VLSI Lab Report Assignment 2

BCSE 4th Year 2nd Semester

Name: ***Priti Shaw***
Roll No. : ***001710501076***
Batch: ***A3***
Submission Date: ***05th April 2021***

1. Design 2 X 1 Encoder using behavioral level Modelling

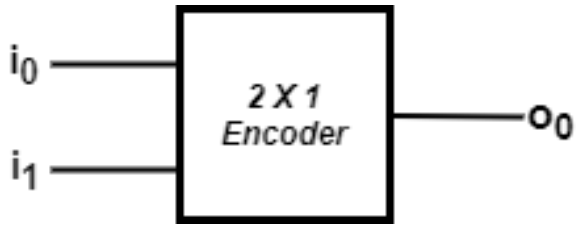
Description

An encoder in digital electronics is a one-hot to binary converter. That is if there are 2^n 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 2-to-1 encoder has 2 input lines and 1 output line.

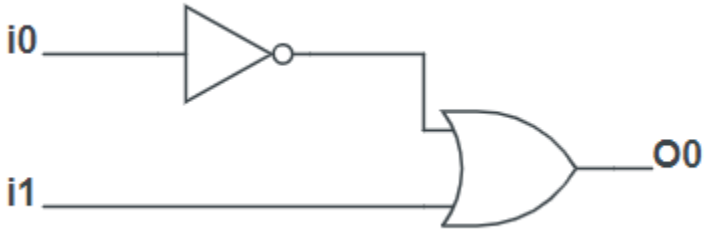
Truth Table

Input		Output
i_1	i_0	o_0
0	1	0
1	0	1

Block Diagram



Circuit Diagram



Architecture

a) Using ***if-else*** statement

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

entity QuestionOne is
    Port ( i : in  STD_LOGIC_VECTOR (1 downto 0);
           o : out  STD_LOGIC);
end QuestionOne;

architecture Behavioral of QuestionOne is

begin
p1:process(i)
            begin
            if i="01" then
                  o<='0';
            elsif i="10" then
                  o<='1';
            else
                  o<='Z';
            end if;
            end process;
end Behavioral;
```

b) Using ***case*** statement

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

entity QuestionOne_Part2 is
    Port ( i : in  STD_LOGIC_VECTOR (1 downto 0);
           o : out  STD_LOGIC);
end QuestionOne_Part2;

architecture Behavioral of QuestionOne_Part2 is

begin
p2:process(i)
```

```
      begin
      case i is
            when "01" => o<='0';
            when "10" => o<='1';
            when others => o<='Z';
      end case;
end process;
end Behavioral;
```

c) Using ***when else*** statement

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

entity QuestionOne_Part3 is
    Port ( i : in  STD_LOGIC_VECTOR (1 downto 0);
           o : out  STD_LOGIC);
end QuestionOne_Part3;

architecture Behavioral of QuestionOne_Part3 is

begin
o<= '0' when i<="01" else
    '1' when i<="10" else
    'Z';
end Behavioral;
```

d) Using ***select when*** statement

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

entity QuestionOne_Part4 is
    Port ( i : in  STD_LOGIC_VECTOR (1 downto 0);
           o : out  STD_LOGIC);
end QuestionOne_Part4;

architecture Behavioral of QuestionOne_Part4 is

begin
with i select
      o<='0' when "01",
```

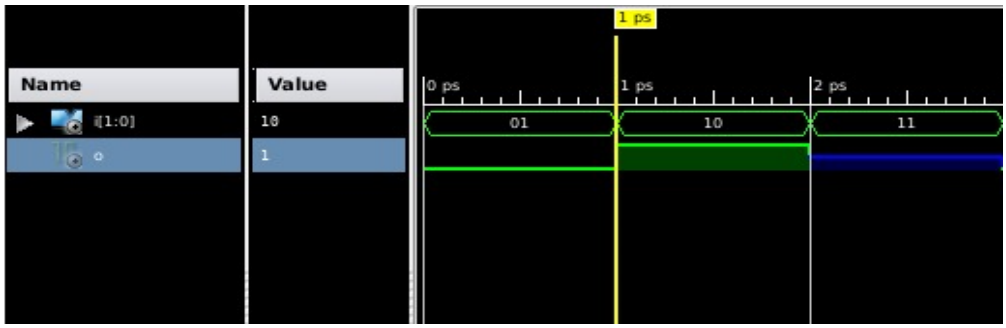
```
            '1' when "10",
            'Z' when others;
end Behavioral;
```

TestBench

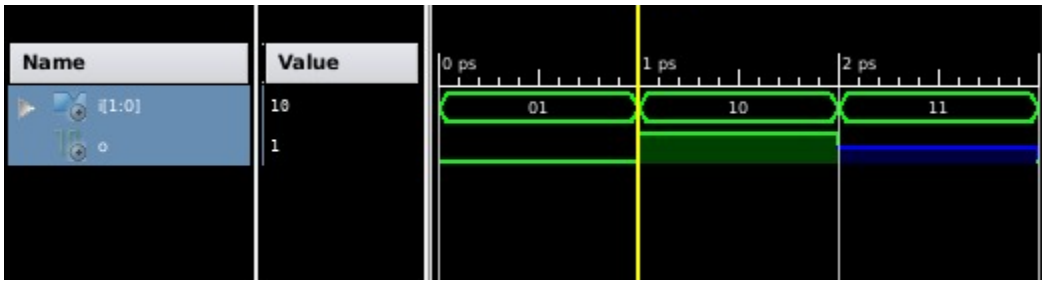
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY QuestionOne_TestBench IS
END QuestionOne_TestBench;

ARCHITECTURE behavior OF QuestionOne_TestBench IS

    COMPONENT QuestionOne
    PORT(
         i : IN  std_logic_vector(1 downto 0);
         o : OUT  std_logic
        );
    END COMPONENT;
   --Inputs
   signal i : std_logic_vector(1 downto 0) := (others => '0');
   --Outputs
   signal o : std_logic;

BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: QuestionOne PORT MAP (
          i => i,
          o => o
        );
   -- Stimulus process
   stim_proc: process
   begin
      i<="01";
            wait for 1 ps;
            i<="10";
      wait for 1 ps;
            i<="11";
            wait for 1 ps;
   end process;
END;
```

Timing Diagram

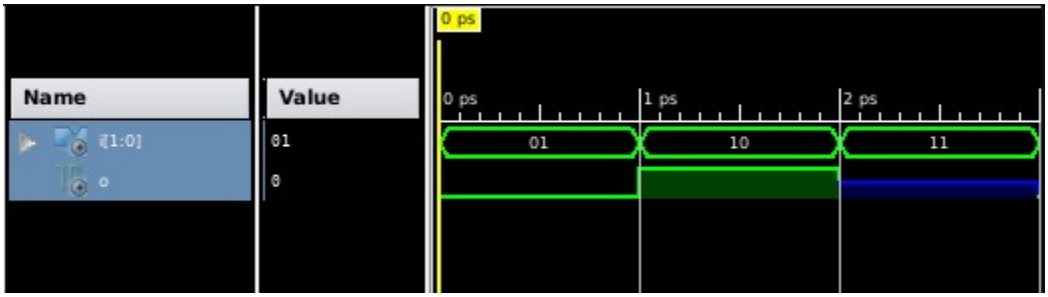
a) Using if-else statement



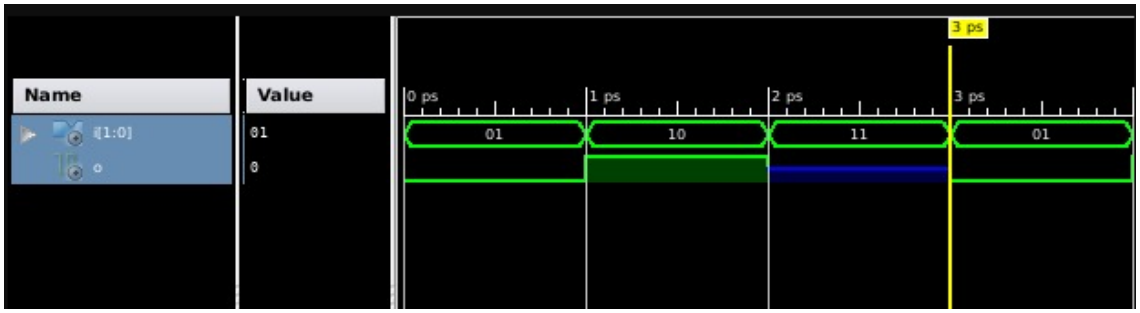
b) Using case statement



c) Using when else statement



d) Using select when statement

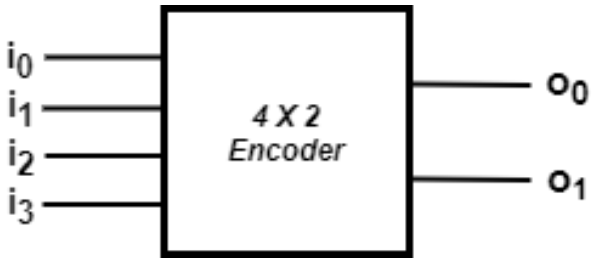


2. Design 4 X 2 Encoder

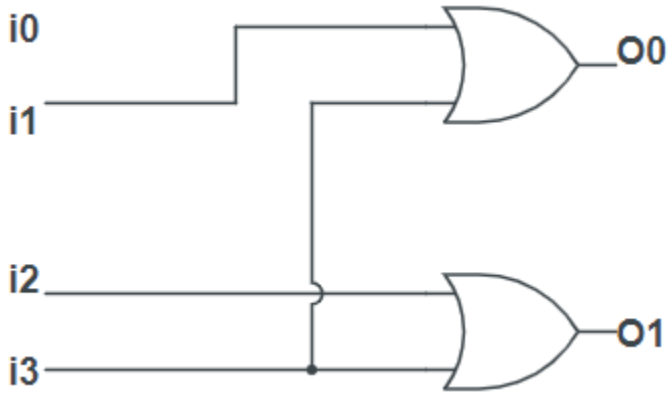
Description

An encoder in digital electronics is a one-hot to binary converter. That is if there are 2^n 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 4-to-2 encoder has 4 input lines and 2 output lines.

Block Diagram



Circuit Diagram



Truth Table

Input				Output	
i3	i2	i1	i0	o1	o0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Architecture

a) Using Gate level Modelling

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

entity QuestionTwo_Part1 is
    Port ( i : in  STD_LOGIC_VECTOR (3 downto 0);
           o : out  STD_LOGIC_VECTOR (1 downto 0));
end QuestionTwo_Part1;

architecture Behavioral of QuestionTwo_Part1 is

begin
      o(0)<=i(1) or i(3);
      o(1)<=i(2) or i(3);
end Behavioral;
```

b) Using behavioral Modelling

i) if-else statement

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

entity QuestionTwo_Part2 is
    Port ( ii : in  STD_LOGIC_VECTOR (3 downto 0);
           oo : out  STD_LOGIC_VECTOR (1 downto 0));
end QuestionTwo_Part2;
architecture Behavioral of QuestionTwo_Part2 is

begin
p1:process(ii)
            begin
            if ii="0001" then
                  oo<="00";
            elsif ii="0010" then
                  oo<="01";
            elsif ii="0100" then
                  oo<="10";
            elsif ii="1000" then
                  oo<="11";
            else
                  oo<="ZZ";
            end if;
            end process;
end Behavioral;
```

ii) case statement

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

entity QuestionTwo_Part2_B is
    Port ( ii : in  STD_LOGIC_VECTOR (3 downto 0);
           oo : out  STD_LOGIC_VECTOR (1 downto 0));
end QuestionTwo_Part2_B;

architecture Behavioral of QuestionTwo_Part2_B is

begin
```

```
p2:process(ii)
      begin
      case ii is
            when "0001" => oo<="00";
            when "0010" => oo<="01";
            when "0010" => oo<="10";
            when "1000" => oo<="11";
            when others => oo<="ZZ";
      end case;
end process;
end Behavioral;
```

iii) when else statement

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

entity QuestionTwo_Part2_C is
    Port ( ii : in  STD_LOGIC_VECTOR (3 downto 0);
           oo : out  STD_LOGIC_VECTOR (1 downto 0));
end QuestionTwo_Part2_C;

architecture Behavioral of QuestionTwo_Part2_C is

begin
oo<="00" when ii<="0001" else
       "01" when ii<="0010" else
       "10" when ii<="0100" else
       "11" when ii<="1000" else
       "ZZ";
end Behavioral;
```

iv) select when statement

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

entity QuestionTwo_Part2_D is
    Port ( ii : in  STD_LOGIC_VECTOR (3 downto 0);
           oo : out  STD_LOGIC_VECTOR (1 downto 0));
end QuestionTwo_Part2_D;
```

```
architecture Behavioral of QuestionTwo_Part2_D is

begin
with ii select
      oo<="00" when "0001",
             "01" when "0010",
             "10" when "0100",
             "11" when "1000",
             "ZZ" when others;
end Behavioral;
```

Test Bench

a) Using Gate level Modelling

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY QuestionTwo_Part1_TestBench IS
END QuestionTwo_Part1_TestBench;

ARCHITECTURE behavior OF QuestionTwo_Part1_TestBench IS
    COMPONENT QuestionTwo_Part1
    PORT(
         i : IN  std_logic_vector(3 downto 0);
         o : OUT  std_logic_vector(1 downto 0)
        );
    END COMPONENT;
   --Inputs
   signal i : std_logic_vector(3 downto 0) := (others => '0');
    --Outputs
   signal o : std_logic_vector(1 downto 0);

BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: QuestionTwo_Part1 PORT MAP (
          i => i,
          o => o
        );
   -- Stimulus process
```

```
   stim_proc: process
   begin
            i<="0001";
            wait for 1 ps;
            i<="0010";
            wait for 1 ps;
            i<="0100";
            wait for 1 ps;
            i<="1000";
            wait for 1 ps;
   end process;
END;
```

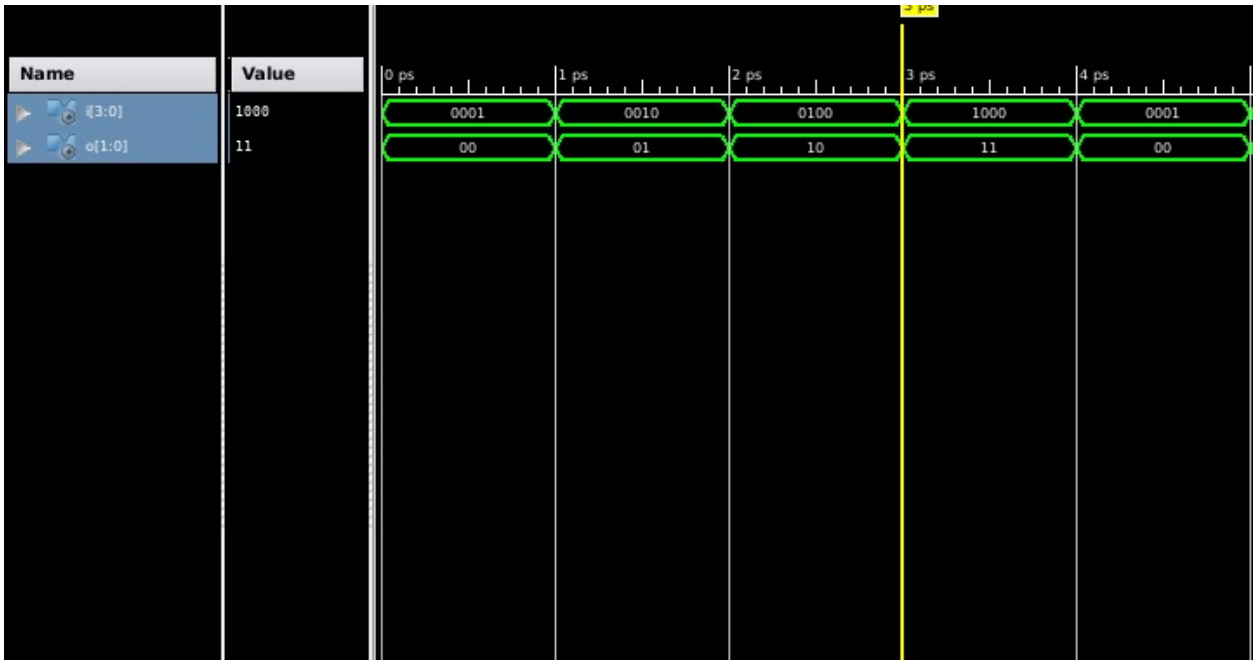
b) Using behavioral Modelling - if-else statement

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY QuestionTwo_Part2_TestBench IS
END QuestionTwo_Part2_TestBench;

ARCHITECTURE behavior OF QuestionTwo_Part2_TestBench IS

    COMPONENT QuestionTwo_Part2
    PORT(
         ii : IN  std_logic_vector(3 downto 0);
         oo : OUT  std_logic_vector(1 downto 0)
        );
    END COMPONENT;
   --Inputs
   signal ii : std_logic_vector(3 downto 0) := (others => '0');
   --Outputs
   signal oo : std_logic_vector(1 downto 0);

BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: QuestionTwo_Part2 PORT MAP (
          ii => ii,
          oo => oo
        );
```

```
   -- Stimulus process
   stim_proc: process
   begin
            ii<="0001";
            wait for 1 ps;
            ii<="0010";
            wait for 1 ps;
            ii<="0100";
            wait for 1 ps;
            ii<="1000";
            wait for 1 ps;
            ii<="1100";
            wait for 1 ps;
   end process;
END;
```

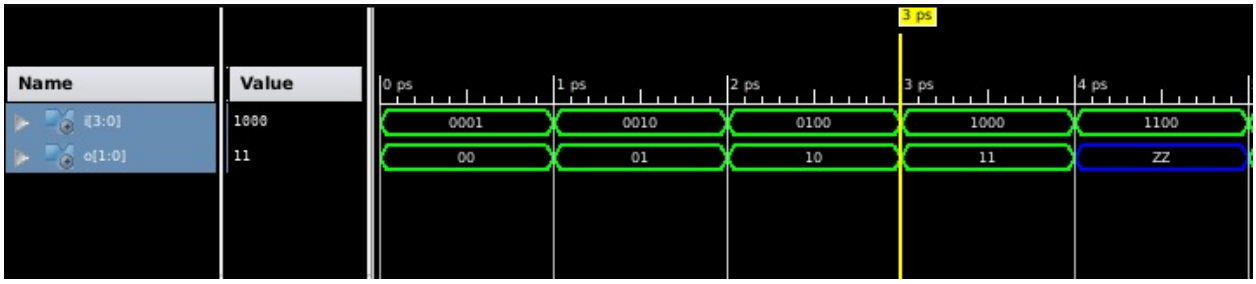
Timing Diagram

a) Gate level Modelling

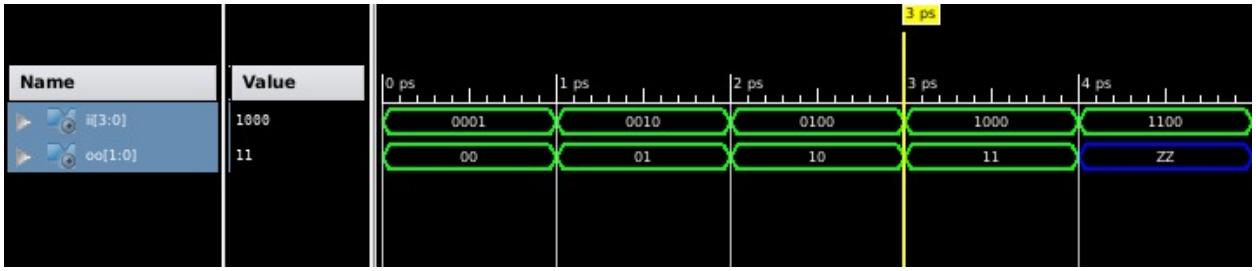


b) Behavioral Level Modelling

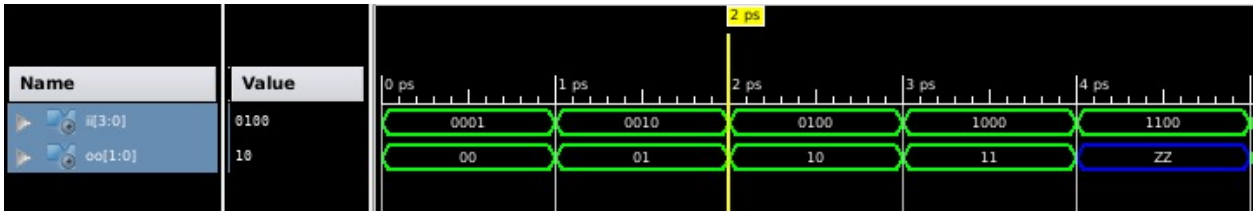
i) if-else statement



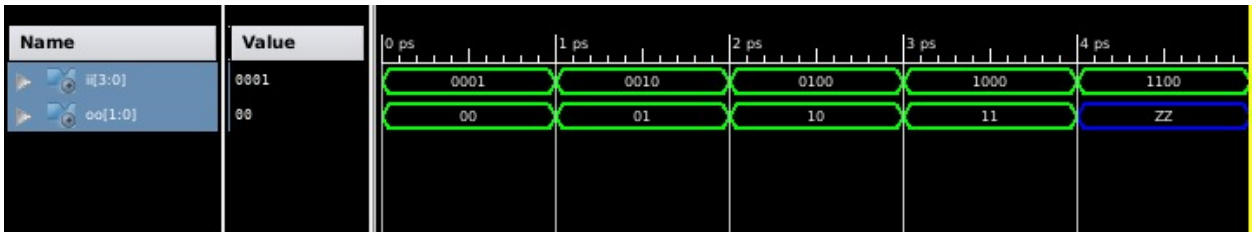
ii) Case statement



iii) When else statement



iv) Switch when statement

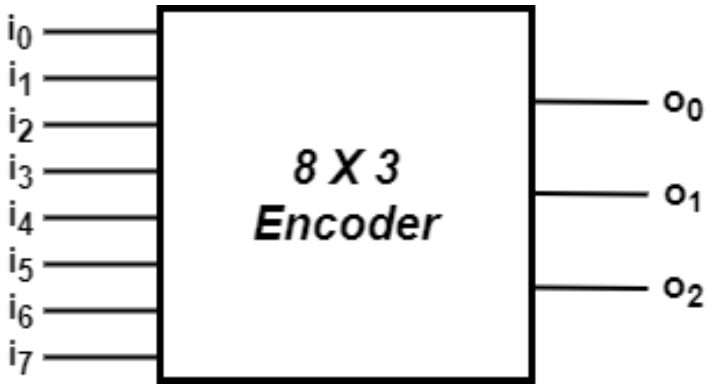


3. Design 8 X 3 Encoder.

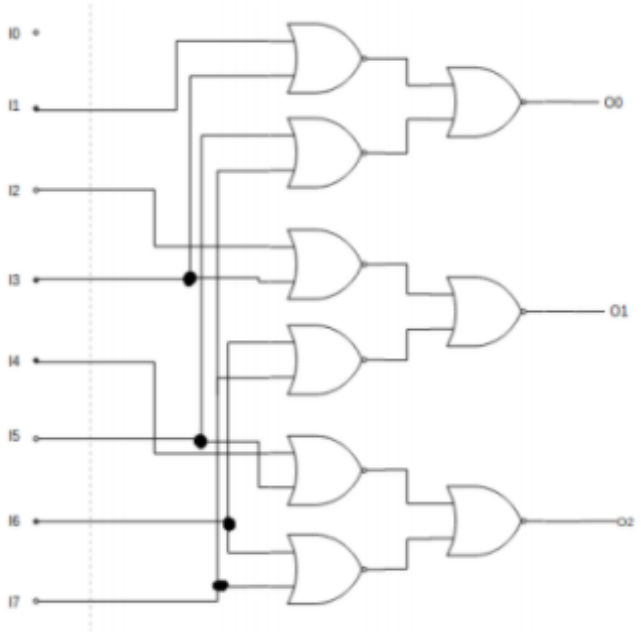
Description

An encoder in digital electronics is a one-hot to binary converter. That is if there are 2^n 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. An 8-to-3 encoder has 8 input lines and 3 output lines.

Block Diagram



Circuit Diagram



Truth Table

Input								Output		
i7	i6	i5	i4	i3	i2	i1	i0	o2	o1	o0
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

Architecture

a) Using Gate level Modelling

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

entity QuestionThree_Part1 is
    Port ( i : in  STD_LOGIC_VECTOR (7 downto 0);
           o : out  STD_LOGIC_VECTOR (2 downto 0));
end QuestionThree_Part1;

architecture Behavioral of QuestionThree_Part1 is

begin
o(2)<=i(7) or i(6) or i(5) or i(4);
o(1)<=i(7) or i(6) or i(3) or i(2);
o(0)<=i(7) or i(5) or i(3) or i(1);
end Behavioral;
```

b) Using Behavioral Modelling - if-else statement

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

entity QuestionThree_Part2 is
    Port ( i : in  STD_LOGIC_VECTOR (7 downto 0);
           o : out  STD_LOGIC_VECTOR (2 downto 0));
end QuestionThree_Part2;

architecture Behavioral of QuestionThree_Part2 is

begin
p1:process(i)
            begin
            if i="00000001" then
                  o<="000";
            elsif i="00000010" then
                  o<="001";
            elsif i="00000100" then
                  o<="010";
            elsif i="00001000" then
                  o<="011";
            elsif i="00010000" then
                  o<="100";
            elsif i="00100000" then
                  o<="101";
            elsif i="01000000" then
                  o<="110";
            elsif i="10000000" then
                  o<="111";
            else
                  o<="ZZZ";
            end if;
            end process;
end Behavioral;
```

Test Bench

a) Using Gate level Modelling

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY QuestionThree_Part1_TestBench IS
END QuestionThree_Part1_TestBench;

ARCHITECTURE behavior OF QuestionThree_Part1_TestBench IS
    COMPONENT QuestionThree_Part1
    PORT(
         i : IN  std_logic_vector(7 downto 0);
         o : OUT  std_logic_vector(2 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal i : std_logic_vector(7 downto 0) := (others => '0');
   --Outputs
   signal o : std_logic_vector(2 downto 0);

BEGIN
   uut: QuestionThree_Part1 PORT MAP (
          i => i,
          o => o
        );
   stim_proc: process
   begin
            i<="00000001";
            wait for 1 ps;
            i<="00000010";
            wait for 1 ps;
            i<="00000100";
            wait for 1 ps;
            i<="00001000";
            wait for 1 ps;
            i<="00010000";
            wait for 1 ps;
            i<="00100000";
```

```
            wait for 1 ps;
            i<="01000000";
            wait for 1 ps;
            i<="10000000";
            wait for 1 ps;
   end process;
END;
```

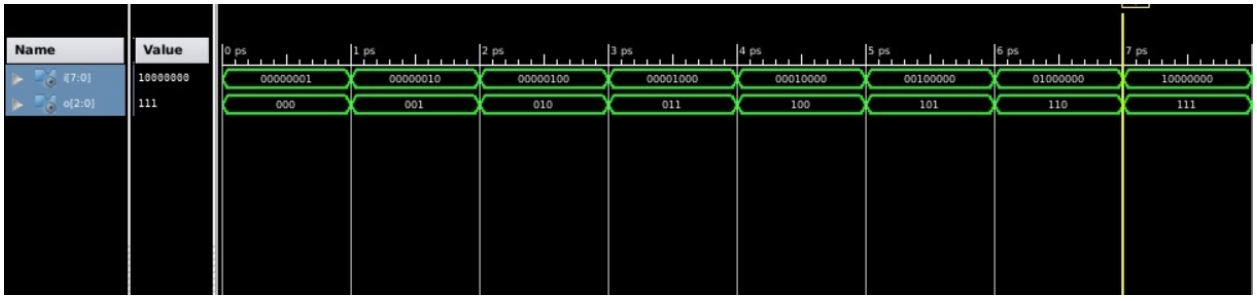
b) Using behavioral Modelling - if-else statement

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY QuestionThree_Part2_TestBench IS
END QuestionThree_Part2_TestBench;

ARCHITECTURE behavior OF QuestionThree_Part2_TestBench IS

    COMPONENT QuestionThree_Part2
    PORT(
         i : IN  std_logic_vector(7 downto 0);
         o : OUT  std_logic_vector(2 downto 0)
        );
    END COMPONENT;
   --Inputs
   signal i : std_logic_vector(7 downto 0) := (others => '0');
   --Outputs
   signal o : std_logic_vector(2 downto 0);
BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: QuestionThree_Part2 PORT MAP (
          i => i,
          o => o
        );
   -- Stimulus process
   stim_proc: process
   begin
      i<="00000001";
            wait for 1 ps;
            i<="00000010";
            wait for 1 ps;
```

```
            i<="00000100";
            wait for 1 ps;
            i<="00001000";
            wait for 1 ps;
            i<="00010000";
            wait for 1 ps;
            i<="00100000";
            wait for 1 ps;
            i<="01000000";
            wait for 1 ps;
            i<="10000000";
            wait for 1 ps;
            i<="10100010";
            wait for 1 ps;
    end process;
END;
```

Timing Diagram

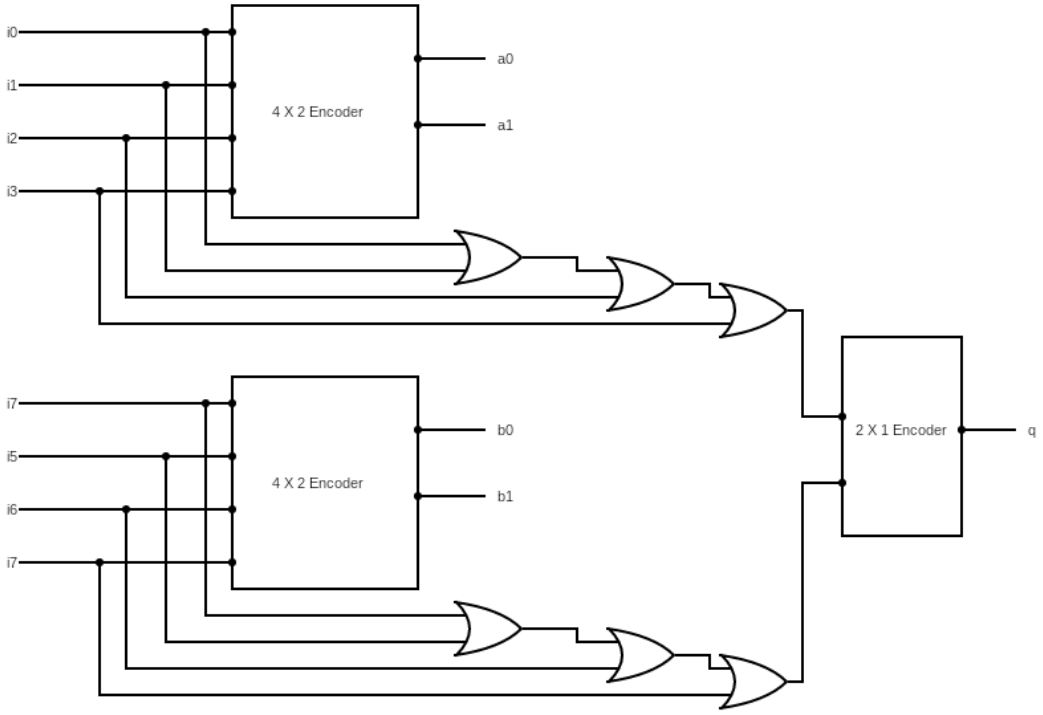
a) For Gate level Modelling



b) For behavioral Modelling - if-else statement

4. Design a 8 X 3 encoder using 4 X 2 encoder and 2 X 1 encoder.

Circuit Diagram



Input								Output		
i7	i6	i5	i4	i3	i2	i1	i0	o2	o1	o0
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

Architecture

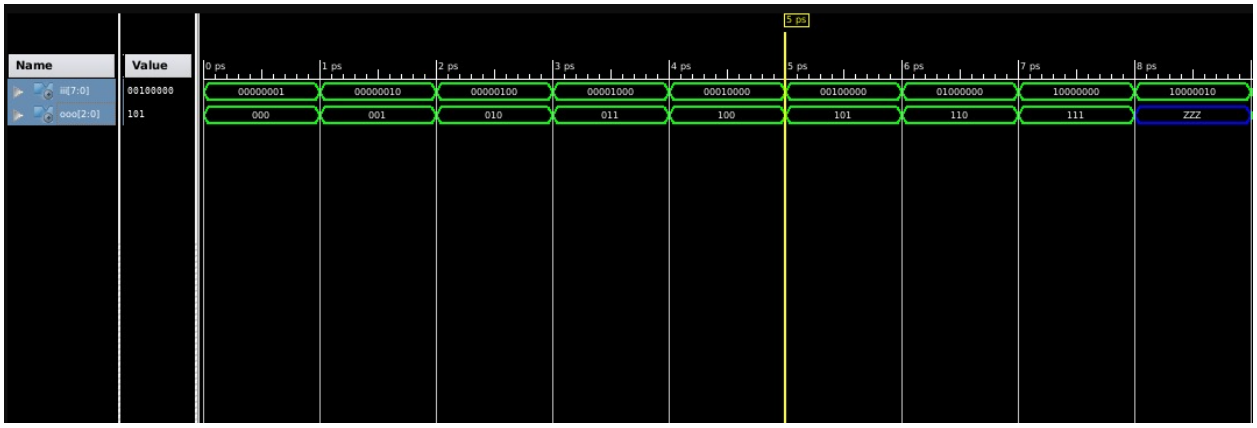
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity QuestionFour is
    Port ( iii : in  STD_LOGIC_VECTOR (7 downto 0);
           ooo : out  STD_LOGIC_VECTOR (2 downto 0));
end QuestionFour;

architecture Behavioral of QuestionFour is
component QuestionTwo_Part2 is
      Port ( ii : in STD_LOGIC_VECTOR (3 downto 0);
                  oo : out STD_LOGIC_VECTOR (1 downto 0));
end component;

component QuestionOne is
      Port ( i : in STD_LOGIC_VECTOR (1 downto 0);
                  o : out STD_LOGIC );
end component;

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

begin
c1: QuestionTwo_Part2 port map(iii(3 downto 0), a);
c2: QuestionTwo_Part2 port map(iii(7 downto 4), b);
c3: QuestionOne port map(p, q);
p(0)<= iii(0) or iii(1) or iii(2) or iii(3);
p(1)<= iii(4) or iii(5) or iii(6) or iii(7);
p1:process(iii,p,q,a,b)
      begin
      if iii(7 downto 4)="0000" then
            ooo<= q & a;
      elsif iii(3 downto 0)="0000" then
            ooo<= q & b;
      else
            ooo<= "ZZZ";
      end if;
      end process;
end Behavioral;
```

Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY QuestionFour_TestBench IS
END QuestionFour_TestBench;
ARCHITECTURE behavior OF QuestionFour_TestBench IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT QuestionFour
    PORT(
         iii : IN  std_logic_vector(7 downto 0);
         ooo : OUT  std_logic_vector(2 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal iii : std_logic_vector(7 downto 0) := (others => '0');

      --Outputs
   signal ooo : std_logic_vector(2 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: QuestionFour PORT MAP (
          iii => iii,
          ooo => ooo
        );

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

```
            iii<="00010000";
            wait for 1 ps;
            iii<="00100000";
            wait for 1 ps;
            iii<="01000000";
            wait for 1 ps;
            iii<="10000000";
            wait for 1 ps;
            iii<="10000010";
            wait for 1 ps;
    end process;
END;
```

Timing Diagram

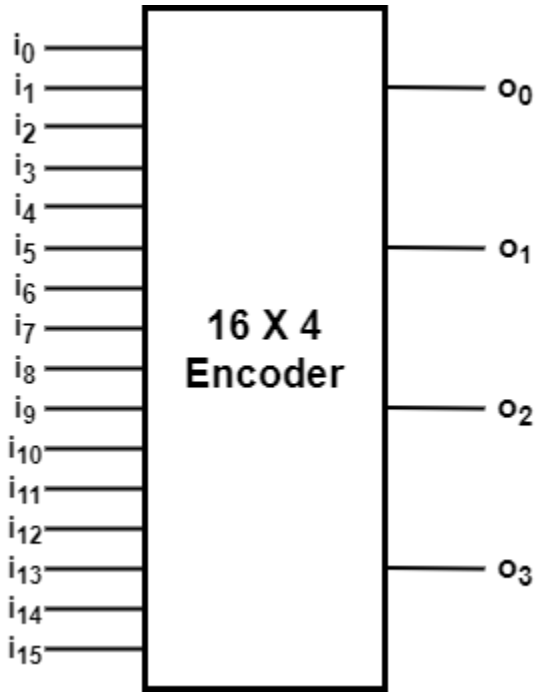


5. Design a 16 X 4 encoder using 4 X 2 encoders only.

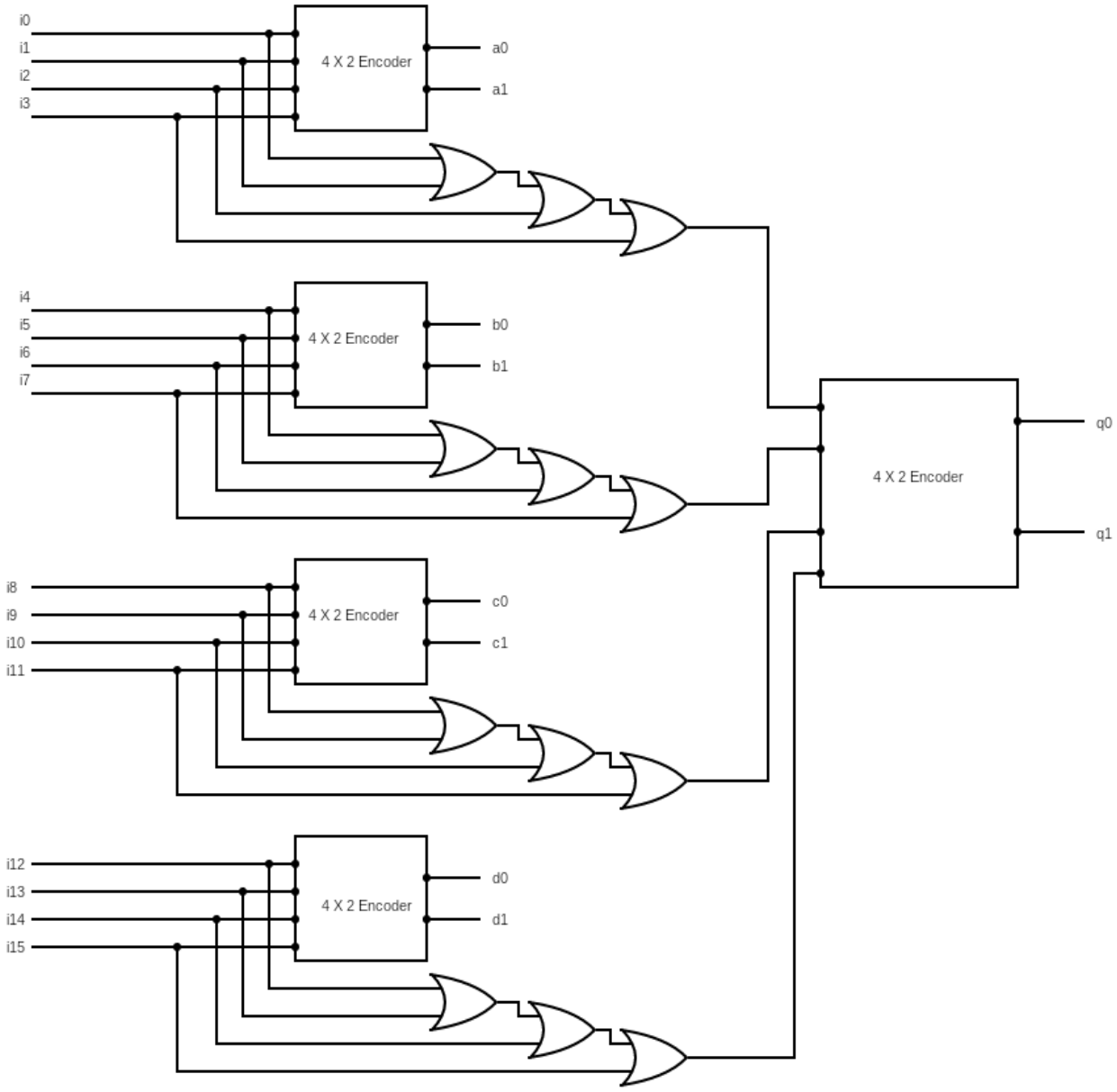
Description

An encoder in digital electronics is a one-hot to binary converter. That is if there are 2^n 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 lines.

Block Diagram



Circuit Diagram


i0
i1
i2
i3
4 X 2 Encoder
a0
a1
i4
i5
i6
i7
4 X 2 Encoder
b0
b1
i8
i9
i10
i11
4 X 2 Encoder
c0
c1
i12
i13
i14
i15
4 X 2 Encoder
d0
d1
4 X 2 Encoder
q0
q1

Truth Table

Input																Output			
i_{15}	i_{14}	i_{13}	i_{12}	i_{11}	i_{10}	i_9	i_8	i_7	i_6	i_5	i_4	i_3	i_2	i_1	i_0	o_3	o_2	o_1	o_0
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	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1
0	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	1	0	1
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1
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	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Architecture

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

entity QuestionFive is
    Port ( i : in  STD_LOGIC_VECTOR (15 downto 0);
           o : out  STD_LOGIC_VECTOR (3 downto 0));
end QuestionFive;

architecture Behavioral of QuestionFive is

component QuestionTwo_Part2 is
      Port ( ii : in STD_LOGIC_VECTOR (3 downto 0);
                  oo : out STD_LOGIC_VECTOR (1 downto 0));
end component;

signal p: STD_LOGIC_VECTOR(3 downto 0);
signal a,b,c,d,q: STD_LOGIC_VECTOR(1 downto 0);

begin
c1: QuestionTwo_Part2 port map(i(3 downto 0), a);
c2: QuestionTwo_Part2 port map(i(7 downto 4), b);
c3: QuestionTwo_Part2 port map(i(11 downto 8), c);
c4: QuestionTwo_Part2 port map(i(15 downto 12), d);
c5: QuestionTwo_Part2 port map(p, q);
p(0)<= i(3) or i(2) or i(1) or i(0);
p(1)<= i(7) or i(6) or i(5) or i(4);
p(2)<= i(11) or i(10) or i(9) or i(8);
p(3)<= i(15) or i(14) or i(13) or i(12);
p1:process(i,p,q,a,b,c,d)
      begin
      if i(7 downto 4)="0000" and i(11 downto 8)="0000" and i(15 downto
12)="0000" then
            o<= q & a;
      elsif i(3 downto 0)="0000" and i(11 downto 8)="0000" and i(15 downto
12)="0000" then
            o<= q & b;
      elsif i(3 downto 0)="0000" and i(7 downto 4)="0000" and i(15 downto
12)="0000" then
            o<= q & c;
      elsif i(3 downto 0)="0000" and i(7 downto 4)="0000" and i(11 downto
8)="0000" then
```

```
            o<= q & d;
      else
            o<="ZZZZ";
      end if;
      end process;
end Behavioral;
```

Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY QuestionFive_TestBench IS
END QuestionFive_TestBench;

ARCHITECTURE behavior OF QuestionFive_TestBench IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT QuestionFive
    PORT(
         i : IN  std_logic_vector(15 downto 0);
         o : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;
   --Inputs
   signal i : std_logic_vector(15 downto 0) := (others => '0');
   --Outputs
   signal o : std_logic_vector(3 downto 0);

BEGIN
      uut: QuestionFive PORT MAP (
          i => i,
          o => o
        );
   stim_proc: process
   begin
            i<="0000000000000001";
            wait for 1 ps;
            i<="0000000000000010";
            wait for 1 ps;
            i<="0000000000000100";
            wait for 1 ps;
```

```
            i<="0000000000001000";
            wait for 1 ps;
            i<="0000000000010000";
            wait for 1 ps;
            i<="0000000000100000";
            wait for 1 ps;
            i<="0000000001000000";
            wait for 1 ps;
            i<="0000000010000000";
            wait for 1 ps;
            i<="0000000100000000";
            wait for 1 ps;
            i<="0000001000000000";
            wait for 1 ps;
            i<="0000010000000000";
            wait for 1 ps;
            i<="0000100000000000";
            wait for 1 ps;
            i<="0001000000000000";
            wait for 1 ps;
            i<="0010000000000000";
            wait for 1 ps;
            i<="0100000000000000";
            wait for 1 ps;
            i<="1000000000000000";
            wait for 1 ps;
            i<="1000000010000010";
            wait for 1 ps;
        end process;
END;
```

Timing Diagram