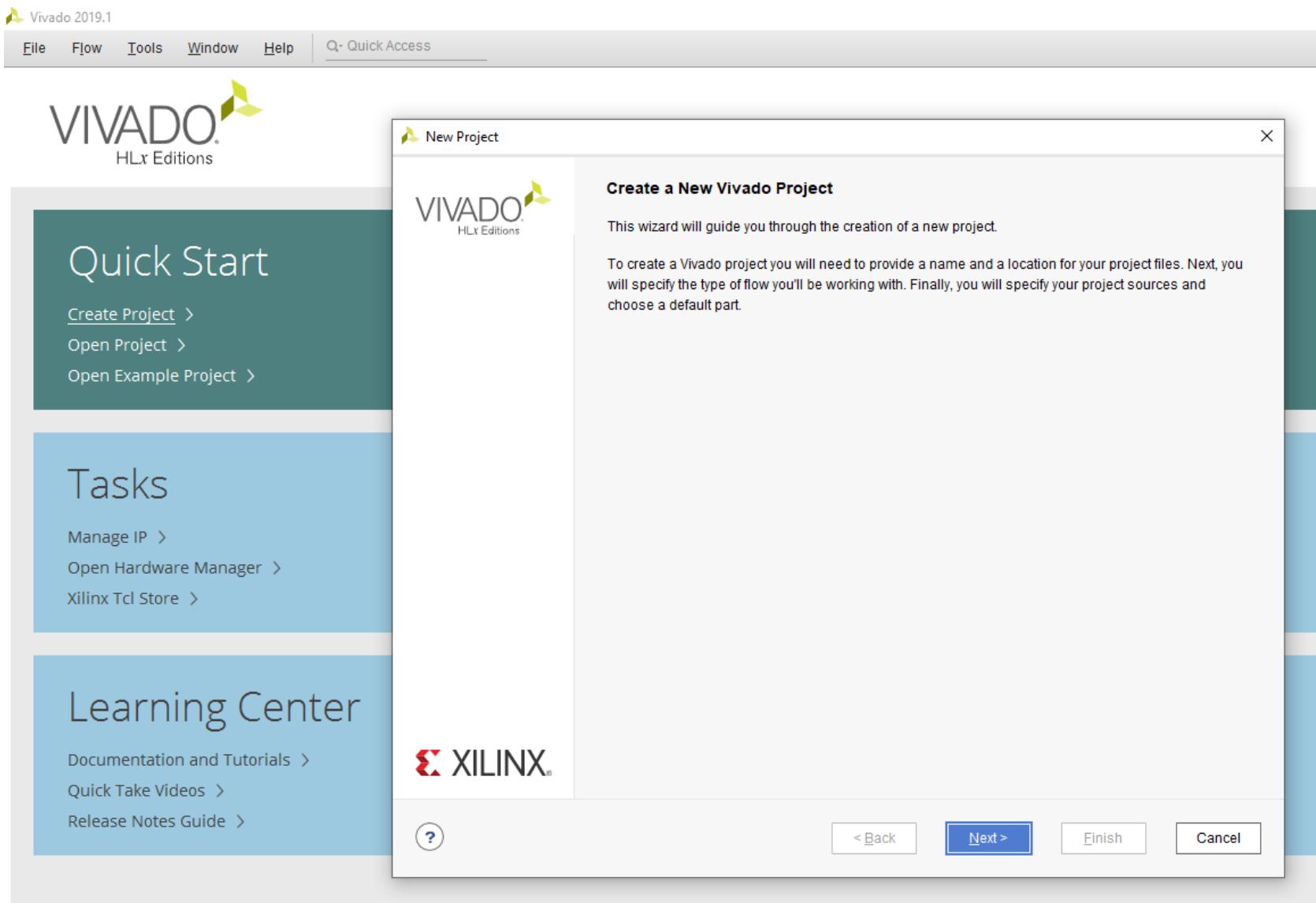
◆ You can also change the radix of signals.



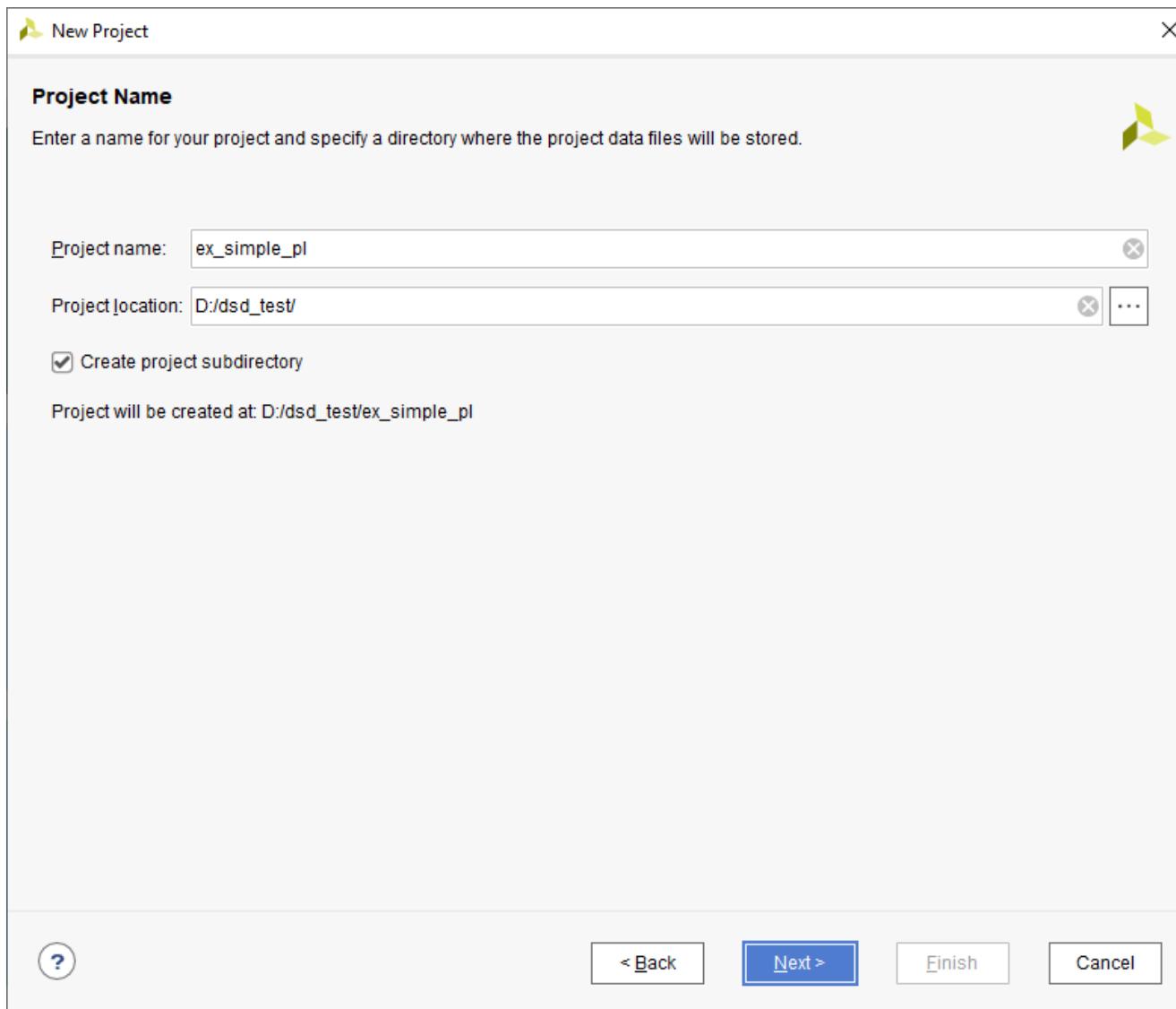
Vivado – Example: Simple PL Design

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ Vivado – Simulation
- ◆ **Vivado – Example: Simple PL Design**
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

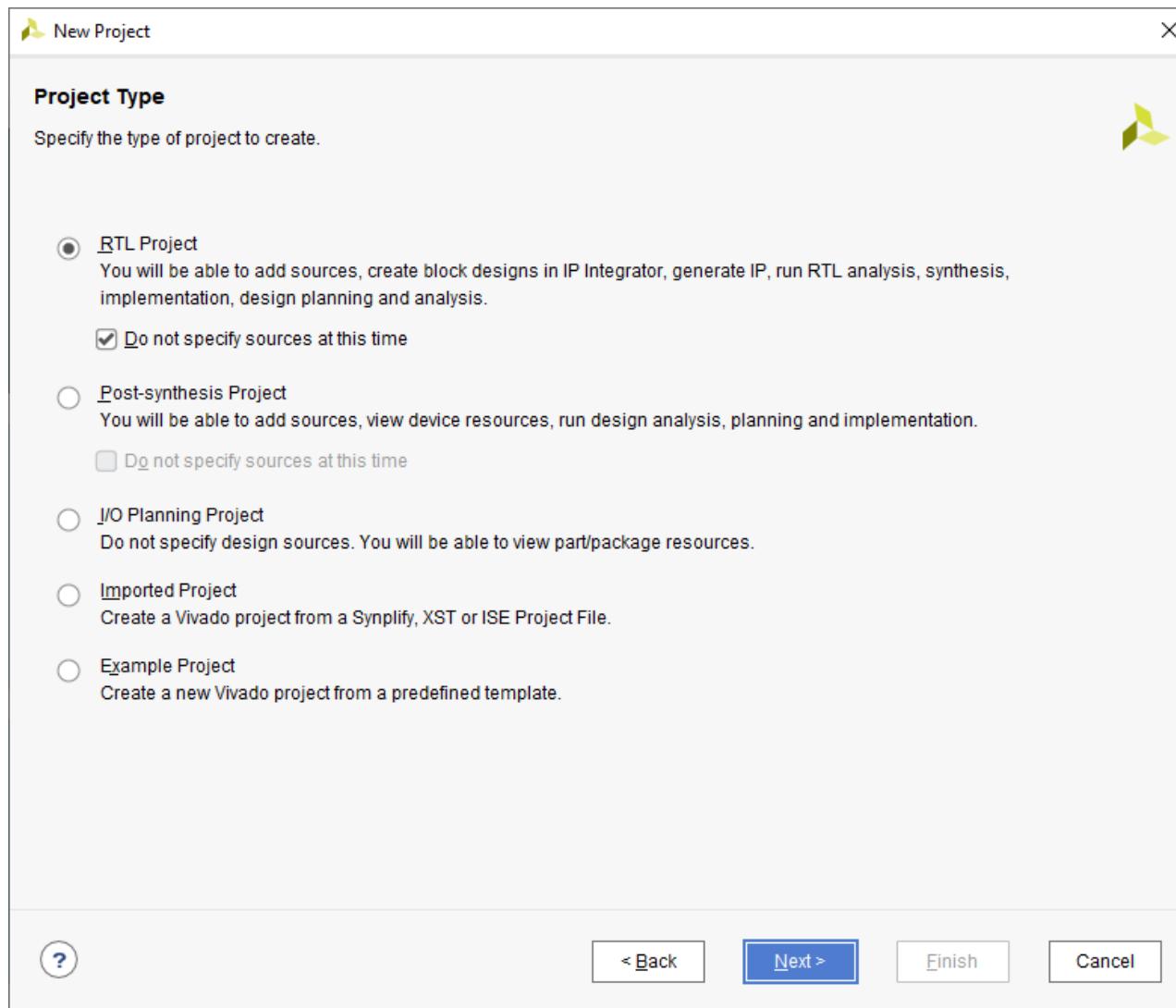
Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design



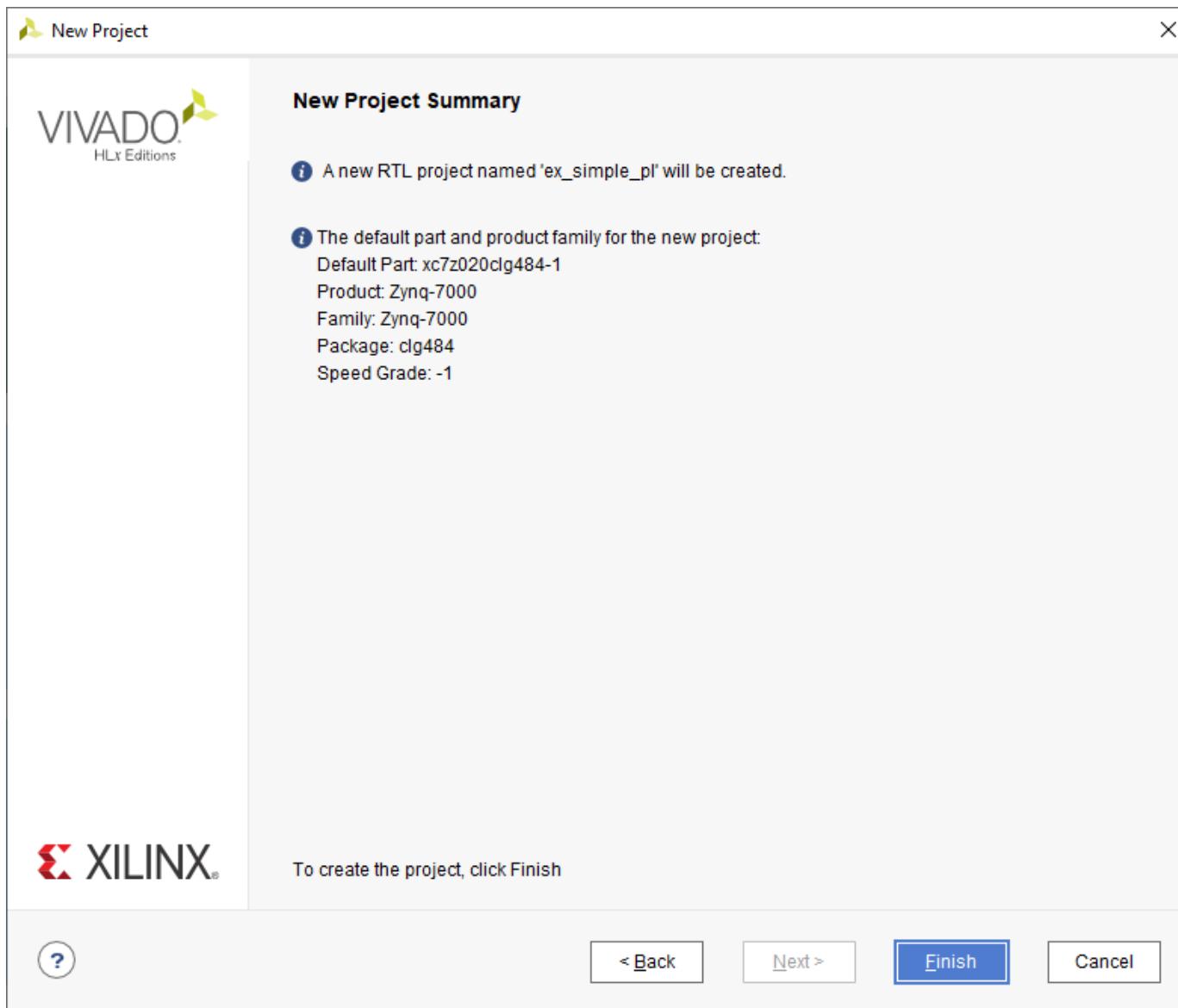
Example: Simple PL Design

The screenshot shows the 'Default Part' dialog in the Xilinx Project Manager. The title bar says 'New Project'. The main area is titled 'Default Part' with the sub-instruction 'Choose a default Xilinx part or board for your project.' Below this is a search bar with the query 'xc7z020clg484-1' and a result count '(1 match)'. A table displays the found part information:

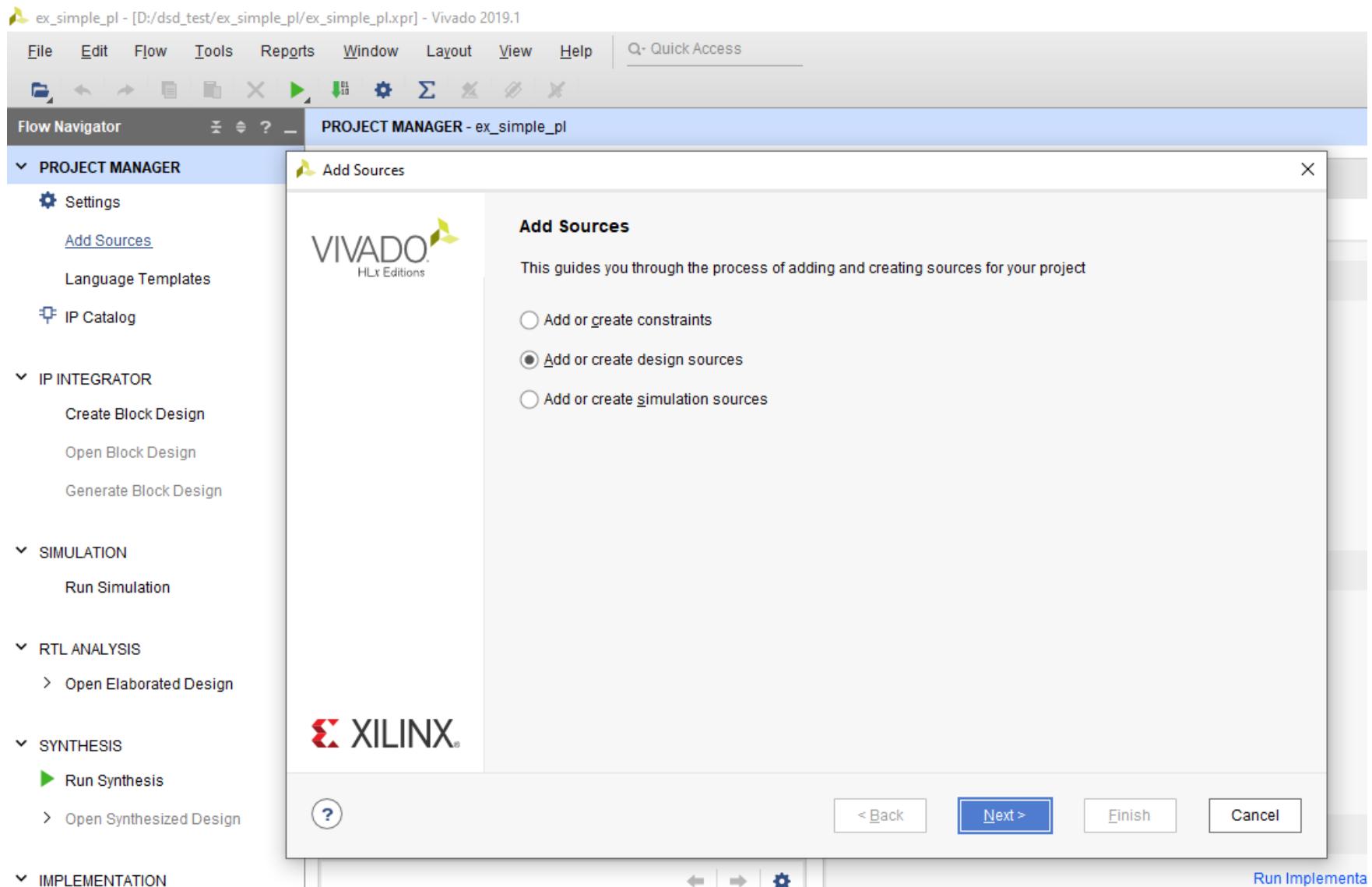
Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gt
xc7z020clg484-1	484	200	53200	106400	140	0	220	0

At the bottom are buttons for '?', '< Back' (disabled), 'Next >', 'Finish', and 'Cancel'.

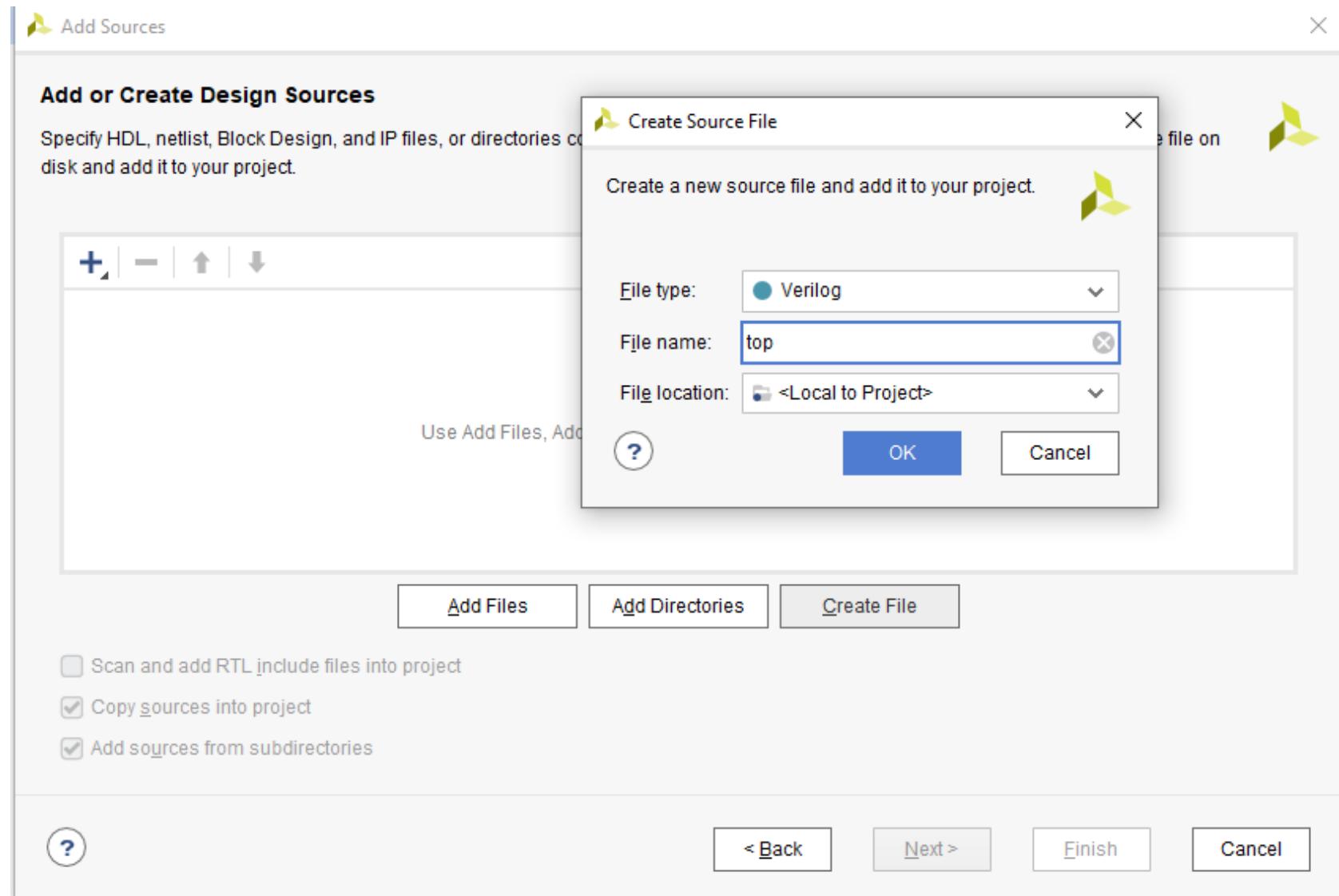
Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design

 Add Sources X

Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.



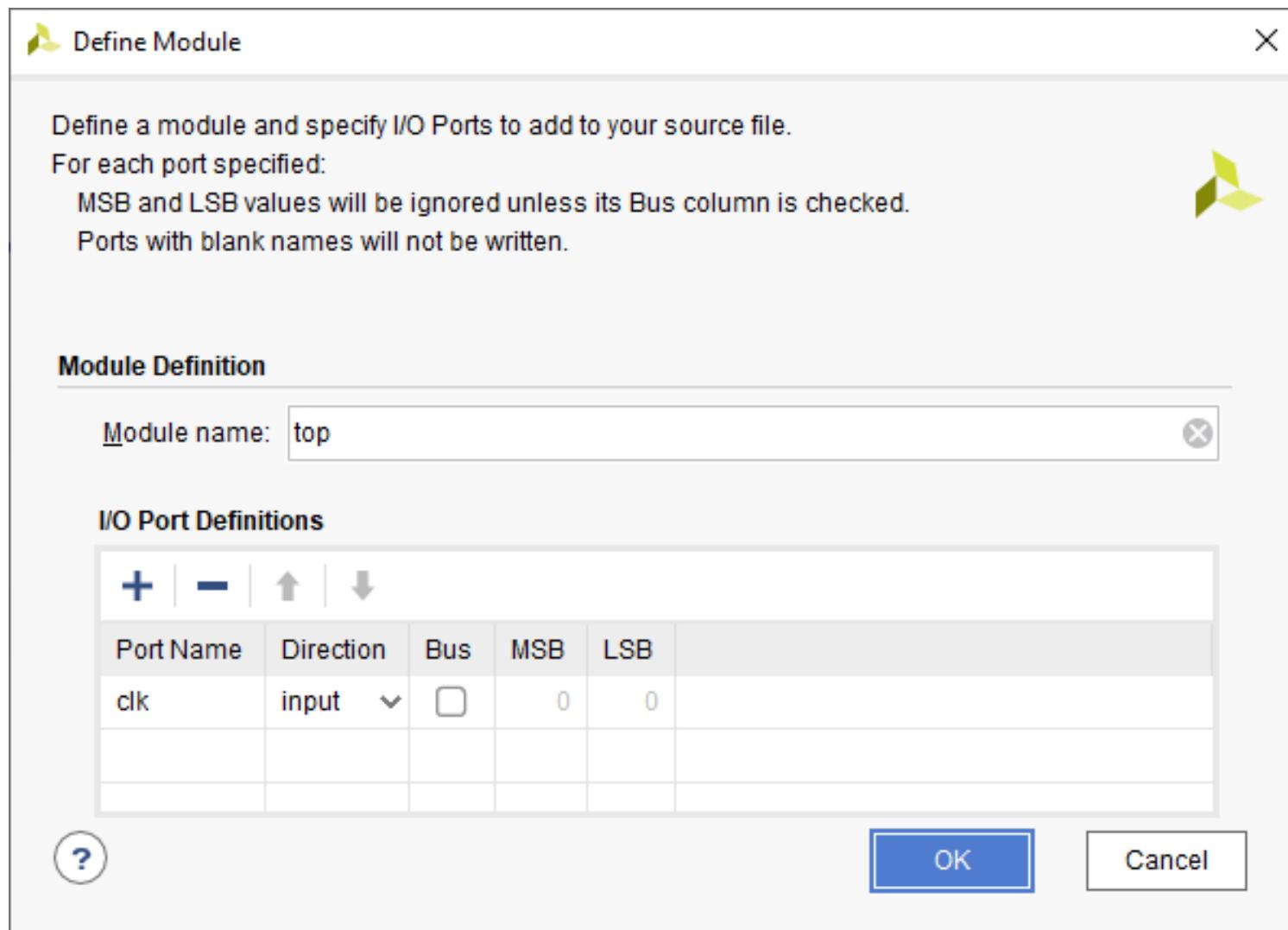
	Index	Name	Library	Location
●	1	top.v	xil_defaultlib	<Local to Project>

[Add Files](#) [Add Directories](#) [Create File](#)

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

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Example: Simple PL Design



Example: Simple PL Design

The screenshot shows a CAD software interface with two main windows. On the left is the 'Sources' browser window, which lists design sources, constraints, simulation sources, and utility sources. The 'Design Sources' section contains one item: 'top (top.v)', which is selected and highlighted with a blue border. On the right is the code editor window for the file 'top.v'. The code editor shows the following Verilog code:

```
1 `timescale 1ns / 1ps
2 // Company:
3 // Engineer:
4 //
5 // Create Date: 05/19/2021 09:08:07 PM
6 // Design Name:
7 // Module Name: top
8 // Project Name:
9 // Target Devices:
10 // Tool Versions:
11 // Description:
12 //
13 // Dependencies:
14 //
15 // Revision:
16 // Revision 0.01 - File Created
17 // Additional Comments:
18 //
19 //
20 //
21 //
22
23 module top(
24     input clk
25 );
26 endmodule
27
```

Example: Simple PL Design

```
1 `timescale 1ns / 1ps
2
3 module top(
4     input wire clk,
5     input wire rstn,
6     input wire [2:0] pushbutton,
7     input wire [7:0] dipswitch,
8     output reg [7:0] led
9 );
10
11 reg [2:0] pushbutton_q;
12 reg [2:0] pushbutton_q2;
13 reg [2:0] pushbutton_q3;
14
15 always @ (posedge clk) begin
16     pushbutton_q <= pushbutton;
17     pushbutton_q2 <= pushbutton_q;
18     pushbutton_q3 <= pushbutton_q2;
19 end
20
21 always @ (posedge clk) begin
22     if (!rstn) begin
23         led <= 0;
24     end
25     else begin
26         if ((pushbutton_q2[0]) && (~pushbutton_q3[0])) begin
27             led <= dipswitch;
28         end
29         else if ((pushbutton_q2[1]) && (~pushbutton_q3[1])) begin
30             led <= {led[0], led[7:1]};
31         end
32         else if ((pushbutton_q2[2]) && (~pushbutton_q3[2])) begin
33             led <= {led[6:0], led[7]};
34         end
35     end
36 end
37
38 endmodule
```

Example: Simple PL Design

```
`timescale 1ns / 1ps

module top(
    input wire clk,
    input wire rstn,
    input wire [2:0] pushbutton,
    input wire [7:0] dipswitch,
    output reg [7:0] led
);

reg [2:0] pushbutton_q;
reg [2:0] pushbutton_q2;
reg [2:0] pushbutton_q3;

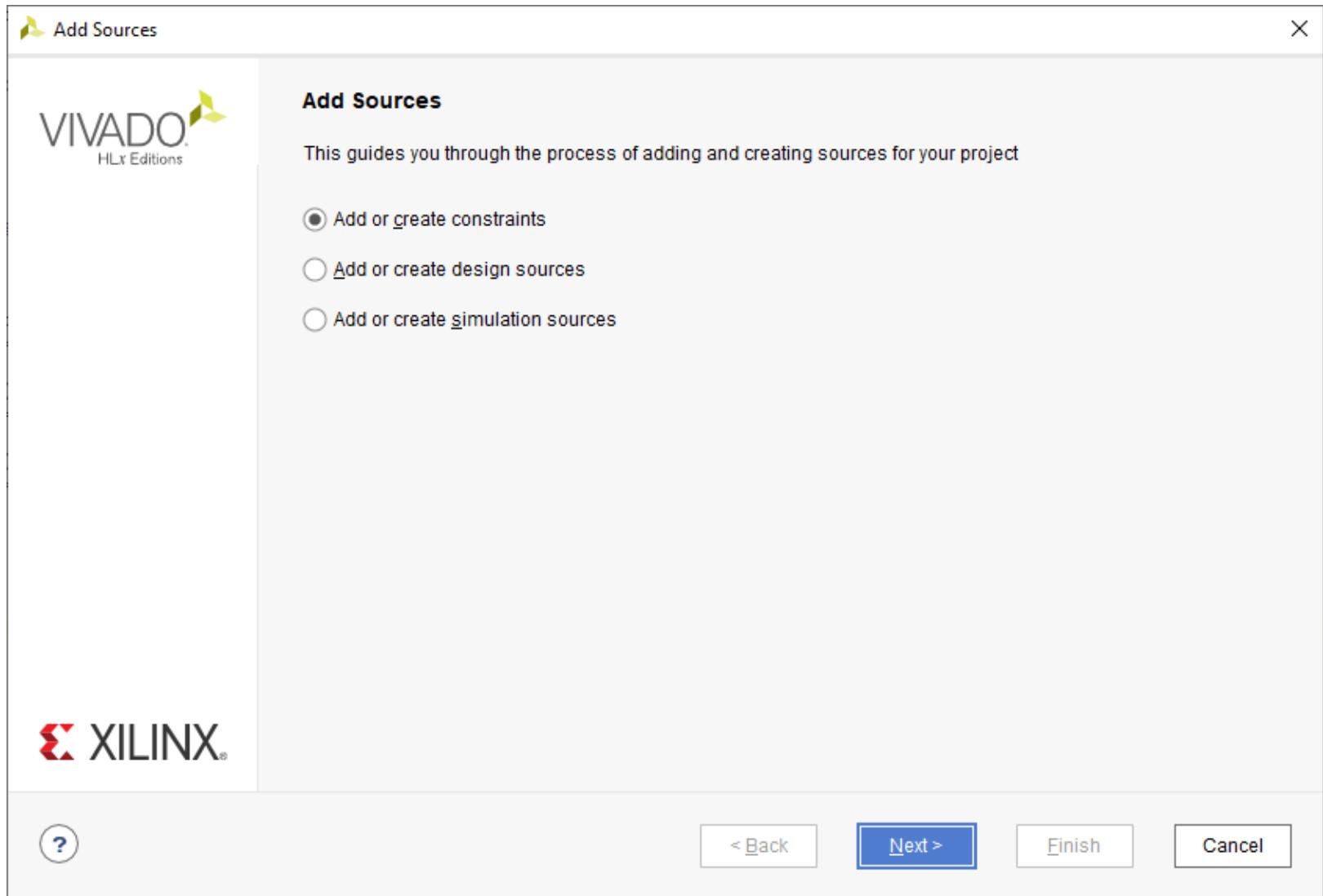
always @ (posedge clk) begin
    pushbutton_q <= pushbutton;
    pushbutton_q2 <= pushbutton_q;
    pushbutton_q3 <= pushbutton_q2;
end
```

Example: Simple PL Design

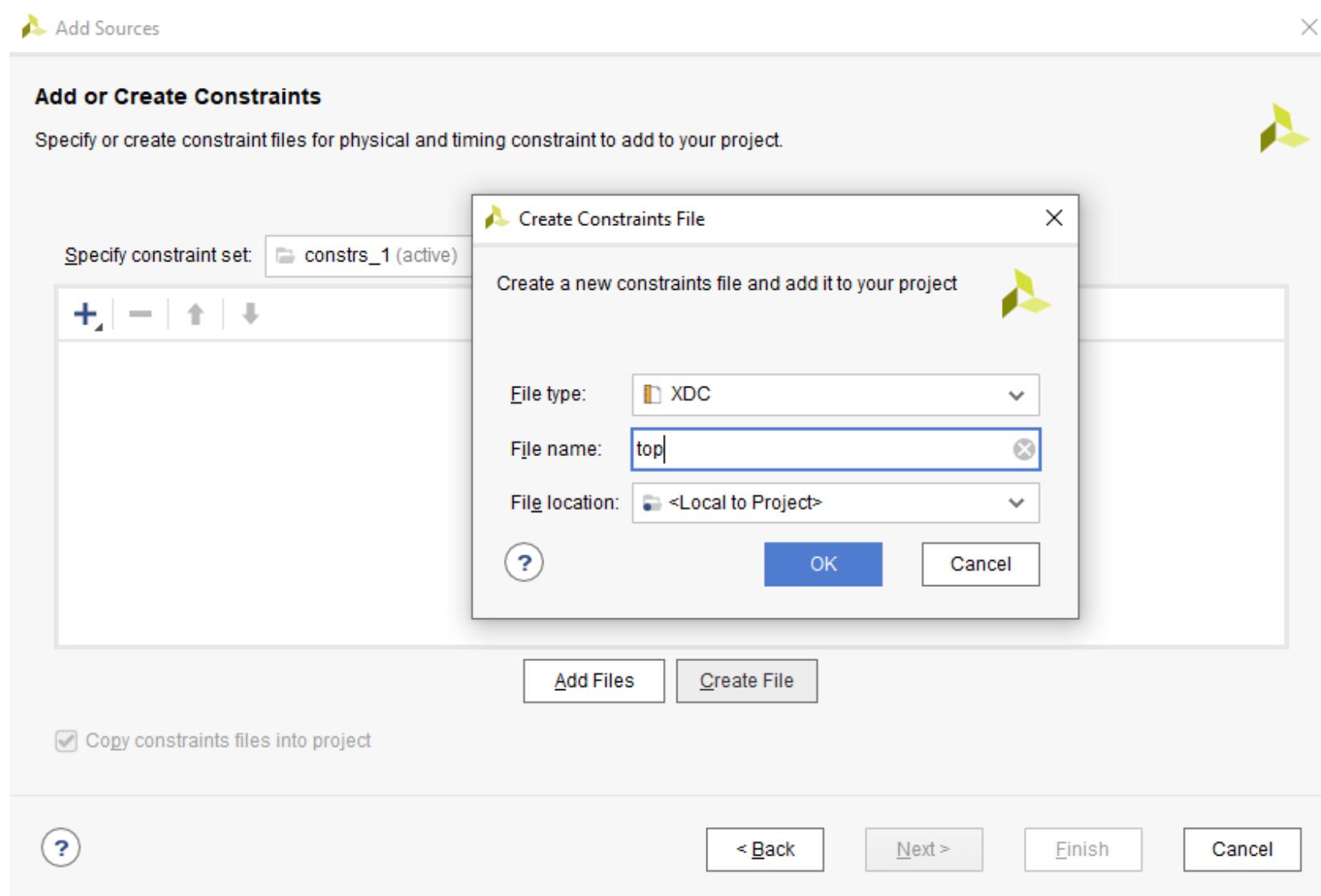
```
always @ (posedge clk) begin
    if (!rstn) begin
        led <= 0;
    end
    else begin
        if ((pushbutton_q2[0]) && (~pushbutton_q3[0])) begin
            led <= dipswitch;
        end
        else if ((pushbutton_q2[1]) && (~pushbutton_q3[1])) begin
            led <= {led[0], led[7:1]};
        end
        else if ((pushbutton_q2[2]) && (~pushbutton_q3[2])) begin
            led <= {led[6:0], led[7]};
        end
    end
end

endmodule
```

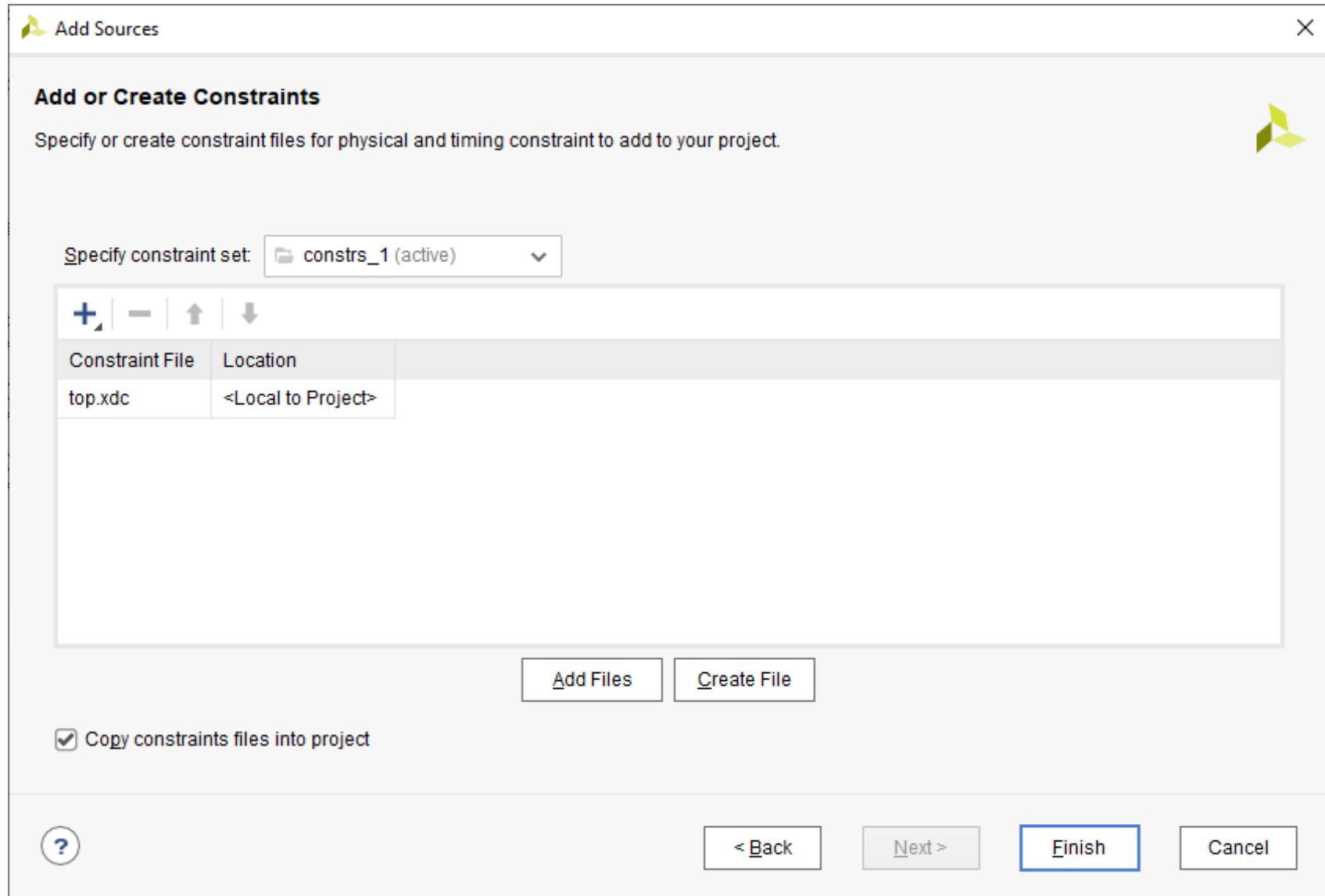
Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design

The screenshot shows a software interface for a digital design project. On the left, there is a 'Sources' panel with a tree view of files:

- Design Sources (1)
 - top (top.v)
- Constraints (1)
 - constrs_1 (1)
 - top.xdc

The main area is titled 'Project Summary' and shows three tabs: 'top.v *', 'top.xdc *', and the currently active tab 'top.xdc *'. The path 'D:/dsd_test/ex_simple_pl/ex_simple_pl.srccs/constrs_1/new/top.xdc' is displayed below the tabs.

The content of the 'top.xdc' file is as follows:

```
1 set_property IOSTANDARD "LVCMOS33" [get_ports "clk"]
2 set_property PACKAGE_PIN "M19" [get_ports "clk"]
3 create_clock -add -name clk_pl -period 40.00 -waveform {0 12.5} [get_ports "clk"]
4
5 set_property IOSTANDARD "LVCMOS33" [get_ports "rstn"]
6 set_property PACKAGE_PIN "Y18" [get_ports "rstn"]
7
8 set_property IOSTANDARD "LVCMOS33" [get_ports "pushbutton[*]"]
9 set_property PACKAGE_PIN "AA18" [get_ports "pushbutton[0]"]
10 set_property PACKAGE_PIN "Y19" [get_ports "pushbutton[1]"]
11 set_property PACKAGE_PIN "AA19" [get_ports "pushbutton[2]"]
12
13 set_property IOSTANDARD "LVCMOS33" [get_ports "dipswitch[*]"]
14 set_property PACKAGE_PIN "Y20" [get_ports "dipswitch[0]"]
15 set_property PACKAGE_PIN "Y21" [get_ports "dipswitch[1]"]
16 set_property PACKAGE_PIN "AB19" [get_ports "dipswitch[2]"]
17 set_property PACKAGE_PIN "AB20" [get_ports "dipswitch[3]"]
18 set_property PACKAGE_PIN "AA22" [get_ports "dipswitch[4]"]
19 set_property PACKAGE_PIN "AB22" [get_ports "dipswitch[5]"]
20 set_property PACKAGE_PIN "AA21" [get_ports "dipswitch[6]"]
21 set_property PACKAGE_PIN "AB21" [get_ports "dipswitch[7]"]
22
23 set_property IOSTANDARD "LVCMOS33" [get_ports "led[*]"]
24 set_property PACKAGE_PIN "T16" [get_ports "led[0]"]
25 set_property PACKAGE_PIN "T17" [get_ports "led[1]"]
26 set_property PACKAGE_PIN "R19" [get_ports "led[2]"]
27 set_property PACKAGE_PIN "T19" [get_ports "led[3]"]
28 set_property PACKAGE_PIN "R18" [get_ports "led[4]"]
29 set_property PACKAGE_PIN "T18" [get_ports "led[5]"]
30 set_property PACKAGE_PIN "P16" [get_ports "led[6]"]
31 set_property PACKAGE_PIN "R16" [get_ports "led[7]"]
```

Example: Simple PL Design

```
set_property IOSTANDARD "LVCMOS33" [get_ports "clk"]
set_property PACKAGE_PIN "M19" [get_ports "clk"]
create_clock -add -name clk_pl -period 40.00 -waveform {0 12.5} [get_ports "clk"]

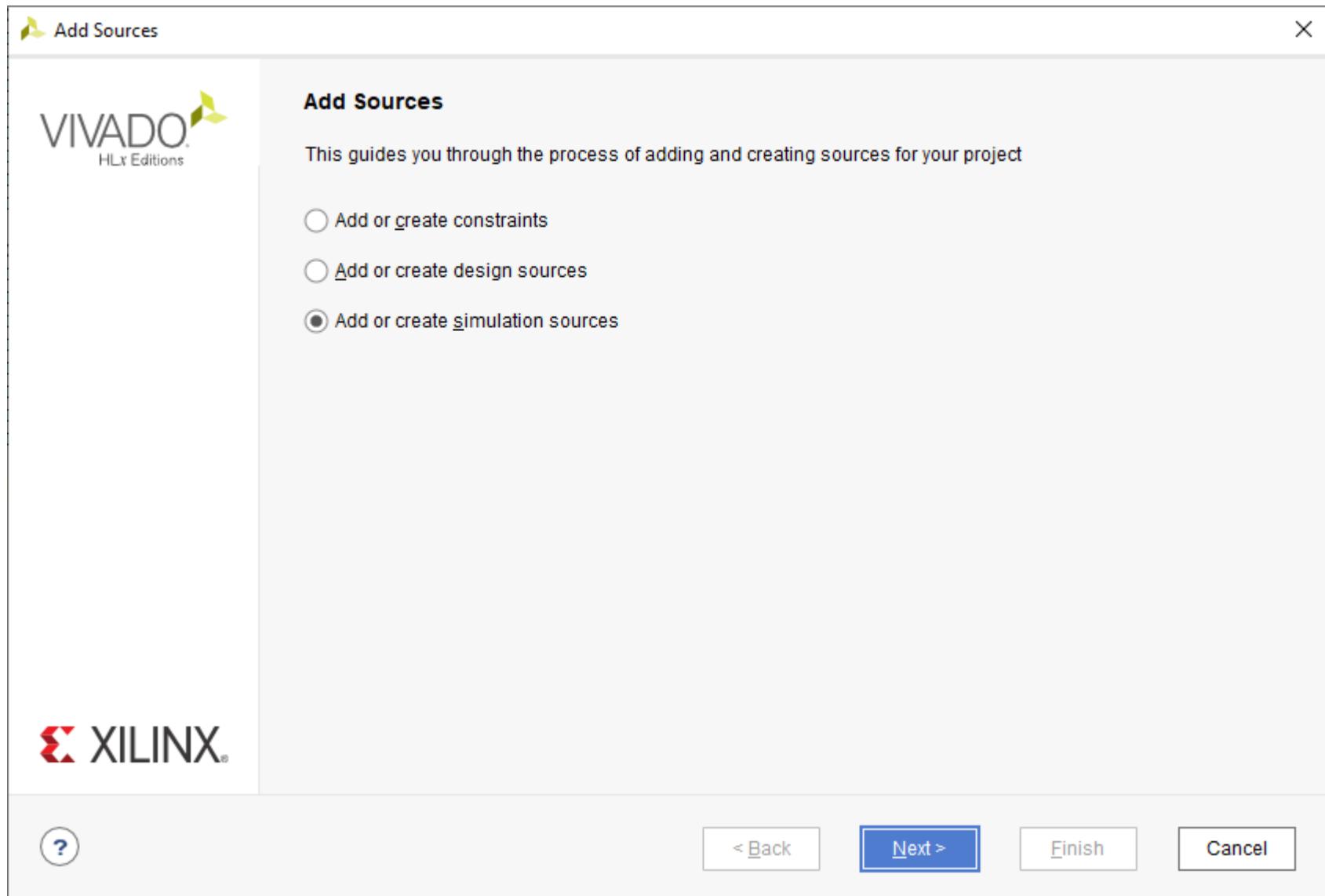
set_property IOSTANDARD "LVCMOS33" [get_ports "rstn"]
set_property PACKAGE_PIN "Y18" [get_ports "rstn"]

set_property IOSTANDARD "LVCMOS33" [get_ports "pushbutton[*]"]
set_property PACKAGE_PIN "AA18" [get_ports "pushbutton[0]"]
set_property PACKAGE_PIN "Y19" [get_ports "pushbutton[1]"]
set_property PACKAGE_PIN "AA19" [get_ports "pushbutton[2]"]

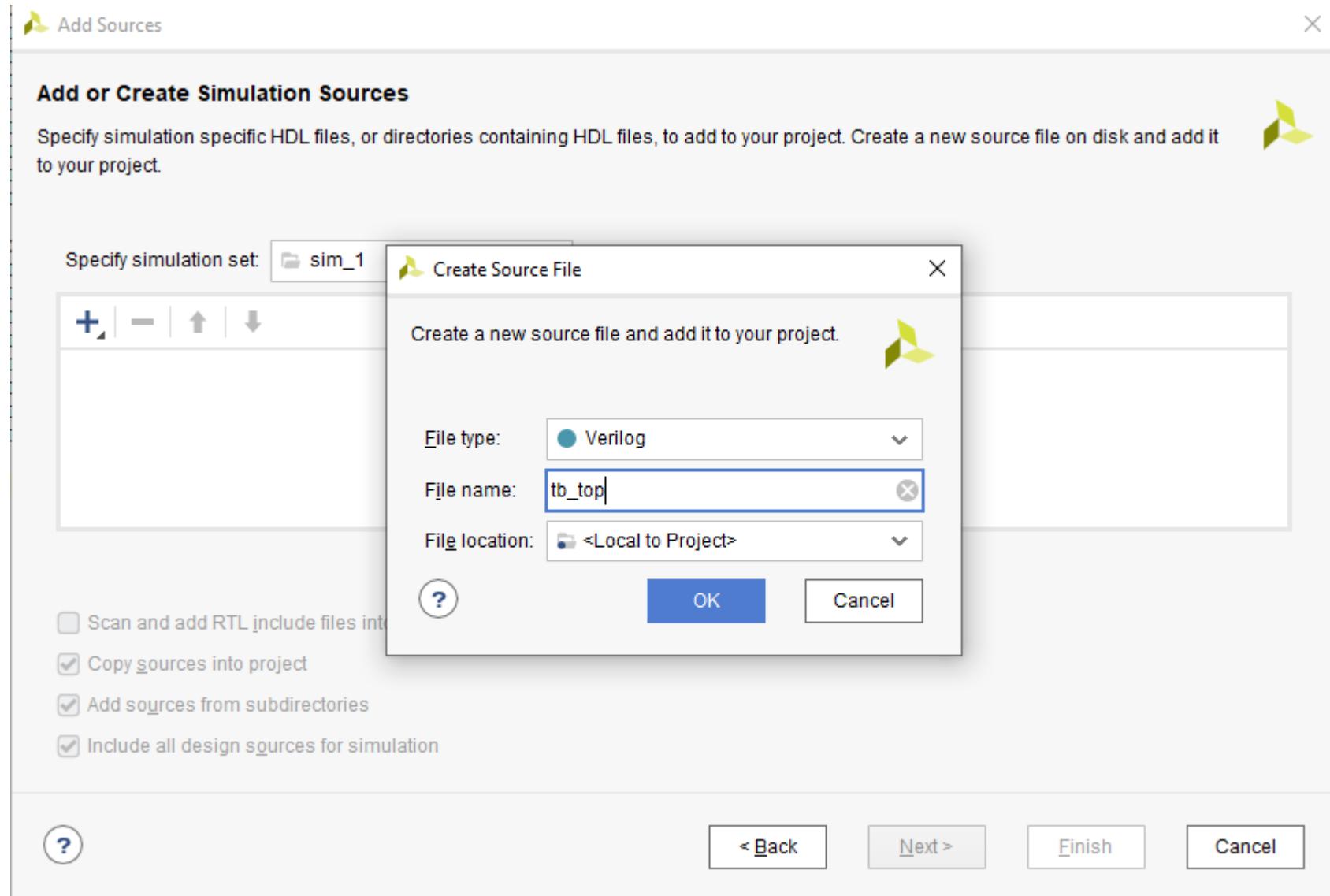
set_property IOSTANDARD "LVCMOS33" [get_ports "dipswitch[*]"]
set_property PACKAGE_PIN "Y20" [get_ports "dipswitch[0]"]
set_property PACKAGE_PIN "Y21" [get_ports "dipswitch[1]"]
set_property PACKAGE_PIN "AB19" [get_ports "dipswitch[2]"]
set_property PACKAGE_PIN "AB20" [get_ports "dipswitch[3]"]
set_property PACKAGE_PIN "AA22" [get_ports "dipswitch[4]"]
set_property PACKAGE_PIN "AB22" [get_ports "dipswitch[5]"]
set_property PACKAGE_PIN "AA21" [get_ports "dipswitch[6]"]
set_property PACKAGE_PIN "AB21" [get_ports "dipswitch[7]"]

set_property IOSTANDARD "LVCMOS33" [get_ports "led[*]"]
set_property PACKAGE_PIN "T16" [get_ports "led[0]"]
set_property PACKAGE_PIN "T17" [get_ports "led[1]"]
set_property PACKAGE_PIN "R19" [get_ports "led[2]"]
set_property PACKAGE_PIN "T19" [get_ports "led[3]"]
set_property PACKAGE_PIN "R18" [get_ports "led[4]"]
set_property PACKAGE_PIN "T18" [get_ports "led[5]"]
set_property PACKAGE_PIN "P16" [get_ports "led[6]"]
set_property PACKAGE_PIN "R16" [get_ports "led[7]"]
```

Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design

 Add Sources X

Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.



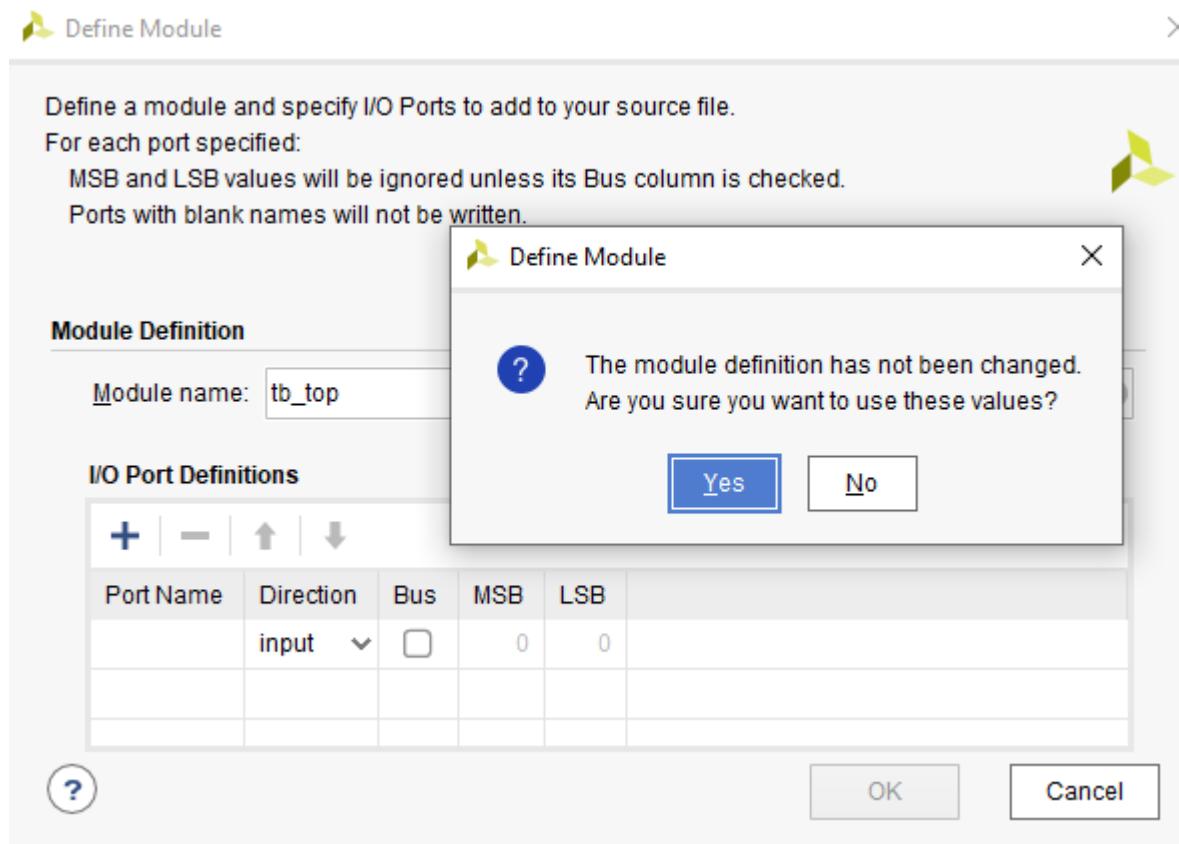
Specify simulation set:

+	Index	Name	Library	Location
	1	tb_top.v	xil_defaultlib	<Local to Project>

Scan and add RTL include files into project
 Copy sources into project
 Add sources from subdirectories
 Include all design sources for simulation

 [Back](#) [Next >](#) Finish [Cancel](#)

Example: Simple PL Design



Example: Simple PL Design

The screenshot shows a CAD tool interface with two main panes. The left pane is titled 'Sources' and displays a tree view of project files:

- Design Sources (1)
 - top (top.v)
- Constraints (1)
 - constrs_1 (1)
 - top.xdc
- Simulation Sources (2)
 - sim_1 (2)
 - top (top.v)
 - tb_top (tb_top.v) [selected]
- Utility Sources

The bottom of the Sources pane has tabs for Hierarchy, Libraries, and Compile Order.

The right pane is titled 'Project Summary' and shows the contents of the tb_top.v file:

```
D:/dsd_test/ex_simple_pl/ex_simple_pl.srcts/sim_1/new/tb_top.v

`timescale 1ns / 1ps
// Company:
// Engineer:
//
// Create Date: 05/19/2021 09:17:10 PM
// Design Name:
// Module Name: tb_top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
module tb_top(
);
endmodule
```

Example: Simple PL Design

```
1 `timescale 1ns / 1ps
2
3 module tb_top();
4
5 reg clk, rstn;
6 reg [2:0] pushbutton;
7 reg [7:0] dipswitch;
8 wire [7:0] led;
9
10 top dut(
11     .clk(clk),
12     .rstn(rstn),
13     .pushbutton(pushbutton),
14     .dipswitch(dipswitch),
15     .led(led)
16 );
17
18 initial begin
19     clk = 0;
20     forever
21         #5 clk = ~clk;
22 end
23
24 initial begin
25     rstn = 1;
26     #200     rstn = 0;
27     #100     rstn = 1;
28 end
29
30 initial begin
31     // test pushbutton[0] after system reset
32     dipswitch = 8'b0000_1111;
33     pushbutton = 3'b000;
34     #400    pushbutton = 3'b001;
35     #50     pushbutton = 3'b000;
36     #50     dipswitch = 8'b1111_1110;
37     #50     pushbutton = 3'b001;
38     #50     pushbutton = 3'b000;
39     // test pushbutton[1]
40     #50     pushbutton = 3'b010;
41     #50     pushbutton = 3'b000;
42     #100    pushbutton = 3'b100;
43     #50     pushbutton = 3'b000;
44     #100    $stop;
45 end
46
47 endmodule
```

Example: Simple PL Design

```
'timescale 1ns / 1ps

module tb_top();
reg clk, rstn;
reg [2:0] pushbutton;
reg [7:0] dipswitch;
wire [7:0] led;

top dut(
    .clk(clk),
    .rstn(rstn),
    .pushbutton(pushbutton),
    .dipswitch(dipswitch),
    .led(led)
);

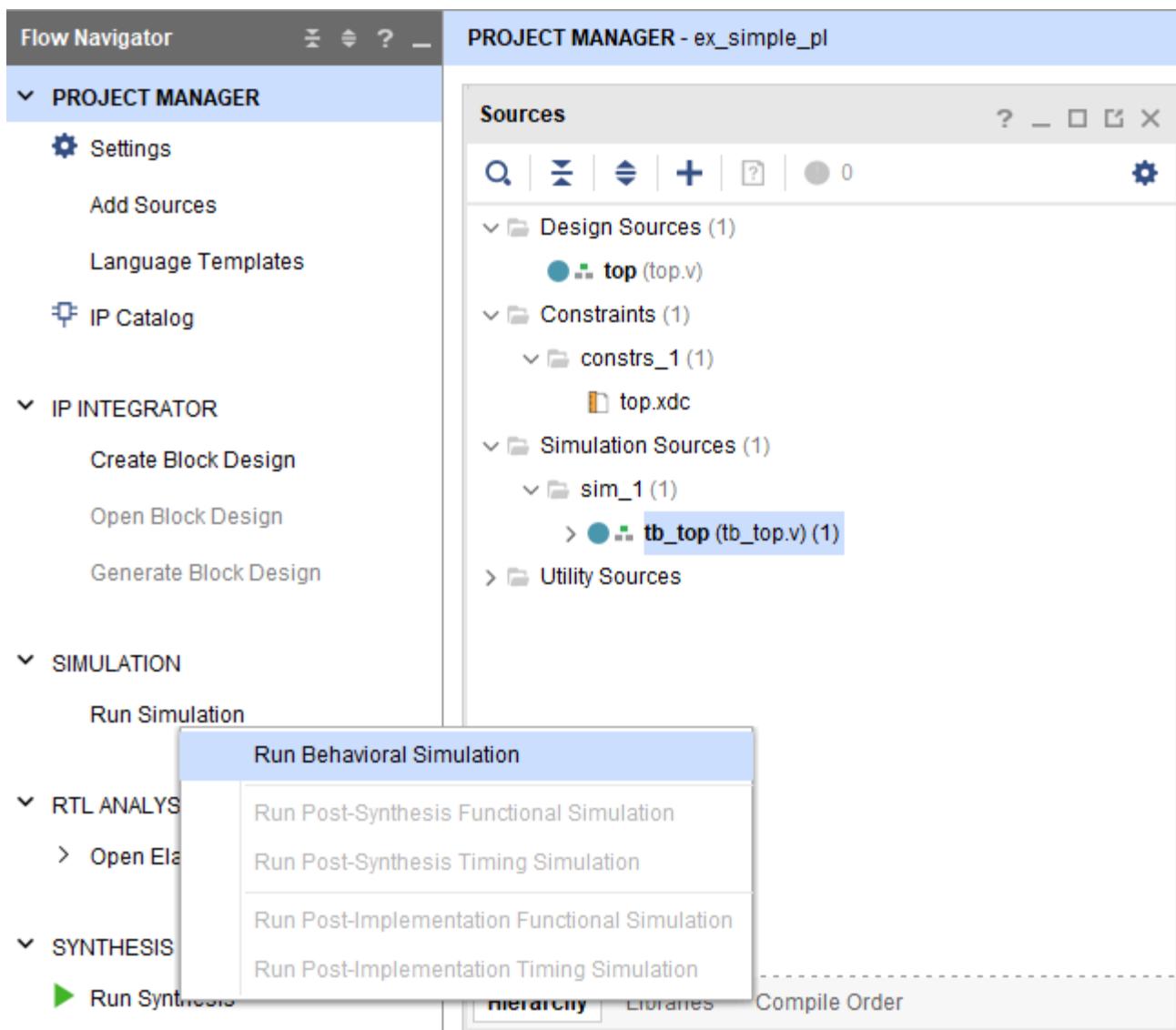
initial begin
    clk = 0;
    forever
        #5 clk = ~clk;
end

initial begin
    rstn = 1;
    #200    rstn = 0;
    #100    rstn = 1;
end
```

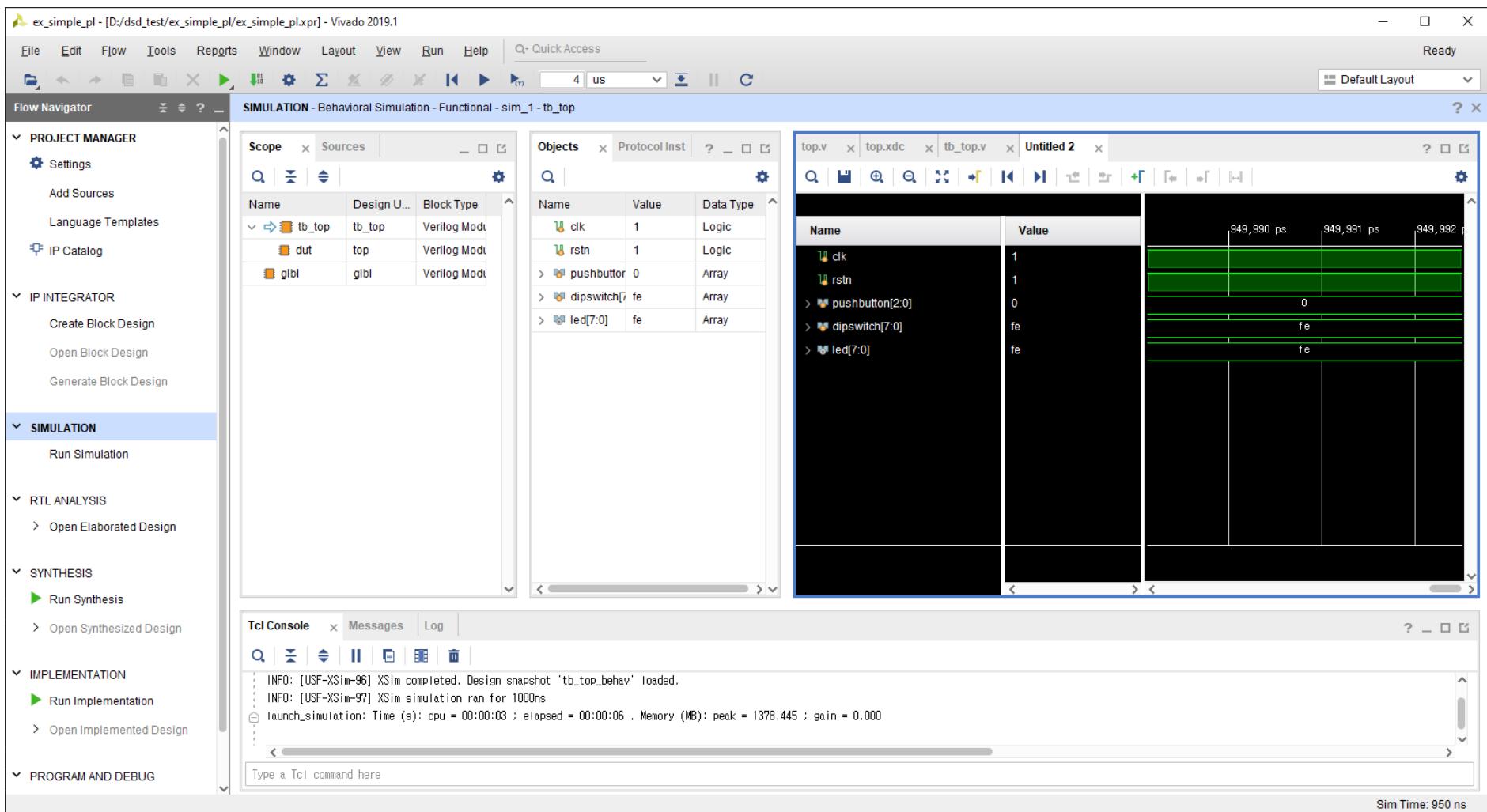
```
initial begin
    // test pushbutton[0] after system reset
    dipswitch = 8'b0000_1111;
    pushbutton = 3'b000;
    #400    pushbutton = 3'b001;
    #50     pushbutton = 3'b000;
    #50     dipswitch = 8'b1111_1110;
    #50     pushbutton = 3'b001;
    #50     pushbutton = 3'b000;
    // test pushbutton[1]
    #50     pushbutton = 3'b010;
    #50     pushbutton = 3'b000;
    #100    pushbutton = 3'b100;
    #50     pushbutton = 3'b000;
    #100    $stop;
end

endmodule
```

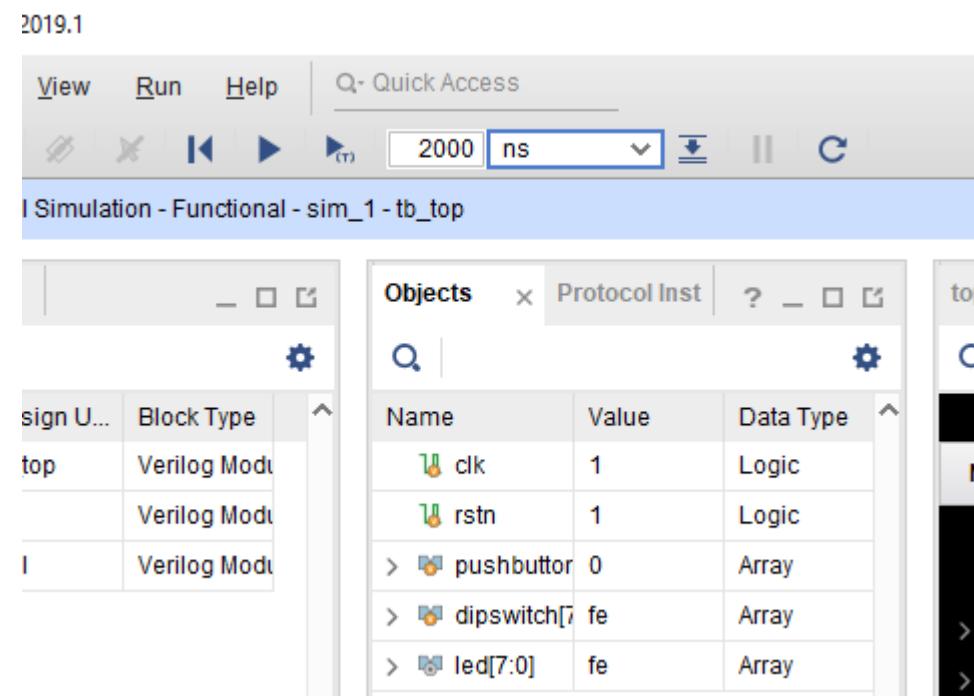
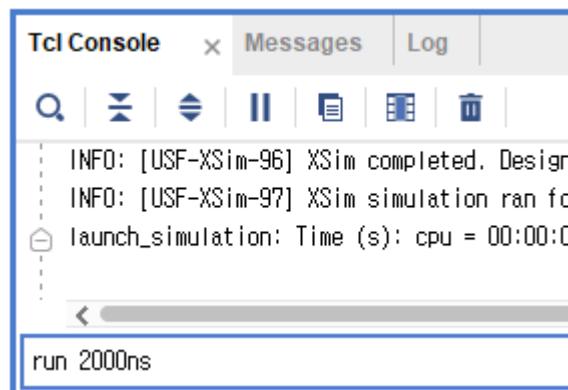
Example: Simple PL Design



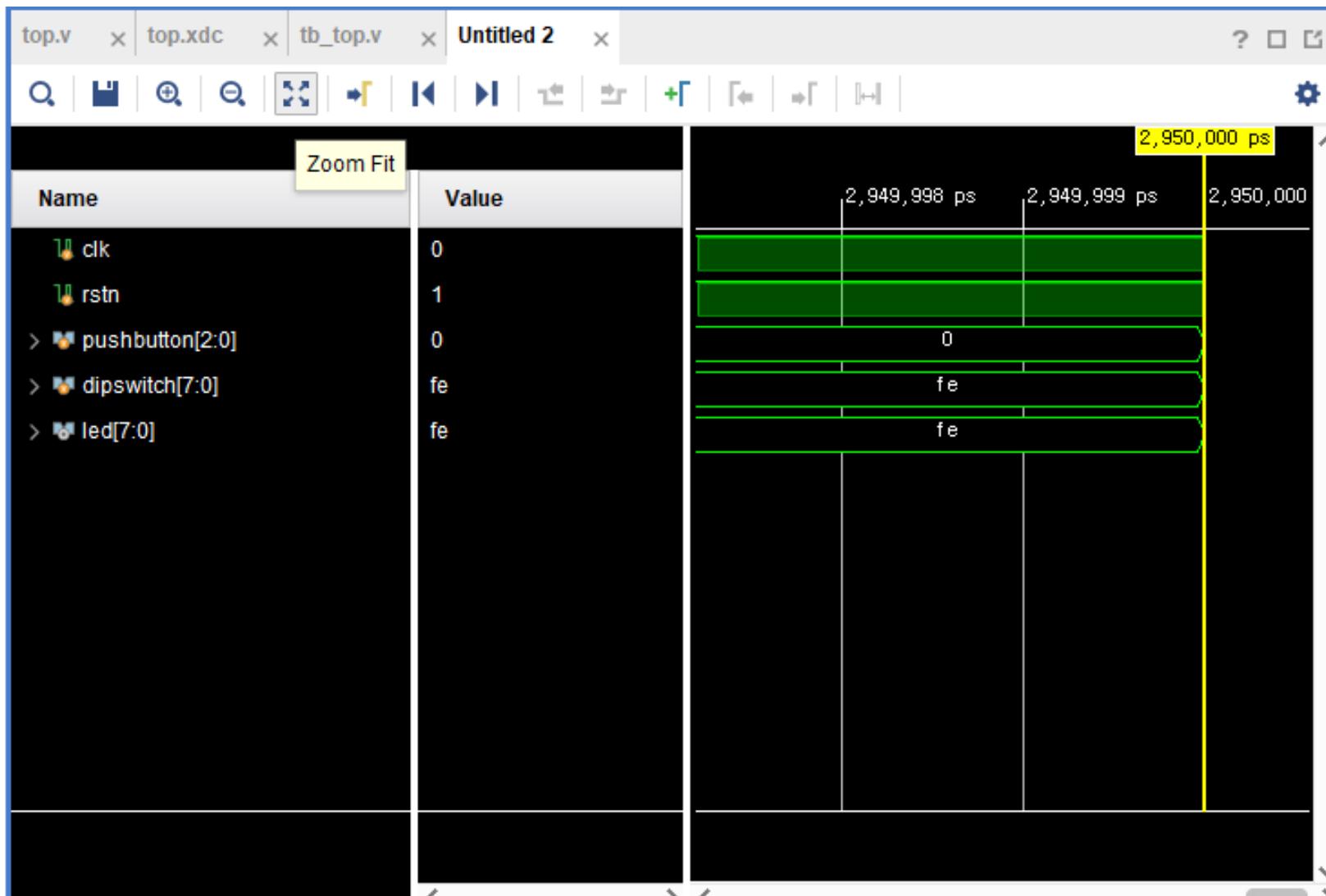
Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design

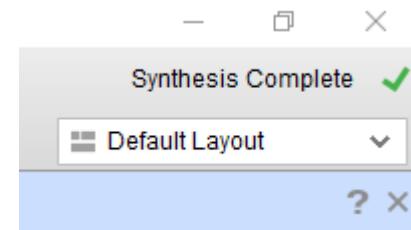
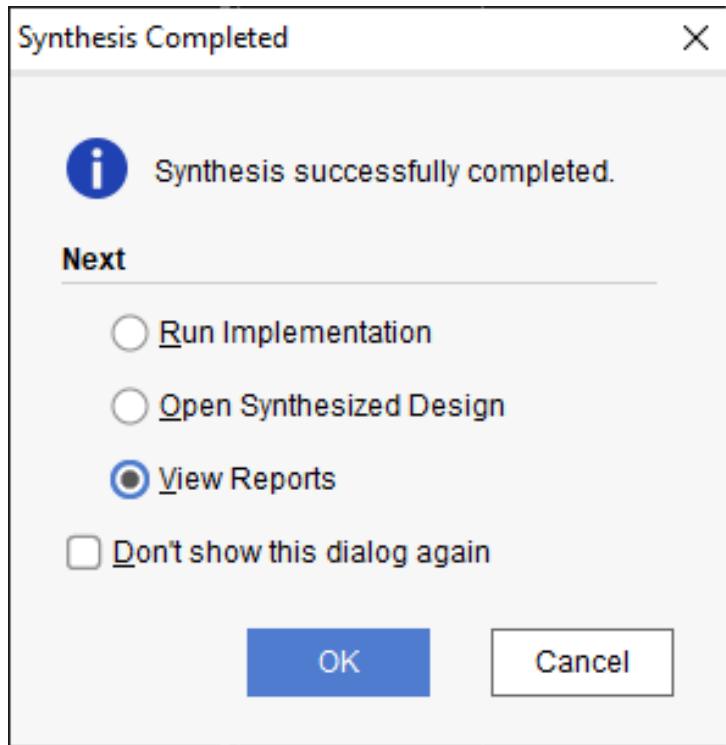


Example: Simple PL Design

The screenshot shows a CAD tool interface with the following components:

- Flow Navigator:** On the left, under the **SIMULATION** section, the **Run Synthesis** option is highlighted with a red box.
- SIMULATION - Behavioral Simulation - Fu:** The main workspace shows a **Scope** window with a table of sources and a waveform viewer below it. The waveform shows a signal labeled 'dut' with values 0, 1, and 2 over time intervals from 500 ns to 600 ns.
- Launch Runs dialog:** A modal dialog titled "Launch Runs" is open, prompting the user to "Launch the selected synthesis or implementation runs." It includes fields for "Launch directory" (set to "<Default Launch Directory>"), options for "Launch runs on local host" (set to 4 jobs) or "Generate scripts only", and checkboxes for "Don't show this dialog again". The "OK" button is highlighted with a red box.
- Top Status Bar:** The status bar at the top right indicates "Running synth_design" with a "Cancel" link and a green circular progress indicator.

Example: Simple PL Design



Example: Simple PL Design

SIMULATION

Run Simulation

RTL ANALYSIS

SYNTHESIS

Run Synthesis

Open Synthesized Design

Constraints Wizard

Edit Timing Constraints

Set Up Debug

Report Timing Summary

Report Clock Networks

Report Clock Interaction

Report Methodology

Report DRC

Report Noise

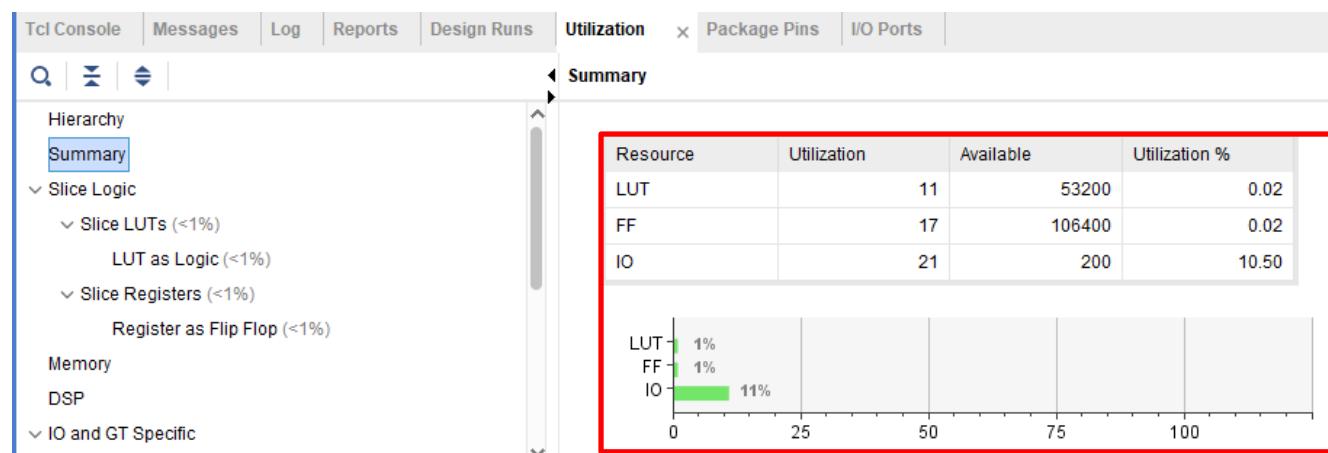
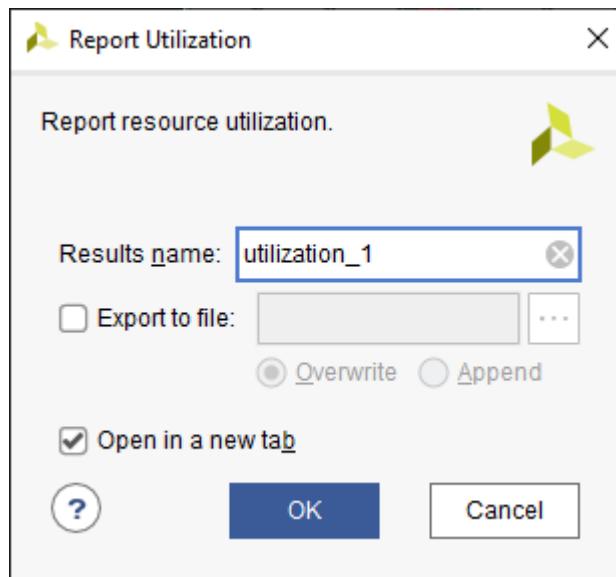
Report Utilization

Report Power

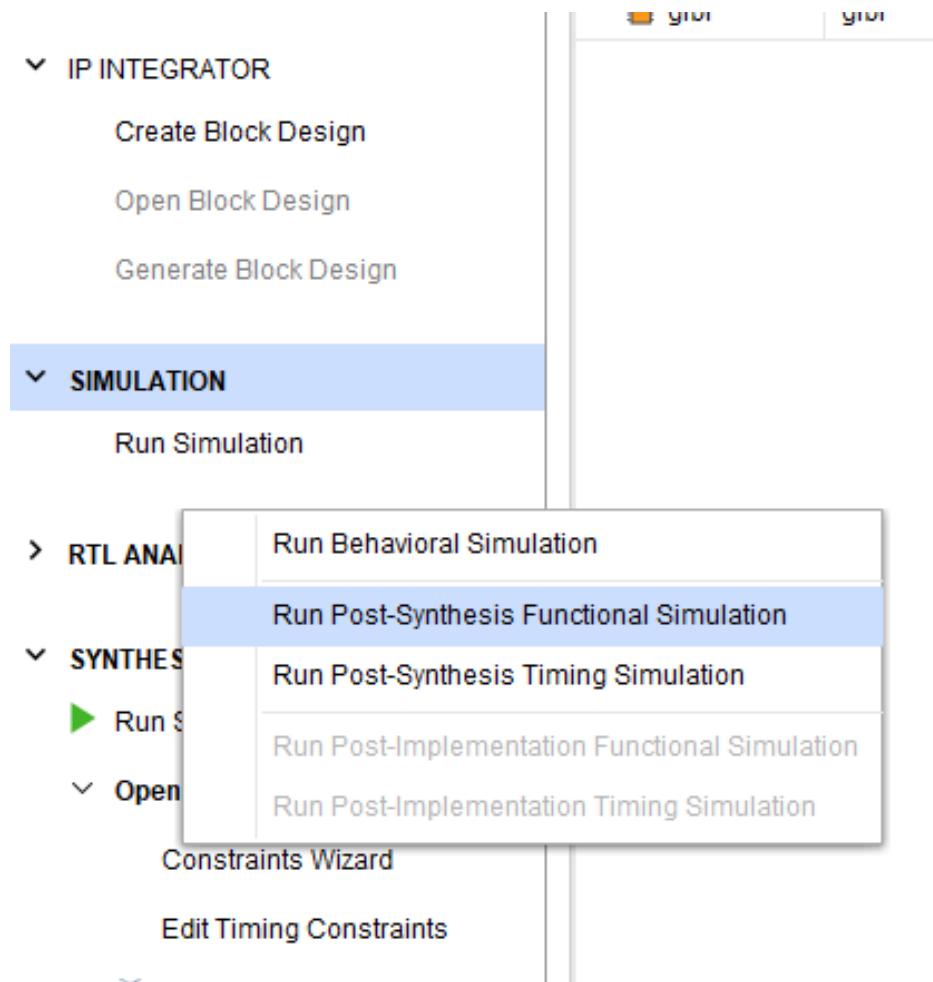
Schematic

IMPLEMENTATION

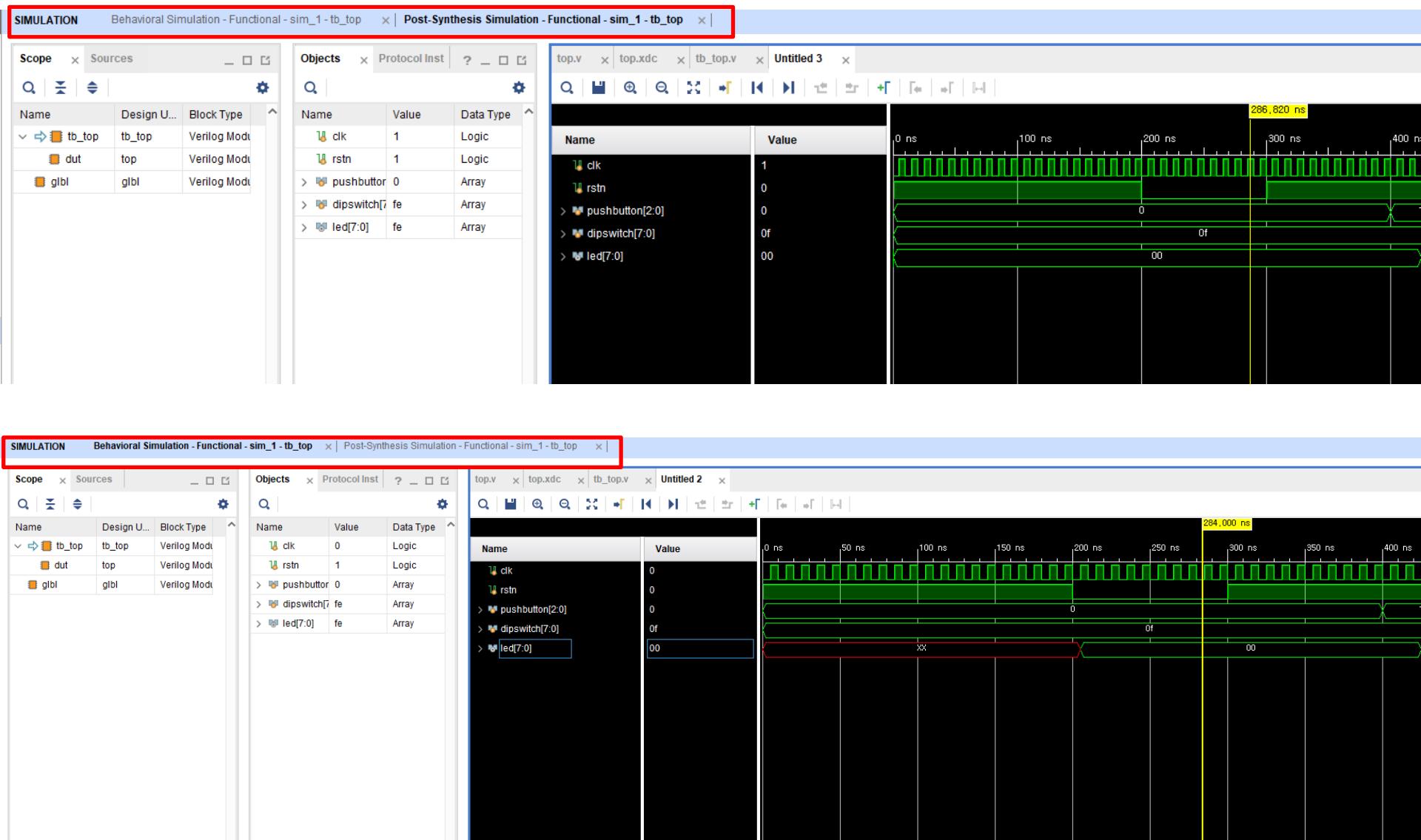
Run Implementation



Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design

The screenshot shows a CAD tool's interface with a sidebar containing navigation links:

- SIMULATION
 - Run Simulation
- RTL ANALYSIS
- SYNTHESIS
- IMPLEMENTATION
 - Run Implementation
 - Open Implemented Design
- PROGRAM AND DEBUG
 - Generate Bitstream
 - Open Hardware Manager

A red box highlights the "Generate Bitstream" option under "PROGRAM AND DEBUG".

Two dialog boxes are displayed over the interface:

- No Implementation Results Available**

There are no implementation results available. OK to launch implementation? 'Generate Bitstream' will automatically start when implementation completes.

Don't show this dialog again

Yes No
- Launch Runs**

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

Options

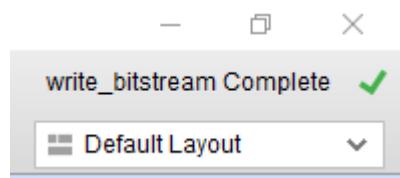
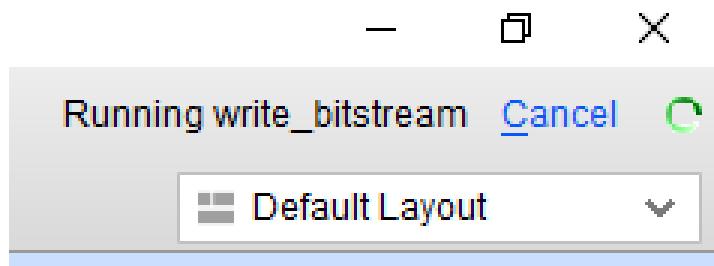
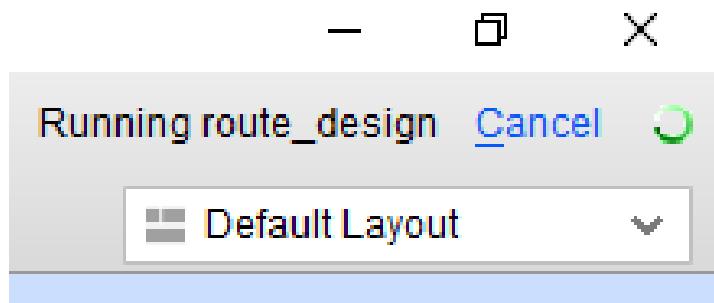
Launch runs on local host: Number of jobs: 4

Generate scripts only

Don't show this dialog again

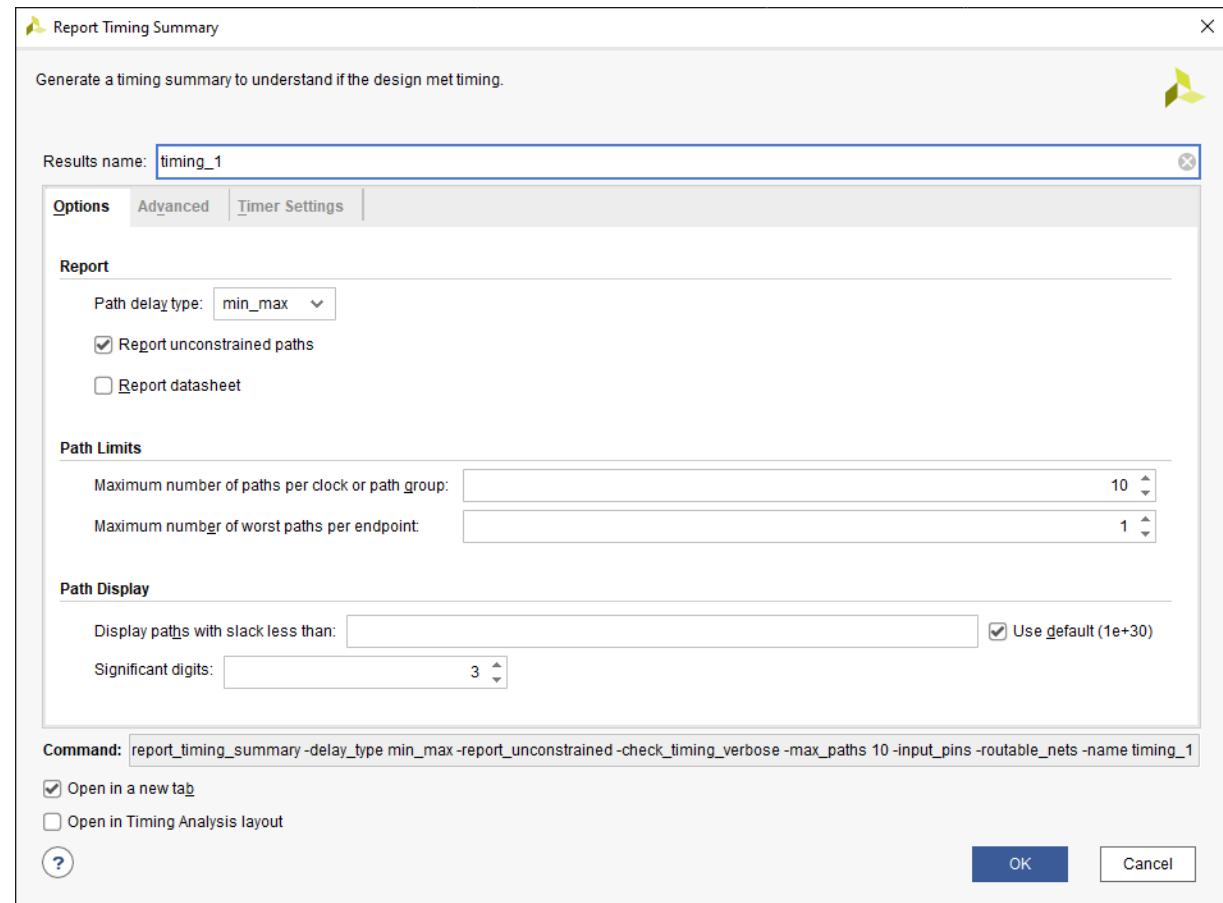
OK Cancel

Example: Simple PL Design



Example: Simple PL Design

- ▼ IMPLEMENTATION
 - ▶ Run Implementation
 - ▼ Open Implemented Design
 - Constraints Wizard
 - Edit Timing Constraints
 - ⌚ Report Timing Summary**
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Utilization
 -  Report Power
 -  Schematic



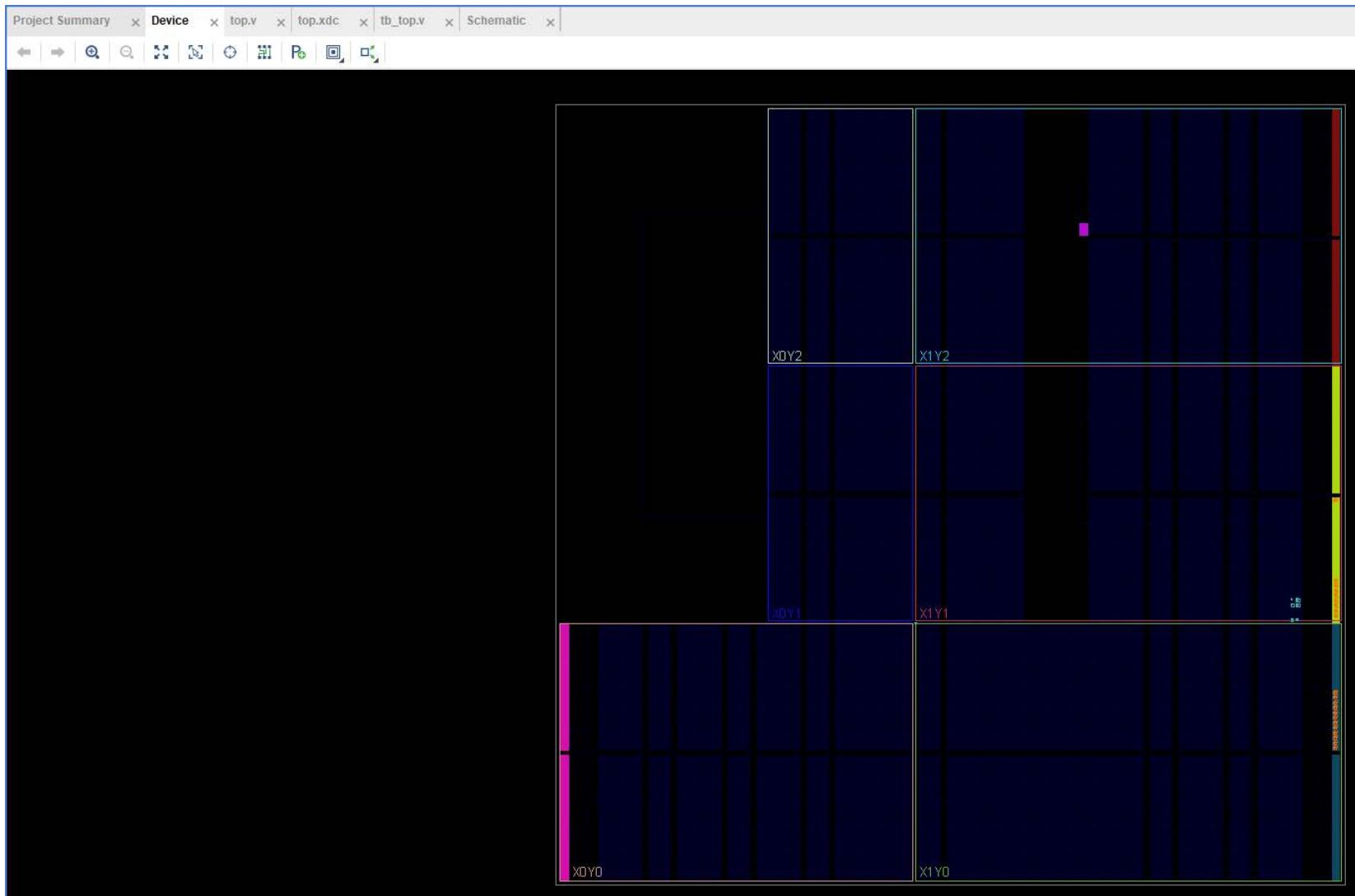
Example: Simple PL Design

The screenshot shows a software interface for timing analysis. The top menu bar includes 'Tcl Console', 'Messages', 'Log', 'Reports', 'Design Runs', 'DRC', 'Methodology', 'Power', and 'Timing'. The 'Timing' tab is selected. On the left, a sidebar lists navigation options: 'General Information', 'Timer Settings', 'Design Timing Summary' (which is selected and highlighted in blue), 'Clock Summary (1)', 'Check Timing (20)', 'Intra-Clock Paths', 'Inter-Clock Paths', and 'Other Path Groups'. The main pane is titled 'Design Timing Summary' and contains a table with three columns: 'Setup', 'Hold', and 'Pulse Width'. The table data is as follows:

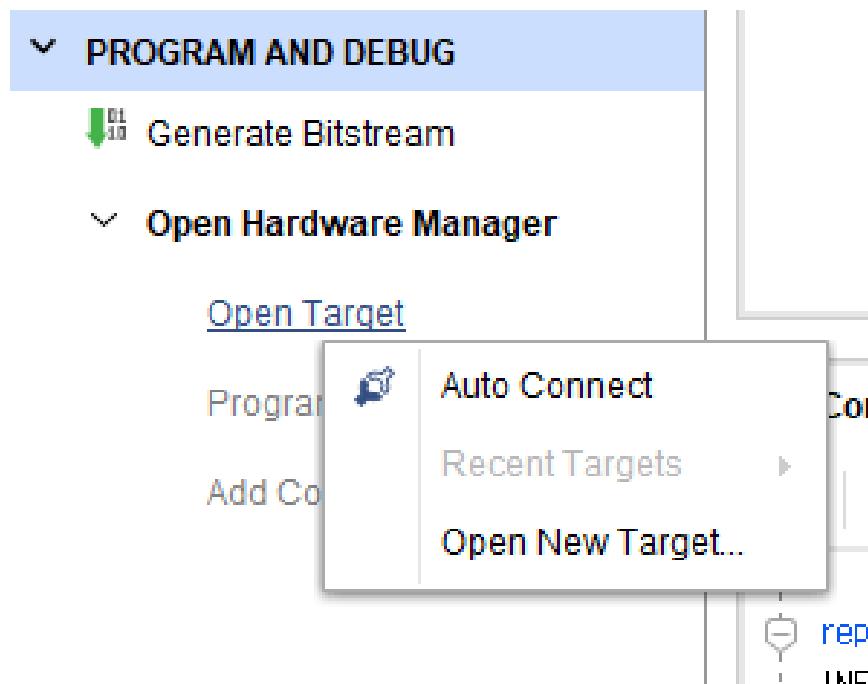
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 37.279 ns	Worst Hold Slack (WHS): 0.207 ns	Worst Pulse Width Slack (WPWS): 12.000 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 22	Total Number of Endpoints: 22	Total Number of Endpoints: 18

A message at the bottom of the table states: 'All user specified timing constraints are met.'

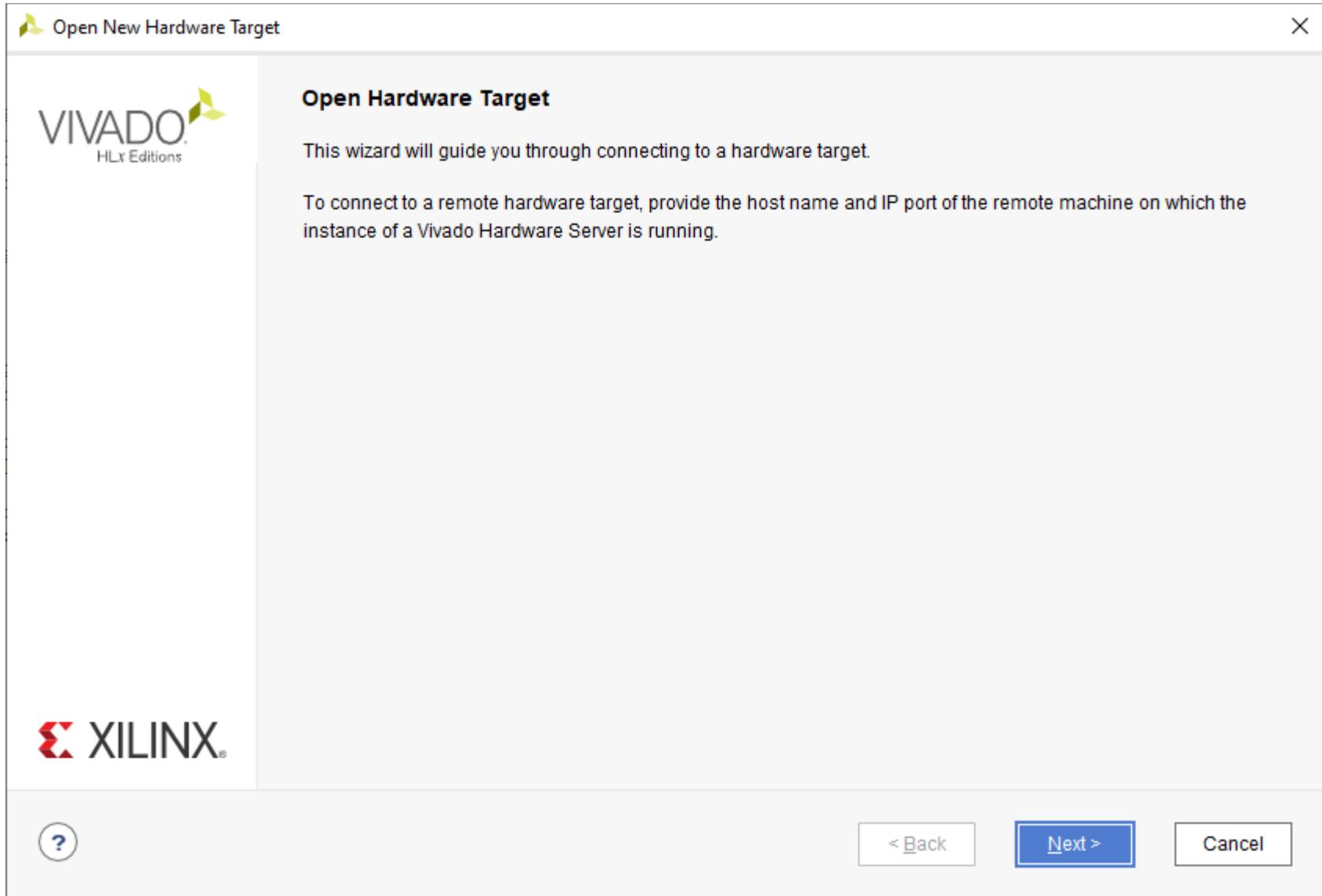
Example: Simple PL Design



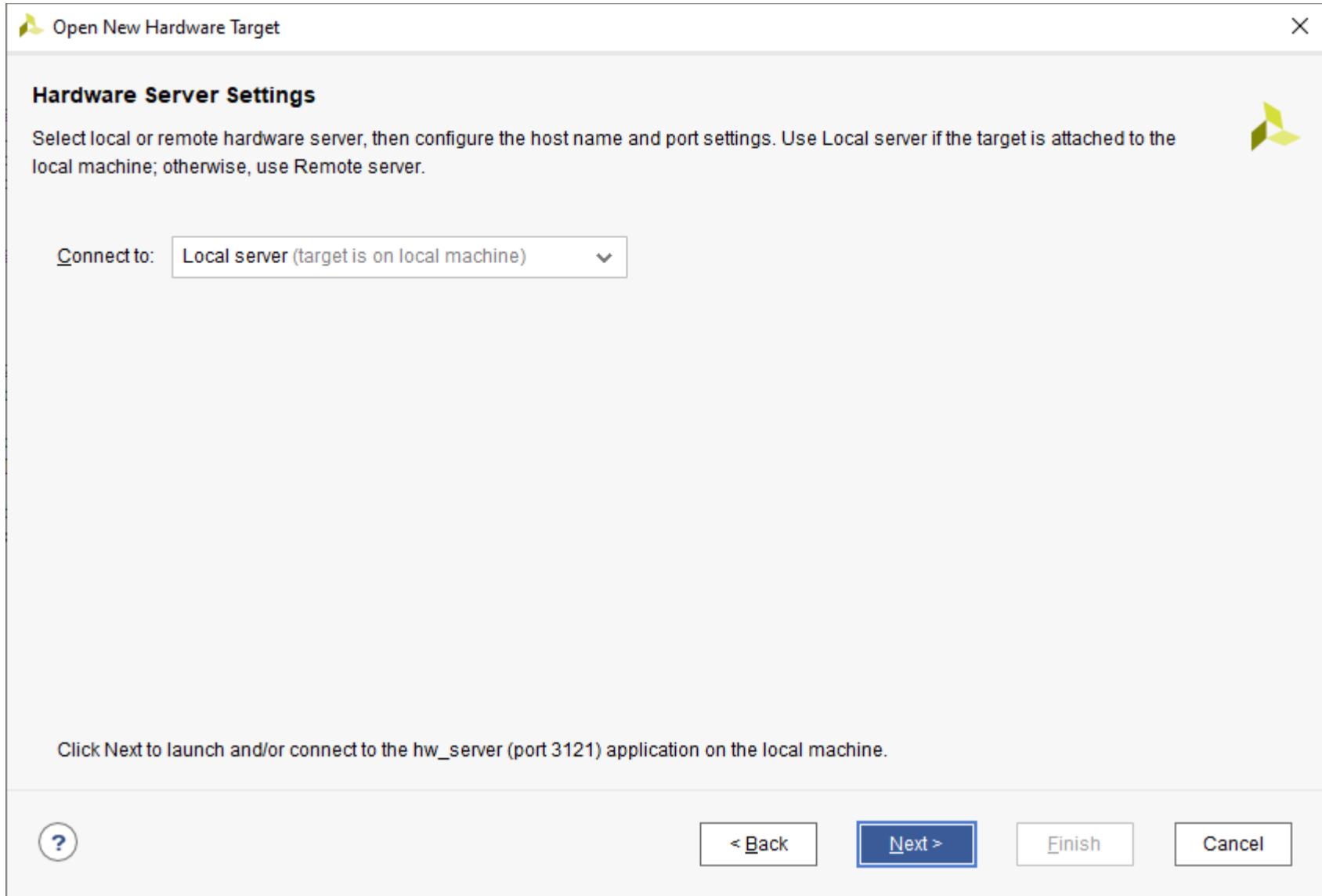
Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design

Open New Hardware Target X

Select Hardware Target

Select a hardware target from the list of available targets, then set the appropriate JTAG clock (TCK) frequency. If you do not see the expected devices, decrease the frequency or select a different target.

Hardware Targets

Type	Name	JTAG Clock Frequency
xilinx_tcf	Digilent/210249938355	10000000

Add Xilinx Virtual Cable (XVC)

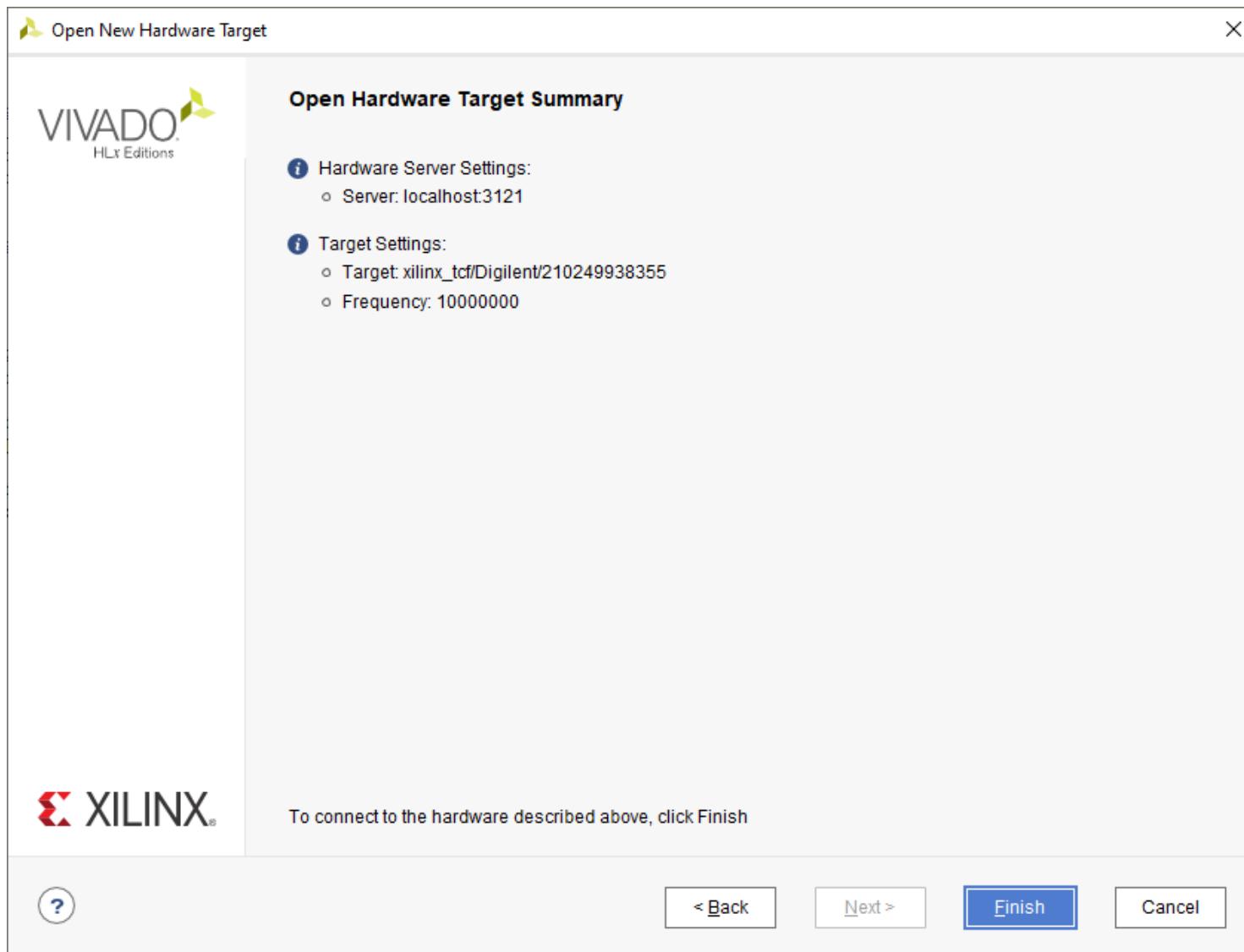
Hardware Devices (for unknown devices, specify the Instruction Register (IR) length)

Name	ID Code	IR Length
arm_dap_0	4BA00477	4
xc7z020_1	23727093	6

Hardware server: localhost:3121

? < Back Next > Finish Cancel

Example: Simple PL Design



Example: Simple PL Design

The screenshot shows the Xilinx Vivado interface. On the left, the 'Hardware Manager' window displays a list of connected devices: 'localhost (1)' (Connected), 'xilinx_tcf/Digilent/210249938355' (Open), 'arm_dap_0 (0)' (N/A), and 'xc7z020_1 (1)' (Not programmed). On the right, the 'top.v' file is open in the code editor, showing the following Verilog code:

```
top.v x top.xdc x tb_top.v x

D:/dsd_test/ex_simple_pl/ex_simple_pl.srcs/s

Q | ↻ | ← | → | X | ⌂ | ⌂ | X

`timescale 1ns / 1ps

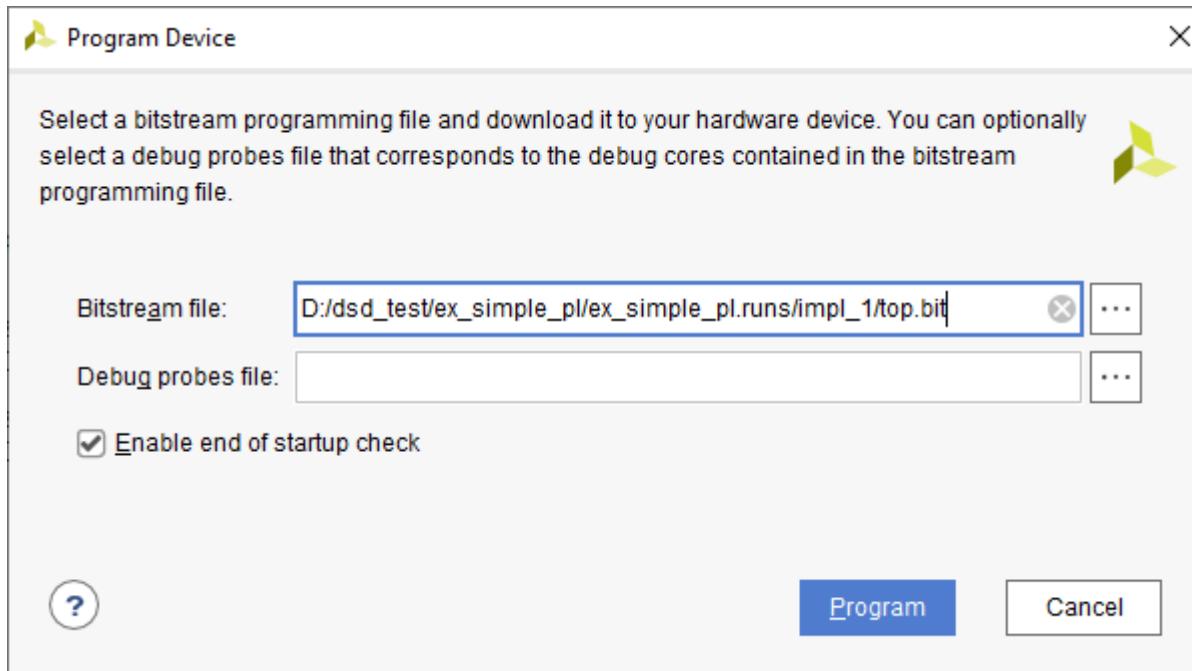
module top(
    input wire clk,
    input wire rstn,
    input wire [2:0] pushbutton,
    input wire [7:0] dipswitch,
    output reg [7:0] led
);

reg [2:0] pushbutton_q;
reg [2:0] pushbutton_q2;
reg [2:0] pushbutton_q3;

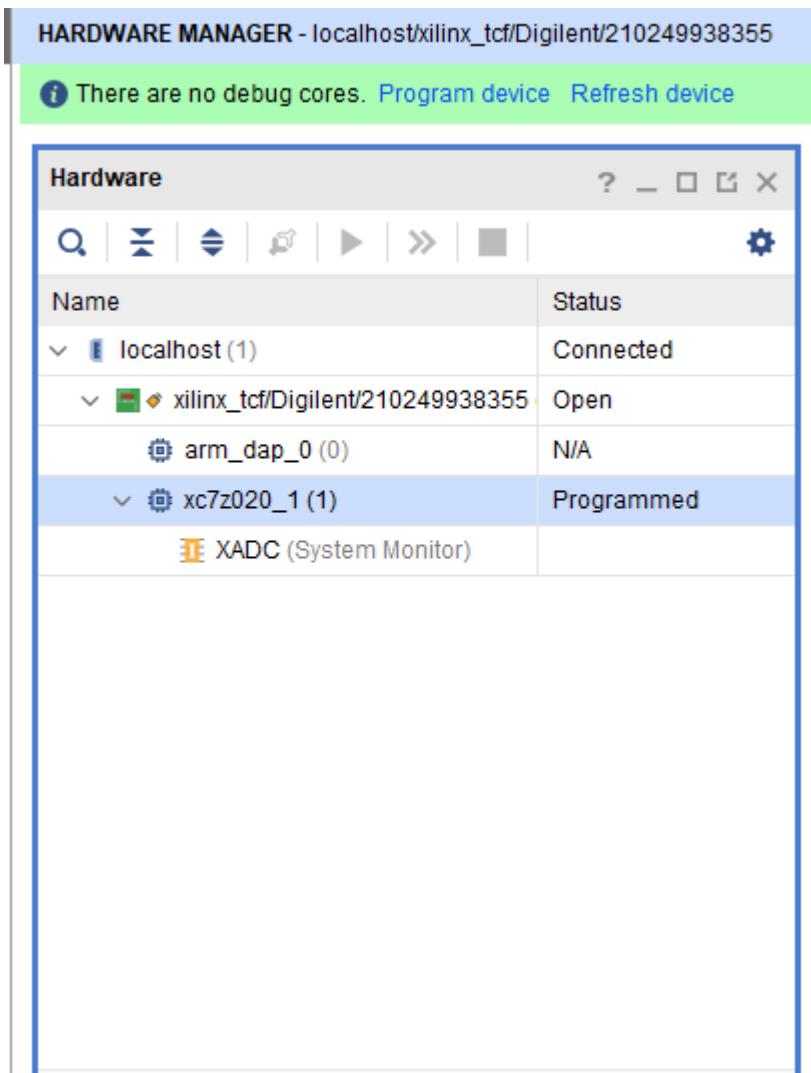
always @ (posedge clk) begin
    pushbutton_q <= pushbutton;
    pushbutton_q2 <= pushbutton_q;
    pushbutton_q3 <= pushbutton_q;
end

always @ (posedge clk) begin
    if (!rstn) begin
        led <= 0;
    end
    else begin
```

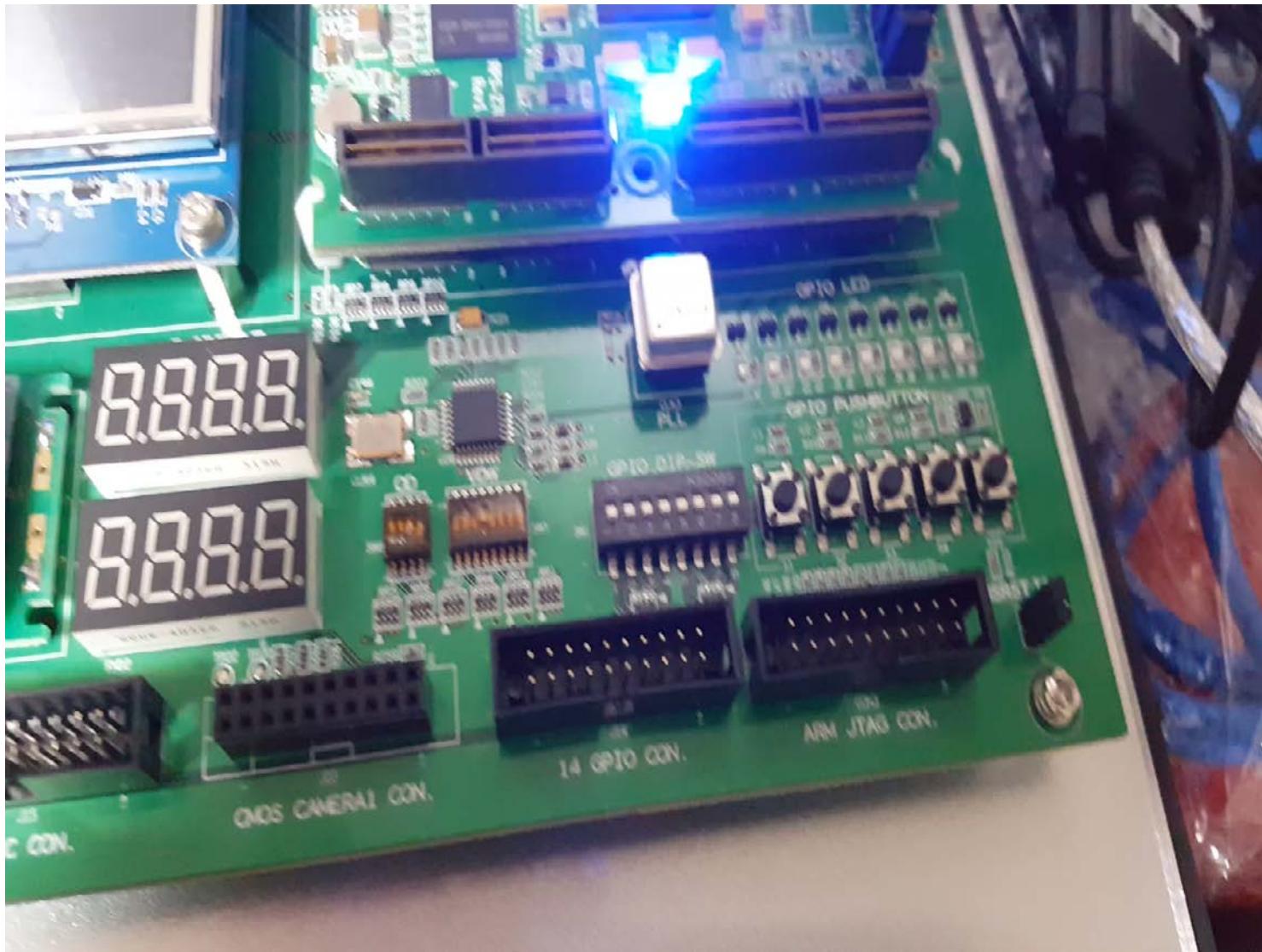
Example: Simple PL Design



Example: Simple PL Design



Example: Simple PL Design



Project – Reference code review

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ Vivado – Simulation
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

Project – Reference code review

```
//////////  
// open original bmp file  
//////////  
  
FILE* fp_lenna_original;  
fopen_s(&fp_lenna_original, "D:\\dsd_test\\lenna_256.bmp", "rb");  
  
unsigned char header_info_lenna_original[54];  
fread(header_info_lenna_original, sizeof(unsigned char), 54, fp_lenna_original);  
  
unsigned int header_start_addr;  
unsigned int header_width_original;  
unsigned int header_height_original;  
  
header_start_addr = *(unsigned int*)& header_info_lenna_original[10];  
header_width_original = *(unsigned int*)& header_info_lenna_original[18];  
header_height_original = *(unsigned int*)& header_info_lenna_original[22];  
  
// let's check data  
// header_start_address should be "54"  
// header_width_original should be "128"  
// header_height_original should be "128"  
printf("header_start_addr: %u\\n", header_start_addr);  
printf("header_width_original: %u\\n", header_width_original);  
printf("header_height_original: %u\\n", header_height_original);  
  
unsigned int size_lenna_original;  
size_lenna_original = 3 * header_width_original * header_height_original;  
  
// load pixel data  
unsigned char* data_lenna_original = (char*)malloc(sizeof(unsigned char) * size_lenna_original);  
fread(data_lenna_original, sizeof(unsigned char), size_lenna_original, fp_lenna_original);
```

Project – Reference code review

```
//////////  
// lenna_256.coe memory file gen.  
//////////  
FILE* fp_lenna_coe;  
fopen_s(&fp_lenna_coe, "D:\\dsd_test\\lenna_256.coe", "wb");  
  
fprintf(fp_lenna_coe, "memory_initialization_radix = 16;\n");  
fprintf(fp_lenna_coe, "memory_initialization_vector=\n");  
unsigned char tmp_byte_data = 0;  
  
for (int row = 0; row < header_height_original; row++) {  
    for (int col = 0; col < header_width_original; col++) {  
        for (int channel = 2; channel >= 0; channel--) {  
            tmp_byte_data = data_lenna_original[row * header_width_original * 3 + col * 3 + channel];  
            fprintf(fp_lenna_coe, "%x", tmp_byte_data);  
        }  
        fprintf(fp_lenna_coe, ",\n");  
    }  
}  
fseek(fp_lenna_coe, -2, SEEK_CUR);  
fprintf(fp_lenna_coe, ";");  
  
fclose(fp_lenna_coe);
```

Project – Reference code review

```
//////////  
// do down-sample and write down-sampled bmp file  
//////////  
  
FILE* fp_lenna_ds;  
fopen_s(&fp_lenna_ds, "D:\\dsd_test\\lenna_ds.bmp", "wb");  
  
// make lenna_ds's header info  
unsigned char header_info_lenna_ds[54];  
  
// copy lenna_128's header info  
for (int i = 0; i < 54; i++) {  
    header_info_lenna_ds[i] = header_info_lenna_original[i];  
}  
  
// change width & height info  
unsigned int header_width_ds;  
unsigned int header_height_ds;  
header_width_ds = header_width_original / 2;  
header_height_ds = header_height_original / 2;  
*(unsigned int*)& header_info_lenna_ds[18] = header_width_ds;  
*(unsigned int*)& header_info_lenna_ds[22] = header_height_ds;  
  
// check whether lenna_ds's header info is changed correctly  
unsigned int test_tmp;  
test_tmp = *(unsigned int*)& header_info_lenna_ds[18];  
printf("ds_header_width: %u\n", test_tmp);  
test_tmp = *(unsigned int*)& header_info_lenna_ds[22];  
printf("ds_header_height: %u\n", test_tmp);
```

Project – Reference code review

```
// do simple down-sampling
// in this example, we will just select(sample) one of the 4 pixels
// you have to develop this part

unsigned int size_lenna_ds;
size_lenna_ds = 3 * header_width_ds * header_height_ds;

unsigned char* data_lenna_ds = (char*)malloc(sizeof(unsigned char) * size_lenna_ds);

int idx = 0;
for (int row = 0; row < header_height_original; row++) {
    for (int col = 0; col < header_width_original; col++) {
        if ((row % 2) == 1) {
            break;
        }
        else {
            if ((col % 2) == 0) {
                data_lenna_ds[idx + 0] = data_lenna_original[row * header_width_original * 3 + col * 3 + 0];
                data_lenna_ds[idx + 1] = data_lenna_original[row * header_width_original * 3 + col * 3 + 1];
                data_lenna_ds[idx + 2] = data_lenna_original[row * header_width_original * 3 + col * 3 + 2];
                idx = idx + 3;
            }
        }
    }
}

// write data
fwrite(header_info_lenna_ds, sizeof(unsigned char), 54, fp_lenna_ds);
fwrite(data_lenna_ds, sizeof(unsigned char), size_lenna_ds, fp_lenna_ds);
```

Project – Reference code review

```
//////////  
// do up-sample and write up-sampled bmp file  
//////////  
  
FILE* fp_lenna_us;  
fopen_s(&fp_lenna_us, "D:\\dsd_test\\lenna_us.bmp", "wb");  
  
// make lenna_ds's header info  
unsigned char header_info_lenna_us[54];  
  
// copy lenna_128's header info  
for (int i = 0; i < 54; i++) {  
    header_info_lenna_us[i] = header_info_lenna_original[i];  
}  
  
// change width & height info  
unsigned int header_width_us;  
unsigned int header_height_us;  
header_width_us = header_width_original;  
header_height_us = header_height_original;
```

Project – Reference code review

```
// do simple up-sampling
// in this example, we will just reproduct data
// you have to develop this part
unsigned int size_lenna_us;
size_lenna_us = 3 * header_width_us * header_height_us;

unsigned char* data_lenna_us = (char*)malloc(sizeof(unsigned char) * size_lenna_us);

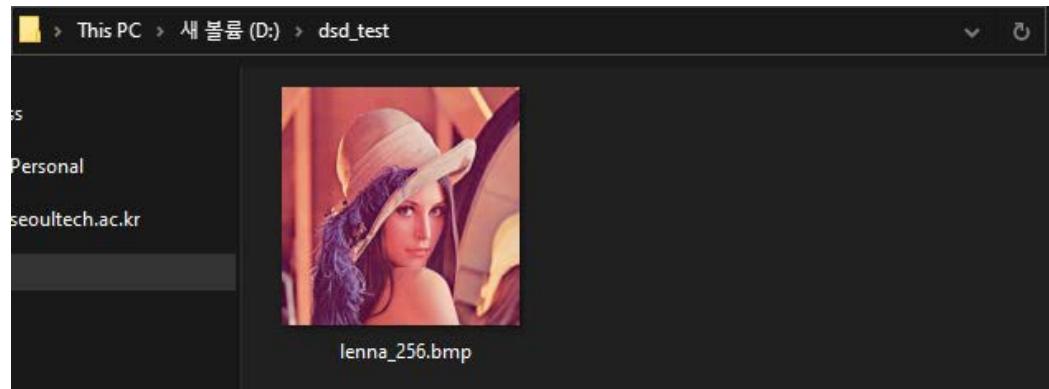
for (int row = 0; row < header_height_ds; row++) {
    for (int col = 0; col < header_width_ds; col++) {
        for (int k = 0; k < 3; k++) {
            data_lenna_us[row * 2 * header_width_us * 3 + col * 2 * 3 + k] = data_lenna_ds[row * header_width_ds * 3 + col * 3 + k];
            data_lenna_us[row * 2 * header_width_us * 3 + col * 2 * 3 + k + 3] = data_lenna_ds[row * header_width_ds * 3 + col * 3 + k];
            data_lenna_us[(row * 2 + 1) * header_width_us * 3 + col * 2 * 3 + k] = data_lenna_ds[row * header_width_ds * 3 + col * 3 + k];
            data_lenna_us[(row * 2 + 1) * header_width_us * 3 + col * 2 * 3 + k + 3] = data_lenna_ds[row * header_width_ds * 3 + col * 3 + k];
        }
    }
}

// write data
fwrite(header_info_lenna_us, sizeof(unsigned char), 54, fp_lenna_us);
fwrite(data_lenna_us, sizeof(unsigned char), size_lenna_us, fp_lenna_us);
```

Project – Reference code review

```
//////////  
// calculate psnr  
//////////  
  
double mse_tmp = 0;  
double mse = 0;  
double psnr = 0;  
int size = 3 * header_height_original * header_width_original;  
  
for (int i = 0; i < size; i++) {  
    mse_tmp += pow(data_lenna_original[i] - data_lenna_us[i], 2);  
}  
mse = mse_tmp / size;  
  
psnr = 20 * log10(255) - 10 * log10(mse);  
printf("psnr: %lf\n", psnr);
```

Project – Reference code review



```
Microsoft Visual Studio Debug Console  
header_start_addr: 54  
header_width_original: 256  
header_height_original: 256  
lenna[0][0][0] = 57  
lenna[0][0][1] = 22  
lenna[0][0][2] = 83  
ds_header_width: 128  
ds_header_height: 128  
psnr: 26.484972
```



Project – Vivado given sample code review

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ Vivado – Simulation
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

Project – Vivado given sample code review

```
    ▼ Simulation Sources (4)
      ▼ sim_1 (4)
        ▼ tb_top_ds_us_test (tb_top_ds_us_test.v) (1)
          ▼ dut : top_ds_us_test (top_ds_us_test.v) (3)
            ▼ rom_inst : block_rom_lenna256 (block_rom_lenna256.xci) (1)
              > block_rom_lenna256 (block_rom_lenna256.v) (1)
            ▼ ds_us_inst : top_ds_us (top_ds_us.v) (2)
            ▼ ram_inst : block_ram_dual_port_test (block_ram_dual_port_test.xci) (1)
              > block_ram_dual_port_test (block_ram_dual_port_test.v) (1)
```

Project – Vivado given sample code review

```
1 `timescale 1ns / 1ps
2
3 module ds(
4     // global signals
5     input wire clk,
6     input wire rstn,
7     // system signals
8     input wire sys_en,
9     // brom interface
10    output reg [15:0] rom_addr,
11    input wire [23:0] rom_data,
12    output wire rom_en,
13    // down-sampled data & control signals
14    output wire [23:0] ds_data,
15    output wire ds_done,
16    output wire ds_wr_en
17 );
18
19 // Notice: to read data is delayed 1-clock cycle
20 // you should be aware of it!
21 // design your own ds code below!
22 // you can add any port or signals
23
24
25 endmodule
--
```

Project – Vivado given sample code review

```
1 `timescale 1ns/1ps
2
3 module us(
4     // global signals
5     input wire clk,
6     input wire rstn,
7     // system signals
8     input wire sys_en,
9     // down-sampled data & control signal
10    input wire [23:0] ds_data,
11    input wire ds_vld,
12    // bram interface
13    output reg [15:0] ram_wr_addr,
14    output wire [31:0] ram_wr_data,
15    output wire ram_en,
16    output wire ram_rst,
17    // ram write enable byte
18    output wire [3:0] ram_web,
19    // up-sampled control signals
20    output wire us_done,
21    output wire us_wr_en
22 );
23
24 // Notice: if you want to write data to bram,
25 // then set "ram_web" as "4'b1111"
26 // otherwise(i.e. you don't want to write data)
27 // set "ram_web" as "4'b0000"
28 // design your own ds code below!
29 // you can add any port or signals
30
31
32
33 endmodule
```

Project – Vivado given sample code review

```
1 `timescale 1ns/1ps
2
3 module top_ds_us(
4     // global signals
5     input wire clk,
6     input wire rstn,
7     // system signals
8     input wire sys_en,
9     // brom interface
10    output wire [15:0] rom_addr,
11    input wire [23:0] rom_data,
12    output wire rom_en,
13    // bram interface
14    output wire [15:0] ram_wr_addr,
15    output wire [31:0] ram_wr_data,
16    output wire ram_en,
17    output wire ram_rst,
18    output wire [3:0] ram_web,
19    // down-sampled data & control signals
20    output wire [23:0] ds_out,
21    output wire ds_done,
22    output wire ds_wr_en,
23    // up-sampled data & control signals
24    output wire [23:0] us_out,
25    output wire us_done,
26    output wire us_wr_en
27 );
28
29 assign us_out = ram_wr_data[23:0];
30
31 wire ds_vld;
32 assign ds_vld = ds_wr_en;
33
34 // down-sample module instantiation
35 ds ds_inst(
36     // global signals
37     .clk(clk),
38     .rstn(rstn),
39     // system signals
40     .sys_en(sys_en),
41     // brom interface
42     .rom_addr(rom_addr),
43     .rom_data(rom_data),
44     .rom_en(rom_en),
45     // down-sampled data & control signals
46     .ds_data(ds_out),
47     .ds_done(ds_done),
48     .ds_wr_en(ds_wr_en)
49 );
50
51 //-----//
52 // you can add control unit(module) here
53 //-----//
54
55 // up-sample module instantiation
56 us us_inst(
57     // global signals
58     .clk(clk),
59     .rstn(rstn),
60     // system signals
61     .sys_en(sys_en),
62     // down-sampled data & control signal
63     .ds_data(ds_out),
64     .ds_vld(ds_vld),
65     // bram interface
66     .ram_wr_addr(ram_wr_addr),
67     .ram_wr_data(ram_wr_data),
68     // port to check whether data is written correctly
69     .ram_rd_data(ram_rd_data),
70     .ram_en(ram_en),
71     .ram_rst(ram_rst),
72     .ram_web(ram_web),
73     // up-sampled control signals
74     .us_done(us_done),
75     .us_wr_en(us_wr_en)
76 );
77
78 endmodule
```

Project – Vivado given sample code review

```
3 module top_ds_us_test(
4     // global signals
5     input wire clk,
6     input wire rstn,
7     // system signals
8     input wire sys_en,
9     // down-sampled data & control signals
10    output wire [23:0] ds_out,
11    output wire ds_done,
12    output wire ds_wr_en,
13    // up-sampled data & control signals
14    output wire [23:0] us_out,
15    output wire [15:0] us_addr,
16    output wire us_done,
17    output wire us_wr_en,
18    // test ram data
19    input wire ram_result_test_en,
20    input wire [15:0] ram_result_test_addr,
21    output wire [31:0] ram_result_test_data
22 );
23
24 wire [15:0] rom_addr;
25 wire [23:0] rom_data;
26 wire rom_en;
27
28 wire [15:0] ram_wr_addr;
29 wire [31:0] ram_wr_data;
30 wire ram_en;
31 wire ram_rst;
32 wire [3:0] ram_web;
33
34 wire ram_rd_en;
35 wire [18:0] ram_rd_addr;
36 wire [31:0] ram_rd_data;
37
38 assign us_addr = ram_wr_addr;
```

```
40     // block rom instantiation
41     block_rom_lenna256      rom_inst(
42         .clka(clk),
43         .ena(rom_en),
44         .addra(rom_addr),
45         .douta(rom_data)
46     );
47
48     top_ds_us      ds_us_inst(
49         // global signals
50         .clk(clk),
51         .rstn(rstn),
52         // system signals
53         .sys_en(sys_en),
54         // brom interface
55         .rom_addr(rom_addr),
56         .rom_data(rom_data),
57         .rom_en(rom_en),
58         // bram interface
59         .ram_wr_addr(ram_wr_addr),
60         .ram_wr_data(ram_wr_data),
61         .ram_en(ram_en),
62         .ram_rst(ram_rst),
63         .ram_web(ram_web),
64         // down-sampled data & control signals
65         .ds_out(ds_out),
66         .ds_done(ds_done),
67         .ds_wr_en(ds_wr_en),
68         // up-sampled data & control signals
69         .us_out(us_out),
70         .us_done(us_done),
71         .us_wr_en(us_wr_en)
72 );
```

Project – Vivado given sample code review

```
74     block_ram_dual_port_test ram_inst(
75         // port A, cpu will read bram data through this port
76         .clka(clk),
77         .ena(ram_result_test_en),
78         .wea(),
79         .addr(a(ram_result_test_addr)),
80         .dina(),
81         .douta(ram_result_test_data),
82         // port B, your output will be written through this port
83         .clkb(clk),
84         .rstb(ram_rst),
85         .enb(ram_en),
86         .web(ram_web),
87         .addrb(ram_wr_addr),
88         .dinb(ram_wr_data),
89         .doutb(),
90         .rsta_busy(),
91         .rstb_busy()
92     );
93
94 endmodule
```

Project – Vivado testbench examples

- ◆ About FPGA
- ◆ Design Flow
- ◆ Project explanation
- ◆ Vivado – Download
- ◆ Vivado – Create project
- ◆ Vivado – Add/Create sources
- ◆ Vivado – Simulation
- ◆ Vivado – Example: Simple PL Design
- ◆ Project – Reference code review (Video)
- ◆ Project – Vivado given sample code review (Video)
- ◆ Project – Vivado testbench examples (Video)

Project – Vivado testbench examples

- ◆ Let's learn to open file and write data

```
1  `timescale 1ns / 1ps
2
3  module tb_file_open_ex();
4      parameter wr_file_name = "D://dsd_test//test.txt";
5      integer fp_wr;
6
7      reg clk;
8      reg file_wr_en;
9      reg [9:0] addr;
10     wire end_of_frame_wr;
11
12 //-----
13 // generate clock
14 //-----
15 initial begin
16     clk = 0;
17 end
18 always begin
19     #5 clk = ~clk;
20 end
21
22 //-----
23 // set file_write_enable as 1
24 //-----
25 initial begin
26     file_wr_en = 0;
27     #100     file_wr_en = 1;
28 end
29
```

Project – Vivado testbench examples

```
30 //---  
31 // set file_write_enable as 0,  
32 // if file_write operation reaches end of frame  
33 //---  
34 always @(*) begin  
35     if (end_of_frame_wr == 1) begin  
36         file_wr_en = 0;  
37         $fclose(fp_wr);  
38         $stop;  
39     end  
40 end  
41 //---  
42 // increase address according to file_write_enable  
43 //---  
44 initial begin  
45     addr = 0;  
46 end  
47 always @(posedge clk) begin  
48     if (file_wr_en) begin  
49         addr <= addr + 1;  
50     end  
51 end
```



```
test.txt - Notepad  
File Edit Format View  
000  
001  
002  
003  
004  
005  
006  
007  
008  
009  
00a  
00b  
00c  
00d  
00e
```

```
52 //---  
53 // in this example, if address == 15,  
54 // then stop file write operation  
55 //---  
56 assign end_of_frame_wr = (addr == 15) ? 1 : 0;  
57 //---  
58 // file open and write  
59 //---  
60 initial begin  
61     fp_wr = $fopen(wr_file_name, "wb");  
62     if (!fp_wr) begin  
63         $display("write bmp file open error\n");  
64         $finish;  
65     end  
66     $display("%s file opened\n", wr_file_name);  
67 end  
68  
69 always @(posedge clk) begin  
70     if (file_wr_en) begin  
71         $fwrite(fp_wr, "%x\n", addr);  
72     end  
73 end  
74  
75  
76  
77 endmodule
```

Project – Vivado testbench examples

- ◆ Let's open bmp file and copy.

```
1  `timescale 1ns/1ps
2
3  module tb_file_copy_ex();
4  //-----
5  // parameter declaration
6  //-----
7  parameter rd_file_name = "D://dsd_test//lenna_256.bmp";
8  parameter wr_file_name = "D://dsd_test//lanna_copied.bmp";
9  parameter img_size = (256 * 256 * 3);
10
11 integer fp_rd, fp_wr;
12
13 reg [7:0] header_rd [53:0];
14 reg [7:0] header_wr [53:0];
15
16 reg [7:0] img_data_rd [((img_size-1)):0];
17 reg [7:0] img_data_wr [((img_size-1)):0];
18
19 reg [7:0] tmp_byte_data;
20 integer r, i, j, k;
21
```

Project – Vivado testbench examples

```
22 initial begin
23     //-----
24     // file open and read original image
25     //-----
26     fp_rd = $fopen(rd_file_name, "rb");
27     if (!fp_rd) begin
28         $display("read bmp file open error\n");
29         $finish;
30     end
31     $display("%s file opend\n", rd_file_name);
32
33     // read header information
34     for (i = 0; i < 54; i = i + 1) begin
35         #1 r = $fread(tmp_byte_data, fp_rd);
36         header_rd[i] = tmp_byte_data;
37     end
38
39     // read image data
40     for (i = 0; i < 196608; i = i + 1) begin
41         #1 r = $fread(tmp_byte_data, fp_rd);
42         img_data_rd[i] = tmp_byte_data;
43     end
44 :
```



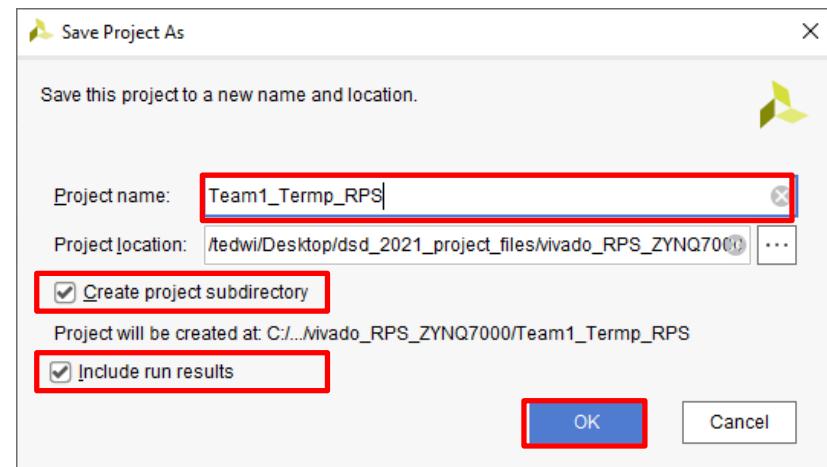
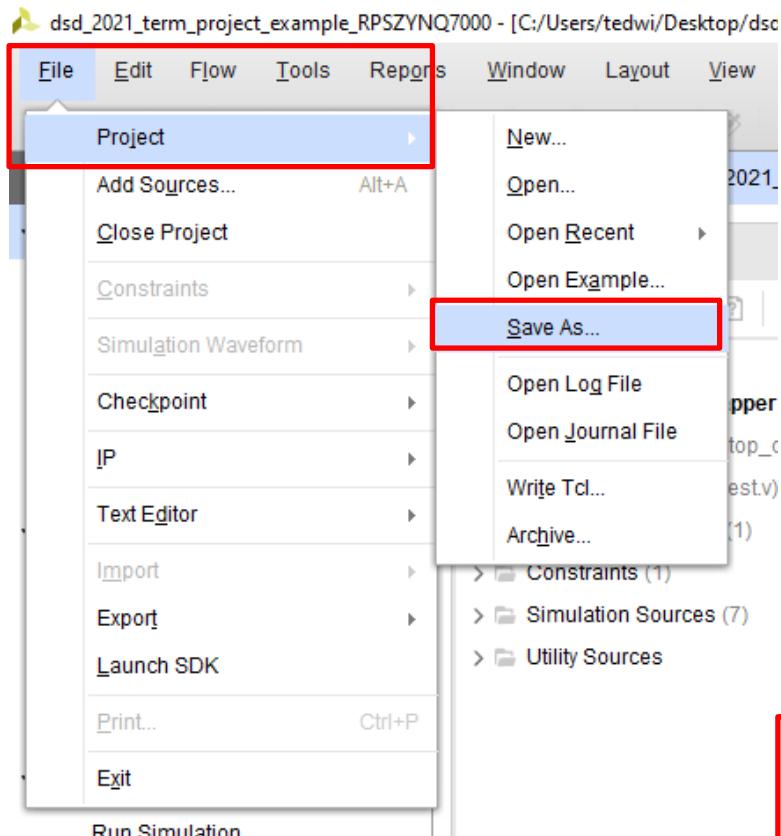
lenna_256.bmp



lenna_copied.bmp

```
45     //-----
46     // copy data
47     //-----
48     for (i = 0; i < 54; i = i + 1) begin
49         header_wr[i] = header_rd[i];
50     end
51     for (i = 0; i < 196608; i = i + 1) begin
52         img_data_wr[i] = img_data_rd[i];
53     end
54
55     //-----
56     // file open and write copied image
57     //-----
58     fp_wr = $fopen(wr_file_name, "wb");
59     if (!fp_wr) begin
60         $display("write bmp file open error\n");
61         $finish;
62     end
63     $display("%s file opend\n", wr_file_name);
64
65     // write header information
66     for (i = 0; i < 54; i = i + 1) begin
67         #1 $fwrite(fp_wr, "%c", header_wr[i]);
68     end
69
70     // write copied data
71     for (i = 0; i < 196608; i = i + 1) begin
72         $fwrite(fp_wr, "%c", img_data_wr[i]);
73     end
74
75     $stop;
76     $fclose(fp_rd);
77     $fclose(fp_wr);
78     $stop;
79 end
80
81 endmodule
```

Vivado project file



Project Name: Team{Number}_Termp_{Board Name}

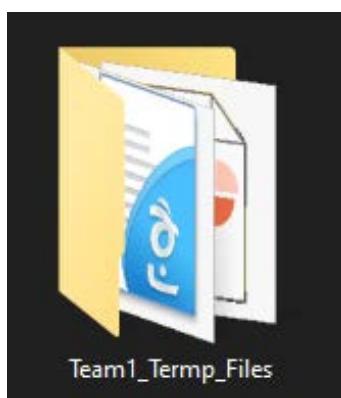
Number: 1 ~ 10

Board Name: "RPS" or "ZYBO"

Ex1: Team1_Termp_RPS

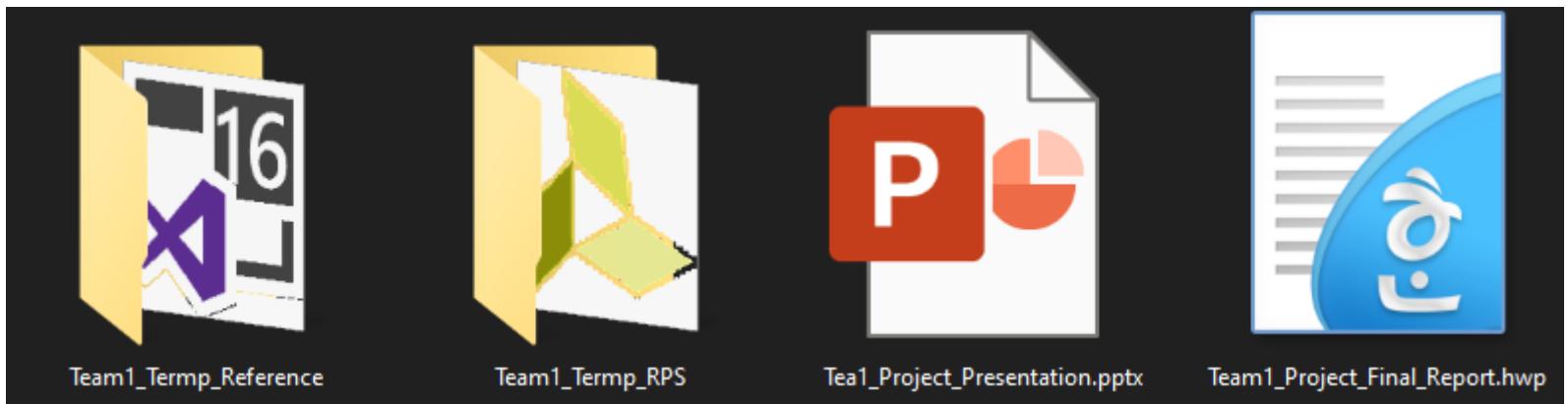
Ex2: Team2_Termp_ZYBO

Submission Form



Team{number}_Termp_Files.zip

- Team{number}_Termp_Reference
- Team{number}_Termp_{board name}
- Team{number}_Project_Presentation
- Team{number}_Project_Final_Report



Team1_Termp_Reference

Team1_Termp_RPS

Tea1_Project_Presentation.pptx

Team1_Project_Final_Report.hwp