Lab 1: ALU Designs

Submission Due Dates:

Demo: 2023/09/19 17:20
Source Code: 2023/09/19 18:30
Report: 2023/09/24 23:59

Objective

Getting familiar with basic Verilog behavior modeling.

Action Items

1 lab1_1.v (20%)

Write a Verilog module and test your module by using the testbench file **lab1_1_t.v**.

a. IO list:

- ✓ Input: **op[1:0], a[3:0], b[3:0]**
- ✓ Output: **d[3:0]**

Perform the operations based on the operation op[1:0].

Operation op[1:0]	Operation
2'b00	d = a AND b
2'b01	d = a << b (ignoring the overflow)
2'b10	d = a OR b
2'b11	d = a >> b

b. You must use the following template for your design. Remember to remove the blue-colored comments.

```
`timescale 1ns/100ps
module lab1_1 (
  input wire [1:0] op,
  input wire [3:0] a,
  input wire [3:0] b,
  output reg [3:0] d
);
  /* Note that d can be either reg or wire.
   * e.g.,    output reg [3:0] d
   * or       output wire [3:0] d
   * It depends on how you design your module. */
  // Add your design here

endmodule
```

2 lab1_1_t.v (20%)

Complete the testbench **lab1_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:  op = XXXX, d = XXXX, correct d should be XXXX
```

where X is the corresponding data bit.

3 lab1_2.v (30%)

Write a Verilog module that models a **4-bit Arithmetic Logic Unit (ALU)** and test your module using the testbench **lab1_2_t.v**. The module **lab1_1** must be reused in **lab1_2**. You must finish both lab1_1 and lab1_2 in one Vivado project.

a. IO list:

- ✓ Inputs: **source_0[3:0]**, **source_1[3:0]**, **source_2[3:0]**, **source_3[3:0]**, **op_0[1:0]**, **op_1[1:0]**, **request[1:0]**

  There are four 4-bit source ports (e.g., **source_X**), two op ports (e.g., **op_X**), and one 2-bit request.

  When request[0]==1, it indicates that there is a request for op_0. Similarly, request[1]==1 means that there is a request for op_1. If there are two requests at the same time, op_0 has a higher priority.

  If there is a request for op_0, perform the operations based on the operation op_0[1:0].

Operation op_0 [1:0]	Operation
2'b00	result = source_0 AND source_1
2'b01	result = source_0 << source_1 (ignoring the overflow)
2'b10	result = source_0 OR source_1
2'b11	result = source_0 >> source_1

  If there is a request for op_1, perform the operations based on the operation op_1[1:0].

Operation op_0 [1:0]	Operation
2'b00	result = source_2 AND source_3
2'b01	result = source_2 << source_3 (ignoring the overflow)
2'b10	result = source_2 OR source_3
2'b11	result = source_2 >> source_3

  typo

  If there isn't any request, the result should be 0.

- ✓ Output: **result[3:0]**

  The 4-bit result.

Remember to reuse lab1_1 module.

b. You must use the following template for your design:

```
`timescale 1ns/100ps
module lab1_2 (
  input wire [3:0] source_0,
  input wire [3:0] source_1,
  input wire [3:0] source_2,
  input wire [3:0] source_3,
  input wire [1:0] op_0,
  input wire [1:0] op_1,
  input wire [1:0] request,
  output reg [3:0] result
);
  /* Note that result can be either reg or wire.
   * It depends on how you design your module. */
  // add your design here
endmodule
```

c. Follow the Appendix to extend the simulation time. You must add the test patterns and finish this lab in only one project.

4 lab1_2_t.v (30%)

Complete the testbench **lab1_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: source_0 = XX, source_1 = XX source_2 = XX, source_3 = XX op_0 = XX, op_1 =XX, request =XX, your result = XX, correct result = XX.

where X is the corresponding data bit.

5 Questions and Discussion

Please answer the following questions in your report.

A. In the testbench **lab1_1_t.v**, please explain why we place **#DELAY** between input assignment and output verification. Hint: Gate delay.

B. If we want to let the 2'b00 operation of op_0 and op_1 have the highest priority, 2'b01 have the 2nd highest priority, and so on. When op_0 and op_1 has same operation, op_0 still has higher priority. How would you modify the code?

6 Guidelines for the report

Your report should include but not be limited to the following items.

Grading policy (subject to change): Part (A): **35%**; Part (B): **50%**; Part (C): **10%**; (D): **5%**

A. **Lab Implementation**
   You may elaborate on the following.
   1. Block diagram of the design with an explanation
   2. Partial code screenshot with an explanation: You don't need to paste the entire code into the report. Just explain the kernel part.
   3. Finite state machine (FSM) with the explanation.

B. **Questions and Discussions**
   Provide your answer to the Questions and Discussions in the lab assignment.

C. **Problem Encountered**
   Describe the problems you encountered, solutions you developed, and the discussion. Explaining them with code segments or diagrams is recommended.

D. **Suggestions**
   Any suggestions for this course are more than welcome.
   (If not, you may also post a joke (a funny one, please). It is not mandatory and has nothing to do with the grading. But it would undoubtedly amuse us. ☺)

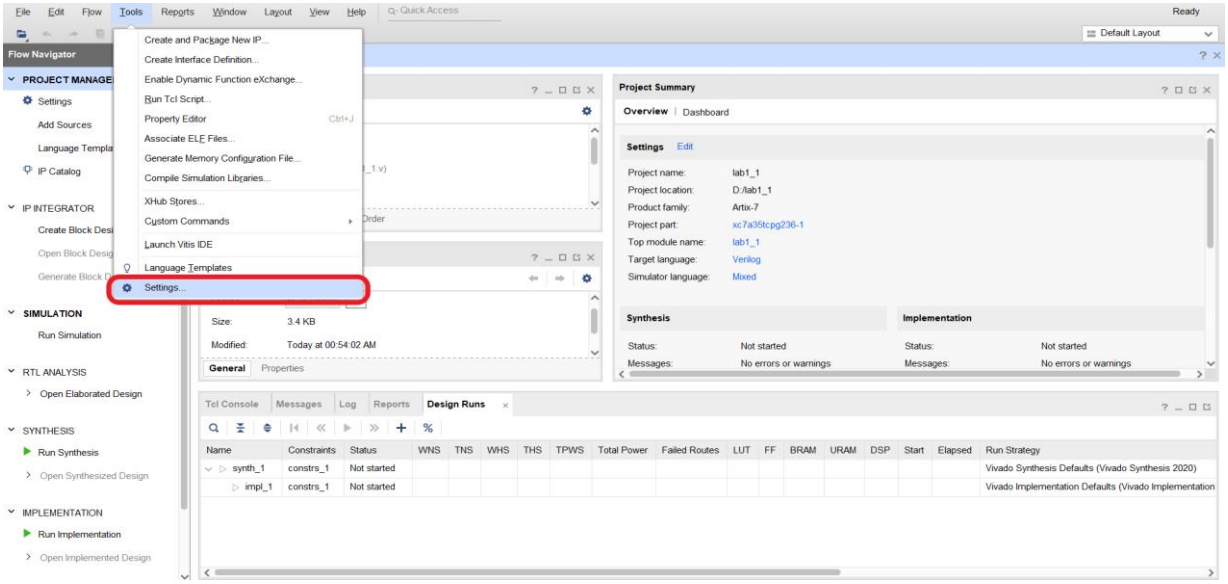
Attention

- ✓ 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 should hand in four source files, including **lab1_1.v, lab1_1_t.v, lab1_2.v, lab1_2_t.v**. Upload each source file individually. DO NOT hand in any compressed ZIP files, which will be considered an incorrect format.
- ✓ **lab1_2.v** must **not** include the module of **lab1_1**. That is, you shouldn't duplicate the module lab1_1 into **lab1_2.v**. Otherwise, you will get a penalty of 20 points.
- ✓ We will use **hidden test patterns** to test your design in this lab.
- ✓ You must also hand in your report as **lab1_report_StudentID.pdf** (i.e., lab1_report_110456789.pdf).
- ✓ You should be able to answer questions from TA during the demo.
- ✓ You may also add a **$monitor** in the testbench to show all the inputs and outputs during the simulation.

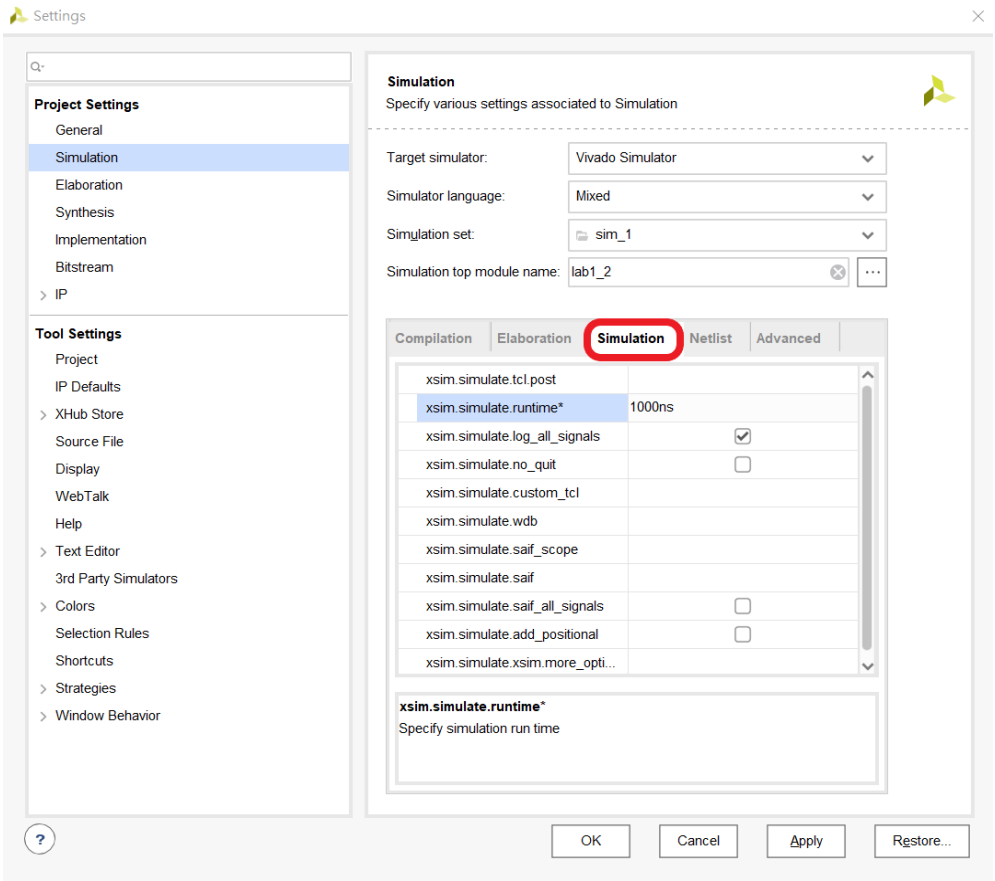
Appendix

✓ Before the simulation, you need to change the runtime to 10000ns (or a large enough period) in “Simulation Settings”. See the guide below.

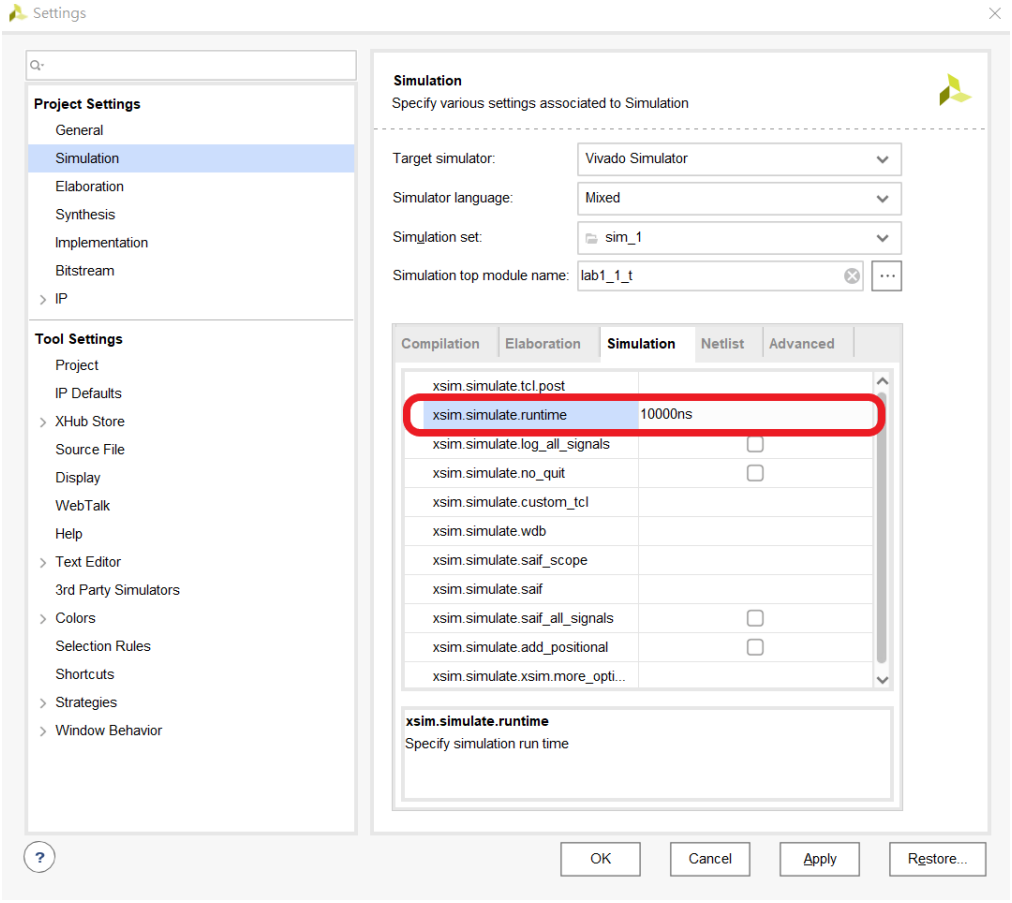
1. Click Tools at the top of Vivado and select settings.



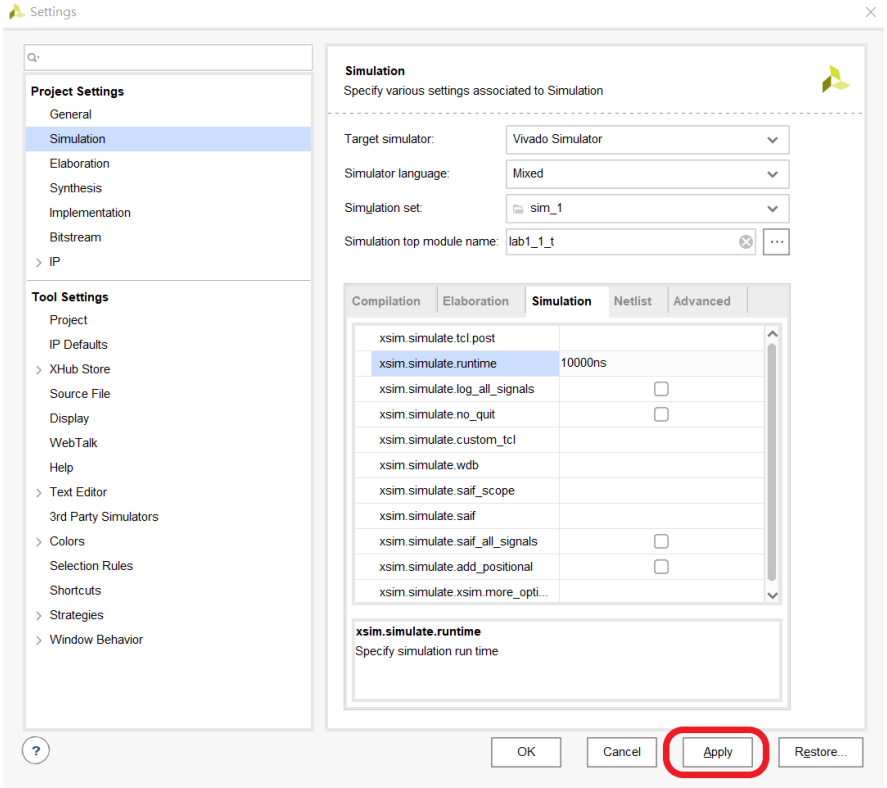
2. Select Simulation.



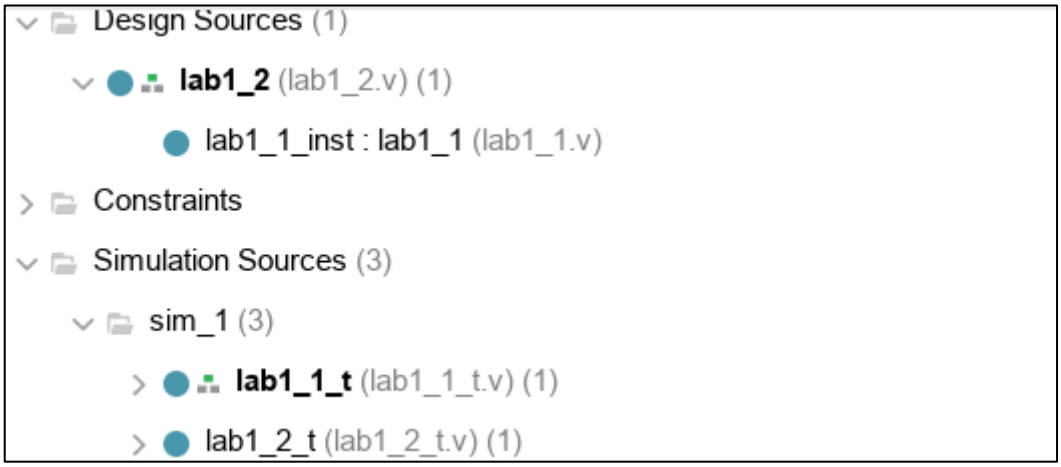
3. Change runtime to 10000ns or a larger value.



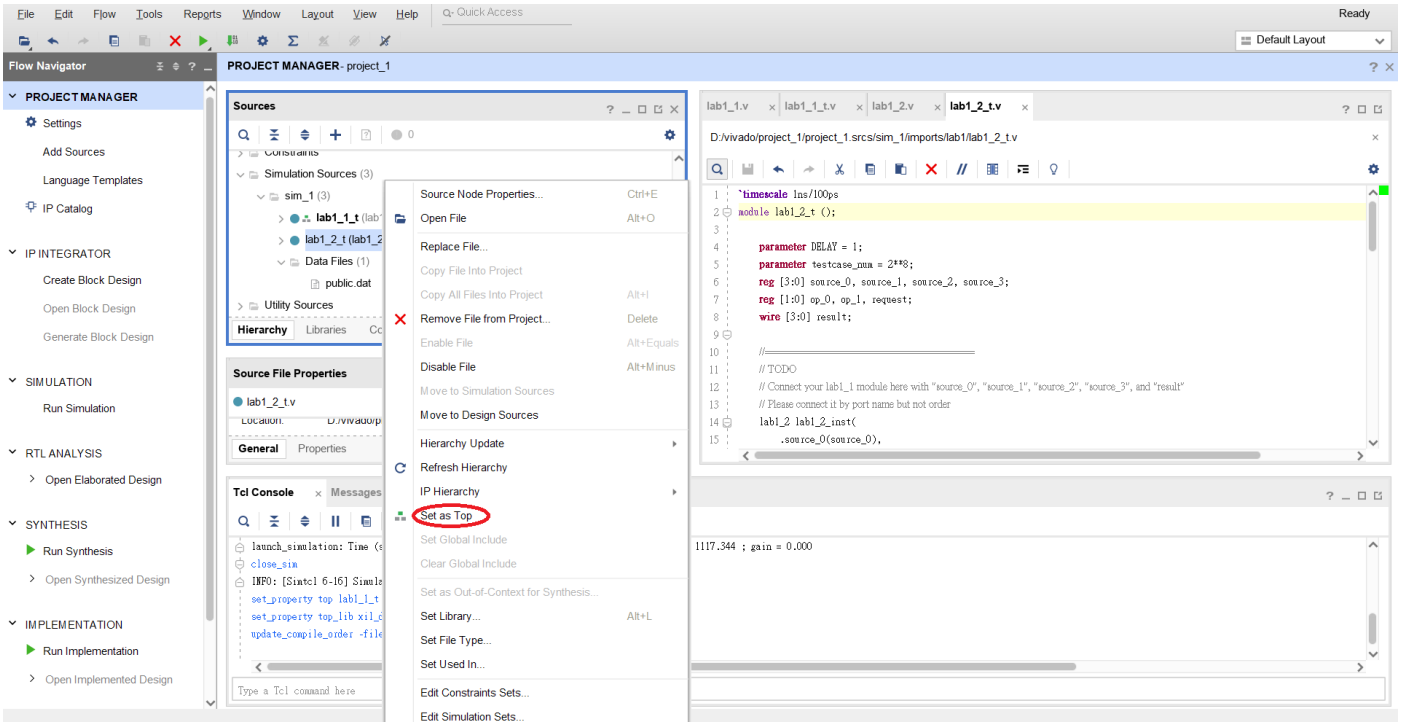
4. Remember to click Apply.



✓ 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 testbench can provide various test patterns, or even integrate different design modules to test. In this lab, 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).



✓ In lab1_2, testing exhausted (all possible) patterns will be time-consuming. So, we provide a public test set of 2^8 patterns in the data file public.dat. Note that we will use additional hidden test patterns to verify your design.