# Sprawozdanie

# Inżynieria Układów Programowalnych

| | |
|---|---|
| **Nazwa ćwiczenia** | Sterowanie wyświetlaczem LED |
| **Numer ćwiczenia** | 6 |
| **Imię** | Mateusz |
| **Nazwisko** | Gabryel |
| **Numer indeksu** | 181329 |
| **Kierunek** | Elektronika i Telekomunikacja |
| **Numer grupy dziekańskiej** | 3 |

## Opis działania zadania:

Układ po zaprogramowaniu, umożliwia sterowanie wyświetlaczem LED. Głównym celem ćwiczenia jest sterowanie czterema cyframi wyświetlacza LED. Zaprojektowany układ poprzez ustawienie przycisków oraz przełączników wyświetla na wyświetlaczu LED dowolną 4 -znakową liczbę szesnastkową wraz z przecinkami. Zaprojektowany układ składa się z dwóch komponentów: display oraz encoder & memory. Pierwszy z nich jest odpowiedzialny za sterowanie wyświetlania na poszczególnych segmentach wyświetlacza. Drugi z nich jest odpowiedzialny za kodowanie wprowadzonych bitów oraz przekazanie do wyświetlenia. W projekcie został zastosowany moduł dzielnika częstotliwości z zadania numer 5, aby umożliwić przełączanie wyświetlacza na kolejną cyfrę co 1 [ms]. W przeprowadzonej weryfikacji na płytce został wprowadzony na wyświetlacz LED numer indeksu studenta w postaci heksadecymalnej, przy czym 4 najstarsze cyfry zostały wprowadzone na kolejne wyświetlacze 7 – segmentowe, a najmłodsza cyfra została przedstawiona za pomocą kropek.

Numer indeksu studenta w postaci decymalnej: 181329

Numer indeksu studenta w postaci heksadecymalnej: 2C451

4 najstarsze cyfry wyświetlane na wyświetlaczach 7 – segmentowych: 2C45

Najmłodsza cyfra przedstawiona za pomocą kropek: 1

Układ posiada łącznie 12 przycisków:

| NUMER PRZYCISKU | PRZYCISK | ZADANIE PRZYCISKU |
|---|---|---|
| 1 | SW7 | Zapalenie kropki wyświetlacza AN3 |
| 2 | SW6 | Zapalenie kropki wyświetlacza AN2 |
| 3 | SW5 | Zapalenie kropki wyświetlacza AN1 |
| 4 | SW4 | Zapalenie kropki wyświetlacza AN0 |
| 5 | SW3 | Ustawienie 4 – bitowej wartości |
| 6 | SW2 | Ustawienie 4 – bitowej wartości |
| 7 | SW1 | Ustawienie 4 – bitowej wartości |
| 8 | SW0 | Ustawienie 4 – bitowej wartości |
| 9 | BTN3 | Wprowadzenie wprowadzonej wartości na wyświetlacz AN3 |
| 10 | BTN2 | Wprowadzenie wprowadzonej wartości na wyświetlacz AN2 |
| 11 | BTN1 | Wprowadzenie wprowadzonej wartości na wyświetlacz AN1 |
| 12 | BTN0 | Wprowadzenie wprowadzonej wartości na wyświetlacz AN0 |

# Opis pliku top_encoder.vhd:

1.

2.

3.

4.

5.

```
1   -- Mateusz Gabryel 181329 EiT 3
2   library IEEE;
3   use IEEE.STD_LOGIC_1164.ALL;
4   use IEEE.STD_LOGIC_SIGNED.ALL;
5   use IEEE.STD_LOGIC_UNSIGNED.ALL;
6   use IEEE.STD_LOGIC_ARITH.ALL;
7   use STD.STANDARD.ALL;
8
9   ENTITY top_encoder IS
10      PORT ( clk_i : in STD_LOGIC;
11             btn_i : in STD_LOGIC_VECTOR (3 downto 0);
12             sw_i : in STD_LOGIC_VECTOR (7 downto 0);
13             led7_an_o : out STD_LOGIC_VECTOR (3 downto 0);
14             led7_seg_o : out STD_LOGIC_VECTOR (7 downto 0)
15            );
16  END top_encoder;
17
18  ARCHITECTURE Behavioral OF top_encoder IS
19
20  COMPONENT top_divider IS
21      PORT ( clk_i: in STD_LOGIC;
22             rst_i: in STD_LOGIC;
23             clk_divider: out STD_LOGIC
24            );
25  END COMPONENT;
26
27  COMPONENT top_display IS
28      PORT ( led7_seg_o: out STD_LOGIC_VECTOR(7 downto 0);
29             led7_an_o: out STD_LOGIC_VECTOR(3 downto 0);
30             digit_i: in STD_LOGIC_VECTOR(31 downto 0);
31             display_clk: in STD_LOGIC;
32             rst_i: in STD_LOGIC
33            );
34  END COMPONENT;
35
36  signal a: STD_LOGIC_VECTOR (6 downto 0):= "1111111";
37  signal digit_i: STD_LOGIC_VECTOR (31 downto 0);
38  signal b: STD_LOGIC;
39
40  BEGIN
41
42  divider: top_divider
43      PORT MAP ( clk_i => clk_i,
44                 rst_i => '0',
45                  clk_divider => b
46                 );
47
48  display: top_display
49       PORT MAP ( led7_seg_o => led7_seg_o,
50                  led7_an_o => led7_an_o,
51                  digit_i => digit_i,
52                  display_clk => b,
53                  rst_i => '0'
54                 );
55
56  WITH sw_i(3 downto 0) SELECT -- realizacja przypisania selektywnego do zak
57        a <= "0000001" WHEN "0000",  -- 0
58             "1001111" WHEN "0001",  -- 1
59             "0010010" WHEN "0010",  -- 2
60             "0000110" WHEN "0011",  -- 3
61             "1001100" WHEN "0100",  -- 4
62             "0100100" WHEN "0101",  -- 5
63             "0100000" WHEN "0110",  -- 6
64             "0001111" WHEN "0111",  -- 7
65             "0000000" WHEN "1000",  -- 8
66             "0000100" WHEN "1001",  -- 9
67             "0001000" WHEN "1010",  -- A
68             "1100000" WHEN "1011",  -- B
69             "0110001" WHEN "1100",  -- C
70             "1000010" WHEN "1101",  -- D
71             "0110000" WHEN "1110",  -- E
72             "0111000" WHEN "1111",  -- F
73             "1111111" WHEN others;  -- ? (nieokreslone)
74
75  PROCESS(clk_i, sw_i, btn_i,a)
76
77  BEGIN
78      if(rising_edge(clk_i)) then
79
80          if(sw_i(4) = '1') then digit_i(0) <= '0'; -- kropka dla AN0
81              else
82                  digit_i(0) <= '1';
83          END if;
84
85          if(sw_i(5) = '1') then digit_i(8) <= '0'; -- kropka dla AN1
86              else
87                  digit_i(8) <= '1';
88          END if;
89
90          if(sw_i(6) = '1') then digit_i(16) <= '0'; -- kropka dla AN2
91              else
92                  digit_i(16) <= '1';
93          END if;
94
95          if(sw_i(7) = '1') then digit_i(24) <= '0'; -- kropka dla AN3
96              else
97                  digit_i(24) <= '1';
98          END if;
99
100         -- wpisanie ustawionej wartosci na odpowiedni segment
101         if(btn_i(0) = '1') then digit_i(7 downto 1) <= a;   -- wpisanie na AN0
102         END if;
103         if(btn_i(1) = '1') then digit_i(15 downto 9) <= a;  -- wpisanie na AN1
104         END if;
105         if(btn_i(2) = '1') then digit_i(23 downto 17) <= a; -- wpisanie na AN2
106         END if;
107         if(btn_i(3) = '1') then digit_i(31 downto 25) <= a; -- wpisanie na AN3
108         END if;
109
110     END if;
111 END PROCESS;
112 END Behavioral;
```

**1.** Deklaracja użycia biblioteki IEEE oraz deklaracja użycia pakietów i ich składników.

**2.** Deklaracja przydziału portów:

- clk_i: zegar wejściowy
- btn_i: przyciski umożliwiające wprowadzenie wartości na poszczególne wyświetlacze
- sw_i:  przyciski wprowadzające wartości
- led7_an_o: pojedyncza część wyświetlacza LED
- led7_seg_o: segmenty pojedynczego wyświetlacza

**3.** Realizacja przypisania selektywnego do zakodowania wprowadzanych wartości z postaci binarnej na postać heksadecymalną.

**4.** Włączenie kropek dziesiętnych danego segmentu wyświetlacza LED.

**5.** Przepisanie wprowadzonej oraz zakodowanej wartości na żądany za pomocą przycisków BTNx segment wyświetlacza LED.

## Do prawidłowego działania układu na płytce FPGA jest potrzebny plik constraint iup6.xdc:

```
# Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk_i }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports { clk_i }];
#virtual DIP Switches
set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { sw_i[0] }];
set_property -dict { PACKAGE_PIN G16   IOSTANDARD LVCMOS33 } [get_ports { sw_i[1] }];
set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { sw_i[2] }];
set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { sw_i[3] }];
set_property -dict { PACKAGE_PIN G18   IOSTANDARD LVCMOS33 } [get_ports { sw_i[4] }];
set_property -dict { PACKAGE_PIN F18   IOSTANDARD LVCMOS33 } [get_ports { sw_i[5] }];
set_property -dict { PACKAGE_PIN E17   IOSTANDARD LVCMOS33 } [get_ports { sw_i[6] }];
set_property -dict { PACKAGE_PIN D17   IOSTANDARD LVCMOS33 } [get_ports { sw_i[7] }];
#virtual Buttons
set_property -dict { PACKAGE_PIN C17   IOSTANDARD LVCMOS33 } [get_ports { btn_i[0] }];
set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 } [get_ports { btn_i[1] }];
set_property -dict { PACKAGE_PIN E18   IOSTANDARD LVCMOS33 } [get_ports { btn_i[2] }];
set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { btn_i[3] }];
#7-segment display
#segments (active low)
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { led7_seg_o[7] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { led7_seg_o[6] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { led7_seg_o[5] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { led7_seg_o[4] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { led7_seg_o[3] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { led7_seg_o[2] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { led7_seg_o[1] }]; #IO_L4P_T0_D04_14 Sch=cg
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { led7_seg_o[0] }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
#digits (active low)
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { led7_an_o[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { led7_an_o[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { led7_an_o[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { led7_an_o[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```

## Opis pliku top_display.vhd:

**1.**

**2.**

**3.**

**4.**

```
1  -- Mateusz Gabryel 181329 EiT 3
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_SIGNED.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;
6  use IEEE.STD_LOGIC_ARITH.ALL;
7  use STD.STANDARD.ALL;
8
9  ENTITY top_display IS
10   PORT ( led7_seg_o: out STD_LOGIC_VECTOR(7 downto 0):= "00000000";
11          led7_an_o: out STD_LOGIC_VECTOR(3 downto 0):= "0000";
12          digit_i: in STD_LOGIC_VECTOR(31 downto 0);
13          display_clk: in STD_LOGIC;
14          rst_i: in STD_LOGIC
15         );
16 END top_display;
17
18 ARCHITECTURE Behavioral OF top_display IS
19 signal counter_display : INTEGER := 0; --licznik do rejestru
20 BEGIN
21 WITH counter_display SELECT --realizacja przypisania selektywnego do realizacji rejestru przesuwne
22     led7_an_o <= "0111" WHEN 1, -- AN3
23                  "1011" WHEN 2, -- AN2
24                  "1101" WHEN 3, -- AN1
25                  "1110" WHEN 4, -- AN0
26                  "0000" WHEN OTHERS;
27     WITH counter_display SELECT
28     led7_seg_o <= digit_i(31 downto 24) WHEN 1, -- AN3
29                   digit_i(23 downto 16) WHEN 2, -- AN2
30                   digit_i(15 downto 8) WHEN 3,  -- AN1
31                   digit_i(7 downto 0) WHEN 4,   -- AN0
32                   "00000000" WHEN OTHERS;
33
34 PROCESS (display_clk, rst_i)
35
36 BEGIN
37     if(rst_i = '1') then -- reset
38         counter_display <= 0;
39     elsif(rising_edge(display_clk)) then
40             if(counter_display=4) then  -- zmiana z AN0 na AN3 (symulacja)
41                 counter_display <= 1;   -- AN3
42             else
43                 counter_display <= counter_display+1; -- dalsze kroki rejestru przesuwnego
44             END if;
45           END if;
46 END PROCESS;
47 END Behavioral;
```

**1.** Deklaracja użycia biblioteki IEEE oraz deklaracja użycia pakietów i ich składników.

**2.** Deklaracja przydziału portów:

- led7_seg_o: segmenty pojedynczego wyświetlacza
- led7_an_o: pojedyncza część wyświetlacza LED
- digit_i: wejście zakodowanych wartości
- display_clk: zegar wyświetlacza
- rst_i: reset asynchroniczny

**3.** Realizacja przypisania selektywnego do realizacji rejestru przesuwnego, potrzebnego do włączania pojedynczych części wyświetlacza LED – ANx oraz potrzebnego do wpisania zakodowanych wartości.

**4.** Realizacja licznika i zmiany segmentów.

## W pliku tb_led_display.vhd została wykonana symulacja:

```
-- Mateusz Gabryel 181329 EiT 3
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

ENTITY tb IS
END tb;

ARCHITECTURE Behavioral OF tb IS

COMPONENT top_encoder IS
    PORT ( clk_i : in STD_LOGIC;
           btn_i : in STD_LOGIC_VECTOR (3 downto 0);
           sw_i : in STD_LOGIC_VECTOR (7 downto 0);
           led7_an_o : out STD_LOGIC_VECTOR (3 downto 0);
           led7_seg_o : out STD_LOGIC_VECTOR (7 downto 0)
          );
END COMPONENT top_encoder;

signal clk_i : STD_LOGIC := '0';
signal btn_i : STD_LOGIC_VECTOR (3 downto 0) := "0000";
signal sw_i : std_logic_vector (7 downto 0) := "00000000";
signal led7_an_o : STD_LOGIC_VECTOR (3 downto 0) :="1111";
signal led7_seg_o : STD_LOGIC_VECTOR (7 downto 0) :="11111111";
constant PERIOD : TIME := 10 ns; -- 100 MHz

BEGIN

uut: top_encoder
    PORT MAP ( clk_i => clk_i,
               btn_i => btn_i,
               sw_i => sw_i,
               led7_an_o => led7_an_o,
               led7_seg_o => led7_seg_o
              );

clk_i <= NOT clk_i AFTER PERIOD/2; -- realizacja prostego zegara

tb:PROCESS
   BEGIN
        WAIT FOR 4 ms; -- AN3
          btn_i(3) <= '1';
          btn_i(2) <= '0';
          btn_i(1) <= '0';
          btn_i(0) <= '0';
          sw_i(0) <= '1';
          sw_i(1) <= '1';
          sw_i(2) <= '1';
          sw_i(3) <= '1';
          -- kropki
          sw_i(4) <= '1';
          sw_i(5) <= '1';
          sw_i(6) <= '0';
          sw_i(7) <= '0';
        WAIT FOR 2 ms; -- AN2
          btn_i(3) <= '0';
          btn_i(2) <= '1';
          btn_i(1) <= '0';
          btn_i(0) <= '0';
          sw_i(0) <= '0';
          sw_i(1) <= '1';
          sw_i(2) <= '0';
          sw_i(3) <= '1';
          -- kropki
          sw_i(4) <= '1';
          sw_i(5) <= '0';
          sw_i(6) <= '0';
          sw_i(7) <= '1';
        WAIT FOR 2 ms; -- AN1
          btn_i(3) <= '0';
          btn_i(2) <= '0';
          btn_i(1) <= '1';
          btn_i(0) <= '0';
          sw_i(0) <= '0';
          sw_i(1) <= '0';
          sw_i(2) <= '0';
          sw_i(3) <= '0';
          -- kropki
          sw_i(4) <= '1';
          sw_i(5) <= '0';
          sw_i(6) <= '1';
          sw_i(7) <= '0';
        WAIT FOR 2 ms; -- AN0
          btn_i(3) <= '0';
          btn_i(2) <= '0';
          btn_i(1) <= '0';
          btn_i(0) <= '1';
          sw_i(0) <= '0';
          sw_i(1) <= '1';
          sw_i(2) <= '0';
          sw_i(3) <= '0';
          -- kropki
          sw_i(4) <= '0';
          sw_i(5) <= '1';
          sw_i(6) <= '0';
          sw_i(7) <= '1';
        WAIT;
END PROCESS;
END Behavioral;
```

## Wyniki symulacji: