Lab 1 Practice: Gate-Level Designs

You do NOT need to submit this practice

Objective

1. Getting familiar with basic Verilog gate-level modeling for combinational/sequential circuits.
2. Practicing modular and hierarchical design in Verilog.

Prerequisites

Concept	Video Lectures	Note
Basic Verilog coding & structural modeling	CT Verilog Series 00~03	
Module instantiation & hierarchical design	CT Verilog Series 06 (22:36~28:25)	You may skip the Dataflow and Behavioral parts for this practice.

Action Items

1 `practice_1.v`

Design a function selector implemented with 2-to-1 multiplexer (MUX) using **gate-level** modeling.

Recall from your Logic Design course that a **2-to-1 multiplexer (MUX)** is a digital circuit that selects one of two input signals (**a** or **b**) and forwards it to the output based on a control signal **c**. Specifically, when **c** = 0, the output will be **a**; when **c** = 1, the output will be **b**.

In this lab, you are required to implement a simple function selector using a 2-to-1 MUX. The MUX takes the outputs of an **AND gate** and an **OR gate** as inputs, which both operate on the same input signals (**A** and **B**). The control signal **C** determines which operation's result—AND or OR—will be selected as the final output. This setup allows switching between logical functions dynamically based on the control signal.

The **practice_1.v** file is divided into two parts:

(i) Implement a **2-to-1 MUX** by primitive gates.

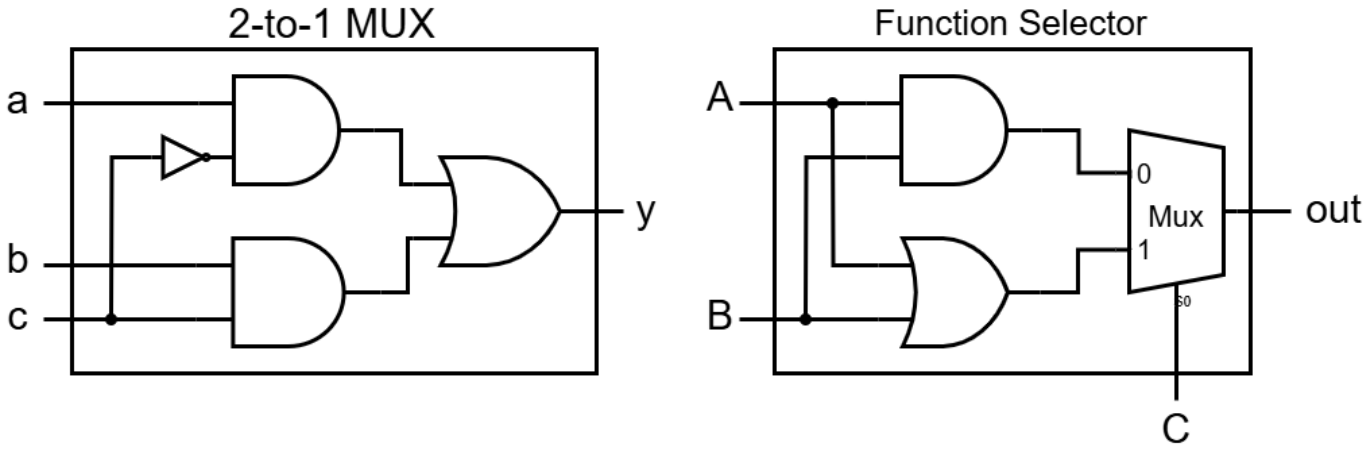
(ii) Instantiate the 2-to-1 MUX module created in (i) and implement the function selector.

a. IO list:

Signal Name	Definition
A	Input (operand 1)
B	Input (operand 2)
C	Input (select signal)
out	Output

b. Implementation Details:

You may come up with your own design, or you may follow the block diagram below to implement a 2-to-1 MUX and the function selector.



For example, if the inputs are (**A**, **B**) = (0, 1), the AND gate produces 0 and the OR gate produces 1. When the control signal **C** = 0, the multiplexer selects the output of the AND gate, resulting in a final output of 0.

Truth Table

Inputs			Outputs
A	**B**	**C**	**out**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

c. Note that only **gate-level** descriptions (e.g., **AND**, **OR**, **NAND**, **NOR**, **NOT, …**) are permitted. **Behavior-level** constructs such as **if-else** statements are not allowed in practice_1.v.

d. You must use the following I/O definition for your design. Remember to remove the blue-colored comments.

```
`timescale 1ns/100ps
module practice_1 (
  input wire A,
  input wire B,
  input wire C,
  output wire out
);
    // Write your code here
endmodule

module mux (
  input wire a,
  input wire b,
  input wire c,
  output wire y
);
    // Write your code here
endmodule
```

2 practice_1_t.v

Complete the testbench **practice_1_t.v** to verify your design. Check the TODO hints in the template code carefully. For incorrect results, you may see error messages like this:

```
Error: a=X, b=X, c=X, y=X
Correct answer should be X
```

or

```
Error: A=X, B=X, C=X, out=X
Correct answer should be X
```

where X is the corresponding data bit.

3 practice_2.v

Write a Verilog module that models a **gated D latch** using **gate-level** description only.

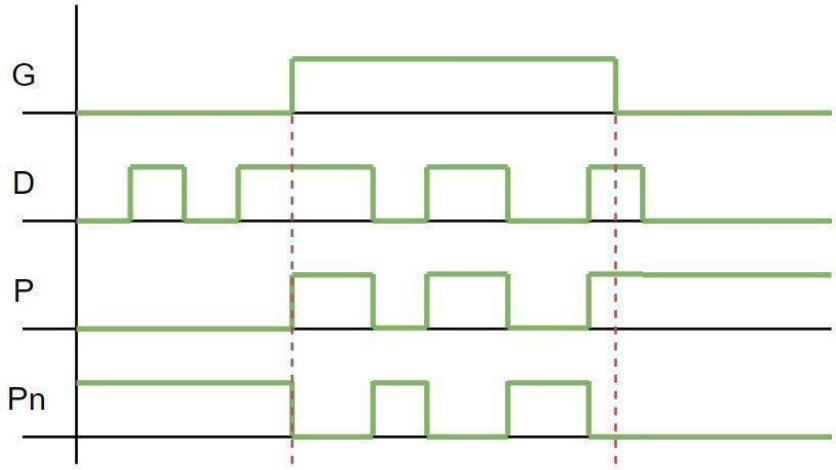
a. IO List:

Signal Name	Definition
G	Gate input.
D	Data input.
P	Output.
Pn	Complementary output.

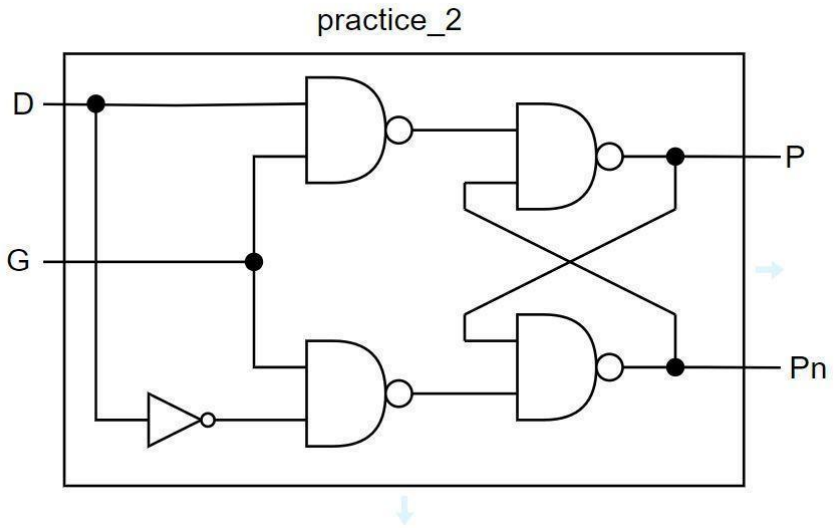
b. Implementation Details:

Recall from your Logic Design course that a gated D latch is a sequential circuit that samples and stores the value of the data input **D** when the Gate input **G** is set to 1. Specifically, when **G** is 1, the gated D latch samples the value of **D** and outputs it through **P**. When **G** is 0, the gated D latch preserves its last sampled value at **P** until the gate **G** is set to 1 again. Additionally, the gated D latch also outputs **Pn**, which is the inverse of **P**.

For example, in the following timing diagram we can see that when **G** is set to 1, the output **P** simply follows the data input **D**. When **G** is later set to 0, the output **P** remains its last sampled value, which is 1.



You may come up with your own design, or you may follow the block diagram below to implement a gated D latch.



c. You must use the following I/O definition for your design. Remember to remove the blue-colored comments.

```
`timescale 1ns/100ps
module practice_2 (
  input wire G,
  input wire D,
  output wire P,
  output wire Pn
);
  // Write your code here
endmodule
```

4 practice_2_t.v

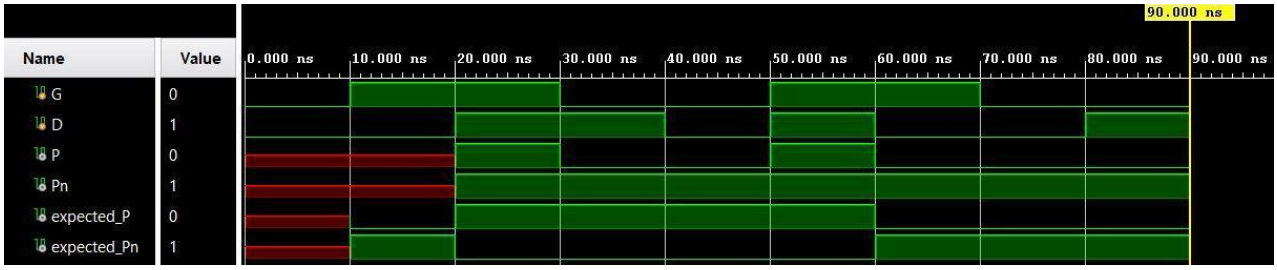
Complete the testbench **practice_2_t.v** to verify your design. Check the TODO hints in the template code carefully. For incorrect results, you may see error messages like this:

```
[ERROR]: There were 8 errors.
     Gate input G: 0 -> 1 -> 1 -> 0 -> 0 -> 1 -> 1 -> 0 -> 0
     Data input D: 0 -> 0 -> 1 -> 1 -> 0 -> 1 -> 0 -> 0 -> 1
    Your output P: x -> x -> 1 -> 0 -> 0 -> 1 -> 0 -> 0 -> 0
   Your output Pn: x -> x -> 1 -> 1 -> 1 -> 1 -> 1 -> 1 -> 1
 Correct output P: x -> 0 -> 1 -> 1 -> 1 -> 1 -> 0 -> 0 -> 0
Correct output Pn: x -> 1 -> 0 -> 0 -> 0 -> 0 -> 1 -> 1 -> 1
```

In this error message, for example, one of the errors can be spotted on the 2nd column: when G is 1 and D is 0 on the 2nd column, the correct output P should be 0 instead of x.

If you think the error message is too hard to read, you may also use the waveform

viewer in Vivado. For the error message example above, you will see the following waveform.

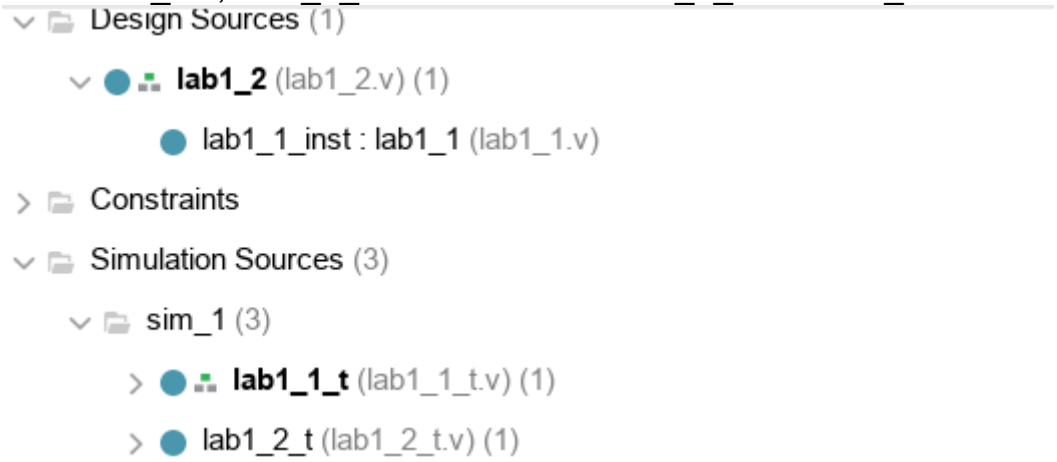


Attention

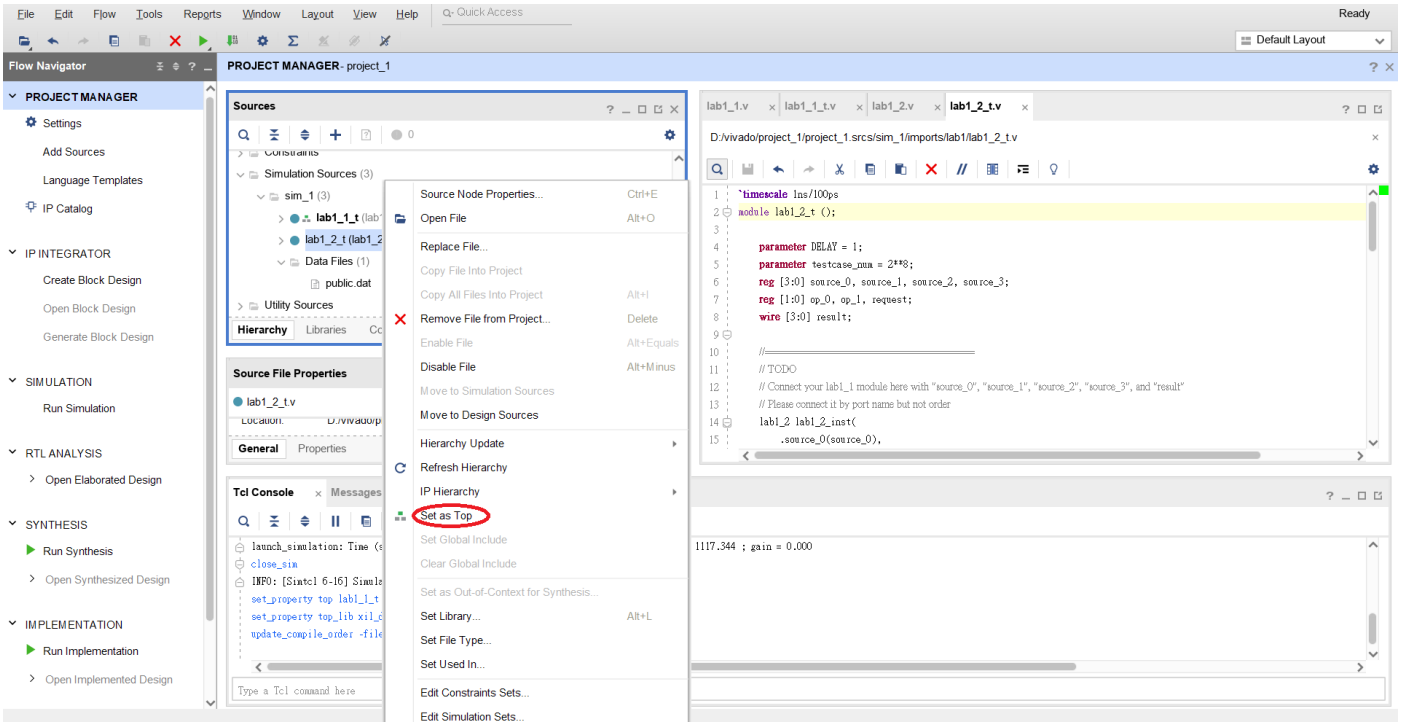
- You are only allowed to use Gate-level modeling, Dataflow modeling and Behavioral modeling are NOT permitted in this lab.
- DO NOT copy-and-paste code segments from the PDF materials to compose your design. It may also paste invisible non-ASCII characters, leading to hard-to-debug syntax errors.
- You may also add a **$monitor** in the testbench to show all the inputs and outputs during the simulation.
- Please note that this practice is highly associated with lab1.

Appendix

- We can add two or more testbenches in one single project, e.g., lab1_1_t and lab1_2_t, in the following example. Different testbenches can provide various test patterns, or even integrate different design modules to test. In this example, lab1_1_t.v is used to test lab1_1.v; lab1_2_t.v is used to test lab_1_2.v + lab1_1.v.

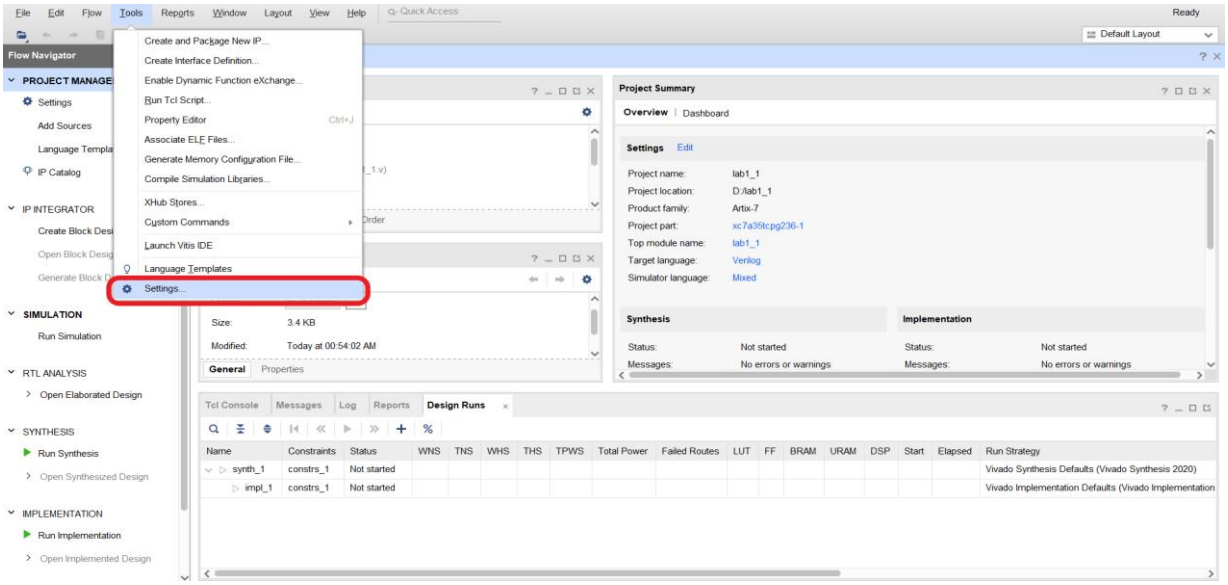


Right click with the mouse on the testbench and select "Set as Top" to activate it. In the following example, we activate the lab1_2_t.v and run the corresponding simulation. (Note: if the two testbenches use the identical top module name, you may need to “Disable File” first in order to activate the other one).

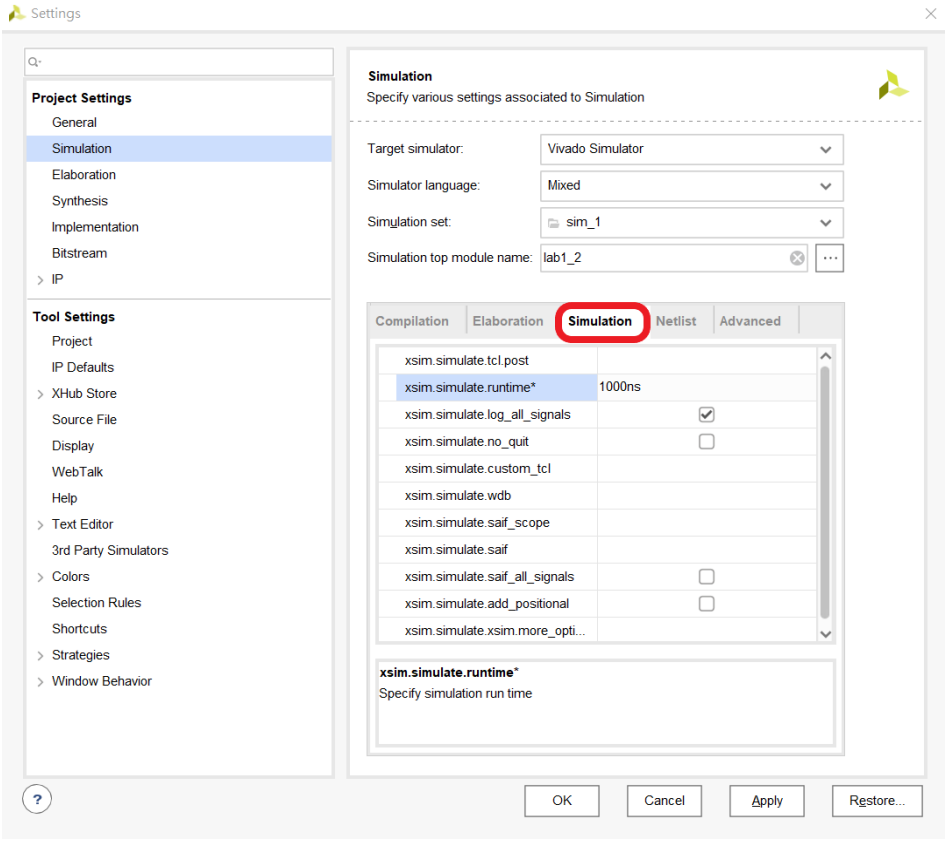


➢ Follow these steps to increase your simulation runtime if you need to.

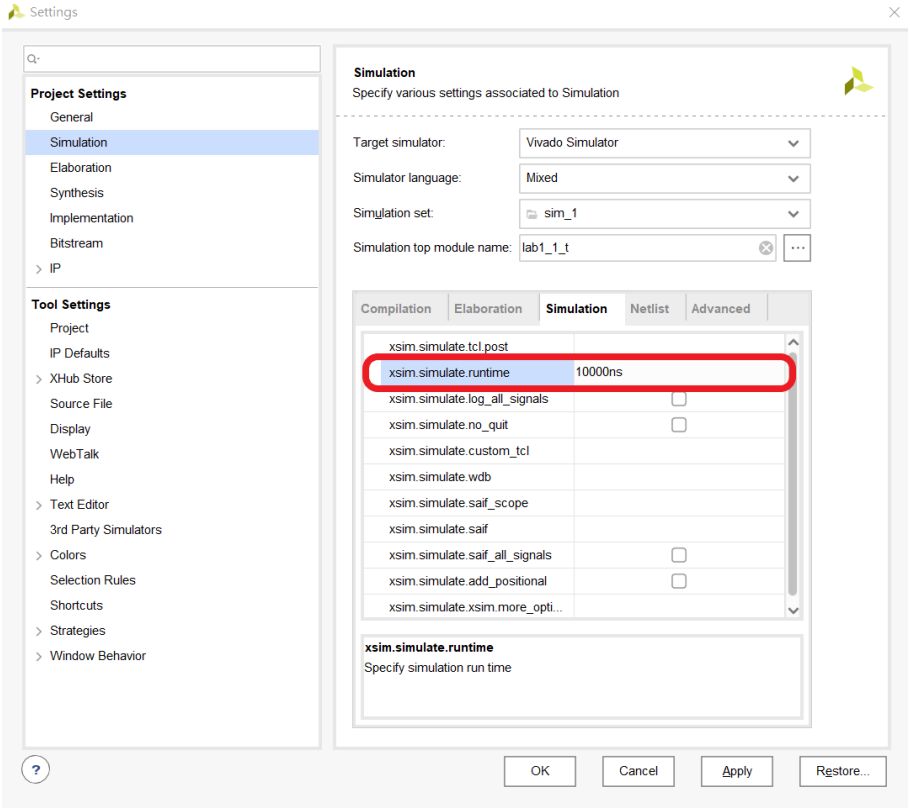
1. Click Tools at the top of Vivado and select settings.



2. Select Simulation.



3. Change runtime to 10000ns or a larger value.



4. Remember to click Apply.