ECE-111: Advanced Digital Design Project: Homework 2

Designs	4-bit ALU (Arithmetic Logic Unit), 4-bit up down binary counter
Deadline	October 16, 2024 at 11:59pm
Max Late Days	2 (20% grade reduction per day)

Overview

In this homework, you will learn how to create parameterized modules, how to instantiate a module in another module and how to connect the primary ports of two modules using an explicit name based binding approach. You will learn the functional behavior of SystemVerilog arithmetic and logical operators and understand the hardware generated for each of the operators post synthesis. You will observe the changes in the values of signals in the sensitivity list of always block and how it impacts the behavior of the circuit.

There will be two parts for this homework. In **homework-2a** you will design a synthesizable SystemVerilog Model of 4-bit ALU (Arithmetic Logic Unit). In **homework-2b**, you will develop a synthesizable SystemVerilog code for a 4-bit up-down binary counter.

We have provided a folder called **Lab2.zip** which contains the followings:

Homework-2a:

1. alu_top.sv template code
2. alu.sv template code
3. alu_top_testbench.sv with full testbench implementation

Homework-2b:

1. up_down_counter.sv template code
2. down_counter.sv with partial code
3. up_counter.sv with full implementation
4. mux2to1.sv with partial implementation
5. up_down_counter_testbench.sv with full testbench implementation

Assignment Tasks

For Homework-2a:

- Design synthesizable SystemVerilog Model of a 4-bit ALU (Arithmetic Logic Unit) which can perform following mentioned functions:

Operation Select	Function	Function Definition
4'b0000	operand1 + operand2	Addition
4'b0001	operand1 - operand2	Subtraction
4'b0010	operand1 * operand2	Multiplication
4'b0011	operand1 % operand2	Modulo
4'b0100	operand1 / operand2	Division
4'b0101	operand1 & operand2	Bitwise AND
4'b0110	operand1 | operand2	Bitwise OR
4'b0111	operand1 ^ operand2	Bitwise XOR
4'b1000	operand1 && operand2	Logical AND
4'b1001	operand1 || operand2	Logical OR
4'b1010	operand1 << 1	Left Shift by 1
4'b1011	operand1 >> 1	Right Shift by 1
4'b1100	operand1 == operand2	Logical Equality
4'b1101	operand1 != operand2	Logical Inequality
4'b1110	operand1 < operand2	Less Than Comparison
4'b1111	operand1 > operand2	Greater Than Comparison

- Create module with name "alu"
  - Declare 4-bit logic type input primary ports: operand1, operand2, operation.
  - Declare 8-bit logic type output primary alu_out.
  - Declare parameter "N" with default value set as 4. N is used when declaring the width of operand1 and operand2 primary ports.
  - Use always block with operation, operand1, operand2 in its sensitivity list.
  - Within the always procedural block, use the "case" statement to select between alu operations.
  - Ensure default case expression is provided with an addition operation.

- Create another module alu_top
  - Declare 4-bit logic type input primary ports: operand1, operand2, select.
  - Declare 8-bit logic type output primary result.
  - Instantiate alu module inside the module alu_top.
  - In the instance of alu, connect the primary ports of alu_top with alu's primary ports using explicit name based binding approach.
  - Add a positive edge triggered flipflop at the output of alu.

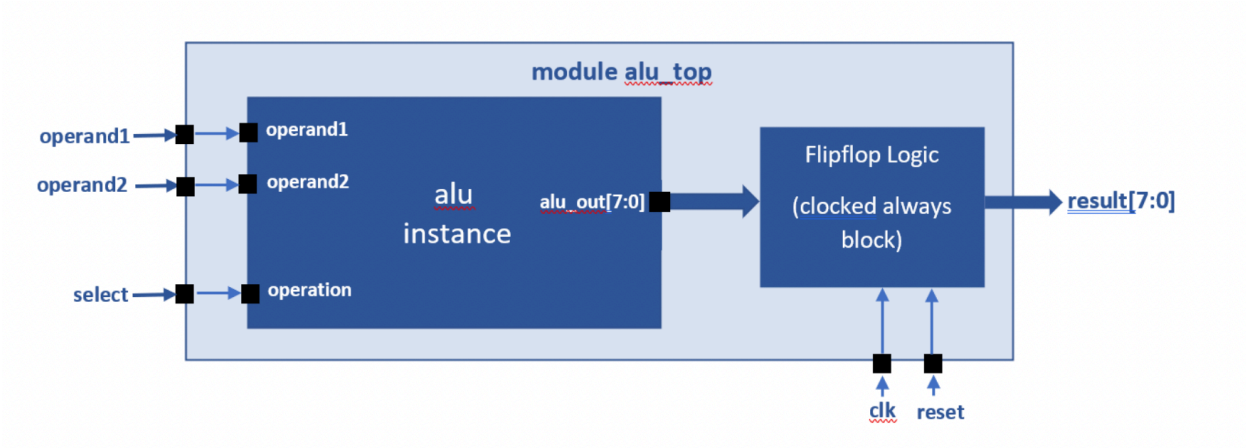
Note: flipflop logic is already implemented in alu_top template code provided in lab folder

- Perform synthesis of alu_top and alu module
  - Review resource usage report, RTL Netlist and Post Mapping Schematics.

- Perform simulation of alu_top using alu_top_testbench provided in lab folder
  - Review all 16 ALU operations' results in waveform.

- Observe the functional behavior of all ALU operations. Especially review the difference between logical and bitwise and/or operations.

- Here is the block diagram of alu_top block:



For Homework-2b:

- In up_down_counter there are two separate counters. These are:
  - 4-bit up counter which counts values from 0 to 15 in increments of 1 each clk cycle.
  - 4-bit down counter which counts values from 15 to 0 in decrements of 1 each clk cycle.

- Outputs of both up and down counters are connected to the inputs of a 2-to-1 Multiplexer.
  - When the "select" input port of the up_down_counter module is driven to '0', then the output of the multiplexer should reflect up_counter count values (i.e. 0 to 15 count values).
  - When the "select" input port of the up_down_counter module is driven to '1', then the output of the multiplexer should generate down_counter count values (i.e. 15 to 0 values).

- Review the code for up_counter module in up_counter.sv file and use it as a reference to update the down_counter module in down_counter.sv file with the required modifications.

- Review mux_2to1.sv file and update the width of its primary ports- in0, in1 and out to 4 bits.

- Instantiate up_counter, down_counter and mux_2x1 modules inside up_down_counter module and connect the ports of modules using an explicit name based binding approach.
  - Output count ports from up_counter and down_counter should be connected to mux_2x1 in0 and in1 ports using wires up_count_value and down_count_value respectively.

    - Output out port of mux should be connected to the primary port result of up_down_counter.
    - Primary input port select of up_down_counter should be connected to sel port of mux_2x1.
    - Primary input port clk and clear should be connected to clk and clear ports of up_counter and down_counter.

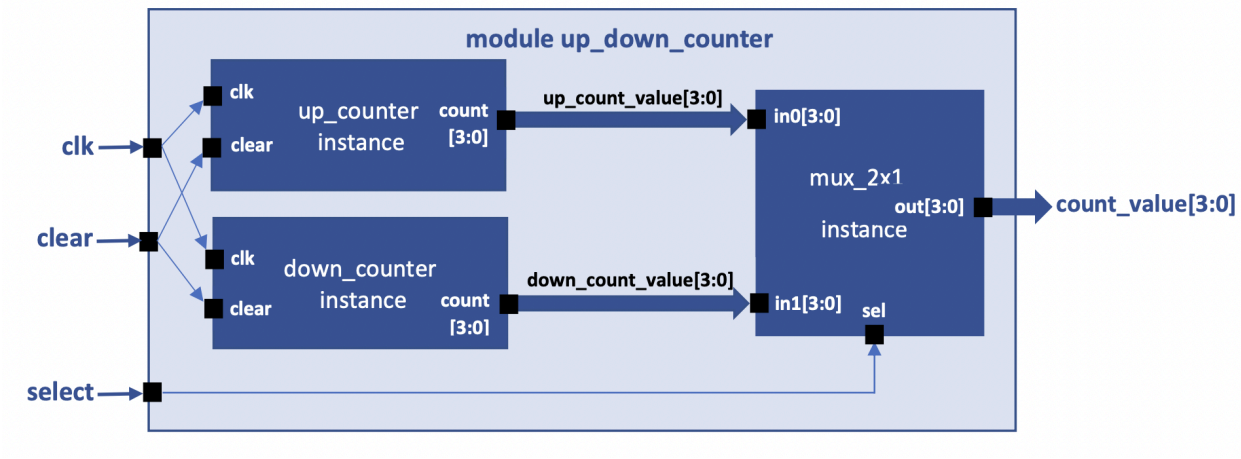
- Synthesize up_down_counter top level module along with sub-modules up-counter, down_counter and mux_2to1.

- Review synthesis results (resource usage and RTL netlist/schematic).

- Run simulation using up_down_counter testbench code provided in the Lab2 folder.

- Review up_down_counter input and output signals in simulation waveform.

- Block Diagram of up_down_counter:



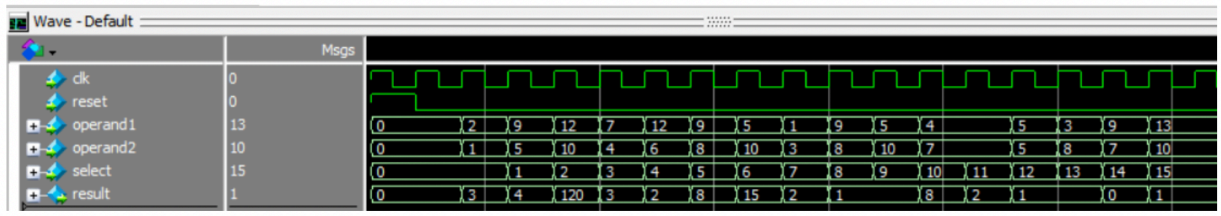
Submission Requirements

Submit a report on Gradescope in PDF format which includes the following:

For Homework-2a:

- Include the SystemVerilog code snapshots of module alu_top and alu.
- Provide a snapshot of FPGA resource usage generated post synthesis.
- Provide a snapshot of the schematic generated from RTL netlist viewer.
- Provide a snapshot of the simulation waveform and explain the simulation results.

Here is a simulation snapshot for reference:



For Homework-2b:

- Include the snapshots of up_down_counter, up_counter, down_counter, mux_2x1 SystemVerilog codes.
- Provide a snapshot of FPGA resource usage generated post synthesis.
- Provide a snapshot of the schematic generated from RTL netlist viewer.
- Provide a snapshot of the simulation waveform and explain the simulation results.

Here is a simulation snapshot for reference: