Sprawozdanie

Inżynieria Układów Programowalnych

Nazwa ćwiczenia	Prosty stoper
Numer ćwiczenia	7
Imię	Mateusz
Nazwisko	Gabryel
Numer indeksu	181329
Kierunek	Elektronika i Telekomunikacja
Numer grupy dziekańskiej	3

Opis działania zadania:

Układ po zaprogramowaniu, umożliwia pomiar czasu z dokładnością do setnych części sekundy. W układzie został zastosowany blok sterowania wyświetlaczem z zadania numer 6 oraz moduł dzielnika częstotliwości z zadania numer 5. Układ posiada dwa przyciski:

- BTN0 – kolejne jego naciśnięcia wywołują: START, STOP RESET
- BTN 3 – reset asynchroniczny

Czas na wyświetlaczu LED jest wyświetlany w postaci: SS.DD, a przekroczenie czasu 59.99 sekund jest sygnalizowane wyświetleniem specjalnego symbolu oznaczającego przepełnienie „--.--„. Układ został zabezpieczony przed drganiami przycisku BTN0.

Numer indeksu studenta: 181329

Dwie ostatnie cyfry: 29

Podczas wykonanej symulacji, zatrzymanie stopera zostało wykonane według instrukcji, czyli po czasie 1,29 sekund.

Opis pliku top_stoper.vhd:

1.

2.

```
-- Mateusz Gabryel 181329 EiT 3
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.NUMERIC_STD.ALL;
use STD.STANDARD.ALL;


ENTITY top_stoper IS
    PORT ( clk_i : in STD_LOGIC;
           rst_i : in STD_LOGIC;
           start_stop_button_i : in STD_LOGIC;
           led7_an_o : out STD_LOGIC_VECTOR (3 downto 0);
           led7_seg_o : out STD_LOGIC_VECTOR (7 downto 0)
          );
END top_stoper;

ARCHITECTURE Behavioral OF top_stoper IS

COMPONENT top_divider is
    PORT ( clk_i: in STD_LOGIC;
           rst_i: in STD_LOGIC;
           clk_divider: out STD_LOGIC
          );
END COMPONENT;

COMPONENT top_display IS
    PORT ( led7_seg_o: out STD_LOGIC_VECTOR(7 downto 0);
           led7_an_o: out STD_LOGIC_VECTOR(3 downto 0);
           digit_i: in STD_LOGIC_VECTOR(31 downto 0);
           display_clk: in STD_LOGIC;
           rst_i: in STD_LOGIC
          );
END COMPONENT;

COMPONENT top_part IS
    PORT ( part: in STD_LOGIC_VECTOR(3 downto 0);
           digit_i: out STD_LOGIC_VECTOR(6 downto 0)
          );
END COMPONENT;

signal digit_i: STD_LOGIC_VECTOR (31 downto 0):= "11000000110000001100000011000000";
signal clk_o: STD_LOGIC;
  type stan_lista IS (START, STOP, RESET);
  signal stan, n_stan: stan_lista := RESET;
  signal ms_AN0: STD_LOGIC_VECTOR (3 downto 0):= "0000"; -- jednosci ms
  signal ms_AN1: STD_LOGIC_VECTOR (3 downto 0):= "0000"; -- dziesiatki ms
  signal s_AN2: STD_LOGIC_VECTOR (3 downto 0):= "0000";  -- jednosci s
  signal s_AN3: STD_LOGIC_VECTOR (3 downto 0):= "0000";  -- dziesiatki s
  signal counter: INTEGER := 0; -- licznik
  signal rst_o: STD_LOGIC := '1';
  signal rst_n: STD_LOGIC :='0';
  signal s_sync, s_sync_old: STD_LOGIC := '0';
  signal s_stable: STD_LOGIC := '0';

  BEGIN

  divider: top_divider
      PORT MAP ( clk_i => clk_i,
                 rst_i => '0',
                 clk_divider => clk_o
                );

  display: top_display
      PORT MAP ( led7_seg_o => led7_seg_o,
                 led7_an_o => led7_an_o,
                 digit_i => digit_i,
                 display_clk => clk_o,
                 rst_i => rst_i
                );

  AN0: top_part
      PORT MAP ( part =>ms_AN0,
                 digit_i => digit_i(7 downto 1)
                );

  AN1: top_part
      PORT MAP ( part => ms_AN1,
                 digit_i => digit_i(15 downto 9)
                );
```

```
AN2: top_part
    PORT MAP ( part =>s_AN2,
               digit_i =>digit_i(23 downto 17)
              );

AN3: top_part
    PORT MAP ( part =>s_AN3,
               digit_i =>digit_i(31 downto 25)
              );

PROCESS(clk_i)
variable opoznienie: INTEGER;
BEGIN

    if(rising_edge(clk_i)) then
        stan <= n_stan;
        s_sync <= start_stop_button_i;
        s_sync_old <= s_stable;

        if (s_sync = s_stable) then
            opoznienie :=0;
        else
            opoznienie := opoznienie + 1;
        END if;
        if (opoznienie = 1000000) then
            s_stable <= s_sync;
            opoznienie := 0;
        END if;

        if(stan = START) then
            if(s_AN3 = "1111" and s_AN2 = "1111" and ms_AN1 = "1111" and ms_AN0 = "1111") then -- start/reset po przepelnieniu
                counter <=0;
            else
                counter <= counter +1;
            END if;

            if(counter = 1000000) then
                counter <=0;
                ms_AN0 <= ms_AN0 + 1;
            END if;

              if(ms_AN0 = "1010") then -- 9 na AN0 - czyli na jednosci ms
                  ms_AN0 <= "0000";
                  ms_AN1 <= ms_AN1 + 1;
              END if;

              if(ms_AN1 = "1010") then -- 9 na AN1 - czyli na dziesiatki ms
                  ms_AN1 <= "0000";
                  s_AN2 <= s_AN2 + 1;
              END if;

              if(s_AN2 = "1010") then -- 9 na AN2 - czyli na jednosci s
                  s_AN2 <= "0000";
                  s_AN3 <= s_AN3 + 1;
              elsif(s_AN3 = "0101" and s_AN2 = "1001" and ms_AN1 = "1001" and ms_AN0 = "1001") then -- przepelnienie
                  s_AN3 <= "1111";
                  s_AN2 <= "1111";
                  ms_AN1 <= "1111";
                  ms_AN0 <= "1111";
                  counter <= 0;
              END if;

          elsif(stan = STOP) then -- STOP, zatrzymanie
              counter <=0;
          elsif(stan = RESET) then -- RESET, wyzerowanie
              counter <=0;
              ms_AN0 <= "0000";
              ms_AN1 <= "0000";
              s_AN2 <= "0000";
              s_AN3 <= "0000";
          END if;
      END if;
  END PROCESS;

  PROCESS(rst_i, stan, s_stable, s_sync_old, rst_o, rst_n) -- dzialanie przycisku po nacisnieciu
  BEGIN
      if (rst_i = '1') then
          rst_o <= '0';
          rst_n <= '1';
      elsif (rst_i = '0') then
          rst_o <= '1';
          rst_n <= '0';
      END if;

       if (s_stable = '1' and s_sync_old = '0') then
           CASE stan IS
               WHEN START => n_stan <= STOP;
               WHEN STOP => n_stan <= RESET;
               WHEN RESET => n_stan <= START;
           END CASE;
       elsif (rst_o = '0' and rst_n = '1') then
           n_stan <= RESET;
       else
           n_stan <= stan;
       END if;
  END PROCESS;

  digit_i(0) <= '1';  -- wylaczenie kropki dla AN0
  digit_i(8) <= '1';  -- wylaczenie kropki dla AN1
  digit_i(16) <= '0'; -- wlaczenie kropki dla AN2
  digit_i(24) <= '1'; -- wylaczenie kropki dla AN3

  END Behavioral;
```

3.

4.

5.

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

2. Deklaracja przydziału portów:

- clk_i: zegar wejściowy
- rst_i: reset asynchroniczny
- start_stop_button_i: przyciski BTN0
- led7_an_o: pojedyncza część wyświetlacza LED
- led7_seg_o: segmenty pojedynczego wyświetlacza

3. Zabezpieczenie przed drganiami.

4. Liczenie oraz przejście na następne segmenty wyświetlacza, realizacja przepełnienia.

5. Obsługa wciśnięć przycisków oraz włączenie i wyłączenie kropek dziesiętnych.

Do prawidłowego działania układu na płytce FPGA jest potrzebny plik constraint iup7.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 Buttons
set_property -dict { PACKAGE_PIN C17   IOSTANDARD LVCMOS33 } [get_ports { start_stop_button_i }];  # virtual BTN0
set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { rst_i }];                # virtual BTN3
#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_part.vhd:

1.

2.

3.

```
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_part IS
10     PORT ( part : in STD_LOGIC_VECTOR (3 downto 0);
11            digit_i : out STD_LOGIC_VECTOR (7 downto 1):="1111111");
12 END top_part;
13
14 ARCHITECTURE Behavioral OF top_part IS
15
16 BEGIN
17
18 WITH part SELECT
19     digit_i <= "0000001" WHEN "0000",   -- 0
20                "1001111" WHEN "0001",   -- 1
21                "0010010" WHEN "0010",   -- 2
22                "0000110" WHEN "0011",   -- 3
23                "1001100" WHEN "0100",   -- 4
24                "0100100" WHEN "0101",   -- 5
25                "0100000" WHEN "0110",   -- 6
26                "0001111" WHEN "0111",   -- 7
27                "0000000" WHEN "1000",   -- 8
28                "0000100" WHEN "1001",   -- 9
29                "1111110" WHEN OTHERS;   -- znak przepelnienia
30 END Behavioral;
```

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

2. Deklaracja przydziału portów:

- part: segmenty pojedynczego wyświetlacza
- digit_i: wartość bitowa

3. Wyświetlanie poszczególnych symboli wyświetlanych na wyświetlaczu (0-9) oraz znak przepełnienia (-).

W pliku tb.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_stoper IS
    PORT ( clk_i : in STD_LOGIC;
           rst_i : in STD_LOGIC;
           start_stop_button_i : in STD_LOGIC;
           led7_an_o : out STD_LOGIC_VECTOR (3 downto 0);
           led7_seg_o : out STD_LOGIC_VECTOR (7 downto 0)
          );
END COMPONENT top_stoper;

COMPONENT bounce IS
   GENERIC ( min_time : TIME := 100 us;
             max_time : TIME := 1 ms;
             max_cnt : INTEGER := 2;
             seed : INTEGER := 777);
    PORT ( in_i : in STD_LOGIC;
           out_o : out STD_LOGIC);
END COMPONENT bounce;

signal clk_i: STD_LOGIC := '0';
signal btn_o: STD_LOGIC:= '0';
signal btn_o_bounce: STD_LOGIC:= '0';
signal rst_i: STD_LOGIC := '0';
signal led7_seg_o : STD_LOGIC_VECTOR (7 downto 0);
signal led7_an_o : STD_LOGIC_VECTOR (3 downto 0);
constant PERIOD : TIME := 10 ns; -- 100 MHz

BEGIN

uut: top_stoper
    PORT MAP ( clk_i => clk_i,
               rst_i =>  rst_i,
              start_stop_button_i => btn_o_bounce,
              led7_seg_o => led7_seg_o,
              led7_an_o => led7_an_o
             );

uut_2: bounce
    PORT MAP ( in_i => btn_o,
               out_o => btn_o_bounce
             );

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

tb:PROCESS
   BEGIN
        WAIT FOR 10 ms;
            btn_o <= '1'; -- start
        WAIT FOR 30 ms;
            btn_o <= '0';
        WAIT FOR  1290 ms; -- wcisniecie startu po 1,29 s - czyli zatrzymanie
            btn_o <= '1';
        WAIT FOR 50 ms;
            btn_o <= '0';
        WAIT FOR 150 ms;
            btn_o <= '1'; -- start
        WAIT FOR 30 ms;
            btn_o <= '0';
        WAIT FOR 50 ms;
            btn_o <= '1'; -- zatrzymanie
        WAIT FOR 30 ms;
            btn_o <= '0';
        WAIT FOR 50 ms;
            rst_i <= '1'; -- reset asynchroniczny
        WAIT FOR 5 ms;
            rst_i <= '0';
        WAIT FOR 15 ms;
     WAIT;
END PROCESS;
END Behavioral;
```

Symulacja została zabezpieczona przed drganiami umieszczonym na stronie plikiem bounce.vhd:

```
----------------------------------------------------------------------------------
--
-- Modelowanie drgań zestyków (do symulacji)
-- Wersja: 0.1
--
-- in_i - wejściowy "czysty" sygnał
-- out_o - sygnał z dodanymi drganiami zestyków
--
-- min_time - minimalny czas pomiędzy drganiami zestyków
-- max_time - maksymalny czas pomiędzy drganiami zestyków
-- max_cnt - maksymalna liczba drgań zestyków (minimalna wynosi zawsze 0)
-- seed - ziarno dla generatora liczb pseudolosowych
--
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.MATH_REAL.ALL;

entity bounce is
    Generic ( min_time : TIME := 100 us;
              max_time : TIME := 1 ms;
              max_cnt : INTEGER := 2;
              seed : INTEGER := 777);
    Port ( in_i : in STD_LOGIC;
           out_o : out STD_LOGIC);
end bounce;

architecture Behavioral of bounce is

begin

process
variable seed1, seed2 : integer := seed;
variable rnd, rnd_scaled, min_real, max_real : real;
variable rnd_cnt : integer;
begin
  min_real := real(min_time / 1 ns);
  max_real := real(max_time / 1 ns);
  out_o <= in_i;
  loop
    wait on in_i;
    uniform(seed1, seed2, rnd);
    rnd_cnt := integer(floor(rnd * real(max_cnt + 1)));
    for i in 0 to rnd_cnt * 2 loop
      uniform(seed1, seed2, rnd);
      rnd_scaled := rnd * (max_real - min_real) + min_real;
      wait for rnd_scaled * 1 ns;
      if i mod 2 = 0 then
        out_o <= in_i;
      else
        out_o <= not in_i;
      end if;
    end loop;
  end loop;
end process;

end Behavioral;
```