Logic design 2

Lab 3

Name & IDs:

كريم فتحي عبد العزيز محمد مصطفى (20011116)

احمد محمد حسن محمد صالح (20010174)

عمر محمود عبد الوهاب السيد(20011027)

محمد احمد علي رياض علي (20011457)

فارس محمد انور رزق (20011065)

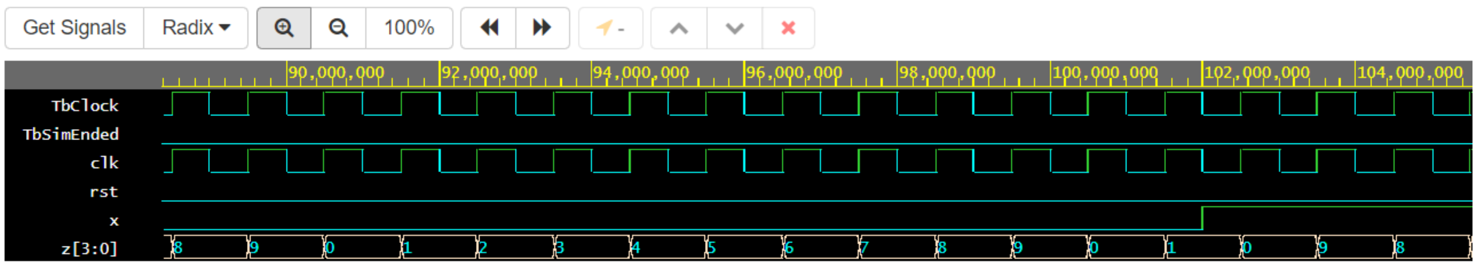
محمد عمرو عبد الفتاح (20011675)

Problem Statement

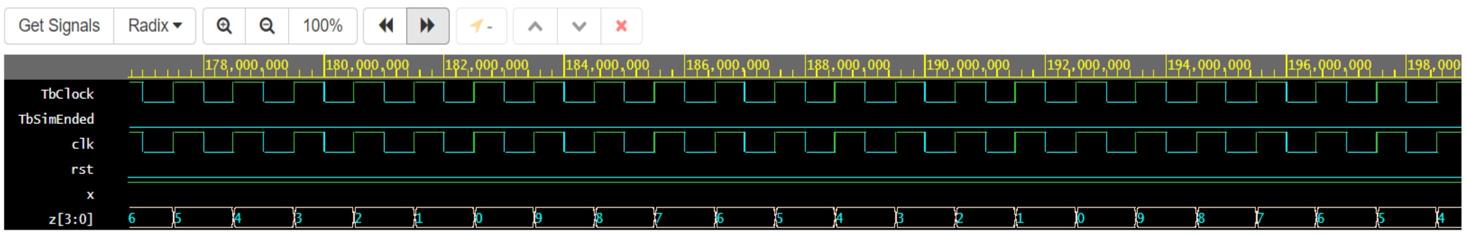
Design and implement a 4-bit BCD up/down counter. The counter should work as follows:

- If input X = 0, the counter counts up. Otherwise, it counts down.
- If counting up, the counter’s value should be: 0000, 0001, 0010...
- If counting down: 0010, 0001, 0000…

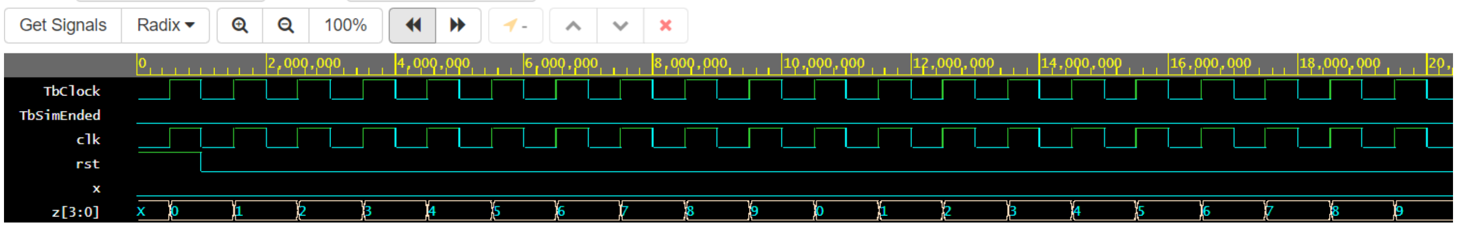
Simulation samples:



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

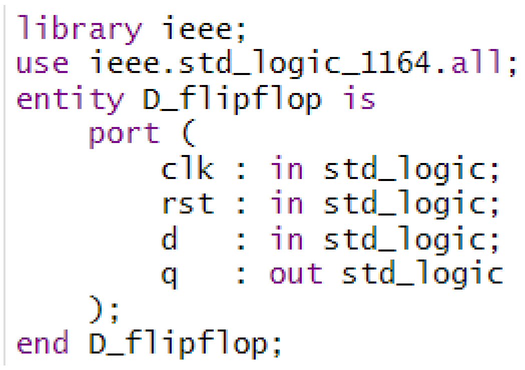


Note: To revert to EPWave opening in a new browser window, set that option on your user page.

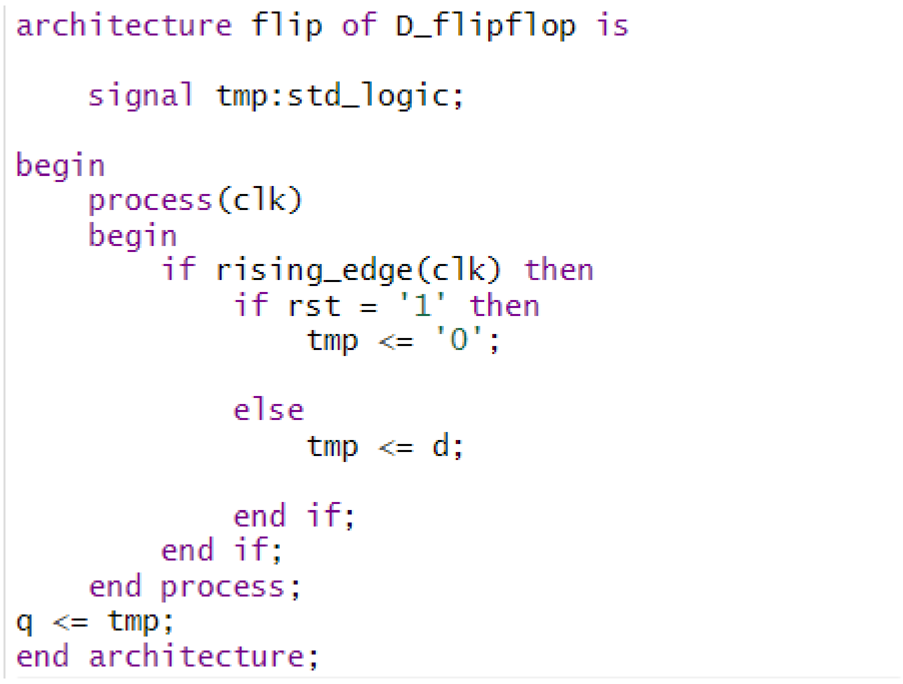


Note: To revert to EPWave opening in a new browser window, set that option on your user page.

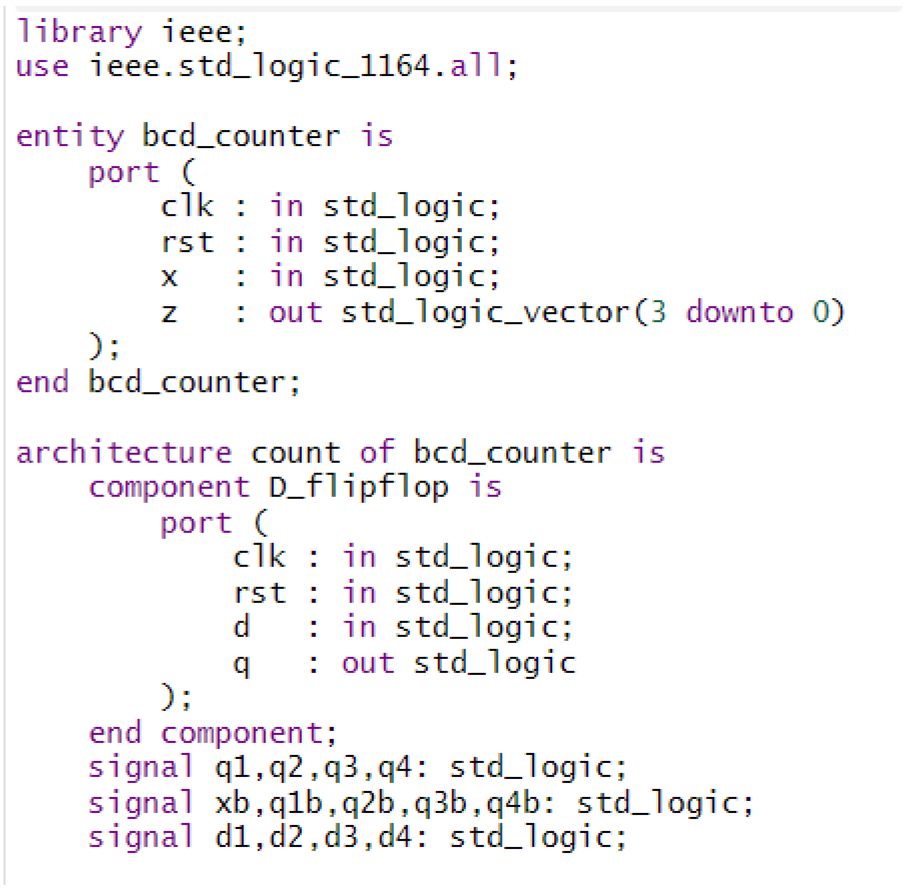
code snippets:



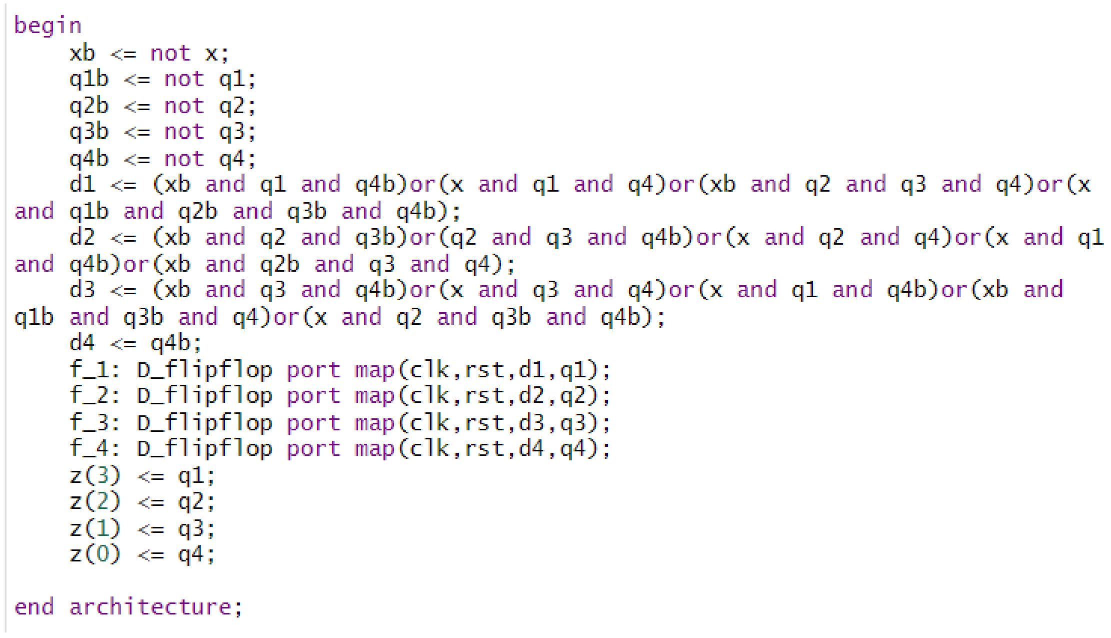
```
library ieee;
use ieee.std_logic_1164.all;
entity D_flipflop is
    port (
        clk : in std_logic;
        rst : in std_logic;
        d   : in std_logic;
        q   : out std_logic
    );
end D_flipflop;
```



```
architecture flip of D_flipflop is

    signal tmp:std_logic;

begin
    process(clk)
    begin
        if rising_edge(clk) then
            if rst = '1' then
                tmp <= '0';

            else
                tmp <= d;

            end if;
        end if;
    end process;
q <= tmp;
end architecture;
```



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

entity bcd_counter is
    port (
        clk : in std_logic;
        rst : in std_logic;
        x   : in std_logic;
        z   : out std_logic_vector(3 downto 0)
    );
end bcd_counter;

architecture count of bcd_counter is
    component D_flipflop is
        port (
            clk : in std_logic;
            rst : in std_logic;
            d   : in std_logic;
            q   : out std_logic
        );
    end component;
    signal q1,q2,q3,q4: std_logic;
    signal xb,q1b,q2b,q3b,q4b: std_logic;
    signal d1,d2,d3,d4: std_logic;
```

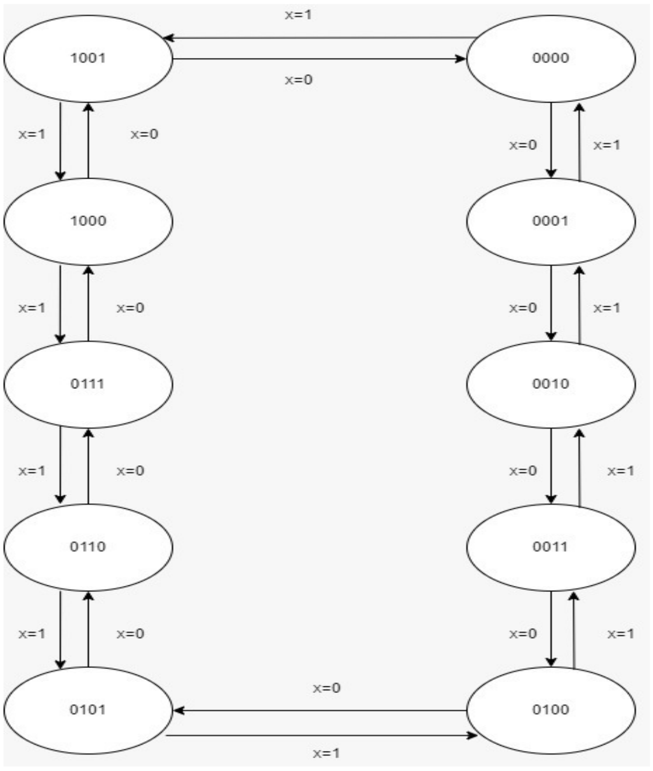


```vhdl
begin
    xb <= not x;
    q1b <= not q1;
    q2b <= not q2;
    q3b <= not q3;
    q4b <= not q4;
    d1 <= (xb and q1 and q4b)or(x and q1 and q4)or(xb and q2 and q3 and q4)or(x
and q1b and q2b and q3b and q4b);
    d2 <= (xb and q2 and q3b)or(q2 and q3 and q4b)or(x and q2 and q4)or(x and q1
and q4b)or(xb and q2b and q3 and q4);
    d3 <= (xb and q3 and q4b)or(x and q3 and q4)or(x and q1 and q4b)or(xb and
q1b and q3b and q4)or(x and q2 and q3b and q4b);
    d4 <= q4b;
    f_1: D_flipflop port map(clk,rst,d1,q1);
    f_2: D_flipflop port map(clk,rst,d2,q2);
    f_3: D_flipflop port map(clk,rst,d3,q3);
    f_4: D_flipflop port map(clk,rst,d4,q4);
    z(3) <= q1;
    z(2) <= q2;
    z(1) <= q3;
    z(0) <= q4;

end architecture;
```

Test :

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

entity tb_bcd_counter is
end tb_bcd_counter;

architecture tb of tb_bcd_counter is

    component bcd_counter
        port (clk : in std_logic;
              rst : in std_logic;
              x   : in std_logic;
              z   : out std_logic_vector (3 downto 0));
    end component;

    signal clk : std_logic;
    signal rst : std_logic;
    signal x   : std_logic;
    signal z   : std_logic_vector (3 downto 0);

    constant TbPeriod : time := 1000 ns; -- EDIT Put right period here
    signal TbClock : std_logic := '0';
    signal TbSimEnded : std_logic := '0';


begin

    dut : bcd_counter
    port map (clk => clk,
              rst => rst,
              x   => x,
              z   => z);

    -- Clock generation
    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

    -- EDIT: Check that clk is really your main clock signal
    clk <= TbClock;

    stimuli : process
    begin
        -- EDIT Adapt initialization as needed
        x <= '0';

        -- Reset generation
        -- EDIT: Check that rst is really your reset signal
        rst <= '1';
        wait for TbPeriod;
        rst <= '0';
        wait for TbPeriod;
        -- EDIT Add stimuli here
        x <='0';
        wait for 100 * TbPeriod;
        x <='1';
        wait for 100 * TbPeriod;
        -- Stop the clock and hence terminate the simulation
        TbSimEnded <= '1';
        wait;
    end process;

end tb;
```

Minimized State diagram:



Transition table:

Present state	Next state	
	X=0	X=1
0000	0001	1001
0001	0010	0000
0010	0011	0001
0011	0100	0010
0100	0101	0011

0101	0110	0100
0110	0111	0101
0111	1000	0110
1000	1001	0111
1001	0000	1000

KMaps & Equation:

D1 = $\bar{X}.Q1.Q4 + X.Q1.Q4 + \bar{X}.Q2.Q3.Q4 + X\overline{Q1}.\overline{Q2}.\overline{Q3}.\overline{Q4}$

0	0	0	0
0	0	1	0
X	X	X	X
1	0	X	X
1	0	0	0
0	0	0	0
X	X	X	X
0	1	X	X

$$D2 = \bar{X}.Q2.\overline{Q3} + Q2.Q3.\overline{Q4} + X.Q2.Q4 + X.Q1.\overline{Q4} + \bar{X}.\overline{Q2}.Q3.Q4$$

0	0	1	0
1	1	0	1
X	X	X	X
0	0	X	X
0	0	0	0
0	1	1	1
X	X	X	X
1	0	X	X

$$D3 = \bar{X}.Q3.\overline{Q4} + X.Q3.Q4 + X.Q1.\overline{Q4} + \bar{X}\overline{Q1}.\overline{Q3}.Q4 + X.Q2.\overline{Q3}.\overline{Q4}$$

0	1	0	1
0	1	0	1
X	X	X	X
0	0	X	X
0	0	1	0
1	0	1	0
X	X	X	X
1	0	X	X

$$D4 = \overline{Q4}$$

1	0	0	1
1	0	0	1
X	X	X	X
1	0	X	X
1	0	0	1
1	0	0	1
X	X	X	X
1	0	X	X