PRURE- dokumentacja projektu „Moduł sterownika świecenia diodami LED”

Autorzy:

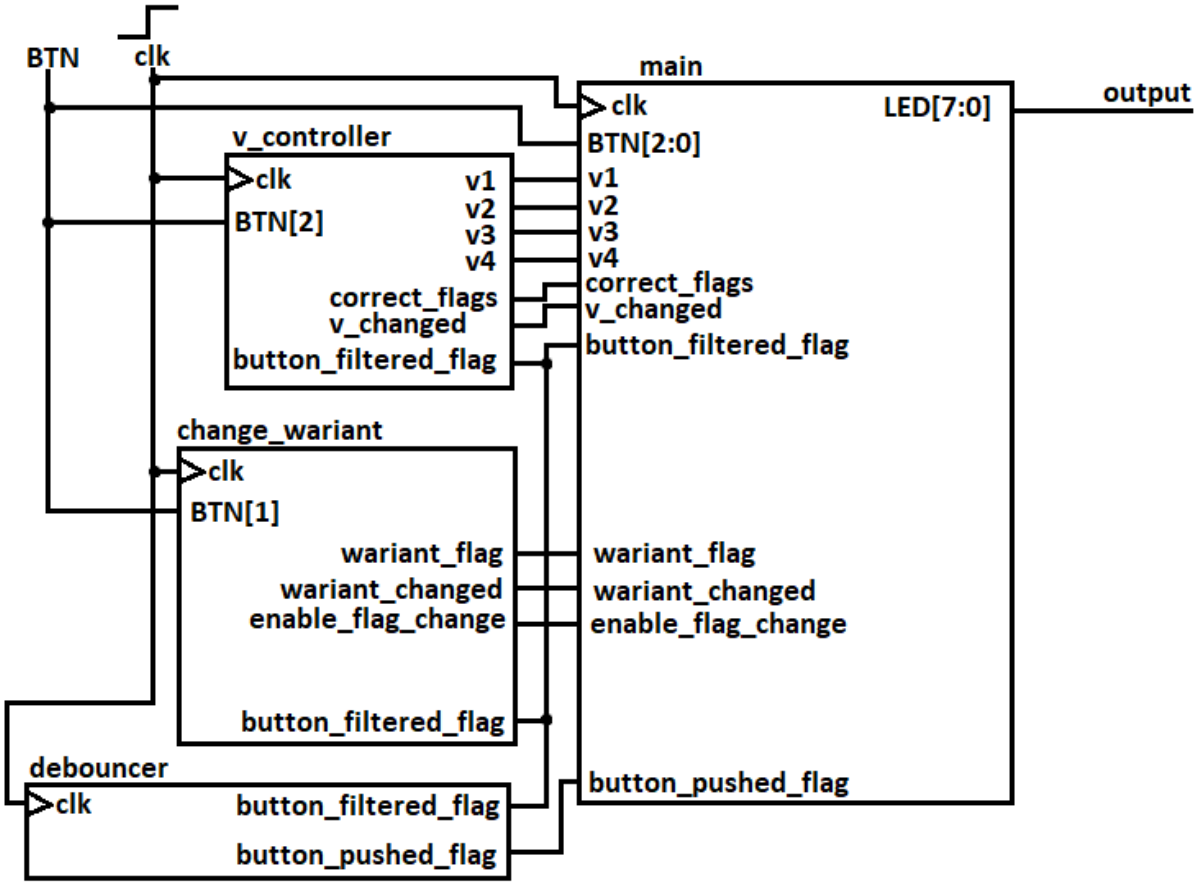
Karol Bogumił

Patryk Zemła

Realizowane schematy świetlne:

Wariant 6: Schemat diod włączonych 1-8;2-7;3-6;4-5 a następnie w odwrotnej kolejności
Wariant 7: Włączające się kolejno diody (maksymalnie 8 naraz włączonych), a następnie kolejno wyłączające się

Schemat blokowy układu:



Opis działania projektu:

Działanie projektu opiera się na ciągłym zapalaniu i gaszeniu diod według konkretnego schematu świetlnego. Wciskając przycisk BTN0, wszystkie diody gasną. Wciskając ponownie BTN0, diody budzą się do życia i ponownie wyświetlają odpowiedni wariant. BTN1 służy do przełączania się pomiędzy wariantami. Po jego naciśnięciu, diody będą świecić innym schematem. BTN2 zmienia prędkość. Są 4 dostępne prędkości: podstawowa prędkość to przełączanie się z częstotliwością 1Hz. Wciśnięcie przycisku spowoduje dwukrotne zwiększenie częstotliwości, kolejne wciśnięcie ponowne dwukrotne przyspieszenie, kolejne tak samo, a jeszcze kolejne spowoduje powrót do prędkości podstawowej.

Nasz projekt zdecydowaliśmy się podzielić na 4 moduły.

Moduł „main” (nazwa wynika z tego, że w nim realizowane są fizyczne wejścia i wyjścia projektu oraz instancje reszty modułów)- w nim opisana jest logika całego projektu. Oprócz pogodzenia ze sobą sygnałów i flag od reszty modułów, w nim realizowany jest przycisk BTN0 (włącz/ wyłącz).

Moduł change_wariant- ten moduł odpowiada za zmianę flagi wskazującej, który wariant powinien być w danej chwili wyświetlany. Po wciśnięciu BTN1, flaga wariant_flag zmienia swoją wartość. Zmiana wartości wariant_flag, uruchamia flagę wariant_changed, która mówi układowi o konieczności zmiany schematu.

Moduł v_controller- jest to kontroler 4 flag, które wskazują prędkości przełączania. Flagi zmieniają swoje wartości po naciśnięciu przycisku BTN2. W danej chwili tylko jedna flaga może mieć wartość „1”. Stan wysoki odpowiedniej flagi decyduje o okresie zliczania głównego licznika w projekcie, a zatem determinuje to szybkość jego przeładowania.

Moduł debouncer- jest to układ, który eliminuje drgania styków przycisków. Jego działanie polega na ładowaniu się 4 rejestrów po kolei, co okres licznika ustawionego na odpowiednią wartość. Jeśli każdy z tych sygnałów ma stan wysoki, to oznacza, że drgania zostały wyeliminowane i można wykonać akcję związaną z danym przyciskiem.

Omówienie najważniejszych elementów kodu:

Główna część projektu:

Generic’i, wejścia/wyjścia:

```
entity main is
generic(parameter: natural:=12; ---ustawienie predkosci --125_000_000
v1param: natural:=8; -- 93_749_999
v2param:natural:=5; -- 62_499_999
v3param:natural:=2; -- 31_249_999
v4param:natural:=11);--124_999_999
  Port ( clk:in bit; --to bedzie zegar
    jd: out bit_vector(7 downto 0);
    btn: in bit_vector(2 downto 0)); --btn(0)- wĹ‚Ä…cznik; btn(1)- zmiana schematu; btn(2)- predkosc );
end main;
```

Lista sygnałów pomocniczych:

```
architecture Behavioral of main is
signal Time_cnt: natural:=0;
signal M: natural:=0; --zmienna do wariantu 6
signal N: natural:=7; --zmienna do wairantu 6
signal x: natural:=0; --zmienna do wariantu 7
--do przycisku on/off
signal power_flag: bit:='0'; --kiedy 0 to uklad wlaczony
-- do zmiany wariantu
signal wariant_flag: bit:='0'; --kiedy 0 to wariant 6 a kiedy 1 to wariant 7
signal wariant_changed: bit:='0'; --flaga sygnalizujaca ze zmienono stan
--do kontrolera predkosci
signal v1: bit:='0'; --predkosc 1
signal v2: bit:='0'; --predkosc 2
signal v3: bit:='0'; --predkosc 3
signal v4: bit:='1'; --predkosc 4- domyslnie 1s
signal period: natural:=parameter;
signal correct_flags: bit:='0';
signal ready_flag: bit:='0';
signal v_changed: bit:='0';
--do debouncera:
signal button_pushed_flag: bit:='0';
signal button_filtered_flag: bit:='0';
-- szybkie zmiany flag:
signal enable_flag_change: bit:='0'; --1- enable, 0- disable
```

Realizacja przycisku włącz/ wyłącz:

```
--przycisk wlaczcz/wylacz:
if(btn(0)='1' and power_flag='0' and button_filtered_flag='1') then --czyli dzialal ale nacisnieto btn(0)
    Time_cnt<=0; --wyzerowanie countera
    M<=0;
    N<=7;
    x<=0;
    if(enable_flag_change='1') then power_flag<='1'; end if;
end if;

if(btn(0)='1' and power_flag='1' and button_filtered_flag='1') then --czyli nie dzialal ale nacisnieto btn(0)
    if(enable_flag_change='1') then power_flag<='0'; end if;
    M<=0;
    N<=7;
    x<=0;
end if;
--koniec przycisku wlaczcz/wylacz
```

Układ działa, jeśli flaga power_flag jest na stanie niskim.

Raz na okres licznika Time_cnt, wystawiana jest flaga ready_flag i wtedy dokonują się zmiany na diodach.

Zmienianie prędkości:

```
-- dostosowanie predkosci
if(v4='0' and v3='0' and v2='0' and v1='0') then correct_flags<='1'; end if;

Time_cnt<=(Time_cnt+1);-- mod 12; --glowny licznik --125_000_000
if(Time_cnt=11) then Time_cnt<=0; end if; --unikniecie modulo

if(Time_cnt=v3param and v3='1') then ready_flag<='1'; Time_cnt<=0; end if;
if(Time_cnt=v2param and v2='1') then ready_flag<='1'; Time_cnt<=0; end if;
if(Time_cnt=v1param and v1='1') then ready_flag<='1'; Time_cnt<=0;end if;
if(Time_cnt=v4param and v4='1') then ready_flag<='1'; end if;
if(v_changed='1') then Time_cnt<=0; end if;
-- koniec dostosowania predkosci

if(ready_flag='1') then --raz na sekunde gdy mod 125_000_00 i if 124_999_999
```

Sama zmiana flag v1,…, v4 dokonuje się w module v_controller.

Opis schematów świecenia:

```
if(wariant_flag='0') then
--wariant 6
M<=(M+1);-- mod 8;
if(M=7) then M<=0; end if;
N<=(N-1);-- mod 8;
if(N=0) then N<=7; end if;
jd <= ( others => '0');
jd(M)<='1';
jd(N)<='1';
--koniec wariantu 6
end if; --koniec ifa od wariant flag=0

if(wariant_flag='1') then
-- wariant 7
x<=(x+1) mod 16;
--if(x=15) then x<=0; end if; --unikniecie modulo
if(x<=7) then
jd(x)<='1';
end if;
if(x>7) then
jd(15-x)<='0';
end if;
-- koniec wariantu 7
-------------------------------------------------
end if; --koniec ifa od wariant flag=1
ready_flag<='0';
```

Instancje reszty modułów:

```
change_wariant_inst: entity work.change_wariant(Behavioral)
port map (btn=>btn(1), wariant_flag=>wariant_flag, clk=> clk, wariant_changed=> wariant_changed, button_filtered_flag=> button_filtered_flag, enable_flag_chan

v_controller_inst: entity work.v_controller(Behavioral)
port map (btn=>btn(2), clk=> clk, v1=>v1,v2=>v2, v3=>v3, v4=>v4, correct_flags=>correct_flags, v_changed=>v_changed, button_filtered_flag=> button_filtered_fl

debouncer_inst: entity work.debouncer(Behavioral)
port map (clk=>clk, button_pushed_flag=>button_pushed_flag, button_filtered_flag=>button_filtered_flag);
```

Moduł change_wariant:

```
entity change_wariant is
  Port (btn: in bit;
 wariant_flag: inout bit;
 clk: in bit;
 wariant_changed: inout bit;
 button_filtered_flag: in bit;
 enable_flag_change: in bit);
end change_wariant;

architecture Behavioral of change_wariant is
begin
process(clk) is
begin
if(clk'event and clk='1') then
if(button_filtered_flag='1') then
if(btn='1') then
if(enable_flag_change='1') then wariant_flag<= not wariant_flag; end if;
wariant_changed<='1';
else wariant_changed<='0';
end if;
end if;
end if;
end process;
end Behavioral;
```

W tym module, przycisk wywołuje zmianę flagi wskazującej na odpowiedni schemat.

Moduł debouncera:

```
architecture Behavioral of debouncer is
signal a,b,c,d: bit;
signal cnt :natural := 0;
begin
process(clk) is
    begin
    if(clk'event and clk='1') then
        cnt<=(cnt+1);-- mod 3; --tutaj wstawic dobra wartosc!
        if(cnt=count) then
            cnt<=0;
            a<=button_pushed_flag;
            b<=a;
            c<=b;
            d<=c;
        end if;
     end if;

     if (button_pushed_flag='1' and a='1' and b='1' and c='1' and d='1') then button_filtered_flag <= '1';
     else button_filtered_flag<='0';
     end if;
        end process;

end Behavioral;
```

Po buton_pushed_flag ładują się po kolei a, b, c oraz d i jeśli wszystkie mają wartość ‘1’, to dopiero wtedy układ interpretuje wciśnięcie przycisku.

Moduł v_controller:

```
architecture Behavioral of v_controller is
begin
process(clk) is
begin
if(clk'event and clk='1') then
if(correct_flags='1' and v4='0' and v3='0' and v2='0' and v1='0') then v4<='1'; v3<='0'; v2<='0'; v1<='0'; end if;
if(button_filtered_flag='1') then
if(btn='1') then
if(v_changed='0') then
if(v4='1') then v4<='0'; v3<='0'; v2<='0'; v1<='1'; end if;
if(v1='1') then v1<='0'; v3<='0'; v1<='0'; v2<='1'; end if;
if(v2='1') then v2<='0'; v1<='0'; v4<='0'; v3<='1'; end if;
if(v3='1') then v3<='0'; v1<='0'; v2<='0'; v4<='1'; end if;
v_changed<='1';
else end if;
else v_changed<='0';
end if;
end if;
end if;
end process;
end Behavioral;
```

Flagi od v1 do v4 oznaczają prędkości. Correct_flags to flaga, która przyjmuje wartość 1, gdy inne flagi mają wartość 0. Jest to rodzaj zabezpieczenia.

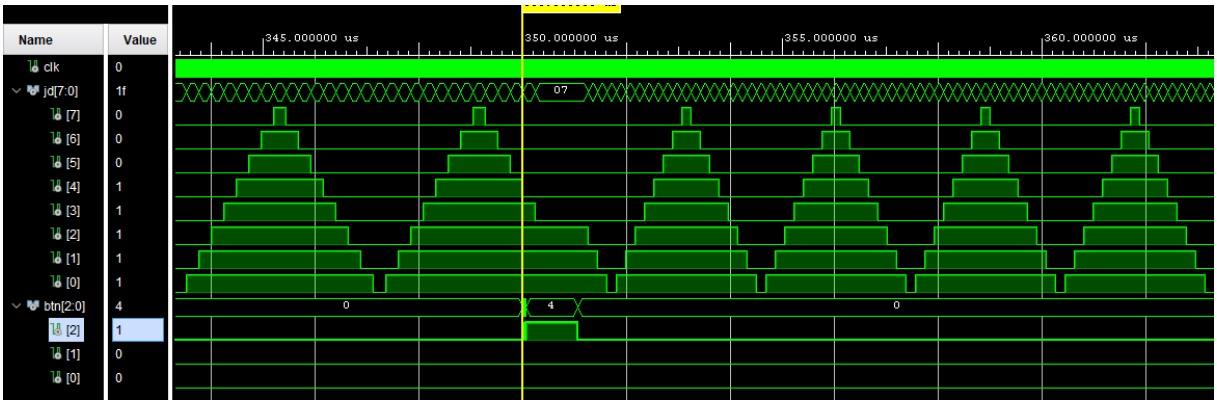
Symulacja:

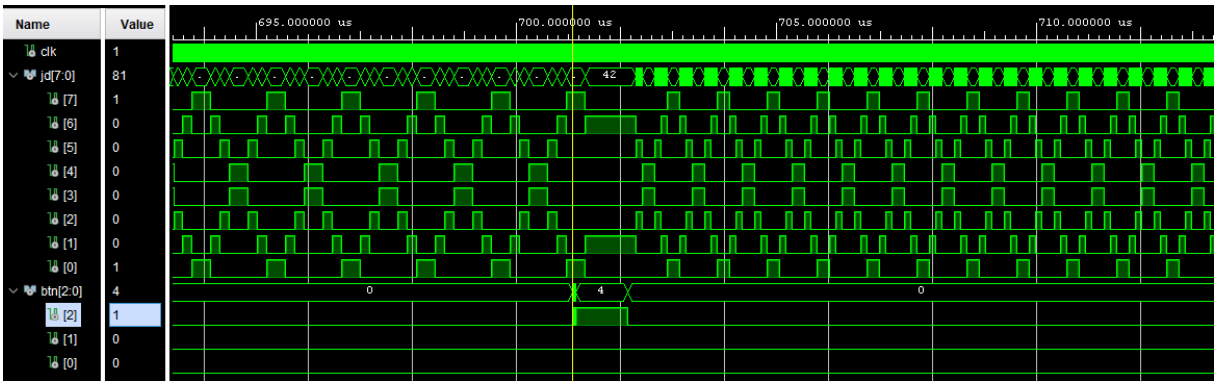
Testbench zawiera w sobie symulację debouncingu:

```
process is ---czesc z przyciskiem reset
begin
btn(0)<= '0'; wait for 1ms;
btn(0)<='1'; wait for 5ns;
btn(0)<='0'; wait for 10ns;
btn(0)<='1'; wait for 20ns;
btn(0)<='0'; wait for 10ns;
btn(0)<='1'; wait for 1us;
end process;

process is --czesc z przyciskiem zmiana wariantu
begin
btn(1)<='0'; wait for 250 us;
btn(1)<='1'; wait for 5ns;
btn(1)<='0'; wait for 10ns;
btn(1)<='1'; wait for 20ns;
btn(1)<='0'; wait for 10ns;
btn(1)<='1'; wait for 40ns;
btn(1)<='0'; wait for 20ns;
btn(1)<='1'; wait for 1us;
end process;
```

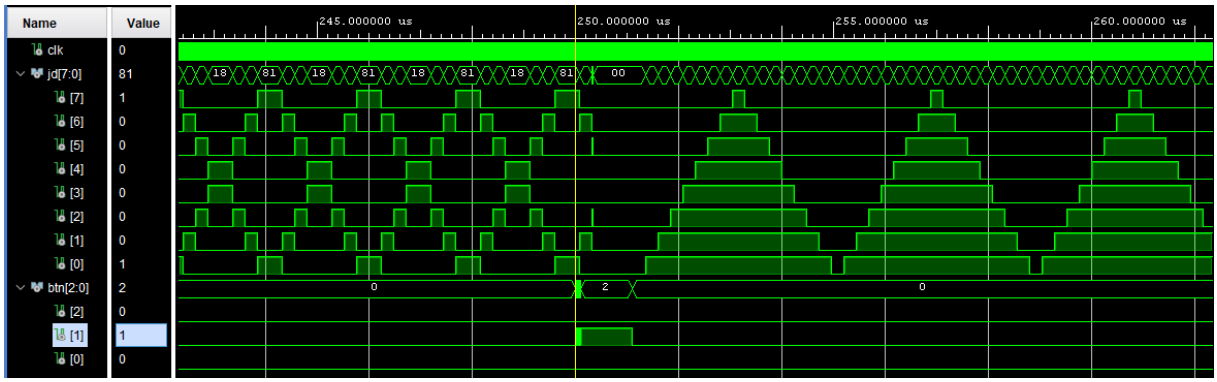
Działanie przycisku BTN2 (zmiana prędkości):





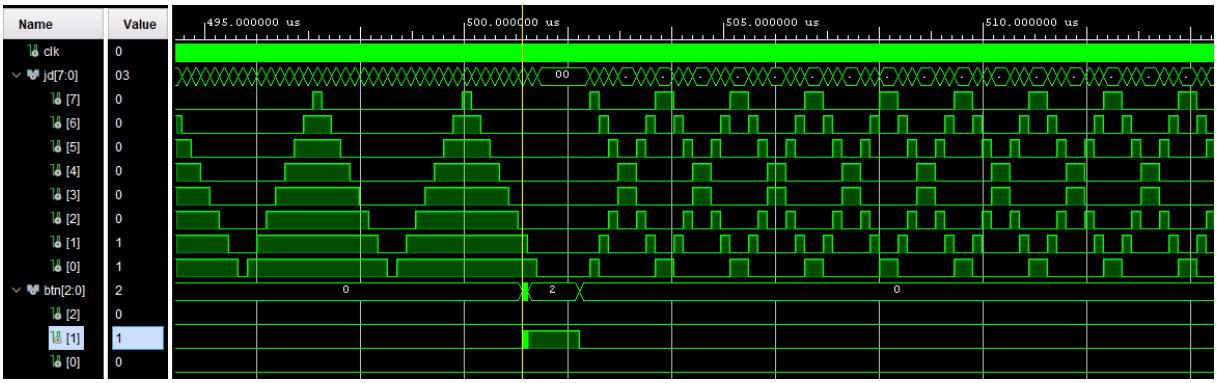
Widać, że okres schematu świecenia po narastającym zboczu BTN2 jest węższe, czyli trwa krócej, co oznacza większą prędkość.

Działanie przycisku BTN1 (zmiana schematu świecenia):

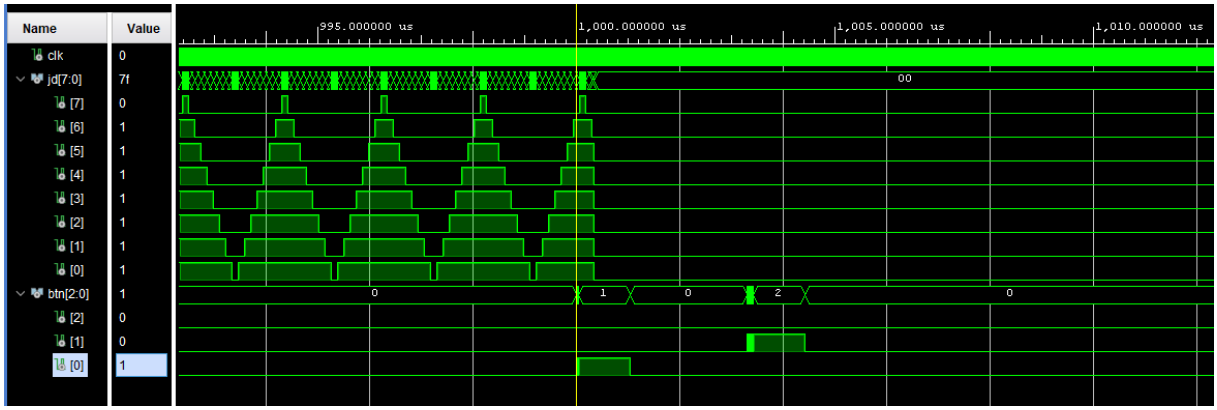


Po narastającym zboczu BTN1 schemat świetlny ulega zmianie.

Dzieje się tak również w drugą stronę:

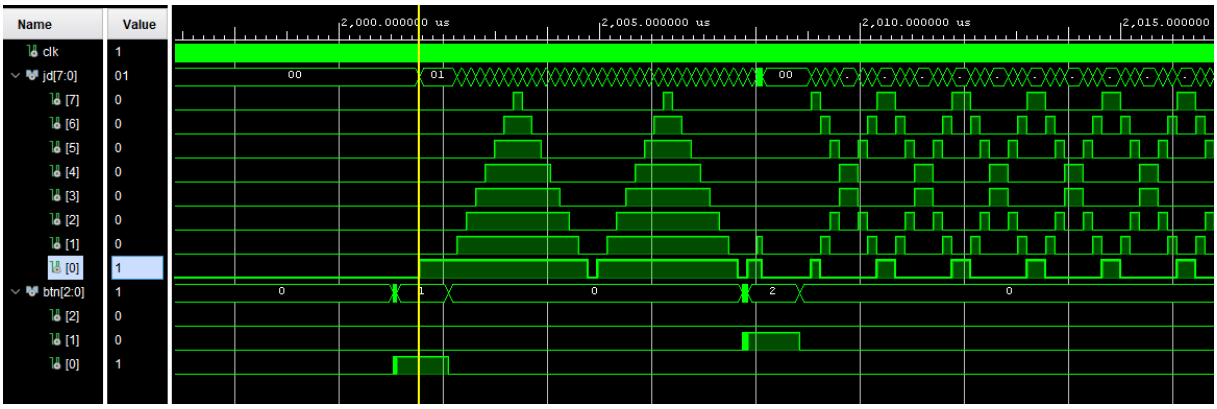


Działanie przycisku BTN0 (włącz/ wyłącz):



Wciśnięcie tego przycisku gasi diody.

Ponowne włączenie:



Udało się zrealizować projekt w całości. Symulacja odbyła się pomyślnie. Na płytce Zybo w laboratorium kod działa prawidłowo, zgodnie z oczekiwaniami.