Name :- Md Shahbaz Alam

Roll number :- 26

Stream :- CSE.

Enrollment number :- 12020009001123

paper name :- Computer Organization & Architecture laboratory.

Subject Code :- PCC-CS492

Section :- D

Assignment - I

Assignment - I

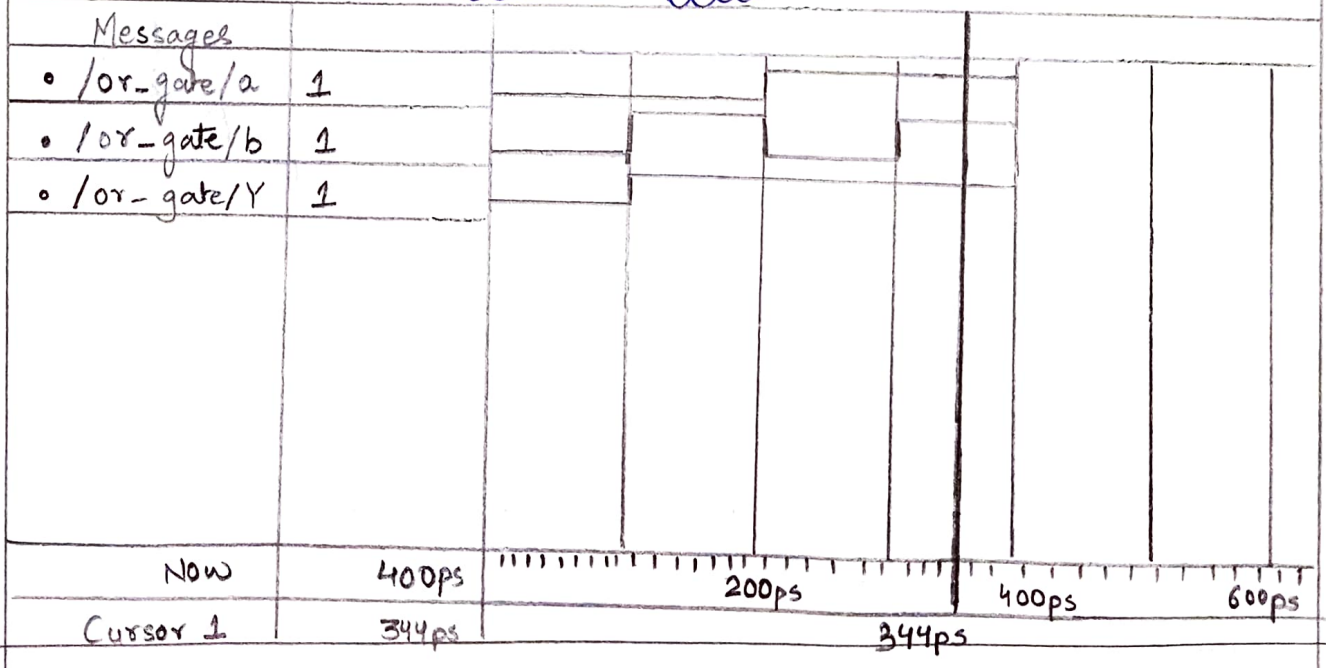
CSE, D, 26

1 Write the VHDL programs to implement all the basis gates :-

(i) OR Gate

```
Library IEEE;
  use IEEE.std_logic_1164.all;
entity OR_gate is
  port (A: in std_logic;
        B: in std_logic;
        Y: out std_logic;
end OR_gate;
architecture orLogic of OR_gate is
  begin
      Y <= A OR B;
  end orLogic;
```

Simulation result



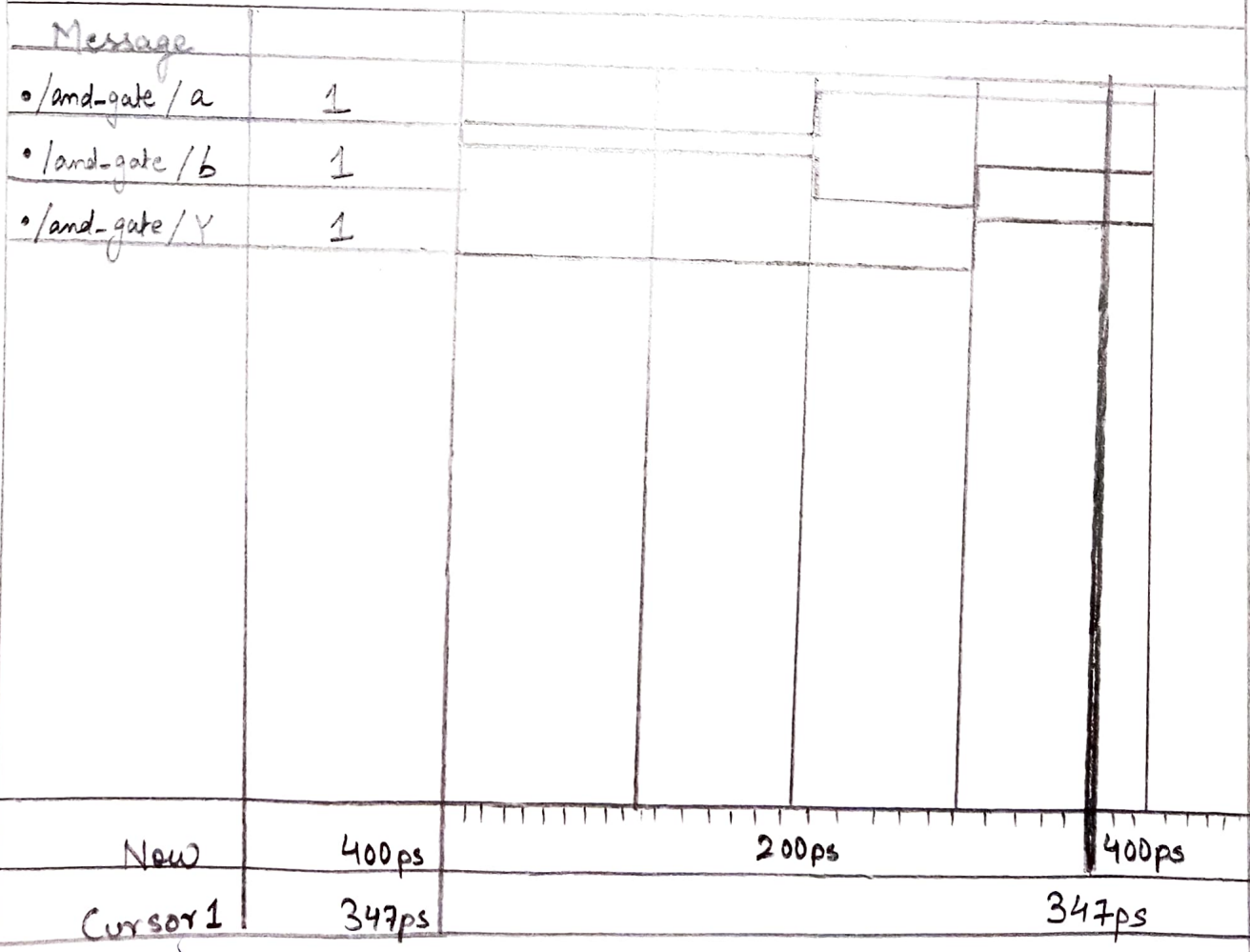
(10 11)

AND Gate

```
Library IEEE;
  use IEEE. std-logic -1164. all;
entity AND_gate is
port (A : in std-logic ;
      B : in std-logic ;
      Y : out std-logic );
end AND_gate ;
architecture ANDlogic of AND-gate is
 begin
      Y <= A AND B ;
end ANDlogic ;
```

(12)

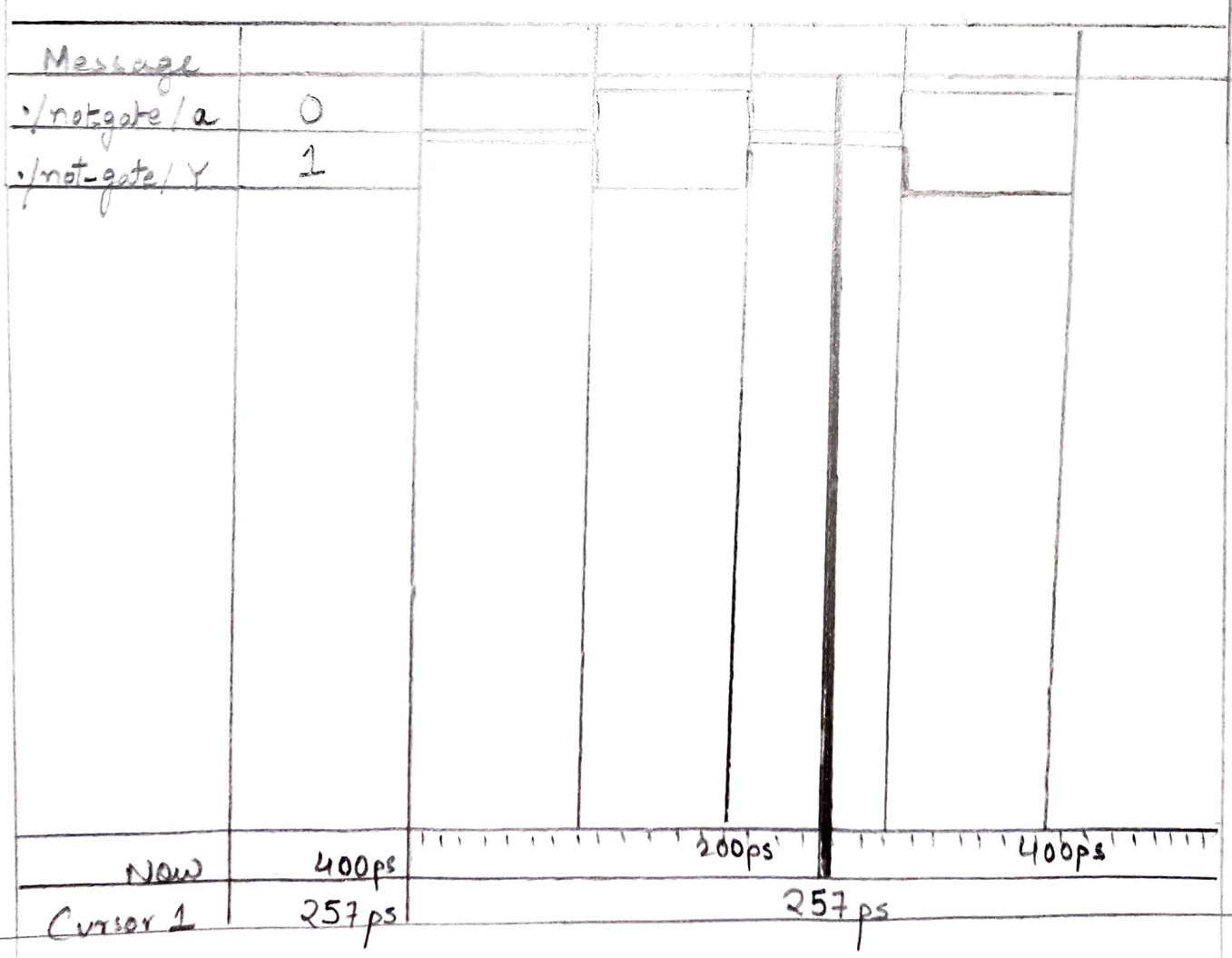
Simulation result



(iii) NOT Gate

```
Library IEEE;
use IEEE. std_logic_1164. all;
entity NOT_gate is
port (A: in std_logic;
      Y: out std_logic);
end NOT_gate;
architecture NOTlogic of NOT_gate is
begin
    Y <= not (A);
end NOTlogic;
```

Simulation result

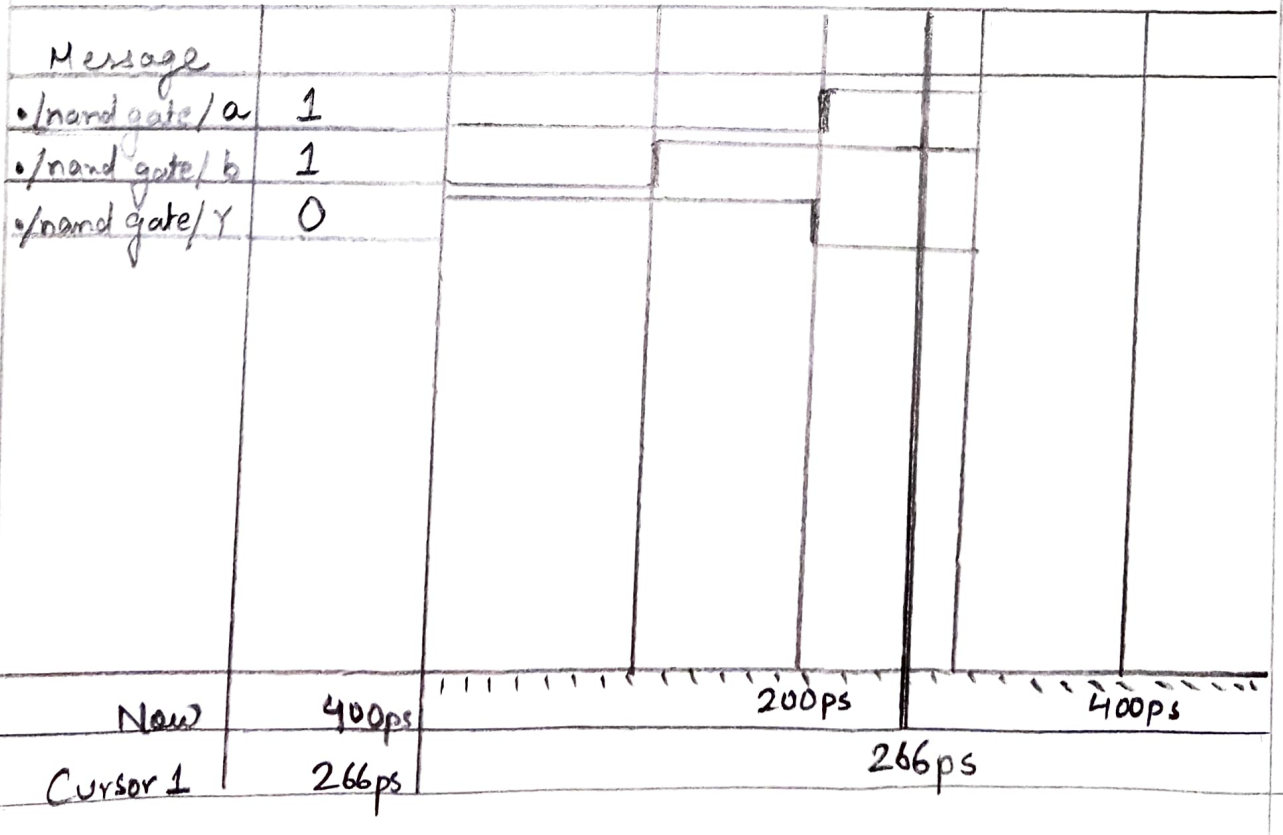


2. Write the VHDL programs to implement universal gate :-

i) NAND

```
Library IEEE;
  use IEEE. std_logic_1164.all;
entity NAND_gate is
port (A : in std_logic;
      B : in std_logic;
      Y : out std_logic);
end NAND_gate;
architecture NANDlogic of NAND_gate is
 begin
      Y <= A NAND B;
end NAND;
```

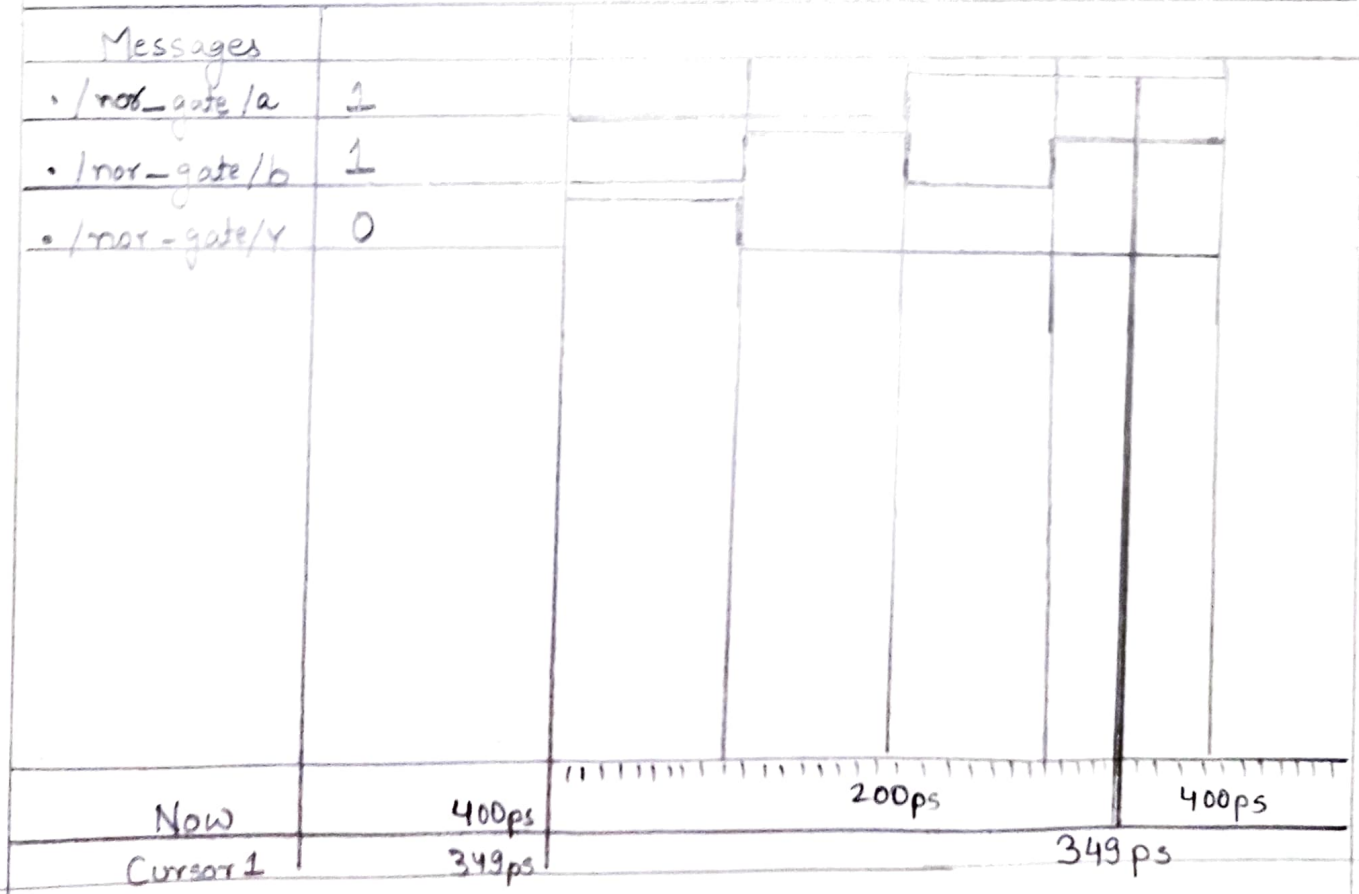
Simulation result



(pp) NOR

```
Library IEEE;
  use IEEE.std_logic_1164.all;
entity NOR_gate is
 port (A: in std_logic;
       B: in std_logic;
       Y: out std_logic);

end NOR_gate;
architecture NORlogic of NOR_gate is
begin
        Y <= not(A OR B);
end NORlogic;
```

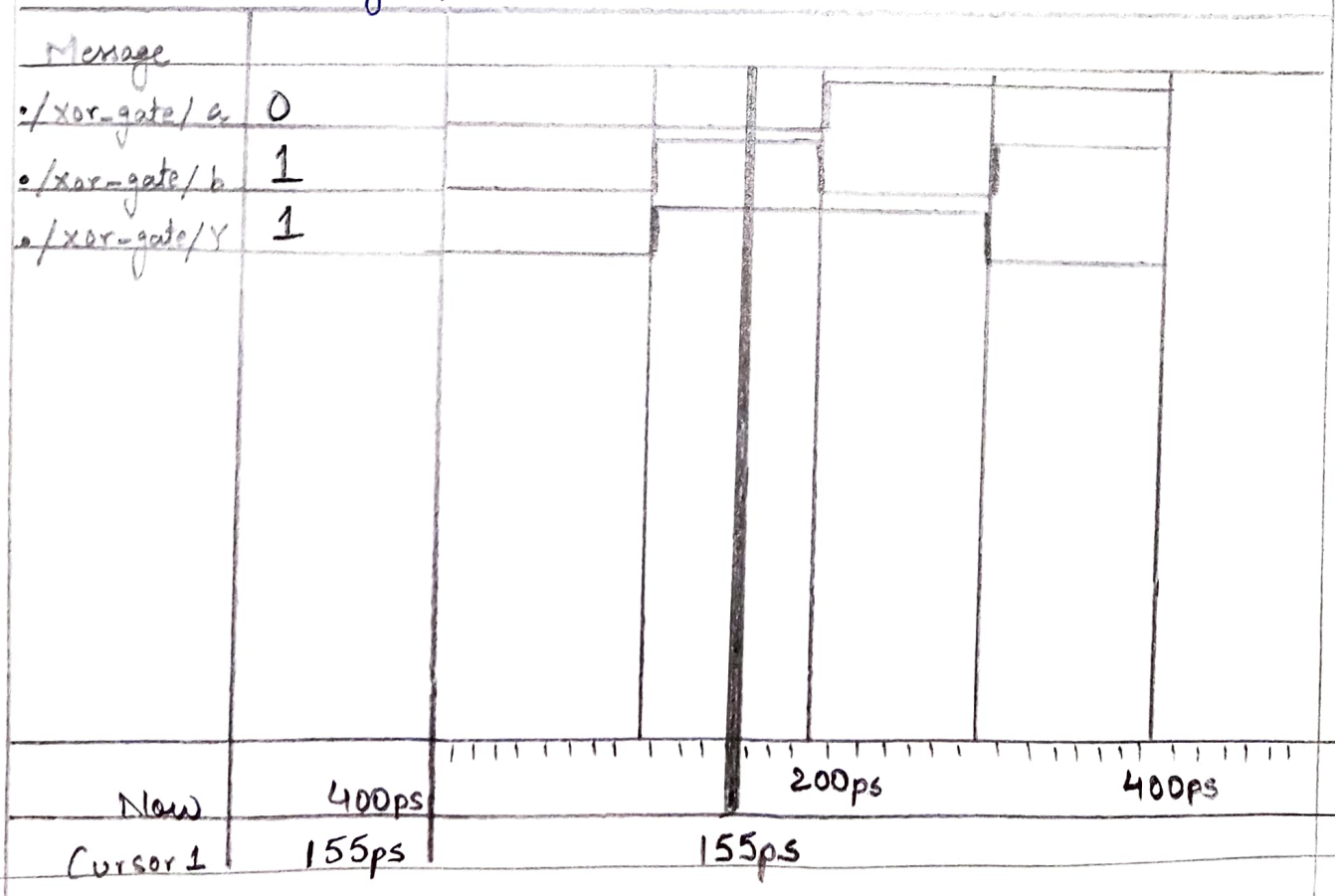
Simulation result



3) Write the VHDL programs to implement

i) XOR

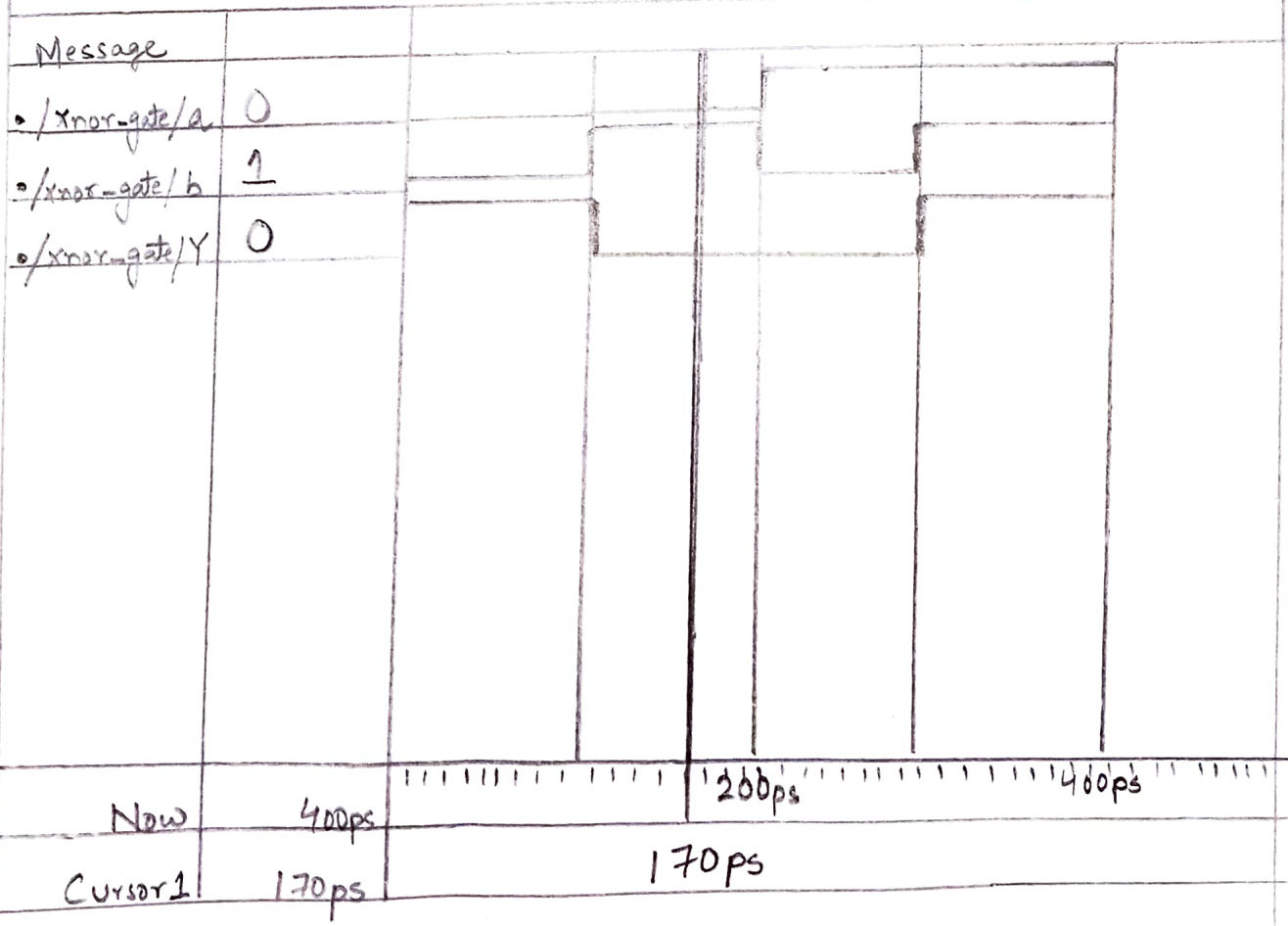
```
Library IEEE;
  use IEEE.std_logic_1164.all;
entity XOR_gate is
  port (A : in std_logic;
        B : in std_logic;
        Y : out std_logic);

end XOR_gate;
architecture XORlogic of XOR_gate is
begin
        Y <= A XOR B;
end XORlogic;
```



(ii) XNOR

```
Library IEEE;
  use IEEE. std_logic_1164.all;
entity XNOR_gate is
 port ( A : in std_logic ;
        B : in std_logic ;
        Y : out std_logic );
end XNOR_gate;
architecture XNORlogic of XNOR_gate is
 begin
        Y <= not (A XOR B);
end XNORlogic ;
```

Simulation result

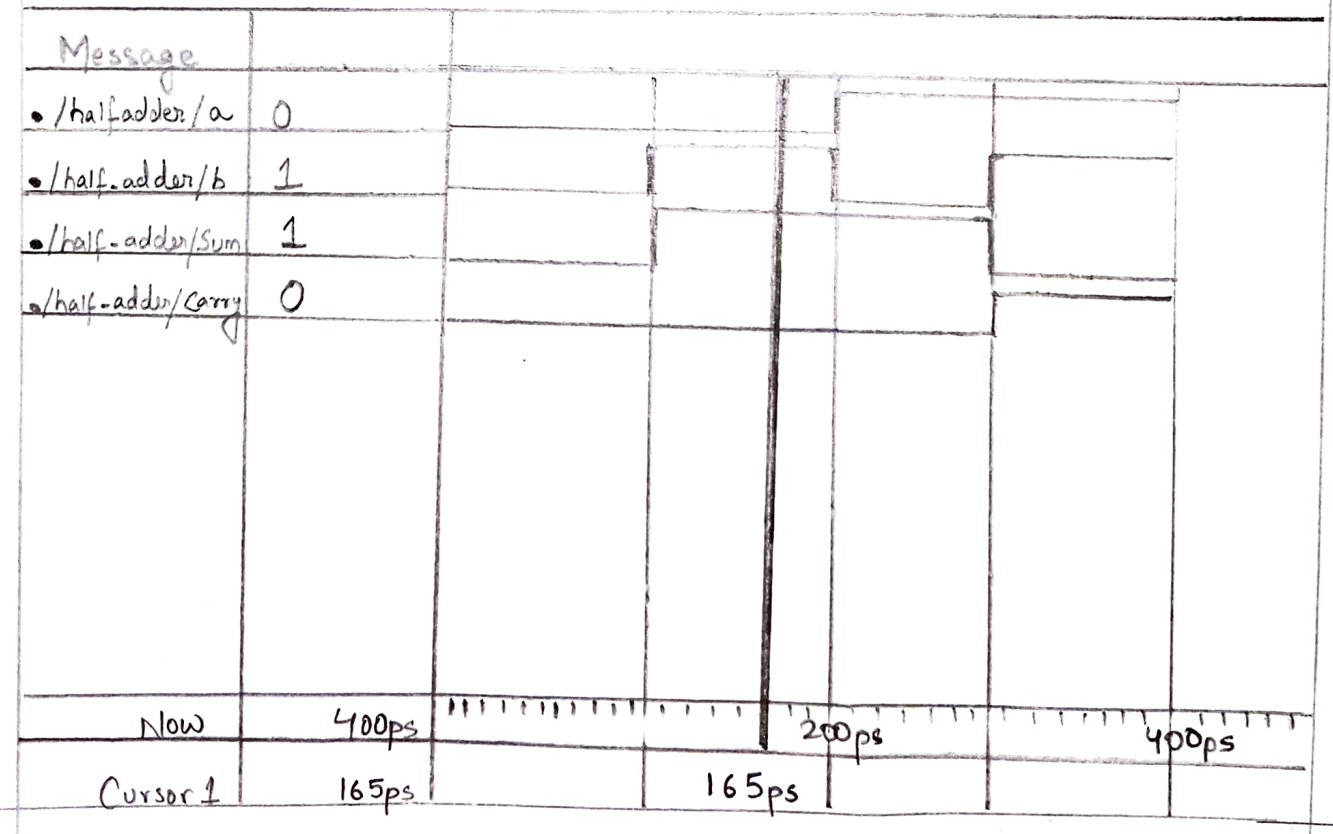


4) Design i) Half adder using VHDL

(i)

```
Library IEEE;
  use IEEE. std-logic-1164. all ;
entity half-adder is
port (A : in bit ;
      B : in bit ;
      Y : out bit ;
      X : out bit);
end half-adder ;
architecture data of half-adder is
begin
      Y <= A XOR B ;
      X <= A AND B;
end data ;
```

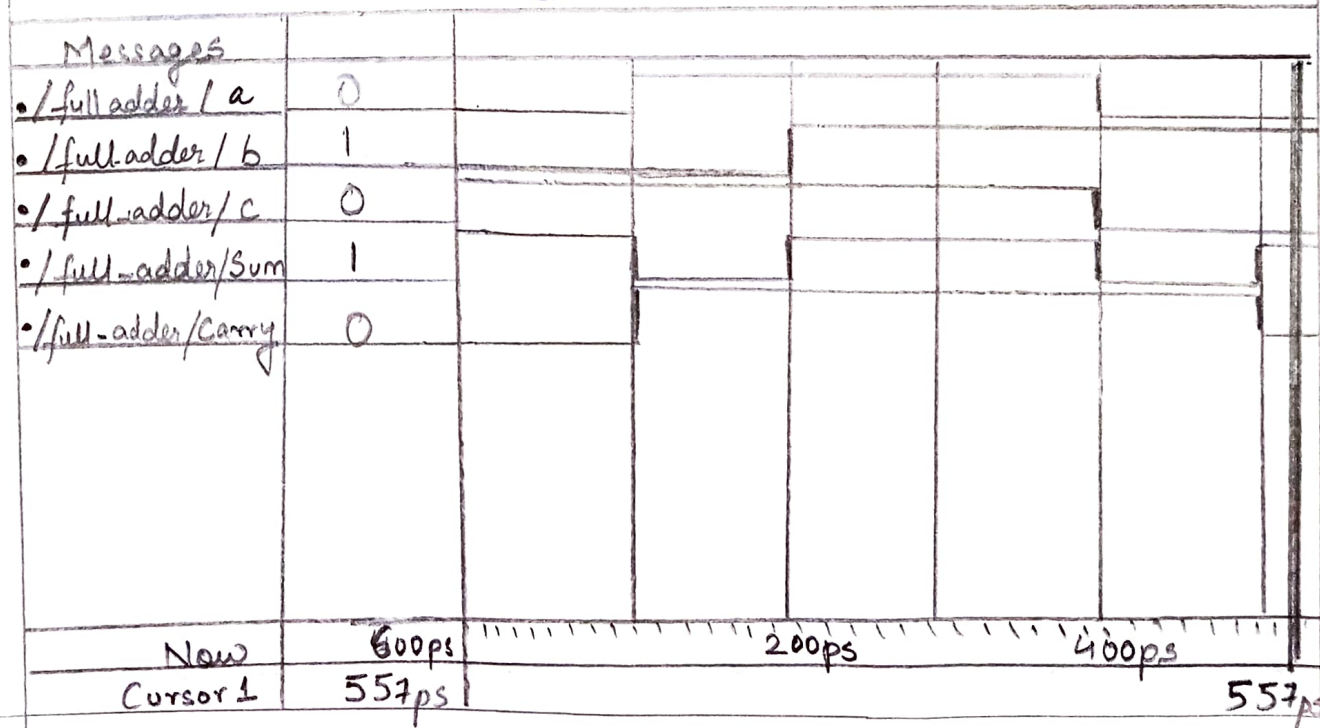
Simulation result



(ii) Full adder using VHDL

```
Library IEEE;
  use IEEE. std_logic_1164. all;
entity full_adder is
  port (A : in bit;
        B : in bit;
        C : in bit;
        Y : out bit;
        X : out bit);
end full_adder;
architecture data of full_adder is
begin
    Y <= A XOR B XOR C;
    X <= ((A AND B) OR (B AND C) OR (A AND C));
end data;
```

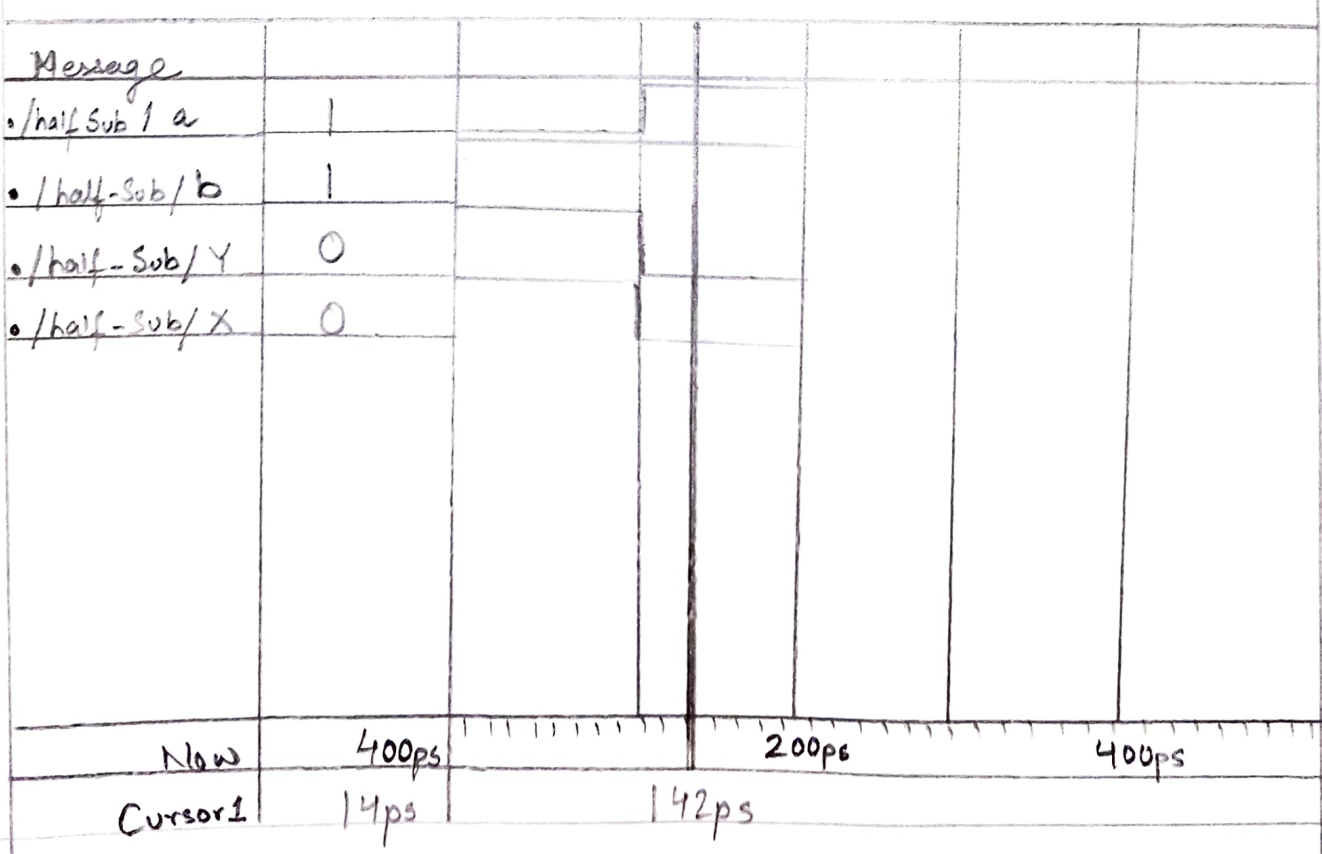
Simulation result



5) Design a) half Subtractor using VHDL.

```
Library IEEE;
  use IEEE. std_logic_1164. all;
entity half_subtractor is
 port ( A : in bit;
        B : in bit;
        Y : out bit;
        X : out bit);
end half_subtractor;
architecture data of half_adder is
begin
        Y <= A XOR B;
        X <= ((not A) AND B);
end data;
```

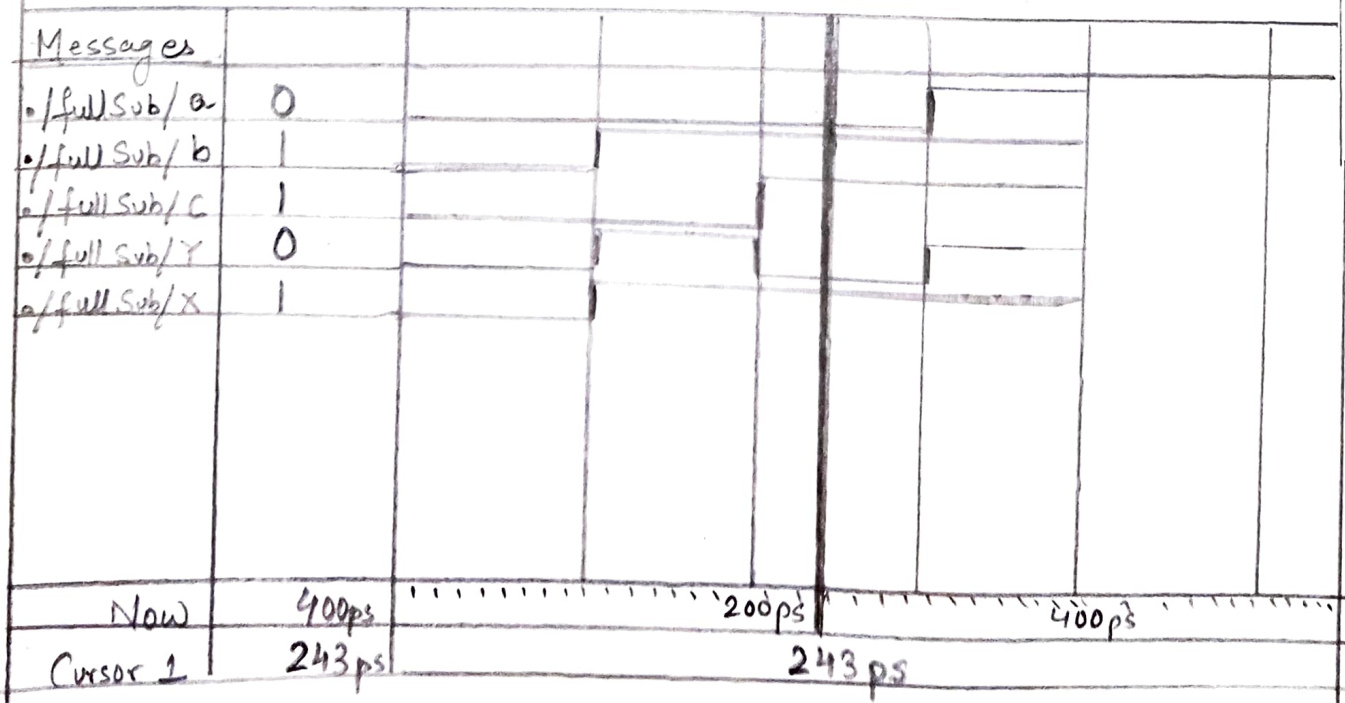
Simulation result



(11) Full Subtractor using VHDL

```
Library IEEE;
use IEEE.std_logic_1164.all;
entity full_Subtractor is
port (A : in bit;
      B : in bit;
      C : in bit;
      Y : out bit;
      X : out bit);
end full_Subtractor;
architecture data of half_adder is
begin
      Y <= A XOR B XOR C;
   X <= ((B AND C) OR ((not A) AND C) OR ((not A) AND B));
end data;
```

Simulation result



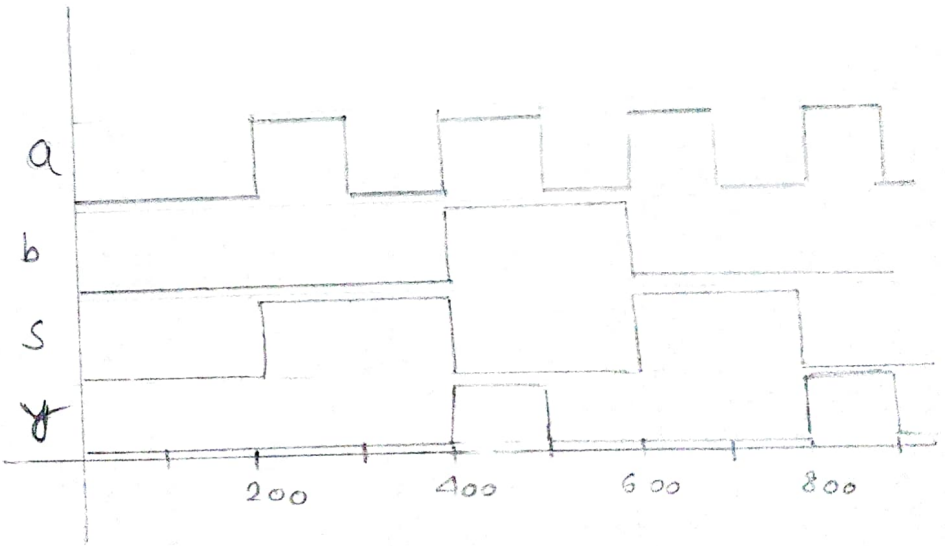
Assignment - 2.

1. Write the VHDL Programs to implement multiplexers : 2:1, 4:1, 8:1 and 16:1 (using when / if-else / vector / testbench)

→ • 2:1 mux

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity test 2to1mux is
    Port ( a: in STD_LOGIC;
           b: in STD_LOGIC;
           s: in STD_LOGIC;
           y: out STD_LOGIC);
end 2to1mux;
architecture Behavioral of 2to1mux is
begin
        y <= a when s = '0' else
             b;
end Behavioural;
```

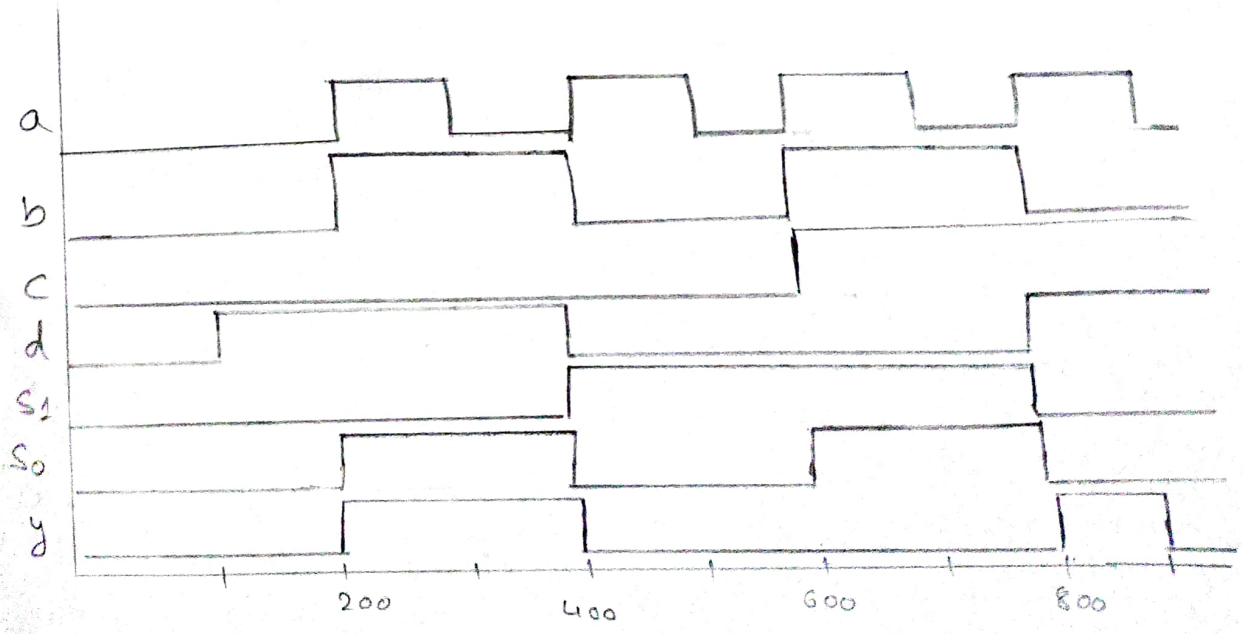
Stimulation Waveform :-



4 : 1 Mux

```
library ieee;
use ieee std_logic_1164.all;
entity 4to1_mux is
   Port (a,b,c,d, s1,s0 : in STD_LOGIC;
         y: out STD_LOGIC);
end 4 to 1_mux;
architecture Behavioral of 4to1_mux is
 Signal s: std_logic_vector (1 downto 0);
 begin
        S<= s1 & s0;
     with S select
          y <= a when "00",
               b when "01",
               c when "10",
               d when "11";
              'z' when others;
    end Behavioral;
```

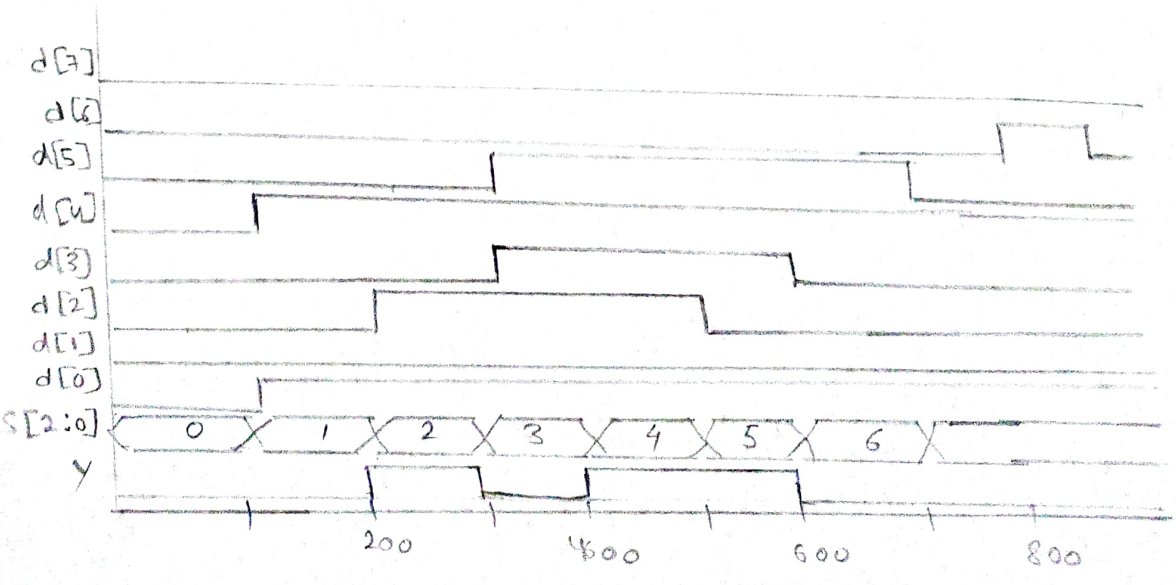
Stimulation Waveform :-



8:1 mux

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity 8to1mux is
  Port ( d: in std_logic_vector (7 downto 0);
         s: in std_logic_vector (2 downto 0);
         y: out std_logic);
end 8to1mux;
architecture Behavioural of 8to1mux is
begin
        y <= d(0) when s = "000" else
             d(1) when s = "001" else
             d(2) when s = "010" else
             d(3) when s = "011" else
             d(4) when s = "100" else
             d(5) when s = "101" else
             d(6) when s = "110" else
             d(7);

end Behavioral;
```

Stimulation Waveform :-



16 : 1 mux

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity mux16 is
Port ( x : in std_logic_vector (15 downto 0);
       s : in std_logic_vector (3 downto 0);
       y : out std_logic);
end mux16;
architecture mux of mux16 is
begin
    Process (x, s)
      begin
       if ( s="0000") then
          y<=x(0);
       else if (s="0001") then
          y<=x(1);
       else if (s="0010") then
          y<=x(2);
       else if (s="0011") then
          y<=x(3);
       else if (s="0100") then
          y<=x(4);
       else if (s="0101") then
          y<=x(5);
```

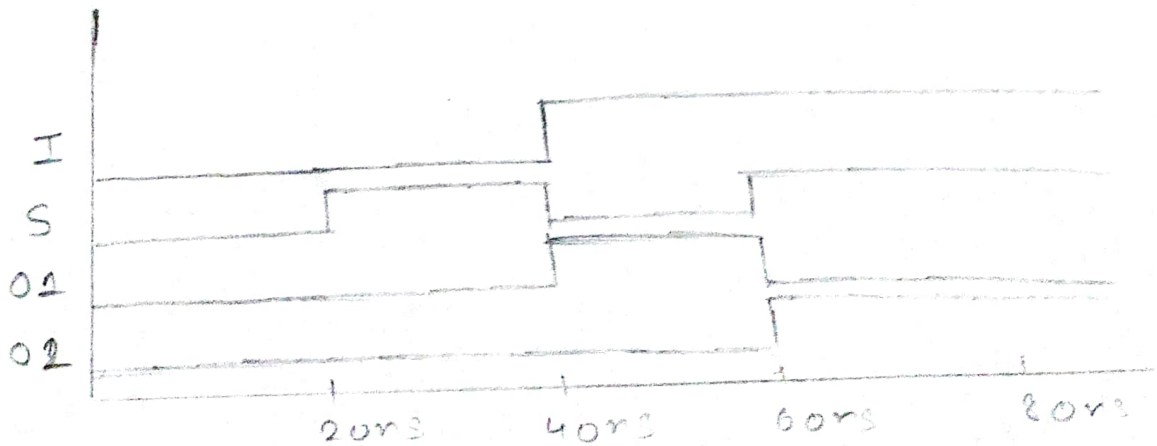
```
elsif (s="0110") then
    y<=x(6);
elsif (s="0111") then
    y<=x(7);
elsif (s="1000") then
    y<=x(8);
elsif (s="1001") then
    y<=x(9);
elsif (s="1010") then
    y<=x(10);
elsif (s="1011") then
    y<=x(11);
elsif (s="1100") then
    y<=x(12);
elsif (s="1101") then
    y<=x(13);
elsif (s="1110") then
    y<=x(14);
else
    y<=x(15);
end if;
end process;
end mux;
```

2. Write the VHDL program to implement de-multiplexers 1:2, 1:4, 1:8 and 1:16 (using when / if-else / vector / testbench).

- 1:2 de-mux

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity Demux 1 to 2 is
    Port ( I, S : in std_logic;
           O1, O2 : out std_logic);
end Demux1to2;
architecture dataflow of Demux1to2 is
begin
        O1 <= I and (not S);
        O2 <= I and S;
end dataflow;
```

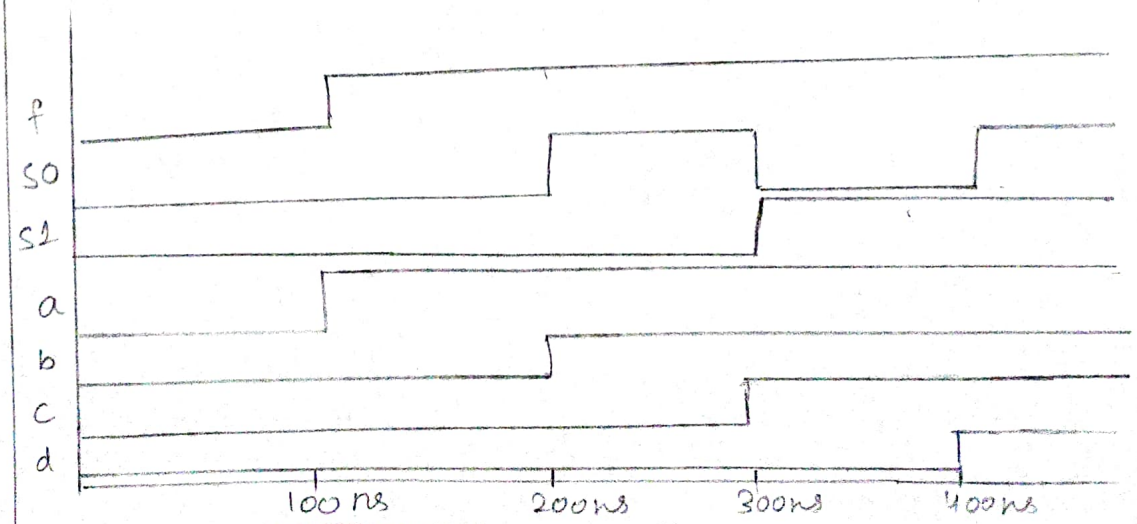
Stimulation Waveform:-



- 1:4 de-mux

```
library ieee;
use ieee.std_logic_1164.all
entity demux_1to4 is
Port( F : in std_logic;
      S0,S1 : in std_logic;
      A,B,C,D : out std_logic);
    end demux_1to4;
architecture bhv of demux_1to4 is
begin
Process (F, S0, S1) is
 begin
      if ( S0 = '0' and S1='0') then
          A<= F;
      elseif (S0 = '1' and S1 = '0') then
          B<= F;
      elseif ( S0 = '0' and S1 = '1') then
          C<= F;
      else
          D<= F;
   end process;
   end bhv;
```

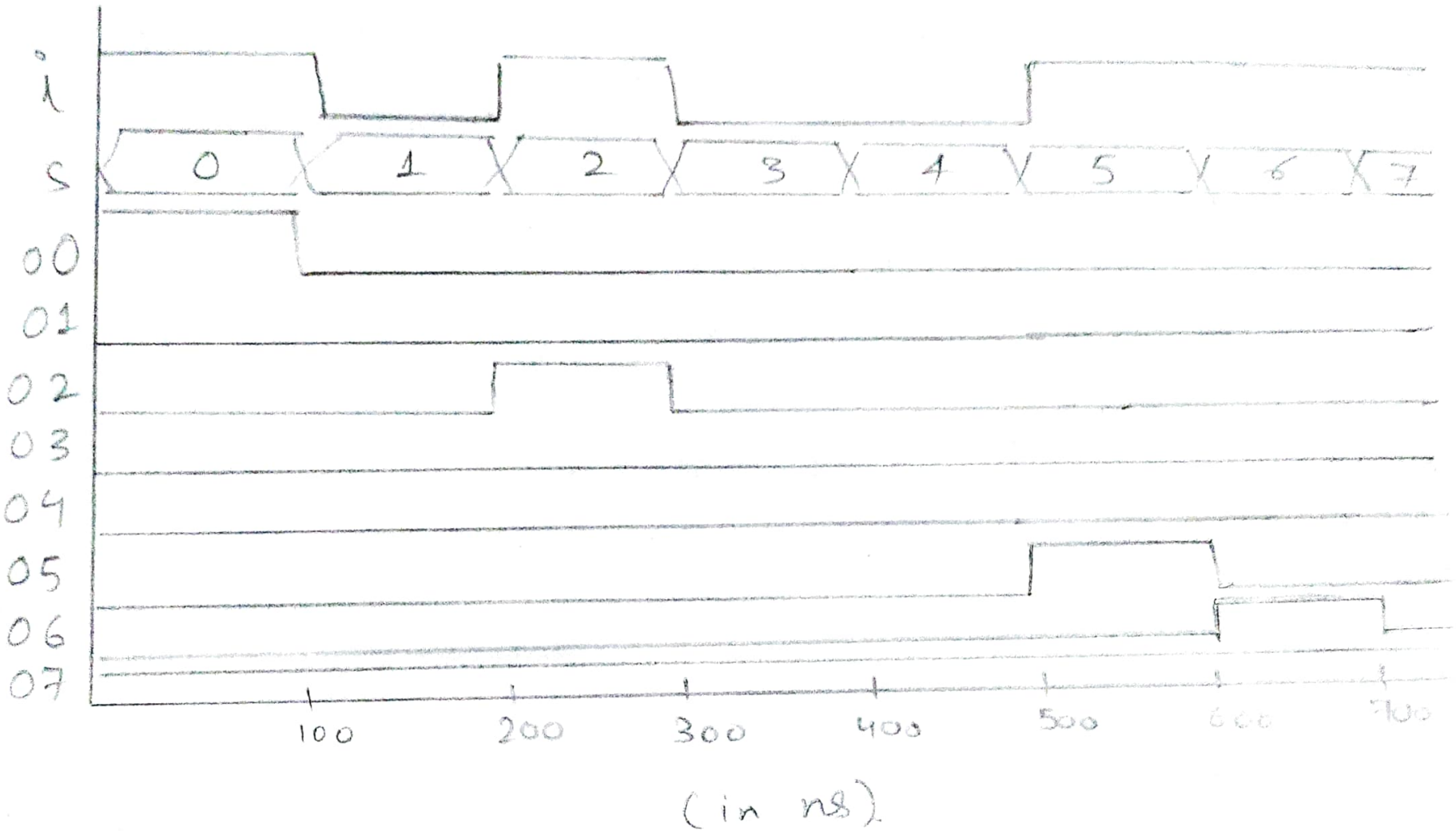
Stimulation Waveform:-



- 1:8 de-mux

```
Library ieee;
use ieee.std_logic_1164.all
entity demux_1to8 is
Port ( i: in std_logic;
       s: in std_logic_vector (0 to 2);
       o: out std_logic_vector (0 to 7);

end demux_1to8;
architecture demux_arch of demux_1x8 is
begin
    process (i,s)
        begin
            o <= "00000000";
            case s is
                when "000" => o(0) <= i;
                when "001" => o(1) <= i;
                when "010" => o(2) <= i;
                when "011" => o(3) <= i;
                when "100" => o(4) <= i;
                when "101" => o(5) <= i;
                when "110" => o(6) <= i;
                when "111" => o(7) <= i;
                when others => o <= "00000000";
            end case;
        end process;
    end demux_arch;
```

Stimulation Waveform:-



1:16 de-mux

```
library ieee;
use ieee.std_logic_1164.all
use ieee.std_logic_arith.all
use ieee.std_logic_unsigned.all
entity demux16 is
port (x : in std_logic ; vector
      s : in std_logic_( 3 downto 0);
      y : out std_logic_vector (15 downto 0);

end demux16;
architecture demux of demux16 is
begin
    process (x, s)
      begin
        case s is
          when '0000' =>
          y(0) <= x ;
          when '0001' =>
            y(1) <= x;
          when '0010' =>
            y(2) <= x;
          when '0011' =>
            y(3) <= x;
          when '0100' =>
            y(4) <= x;
          when '0101' =>
            y(5) <= x;
          when '0110' =>
            y(6) <= x;
          when '0111' =>
            y(7) <= x ;
```

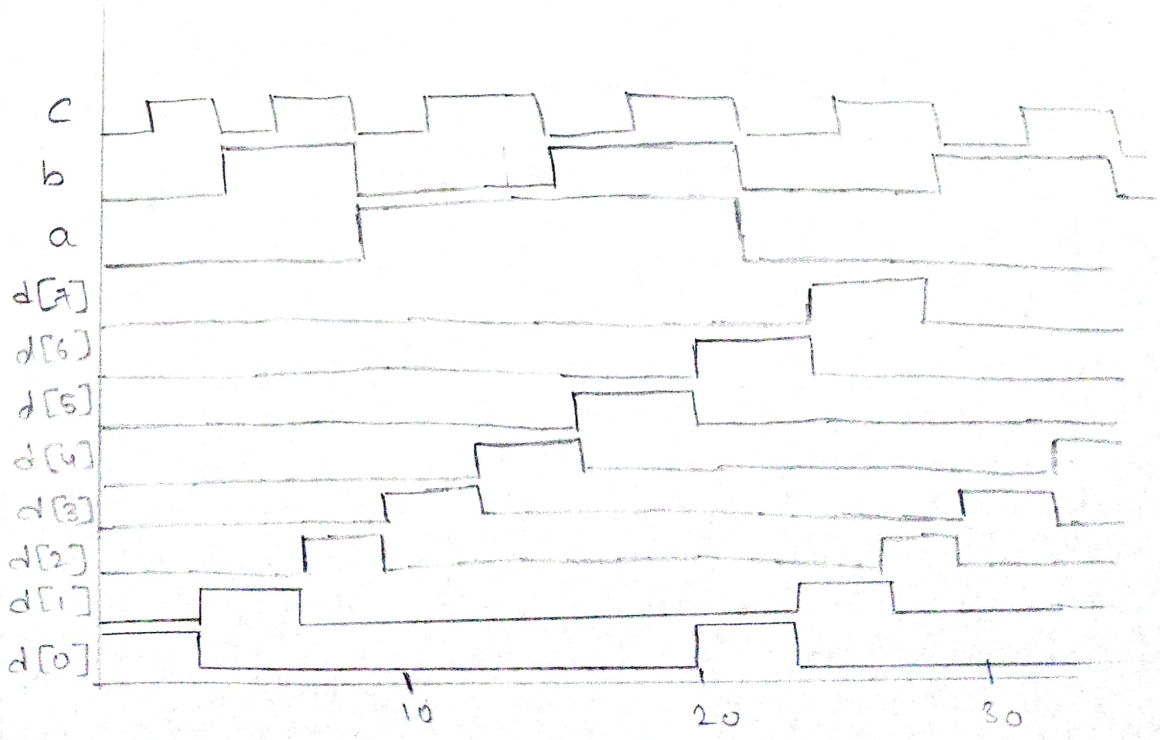
```
when "1000" =>
    y(8) <= x;
when "1001" =>
    y(9) <= x;
when "1010" =>
    y(10) <= x;
when "1100" =>
    y(12) <= x;
when "1011" =>
    y(11) <= x;
when "1101" =>
    y(13) <= x;
when "1110" =>
    y(14) <= x;
when others =>
    y(15) <= x;
end case;
end process;
end demux;
```

3. Write the VHDL program to implement encoder circuit : 8:3, 4:2 and parity encoder

- 8:3 encoder

```
Library ieee;
use ieee. std_logic_1164. all;
entity encoder_8x3 is
Port (d: in std_logic vector (7 downto 0);
      a, b, c : out std_logic);
end encoder_8x3;
architecture encode_8x3_arch of encoder_8x3 is
begin
    process(d)
      begin
        a <= d(4) or d(5) or d(6) or d(7);
        b <= d(2) or d(3) or d(6) or d(7);
        c <= d(1) or d(3) or d(5) or d(7);
    end process;
end encoder_arch;
```

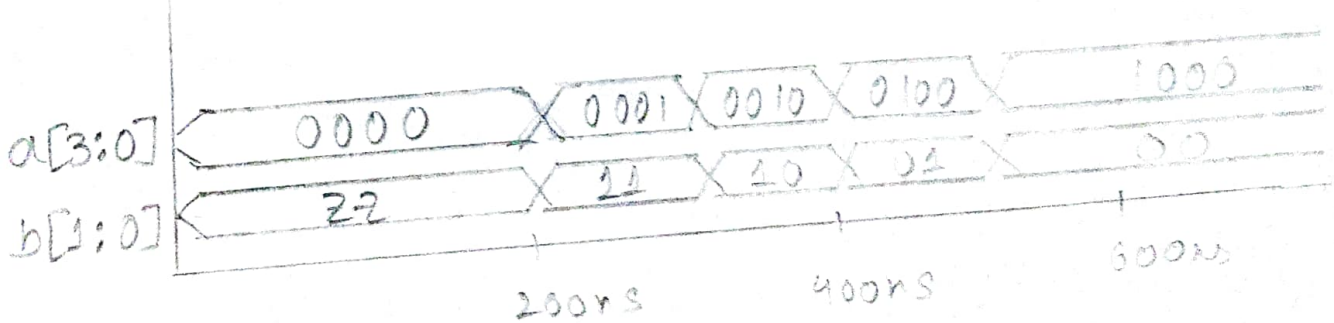
Stimulation Waveform :-



- ## 4:2 Encoder

```
library ieee;
use ieee.std_logic_1164.all;
entity encoder2 is
Port ( a: in std_logic_vector (3 downto 0);
       b: out std_logic_vector (1 downto 0));

end encoder2;
architecture bhv of encoder2 is
begin
        b(0) <= a(1) or a(2);
        b(1) <= a(1) or a(3);

end bhv;
```

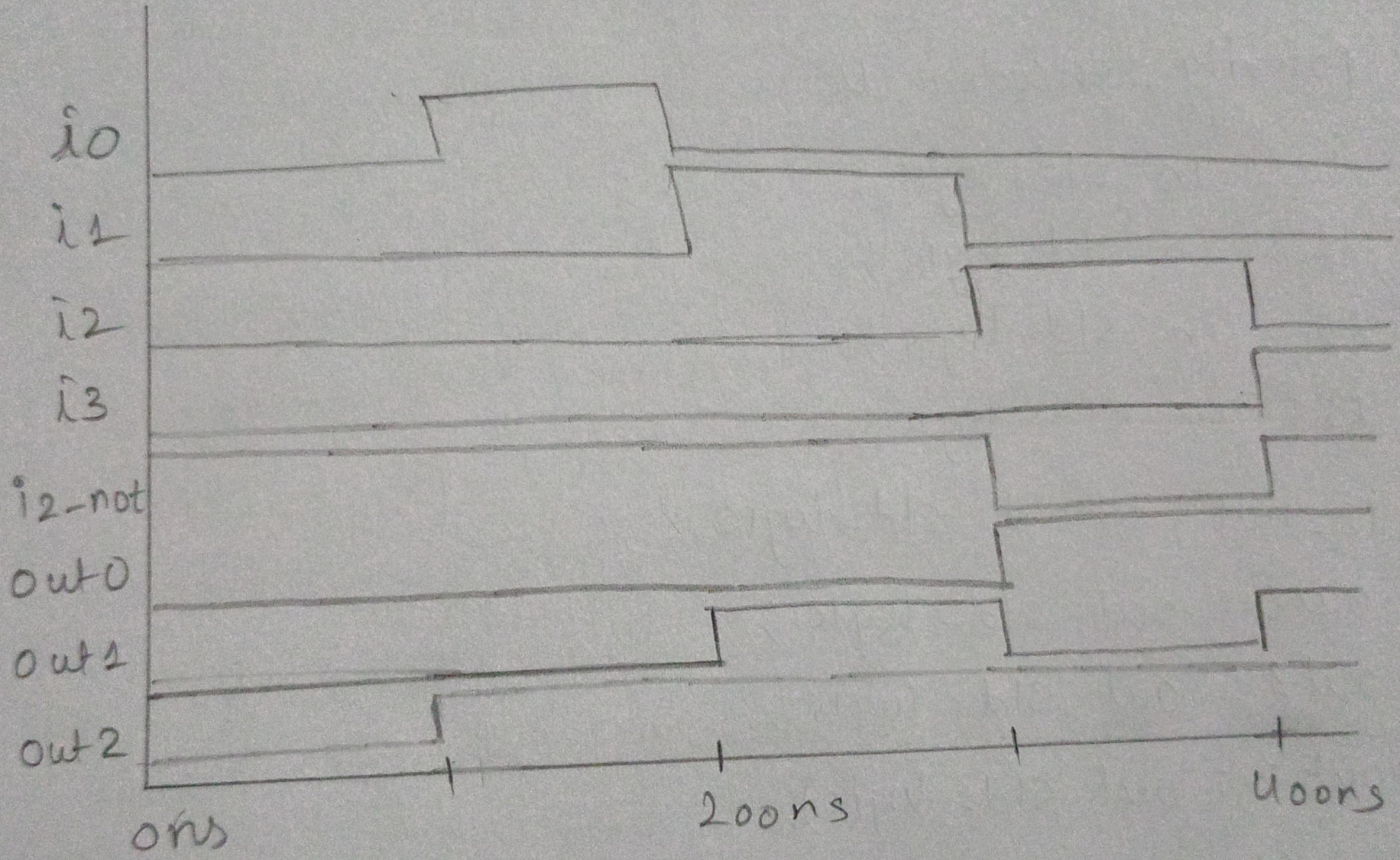
Stimulation Waveform :-



Priority Encoder 4:2

```
library ieee;
use ieee, std_logic_1164.all;
entity priority_encoder is
Port ( i0 : in std_logic;
       i1 : in std_logic;
       i2 : in std_logic;
       i3 : in std_logic;
       i2_not : inout std_logic;
       out0 : out std_logic;
       out1 : out std_logic;
       out2 : out std_logic);
end Priority_encoder;
architecture Dataflow of priority_encoder is
begin
    process(i2)
    begin
        if i2 = '0' then
         i2_not <= '1';
        else
          i2_not <= '0';
        end if;
        end process;
       out2 <= i0 OR i1 OR i2 OR i3;
       out1 <= (i1 AND i2_not) OR i3;
       out0 <= i2 OR i3;
       end Data flow;
```

Stimulation Waveform:-

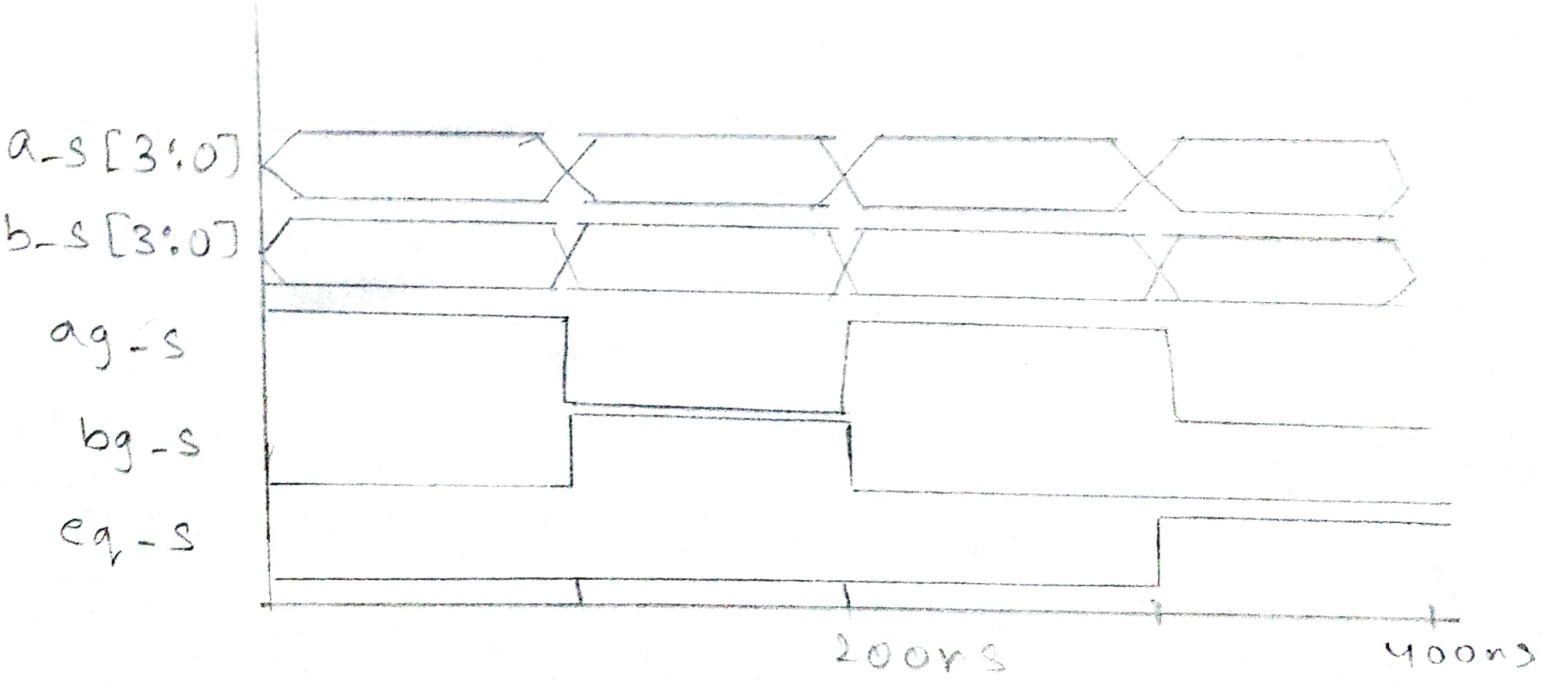


5. Write a VHDL program to implement magnitude Comparator circuit.

```
library ieee;
use ieee.std_logic_1164.all
entity mag_comp is
end mag_comp;
architecture beh of mag_comp is
 component mag_comp
  Port ( a,b : in std_logic_vector (3 downto 0);
         ag, bg, eq : out std_logic);
  end component;
Signal a_s, b_s : std_logic_vector (3 downto 0);
Signal ag_s, bg_s, eq_s : std_logic;
begin -- beh
    u1 : mag_comp port map (
        a => a_s,
        b => b_s,
        ag => ag_s,
        eq => eq_s,
        bg => bg_s);
    tst_p : process
        begin
         a_s <= "1111";
         b_s <= "0000";
         wait for 100 ns;
            a_s <= "1010";
            b_s <= "1100";
         wait for 100 ns;
            a_s <= "1001";
            b_s <= "0011";
```

```
        wait for 100 ns;
            a_s <= "1000";
            b_s <= "1000";
        wait for 100 ns;
    end process;
end beh;
```

Stimulation Waveform :-

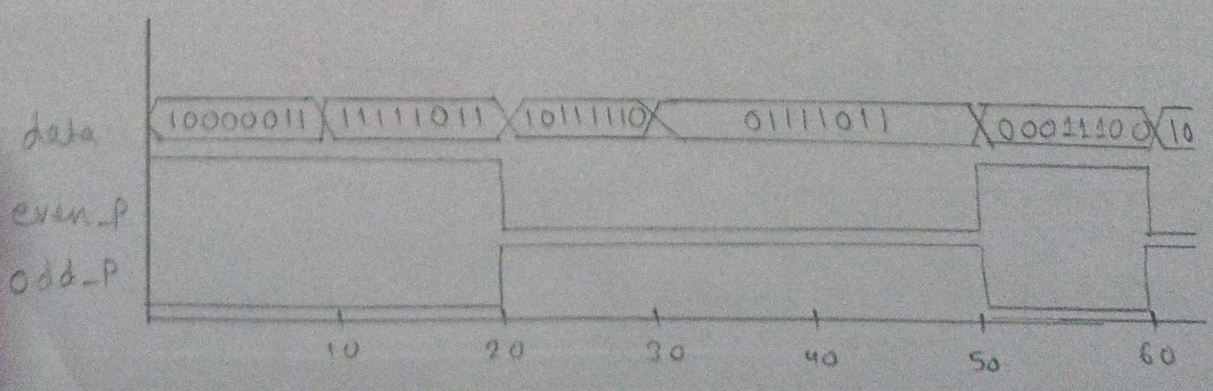


6. Write the VHDL programs to implement parity generator and parity checker

- 8-bit parity generator

```
library ieee;
use ieee.std_logic_1164.all;
entity parity is
    Port ( data : in bit_vector (7 downto 0);
           even_p, odd_p : out bit);

end parity;
architecture parity_gen of parity is
signal temp : bit_vector (5 downto 0);
begin
    temp(0) <= data(0) xor data(1);
    temp(1) <= temp(0) xor data(2);
    temp(2) <= temp(1) xor data(3);
    temp(3) <= temp(2) xor data(4);
    temp(4) <= temp(3) xor data(5);
    temp(5) <= temp(4) xor data(6);
    even_p <= temp(5) xor data(7);
    odd_p <= not(temp(5) xor data(7));
end parity_gen;
```

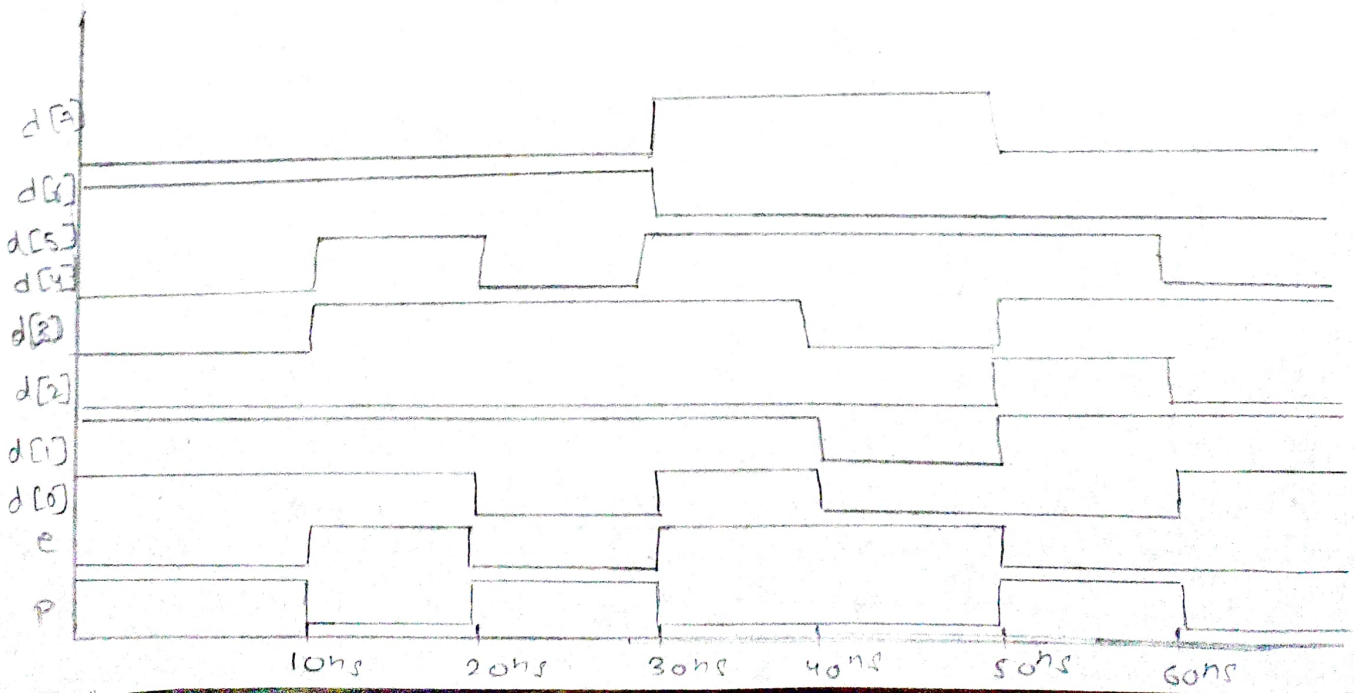
Stimulation waveform :-



- 8-bit parity checker

```
library ieee;
use ieee.std_logic_1164.all
entity parity_chk is
Port ( data : in bit_vector (7 downto 0);
       p : in bit;
       e : out bit) ;
end parity_chk;
architecture parity_arch of parity_chk is
Signal temp : bit_vector (6 downto 0);
begin
    temp(0) <= data(0) xor data(1);
    temp(1) <= temp(0) xor data(2);
    temp(2) <= temp(1) xor data(3);
    temp(3) <= temp(2) xor data(4);
    temp(4) <= temp(3) xor data(5);
    temp(5) <= temp(4) xor data(6);
    temp(6) <= temp(5) xor data(7);
     e <= p xor temp(6);
end parity_arch;
```

Stimulation Waveform :-

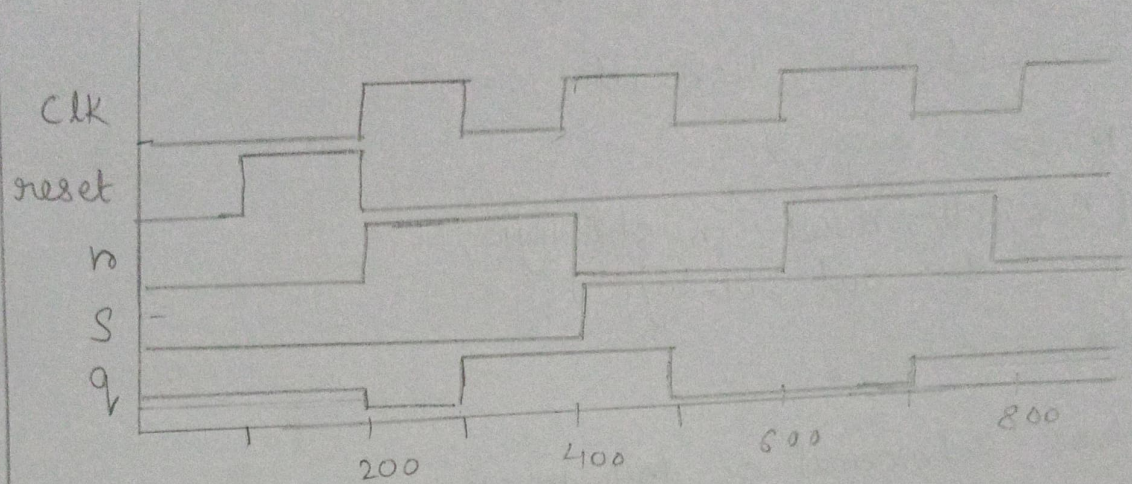


7. Write the VHDL programs to implement flip flop units : SR, JK, D, T.

SR Flip Flop

```
library ieee;
use ieee.std_logic_1164.all

entity sr is
  Port ( r,s, clk, reset : in std_logic;
         q : out std_logic);
end sr;

architecture Behavioral of sr is
signal p: std_logic;
begin
    process (clk, reset)
  begin
    if (reset = '1') then
      P <= '0';
   else if (clk'event and clk='1') then
       if (r='0' and s='0') then
          P <= P;
       else if (r='0' and S='1') then
          P<='0';
       else if (r='1' and S='0') then
          P<='1';
       else if (r='1' and s='1') then
          P<='-';
       end if;
   end if;
     q <= P;
  end process;
end Behavioral;
```

Stimulation Waveform :-

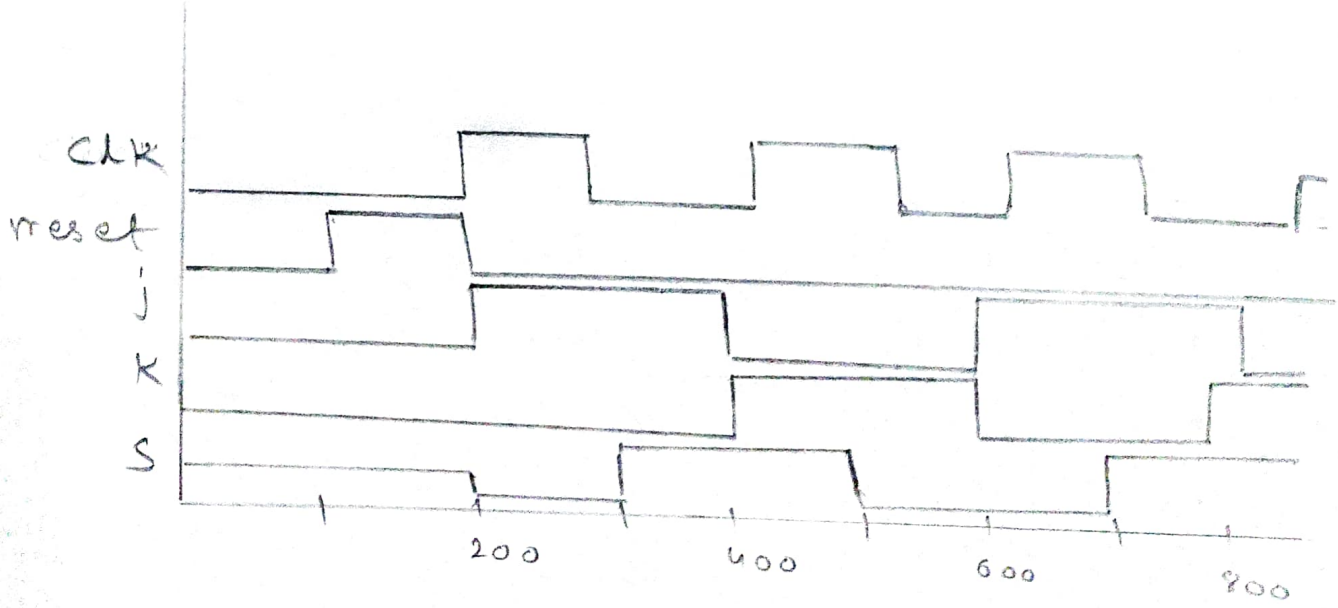


- ## JK Flip Flop

```
library ieee;
use ieee.std_logic_1164.all
entity jk is
    Port ( j,k, clk, reset : in std_logic;
           q : out STD_LOGIC);
end jk;
architecture Behavioral of jk is
signal s: std_logic;
begin
process (clk, reset)
begin
if(reset = '1') then
    s <= '0';
else if ( clk'event and clk = '1') then
```

```
        if (j='0' and k='0') then
            s<=s;
        elsif (j='0' and k='1') then
            s<='0';
        else if (j='1' and k='0') then
            s<='1';
        else if (j='1' and k='1') then
            s<= not s;
        end if;
    end if;
    q<=s;
  end process;
end Behavioral;
```

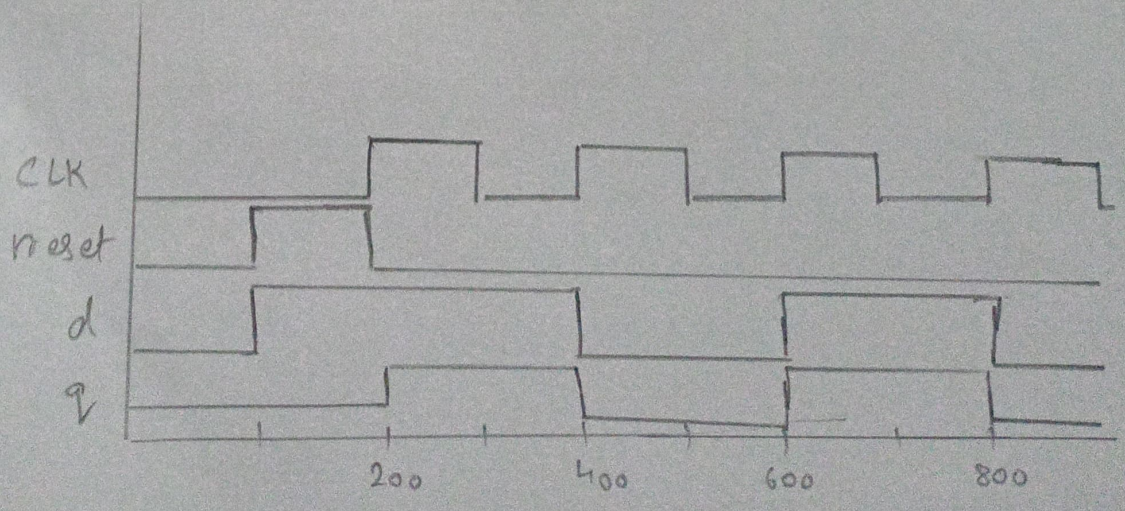
Stimulation Waveform :-



- D-flipflop

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

entity dff is
Port (d, clk, reset : in STD_LOGIC;
                  q : out STD_LOGIC);

end dff;

architecture Behavioral dff is
begin
 Process(clk)
 begin
 if (clk'event and clk = '1') then
      if (reset = '1') then
          q<='0';
      else
          q<=d;
      end if;
   end if;
      end process;
end Behavioral;
```

Stimulation Waveform :-



- T - flip-flop

```
library ieee;
use ieee std_logic_1164.all
entity tff is
  Port ( t, clk, reset : in std_logic;
         q : out std_logic);
  end tff;
architecture Behavioral tff is
  Signals : std_logic;
begin
    process ( clk, reset)
  begin
  if ( reset = '1' ) then
      S <= '0';
  else if (clk'event and clk = '1') then
    if ( t= '1') then
          S <= not S;
    else
          S <= S;
    end if;
  end if;
  end process;

  end Behavioral;
```

Stimulation Waveforms:-