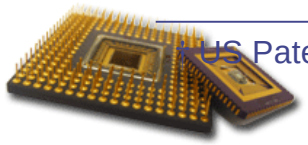
Introduction to Vivado

Lan-Da Van and Chun-Jen Tsai
Department of Computer Science
National Yang Ming Chiao Tung University
Taiwan, R.O.C.

Fall, 2024

Target Technology of Digital Labs

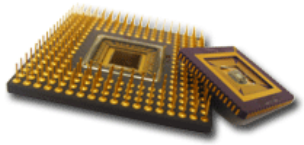
- Digital circuits can be implemented in different ways.
  - Circuits Boards
    - Circuit board design using standard IC parts (e.g., 74SLxx)
  - Application Specific ICs
    - Full-custom and Cell-based IC designs
  - Programmable logics
    - Field Programmable Gate Array (FPGA) design
- Here, we 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 1985†.



† US Patent 4,870,302

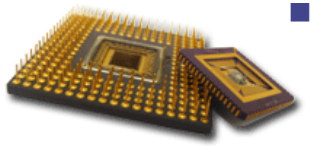
Xilinx Vivado Design Suite

- 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.



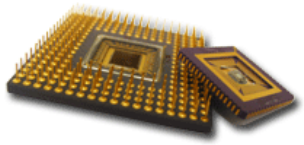
Vivado Circuit Implementation Flow

- 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.

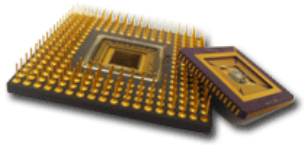


Install Your Own Vivado Design Suite

- Install a copy of Vivado ML Edition - 2024.1 Full Product Installation onto your computer.
  - You can download it from:

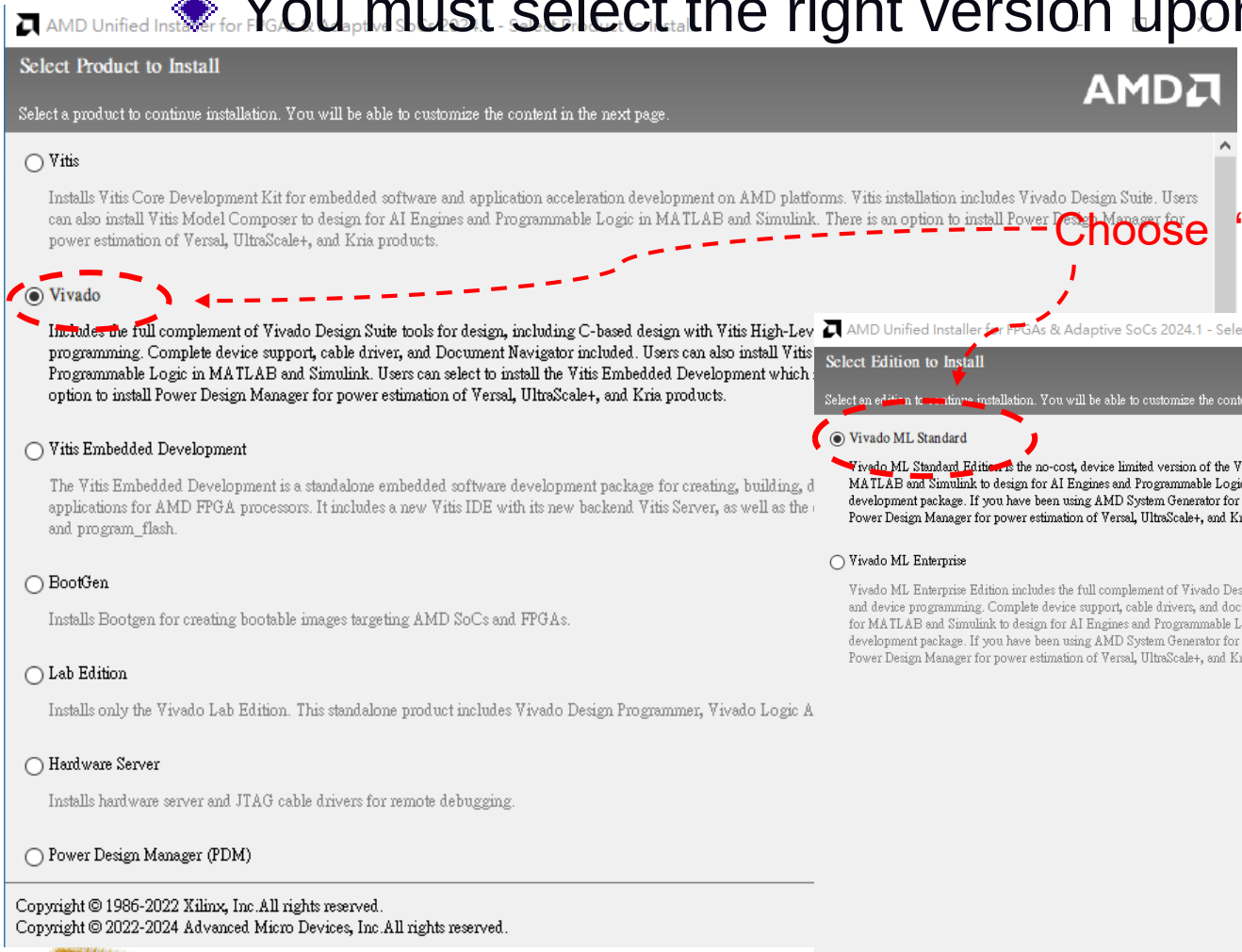
    https://www.Xilinx.com/support/download.html

- The installation requires about 60 GB of disk space.
  - Please install the “Web Installer” version and register online for a free license.

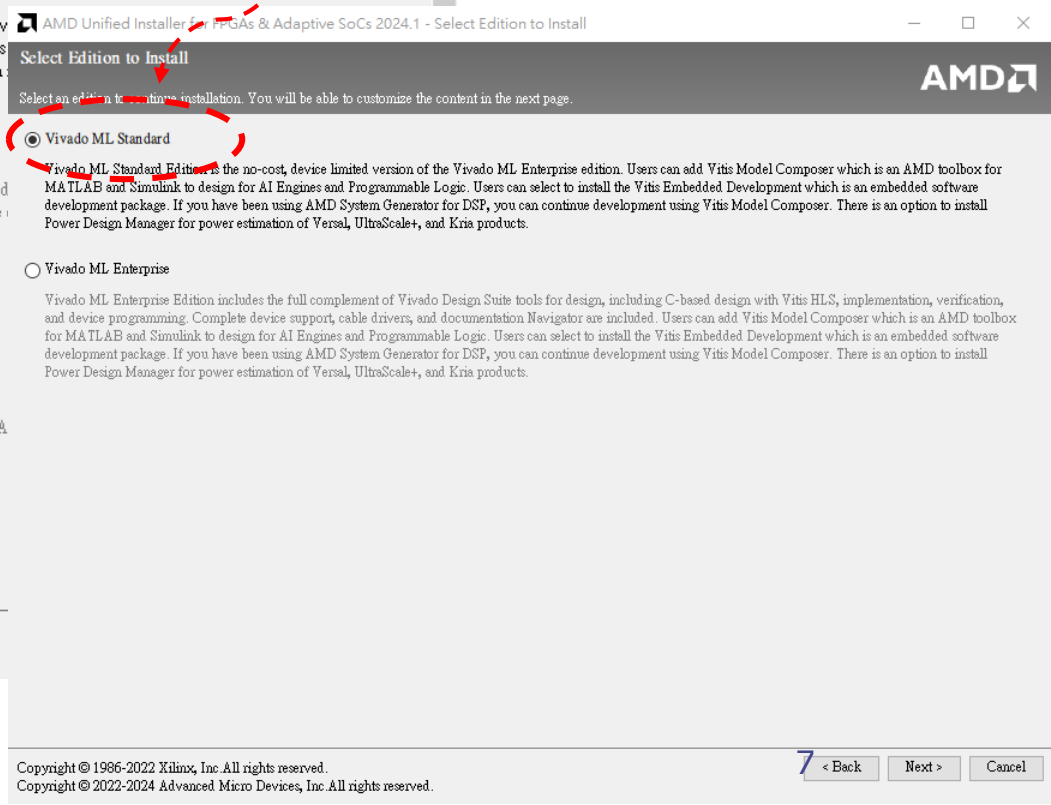


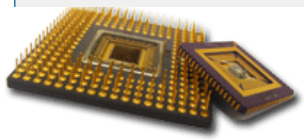
Vivado Installation Guide (1/2)

- You must select the right version upon installation:



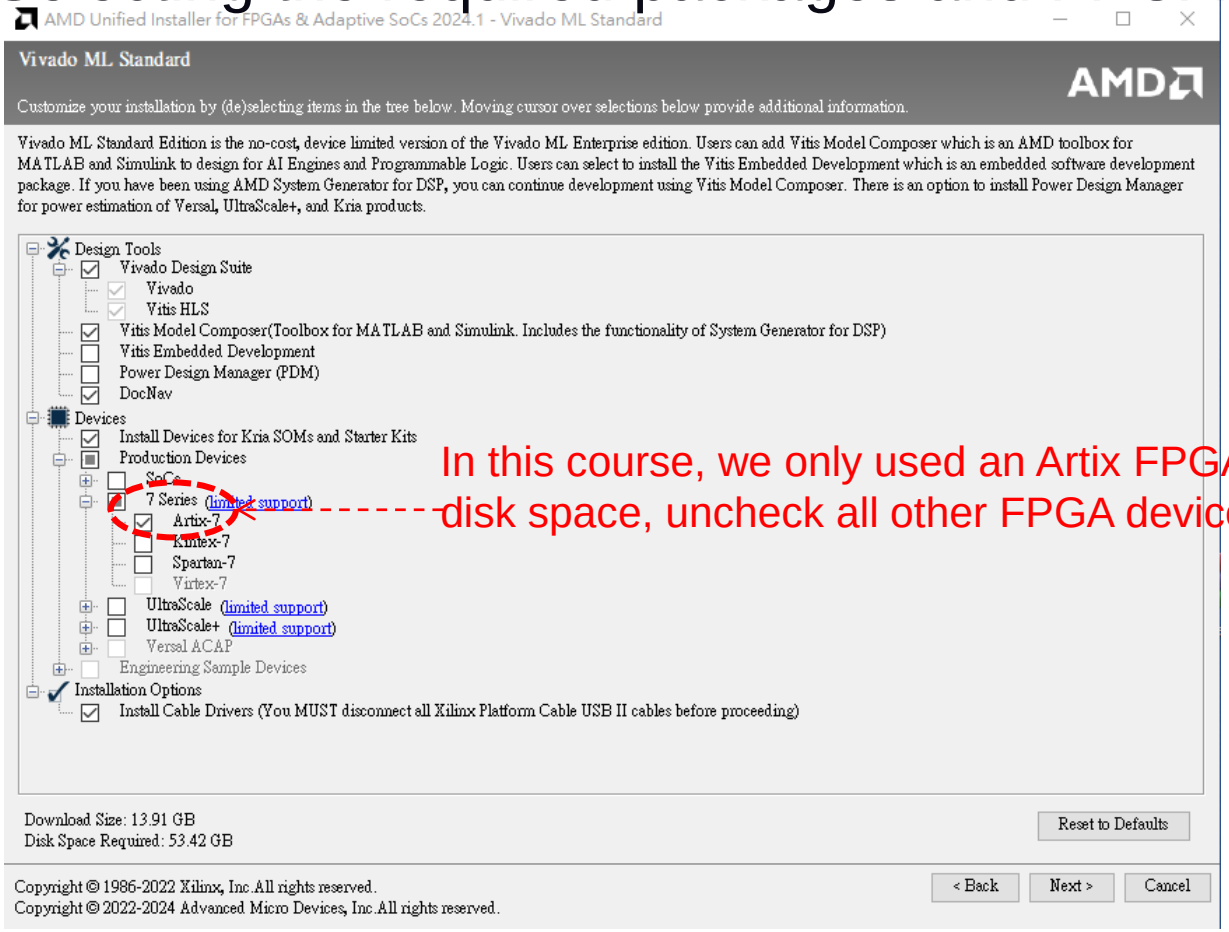
Choose “Vivado” and “ML Standard”



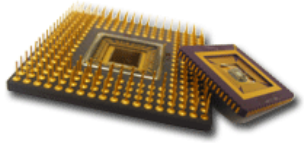


Vivado Installation Guide (2/2)

- Selecting the required packages and FPGA devices:



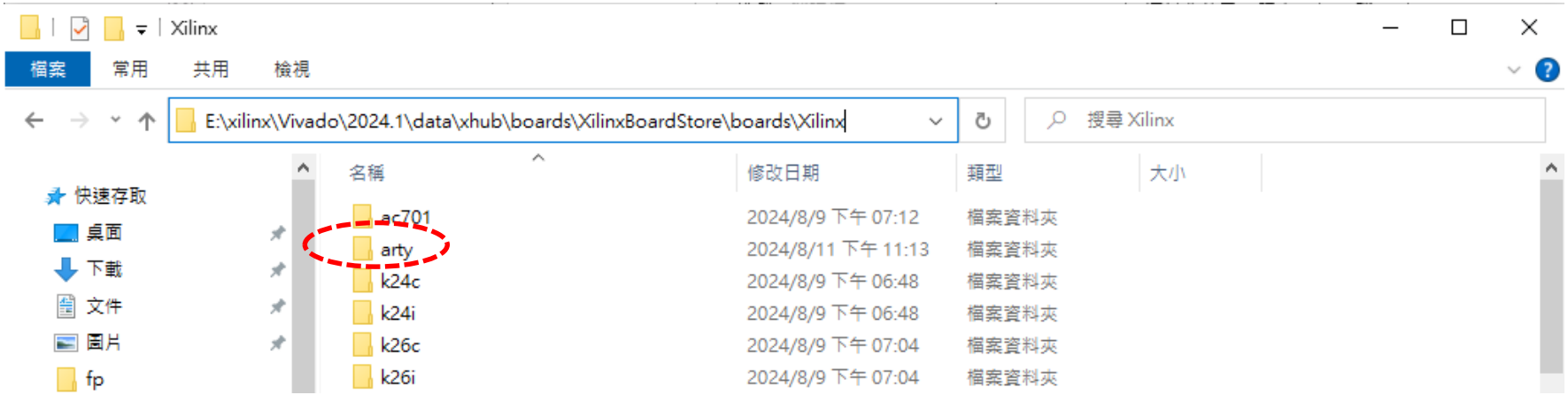
In this course, we only used an Artix FPGA. To save disk space, uncheck all other FPGA devices.



Installation of Arty Board Definitions

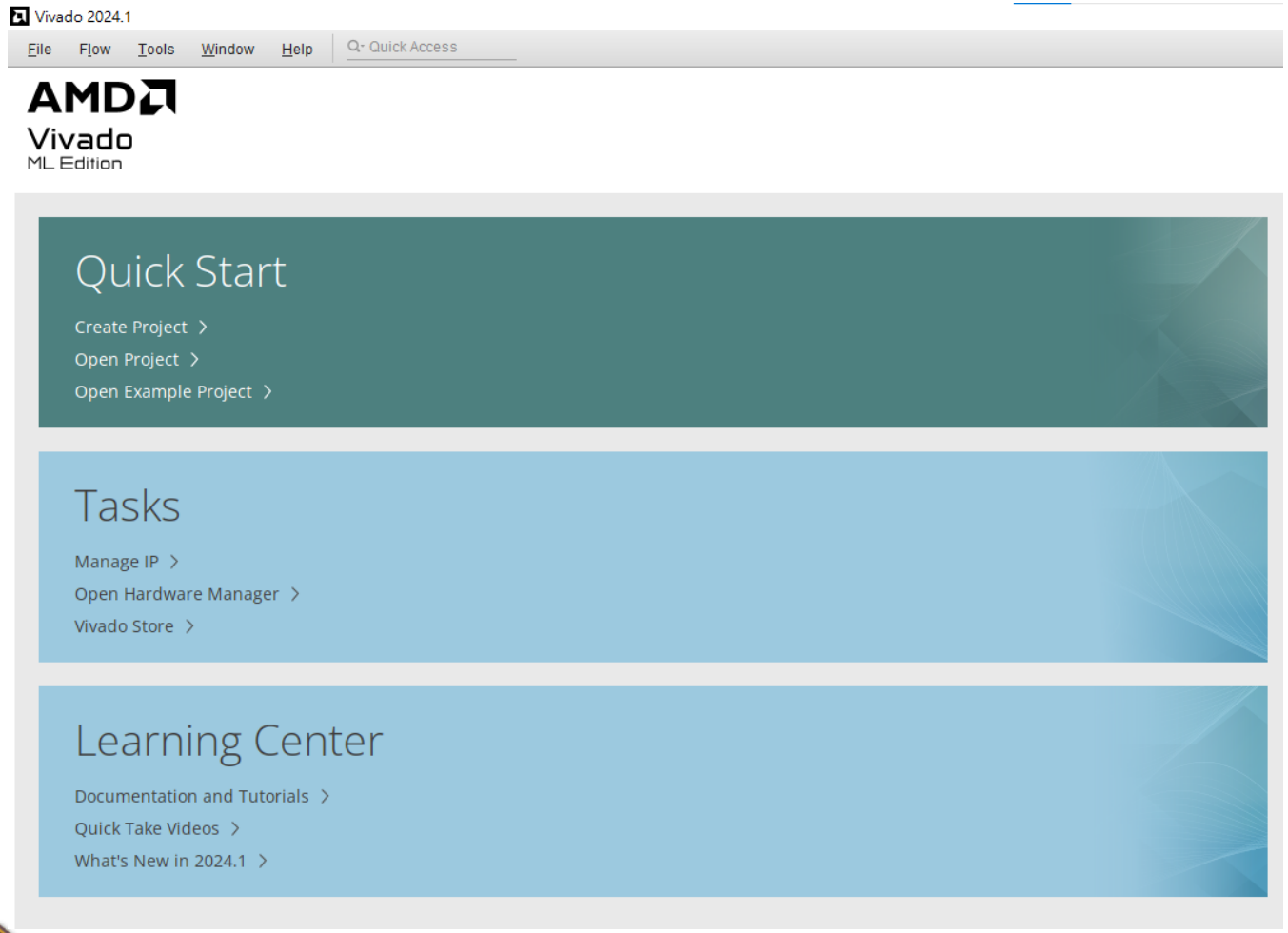
- After the installation of Vivado, you must install the board definition file of Arty:
  - Download arty.zip from E3.
  - Unzip arty.zip to the following directory:

C:/<INST_DIR>/Vivado/2024.1/data/xhub/boards/XilinxBoardStore/boards/Xilinx

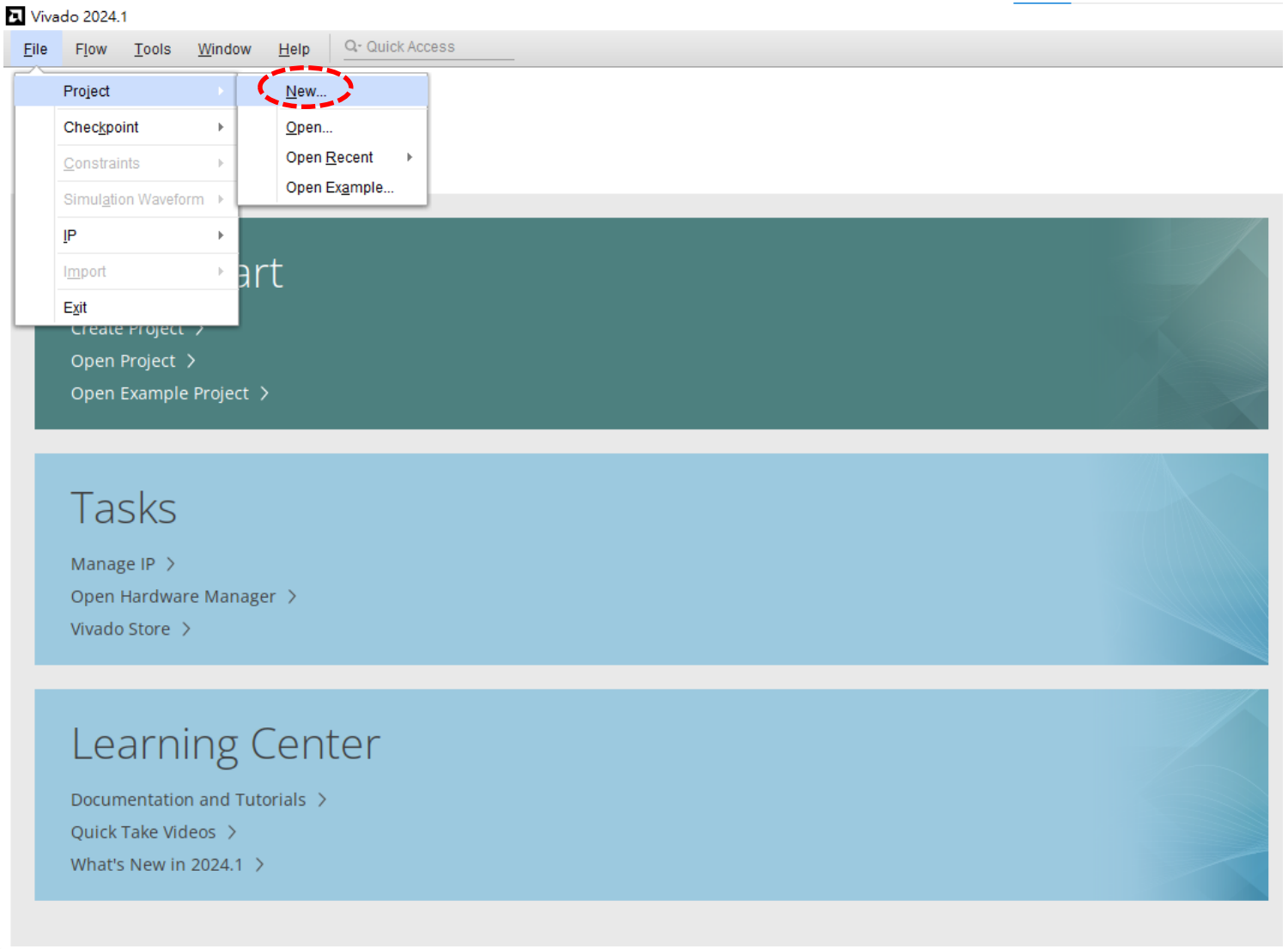


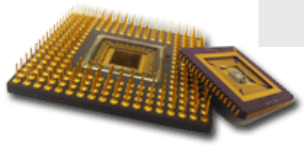
Launch Vivado 2024.1

- Double-click the Vivado 2024.1 icon on the desktop:



Create a New Project in Vivado





Select Project Type

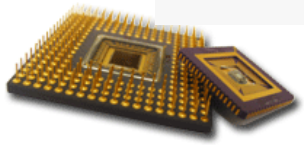
New Project

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.
  - [x] Do not specify sources at this time
  - [ ] Project is an extensible Vitis platform
- ( ) 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 Project File.
- ( ) Example Project
  Create a new Vivado project from a predefined template.

< Back | Next > | Finish | Cancel



Select the Target FPGA Board

New Project

Default Part

Choose a default AMD part or board for your project.

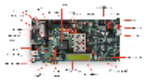
Parts | **Boards**

To fetch the latest available boards from git repository, click on 'Refresh' button. Dismiss

Reset All Filters

Vendor: All Name: All Board Rev: Latest

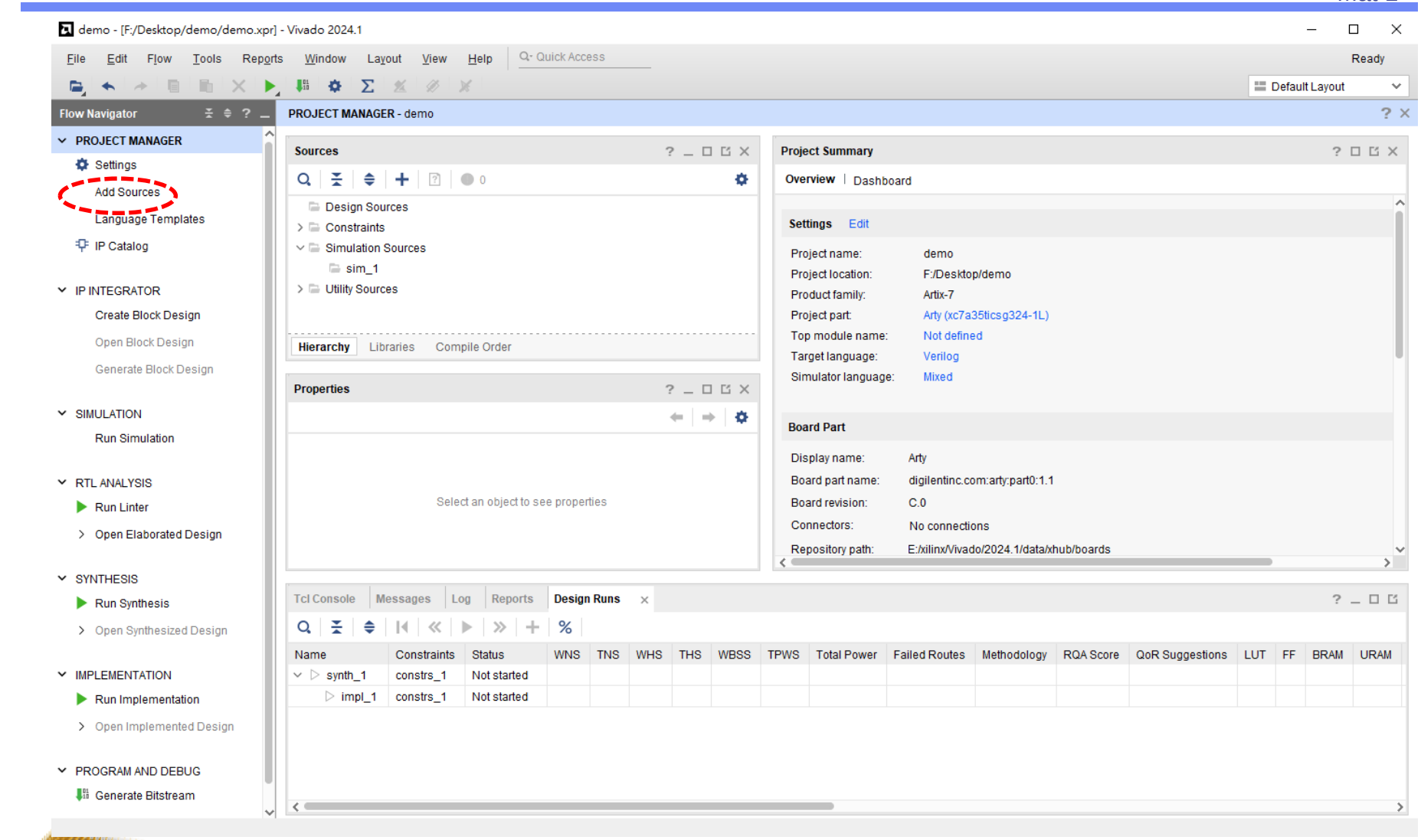
Search:

Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Count	Board Re
Arty		Installed	digilentinc.com	1.1	xc7a35ticsg324-1L	324	C.0
Artix-7 AC701 Evaluation Platform Add Companion Card Connections		Installed	xilinx.com	1.4	xc7a200tfbg676-2	676	1.1
Kria K24C SOM Add Companion Card Connections		Installed	xilinx.com	1.0	Commercial temperature grade K24 SOM	530	Rev_A01

Refresh

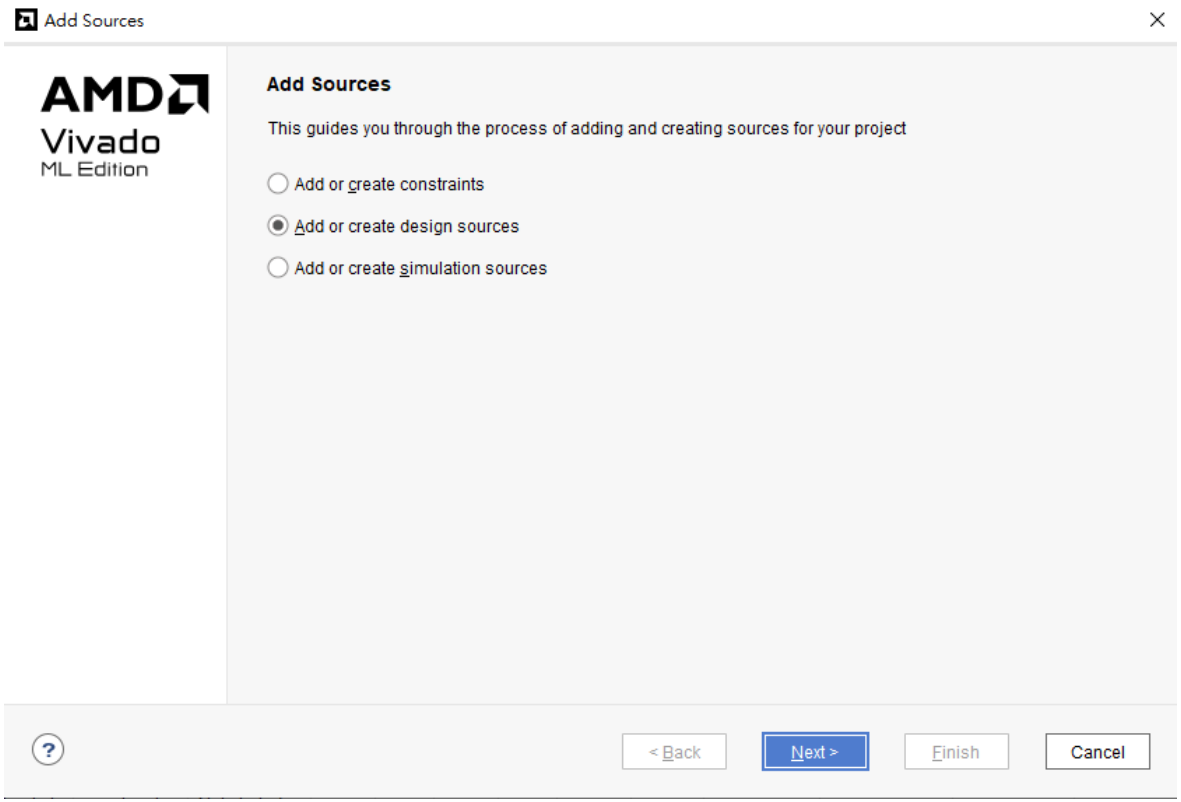
< Back Next > Finish Cancel

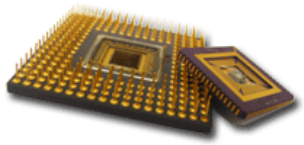
Add a New HDL Source Code



Specifying the Source Type to Create

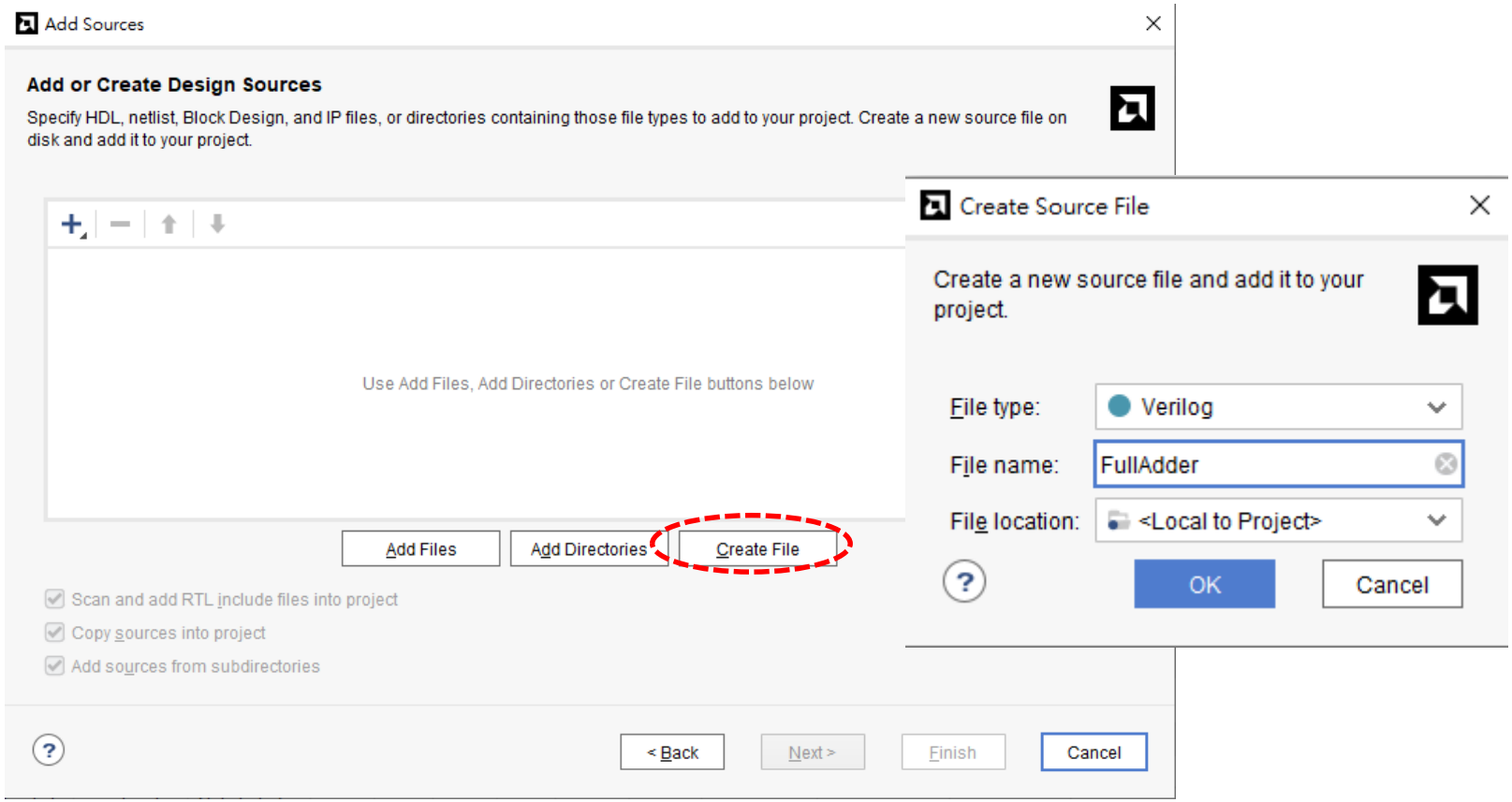
- There are several types of source files in a circuit design project: design sources, constraint sources, simulation sources, and memory sources, etc.





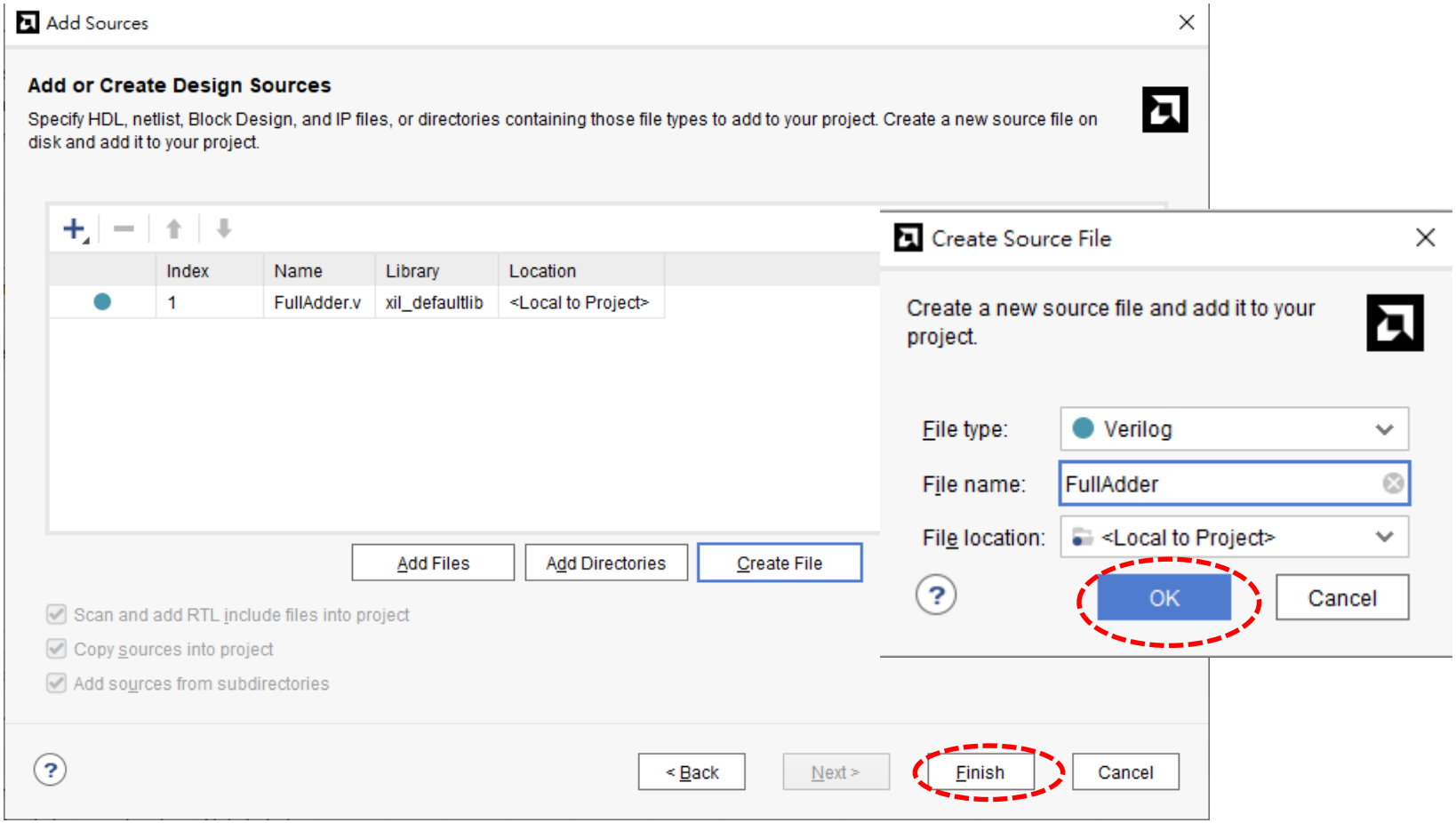
Create a 4-bit Full Adder Design

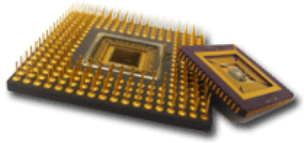
- Let's create a design source from scratch!



Confirm to Create the Verilog Module

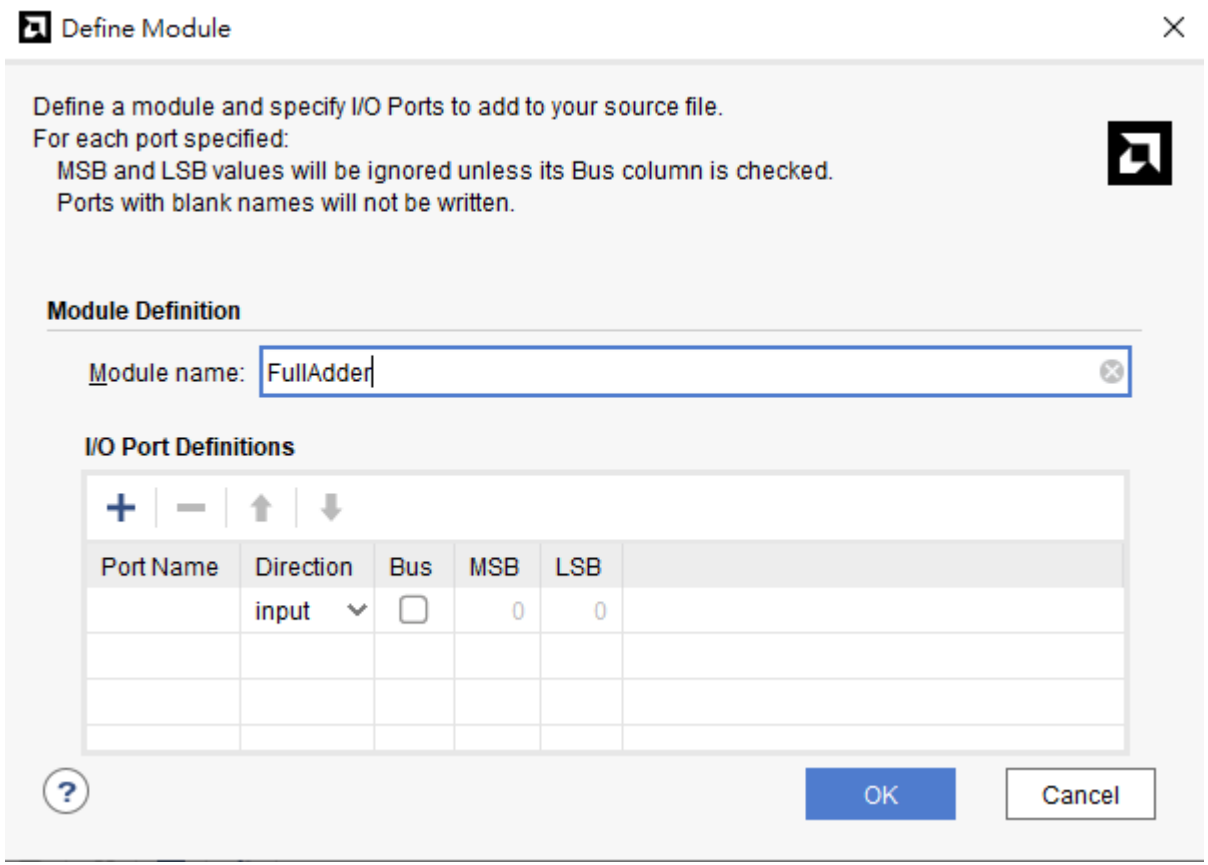
- Let's create a design source from scratch!

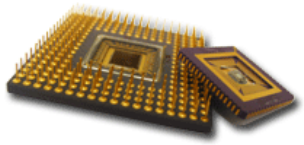




Confirm to Create the Verilog Module

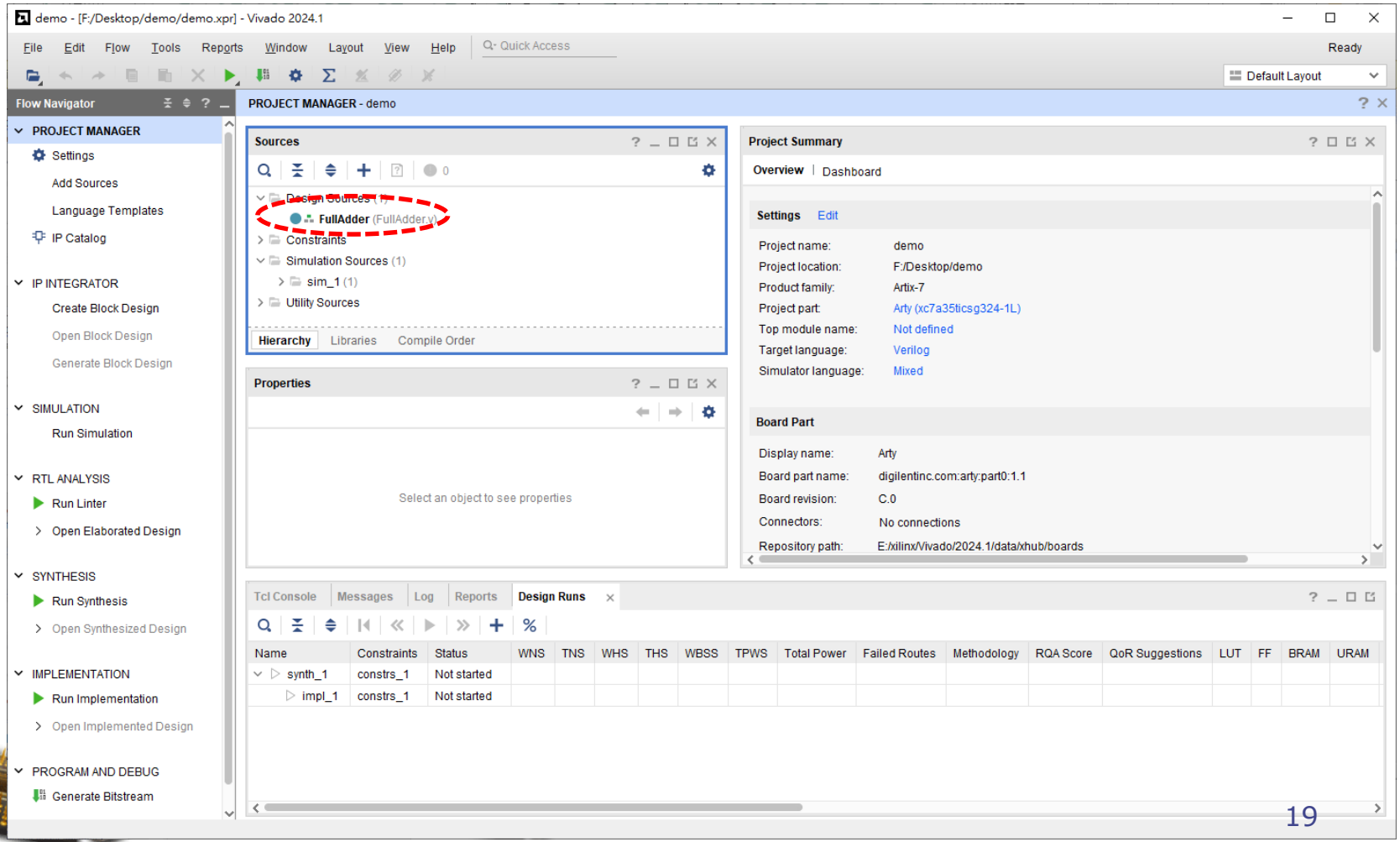
- You can define your ports here or do it in the HDL code:





An Empty Verilog Template is Created

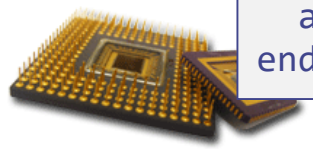
- You can type in your Verilog code in the editor window:



Type in the HDL Source Code

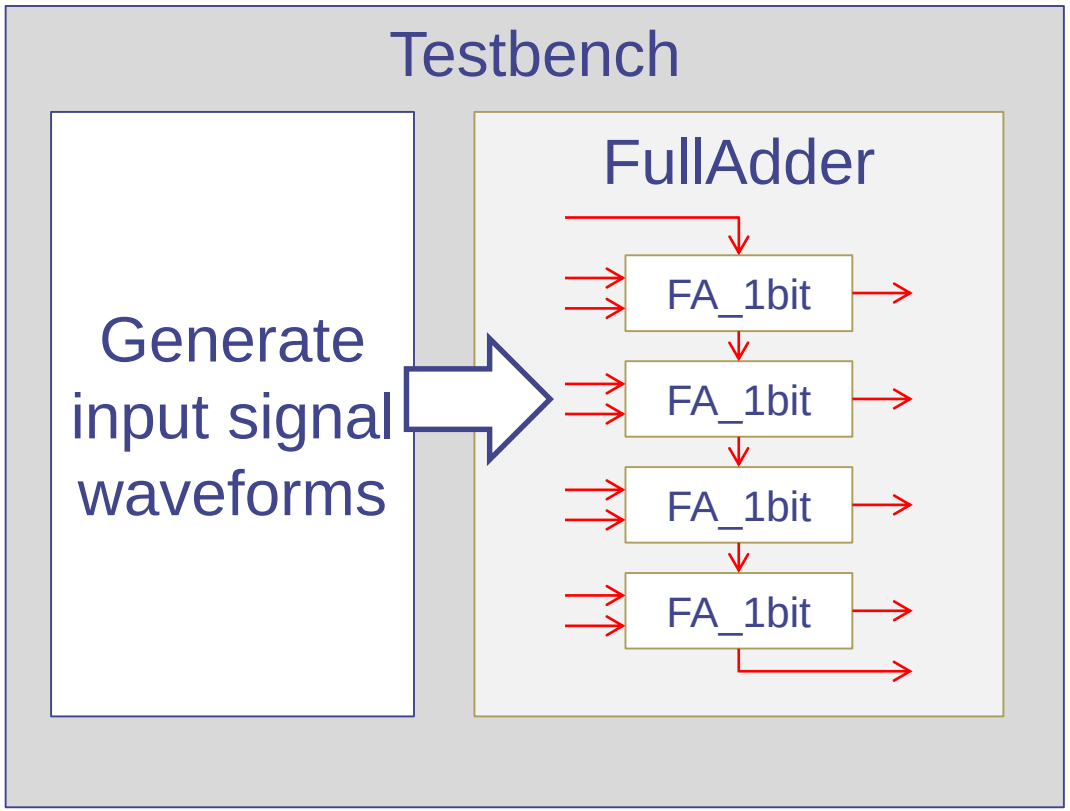
- The complete code for a 4-bit full adder is as follows:

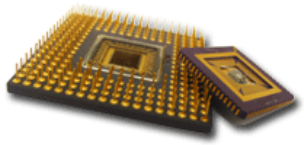
```
// ------------- A four-bit full adder ----------------------------
module FullAdder(A, B, Cin, S, Cout);
  input [3:0] A, B;
  input Cin;
  output [3:0] S;
  output Cout;
  wire [2:0] t;

  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

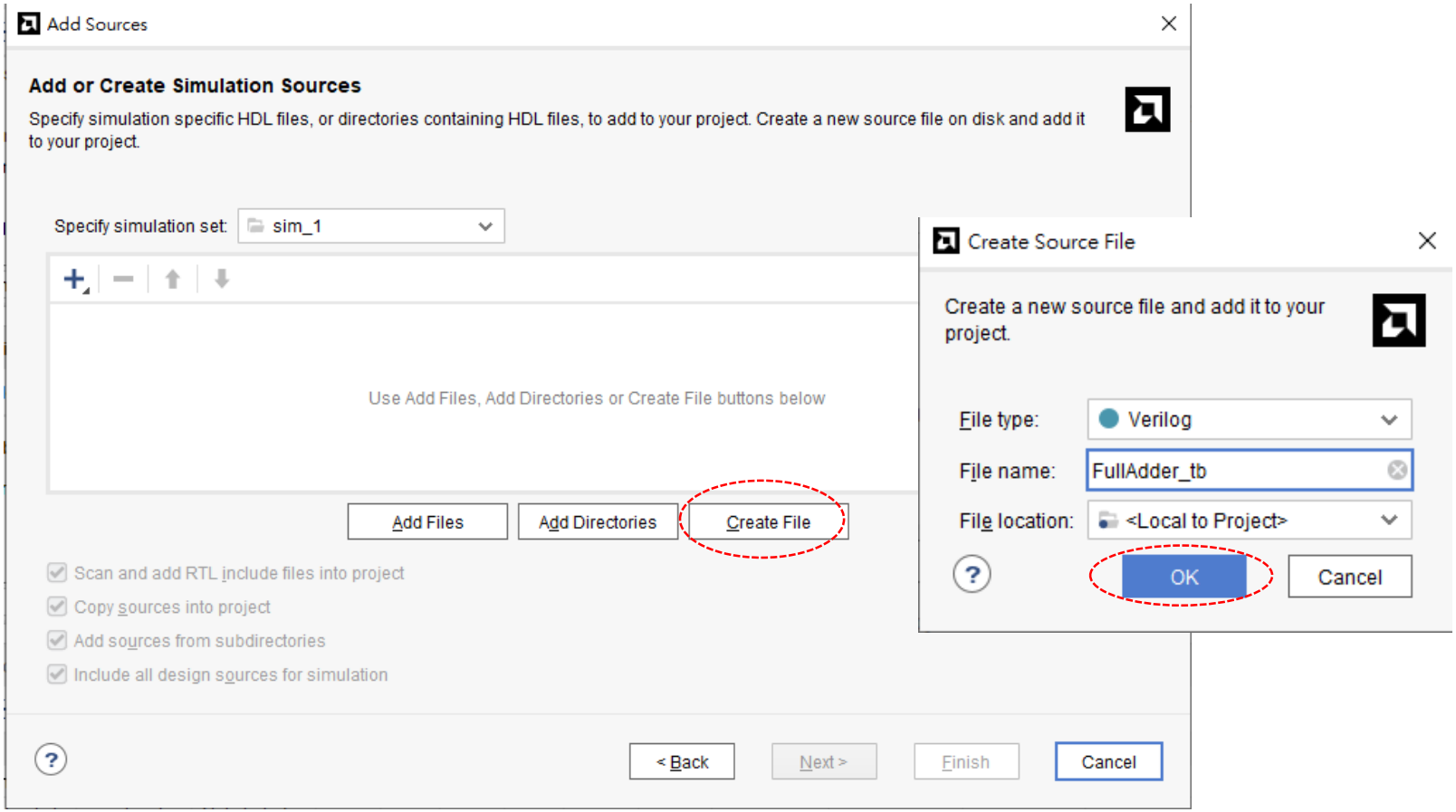
- You must create a testbench to generate input signals that can feed into your circuit module, such that you can analyze the output to verify its correctness.





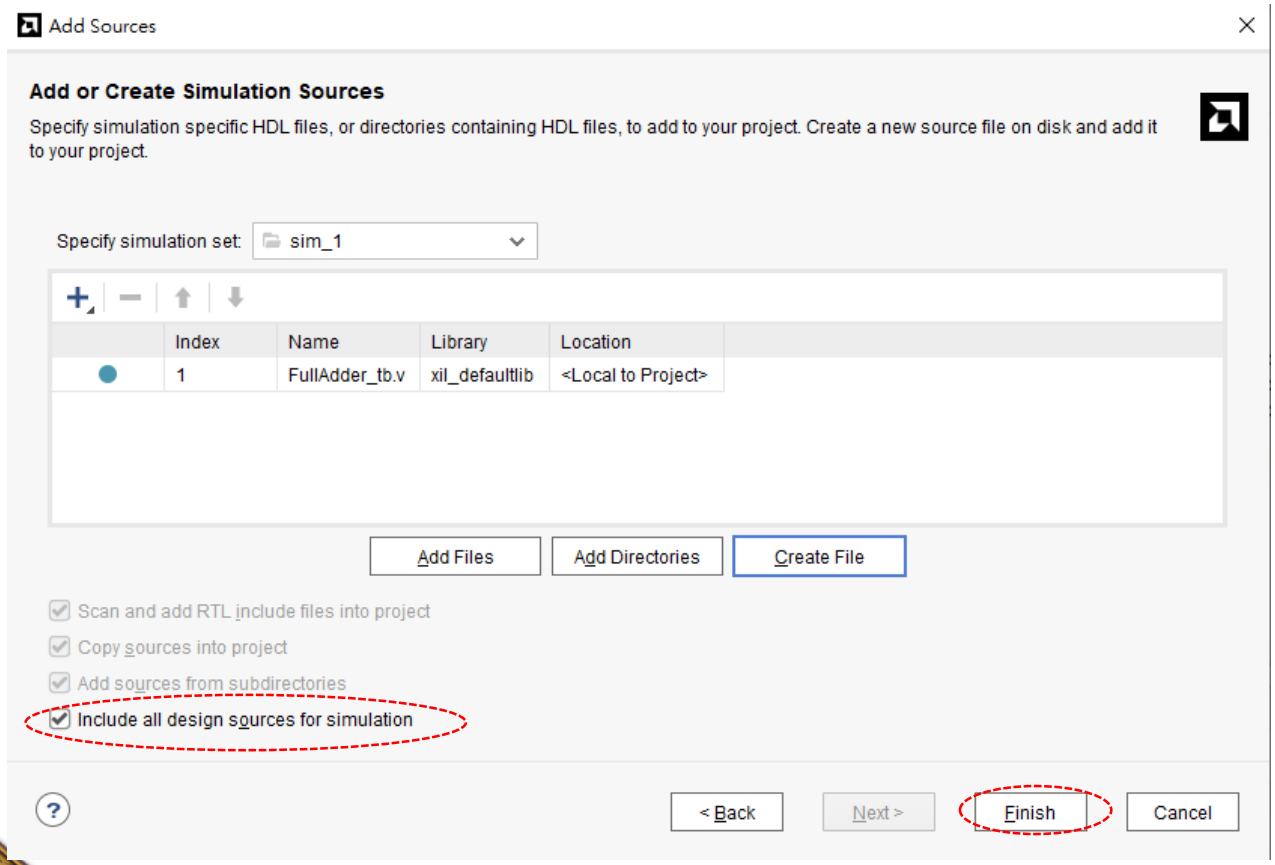
Create the Testbench Source Code

- Click “Add Sources” button again, and this time, select “Add or create simulation sources”



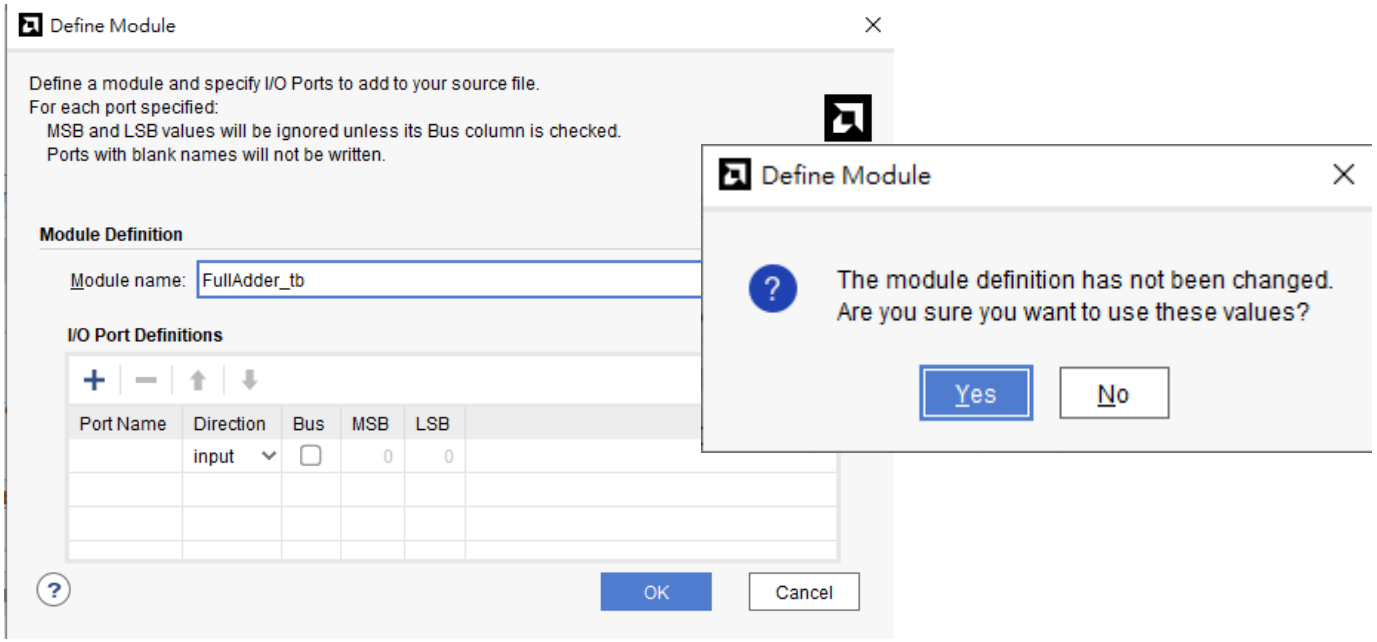
Confirm the Creation of the Testbench

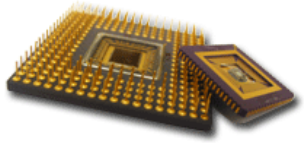
- Here, we include the design sources into the simulation set so that we can test the modules under development.



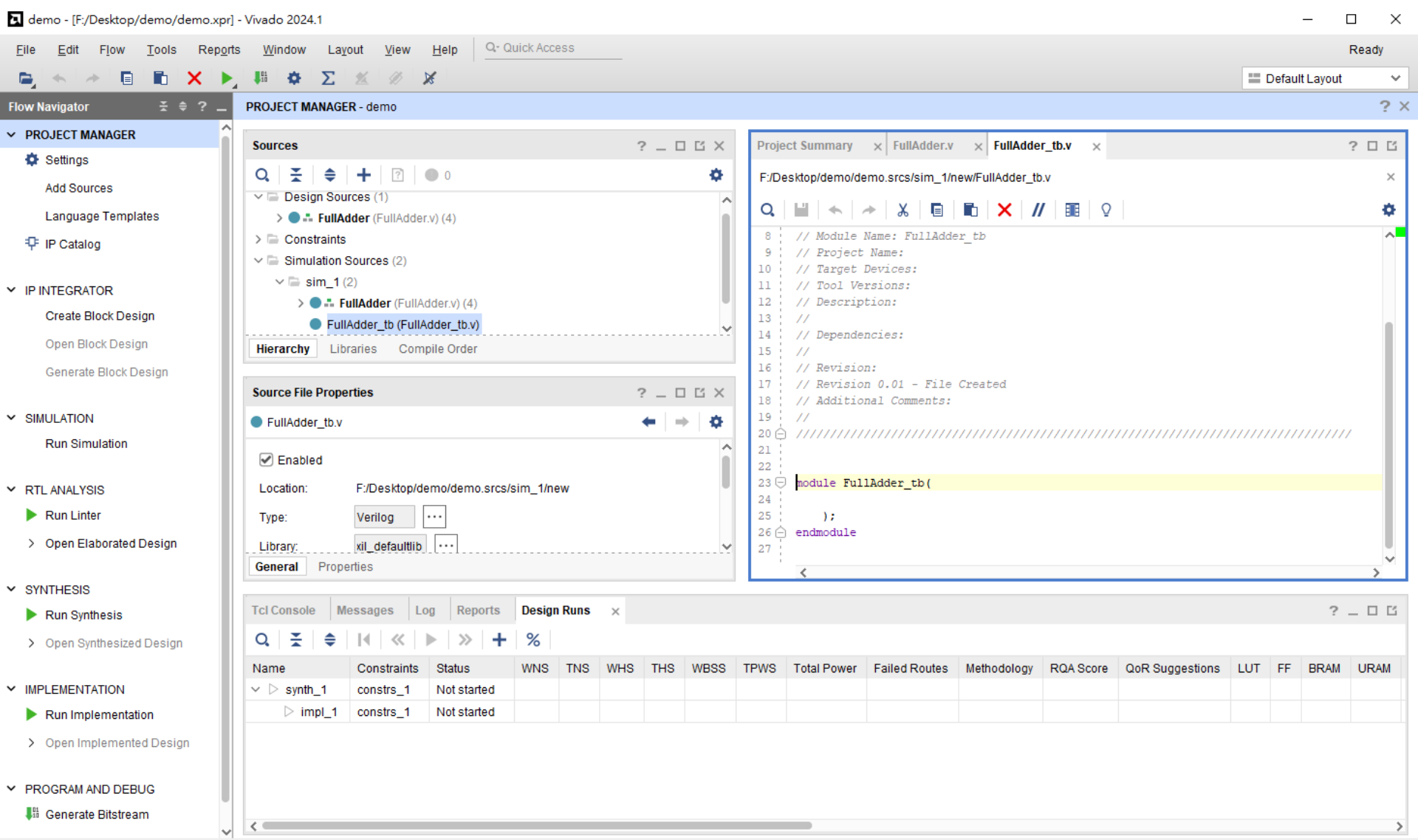
Create the Testbench Template

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





Type in the Testbench Source Code

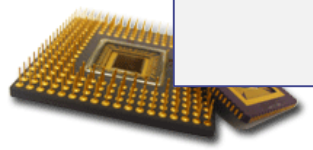


The Sample Testbench Code

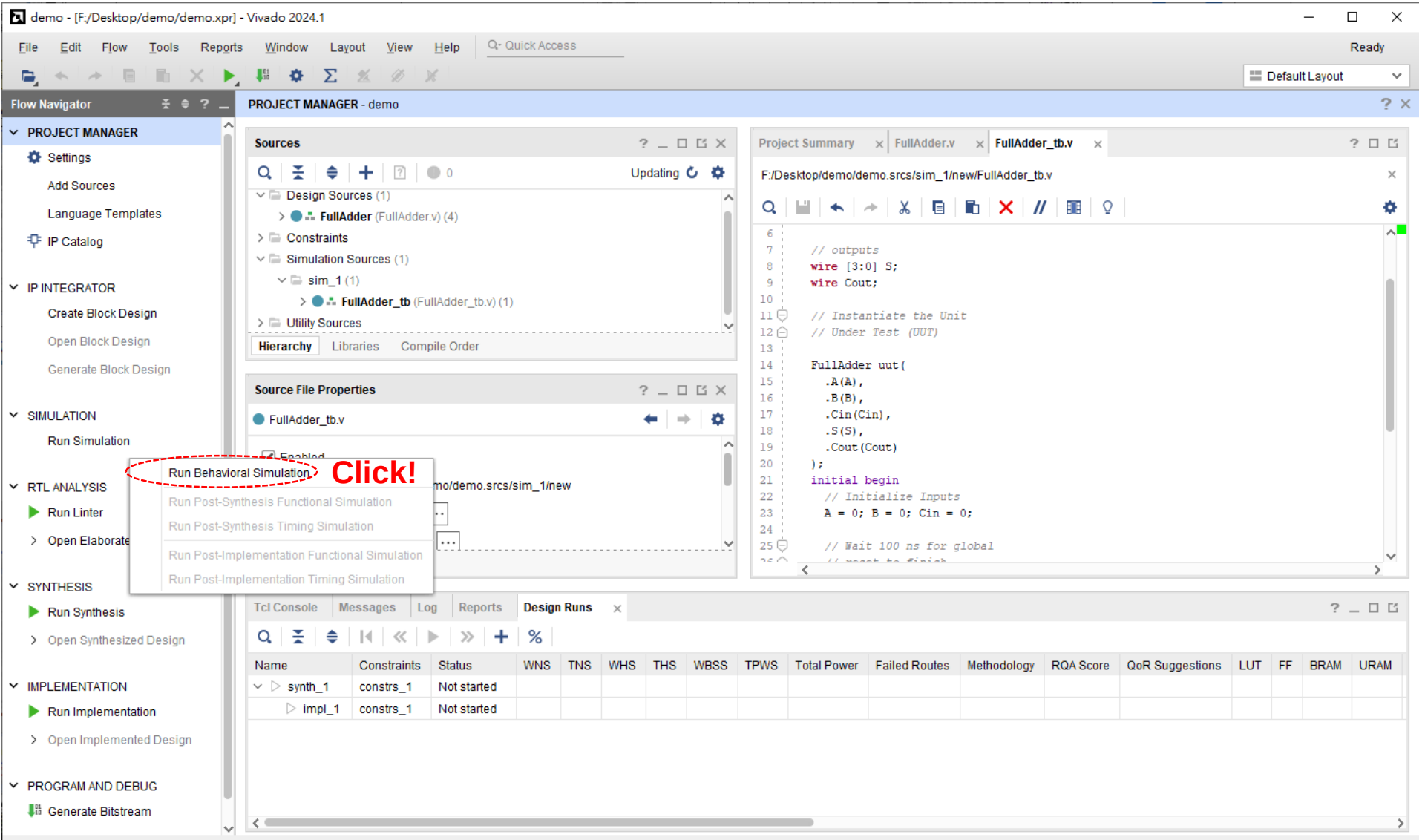
- The template created by Vivado is an empty module; you must add test pattern generators in the module.

```
module FullAdder_tb;

 // inputs
  reg [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)
  );
```

```
  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



Vivado Simulator Window