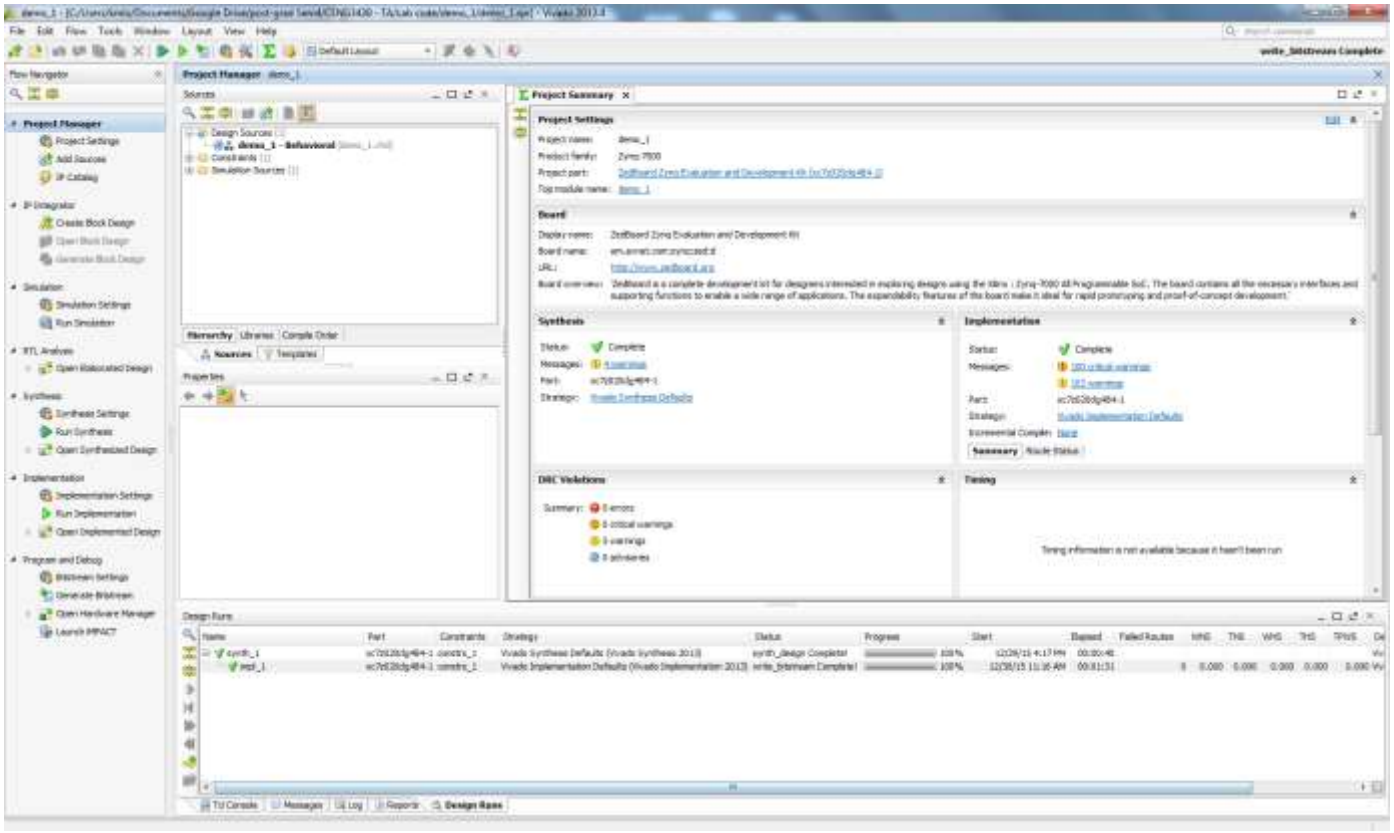
香港中文大學
The Chinese University of Hong Kong

CENG3430 Rapid Prototyping of Digital Systems

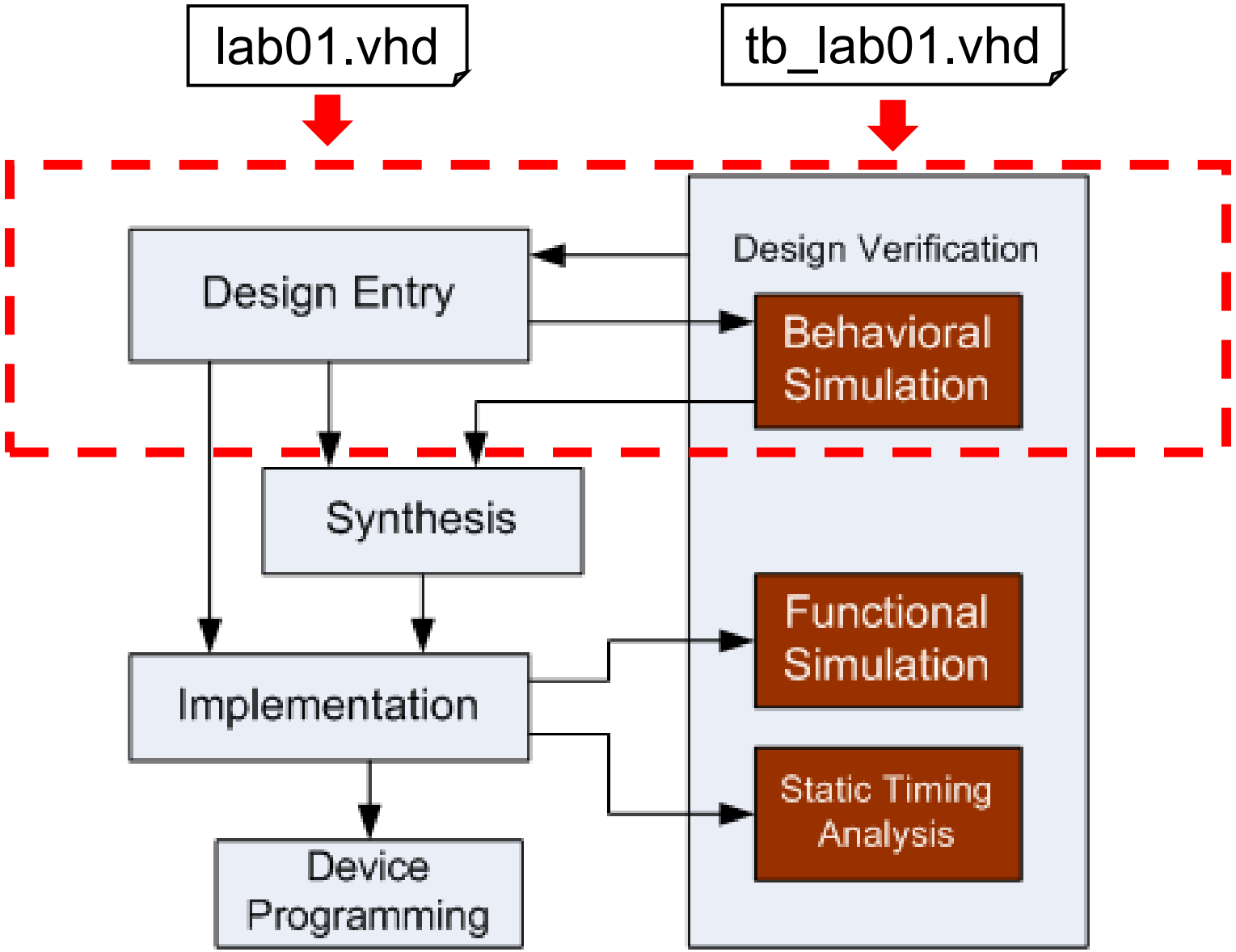
Lab01: Software Simulation

Vivado

- **Programming** using IDE for VHDL (Lab00)
- Running **simulation** to test your design (Lab01)
- **Downloading** the compiled code to FPGA (Lab02)



Focus of Lab01: Behavioral Simulation



Lab01 Example

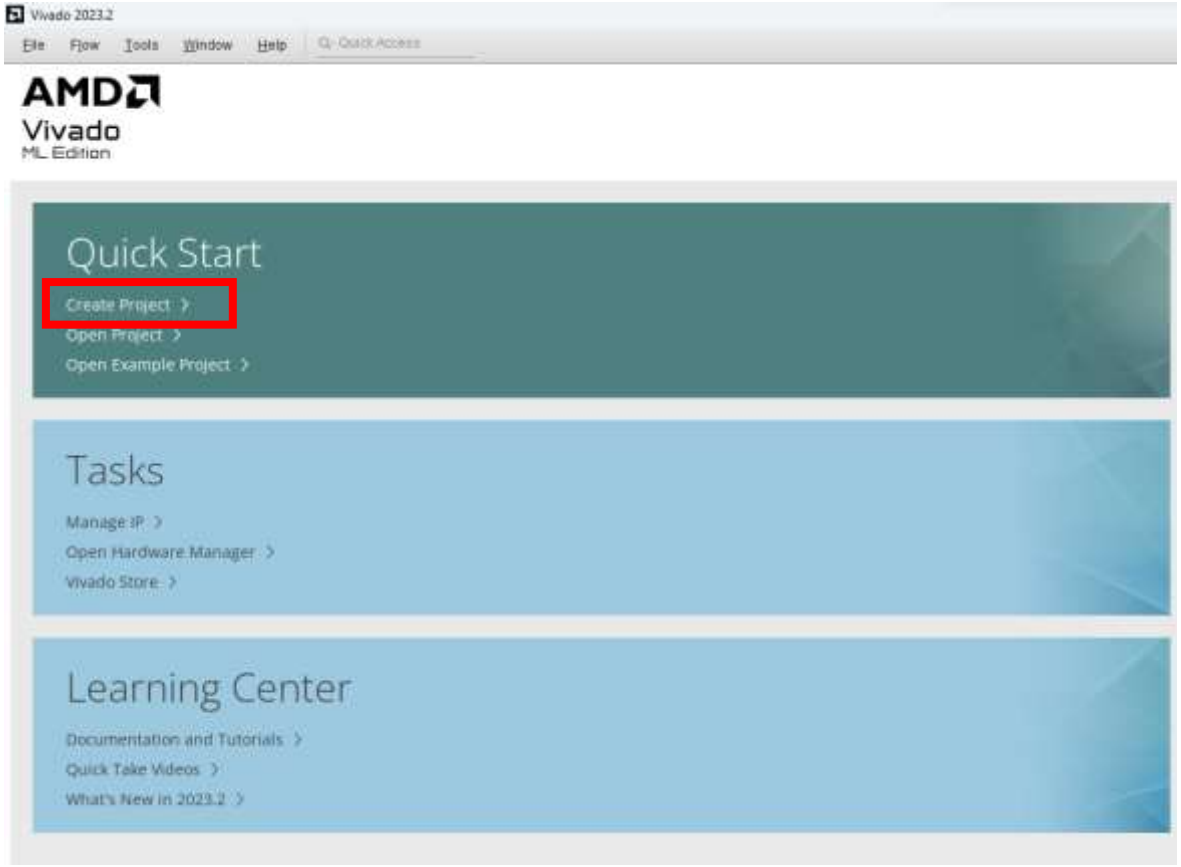
I. Create a Vivado Project (Lab00)

Step 1: Start the Software (Lab00)

- Double click the icon of “Vivado”
- Or start “Vivado” from the Start Menu

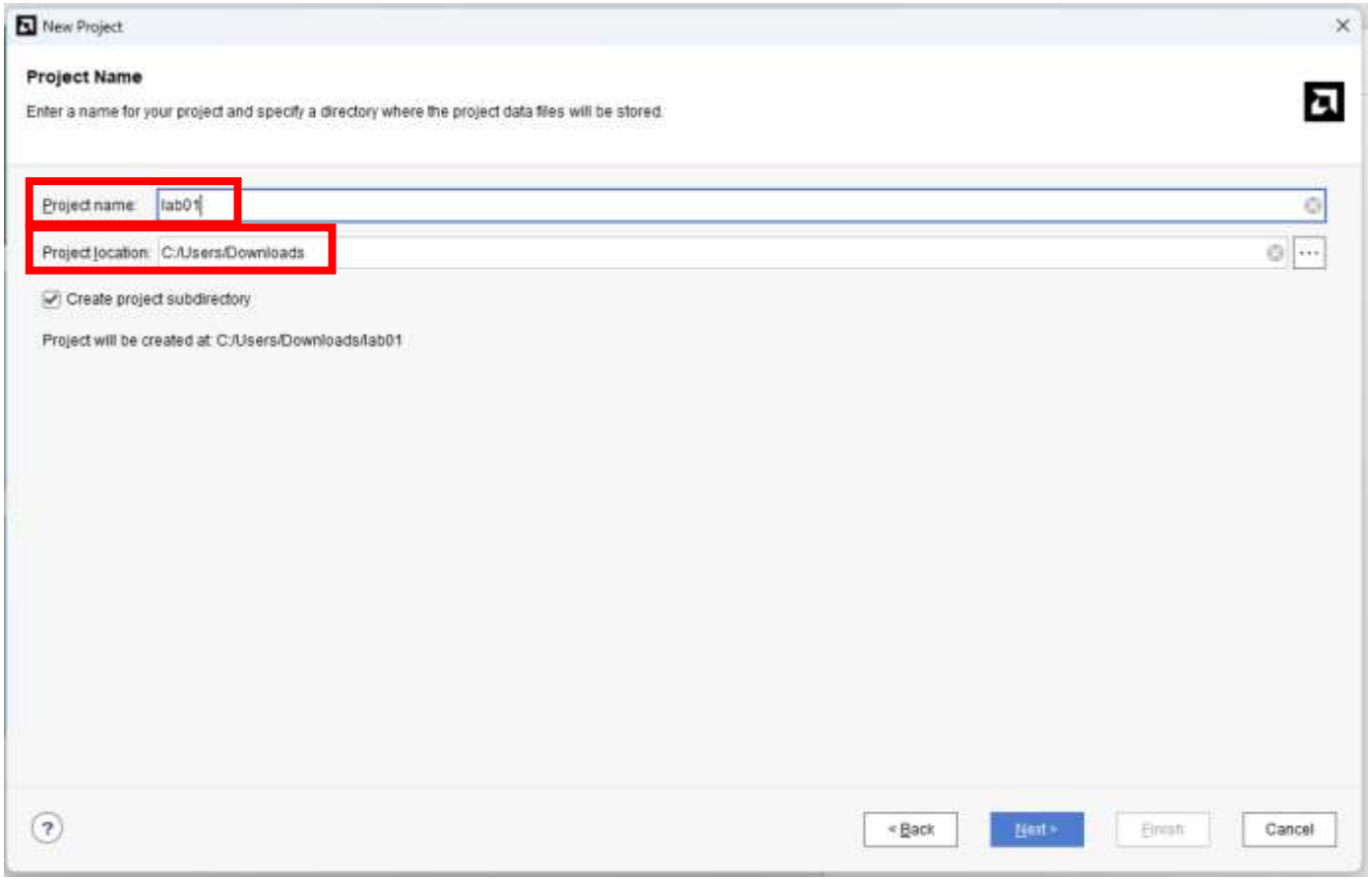
Step 2: Create a New Project (Lab00)

- Click “Create Project”
- Or click “File” -> “Project” -> “New”



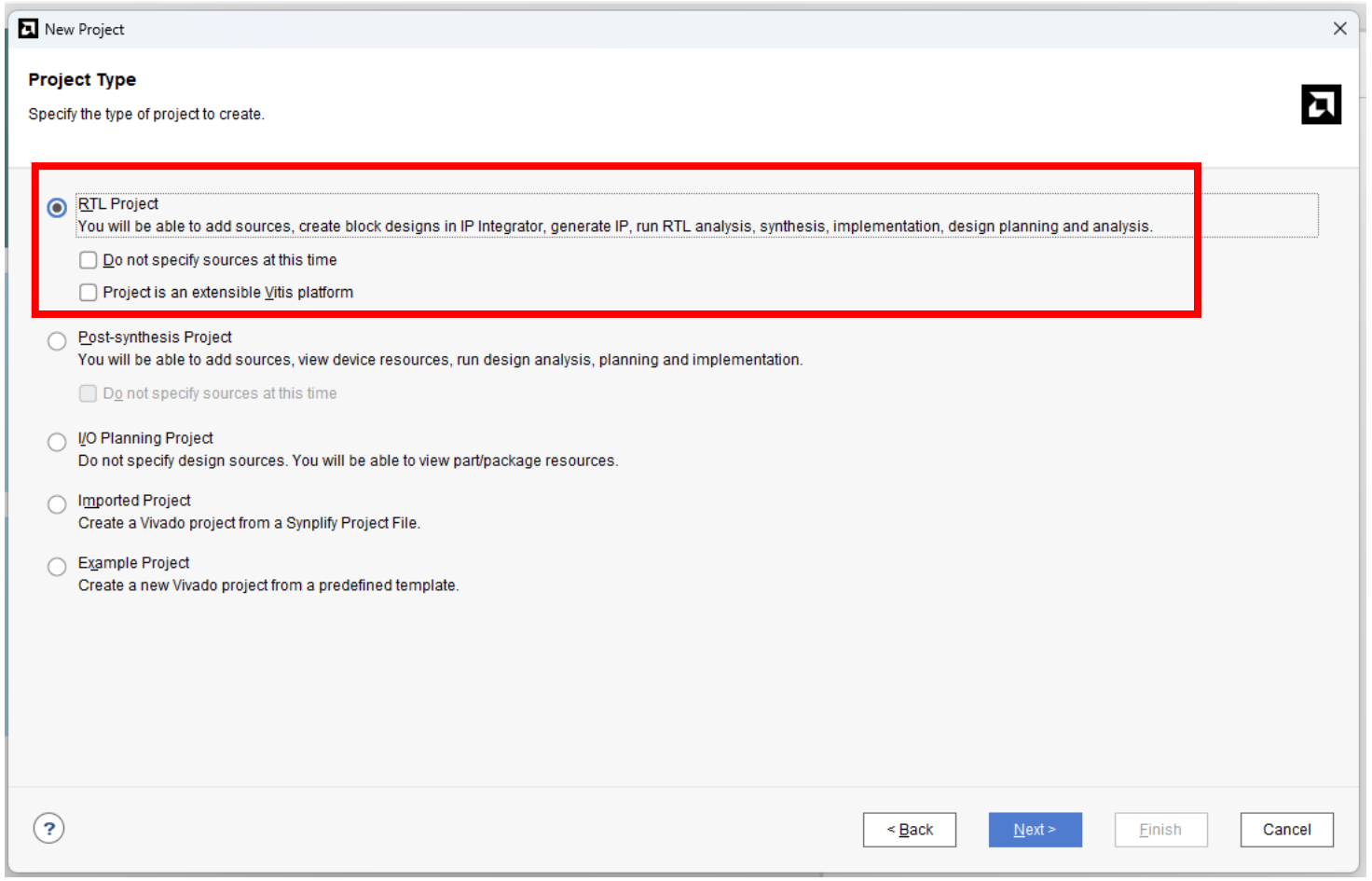
Step 2: Create a New Project (Lab00)

- Enter the “Project name” (e.g., lab01)
- Specify a proper “Project location” for storage

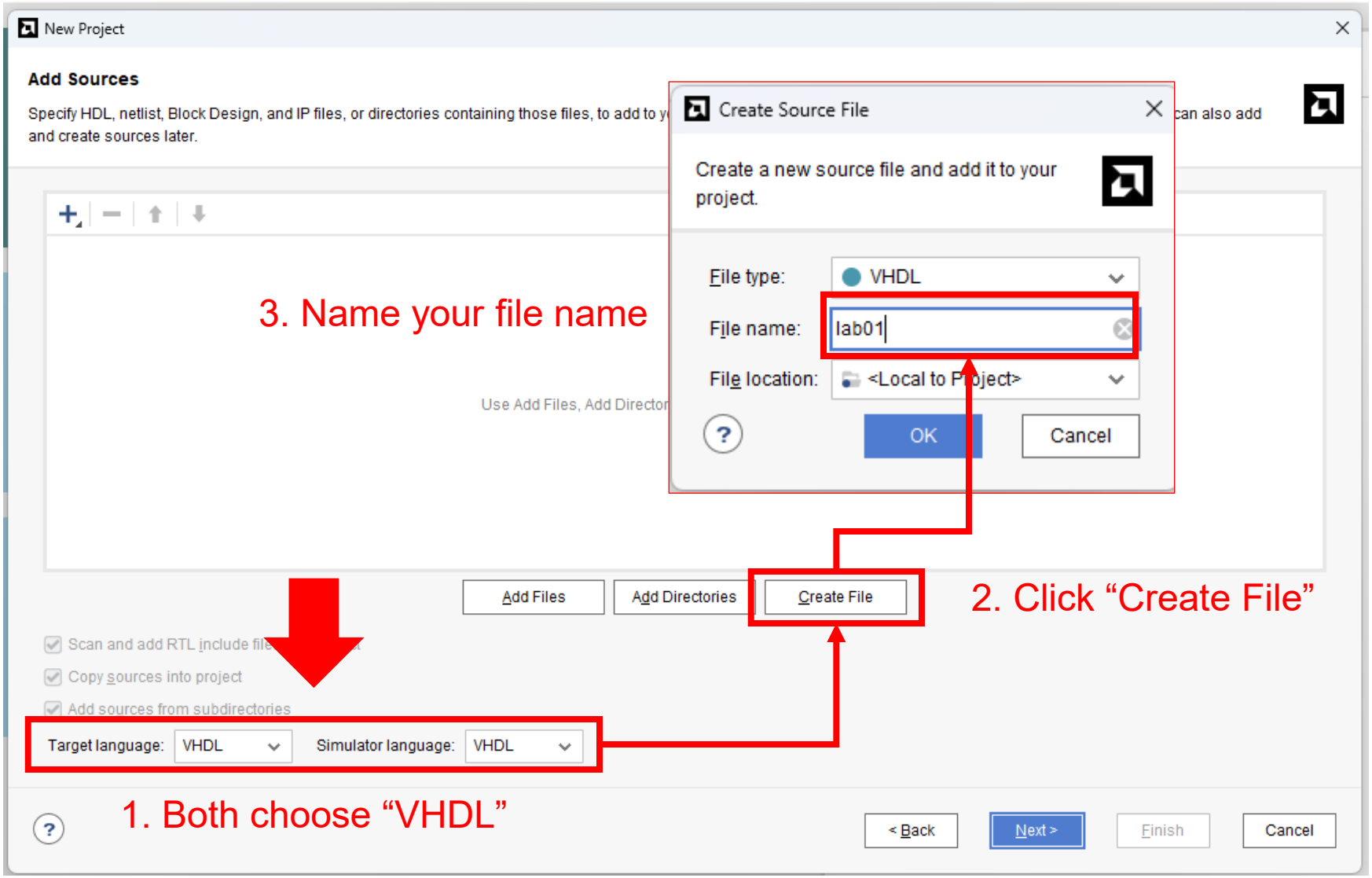


Step 2: Create a New Project (Lab00)

- Select “RTL Project” -> “Next”

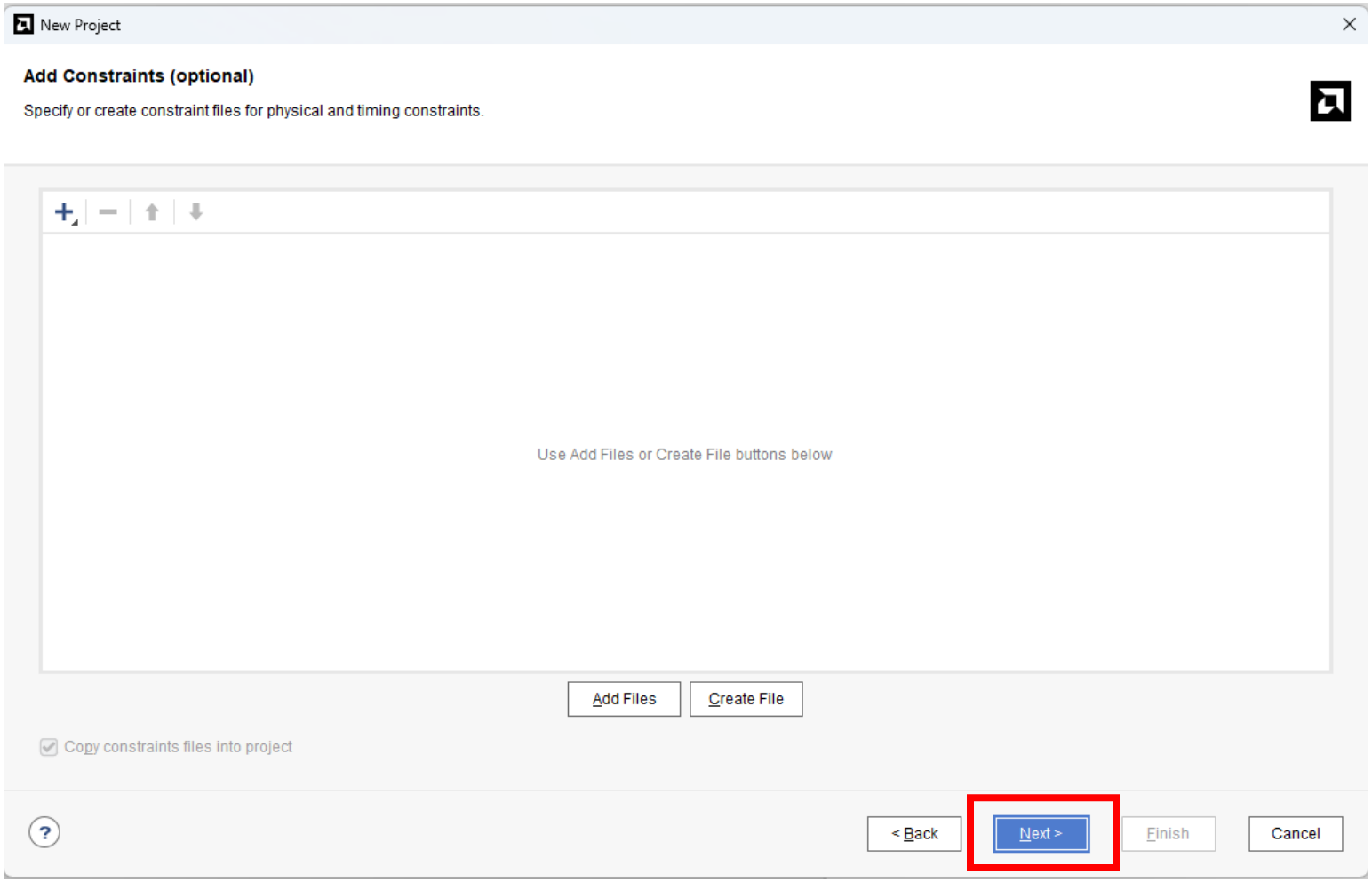


Step 2: Create a New Project (Lab00)



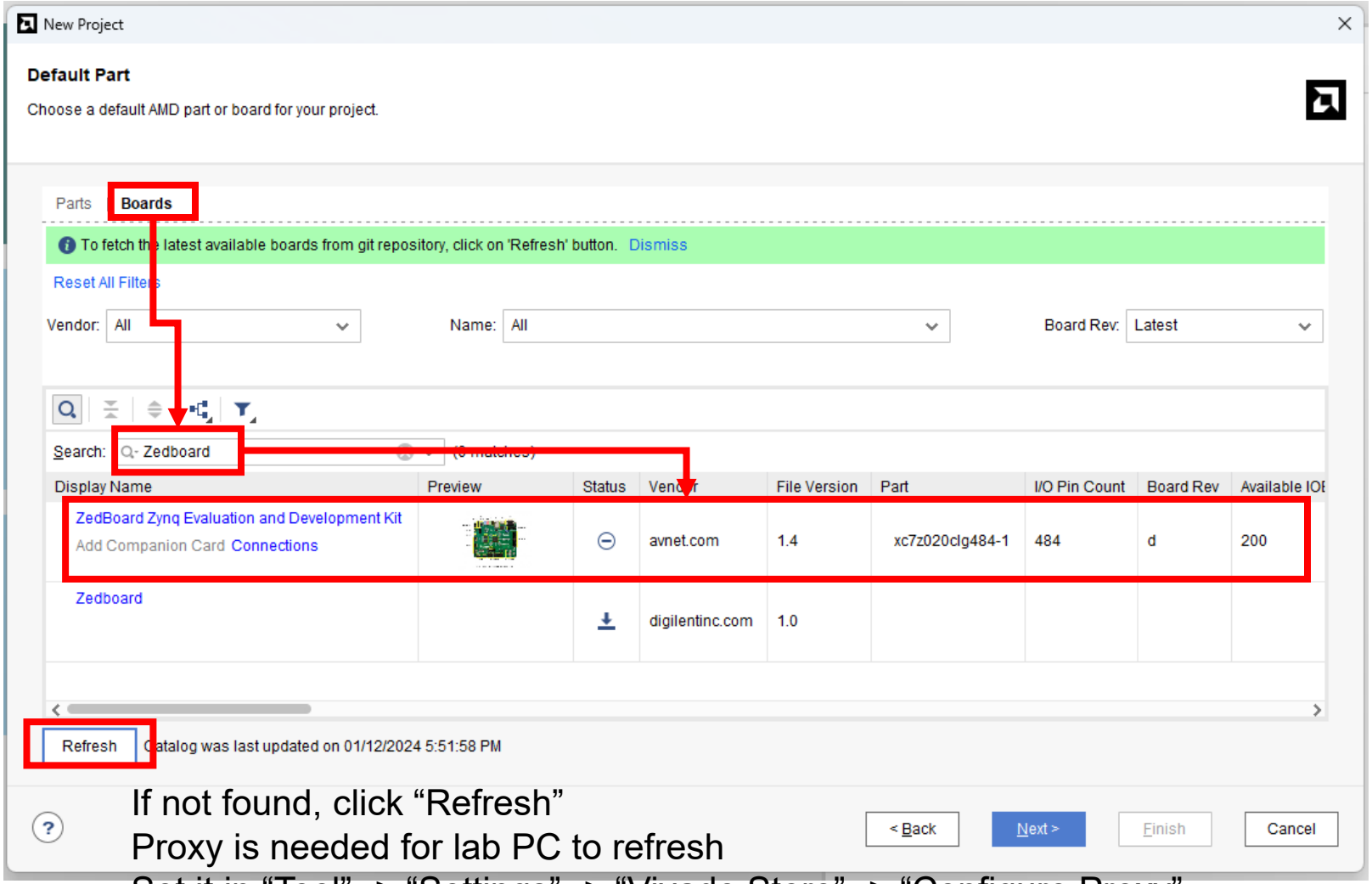
Step 2: Create a New Project (Lab00)

- Click “Next”



Step 2: Create a New Project (Lab00)

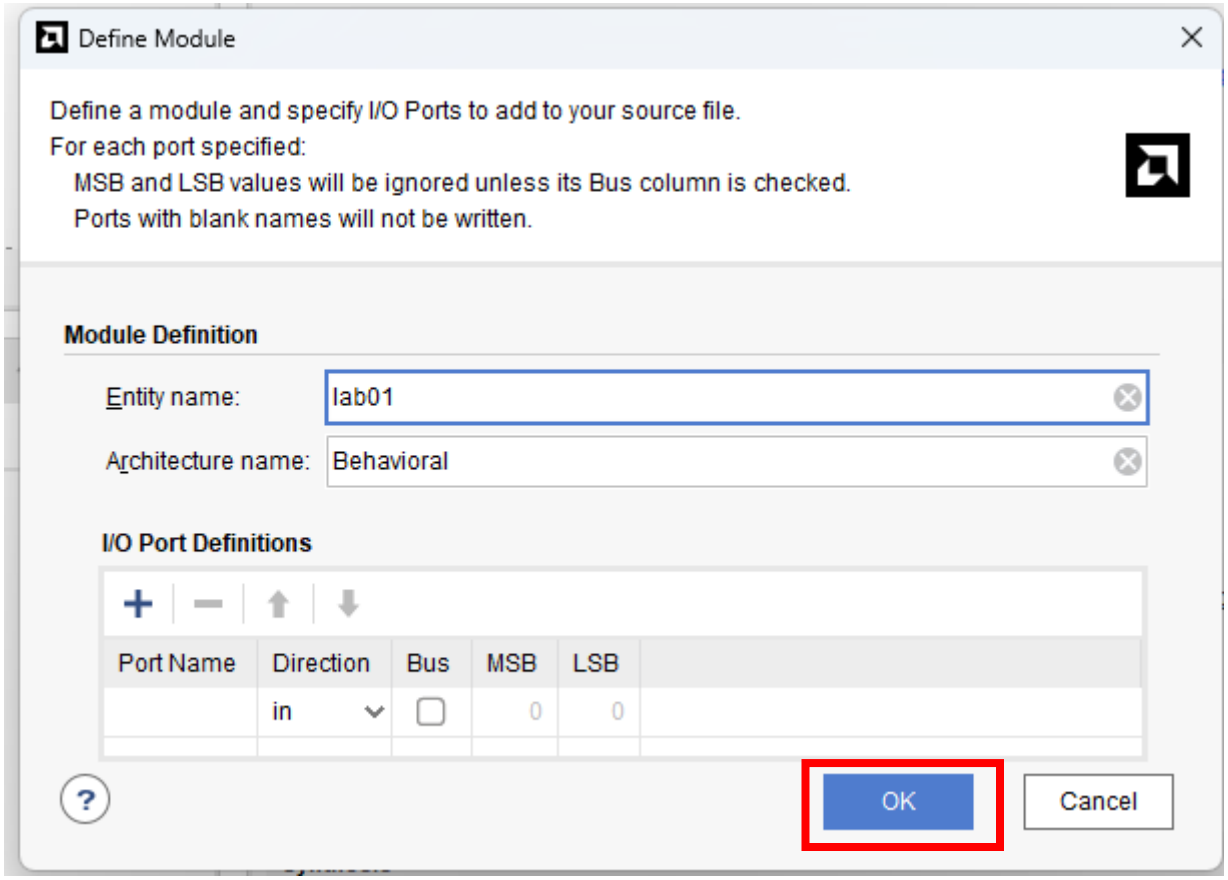
- Click “Boards” -> Select “ZedBoard” -> Click “Next”



If not found, click “Refresh”
Proxy is needed for lab PC to refresh
Set it in “Tool” -> “Settings” -> “Vivado Store” -> “Configure Proxy”

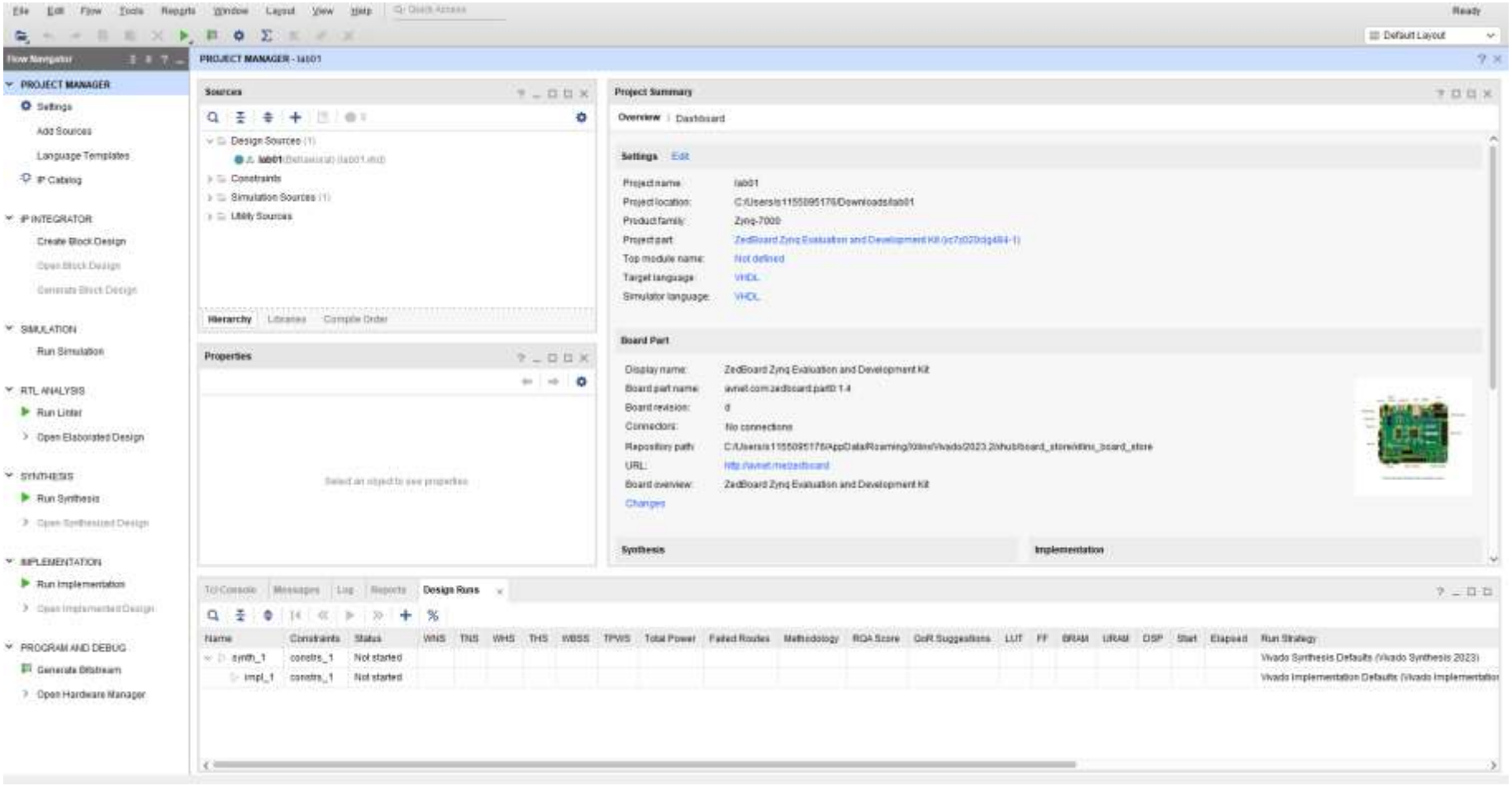
Step 2: Create a New Project (Lab00)

- If you see this box, just click “OK”



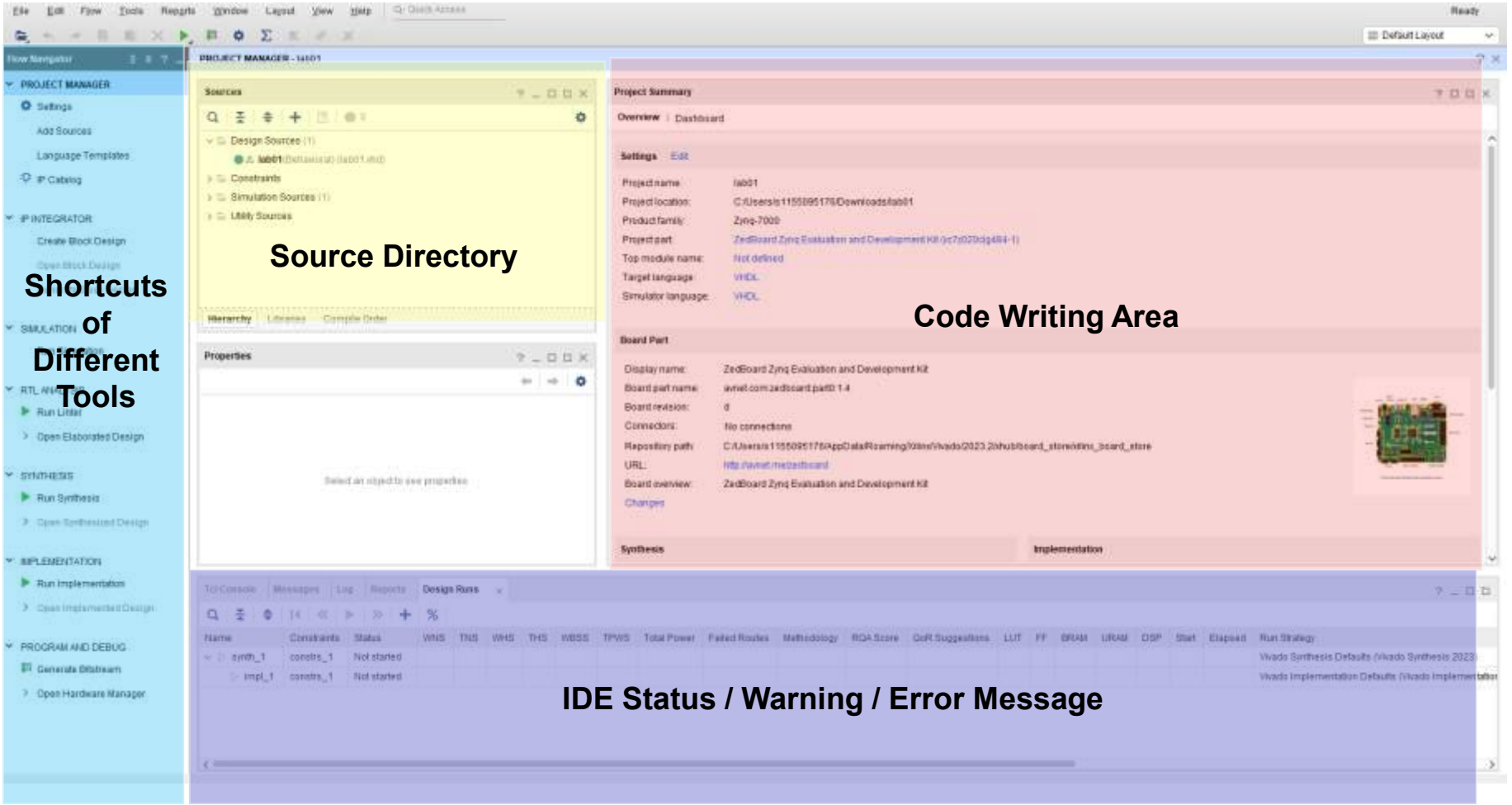
Step 2: Create a New Project (Lab00)

- This is the programming interface of Vivado:



II. Write the VHDL Code (Lab00)

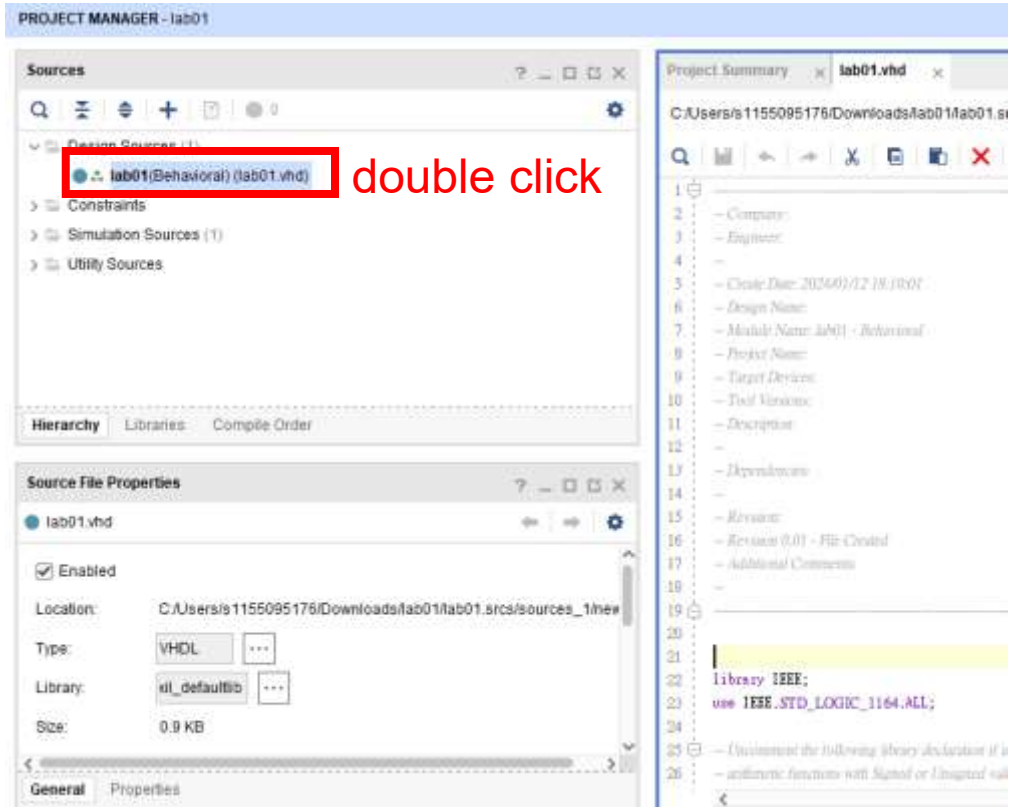
Step 3: Start Programming (Lab00)



- This is the programming interface of Vivado:

Step 3: Start Programming (Lab00)

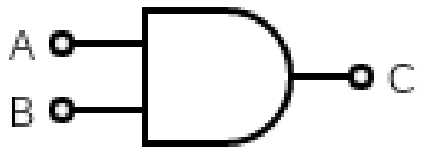
- In the “Source Directory”, double click the source file (e.g., lab01.vhd) created.
  - *Seeing the file name appears twice in “Syntax Error Files” and “Non-module Files” is normal if the source code is empty or having syntax errors.*



Step 3: Start Programming (Lab00)

- Implement a simple AND logic and save

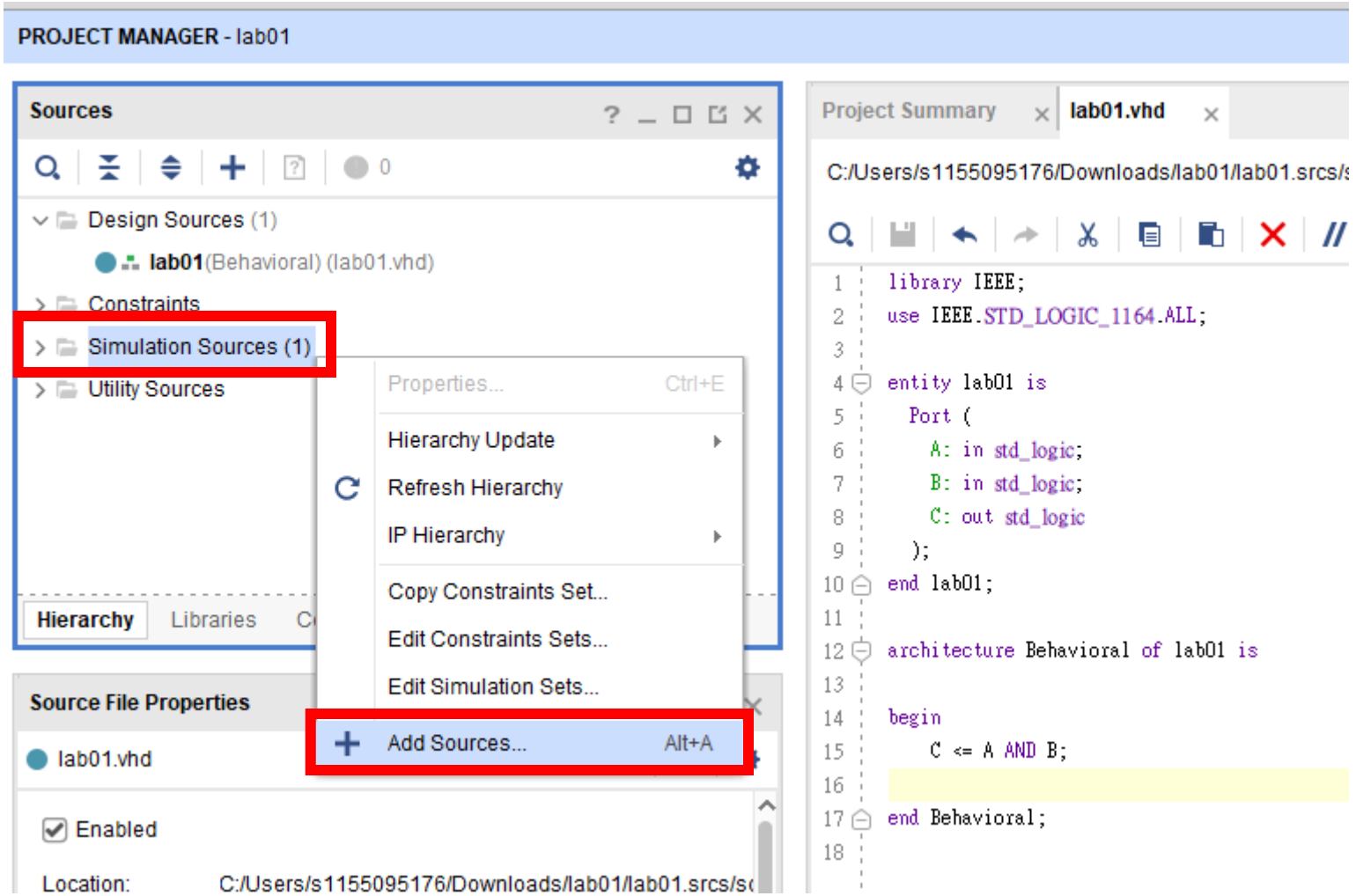
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lab01 is
  Port (
    A: in std_logic;
    B: in std_logic;
    C: out std_logic
  );
end lab01;

architecture Behavioral of lab01 is

begin
    C <= A AND B;

end Behavioral;
```



You can copy this

III. Perform the Simulation

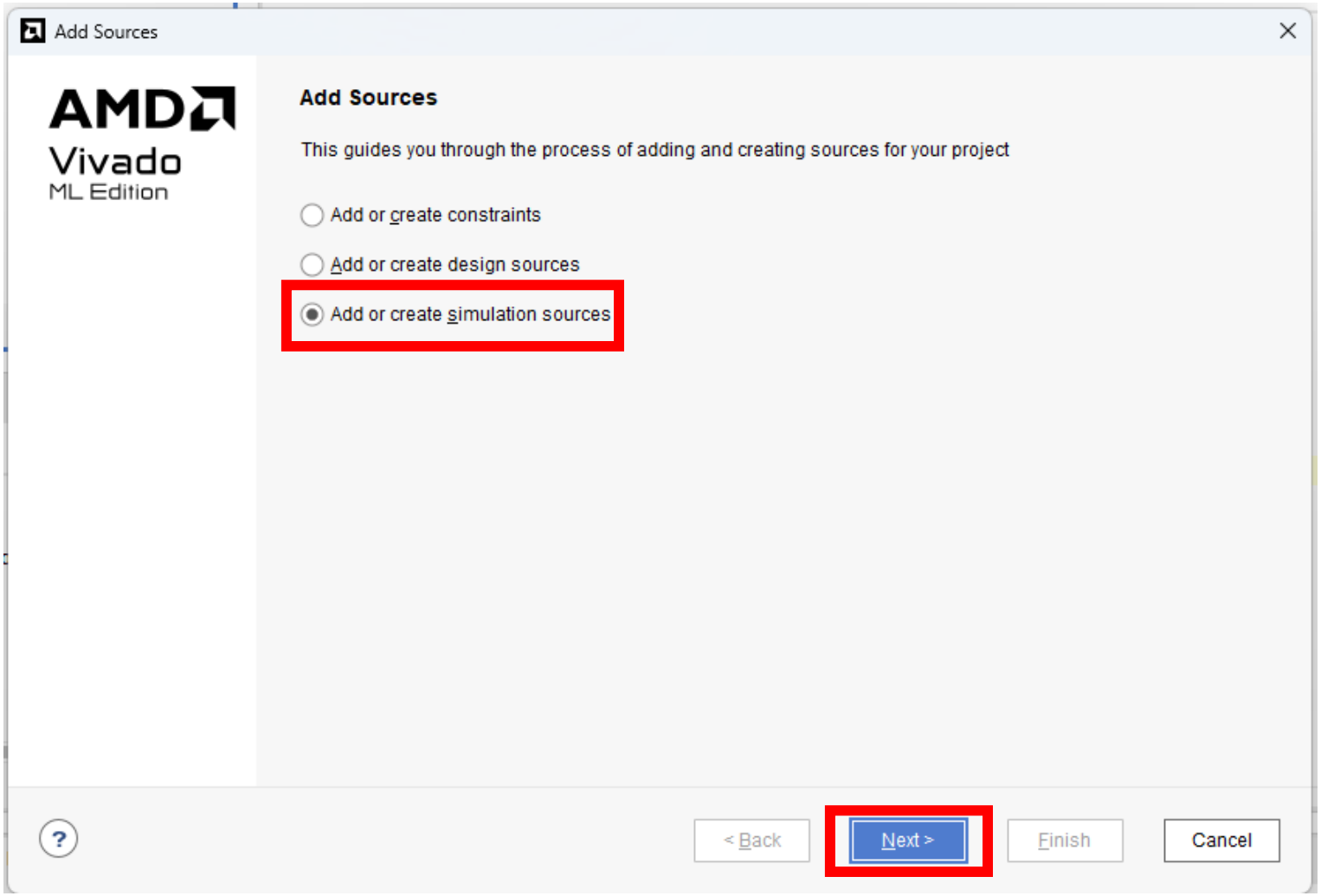
Step 4: Write a Testbench



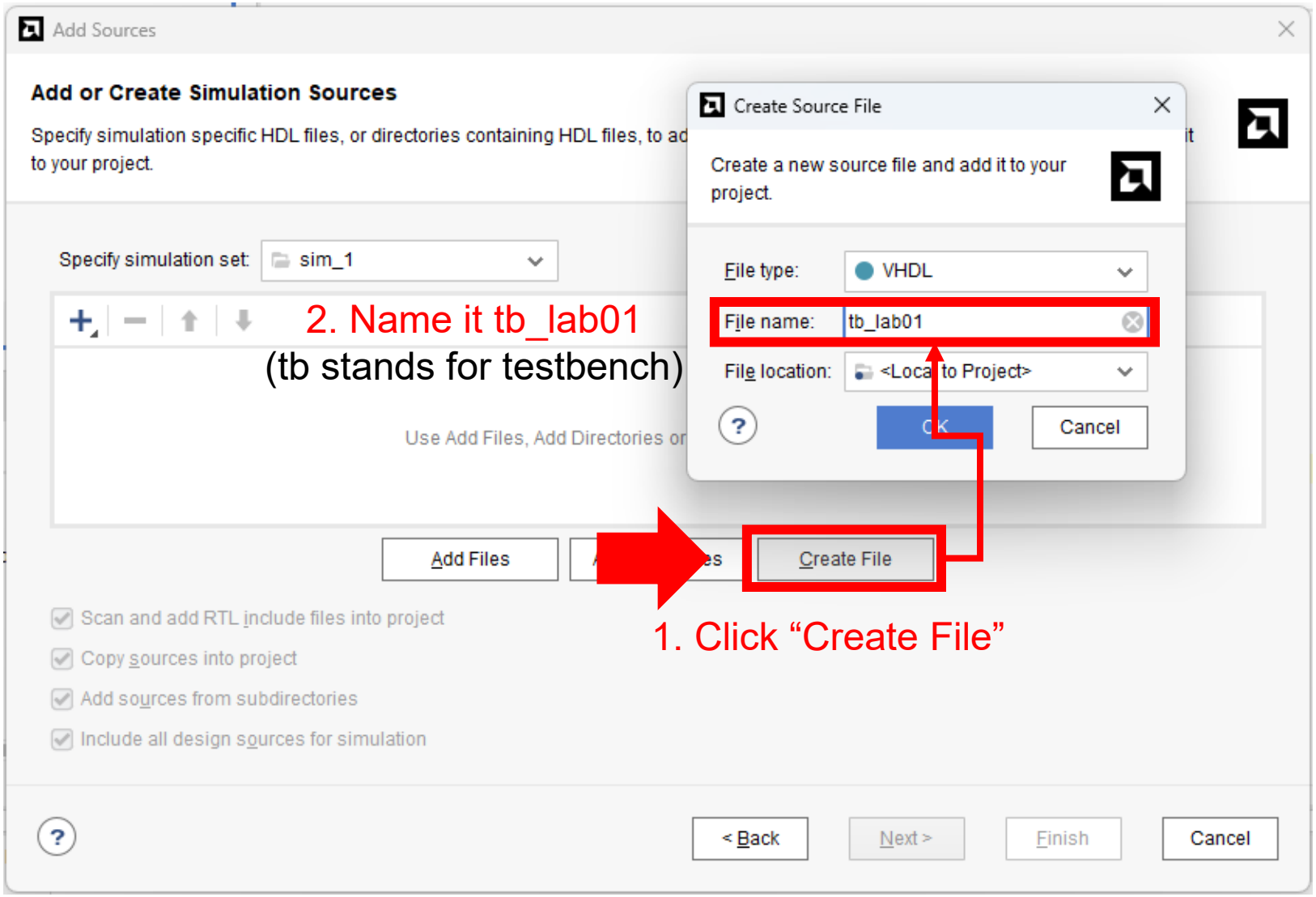
- Under the source window, right click on the “Simulation Sources” -> “Add sources…”

Step 4: Write a Testbench

- Select “Add or Create Simulation Sources” -> “Next”

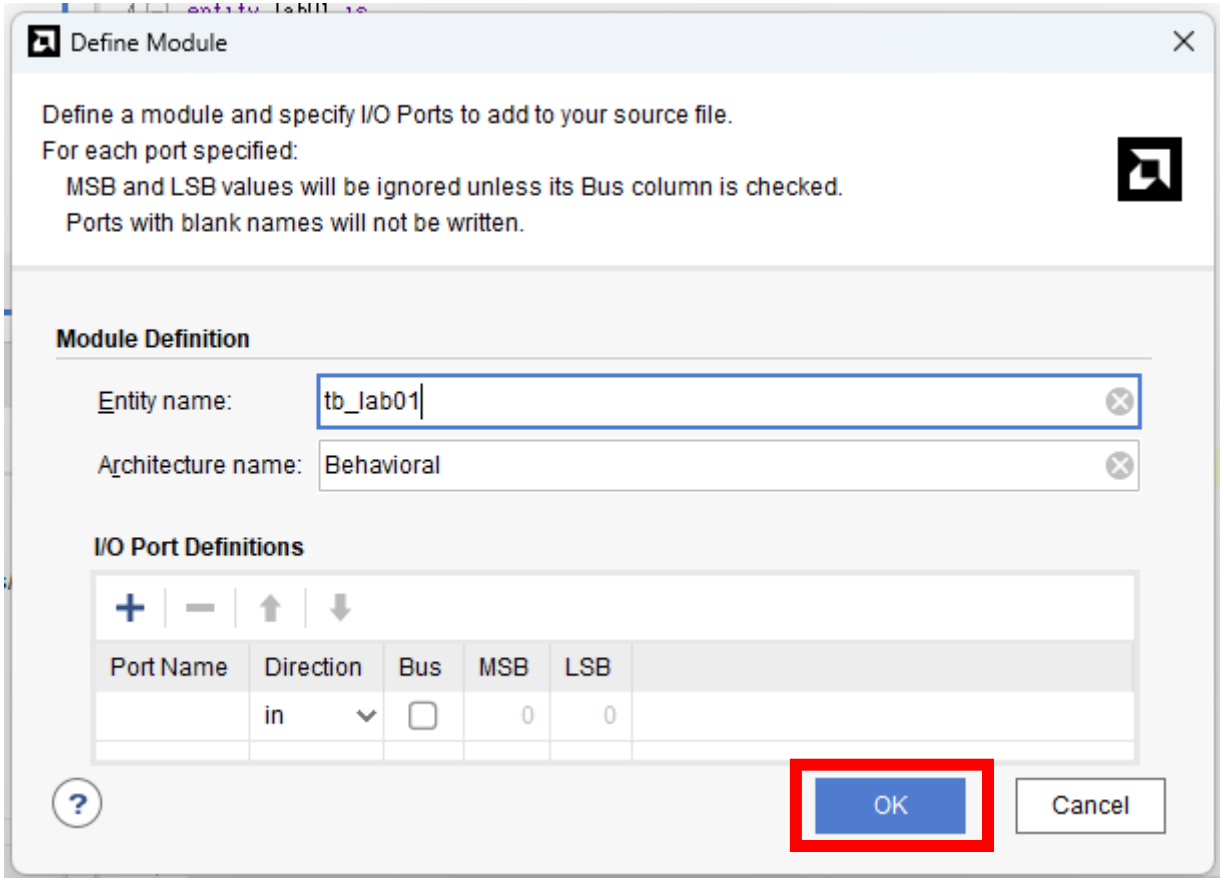


Step 4: Write a Testbench



Step 4: Write a Testbench

- If you see this box, just click “OK”

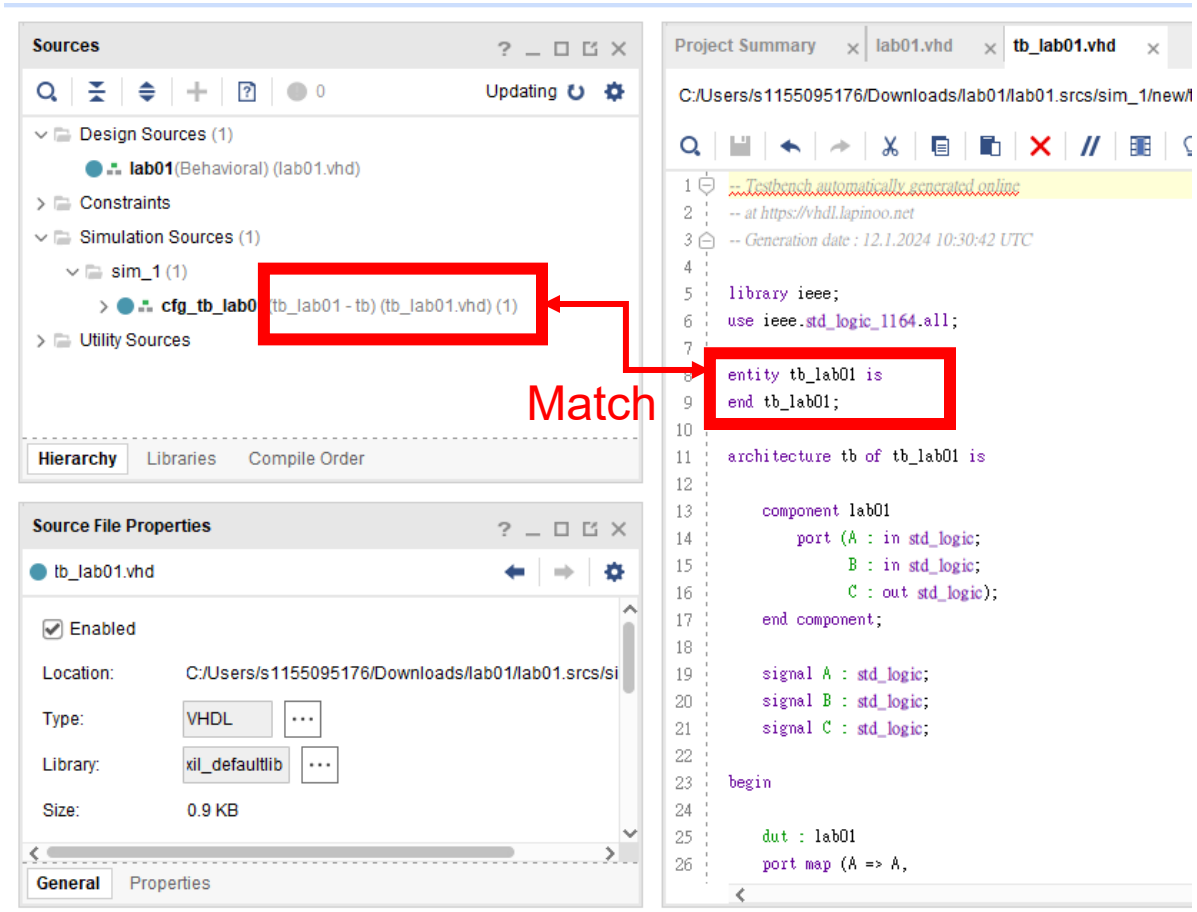


Step 4: Write a Testbench

- The testbench file will not be generated automatically
- An easy way is to use Testbench Template Generator
  1. Visit https://vhdl.lapinoo.net/testbench/
  2. Paste your source code into it, and the website will generate the testbench file for you
     - ✓ Select "No clock generation"
     - ✓ Select "No reset generation"
  3. Copy the content generated by the website, and replace the content on tb_lab01.vhd

Step 4: Write a Testbench

- Make sure that the entity name (i.e., tb_lab0) in the testbench file matches the testbench file name



Step 4: Write a Testbench

- Finish the *stimuli* process with one style below

```
26      port map (A => A,
27                B => B,
28                C => C);
29
30      stimuli : process
31      begin
32          -- EDIT Adapt initialization as needed
33          A <= '0';
34          B <= '0';
35          wait for 100ns;
36          A <= '0';
37          B <= '1';
38          wait for 100ns;
39          A <= '1';
40          B <= '0';
41          wait for 100ns;
42          A <= '1';
43          B <= '1';
44          wait for 100ns;
45
46          wait;
47      end process;
48
49  end tb;
```

Style 1 (Simpler)
(copyable version on the next slide)

```
30      stimuli : process
31      begin
32          -- EDIT Adapt initialization as needed
33          A <= '0';
34          B <= '0';
35          wait for 100ns;
36          assert(C = '0')
37          report "Test failed for input 00" severity error;
38
39          A <= '0';
40          B <= '1';
41          wait for 100ns;
42          assert(C = '0')
43          report "Test failed for input 01" severity error;
44
45          A <= '1';
46          B <= '0';
47          wait for 100ns;
48          assert(C = '0')
49          report "Test failed for input 10" severity error;
50
51          A <= '1';
52          B <= '1';
53          wait for 100ns;
54          assert(C = '1')
55          report "Test failed for input 11" severity error;
56
57          wait;
58      end process;
```

Style 2
(With assertion)

Step 4: Write a Testbench

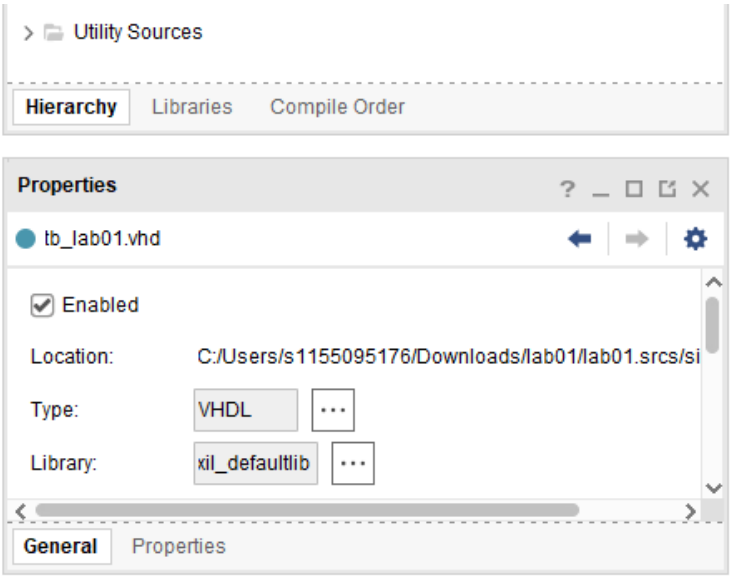
- Testbench Style 1

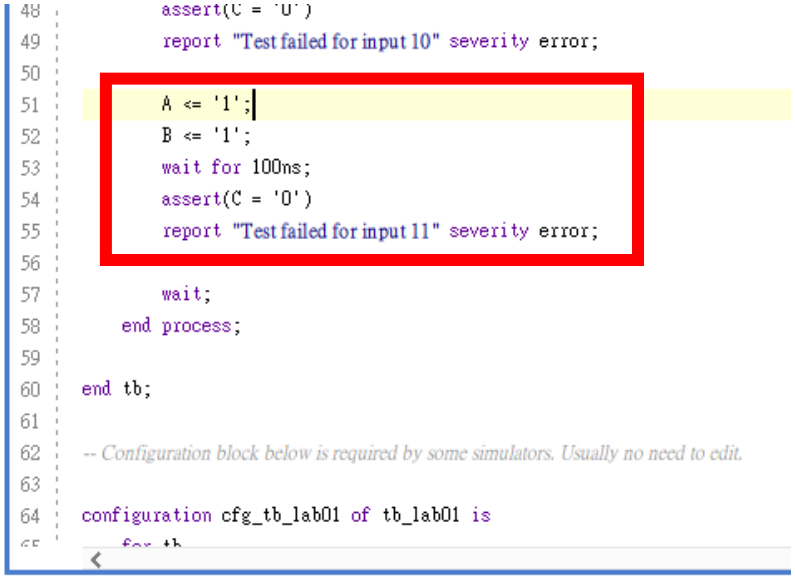
```
stimuli : process
    begin
        A <= '0';
        B <= '0';
        wait for 100ns;
        A <= '0';
        B <= '1';
        wait for 100ns;
        A <= '1';
        B <= '0';
        wait for 100ns;
        A <= '1';
        B <= '1';
        wait for 100ns;
        wait;
    end process;
```

You can copy this

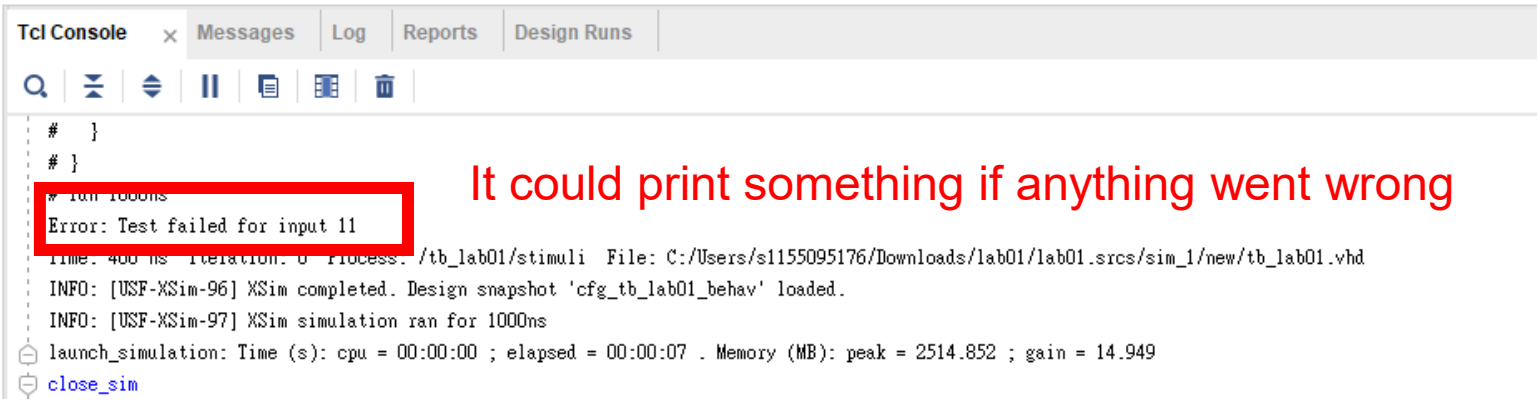
Step 4: Write a Testbench

- Why Style 2 ?





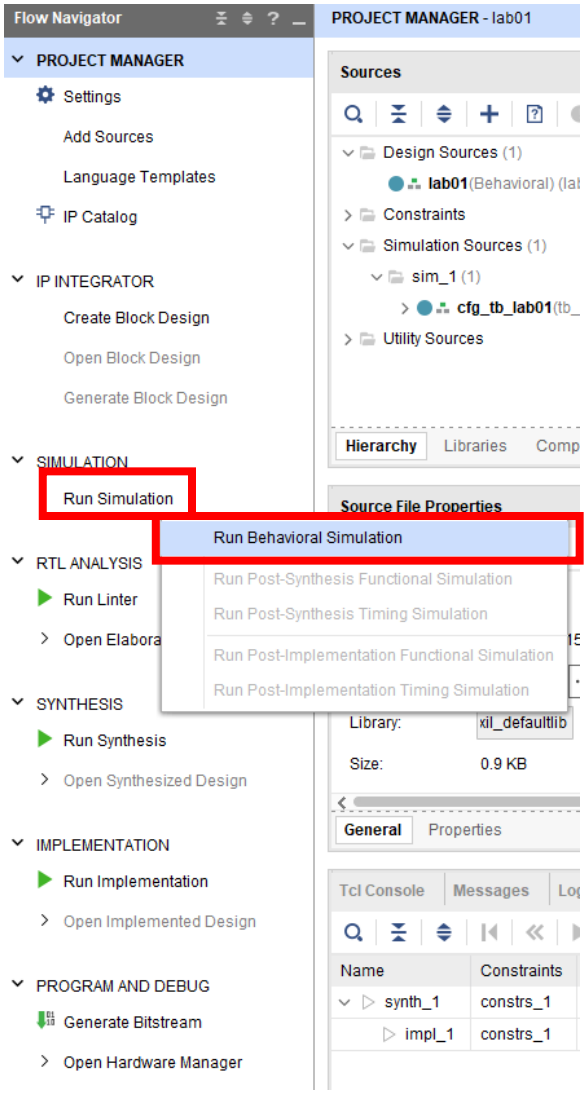
```
48          assert(C = '0')
49          report "Test failed for input 10" severity error;
50
51          A <= '1';
52          B <= '1';
53          wait for 100ns;
54          assert(C = '0')
55          report "Test failed for input 11" severity error;
56
57          wait;
58      end process;
59
60  end tb;
61
62  -- Configuration block below is required by some simulators. Usually no need to edit.
63
64  configuration cfg_tb_lab01 of tb_lab01 is
```



It could print something if anything went wrong

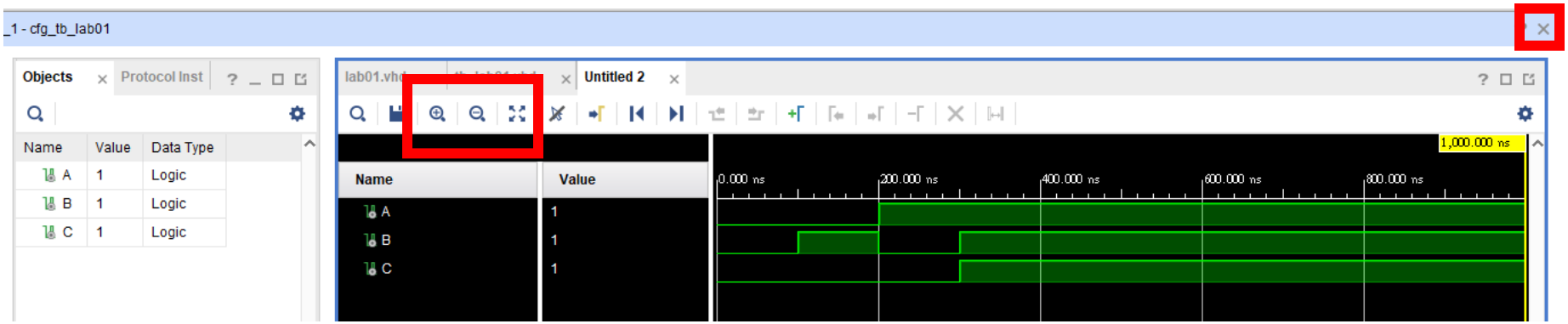
Step 5: Run Simulation

- Click “Run simulation” and select “Run Behavioral Simulation”



Step 5: Run Simulation

- Verify the output(s) by checking all possible input(s)
  - Click  first, and use  to adjust the timeline
  - If everything is alright, click X to exit



A	0	0	1	1
B	0	1	0	1
C	0	0	0	1

Lab01 Task

Lab01: 2-bit Comparator

- **Objectives and Aims:**
  - Get familiar with VHDL and Vivado
- **Requirements:**
  1. Create a new project named "lab01"
  2. Implement a **2-bit comparator** using VHDL
     - 3 outputs are required (Both are std_logic)
       - **less**: When A < B, the output would be 1, otherwise 0
       - **equal**: When A = B, the output would be 1, otherwise 0
       - **greater**: When A > B, the output would be 1, otherwise 0
     - Hint: Refer the 4-bit comparator in Lec01 (Page 31)
  3. Write a **testbench** file to test **all possible** input(s)
  4. Run the **behavioral simulation** and screen capture the resulting waveform

Hint

- **For 2.**
  One possible solution is using when else syntax

```
less <= '1' when (A < B) else '0';
equal <= '1' when (A = B) else '0';
greater <= '1' when (A > B) else '0';
```

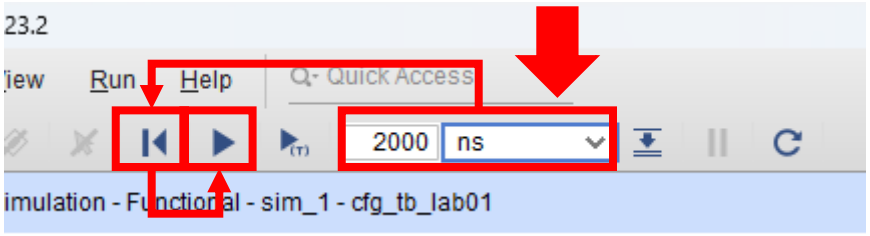
- **For 3.**
  You can check all test cases with this spreadsheet

	A <= "00";	A <= "01";	A <= "10";	A <= "11";
B <= "00";				
B <= "01";				
B <= "10";				
B <= "11";				

Lab01: 2-bit Comparator

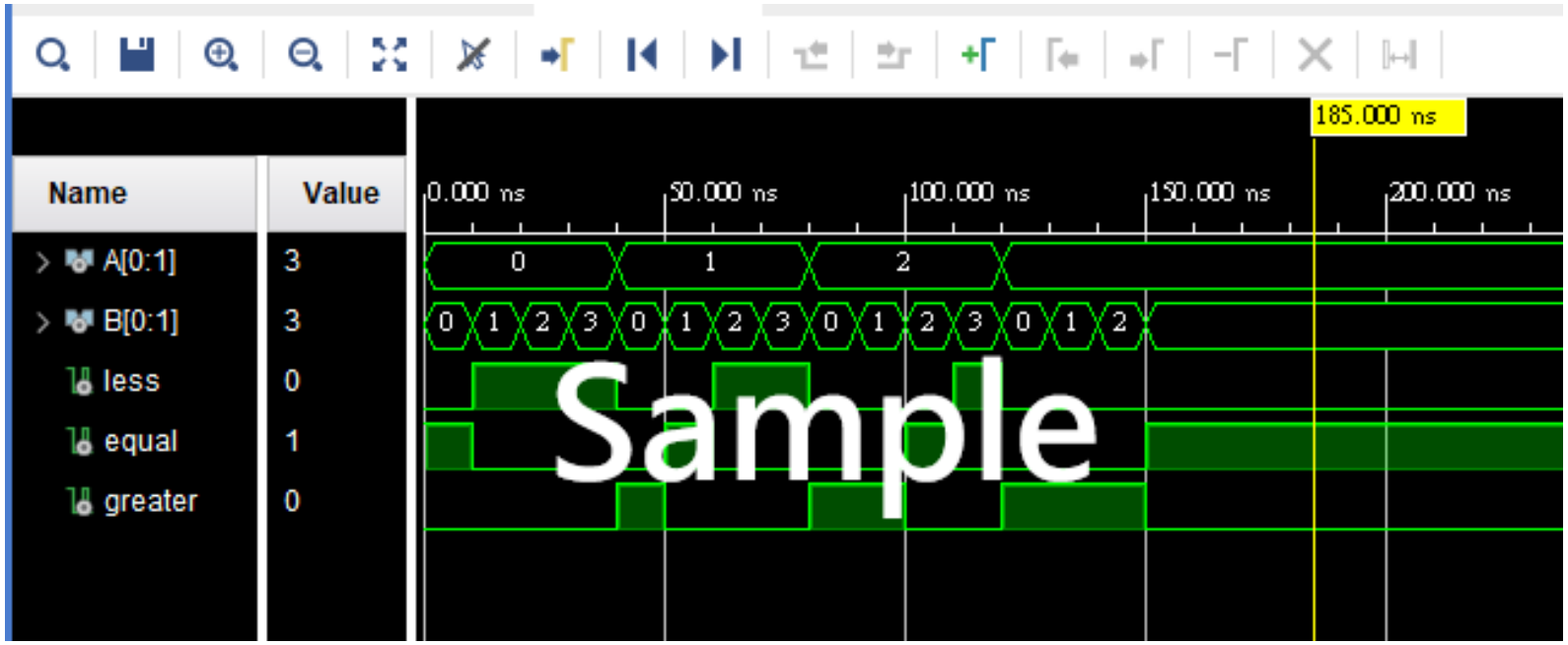
- One possible problem: The simulation might stop earlier than you expect

- Solution 1: Shorter the wait time (wait for 100ns -> 10ns)
- Solution 2:
  - Change to a longer duration
  - Click “Restart” then “Run All”

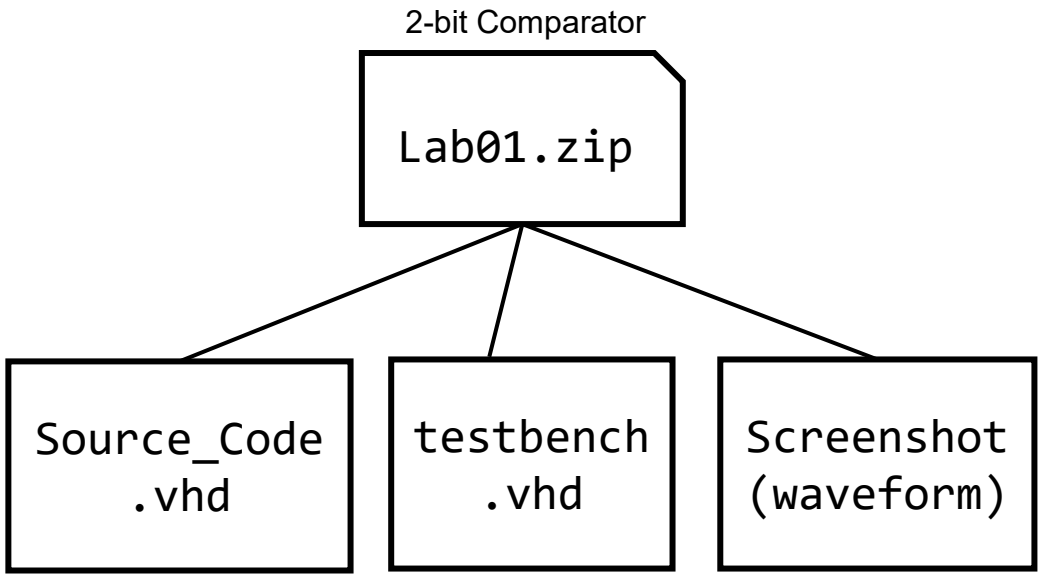


Lab01: 2-bit Comparator

- A sample simulation result of 2-bit comparator



Lab01: Submission File Checklist



Where to find the .vhd?

{Project Folder}
➔{Project Name}.srcs
➔sources_1
➔new
➔{source_code}.vhd
And
➔sim_1
➔new
➔{testbench}.vhd

- **Submission Rule:**
  1. Submit the zip file following the above format to Blackboard
     Deadline: **12:30 on 21 Jan. 2026**
     - Late submission is NOT acceptable (unless otherwise approved)

香港中文大學
The Chinese University of Hong Kong

Thank You!