CSCI 113 - Lab 2

Basic gate/component implementation (behavioral) in VHDL and testing

1. Download "AND_gate.vhd" and AND_gate_TB.vhd" from the Canvas of this course, and compile/run.
   When you fully understand the operations manipulated in the testbench program (AND_gate_TB.vhd) and the wave form, proceed to the next step.

2. Update AND_gate.vhd with the following logic:

```
if ((x='0') and (y='0')) then
   z <= '1';
elsif ((x='1') and (y='1')) then
   z <= '1';
else
   z <= '0';
end if;
```

   Also, update the testbench program (AND_gate_TB.vhd) with the following for --case1:

```
--case1
x <= '0';
y <= '0';
wait for 10 ns;
assert (z = '0') --desired output value
report "case1 failed!" severity error;  --if not desired output, display error msg
```

   Then, compile/run and see what are different from the previous run.

3. Restore AND_gate.vhd and AND_gate_TB.vhd to their original codes;

4. Write VHDL codes for 2-input (1-bit each) OR gate, i.e., the gate (OR_gate.vhd) and testbench (OR_gate_TB.vhd), and compile/run until they are correct;

5. Write VHDL codes for 1-input (1-bit) Negator (NOT gate), i.e., the gate (NOT_gate.vhd) and testbench (NOT_gate_TB.vhd), and compile/run until correct;

6. Now, challenge to a little more complex logic for <u>2x1 Mux</u>, which you studied in class; Name the component "MUX_2x1", i.e.,

```
entity MUX_2x1 is
port(. . .  );
end MUX_2x1;

architecture behav of MUX_2x1 is
. . . .
. . . .
end behav;
```

   You should also write a testbench program, which tests all (8) cases of inputs, i.e.,

   (A, B, Sel) = (0,0,0), (0,0,1), (0,1,0), . . . , (1,1,1).

6. Complete the report in the next page and submit it.

CSCI 113 Lab2 – Report **(10 pts)** Name:

What have you learned from this lab?

Briefly describe your understanding from Step 2:

Draw wave forms from your OR gate test run and the MUX_2x1 test run:
//Show all signal names and mark (circle) the output signal name(s).

OR_gate:

MUX_2x1: