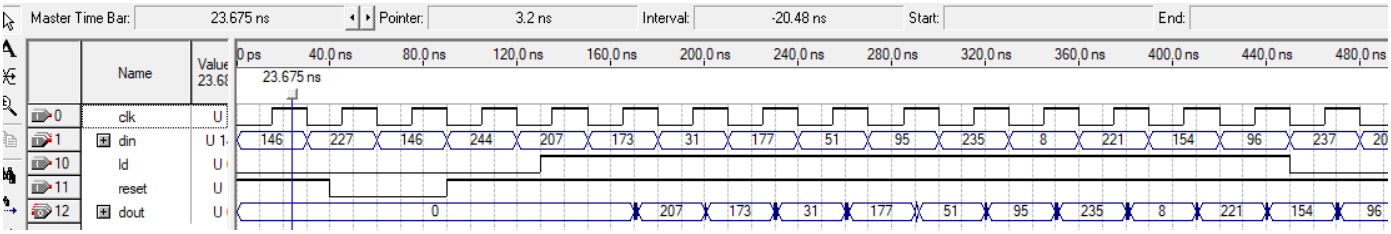
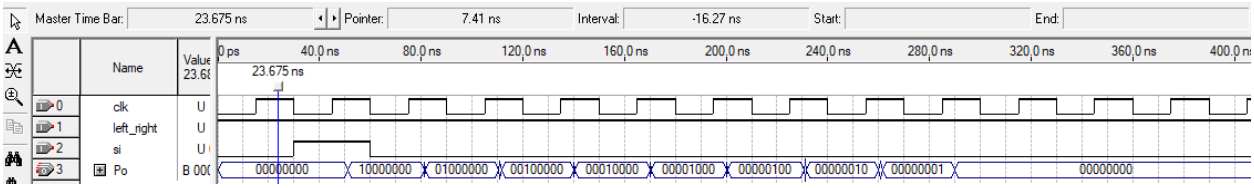
1. Registru pe 8 biți cu încărcare paralelă şi resetare asincronă

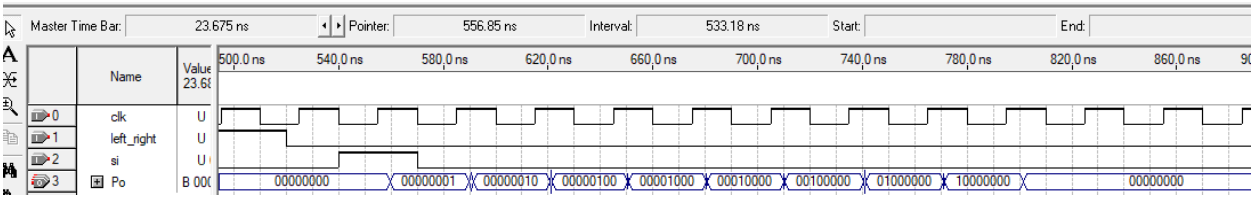
```
library ieee;
use ieee.std_logic_1164.all;
entity registru1 is
  port ( clk, reset, ld: in std_logic;
         din: in std_logic_vector(7 downto 0);
        dout: out std_logic_vector(7 downto 0));
end registru1;
architecture behavior of registru1 is
 signal n_state: std_logic_vector(7 downto 0);
 signal p_state : std_logic_vector(7 downto 0);
  begin
   process(clk, reset)
     begin
       if (reset = '0') then p_state <= (others => '0'); --resetare asincrona
        elsif (clk'event and clk = '1') then
          if (ld='1') then n_state <= din; --încarcare paralela
            else null;
         end if;
      p_state <= n_state;
    end if;
  end process;
dout <= p_state;
end behavior;
```



2. Registru bidirectional cu incarcare seriala

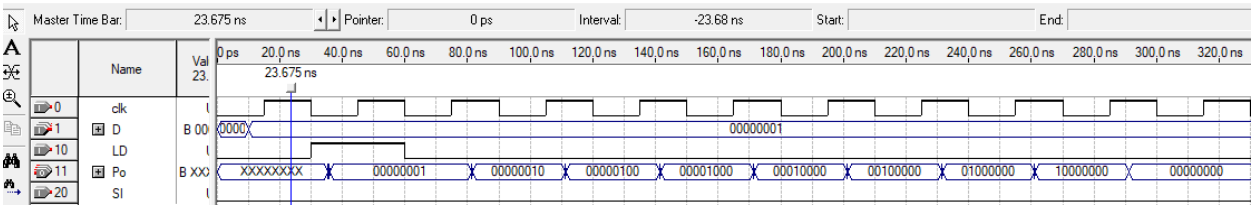
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity registru2 is
        port (clk, si, left_right: in std_logic;
        Po: out std_logic_vector (7 downto 0));
end registru2;
architecture beh of registru2 is
signal tmp: std_logic_vector (7 downto 0);
  begin
    process (clk)
       begin
          if (clk'event and clk='1') then
            if (left_right='0') then
               tmp<=tmp(6 downto 0) & si; --deplasare stanga si incarcare seriala
            else
              tmp<=si & tmp(7 downto 1); --deplasare dreapta si incarcare seriala
            end if;
         end if;
     end process;
   po<=tmp; --iesire paralela
end beh;
```

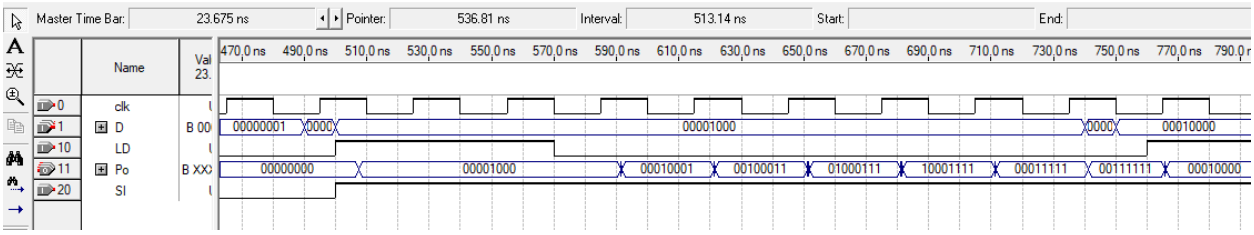


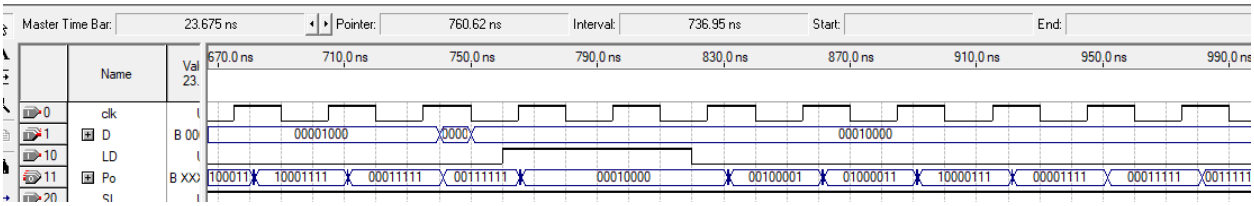


3.Registru deplasare stanga cu incarcare paralela, activ pe frontul pozitiv al semnalului de ceas

```
    library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity registru3 is
        port (clk, LD, SI : in std_logic;
        D: in std_logic_vector (7 downto 0);
        Po: out std_logic_vector (7 downto 0));
end registru3;
architecture beh of registru3 is
signal tmp: std_logic_vector (7 downto 0);
  begin
            process (CLK, LD, D)
           begin
              if (LD='1') then
                  tmp <= D; --încarcare paralela
               elsif (CLK'event and CLK='1') then
                    tmp <= tmp(6 downto 0) & SI;
          end if;
      end process;
    Po<=tmp; --iesire paralela
end beh;
```







Proiectați un registru de deplasare pe 3 biți, activ pe frontul negativ al semnalului de ceas, cu resetare sincronă, încărcare paralelă, care efectuează operația de împărțire la 2.