PRZYKŁADY PROJEKTOWE

Binarny dzielnik częstotliwości

Sterowanie diodą LED (migotki, choinki, PWM)

Zastosowania rejestrów przesuwnych

Cyfrowy pomiar wielkości fizycznej (metoda licznikowa)

Licznik w kodzie BCD

Sterownik wyświetlacza 7-segmentowego LED (4 cyfry)

Cyfrowy nadajnik radiowy (AM/FM)

Sterownik VGA

Cyfrowa synteza sygnału (DDS)

Eliminacja drgań styków mechanicznych

BINARNY DZIELNIK CZĘSTOTLIWOŚCI

10 bitów => 1024 => 1/1024 = 0,0009765625 ≈ 0,001 (:1000)

Zatem każde dodatkowe 10 bitów licznika, stanowi dzielnik częstotliwości przez 1000 (właściwie 1024 :-)

Np. dla 100 MHz:

1 0 0 | 0 0 0 | 0 0 0 Hz

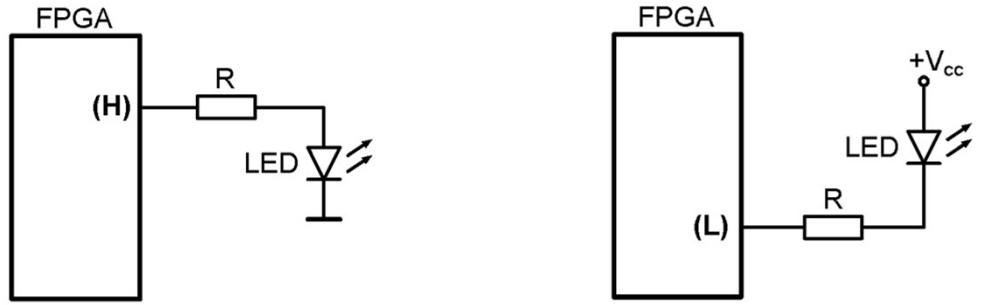
n-bitów + 10-bitów + 10-bitów

wówczas szacowanie częstotliwości wyjściowej to podzielenie liczby 100 przez np. 32 => 3 Hz, 64 => 1.5 Hz, 128 => 0.75 Hz ...

(5 bitów) (6 bitów) (7 bitów)

STEROWANIE DIODĄ LED

Sposób dołączenia wymusza stan aktywny, w którym dioda świeci.



Który sposób lepszy?

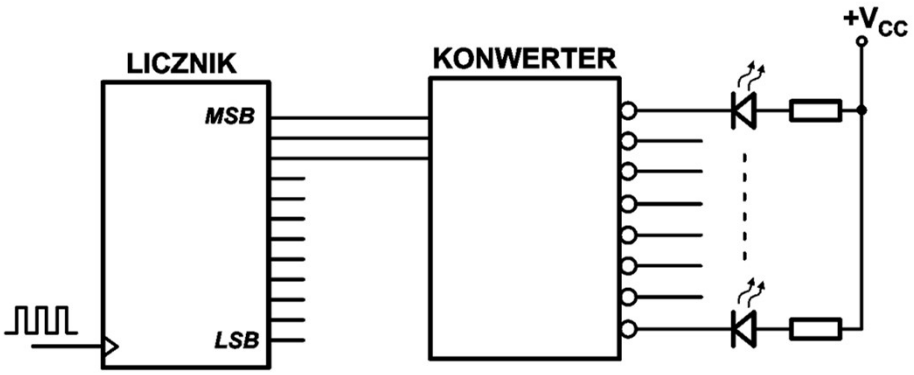
Zależy od „dalszego” użycia sygnału sterującego diodą...
(wpływ obciążenia na wartości napięć stanowiące poziomy logiczne)

STEROWNIK MIGOTKOWY...

Można dzielnik częstotliwości i licznik połączyć w jeden układ.

Wówczas jego najbardziej znaczące bity stanowią sygnały wejściowe dla konwertera, do którego dołączono diody LED.

Pozostałe bity licznika tworzą dzielnik częstotliwości zegara.



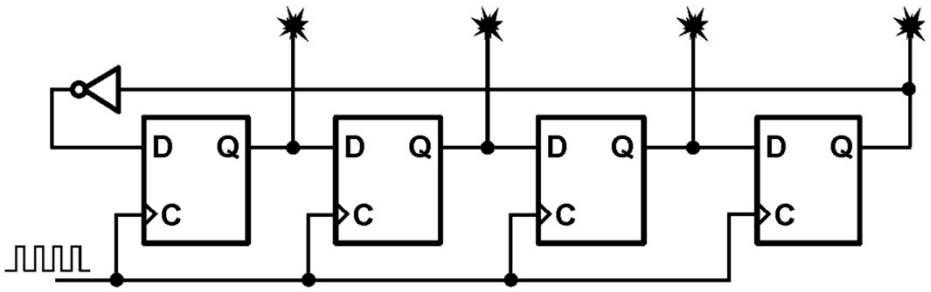
STEROWNIK MIGOTKOWY...

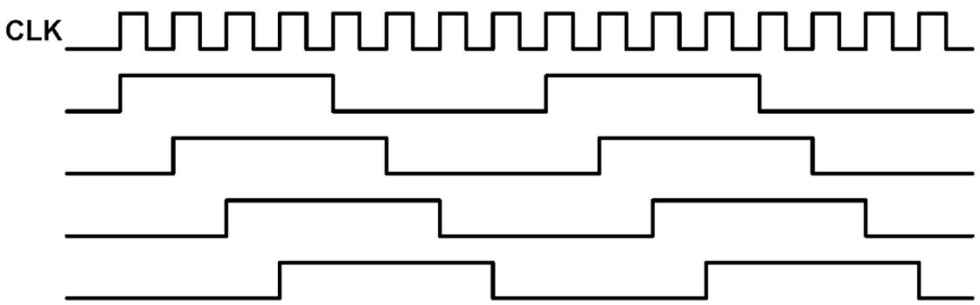
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity MIG is
  port ( Zegar : in std_logic;   -- 100 MHz
         LED : out std_logic_vector(7 downto 0) );
end MIG;

architecture Bech of MIG is
  signal Q : std_logic_vector( 25 downto 0 );
begin

 process( Zegar )  begin
   if rising_edge(Zegar) then Q <= Q + 1; end if;
 end process;

 process( Q(25 downto 23) )  begin        -- 3 najbardziej
   case Q(25 downto 23) is                -- znaczące bity
     when ”000”  =>  LED  <= ”00000111”;
     when ”001”  =>  LED  <= ”00011100”;
     when ”010”  =>  LED  <= ”00111000”;
     when ”011”  =>  LED  <= ”01110000”;
     when ”100”  =>  LED  <= ”11100000”;
     when others =>  LED  <= ”11111111”;
   end case;
 end process;

end Bech;
```

Licznik >

Konwerter >

LICZNIK JOHNSONA (np. 4-bitowy)

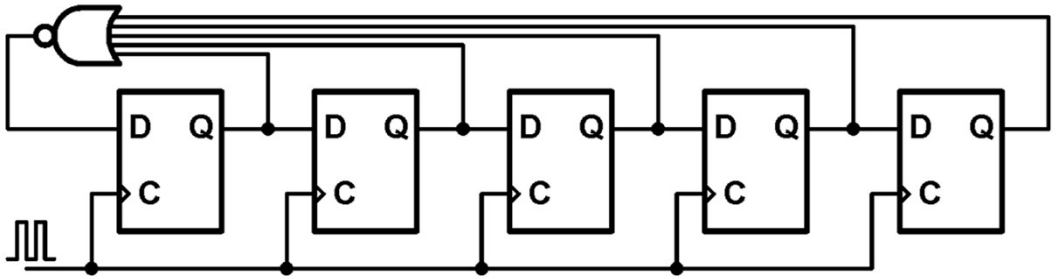


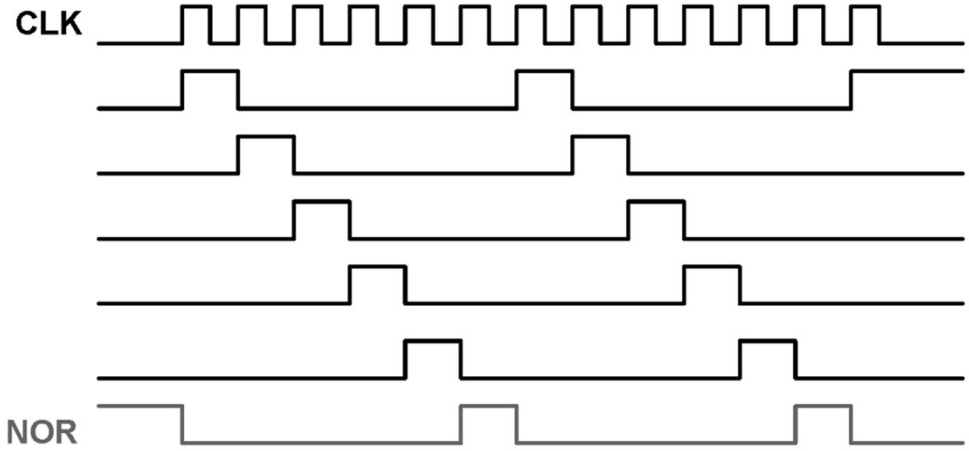



```
process(CLK)
begin
    if  rising_edge(CLK)  then
        Q <= Q(2 downto 0) & ( not Q(3) ) ;
    end if;
 end process;
```

ONE HOT „1”

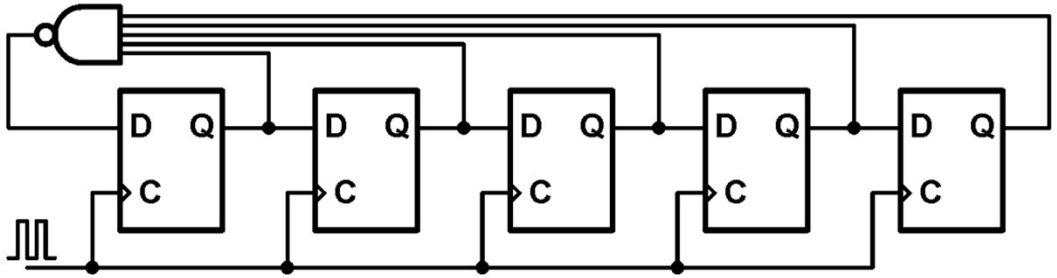
czyli układ krążącej jedynki z samokorekcją kodu...

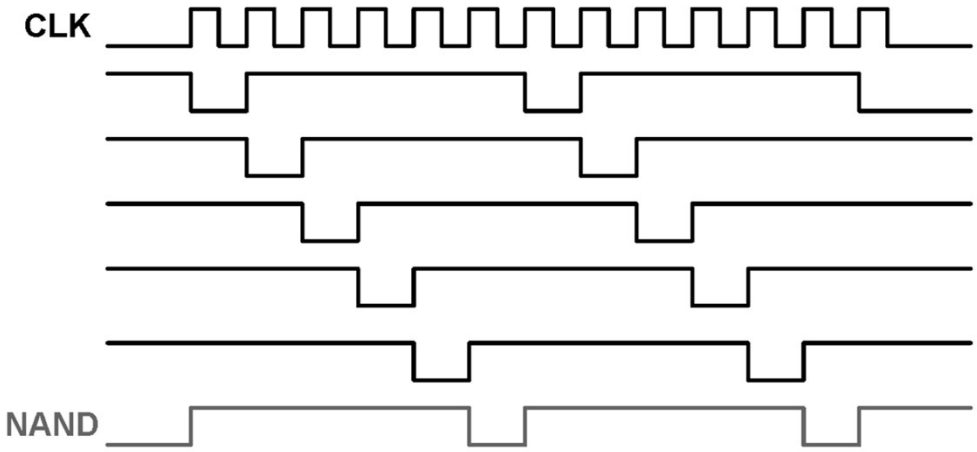




ONE HOT „0”

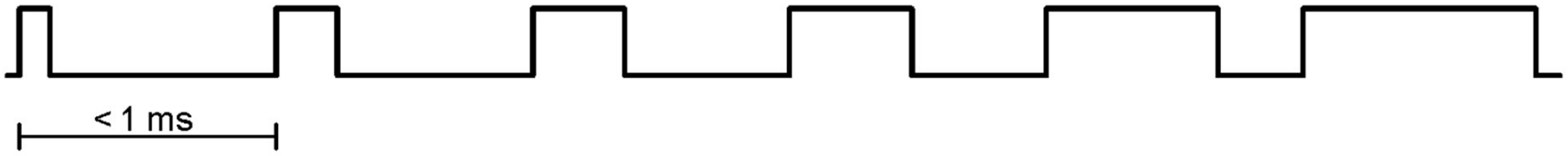
czyli układ krążącego zera z samokorekcją kodu...



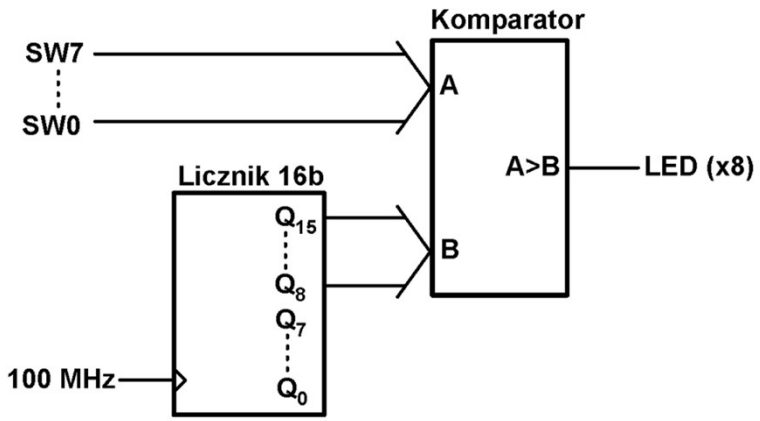


STEROWANIE JASNOŚCIĄ ŚWIECENIA DIODY LED

Jasność świecenia można zmieniać stosując sterowanie impulsowe PWM (*Pulse Width Modulation*), które polega na zmianie wypełnienia okresowego sygnału cyfrowego przy stałej częstotliwości (min. 1 kHz).



PWM – wypełnienie ustawiane przełącznikami zewnętrznymi



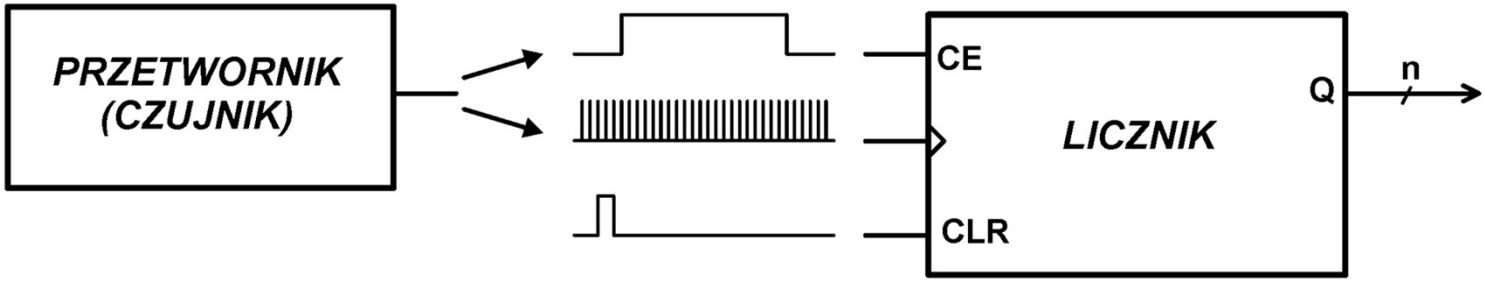
Q_7: 100 MHz / 256 = 390 625 Hz

Q_{15}: 390 625 Hz / 256 = 1525,9 Hz

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity PWM is
  port( CLK_100M : in std_logic;
        A : in std_logic_vector(7 downto 0);
        LED : out std_logic_vector(7 downto 0) );
end;

architecture Beh of PWM is
  signal Q : std_logic_vector(15 downto 0);
  signal B : std_logic_vector(7 downto 0);
  signal Y : std_logic;

begin

  process(CLK_100M)  begin       -- licznik 16-bitowy
    if rising_edge(CLK_100M) then Q<=Q+1; end if;
  end process;

  B <= Q(15 downto 8);

  Y <= '1' when A>B else '0';       -- komparator

  LED <= (others => Y);

end Beh;
```

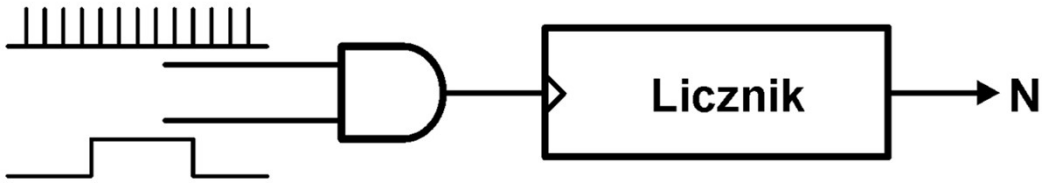
LICZNIKOWA METODA POMIARU WIELKOŚCI FIZYCZNEJ

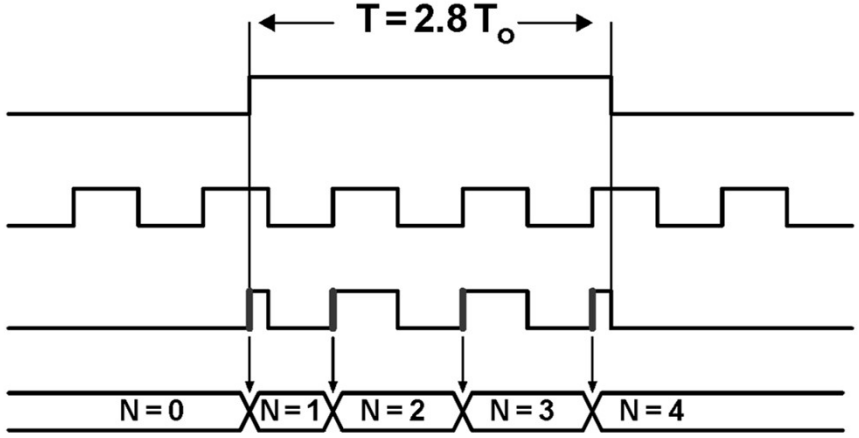
poprzez pomiar odcinka czasu lub częstotliwości sygnału...



Licznik powinien posiadać wejście synchronicznego zezwolenia na liczenie !!!

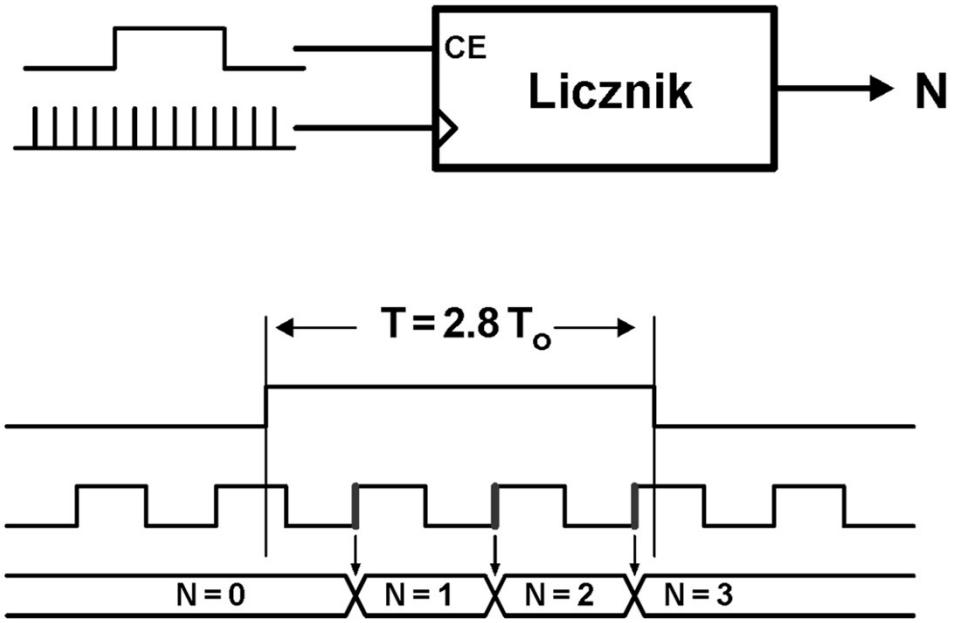
BRAMKA NA WEJŚCIU ZEGAROWYM LICZNIKA





NIE WOLNO STOSOWAĆ !!!

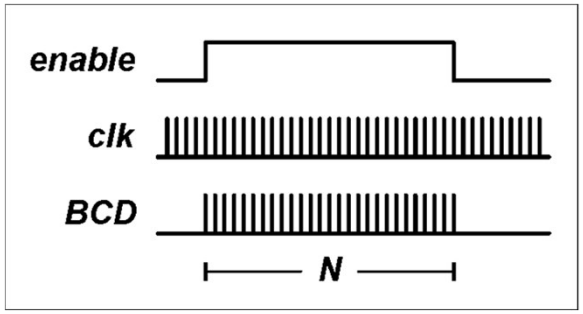
SYNCHRONIZACJA WYZWOLENIA LICZNIKA



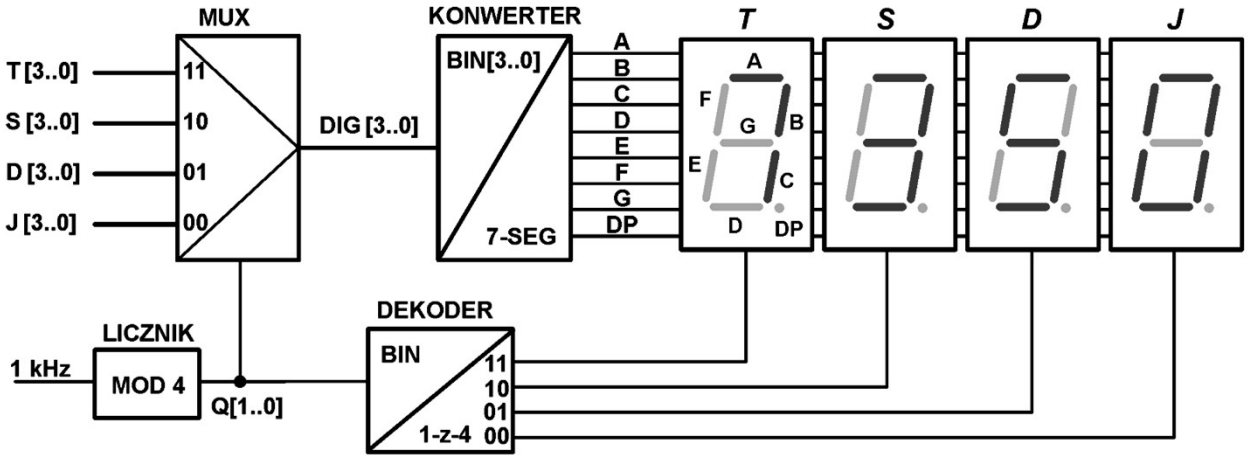

```
process(CLK)  begin
   if  rising_edge(CLK)  then
       if  CE='1'  then
           N <= N + 1;
       end if;
   end if;
end process;
```

MODUŁ 4-CYFROWEGO LICZNIKA BCD

```
process( reset, enable, clk )
begin
  if reset='1' then J<="0000"; D<="0000"; S<="0000"; T<="0000";
  elsif rising_edge(clk) then
    if enable='1' then
        if J<9 then J<=J+1;       -- Jedności
        else J<="0000";
              if D<9 then D<=D+1;     -- Dziesiątki
              else D<="0000";
                    if S<9 then S<=S+1;      -- Setki
                    else S<="0000";
                          if T<9 then T<=T+1;     -- Tysiące
                          else T<="0000";
                          end if;
                    end if;
              end if;
        end if;
    end if;
  end if;
end process;
BCD <= T & S & D & J;
```

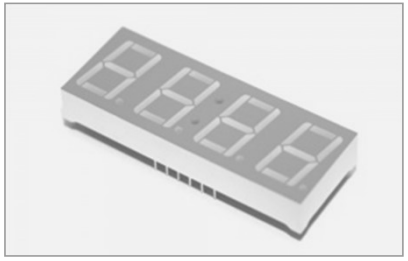


DYNAMICZNE STEROWANIE CZTEROCYFROWYM WYŚWIETLACZEM 7-SEGMENTOWYM LED



Q[1..0]

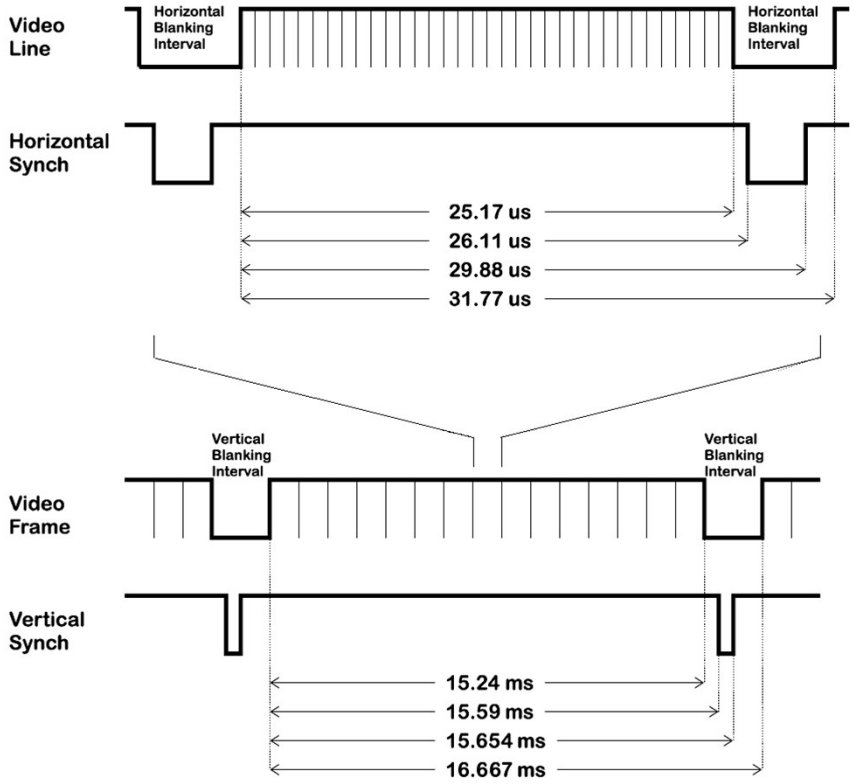
0 0 → J
0 1 → D
1 0 → S
1 1 → T



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity STEROWNIK is
  port( J,D,S,T : in  std_logic_vector(3 downto 0);
        SEG : out  std_logic_vector(6 downto 0);
        DP  : out  std_logic;
        EN  : out  std_logic_vector(3 downto 0) );
end;

architecture BEH of STEROWNIK is
  signal Q : std_logic_vector(1 downto 0);
  signal DIG : std_logic_vector(3 downto 0);

begin

  DP<='1';   -- wygaszenie kropek

  -- LICZNIK
  process(CLK)  begin
    if rising_edge(CLK) then Q<=Q+1; end if;
  end process;

-- MUX
  with Q select
    DIG <= J when "00",  D when "01",
           S when "10",  T when others;

-- KONWERTER
  process(DIG)  begin
    case DIG is
      when "0000" => SEG <= "0000001";  -- 0
      when "0001" => SEG <= "1001111";  -- 1
      when "0010" => SEG <= "0010010";  -- 2
      when "0011" => SEG <= "0000110";  -- 3
      when "0100" => SEG <= "1001100";  -- 4
      when "0101" => SEG <= "0100100";  -- 5
      when "0110" => SEG <= "0100000";  -- 6
      when "0111" => SEG <= "0001111";  -- 7
      when "1000" => SEG <= "0000000";  -- 8
      when "1001" => SEG <= "0000100";  -- 9
      when others => SEG <= "1111110";  -- –
    end case;
  end process;

-- DEKODER
  with Q select
    EN <= "1110" when "00", "1101" when "01",
          "1011" when "10", "0111" when others ;

end BEH;
```

STEROWNIK VGA

640x480, 60 Hz, 31.5 kHz



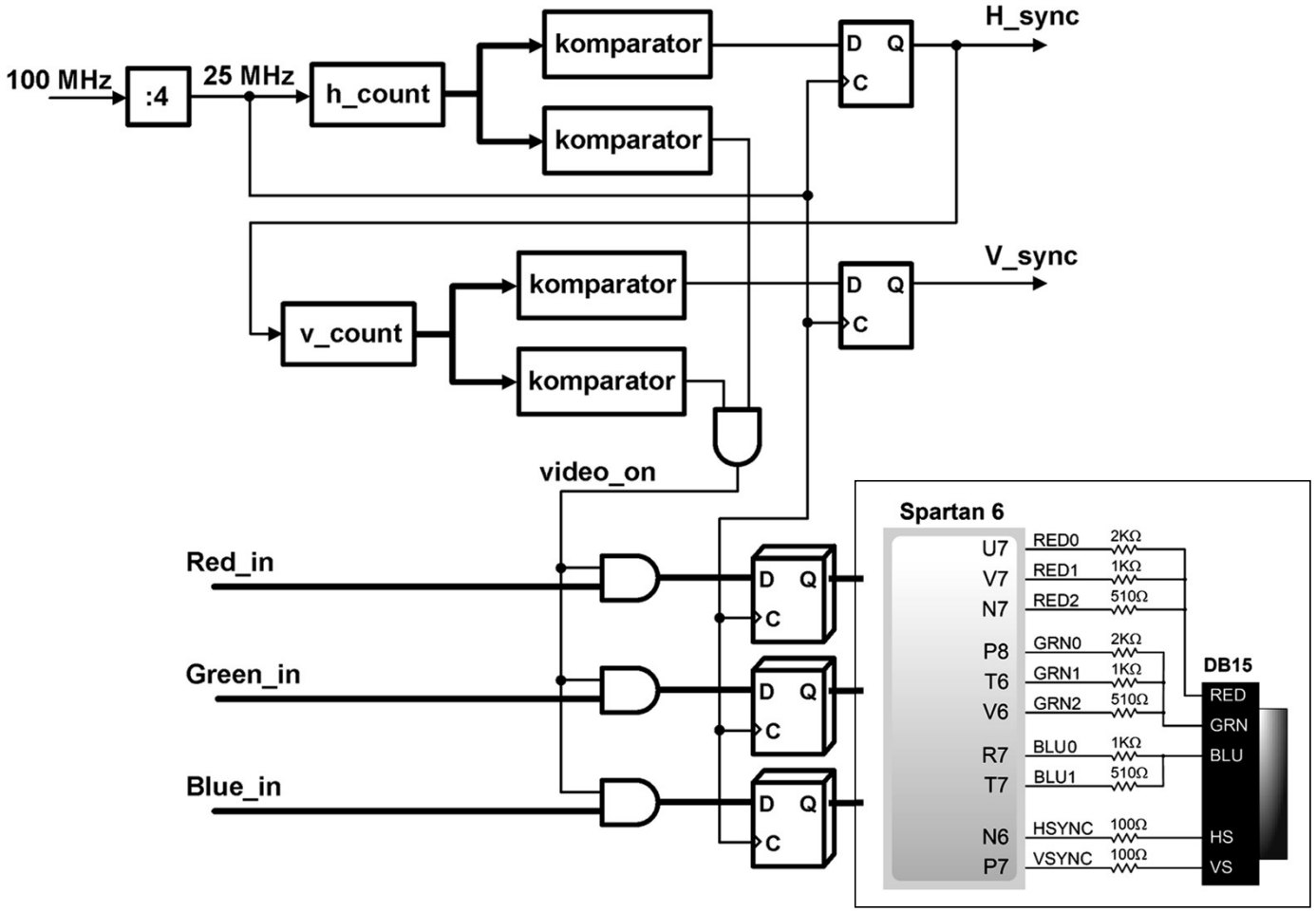
ZEGAR
25,17 ms / 640 = 39,328 ns
stąd 25,427 MHz

Ale łatwiej:
100 MHz / 4 = 25 MHz (czyli 40 ns)

Zatem:
LICZNIK KOLUMN
31,77 ms / 40 ns = 794,25

LICZNIK WIERSZY
16,667 ms / (794 • 40 ns) = 524,779

Schemat blokowy...

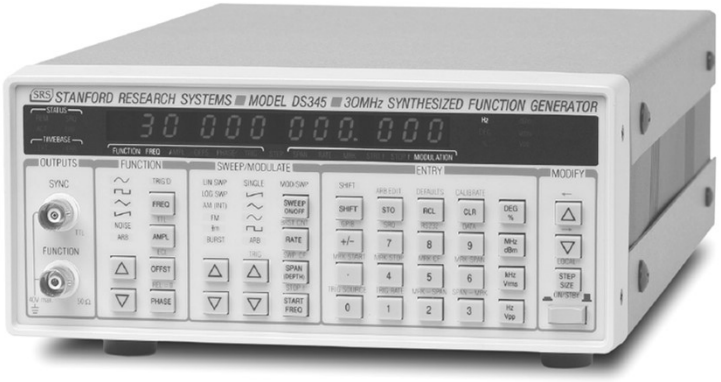


```
entity VGA is
  port( ...  );
end;

architecture beh of VGA is
  signal ... ;
begin

 process(CLOCK_100MHz)  -- DZIELNIK PRZEZ 4
  variable q_int : std_logic_vector(1 downto 0);
 begin
  if rising_edge(CLOCK_100MHz) then q_int:=q_int+1; end if;
  clock_25MHz<=q_int(1);
 end process;

 process(clock_25MHz) begin -- LICZNIK KOLUMN
  if rising_edge(clock_25MHz) then
   if h_count<793 then h_count<=h_count+1; else h_count<=(others=>'0'); end if;
  end if;
 end process;

 process(H_SYNC) begin -- LICZNIK WIERSZY
  if rising_edge(H_SYNC) then
   if v_count<524 then v_count<=v_count+1; else v_count<=(others=>'0'); end if;
  end if;
 end process;

 process(clock_25MHz) begin -- IMPULSY SYNCHRONIZACJI
  if rising_edge(clock_25MHz) then
   if (h_count>=660) and (h_count<=750) then H_SYNC<='0'; else H_SYNC<='1'; end if;
   if (v_count>=491) and (v_count<=492) then V_SYNC<='0'; else V_SYNC<='1'; end if;
  end if;
 end process;

 video_on<='1' when (h_count<640 and v_count<480) else '0';  -- WYGASZANIE

 process(clock_25MHz) begin -- GENEROWANIE SYGNAŁÓW REDn, GRNn i BLUn
  if rising_edge(clock_25MHz) then
   RED2 <= RGB(7) and video_on;   RED1 <= RGB(6) and video_on;    RED0 <= RGB(5) and video_on;
   GRN2 <= RGB(4) and video_on;   GRN1 <= RGB(3) and video_on;   GRN0 <= RGB(2) and video_on;
   BLU1 <= RGB(1) and video_on;    BLU0 <= RGB(0) and video_on;
  end if;
 end process;

end beh;
```

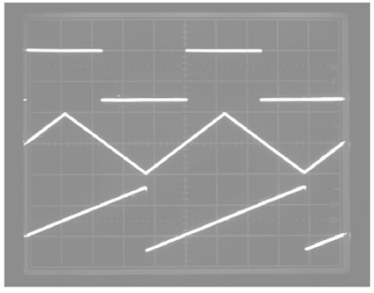
CYFROWA SYNTEZA SYGNAŁU
(DDS – *Direct Digital Synthesis*)

Synthesized Function Generator

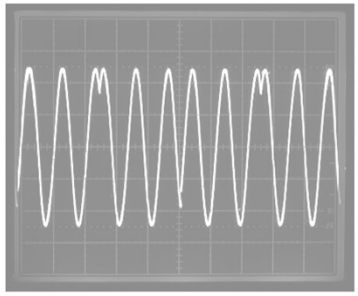
DS345 — 30 MHz function and arbitrary waveform generator



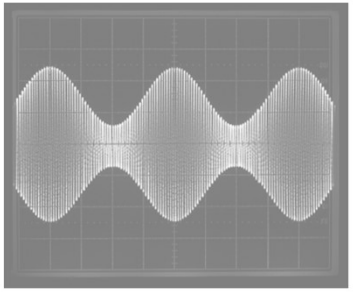
- **1 µHz to 30.2 MHz frequency range**
- **1 µHz frequency resolution**
- **Sine, square, ramp, triangle & noise**
- **Phase-continuous frequency sweeps**
- **AM, FM, burst and phase modulation**
- **16,300 point arbitrary waveforms**
- **10 MHz reference input**
- **RS-232 and GPIB interfaces (opt.)**
- **DS345 ... $1595** *(U.S. list)*



Square, triangle and ramp waveforms

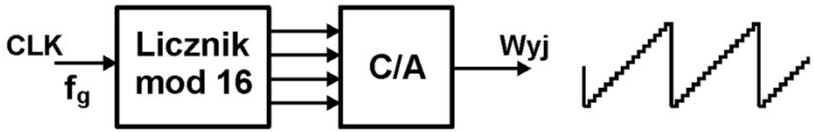


Phase modulation



Amplitude modulation

Najprostszy generator sygnału piłokształtnego to licznik binarny...




```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity PILA is
port( CLK : in  std_logic;                          -- 100 MHz
      S : out std_logic_vector(7 downto 0) );    -- 8-bitowy C/A
end PILA ;

architecture Beh of PILA is
  signal Q : std_logic_vector(19 downto 0);       -- 20-bitowy licznik binarny
begin

  process(CLK) begin
     if rising_edge(CLK) then Q<=Q+1; end if;
  end process;

  S <= Q(19 downto 12);   -- piła

end Beh;
```

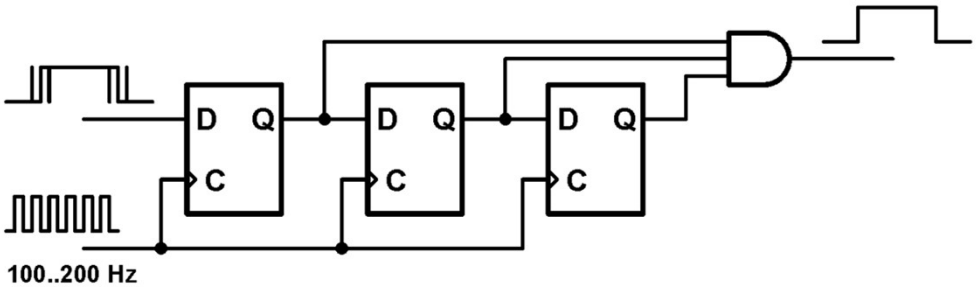
Generator sygnału o dowolnym kształcie...


```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


entity SINUS is
port( CLK : in  std_logic;
        S : out std_logic_vector(7 downto 0) );
end SINUS;


architecture Beh of SINUS is
  signal N : std_logic_vector(15 downto 0);
begin


 process(CLK) begin     --  Licznik
   if rising_edge(CLK) then N<=N+1; end if;
 end process;
```

```
 process(N (15 downto 12))  begin  -- ROM sinus
   case conv_integer( N(15 downto 12) ) is
      when  0 =>  S<="10000000";          -- 0
      when  1 =>  S<="10110000";          -- 22.5
      when  2 =>  S<="11011001";          -- 45
      when  3 =>  S<="11110101";          -- 67.5
      when  4 =>  S<="11111111";          -- 90
      when  5 =>  S<="11110101";          -- 112.5
      when  6 =>  S<="11011001";          -- 135
      when  7 =>  S<="10110000";          -- 157.5
      when  8 =>  S<="10000000";          -- 180
      when  9 =>  S<="01001111";          -- 202.5
      when 10 =>  S<="00100101";          -- 225
      when 11 =>  S<="00001010";          -- 247.5
      when 12 =>  S<="00000001";          -- 270
      when 13 =>  S<="00001010";          -- 292.5
      when 14 =>  S<="00100101";          -- 315
      when 15 =>  S<="01001110";          -- 337.5
      when others => S<=(others=>'0');
   end case;
 end process;

end Beh;
```

DEBOUNCER

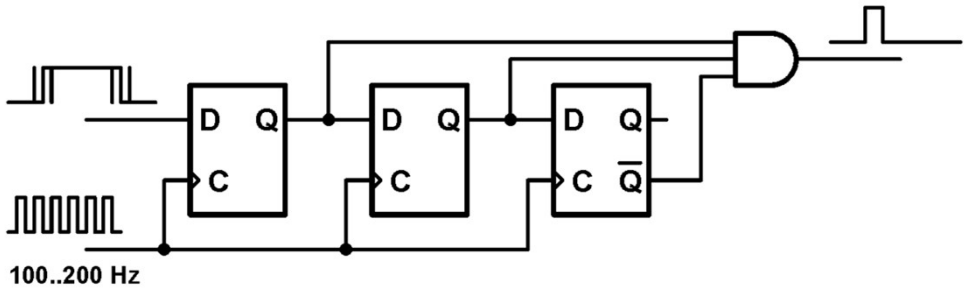
...czyli układ do eliminacji drgań styków mechanicznych...




```
process(CLK)
begin
     if  rising_edge(CLK)  then
          Q <= Q(1) & Q(0) & BTN ;
     end if;
end process;
BTN_sync <= Q(2) and Q(1) and Q(0);
```

POJEDYNCZY IMPULS ZEGAROWY

...czyli układ generujący pojedynczy impuls pod wpływem naciśnięcia przycisku mechanicznego...




```
process(CLK)
begin
     if  rising_edge(CLK)  then
          Q <= Q(1) & Q(0) & BTN ;
     end if;
end process;
PULSE <= (not Q(2)) and Q(1) and Q(0);
```