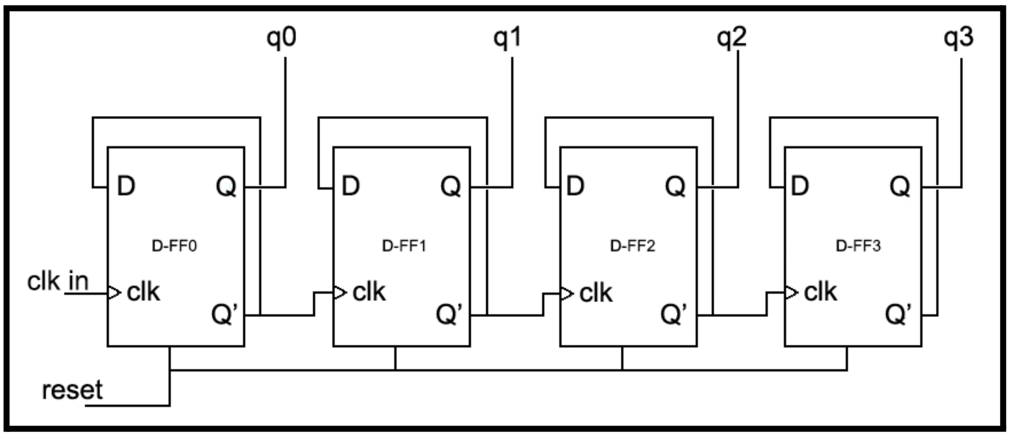
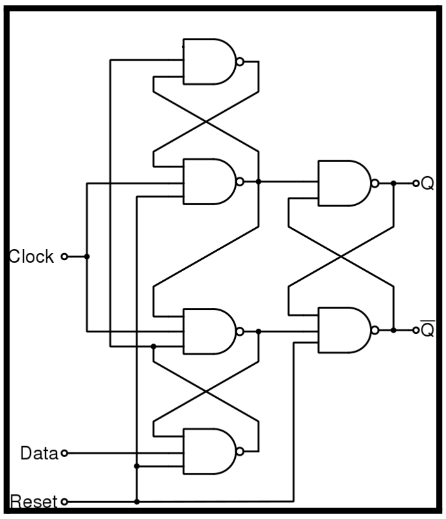
4 bit Counter

Asynchronous 4 bit Counter

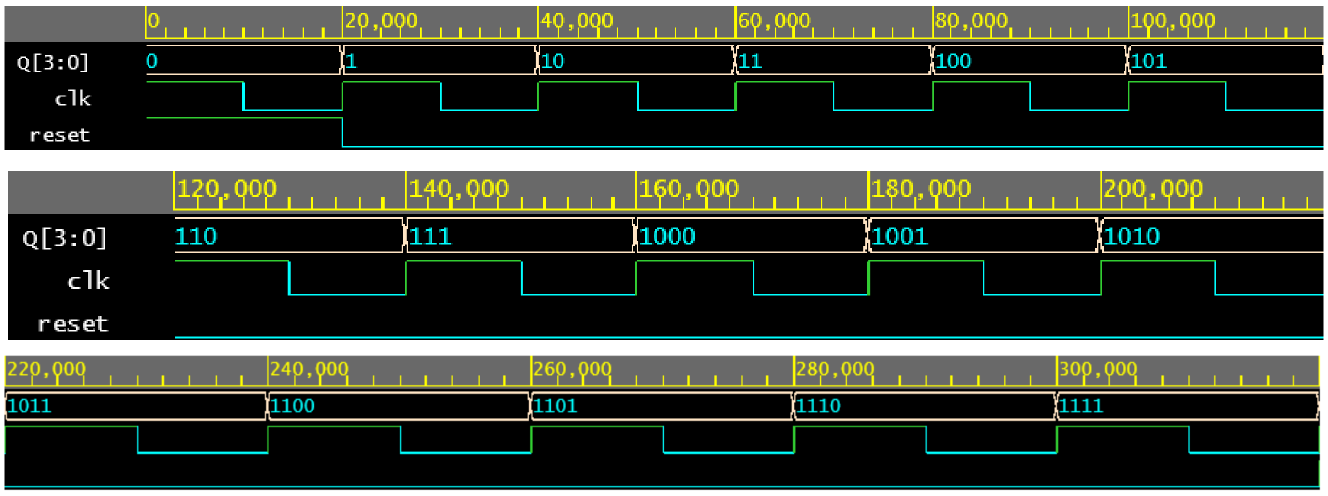


D flip-flop

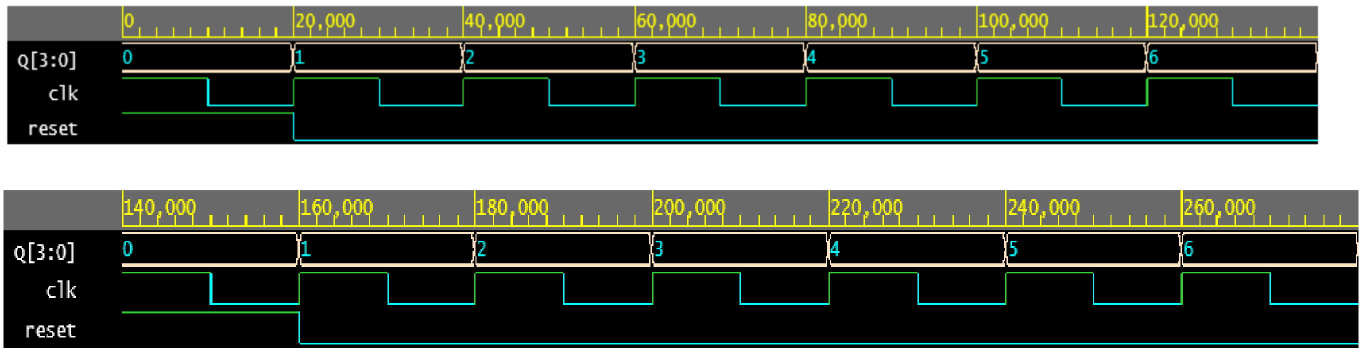


Implementing a 4-bit asynchronous counter requires the interconnection of four D Flip-Flops. The clock input is connected to the first Flip-Flop. The remaining Flip-Flops receive their clock signal input from the Q' output of the previous Flip-Flop. In the counter the Flip-Flops are connected in toggle mode, so when the clock input is connected to the first flip flop D-FF0, then its output after one clock pulse will become 20ns. The q outputs of each D-FF represent the metre reading.
The simulation results are as follows

Count from 0 to 15 - Run Time: 320ns



Count from 0 to 6 with repetition - Run Time: 270ns



Behavioural/Algorithmic model:

```
library IEEE;
use IEEE.std_logic_1164.all;
-- Creating a Positively Triggered D Flip - Flop
entity D_FlipFlop is
   port ( FFclk,FFreset,FFD: in bit; FFq: out bit);
end D_FlipFlop;

architecture Algorithmic of D_FlipFlop is
begin
            process(FFd,FFclk, FFreset)
            begin
            -- Create a rollback feature
            if (FFreset = '1') then
                        FFq <= '0';
            -- Value change event
            elsif (FFclk = '1')  and FFclk'event then
                        FFq <= FFd;
            end if;
            end process;
end Algorithmic;

-- Creating a 4-bit counter
entity counter_4bit is
            Port ( clk,reset : in  bit;
            Q: out  bit_vector(3 downto 0));
end counter_4bit;

architecture Algorithmic of counter_4bit is
  component D_FlipFlop
            port ( FFclk,FFreset,FFD: in bit; FFq: out bit);
  end component;

signal q0, q1, q2,q3: bit;
begin
-- Creation of the 4 Flip-Flops and the corresponding bit assignment between them.
-- eg FF0 output FFq to be assigned to q0 and then q0 to be mapped as FF1's clk input
-- The output FFq of FF1 is assigned to q1 and then q1 is assigned as the clk input of FF2. ….
-- etc.
  FF0: D_FlipFlop
  port map(FFclk => clk, FFreset => reset, FFd => (not q0), FFq => q0);
  FF1: D_FlipFlop
  port map(FFclk =>(not q0), FFreset => reset, FFd => (not q1), FFq => q1);
  FF2: D_FlipFlop
  port map(FFclk => (not q1), FFreset => reset, FFd => (not q2), FFq => q2);
  FF3: D_FlipFlop
  port map(FFclk => (not q2),FFreset => reset, FFd => (not q3),FFq => q3);
 -- Output
      Q <= q3 & q2 & q1 & q0;
end Algorithmic;
```

Testbench

```
Testbench
library IEEE;
use IEEE.std_logic_1164.all;

entity test_counter_4bit is
end test_counter_4bit;

architecture testbench  of test_counter_4bit is
   component counter_4bit
            port(
            clk,reset : in  bit;
            Q: out  bit_vector(3 downto 0));
  end component;

 signal clk:bit;
 signal reset:bit;
 signal Q:bit_vector(3 downto 0);

begin
  test: counter_4bit port map (
  clk=>clk,
  reset=>reset,
  Q=>Q);

process
  begin
            -- This particular testbench counts up to 6
            -- Reset assignment. Depending on the number you wish to count
            -- need to add appropriate number clk<='1'? reset<='0';  and clk<='0'; reset <='0';

            clk<='1'; reset<='1';
            wait for 10 ns;
            clk<='0'; reset<='1';
            wait for 10 ns;
            --1
            clk<='1'; reset<='0';
            wait for 10 ns;
            clk<='0'; reset <='0';
            wait for 10 ns;
            --2
            clk<='1'; reset<='0';
            wait for 10 ns;
            clk<='0'; reset<='0';
            wait for 10 ns;
            --3
            clk<='1'; reset<='0';
            wait for 10 ns;
            clk<='0'; reset<='0';
            wait for 10 ns;
```

```
              --4
            clk<='1'; reset<='0';
            wait for 10 ns;
            clk<='0'; reset<='0';
            wait for 10 ns;
            --5
            clk<='1'; reset<='0';
            wait for 10 ns;
            clk<='0'; reset<='0';
            wait for 10 ns;
            --6
            clk<='1'; reset<='0';
            wait for 10 ns;
            clk<='0'; reset<='0';
            wait for 10 ns;

   end process;
end testbench ;
```