LIST OF EXPERIMENT

EXPERIMENT NO. I

AIM:-

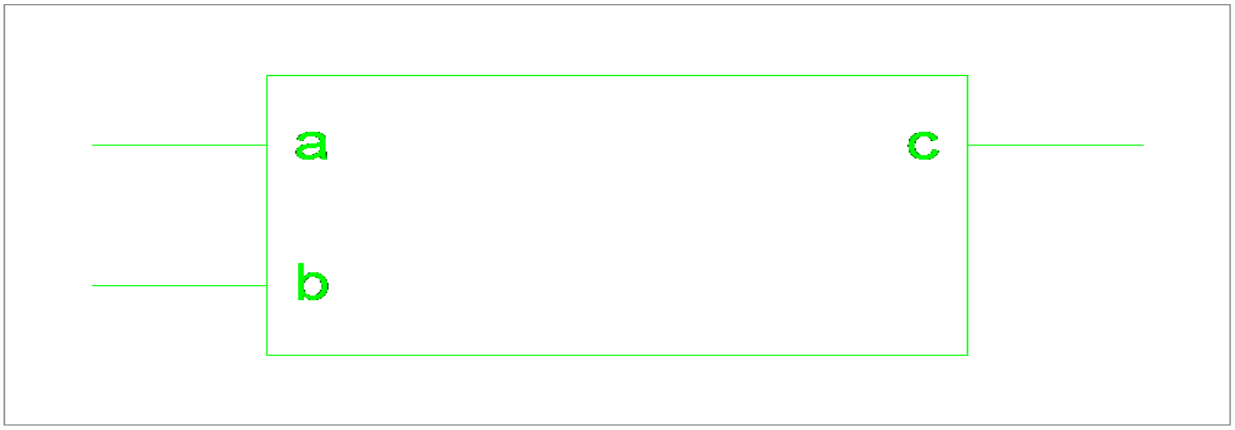
Synthesis & simulation of all basic logic gates using VHDL.

PROGRAM:-

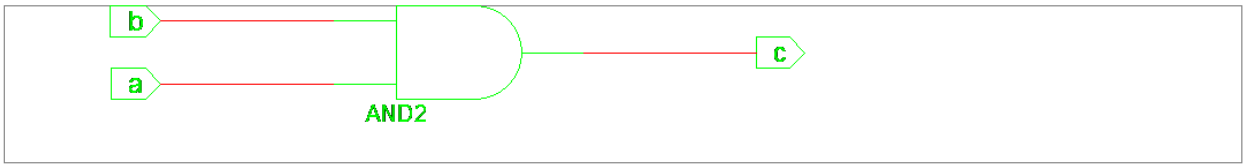
1) and_gate.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity and_gate is
Port ( a : in  STD_LOGIC;
b : in  STD_LOGIC;
c : out  STD_LOGIC);
end and_gate;
architecture Behavioral of and_gate is
begin
c <= a and b;
end Behavioral;
```

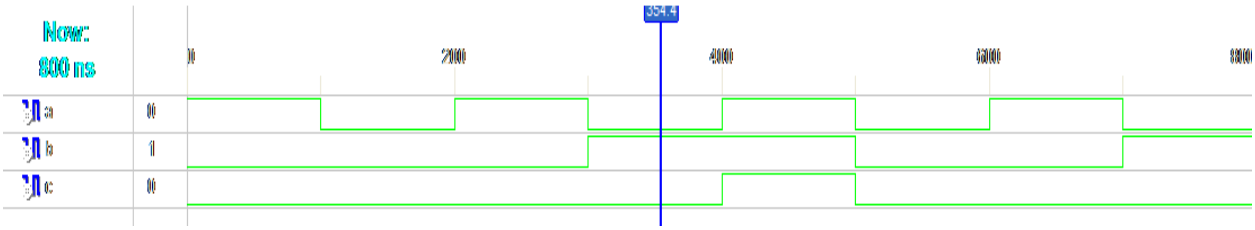
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



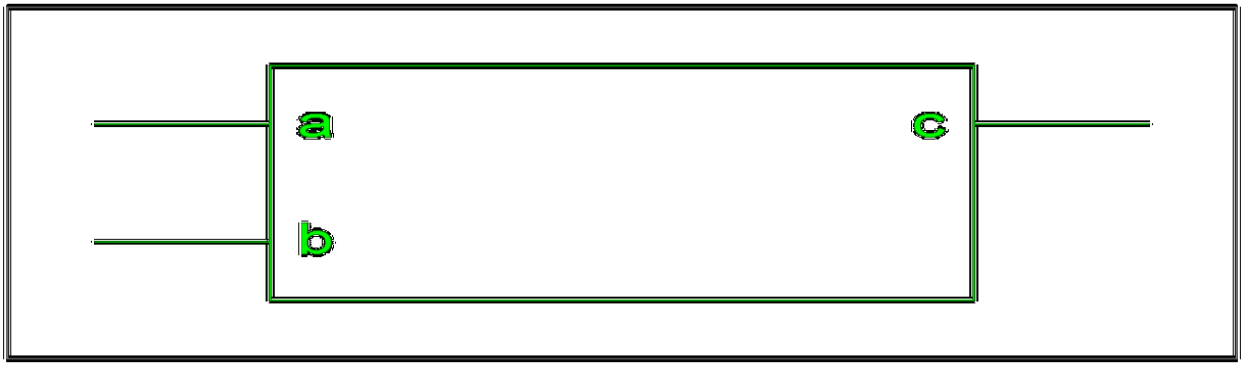
(c) SIMULATION WAVEFORM



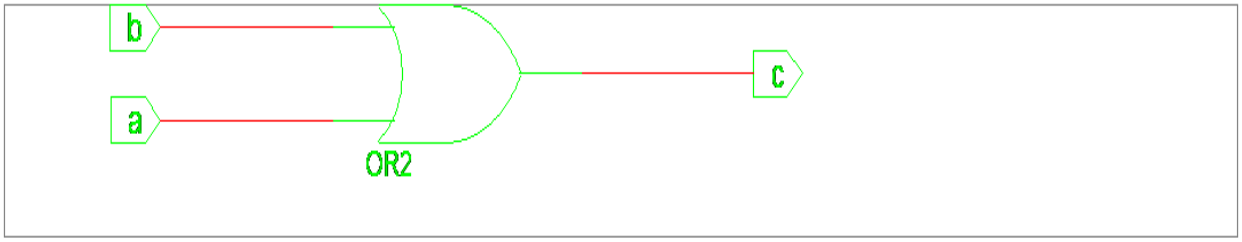
2) or_gate.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity or_gate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          c : out  STD_LOGIC);
end or_gate;
architecture Behavioral of or_gate is
begin
c <= a or b;
end Behavioral
```

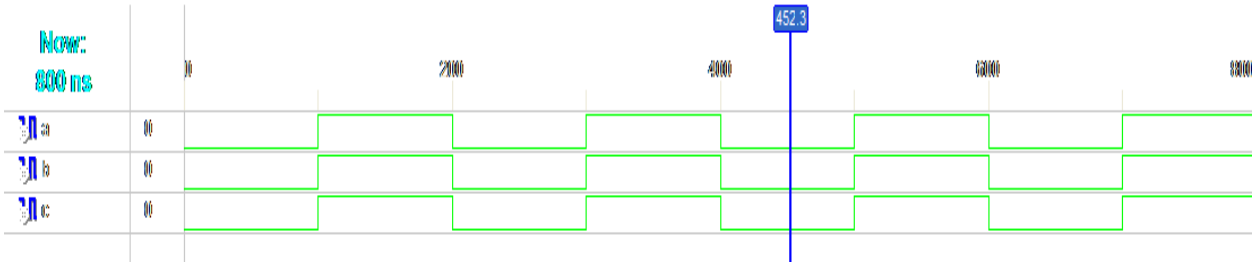
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM

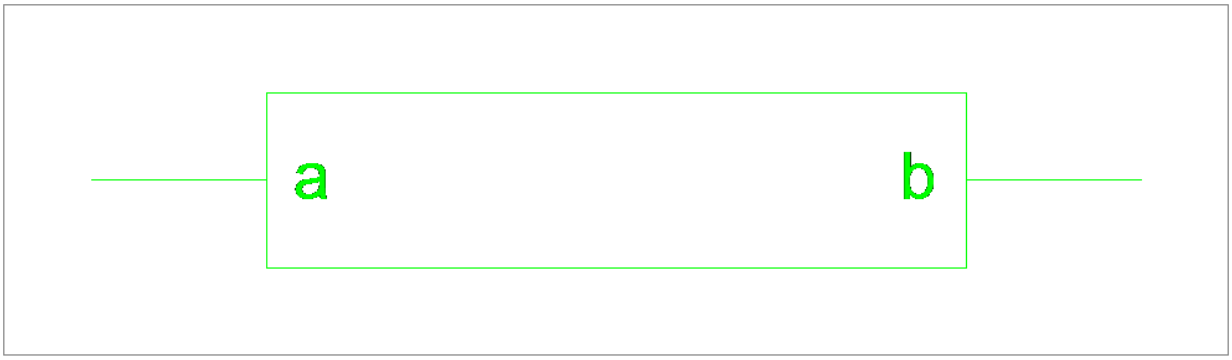


3) inverter.vhd

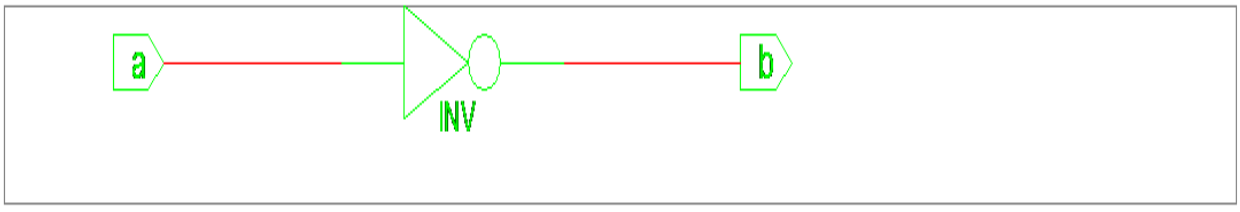
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity inverter is
```

```
Port ( a : in  STD_LOGIC;
       b : out  STD_LOGIC);
end inverter;
architecture Behavioral of inverter is
begin
b <= not a;
end Behavioral;
```

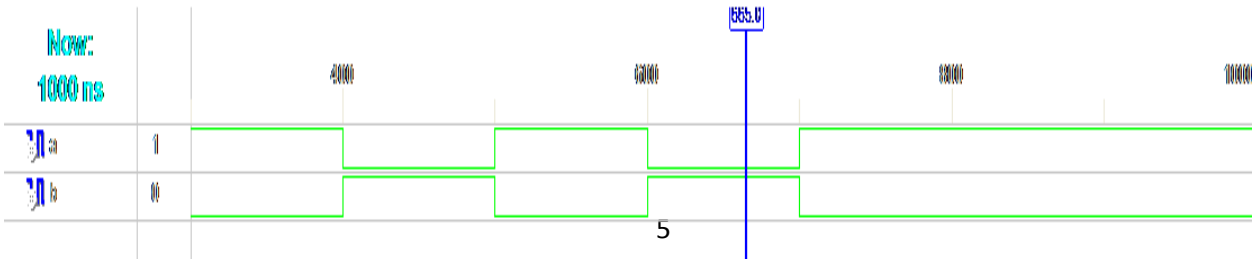
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



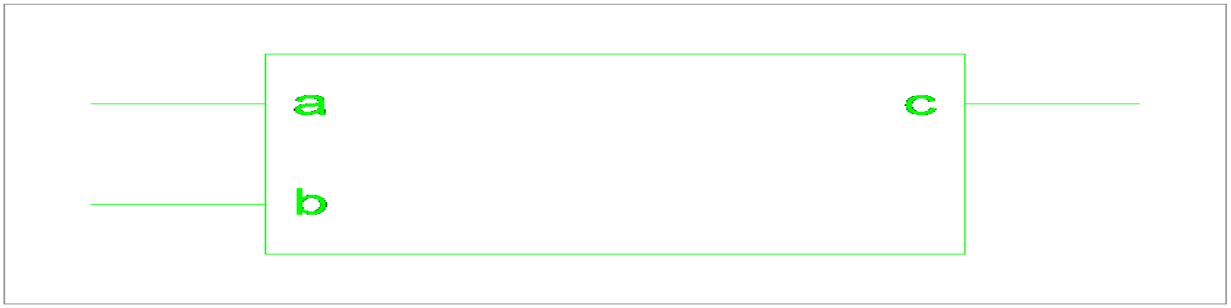
(c) SIMULATION WAVEFORM



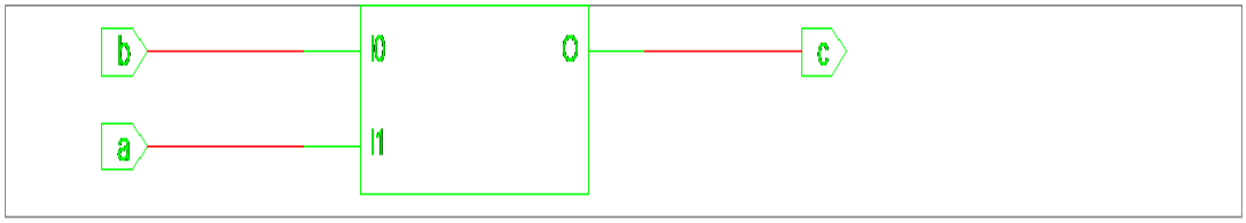
4) xor_gate.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity xor_gate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          c : out  STD_LOGIC);
end xor_gate;
architecture Behavioral of xor_gate is
begin
c <= a xor b;
end Behavioral;
```

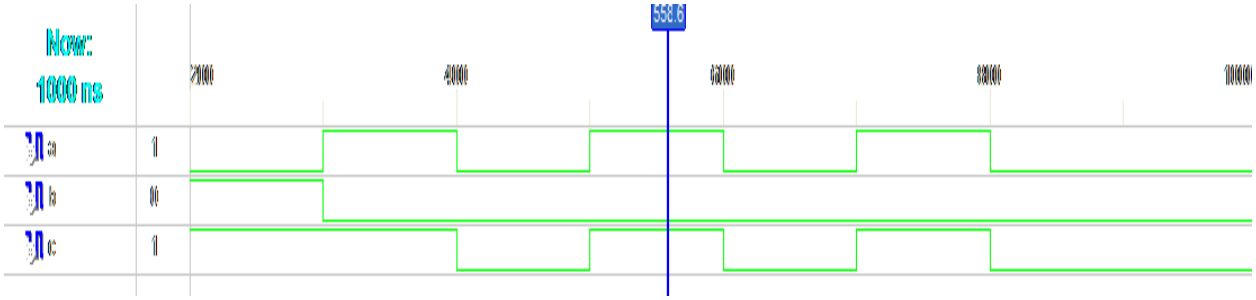
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM



EXPERIMENT NO. II

AIM:-

Synthesis & simulation of D-flip flop using VHDL.

PROGRAM:-

d_flipflop.vhd

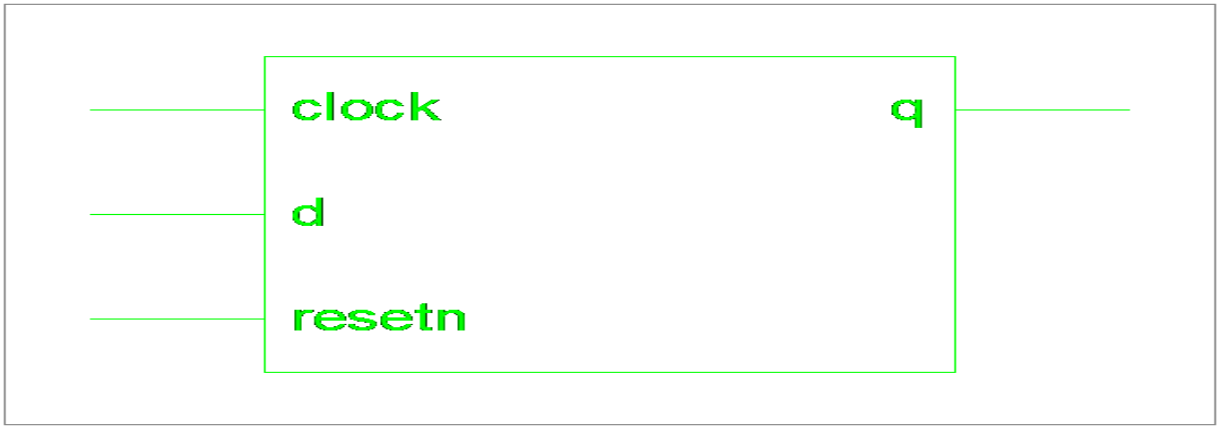
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity d_flipflop is
   Port ( d : in  STD_LOGIC;
          resetn : in  STD_LOGIC;
          clock : in  STD_LOGIC;
          q : out  STD_LOGIC);
end d_flipflop;
architecture Behavioral of d_flipflop is
begin
process(resetn,clock)
begin
if resetn='0'then
      q <= '0';
 elsif clock'event and clock ='1' then
       q <= d;
```

end if;

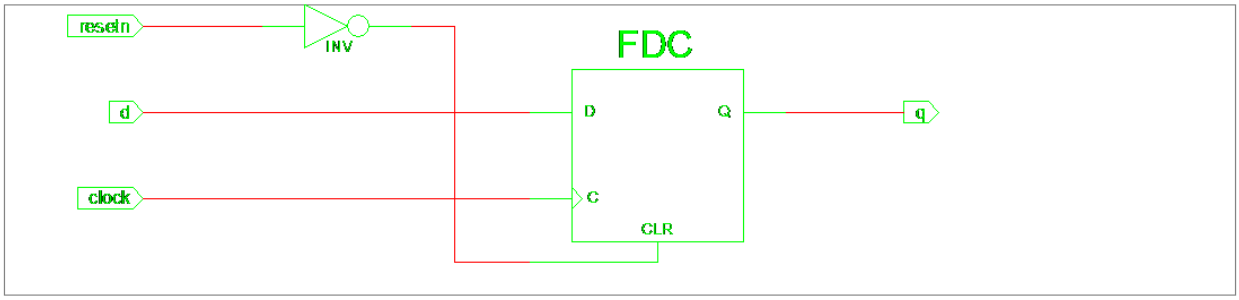
end process;

end Behavioral;

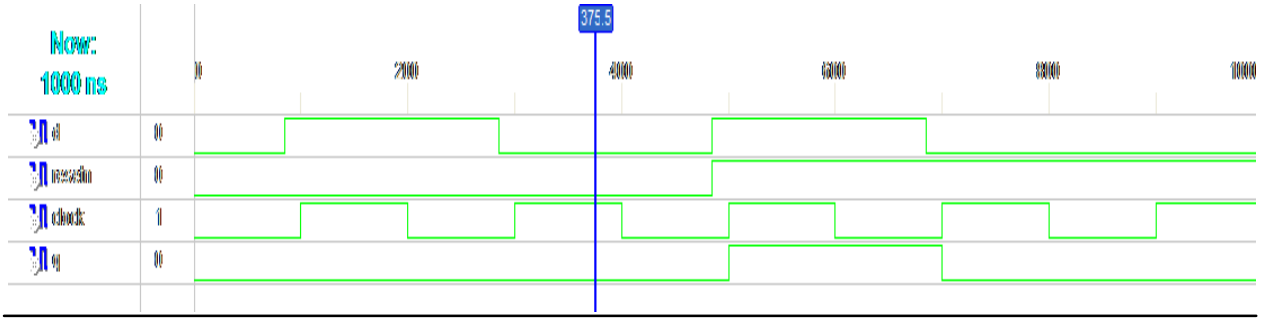
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM



EXPERIMENT NO. III

AIM:-

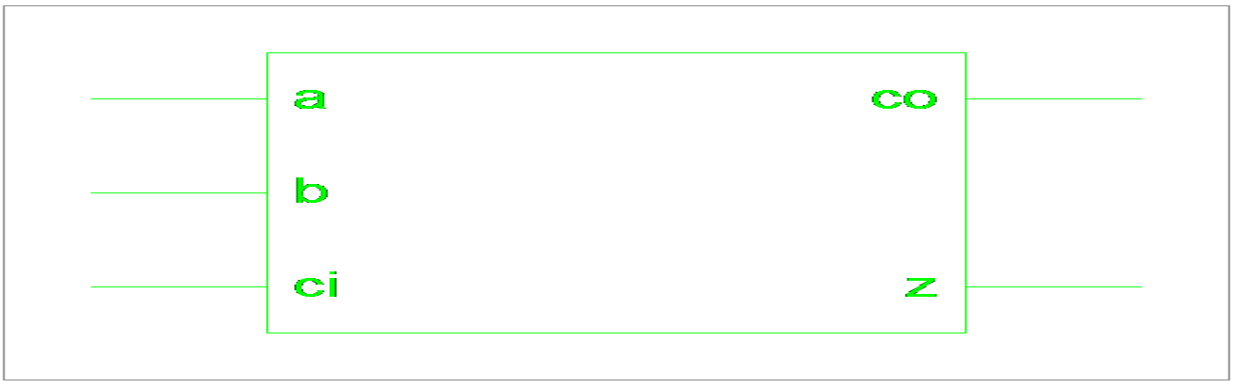
Synthesis & simulation of full-adder using VHDL.

PROGRAM:-

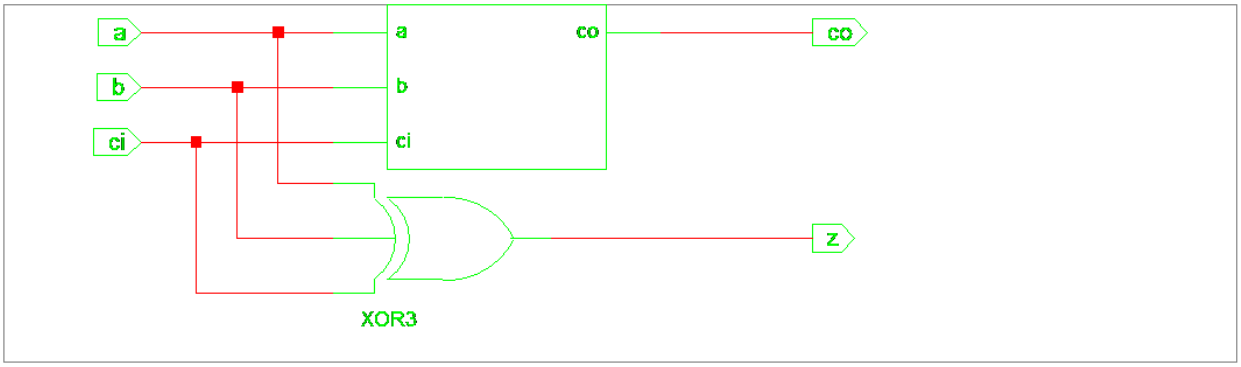
fulladd.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
 entity fulladd is
   Port ( a : in  STD_LOGIC;
        b : in  STD_LOGIC;
        ci : in  STD_LOGIC;
        z : out  STD_LOGIC;
        co : out  STD_LOGIC);
end fulladd;
architecture Behavioral of fulladd is
begin
z <= a xor b xor ci;
co <= (a and b) or (b and ci) or (ci and a);
end Behavioral;
```

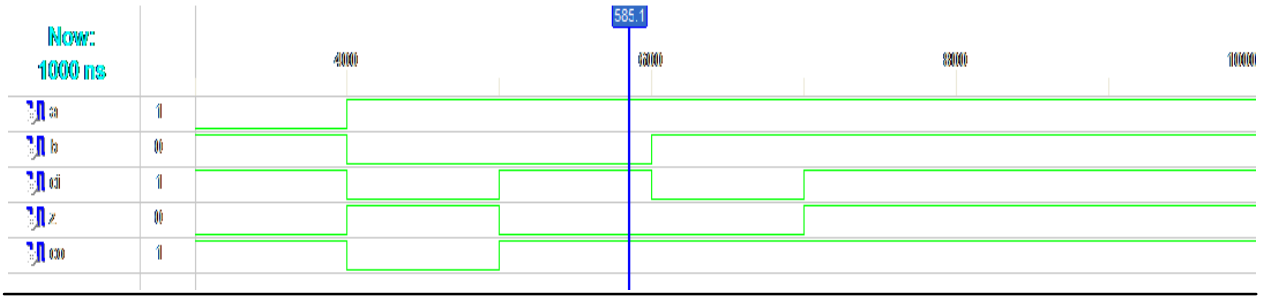
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM



EXPERIMENT NO. IV

AIM:-

Synthesis & simulation of multiplexer (4:1) using VHDL.

PROGRAM:-

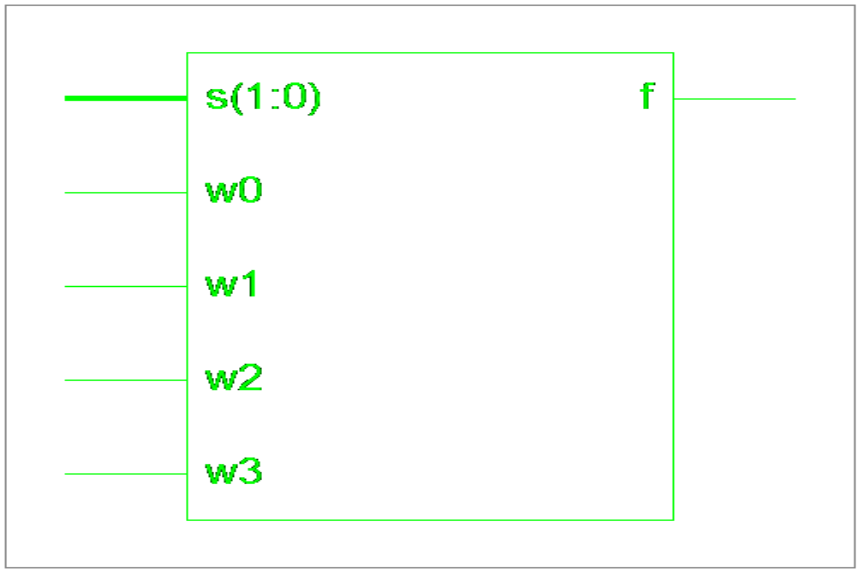
mux4to1.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
 entity mux4to1 is
   Port ( w0 : in  STD_LOGIC;
        w1 : in  STD_LOGIC;
        w2 : in  STD_LOGIC;
        w3 : in  STD_LOGIC;
        s : in  STD_LOGIC_VECTOR (1 downto 0);
        f : out  STD_LOGIC);
end mux4to1;
architecture Behavioral of mux4to1 is
begin
    with s select
                f <=  w0 when "00",
                      w1 when "01",
                      w2 when "10",
```

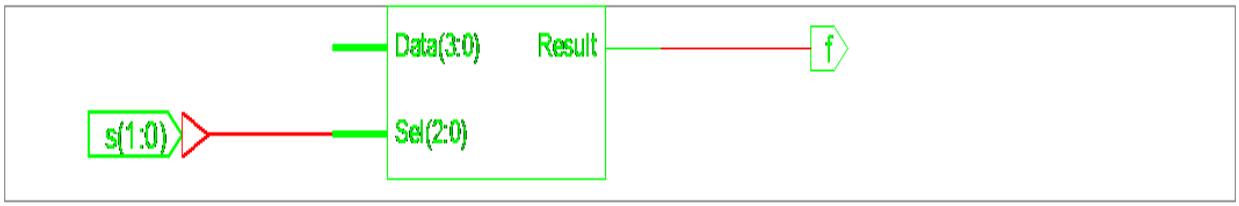
w3 when others;

end Behavioral;

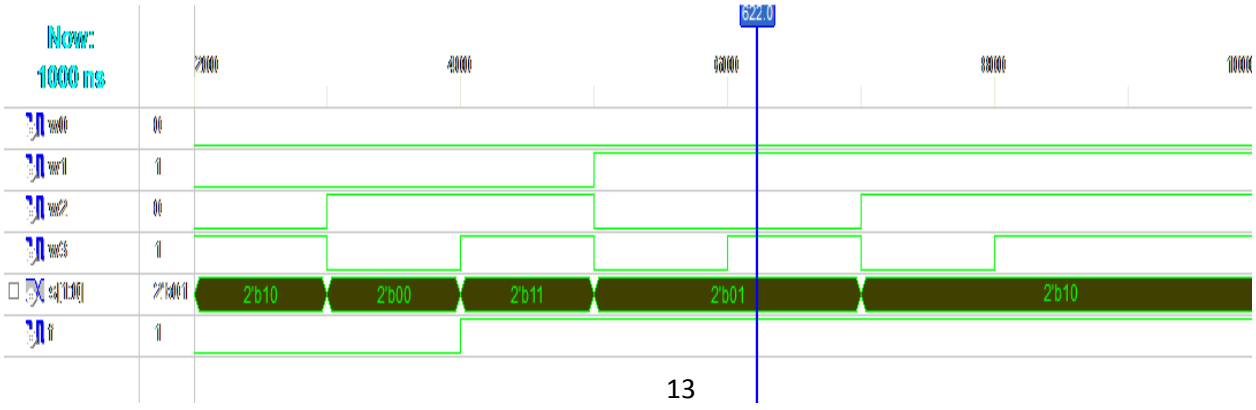
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM



EXPERIMENT NO. V

AIM:-

Synthesis & simulation of decoder (2:4) using VHDL.

PROGRAM:-

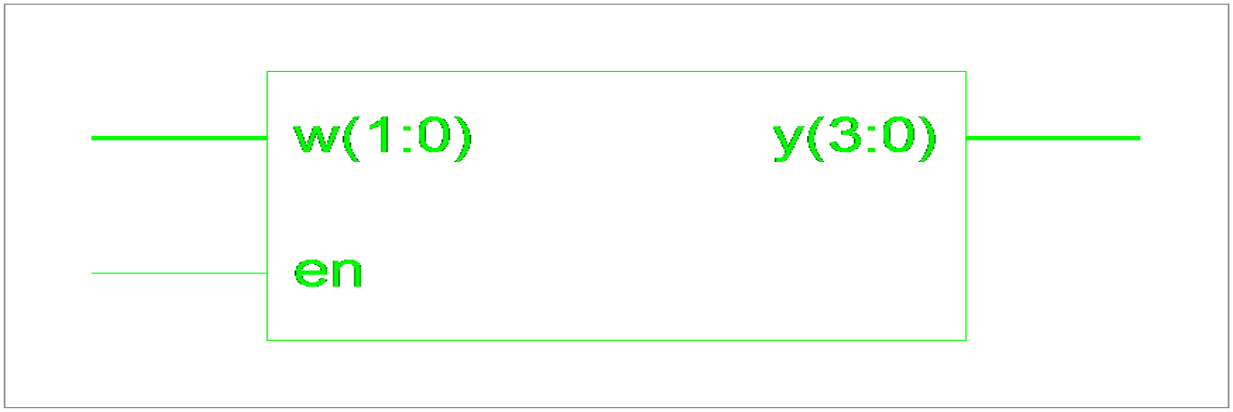
dec2to4.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity dec2to4 is
   Port ( w : in  STD_LOGIC_VECTOR (1 downto 0);
          en : in  STD_LOGIC;
          y : out  STD_LOGIC_VECTOR (3 downto 0));
end dec2to4;
architecture Behavioral of dec2to4 is
signal enw : STD_LOGIC_VECTOR (2 downto 0);
begin
enw <= en & w;
  with enw select
    y <= "1000" when "100",
          "0100" when "101",
          "0010" when "110",
           "0001" when "111",
```

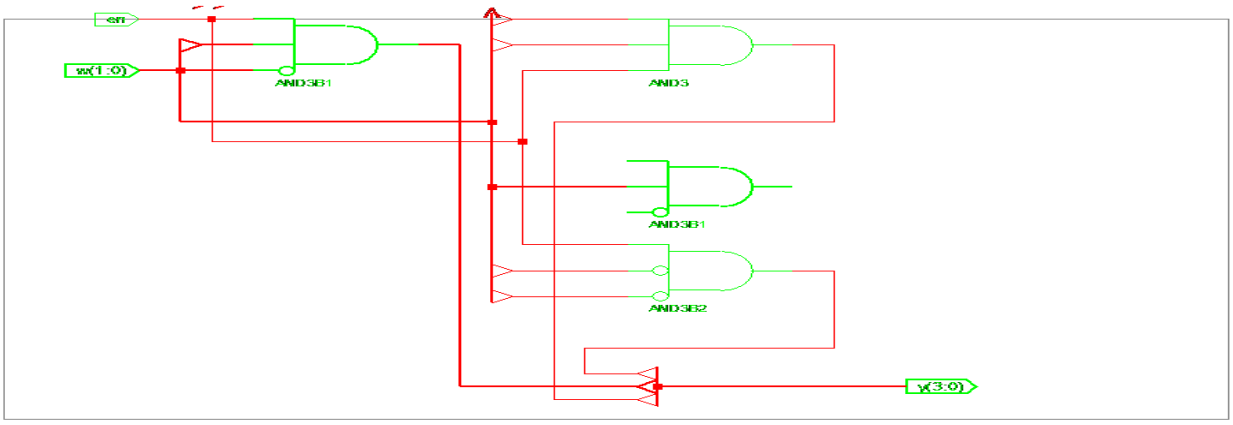
"0000" when others;

end Behavioral;

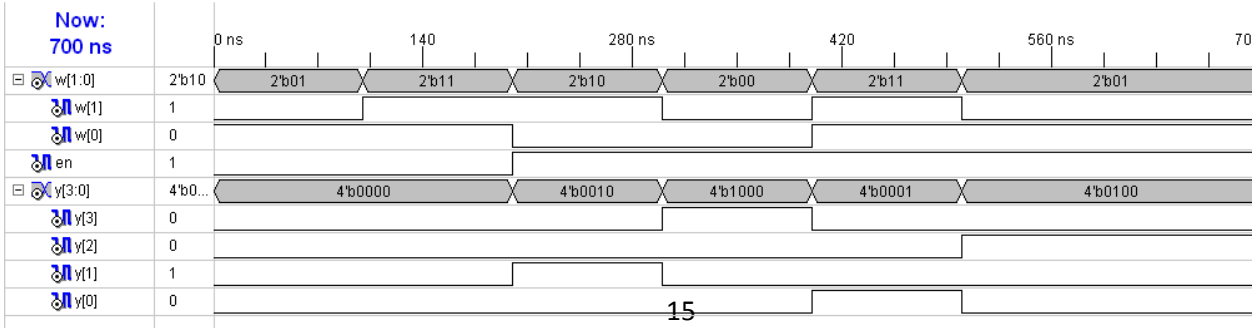
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM



EXPERIMENT NO. VI

AIM:-

Synthesis & simulation of 8-bit adder using VHDL.

PROGRAM:-

adder8.vhd

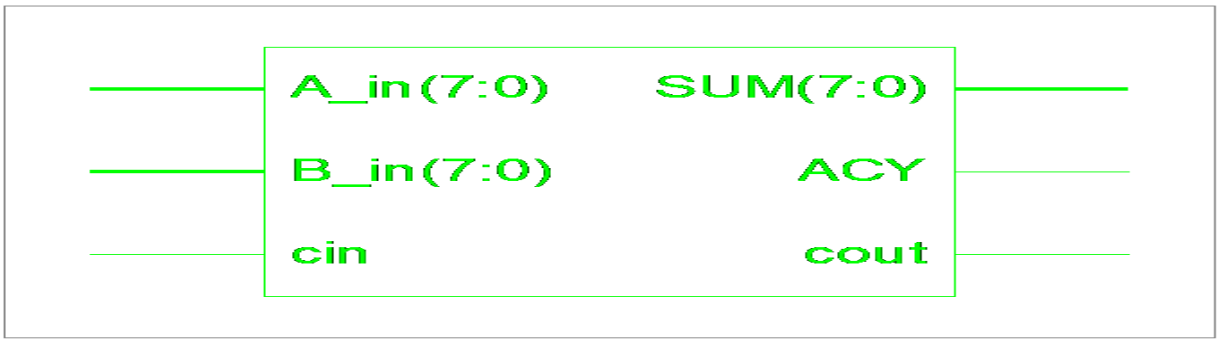
```
library ieee;
use ieee.std_logic_1164.all;
entity adder8 is
 generic(n:integer := 7 ) ;
 port( A_in : in std_logic_vector(n downto 0 );
     B_in : in std_logic_vector(n downto 0 );
     SUM  : out std_logic_vector(n downto 0 );
     cin  : in std_logic ;
     ACY  : out std_logic ;
     cout : out std_logic ) ;
end adder8 ;
architecture struct of adder8 is
  component FULLADD
  port(
    Z                : out   STD_LOGIC;
    CO               : out   STD_LOGIC;
    A                : in   STD_LOGIC;
    B                : in    STD_LOGIC;
    CI               : in    STD_LOGIC);
```

```
end component;
signal A_in_s :std_logic_vector(n downto 0 );
signal B_in_s :std_logic_vector(n downto 0 );
signal SUM_s  :std_logic_vector(n downto 0 );
 signal cin_s  :std_logic_vector(n downto 0 );
signal cout_s :std_logic_vector(n downto 0 );
   begin
  cin_s(0)  <= cin ;
    ADD0: FULLADD port map (  Z   => SUM_s(0) ,
                CO  => cout_s(0) ,
                A   => A_in_s(0) ,
                B   => B_in_s(0) ,
                CI  => cin_s(0)) ;
  G1 : for i in n downto 1 generate
      ADD1: FULLADD port map( Z   => SUM_s(i) ,
                CO  => cout_s(i) ,
                A   => A_in_s(i) ,
                B   => B_in_s(i) ,
                CI  => cout_s(i-1)) ;
   end generate ;
SUM <= SUM_S ;
cout <= cout_s(7) ;
A_in_s <= A_in ;
B_in_s <= B_in ;
```

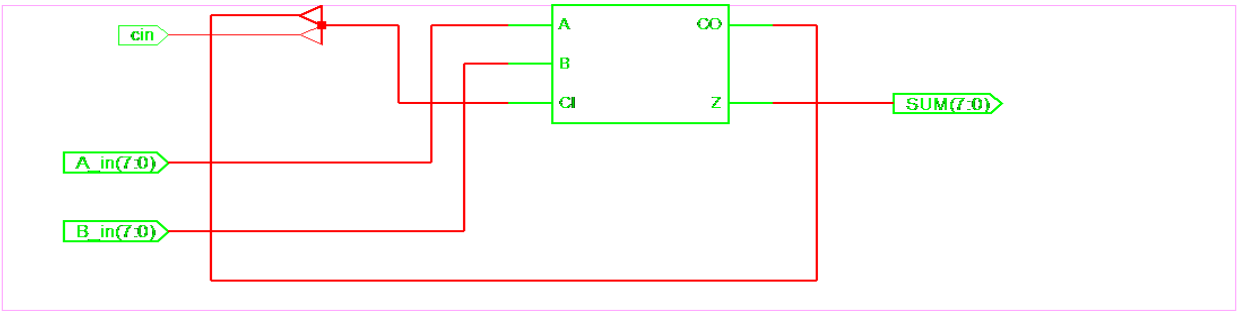
```
ACY <= cout_s(3) ;

end struct ;
```

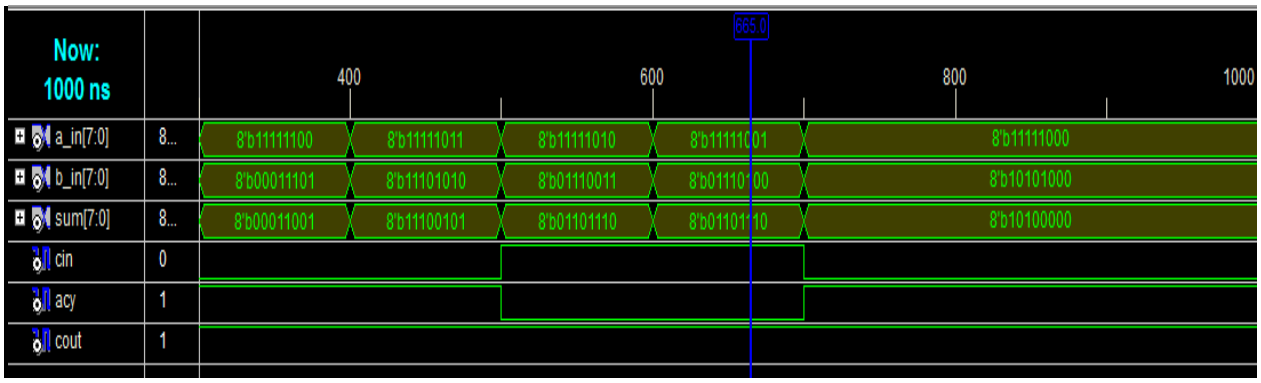
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM



EXPERIMENT NO. VII

AIM:-

Synthesis & simulation of 8-bit ALU using VHDL.

PROGRAM:-

alu.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all ;
entity alu is
    generic ( n : integer := 7 ;
          s : integer := 3) ;
    port(dat_a  : in std_logic_vector(n downto 0);
       dat_b  : in std_logic_vector(n downto 0);
       dat_out: out std_logic_vector(n downto 0);
       cout   : out std_logic;
       ACY    : out std_logic;
       SIGN   : out std_logic;
       PARITY : out std_logic;
       ZERO   : out std_logic;
       cin    : in std_logic;
       op_sel : in std_logic_vector(s downto 0));
end alu;
```

```
architecture behav of alu is
  signal nd    : std_logic_vector(n downto 0 );
  signal A_in_s : std_logic_vector(n downto 0 );
signal B_in_s  : std_logic_vector(n downto 0 );
  signal dat_b_n_s : std_logic_vector(n downto 0 );
  signal dat_a_n_s : std_logic_vector(n downto 0 );
  signal B_in_s_r  : std_logic_vector(n downto 0 );
  signal A_in_s_r  : std_logic_vector(n downto 0 );
 signal SUM_s : std_logic_vector(n downto 0 );
 signal cout_s : std_logic ; _vector(n downto 0 );
component ADDER8
 port( A_in : in std_logic_vector(n downto 0 );
      B_in : in std_logic_vector(n downto 0 );
      SUM  : out std_logic_vector(n downto 0 );
      cin  : in std_logic ;
      ACY  : out std_logic ;
      cout : out std_logic ) ;
end component;
 begin
A_in_s <= dat_a ;
B_in_s <= dat_b ;
dat_b_n_s <= not(dat_b) ;
dat_a_n_s <= not(dat_a) ;
A8_1 : adder8 port map (A_in => A_in_s_r ,
```

```
    B_in => B_in_s_r ,
    SUM  => SUM_s ,
    cin  => cin ,
    ACY  => ACY ,
    cout => cout_s) ;
process(dat_a,dat_b,op_sel,A_in_s,B_in_s,dat_b_n_s,SUM_s)
variable dat_xor_res:std_logic_vector(n downto 0) ;
variable dat_shr_res:std_logic_vector(n downto 0) ;
variable dat_xnor_res:std_logic_vector(n downto 0);
variable dat_and_res:std_logic_vector(n downto 0);
variable dat_nand_res:std_logic_vector(n downto 0);
variable dat_or_res:std_logic_vector(n downto 0);
variable dat_nor_res:std_logic_vector(n downto 0);
variable dat_out_var:std_logic_vector(n downto 0);
variable op_sel_int : integer := 0 ;
begin
 op_sel_int := CONV_INTEGER(op_sel);
for i in n downto 0 loop
 dat_xor_res(i) := dat_a(i) xor dat_b(i);
end loop;
for i in n-1 downto 0 loop
 dat_shr_res(i) := dat_a(i+1) ;
end loop;
```

```
 dat_shr_res(7) := '0' ;
for i in n downto 0 loop
 dat_xnor_res(i) := not (dat_a(i) xor dat_b(i));
end loop;
for i in n downto 0 loop
 dat_and_res(i) := dat_a(i) and dat_b(i);
end loop;
  for i in n downto 0 loop
   dat_nand_res(i) := not (dat_a(i) and dat_b(i));
  end loop;
  for i in n downto 0 loop
   dat_or_res(i) := dat_a(i) or dat_b(i);
  end loop;
  for i in n downto 0 loop
   dat_nor_res(i) := not (dat_a(i) or dat_b(i));
  end loop;
case op_sel_int is
  when 1 =>
   dat_out_var := dat_xor_res ;
   B_in_s_r <= B_in_s ;
   A_in_s_r <= A_in_s ;
   cout <= '0' ;
  when 2 =>
   dat_out_var := dat_xnor_res ;
```

```
 B_in_s_r <= B_in_s ;
 A_in_s_r <= A_in_s ;
 cout <= '0' ;
when 3 =>
 dat_out_var := dat_and_res ;
 B_in_s_r <= B_in_s ;
 A_in_s_r <= A_in_s ;
 cout <= '0' ;
when 4 =>
dat_out_var := dat_nand_res ;
B_in_s_r <= B_in_s ;
A_in_s_r <= A_in_s ;
cout <= cout_s ;
when 5 =>
dat_out_var := dat_or_res ;
B_in_s_r <= B_in_s ;
A_in_s_r <= A_in_s ;
cout <= '0' ;
when 6 =>
dat_out_var := dat_nor_res ;
B_in_s_r <= B_in_s ;
A_in_s_r <= A_in_s ;
cout <= '0' ;
```

```
when 0 =>  -- add A + B
 B_in_s_r <= B_in_s ;
 dat_out_var := SUM_s ;
 A_in_s_r <= A_in_s ;
 cout <= cout_s ;
when 7 =>  -- sub A - B
 B_in_s_r <= dat_b_n_s ;
 dat_out_var := SUM_s ;
 A_in_s_r <= A_in_s ;
 cout <= cout_s ;
when 8 => -- not not(A)
 B_in_s_r <= "00000000" ;
 dat_out_var := SUM_s ;
 A_in_s_r <= dat_a_n_s ;
 cout <= cout_s ;
when 9 =>  -- inr REG + 1
 B_in_s_r <= B_in_s ;
 dat_out_var := SUM_s ;
 A_in_s_r <= "00000000" ;
 cout <= cout_s ;
when 10 =>  -- dcr REG - 1
 B_in_s_r <= "11111110" ;
 dat_out_var := SUM_s ;
 A_in_s_r <= B_in_s;
```

```
 cout <= cout_s ;
when 11 =>  -- shl A <- shl(A)
 B_in_s_r <= A_in_s ;
 dat_out_var := SUM_s ;
 A_in_s_r <= A_in_s;
 cout <= cout_s ;
when 12 =>  -- zero/clear
 B_in_s_r <= "00000000" ;
 dat_out_var := SUM_s ;
 A_in_s_r <= "00000000";
 cout <= cout_s ;
  when 13 =>
   dat_out_var := dat_shr_res ; -- SHR(A)
   B_in_s_r <= B_in_s ;
   A_in_s_r <= A_in_s ;
   cout <= cout_s ;
  when others =>
   dat_out_var := SUM_s ;
   B_in_s_r <= "00000000" ;
   A_in_s_r <= "00000000";
   cout <= cout_s ;
end case ;
dat_out <= dat_out_var ;
```

PARITY <= dat_out_var(0) xor dat_out_var(1) xor dat_out_var(2) xor dat_out_var(3) xor dat_out_var(4) xor dat_out_var(5) xor dat_out_var(6) xor dat_out_var(7) ;

nd <= dat_out_var ;

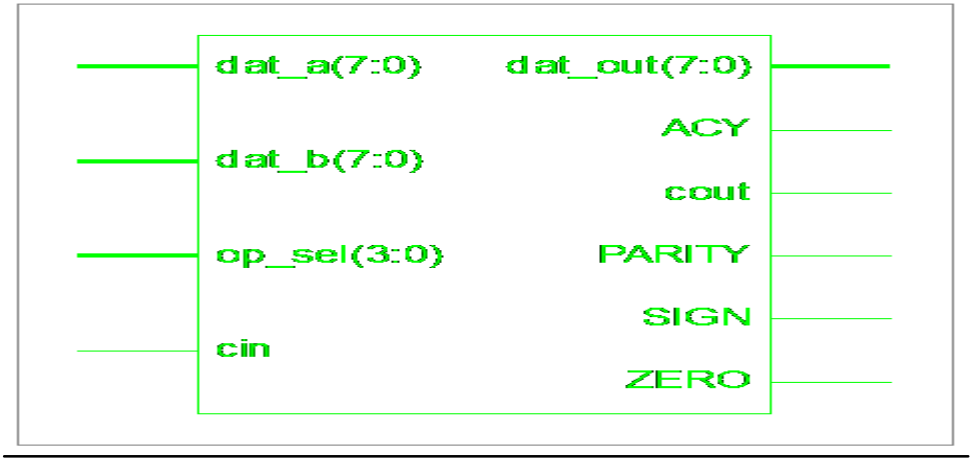
end process;

ZERO <= not (nd(0) and nd(1) and nd(2) and nd(3) and nd(4) and nd(5) and nd(6) and nd(7)) ;

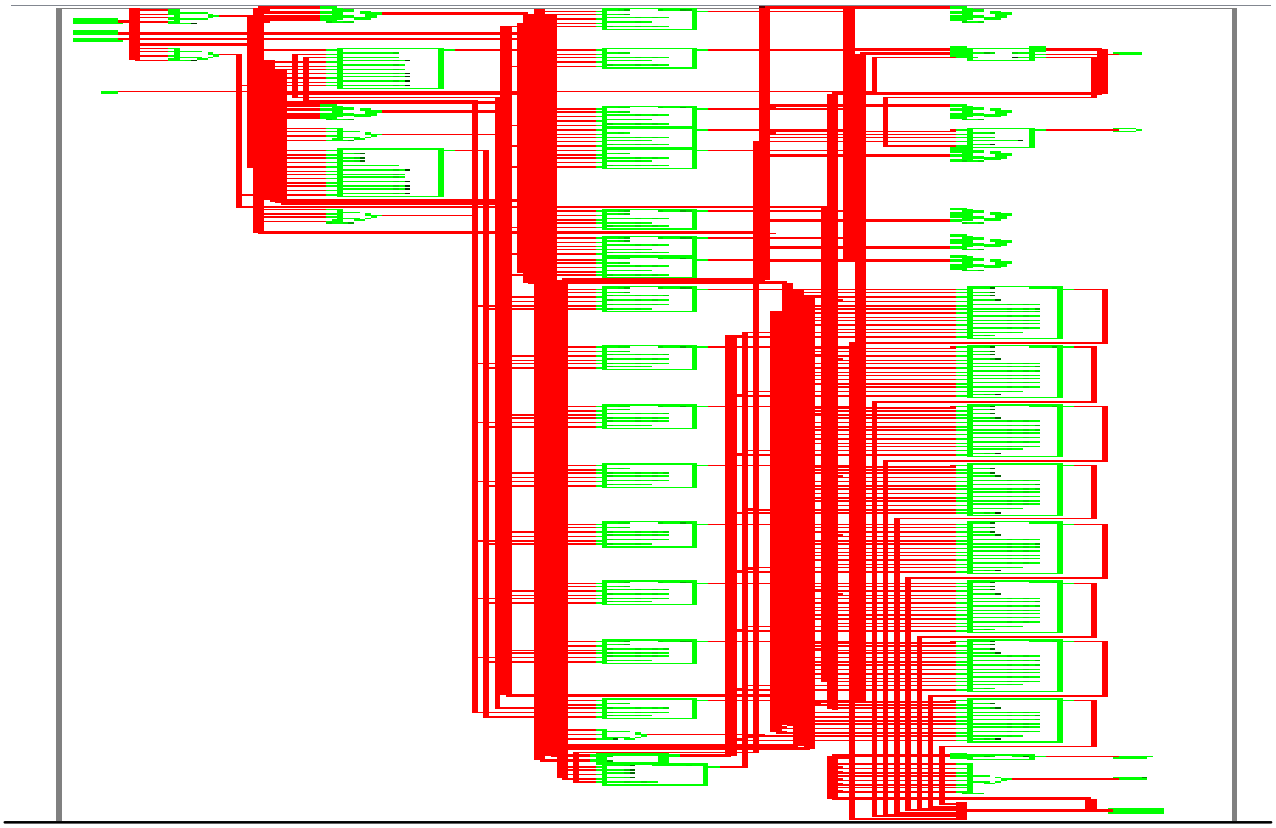
SIGN <= nd(7) ;

end behav;

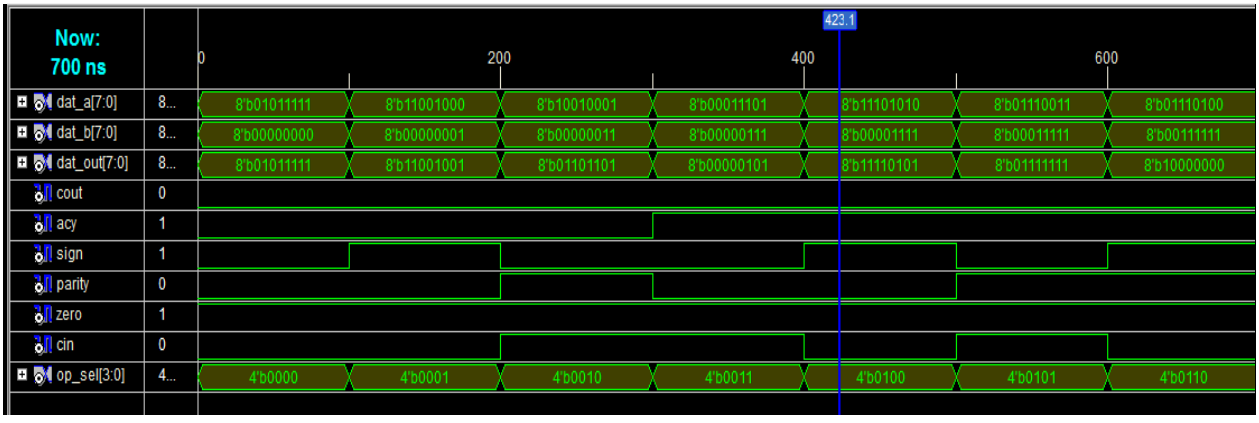
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM



EXPERIMENT NO. VIII

AIM:-

Synthesis & simulation of FSM using VHDL.

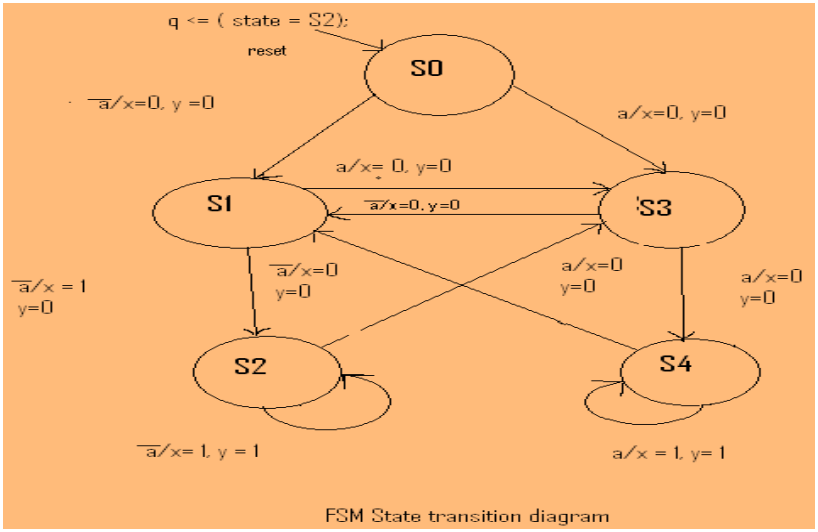
PROGRAM:-

FSM.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity FSM is
   Port ( clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          a : in  STD_LOGIC;
          x : out  STD_LOGIC;
          y : out  STD_LOGIC);
end FSM;
architecture Behavioral of FSM is
type statetype is (S0, S1, S2, S3, S4);
signal state, nextstate: statetype;
begin
    -- state register
          process (clk, reset) begin
          if reset = '1' then state <= S0;
          elsif clk'event and clk = '1' then state <= nextstate;
```

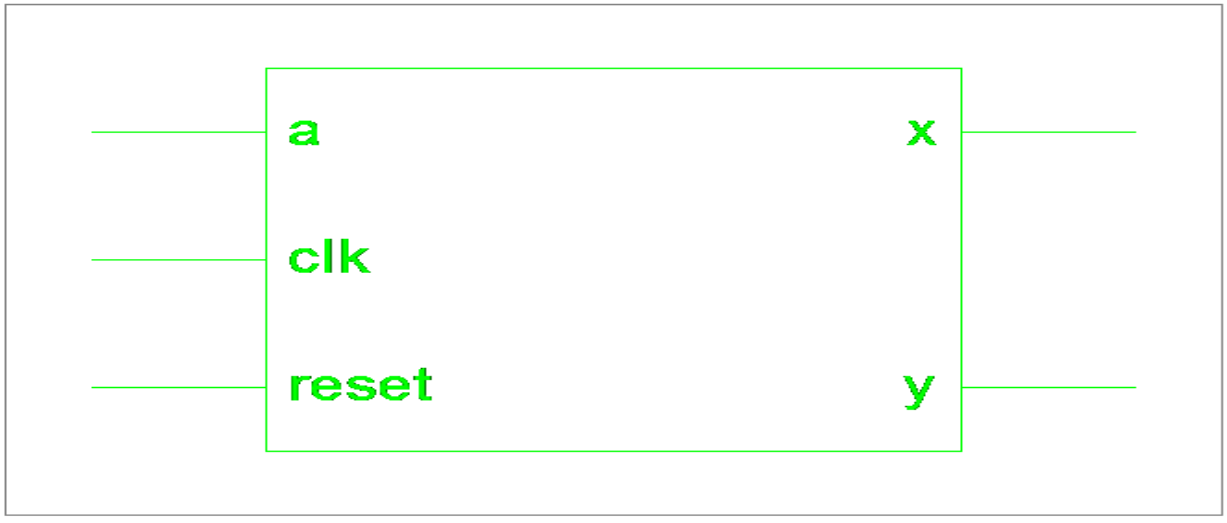
```
            end if;
            end process;
 -- next state logic
        process (state, a) begin
        case state is
           when S0 => if a = '1' then nextstate <= S3;
                        else nextstate <= S1;
                        end if;
       when S1 => if a = '1' then nextstate <= S3;
                        else nextstate <= S2;
                       end if;
      when S2 => if a = '1' then nextstate <= S3;
                        else  nextstate <= S2;
                        end if;
     when S3 => if a = '1' then nextstate <= S4;
                        else         nextstate <= S1;
                        end if;
     when S4 => if a = '1' then nextstate <= S4;
                        else         nextstate <= S1;
                        end if;
     when others =>         nextstate <= S0;
  end case;
end process;
--output logic
```

```
x <= '1' when ((state = S1 or state = S2) and a = '1') or
        ((state = S3 or state = S4) and a = '1')
    else '1';
y <= '1' when (state = S2 and a = '1') or (state = S4 and a='1')
        else '1';
end Behavioral;
```

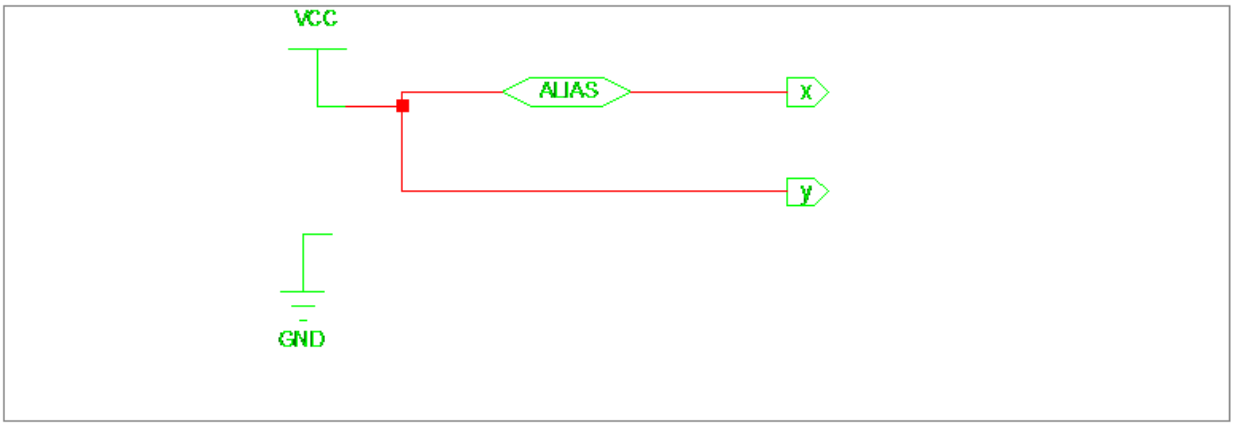


FSM State transition diagram

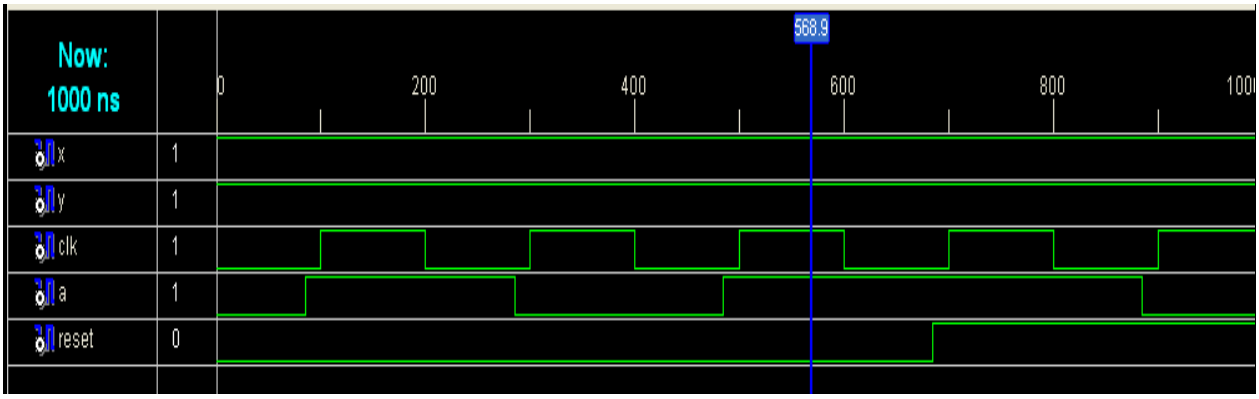
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM



EXPERIMENT NO. IX

AIM:-

Synthesis & simulation of 8- bit multiplier using VHDL.

PROGRAM:-

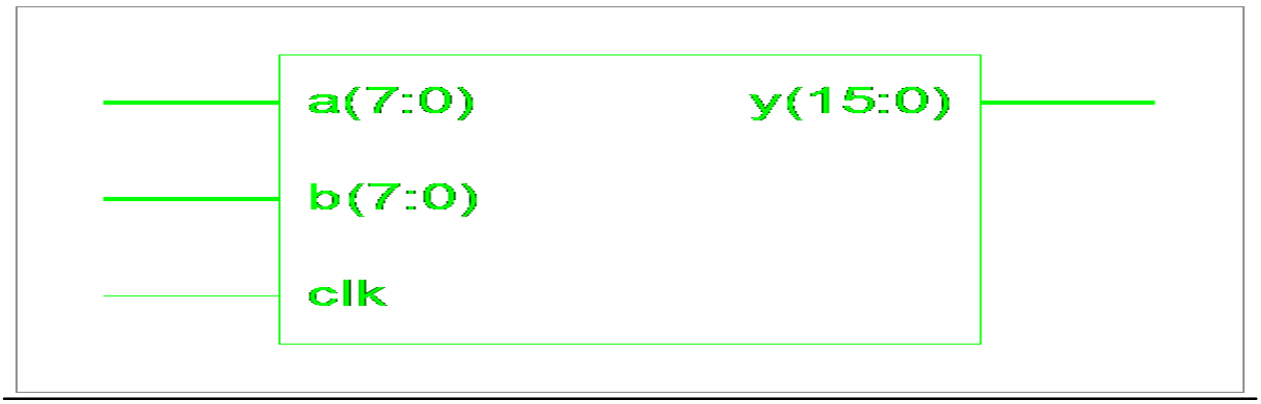
MULTIPLIER_8.VHD

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity multiplier_8 is
   Port ( clk : in  STD_LOGIC;
        a : in  STD_LOGIC_VECTOR (7 downto 0);
        b : in  STD_LOGIC_VECTOR (7 downto 0);
        y : out  STD_LOGIC_VECTOR (15 downto 0));
end multiplier_8;
architecture Behavioral of multiplier_8 is
begin
p1: process(clk)
variable n0,n1,n2,n3,n4,n5,n6,n7: std_logic_vector(15 downto 0);
variable m: std_logic_vector(7 downto 0);
begin
m:="00000000";
if clk'event and clk='1' then
if a(0)<='0' then
```

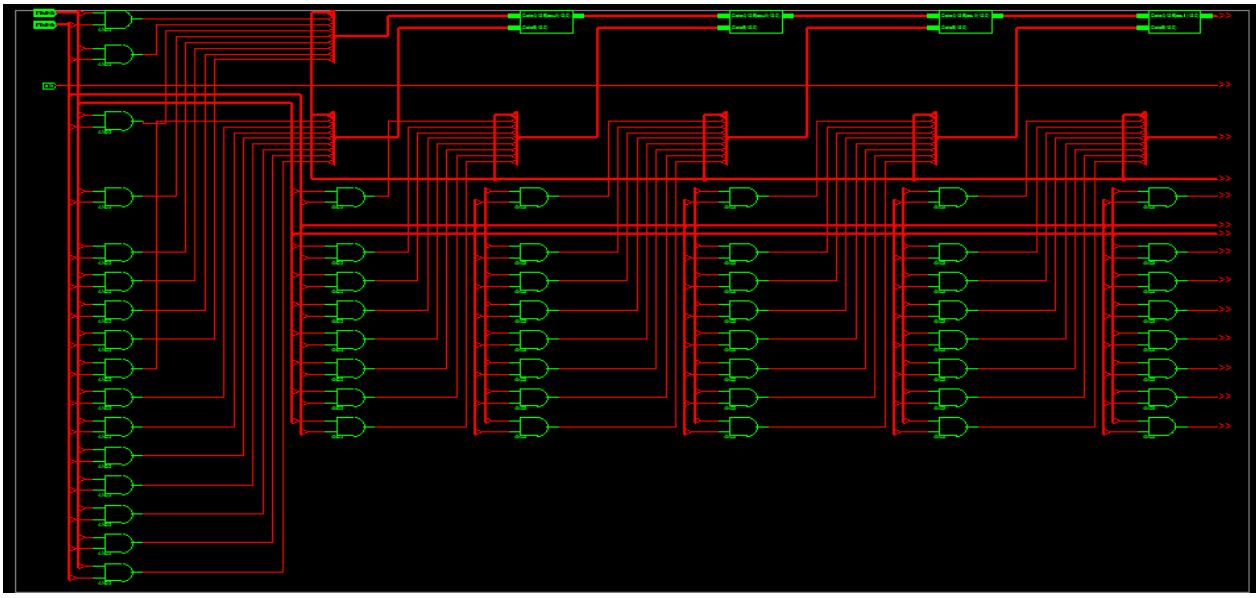
```
n0:="0000000000000000";
else
n0:=m&b;
end if;
if a(1)<='0' then
n1:="0000000000000000";
else
n1:=m(7 downto 1)&b&m(0);
end if;
if a(2)<='0' then
n2:="0000000000000000";
else
n2:=m(7 downto 2)&b&m(1 downto 0);
end if;
if a(3)<='0' then
n3:="0000000000000000";
else
n3:=m(7 downto 3)&b&m(2 downto 0);
end if;
if a(4)<='0' then
n4:="0000000000000000";
else
n4:=m(7 downto 4)&b&m(3 downto 0);
end if;
if a(5)<='0' then
```

```
n5:="0000000000000000";
else
n5:=m(7 downto 5)&b&m(4 downto 0);
end if;
if a(6)<='0' then
n6:="0000000000000000";
else
n6:=m(7 downto 6)&b&m(5 downto 0);
end if;
if a(7)<='0' then
n7:="0000000000000000";
else
n7:=m(7)&b&m(6 downto 0);
end if;
y<=n0+n1+n2+n3+n4+n5+n6+n7;
end if;
end process p1;
 end Behavioral;
```

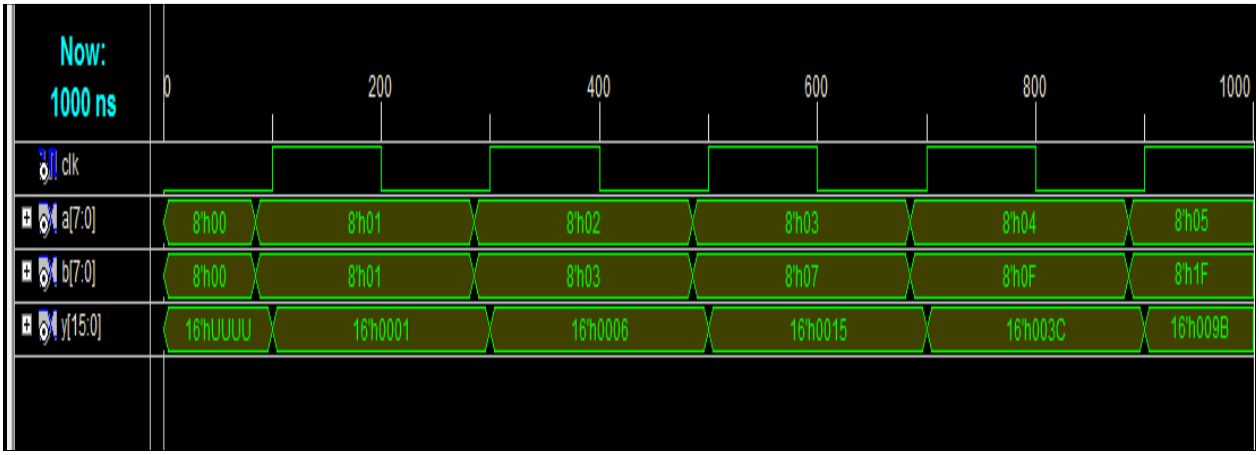
(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM



EXPERIMENT NO. X

AIM:-

Synthesis & simulation of using 8-bit up/down counter using VHDL.

PROGRAM:-

BIT8CNT.VHD

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity BIT8CNT is
  Port ( EN : in  STD_LOGIC;
       CLR : in  STD_LOGIC;
       CLK : in  STD_LOGIC;
       CE : in std_logic;
       Carry : out std_logic;
       COUNTER : out  STD_LOGIC_VECTOR (7 downto 0);
       UD : in std_logic);
end BIT8CNT;
architecture Behavioral of BIT8CNT is
component BIT4CNT
        PORT ( EN : in  STD_LOGIC;
               CLK : in  STD_LOGIC;
               CLR : in  STD_LOGIC;
```

```
        CE : in std_logic;
        CARRY: out std_logic;
        CNT : out  std_logic_vector(3 downto 0);
        UD : in std_logic);
end component;
signal C0_IN,C0_OUT : std_logic;
begin
u0: BIT4CNT Port map(
CLK=>CLK,CLR=>CLR,CE=>CE,EN=>EN,CARRY=>C0_OUT,CNT=>COUNTER(
3 downto 0),UD=>UD);
u1: BIT4CNT Port map(
CLK=>CLK,CLR=>CLR,CE=>CE,EN=>C0_IN,CARRY=>Carry,CNT=>COUNTER(
7 downto 4),UD=>UD);
C0_IN <= C0_OUT AND EN;
end Behavioral;
```

BIT4CNT.VHD

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity BIT4CNT is
   Port ( EN : in  STD_LOGIC;
        CLK : in  STD_LOGIC;
        CLR : in  STD_LOGIC;
```

```
        CE : in std_logic;
        CARRY: out std_logic;
        CNT : out  std_logic_vector(3 downto 0);
        UD : in std_logic);
end BIT4CNT;
architecture Behavioral of BIT4CNT is
component EX3
          Port( T: in std_logic;
                        CLR : in std_logic;
                        CLK : in std_logic;
                        CE : in std_logic;
                        Q: out std_logic);
end component;
signal c0,c1,c2: std_logic;
signal Q :std_logic_vector(3 downto 0);
begin
u0 : EX3 Port map (CLK=>CLK,CLR=>CLR,CE =>CE,T=>EN,Q=>Q(0));
u1 : EX3 Port map (CLK=>CLK,CLR=>CLR,CE =>CE,T=>c0,Q=>Q(1));
u2 : EX3 Port map (CLK=>CLK,CLR=>CLR,CE =>CE,T=>c1,Q=>Q(2));
u3 : EX3 Port map (CLK=>CLK,CLR=>CLR,CE =>CE,T=>c2,Q=>Q(3));
          process(EN,CLK,CE,UD,Q,c0,c1,c2)
          begin
          if(UD = '1') then
                    c0<=EN and Q(0);
```

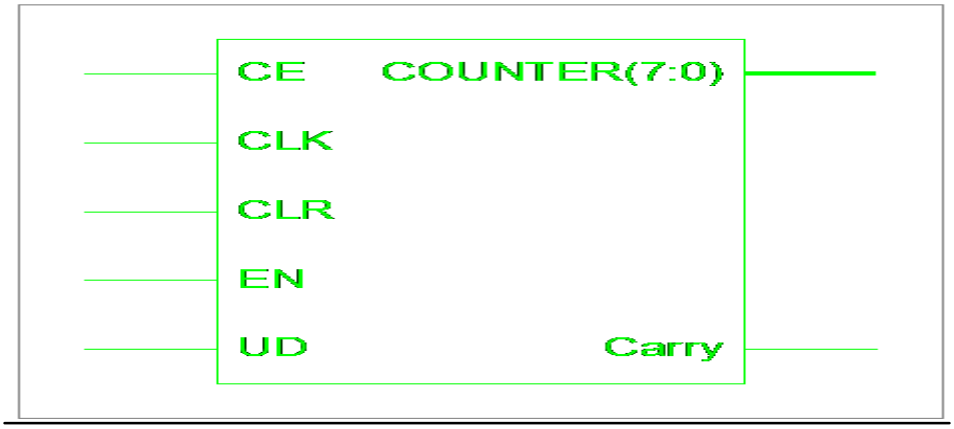
```
            c1<=c0 and Q(1);
            c2<=c1 and Q(2);
            CARRY<=c2 and Q(3);
        else
            c0<=EN and not Q(0);
            c1<=c0 and not Q(1);
            c2<=c1 and not Q(2);
            CARRY<=c2 and not Q(3);
end if;
        CNT <= Q;
        end process;
end Behavioral;
```

EX3.VHD

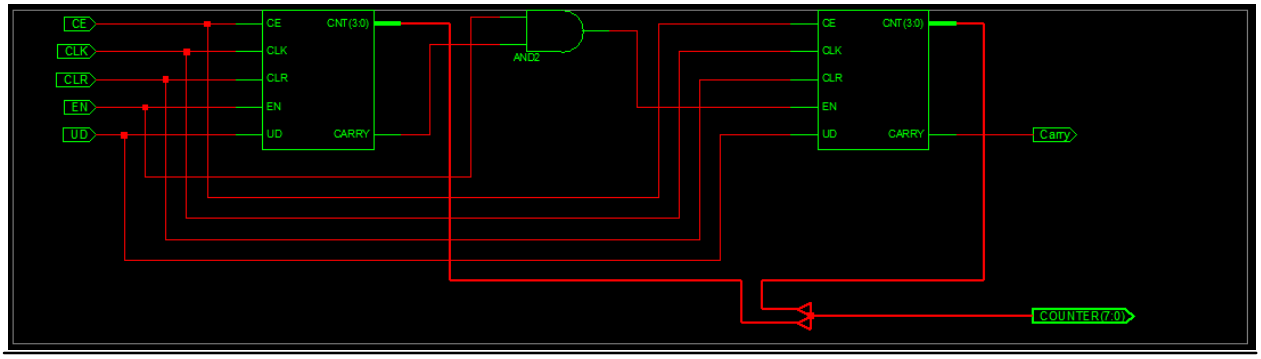
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
 entity EX3 is
   Port ( T : in  STD_LOGIC;
       CLR : in  STD_LOGIC;
       CLK : in  STD_LOGIC;
       CE : in std_logic;
       Q : out  STD_LOGIC);
```

```
end EX3;
architecture Behavioral of EX3 is
begin
        process(CLK,CLR)
        variable temp : std_logic := '0';
        begin
        if(CLR = '1') then
        temp := '0';
        elsif(CLK'EVENT and CLK = '1') then
                if(CE = '1') then
                        if(T = '0') then
                        temp := temp;
                        else
                        temp := not temp;
                        end if;
                end if;
        end if;
        Q <= temp;
        end process;
end Behavioral;
```

(a) RTL SCHEMATIC



(b) SOFTWARE GENERATED INTERNAL ARCHITECTURE



(c) SIMULATION WAVEFORM