Name: Anil Kumar

ROLL_NO. : 2020UCD2101

# DIGITAL LOGIC DESIGN PRACTICAL

# VHDL PROGRAMMING

# Exercise 1

Write vhdl code and testbench of 4:16 decoder using 3:8 decoder.

```
-- Code your design here
library IEEE;
use IEEE.std_logic_1164.all;

entity three_eight_decode is
port(
    enable: in Bit;
    i: in Bit_vector(2 downto 0);
    o: out Bit_vector(7 downto 0));
end three_eight_decode;
architecture ted of three_eight_decode is
begin
    process(i) is
    begin
        if enable = '1' then
          case i is
              when b"000" => o <= b"00000001";
              when b"001" => o <= b"00000010";
              when b"010" => o <= b"00000100";
              when b"011" => o <= b"00001000";
              when b"100" => o <= b"00010000";
              when b"101" => o <= b"00100000";
              when b"110" => o <= b"01000000";
              when b"111" => o <= b"10000000";
              when others => o <= b"00000000";
          end case;
        else
            o <= b"00000000";
        end if;
    end process;
end ted;
entity four_sixteen_decode is
port(
    msb_input: in Bit;
    input: in Bit_vector(2 downto 0);
    output1: out Bit_vector(7 downto 0);
    output2: out Bit_vector(7 downto 0));
```

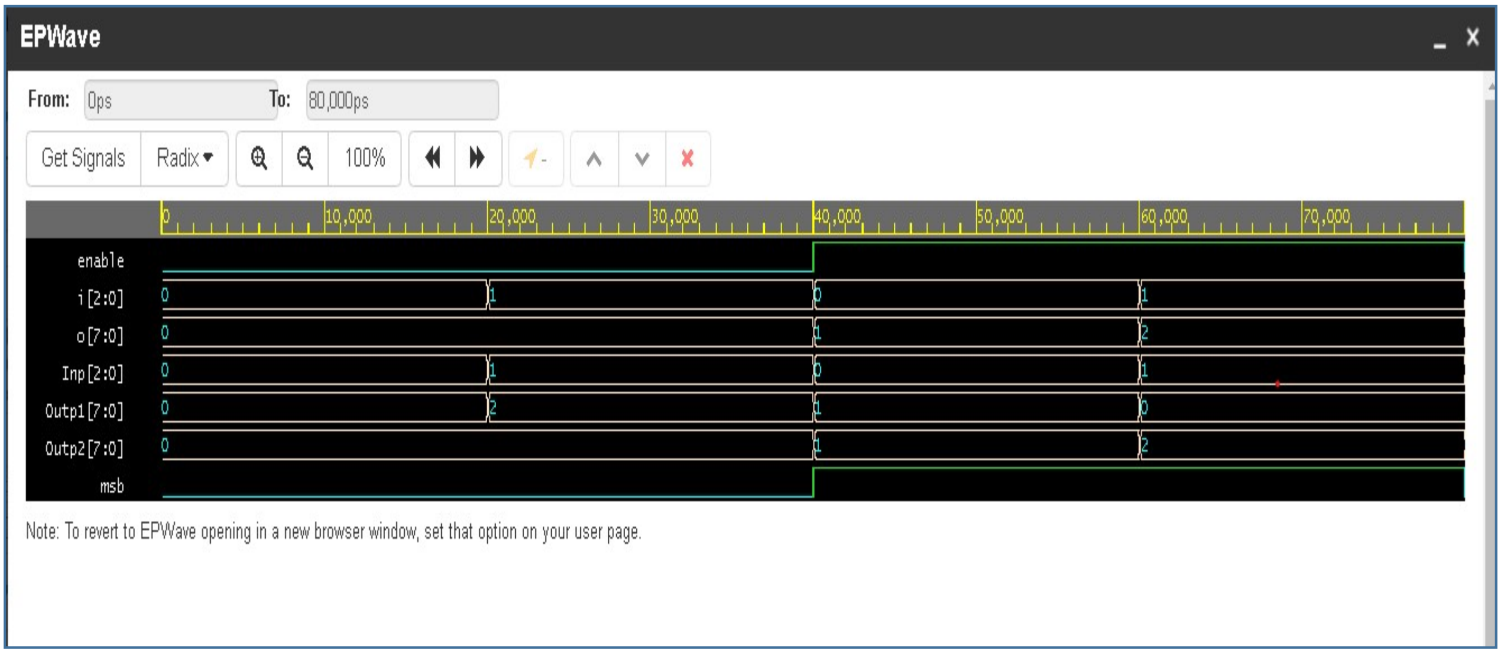
```
end four_sixteen_decode;
architecture fsd of four_sixteen_decode is
component three_eight_decode is
port(
    enable: in Bit;
    i: in Bit_vector(2 downto 0);
    o: out Bit_vector(7 downto 0));
end component;
begin
    --if msb_input = '0' then
    ted1: three_eight_decode port map ((not msb_input), input, output1);
    ted2: three_eight_decode port map (msb_input, input, output2);
end fsd;
```

```
-- Code your testbench here
library IEEE;
use IEEE.std_logic_1164.all;


entity decoder4x16_tb is
end decoder4x16_tb;
```

```
architecture be of decoder4x16_tb is
```

```
component four_sixteen_decode is
port(
msb_input: in Bit;
input: in Bit_vector(2 downto 0);
output1: out Bit_vector(7 downto 0);
output2: out Bit_vector(7 downto 0));
end component;
signal msb: bit;
signal Inp: Bit_vector(2 downto 0);
signal Outp1: Bit_vector(7 downto 0);
signal Outp2: Bit_vector(7 downto 0);
begin
DUT: four_sixteen_decode port map(msb,Inp,Outp1,Outp2);
process
    begin
    msb<='0';
    Inp<=b"000";
    wait for 20ns;
    assert(Outp1="10000000" and Outp2="00000000") report "Fail 0" sever
ity error;

    msb<='0';
    Inp<=b"001";
    wait for 20ns;
    assert(Outp1="01000000" and Outp2="00000000") report "Fail 0" sever
ity error;

    msb<='1';
    Inp<=b"000";
    wait for 20ns;
    assert(Outp1="00000000" and Outp2="10000000") report "Fail 0" sever
ity error;

    msb<='1';
    Inp<=b"001";
    wait for 20ns;
    assert(Outp1="00000000" and Outp2="01000000") report "Fail 0" sever
ity error;

    --clear input
    msb<='0';
    Inp<=b"000";
    assert false report "Test done." severity note;
    wait;
end process;
end be;
```

## OUTPUT


EPWave

From: 0ps To: 80,000ps

Note: To revert to EPWave opening in a new browser window, set that option on your user page.