Maciej Byczko Bartosz Matysiak	Prowadzący: dr inż. Jacek Mazurkiewicz	Numer ćwiczenia 6
PN 10:50 TP	Temat ćwiczenia: Licznik synchroniczny sterowany	Ocena:
Grupa: B	Data wykonania: 15 Grudnia 2021r.	

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.

Oznacza to, że do przystawki dotrze kod ASCII naciśniętego klawisza poprzez port szeregowy RS232 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 w postaci 2-cyfrowej liczby szesnastkowej na wyświetlaczu 7-segmentowym. 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 oraz obsługi wyświetlacza 7-segmentowego w wersji wielocyfrowej.

2 Rozwiązanie

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

Wykonaliśmy to w ”sprytny” sposób wiedząc że możemy ikrementować bądź dekrementować wartość bitową więc nie potrzebujemy żadnych dodatkowych zmiennych. Za pomocą zmiennej splitter wykonaliśmy dzielnik częstotliwości aby wartość wyświetlana na ekranie była czytelna dla użytkownika

2.1 Kod VHDL

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use ieee.std_logic_unsigned.all;
4 use IEEE.STD_LOGIC_ARITH.ALL;
5 use ieee.numeric_std.all;
6
7 entity counter_mod is
8     Port ( WEJ : in  STD_LOGIC_VECTOR (7 downto 0);
9            REVERSE : in  STD_LOGIC;
```

```
10              CLK_LF : in  STD_LOGIC;
11              RESET : in  STD_LOGIC;
12              WYJ : out  STD_LOGIC_VECTOR (7 downto 0));
13  end counter_mod;
14
15  architecture Behavioral of counter_mod is
16
17    signal current_state: STD_LOGIC_VECTOR(7 downto 0);
18    signal splitter: STD_LOGIC_VECTOR(7 downto 0);-- := "00000000";
19
20  begin
21
22    process1: process(CLK_LF, RESET, REVERSE, WEJ)--, splitter,
        current_state)
23    begin
24      -- gdy wcisniety reset, pobranie wartosci z klawiatury
25      if RESET = '1' then
26        current_state <= WEJ;
27      elsif rising_edge(CLK_LF) then
28        splitter <= splitter + 1;
29        if splitter = "10000000" then
30          splitter <= "00000000";
31          if REVERSE = '0' then
32            if current_state = "11111111" then
33              current_state <= "00000000";
34            else
35              current_state <= current_state + 1;
36            end if;
37          else
38            if current_state = "00000000" then
39              current_state <= "11111111";
40            else
41              current_state <= current_state - 1;
42            end if;
43          end if;
44        end if;
45      end if;
46
47    end process process1;
48
49    WYJ <= current_state;
50
51  end Behavioral;
```

2.2 Kod VHDL TestBench

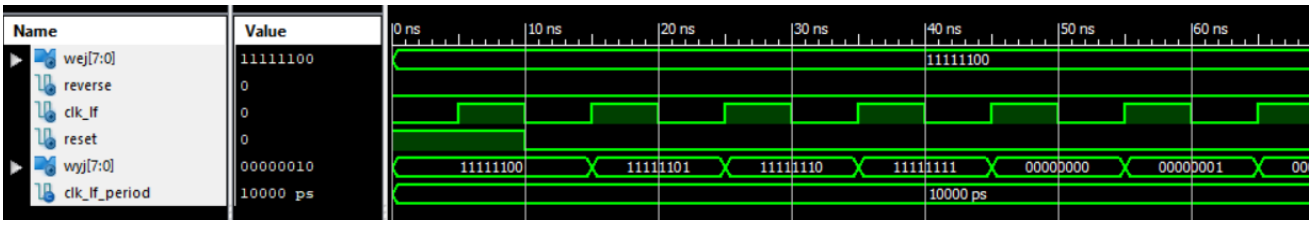
```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY counter_mod_benchmark IS
5  END counter_mod_benchmark;
6
```

```
7  ARCHITECTURE behavior OF counter_mod_benchmark IS
8
9      COMPONENT counterMod
10     PORT(
11           WEJ : IN  std_logic_vector(7 downto 0);
12           REVERSE : IN  std_logic;
13           CLK_LF : IN  std_logic;
14           RESET : IN  std_logic;
15           WYJ : OUT  std_logic_vector(7 downto 0)
16          );
17     END COMPONENT;
18
19    --Inputs
20    signal WEJ : std_logic_vector(7 downto 0) := (others => '0');
21    signal REVERSE : std_logic := '0';
22    signal CLK_LF : std_logic := '0';
23    signal RESET : std_logic := '0';
24
25   --Outputs
26    signal WYJ : std_logic_vector(7 downto 0);
27
28    -- Clock period definitions
29    constant CLK_LF_period : time := 10 ns;
30
31 BEGIN
32
33   -- Instantiate the Unit Under Test (UUT)
34    uut: counterMod PORT MAP (
35            WEJ => WEJ,
36            REVERSE => REVERSE,
37            CLK_LF => CLK_LF,
38            RESET => RESET,
39            WYJ => WYJ
40          );
41
42    -- Clock process definitions
43    CLK_LF_process :process
44    begin
45     CLK_LF <= '0';
46     wait for CLK_LF_period/2;
47     CLK_LF <= '1';
48     wait for CLK_LF_period/2;
49    end process;
50
51    -- Stimulus process
52    stim_proc: process
53    begin
54       -- hold reset state for 100 ns.
55
56     WEJ <= "11111100", "11110011" after 650ns;
57     --CLK_LF <= not CLK_LF after 50ns;
```

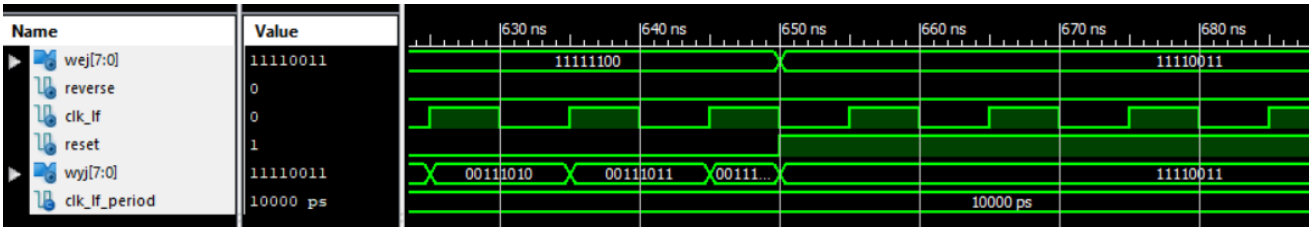
```
58      RESET <= '1', '0' after 10ns, '1' after 650ns, '0' after 750ns;
59      REVERSE <= '0', '1' after 650ns;
60         wait;
61      end process;
62
63  END;
```

2.3 Symulacja

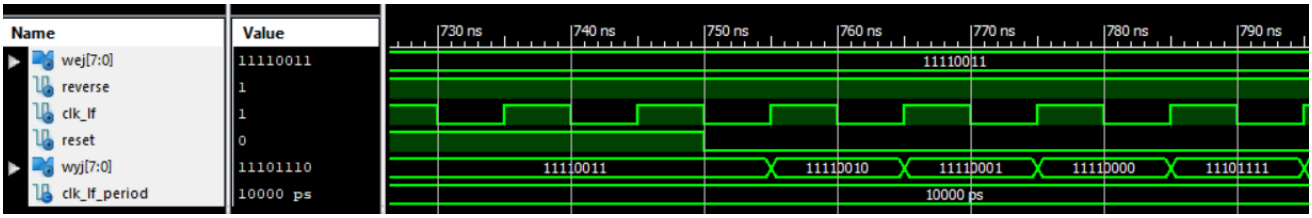
Rysunek 1: Początek symulacji



Rysunek 2: Wprowadzenie nowej wartości

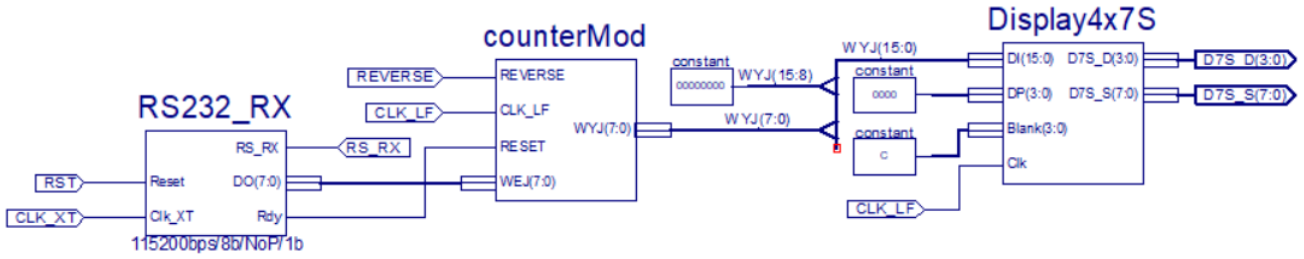


Rysunek 3: Odwrócenie kolejności odliczania



2.4 Schemat układu z wykorzystaniem zaprojektowanego modułu

Rysunek 4: Schemat z podłączoną klawiaturą oraz wyświetlaczem LED



3 Fizyczna implementacja

3.1 Kod UCF

```
# Clocks
NET "Clk_LF" LOC = "P7" | BUFG = CLK | PERIOD = 5ms HIGH 50%;
NET "Clk_XT" LOC = "P5" | BUFG = CLK | PERIOD = 500ns HIGH 50%;

# Keys
NET "REVERSE" LOC = "P42";
NET "RST" LOC = "P40";

# DISPL. 7-SEG
NET "D7S_D(0)" LOC = "P8" | SLEW = "SLOW";
NET "D7S_D(1)" LOC = "P6" | SLEW = "SLOW";
NET "D7S_D(2)" LOC = "P4" | SLEW = "SLOW";
NET "D7S_D(3)" LOC = "P9" | SLEW = "SLOW";
NET "D7S_S(0)" LOC = "P12"; # Seg. A; shared with LED<10>
NET "D7S_S(1)" LOC = "P13"; # Seg. B; shared with LED<8>
NET "D7S_S(2)" LOC = "P22"; # Seg. C; shared with LED<12>
NET "D7S_S(3)" LOC = "P19"; # Seg. D; shared with LED<14>
NET "D7S_S(4)" LOC = "P14"; # Seg. E; shared with LED<15>
NET "D7S_S(5)" LOC = "P11"; # Seg. F; shared with LED<9>
NET "D7S_S(6)" LOC = "P20"; # Seg. G; shared with LED<13>
NET "D7S_S(7)" LOC = "P18"; # Seg. DP;shared with LED<11>

# RS-232
NET "RS_RX" LOC = "P1";
```

4 Wnioski

W trakcie wykonywania tak zaawansowanego zadania laboratoryjnego, jakim okazał się opisywany tutaj licznik synchroniczny, napotkaliśmy na szereg pomniejszych wyzwań, z którymi przyszło nam się zmierzyć; jednym z nich okazało się umiejscowienie jednej w instrukcji przypisania w kodzie VHDL tworzonego modułu. Początkowo umieściliśmy przypisanie do wejścia w zakresie procesu; powodowało to, początkowo dla nas niezrozumiałe, opóźnienie przypisania tego sygnału o pół okresu naszego licznika. Po wnikliwej analizie udało nam się jednak sprostować popełnioną gafę, i umieścić owe przypisanie jako niezależną instrukcję współbieżną. Przykład ten pozwolił nam na utrwalenie wiedzy wyniesionej z wykładu.

Kolejnym wyzwaniem, z jakim przyszło nam się zmierzyć w trakcie laboratoriów, była konieczność podziału częstotliwości 240Hz zegara **CLK_LF**. Była ona podyktowana tym, że dwa z modułów użytych w projekcie (Display4x7S oraz nasz własny) używały różnych ustawień tego samego zegara. Finalnie, problem zdecydowaliśmy się rozwiązać poprzez wprowadzenie w kodzie naszego układu dodatkowego stanu, który zliczałby cykle zegara wejściowego. Każdy bit dodatkowego stanu umożliwił spowolnienie układu licznika 2 razy; dla podanej częstotliwości zegara uznaliśmy zatem rozmiar 8-bitów za odpowiedni.

Wykonaliśmy to wewnątrz kodu VHDL ponieważ gdy próbowaliśmy wykonać to za pomocą przerzutników typu T to natrafiliśmy na błąd braku pamięci czyli brak możliwości dokładania dodatkowych elementów do schematu.