National Tsing Hua University
Department of Electrical Engineering
EE429200 IC Design Laboratory, Fall 2021

Lab 02: NC-Verlilog
Assigned on Sep 23, 2021
Due day on Sep 30, 2021

Objective

In this lab, you will learn:

1. Verilog HDL coding for basic digital circuit elements.
2. Construct a simple ALU with synthesizable Verilog HDL.
3. Simulate Verilog circuit with simulator NC-Verilog.

Demo checklist:

- ☐ Schematic view of your ALU.
- ☐ Simulation result of ALU.
- ☐ Spyglass clean.
- ☐ Answer the following question.
    1. Why do we need combinational part and sequential part for digital circuits?
    2. What's the difference between combinational and sequential circuits in implementation?

Environment Setup

1. Copy the default *.tcshrc* into your home directory (Remember the last dot).
   ```
   $ cd ~
   $ cp ~ee4292/iclab_tcshrc.txt .
   $ cat ./iclab_tcshrc.txt >> ~/.tcshrc
   ```

Note: (**Important**) The new iclab_tcshrc.txt was upload to activate CAD tool.

2. Copy lab file packages from ee4292. Decompress the package and enter it. You can check the file list in Appendix.
   ```
   $ cp ~ee4292/iclab2021/lab02.zip .
   $ unzip lab02.zip
   $ cd lab02
   ```

Description

Open the files in reference with *Verdi*. You will find some Verilog example for the basic

digital elements. Besides, you can also find suggestions for using these elements in comment. Follow the coding styles in the reference and what you learn in class, and then construct an ALU.

Figure 1 is a simple ALU with two 8-bit input signals and six instructions. You can input signals and select the functions with instruction. The functions and their corresponding instructions is defined in Table 1.

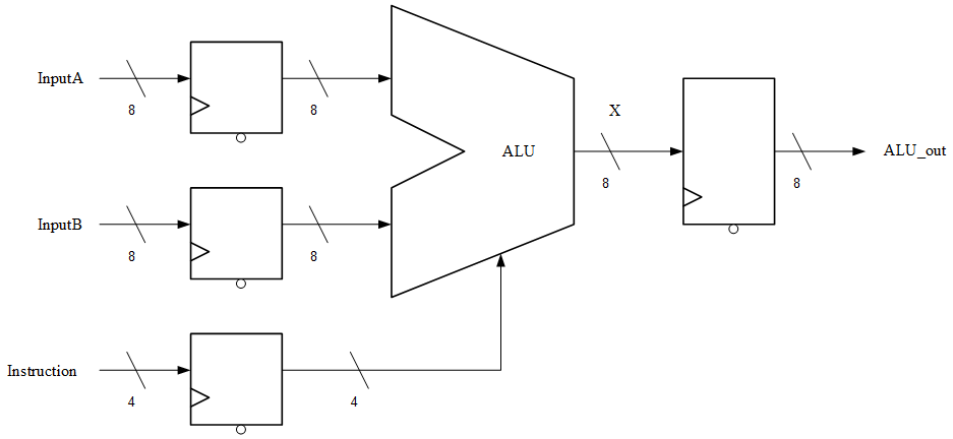

Figure 1. ALU architecture.

Instruction[3:0]	Function
4'b0000	Addition
4'b0001	Subtraction(*A-B*)
4'b0010	*B* Inversion
4'b0011	AND
4'b0100	OR
default	Exclusive OR

Table 1. ALU function table.

Action Items

I. View the schematic of basic elements.

1. Open Verdi to view the schematic diagram.
   $ Verdi &
2. Open *basic_elements.v* in reference with Verdi.

II. Construct the ALU.

1. Open ***lab2_alu.v*** with the text editor (i.e. *gedit*, *vim* etc.).
2. Construct the ALU described in Figure 1. Start from the module template ***lab2_alu.v***, which defines the I/O of the module.
3. Run NC-Verilog to do lint for Verilog and fix all of the language errors and warnings with the messages shown on the terminal or in ***ncverilog.log***.
   $ ncverilog lab2_alu.v

III. Run simulation with testbench.

Run simulation with the following command.

```
$ ncverilog –f lab2_run.f
```

or

```
$ ncverilog lab2_alu_test.v lab2_alu.v +access+r
```

Note: "+access+r" allows other tools to access the result of ncverilog.

IV. Run synthesizable check with *spyglass*.

1. Run *spyglass* in terminal. Click *Add Files* import files, select *lint* and *adv_lint* in *Goal Setup*. Press *Run Goals* button and you may check the code is synthesizable or not.

   ```
   $ spyglass &
   ```

2. Fix all the errors and warnings.

Appendix

Filename	**Description**
lab2_alu.v	RTL code for ALU
lab2_alu_test.v	Test bench for ALU
lab2_run.f	Filelist
golden.dat	Correct respond
reference/	Verilog example code