16x4 Encoder

VLSI Systems
Assignment-2 Annexure-II

PREPARED BY

Md Sahil

BCSE-IV

001710501029

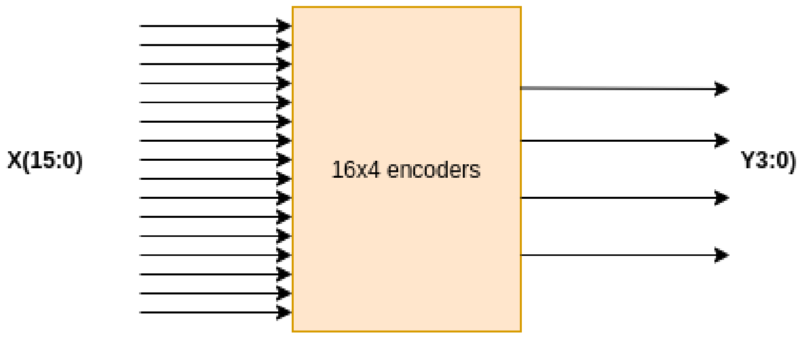
Jadavpur University

Description

Designing 16x4 encoder using 4x2 encoders with

1. Function of 4x2 encoders
2. Procedure of 4x2 encoders

Block Diagram

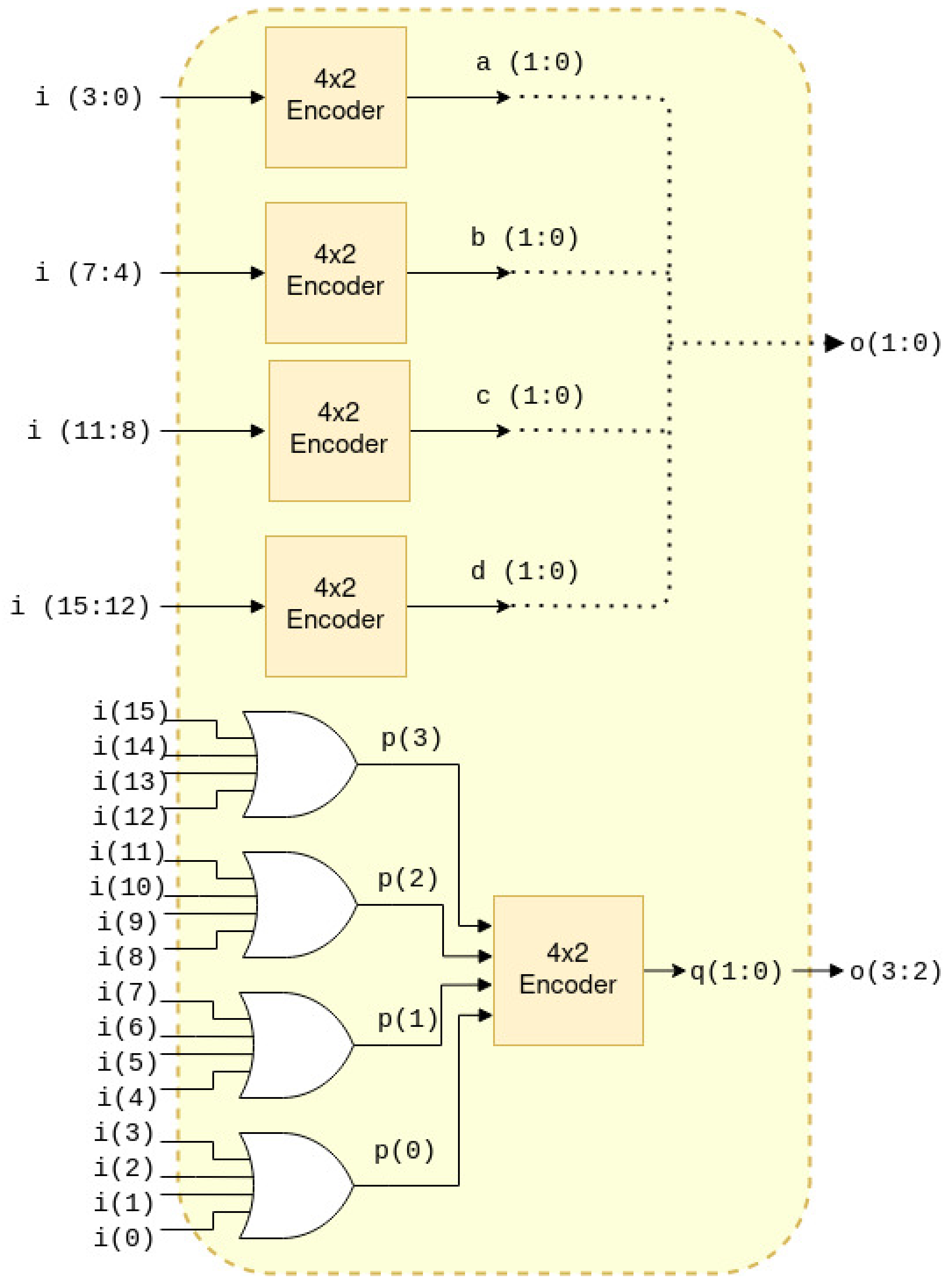


16x4 Encoder

Truth Table

X(15:0)																Y(3:0)			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	z	z	z	z
0	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	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Circuit Diagram



Code

1. Using functions of 4x2 encoders

```
architecture Behavioral of ass2_annex2a is
function mux4x2(x: in std_logic_vector) return std_logic_vector is
   variable a: std_logic_vector(1 downto 0);
   begin
       if x = "0001" then
           a := "00";
       elsif x = "0010" then
           a := "01";
       elsif x = "0100" then
           a := "10";
       elsif x = "1000" then
           a := "11";
       else
           a := "ZZ";
       end if;
       return a;
   end function;

begin

   p1: process(X)
   variable j,k: integer;
   variable a: std_logic_vector(7 downto 0);
   variable q: std_logic_vector(1 downto 0);
   variable p: std_logic_vector(3 downto 0);

   begin
       for k in 0 to 3 loop
           a(2*k+1 downto 2*k) := mux4x2(X(4*k+3 downto 4*k));
           p(k) := '0';
           for j in 0 to 3 loop
               p(k) := p(k) or X(4*k + j);
           end loop;
       end loop;

       q := mux4x2(p);
       if q = "ZZ" then
           Y <= "ZZZZ";
       else
           for k in 0 to 3 loop
```

```
                        if p(k) = '1' then
                            if a(2*k+1 downto 2*k) = "ZZ" then
                                Y <= "ZZZZ";
                            else
                                Y <= q & a(2*k+1 downto 2*k);
                            end if;
                        end if;
                    end loop;
                end if;
            end process;
        end Behavioral;
```

2. Using procedures of 4x2 encoders

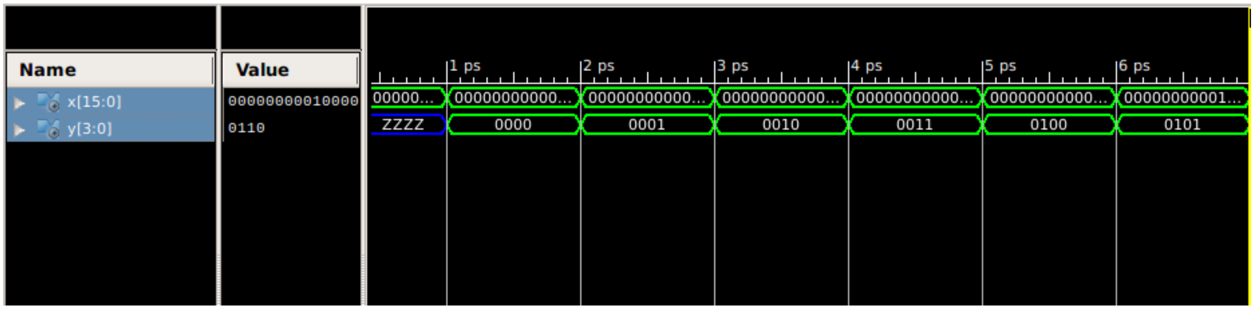
```
architecture Behavioral of ass2_annex2b is
      procedure mux4x2(x:in std_logic_vector; y:out std_logic_vector) is
            variable a: std_logic_vector(1 downto 0);
            begin
                   if x = "0001" then
                       a := "00";
                   elsif x = "0010" then
                       a := "01";
                   elsif x = "0100" then
                       a := "10";
                   elsif x = "1000" then
                       a := "11";
                   else
                       a := "ZZ";
                   end if;
                   y := a;
      end procedure;

begin

   p1: process(X)
   variable j,k: integer;
   variable a: std_logic_vector(7 downto 0);
   variable q: std_logic_vector(1 downto 0);
   variable p: std_logic_vector(3 downto 0);
   begin
       for k in 0 to 3 loop
           proc1: mux4x2(X(4*k+3 downto 4*k), a(2*k+1 downto 2*k));
           p(k) := '0';
           for j in 0 to 3 loop
```

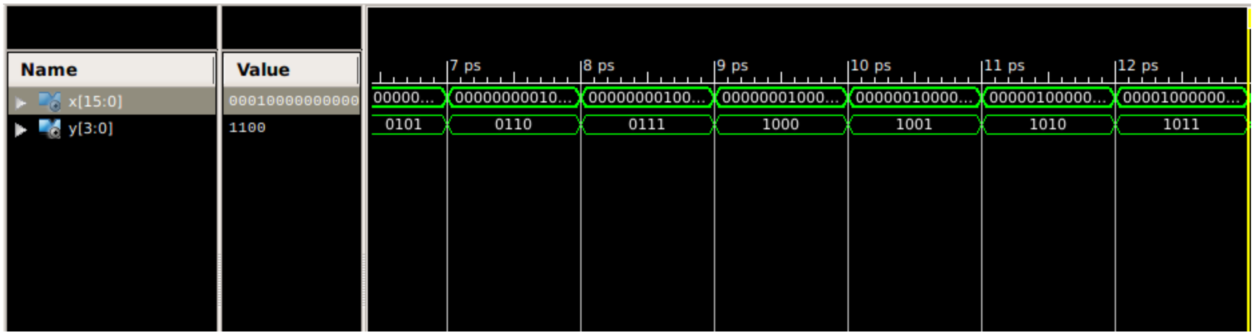
```
                p(k) := p(k) or X(4*k+j);
            end loop;
        end loop;
        proc2: mux4x2(p, q);
        if q = "ZZ" then
            Y <= "ZZZZ";
        else
            for k in 0 to 3 loop
                if p(k) = '1' then
                    if a(2*k+1 downto 2*k) = "ZZ" then
                        Y <= "ZZZZ";
                    else
                        Y <= q & a(2*k+1 downto 2*k);
                    end if;
                end if;
            end loop;
        end if;
    end process;
end Behavioral;
```

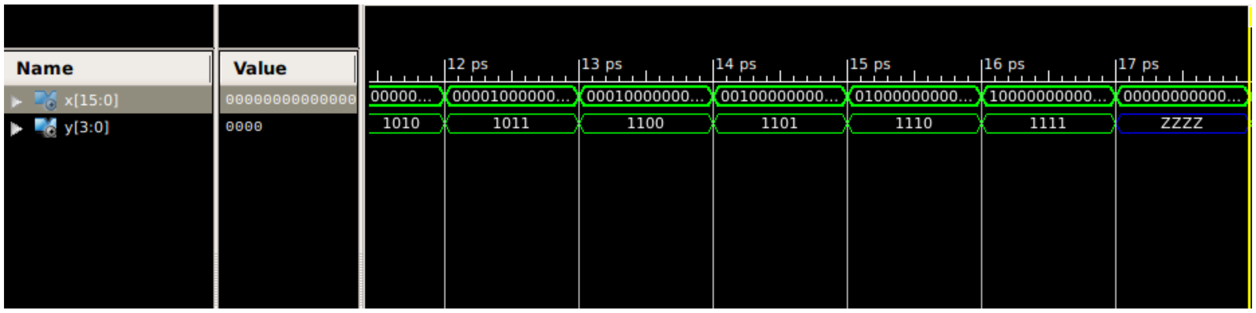
Test Bench

```
ARCHITECTURE behavior OF ass2_annex1_test_bench IS
   -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT ass2_annex1
   PORT(
        X : IN  std_logic_vector(15 downto 0);
        Y : OUT  std_logic_vector(3 downto 0)
       );
   END COMPONENT;


  --Inputs
  signal X : std_logic_vector(15 downto 0) := (others => '0');

  --Outputs
  signal Y : std_logic_vector(3 downto 0);
BEGIN
  -- Instantiate the Unit Under Test (UUT)
  uut: ass2_annex1 PORT MAP (
         X => X,
         Y => Y
       );
  -- Stimulus process
  stim_proc: process
  begin
     X <= "0000000000000000";
     wait for 1 ps;
     for i in 0 to 15 loop
        X(i) <= '1';
        wait for 1ps;
        X(i) <= '0';
     end loop;
  end process;
END;
```

Timing Diagram


Name
Value
x[15:0]
y[3:0]
0000000000010000
0110
1 ps
2 ps
3 ps
4 ps
5 ps
6 ps
00000...
000000000000...
000000000000...
000000000000...
000000000000...
000000000000...
000000000001...
ZZZZ
0000
0001
0010
0011
0100
0101


Name
Value
x[15:0]
y[3:0]
0001000000000000
1100
7 ps
8 ps
9 ps
10 ps
11 ps
12 ps
00000...
000000000010...
000000000100...
000000001000...
000000010000...
000000100000...
000001000000...
0101
0110
0111
1000
1001
1010
1011


Name
Value
x[15:0]
y[3:0]
0000000000000000
0000
12 ps
13 ps
14 ps
15 ps
16 ps
17 ps
00000...
000010000000...
000100000000...
001000000000...
010000000000...
100000000000...
000000000000...
1010
1011
1100
1101
1110
1111
ZZZZ