UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

ECE-111 Advanced Digital Design Projects

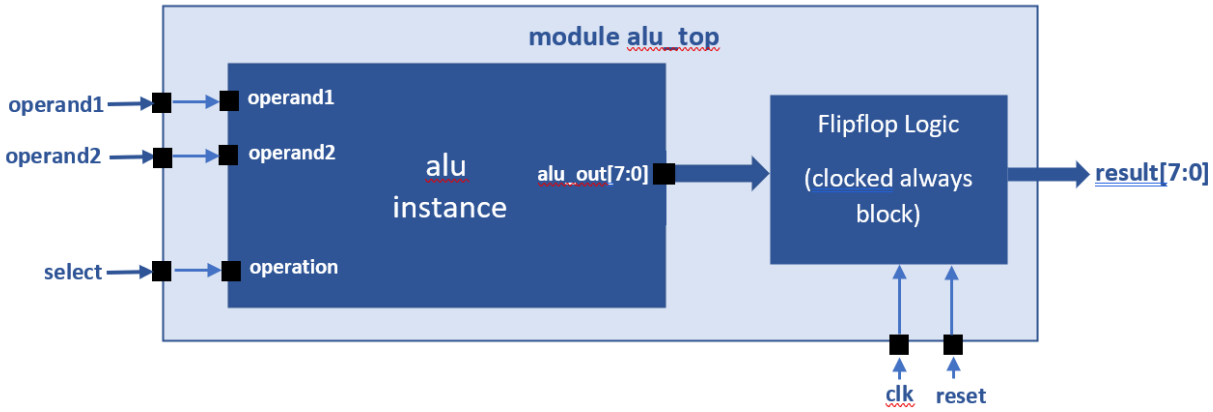
- **Objective of Homework-2:** Learn how to create parameterized modules, how to instantiate a module in another module, how to connect primary ports of two modules using explicit name based binding approach. Learn functional behavior of SystemVerilog arithmetic and logical operators and understand hardware generated for each of the operator post synthesis. Observe change in value of signals in sensitivity list of always block how it impacts the behavior of the circuit.

Homework-2a

- Design synthesizable SystemVerilog Model of 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	Division
4'b0100	operand1 / operand2	Modulo
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

Block Diagram of alu_top block



- 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 width of operand1 and operand2 primary ports
  - Use always block with operation, operand1, operand2 in its sensitivity list
  - Within always procedural block, use "case" statement to select between alu operations
  - Ensure default case expression is provided with 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 module alu_top
  - In instance of alu, connect primary ports of alu_top with alu primary ports using explicit name based binding approach
  - Add positive edge triggered flipfop 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 alut_top using alu_top_testbench provided in lab folder
  - Review all 16 ALU operation results in waveform
  - Observe functional behavior of all ALU operations. Especially review the difference between logical and bitwise and / or operations

Homework Submission:

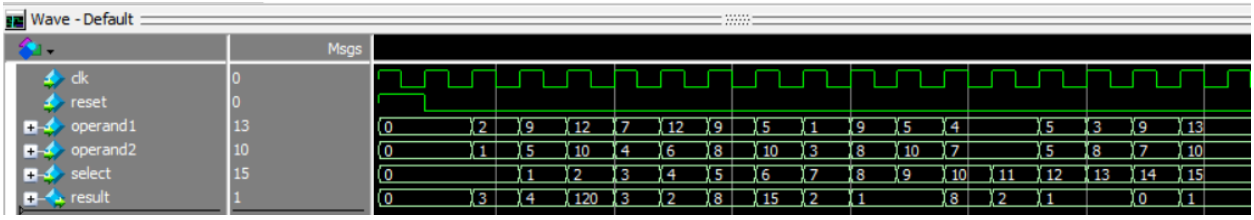
Submit report (PDF file) which should include:

- SystemVerilog code snapshot of module alu_top and alu
- Provide snapshot of FPGA resource usage generated post synthesis
- Provide snapshot of schematic generated from RTL netlist viewer
- Provide snapshot of simulation waveform and explain simulation result

Note: Lab2.zip folder has following mentioned files:

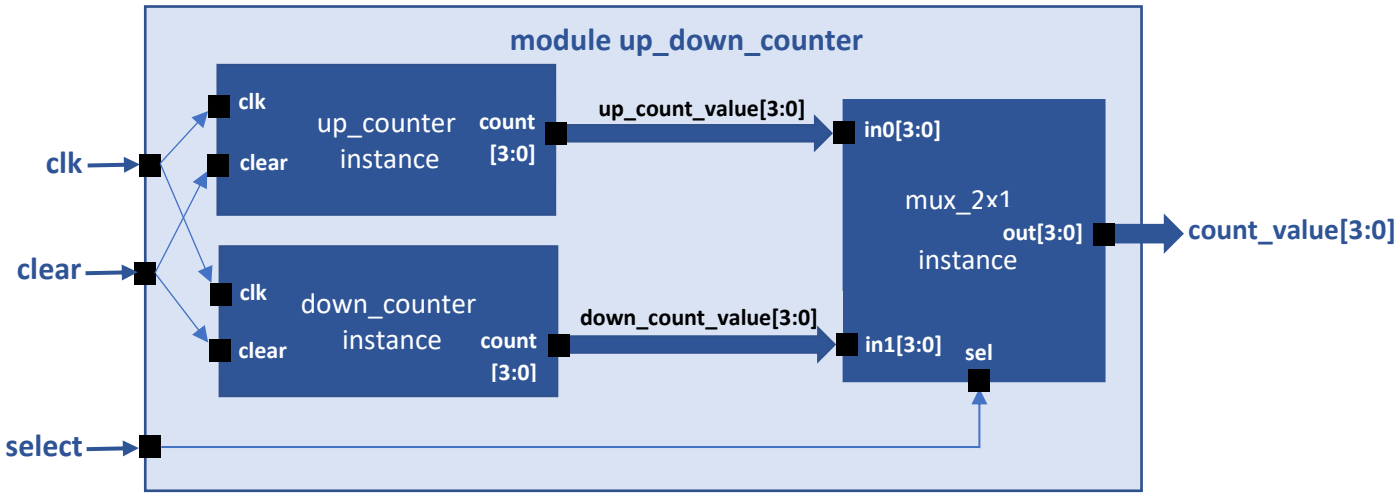
- alu_top.sv template code
- alu.sv template code
- alu_top_testbench.sv with full testbench implementation

Reference Simulation Snapshot



Homework-2b

Develop synthesizable SystemVerilog code for 4-bit up down binary counter. Refer to below mentioned block diagram of up_down_counter when developing SystemVerilog code.



- In up_down_counter there are two separate counters. These are :
  - 4-bit up counter which counts values from 0 to 15 in increment by 1 each clk cycle
  - 4-bit down counter which counts values from 15 to 0 in decrement by 1 each clk cycle
- Output of both up and down counters are connected to inputs of a 2-to-1 Multiplexer
  - When "select" input port of up_down_counter module is driven to '0' then output of multiplexer should reflect up_counter count values (i.e. 0 to 15 count values)
  - When "select" input port of up_down_counter module is driven to '1' then output of multiplexer should generate down_counter count values (i.e. 15 to 0 values)
- Review code for up_counter module in up_counter.sv file and using it as a reference update down_counter module in down_counter.sv file with required modification
- Review mux_2to1.sv file and make update width of it 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 explicit name based binding approach
  - Output count port from up_counter and down_counter should be connected to mux_2x1 in0 and ini1 ports using wires up_count_value and down_count_value
  - Output out port of mux should be connected to 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, mux_2to1
- Review synthesis results (resource usage and RTL netlist/schematic)
- Run simulation using up_down_counter testbench code provided in Lab2 folder
- Review up down counter input output signals in simulation waveform

Homework Submission

Submit report (PDF file) which should include:

- up_down_counter, up_counter, down_counter, mux_2x1 SystemVerilog code
- Provide snapshot of FPGA resource usage generated post synthesis
- Provide snapshot of schematic generated from RTL netlist viewer
- Provide snapshot of simulation waveform and explain simulation result

Note: Lab2.zip folder has following mentioned files:

- up_down_counter.sv template code
- down_coutner.sv with partial code
- up_counter.sv with full implementation
- mux2to1.sv with full implementation
- up_down_counter_testbench.sv with full testbench implementation

Reference Simulation Snapshot