Lab1: Introduction to the EDA Tools

National Chiao Tung University
Chun-Jen Tsai
9/18/2018

Lab 1 Goal: Simulate an 8-bit Multiplier

- In this lab, you must simulate the operations of a sequential binary multiplier using the Vivado Simulator
  - You should review your old textbook on Digital Circuit Design by Mano. Some design guideline of the sequential binary multiplier is in section 8.7 of Mano's book
  - The multiplier is designed using only adder, shifter, multiplexor, and gate-level operators. You can not use the multiplication operator of Verilog.

- You should complete the simulation and show some waveforms on 9/18

The Target Technology of Digital Labs

- Digital circuits can be implemented in different ways
  - Circuits Boards
    - Circuit board design using standard IC parts (e.g., 74SLxx)
  - Application Specific ICs
    - Full-custom and Cell-based IC designs
  - Programmable logics
    - Field Programmable Gate Array (FPGA) design
- Here, we use Xilinx FPGAs for circuit implementation
  - Xilinx is the largest FPGA manufacturing company in the world
  - Ross Freeman, the co-founder of Xilinx, invented the very first FPGA in 1986[†]

† US Patent 4,870,302

Xilinx Vivado Design Suite

- Xilinx has two different EDA tools for FPGA-based digital system designs
  - Vivado Design Suite
    - Only for 7th-generation FPGAs and above
    - Unified IDE for both “SoC” and “digital circuit” designs
  - ISE Design Suite
    - For 7th- and older generations of FPGAs
    - ISE EDK for SoC designs
    - ISE Project Navigator for digital circuit designs
- In this course, we use the Vivado Design Suite for digital circuit design

Vivado Circuit Implementation Flow

- Step 1: Design Entry
  - Input your circuit design using Hardware Description Language (HDL), such as Verilog or VHDL
- Step 2: Synthesis
  - Convert from the HDL programs or schematics to a netlist file that define a list of circuit blocks and how they are connected
- Step 3: Mapping
  - Determine what FPGA resource will be used to implement which part of the netlist
- Step 4: Place-and-Route
  - Determine physical location and routing of the circuit resource
  - A “*.bit” file will be generated for the FPGA device

Vivado Circuit Debug Flow

- Your design may not be perfect in the first try!
  - Circuit debugging is done via “simulation” or “signal probing”
- Vivado supports several simulation types. In particular:
  - Behavioral simulation
    - Functional simulation before synthesis; assumes zero delay
  - Post implementation functional simulation
    - Functional simulation after synthesis; assumes zero delay
  - Post implementation timing simulation
    - Simulate signal switching of your circuit with exact signal delays on the target devices
    - Also called “post-sim”
- Vivado Logic Analyzer can analyze runtime signals

Install Your Own Vivado Design Suite

- Install a copy of Vivado Design Suite – HLx Editions 2018.2 onto your computer.
  - You can download it from:

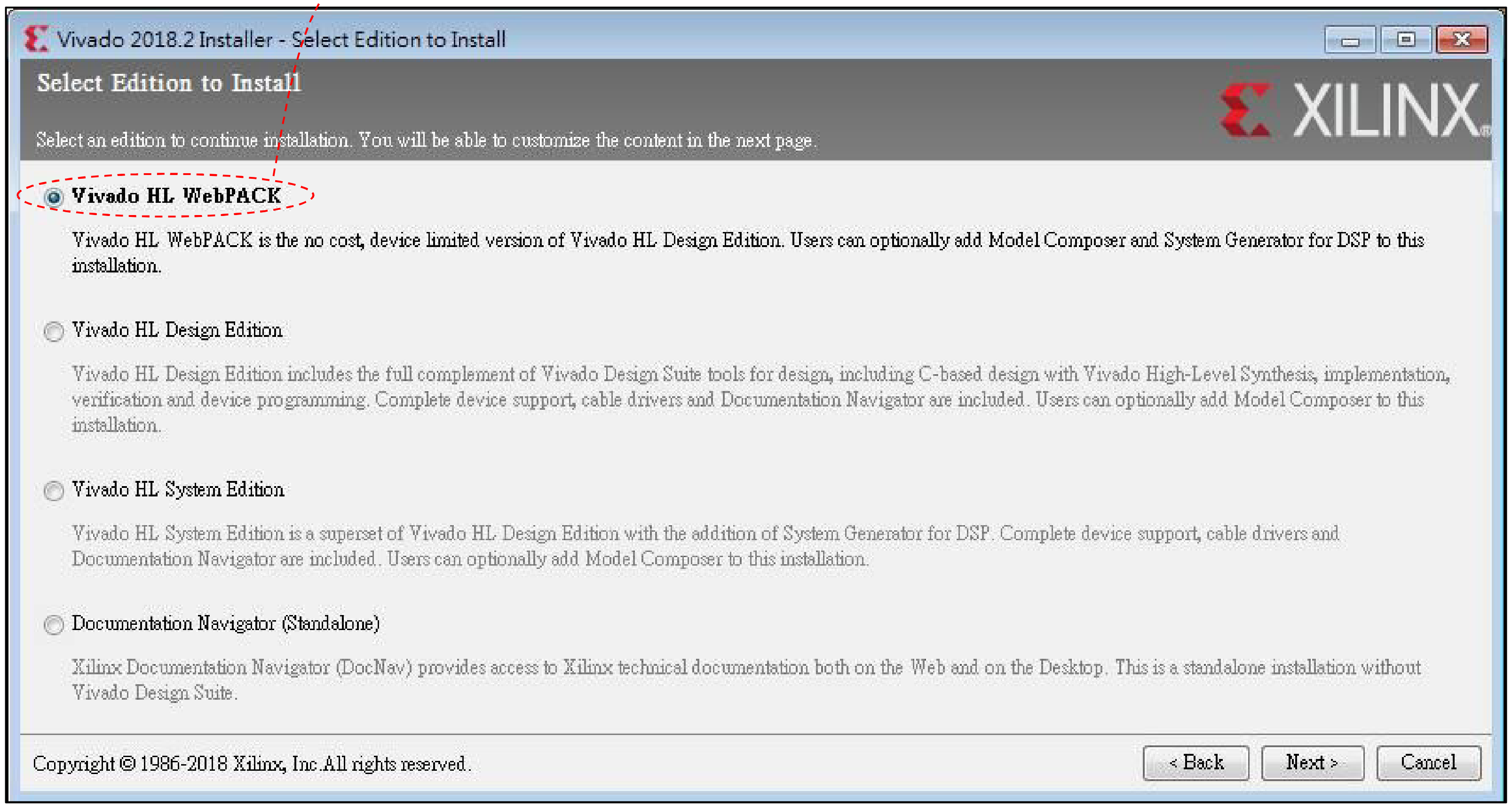
    https://www.xilinx.com/support/download.html

- The installation requires about 16 GiB of disk space
  - Please install the “WebPACK” version and register online for a free license

Vivado Installation Guide (1/2)

- You must select the right version upon installation:

Free license version, for personal use.



Vivado 2018.2 Installer - Select Edition to Install

Select Edition to Install

Select an edition to continue installation. You will be able to customize the content in the next page.

[✓ **Vivado HL WebPACK**]

Vivado HL WebPACK is the no cost, device limited version of Vivado HL Design Edition. Users can optionally add Model Composer and System Generator for DSP to this installation.

Vivado HL Design Edition

Vivado HL Design Edition includes the full complement of Vivado Design Suite tools for design, including C-based design with Vivado High-Level Synthesis, implementation, verification and device programming. Complete device support, cable drivers and Documentation Navigator are included. Users can optionally add Model Composer to this installation.

Vivado HL System Edition

Vivado HL System Edition is a superset of Vivado HL Design Edition with the addition of System Generator for DSP. Complete device support, cable drivers and Documentation Navigator are included. Users can optionally add Model Composer to this installation.

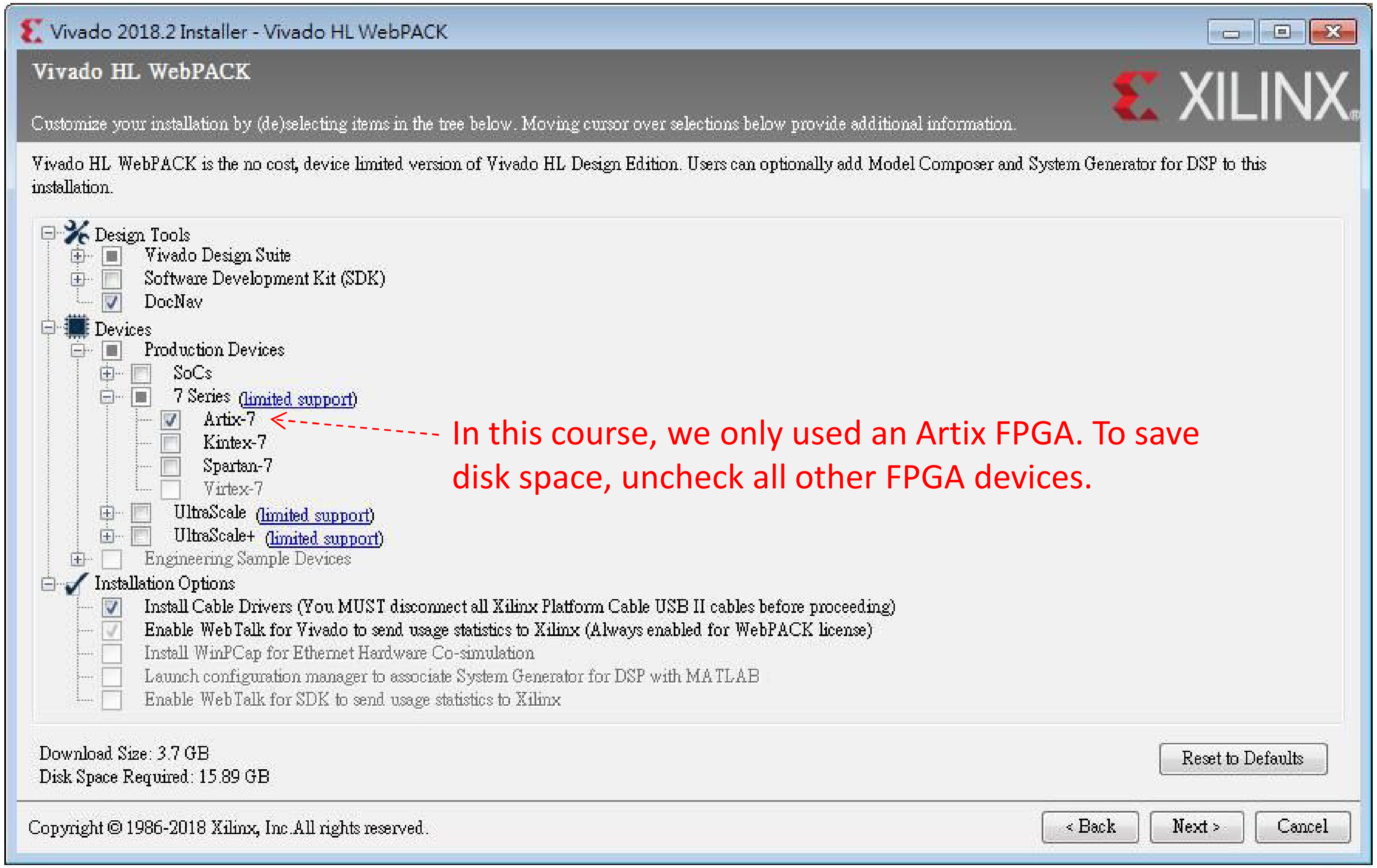
Documentation Navigator (Standalone)

Xilinx Documentation Navigator (DocNav) provides access to Xilinx technical documentation both on the Web and on the Desktop. This is a standalone installation without Vivado Design Suite.

Copyright © 1986-2018 Xilinx, Inc. All rights reserved.

Vivado Installation Guide (2/2)

- Selecting the required packages and FPGA devices:



Customize your installation by (de)selecting items in the tree below. Moving cursor over selections below provide additional information.

Vivado HL WebPACK is the no cost, device limited version of Vivado HL Design Edition. Users can optionally add Model Composer and System Generator for DSP to this installation.

- Design Tools
  - ▪ Vivado Design Suite
  - ☐ Software Development Kit (SDK)
  - ☑ DocNav
- Devices
  - ▪ Production Devices
    - ☐ SoCs
    - ▪ 7 Series (limited support)
      - ☑ Artix-7
      - ☐ Kintex-7
      - ☐ Spartan-7
      - ☐ Virtex-7
    - ☐ UltraScale (limited support)
    - ☐ UltraScale+ (limited support)
  - ☐ Engineering Sample Devices
- Installation Options
  - ☑ Install Cable Drivers (You MUST disconnect all Xilinx Platform Cable USB II cables before proceeding)
  - ☑ Enable WebTalk for Vivado to send usage statistics to Xilinx (Always enabled for WebPACK license)
  - ☐ Install WinPCap for Ethernet Hardware Co-simulation
  - ☐ Launch configuration manager to associate System Generator for DSP with MATLAB
  - ☐ Enable WebTalk for SDK to send usage statistics to Xilinx

In this course, we only used an Artix FPGA. To save disk space, uncheck all other FPGA devices.

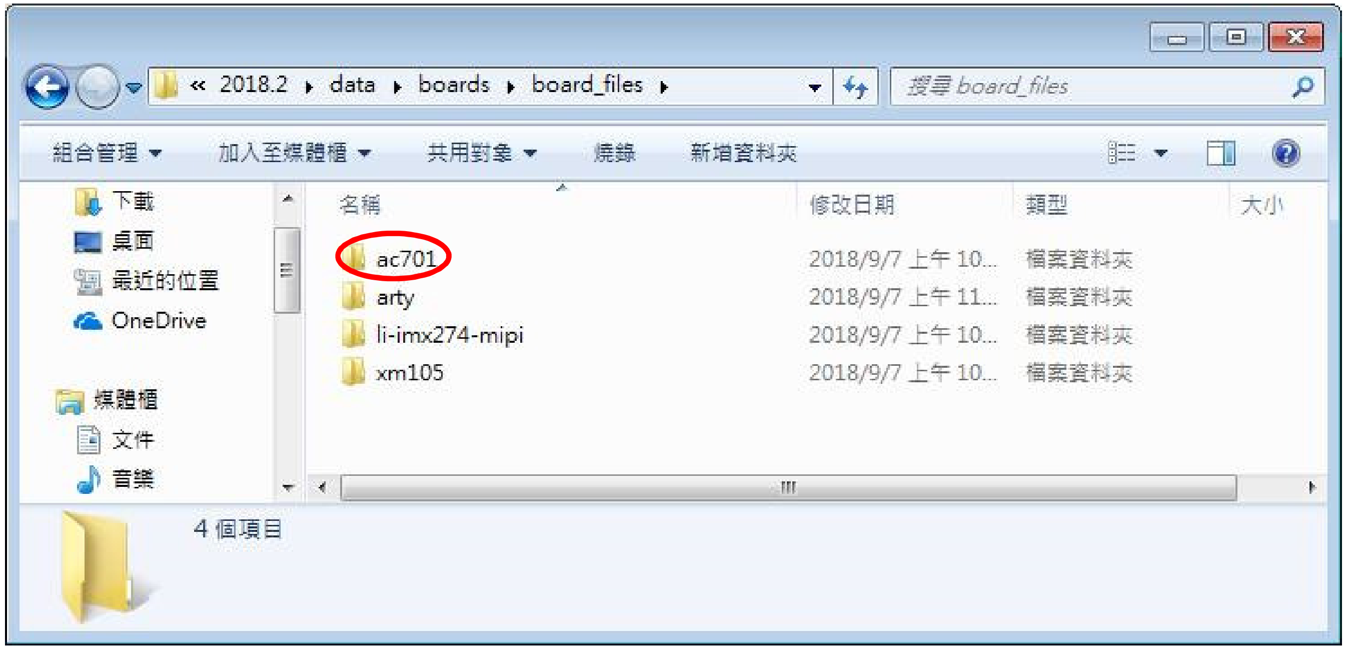
Download Size: 3.7 GB
Disk Space Required: 15.89 GB

Copyright © 1986-2018 Xilinx, Inc. All rights reserved.

Installation of Arty Board Definitions

- After the installation of Vivado, you must install the board definition file of Arty:
  - Download arty.zip from E3.
  - Unzip arty.zip to the following directory:

C:\<INST_DIR>\Xilinx\Vivado\2018.2\data\boards\board_files

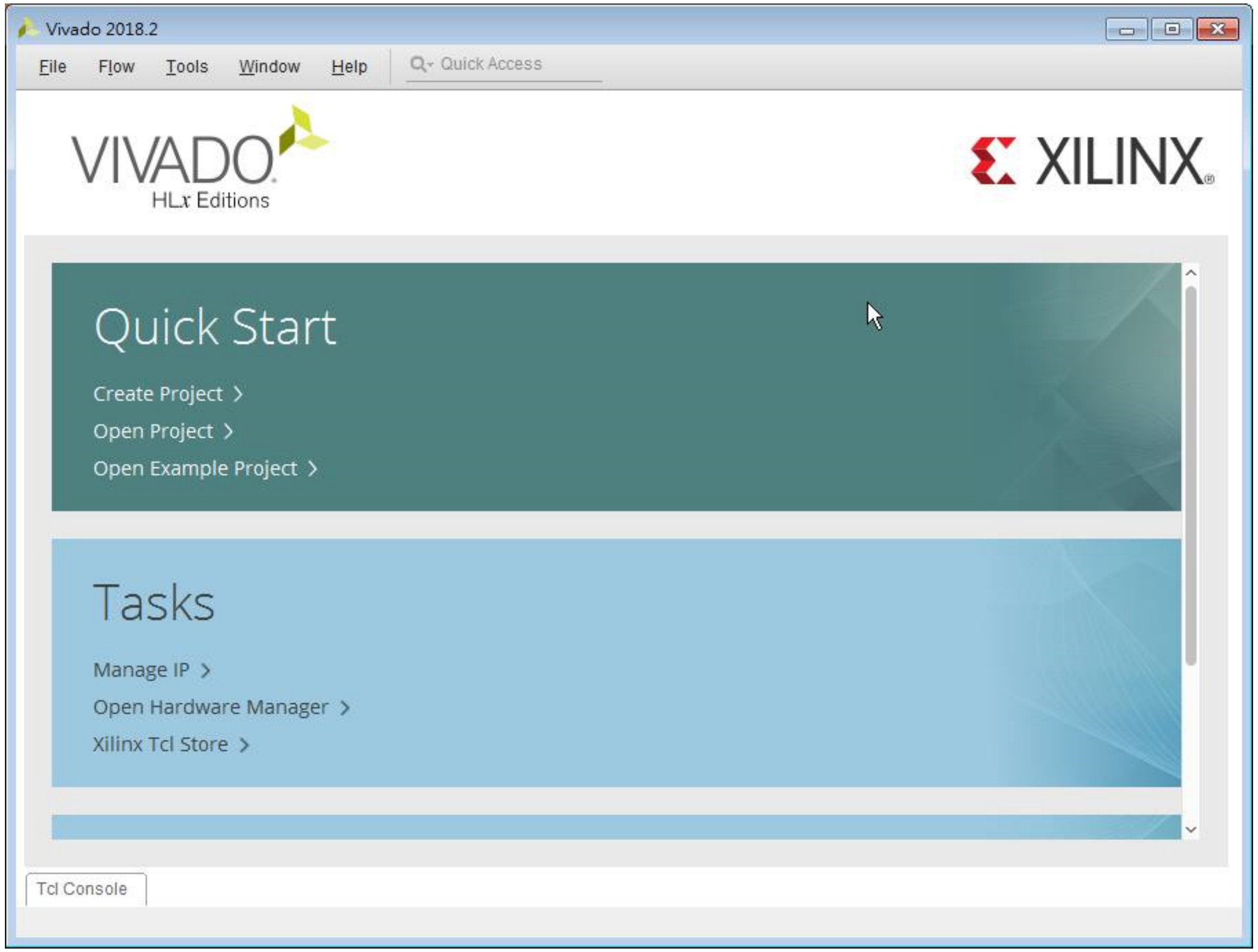

« 2018.2 ▸ data ▸ boards ▸ board_files ▸

名稱	修改日期	類型	大小
ac701	2018/9/7 上午 10...	檔案資料夾	
arty	2018/9/7 上午 11...	檔案資料夾	
li-imx274-mipi	2018/9/7 上午 10...	檔案資料夾	
xm105	2018/9/7 上午 10...	檔案資料夾	

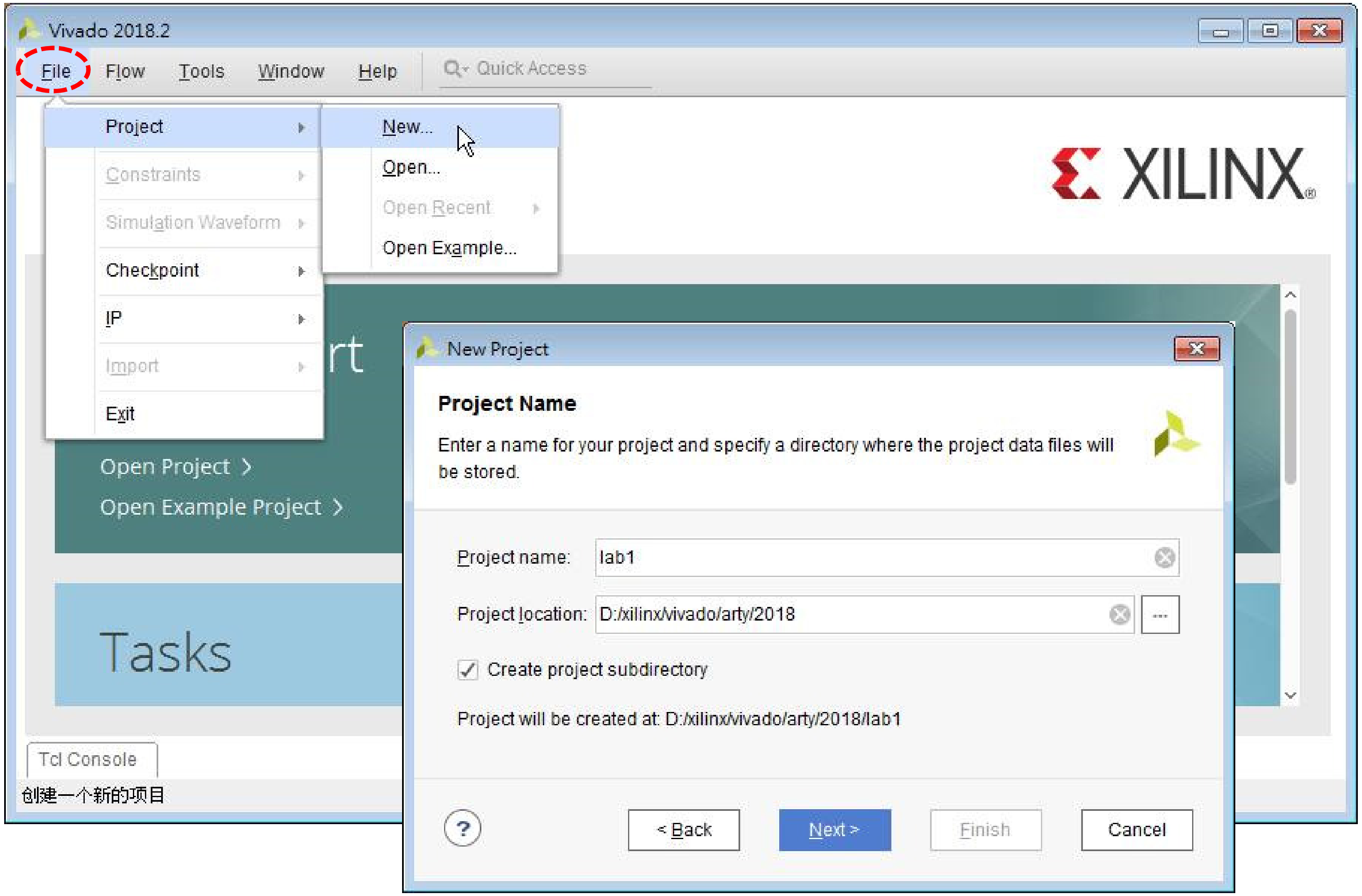
4 個項目

Launch Vivado 2018.2

❑ Double-click the Vivado 2018.2 icon on the desktop:



Create a New Project in Vivado



Project Name

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

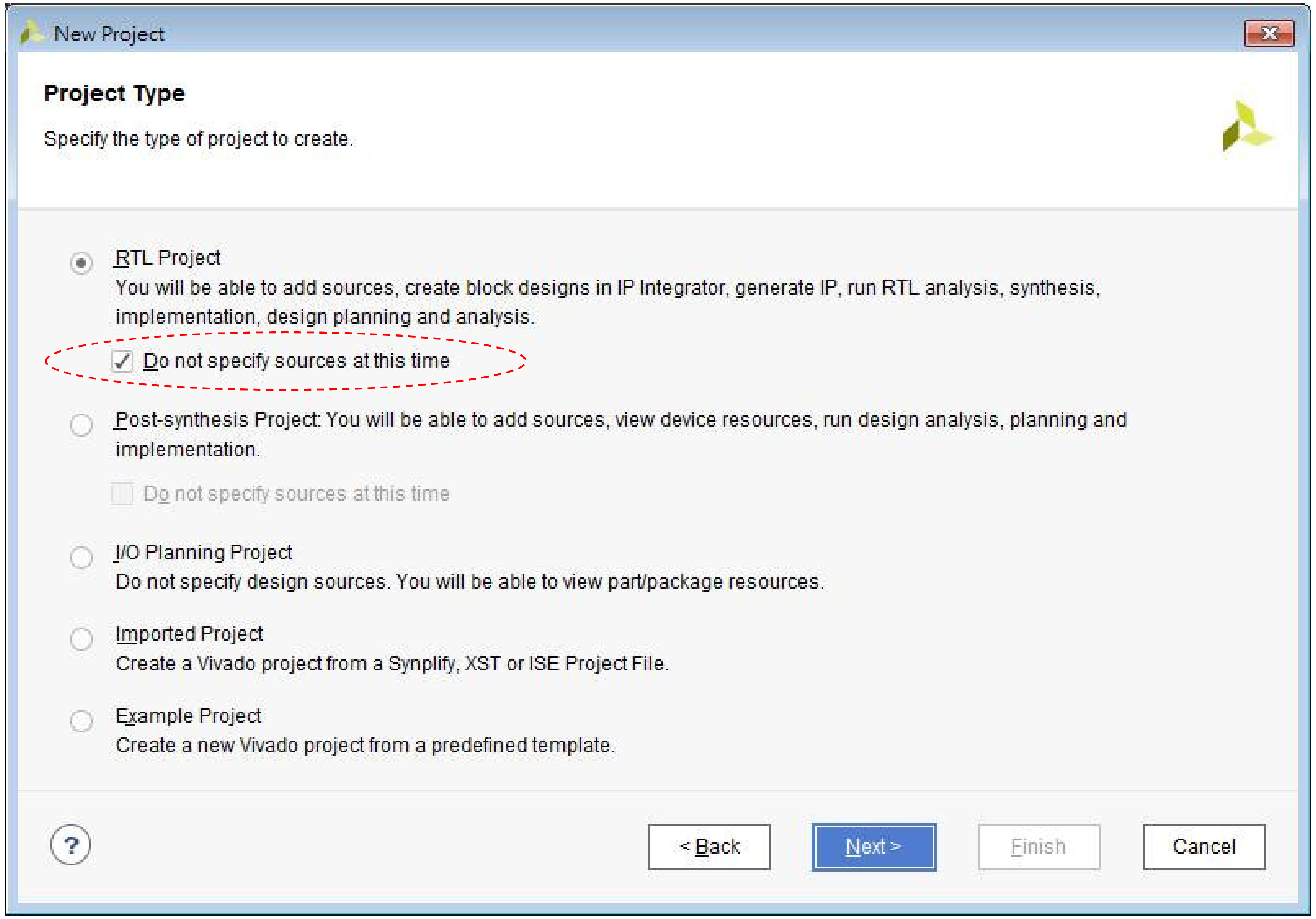
Project name: lab1

Project location: D:/xilinx/vivado/arty/2018

☑ Create project subdirectory

Project will be created at: D:/xilinx/vivado/arty/2018/lab1

Select Project Type



New Project

Project Type

Specify the type of project to create.

(●) RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

[✓] Do not specify sources at this time

( ) Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.

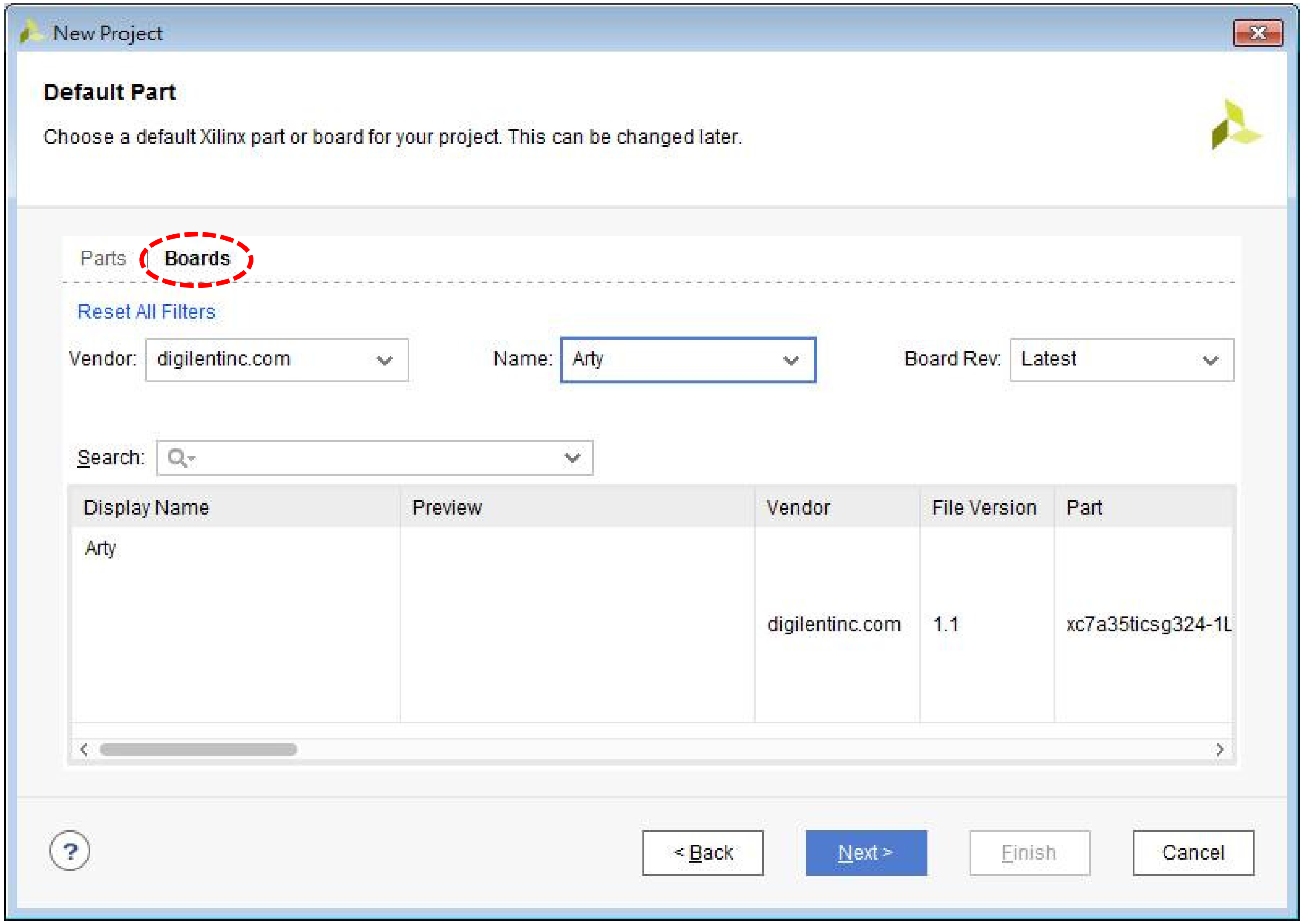
[ ] Do not specify sources at this time

( ) I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

( ) Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

( ) Example Project
Create a new Vivado project from a predefined template.

Select the Target FPGA Board



New Project

Default Part

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

Vendor: digilentinc.com

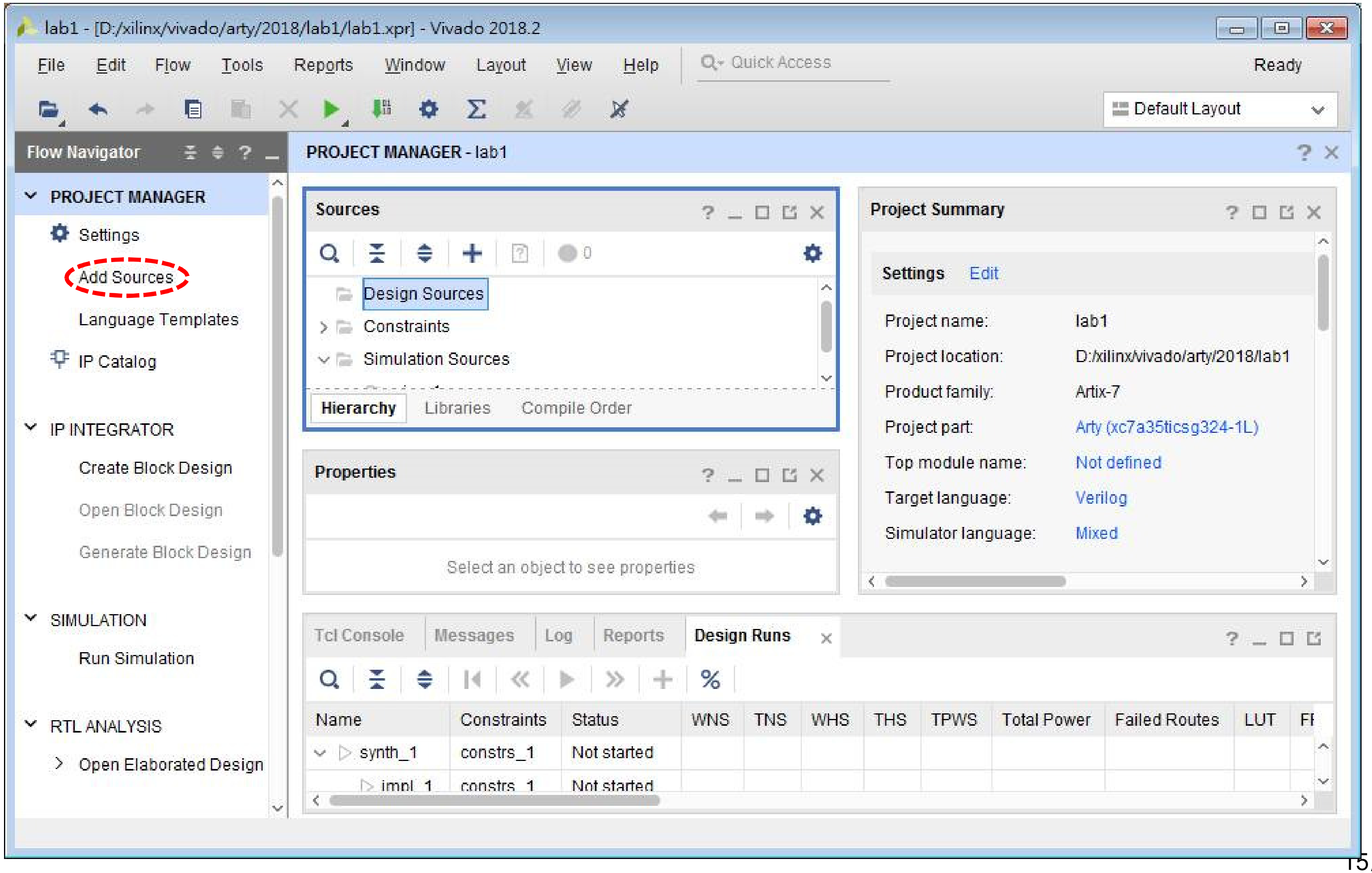
Name: Arty

Board Rev: Latest

Search:

Display Name	Preview	Vendor	File Version	Part
Arty		digilentinc.com	1.1	xc7a35ticsg324-1L

Add a New HDL Source Code



Project Summary

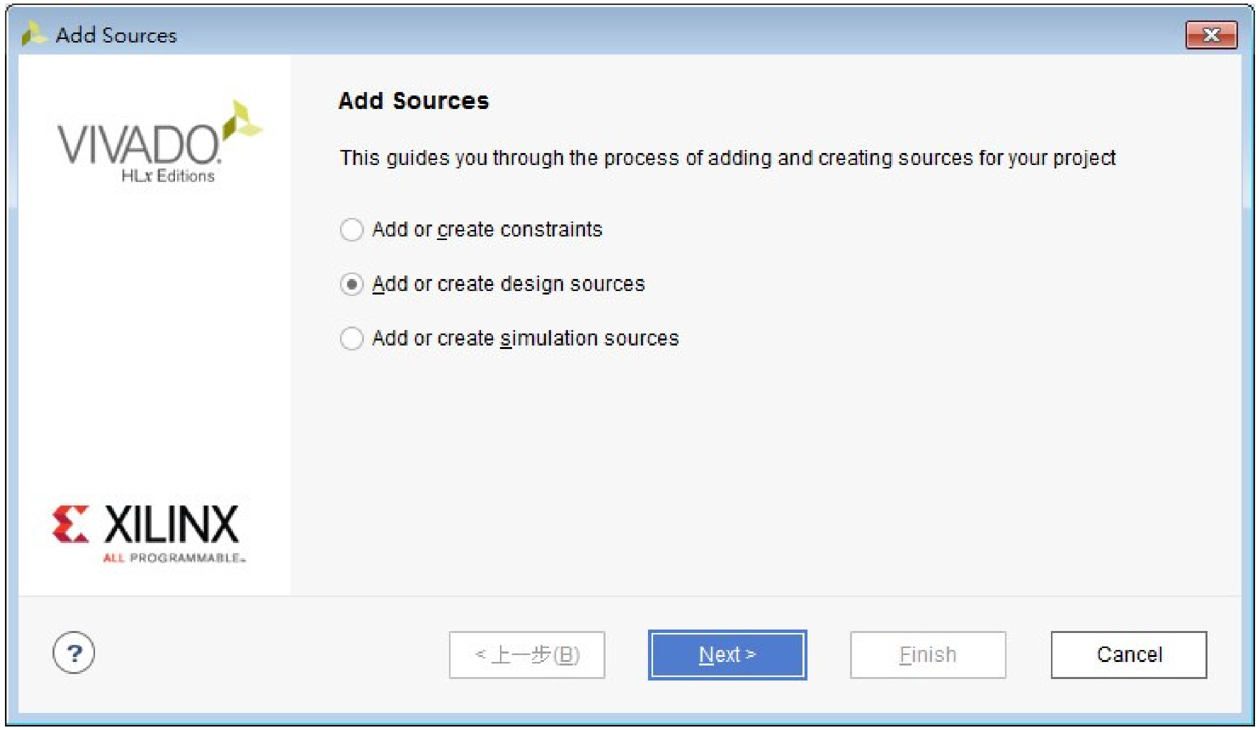
Settings Edit

Project name:	lab1
Project location:	D:/xilinx/vivado/arty/2018/lab1
Product family:	Artix-7
Project part:	Arty (xc7a35ticsg324-1L)
Top module name:	Not defined
Target language:	Verilog
Simulator language:	Mixed

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF
synth_1	constrs_1	Not started									
impl_1	constrs_1	Not started									

Specifying the Source Type to Create

- There are several types of source files in a circuit design project: design sources, constraint sources, simulation sources, and memory sources, etc.



Add Sources

Add Sources

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

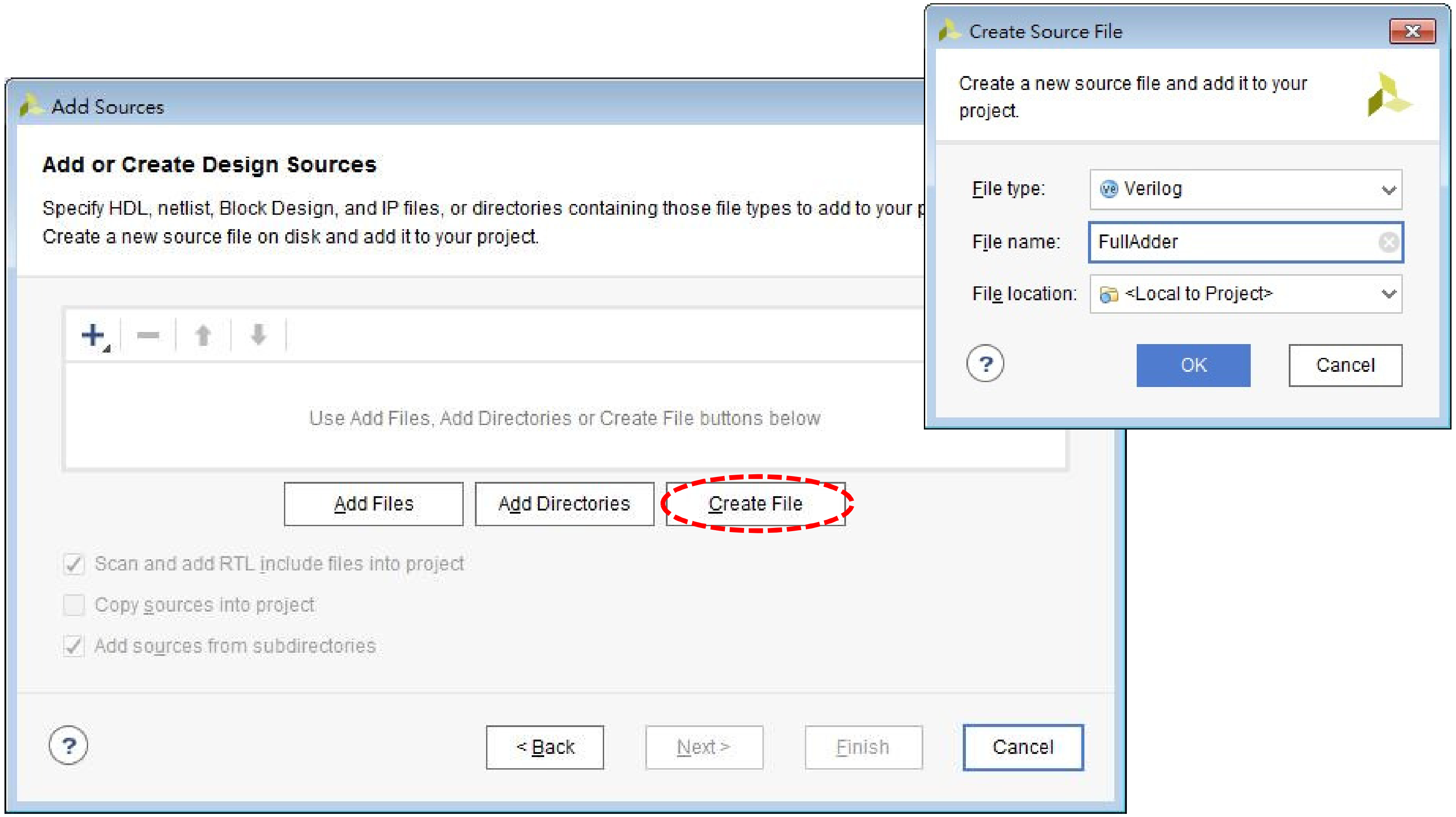
( ) Add or create constraints

(●) Add or create design sources

( ) Add or create simulation sources

Create a 4-bit Full Adder Design

- Let's create a design source from scratch!



Add Sources

Add or Create Design Sources

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

Use Add Files, Add Directories or Create File buttons below

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

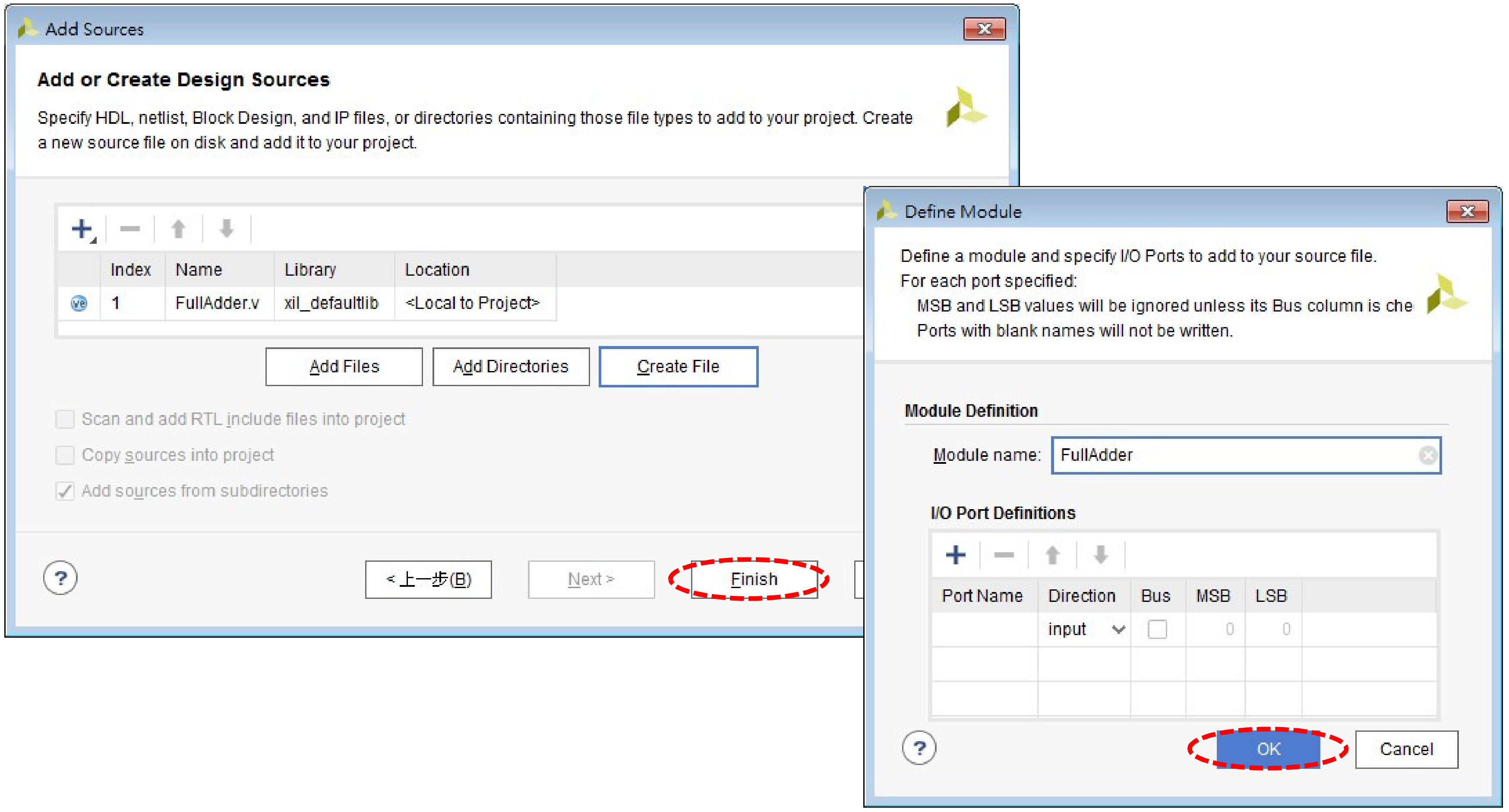
Create Source File

Create a new source file and add it to your project.

File type: Verilog
File name: FullAdder
File location: <Local to Project>

Confirm to Create the Verilog Module

❑ You can define your ports here or do it in the HDL code:


Add Sources

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	FullAdder.v	xil_defaultlib	<Local to Project>

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

Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is che
Ports with blank names will not be written.

Module Definition

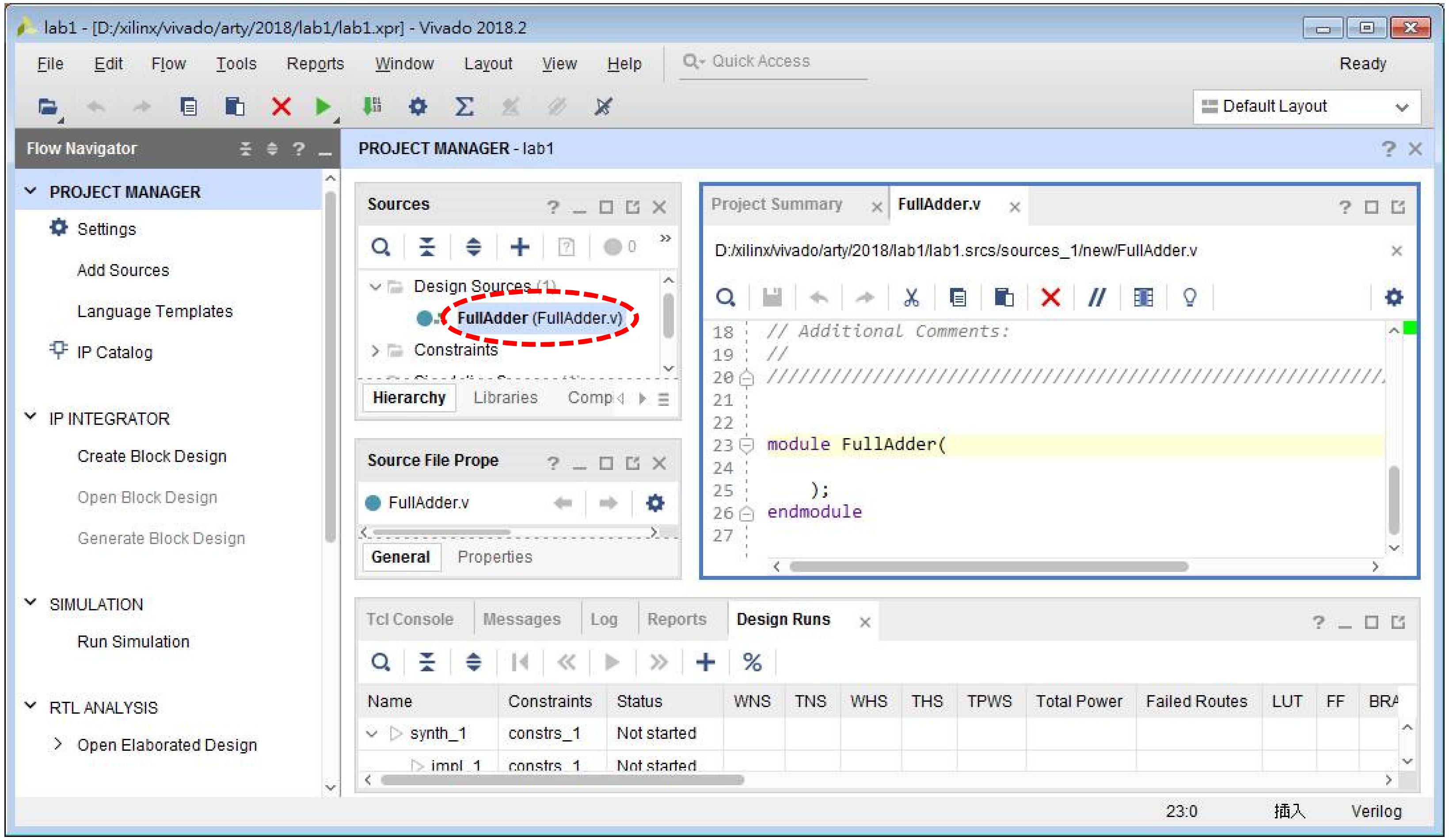
Module name: FullAdder

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
	input	☐	0	0

An Empty Verilog Template is Created

- You can type in your Verilog code in the editor window:



lab1 - [D:/xilinx/vivado/arty/2018/lab1/lab1.xpr] - Vivado 2018.2

D:/xilinx/vivado/arty/2018/lab1/lab1.srcs/sources_1/new/FullAdder.v

```
// Additional Comments:
//
//////////////////////////////////////////////////////////////////


module FullAdder(

    );
endmodule
```

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRA
synth_1	constrs_1	Not started										
impl_1	constrs_1	Not started										

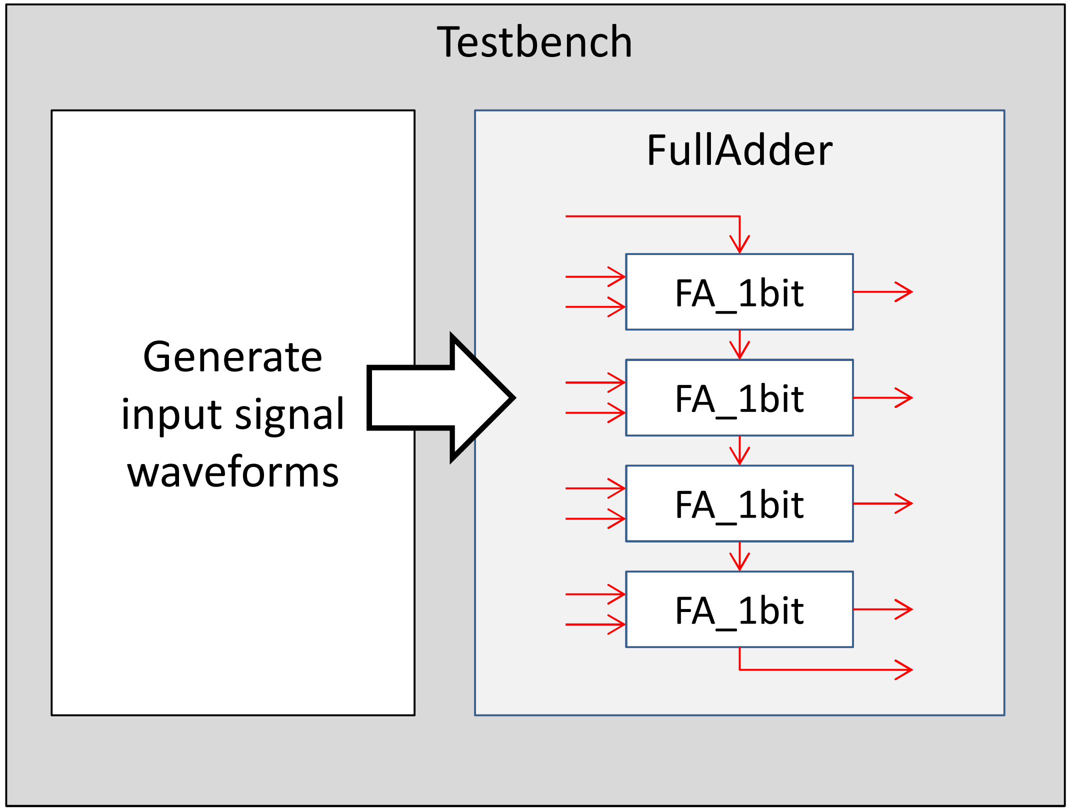
Type in the HDL Source Code

- The complete code for a 4-bit full adder is as follows:

```
// ------------- A four-bit full adder -------------------------------
module FullAdder(A, B, Cin, S, Cout);
  input [3:0] A, B;
  input Cin;
  output [3:0] S;
  output Cout;
  wire [2:0] t;

  FA_1bit FA0(.A(A[0]), .B(B[0]), .Cin(Cin), .S(S[0]), .Cout(t[0]));
  FA_1bit FA1(.A(A[1]), .B(B[1]), .Cin(t[0]), .S(S[1]), .Cout(t[1]));
  FA_1bit FA2(.A(A[2]), .B(B[2]), .Cin(t[1]), .S(S[2]), .Cout(t[2]));
  FA_1bit FA3(.A(A[3]), .B(B[3]), .Cin(t[2]), .S(S[3]), .Cout(Cout));
endmodule

// ----------------- A 1-bit full adder -----------------------------
module FA_1bit(A, B, Cin, S, Cout);
  input A, B, Cin;
  output S, Cout;

  assign S = Cin ^ A ^ B;
  assign Cout = (A & B) | (Cin & B) | (Cin & A);
endmodule
```

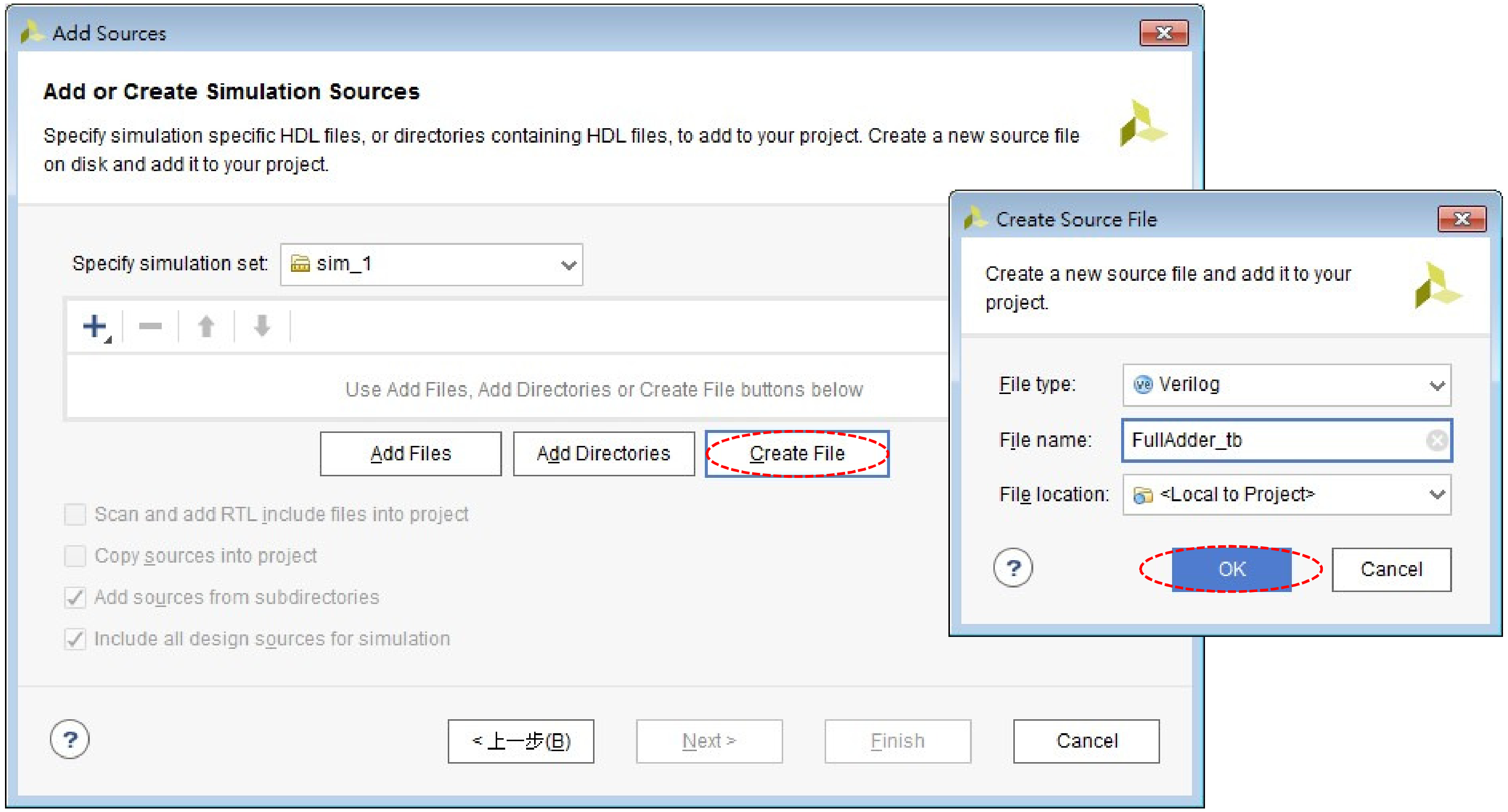
TestBench Design

- You must create a testbench to generate input signals that can feed into your circuit module, such that you can analyze the output to verify its correctness



Create the Testbench Source Code

❑ Click "Add Sources" button again, and this time, select "Add or create simulation sources"


Add Sources

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: sim_1

Use Add Files, Add Directories or Create File buttons below

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

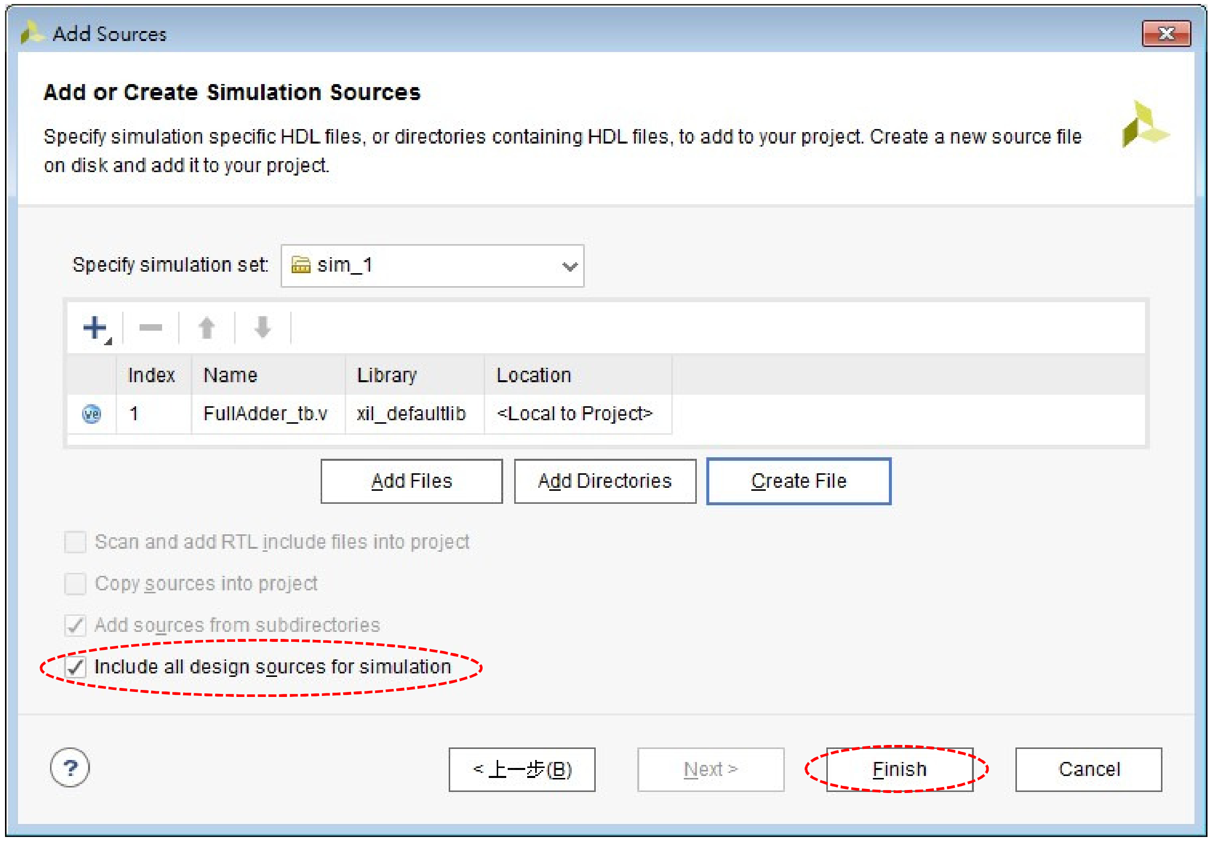
Create Source File

Create a new source file and add it to your project.

File type: Verilog
File name: FullAdder_tb
File location: <Local to Project>

Confirm the Creation of the Testbench

- Here, we include the design sources into the simulation set so that we can test the modules under development



Add Sources

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.

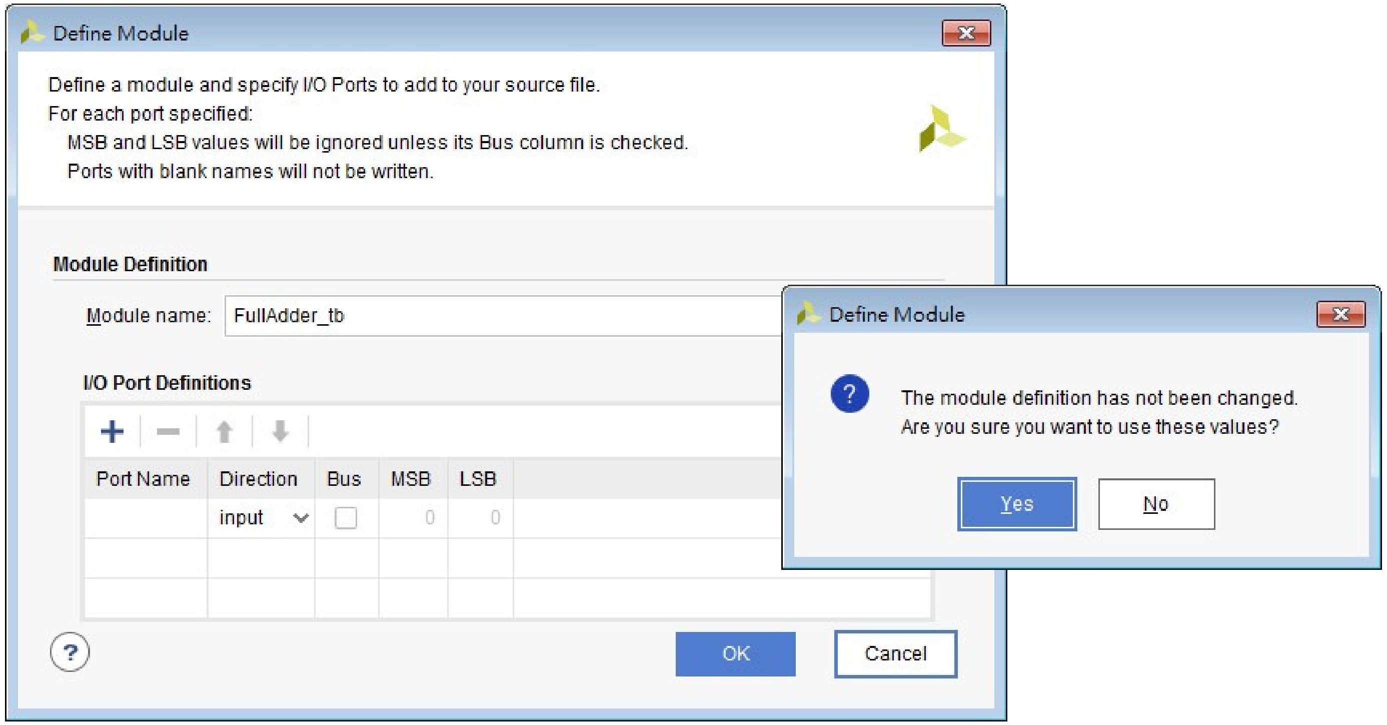
Specify simulation set: sim_1

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

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

Create the Testbench Template

- Hit “OK” then “Yes” to create an empty testbench template $\rightarrow$ top-level of the testbench template usually has no I/O ports



Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Module name: FullAdder_tb

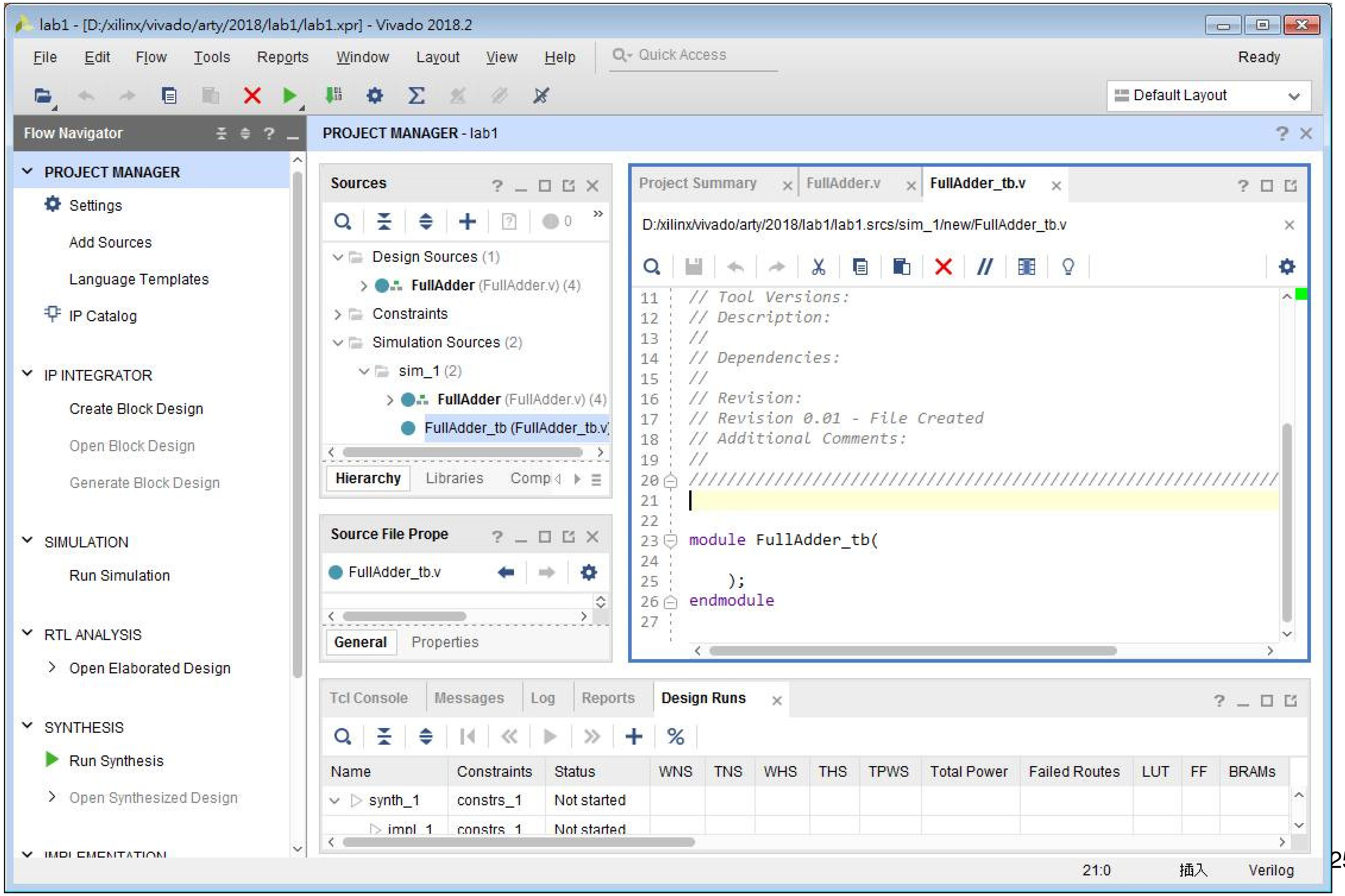
I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
	input	☐	0	0

Define Module

The module definition has not been changed.
Are you sure you want to use these values?

Type in the Testbench Source Code



lab1 - [D:/xilinx/vivado/arty/2018/lab1/lab1.xpr] - Vivado 2018.2

File Edit Flow Tools Reports Window Layout View Help Quick Access Ready

Default Layout

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

PROJECT MANAGER - lab1

Sources

- Design Sources (1)
  - FullAdder (FullAdder.v) (4)
- Constraints
- Simulation Sources (2)
  - sim_1 (2)
    - FullAdder (FullAdder.v) (4)
    - FullAdder_tb (FullAdder_tb.v)

Hierarchy Libraries Comp

Source File Prope

FullAdder_tb.v

General Properties

Project Summary | FullAdder.v | FullAdder_tb.v

D:/xilinx/vivado/arty/2018/lab1/lab1.srcs/sim_1/new/FullAdder_tb.v

```
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module FullAdder_tb(

    );
endmodule

```

Tcl Console | Messages | Log | Reports | Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs
synth_1	constrs_1	Not started										
impl_1	constrs_1	Not started										

21:0 插入 Verilog

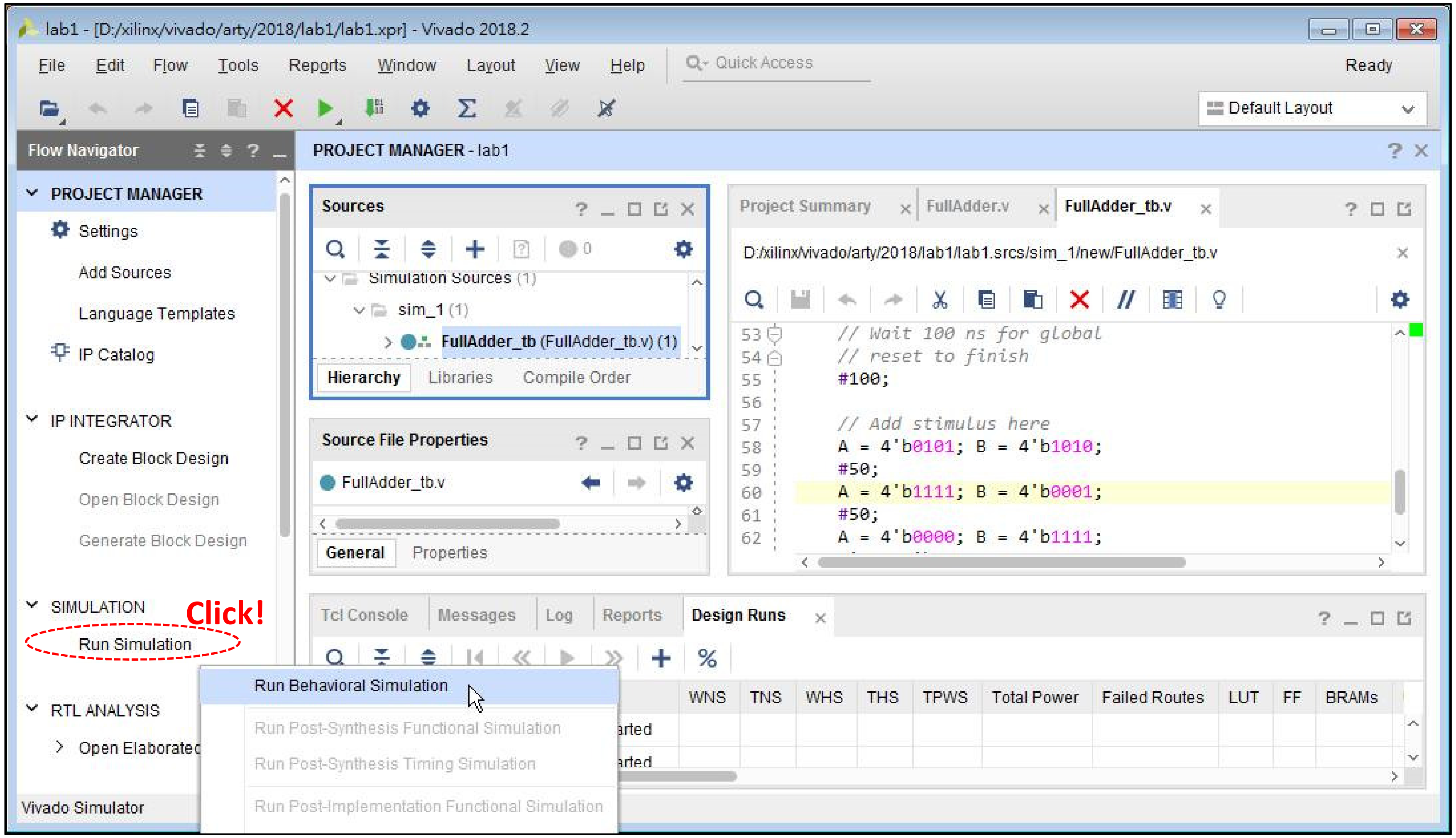
The Sample Testbench Code

- The template created by Vivado is an empty module; you must add test pattern generators in the module

```
module FullAdder_tb;

 // inputs
  reg clk = 1;
  reg [3:0] A, B;
  reg Cin;

  // outputs
  wire [3:0] S;
  wire Cout;

  // Instantiate the Unit
  // Under Test (UUT)

  FullAdder uut(
    .A(A),
    .B(B),
    .Cin(Cin),
    .S(S),
    .Cout(Cout)
  );
```

```
  // 100MHz clock generator
  always
    #5 clk = !clk;

  initial begin
    // Initialize Inputs
    A = 0; B = 0; Cin = 0;

    // Wait 100 ns for global
    // reset to finish
    #100;

    // Add stimulus here
    A = 4'b0101; B = 4'b1010;
    #50;
    A = 4'b1111; B = 4'b0001;
    #50;
    A = 4'b0000; B = 4'b1111;
    Cin = 1'b1;
    #50;
    A = 4'b0110; B = 4'b0001;
  end
endmodule
```

Run the Simulation



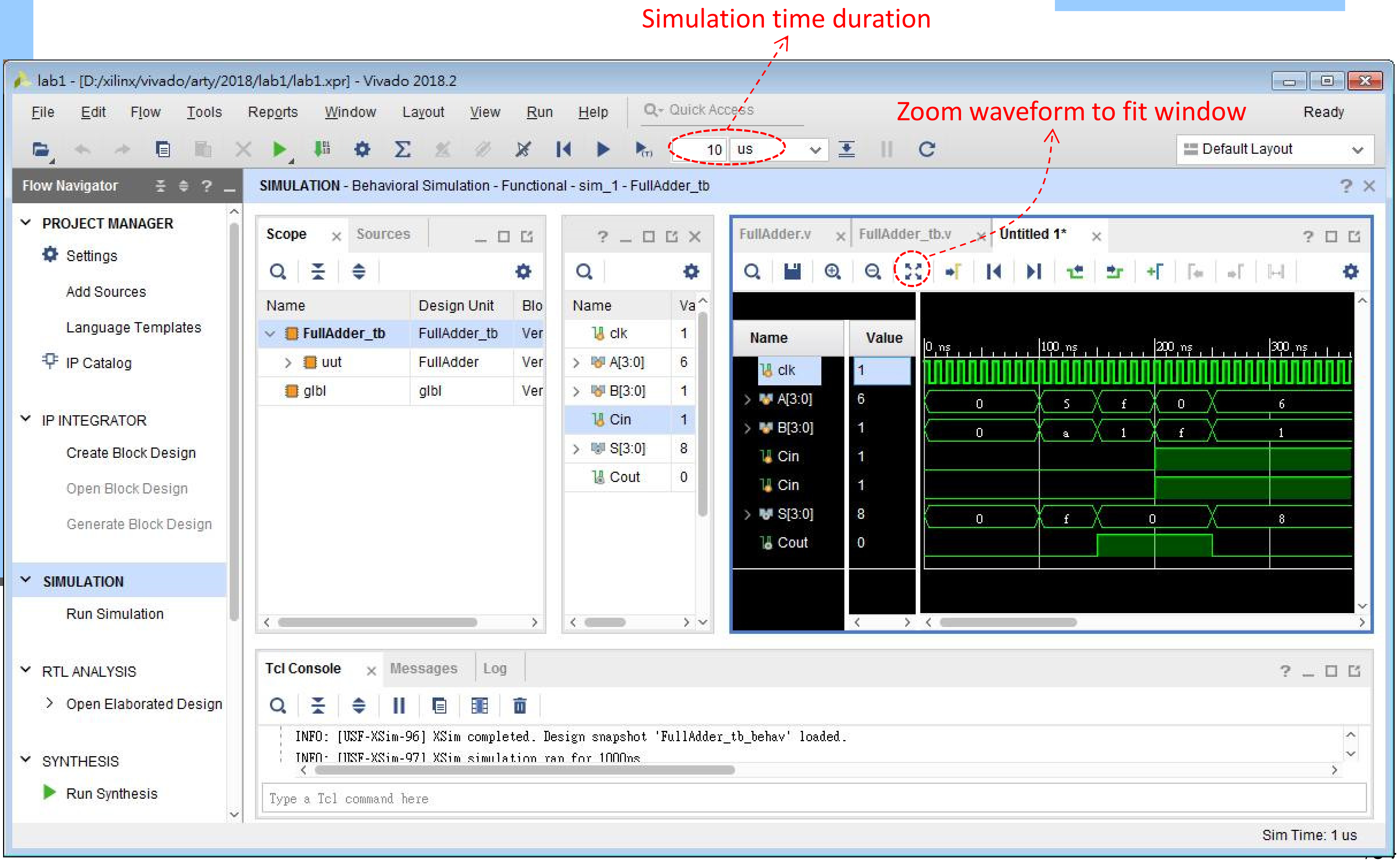
lab1 - [D:/xilinx/vivado/arty/2018/lab1/lab1.xpr] - Vivado 2018.2

File Edit Flow Tools Reports Window Layout View Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Click!
Run Simulation
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
RTL ANALYSIS
Open Elaborated
Vivado Simulator
PROJECT MANAGER - lab1
Sources
Simulation Sources (1)
sim_1 (1)
FullAdder_tb (FullAdder_tb.v) (1)
Hierarchy Libraries Compile Order
Source File Properties
FullAdder_tb.v
General Properties
Project Summary FullAdder.v FullAdder_tb.v
Tcl Console Messages Log Reports Design Runs
WNS TNS WHS THS TPWS Total Power Failed Routes LUT FF BRAMs

D:/xilinx/vivado/arty/2018/lab1/lab1.srcs/sim_1/new/FullAdder_tb.v

```
    // Wait 100 ns for global
    // reset to finish
    #100;

    // Add stimulus here
    A = 4'b0101; B = 4'b1010;
    #50;
    A = 4'b1111; B = 4'b0001;
    #50;
    A = 4'b0000; B = 4'b1111;
```

Vivado Simulator Window



Name	Design Unit	Blo
FullAdder_tb	FullAdder_tb	Ver
uut	FullAdder	Ver
glbl	glbl	Ver

Name	Va
clk	1
A[3:0]	6
B[3:0]	1
Cin	1
S[3:0]	8
Cout	0

Name	Value
clk	1
A[3:0]	6
B[3:0]	1
Cin	1
Cin	1
S[3:0]	8
Cout	0

```
INFO: [USF-XSim-96] XSim completed. Design snapshot 'FullAdder_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Type a Tcl command here

Register Transfer-Level (RTL) Coding

- For our labs, you do not have to design the circuits purely at gate-level; you can write RTL codes
- For example, the four-bit adder module can be implemented using the RTL code simply as follows:

```
// ------------- A four-bit full adder ------------------------------
module FullAdder(A, B, Cin, S, Cout);
  input [3:0] A, B;
  input Cin;
  output [3:0] S;
  output Cout;

  assign { Cout, S } = A + B + Cin;
endmodule
```

Now, Write Simulation for a Multiplier

- The input/output ports of the 8-bit multiplier is as follows:

```
module SeqMultiplier(
    input wire clk,
    input wire enable,
    input wire [7:0] A,
    input wire [7:0] B,
    output wire [15:0] C
    );
```

‘`clk`’ is the system clock,
‘`enable`’ activates the multiplication operation,
‘`A`’ is the 8-bit unsigned multiplicand input,
‘`B`’ is the 8-bit unsigned multiplier input, and
‘`C`’ is the 16-bit unsigned product output.

Sequential Binary Multiplier Behavior

- Sequential binary multiplication of 10111 and 10011:

```
      00010111  → multiplicand
×     00010011  → multiplier
──────────────────────
      00000000
       00000000
        00000000
         00010111
          00000000
           00000000
            00010111
+            00010111
──────────────────────
         000110110101  → product
```

C Model of the Multiplier

- The C code that performs the sequential multiplication:

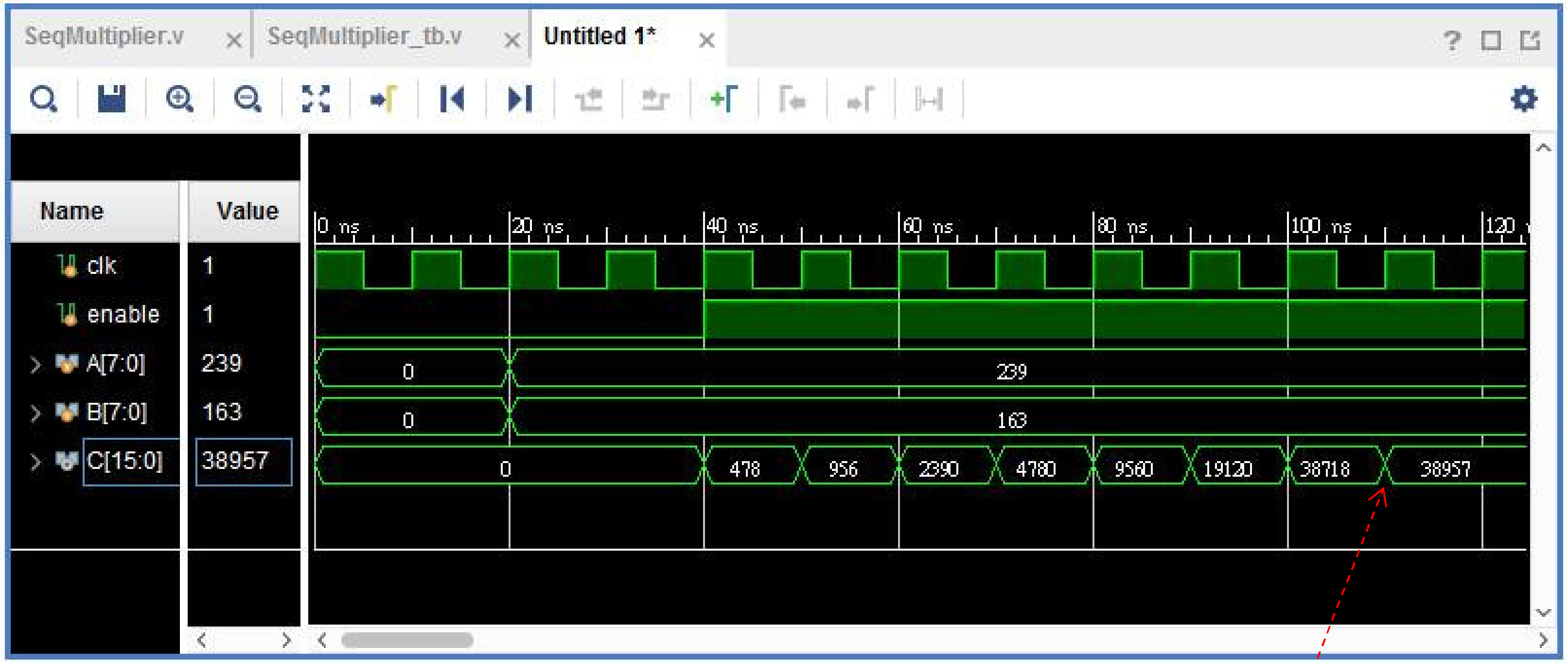
```
typedef unsigned char Byte;
typedef unsigned short Word;

SeqMultiplier(Byte A, Byte B, Word *C)
{
    int idx;

    *C = 0;
    for (idx = 8; idx != 0; idx--)
    {
        *C = *C << 1;
        if ((B & 0x80) == 0x80)
        {
            *C += A;
        }
        B = B << 1;
    }
}
```

Verilog Module of the Multiplier

```
module SeqMultiplier(input wire clk, input wire enable,
    input wire [7:0] A, input wire [7:0] B,
    output wire [15:0] C);

reg  [15:0] prod;
reg  [7:0]  mult;
reg  [3:0]  counter;
wire        shift;

assign C = prod;
assign shift = |(counter^7);

always @(posedge clk) begin
  if (!enable) begin
    mult <= B;
    prod <= 0;
    counter <= 0;
  end
  else begin
    mult <= mult << 1;
    prod <= (prod + (A & {8{mult[7]}})) << shift;
    counter <= counter + shift;
  end
end

endmodule
```

Example of Simulated Waveforms

- An example of the timing diagram of 239×163 = 38957



Stable output 8365 happens at the 8th cycle after enable == 1