VLSI Lab Report Assignment 2
Annexure 2

BCSE 4th Year 2nd Semester

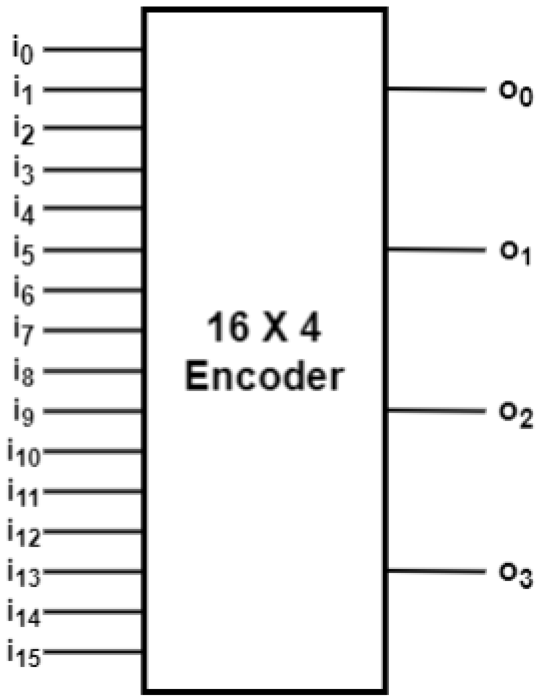
Name: ***Priti Shaw***
Roll No. : ***001710501076***
Batch: ***A3***

1. Design 16X4 Encoder using function and procedure.

Description

An encoder in digital electronics is a one-hot to binary converter. That is if there are 2n input lines, and at most only one of them will ever be high, the binary code of this 'hot' line is produced on the n-bit output lines. A 16-to-4 encoder has 16 input lines and 4 output line.

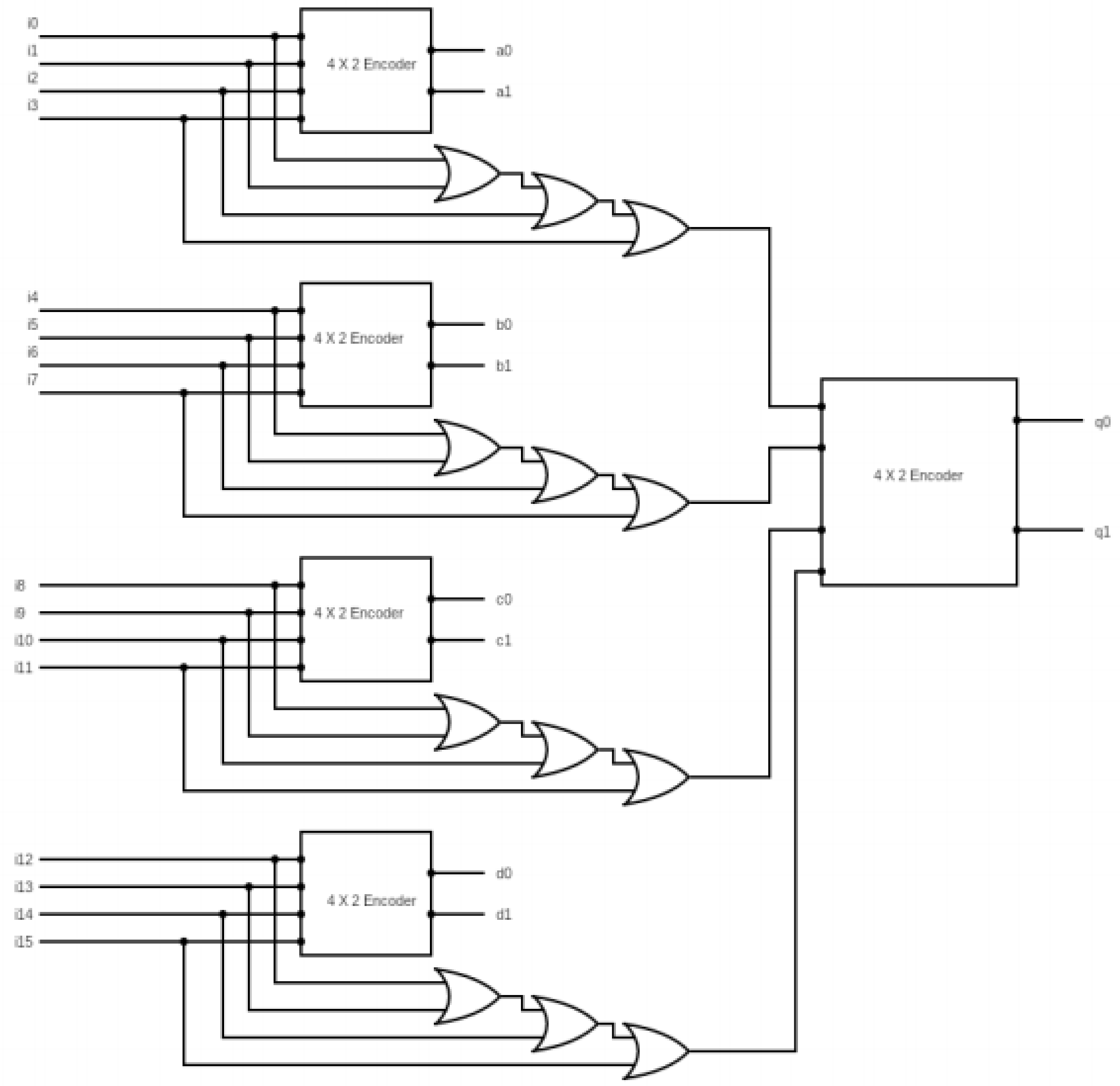
Block Diagram



Entity

```
entity 16X4_Encoder is
    Port ( ii : in  STD_LOGIC_VECTOR (15 downto 0);
           oo : out  STD_LOGIC_VECTOR (3 downto 0));
end Function_16X4_Encoder;
```

Circuit Diagram



Architecture

a) Using function of 4X2 encoders only

```
architecture Behavioral of Function_16X4_Encoder is
function Encoder_4X2_function(i:std_logic_vector)return std_logic_vector;

function Encoder_4X2_function(i:std_logic_vector)return std_logic_vector is
variable a:std_logic_vector(1 downto 0);
      begin
            if i="0001" then
                  a:="00";
```

```vhdl
            elsif i="0010" then
                  a:="01";
            elsif i="0100" then
                  a:="10";
            elsif i="1000" then
                  a:="11";
            else
                  a:="ZZ";
            end if;
      return a;
end function;

begin
process(ii)
      variable p:std_logic_vector(3 downto 0);
      variable k,j:integer;
      variable y:std_logic_vector(9 downto 0);
      begin
            for k in 0 to 3 loop
                  p(k):='0';
                  for j in 0 to 3 loop
                        p(k):=p(k) or ii(4*k+j);
                  end loop;
                  y(2*k+1 downto 2*k):=Encoder_4X2_function(ii(4*k+3 downto
4*k));
            end loop;
            y(9 downto 8):=Encoder_4X2_function(p);
            if y(9 downto 8)="ZZ" then
                  oo<="ZZZZ";
            else
                  for k in 0 to 3 loop
                        if p(k)='1' then
                              if y(2*k+1 downto 2*k)="ZZ" then
                                    oo<="ZZZZ";
                              else
                                    oo<=y(9 downto 8) & y(2*k+1 downto
2*k);
                              end if;
                        end if;
                  end loop;
            end if;
end process;
end Behavioral;
```

b) Using procedure of 4X2 encoders only

```
architecture Behavioral of Procedure_16X4_Encoder is

procedure Encoder_4X2_procedure(i:in std_logic_vector;o:out
std_logic_vector);

procedure Encoder_4X2_procedure(i:in std_logic_vector;o:out
std_logic_vector) is
begin
      if i="0001" then
            o:="00";
      elsif i="0010" then
            o:="01";
      elsif i="0100" then
            o:="10";
      elsif i="1000" then
            o:="11";
      else
            o:="ZZ";
      end if;
end procedure;

begin
      process(ii)
      variable p:std_logic_vector(3 downto 0);
      variable k,j:integer;
      variable y:std_logic_vector(9 downto 0);
      begin
      for k in 0 to 3 loop
            p(k):='0';
            for j in 0 to 3 loop
                  p(k):=p(k) or ii(4*k+j);
            end loop;
            Encoder_4X2_procedure(ii(4*k+3 downto 4*k),y(2*k+1 downto
2*k));
      end loop;
      Encoder_4X2_procedure(p,y(9 downto 8));
      if y(9 downto 8)="ZZ" then
            oo<="ZZZZ";
      else
            for k in 0 to 3 loop
```

```
                  if p(k)='1' then
                        if y(2*k+1 downto 2*k)="ZZ" then
                              oo<="ZZZZ";
                        else
                              oo<=y(9 downto 8) & y(2*k+1 downto 2*k);
                        end if;
                  end if;
            end loop;
      end if;
end process;
end Behavioral;
```

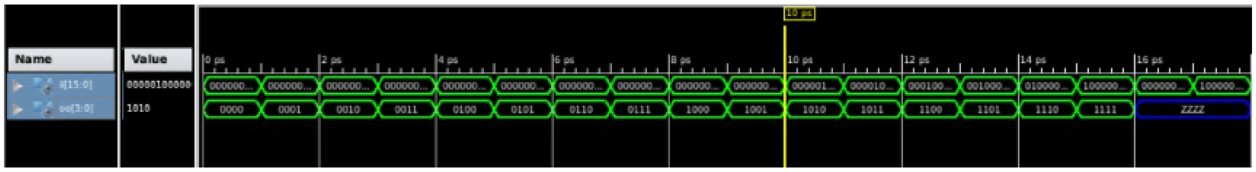
TestBench

```
ENTITY Function_16X4_Encoder_TestBench IS
END Function_16X4_Encoder_TestBench;

ARCHITECTURE behavior OF Function_16X4_Encoder_TestBench IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT Function_16X4_Encoder
    PORT(
         ii : IN  std_logic_vector(15 downto 0);
         oo : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal ii : std_logic_vector(15 downto 0) := (others => '0');
      --Outputs
   signal oo : std_logic_vector(3 downto 0);

BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: Function_16X4_Encoder PORT MAP (
          ii => ii,
          oo => oo
        );
   -- Stimulus process
   stim_proc: process
      variable a,k:integer;
   begin
```

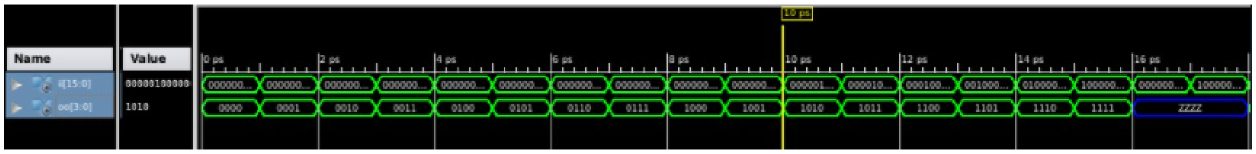
```vhdl
            for k in 0 to 15 loop
                  for a in 0 to 15 loop
                        ii(a)<='0';
                  end loop;
                  ii(k)<='1';
                  wait for 1 ps;
            end loop;
            ii<=(others=>'0');
            ii(0)<='1';
            ii(1)<='1';
            wait for 1 ps;
            ii<=(others=>'0');
            ii(0)<='1';
            ii(15)<='1';
            wait for 1 ps;
   end process;
END;
```

Timing Diagram

a) Using function of 4X2 encoders only


Name
Value
ii[15:0]
oo[3:0]
0000100000
1010
10 ps
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 ZZZZ

b) Using procedure of 4X2 encoders only


Name
Value
ii[15:0]
oo[3:0]
0000100000
1010
10 ps
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 ZZZZ