Raport z Laboratorium Układy programowalne 2

Tytuł ćwiczenia: Licznik częstotliwości

Data wykonania: 11 kwietnia 2025

Autor: Patryk Kowalski 267671

Spis treści

1 Cel ćwiczenia

Celem ćwiczenia jest zaprojektowanie licznika częstotliwości z akwizycją co 1000 ms.

2 Przebieg ćwiczenia

Do wykonania ćwiczenia wykorzystano do tego płytkę Xilinx NEXYS A7, środowisko Vivado, język programowania VHDL. Do przetestowania działania wykorzystano zewnętrzny generator analog discovery 2 wraz z oprogramowaniem Waveforms.

Zaprojektowany licznik składa się z trzech głównych modułów:

- Dzielnik częstotliwości 100 MHz do generowania sygnału 1-sekundowego (ClockDivider)
- Dzielnik częstotliwości do obsługi wyświetlacza (DisplayClockDivider)
- Główny moduł licznika częstotliwości (FrequencyCounter)

Sygnał wejściowy dostarczany jest na pin JB(1) z zewnętrznego generatora. Licznik zlicza impulsy przez 1 sekundę, a następnie wyświetla wartość na 7-segmentowym wyświetlaczu płytki NEXYS A7.

3 Kod VHDL

Poniżej przedstawiono kod VHDL zaprojektowanego licznika.

3.1 Moduł główny - FrequencyCounter

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity FrequencyCounter is
7      Port (
8          CLK100MHZ : in STD_LOGIC;
9          JB : in STD_LOGIC_VECTOR(1 to 10);
10         SEG : out STD_LOGIC_VECTOR(6 downto 0);
11         AN  : out STD_LOGIC_VECTOR(7 downto 0)
12     );
13 end FrequencyCounter;
14
15 architecture Behavioral of FrequencyCounter is
16     signal one_sec : STD_LOGIC := '0';
17     signal disp_clk : STD_LOGIC := '0';
18     signal pulse_cnt : INTEGER := 0;
19     signal display_value : INTEGER := 0;
20     signal digit_index : INTEGER range 0 to 7 := 0;
21     signal current_digit_value : INTEGER range 0 to 9 := 0;
22     signal JB1_sync1, JB1_sync2 : STD_LOGIC := '0';
23     signal JB1_prev : STD_LOGIC := '0';
24
25     component ClockDivider
26         Port (CLK100MHZ : in STD_LOGIC; ONE_SEC : out STD_LOGIC);
27     end component;
28
29     component DisplayClockDivider
30         Port (CLK100MHZ : in STD_LOGIC; DISP_CLK : out STD_LOGIC);
31     end component;
32
33 begin
34     ClockDiv_Inst : ClockDivider port map (CLK100MHZ => CLK100MHZ, ONE_SEC =>
           one_sec);
```

```
35      DispClk_Inst : DisplayClockDivider port map (CLK100MHZ => CLK100MHZ, DISP_CLK =>
            disp_clk);
36
37      process (CLK100MHZ)
38      begin
39          if rising_edge(CLK100MHZ) then
40              JB1_sync1 <= JB(1);
41              JB1_sync2 <= JB1_sync1;
42          end if;
43      end process;
44
45      process (CLK100MHZ)
46      begin
47          if rising_edge(CLK100MHZ) then
48              if JB1_prev = '0' and JB1_sync2 = '1' then
49                  pulse_cnt <= pulse_cnt + 1;
50              end if;
51              JB1_prev <= JB1_sync2;
52
53              if one_sec = '1' then
54                  display_value <= pulse_cnt;
55                  pulse_cnt <= 0;
56              end if;
57          end if;
58      end process;
59
60      process (CLK100MHZ)
61      begin
62          if rising_edge(CLK100MHZ) then
63              if disp_clk = '1' then
64                  digit_index <= (digit_index + 1) mod 8;
65              end if;
66          end if;
67      end process;
68
69      process (digit_index)
70      begin
71          case digit_index is
72              when 0 => AN <= "11111110";
73              when 1 => AN <= "11111101";
74              when 2 => AN <= "11111011";
75              when 3 => AN <= "11110111";
76              when 4 => AN <= "11101111";
77              when 5 => AN <= "11011111";
78              when 6 => AN <= "10111111";
79              when 7 => AN <= "01111111";
80          end case;
81      end process;
82
83      process (digit_index, display_value)
84      begin
85          case digit_index is
86              when 0 => current_digit_value <= display_value mod 10;
87              when 1 => current_digit_value <= (display_value / 10) mod 10;
88              when 2 => current_digit_value <= (display_value / 100) mod 10;
89              when 3 => current_digit_value <= (display_value / 1000) mod 10;
90              when 4 => current_digit_value <= (display_value / 10000) mod 10;
91              when 5 => current_digit_value <= (display_value / 100000) mod 10;
92              when 6 => current_digit_value <= (display_value / 1000000) mod 10;
93              when 7 => current_digit_value <= (display_value / 10000000) mod 10;
94          end case;
95      end process;
96
```

```
    process (current_digit_value)
    begin
        case current_digit_value is
            when 0 => SEG <= "1000000";
            when 1 => SEG <= "1111001";
            when 2 => SEG <= "0100100";
            when 3 => SEG <= "0110000";
            when 4 => SEG <= "0011001";
            when 5 => SEG <= "0010010";
            when 6 => SEG <= "0000010";
            when 7 => SEG <= "1111000";
            when 8 => SEG <= "0000000";
            when 9 => SEG <= "0010000";
            when others => SEG <= "1111111";
        end case;
    end process;

end Behavioral;
```

Listing 1: Kod modułu głównego licznika częstotliwości

3.2 Moduł dzielnika zegara - ClockDivider

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ClockDivider is
    Port (
        CLK100MHZ : in STD_LOGIC;
        ONE_SEC : out STD_LOGIC
    );
end ClockDivider;

architecture Behavioral of ClockDivider is
    signal clk_div : INTEGER := 0;
begin
    process (CLK100MHZ)
    begin
        if rising_edge(CLK100MHZ) then
            if clk_div = 99999999 then
                clk_div <= 0;
                ONE_SEC <= '1';
            else
                clk_div <= clk_div + 1;
                ONE_SEC <= '0';
            end if;
        end if;
    end process;
end Behavioral;
```

Listing 2: Kod modułu dzielnika zegara (1-sekundowy)

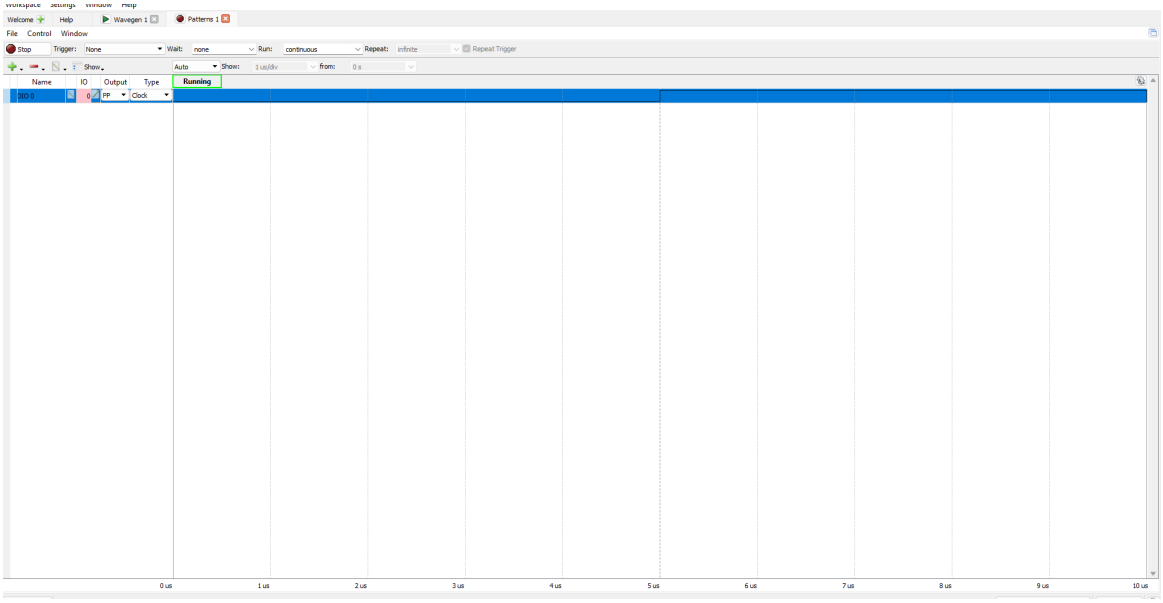
3.3 Moduł sterownika wyświetlacza - DisplayClockDivider

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DisplayClockDivider is
    Port (
        CLK100MHZ : in STD_LOGIC;
        DISP_CLK : out STD_LOGIC
```

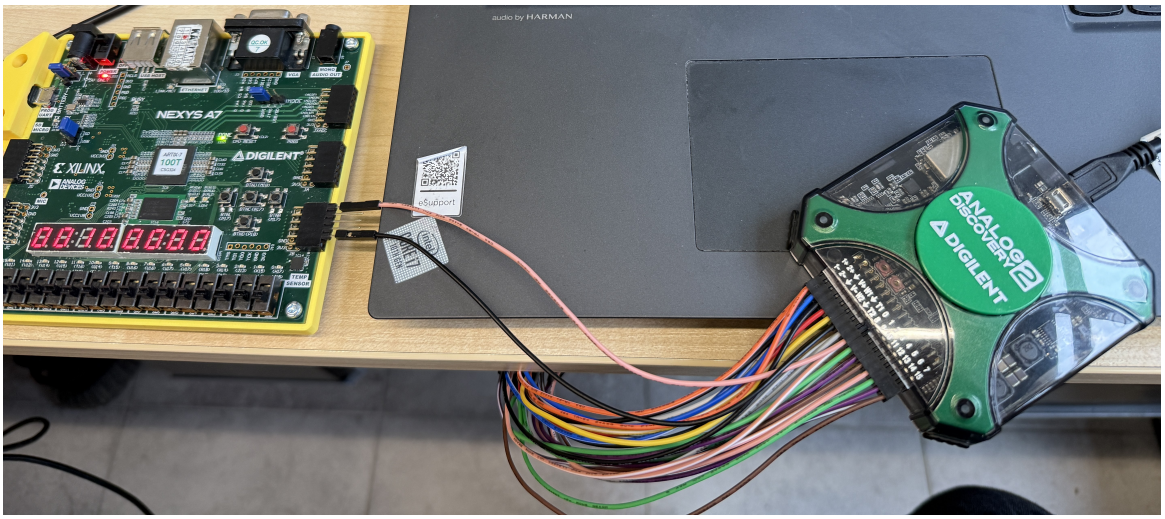
```
    );
end DisplayClockDivider;

architecture Behavioral of DisplayClockDivider is
    signal clk_div : INTEGER := 0;
begin
    process (CLK100MHZ)
    begin
        if rising_edge(CLK100MHZ) then
            if clk_div = 99999 then
                clk_div <= 0;
                DISP_CLK <= '1';
            else
                clk_div <= clk_div + 1;
                DISP_CLK <= '0';
            end if;
        end if;
    end process;
end Behavioral;
```

Listing 3: Kod modułu sterownika wyświetlacza

4 Wynik



Rysunek 1: Zrzut ekranu z programu WaveForms prezentujący sygnał testowy



Rysunek 2: Test z płytką NEXYS A7 i generatorem Analog Discovery 2

5 Wnioski

Zrealizowany układ poprawnie zlicza impulsy zewnętrznego sygnału i wyświetla częstotliwość (liczbę impulsów na sekundę) na wyświetlaczu 7-segmentowym płytki NEXYS A7.