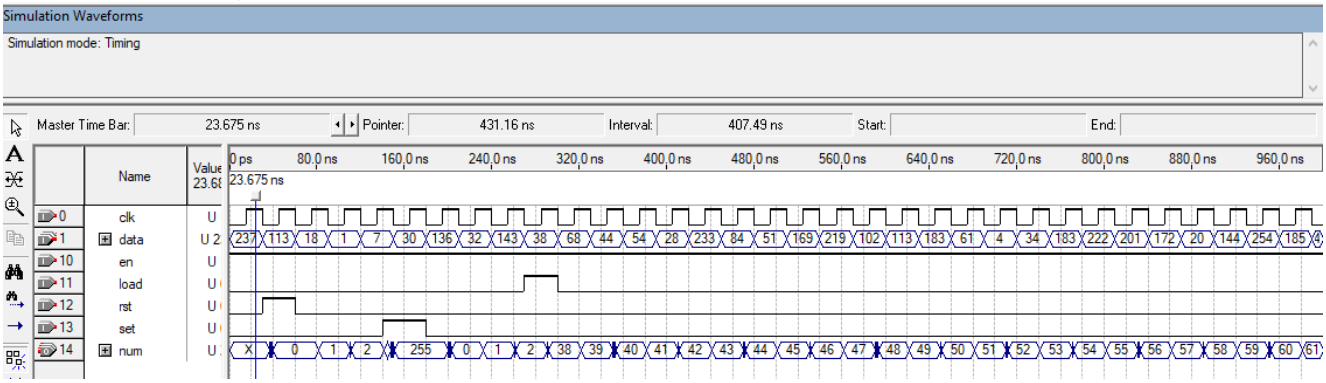
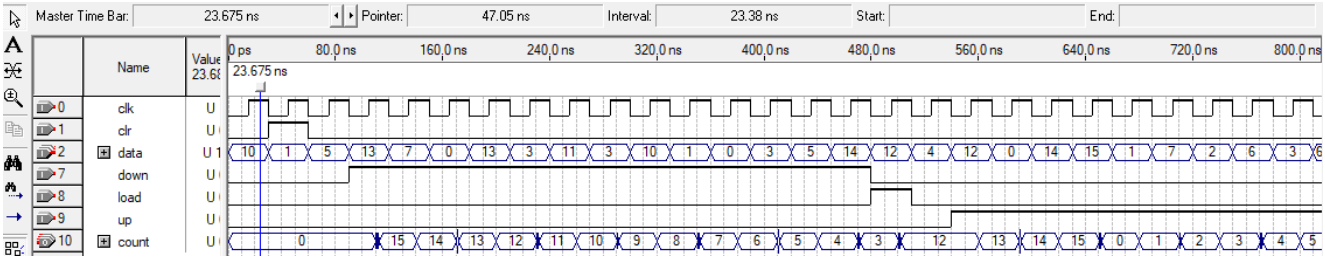
1. Numărător direct de 8 biţi cu semnale asincrone de resetare şi setare.

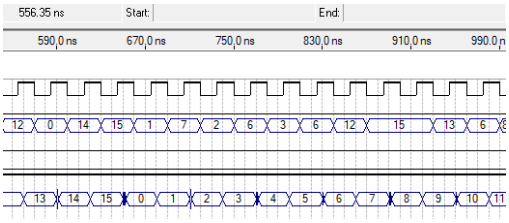
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity num1 is
  port (clk: in std_logic;
        rst, set: in std_logic;
        en, load: in std_logic;
        data: in std_logic_vector (7 downto 0);
         num: out std_logic_vector (7 downto 0));
end num1;
architecture arh_num1 of num1 is
   signal tmp: std_logic_vector (7 downto 0);
     begin
       cnt: process (rst, set, clk)
            begin
               if (rst = '1') then
                   tmp <= (others => '0');
                elsif (set = '1') then
                   tmp <= (others => '1');
                elsif (clk'event and clk = '1') then
                   if (load = '1') then
                       tmp <= data;
                elsif (en = '1') then
                       tmp <= tmp + 1;
                end if;
              end if;
        end process cnt;
          num <= tmp;
end arh_num1;
```



2.Numărător pe 4 biți reversibil cu încărcare sincronă

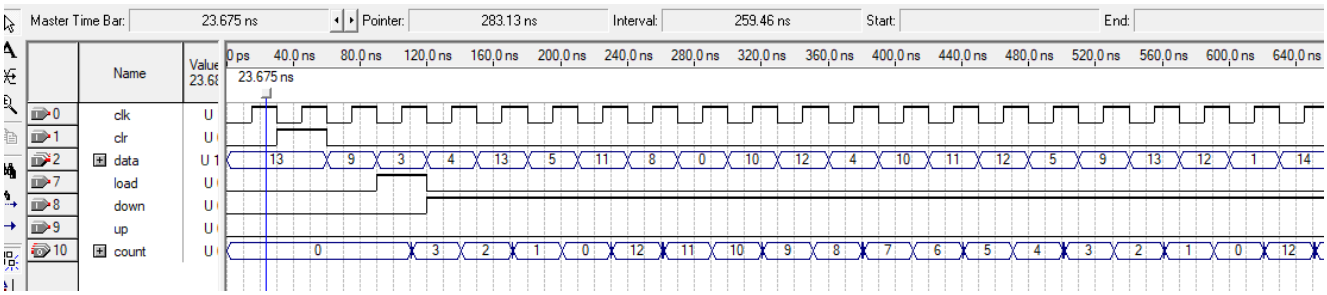
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity num2 is
port (clk, clr, load, up, down: in std_logic;
        data: in std_logic_vector(3 downto 0);
       count: out std_logic_vector(3 downto 0));
end num2;
architecture count4 of num2 is
signal cnt: std_logic_vector(3 downto 0);
  begin
    process (clr, clk)
       begin
         if clr='1' then cnt<="0000";
         elsif (clk'event and clk='1') then
               if load='1' then cnt<=data;
          elsif up='1' then
                    if cnt="1111" then cnt <="0000";
                   else cnt<=cnt+1;
                  end if;
              elsif down='1' then
                if cnt="0000" then cnt<="1111";
              else cnt<=cnt-1;
            end if;
           else
        cnt<=cnt;
end if;
end if;
count<=cnt;
end process;
end count4;
```

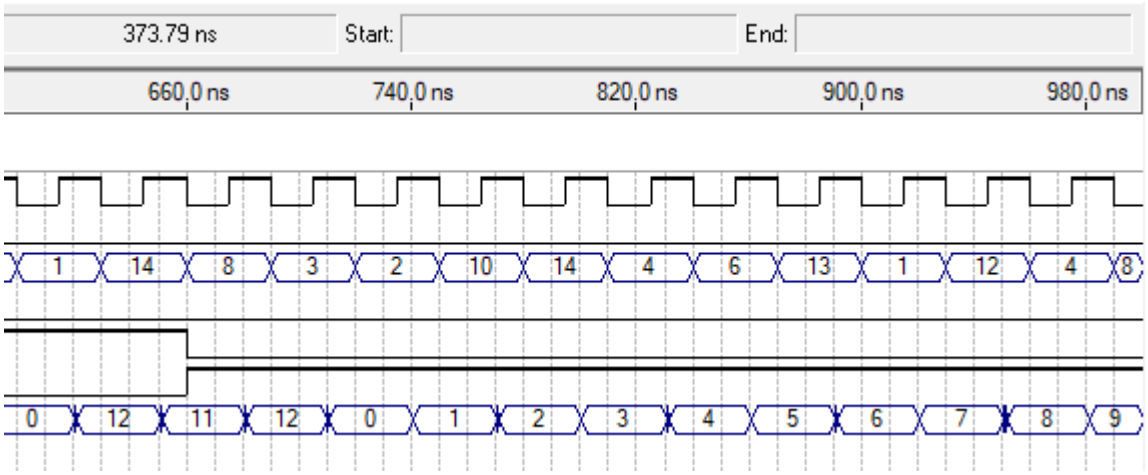




3.Numarator reversibil cu modul arbitrar.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity num3 is
port (clk, clr, load, up, down: in std_logic;
        data: in std_logic_vector(3 downto 0);
       count: out std_logic_vector(3 downto 0));
end num3;
architecture count4 of num3 is
signal cnt: std_logic_vector(3 downto 0);
  begin
    process (clr, clk)
       begin
         if clr='1' then cnt<="0000";
         elsif (clk'event and clk='1') then
               if load='1' then cnt<=data;
          elsif up='1' then
                    if cnt="1100" then cnt <="0000";
                   else cnt<=cnt+1;
                  end if;
                elsif down='1' then
                   if cnt="0000" then cnt<="1100";
                else cnt<=cnt-1;
             end if;
            else
        cnt<=cnt;
end if;
end if;
count<=cnt;
end process;
end count4;
```





Proiectați un numărător cu semnal asincron de resetare, încărcare paralelă, activ pe frontul negativ al semnalului de ceas, care numara din 2 in 2 . Starea initiala 4. Starea finala 14.