Binary Encoders

VLSI Systems
Assignment-2

PREPARED BY

Priyank Lohariwal

BCSE-IV

001710501055

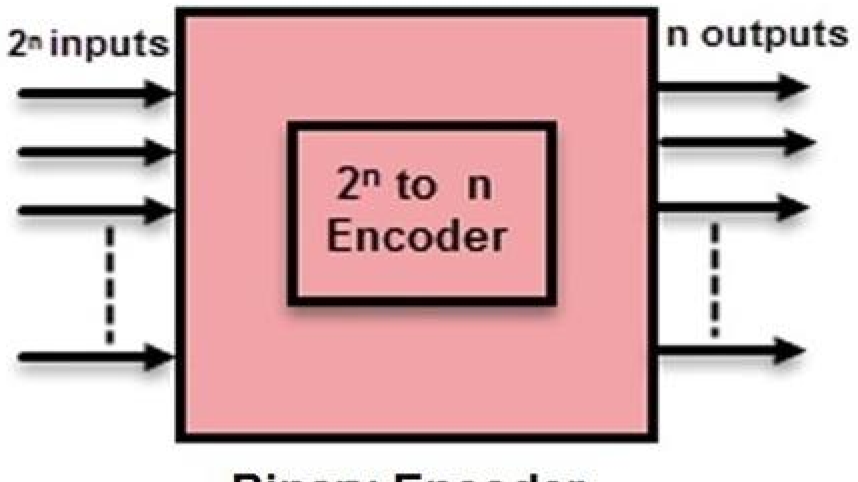
Jadavpur University

Description

Design various types of encoders

1. 2x1 encoder using behavioural modelling
    1.1. Using if-else statement
    1.2. Using case statements
    1.3. Using when-else statements
    1.4. Using select statements
2. 4x2 encoder
    2.1. Using gate-level modelling
    2.2. Using behavioural modelling
        2.2.1. Using if-else statements
        2.2.2. Using case statements
        2.2.3. Using when-else statements
        2.2.4. Using select statements
3. 8x3 encoder
    3.1. Using gate-level modelling
    3.2. Using behavioural modelling
4. 8x3 encoder using 4x2 and 2x1 encoders using component instantiation
5. 16x4 encoder using 4x2 encoders.

Block Diagram



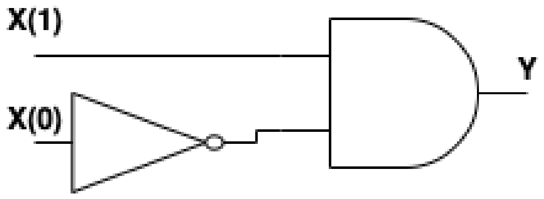
Binary Encoder

2x1 Encoders

Truth Table

X(1)	X(0)	Y
0	0	z
0	1	0
1	0	1
1	1	z

Circuit Diagram



Code

1.1. Using if-else statements

```
entity encoder2x1 is
   Port ( X : in  STD_LOGIC_VECTOR (1 downto 0);
          Y : out  STD_LOGIC);
end encoder2x1;

architecture Behavioral of encoder2x1 is
begin
  p1: process(X)
   begin
       if X = "01" then
           Y <= '0';
       elsif X = "10" then
           Y <= '1';
       else
           Y <= 'Z';
       end if;
   end process p1;

end Behavioral;
```

1.2. Using case statements

```
entity ass2_1b is
   Port ( X : in  STD_LOGIC_VECTOR (1 downto 0);
          Y : out  STD_LOGIC);
end ass2_1b;

architecture Behavioral of ass2_1b is
begin
   p1: process(X)
   begin
       case X is
           when "01" => Y <= '0';
           when "10" => Y <= '1';
           when others => Y <= 'Z';
       end case;
   end process p1;
end Behavioral;
```

1.3. Using when-else statements

```
entity ass2_1c is
   Port ( X : in  STD_LOGIC_VECTOR (1 downto 0);
          Y : out  STD_LOGIC);
end ass2_1c;
architecture Behavioral of ass2_1c is
begin
   Y <= '0' when X = "01" else
         '1' when X = "10" else
         'Z';
end Behavioral;
```

1.4. Using select statements

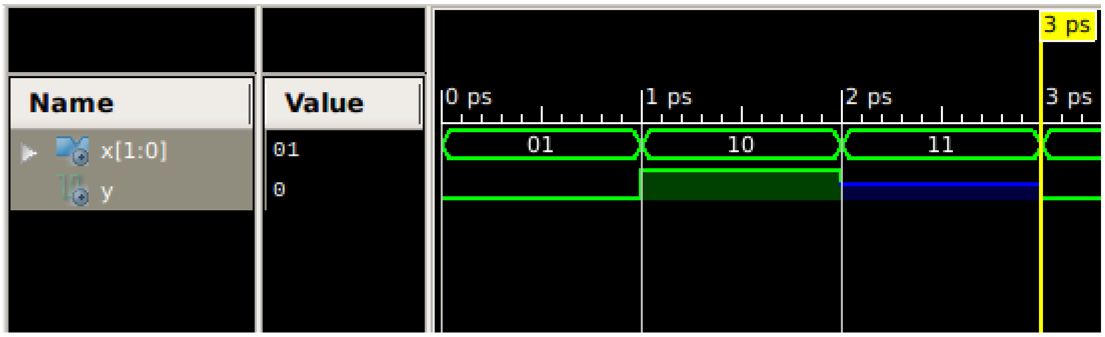
```
entity ass2_1d is
   Port ( X : in  STD_LOGIC_VECTOR (1 downto 0);
          Y : out  STD_LOGIC);
end ass2_1d;
architecture Behavioral of ass2_1d is
begin
   with X select
       Y <= '0' when "01",
            '1' when "10",
             'Z' when others;
end Behavioral;
```

Test Bench

```
ENTITY encoder2x1_test_bench IS
END encoder2x1_test_bench;
ARCHITECTURE behavior OF encoder2x1_test_bench IS
   COMPONENT encoder2x1
   PORT(
        X : IN  std_logic_vector(1 downto 0);
        Y : OUT  std_logic
       );
   END COMPONENT;

  signal X : std_logic_vector(1 downto 0) := (others => '0');
  signal Y : std_logic;
BEGIN
  uut: encoder2x1 PORT MAP (
         X => X,
         Y => Y
       );
  stim_proc: process
  begin
   X <= "01";
   wait for 1 ps;
   X <= "10";
   wait for 1 ps;
   X <= "11";
   wait for 1 ps;
  end process;
END;
```

Note: Test Bench would be same for all the different implementations of encoders mentioned here with only components name changed

Timing Diagram

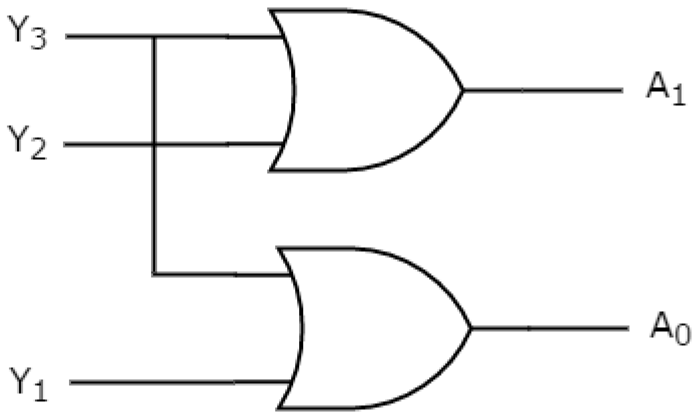


4x2 Encoders

Truth Table

X(3)	X(2)	X(1)	X(0)	Y(1)	Y(0)
0	0	0	0	z	z
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Circuit Diagram



Code

2.1. Using gate-level modelling

```
entity ass2_2a is
   Port ( X : in  STD_LOGIC_VECTOR (3 downto 0);
          Y : out  STD_LOGIC_VECTOR (1 downto 0));
end ass2_2a;

architecture Behavioral of ass2_2a is
begin
   Y(0) <= X(1) or X(3);
   Y(1) <= X(2) or X(3);
end Behavioral;
```

2.2.1 Using if-else statements

```
entity ass2_2ba is
   Port ( X : in  STD_LOGIC_VECTOR (3 downto 0);
          Y : out  STD_LOGIC_VECTOR (1 downto 0));
end ass2_2ba;

architecture Behavioral of ass2_2ba is
begin
   p1: process(X)
   begin
       if X = "0001" then
           Y <= "00";
       elsif X = "0010" then
           Y <= "01";
       elsif X = "0100" then
           Y <= "10";
       elsif X = "1000" then
           Y <= "11";
       else
           Y <= "ZZ";
       end if;
   end process p1;
end Behavioral;
```

2.2.2. Using case statements

```
entity ass2_2bb is
   Port ( X : in  STD_LOGIC_VECTOR (3 downto 0);
          Y : out  STD_LOGIC_VECTOR (1 downto 0));
end ass2_2bb;

architecture Behavioral of ass2_2bb is
begin
   p1: process(X)
   begin
       case X is
           when "0001" => Y <= "00";
           when "0010" => Y <= "01";
           when "0100" => Y <= "10";
           when "1000" => Y <= "11";
           when others => Y <= "ZZ";
       end case;
   end process p1;
end Behavioral;
```

2.2.3. Using when-else statements

```
entity ass2_2bc is
   Port ( X : in  STD_LOGIC_VECTOR (3 downto 0);
          Y : out  STD_LOGIC_VECTOR (1 downto 0));
end ass2_2bc;

architecture Behavioral of ass2_2bc is
begin
   Y <= "00" when X = "0001" else
        "01" when X = "0010" else
        "10" when X = "0100" else
        "11" when X = "1000" else
        "ZZ";
end Behavioral;
```

2.2.4. Using select statements

```
entity ass2_2bd is
   Port ( X : in  STD_LOGIC_VECTOR (3 downto 0);
          Y : out  STD_LOGIC_VECTOR (1 downto 0));
end ass2_2bd;

architecture Behavioral of ass2_2bd is
begin
   with X select
       Y <= "00" when "0001",
             "01" when "0010",
             "10" when "0100",
            "11" when "1000",
             "ZZ" when others;
end Behavioral;
```

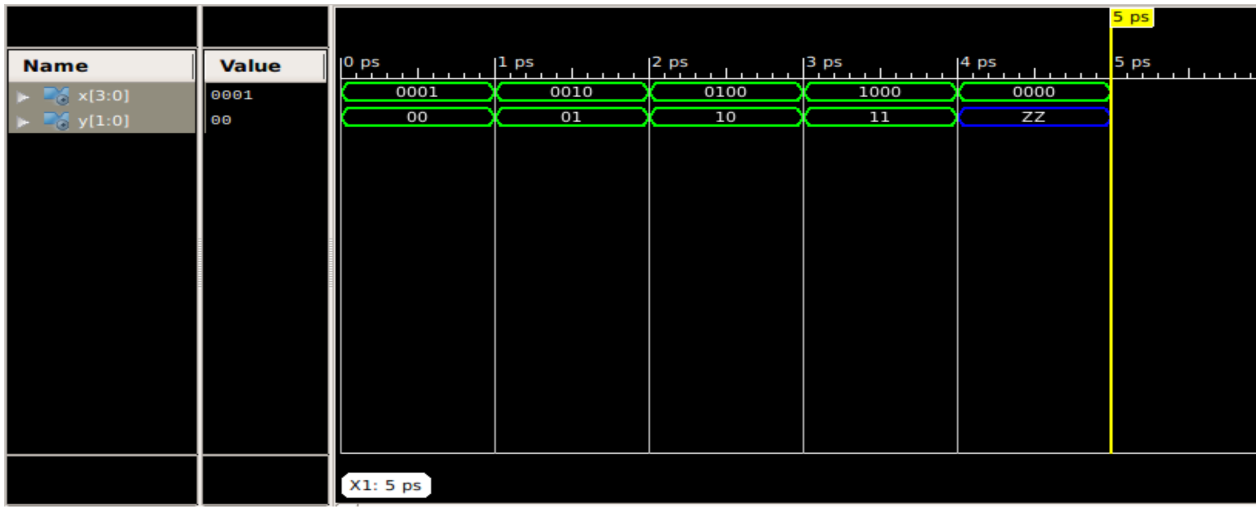
Test Bench

```
ARCHITECTURE behavior OF ass2_2a_test_bench IS
   COMPONENT ass2_2a
   PORT(
        X : IN  std_logic_vector(3 downto 0);
        Y : OUT  std_logic_vector(1 downto 0)
       );
   END COMPONENT;

  signal X : std_logic_vector(3 downto 0) := (others => '0');
  signal Y : std_logic_vector(1 downto 0);

BEGIN

  uut: ass2_2a PORT MAP (
         X => X,
         Y => Y
       );

  -- Stimulus process
  stim_proc: process
  begin
     X <= "0001";
     wait for 1 ps;
     X <= "0010";
     wait for 1 ps;
```

```
            X <= "0100";
            wait for 1 ps;
            X <= "1000";
            wait for 1 ps;
         end process;

      END;
```

Note: Test Bench would be same for all the different implementations of encoders mentioned here with only components name changed

Timing Diagram

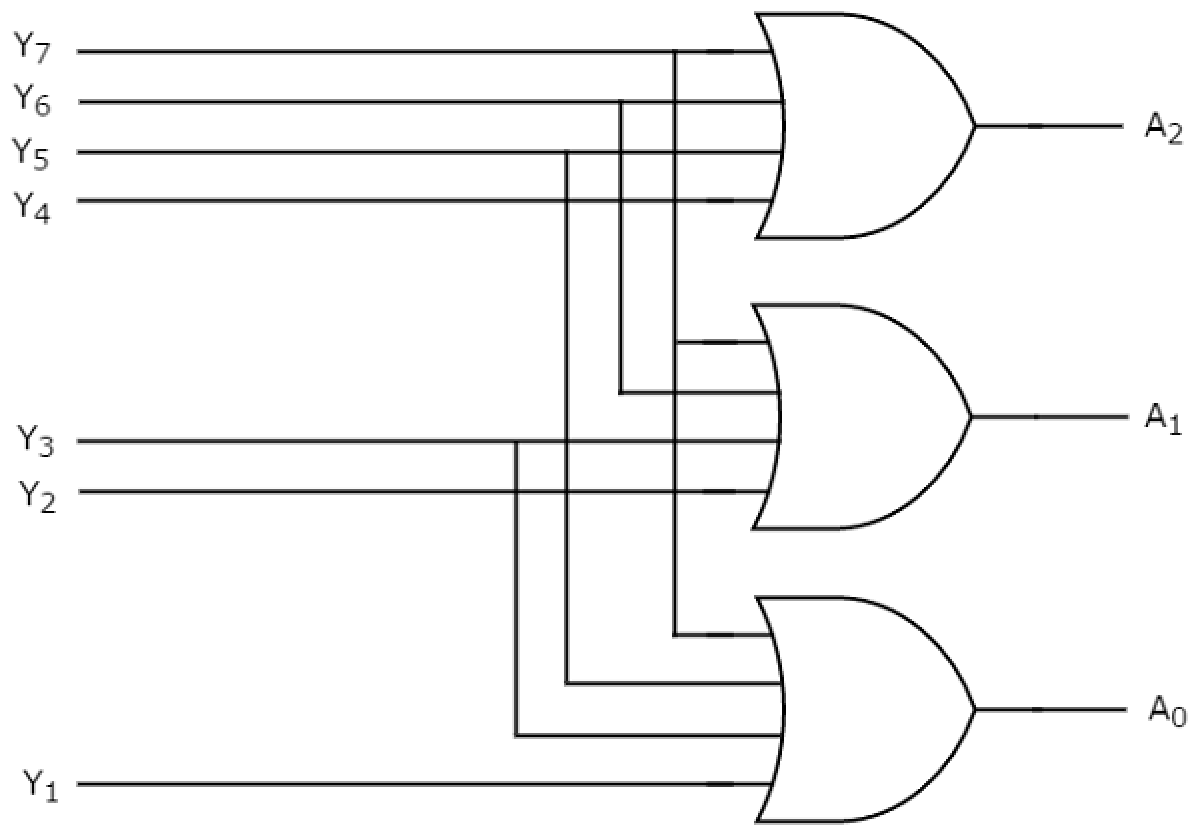


8x3 Encoders

Truth Table

X(7)	X(6)	X(5)	X(4)	X(3)	X(2)	X(1)	X(0)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	0	z	z	z
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Circuit Diagram



Code

3.1. Using gate-level modelling

```
entity ass2_3a is
   Port ( X : in  STD_LOGIC_VECTOR (7 downto 0);
          Y : out  STD_LOGIC_VECTOR (2 downto 0));
end ass2_3a;

architecture Behavioral of ass2_3a is
begin
   Y(2) <= X(4) or X(5) or X(6) or X(7);
   Y(1) <= X(2) or X(3) or X(6) or X(7);
   Y(0) <= X(1) or X(3) or X(5) or X(7);

end Behavioral;
```

3.2. Using behavioural modelling

```
entity ass2_3b is
   Port ( X : in  STD_LOGIC_VECTOR (7 downto 0);
          Y : out  STD_LOGIC_VECTOR (2 downto 0));
end ass2_3b;

architecture Behavioral of ass2_3b is
begin
   with X select
       Y <= "000" when "00000001",
            "001" when "00000010",
             "010" when "00000100",
             "011" when "00001000",
             "100" when "00010000",
             "101" when "00100000",
             "110" when "01000000",
             "111" when "10000000",
             "ZZZ" when others;
end Behavioral;
```

4. Using 4x2 and 2x1 encoders using component instantiation

```
architecture Behavioral of ass2_4 is
    component ass2_2ba is
        PORT( X: IN STD_LOGIC_VECTOR(3 downto 0);
                Y: OUT STD_LOGIC_VECTOR(1 downto 0));
   end component;
   component encoder2x1 is
        PORT( X: IN STD_LOGIC_VECTOR(1 downto 0);
                Y: OUT STD_LOGIC);
   end component;

   signal a,b,p: STD_LOGIC_VECTOR(1 downto 0);
   signal q: STD_LOGIC;

begin
c1: ass2_2ba port map(X(3 downto 0), a);
c2: ass2_2ba port map(X(7 downto 4), b);
c3: encoder2x1 port map(p, q);

p(0) <= X(0) or X(1) or X(2) or X(3);
p(1) <= X(4) or X(5) or X(6) or X(7);
```

```
p1: process(X, p, q, a, b)
begin
   if X(7 downto 4) = "0000" then
       Y <= q & a;
   elsif X(3 downto 0) = "0000" then
       Y <= q & b;
   else
       Y <= "ZZZ";
   end if;
end process;

end Behavioral;
```

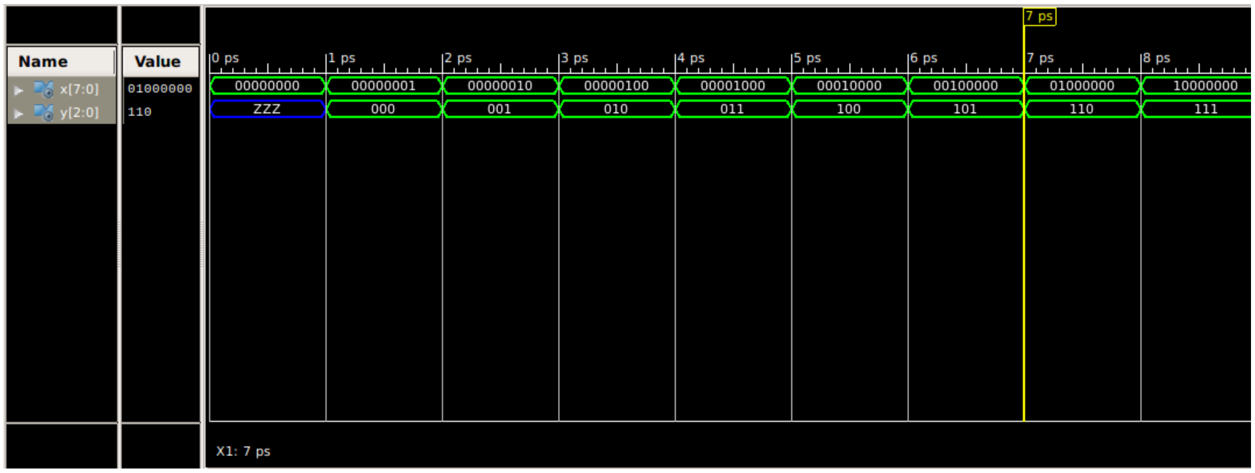
Test Bench

```
ARCHITECTURE behavior OF ass2_3b_test_bench IS
   COMPONENT ass2_3b
   PORT(
        X : IN  std_logic_vector(7 downto 0);
        Y : OUT  std_logic_vector(2 downto 0)
       );
   END COMPONENT;
  signal X : std_logic_vector(7 downto 0) := (others => '0');
  signal Y : std_logic_vector(2 downto 0);
BEGIN
  uut: ass2_3b PORT MAP (
         X => X,
         Y => Y
       );

  -- Stimulus process
  stim_proc: process
  begin
     X <= "00000000";
     wait for 1 ps;
     X <= "00000001";
     wait for 1 ps;
     X <= "00000010";
     wait for 1 ps;
     X <= "00000100";
     wait for 1 ps;
     X <= "00001000";
     wait for 1 ps;
```

```
            X <= "00010000";
            wait for 1 ps;
            X <= "00100000";
            wait for 1 ps;
            X <= "01000000";
            wait for 1 ps;
            X <= "10000000";
            wait for 1 ps;
        end process;

    END;
```

Note: Test Bench would be same for all the different implementations of encoders mentioned here with only components name changed

Timing Diagram



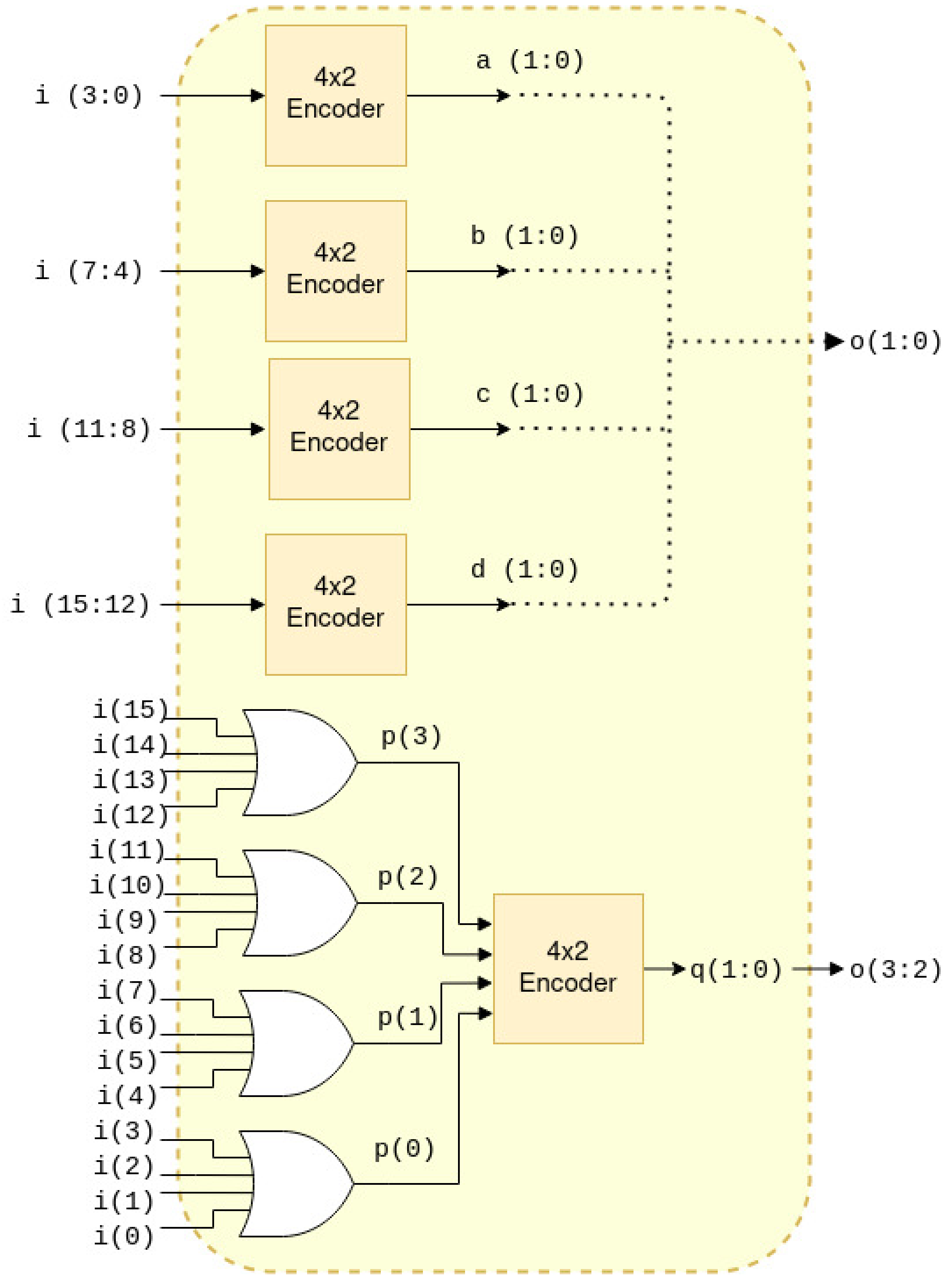
16x4 Encoders

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

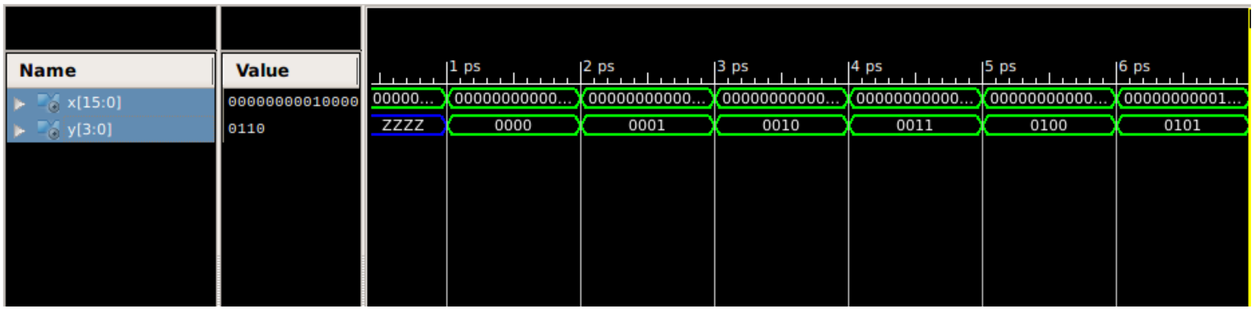
5. Using 4x2 encoders

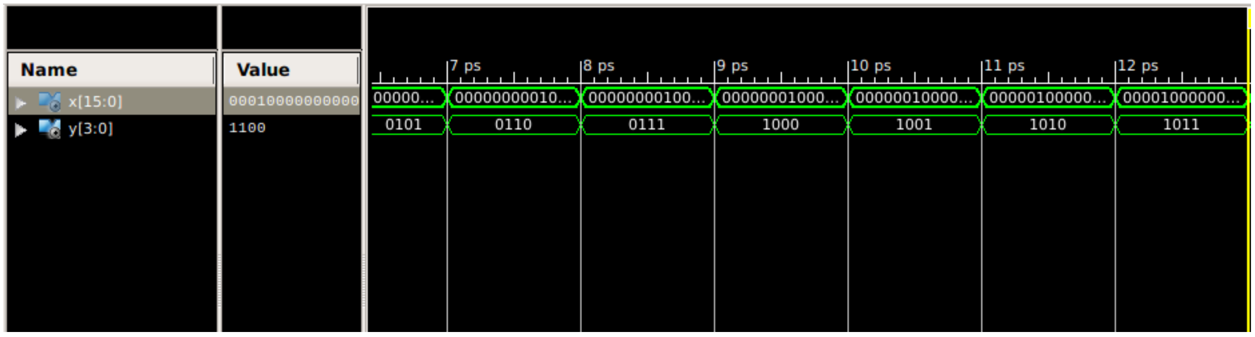
```
entity ass2_5 is
   Port ( X : in  STD_LOGIC_VECTOR (15 downto 0);
          Y : out  STD_LOGIC_VECTOR (3 downto 0));
end ass2_5;

architecture Behavioral of ass2_5 is
component ass2_2ba is
   port( X: in STD_LOGIC_VECTOR (3 downto 0);
           Y: out STD_LOGIC_VECTOR(1 downto 0));
end component;
signal a,b,c,d,q: STD_LOGIC_VECTOR(1 downto 0);
signal p: STD_LOGIC_VECTOR(3 downto 0);

begin
   p(3) <= X(15) or X(14) or X(13) or X(12);
   p(2) <= X(11) or X(10) or X(9) or X(8);
   p(1) <= X(7) or X(6) or X(5) or X(4);
   p(0) <= X(3) or X(2) or X(1) or X(0);

   c1: ass2_2ba port map(X(15 downto 12), a);
   c2: ass2_2ba port map(X(11 downto 8), b);
   c3: ass2_2ba port map(X(7 downto 4), c);
   c4: ass2_2ba port map(X(3 downto 0), d);
   c5: ass2_2ba port map(p, q);

   p1: process(X, a, b, c, d, q, p)
   begin
       if X(11 downto 0) = "000000000000" then
           Y <= q & a;
       elsif X(15 downto 12) = "0000" and X(7 downto 0) = "00000000" then
           Y <= q & b;
       elsif X(15 downto 8) = "00000000" and X(3 downto 0) = "0000" then
           Y <= q & c;
       elsif X(15 downto 4) = "000000000000" then
           Y <= q & d;
       else
           Y <= "ZZZZ";
       end if;
   end process;
end Behavioral;
```

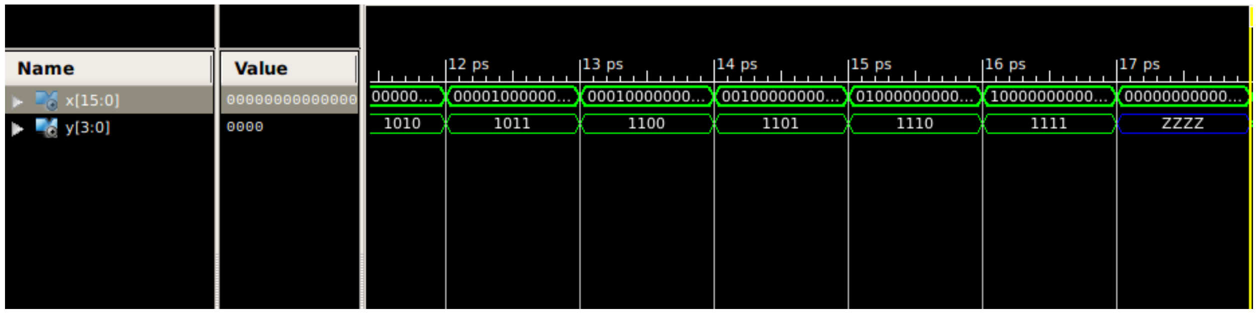
Test Bench

```
ARCHITECTURE behavior OF ass2_5_test_bench IS
    COMPONENT ass2_5
    PORT(
         X : IN  std_logic_vector(15 downto 0);
         Y : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;

  signal X : std_logic_vector(15 downto 0) := (others => '0');
  signal Y : std_logic_vector(3 downto 0);
BEGIN
  uut: ass2_5 PORT MAP (
          X => X,
          Y => Y
        );
  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]
00000000010000
0110
1 ps
2 ps
3 ps
4 ps
5 ps
6 ps
00000...
00000000000...
00000000001...
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...
00000000010...
00000000100...
00000001000...
00000010000...
00000100000...
00001000000...
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...
00001000000...
00010000000...
00100000000...
01000000000...
10000000000...
00000000000...
1010
1011
1100
1101
1110
1111
ZZZZ