Termin zajęć Wtorek NP 7:30 – 11:00	Układy cyfrowe i systemy wbudowane	
Osoby wykonujące ćwiczenie: Jakub Suski 264028, Adam Czekalski 264488		**Grupa:** **D**
Tytuł ćwiczenia: Licznik synchroniczny sterowany		**Laboratorium nr:** **6**
Data wykonania ćwiczenia	5.12.2023	**Ocena:**
Data oddania sprawozdania	19.12.2023	

Na zajęciach laboratoryjnych, wykorzystano cały szereg nabytych do tej pory umiejętności, konstruując 8-bitowy licznik NKB mogący liczyć zarówno w górę jak i w dół na płytce CPLD.

1. Plik modułowy VHDL

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.NUMERIC_STD.ALL;
23
24  -- Uncomment the following library declaration if using
25  -- arithmetic functions with Signed or Unsigned values
26  --use IEEE.NUMERIC_STD.ALL;
27
28  -- Uncomment the following library declaration if instantiating
29  -- any Xilinx primitives in this code.
30  --library UNISIM;
31  --use UNISIM.VComponents.all;
32
33  entity counter is
34      Port ( START_STOP : in  STD_LOGIC;
35             PAUSE : in  STD_LOGIC;
36             LOAD : in  STD_LOGIC;
37             RESET : in  STD_LOGIC;
38             OUT_O : out  STD_LOGIC_VECTOR (7 downto 0);
39             IN_I : in  STD_LOGIC_VECTOR (7 downto 0);
40             REVERSE : in STD_LOGIC;
41             CLOCK : in  STD_LOGIC);
42  end counter;
43
44  architecture Behavioral of counter is
45
46     signal number : UNSIGNED(7 downto 0) := (others => '0');
47     signal isStarted : STD_LOGIC := '0';
48
49
50  begin
51
52     process(START_STOP)
53     begin
54        if START_STOP = '1' then
55           isStarted <= '1';
56
```

Rysunek 1 Pierwszy fragment pliku modułowego VHDL

```
57          elsif START_STOP = '0' then
58              isStarted <= '0';
59
60          end if;
61
62      end process;
63
64      process(CLOCK, PAUSE, LOAD, RESET)
65      begin
66          if RESET = '1' then
67              number <= (others => '0');
68
69          elsif PAUSE = '0' and isStarted = '1' then
70              if LOAD = '1' then
71                  number <= UNSIGNED(IN_I);
72              --end if;
73              elsif rising_edge(CLOCK) then
74                  if REVERSE = '0' then
75                      number <= number + 1;
76                  elsif REVERSE = '1' then
77                      number <= number - 1;
78                  end if;
79              end if;
80          end if;
81      end process;
82
83      OUT_O <= STD_LOGIC_VECTOR(number);
84
85  end Behavioral;
86
87
```

Rysunek 2 Drugi fragment pliku modułowego VHDL

Licznik składa się z następujących portów:

- *START_STOP* – kiedy podajemy na linię sygnał o wartości ‘1’, licznik odlicza, kiedy ‘0’ – licznik zatrzymuje się
- *LOAD* – pozwala załadować liczbę podaną na porcie *IN_I*, gdy podamy sygnał ‘1’
- *RESET* – pozwala zresetować licznik do stanu początkowego, gdy podamy na linię sygnał ‘1’
- *OUT_O* – port podający liczbę na wyjście
- *IN_I* – pozwala wprowadzić 7 bitową liczbę, od której licznik ma rozpocząć liczenie
- *REVERSE* – określa w którą stronę odlicza licznik – jeśli REVERSE = ‘0’, wtedy w górę, jeśli REVERSE = ‘1’, wtedy w dół
- *CLOCK* – zegar działający analogicznie jak w układach sekwencyjnych

W kodzie VHDL wykorzystano też zmienne pomocnicze:

- *isStarted* – zmienna typu bool sygnalizująca czy licznik rozpoczął liczenie, jeśli tak, to przyjmuje wartość ‘1’, jeśli nie, to ‘0’

- *number* – zmienna typu UNSIGNED, która przechowuje aktualny stan w jakim znajduje się licznik, która jest parsowana na zmienną typu wektor, w celu podania jej na port *OUT_O*

2. Plik testowy VHDL

```
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY test IS
36  END test;
37
38  ARCHITECTURE behavior OF test IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT counter
43      PORT(
44           START_STOP : IN  std_logic;
45           PAUSE : IN  std_logic;
46           LOAD : IN  std_logic;
47           RESET : IN  std_logic;
48           OUT_O : OUT  std_logic_vector(7 downto 0);
49           IN_I : IN  std_logic_vector(7 downto 0);
50           REVERSE : IN  std_logic;
51           CLOCK : IN  std_logic
52          );
53      END COMPONENT;
54
55
56     --Inputs
57     signal START_STOP : std_logic := '0';
58     signal PAUSE : std_logic := '0';
59     signal LOAD : std_logic := '0';
60     signal RESET : std_logic := '0';
61     signal IN_I : std_logic_vector(7 downto 0) := (others => '0');
62     signal REVERSE : std_logic := '0';
63     signal CLOCK : std_logic := '0';
64
65      --Outputs
66     signal OUT_O : std_logic_vector(7 downto 0);
67
68  BEGIN
69
70      -- Instantiate the Unit Under Test (UUT)
71     uut: counter PORT MAP (
72           START_STOP => START_STOP,
73           PAUSE => PAUSE,
74           LOAD => LOAD,
75           RESET => RESET,
76           OUT_O => OUT_O,
77           IN_I => IN_I,
78           REVERSE => REVERSE,
79           CLOCK => CLOCK
80          );
81
82     CLOCK <= not CLOCK after 10 ns;
83      START_STOP <= '1', '0' after 50 ns, '1' after 100 ns, '0' after 500 ns;
84      IN_I <= "10000001";
85      LOAD <= '1' after 40 ns, '0' after 55 ns;
86      PAUSE <= '1' after 170 ns, '0' after 250 ns;
87      RESET <= '1' after 1000 ns, '0' after 1050 ns;
88      REVERSE <= '1' after 80 ns, '0' after 200 ns;
89
90  END;
91
```

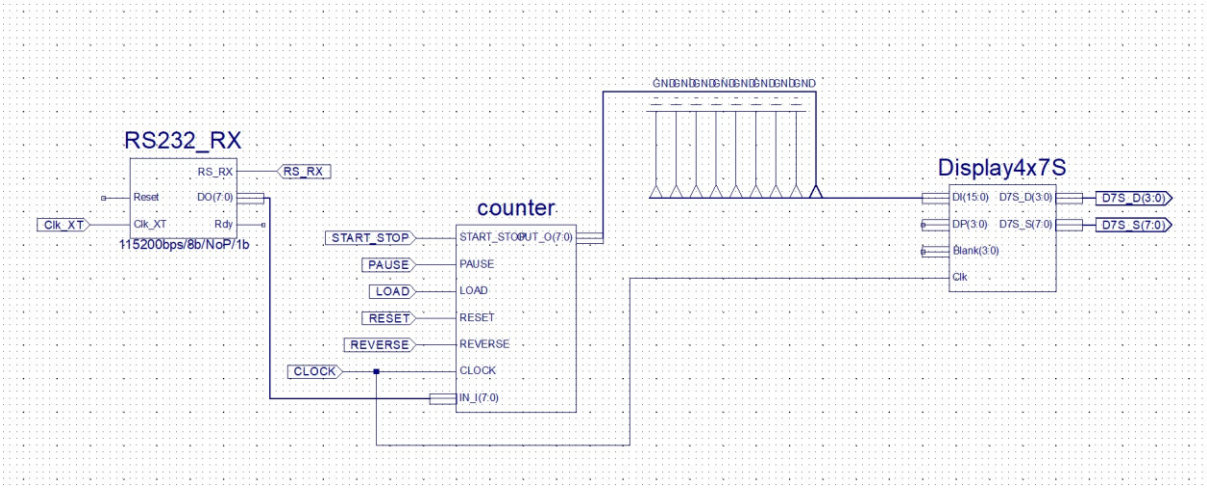
Rysunek 3 Plik testowy VHDL

3. Symulacja

Rysunek 4 Symulacja układu licznika

Od początku symulacji idąc, widać że jeśli na porcie *start_stop* pojawi się zbocze wyzwalające, licznik liczy do przodu. Następnie gdy magistrali *in_i* podajemy wektor o wartości „10000001” oraz na port *load* zbocze wyzwalające, do licznika zostaje załadowany wektor z magistrali. Następnie pojawia się zbocze wyzwalające na porcie *reverse* oraz *start_stop* co jest równoważne z rozpoczęciem liczenia w tył. Następnie zostaje podana wartość logiczna ‘1’ na port *pause*, nie zmieniając stanu portu *start_stop* co powoduje wstrzymanie liczenia do momentu podania wartości ‘0’ na port *pause*. Kolejno znów zmienia się kierunek liczenia, poprzez podanie na port *reverse* ‘0’, więc licznik liczy znów w przód.

4. Podpięcie klawiatury i wyświetlacza



Rysunek 5 Plik z rozszerzeniem .sch

W celu podpięcia klawiatury i wyświetlacza do stworzonej wcześniej jednostki licznika w języku VHDL, skorzystano z pliku z rozszerzeniem .sch, w którym można było umieścić skonstruowaną wcześniej czarną skrzynkę. Podpięto moduł *RS232_RX* oraz *Display4x7s*. Czarna skrzynka odpowiedzialna za klawiaturę jest „napędzana” generatorem pracującym w częstotliwości 1,8432MHz, natomiast licznik oraz wyświetlacz generatorem pracującym w częstotliwości 240Hz. Z klawiatury można wprowadzić wartość inicjującą licznik, na wyświetlaczu natomiast widoczny jest aktualny stan licznika w postaci 2-cyfrowej liczby szesnastkowej.

5. Plik z rozszerzeniem .ucf

```
#------------------------------------------------------------
#  ZL-9572 CPLD board,  J.Sugier 2009
#------------------------------------------------------------

# Clocks
NET "CLOCK" 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 "RESET" LOC = "P40";
#NET "LOAD" LOC = "P43";
NET "PAUSE" LOC = "P38";
NET "START_STOP" LOC = "P37";
#NET "Key<5>" LOC = "P36";  # shared with ROT_A
#NET "Key<6>" LOC = "P24";  # shared with ROT_B
#NET "Key<7>" LOC = "P39";  # GSR

# LEDS
#NET "LED<0>"  LOC = "P35";
#NET "LED<1>"  LOC = "P29";
#NET "LED<2>"  LOC = "P33";
#NET "LED<3>"  LOC = "P34";
#NET "LED<4>"  LOC = "P28";
#NET "LED<5>"  LOC = "P27";
#NET "LED<6>"  LOC = "P26";
#NET "LED<7>"  LOC = "P25";

#NET "LED<8>"  LOC = "P13"; # shared with seg. B
#NET "LED<9>"  LOC = "P11"; # shared with seg. F
#NET "LED<10>" LOC = "P12"; # shared with seg. A
#NET "LED<11>" LOC = "P18"; # shared with seg. DP
#NET "LED<12>" LOC = "P22"; # shared with seg. C
#NET "LED<13>" LOC = "P20"; # shared with seg. G
#NET "LED<14>" LOC = "P19"; # shared with seg. D
#NET "LED<15>" LOC = "P14"; # shared with seg. E

# 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>

# Rotary encoder
#NET "ROT_A" LOC = "P36";   # shared with Key<5>
#NET "ROT_B" LOC = "P24";   # shared with Key<6>

# PS/2
#NET "PS2_Clk"   LOC = "P3";
#NET "PS2_Data"  LOC = "P2";

# RS-232
NET "RS_RX" LOC = "P1";
#NET "RS_TX" LOC = "P44";
```

Rysunek 6 Plik z rozszerzeniem .ucf

Kolejne przyciski odpowiedzialne są za:

- K0 – zmiana kierunku liczenia
- K1 – zresetowanie licznika
- K2 – załadowanie wartości do licznika
- K3 – wstrzymanie liczenia
- K4 – rozpoczęcie liczenia

Odkomentowano sekcje odpowiedzialne za klawiaturę, wyświetlacz oraz zegary.

6. Wnioski

Licznik działał poprawnie, według przyjętych założeń. Wszystkie założenia udało się zrealizować.