VLSI Lab Report Assignment 3

BCSE 4th Year 2nd Semester

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

1. Design 1 X 2 Decoder

Description

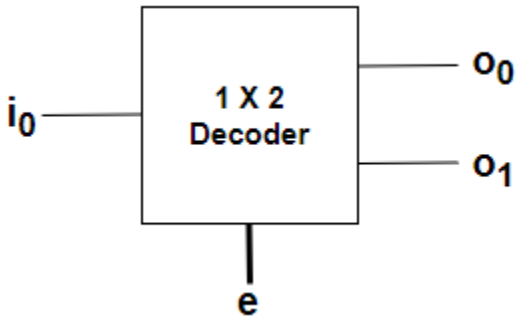
Design 1 X 2 decoder using gate and behavioral level modeling.

A decoder is a combinational circuit that has N input lines and a maximum of 2^N output lines. One of these outputs will be active high based on the combination of inputs present when the decoder is enabled. That means the decoder detects a particular code. A 1 to 2 decoder has 1 input line and 2 output lines. An enable input is provided to switch the decoder on and off.

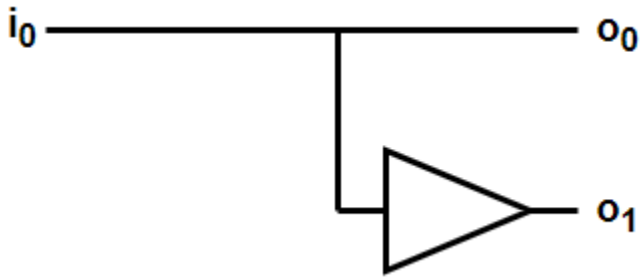
Truth Table

Input		Output	
e	**i_0**	**o_1**	**o_0**
0	X	X	X
1	0	0	1
1	1	1	0

Block Diagram



Circuit Diagram



Entity

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

Architecture

a) Using ***gate-level*** modeling

```
architecture Behavioral of QuestionOne_PartA is

begin
o(0) <= e and not(i);
o(1) <= e and i;
end Behavioral;
```

b) Using ***behavioral-level*** modeling

i. Using **if-else** statement

```
architecture Behavioral of QuestionOne_PartB_One is

begin
p1:process(e, i)
                  begin
                        if e='0' then
```

```
                              o<="00";
                        elsif i='0' then
                              o<="01";
                        elsif i='1' then
                              o<="10";
                        end if;
                  end process;
end Behavioral;
```

ii. Using **case** statement

```
architecture Behavioral of QuestionOne_PartB_two is

begin
p2:process(e,i)
      begin
            case e is
                  when '0' => o<="00";
                  when '1' =>
                  case i is
                        when '0' => o<="01";
                        when '1' => o<="10";
                        when others =>o<="ZZ";
                  end case;
                  when others =>o<="ZZ";
            end case;
      end process;
end Behavioral;
```

iii. Using **when else** statement

```
architecture Behavioral of QuestionOne_PartB_three is

begin
o<="00" when e<='0' else
      "01" when i<='0' else
      "10" when i<='1' else
      "ZZ";
end Behavioral;
```

iv. Using **switch** when statement

```
architecture Behavioral of QuestionOne_PartB_four is

begin
      with (e&i) select
            o <= "01" when "10",
                    "10" when "11",
                    "00" when "00",
                    "ZZ" when others;
end Behavioral;
```

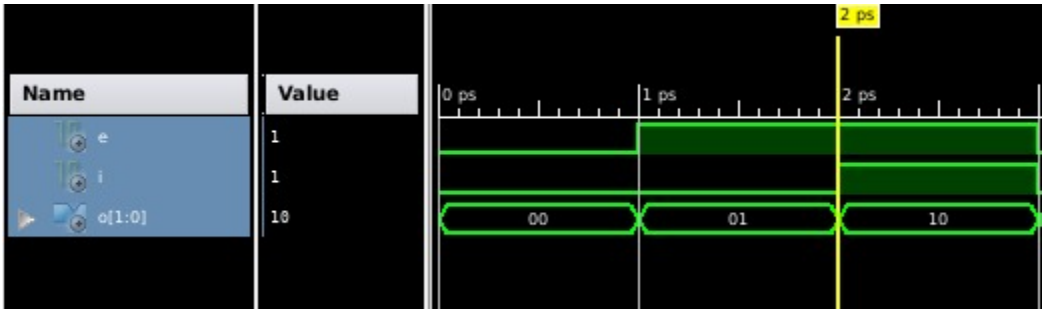
TestBench

```
ENTITY QuestionOne_Testbench IS
END QuestionOne_Testbench;

ARCHITECTURE behavior OF QuestionOne_Testbench IS
    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT QuestionOne
    PORT(
         e : IN  std_logic;
         i : IN  std_logic;
         o : OUT  std_logic_vector(1 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal e : std_logic := '0';
   signal i : std_logic := '0';
   --Outputs
   signal o : std_logic_vector(1 downto 0);

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

```
   begin
            e<='0';
            i<='0';
            wait for 1 ps;
            e<='1';
            i<='0';
            wait for 1 ps;
            i<='1';
            wait for 1 ps;
   end process;
END;
```

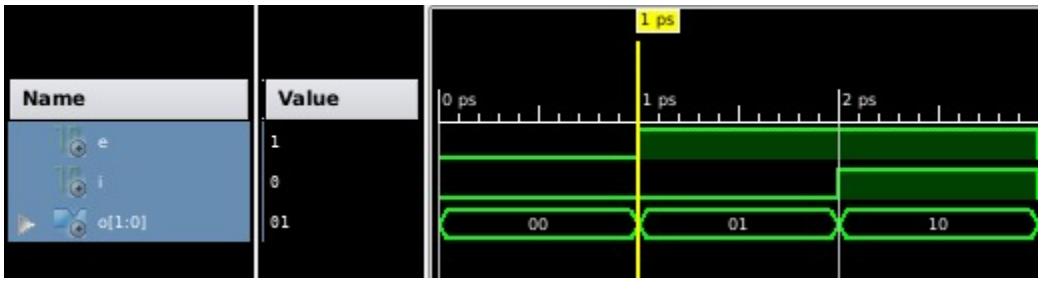
Timing Diagram

a) Using ***gate-level*** modeling



b) Using ***behavioral-level*** modeling

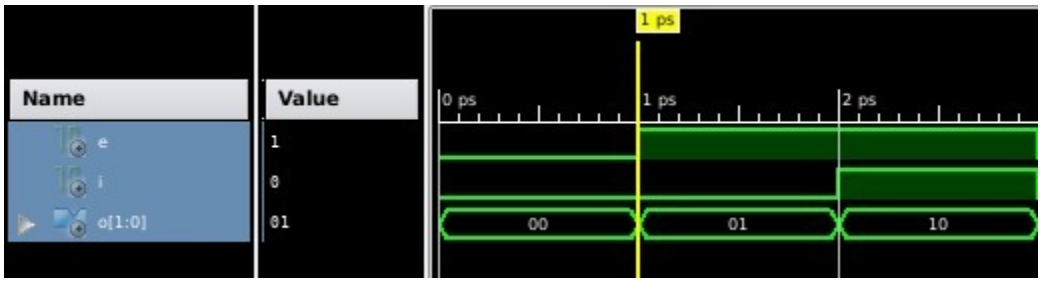
i) Using **if-else** statement



ii) Using **case** statement

iii) Using **when else** statement

iv) Using **select when** statement



2. Design 2 X 4 Decoder

Description

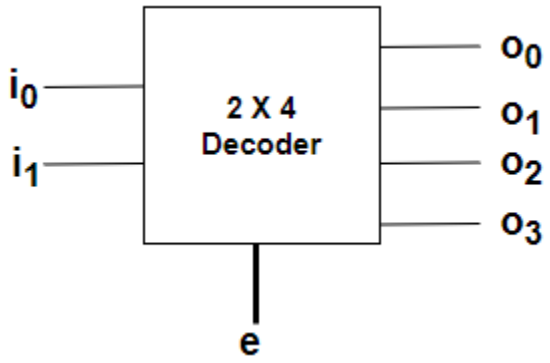
Design 2 X 4 decoder using gate and behavioral level modeling.

A decoder is a combinational circuit that has N input lines and a maximum of 2^N output lines. A 2 to 4 decoder has 2 input lines and 4 output lines. An enable input is provided to switch the decoder on and off.

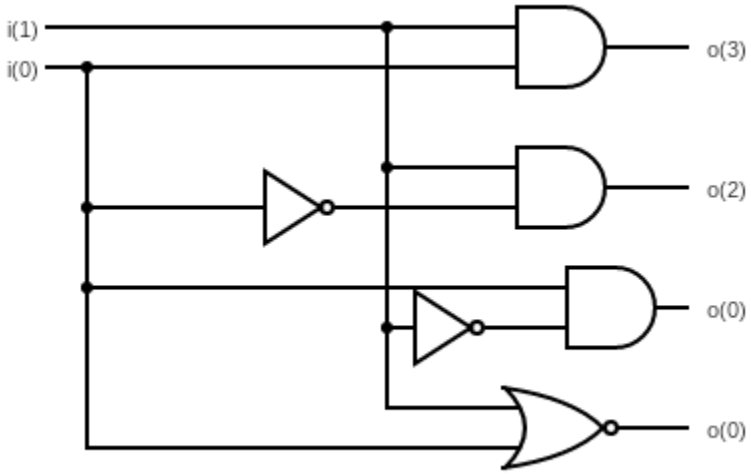
Truth Table

Input			Output			
e	**i_1**	**i_0**	**o_3**	**o_2**	**o_1**	**o_0**
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Block Diagram



Circuit Diagram



Entity

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

Architecture

a) Using ***gate-level*** modeling

```
architecture Behavioral of QuestionTwo_PartA is

```

```
begin
      oo(0)<= e and not(ii(1)) and not(ii(0));
      oo(1)<= e and not(ii(1)) and ii(0);
      oo(2)<= e and (ii(1)) and not(ii(0));
      oo(3)<= e and (ii(1)) and ii(0);
end Behavioral;
```

b) Using ***behavioral-level*** modeling

i. Using **if-else** statement

```
architecture Behavioral of QuestionTwo_PartB_one is

begin
p1:process(ee, ii)
                  begin
                        if ee='0' then
                              oo<="0000";
                        elsif ii="00" then
                              oo<="0001";
                        elsif ii="01" then
                              oo<="0010";
                        elsif ii="10" then
                              oo<="0100";
                        elsif ii="11" then
                              oo<="1000";
                        else
                              oo<="ZZZZ";
                        end if;
                  end process;
end Behavioral;
```

ii. Using **case** statement

```
architecture Behavioral of QuestionOne_PartB_two is

begin
p2:process(e,i)
      begin
            case e is
                  when '0' => o<="00";
                  when '1' =>
```

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

iii. Using **when else** statement

```
architecture Behavioral of QuestionTwo_PartB_three is

begin
oo<="0000" when e<='0' else
       "0001" when ii<="00" else
       "0010" when ii<="01" else
       "0100" when ii<="10" else
       "1000" when ii<="11" else
       "ZZZZ";
end Behavioral;
```

iv. Using **switch** when statement

```
architecture Behavioral of QuestionTwo_PartB_Four is

begin
with (e&ii) select
            oo <= "0001" when "100",
                        "0010" when "101",
                        "0100" when "110",
                        "1000" when "111",
                    "0000" when others;
end Behavioral;
```

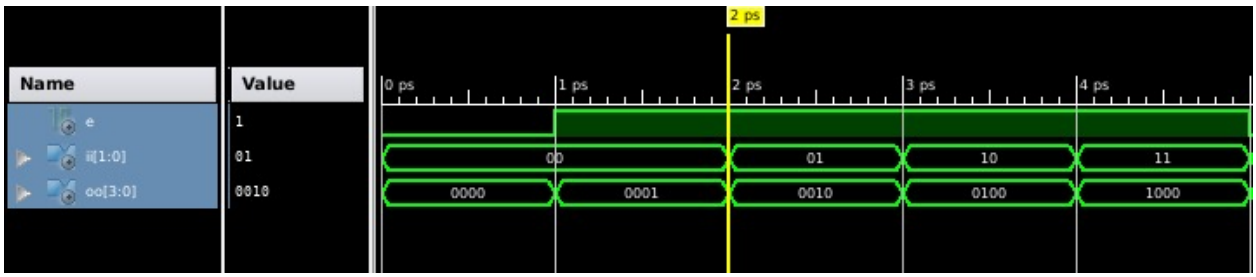
TestBench

```
ENTITY QuestionTwo_TestBench IS
END QuestionTwo_TestBench;
```

```
ARCHITECTURE behavior OF QuestionTwo_TestBench IS
    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT QuestionTwo_PartB_Four
    PORT(
         e : IN  std_logic;
         ii : IN  std_logic_vector(1 downto 0);
         oo : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal e : std_logic := '0';
   signal ii : std_logic_vector(1 downto 0) := (others => '0');

   --Outputs
   signal oo : std_logic_vector(3 downto 0);

BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: QuestionTwo_PartB_Four PORT MAP (
          e => e,
          ii => ii,
          oo => oo
        );
   -- Stimulus process
   stim_proc: process
   begin
      e<='0';
            ii<="00";
            wait for 1 ps;
            e<='1';
            ii<="00";
            wait for 1 ps;
            ii<="01";
            wait for 1 ps;
            ii<="10";
            wait for 1 ps;
            ii<="11";
            wait for 1 ps;
   end process;
END;
```

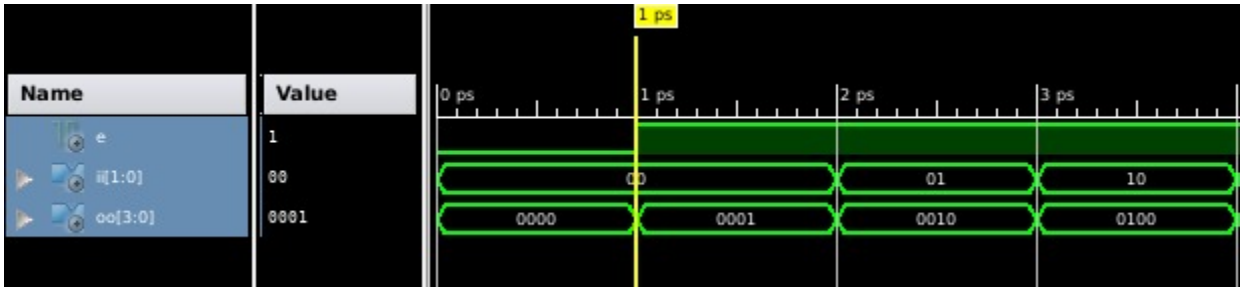
Timing Diagram

a) Using ***gate-level*** modeling

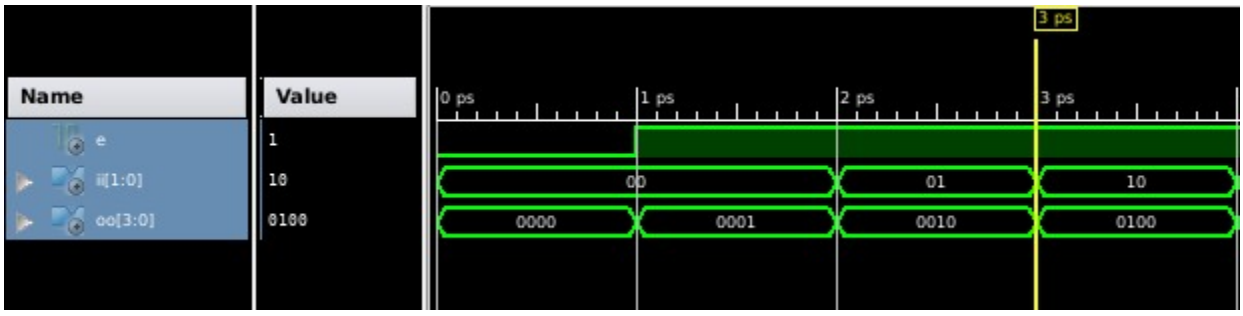


b) Using ***behavioral-level*** modeling

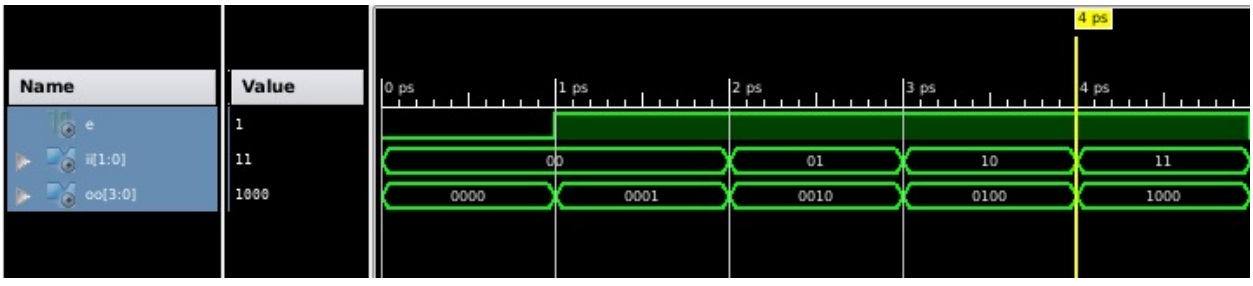
i) Using **if-else** statement



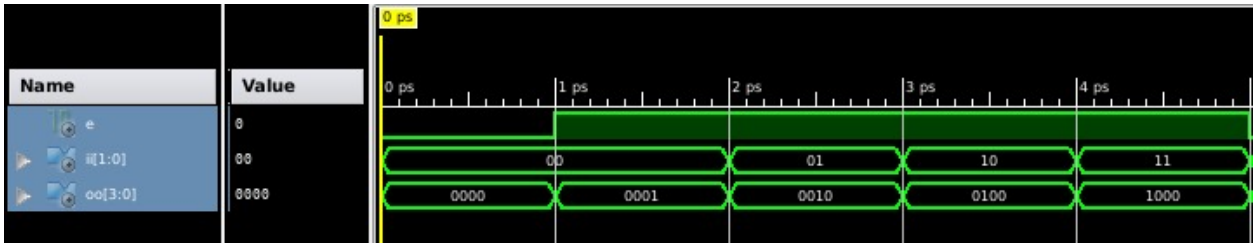
ii) Using **case** statement



iii) Using **when else** statement



iv) Using **select when** statement



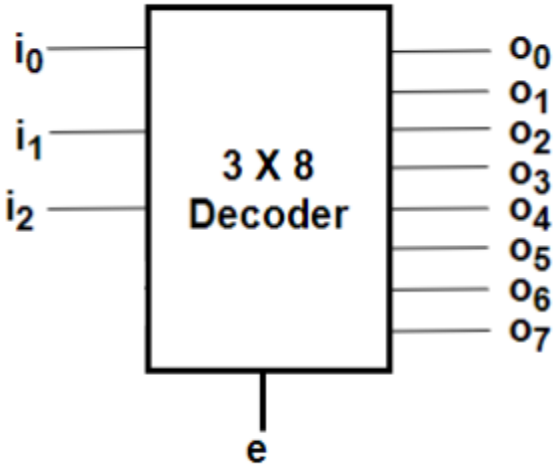
3. Design 3 X 8 Decoder

Description

Design 3 X 8 decoder using gate and behavioral level modeling.

A decoder is a combinational circuit that has N input lines and a maximum of 2^N output lines. One of these outputs will be active high based on the combination of inputs present when the decoder is enabled. That means the decoder detects a particular code. A 3 to 8 decoder has 3 input lines and 8 output lines. An enable input is provided to switch the decoder on and off.

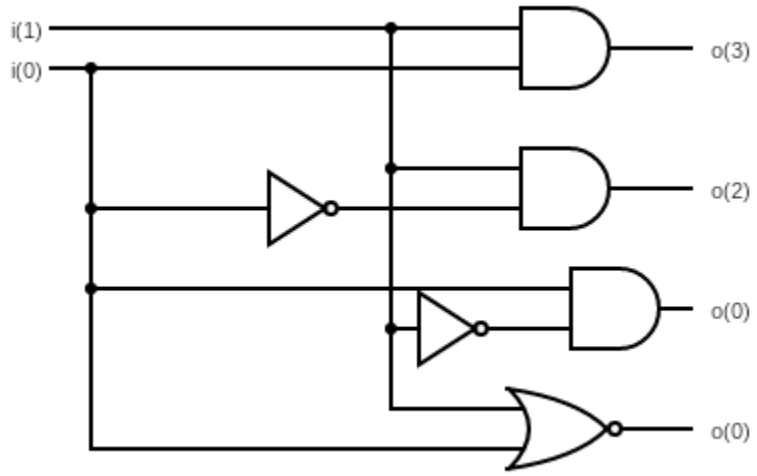
Block Diagram



Truth Table

Input				Output							
e	**i_2**	**i_1**	**i_0**	**o_7**	**o_6**	**o_5**	**o_4**	**o_3**	**o_2**	**o_1**	**o_0**
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Circuit Diagram



Entity

```
entity QuestionThree is
    Port ( e : in  STD_LOGIC;
           iii : in  STD_LOGIC_VECTOR (2 downto 0);
```

```
           ooo : out  STD_LOGIC_VECTOR (7 downto 0));
end QuestionThree;
```

Architecture

a) Using ***gate-level*** modeling

```
architecture Behavioral of QuestionThree_PartA is

begin
ooo(7)<= e and iii(2) and iii(1) and iii(0);
ooo(6)<= e and iii(2) and iii(1) and (not(iii(0)));
ooo(5)<= e and iii(2) and iii(0) and not(iii(1));
ooo(4)<= e and iii(2) and (iii(1) nor iii(0));
ooo(3)<= e and iii(1) and iii(0) and (not iii(2));
ooo(2)<= e and iii(1) and (iii(2) nor iii(0));
ooo(1)<= e and iii(0) and (iii(2) nor iii(1));
ooo(0)<= e and not(iii(0) or iii(1) or iii(2));
end Behavioral;
```

b) Using ***behavioral-level*** modeling

i. Using **if-else** statement

```
architecture Behavioral of QuestionThree_PartB_one is

begin
p1:process(e, iii)
                  begin
                        if e='0' then
                              ooo<="00000000";
                        elsif iii="000" then
                              ooo<="00000001";
                        elsif iii="001" then
                              ooo<="00000010";
                        elsif iii="010" then
                              ooo<="00000100";
                        elsif iii="011" then
                              ooo<="00001000";
                        elsif iii="100" then
```

```
                              ooo<="00010000";
                        elsif iii="101" then
                              ooo<="00100000";
                        elsif iii="110" then
                              ooo<="01000000";
                        elsif iii="111" then
                              ooo<="10000000";
                        end if;
                  end process;
end Behavioral;
```

ii. Using **case** statement

```
architecture Behavioral of QuestionThree_PartB_two is

begin
p2:process(e,iii)
      begin
            case e is
                  when '0' => ooo<="00000000";
                  when '1' =>
                  case iii is
                        when "000" => ooo<="00000001";
                        when "001" => ooo<="00000010";
                        when "010" => ooo<="00000100";
                        when "011" => ooo<="00001000";
                        when "100" => ooo<="00010000";
                        when "101" => ooo<="00100000";
                        when "110" => ooo<="01000000";
                        when "111" => ooo<="10000000";
                        when others =>ooo<="ZZZZZZZZ";
                  end case;
                  when others =>ooo<="ZZZZZZZZ";
            end case;
      end process;
end Behavioral;
```

iii. Using **when else** statement

```
architecture Behavioral of QuestionThree_PartB_three is

begin
```

```
ooo<="00000000" when e<='0' else
        "00000001" when iii<="000" else
        "00000010" when iii<="001" else
        "00000100" when iii<="010" else
        "00001000" when iii<="011" else
        "00010000" when iii<="100" else
        "00100000" when iii<="101" else
        "01000000" when iii<="110" else
        "10000000" when iii<="111" else
        "ZZZZZZZZ";
end Behavioral;
```

iv. Using **switch** when statement

```
architecture Behavioral of QuestionThree_PartB_four is

begin
with (e&iii) select
            ooo <= "00000001" when "1000",
                        "00000010" when "1001",
                        "00000100" when "1010",
                        "00001000" when "1011",
                        "00010000" when "1100",
                        "00100000" when "1101",
                        "01000000" when "1110",
                        "10000000" when "1111",
                    "00000000" when others;
end Behavioral;
```

TestBench

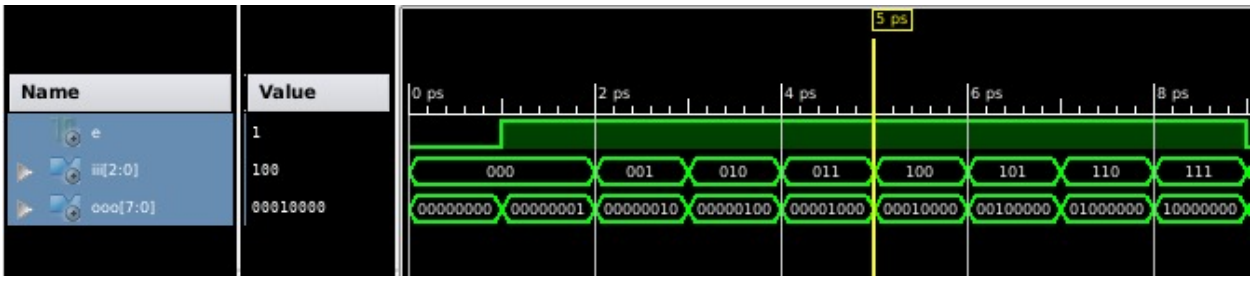
```
ENTITY QuestionThree_Testbench IS
END QuestionThree_Testbench;

ARCHITECTURE behavior OF QuestionThree_Testbench IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT QuestionThree_PartB_one
    PORT(
         e : IN  std_logic;
```

```
         iii : IN  std_logic_vector(2 downto 0);
         ooo : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal e : std_logic := '0';
   signal iii : std_logic_vector(2 downto 0) := (others => '0');

   --Outputs
   signal ooo : std_logic_vector(7 downto 0);

BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: QuestionThree_PartB_one PORT MAP (
          e => e,
          iii => iii,
          ooo => ooo
        );
   -- Stimulus process
   stim_proc: process
   begin
      e<='0';
            iii<="000";
            wait for 1 ps;
            e<='1';
            iii<="000";
            wait for 1 ps;
            iii<="001";
            wait for 1 ps;
            iii<="010";
            wait for 1 ps;
            iii<="011";
            wait for 1 ps;
            iii<="100";
            wait for 1 ps;
            iii<="101";
            wait for 1 ps;
            iii<="110";
            wait for 1 ps;
            iii<="111";
            wait for 1 ps;
```

```
   end process;

END;
```

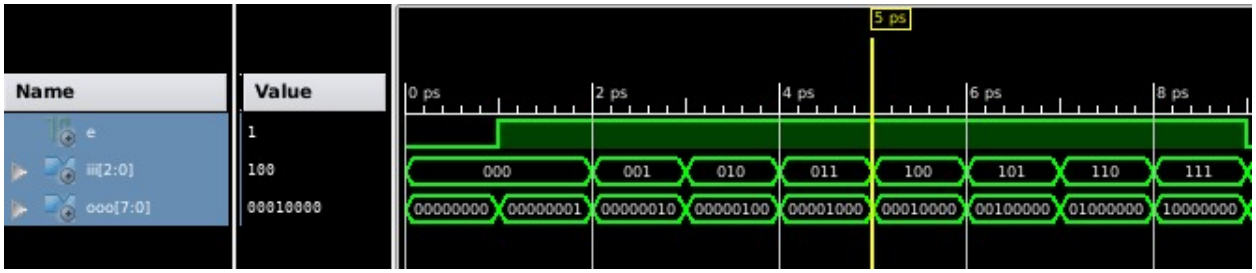
Timing Diagram

a) Using ***gate-level*** modeling

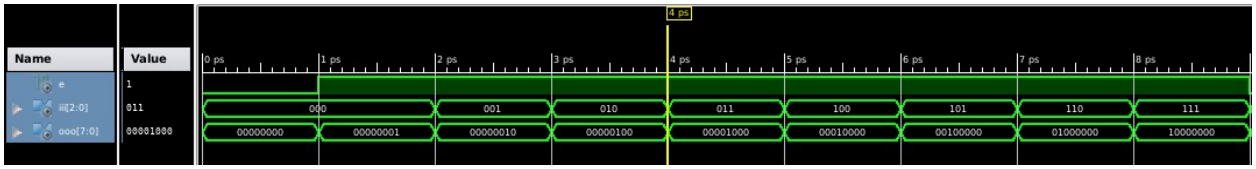


b) Using ***behavioral-level*** modeling

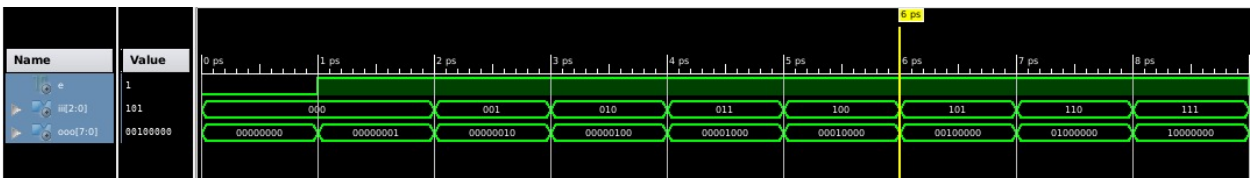
i) Using **if-else** statement



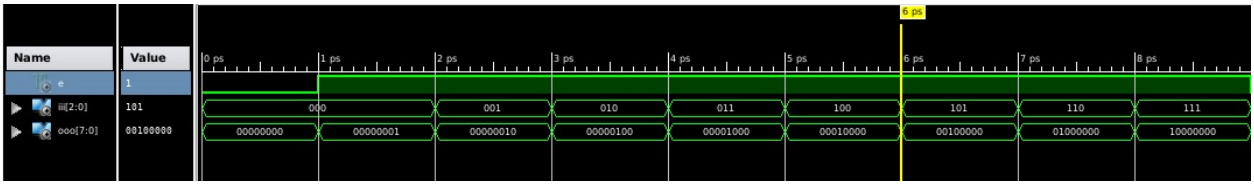
ii) Using **case** statement



iii) Using **when else** statement



iv) Using **select when** statement



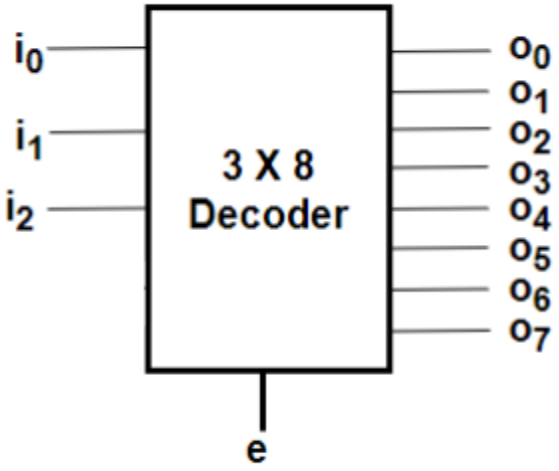
4. Design 3 X 8 Decoder using Component Instantiation.

Description

Design 3 X 8 decoder using 2 X 4 decoder and 1 X 2 decoder by component instantiation.

A 3 to 8 decoder has 3 input lines and 8 output lines. An enable input is provided to switch the decoder on and off.

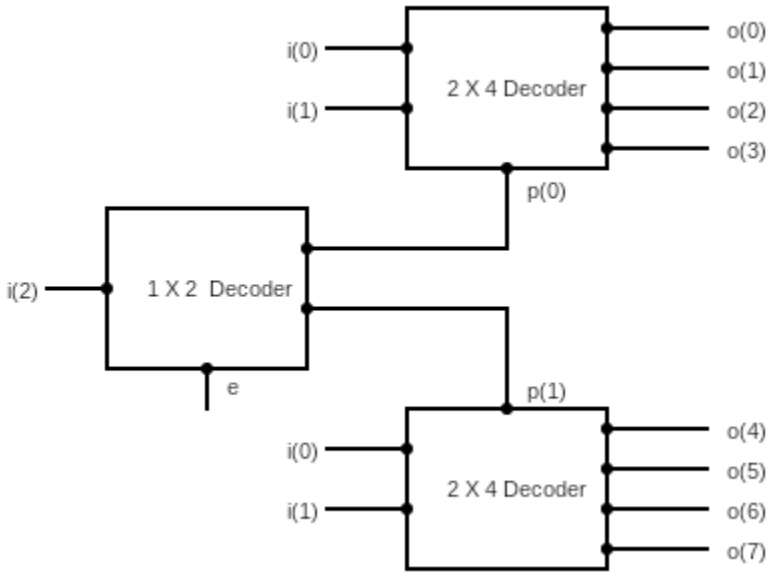
Block Diagram



Truth Table

Input				Output							
e	**i_2**	**i_1**	**i_0**	**o_7**	**o_6**	**o_5**	**o_4**	**o_3**	**o_2**	**o_1**	**o_0**
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Circuit Diagram



Entity

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

Architecture

```
architecture Behavioral of QuestionFour is

-- 2 X 4 Decoder
component QuestionTwo_PartB_one is
    Port ( ee : in  STD_LOGIC;
           ii : in  STD_LOGIC_VECTOR (1 downto 0);
           oo : out  STD_LOGIC_VECTOR (3 downto 0));
end component;

-- 1 X 2 Decoder
component QuestionOne_PartB_One is
    Port ( e : in  STD_LOGIC;
           i : in  STD_LOGIC;
           o : out  STD_LOGIC_VECTOR (1 downto 0));
end component;
signal p: STD_LOGIC_VECTOR(1 downto 0);

begin
c1: QuestionTwo_PartB_one port map(p(0), iiii(1 downto 0), oooo(3 downto
0));
c2: QuestionTwo_PartB_one port map(p(1), iiii(1 downto 0), oooo(7 downto
4));
c3: QuestionOne_PartB_One port map(eeee, iiii(2), p);
end Behavioral;
```

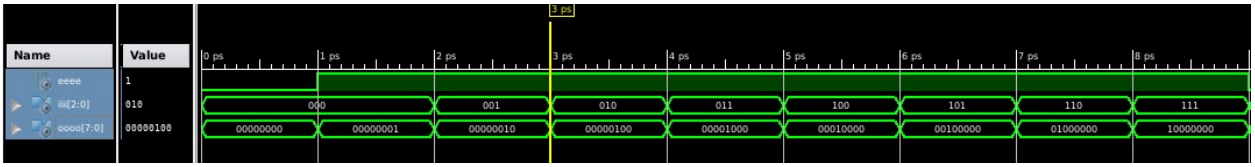
TestBench

```
ENTITY QuestionFive_TestBench IS
END QuestionFive_TestBench;

ARCHITECTURE behavior OF QuestionFive_TestBench IS
```

```
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT QuestionFive
    PORT(
         eeeee : IN  std_logic;
         iiiii : IN  std_logic_vector(3 downto 0);
         oooo : OUT  std_logic_vector(15 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal eeeee : std_logic := '0';
   signal iiiii : std_logic_vector(3 downto 0) := (others => '0');

   --Outputs
   signal oooo : std_logic_vector(15 downto 0);

BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: QuestionFive PORT MAP (
          eeeee => eeeee,
          iiiii => iiiii,
          oooo => oooo
        );
   -- Stimulus process
   stim_proc: process
   begin
            eeeee<='0';
            iiiii<="0000";
            wait for 1 ps;
            eeeee<='1';
            iiiii<="0000";
            wait for 1 ps;
            iiiii<="0001";
            wait for 1 ps;
            iiiii<="0010";
            wait for 1 ps;
            iiiii<="0011";
            wait for 1 ps;
            iiiii<="0100";
            wait for 1 ps;
            iiiii<="0101";
            wait for 1 ps;
            iiiii<="0110";
```

```
            wait for 1 ps;
            iiiii<="0111";
            wait for 1 ps;
            iiiii<="1000";
            wait for 1 ps;
            iiiii<="1001";
            wait for 1 ps;
            iiiii<="1010";
            wait for 1 ps;
            iiiii<="1011";
            wait for 1 ps;
            iiiii<="1100";
            wait for 1 ps;
            iiiii<="1101";
            wait for 1 ps;
            iiiii<="1110";
            wait for 1 ps;
            iiiii<="1111";
            wait for 1 ps;
      end process;
END;
```

Timing Diagram



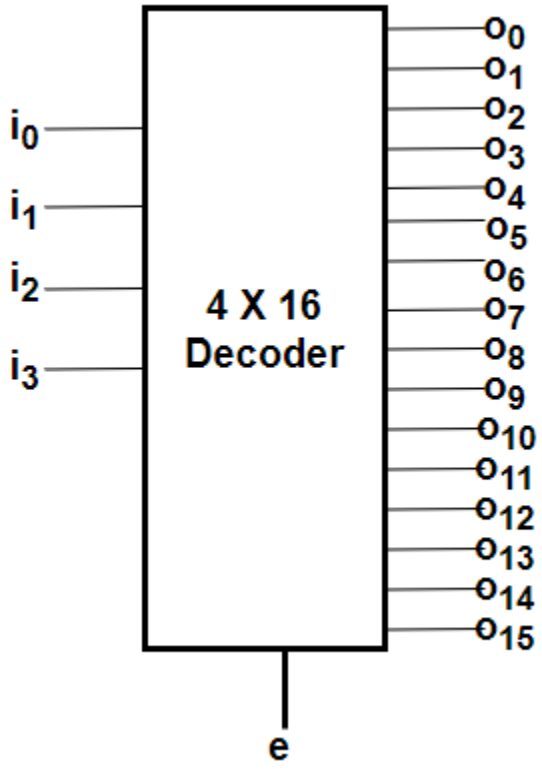
5. Design 4 X 16 Decoder using Component Instantiation.

Description

Design 4 X 16 decoder using 2 X 4 decoder by component instantiation.

A decoder is a combinational circuit that has N input lines and a maximum of 2^N output lines. One of these outputs will be active high based on the combination of inputs present when the decoder is enabled. That means the decoder detects a particular code. A 4 to 16 decoder has 4 input lines and 16 output lines. An enable input is provided to switch the decoder on and off.

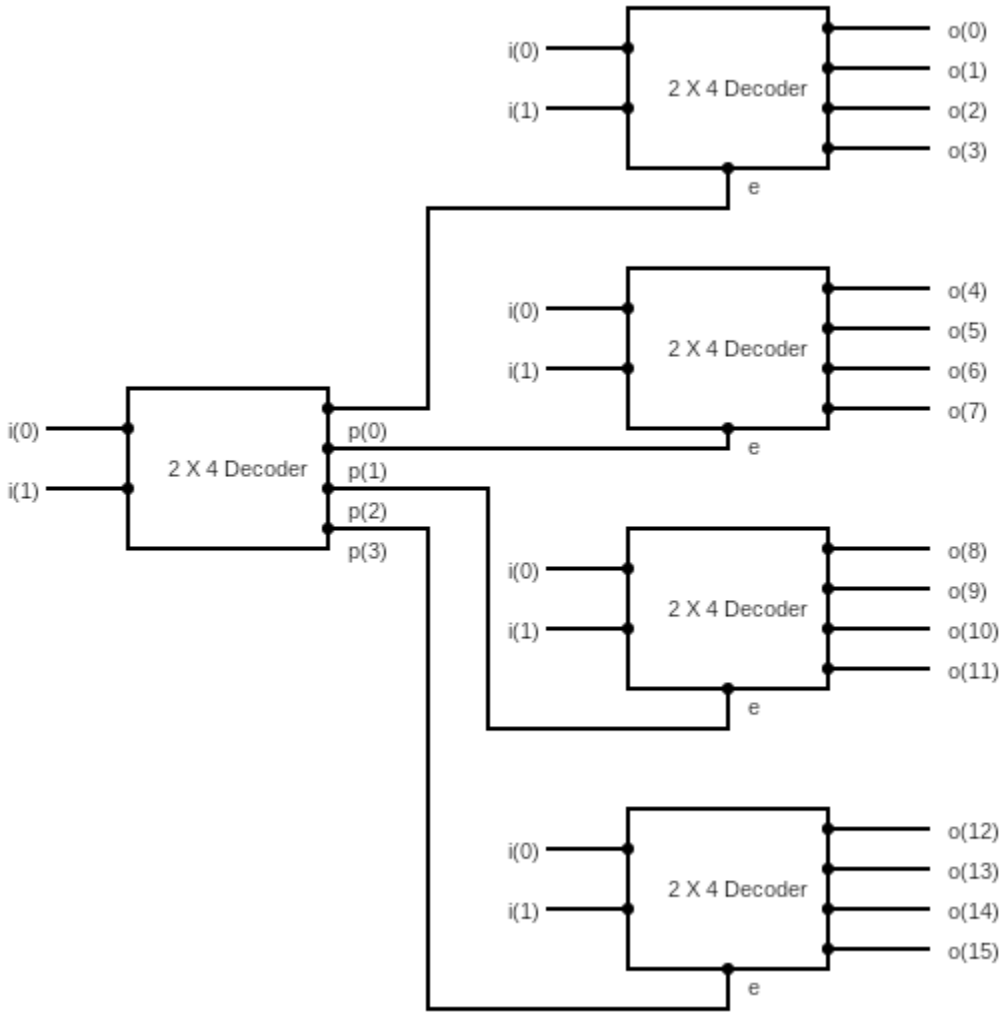
Block Diagram



Truth Table

Input					Output														
e	**i_3**	**i_2**	**i_1**	**i_0**	**o_7**	**o_6**	**o_5**	**o_4**	**o_3**	**o_2**	**o_1**	**o_0**	**o_7**	**o_6**	**o_5**	**o_4**	**o_3**	**o_2**	**o_1**	**o_0**
0	X	X	X	X	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	0	1
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	1	1	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	0	1	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Circuit Diagram



Entity

```
entity QuestionFive is
    Port ( eeeee : in  STD_LOGIC;
           iiiii : in  STD_LOGIC_VECTOR (3 downto 0);
           oooo : out  STD_LOGIC_VECTOR (15 downto 0));
end QuestionFive;
```

Architecture

```
architecture Behavioral of QuestionFive is

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

signal p: STD_LOGIC_VECTOR(3 downto 0);

begin
c1: QuestionTwo_PartB_one port map(p(0), iiiii(1 downto 0), oooo(3 downto
0));
c2: QuestionTwo_PartB_one port map(p(1), iiiii(1 downto 0), oooo(7 downto
4));
c3: QuestionTwo_PartB_one port map(p(2), iiiii(1 downto 0), oooo(11 downto
8));
c4: QuestionTwo_PartB_one port map(p(3), iiiii(1 downto 0), oooo(15 downto
12));
c5: QuestionTwo_PartB_one port map(eeeee, iiiii(3 downto 2), p);

end Behavioral;
```

TestBench

```
ENTITY QuestionFive_TestBench IS
END QuestionFive_TestBench;

ARCHITECTURE behavior OF QuestionFive_TestBench IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT QuestionFive
    PORT(
         eeeee : IN  std_logic;
         iiiii : IN  std_logic_vector(3 downto 0);
         oooo : OUT  std_logic_vector(15 downto 0)
        );
    END COMPONENT;
```

```
   --Inputs
   signal eeeee : std_logic := '0';
   signal iiiii : std_logic_vector(3 downto 0) := (others => '0');
   --Outputs
   signal oooo : std_logic_vector(15 downto 0);

BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: QuestionFive PORT MAP (
          eeeee => eeeee,
          iiiii => iiiii,
          oooo => oooo
        );
   -- Stimulus process
   stim_proc: process
   begin
            eeeee<='0';
            iiiii<="0000";
            wait for 1 ps;
            eeeee<='1';
            iiiii<="0000";
            wait for 1 ps;
            iiiii<="0001";
            wait for 1 ps;
            iiiii<="0010";
            wait for 1 ps;
            iiiii<="0011";
            wait for 1 ps;
            iiiii<="0100";
            wait for 1 ps;
            iiiii<="0101";
            wait for 1 ps;
            iiiii<="0110";
            wait for 1 ps;
            iiiii<="0111";
            wait for 1 ps;
            iiiii<="1000";
            wait for 1 ps;
            iiiii<="1001";
            wait for 1 ps;
            iiiii<="1010";
            wait for 1 ps;
            iiiii<="1011";
```

```
            wait for 1 ps;
            iiiii<="1100";
            wait for 1 ps;
            iiiii<="1101";
            wait for 1 ps;
            iiiii<="1110";
            wait for 1 ps;
            iiiii<="1111";
            wait for 1 ps;
      end process;
END;
```

Timing Diagram