Experiment -1

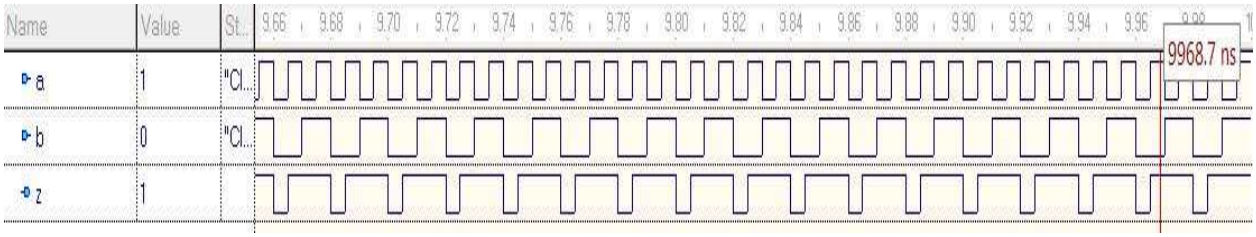
Aim:- Design all gates using VHDL.

1. OR gate

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity  dee_or is
        port(
                a : in STD_LOGIC;
                b : in STD_LOGIC;
                z : out STD_LOGIC
            );
end dee_or;
--}} End of automatically maintained section

architecture Behavioral of dee_or is
begin
        z<= a or b;
end Behavioral;
```

Output:-

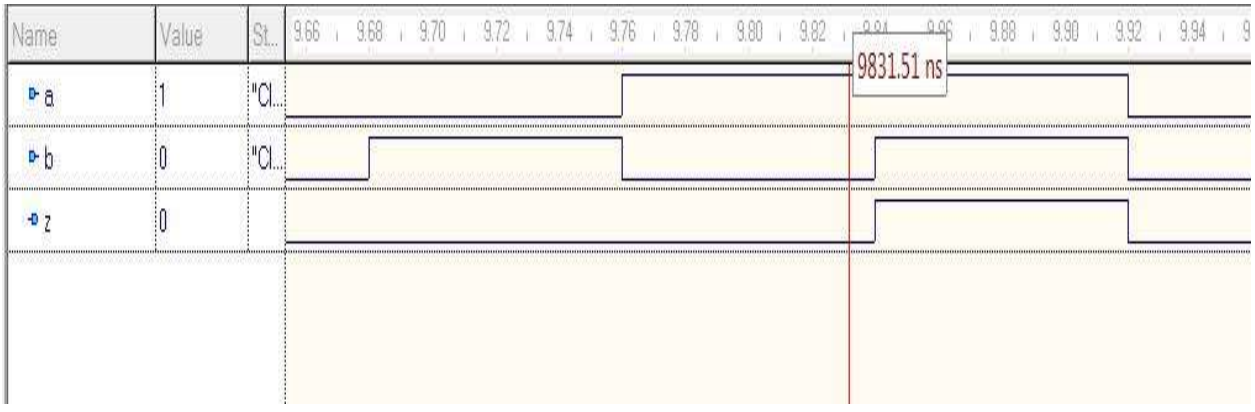


2. AND gate

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity dee_and is
        port(
                a : in STD_LOGIC;
                b : in STD_LOGIC;
                z : out STD_LOGIC
            );
end dee_and;
--}} End of automatically maintained section

architecture Behavioral of dee_and is
begin
        z<=a and b;
end Behavioral;
```

Output:-

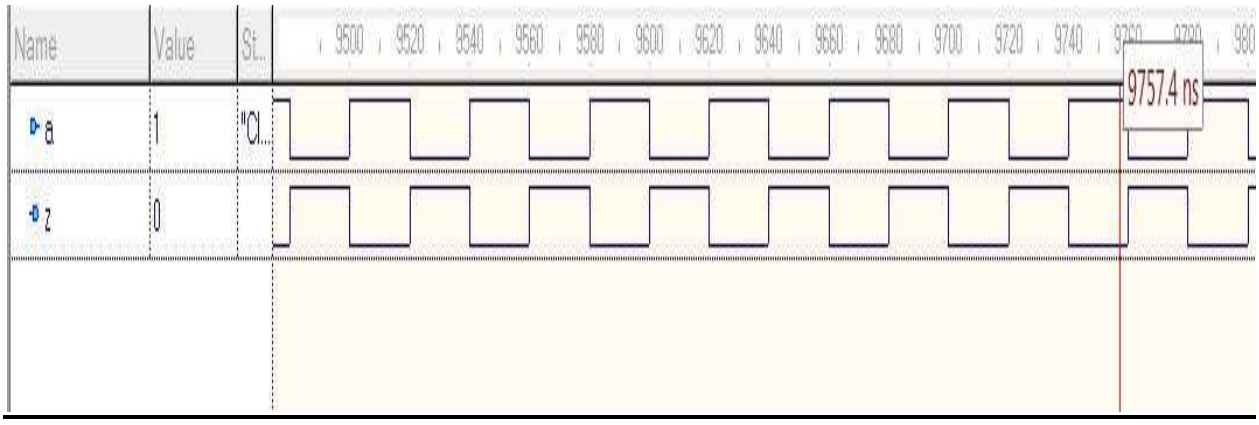


3. NOT gate

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity dee_not is
        port(
                a : in STD_LOGIC;
                z : out STD_LOGIC
            );
end dee_not;
--}} End of automatically maintained section

architecture Behavioral of dee_not is
begin
        z<= not a;
end Behavioral;
```

Output:-

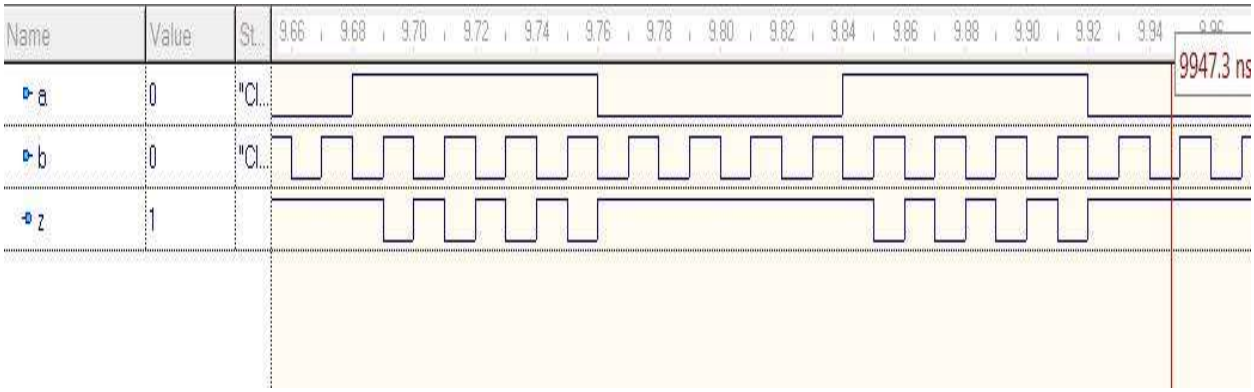


4. NAND gate

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity dee_nand is
        port(
                a : in STD_LOGIC;
                b : in STD_LOGIC;
                z : out STD_LOGIC
            );
end dee_nand;
--}} End of automatically maintained section

architecture Behavioral of dee_nand is
begin
        z<= a nand b;
end Behavioral;
```

Output:-

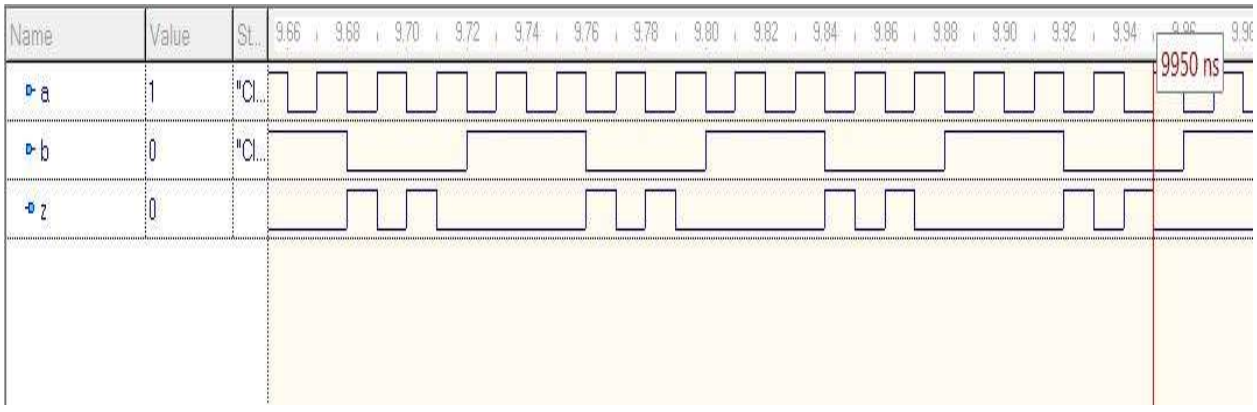


5. NOR gate

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity dee_nor is
        port(
                a : in STD_LOGIC;
                b : in STD_LOGIC;
                z : out STD_LOGIC
            );
end dee_nor;
--}} End of automatically maintained section

architecture Behavioral of dee_nor is
begin
        z<= a nor b;
end Behavioral;
```

Output:-

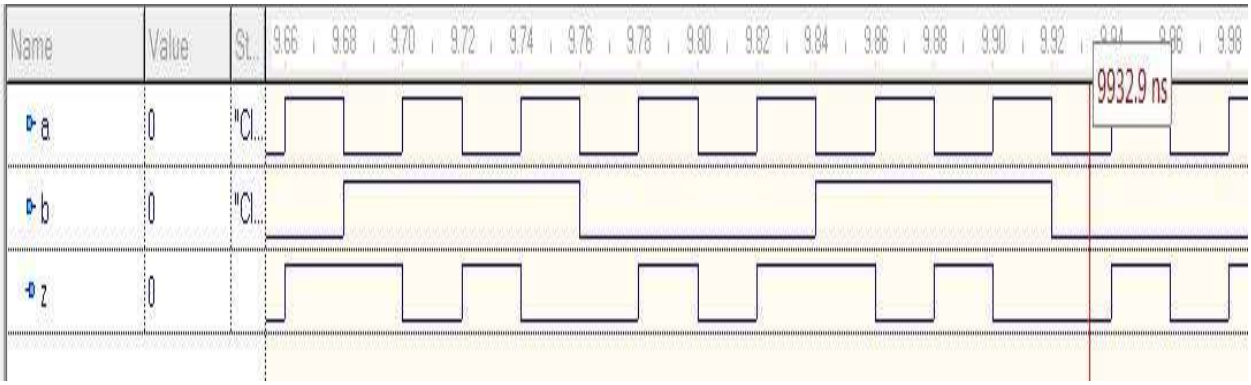


6. EX-OR gate

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity dee_exor is
        port(
                a : in STD_LOGIC;
                b : in STD_LOGIC;
                z : out STD_LOGIC
            );
end dee_exor;
--}} End of automatically maintained section

architecture Behavioral of dee_exor is
begi
        z<= a xor b;
end Behavioral;
```

Output:-

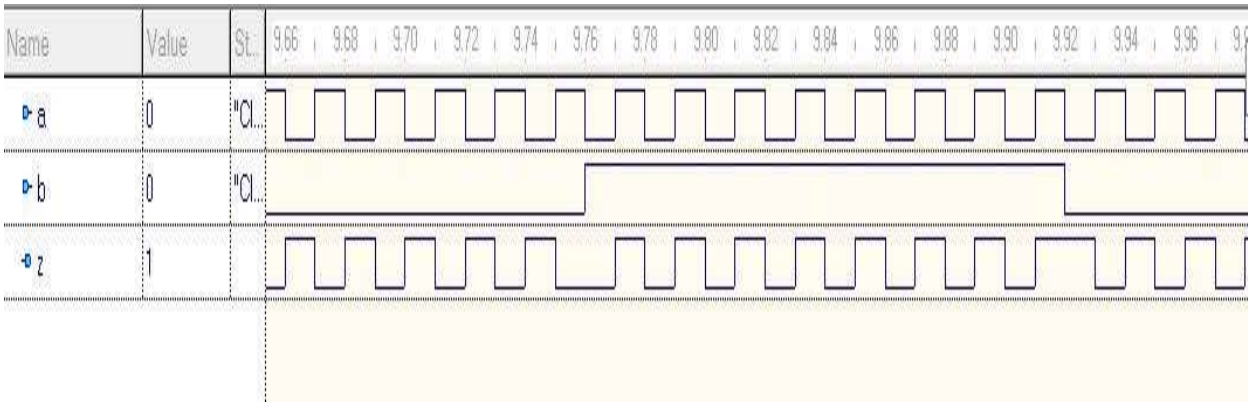


7. EX-NOR gate

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity dee_exnor is
        port(
                a : in STD_LOGIC;
                b : in STD_LOGIC;
                z : out STD_LOGIC
            );
end dee_exnor;
--}} End of automatically maintained section

architecture Behavioral of dee_exnor is
begin
        z<= a  xnor  b;
end Behavioral;
```

Output:-



Experiment – 2

Aim :- Write VHDL programs for the following circuits, check the wave forms and the hardware generated

i) half adder

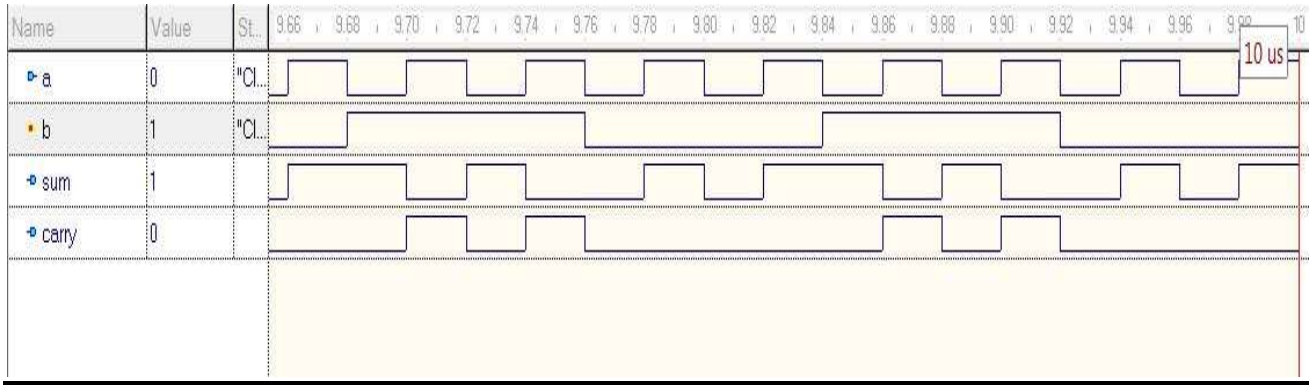
ii) full adder

1. Half-adder

Code :-

```
library IEEE;

use IEEE.STD_LOGIC_1164.all;


entity \dee_half-adder\ is

        port(

                a : in STD_LOGIC;

                b : in STD_LOGIC;

                sum : out STD_LOGIC;

                carry : out STD_LOGIC

            );

end \dee_half-adder\;

--}} End of automatically maintained section


architecture Behavioral of \dee_half-adder\ is

begin

        sum<= a xor b;

        carry <= a and b;

end Behavioral;
```

Output :-



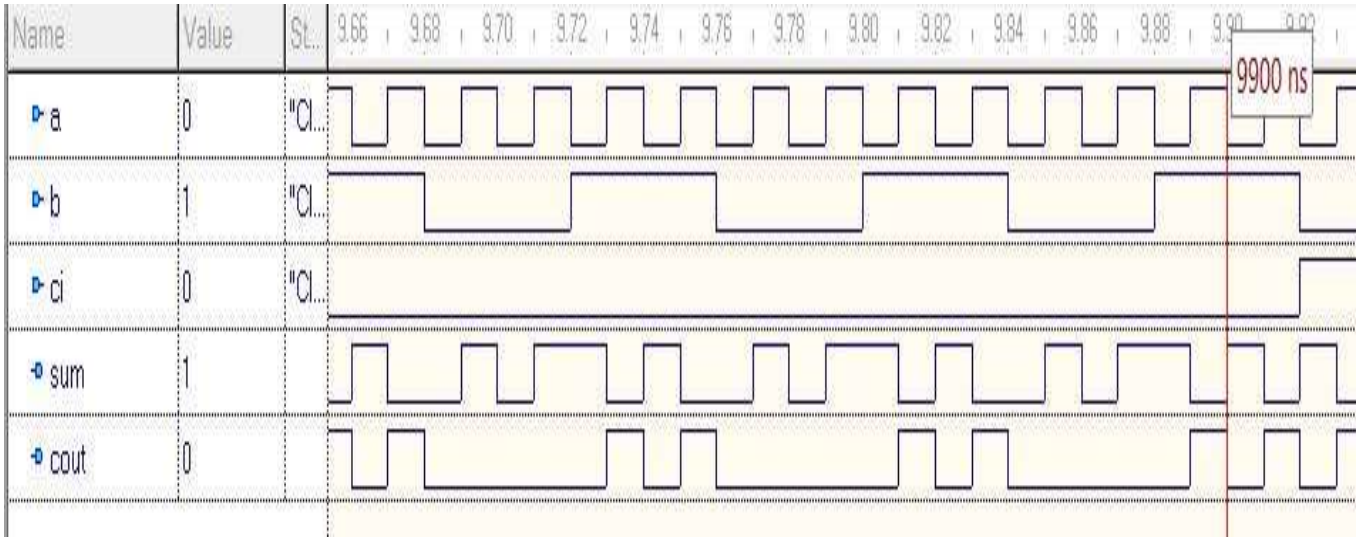
2. Full-adder

Code :-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;


entity \dee_full-adder\ is
        port(
                a : in STD_LOGIC;
                b : in STD_LOGIC;
                ci : in STD_LOGIC;
                sum : out STD_LOGIC;
                cout : out STD_LOGIC
            );
end \dee_full-adder\;

--}} End of automatically maintained section
```

```
architecture Behavioral of \dee_full-adder\ is
begin

        sum<= a xor b xor ci;
        cout <= (a and b) or (b and ci) or ( a and ci);

end Behavioral;
```

Output :-



Experiment – 3

Aim :- Write VHDL programs for the following circuits, check the wave forms and the hardware generated

i) half subtractor

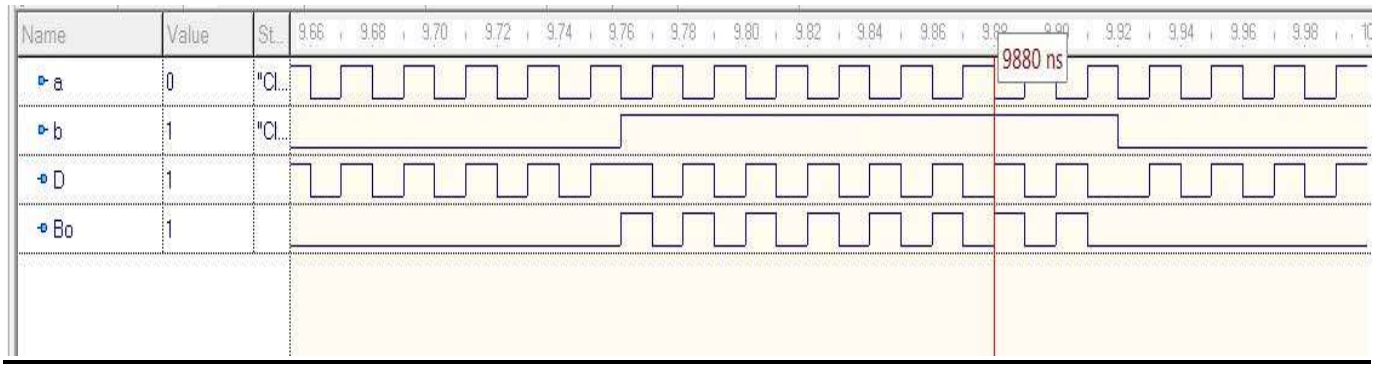
ii) full subtractor

1. Half-subtractor

Code :-

```
library IEEE;

use IEEE.STD_LOGIC_1164.all;

entity dee_half_sub is

        port(   a : in STD_LOGIC;

                b : in STD_LOGIC;

                D : out STD_LOGIC;

                Bo : out STD_LOGIC

            );

end dee_half_sub;

--}} End of automatically maintained section

architecture Behavioral of dee_half_sub is

begin

        D<=a xor b;

        Bo<= (not a) and b;

end Behavioral;
```

Output :-



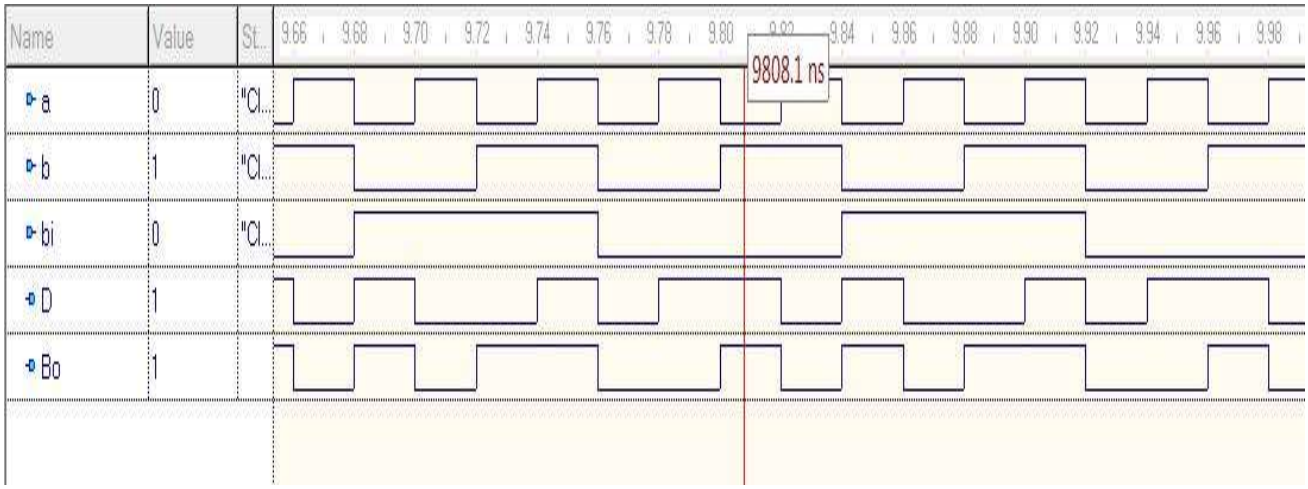
2. Full-subtractor

Code :-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity dee_full_sub is
        port(   a : in STD_LOGIC;
                b : in STD_LOGIC;
                bi : in STD_LOGIC;
                D : out STD_LOGIC;
                Bo : out STD_LOGIC
            );
end dee_full_sub;
--}} End of automatically maintained section

architecture Behavioral of dee_full_sub is
begin
        D<= a xor b xor bi;
        Bo<= ((not a) and b)   or  ((not a) and bi)  or  (bi and b) ;
```

end Behavioral;

Output :-



Experiment – 4

AIM :- Write a VHDL program for the converting bits in following ways and check the waveforms generated.

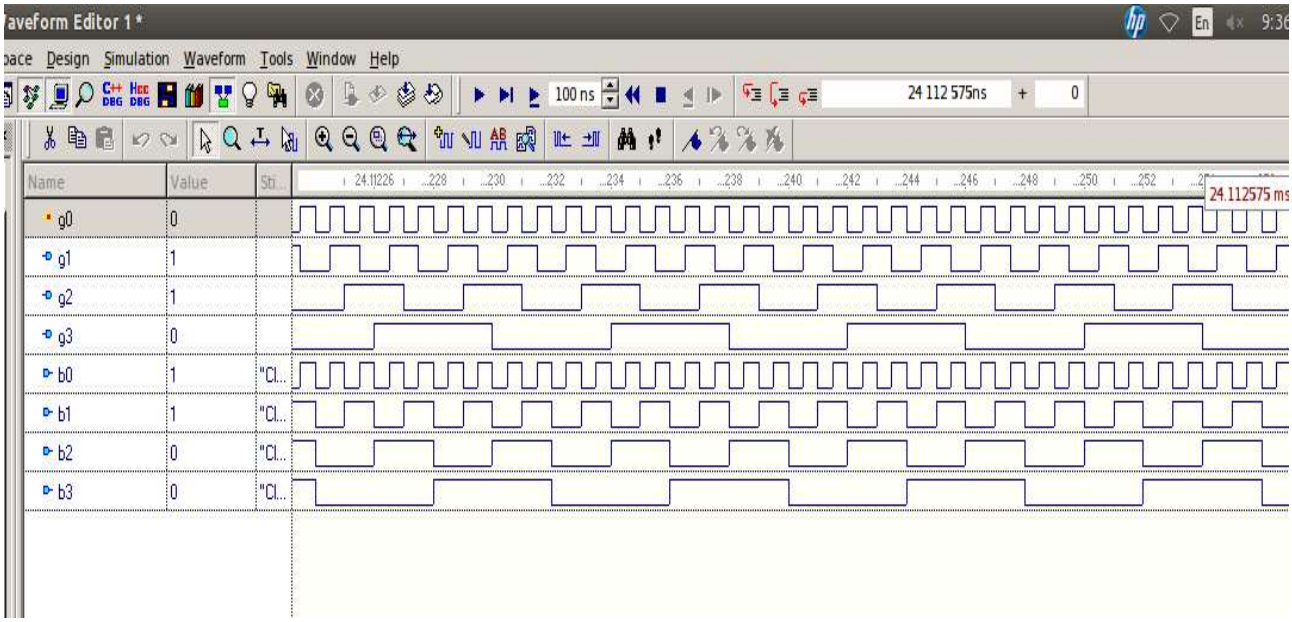
1.Binary to Gray
2.Gray to Binary

CODE:-

1. Binary to gray

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity binary_gray is
        port(
                b0 : in STD_LOGIC;
                b1 : in STD_LOGIC;
                b2 : in STD_LOGIC;
                b3 : in STD_LOGIC;
                g0 : out STD_LOGIC;
                g1 : out STD_LOGIC;
                g2 : out STD_LOGIC;
                g3 : out STD_LOGIC

end binary_gray;

--}} End of automatically maintained section

architecture behavioral of binary_gray is

begin

g0<= b0;
g1<=b0 xor b1;
g2<=b1 xor b2;
g3<= b2 xor b3;

end behavioral;
```

OUTPUT :-

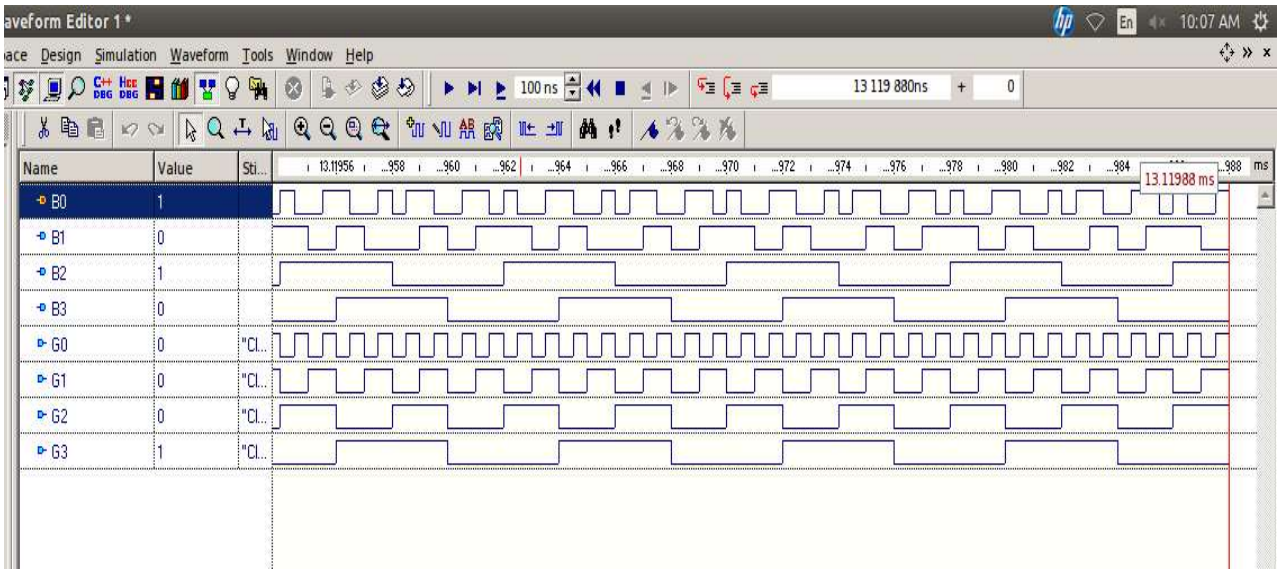


2.Gray to Binary

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity GRAY_BINARY is
        port(
                G0 : in STD_LOGIC;
                G1 : in STD_LOGIC;
                G2 : in STD_LOGIC;
                G3 : in STD_LOGIC;
                B0 : out STD_LOGIC;
                B1 : out STD_LOGIC;
                B2 : out STD_LOGIC;
                B3 : out STD_LOGIC
            );
end GRAY_BINARY;

--}} End of automatically maintained section

architecture BEHAVIORAL of GRAY_BINARY is
begin

        B0<=G0;
        B1<=B0 XOR G1;
```

```
        B2<=B1 XOR G2;
        B3<= B2 XOR G3;

end BEHAVIORAL;
```

OUTPUT:-



Experiment – 5

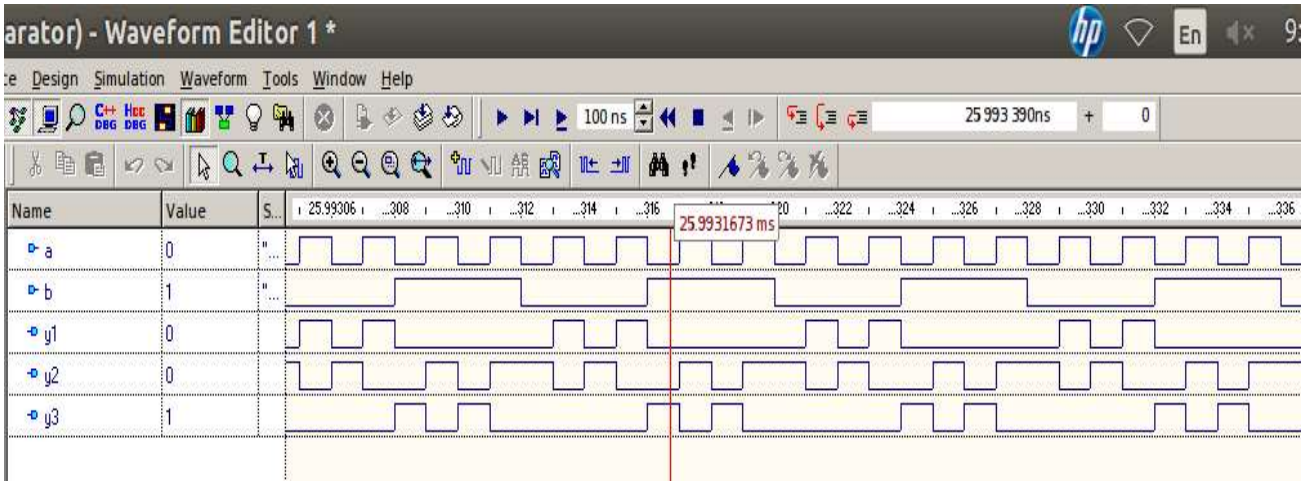
Aim :- Write a VHDL program for a comparator and check the wave forms.

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity comparator is
        port(   a : in STD_LOGIC;
                b : in STD_LOGIC;
                y1 : out STD_LOGIC;             -- y1= (a>b)
                y2 : out STD_LOGIC;             -- y2= (a=b)
                y3 : out STD_LOGIC              -- y3= (a<b)
            );
end comparator;
--}} End of automatically maintained section
architecture Behavioral of comparator is
begin
        process(a,b)
        begin
                if a='0' and b='0' then
                  y1<='0';                                      -- y1= (a>b)
                  y2<='1';                                      -- y2= (a=b)
                  y3<='0';                                      -- y3= (a<b)
                elsif a='0' and b='1' then
                  y1<='0';
```

```
            y2<='0';
            y3<='1';
        elsif a='1' and b='0' then
            y1<='1';
            y2<='0';
            y3<='0';
        elsif a='1' and b='1' then
            y1<='0';
            y2<='1';
            y3<='0';
        else
            y1<='0';
            y2<='0';
            y3<='0';
    end if;
    end process;
end Behavioral;
```

Output :-



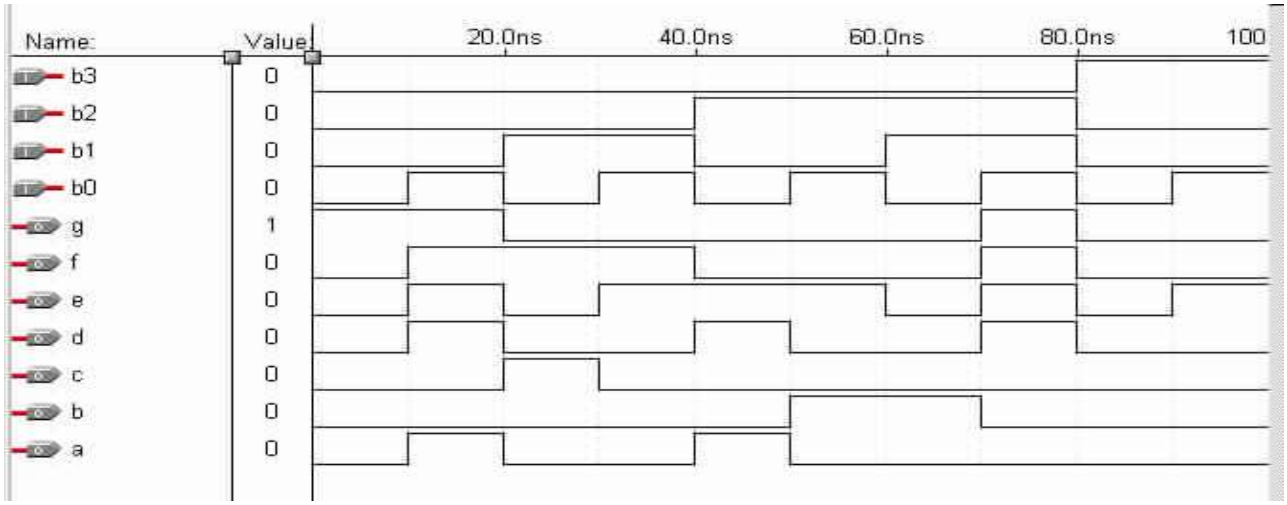
Experiment – 6

Aim :- Write a VHDL program for BCD to 7 segment display and checks the waveform.

Code:-

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

entity bcd_7seg is
Port ( b0,b1,b2,b3 : in STD_LOGIC;
a,b,c,d,e,f ,g: out STD_LOGIC);
end bcd_7seg;

architecture Behavioral of bcd_7seg is

begin

a <= b0 OR b2 OR (b1 AND b3) OR (NOT b1 AND NOT b3);
b <= (NOT b1) OR (NOT b2 AND NOT b3) OR (b2 AND b3);
c <= b1 OR NOT b2 OR b3;
d <= (NOT b1 AND NOT b3) OR (b2 AND NOT b3) OR (b1 AND NOT b2 AND b3) OR (NOT b1 AND b2) OR
b0;
e <= (NOT b1 AND NOT b3) OR (b2 AND NOT b3);
f <= b0 OR (NOT b2 AND NOT b3) OR (b1 AND NOT b2) OR (b1 AND NOT b3);
g <= b0 OR (b1 AND NOT b2) OR ( NOT b1 AND b2) OR (b2 AND NOT b3);

end Behavioral;
```

Output :-



Experiment – 7

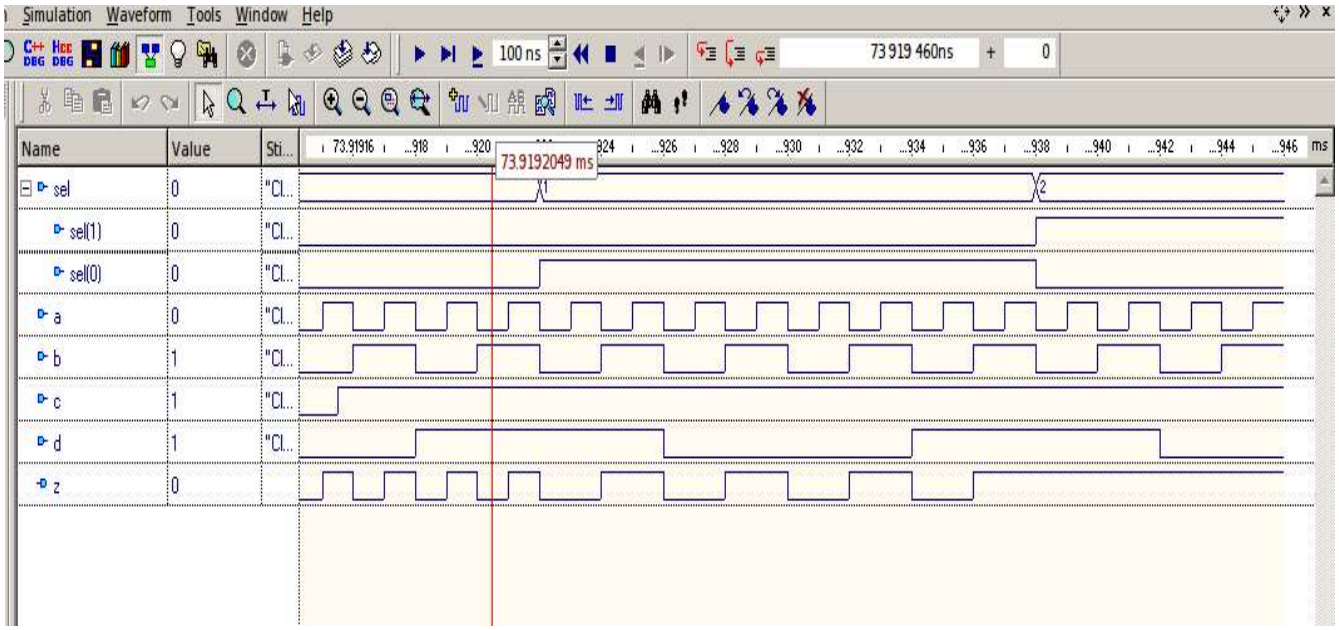
Aim :- Write a VHDL program for a 4 bit multiplexer and check the wave forms.

Code:- Using selective statement

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;


entity selective_mux4 is
	port(
		sel : in STD_LOGIC_VECTOR(1 DOWNTO 0);
		a : in STD_LOGIC;
		b : in STD_LOGIC;
		c : in STD_LOGIC;
		d : in STD_LOGIC;
		z : out STD_LOGIC
	  );
end selective_mux4;


--}} End of automatically maintained section


architecture Dataflow of selective_mux4 is
begin


	with sel select
```

```
        z<= a when "00" ,
        b when "01"     ,
        c when "10"  ,
        d when others;
end Dataflow;
```

Output :-

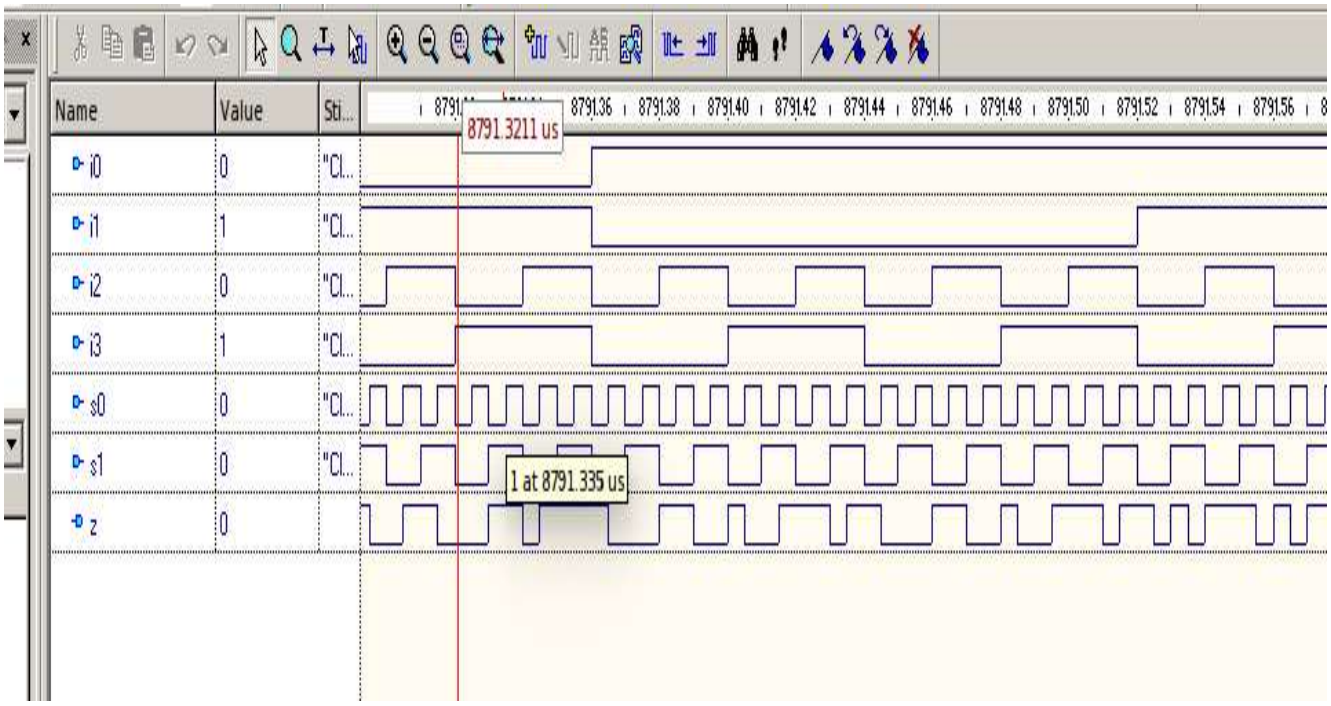


Code :- using Boolean expression

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity mux4 is
        port(
                i0 : in STD_LOGIC;
                i1 : in STD_LOGIC;
```

```
        i2 : in STD_LOGIC;
        i3 : in STD_LOGIC;
        s0 : in STD_LOGIC;
        s1 : in STD_LOGIC;
        z : out STD_LOGIC
    );
end mux4;
--}} End of automatically maintained section
architecture Dataflow of mux4 is
begin
    z<=( (not s0 and not s1) and i0 )or  ((not s0 and s1) and i1)  or ( ( s0 and not s1) and i2) or ( ( s0 and  s1) and i3);
end Dataflow;
```

Output :-



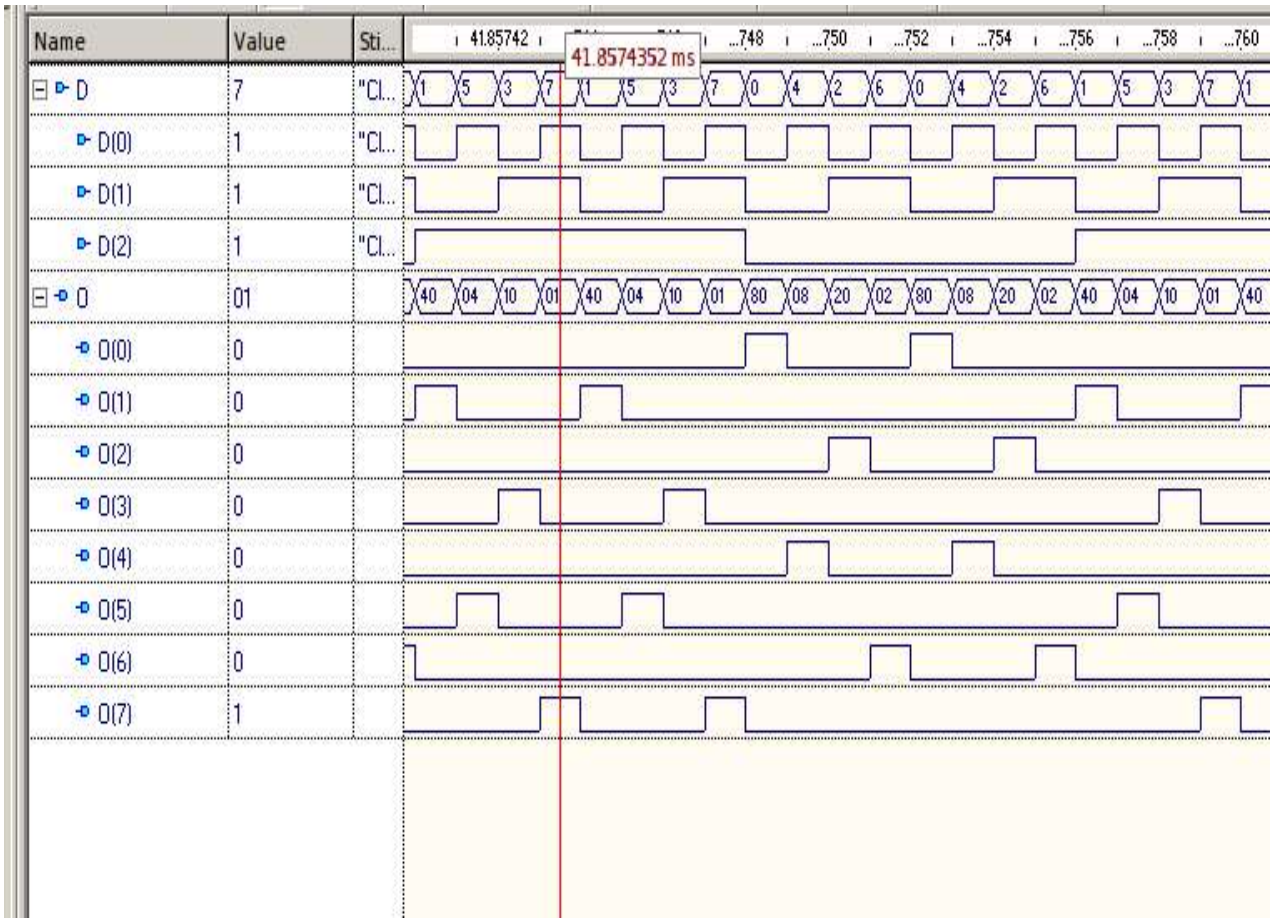
Experiment – 8

AIM :- Write a VHDL program for a 3x8 decoder and check the waveforms.

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity deco3_8 is
        port(
                D : in STD_LOGIC_VECTOR(0 TO 2);
                O : out STD_LOGIC_VECTOR( 0 TO 7)
            );
end deco3_8;

--}} End of automatically maintained section


architecture selective of deco3_8 is


begin

        with  D select
        O<= "10000000" when "000",
        "01000000" when "001",
        "00100000" when "010",
        "00010000" when "011",
        "00001000" when "100",
        "00000100" when "101",
        "00000010" when "110",
        "00000001" when "111",
        "00000000" when others;


end selective;
```

Output:-


Name
Value
Sti...
41.85742
41.8574352 ms
...748
...750
...752
...754
...756
...758
...760
D
7
D(0)
1
D(1)
1
D(2)
1
O
01
O(0)
0
O(1)
0
O(2)
0
O(3)
0
O(4)
0
O(5)
0
O(6)
0
O(7)
1

Experiment – 9

AIM :- Write a VHDL program for T-flipflop and check the waveforms.

Code:-

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity T_flipflop is
        port(
                p : in STD_LOGIC;
                c : in STD_LOGIC;
                clk : in STD_LOGIC;
                t : in STD_LOGIC;
                q : inout STD_LOGIC;
                qb : inout STD_LOGIC
            );
end T_flipflop;

--}} End of automatically maintained section

architecture behavioral of T_flipflop is
begin

        process(clk,p,c)
        begin
                if (p='0') then
                        q<='1';
                        qb<='0';
                elsif (c='0') then
                        q<='0';
                        qb<='1';
                elsif(p='0' and c='0') then
                        q<='X';
                        qb<='X';
                elsif(clk='0') then
                        if (t='1') then
                                q<=not q;
```

```
            qb<=not qb;
        else
                q<=q;
            qb<=not q;
        end if;
        end if;
    end process;

end behavioral;
```

Output:-