1. Licznik synchroniczny modulo 2^8, wyzwalany zboczem opadającym, pracujący w kodzie binarnym z dodatkowym wejściem zezwolenia zegara i asynchronicznym resetem.

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use IEEE.std_logic_unsigned.all;  -- WAŻNA BIBLIOTEKA !!!
4
5   entity Licznik_mod256_CE_aRES is
6        port(
7            clk : in STD_LOGIC;      -- sygnał zegarowy
8            ce : in STD_LOGIC;       -- sygnał zezwolenia Clock Enable
9            clr : in STD_LOGIC;      -- sygnał asynchronicznego resetu
10           Q : out STD_LOGIC_VECTOR(7 downto 0)
11           );
12  end Licznik_mod256_CE_aRES;
13
14  architecture MojaWyczesanaNazwaArchitektury of Licznik_mod256_CE_aRES is
15  signal StanLicznika : std_logic_vector(7 downto 0) := "00000000"; -- zmienna przechowująca stan licznika
16  begin
17
18      process (clk, clr)  --każda zmiana sygnału clk lub clr uruchomi process
19      begin
20          if (clr = '1') then                 -- po pierwsze sprawdź reset
21              StanLicznika <= "00000000";
22          else                                -- jak nie ma resetu, to:
23              if falling_edge(clk) then                   -- szukaj zbocza opadającego
24                  if (ce = '1') then                      -- sprawdź czy Clock Enable jest w stanie Hi
25                      StanLicznika <= StanLicznika + 1;   -- zwiększ stan licznika o 1
26                  end if;
27              end if;
28          end if;
29      end process;
30      Q <= StanLicznika;              -- podaj stan licznika na wyjście
31
32  end MojaWyczesanaNazwaArchitektury;
```

2. Licznik synchroniczny modulo 2^8, wyzwalany zboczem opadającym, pracujący w kodzie binarnym z dodatkowym wejściem zezwolenia zegara i synchronicznym resetem.

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use IEEE.std_logic_unsigned.all;  -- WAŻNA BIBLIOTEKA !!!
4
5   entity JakasNazwa is
6        port(
7            clk : in STD_LOGIC;      -- sygnał zegarowy
8            ce : in STD_LOGIC;       -- sygnał zezwolenia Clock Enable
9            clr : in STD_LOGIC;      -- sygnał synchronicznego resetu
10           Q : out STD_LOGIC_VECTOR(7 downto 0)
11           );
12  end JakasNazwa;
13
14  architecture NazwaArchitektury of JakasNazwa is
15  signal StanLicznika : std_logic_vector(7 downto 0) := "00000000"; -- zmienna przechowująca stan licznika
16  begin
17
18      process (clk)       --każda zmiana sygnału clk uruchomi process
19      begin
20          if falling_edge(clk) then           -- po pierwsze, szukaj zbocza opadającego
21              if (clr = '1') then                 -- sprawdź rest
22                  StanLicznika <= "00000000";
23              else                                -- a jak nie ma resetu, to:
24                  if (ce = '1') then                      -- sprawdź czy Clock Enable jest w stanie Hi
25                      StanLicznika <= StanLicznika + 1;   -- zwiększ stan licznika o 1
26                  end if;
27              end if;
28          end if;
29      end process;
30      Q <= StanLicznika;              -- podaj stan licznika na wyjście
31
32  end NazwaArchitektury;
```

3. Licznik synchroniczny modulo 34 zliczający w górę z dodatkowym wejściem zezwolenia zegara

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_unsigned.all;  -- WAŻNA BIBLIOTEKA !!!

entity Licznik_mod34_CE is
     port(
         clk : in STD_LOGIC;
         ce : in STD_LOGIC;
         Q : out STD_LOGIC_VECTOR(5 downto 0)
         );
end Licznik_mod34_CE;

architecture Licznik_mod34_CE of Licznik_mod34_CE is
signal StanLicznika : std_logic_vector(5 downto 0) := "000000"; -- zmienna przechowująca stan licznika
begin

    process (clk)   --każda zmiana sygnału clk uruchomi process
    begin
        if rising_edge(clk) then    -- szukaj zbocza narastającego
            if (ce = '1') then      -- sprawdź czy Clock Enable jest w stanie Hi
                if (StanLicznika = 33) then       -- jak poprzednio było max,
                    StanLicznika <= "000000";                           -- to się wyzeruj
                else                              -- a w przeciwnym wypadku
                    StanLicznika <= StanLicznika + 1;                   -- zwiększ stan licznika o 1
                end if;
            end if;
        end if;
    end process;
    Q <= StanLicznika;               -- podaj stan licznika na wyjście

end Licznik_mod34_CE;
```

4. Układ PISO (Parallel-In Serial-Out) z dodatkowym sygnałem zezwolenia zegara (Clock Enable).

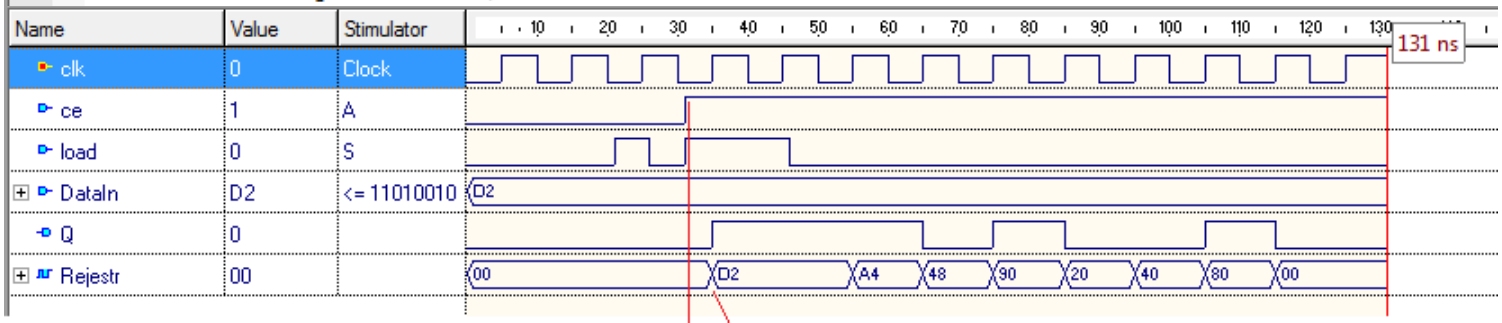
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity PISO is
     port(
          clk : in STD_LOGIC;      -- sygnał zegarowy
          ce : in STD_LOGIC;       -- sygnał zezwolenia Clock Enable
          load : in STD_LOGIC;     -- sygnał sterujący załadowaniem rejestru
          DataIn : in STD_LOGIC_VECTOR(7 downto 0); -- wejście równoległe
          Q : out STD_LOGIC        -- wyjście szeregowe
          );
end PISO;

architecture JednoZMozliwychRozwiazan of PISO is
signal Rejestr : std_logic_vector(7 downto 0) := "00000000"; -- zmienna przechowująca stan rejestru
begin

    process (clk)       --każda zmiana sygnału clk uruchomi process
    begin
        if rising_edge(clk) then            -- po pierwsze, szukaj zbocza narastającego
            if (ce = '1') then                  -- sprawdź czy Clock Enable jest w stanie Hi
                if (load = '1') then            -- sprawdź czy nastąpi załadowanie rejestru
                    Rejestr <= DataIn;
                else                            -- w przeciwnym wypadku przesuń dane o jeden w stronę MSB
                    Rejestr <= (Rejestr(6 downto 0) & '0');     -- na LSB wpisz '0'
                end if;
            end if;
        end if;
    end process;
    Q <= Rejestr(7);                -- podaj najstarszy bit rejestru na wyjście

end JednoZMozliwychRozwiazan;
```



gdy ce=0, to układ nie reaguje | **load = 1, wpisuję do rejestru z wejścia DataIn**

5. Układ SIPO (Serial-In-Parallel-Out) z dodatkowym sygnałem zezwolenia zegara (Clock Enable).

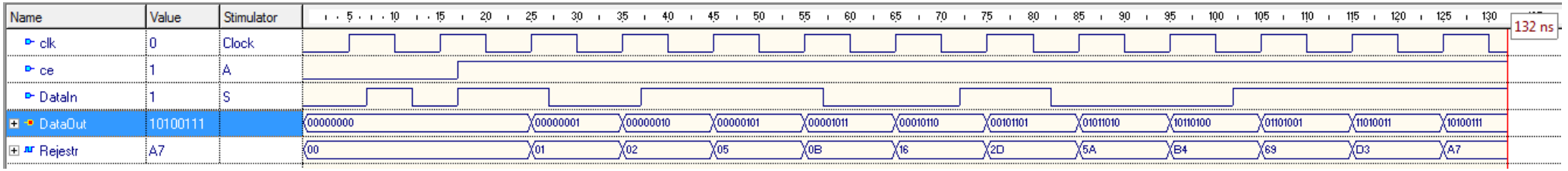
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity SIPO is
     port(
          clk : in STD_LOGIC;      -- sygnał zegarowy
          ce : in STD_LOGIC;       -- sygnał zezwolenia Clock Enable
          DataIn : in STD_LOGIC;   -- wejście szeregowe
          DataOut : out STD_LOGIC_VECTOR(7 downto 0) -- wyjście równoległe
          );
end SIPO;

architecture JednoZMozliwychRozwiazan of SIPO is
signal Rejestr : std_logic_vector(7 downto 0) := "00000000"; -- zmienna przechowująca stan rejestru
begin

    process (clk)       --każda zmiana sygnału clk uruchomi process
    begin
        if rising_edge(clk) then            -- po pierwsze, szukaj zbocza narastającego
            if (ce = '1') then                  -- sprawdź czy Clock Enable jest w stanie Hi
                Rejestr <= (Rejestr(6 downto 0) & DataIn);      -- przesuń reestr "w lewo" a na LSB wpisz poziom logiczny z wejścia szeregowego
            end if;
        end if;
    end process;
    DataOut <= Rejestr;             -- podaj zawartośc rejestru na wyjście równoległe

end JednoZMozliwychRozwiazan;
```



6. Licznik pracujący w kodzie innym niż binarny, np. w kodzie 001, 010, 101, 011, 111, 110, 100, 001, ...

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3
4   entity DziwnyLicznik is
5        port(
6            clk : in STD_LOGIC;      -- sygnał zegarowy
7            ce : in STD_LOGIC;       -- sygnał zezwolenia Clock Enable
8            clr : in STD_LOGIC;      -- sygnał asynchronicznego resetu
9            Q : out STD_LOGIC_VECTOR(2 downto 0)
10           );
11  end DziwnyLicznik;
12
13  architecture TopSecret of DziwnyLicznik is
14  signal StanLicznika : std_logic_vector(2 downto 0) := "001"; -- zmienna przechowująca stan automatu
15  begin
16
17      process (clk, clr)  --każda zmiana sygnału clk lub clr uruchomi process
18      begin
19          if (clr = '1') then                -- po pierwsze sprawdź reset
20              StanLicznika <= "001";                    -- wprowadź uklad do znanego/poprawnego stanu
21          else                               -- jak nie ma resetu, to:
22              if rising_edge(clk) then                  -- szukaj zbocza narastającego
23                  if (ce = '1') then                    -- sprawdź czy Clock Enable jest w stanie Hi
24                      case StanLicznika is              -- zobacz w jakim stanie teraz jesteś
25                          when "001" => StanLicznika <= "010";          -- i bezwarunkowo przejdź do kolejnego stanu
26                          when "010" => StanLicznika <= "101";
27                          when "101" => StanLicznika <= "011";
28                          when "011" => StanLicznika <= "111";
29                          when "111" => StanLicznika <= "110";
30                          when "110" => StanLicznika <= "100";
31                          when "100" => StanLicznika <= "001";
32                          when others => StanLicznika <= "000";   --nigdy nie powinien tu sie znaleźć
33                      end case;
34                  end if;
35              end if;
36          end if;
37      end process;
38      Q <= StanLicznika;              -- podaj stan licznika na wyjście
39
40  end TopSecret;
```