Programowalne układy cyfrowe **2014/2015**

Prowadzący: mgr. Tomaszewski Grzegorz poniedziałek, 14:00

Data oddania: _____ Ocena: _____

Witold Olechowski 127517
Tomasz Marecik 127374

Zadanie : Projekt licznika rewersyjnego.

1. Cel ćwiczenia:

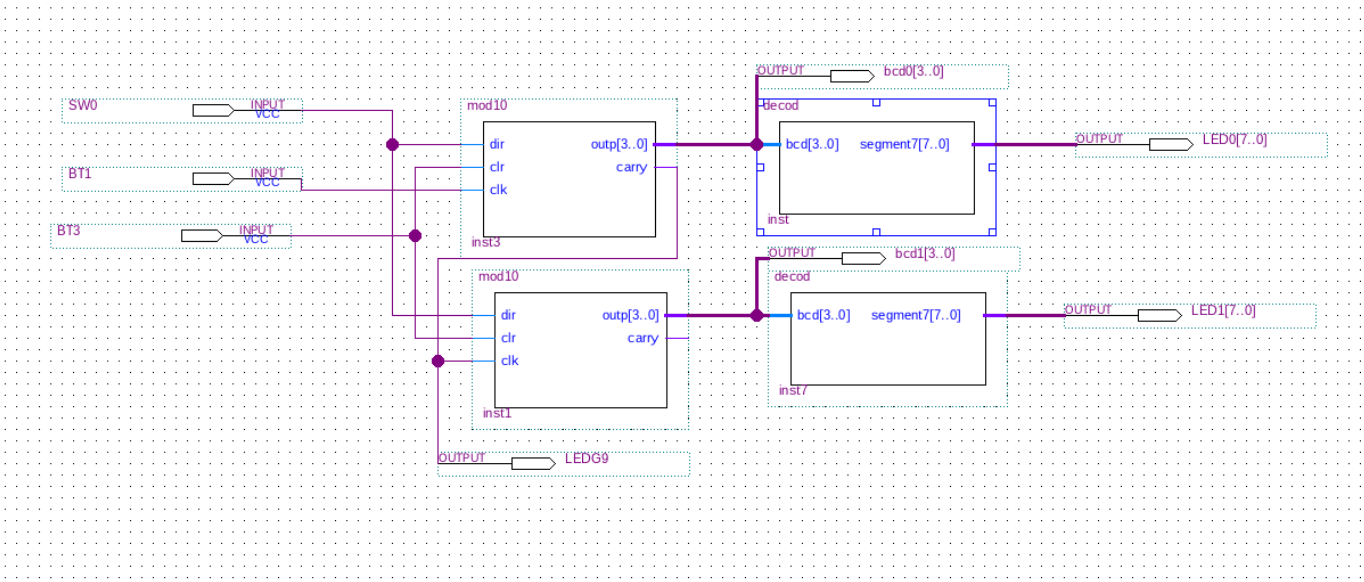
Używając języka VHDL zrealizowad projekt licznika rewersyjnego dla dwóch możliwych wariantów:

- stan licznika wyświetlany na dwóch wyświetlaczach siedmiosegmentowych, zliczanie układu w górę oraz w dół sterowane za pomocą przełącznika suwakowego SW0, przycisk BUTTON1 źródłem zliczania impulsów

- stan licznika wyświetlany na dwóch wyświetlaczach siedmiosegmentowych, zliczanie układu w górę po wciśnięciu przycisku BUTTON1, natomiast zliczanie układu w dół po wciśnięciu przycisku BUTTON0

2. Pierwszy wariant realizacji zadania

Realizacja pierwszego wariantu polegała na połączeniu dwóch liczników modulo 10 do zliczania impulsów jedności i dziesiątek, odpowiednie przypisanie wejśd liczników skutkuje zwiększaniem się liczb na wyświetlaczu po ustawieniu przełącznika suwakowego w pozycji górnej, natomiast zmniejszaniu się wartości po ustawieniu przełącznika SW0 w pozycji dolnej. Zliczanie impulsów odbywa się poprzez wciskanie przycisku BUTTON1.



Rysunek 1: Schemat blokowy licznika dla pierwszego wariantu.

Na listingu 1 zamieszczona została, implementacja w jezyku VHDL rewersyjnego licznika modulo 10.

Listing 1 Rewersyjny licznik modulo 10

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

entity mod10 is --porty we / wy
    Port(outp : out std_logic_vector(3 downto 0);
         dir : in std_logic;
         carry: out std_logic;
         clr : in std_logic;
         clk : in std_logic);
end mod10;

architecture Behavioral of mod10 is
signal temp : std_logic_vector(3 downto 0);
begin

process(clk,clr, dir)
begin    -- wyzeruj licznik oraz pozyczke
         if clr = '1' then
                 temp<="0000";
                 carry <= '0';
         -- licz gdu narastajace zbocze na zegarze
         elsif rising_edge(clk) then
                 --jesli dir "1" liczy do gory, oraz sprawdza stan licznika
                 --po przepelnieni zeruje sie i ustawia pozyczke na "1"
                 --analogicznie dla dir "0"
                 if dir = '1' then
                         if temp <= "1000" then
                                 temp <= temp+1;
                                 carry <='0';
                         else
                                 temp <= "0000";
                                 carry <= '1';
                         end if;
                 elsif dir = '0' then
                                 if temp >= "0001" then
                                 temp <= temp - 1;
                                 carry <='0';
                         else
                                 temp <= "1001";
                                 carry <= '1';
                         end if;
                 end if;
         end if;
         outp<=temp;
end process;
end Behavioral;
```

Na listingu 2 zamieszczona została, implementacja w jezyku VHDL dekodera BCD na kod wskaznika 7-segmentowego.

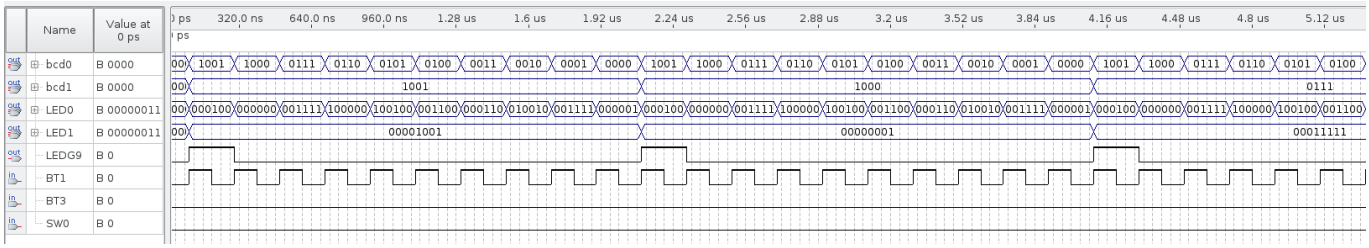
Listing 2 Dekoder bcd na 7 segment

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

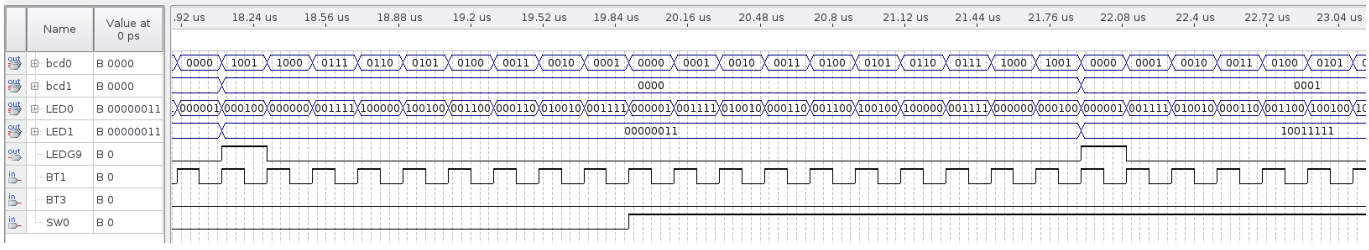
```
entity decod is
port (
        --wejscie BCD.
        bcd : in std_logic_vector(3 downto 0);
        -- wyjscie dekodera 8 bit.
        segment7 : out std_logic_vector(7 downto 0)
    );
end decod;

architecture Behavioral of decod is

begin
process (bcd)
BEGIN
case  bcd is
        when "0000"=> segment7 <="00000011";   -- '0'
        when "0001"=> segment7 <="10011111";   -- '1'
        when "0010"=> segment7 <="00100101";   -- '2'
        when "0011"=> segment7 <="00001101";   -- '3'
        when "0100"=> segment7 <="10011001";   -- '4'
        when "0101"=> segment7 <="01001001";   -- '5'
        when "0110"=> segment7 <="01000001";   -- '6'
        when "0111"=> segment7 <="00011111";   -- '7'
        when "1000"=> segment7 <="00000001";   -- '8'
        when "1001"=> segment7 <="00001001";   -- '9'
        -- stany wyzsze od 9 wygaczaja segement
        when others=> segment7 <="11111111";
end case;
end process;
end Behavioral;
```

2.1. Przebiegi czasowe układu:

Na rysunkach 2, 3, 4, 5 zostały przedstawione przebiegi czasowe potwierdzające poprawne działanie licznika rewersyjnego dla pierwszego wariantu. Przycisk BT1 generuje kolejne impulsy zliczające, natomiast przełącznik suwakowy SW1 steruje kierunkiem zliczanych impulsów (góra, dół).

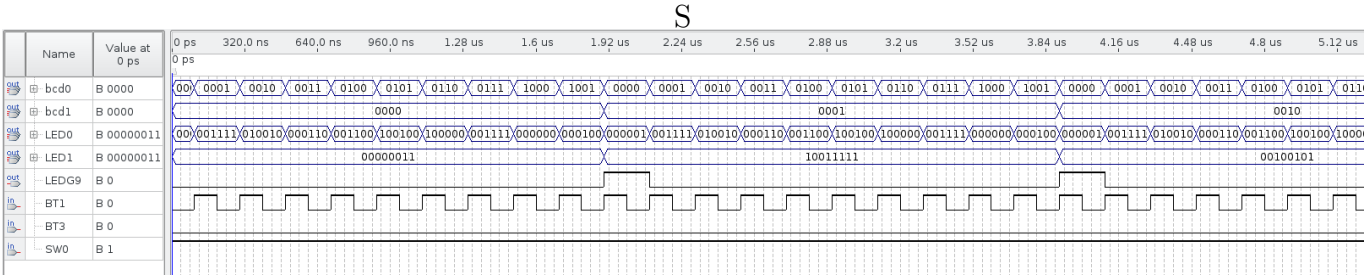


Rysunek 2: Przebiegi czasowe – zliczanie w dół (SW0=0)

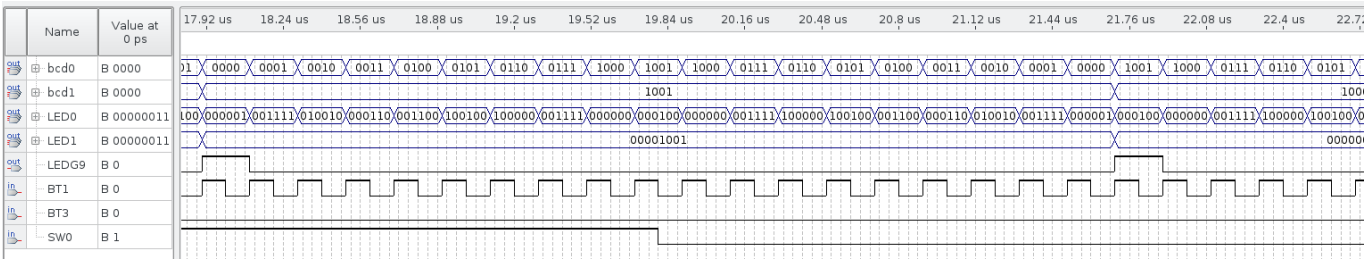


Rysunek 3: Przebiegi czasowe – zliczanie w dół (SW0=0 $\longrightarrow$ SW0=1) zmiana pozycji przełącznika suwakowego SW0 przy skrajnych wartościach

S



Rysunek 4: Przebiegi czasowe – zliczanie w górę (SW0=1)

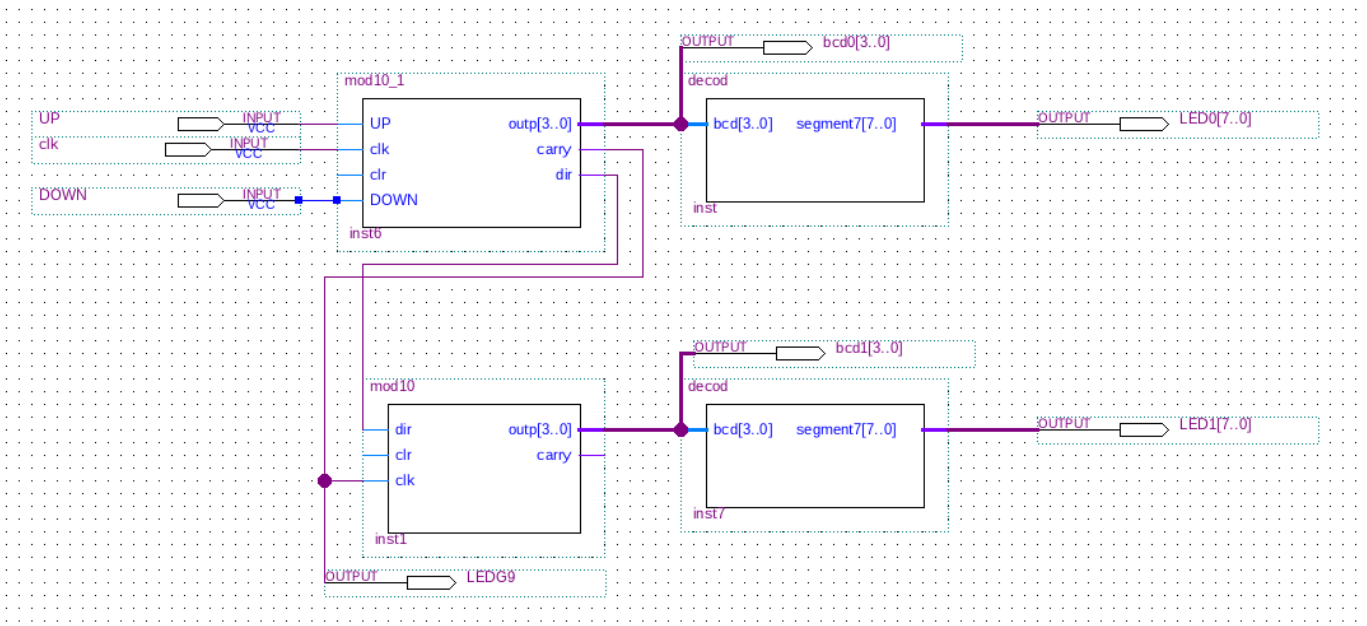


Rysunek 5: Przebiegi czasowe – zliczanie w górę (SW0=1 $\longrightarrow$ SW0=0) zmiana pozycji przełącznika suwakowego SW0 przy skrajnych wartościach

3. Drugi wariant realizacji projektu licznika

Realizacja drugiego wariantu polegała na połączeniu dwóch liczników modulo 10 do zliczania impulsów jedności i dziesiątek, odpowiednie przypisanie wejśd liczników skutkuje odpowiednio zwiększaniem się liczb na wyświetlaczu po wciśnięciu przycisku BUTTON1, natomiast zmniejszaniu się wartości po wciśnięciu przycisku BUTTON0.

Do poprawnego działania drugiego wariantu konieczna była modyfikacja schematu blokowego co widzimy na Rysunku 6. Właściwe działanie bloku mod10_1 (lst. 3) wynika z odpowiednich modyfikacji kodu w języku VHDL.

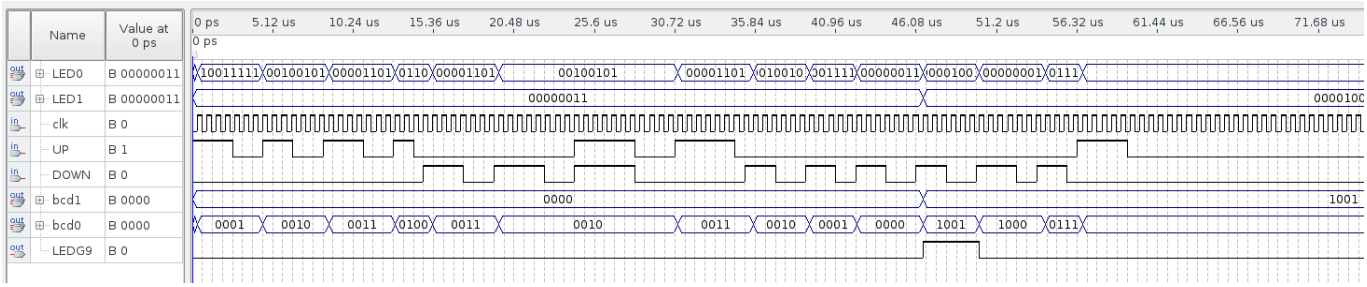


Rysunek 6: Schemat blokowy licznika dla drugiego wariantu.

Listing 3 Rewersyjny licznik modulo 10

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

entity mod10_1 is
Port (  outp : out std_logic_vector(3 downto 0);
        UP   : in std_logic;
        clk  : in std_logic;
        carry: out std_logic;
        clr  : in std_logic;
        dir  : out std_logic;
        DOWN : in std_logic);
end mod10_1;

architecture Behavioral of mod10_1 is
signal temp : std_logic_vector(3 downto 0);
-- pomocnicze zmienne zabezpieczja wy przed inkrementowaniem z kazdym
-- narastajacym zboczem zegara
signal up1, dw1    : std_logic;

begin
process(clk,clr)
begin
if clr = '1' then
        temp<="0000";
        carry <= '0';
elsif rising_edge(clk) then
        -- licz w gore jesli narastajace zbocze zegara, na sygnale UP "1"
        -- poprzednia wartosc UP "0" patrz komentarz wyzej oraz DOWN "0"
        if (UP='1' and up1='0' and DOWN='0') then
                dir <= '1';
                if temp <= "1000" then
                        temp <= temp+1;
                        carry <='0';
                else
                        temp <= "0000";
                        carry <= '1';
                end if;
        elsif (DOWN='1' and dw1='0' and UP='0') then
                dir <= '0';
                if temp >= "0001" then
                        temp <= temp - 1;
                        carry <='0';
                else
                        temp <= "1001";
                        carry <= '1';
                end if;
        end if;
        up1 <= UP;
        dw1 <= DOWN;
end if;
outp<=temp;
end process;
end Behavioral;
```

Przebiegi czasowe dla drugiego wariantu obrazuje Rysunek 7 , naciskanie przycisków BT0 oraz BT1 odpowiada zwiększaniu lub zmniejszaniu się wartości wyświetlanych na wyświetlaczu. Przebiegi potwierdzają poprawne działanie licznika dla drugiego wariantu realizowanego projektu.



Rysunek 7: Przebiegi czasowe dla drugiego wariantu – ciąg impulsów BT0 oraz BT1 i widoczne wartości na wyświetlaczach

4. Wnioski

Celem wykonywanego dwiczenia było zapoznanie się z zasadą działania licznika, oraz jego Celem wykonywanego dwiczenia było zapoznanie się z zasadą działania licznika, oraz jego budową koniecznymi do realizacji dwóch wariantów licznika rewersyjnego w oprogramowaniu Quartus II. Przy wykorzystaniu języka VHDL utworzono nowy bloczek w postaci licznika modulo 10, odpowiednie połączenie dwóch takich elementów współpracującymi z dekoderami z poprzedniego projektu w efekcie umożliwiało zaobserwowanie poprawnego działania licznika rewersyjnego.

Odpowiednio zrealizowany schemat blokowy wraz z funkcjonalnym kodem VHDL umożliwia poprawne działanie pierwszego wariantu. Utworzony projekt poddawaliśmy testom na poprawnośd działania, przeprowadzone symulacje są potwierdzeniem założeo projektowych. Drugi wariant projektu jest odpowiednią modyfikacją schematu blokowego pierwszej opcji, konieczne było odpowiednie zaimplementowanie przycisków BT0 oraz BT1 do generowania impulsów zliczających w górę o raz w dół. Kilka zmian w listingu bloczku modulo 10 zapewniło finalny efekt w postaci poprawnie działającego układu zliczającego licznika.

Podobnie jak w pierwszym przypadku po zaprogramowaniu układu poddawaliśmy go testom na poprawnośd. Jednym z pojawiających się problemów przy realizacji projektu było nie poprawne zliczanie impulsów w przypadku zmiany kierunku zliczania dla skrajnych wartości na wyświetlaczu (np. dla wartości 9, 19, 10, 20 itp.). Pojawiający problem u dało się jednak wyeliminowad poprzez odpowiednią modyfikację kodu VHDL bloczku mod_10.

Literatura

[1] John Wiley and Sons Publishers. *Digital Design,* University of California, Riverside, 2007