ELECTRICAL & COMPUTER ENGINEERING

School of Engineering

EGRE 365 – Digital Systems

Homework No. 2

Name: Luis Barquero

Major: Computer Engineering

Due Date: 10/02/17

Honor Pledge: *I have neither given nor received any unauthorized help on this lab.*
Signed:

_____*Luis Barquero*_____

Since both subprograms will use the same inputs to find both the maximum and minimum values, alongside their indices, both subprograms were combined into a single VHDL model for a better implementation.

1) **Write a VHDL subprogram that implements the function of finding the maximum of 4 inputs.**

   The first step in implementing the subprogram was to first define the port, which includes 4 standard logic vectors size 8 (7 down to 0) for each one of the four inputs. From there, the initialization of the maximum and minimum index was created, both of standard logic type of size 2 (1 down to 0).

   Next, a procedure labeled finding_max was created that takes in inputs 0-3, finds the maximum value, and outputs the index of that maximum value.

   The way the procedure works is that it checks to see if current input is greater than the rest of the inputs. For example, it first checks if input0 is greater than input1, input2, and input3. If so, the input0 is considered the max value, and it outputs the index of input0. The same method is used for the rest of the inputs.

   At the same time, the procedure also outputs the value contained in the index, or in other words, it outputs the maximum number.

   See Appendix A for the VHDL Model code.

2) **Write a VHDL subprogram that implements the function of finding the minimum of 4 inputs.**

   Given that all inputs and outputs have already been initialized, then in order to find the minimum value, and a procedure labeled finding_min is created that takes in inputs 0-3, finds the minimum value, and outputs the index of the minimum value.

   This procedure works in a similar way as the finding_max procedure, except instead of checking if the current input is greater than the rest of the inputs, it checks if the current input is less than the rest of the inputs. If the condition is met, then the procedure outputs the index of the minimum value, alongside the value contained in the index.

   See Appendix A for the VHDL Mode code.

3) Write a VHDL entity and architecture that implements a comparator that finds out the maximum number and minimum number in 4 8-bit inputs.

For this part, there is a procedure call in the architecture for both the max and min finders that sends in the inputs, determines the max/min with their indices, and outputs the results.

4) Write a test bench to test your code for a minimum of 10 test cases.

In order to properly test out the model, the following 10 cases were constructed, as illustrated by table 1, where it shows all inputs alongside their max/min index and their max/min value.

Table 1 – Table 1 shows all 10 cases with their corresponding max/min index/value.

Case	Input0	Input1	Input2	Input3	Max_Index	Min_Index	Max_Value	Min_Value
1	0010_0010 (34)	0011_0011 (51)	0000_0000 (0)	1111_1111 (255)	11	10	1111_1111 (255)	0000_0000 (0)
2	0000_0000 (0)	0000_0000 (0)	1111_1111 (255)	1111_1111 (255)	10	00	1111_1111 (255)	0000_0000 (0)
3	0000_0001 (1)	0001_1001 (25)	0100_0000 (64)	0100_0000 (64)	10	00	0100_0000 (64)	0000_0001 (1)
4	0010_0100 (36)	0010_0100 (36)	0100_0000 (64)	0110_0101 (101)	11	00	0110_0101 (101)	0010_0100 (36)
5	0000_0011 (3)	0000_1100 (12)	0000_0110 (6)	0000_1001 (9)	01	00	0000_1100 (12)	0000_0011 (3)
6	0111_1010 (122)	0111_1000 (120)	0111_1011 (123)	0111_1001 (121)	10	01	0111_1011 (123)	0111_1000 (120)
7	1001_0010 (146)	1001_0010 (146)	1001_0010 (146)	1001_0010 (146)	00	00	1001_0010 (146)	1001_0010 (146)
8	1000_1000 (136)	1000_1000 (136)	1000_1000 (136)	1000_1000 (136)	00	00	1000_1000 (136)	1000_1000 (136)
9	0010_0100 (36)	0010_0100 (36)	0100_1000 (72)	0100_1000 (72)	10	00	0100_1000 (72)	0010_0100 (36)
10	0000_0010 (2)	0000_0101 (5)	0000_0001 (1)	0000_0111 (7)	11	10	0000_0111 (7)	0000_0001 (1)

Appendix B contains the simulation results from the testbench.

Appendix A

VHDL Code

Figure 1 – Figure 1 shows the code used for the VHDL Model

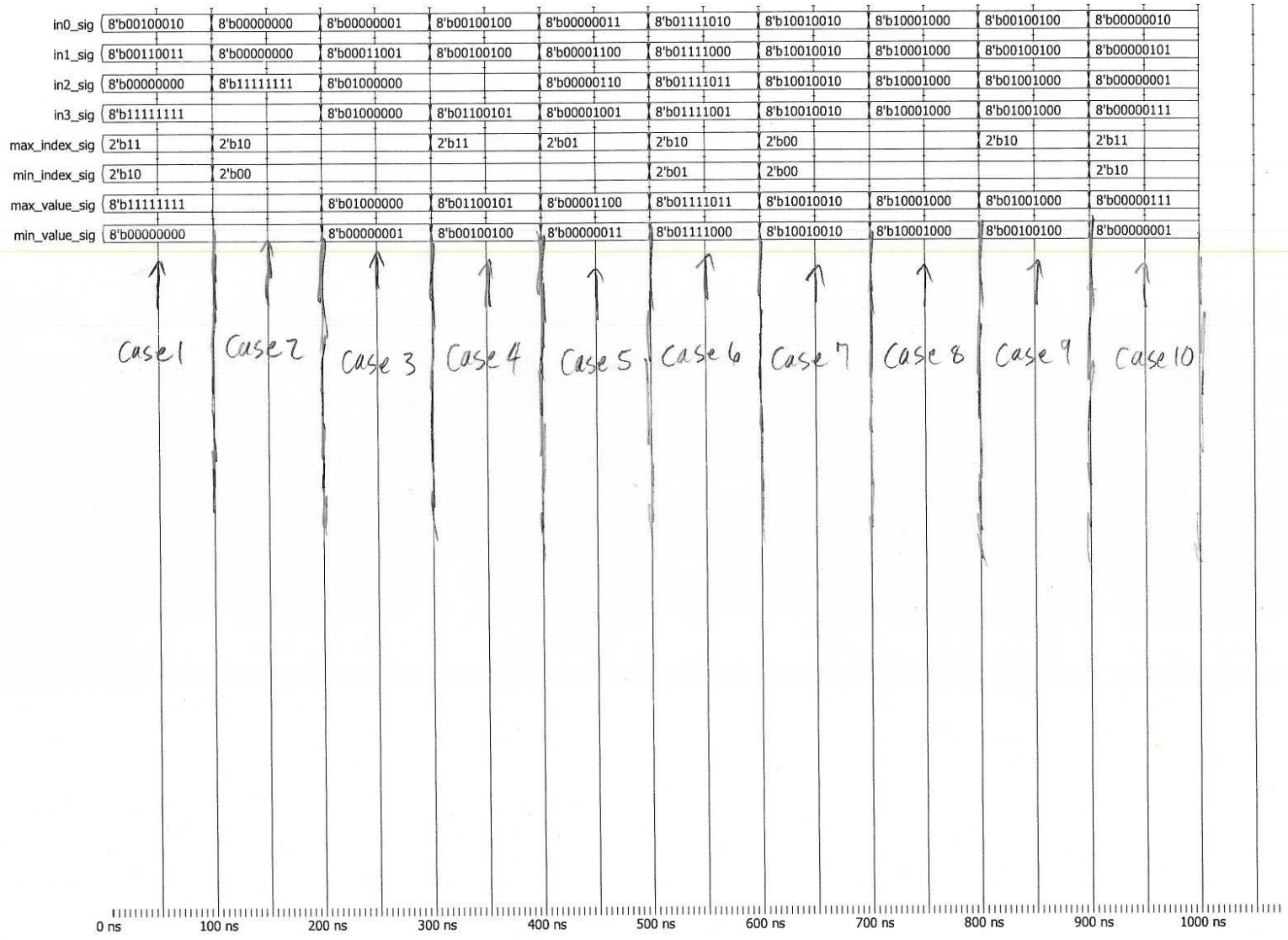
```
--Programmed by: Luis Baquero
--Purpose: This VHDL Model will take 4 inputs and will determine the maximum value, the minimum value
--         the index of the maximum value, and the index of the minimum value.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.ALL;

entity compareENT is
    Port ( in0     : in   STD_LOGIC_VECTOR (7 downto 0);               --Input 0
           in1     : in   STD_LOGIC_VECTOR (7 downto 0);               --Input 1
    signal max_index : out STD_LOGIC_VECTOR (1 downto 0)) is    --Signal to find the index of the max value

  begin
    -- Find maximum value
        if (in0 >= in1 and in0 >= in2 and in0 >= in3) then  --This will check to see if input 0 is greater than the rest of the inputs.
                max_value <= in0;                          --If input 0 is greater than the rest, it outputs the max value and the index.
                max_index <= b"00";
        elsif (in1 >= in0 and in1 >= in2 and in1 >= in3) then  --This will check to see if input 0 is greater than the rest of the inputs.
                max_value <= in1;                              --If input 1 is greater than the rest, it outputs the max value and the index.
                max_index <= b"01";
        elsif (in2 >= in0 and in2 >= in1 and in2 >= in3) then  --This will check to see if input 0 is greater than the rest of the inputs.
                max_value <= in2;                              --If input 2 is greater than the rest, it outputs the max value and the index.
                max_index <= b"10";
        elsif (in3 >= in0 and in3 >= in1 and in3 >= in2) then  --This will check to see if input 0 is greater than the rest of the inputs.
                max_value <= in3;                              --If input 3 is greater than the rest, it outputs the max value and the index.
                max_index <= b"11";
        end if;
  end finding_max;

procedure finding_min (signal input0, input1, input2, input3 : in STD_LOGIC_VECTOR (7 downto 0);
   signal min_value : out STD_LOGIC_VECTOR (7 downto 0);                --Signal to find min value of the inputs
   signal min_index : out STD_LOGIC_VECTOR (1 downto 0)) is             --Signal to find the index of the min value

begin
      -- Find minimum value
      if (in0 <= in1 and in0 <= in2 and in0 <= in3) then   --This will check to see if input 0 is greater than the rest of the inputs.
        min_value <= in0;                                 --If input 0 is greater than the rest, it outputs the max value and the index.
        min_index <= b"00";
      elsif (in1 <= in0 and in1 <= in2 and in1 <= in3) then  --This will check to see if input 0 is greater than the rest of the inputs.
        min_value <= in1;                                    --If input 1 is greater than the rest, it outputs the max value and the index.
        min_index <= b"01";
      elsif (in2 <= in0 and in2 <= in1 and in2 <= in3) then  --This will check to see if input 0 is greater than the rest of the inputs.
        min_value <= in2;                                    --If input 2 is greater than the rest, it outputs the max value and the index.
        min_index <= b"10";
      elsif (in3 <= in0 and in3 <= in1 and in3 <= in2) then  --This will check to see if input 0 is greater than the rest of the inputs.
        min_value <= in3;                                    --If input 3 is greater than the rest, it outputs the max value and the index.
        min_index <= b"11";
        end if;
end finding_min;

begin
    process(in0,in1, in2, in3)
      begin
        finding_max(in0, in1, in2, in3,         --Calls the procedure max_finder
                    max_val,max_idx);
        finding_min(in0, in1, in2, in3,         --Calls the procedure min_finder
                    min_val,min_idx);
    end process;

end simple;
```

Figure 2 – Figure 2 shows the code used for the VHDL Testbench

```
--Programmed by: Luis Baquero
--Purpose: This VHDL Testbench will take 4 inputs and will determine the maximum value, the minimum value
--         the index of the maximum value, and the index of the minimum value.
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY compareTB IS
END compareTB;

ARCHITECTURE behavior OF compareTB IS

   -- define the maximum delay for the DUT
   constant MAX_DELAY : time := 100 ns;
   constant BIT_WIDTH : integer := 8;
   constant NO_OF_VALUES : integer := 10;

 -- define signals that connect to DUT
   signal in0_sig  : std_logic_vector((BIT_WIDTH-1) downto 0);  --Input 0
   signal in1_sig  : std_logic_vector((BIT_WIDTH-1) downto 0);  --Input 1
   signal in2_sig  : std_logic_vector((BIT_WIDTH-1) downto 0);  --Input 2
   signal in3_sig  : std_logic_vector((BIT_WIDTH-1) downto 0);  --Input 3
   signal max_index_sig : std_logic_vector(1 downto 0);     --Max Index
   signal min_index_sig : std_logic_vector(1 downto 0);     --Min Index
   signal max_value_sig : std_logic_vector((BIT_WIDTH-1) downto 0);     --Max Value
   signal min_value_sig : std_logic_vector((BIT_WIDTH-1) downto 0);     --Min Value


   --declare a constant to hold arrays of input values for each input signal
   type input_value_array is array (0 to (NO_OF_VALUES-1)) of std_logic_vector ((BIT_WIDTH-1) downto 0);
   constant in0_sig_values : input_value_array := ("00100010","00000000",   --Test Cases 1 and 2 for Input 0
                                                   "00000001","00100100",   --Test Cases 3 and 4 for Input 0
                                                   "00000011","01111010",   --Test Cases 5 and 6 for Input 0
                                                   "10010010","10001000",   --Test Cases 7 and 8 for Input 0
                                                   "00100100","00000010");  --Test Cases 9 and 10 for Input 0

   constant in1_sig_values : input_value_array := ("00110011","00000000",       --Test Cases 1 and 2 for Input 1
                                                   "00011001","00100100",       --Test Cases 3 and 4 for Input 1
                                                   "00001100","01111000",       --Test Cases 5 and 6 for Input 1
                                                   "10010010","10001000",       --Test Cases 7 and 8 for Input 1
                                                   "00100100","00000101");      --Test Cases 9 and 10 for Input 1

   constant in2_sig_values : input_value_array := ("00000000","11111111",       --Test Cases 1 and 2 for Input 2
                                                   "01000000","01000000",       --Test Cases 3 and 4 for Input 2
                                                   "00000110","01111011",       --Test Cases 5 and 6 for Input 2
                                                   "10010010","10001000",       --Test Cases 7 and 8 for Input 2
                                                   "01001000","00000001");      --Test Cases 9 and 10 for Input 2

   constant in3_sig_values : input_value_array := ("11111111","11111111",       --Test Cases 1 and 2 for Input 3
                                                   "01000000","01100101",       --Test Cases 3 and 4 for Input 3
                                                   "00001001","01111001",       --Test Cases 5 and 6 for Input 3
                                                   "10010010","10001000",       --Test Cases 7 and 8 for Input 3
                                                   "01001000","00000111");      --Test Cases 9 and 10 for Input 3

   begin

        -- this is the process that will generate the inputs
    stimulus : process
      begin
        for i in 0 to (NO_OF_VALUES-1) loop
          in0_sig <= in0_sig_values(i);
          in1_sig <= in1_sig_values(i);
          in2_sig <= in2_sig_values(i);
          in3_sig <= in3_sig_values(i);
          wait for MAX_DELAY;
        end loop;
        wait; -- stop the process to avoid an infinite loop
    end process stimulus;
     -- this is the component instantiation for the
     -- DUT - the device we are testing
     DUT : entity work.compareENT(simple)
         port map(  in0     => in0_sig,
                    in1     => in1_sig,
                    in2     => in2_sig,
                    in3     => in3_sig,
                    max_idx => max_index_sig,
                    min_idx => min_index_sig,
                    max_val => max_value_sig,
                    min_val => min_value_sig);
end behavior;


```

Appendix B

VHDL Simulations



Entity:comparetb Architecture:behavior Date: Mon Oct 02 18:09:03 EDT 2017 Row: 1 Page: 1