1. Design a 2 X 1 Multiplexer that will select the binary information from one of the two input lines and direct it to a single output line based on the value of a selection line.

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

-- Entity declaration
entity MUX2to1 is
    Port (
        A : in STD_LOGIC;    -- First input
        B : in STD_LOGIC;    -- Second input
        SEL : in STD_LOGIC;  -- Selection line
        Y : out STD_LOGIC    -- Output
    );
end MUX2to1;

-- Architecture declaration
architecture Behavioral of MUX2to1 is
begin
    -- Process implementing the 2-to-1 multiplexer logic
    process (A, B, SEL)
    begin
        if SEL = '0' then
            Y <= A;  -- Select input A
        else
            Y <= B;  -- Select input B
        end if;
    end process;
end Behavioral;
```

2. Design a 3-to-8-line decoder with active low enable input using NAND gates only.

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

entity decoder3to8_nand is
   Port (
      A : in STD_LOGIC_VECTOR(2 downto 0); -- 3-bit input
      EN : in STD_LOGIC;                   -- Active low enable input
      Y : out STD_LOGIC_VECTOR(7 downto 0) -- 8 outputs
   );
end decoder3to8_nand;

architecture Behavioral of decoder3to8_nand is
begin
   process(A, EN)
   begin
      if EN = '0' then -- Active low enable
         Y(0) <= not (not A(2) and not A(1) and not A(0));
         Y(1) <= not (not A(2) and not A(1) and A(0));
         Y(2) <= not (not A(2) and A(1) and not A(0));
         Y(3) <= not (not A(2) and A(1) and A(0));
         Y(4) <= not (A(2) and not A(1) and not A(0));
         Y(5) <= not (A(2) and not A(1) and A(0));
         Y(6) <= not (A(2) and A(1) and not A(0));
         Y(7) <= not (A(2) and A(1) and A(0));
      else
         Y <= (others => '1'); -- Outputs are inactive when EN is high
      end if;
   end process;

end Behavioral;
```

3. Design a full adder using a 3-to-8-line decoder and external OR gates.

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

entity full_adder_decoder is
   Port (
      A : in STD_LOGIC;  -- First input bit
      B : in STD_LOGIC;  -- Second input bit
      Cin : in STD_LOGIC; -- Carry input
      Sum : out STD_LOGIC; -- Sum output
      Cout : out STD_LOGIC -- Carry output
   );
end full_adder_decoder;

architecture Behavioral of full_adder_decoder is
   signal decoder_out : STD_LOGIC_VECTOR(7 downto 0); -- Output of the 3-to-8 decoder
begin
   -- Instantiate 3-to-8 decoder
   process(A, B, Cin)
   begin
      decoder_out <= (others => '0');
      case (A & B & Cin) is
         when "000" => decoder_out(0) <= '1';
         when "001" => decoder_out(1) <= '1';
         when "010" => decoder_out(2) <= '1';
         when "011" => decoder_out(3) <= '1';
         when "100" => decoder_out(4) <= '1';
         when "101" => decoder_out(5) <= '1';
         when "110" => decoder_out(6) <= '1';
         when "111" => decoder_out(7) <= '1';
         when others => decoder_out <= (others => '0');
      end case;
   end process;

   -- Generate Sum output using external OR gates
   Sum <= decoder_out(1) or decoder_out(2) or decoder_out(4) or decoder_out(7);

   -- Generate Cout output using external OR gates
   Cout <= decoder_out(3) or decoder_out(5) or decoder_out(6) or decoder_out(7);

end Behavioral;
```

4. Design a 4-bit priority encoder with inputs D3 (MSB), D2, D1 and D0 (LSB) and outputs X, Y and V. The priority assigned to inputs is D3 > D2 > D1 > D0. The output V shows a value 1 when one or more inputs are equal to one. If all inputs are 0, V is equal to 0. When V=0, then other two outputs are not inspected and are specified as don’t care conditions.

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

entity priority_encoder_4bit is
   Port (
      D : in STD_LOGIC_VECTOR(3 downto 0); -- Inputs D3 (MSB) to D0 (LSB)
      X : out STD_LOGIC;                -- Output X
      Y : out STD_LOGIC;                -- Output Y
      V : out STD_LOGIC                 -- Valid output V
   );
end priority_encoder_4bit;

architecture Behavioral of priority_encoder_4bit is
begin
   process(D)
   begin
      if D = "0000" then
         V <= '0';  -- No input is active
         X <= 'X'; -- Don't care
         Y <= 'X'; -- Don't care
      else
         V <= '1';  -- At least one input is active
         if D(3) = '1' then
            X <= '1';
            Y <= '1';
         elsif D(2) = '1' then
            X <= '1';
            Y <= '0';
         elsif D(1) = '1' then
            X <= '0';
            Y <= '1';
         elsif D(0) = '1' then
            X <= '0';
            Y <= '0';
         end if;
      end if;
   end process;

end Behavioral;
```