



香港中文大學  
The Chinese University of Hong Kong

*CENG3430 Rapid Prototyping of Digital Systems*

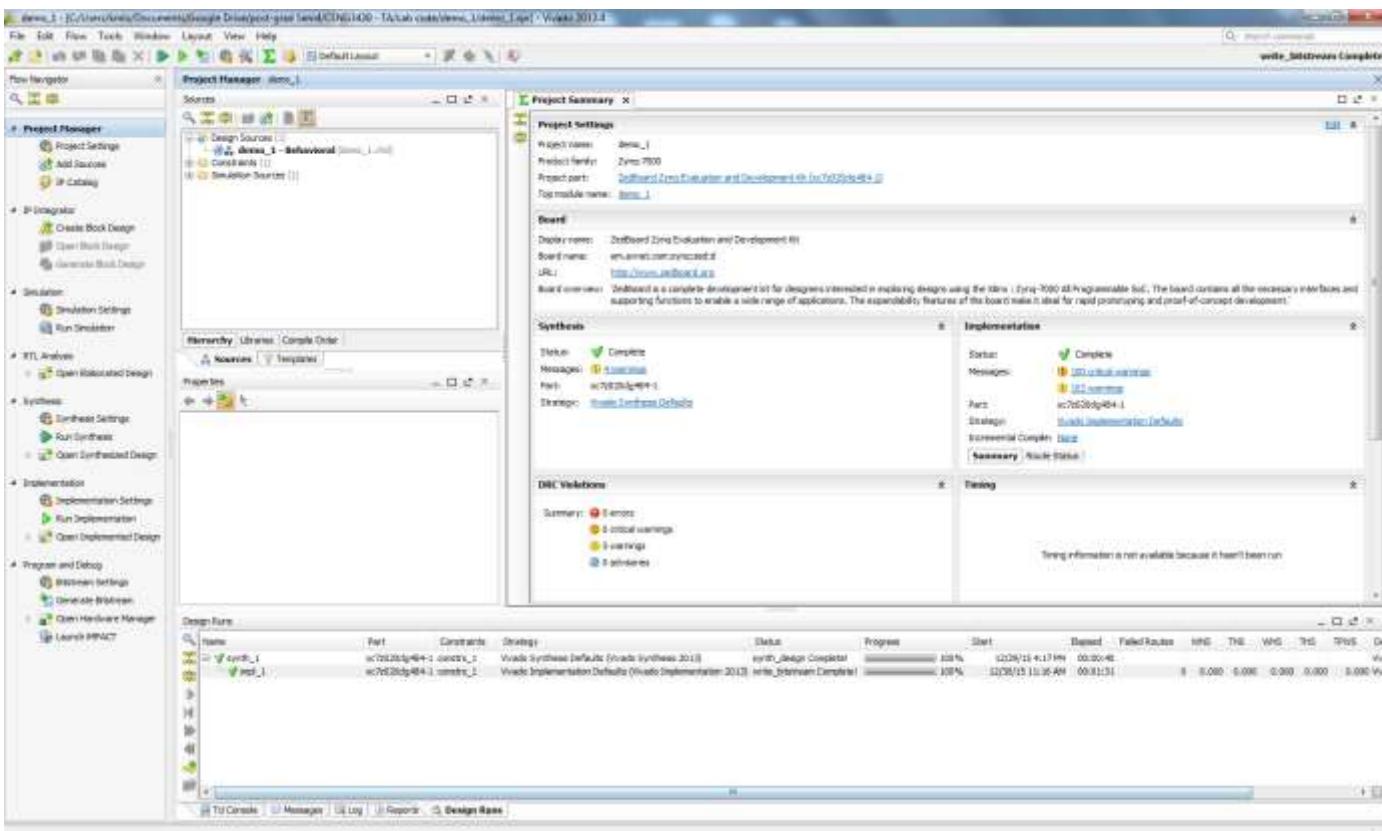
# Lab01: Software Simulation



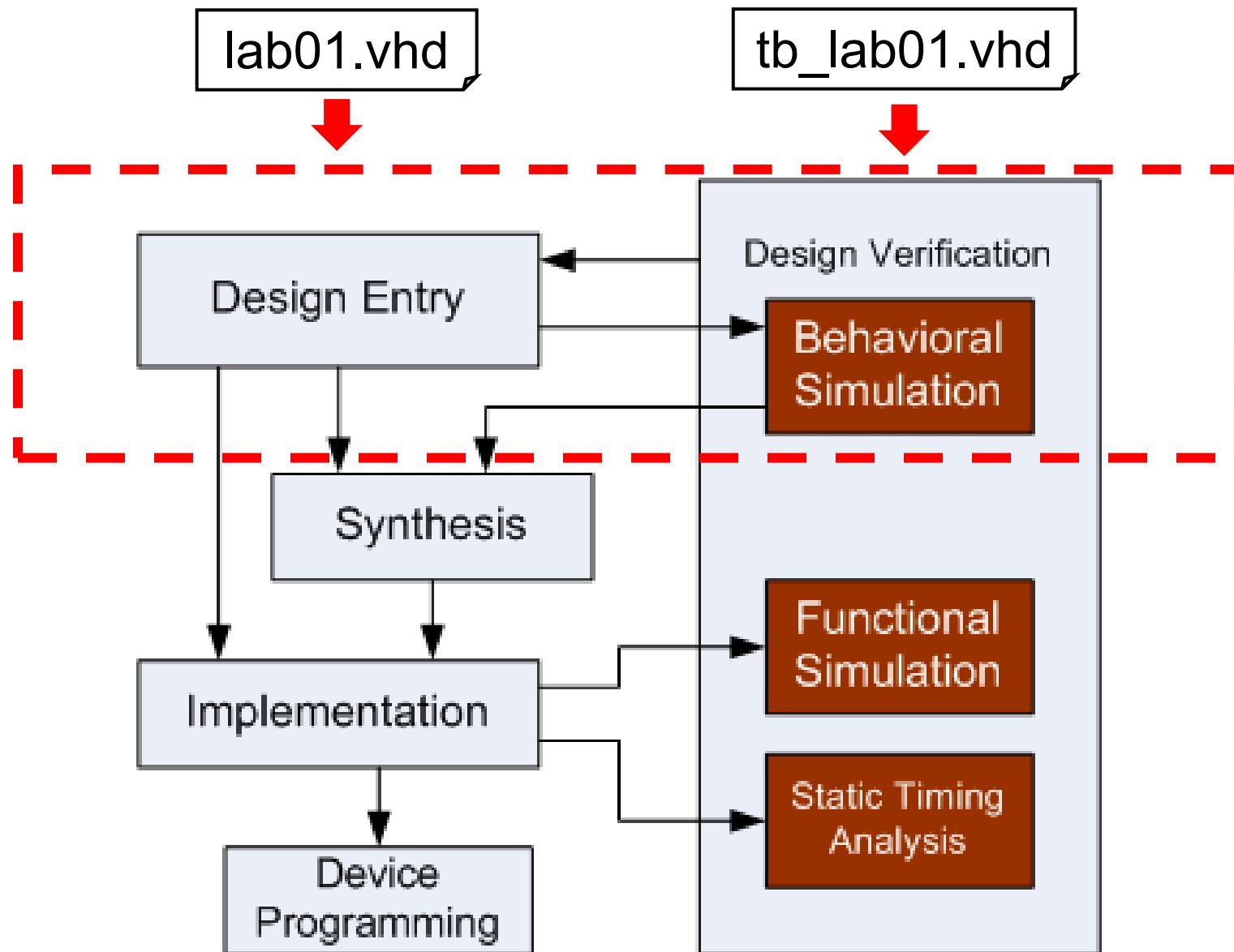


# Vivado

- 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)



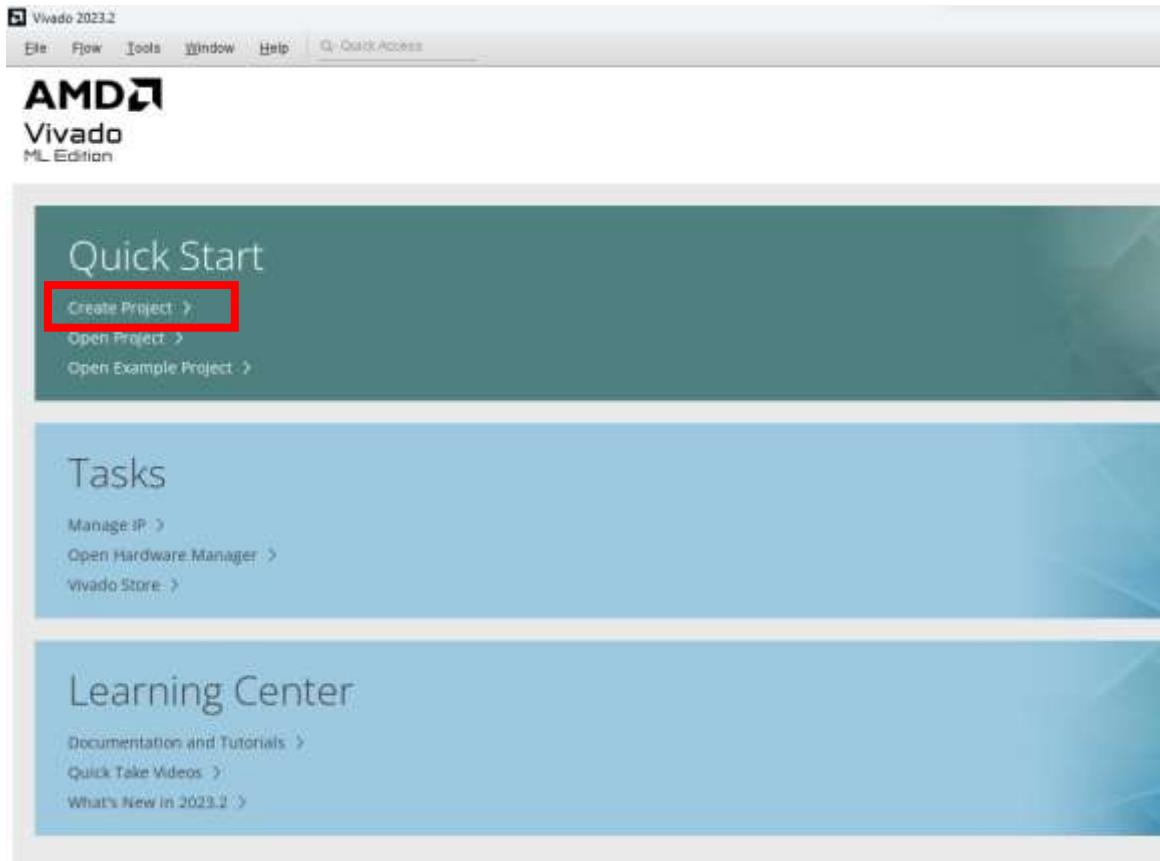
- Double click the icon of “Vivado”
- Or start “Vivado” from the Start Menu



# Step 2: Create a New Project (Lab00)



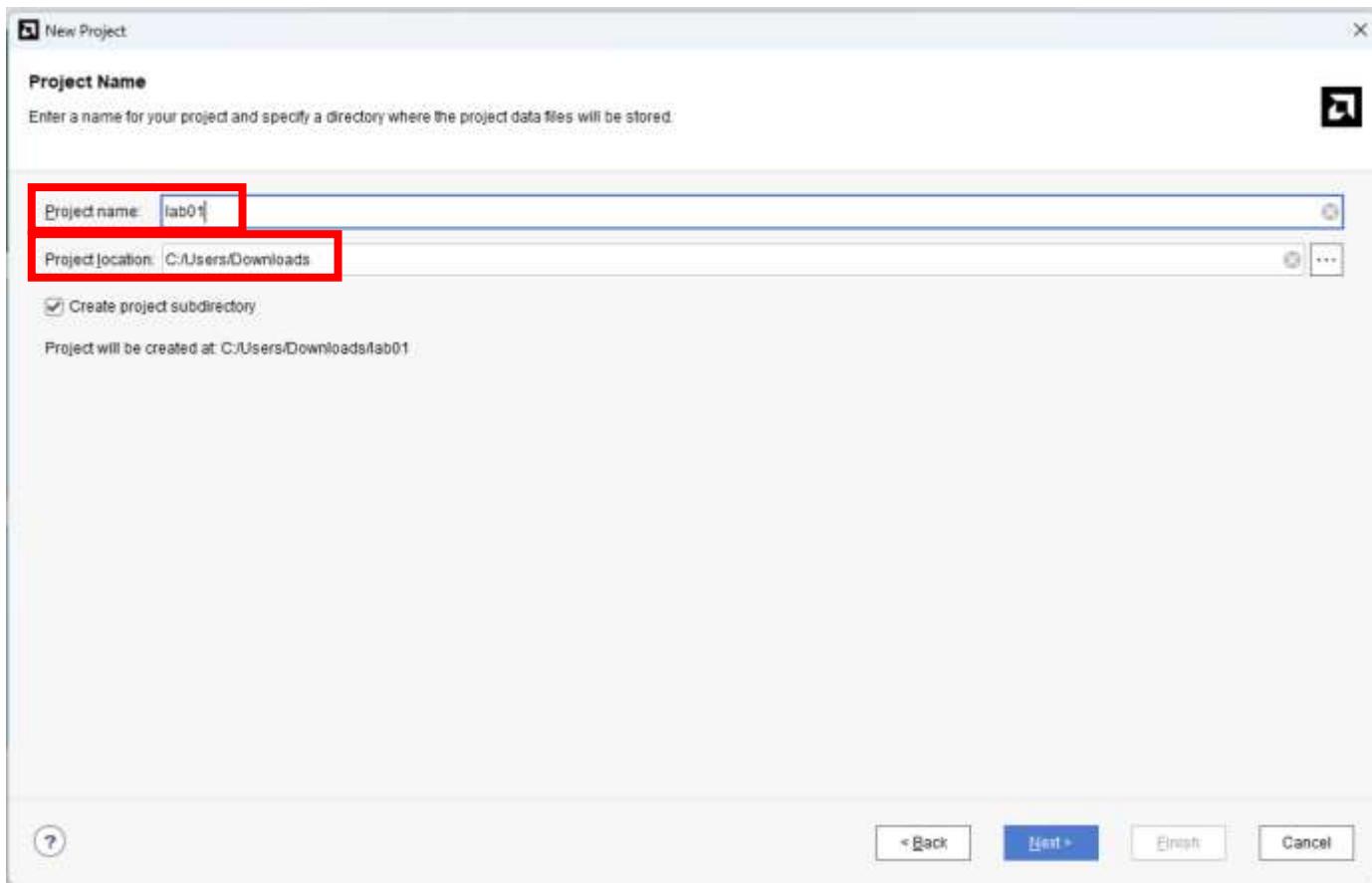
- Click “Create Project”
- Or click “File” -> “Project” -> “New”



# Step 2: Create a New Project (Lab00)



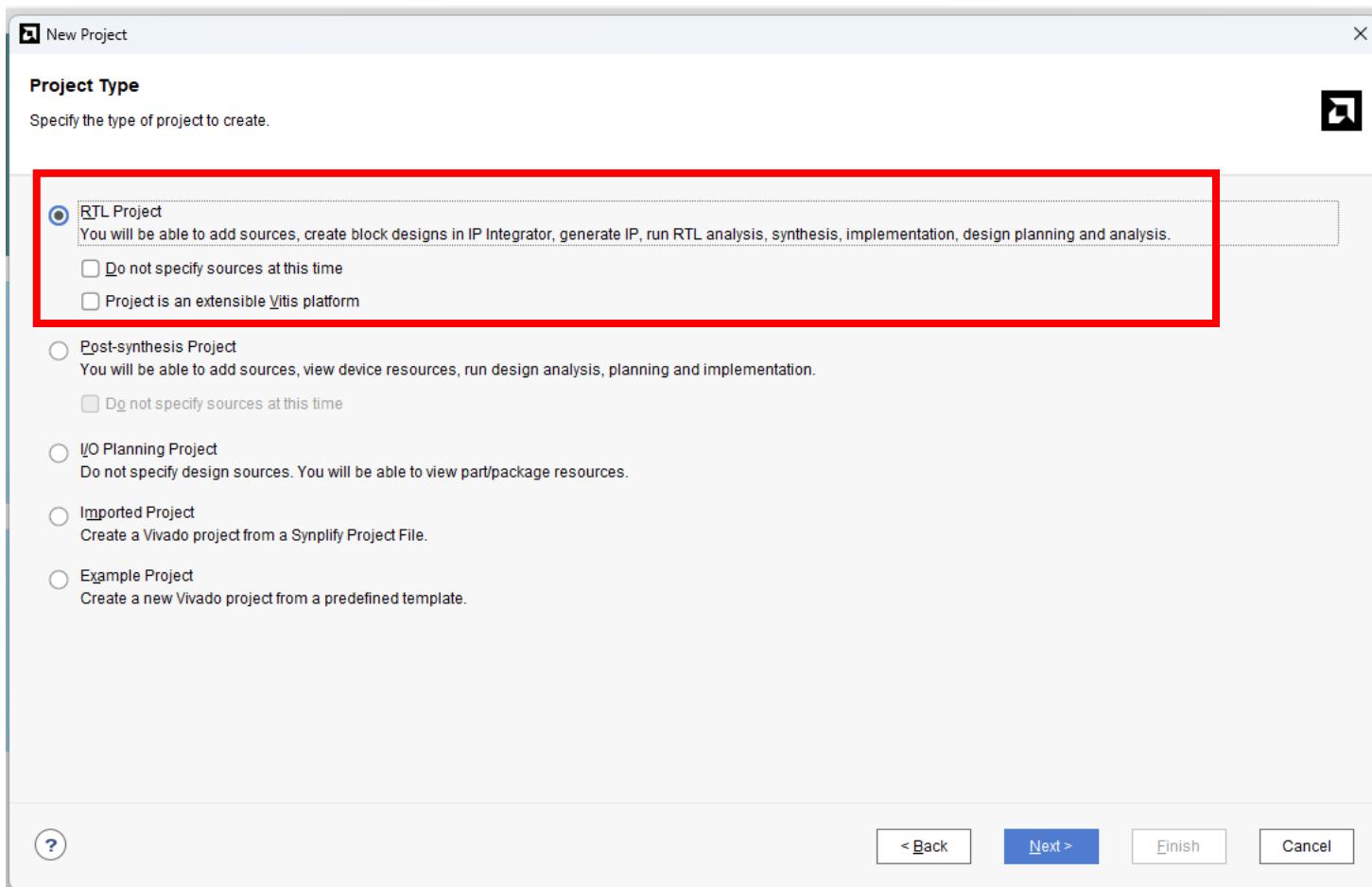
- Enter the “Project name” (e.g., lab01)
- Specify a proper “Project location” for storage



# Step 2: Create a New Project (Lab00)



- Select “RTL Project” -> “Next”



# Step 2: Create a New Project (Lab00)



1. Both choose “VHDL”

2. Click “Create File”

3. Name your file name

New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project and create sources later.

+ | - | ↑ | ↓

Use Add Files, Add Directories

Scan and add RTL include files

Copy sources into project

Add sources from subdirectories

Target language: VHDL Simulator language: VHDL

Create Source File

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

File type: VHDL

File name: lab01

File location: <Local to Project>

OK Cancel

Add Files Add Directories Create File

< Back Next > Finish Cancel

# Step 2: Create a New Project (Lab00)



- Click “Next”

New Project

Add Constraints (optional)  
Specify or create constraint files for physical and timing constraints.

+ | - | ↑ | ↓

Use Add Files or Create File buttons below

Add Files    Create File

Copy constraints files into project

?

< Back    Next >    Finish    Cancel

A screenshot of a software interface titled "New Project". It shows a section for "Add Constraints (optional)" with a note to "Specify or create constraint files for physical and timing constraints." Below this is a large empty list area with a toolbar above it containing buttons for adding, removing, and reordering items. A note says "Use Add Files or Create File buttons below" with two buttons shown: "Add Files" and "Create File". There is a checked checkbox for "Copy constraints files into project". At the bottom, there are navigation buttons: "< Back", "Next >" (which is highlighted with a red box), "Finish", and "Cancel".

# Step 2: Create a New Project (Lab00)



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

The screenshot shows the 'New Project' dialog with the 'Boards' tab selected. A red box highlights the 'Boards' tab. Another red box highlights the search bar containing 'Q- Zedboard'. A third red box highlights the 'Refresh' button at the bottom left.

**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: Q- Zedboard (3 matches)

Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Count	Board Rev	Available IOB
ZedBoard Zynq Evaluation and Development Kit <a href="#">Add Companion Card</a> <a href="#">Connections</a>		<a href="#">-</a>	avnet.com	1.4	xc7z020clg484-1	484	d	200
Zedboard		<a href="#">-</a>	digilentinc.com	1.0				

Refresh Catalog was last updated on 01/12/2024 5:51:58 PM

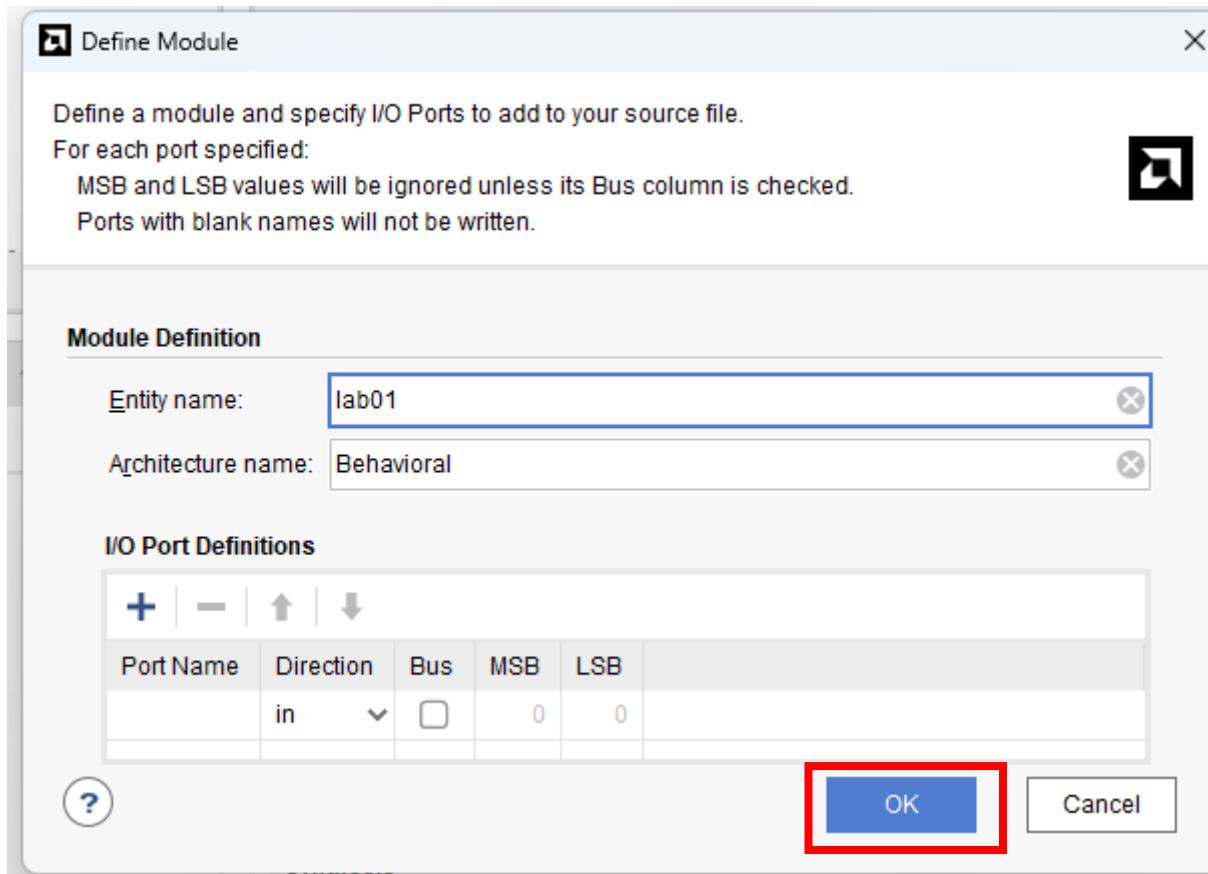
If not found, click “Refresh”  
Proxy is needed for lab PC to refresh  
Set it in “Tool” -> “Settings” -> “Vivado Store” -> “Configure Proxy”

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

# 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:

The screenshot shows the Vivado 2023.2 IDE interface. The left sidebar contains a tree view of project management, IP integrator, simulation, RTL analysis, synthesis, implementation, and program/debug options. The main area is divided into several tabs: Sources, Project Summary, Properties, Board Part, Synthesis, Implementation, and Design Runs.

**Sources Tab:**

- Design Sources (1): lab01 (lab01.srcs) (lab01.lib)
- Constraints
- Simulation Sources (1)
- Utility Sources

**Project Summary Tab:**

Projectname: lab01  
Project location: C:\Users\1155095176\Downloads\lab01  
Product family: Zynq-7000  
Project part: ZedBoard Zyng Evaluation and Development Kit (c7027d484-1)  
Top module name: Not defined  
Target language: VHDL  
Simulator language: VHDL

**Properties Tab:**

Display name: ZedBoard Zyng Evaluation and Development Kit  
Board part name: avnet.com:zedboard:part0:1.4  
Board revision: d  
Connectors: No connections  
Repository path: C:\Users\1155095176\AppData\Roaming\Xilinx\Vivado\2023.2\hub\board\_stencil\line\_board\_store  
URL: http://avnet.metaboard.com  
Board overview: ZedBoard Zyng Evaluation and Development Kit  
Changes

**Board Part Tab:**

Image of the ZedBoard Zyng Evaluation and Development Kit board.

**Synthesis Tab:**

To Console, Messages, Log, Reports, Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RDA Score	CoR Suggestions	LUT	FF	DRAM	SRAM	DSP	Start	Elapsed	Run Strategy
synth_1	constraints_1	Not started																			Vivado Synthesis Defaults (Vivado Synthesis 2023)
impl_1	constraints_1	Not started																			Vivado Implementation Defaults (Vivado Implementation 2023)

**Implementation Tab:**

To Console, Messages, Log, Reports, Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RDA Score	CoR Suggestions	LUT	FF	DRAM	SRAM	DSP	Start	Elapsed	Run Strategy
synth_1	constraints_1	Not started																			Vivado Synthesis Defaults (Vivado Synthesis 2023)
impl_1	constraints_1	Not started																			Vivado Implementation Defaults (Vivado Implementation 2023)



## II. Write the VHDL Code (Lab00)

# Step 3: Start Programming (Lab00)



- This is the programming interface of Vivado:

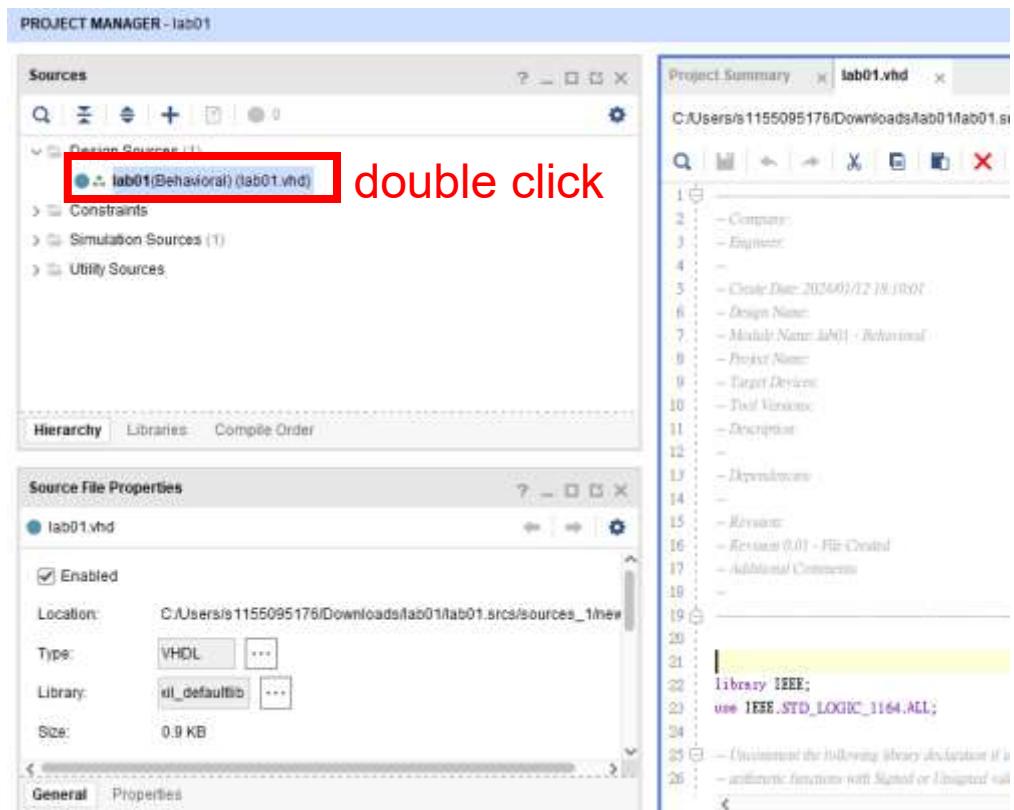
The screenshot shows the Vivado 2023.2 IDE interface. On the left, a vertical toolbar titled "Shortcuts of Different Tools" lists various design flows: Project Manager, IP Integrator, Simulation, RTL Editor, Synthesis, Implementation, and Program and Debug. The "Project Manager" section is currently active, displaying a "Source Directory" tree with a single entry: "lab01 (initial state) (lab01.lib)". The main workspace is divided into several panes:

- Source Directory**: A tree view of project files.
- Properties**: A panel showing properties for selected items in the source directory.
- Code Writing Area**: A large pane where VHDL code is being written. It includes sections for "Board Part" (ZedBoard Zynq Evaluation and Development Kit) and "Synthesis" (Implementation).
- IDE Status / Warning / Error Message**: A bottom pane showing the status of design runs, with one entry: "synth\_1" (status: Not started).

# 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)

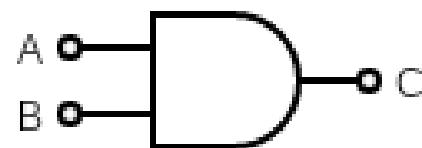


- Implement a simple AND logic and **save**

```
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...”

The screenshot shows the Quartus Project Manager interface. On the left, the 'PROJECT MANAGER - lab01' window displays the 'Sources' panel. The 'Simulation Sources' folder (containing 'lab01.vhd') is selected and highlighted with a red box. A context menu is open over this folder, with the 'Add Sources...' option also highlighted with a red box. On the right, the 'Project Summary' window shows the VHDL code for 'lab01.vhd'. The code defines an entity 'lab01' with three ports: A, B, and C, and an architecture 'Behavioral' with a simple assignment statement C <= A AND B.

```
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;

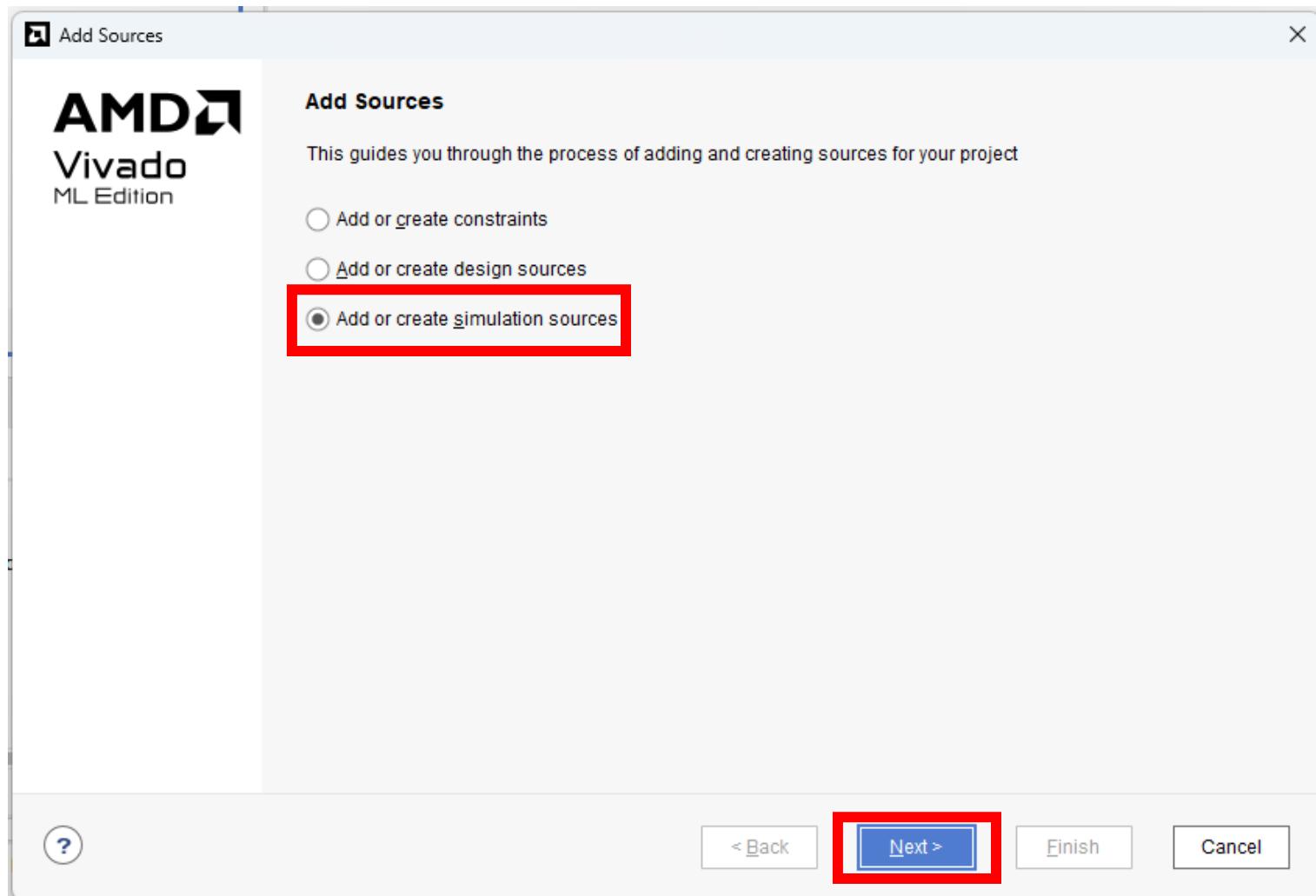
architecture Behavioral of lab01 is

begin
    C <= A AND B;
end Behavioral;
```



# Step 4: Write a Testbench

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





# Step 4: Write a Testbench

The screenshot shows a software interface for managing simulation sources. On the left, the 'Add Sources' dialog box is open, displaying a list of simulation sets (sim\_1) and options for adding files or directories. A red arrow points from the 'Create File' button in this dialog to the 'Create Source File' dialog box on the right. The 'Create Source File' dialog box has 'VHDL' selected as the file type and 'tb\_lab01' entered as the file name. A red box highlights the 'File name' field. Another red arrow points from the 'OK' button in the 'Create Source File' dialog back to the 'Create File' button in the main dialog. The 'Create File' button is also highlighted with a red box.

**Add or Create Simulation Sources**  
Specify simulation specific HDL files, or directories containing HDL files, to add to your project.

Specify simulation set: sim\_1

+ | - | ↑ | ↓

2. Name it tb\_lab01  
(tb stands for testbench)

Use Add Files, Add Directories or

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

Add Files    Create File    OK    Cancel

?

< Back    Next >    Finish    Cancel

Create Source File

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

File type: VHDL

File name: tb\_lab01

File location: <Local to Project>

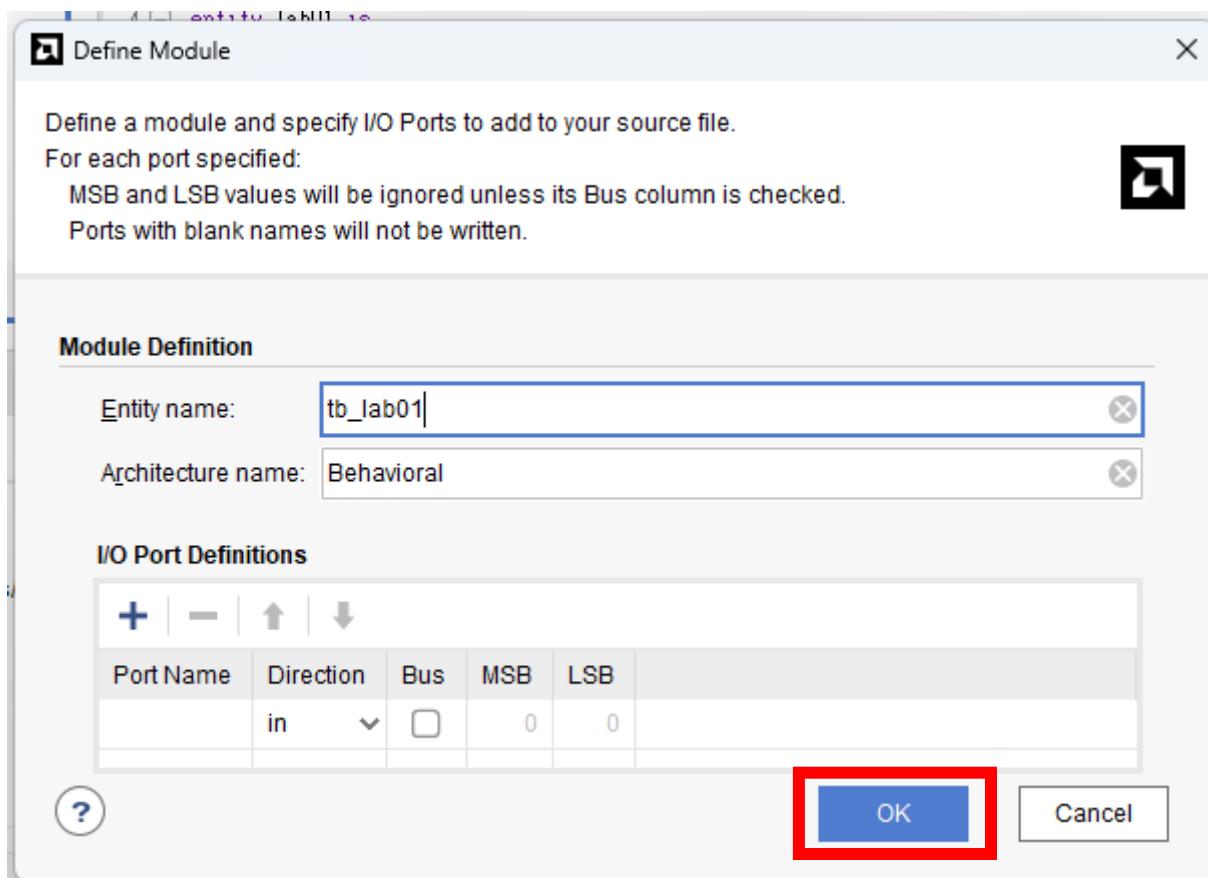
?

OK    Cancel



# 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

The screenshot shows a VHDL development environment with three main windows:

- Sources Browser:** Shows the project structure. A file named "cfg\_tb\_lab0 (tb\_lab01 - tb) (tb\_lab01.vhd) (1)" is selected. A red box highlights this file, and another red box highlights the entity declaration "entity tb\_lab01 is" in the code editor below.
- Project Summary:** Displays the current project path: C:/Users/s1155095176/Downloads/lab01/lab01.srcts/sim\_1/new/. It also shows generation details: "Testbench automatically generated online" at https://vhdl.lapinoo.net, and "Generation date : 12.1.2024 10:30:42 UTC".
- Code Editor:** Shows the VHDL code for the testbench. The entity declaration "entity tb\_lab01 is" is highlighted with a red box. A red arrow labeled "Match" points from the highlighted file in the Sources browser to this entity declaration in the code editor.

```
1 -- Testbench automatically generated online
2 -- at https://vhdl.lapinoo.net
3 -- Generation date : 12.1.2024 10:30:42 UTC
4
5 library ieee;
6 use ieee.std_logic_1164.all;
7
8 entity tb_lab01 is
9 end tb_lab01;
10
11 architecture tb of tb_lab01 is
12
13 component lab01
14     port (A : in std_logic;
15             B : in std_logic;
16             C : out std_logic);
17 end component;
18
19 signal A : std_logic;
20 signal B : std_logic;
21 signal C : std_logic;
22
23 begin
24
25     dut : lab01
26     port map (A => A,
```



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

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 ?

The screenshot shows the Xilinx Vivado IDE interface. On the left, the 'Properties' panel is open for 'tb\_lab01.vhd', showing it is enabled, located at C:/Users/s1155095176/Downloads/lab01/lab01.srcs/sim\_1/new/tb\_lab01.vhd, and is a VHDL file in the xil\_defaultlib library. On the right, the code editor displays a VHDL testbench script. A red box highlights the following code block:

```
48 : assert(C = 'U')  
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  
65 :   for tb
```

Below the code editor is the 'Tcl Console' window. A red box highlights the error message 'Error: Test failed for input 11' in the console output, which is preceded by a timestamp and iteration information.

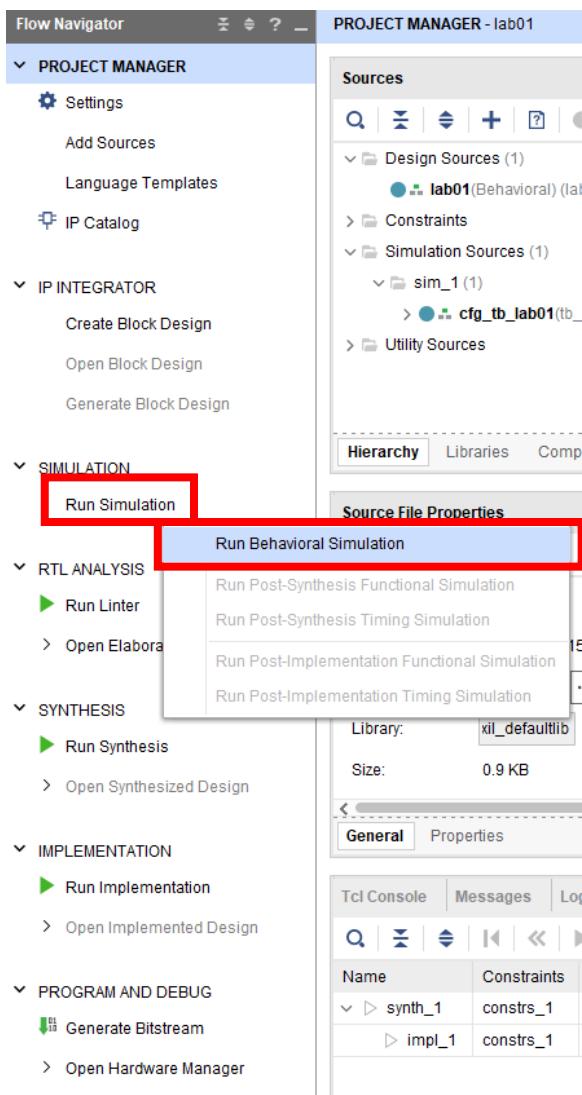
It could print something if anything went wrong

```
# }  
# }  
# 100 ns  
Error: Test failed for input 11  
TIME: 400 ms Iteration: 0 Process: ./tb_lab01/stimuli File: C:/Users/s1155095176/Downloads/lab01/lab01.srcs/sim_1/new/tb_lab01.vhd  
INFO: [USF-XSim-96] XSim completed. Design snapshot 'cfg_tb_lab01_behav' loaded.  
INFO: [USF-XSim-97] XSim simulation ran for 1000ns  
launch_simulation: Time (s): cpu = 00:00:00 ; elapsed = 00:00:07 . Memory (MB): peak = 2514.852 ; gain = 14.949  
close_sim
```



# Step 5: Run Simulation

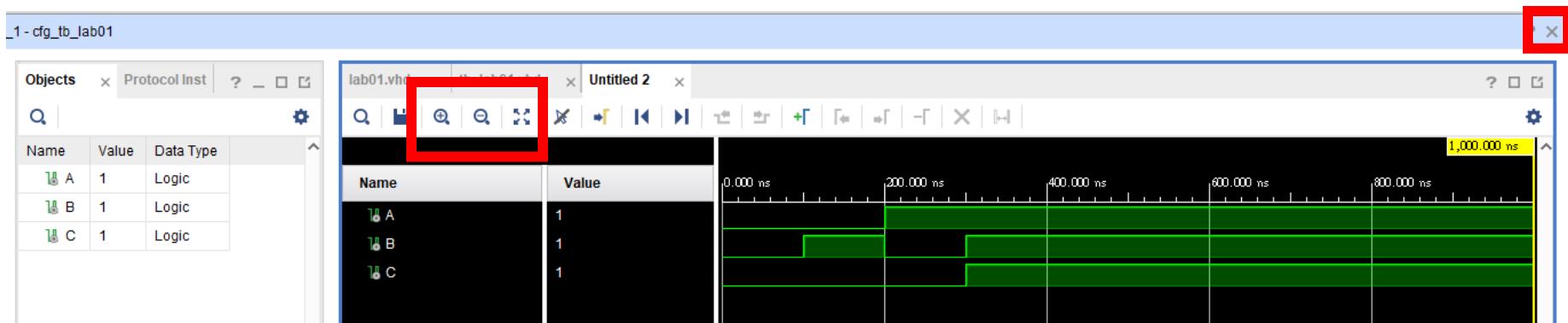
- Click “Run simulation” and select “Run Behavioral Simulation”





# 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";				

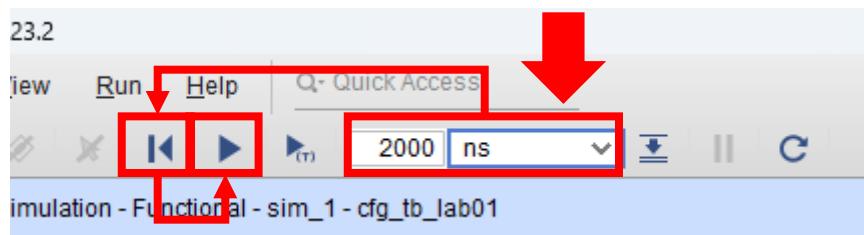


# Lab01: 2-bit Comparator

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



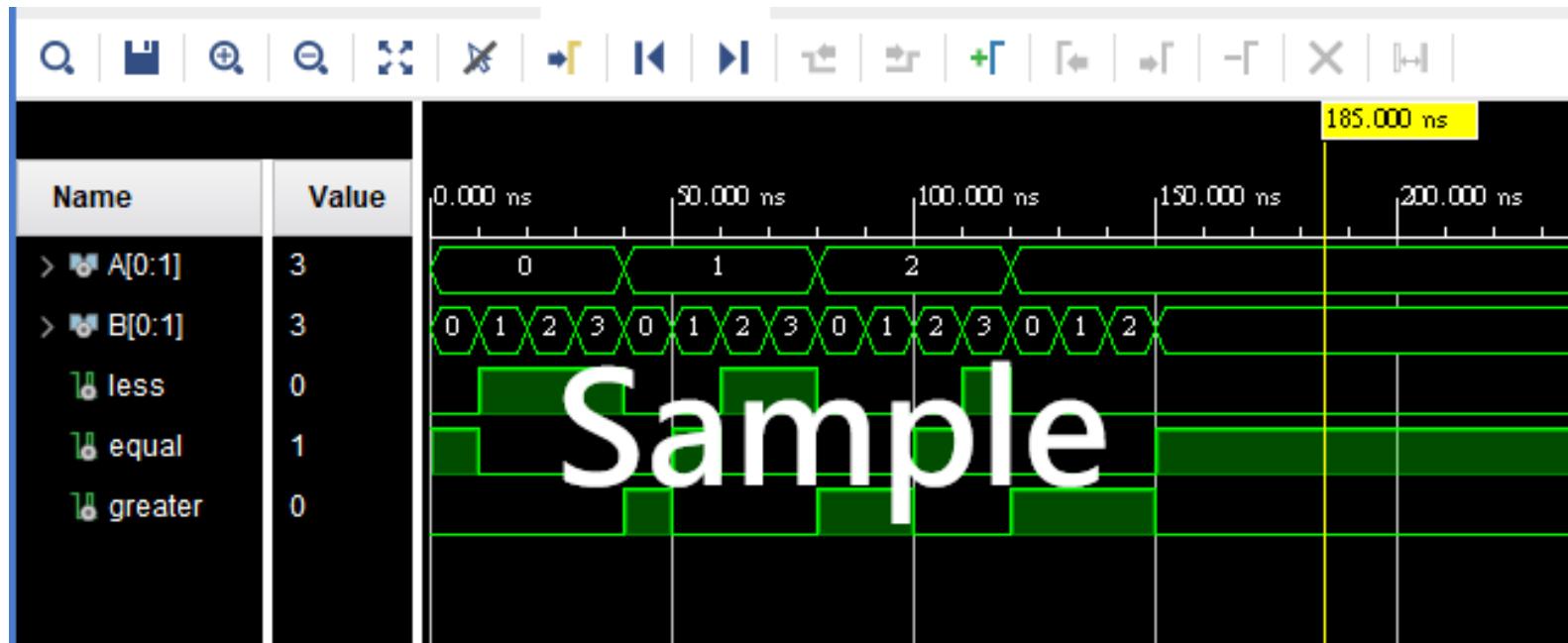
- Solution 1: Shorter the wait time (wait for 100ns -> 10ns)
- Solution 2:
  - Change to a longer duration
  - Click “Restart” then “Run All”



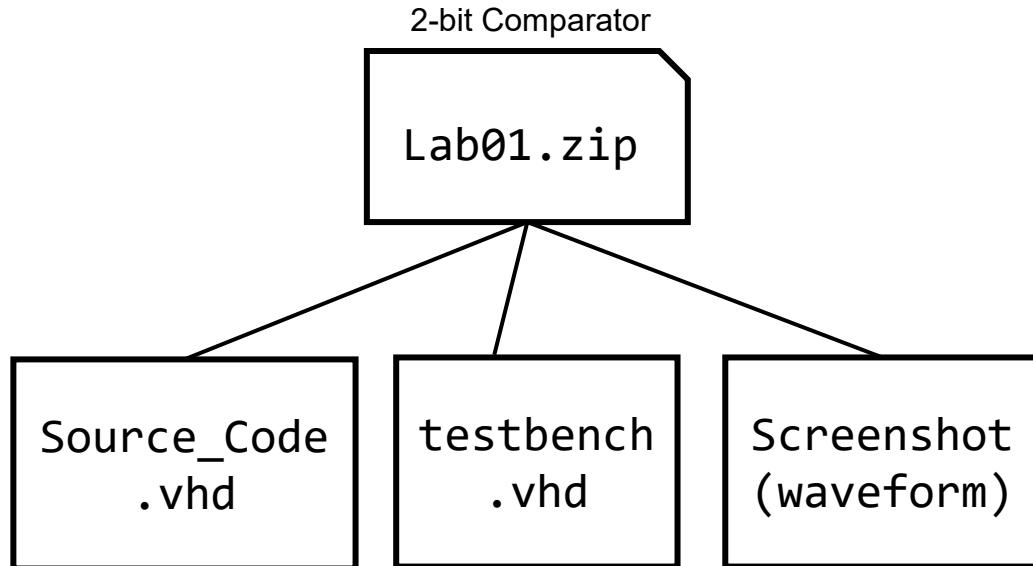


# Lab01: 2-bit Comparator

- 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!

