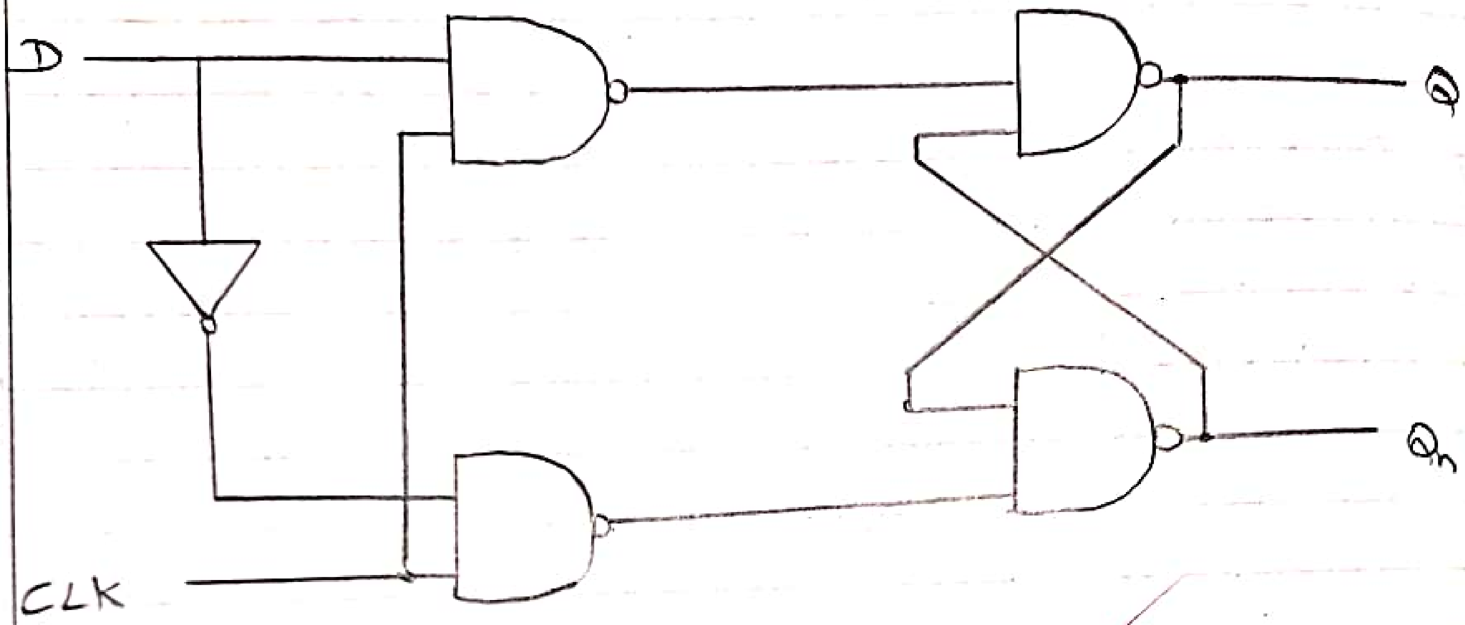


✓

D Flip Flop

Date

Expt No. 1. Page No. 1.

D Flip-Flop

• Objective :-

The D Flip-Flop Stores the value of the D input on the rising edge of the clk signal and outputs the complement on Qn. It also has an asynchronous RESET input to clear the outputs.

• Description:

• Inputs
 - D : Data input
 - Clk : clock signal (triggers state change on rising edge).
 - RESET: Asynchronous reset.

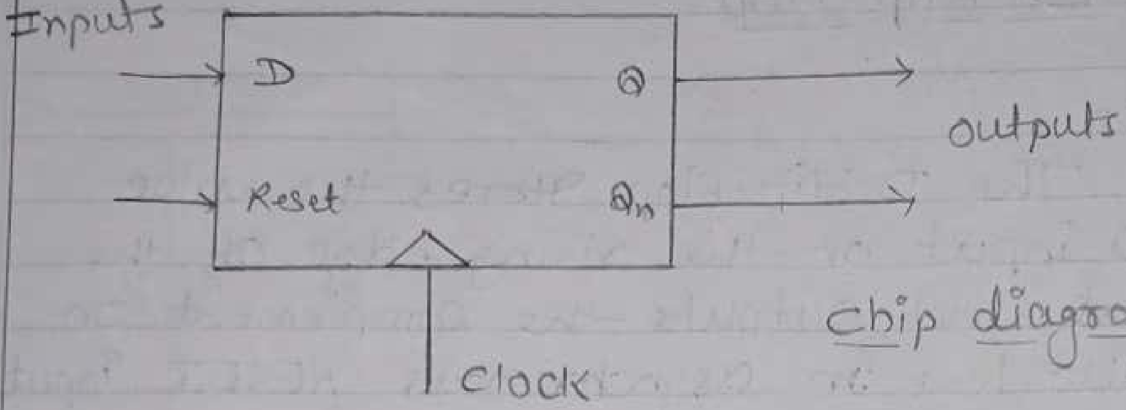
✓

• Outputs
 - Q : Stored value
 - Qn : Complement of Q

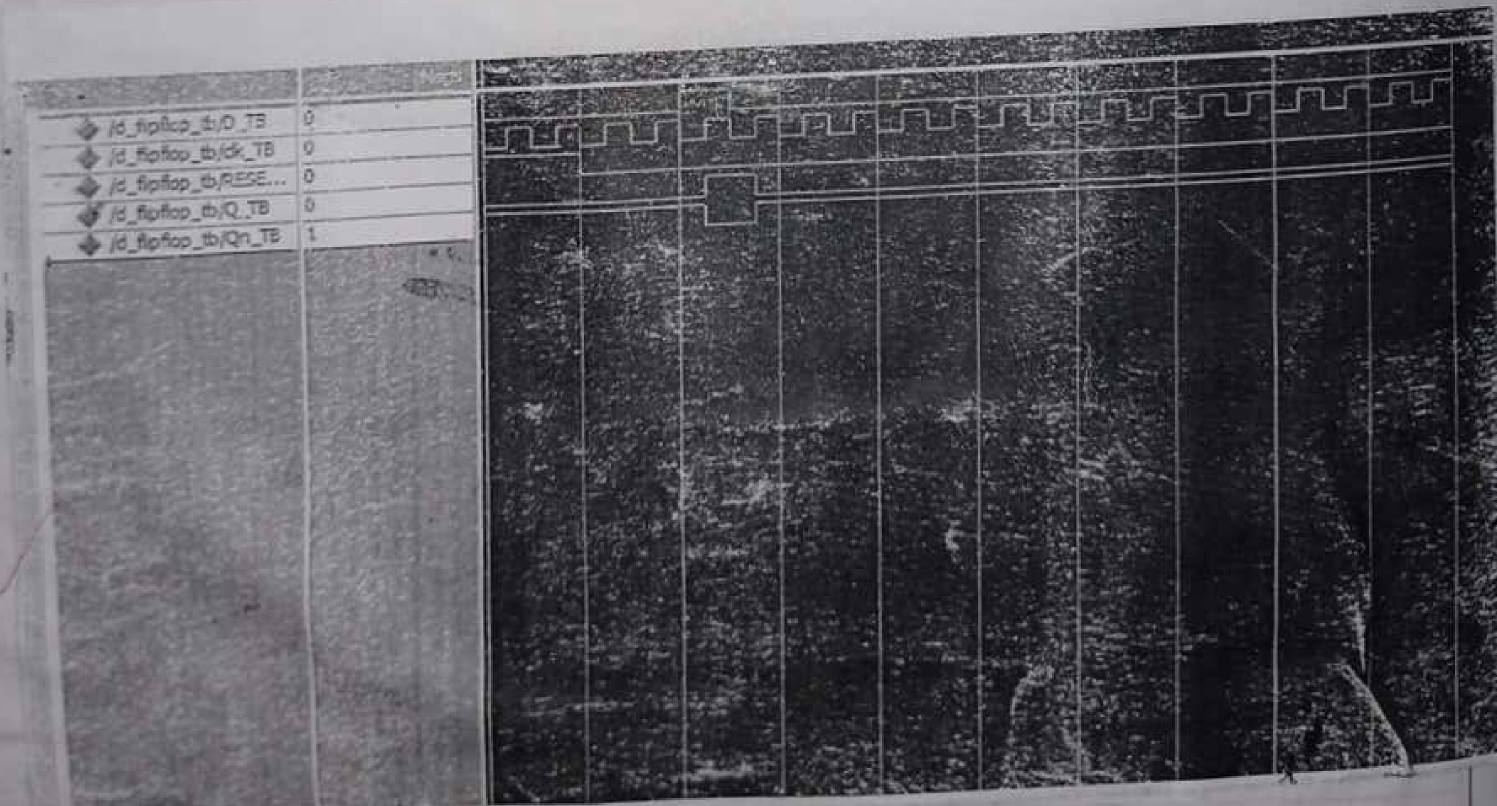
• Behaviour:

On a rising clock edge, Q = D, and Q_n = NOT D. If RESET is high, Q is reset to 0 and Qn to 1.

Teacher's Signature



Chip diagram



```
/d_flipflop_tb/D_TB      0
/d_flipflop_tb/clk_TB    0
/d_flipflop_tb/RESE...   0
/d_flipflop_tb/Q_TB      0
/d_flipflop_tb/Qn_TB     1
```

D Flip-flop

• Truth Table :-

D	RESET	Clk (rising edge)	Q	Qn
0	0	↑	0	1
1	0	↑	1	0
X	1	↑	0	1
0	0	↓	0	1
1	0	↓	0	1

• VHDL Code :-

```
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;
              Clk : in Std_logic;
            RESET : in Std_logic;
                Q : out Std_logic;
               Qn : out std_logic );
end D_FlipFlop;
```

Date
Expt No.

```
architecture behavioral of D-flipflop is
signal Q reg, Qn_reg : std logic := '0';

begin

process (clk, RESET)
begin

    if (RESET = '1') then
                Q_reg <= '0';
                Qn reg <= '1';
    elsif rising_edge (clk) then
✓               Q_reg <= D;
                Qn_reg <= not D;
    end if;
    end process;

            Q <= Q reg;
            Qn <= Qn_reg;

end behavioral;
```

• Test bench :-

Teacher's Signature

Date

Expt No.

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

entity D_FlipFlop_TB is
end D_FlipFlop_TB;

architecture behavioral of D_FlipFlop_TB is

    Signal D_TB, Clk_TB, RESET_TB : std_logic := '0';
    Signal Q_TB, Qn_TB : std_logic;
    Constant Clk_period : time := 10ns;

begin

✓ D_FlipFlop_uut : entity work.D_FlipFlop
    port map (

        D => D_TB,
        Clk => Clk_TB,
      RESET => RESET_TB,
          Q => Q_TB,
          Qn => Qn_TB  );

      Clk_process : process
      begin
```

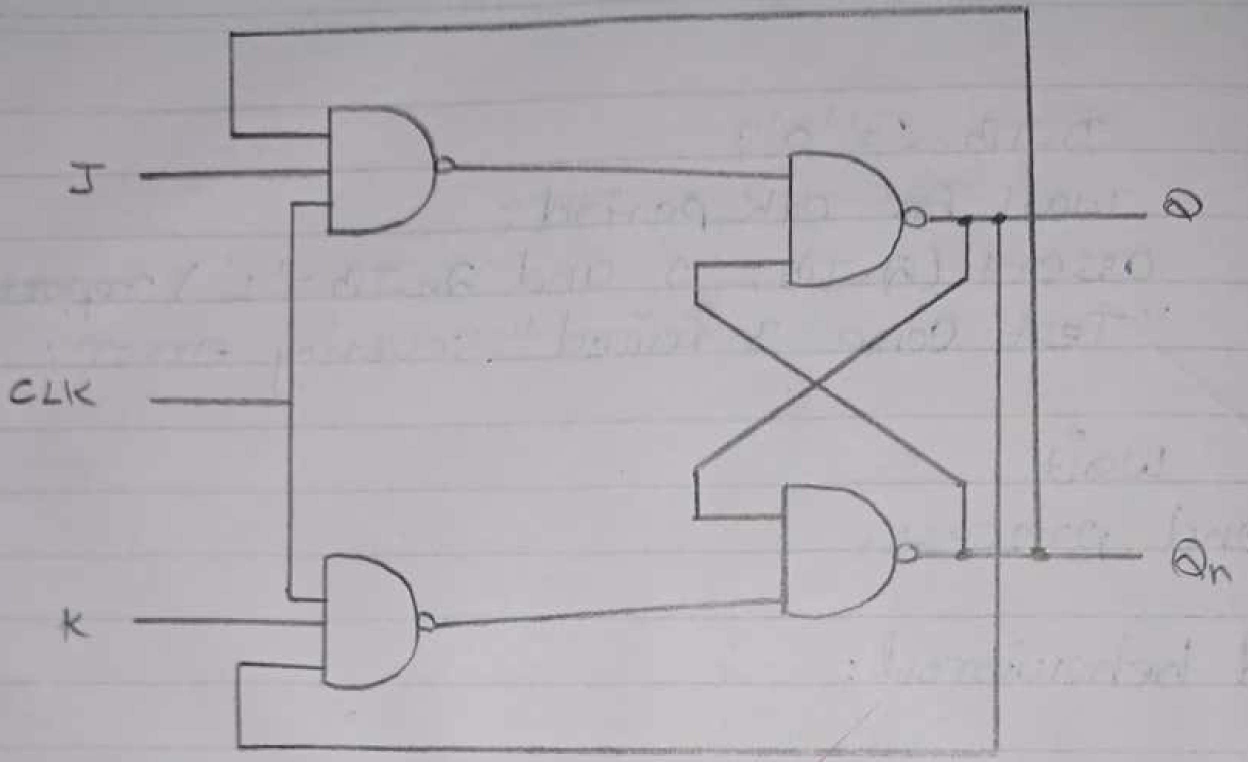
Teacher's Signature

Date

Expt No.

```
    clK_TB <= '0';
    wait for clk_period/2;
    clK_TB  <= '1';
    wait for clk_period/2;
end process;

Stimulus_process : process
begin
    -- Apply reset
    RESET_TB <= '1';
    wait for 20ns;

    -- Deassert reset after Some time
    RESET_TB <= '0';
    wait for 10ns;

    -- Apply Stimulus
    D_TB <= '0';
    wait for clk period;
    assert (Q_TB = '0' and Qn_TB = '1') report
    " Test case 1 failed" Severity error;

    D_TB <= '1';
    wait for clk_period
    assert (Q_TB = '1' and Qn_TB = '0') report
    Case 2 faild" Severity error;
```

Teacher's Signature



JK. FlipFlop

Date

Expt No. 2.

JK FlipFlop

- Objective :-

The Objective of the JK Flip-Flop is to store and toggle a value based on the inputs J and K, triggered by the rising edge of the ClK signal. It provides functionality similar to the SR FlipFlop but with the ability to toggle when both J and K are high.

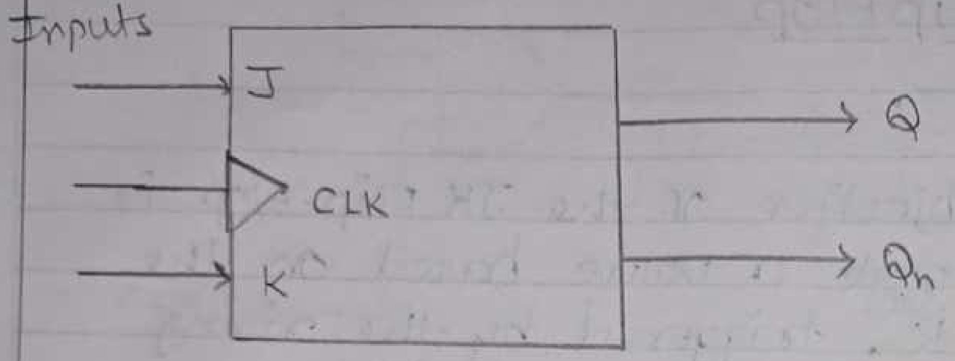
- Description :

  - Inputs
    - J : First input (controls setting of Q).
    - K : Second input (controls resetting of Q).
    - ClK: Clock signal that triggers the flip-flop on the rising edge.

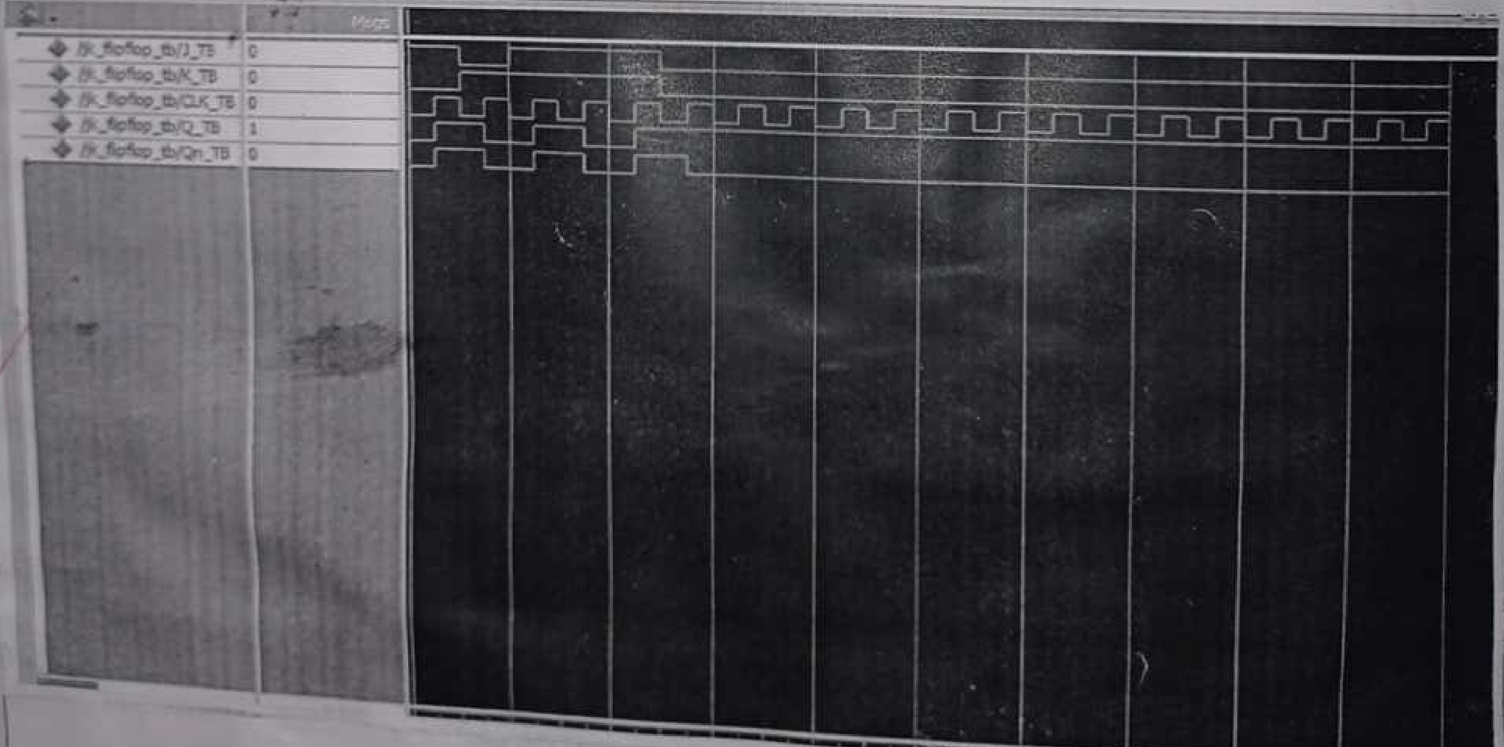
  - Outputs
    - Q : Stored value
    - Q_n : Complement of Q.

  - Behavior :
    - On the rising edge of ClK :

Teacher's Signature



Chip Diagram



```
/jk_flipflop_tb/J_TB     0
/jk_flipflop_tb/K_TB     0
/jk_flipflop_tb/CLK_TB   0
/jk_flipflop_tb/Q_TB     1
/jk_flipflop_tb/Qn_TB    0
```

JK Flip-flop

Expt No.

Date

- If J=1 and K=1, Q toggles.
- If J=1 and K=0, Q is set to 1.
- If J=0 and K=1, Q is reset to 0.
- If J=0 and K=0, Q remains unchanged.

- Truth Table

J	K	CLK (rising edge)	Q	Qn
0	0	↑	Q	Qn
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	T	Tn

- T : Toggles between 0 and 1 on each clock cycle.

- VHDL Code :

```
library ieee;
use ieee.Std_logic_1164.all;
use ieee.Numeric_Std.all;
```

Teacher's Signature

Date

Expt No.

```
entity JK FlipFlop is
        port (
                J : in std_logic;
                K : in std_logic;
              CLK : in std_logic;
                Q : out std_logic;
               Qn : out std_logic );

end JK_FlipFlop;

architecture behavioral of JK_FlipFlop is
       signal Q_reg, Qn_reg : std_logic := '0';
begin
      process (CLK)
       begin
          if rising_edge (CLK) then
             if J='1' and K='1' then
                Q_reg <= not Q_reg;
             elsif J='1' then
                 Q_reg <= '1';
             elsif K='1' then
                 Q_reg <= '0';
             end if
             Qn_reg <= not Q_reg;
          end if
      end process;
```

Teacher's Signature

```
        Q <= Q_reg;
        Qn <= Qn_reg;

end behavioral;
```

• Testbench:

```
library ieee;
use ieee.Std_logic_1164.all;
use ieee.Numeric_std.all;

entity JK_FlipFlop_TB is
end JK_FlipFlop_TB;

architecture Behavioral of JK_FlipFlop_TB is
        signal J_TB, K_TB, CLK_TB : Std_logic := '0';
        signal Q_TB, Qn_TB : Std_logic;

        Constant Clk_Period : time := 10 ns;

begin
        --
        JK_FlipFlop_UUT : entity work.JK_FlipFlop
                port map (
```

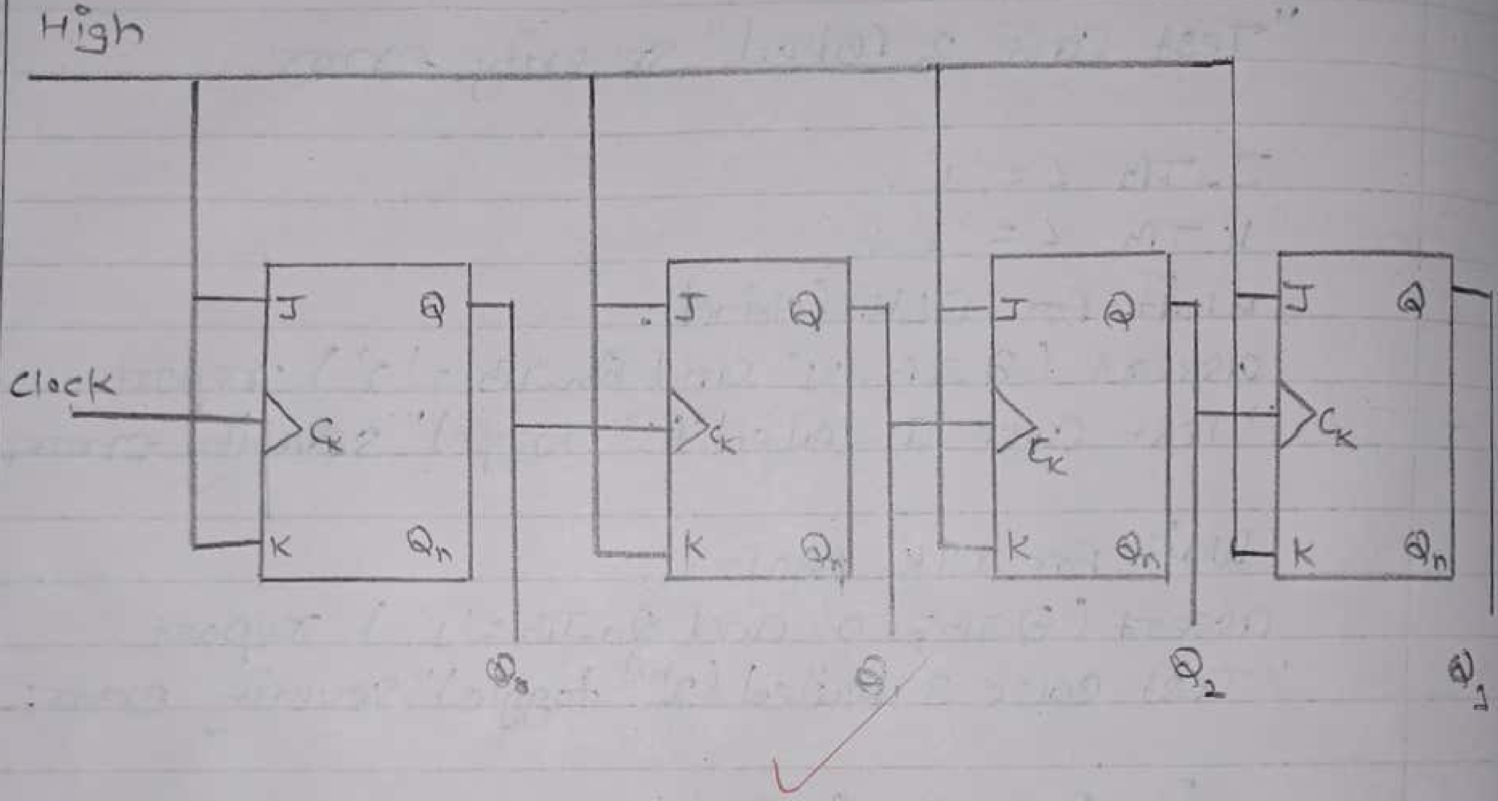
Date

Expt No.

```
        J => J_TB,
        K => K_TB,
      ClK => ClK_TB,
        Q => Q_TB,
       Qn => Qn_TB );

Clock_Process : process
begin
    ClK_TB <='0';
    wait for ClK_period /2;
    ClK_TB <='1';
    wait for ClK_Period /2;
end process;

stimulus_process : process
begin
    J_TB <='1';
    K_TB <='0';
    wait for ClK_Period;
    assert (Q_TB ='1' and Qn_TB ='0') report
    "Test case 1 failed" Severity error;

    J_TB <='0';
    K_TB <='1';
    wait for ClK_Period;
    assert (Q_TB='0' and Qn_TB='1') report
```

Teacher's Signature

```
"Test Case 2 failed" severity error;

J_TB <='1';
K_TB <='1';
wait for CLK_Period;
assert (Q_TB,'1' and Qn_TB ='0') report
"Test Case 3 failed (1st Toggle)" severity error;

wait for clk_period;
assert (Q_TB='0' and Qn_TB ='1') report
"Test case 3 failed (2nd toggle)" severity error;

wait for clk_Period;
assert (Q_TB ='1' and Qn_TB ='0') report
"Test Case 3 failed (3rd toggle)" Severity error;

J_TB <='0';
K_TB <='0';
wait for Clk_Period;
assert (Q_TB ='1' and Qn_TB ='0') report
"Test Case 4 failed (hold previous value)"
Severity error;

wait;
end process;
end Behavioral;
```



4-bit Asynchronous Counter

Date

Expt No. 3. Page No. 13

Counter

- Objective :-

The Objective Of this is to implement a 4-bit binary Counter that increments on each rising edge of the clk Signal and can be reset asynchronously by the rst Signal.

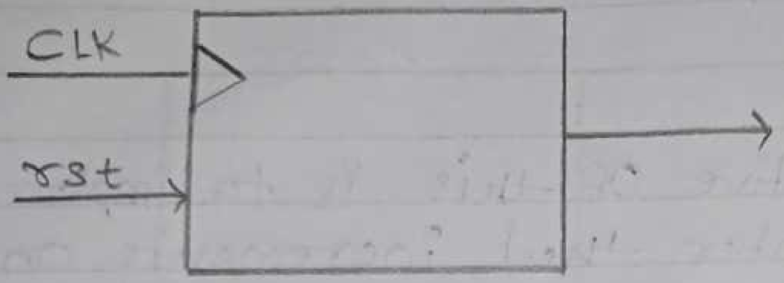
- Description:

  - Inputs:
    - clk : The clock Signal that triggres the Counter to increment.
    - rst : The reset Signal that resets the Counter value to 0 when hight ('1').

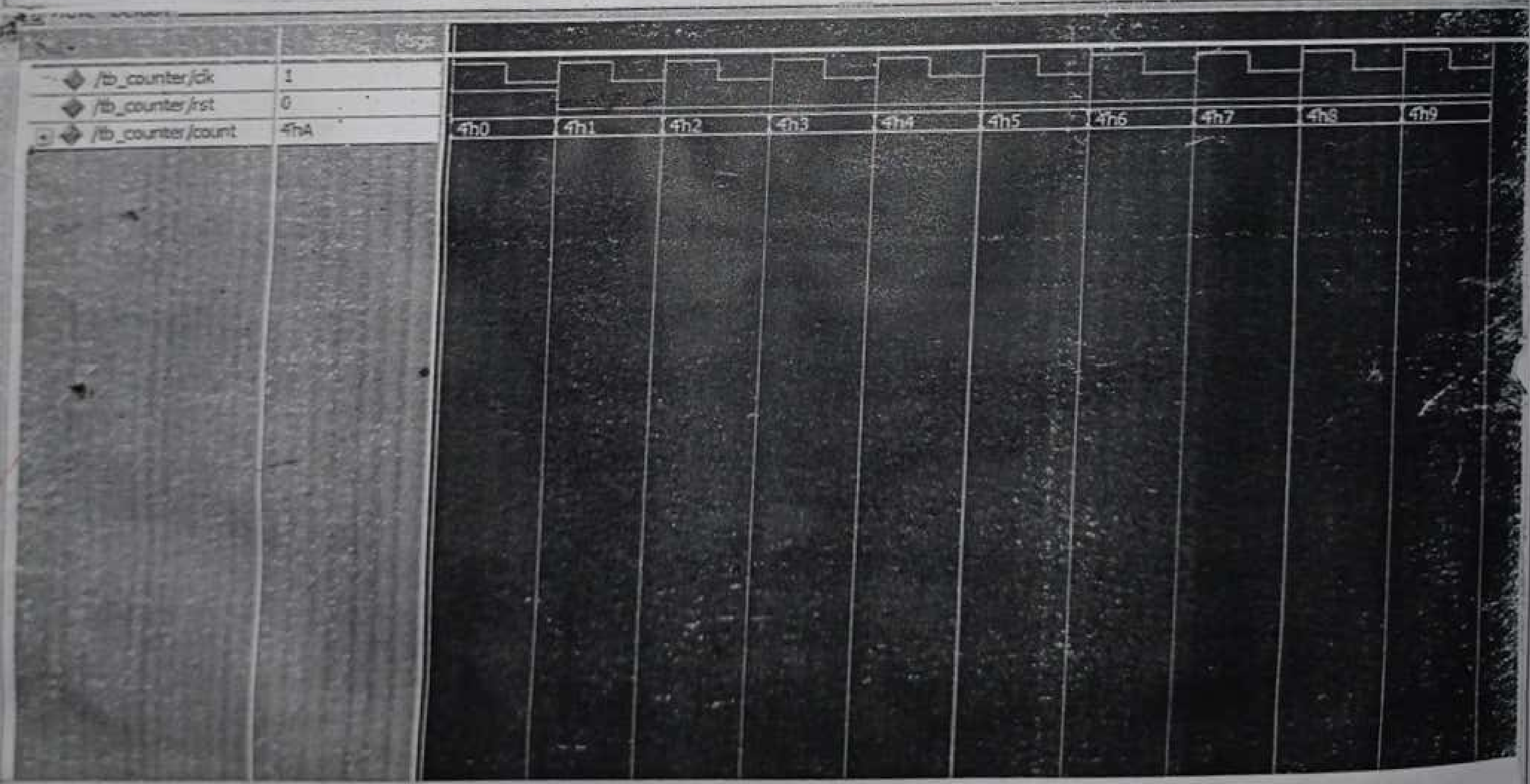
  - Outputs:
    - Count : A 4-bit vector that holds the current Count value.

  - Behaviour:
    - The Counter Starts from "0000"
    - On each rising edge of clk, the Counter increments by 1.
    - When rst is high ('1'), the counter resets to "0000" asynchronously, regardless of the clock signal.

Teacher's Signature



Chip diagram



```
/tb_counter/clk     1
/tb_counter/rst     0
/tb_counter/count   4'hA
```

Counter

• Truth Table :-

clk (rising edge)	rst	Count (4-bit)
↑	0	Cnt + 1
↑	1	0000
(no edge)	0	no change
(no edge)	1	0000

• ↑ : Rising edge of clk.
• Cnt +1 : The Counter increments by 1 on each rising clock edge if rst is 0.

• VHDL Code :

```
library ieee;
use IEEE.Std_logic_1164.all;
use IEEE.Std_logic_Arith.all;
use IEEE.Std_logic_Unsigned.all;

entity Counter is
        Port ( rst : in Std_logic ;
               clk : in Std_logic ;
               Count : out Std_logic_vector (3 downto 0) );
end Counter;
```

Date

Expt No.

```
architecture behavioral of counter is
    signal cnt : std_logic_vector (3 downto 0) := "0000";

begin
        process (clk, rst)
      begin
            if rst = '1' then
                cnt <= "0000";
           elsif rising_edge (clk) then
                cnt <= cnt + 1;
          end if;
      end process;

      count <= cnt;
end behavioral;
```

• Testbench Code:

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

entity tb_counter is
end tb_counter;
```

Teacher's Signature

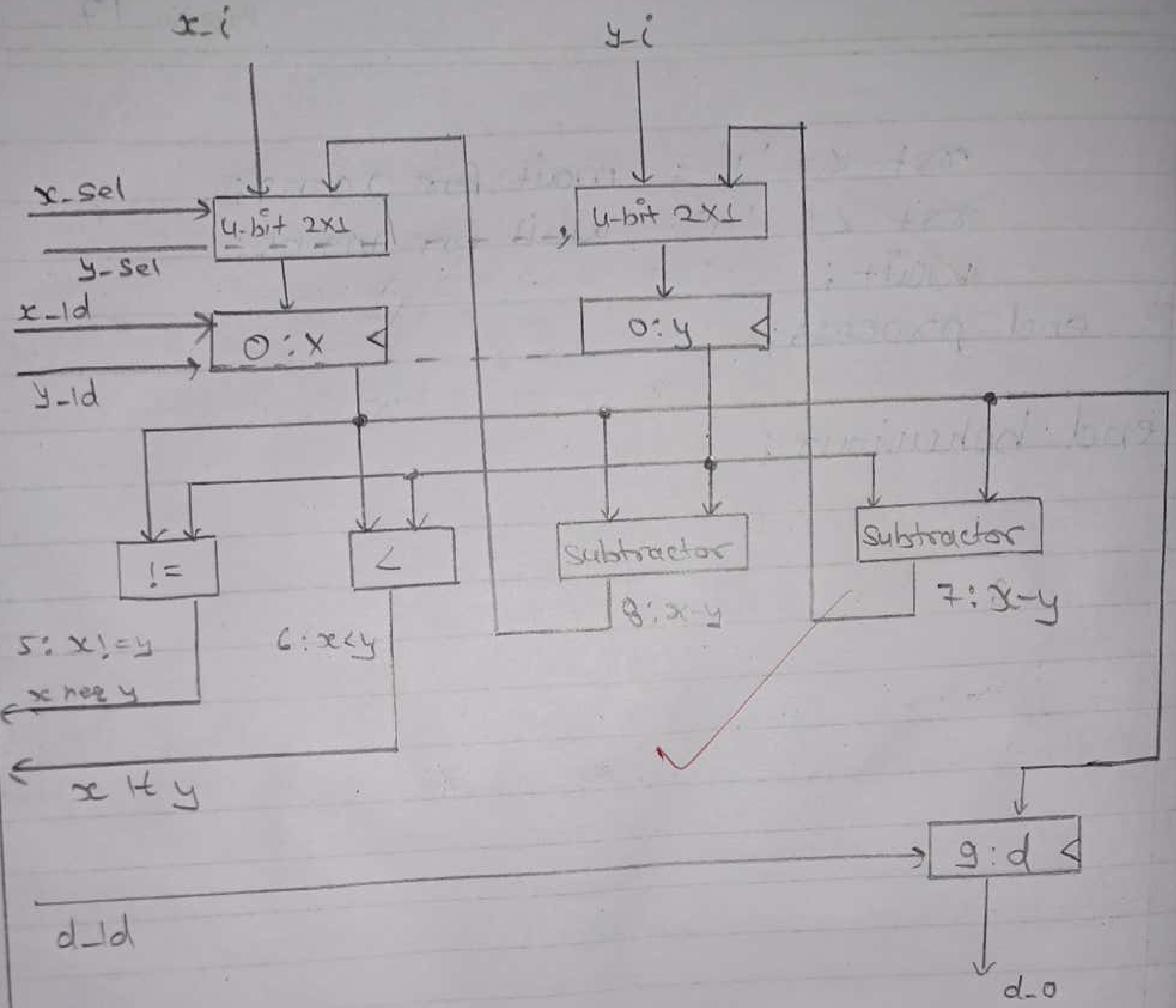
```
architecture behaviour of tb-counter is

    Signal clk : std_logic := '0';
    signal rst : std_logic := '0';
    signal Count : std_logic_vector (3 downto 0);

    Component Counter is
        port (
                clk : in std_logic;
                rst : in std_logic;
                Count : out std_logic vector (3 downto 0));
    end Component;

begin
    uut : Counter Port map (clk => clk, rst => rst,
            Count => Count );

    clk_process : process
    begin
        clk <= not clk ;
        wait for 10ns;
    end process;

    Stim_proc : process
    begin
```

Date

xpt No.   Page No. 17

```
        rst <= '1'; wait for 20 ns;
        rst <= '0'; wait for 100 ns;
        wait;
    end process;
end behaviour;
```

Teacher's Signature



x_i
y_i
x_sel
y_sel
4-bit 2x1
4-bit 2x1
x_ld
y_ld
0 : x
0 : y
!=
<
subtractor
subtractor
5: x!=y
x neq y
6: x<y
x lt y
8: x-y
7: x-y
9 : d
d_ld
d_o

Date

Expt No. 4. Page No. 18

GCD

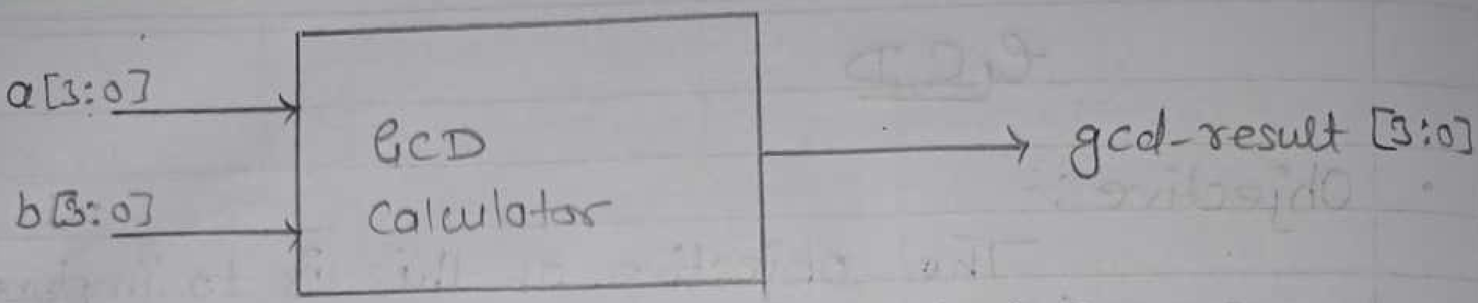
- Objective :-

  The objective of this is to implement a GCD (Greatest Common Divisor) Calculator for two 4-bit inputs. It Computes the GCD of two numbers a and b and outputs the result as a 4-bit vector.

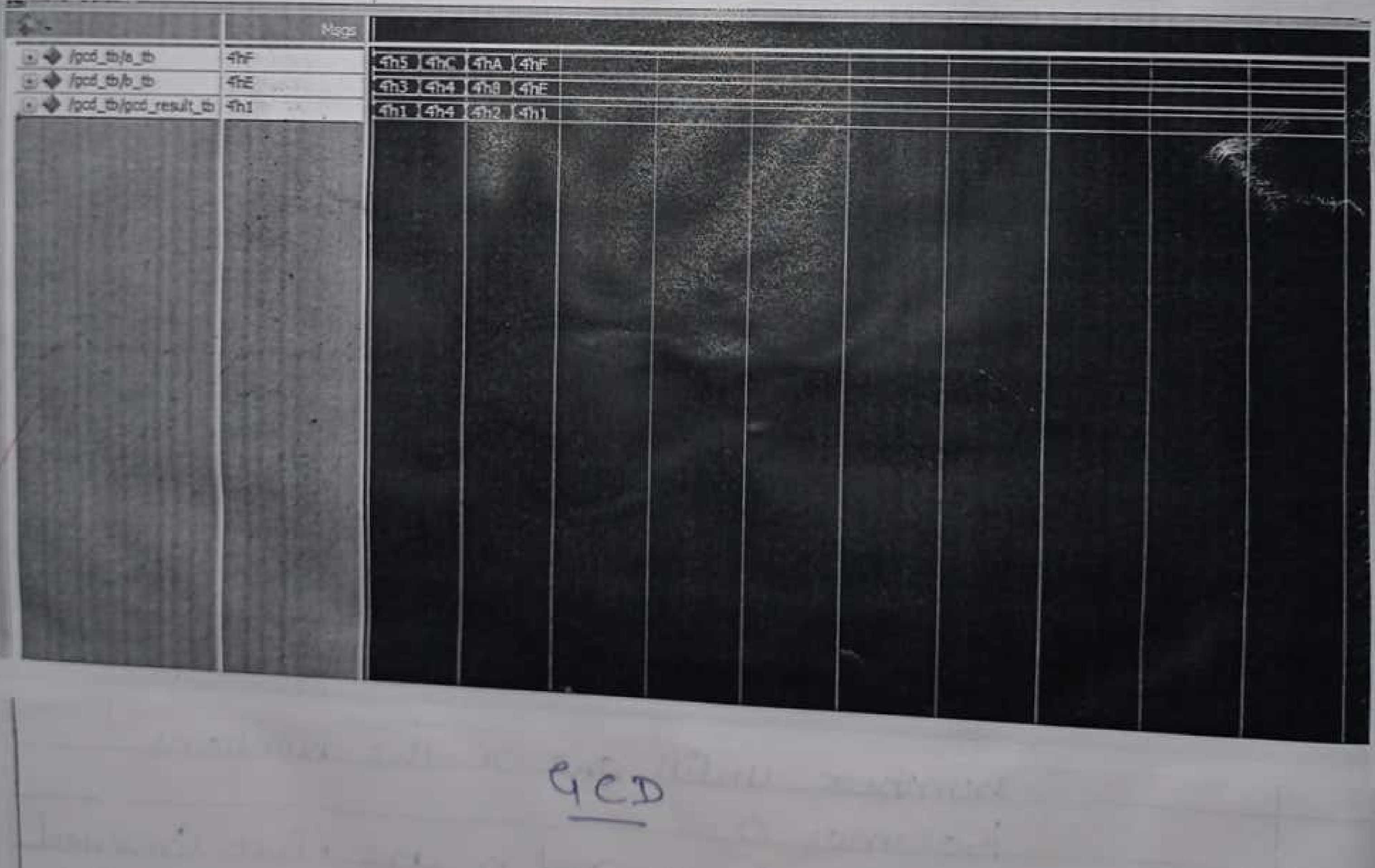
- Description :-
  - Inputs :
    - a : 4-bit first number
    - b : 4-bit Second number
  - Outputs
    - Gcd result : A 4-bit Vector that holds the computed GCD of a and b.

  - Behavior:
    - The GCD is Computed using the Euclidean algorithm, which repeatedly subtracts the Smaller number from the larger number untill one of the numbers becomes 0.
    - The inputs a and b are first Converted to integers. The GCD function Calculates the GCD of these Integers.

Teacher's Signature



chip Diagram



```
/gcd_tb/a_tb           4'hF    4'h5  4'hC  4'hA  4'hF
/gcd_tb/b_tb           4'hE    4'h3  4'h4  4'h8  4'hE
/gcd_tb/gcd_result_tb  4'h1    4'h1  4'h4  4'h2  4'h1
```

GCD

The result is then Converted back to a 4-bit std logic-vector and assigned to gcd-result.

- Truth Table :-

a(4-bit)	b(4-bit)	Gcd result (4-bit)
0000	0000	0000
0001	0001	0001
0010	0010	0010
0100	0010	0010
0101	0010	0001
0110	0011	0001
1110	1100	0010
1111	1011	0001

- VHDL code :-

```
library ieee;
use ieee.std-logic 1164.all;
use ieee.numeric std.all;
```

Date

Expt No.

```
entity gcd is
    port (
          a, b : in Std_logic_Vector (3 downto 0);
      GCd_result: out Std_logic_Vector (3 downto 0)
    );
end gcd;

architecture Behavioral of GCD is
      function GCD_function (a_val, b_val : integer)
      return integer is
            variable a_temp, b_temp : integer;
      begin
          a_temp := a_val;
          b_temp := b_val;
          while b_temp /= 0 loop
              if a_temp > b_temp then
                    a_temp := a_temp - b_temp;
              else
                   b_temp := b_temp - a_temp;
              end if;
          end loop;
          return a_temp;
      end GCD_function;

  begin
```

Teacher's Signature

Date

Expt No.

```
    process (a,b)
        variable a_int, b_int, gcd_val : integer;
    begin
        a_int := to_integer(unsigned(a));
        b_int := to_integer (unsigned (b));
        gcd_val := GCD_Function(a_int, b_int);
        gcd_result <= std_logic_vector (to_unsigned(
                        gcd_val, 4));
    end process;
end behavioral;
```

• Testbench code :-

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

entity gcd_tb is
end gcd_tb;

architecture testbench of gcd_tb is

    component gcd
        port (
```

Teacher's Signature

Date

Expt No.

```
        ·a, b : in Std-logic vector (7 downto 0);
     gcd-result: out Std logic vector (7 downto 0)
   );
end component;

Signals a-tb, b-tb : std-logic vector (7 downto 0);
Signal gcd-result tb : Std-logic vector (7 downto 0);

Constant CLOCK-PERIOD : time := 10 ns;
begin
CLK Process : process
 begin
     while true loop
           wait for CLOCK-PERIOD /2;
     end loop;
 end process ClK-process;

 stim-proc : process
 begin

     a-tb <= "0101";
     b-tb <= "0011";
     wait for 10 ns;
```

Teacher's Signature

Expt No.

Date

Page No. 23

```
        a_tb <= "1100";
        b_tb <= "0100";
        wait for 10 ns;

        a_tb <= "1010";
        b_tb <= "1000";
        wait for 10 ns;

        a_tb <= "1111";
        b_tb <= "1110";
        wait for 10 ns;

        wait;
    end process stim_proc;

    DUT: gcd
    port map (

            a => a_tb;
            b => b_tb;
            gcd_result => gcd_result_tb
    );

end testbench;
```

[illegible] 28/1/25

Teacher's Signature