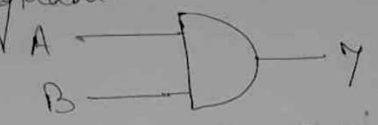
Assignment - 1

1) Write a VHDL program to implement all the AND gates using data flow.

Title - Implementation of AND gate using data flow modelling.

Symbolic diagram -



Boolean Expression - [A.B].

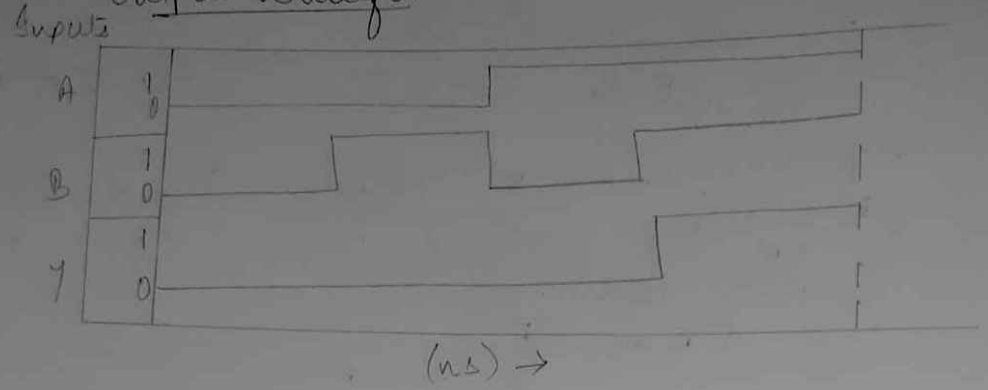
Truth Table

A	B	Y = (A.B)
0	0	0
0	1	0
1	0	0
1	1	1

code

```
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;
```

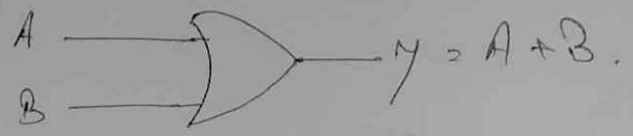
Output Waveform



b) Write a VHDL program to implement all OR Gate using data flow modelling.

Title - Implementation of OR Gate using data flow modelling.

Symbolic diagram



Boolean Expression - (A + B).

Truth Table

A	B	Y = (A + B)
0	0	0
0	1	1
1	0	1
1	1	1

Code

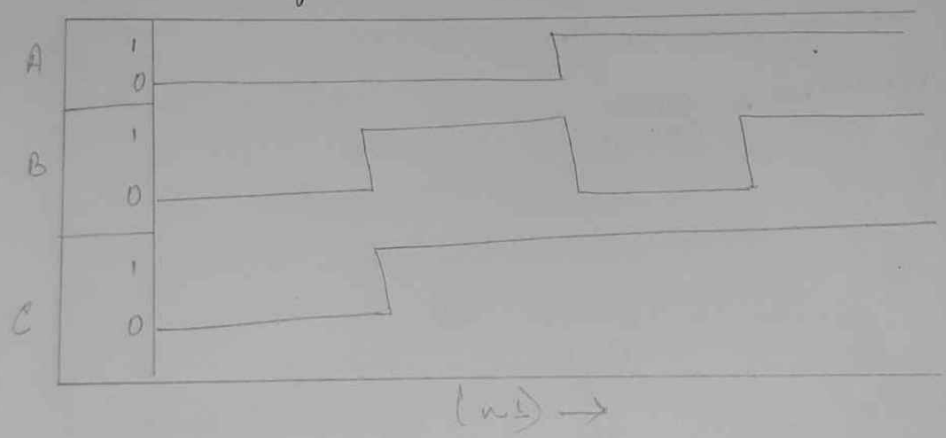
```
library IEEE;
use IEEE.std_logic_1164.all;
entity OR_gate is
  Port ( A: in std_logic;
         B: in std_logic;
         Y: out std_logic);
end OR_gate
architecture OR_gateLogic of OR_gate is
begin
  Y <= A AND B;
end andLogic;
```

```
Y <= A AND B;
end OR_gateLogic;
```

all OR

dataflow

Output waveform



5) Write a VHDL program to implement all NOT Gate using data flow modelling.

Title - Implementation of NOT Gate using data flow modelling.

Symbolic diagram

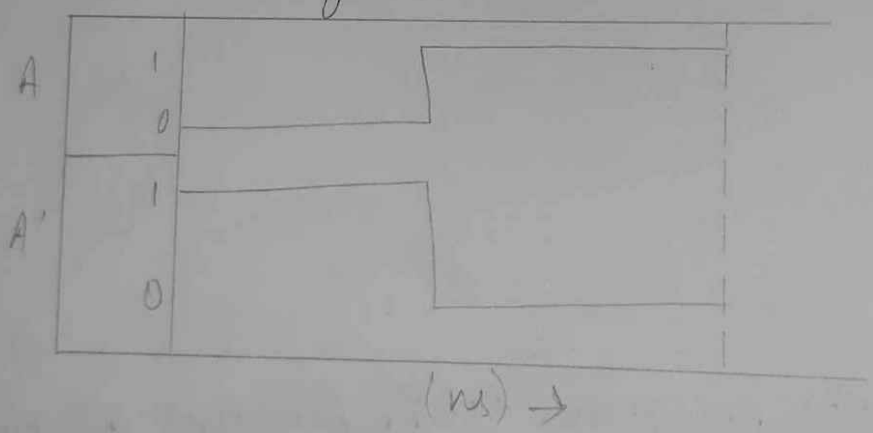
Truth Table

A	A' = Y
0	1
1	0

Code

```
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;
```

Output Waveform

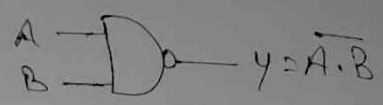


1) Write a VHDL program to implement all NAND gate using data flow modelling.

Title - Implementation of NAND gate using data flow modelling.

Boolean Expression - $Y = \overline{AB}$

Symbolic Diagram -



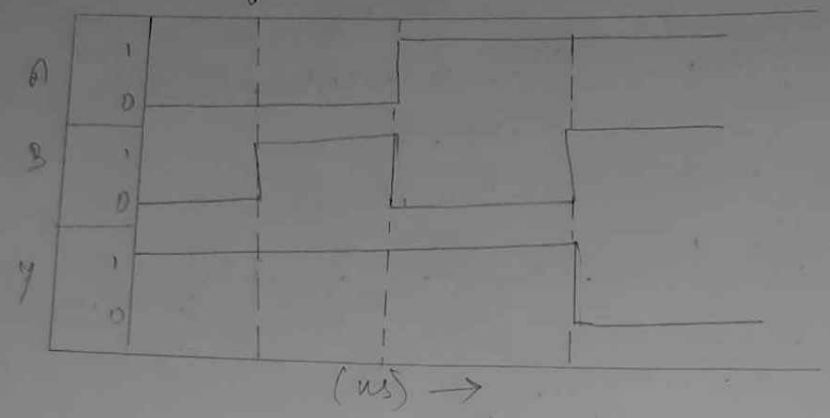
Truth Table

A	B	$Y = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Code

```
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 NANDLogic;
```

Output Waveform

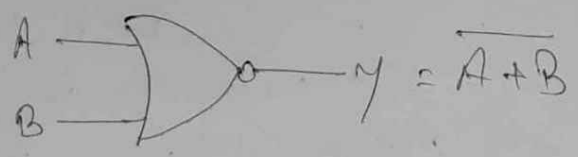


5) Write a VHDL program to implement all NOR GATE using data flow modelling.

Title - Implementation of NOR gate using data flow modelling.

Boolean expression - $Y = \overline{A+B}$

Symbolic Diagram -



Truth Table

A	B	$Y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

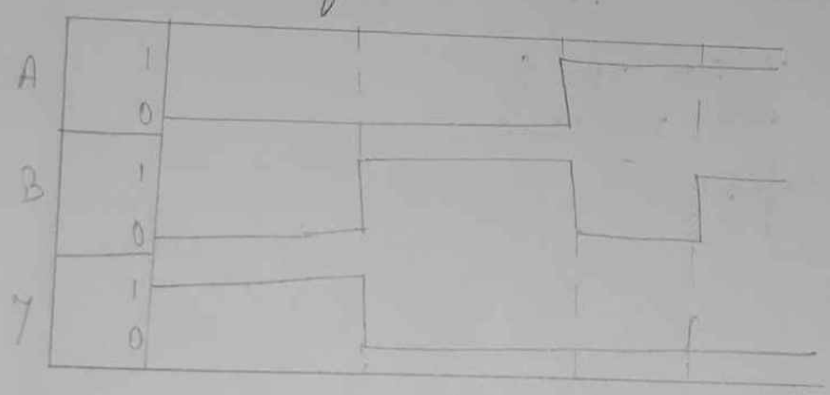
Code

```
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 NORgate is
begin
  Y <= A NOR B;
end NORLogic;
```

NOR

data

Output waveform

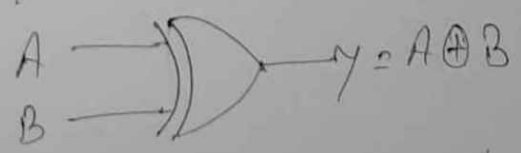


b) Write a VHDL program to implement all XOR gate using data flow modeling.

Title - Implementation of XOR gate using data flow modeling.

Boolean Expression - $Y = A \oplus B$

Symbolic Diagram -



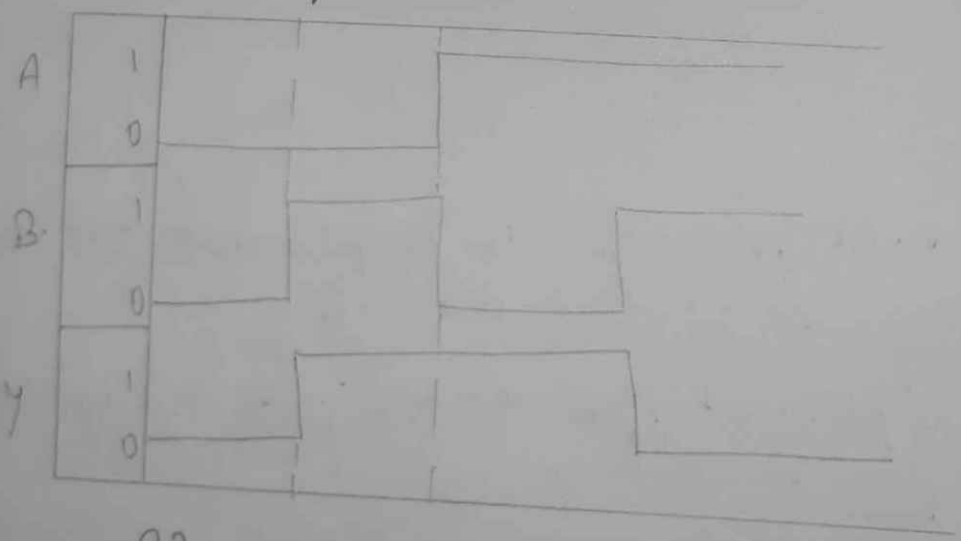
Truth Table

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Code

```
Library IEEE;
use IEEE_std_logic_1164.all;
entity XORgate is
 port ( A: in std_logic;
        B: in std_logic;
        Y: out std_logic);
end XORg;
architecture XORLogic of XORg is
 begin
        Y <= A XOR B
        end XORLogic;
```

Output waveform

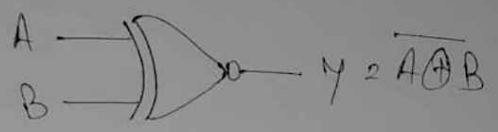


7) Write a VHDL program to implement an XNOR Gate using data flow modeling.

Title - Implementation of XNOR gate using data flow modeling.

Boolean Expression - $Y \Rightarrow Y = \overline{A \oplus B}$

Symbolic Diagram -



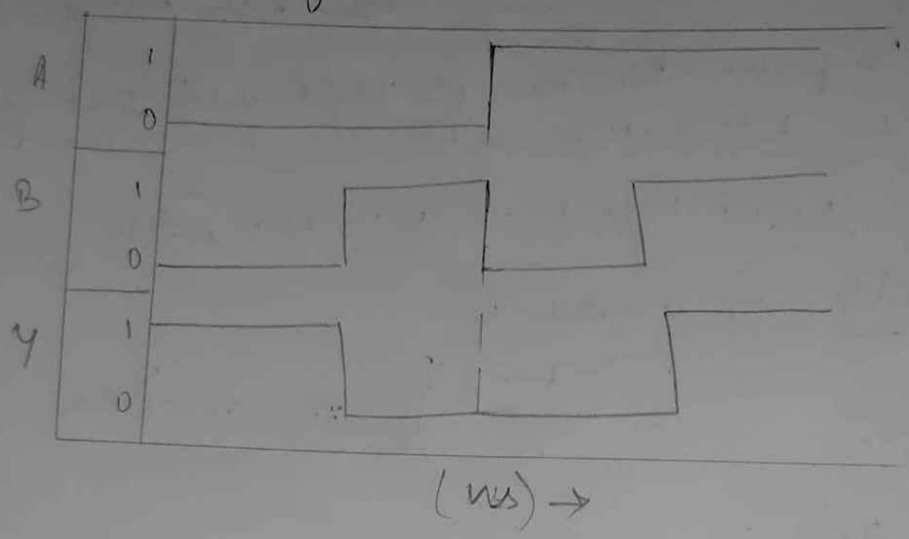
Truth Table -

A	B	$\overline{A \oplus B} = Y$
0	0	1
0	1	0
1	0	0
1	1	1

Code

```
library IEEE;
use IEEE_std_logic_1164.all;
entity XNORg is
    Port( A: in std_logic;
          B: in std_logic;
          Y: out std_logic);
end XNORg;
architecture XNORLogic of XNORg is
begin
    Y <= A XNOR B;
end XNORLogic;
```

Output waveform



31.8.22

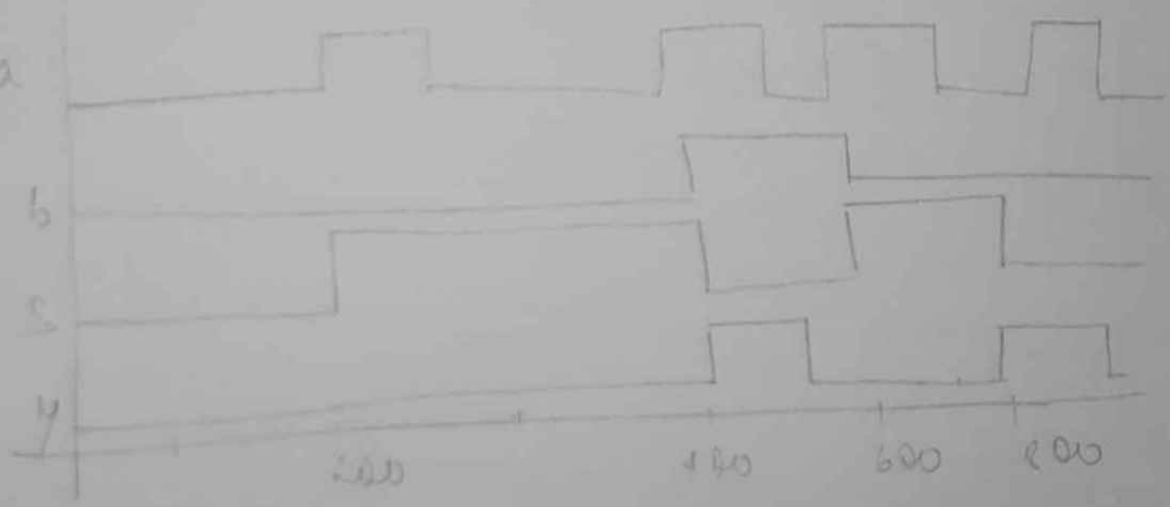
Assignment - 2

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

→ · 2:1 mux

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity test 2to1 mux is
  Port( a : in STD_LOGIC;
        b : in STD_LOGIC;
        c : in STD_LOGIC;
        y : out STD_LOGIC);
end 2to1 mux
architecture behavioral of 2to1 mux is
begin  y <= a when s = '0' else
end Behavioral;
```

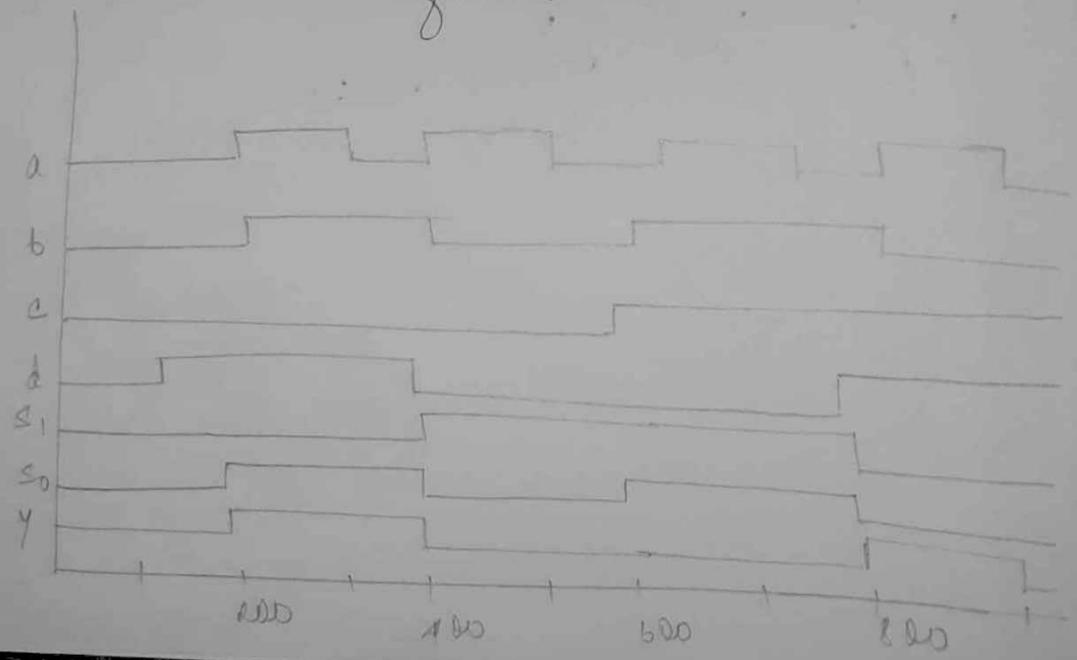
Stimulation waveform:-



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 4to1_mux;
architecture Behavioral of 4to1_mux is
signal s : std_logic_vector(1 down to 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
```

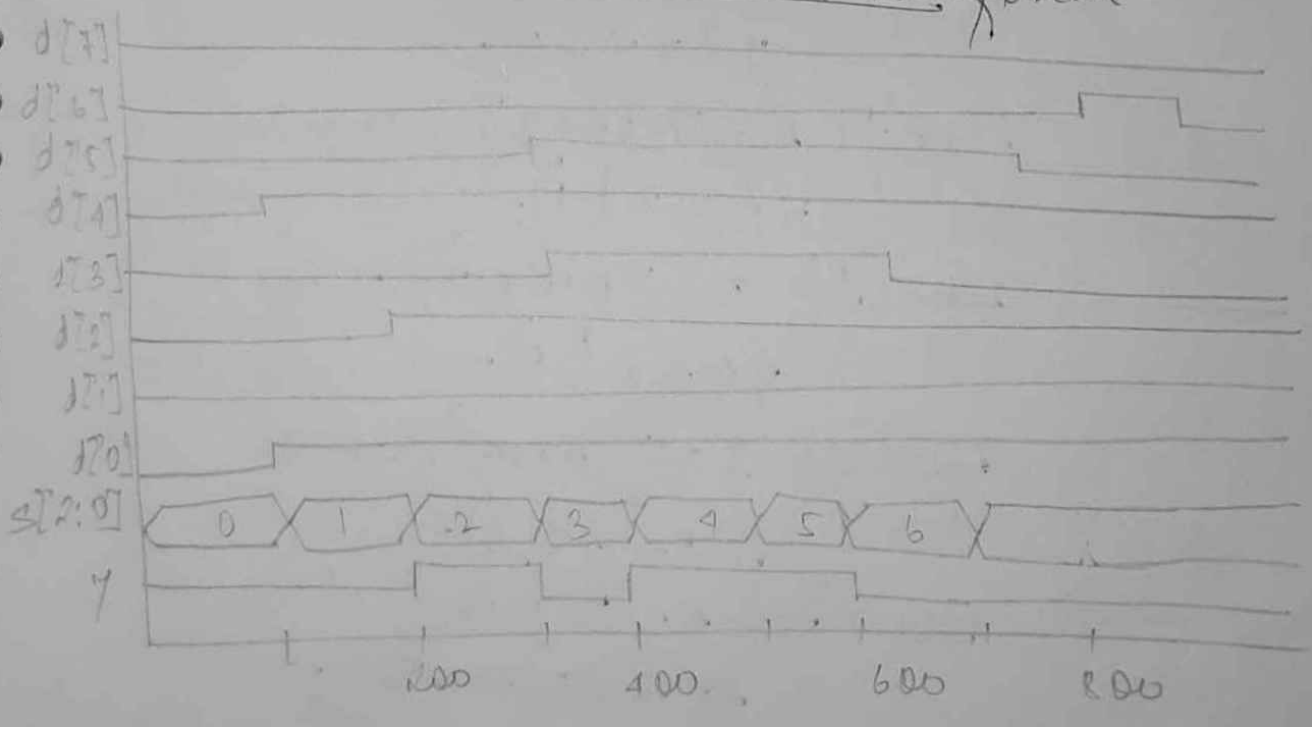
Simulation waveform-



8:1 Mux

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity 8 to 1 mux is
Port( d : in std_logic_vector(7 down to 0);
      s : in std_logic_vector(2 down to 0);
      y : out std_logic);
end 8 to 1 mux;
architecture Behavioral of 8 to 1 mux 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 down to 0);
     s: in std_logic_vector(3 down to 0);
     y: out std_logic);
end mux16;
architecture Behavioral 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);
else if(s="0110") then
    y<=x(6);
else if(s="0111") then
    y<=x(7);
```

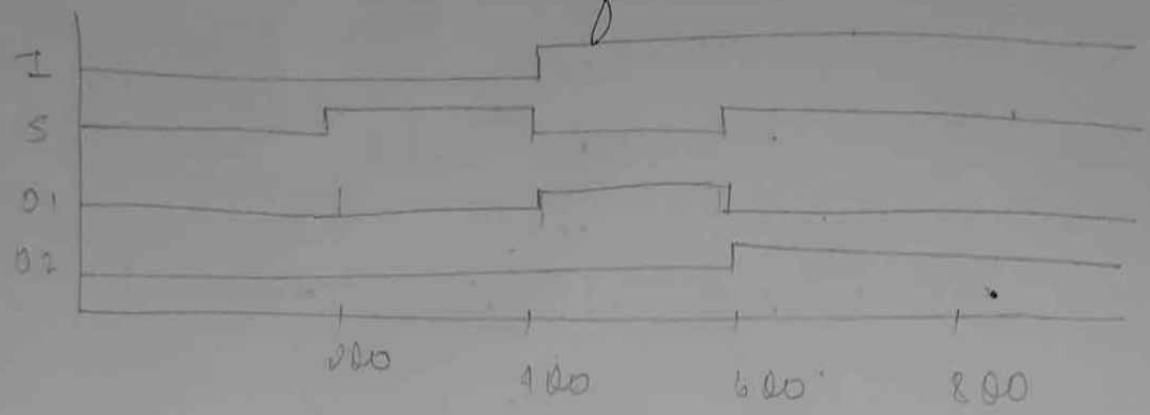
```
elsif (s = "1000") then
        y <= a(8);
elsif (s = "1001") then
        y <= a(9);
elsif (s = "1010") then
        y <= a(10);
elsif (s = "1011") then
        y <= a(11);
elsif (s = "1100") then
        y <= a(12);
elsif (s = "1101") then
        y <= a(13);
elsif (s = "1110") then
        y 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

1) Write a VHDL program to implement de-multiplexers 1:2, 1:4, 1:8 and 1:16.

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;
```

architecture ~~behavioral~~ of dataflow of
Demux 1 to 2 is
begin
O1 <= I and (not S);
O2 <= I and S;
end dataflow;
Stimulation waveform-

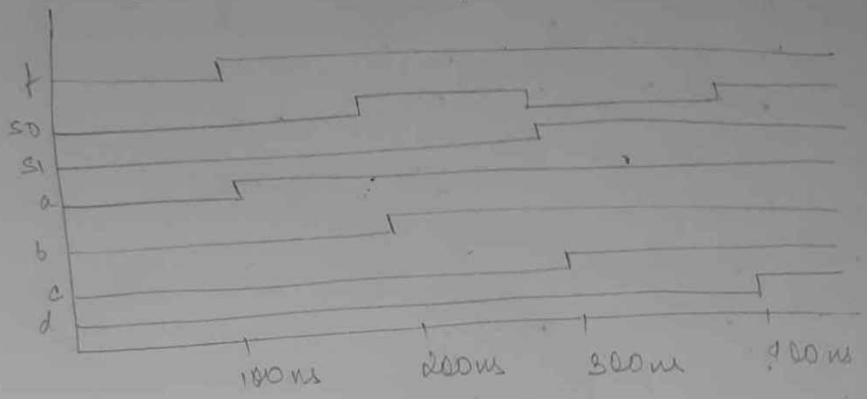


1:4 demux

```
Library ieee;
use ieee.std_logic_1164.all;
entity demux 1 to 4 is
Port ( F : in std_logic;
       S0, S1 : in std_logic;
       A, B, C, D : out std_logic);
end demux_1 to 4;
architecture bhv of demux 1 to 4 is
begin
  Process (F, S0, S1) is
  begin
    if (S0 = '0' and S1 = '0') then
      A <= F;
    else if (S0 = '1' and S1 = '0') then
```

```
        B <= f;
    else
        D <= f;
    end process;
    end bhv;
```

Simulation waveform

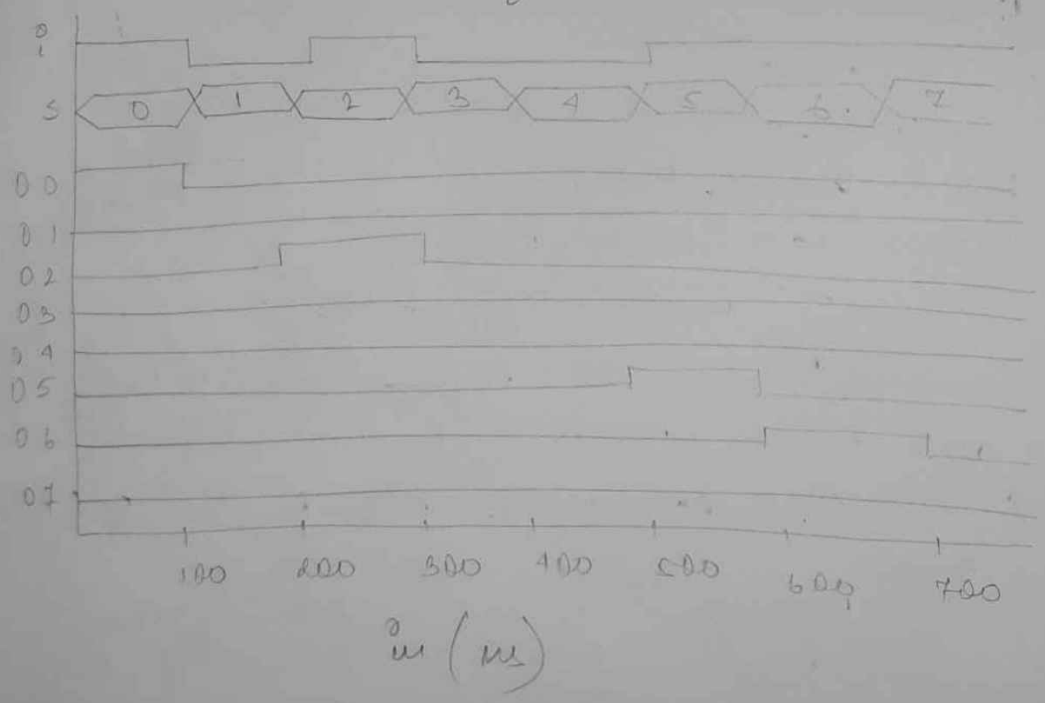


1: 8 demux

```
library ieee;
use ieee.std_logic_1164.all
entity demux1to8 is
    Port( in : 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 demux1to8 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 demux

```
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;
        s: in std_logic (3 down to 0);
        y: out std_logic_vector(15 down to 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 '0111' =>
          y(7) <= x;
```

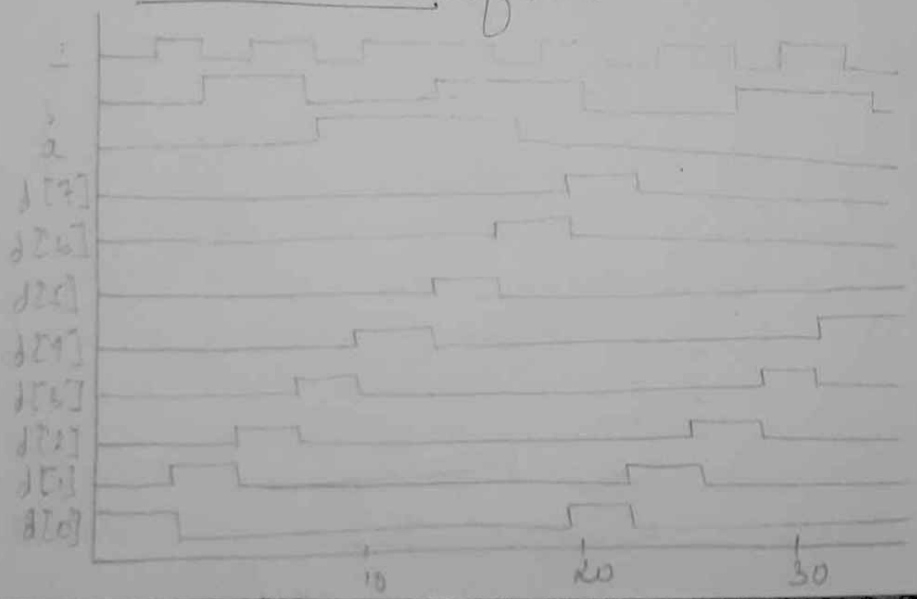
```
when "1000" =>
    y(8) <= x;
when "1001" =>
    y(9) <= x;
when "1010" =>
    y(10) <= x;
when "1100" =>
    y(12) <= x;
when "1011" =>
    y(11) <= x;
when "1101" =>
    y(13) <= x;
when "1110" =>
    y(14) <= x;
when "1111" =>
    y(15) <= x;
end case;
end process;
end behavior;
```

5) 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 down to 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_~~output~~ arch;
```

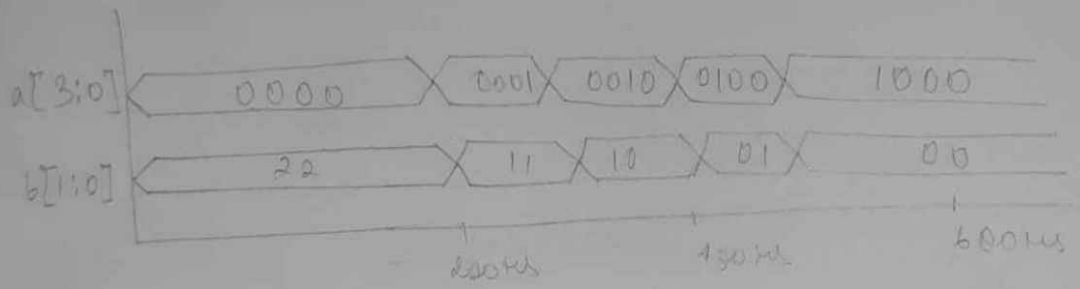
Simulation Waveform



4:2 encoder

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

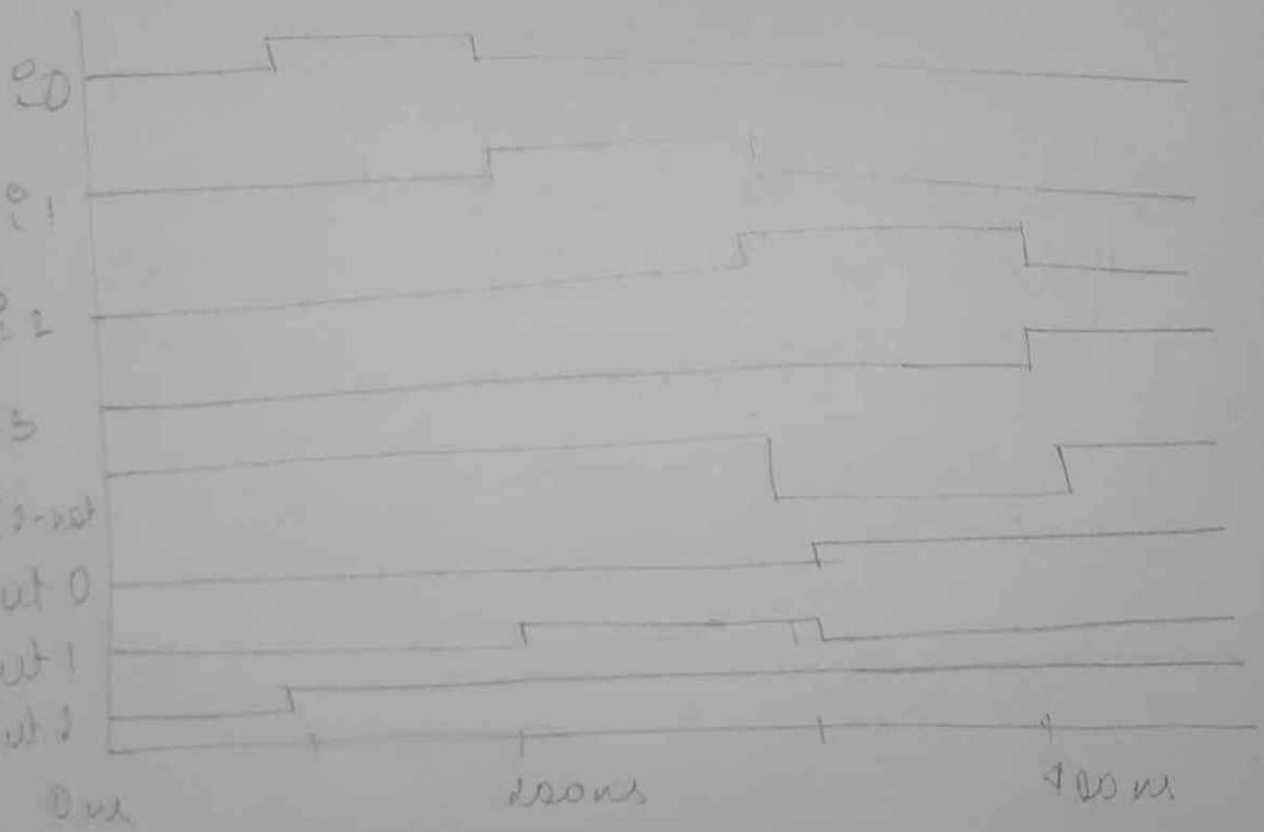


Priority Encoder 4:2

```
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 Dataflow;
```

Simulation Waveform

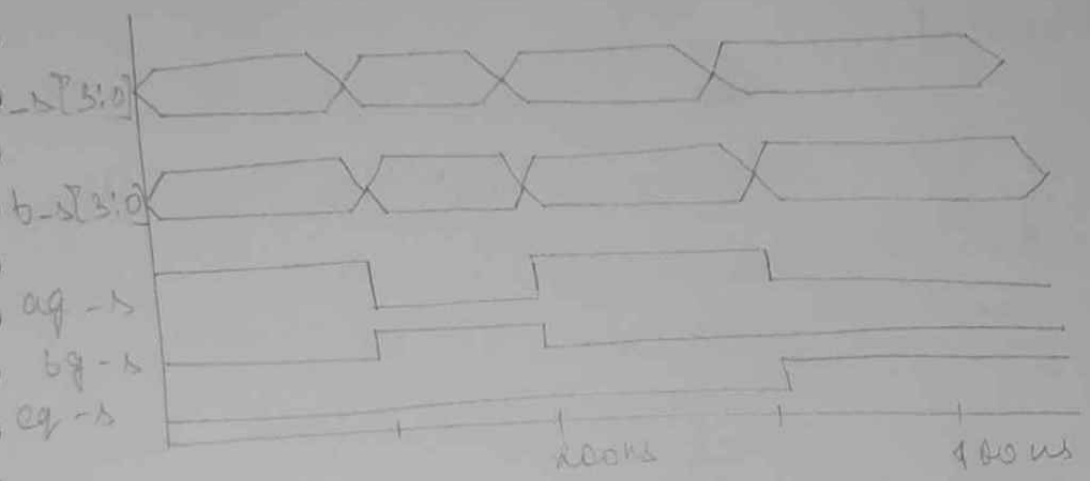


5) Write a VHDL program to implement magnitude comparator circuit

```
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 down to 0);
        ag, bg, eq : out std_logic);
 end component;
 signal a_s, b_s : std_logic_vector(3 down to 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);
  ut_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 bxn;
```

Stimulation Waveform



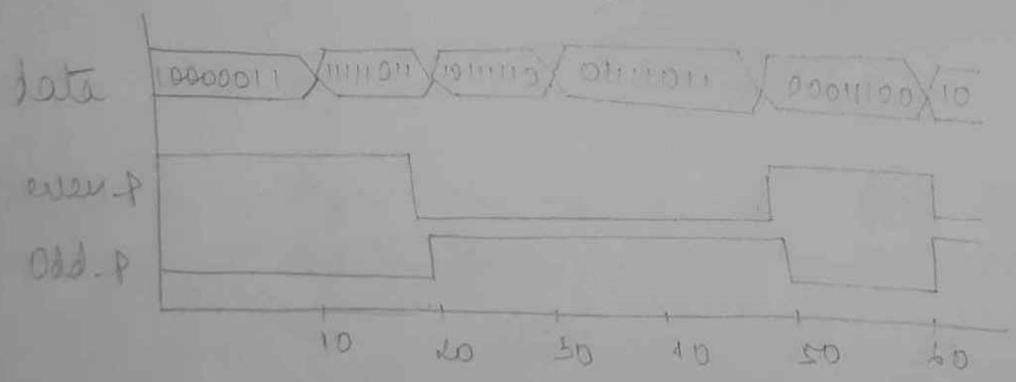
b) Write a VHDL program to implement parity generator and parity checker.

- 8 bit - parity generator

```
library ieee;
use ieee.std_logic_1164.all;
entity parity is
port (data: in bit_vector(7 down to 0)
        even_P, odd_P: out bit);
```

```
end parity;
architecture parity_gen of parity is
  signal temp: bit_vector(5 down to 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)~~
  even_p <= temp(5) xor data(7);
  odd_p <= not (temp(5)) xor data(7));
end parity_gen;
```

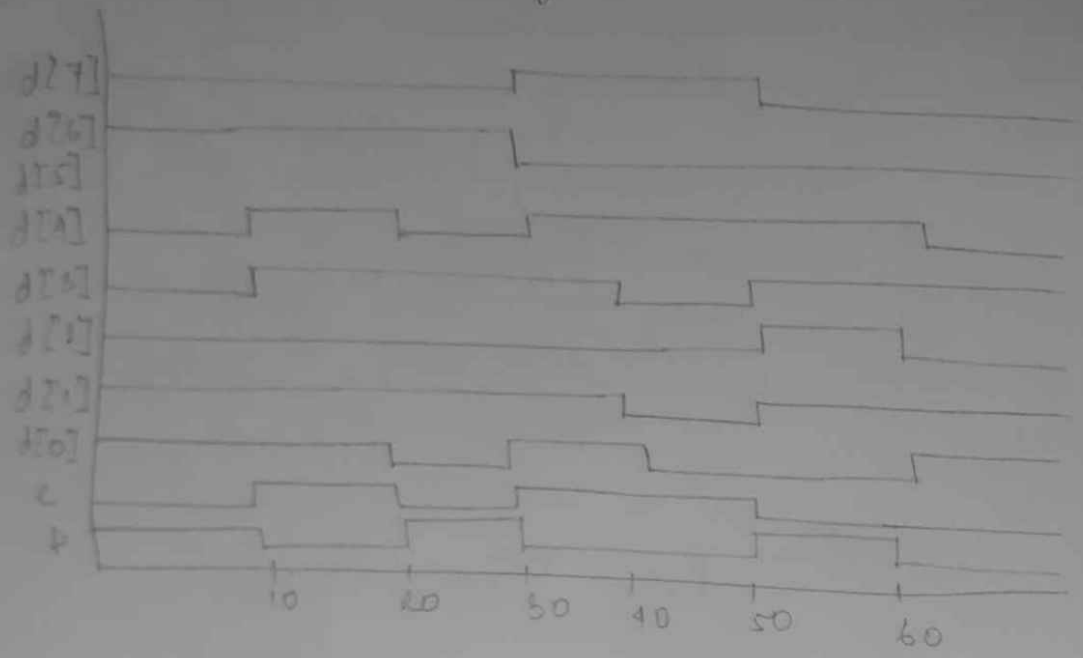
Stimulation waveform



· 8 bit parity checker

```
library ieee;
use ieee.std_logic_1164.all;
entity parity chk is
Port (data : in bit_vector(7 down to 0);
        P : in bit;
        e : out bit);
end parity_chk;
architecture parity_arch of parity_chk is
signal temp : bit_vector(6 down to 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;
```

Simulation waveform



7) Write the VHDL program 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';
```

```
elsif (clk event and clk = '1') then
    ~~P<=Q~~
    elsif (x = '0' and s = '0') then
            P <= P;
    elsif (x = '0' and s = '1') then
            P <= '0';
    elsif (x = '1' and s = '0') then
            P <= '1';
    elsif (x = '1' and s = '1') then
            P <= '-';
    end if;
    end if;
        Q <= P;
end process;
end Behavioral;
```

Simulation Waveform