CPE 129 **Homework 13_A Solution** **W. Pilkington**

1. Write the VHDL Code for a 3-to-8 Binary Decoder, with **Active Low outputs and with an Active Low strobe enable**.

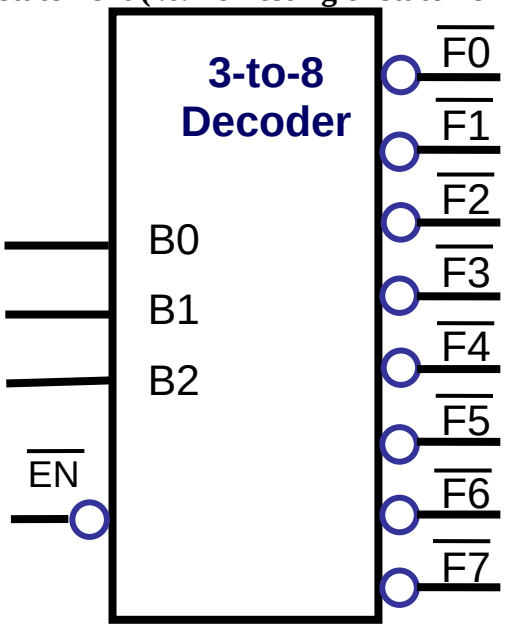
THIS TIME USE BEHAVIORAL MODELS (with a "Process") FOR YOUR VHDL DESCRIPTIONS. Compose 4 different Behavioral Models:

a) Using <u>one</u> IF-THEN-ELSIF-ELSE statement (inside a process) to describe the entire 3-to-8 Decoder device.

b) Using an IF-THEN-ELSIF-ELSE statement for the decoder logic (how the "B" inputs affect the outputs), <u>inside</u> of an IF-THEN-ELSE statement for the Enable logic (how "EN_L" affects the outputs). ["Nested" IF statements]

c) Using <u>both</u> a CASE statement and an "IF" statement in a single description for this device. Use the CASE statement for the decoder logic (how the "B" inputs affect the outputs), <u>inside</u> of an IF-THEN-ELSE statement for the Enable logic (how "EN_L" affects the outputs). [CASE nested inside an IF]

d) Using <u>separate</u> CASE and IF statements in a single process for this device. This time the CASE should <u>not</u> be inside the IF statement; <u>nor</u> should the IF be inside the CASE statement (*i.e.* no nesting of statements.)



So the point here was to show you that there are many different ways to describe the same device in VHDL; and to help you learn all the different programming constructs available to you.

EN′	B2	B1	B0	F0′	F1′	F2′	F3′	F4′	F5′	F6′	F7′
0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	0
1	-	-	-	1	1	1	1	1	1	1	1

```
entity BinaryDecoder3to8 is
   Port ( Binary_In : in  STD_LOGIC_VECTOR (2 downto 0);
          Enable_L: in STD_LOGIC;  -- active low
          Decode_Out_L : out  STD_LOGIC_VECTOR (7 downto 0));  --active low
end BinaryDecoder3to8;
```

a) Using one IF-THEN-ELSIF-ELSE statement (inside a process) to describe the device.

```
architecture Behavioral of BinaryDecoder3to8 is
        signal Enable: STD_LOGIC;                                   -- active high version
        signal Decode_Out :  STD_LOGIC_VECTOR (7 downto 0);  -- active high version
begin

                        -- Using a Behavioral Description (Process)
        decoder_behav: process (Binary_In, Enable) is
        begin
                        -- Handle the Enable first
            if  (Enable = '0')                  then Decode_Out <=  "00000000";

                        -- If enabled, then decode the input
               elsif    (Binary_In = "111")  then  Decode_Out <= "10000000";
               elsif    (Binary_In = "110")  then  Decode_Out <= "01000000";
               elsif    (Binary_In = "101")  then  Decode_Out <= "00100000";
               elsif    (Binary_In = "100")  then  Decode_Out <= "00010000";
               elsif    (Binary_In = "011")  then  Decode_Out <= "00001000";
               elsif    (Binary_In = "010")  then  Decode_Out <= "00000100";
               elsif    (Binary_In = "001")  then  Decode_Out <= "00000010";
               elsif    (Binary_In = "000")  then  Decode_Out <= "00000001";
               else                                Decode_Out <= "00000000";
            end if;

        end process decoder_behav;

                -- Convert to active low Enable and outputs
        Enable <= not( Enable_L);               -- This MUST be OUTSIDE the process.
        Decode_Out_L <= not(Decode_Out);   -- This MUST be OUTSIDE the process.

end Behavioral;
```

- - - - - - - - - - - - - - - - - - - OR - - - - - - - - - - - - - - - - - - -

(Alternative solution)

```
architecture Behavioral of BinaryDecoder3to8 is
        signal Enable: STD_LOGIC;
        signal Decode_Out :  STD_LOGIC_VECTOR (7 downto 0);
begin

                        -- Using a Behavioral Description (Process)
        decoder_behav: process (Binary_In, Enable) is
        begin

                        -- Set up a “default” that handles most output bits
                Decode_Out <=  "00000000"; -- we can get away with this in a process!

                        -- Handle the not Enabled case first
                if  (Enable = '0')                      then Decode_Out <=  "00000000";

                        -- else, Only change the 1 specific bit that is selected
                        -- (relying on the values provided above for the other bits)
                   elsif        (Binary_In = "111")  then  Decode_Out(7) <= ‘1’;
                   elsif        (Binary_In = "110”)  then  Decode_Out(6) <= ‘1’;
                   elsif        (Binary_In = "101”)  then  Decode_Out(5) <= ‘1’;
                   elsif        (Binary_In = "100”)  then  Decode_Out(4) <= ‘1’;
                   elsif        (Binary_In = "011”)  then  Decode_Out(3) <= ‘1’;
                   elsif        (Binary_In = "010”)  then  Decode_Out(2) <= ‘1’;
                   elsif        (Binary_In = "001”)  then  Decode_Out(1) <= ‘1’;
                   elsif        (Binary_In = "000”)  then  Decode_Out(0) <= ‘1’;
                   else                                    Decode_Out <= "00000000";
                 end if;
        end process decoder_behav;

                        -- Convert to active low Enable and outputs
        Enable <= not( Enable_L);               -- This MUST be OUTSIDE the process.
        Decode_Out_L <= not(Decode_Out);  -- This MUST be OUTSIDE the process.

end Behavioral;
```

b) Using an IF-THEN-ELSIF-ELSE statement for the decoder logic (how the "B" inputs affect the outputs), inside of an IF-THEN-ELSE statement for the Enable logic (how "EN′" affects the outputs).

```
architecture NestedBehavioral of BinaryDecoder3to8 is
        signal Enable: STD_LOGIC;
        signal Decode_Out :  STD_LOGIC_VECTOR (7 downto 0);
begin
        decoder_behav: process (Binary_In, Enable) is
        begin
                                -- Handle the Enable first
            if  (Enable = '0')          then Decode_Out <=  "0000000"; -- if disabled
            else
                        -- If enabled, decode the input
                if              (Binary_In = "111")  then  Decode_Out <= "10000000";
                   elsif        (Binary_In = "110")  then  Decode_Out <= "01000000";
                   elsif        (Binary_In = "101")  then  Decode_Out <= "00100000";
                   elsif        (Binary_In = "100")  then  Decode_Out <= "00010000";
                   elsif        (Binary_In = "011")  then  Decode_Out <= "00001000";
                   elsif        (Binary_In = "010")  then  Decode_Out <= "00000100";
                   elsif        (Binary_In = "001")  then  Decode_Out <= "00000010";
                   elsif        (Binary_In = "000")  then  Decode_Out <= "00000001";
                   else                                    Decode_Out <= "00000000";
                end if;
            end if;
        end process decoder_behav;

                        -- Convert to active low Enable and outputs
        Enable <= not( Enable_L);                 -- This MUST be OUTSIDE the process;
        Decode_Out_L <= not(Decode_Out);   -- This MUST be OUTSIDE the process;
end NestedBehavioral;
```

- - - - - - - - - - - - - - - - - - - OR - - - - - - - - - - - - - - - - - - -

(Alternative solution)

```
architecture NestedBehavioral2 of BinaryDecoder3to8 is
        signal Enable: STD_LOGIC;
        signal Decode_Out :  STD_LOGIC_VECTOR (7 downto 0);
begin

        decoder_behav: process (Binary_In, Enable) is

        begin
```

```
                    -- Handle the Enable first
        if  (Enable = '1')
        then      -- If enabled, decode the input

            if      (Binary_In = "111")  then  Decode_Out <= "10000000";
              elsif (Binary_In = "110")  then  Decode_Out <= "01000000";
              elsif (Binary_In = "101")  then  Decode_Out <= "00100000";
              elsif (Binary_In = "100")  then  Decode_Out <= "00010000";
              elsif (Binary_In = "011")  then  Decode_Out <= "00001000";
              elsif (Binary_In = "010")  then  Decode_Out <= "00000100";
              elsif (Binary_In = "001")  then  Decode_Out <= "00000010";
              elsif (Binary_In = "000")  then  Decode_Out <= "00000001";
              else                             Decode_Out <= "00000000";
            end if;

        else      Decode_Out <=  "00000000";  -- if not enabled
        end if;

    end process decoder_behav;

                -- Convert to active low Enable and outputs
    Enable <= not( Enable_L);              -- This MUST be OUTSIDE the process;
    Decode_Out_L <= not(Decode_Out);   -- This MUST be OUTSIDE the process;

end NestedBehavioral2;
```

c) Using both a CASE statement and an "IF" statement in a single description for this device. Use the CASE statement for the decoder logic (how the "B" inputs affect the outputs), inside of an IF-THEN-ELSE statement for the Enable logic (how EN′ affects the outputs). [CASE nested inside an IF]

```
architecture Behavioral of BinaryDecoder3to8 is
        signal Enable: STD_LOGIC;
        signal Decode_Out :  STD_LOGIC_VECTOR (7 downto 0);
begin

                        -- Using a Behavioral Description (Process)
        decoder_behav: process (Binary_In, Enable) is
        begin
                        -- Handle the Enable first
                if  (Enable = '0') then Decode_Out <=  "00000000";
                else
                        -- If enabled, decode the input
                        case (Binary_In) is
                                when "111" =>  Decode_Out <= "10000000";
                                when "110" =>  Decode_Out <= "01000000";
                                when "101" =>  Decode_Out <= "00100000";
                                when "100" =>  Decode_Out <= "00010000";
                                when "011" =>  Decode_Out <= "00001000";
                                when "010" =>  Decode_Out <= "00000100";
                                when "001" =>  Decode_Out <= "00000010";
                                when "000" =>  Decode_Out <= "00000001";
                                when others => Decode_Out <= "00000000";
                        end case;
                end if;
        end process decoder_behav;

                        -- Convert to active low Enable and outputs
        Enable <= not( Enable_L);
        Decode_Out_L <= not( Decode_Out );

end Behavioral;
```

[NOTE: The Xilinx VHDL compiler produced an identical circuit implementation for this Behavioral VHDL code as was generated in Assignment #12 from a Concurrent (Data-Flow) VHDL model. (This is a "good" thing!) The code above is actually a pretty well-constructed, understandable description of a decoder.]

d) Using separate CASE and IF statements in a single process for this device. This time the CASE should not be inside the IF statement; nor should the IF be inside the CASE statement (*i.e.* no nesting of statements.)

```
architecture Behavioral of BinaryDecoder3to8 is
        signal Enable: STD_LOGIC;
        signal Decode_Out :  STD_LOGIC_VECTOR (7 downto 0);
begin

                        -- Using a Behavioral Description (Process)
        decoder_behav: process (Binary_In, Enable) is
        begin
                -- decoder the input
                case (Binary_In) is
                        when "111" =>  Decode_Out <= "10000000";
                        when "110" =>  Decode_Out <= "01000000";
                        when "101" =>  Decode_Out <= "00100000";
                        when "100" =>  Decode_Out <= "00010000";
                        when "011" =>  Decode_Out <= "00001000";
                        when "010" =>  Decode_Out <= "00000100";
                        when "001" =>  Decode_Out <= "00000010";
                        when "000" =>  Decode_Out <= "00000001";
                        when others => Decode_Out <= "00000000";
                end case;
                        -- Handle the Enable LAST! in the process
                if  (Enable = '0')
                        then Decode_Out <=  "00000000";
                end if;

        end process decoder_behav;

                        -- Convert to active low Enable and outputs
        Enable <= not( Enable_L);
        Decode_Out_L <= not(Decode_Out);

end Behavioral;
```

This version is a little trickier than the previous VHDL solutions. Here we are relying on the peculiar behavior of statements inside a PROCESS. For multiple assignments to the same signal inside of a process, only the last assignment made will take effect. Therefore, if the decoder is enabled (Enable = 1), then the final

```
                        then Decode_Out <=  "00000000";
```

will never take place. However, if the decoder is not enabled (Enable=0), then this code will run and "cancel out" the decoding logic results from the CASE statement above this point in the PROCESS.

This is also one case where we can get away with not using an "ELSE", since the CASE statement tells VHDL what to do with Decode_Out *for all other possibilities (when Enable ≠ 0).*