Assignment 7

Link to this Assignment
Link to top of repository

1. Preparation tasks

1.1 D-type flip-flop

$q^D_{n+1} = D$

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

1.2 JK-type flip-flop

$q^{JK}_{n+1} = J \cdot \overline{Q(n)} + \overline{K} \cdot Q(n)$

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	Inverse
1	1	1	0	Inverse

1.3 T-type flip-flop

$q^T_{n+1} = T \cdot \overline{Q(n)} + \overline{T} \cdot Q(n)$

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

2. D latch

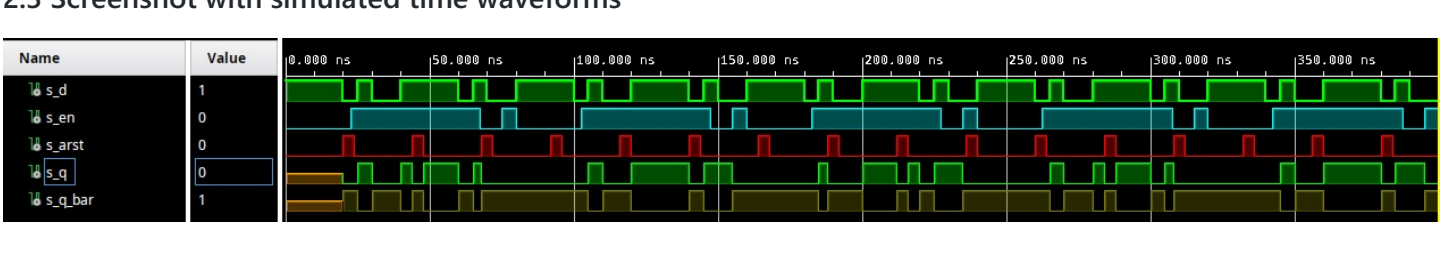
2.1 Listing of VHDL code of the process `p_d_latch`

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

2.2 Listing of VHDL reset and stimulus processes from the testbench `tb_d_latch.vhd`

```
--------------------------------------------------------------------
    -- Enable generation process
--------------------------------------------------------------------
 p_en_gen : process
    begin
        s_en <= '0';
        wait for 22.5ns;
        s_en <= '1';
        wait for 45ns;
        s_en <= '0';
        wait for 7.5ns;
        s_en <= '1';
        wait for 5ns;
    end process p_en_gen;
-------------------------------------------------------
    -- Reset generation process
--------------------------------------------------------------------
tb_arst : process
    begin
        s_arst <= '0';
        wait for 20 ns;
        s_arst <= '1';
        wait for 4ns;
    end process tb_arst;
--------------------------------------------------------------------
    -- Stimulus
--------------------------------------------------------------------
tb_stimulus : process
    begin
        s_d <= '1';
        wait for 20ns;
        s_d <= '0';
        wait for 5ns;
        s_d <= '1';
        wait for 5ns;
        s_d <= '0';
        wait for 10ns;
    end process tb_stimulus;

tb_check : process
    begin
    wait until ((s_d = '0') and (s_en = '0') and (s_arst = '1'));
    wait for 0.4ns;
    assert (s_q = '0') and (s_q_bar = '1')
    report "Reset fault (D LOW)" severity error;

    wait until ((s_d = '1') and (s_en = '0') and (s_arst = '1'));
    wait for 0.4ns;
    assert (s_q = '0') and (s_q_bar = '1')
    report "Reset fault (D HIGH)" severity error;

    wait until (rising_edge(s_d) and (s_en = '0') and (s_arst = '0') and (s_q = '0'));
    wait for 0.4ns;
    assert (s_q = '0') and (s_q_bar = '1')
    report "Hold fault (Hold when EN is LOW)" severity error;

    wait until (falling_edge(s_d) and (s_en = '0') and (s_arst = '0') and (s_q = '1'));
    wait for 0.4ns;
    assert (s_q = '1') and (s_q_bar = '0')
    report "Hold fault (Hold when EN is LOW)" severity error;

    wait until ((s_d = '1') and (s_en = '1') and (s_arst = '0'));
    wait for 0.4ns;
    assert (s_q = '1') and (s_q_bar = '0') --transparency check
    report "Data fault" severity error;

    end process tb_check;
```

2.3 Screenshot with simulated time waveforms



3. Flip-flops

3.1 Listing of VHDL code of the processes

3.1.1 Listing of VHDL code of the process `p_d_ff_arst`

```
p_d_ff_arst: 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_ff_arst;
```

3.1.2 Listing of VHDL code of the process `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;
```

3.1.3 Listing of VHDL code of the process `p_jk_ff_rst`

```
p_jk_ff_rst: process(clk)
    variable q_local : std_logic;
    begin
        if rising_edge(clk) then
            if (rst = '1') then
                q_local := '0';
            elsif ((j = '1')and(k = '1')) then
                q_local := not q_local;
            elsif (j = '1') then
                q_local := '1';
            elsif (k = '1') then
                q_local := '0';
            end if;
            q <= q_local;
            q_bar <= not q_local;
        end if;
    end process p_jk_ff_rst;
```

3.1.4 Listing of VHDL code of the process `p_t_ff_rst`

```
p_t_ff_rst: process(clk)
   variable q_local : std_logic;
   begin
        if rising_edge(clk) then
          if (rst = '1') then
            q_local := '0';
          elsif (t = '1') then
            q_local := not q_local;
          end if;
          q <= q_local;
          q_bar <= not q_local;
        end if;
    end process p_t_ff_rst;
```

3.2 Listing of Testbench processes `clock` , `reset` , `stimulus`

3.2.1 Listing of testbench for `p_d_ff_arst`

```
--------------------------------------------------------------------
    -- Clock generation process
--------------------------------------------------------------------
    p_clk_gen: process
    begin
        s_clk <= '0';
        wait for 10ns;
        s_clk <= '1';
        wait for 10ns;
    end process p_clk_gen;
--------------------------------------------------------------------
    -- Reset generation process
--------------------------------------------------------------------
    p_reset: process
    begin
        s_arst <= '0';
        wait for 30 ns;
        s_arst <= '1';
        wait for 4ns;
    end process p_reset;
--------------------------------------------------------------------
    -- Stimulus
--------------------------------------------------------------------
    p_stimulus: process
    begin
        s_d <= '0';
        wait for 25ns;
        s_d <= '1';
        wait for 35ns;
        s_d <= '0';
        wait for 15ns;
        s_d <= '1';
        wait for 55ns;
        s_d <= '0';
        wait for 75ns;
        s_d <= '1';
        wait for 10ns;
    end process p_stimulus;

    p_check : process
        begin
        wait until ((s_d = '0') and (s_clk = '0') and (s_arst = '1'));
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Reset fault (D LOW)" severity error;

        wait until ((s_d = '1') and (s_clk = '0') and (s_arst = '1'));
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Reset fault (D HIGH)" severity error;

        wait until (rising_edge(s_d) and not(rising_edge(s_clk)) and (s_arst = '0') and (s_q = '0'));
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Hold fault (Hold when CLK is not rising)" severity error;

        wait until (falling_edge(s_d) and not(rising_edge(s_clk)) and (s_arst = '0') and (s_q = '1'));
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0')
        report "Hold fault (Hold when CLK is not rising)" severity error;

        wait until ((s_d = '1') and rising_edge(s_clk) and (s_arst = '0'));
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0') --transparency check
        report "Data fault" severity error;
        report "Test complete" severity note;

        end process p_check;
```

3.2.2 Listing of testbench for `p_d_ff_rst`

```
--------------------------------------------------------------------
    -- Clock generation process
--------------------------------------------------------------------
    p_clk_gen: process
    begin
        s_clk <= '0';
        wait for 10ns;
        s_clk <= '1';
        wait for 10ns;
    end process p_clk_gen;
--------------------------------------------------------------------
    -- Reset generation process
--------------------------------------------------------------------
    p_reset: process
    begin
        s_rst <= '0';
        wait for 30 ns;
        s_rst <= '1';
        wait for 4ns;
    end process p_reset;
--------------------------------------------------------------------
    -- Stimulus
--------------------------------------------------------------------
    p_stimulus: process
    begin
        s_d <= '0';
        wait for 25ns;
        s_d <= '1';
        wait for 35ns;
        s_d <= '0';
        wait for 15ns;
        s_d <= '1';
        wait for 55ns;
        s_d <= '0';
        wait for 75ns;
        s_d <= '1';
        wait for 10ns;
    end process p_stimulus;

    p_check : process
        begin
        wait until ((s_d = '0') and rising_edge(s_clk) and (s_rst = '1'));
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Reset fault (D LOW)" severity error;

        wait until ((s_d = '1') and rising_edge(s_clk) and (s_rst = '1'));
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Reset fault (D HIGH)" severity error;

        wait until (rising_edge(s_d) and not(rising_edge(s_clk)) and (s_rst = '0') and (s_q = '0'));
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Hold fault (Hold when CLK is not rising)" severity error;

        wait until (falling_edge(s_d) and not(rising_edge(s_clk)) and (s_rst = '0') and (s_q = '1'));
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0')
        report "Hold fault (Hold when CLK is not rising)" severity error;

        wait until ((s_d = '1') and rising_edge(s_clk) and (s_rst = '0'));
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0') --transparency check
        report "Data fault" severity error;
        report "Test complete" severity note;

        end process p_check;
```

3.2.3 Listing of testbench for `p_jk_ff_rst`

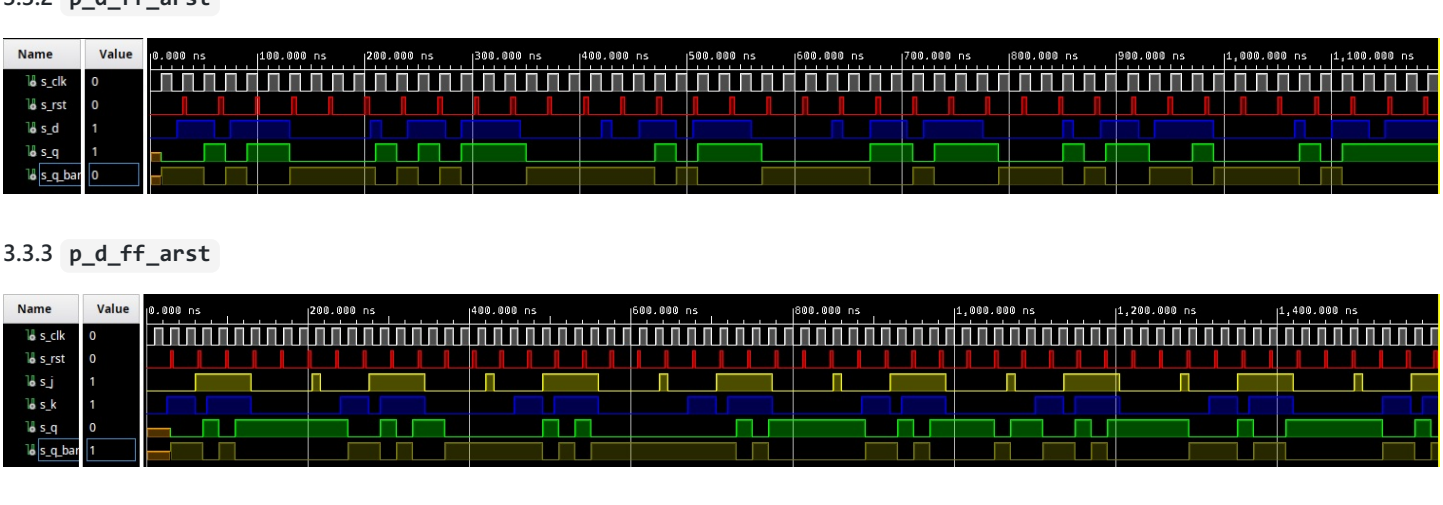
```
--------------------------------------------------------------------
    -- Clock generation process
--------------------------------------------------------------------
    p_clk_gen: process
    begin
        s_clk <= '0';
        wait for 10ns;
        s_clk <= '1';
        wait for 10ns;
    end process p_clk_gen;
--------------------------------------------------------------------
    -- Reset generation process
--------------------------------------------------------------------
    p_reset: process
    begin
        s_rst <= '0';
        wait for 30 ns;
        s_rst <= '1';
        wait for 4ns;
    end process p_reset;
--------------------------------------------------------------------
    -- Stimulus
--------------------------------------------------------------------
    p_stimulus: process
    begin
        s_j <= '0';
        s_k <= '0';
        wait for 25ns;
        s_j <= '0';
        s_k <= '1';
        wait for 35ns;
        s_j <= '1';
        s_k <= '0';
        wait for 15ns;
        s_j <= '1';
        s_k <= '1';
        wait for 55ns;
        s_j <= '0';
        s_k <= '0';
        wait for 75ns;
        s_j <= '1';
        s_k <= '0';
        wait for 10ns;
    end process p_stimulus;

    p_check : process
        begin
        wait until (rising_edge(s_clk) and s_rst = '1' and s_q = '0');
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Reset fault (Q LOW)" severity error;

        wait until (rising_edge(s_clk) and s_rst = '1' and s_q = '1');
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Reset fault (Q HIGH)" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_j = '1' and s_k = '0');
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0')
        report "J fault" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_j = '0' and s_k = '1');
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "K fault" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_j = '0' and s_k = '0' and s_q = '1');
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0')
        report "Hold fault (Q HIGH)" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_j = '0' and s_k = '0' and s_q = '0');
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Hold fault (Q LOW)" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_j = '1' and s_k = '1' and s_q = '1');
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Hold fault (Q HIGH)" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_j = '1' and s_k = '1' and s_q = '0');
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0')
        report "Hold fault (Q LOW)" severity error;

        wait until (not(rising_edge(s_clk)) and (rising_edge(s_rst) or falling_edge(s_rst)) and (rising_edge(s_j) or falling_e
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0')
        report "clock/sync fault (async change when Q HIGH)" severity error;

        wait until (not(rising_edge(s_clk)) and (rising_edge(s_rst) or falling_edge(s_rst)) and (rising_edge(s_j) or falling_e
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "clock/sync fault (async change when Q LOW)" severity error;

        report "Test complete" severity note;

        end process p_check;
```

3.2.4 Listing of testbench for `p_t_ff_rst`

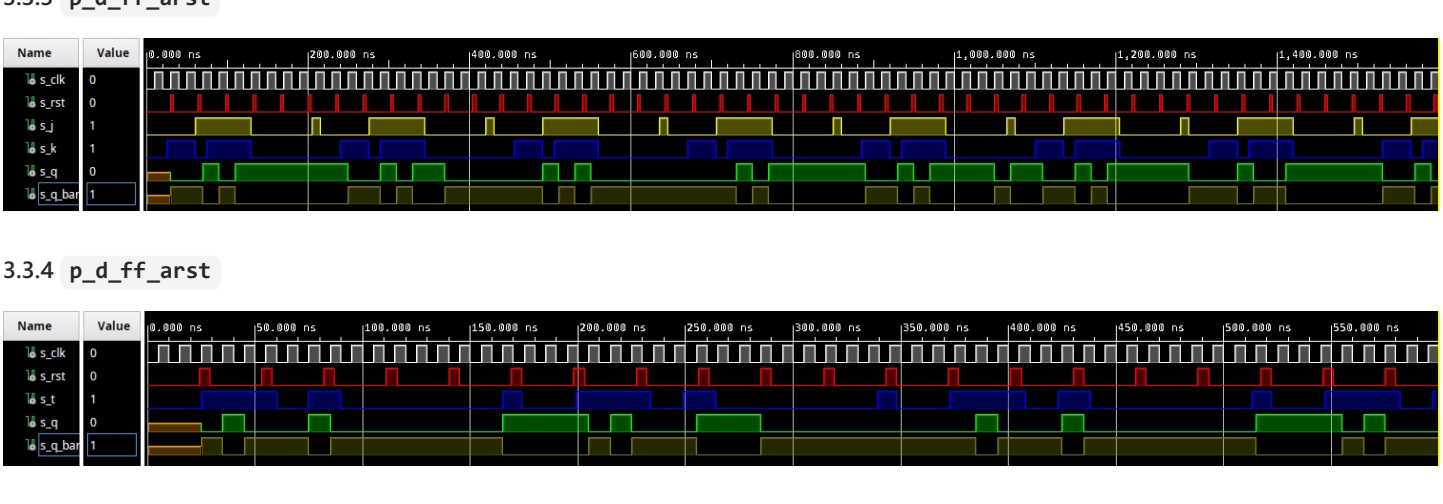
```
--------------------------------------------------------------------
    -- Clock generation process
--------------------------------------------------------------------
    p_clk_gen: process
    begin
        s_clk <= '0';
        wait for 5ns;
        s_clk <= '1';
        wait for 5ns;
    end process p_clk_gen;
--------------------------------------------------------------------
    -- Reset generation process
--------------------------------------------------------------------
    p_reset: process
    begin
        s_rst <= '0';
        wait for 24 ns;
        s_rst <= '1';
        wait for 5ns;
    end process p_reset;
--------------------------------------------------------------------
    -- Stimulus
--------------------------------------------------------------------
    p_stimulus: process
    begin
        s_t <= '0';
        wait for 25ns;
        s_t <= '1';
        wait for 35ns;
        s_t <= '0';
        wait for 15ns;
        s_t <= '1';
        wait for 15ns;
        s_t <= '0';
        wait for 75ns;
        s_t <= '1';
        wait for 9ns;
    end process p_stimulus;

    p_check : process
        begin
        wait until (rising_edge(s_clk) and s_rst = '1');
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Reset fault" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_t = '0' and s_q = '0');
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Hold fault (Q High)" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_t = '0' and s_q = '1');
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0')
        report "Hold fault (Q Low)" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_t = '1' and s_q = '0');
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0')
        report "Invert fault (Q Low)" severity error;

        wait until (rising_edge(s_clk) and s_rst = '0' and s_t = '1' and s_q = '1');
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "Invert fault (Q High)" severity error;

        wait until (not(rising_edge(s_clk)) and (rising_edge(s_rst) or falling_edge(s_rst)) and (rising_edge(s_t) or falling_e
        wait for 0.4ns;
        assert (s_q = '0') and (s_q_bar = '1')
        report "clock/sync fault (async change when Q HIGH)" severity error;

        wait until (not(rising_edge(s_clk)) and (rising_edge(s_rst) or falling_edge(s_rst)) and (rising_edge(s_t) or falling_e
        wait for 0.4ns;
        assert (s_q = '1') and (s_q_bar = '0')
        report "clock/sync fault (async change when Q HIGH)" severity error;
        report "Test complete" severity note;

        end process p_check;
```

3.3 Screenshots with simulated time waveforms

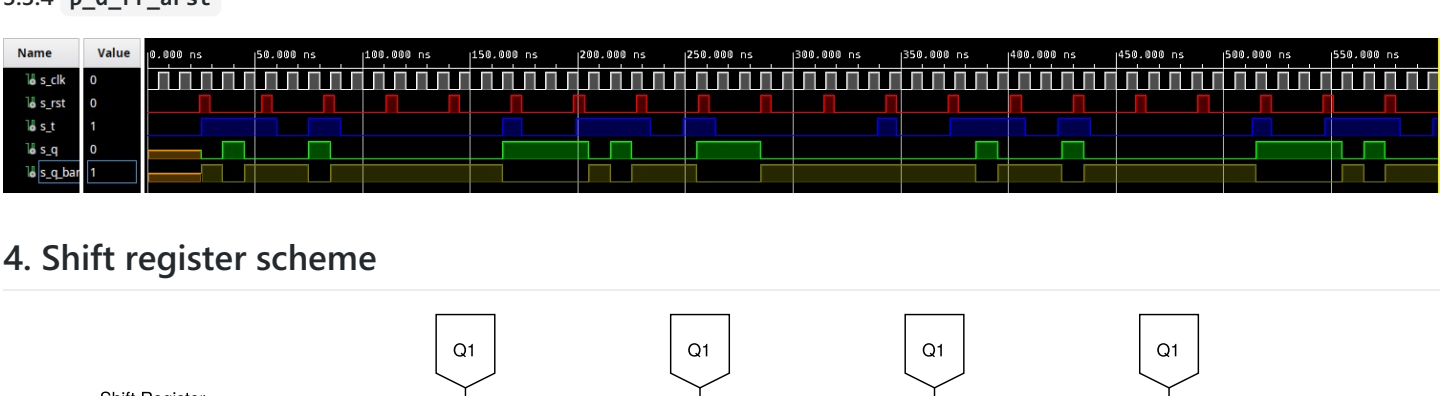
3.3.1 `p_d_ff_arst`



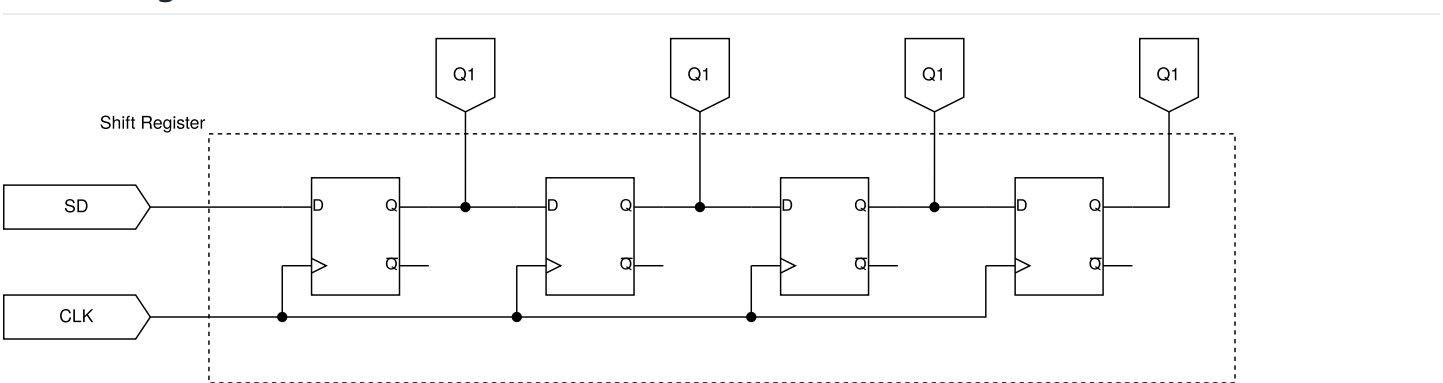
3.3.2 `p_d_ff_arst`



3.3.3 `p_d_ff_arst`



3.3.4 `p_d_ff_arst`



4. Shift register scheme