FPGA course MCI Summer Term May/June 2025.

J. Sid Clements (clementsjs@appstate.edu)

Note: You should review slides 1 to 45 in my power point reader MCI_VHDL Language_Sakai Summer 2023 Part 1 with Fundamentals Combinatorial READER .pptx **before doing this lab 1b.** You can also watch my recorded lecture on it (or skip to certain parts of it). Lecture1stHalfPart1FundamentalsCombinatorial (1 hour).mp4. The last few slides are covered in the first two minutes of the Part2 recording.

FPGA engineers usually simulate their designs before implementing them on FPGA hardware. ModelSim is the most popular simulation software. Therefore, we will use both Quartus and ModelSim in this minicourse. You will use ModelSim for this 1st lab (no boards or Quartus). Do both parts: 1a and 1b.

Lab 1a - ModelSim Testbench NAND gate.

Go to www.NANDLAND.com, *click Learn VHDL,* and work through the "Introduction to ModelSim for Beginners" tutorial using the and_gate.vhd and and_gate_tb.vhd files. To create these files, in the "Add items to the Project" pop-up window, Create New File (instead of Use Existing File), name it and_gate, double click it, and copy and paste the code from the NANDland website into the editing panel on the right. Save As them in the C:/intelFPGA/20.1/**work** directory. If the "Add Items …" window does not pop up, right click in the Project Window (the tab for it is at the bottom of the left window), and select "Add To Project". If you add an Existing File to your project, make sure to select the radio button "copy to project directory".

Make sure the files in the editor window correspond to the ones in the project window by clicking the file in the project window to see if it opens in the editor window.

Add these commands to the end of the testbench file just before the End Process line:

assert false
report "Simulation Completed"
severity FAILURE;

This will stop the simulation at the end and display the message: Failure: Simulation Completed, otherwise it can run too long and make it difficult to find the part of the output waveform that matters.

Make sure both the and_gate vhdl and and_gate_tb files are in the project window (and in the C:/intelFPGA/20.1/work directory). Select Compile All. Select menu COMPILE/COMPILE REPORT to see errors. Fix any errors and recompile. After both files compile successfully, start the simulation. Expand the **+work** line and select the and_gate_**tb** file. Click OK. In the sim window, right-click and_gat_tb and select ADD Waveform (this adds a waveform based on your tb file code). To run the simulation, use the menu SIMULATE/RUN/RUN-ALL, or click the 3rd icon (run all) to the right of the time step value window. Select the wave tab on the bottom left of the main window to show the wave window, then right-click the window and select ZOOM FULL to show the entire simulation waveform (all 4 time steps), or you can use the Zoom Full icon 🔍. You will probably need to drag the window borders to the left to make the waveform window bigger.

Save an image of the waveform using these 3 steps in order:

1) Drag the vertical border of the signal pane so that the entire signals NAMES are visible.

2) Right click on the waveform and select Zoom Full (makes the entire WAVEFORM visible).
3) Then Use windows to snip a screenshot of the waveform.
I do not need the waveform.do file, i.e., do NOT do File, Save Format.

- For credit for this assignment, upload to your SAKA assignments:

1. The waveform IMAGE for the AND gate created in VHDL.
2. The and_gate.vhd file with the recommended coding style applied
3. The and_gate_tb.vhd file (prefixes are not necessary)

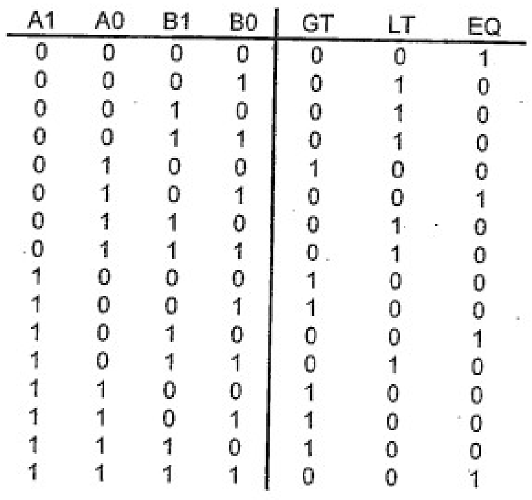
Lab 1b and1c - ModelSim Testbench 2-bit comparator made of sub-components wired with port maps.

Lab 1b is to only design the A Greater Than B circuit GT using the IF statement method (See below). Do not make a test bench for it. (You will make a top-level test bench in lab 1c.) The rest of the lab is 1c.

For lab 1c, You will be using a slightly different "template" for the testbench file. The new template is from an FPGA book by Chu. It is on slides 42 – 45 in my MCI VHDL part 1 with fundamentals combinatorial powerpoint READER.

You must use NANDLAND's VHDL naming styles: https://www.nandland.com/articles/coding-style-recommendations-vhdl-verilog.html
The lab assignment is to design a comparator circuit to compare the magnitudes of two 2-bit numbers (A1 A0 and B1 B0). The circuit will have three output signals: o_aeq2b, o_agtb, and o_altb. The signal o_aeq2b will be high if the two 2-bit values are equal, o_agtb will be high to indicate that the 2-bit A value is greater than the 2-bit B value, and o_altb will be high if the 2-bit A value is less than the 2-bit B value. Use 2-bit standard logic vectors for inputs A and B. The desired 3-output function is defined in a truth table shown below. Design three separate components (EQ, GT, LT). **You must use a CASE statement for the EQ component, an IF-ELSE statement for the GT, and a WITH SELECT for the LT.** All three are covered in my VHDL powerpoint reader (part 1 Combinatorial). **For EQ and LT, use a 4-bit std_logic_vector for an internal signal w_BA consisting of i_B & i_A (concatenated).** Write the VHDL code for these with the information and coding style in my reader. You can get help on VHDL commands from https://www.ics.uci.edu/~jmoorkan/vhdlref/.

Make sure you don't just copy code from the internet, I probably will not accept it (you don't learn much that way).



A1	A0	B1	B0	GT	LT	EQ
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	1

Use my powerpoint reader (MCI VHDL Language Part 1 with Fundamentals Combinatorial) slides 31 - 41 to **put all three sub-components into a top-level component named twobitcomp using PORT MAPS.** The top level component has two 2-bit vector inputs (i_a and i_b) and three standard logic outputs (o_aeq2b, o_agtb, o_altb). **When making the top-level component, for the port mapping, use the old VHDL-87 method (COMPONENT) to declare the EQ component, and the new VHDL-93 method (ENTITY) to declare the GT and LT components.**

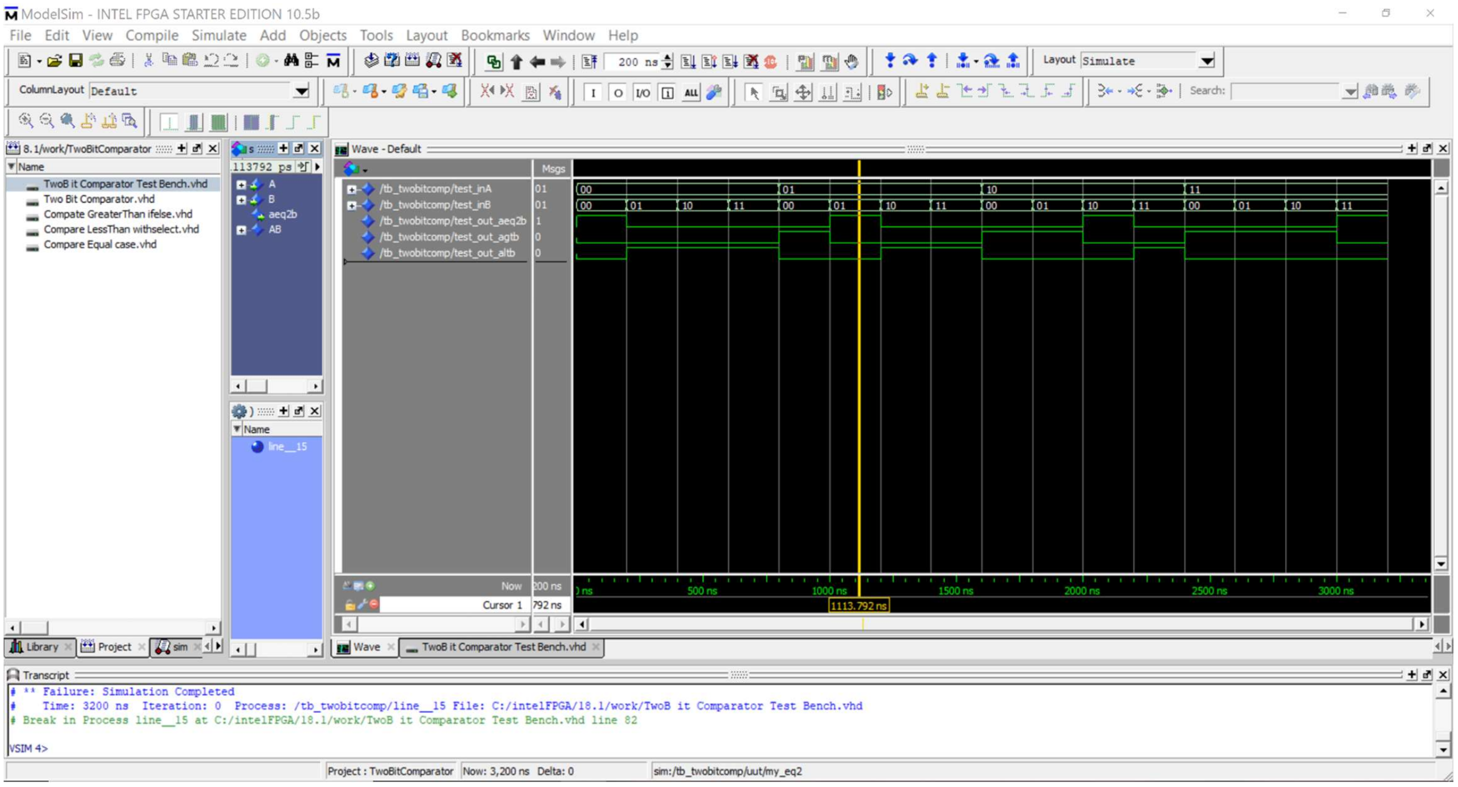
Use ModelSim (you don't need Quartus for this lab) to create **one** project named two-bit_comp with five VHDL files: 3 for the sub-components (EQ, GT, LT), 1 for the top-level component (twobitcomp), and 1 for a testbench (twobitcomp_tb). Remember that the Entity name must match the filename. Create a new project called twobitcomp (if necessary, change the work directory to C:/intelFPGA/20.1/work). You will add new files (right-click in the project menu) for each of the 5 vhdl designs as you progress through this lab. You can click the project tab at the bottom of the project window to show the new files; see the snipped image at the end of this explanation (use the word zoom control to expand it). Double click the filename to open the editor. I suggest doing the LT component first with IF-ELSE (it is the easiest). After typing the VHDL code into the ModelSim editor, select the file in the project window, right-click it, and then COMPILE SELECTED. *View your compile errors by clicking COMPILE from the menu bar, then COMPILE REPORT.* (Menus are context-sensitive, so if PROJECT is not on the menu bar, select a file in the project window to make it appear.)

Use my powerpoint reader (MCI VHDL Language Part 1 with Fundamentals Combinatorial) slides 42-45 on the testbench in the Chu book as a template and create the tb_twobitcomparator testbench. Assign all 16 combinations of A and B to the input signals (name them test_inA and test_inB) as you connect them to the twobitcomp unit under test (UUT). Never *assign* values to testbench *output* signals. Instead, name them test_out_aeq2b, etc., connect them to the UUT outputs, and then l*ook* at them in the wave window after you run the simulation.

Make sure all 5 vhdl files are in the project window (and in the work directory). Start the simulation. Expand the +work line and select the twobitcomp_tb file. Right-click it and select ADD Waveform. Make sure your testbench has the *assert false, report and severity failure commands* at the end. These will stop the simulation at the end; so to run it, change the timestep in the menu bar to 200 ns, then click the 3rd icon (run all) to the right of it to run the simulation. Select the wave tab on the bottom left of the main window to show the wave window, then right-click the window and select ZOOM FULL to show the entire simulation (all 16 time steps). If ZOOM FULL does not show them correctly, then select ZOOM RANGE (0 to 4 us). You can left-click in the waveform window to show the yellow vertical cursor line. Drag it across the waveforms to show the binary input and output values at that time in the grey msgs section on the left of the waveform window. Right-click the yellow box at the bottom of the cursor line. Select “Grid, timeline and cursor control”. Change the time units to ns. Your waveform should look like the snipped image below. (The cursor is on the A = 01 and B = 01 step. Notice that this causes the test_out_aeq2b signal to be a 1 in the grey msgs column.)

- For credit for this assignment, upload to SAKAI:

1. The ModelSim waveform (zoomed full) for the comparator showing the16 time steps and the full input names.
2. The 5 vhd files with the NANDland recommended naming style applied (as well as the coding style in my powerpoints).



```
/tb_twobitcomp/test_inA            01
/tb_twobitcomp/test_inB            01
/tb_twobitcomp/test_out_aeq2b      1
/tb_twobitcomp/test_out_agtb       0
/tb_twobitcomp/test_out_altb       0
```

```
# ** Failure: Simulation Completed
#    Time: 3200 ns  Iteration: 0  Process: /tb_twobitcomp/line__15 File: C:/intelFPGA/18.1/work/TwoB it Comparator Test Bench.vhd
# Break in Process line__15 at C:/intelFPGA/18.1/work/TwoB it Comparator Test Bench.vhd line 82

VSIM 4>
```

Project : TwoBitComparator   Now: 3,200 ns Delta: 0   sim:/tb_twobitcomp/uut/my_eq2

.

```
library ieee;
use ieee.std_logic_1164.all;

entity and_gate is
  port (
    input_1    : in  std_logic;
    input_2    : in  std_logic;
    and_result : out std_logic
    );
end and_gate;

architecture rtl of and_gate is
  signal and_gate : std_logic;
begin
  and_gate   <= input_1 and input_2;
  and_result <= and_gate;
end rtl;
```

```vhdl
-- EQ Component: Checks if A == B
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity eq_comp is
  Port (
    A : in STD_LOGIC_VECTOR(1 downto 0);
    B : in STD_LOGIC_VECTOR(1 downto 0);
    EQ : out STD_LOGIC
  );
end eq_comp;

architecture Behavioral of eq_comp is
begin
  process(A, B)
  begin
    case A is
      when "00" => if B = "00" then EQ <= '1'; else EQ <= '0'; end if;
      when "01" => if B = "01" then EQ <= '1'; else EQ <= '0'; end if;
      when "10" => if B = "10" then EQ <= '1'; else EQ <= '0'; end if;
      when "11" => if B = "11" then EQ <= '1'; else EQ <= '0'; end if;
      when others => EQ <= '0';
    end case;
  end process;
end Behavioral;
```

```
-- GT Component: Checks if A > B
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity gt_comp is
  Port (
    A : in STD_LOGIC_VECTOR(1 downto 0);
    B : in STD_LOGIC_VECTOR(1 downto 0);
    GT : out STD_LOGIC
  );
end gt_comp;

architecture Behavioral of gt_comp is
begin
  process(A, B)
  begin
    if A > B then
      GT <= '1';
    else
      GT <= '0';
    end if;
  end process;
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lt_comp is
   port (
      a : in std_logic_vector(1 downto 0);
      b : in std_logic_vector(1 downto 0);
      lt : out std_logic
   );
end lt_comp;

architecture Behavioral of lt_comp is
   begin

      with a < b select
         lt <= '1' when true,   -- Cases where A < B
         '0' when others;              -- Cases where A >= B

end Behavioral;
```

```
-- Top-Level Component (twobitcomp)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity twobitcomp is
  Port (
    i_a : in STD_LOGIC_VECTOR(1 downto 0);
    i_b : in STD_LOGIC_VECTOR(1 downto 0);
    o_aeq2b : out STD_LOGIC;
    o_agtb : out STD_LOGIC;
    o_altb : out STD_LOGIC
  );
end twobitcomp;

architecture Structural of twobitcomp is
  component eq_comp
    Port (A, B : in STD_LOGIC_VECTOR(1 downto 0); EQ : out STD_LOGIC);
  end component;
  component gt_comp
    Port (A, B : in STD_LOGIC_VECTOR(1 downto 0); GT : out STD_LOGIC);
  end component;
  component lt_comp
    Port (A, B : in STD_LOGIC_VECTOR(1 downto 0); LT : out STD_LOGIC);
  end component;
begin
  eq_inst: eq_comp port map (i_a, i_b, o_aeq2b);
  gt_inst: gt_comp port map (i_a, i_b, o_agtb);
  lt_inst: lt_comp port map (i_a, i_b, o_altb);
```

```
end Structural;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity tb_twobitcomp is
end entity tb_twobitcomp;

architecture behaviour of tb_twobitcomp is

  -- use UNIT UNDER TEST (UUT)
-- has to match my entity from twobitcomp
  component twobitcomp
    port (
      i_A       : in  std_logic_vector(1 downto 0);
      i_B       : in  std_logic_vector(1 downto 0);
      o_aeq2b   : out std_logic;
      o_agtb    : out std_logic;
      o_altb    : out std_logic
    );
  end component;

signal test_inA       : std_logic_vector(1 downto 0) := "00";
  signal test_inB       : std_logic_vector(1 downto 0) := "00";
  signal test_out_aeq2b : std_logic; -- dont assign a value to an output
  signal test_out_agtb  : std_logic;
  signal test_out_altb  : std_logic;


begin

  uut: twobitcomp
    port map (
      i_A       => test_inA,
      i_B       => test_inB,
      o_aeq2b   => test_out_aeq2b,
      o_agtb    => test_out_agtb,
      o_altb    => test_out_altb
    );



  process
  begin
    -- Apply all 16 combinations of i_a and i_b
```

```
test_inA <= "00";
test_inB <= "00";
wait for 50 ns;

test_inA <= "00";
test_inB <= "01";
wait for 50 ns;

test_inA <= "00";
test_inB <= "10";
wait for 50 ns;

test_inA <= "00";
test_inB <= "11";
wait for 50 ns;

test_inA <= "01";
test_inB <= "00";
wait for 50 ns;

test_inA <= "01";
test_inB <= "01";
wait for 50 ns;

test_inA <= "01";
test_inB <= "10";
wait for 50 ns;

test_inA <= "01";
test_inB <= "11";
wait for 50 ns;

test_inA <= "10";
test_inB <= "00";
wait for 50 ns;

test_inA <= "10";
test_inB <= "01";
wait for 200 ns;

test_inA <= "10";
test_inB <= "10";
wait for 50 ns;
```

```
test_inA <= "10";
test_inB <= "11";
wait for 50 ns;

test_inA <= "11";
test_inB <= "00";
wait for 50 ns;

test_inA <= "11";
test_inB <= "01";
wait for 50 ns;

test_inA <= "11";
test_inB <= "10";
wait for 50 ns;

test_inA <= "11";
test_inB <= "11";
wait for 50 ns;



assert false
report "End of simulation."
severity failure;
  end process;
end behaviour;
```