Maciej Byczko Bartosz Matysiak	Prowadzący: dr inż. Jacek Mazurkiewicz	Numer ćwiczenia 7
PN 10:50 TP	Temat ćwiczenia: Licznik synchroniczny sterowany - FPGA	Ocena:
Grupa: B	Data wykonania: 10 Stycznia 2022r.	

Spis treści

1 Polecenie

Licznik synchroniczny rewersyjny 8-bitowy pracujący w kodzie naturalnym binarnym. Wartość inicjująca licznik ma być ładowana z klawiatury komputera PC poprzez uruchomiony na nim terminal. Można także użyć klawiatury PS/2 - uwaga na inne wartości podawane przez klawiaturę - kody skaningowe - oraz inny moduł wejściowy do obsługi portu PS/2.

Oznacza to, że do przystawki dotrze kod naciśniętego klawisza poprzez port szeregowy RS232 lub port PS/2 i ten właśnie kod ma inicjować licznik. Licznik po przyjęciu kodu zaczyna liczyć - grupa wybiera czy będzie zwiększał swój stan - będzie początkowo - pozytywny, czy też zmniejszał swój stan - będzie początkowo - negatywny.

Bieżący stan licznika ma być wyświetlany na wyświetlaczu LCD w dowolnej, ale jednoznacznej i komunikatywnej formie. W dowolnym momencie pracy licznika możemy zmieniać kierunek zliczania wybranym guzikiem z przystawki. Może to być jeden guzik - przełącznik góra/dół, mogą być użyte dwa guziki - jeden włącza zliczanie w górę, drugi - zliczanie w dół.

Realizacja zadania wymaga zatem napisania w VHDL-u własnego modułu, który będzie realizował działanie licznika oraz spięcie tego modułu z gotowymi modułami obsługi urządzeń wejścia/wyjścia przystawki: odbiornika portu RS232 lub portu PS/2 oraz obsługi wyświetlacza LCD stanowiącego integralną część przystawki Spartan FPGA.

2 Rozwiązanie

Projekt licznika się nie zmienia względem poprzednich zajęć, więc dla przypomnienia:

aby wykonać w pełni działający licznik wiemy że potrzebujemy następujące wejścia/wyjścia:

- Wejście na parametry podane z klawiatury
- Zegar na podstawie którego wywołamy kolejny stan
- Reset za pomocą którego będziemy informować licznik że chcemy wprowadzić nową wartość
- Kontrolę w jaki sposób licznik będzie liczyć (do przodu/do tyłu)

Licznik bazuje na 8 bitach więc w momencie przepełnienia ustawiliśmy że licznik wraca do wartości:

- Zerowej gdy liczy do przodu
- Maksymalnej gdy liczy do tyłu

Jedyna różnica w kodzie jest taka, że Spartan posiada zegar ze znacznie większą częstotliwością, więc na podstawie wykonania działania:

$$1Mhz = 10^6 Hz \rightarrow 50Mhz = 5 * 10^7 Hz$$

$$\log_2(5 * 10^7 Hz) \approx 25.57$$

$$\frac{5 * 10^7 Hz}{2^{25}} \approx 1.49 Hz$$

Dzięki tym obliczeniom wiemy, że musimy zastosować 25 bitowy dzielnik aby uzyskać częstotliwość w przybliżeniu $1Hz$.

2.1 Kod VHDL

```
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;

entity mainModule is
    Port ( REVERSE : in  STD_LOGIC;
           CLK_LF : in  STD_LOGIC;
           RESET : in  STD_LOGIC;
           WEJ : in  STD_LOGIC_VECTOR (7 downto 0);
           WYJ : out  STD_LOGIC_VECTOR (7 downto 0));
end mainModule;

architecture Behavioral of mainModule is

signal current_state: STD_LOGIC_VECTOR(7 downto 0);
signal splitter: STD_LOGIC_VECTOR(24 downto 0);

begin

  process1: process(CLK_LF, RESET, REVERSE, WEJ)
  begin
    -- gdy wcisniety reset, pobranie wartosci z klawiatury
    if RESET = '1' then
      current_state <= WEJ;
    elsif rising_edge(CLK_LF) then
      splitter <= splitter + 1;
      if splitter = "1000000000000000000000000" then
        splitter <= "0000000000000000000000000";
        if REVERSE = '0' then
          if current_state = "11111111" then
            current_state <= "00000000";
          else
            current_state <= current_state + 1;
          end if;
        else
          if current_state = "00000000" then
            current_state <= "11111111";
          else
            current_state <= current_state - 1;
          end if;
        end if;
      end if;
    end if;

  end process process1;

  WYJ <= current_state;
```

```
50
51 end Behavioral;
```

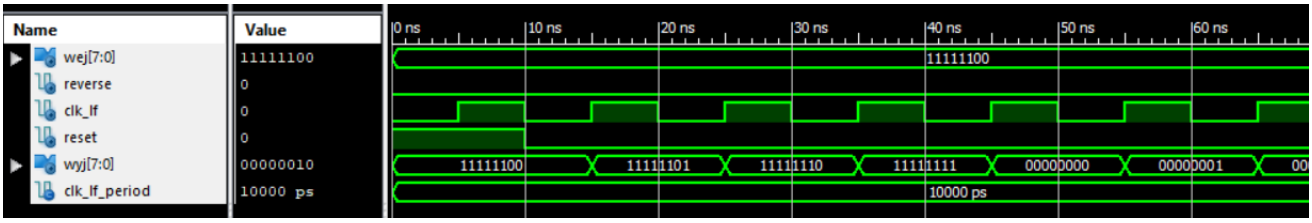
2.2 Kod VHDL TestBench

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY counter_testbench IS
5  END counter_testbench;
6
7  ARCHITECTURE behavior OF counter_testbench IS
8
9      -- Component Declaration for the Unit Under Test (UUT)
10     COMPONENT counter_mod
11     PORT(
12          WEJ : IN  std_logic_vector(7 downto 0);
13          REVERSE : IN  std_logic;
14          CLK_LF : IN  std_logic;
15          RESET : IN  std_logic;
16          WYJ : OUT  std_logic_vector(7 downto 0)
17         );
18     END COMPONENT;
19
20
21    --Inputs
22    signal WEJ : std_logic_vector(7 downto 0) := (others => '0');
23    signal REVERSE : std_logic := '0';
24    signal CLK_LF : std_logic := '0';
25    signal RESET : std_logic := '0';
26
27   --Outputs
28    signal WYJ : std_logic_vector(7 downto 0);
29
30    -- Clock period definitions
31    constant CLK_LF_period : time := 10 ns;
32
33 BEGIN
34
35   -- Instantiate the Unit Under Test (UUT)
36    uut: counter_mod PORT MAP (
37            WEJ => WEJ,
38            REVERSE => REVERSE,
39            CLK_LF => CLK_LF,
40            RESET => RESET,
41            WYJ => WYJ
42          );
43
44    -- Clock process definitions
45    CLK_LF_process :process
46    begin
47     CLK_LF <= '0';
```

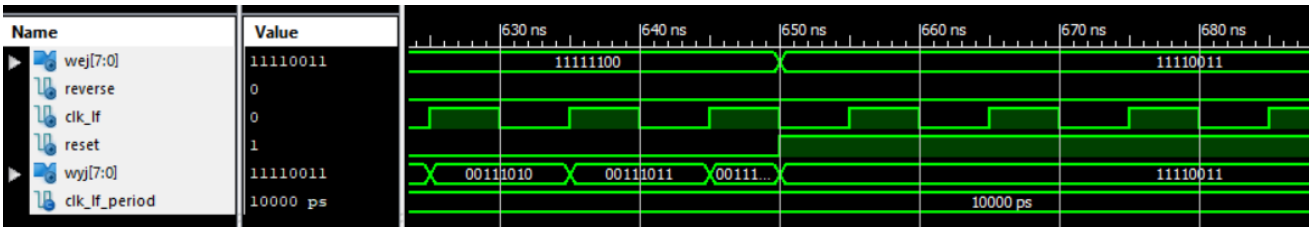
```
48      wait for CLK_LF_period/2;
49      CLK_LF <= '1';
50      wait for CLK_LF_period/2;
51    end process;
52
53    -- Stimulus process
54    stim_proc: process
55    begin
56
57        WEJ <= "11111111";
58        RESET <= '1', '0' after 10 ns;
59        REVERSE <= '0', '1' after 100 ns;
60
61        wait;
62    end process;
63
64 END;
```

2.3 Symulacja

Rysunek 1: Początek symulacji



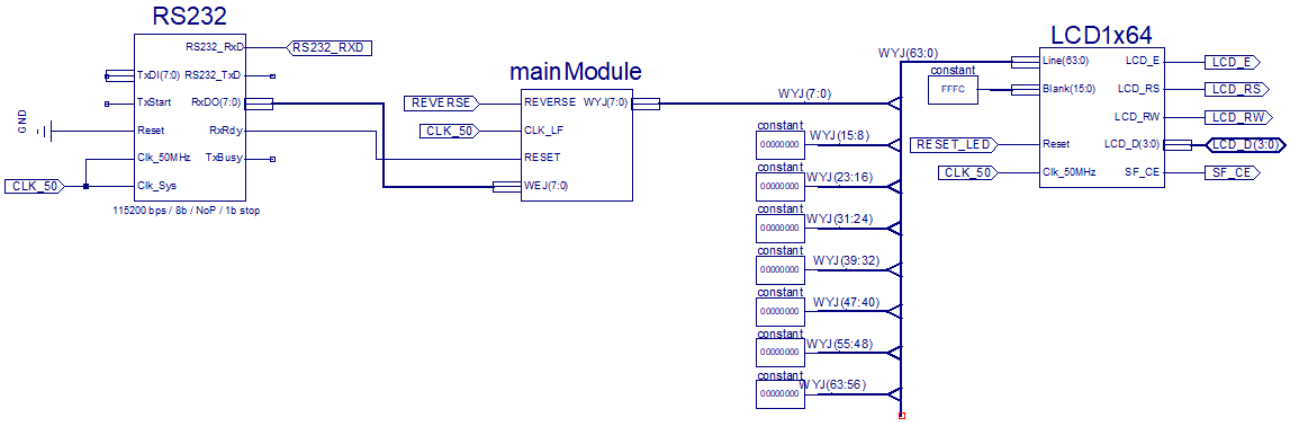
Rysunek 2: Wprowadzenie nowej wartości



Rysunek 3: Odwrócenie kolejności odliczania

2.4 Schemat układu z wykorzystaniem zaprojektowanego modułu

Rysunek 4: Schemat z podłączoną klawiaturą oraz wyświetlaczem LED



3 Fizyczna implementacja

3.1 Kod UCF

Normalnie Kod byłby w dwóch plikach: GenIO.ucf oraz LDC.ucf lecz w celu poprawienia czytelności kody zostały umieszczone w jednym bloku

```
# soldered 50MHz Clock.
NET "Clk_50" LOC = "C9" | IOSTANDARD = LVTTL;
NET "Clk_50" PERIOD = 20.0ns HIGH 50%;


# Slide switches (Up = Hi)
NET "REVERSE" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP;

# RS-232 Serial Port: DCE
NET "RS232_RXD" LOC = "R7" | IOSTANDARD = LVTTL ;


# ==== Character LCD (LCD) ====
NET "LCD_E" LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW =
    SLOW ;
NET "LCD_RS" LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW =
    SLOW ;
NET "LCD_RW" LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW =
    SLOW ;
# LCD data connections are shared with StrataFlash connections SF_D
    <11:8>
NET "LCD_D<0>" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW
    = SLOW ;
NET "LCD_D<1>" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW
    = SLOW ;
NET "LCD_D<2>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW
    = SLOW ;
```

```
21 NET "LCD_D<3>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW
     = SLOW ;
22 NET "SF_CE" LOC = "D16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW =
     SLOW ;
```

4 Wnioski

Niestety przez zajęcia zdalne nie mieliśmy możliwości przetestowania zaprojektowanego układu. Podczas wykonywania zadania pamiętaliśmy że przy włączonym dzielniku częstotliwości rezultaty na symulacji nie byłyby widoczne dlatego podczas symulacji wykomentowaliśmy część odpowiadającą za dzielenie.

Moduły wykorzystane w schemacie od Spartana nie różniły się znacznie od modułów ZL-9572 dzięki czemu nie było z nimi problemów - w przypadku układu RS232 odwoływaliśmy się niemalże do tych samych zestawów wejść i wyjść, co w wersji układu dla platformy CPLD (pozostawiliśmy jedynie niepotrzebne w bieżącym zadaniu wyprowadzenia do obsługi wysyłu danych); w przypadku obsługi wyświetlacza, zmianie uległy szerokości magistrali wejściowych oraz liczba wyjść układu.

Dzięki zadaniu zapoznaliśmy się ze szczegółami dotyczącymi modułów dla platformy FPGA oraz przekonaliśmy się ponownie, jak istotne w projektach układów cyfrowych właściwe wysterowanie domen zegarowych.