Lab 1: Using Testbench and Waveform Debugging

COMPENG 4DV4: Very Large Scale Integration System Design

January 20, 2025

Objective

This lab introduces a synthesizable Verilog implementation of a 4-bit Arithmetic Logic Unit (ALU) and its verification using NC-Verilog. Students will debug syntax and functional errors and validate correctness using a file-based testbench and waveforms.

1 Tools Used

- Cadence Xcelium
- Synopsys Verdi

2 Environment Setup and Data Preparation

2.1 Extracting Lab Files

Upload `Lab1.tar` to your working directory and extract it:

```
tar -xvf Lab1.tar.xz
```

2.2 Loading Environment Scripts

Load the Cadence Xcelium and Synopsys Verdi tools environment in your working directory:

```
source /CMC/scripts/cadence.xceliummain24.03.013.csh
source /CMC/scripts/synopsys.verdi.2021.09-SP1-1.csh
```

2.3 Directory Contents

- `Lab1_alu.v` – ALU RTL design
- `Lab1_alu_tb.sv` – Basic testbench
- `Lab1_alu_run.f` – NC-Verilog file list
- `alu_test_vectors.hex` – Golden output

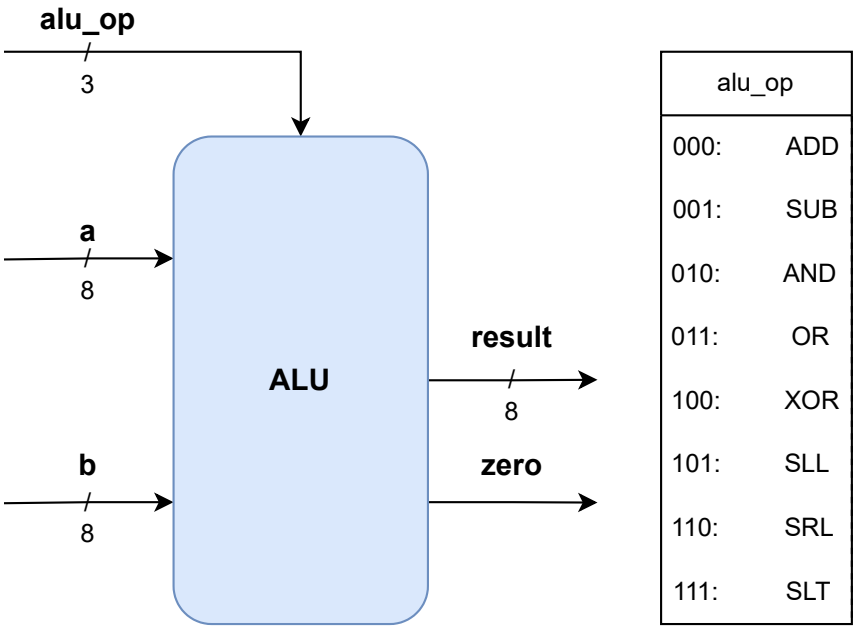
3 Part A - Run Testbench

3.1 Inspecting the RTL

Enter the lab directory and open the ALU source file:

```
cd Lab1
gedit Lab1_alu.v &
```

The ALU supports eight instructions on two 8-bit operands. open the design file to check the instructions and there function .



3.2 Fixing Syntax Errors

Compile the RTL:

```
ncverilog Lab1_alu.v
```

NC-Verilog reports an error indicating that `result = 0` is not a legal lvalue in this context.

3.2.1 Declare the Missing Signal

Edit `Lab1_alu.v` and change the declaration for the output `result`:

```
output reg [WIDTH-1:0] result,
```

Recompile to confirm the fix:

```
ncverilog Lab1_alu.v
```

3.3 Running File-Based Simulation

Open the file-based testbench to understand the I/O mechanism:

```
gedit Lab1_alu_tb.sv &
```

Run the simulation:

```
ncverilog Lab1_alu_tb.sv Lab1_alu.v
```

The simulator reports multiple mismatches between expected and actual outputs.

3.4 Fixing Functional Errors

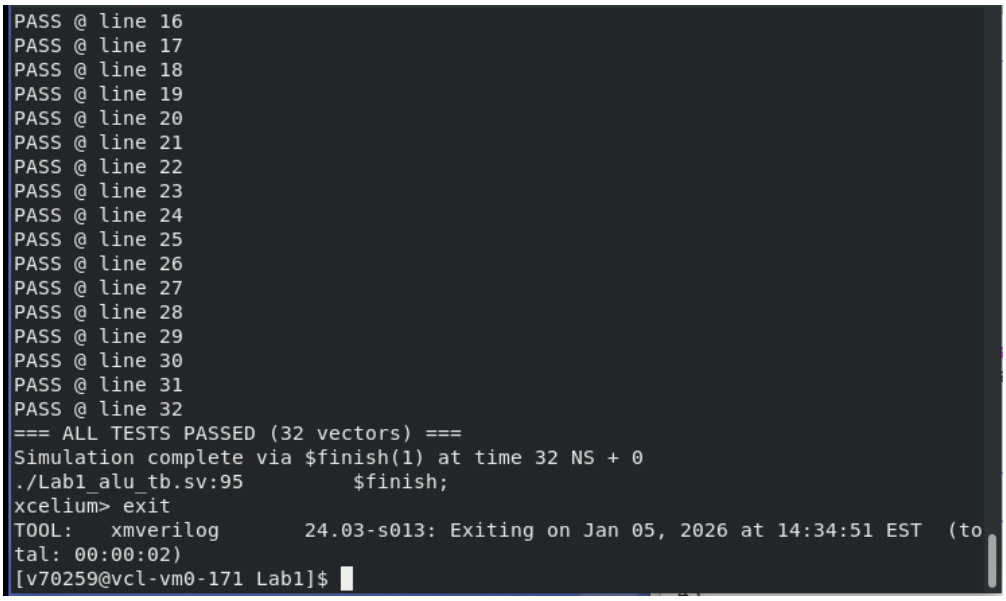
Open your design `Lab1_alu.v` and try to find the functional error in the code.

Modify the computation logic in `Lab1_alu.v` to correctly implement addition and subtraction.

3.4.1 Re-run Simulation

```
ncverilog Lab1_alu_tb.sv Lab1_alu.v
```

The simulation should now complete with no errors.



3.5 Using Command File

Alternatively, run the simulation using the provided file list:

```
ncverilog -f Lab1_alu_run.f
```

4 Part B — Generate Value Change Dump (VCD) File

Open `Lab1_alu_tb.sv` and add a VCD dump in the testbench in a separate initial block. For example:

```
initial begin
    $dumpfile("alu_tb.vcd");
    $dumpvars(0, tb_alu_verify);
end
```

Run RTL simulation (example with file list):

```
# RTL simulation (using the provided filelist)
ncverilog +access+rwc -f Lab1_alu_run.f
```

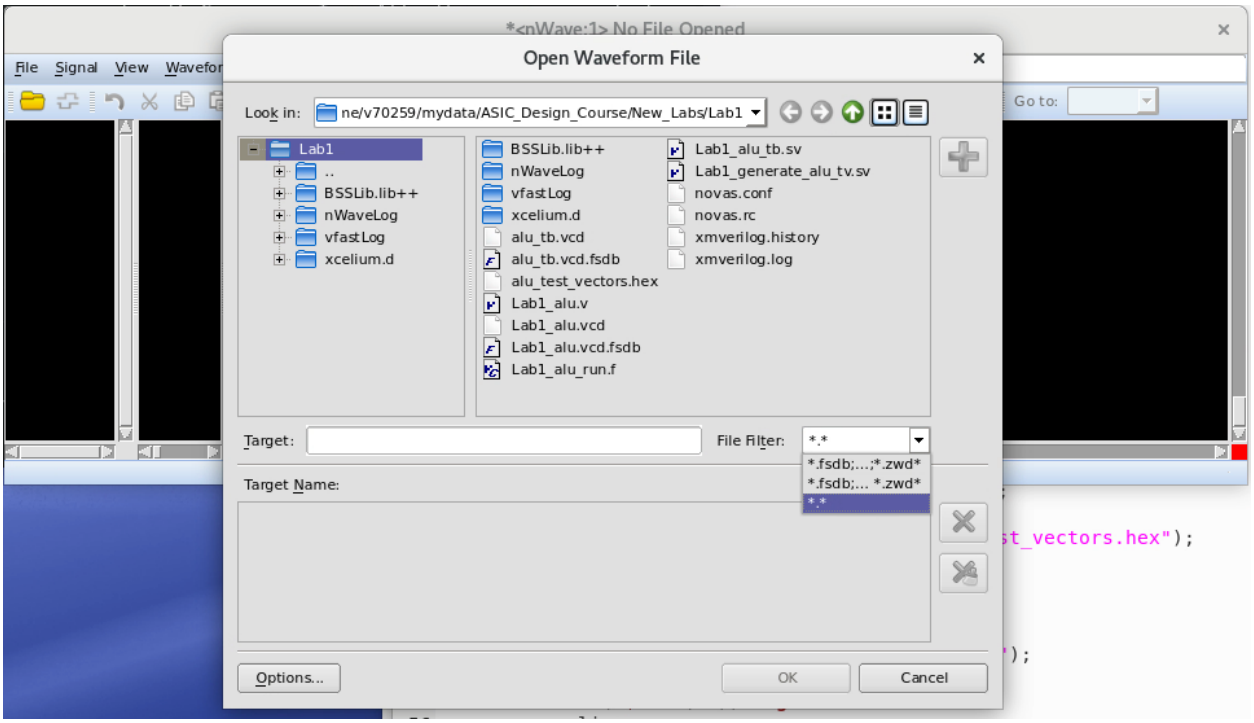
Confirm that `Lab1_alu.vcd` exists.

4.1 Use nWave / Verdi to view waveforms

Open nWave:

```
nWave &
```

Then File – Open – File Filter = "." – select the vcd file.

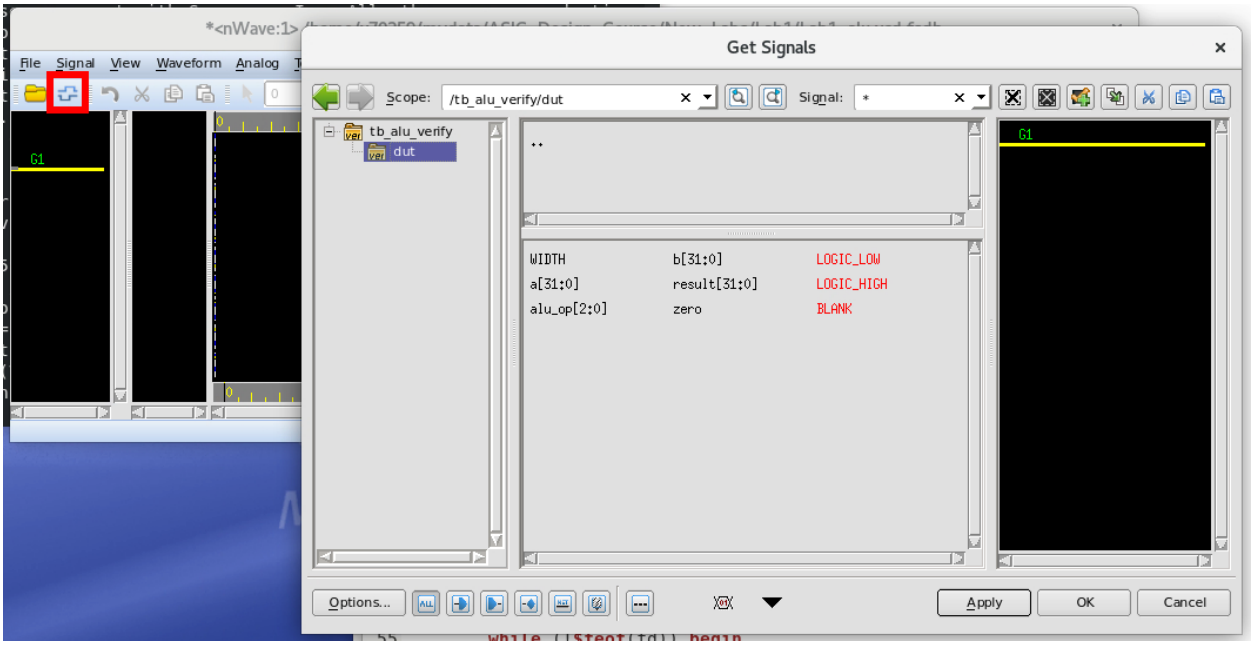


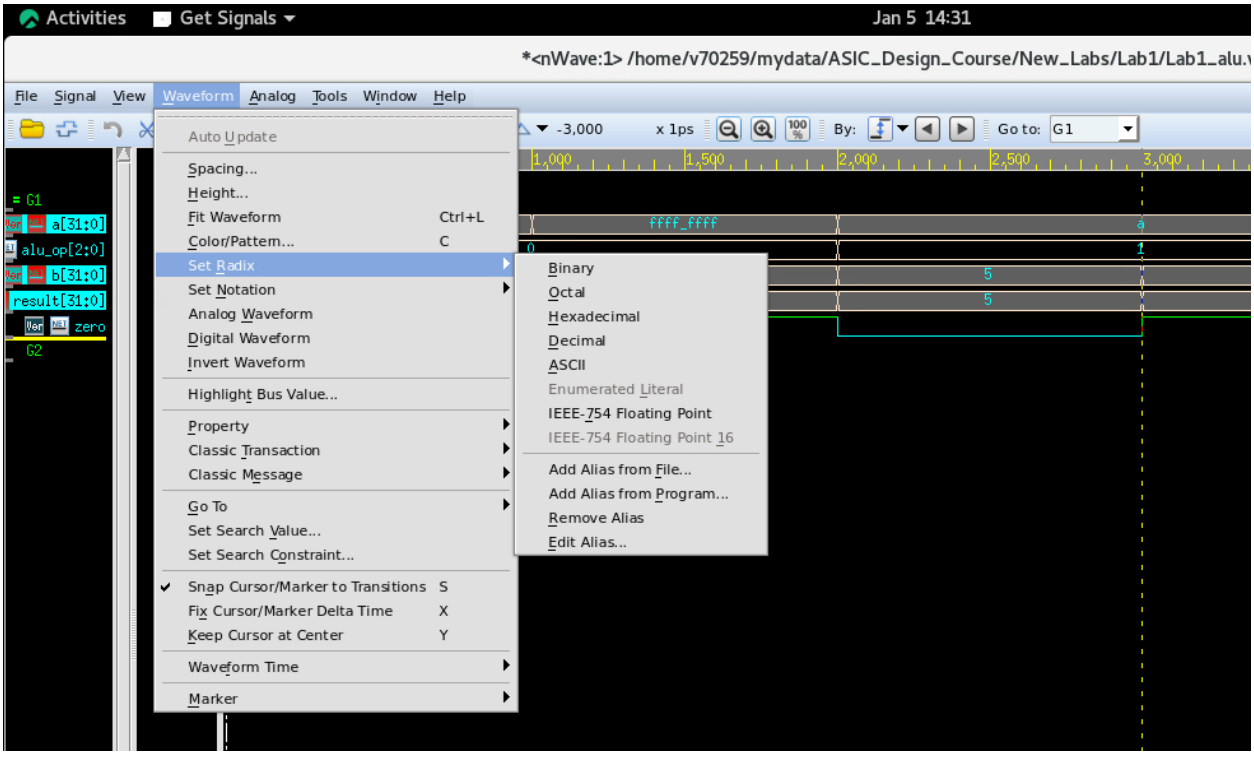
Choose the signals `result`, `zero`, `a`, `b`, `alu_op` to simulate. change the radix of the inputs and outputs to decimal. Check the results of the alu, are the correct?

5 Checkpoints

Before leaving the lab, verify:

1. The syntax error has been fixed in Part A 20%
2. All simulation errors are resolved in Part A 20%





3. The waveform is generated for the wanted signals in Part B 20%
4. Console output and waveform are shown to the TA 20%
5. Report about your results 20%

6 Submission

- A report with all the screenshots stated in the checkpoints and lab steps with annotation
- Name it `lab1_studentID`
- Upload it to its corresponding Dropbox on avenue