1. Preparation tasks

D	Qn	Q(n+1)	Comments
0	0	0	No change
0	1	0	Change
1	1	1	No change
1	0	1	Change

J	K	Qn	Q(n+1)	Comments
0	0	0	0	No change
0	0	1	1	No change
0	1	0	0	Reset
0	1	1	0	Reset
1	0	0	1	Set
1	0	1	1	Set
1	1	0	1	Toggle

J	K	Qn	Q(n+1)	Comments
1	1	1	0	Toggle

T	Qn	Q(n+1)	Comments
0	0	0	No change
0	1	1	No change
1	0	1	Toggle
1	1	0	Toggle

2. D latch

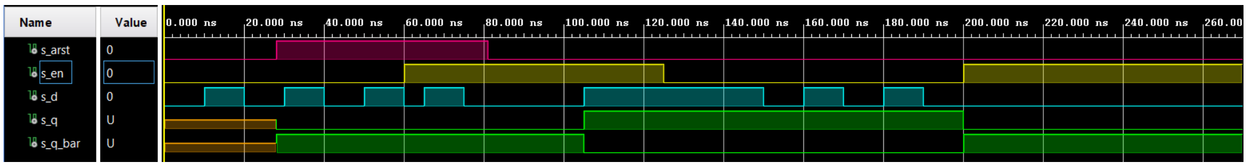
VHDL code listing of the process (`p_d_latch`)

```
p_d_latch : process(d, arst, en)
    begin

        if (arst = '1') then
            q     <= '0';
            q_bar <= '1';

       elsif (en = '1') then
            q     <= d;
            q_bar <= not d;

       end if;
end process p_d_latch;
```

VHDL reset and stimulus processes (`tb_d_latch`)

```
--------------------------------------------------------------------
    -- Reset generation process
--------------------------------------------------------------------
    p_reset_gen : process
    begin
        s_arst <= '0';
        wait for 38 ns;

        -- Reset activated
        s_arst <= '1';
        wait for 53 ns;

        -- Reset activated
        s_arst <= '0';
        wait for 300 ns;
        s_arst <= '1';

        wait;
    end process p_reset_gen;

    --------------------------------------------------------------------
    -- Data generation process
    --------------------------------------------------------------------
    p_stimulus : process
    begin
        report "Stimulus process started" severity note;
        s_en    <= '0';
        s_d     <= '0';

        wait for 10 ns;
        s_d <= '1';
        wait for 10 ns;
        s_d <= '0';
        wait for 10 ns;
        s_d <= '1';
        wait for 10 ns;
        s_d <= '0';
        wait for 10 ns;
        s_d <= '1';
        wait for 10 ns;
        s_d <= '0';

        s_en <= '1'; wait for 5 ns;
        assert(s_q = '0' and s_q_bar = '1')
        report "expected: s_q 0, q_bar 1" severity error;

        s_d <= '1';
        wait for 10 ns;
        s_d <= '0';
        wait for 10 ns;
        s_d <= '0';
        wait for 10 ns;
        s_d <= '0';
        wait for 10 ns;
        s_d <= '1';
        wait for 10 ns;
        s_d <= '1';
        wait for 10 ns;
        s_d <= '1';

        s_en <= '0'; wait for 5 ns;
        assert(s_q = '1' and s_q_bar = '0')
        report "expected: asrt 1" severity error;

        wait for 10 ns;
        s_d <= '1';
        wait for 10 ns;
        s_d <= '0';
        wait for 10 ns;
        s_d <= '1';
        wait for 10 ns;
        s_d <= '0';
        wait for 10 ns;
        s_d <= '1';
        wait for 10 ns;
        s_d <= '0';
        wait for 10 ns;
        s_d <= '0';

        s_en <= '1'; wait for 5 ns;
        assert(s_q = '0' and s_q_bar = '1')
        report "expected: asrt 1" severity error;

        report "Stimulus process finished" severity note;
      wait;
    end process p_stimulus;
```

Screenshot with simulated time waveforms



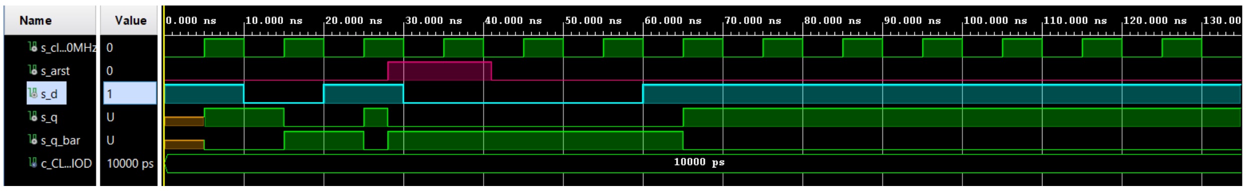
3. Flip-flops

VHDL code listing of the processes (`p_d_ff_rst`)

```
 p_d_latch : process(clk, arst)
    begin

        if (arst = '1') then
            q     <= '0';
            q_bar <= '1';

       elsif rising_edge(clk) then
            q     <= d;
            q_bar <= not d;

       end if;
end process p_d_latch;
```

Listing of VHDL clock, reset and stimulus processes from the testbench

```
    p_clk_gen : process
    begin
        while now < 750 ns loop
            s_clk_100MHz <= '0';
            wait for c_CLK_100MHZ_PERIOD / 2;
            s_clk_100MHz <= '1';
            wait for c_CLK_100MHZ_PERIOD / 2;
        end loop;
        wait;
    end process p_clk_gen;
    --------------------------------------------------------------------
    -- Reset
    --------------------------------------------------------------------
    p_reset_gen : process
    begin
        s_arst <= '0';
        wait for 28 ns;
        s_arst <= '1';
        wait for 13 ns;
        s_arst <= '0';
        wait;
    end process p_reset_gen;

    --------------------------------------------------------------------
    -- Stimulus
    --------------------------------------------------------------------
    p_stimulus : process
    begin
        report "Stimulus process started" severity note;

        s_d <= '1';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        s_d <= '0';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        s_d <= '1';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        s_d <= '0';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        wait for 20ns;
        s_d <= '1';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        report "Stimulus process finished" severity note;

        wait;
    end process p_stimulus;
```



VHDL code listing of the processes (`p_d_ff_rst`)

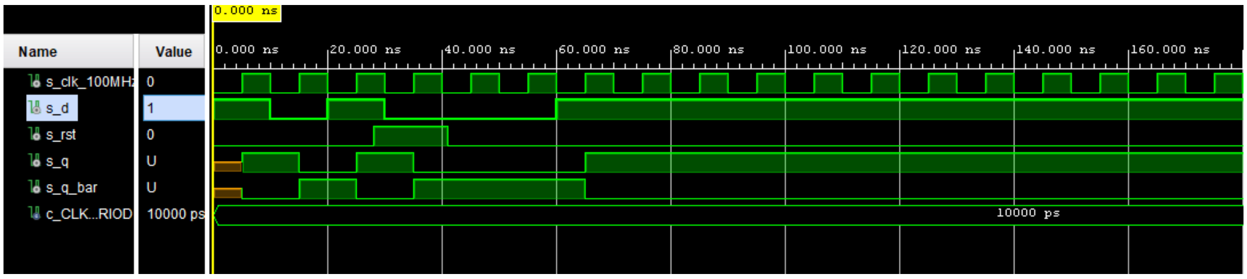
```
    p_d_ff_rst : process (clk)
    begin
        if rising_edge(clk) then
            if (rst = '1') then
                q       <= '0';
                q_bar   <= '1';
            else
                q       <= d;
                q_bar   <= not d;
            end if;
        end if;
    end process p_d_ff_rst;
```

Listing of VHDL clock, reset and stimulus processes from the testbench

```
    --------------------------------------------------------------------
    -- Clock generation process
    --------------------------------------------------------------------
    p_clk_gen : process
    begin
        while now < 750 ns loop
            s_clk_100MHz <= '0';
            wait for c_CLK_100MHZ_PERIOD / 2;
            s_clk_100MHz <= '1';
            wait for c_CLK_100MHZ_PERIOD / 2;
        end loop;
        wait;
    end process p_clk_gen;
    --------------------------------------------------------------------
    -- Reset
    --------------------------------------------------------------------
    p_reset_gen : process
    begin
```

```
        s_rst <= '0';
        wait for 28 ns;
        s_rst <= '1';
        wait for 13 ns;
        s_rst <= '0';
        wait;
    end process p_reset_gen;
    --------------------------------------------------------------------
    -- Stimulus
    --------------------------------------------------------------------
    p_stimulus : process
    begin
        report "Stimulus process started" severity note;

        s_d <= '1';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        s_d <= '0';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        s_d <= '1';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        s_d <= '0';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        wait for 20ns;
        s_d <= '1';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        report "Stimulus process finished" severity note;

        wait;
    end process p_stimulus;
```

Screenshot with simulated time waveforms



VHDL code listing of the processes (`p_jk_ff_rst`)

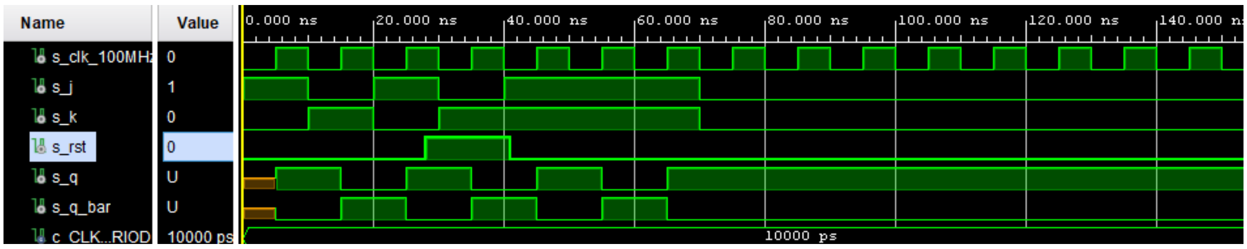
```
architecture Behavioral of jk_ff_rst is
    signal  s_q     : STD_LOGIC;
    signal  s_q_bar : STD_LOGIC;
begin
    p_jk_ff_rst : process (clk)
    begin
        if rising_edge(clk) then
            if (rst = '1') then
              s_q     <= '0';
              s_q_bar <= '1';
            else
            if (j = '0' and k = '0') then
                s_q     <= s_q;
                s_q_bar <= s_q_bar;
            elsif (j = '0' and k = '1') then
                s_q     <= '0';
                s_q_bar <= '1';
            elsif (j = '1' and k = '0') then
                s_q     <= '1';
                s_q_bar <= '0';
            else
                s_q     <= not s_q;
                s_q_bar <= not s_q_bar;
            end if;
          end if;
        end if;
    end process p_jk_ff_rst;

    q       <=  s_q;
    q_bar   <=  s_q_bar;
end Behavioral;
```

Listing of VHDL clock, reset and stimulus processes from the testbench

```
   --------------------------------------------------------------------
    -- Closk
    --------------------------------------------------------------------
    p_clk_gen : process
    begin
```

```
        while now < 750 ns loop
            s_clk_100MHz <= '0';
            wait for c_CLK_100MHZ_PERIOD / 2;
            s_clk_100MHz <= '1';
            wait for c_CLK_100MHZ_PERIOD / 2;
        end loop;
        wait;
    end process p_clk_gen;

    --------------------------------------------------------------------
    -- Reset
    --------------------------------------------------------------------
    p_reset_gen : process
    begin
        s_rst <= '0';
        wait for 28 ns;
        s_rst <= '1';
        wait for 13 ns;
        s_rst <= '0';
        wait;
    end process p_reset_gen;

    --------------------------------------------------------------------
    -- Stimulus
    --------------------------------------------------------------------
    p_stimulus : process
    begin
        report "Stimulus process started" severity note;

        s_j <= '1';
        s_k <= '0';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        s_j <= '0';
        s_k <= '1';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        s_j <= '1';
        s_k <= '0';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        s_j <= '0';
        s_k <= '1';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        s_j <= '1';
        s_k <= '1';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        s_j <= '1';
        s_k <= '1';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        s_j <= '1';
        s_k <= '1';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        s_j <= '0';
        s_k <= '0';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        report "Stimulus process finished" severity note;

        wait;
    end process p_stimulus;
```

Screenshot with simulated time waveforms



VHDL code listing of the processes (`p_t_ff_rst`)

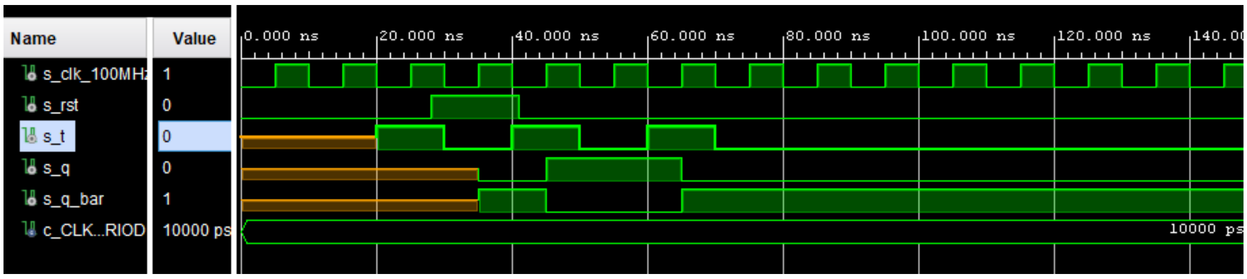
```
architecture Behavioral of t_ff_rst is
    signal  s_q     :   STD_LOGIC;
    signal  s_q_bar :   STD_LOGIC;
begin

    t_ff_rst : process(clk)
    begin
        if rising_edge(clk) then
            if (rst = '1') then
                s_q     <=  '0';
                s_q_bar <=  '1';
             else
                if (t = '0') then
```

```vhdl
                    s_q     <=  s_q;
                    s_q_bar <=  s_q_bar;
                else
                    s_q     <=  not s_q;
                    s_q_bar <=  not s_q_bar;
                end if;
            end if;
        end if;
    end process t_ff_rst;

    q       <=  s_q;
    q_bar   <=  s_q_bar;
end Behavioral;
```

Listing of VHDL clock, reset and stimulus processes from the testbench

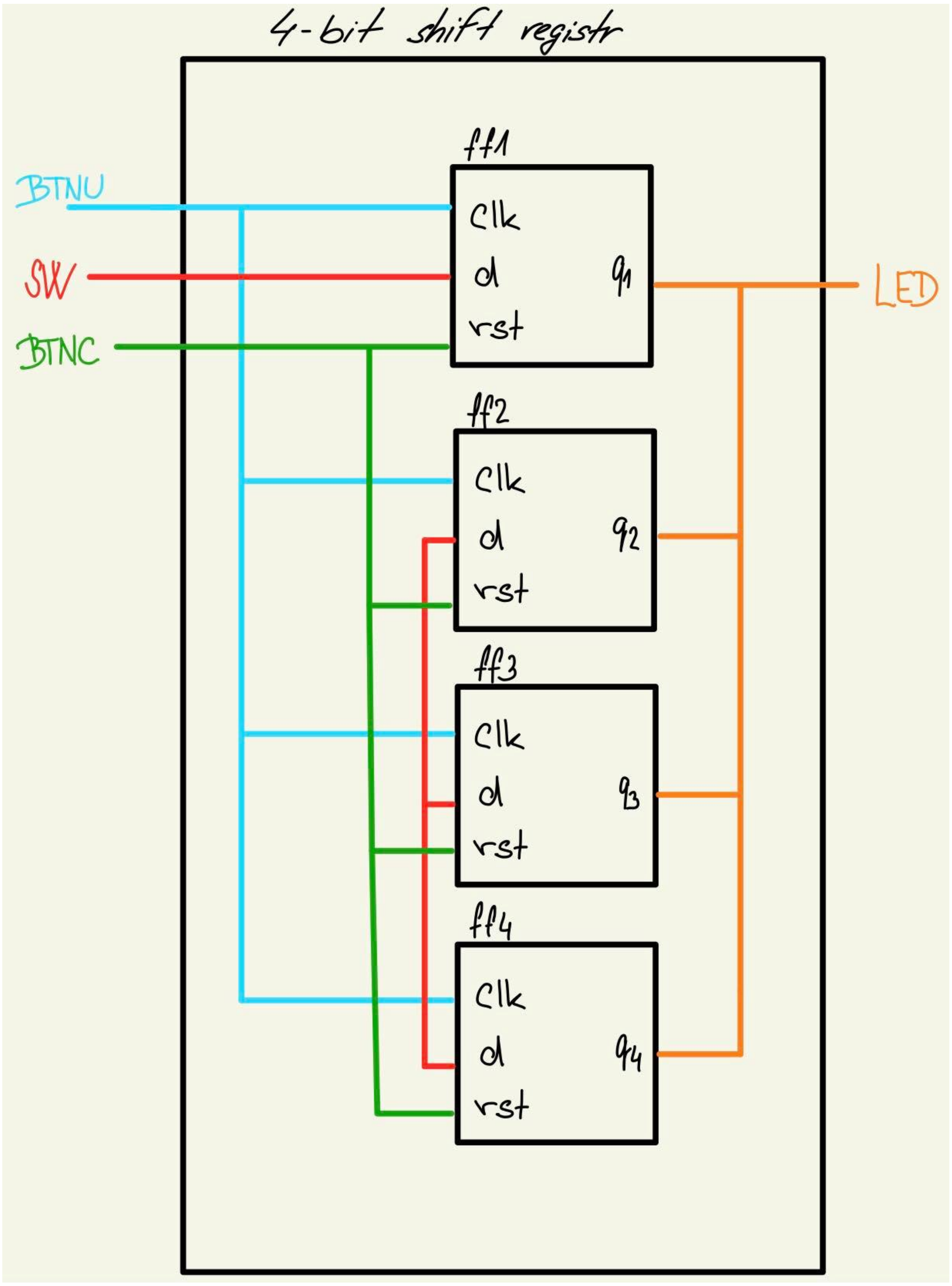
```vhdl
    --------------------------------------------------------------------
    -- Closk
    --------------------------------------------------------------------
    p_clk_gen : process
    begin
        while now < 750 ns loop
            s_clk_100MHz <= '0';
            wait for c_CLK_100MHZ_PERIOD / 2;
            s_clk_100MHz <= '1';
            wait for c_CLK_100MHZ_PERIOD / 2;
        end loop;
        wait;
    end process p_clk_gen;
    --------------------------------------------------------------------
    -- Reset
    --------------------------------------------------------------------
    p_reset_gen : process
    begin
        s_rst <= '0';
        wait for 28 ns;
        s_rst <= '1';
        wait for 13 ns;
        s_rst <= '0';
        wait;
    end process p_reset_gen;
    --------------------------------------------------------------------
    -- Stimulus
    --------------------------------------------------------------------
    p_stimulus : process
    begin
        report "Stimulus process started" severity note;

        wait for 20ns;
        s_t <= '1';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        s_t <= '0';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        s_t <= '1';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        s_t <= '0';
        wait for 10ns;
        assert (s_q = '0' and s_q_bar = '1')
        report "Error" severity note;

        s_t <= '1';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        s_t <= '0';
        wait for 10ns;
        assert (s_q = '1' and s_q_bar = '0')
        report "Error" severity note;

        report "Stimulus process finished" severity note;

        wait;
    end process p_stimulus;
end Behavioral;
```

Screenshot with simulated time waveforms



4. Shift register

Image of the shift register schematic


4-bit shift registr
ff1
ff2
ff3
ff4
BTNU
SW
BTNC
LED
clk
d
rst
q1
q2
q3
q4