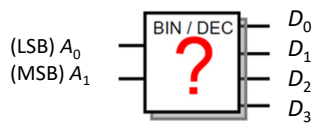


Tut 08

EE 2000

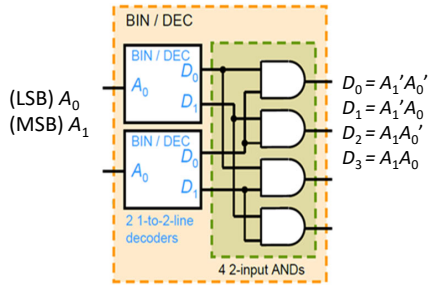
1. 2-to-4 Line Decoder (using 1-to-2 Line Decoder)



- a) Write the entity declaration for a 2-to-4 Line decoder module named **LDecoder2_4** that has two input named A0 and A1, and four outputs named D0, D1, D2, D3.

Solution:

```
entity LDecoder2_4 is
    Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           D0 : out STD_LOGIC;
           D1 : out STD_LOGIC;
           D2 : out STD_LOGIC;
           D3 : out STD_LOGIC);
end LDecoder2_4;
```



Inputs		Outputs			
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

b) Write the architecture declaration for the above 2-to-4 Line decoder named LDecoder2_4 with an architecture name "LDecoder1_2" using 2 1-to-2 Line decoders

```

43 architecture Behavioral of LDecoder2_4 is
44
45 component LDecoder1_2 is
46     Port ( A : in STD_LOGIC;
47           D0 : out STD_LOGIC;
48           D1 : out STD_LOGIC;
49     end component;
50
51 -- declare intermediate signals
52 signal SigA0: std_logic_vector(1 downto 0);
53 signal SigA1: std_logic_vector(1 downto 0);
54
55 begin
56
57 uut1: LDecoder1_2 port map(A0, SigA0(0), SigA0(1));
58 uut2: LDecoder1_2 port map(A1, SigA1(0), SigA1(1));
59
60 D0 <= SigA1(0) and SigA0(0);
61 D1 <= SigA1(0) and SigA0(1);
62 D2 <= SigA1(1) and SigA0(0);
63 D3 <= SigA1(1) and SigA0(1);
64
65 end Behavioral;

```

Virtual Circuit

1. Port for Unit Circuit

2. Signals for Circuits

3. Port map for virtual signals

4. Output for virtual circuits

2. 3-to-8 Decoder

a) Write the entity and architecture declaration for a 3-to-8 decoder module named Decoder3_8 that has one 3-bit input named S and one 8-bit output named Q.

Solution:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using...

entity Decoder3_to_8 is
    Port (
        S : in  STD_LOGIC_VECTOR (2 downto 0);
        Q : out STD_LOGIC_VECTOR (7 downto 0)
    );
end Decoder3_to_8;

architecture Behavioral of Decoder3_to_8 is
begin
    process(S)
    begin
        case S is
            when "000" => Q<="00000001";
            when "001" => Q<="00000010";
            when "010" => Q<="00000100";
            when "011" => Q<="00001000";
            when "100" => Q<="00010000";
            when "101" => Q<="00100000";
            when "110" => Q<="01000000";
            when "111" => Q<="10000000";
            when others => null;
        end case;
    end process;
end Behavioral;

```

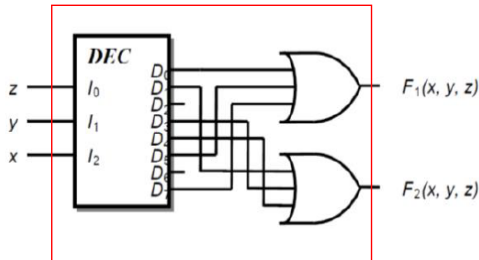
1. Use vectors to present 3 and 8 bits

2. Use case statement to present output Q.

b) Write the circuit named Q1 using VHDL with one 3-to-8 decoder and external gates.

$$F_1(x, y, z) = x'y'z' + xz$$

$$F_2(x, y, z) = xy'z' + x'z$$



```
entity Q1 is
    port (
        X : in STD_LOGIC;
        Y : in STD_LOGIC;
        Z : in STD_LOGIC;
        F1 : out STD_LOGIC;
        F2 : out STD_LOGIC);
end Q1;

architecture Behavioral of Q1 is
    component Decoder3_to_8
        port (
            S : in STD_LOGIC_VECTOR (2 downto 0);
            Q : out STD_LOGIC_VECTOR (7 downto 0));
    end component;
    signal S : std_logic_vector(2 downto 0);
    signal Q : std_logic_vector(7 downto 0);
begin
    uut: Decoder3_to_8 port map(S, Q);
    -- concatenation
    S <= X & Y & Z;
    F1 <= Q(0) or Q(5) or Q(7);
    F2 <= Q(1) or Q(3) or Q(4);
end Behavioral;
```

1. Entity Port for the Circuit

2. Define component for a virtual circuit

3. Signals for Circuits

4. Port map for vectors in virtual signals

5. Output the functions

3. Encoder

Write the entity and architecture declaration for a decimal-to-gray code encoder module named DTG_encoder that has one 10-bit input named D_in and one 4-bit output named G_out.

Inputs										Outputs			
0	1	2	3	4	5	6	7	8	9	A ₃	A ₂	A ₁	A ₀
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	0	0	0	0	1	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0	1	1	1	0	1

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

```
entity DTG_encoder is
  port (
    D_in : in std_logic_vector(9 downto 0);
    G_out : out std_logic_vector(3 downto 0)
  );
end DTG_encoder;
```

```
architecture Behavior of DTG_encoder is
begin
```

```
  process (D_in)
  begin
    G_out(3) <= D_in(8) OR D_in(9);
    G_out(2) <= D_in(4) OR D_in(5) OR D_in(6) OR D_in(7) OR
    D_in(8) OR
    D_in(9);
    G_out(1) <= D_in(2) OR D_in(3) OR D_in(4) OR D_in(5);
    G_out(0) <= D_in(1) OR D_in(2) OR D_in(5) OR D_in(6) OR D_in(9);
  end process;
end Behavior
```

Inputs										Outputs			
0	1	2	3	4	5	6	7	8	9	A ₃	A ₂	A ₁	A ₀
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	0	0	0	0	1	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0	1	1	1	0	1

4. JK Flip-Flop

Write the complete VHDL code for a negative-edge triggered JK Flip-Flop with asynchronous CLR.

C	J	K	Q_{t+1}	$\overline{Q}_{t+1}$	State
0	x	x	Q_t	$\overline{Q}_t$	Hold
1	0	0	Q_t	$\overline{Q}_t$	Hold
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	$\overline{Q}_t$	Q_t	Toggle

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

entity JKFF is
  port(
    j: in std_logic;
    k: in std_logic;
    clk: in std_logic;
    rst: in std_logic;
    q: out std_logic;
  end JKFF;
```

```
architecture behavioral of JKFF is
  signal tmp : std_logic;
begin
  process(j,k,clk,rst)
  begin
    if(rst = '1') then
      tmp <= '0';
    elsif (falling_edge(clk)) then
      if(j /= k) then
        tmp <= j;
      elsif(j = '1' and k = '1') then
        tmp <= not tmp;
      end if;
    end if;
    q <= tmp;
  end process;
end behavioral;
```