POLITECHNIKA WROCŁAWSKA
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

Układy cyfrowe i systemy wbudowane

Laboratorium 7

Licznik synchroniczny sterowany w Spartanie

Termin zajęć: Czwartek TP, 7:30

Autorzy:
Daria Jeżowska, 252731
Kacper Śleziak, 252703

Prowadzący zajęcia:
dr inż. Jacek Mazurkiewicz

1.Zadania do wykonania

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. Licznik po przyjęciu kodu zaczyna liczyć – grupa wybiera czy będzie zwiększał swój stan – będzie początkowo – pozytywny, czy też zmniejszał swój stan będzie początkowo negatywny. Jest to na dobrą sprawę licznik zbudowany przez nas w ramach laboratorium numer 6. Względem poprzedniego laboratorium miały zmieniać się modły do obsługi portu RS232 bądź PS/2.

2 Kod w języku VHDL

Do stworzenia licznika synchronicznego potrzebne były zmienne reprezentujące zmianę kierunku zliczania (up, 0 oznacza zliczanie w dół, a 1 w górę), załadowania nowej wartości (load), liczba do załadowania (num_to_load) oraz liczba pokazywana na wyjściu (num_out). Stworzony został także sygnał o nazwie Number, który jest 8 bitowym wektorem. Jego startową wartością jest 0. W bloku process rozpoczyna się działanie naszego programu. Aby wypisywanie rozpoczęło się od zera wypisujemy więc najpierw liczbę, a następnie sprawdzamy czy load ma wartość 1 - jeśli tak jest to Number przyjmuje wartość num_to_load. W następnym kroku sprawdzamy czy up jest jedynką. Jeśli jest to mamy warunek zabezpieczający - sprawdzamy czy ma wartość 11111111 - w takim przypadku zmieniamy wartość Number na 00000000, aby licznik mógł działać dalej. Następnie Number jest zwiększane o 1. W kolejnym warunku elsif sprawdzamy czy up jest zerem - jeśli jest to musimy sprawdzić czy Number ma wartość 00000000 - aby zabezpieczyć kod i aby program mógł dalej działać wektor zostaje zamieniony na 11111111. Po sprawdzeniu tego warunku Number zostaje pomniejszone o 1.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity lab7 is
    Port ( load : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           num_to_load : in  STD_LOGIC_VECTOR (7 downto 0);
           up : in  STD_LOGIC;
           num_out : out  STD_LOGIC_VECTOR (7 downto 0));
end lab7;

architecture Behavioral of lab7 is

   signal Number: UNSIGNED(7 downto 0) := "00000000";
   signal Started: STD_LOGIC := '0';

begin


   process(Clk, Load)
   begin
      if load = '1' then
            Number <= Unsigned(num_to_load);

      elsif up = '1' then
         if Number <= "11111111" then
            Number <= "00000000";
         end if;
         Number <= Number + 1;

      elsif up = '0' then
         if Number <= "00000000" then
            Number <= "11111111";
         end if;
         Number <= Number - 1;
      end if;
   end process;
   num_out <= STD_LOGIC_VECTOR(Number);

end Behavioral;
```

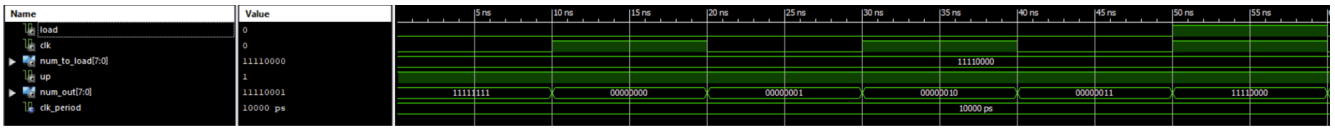
3 Testbench

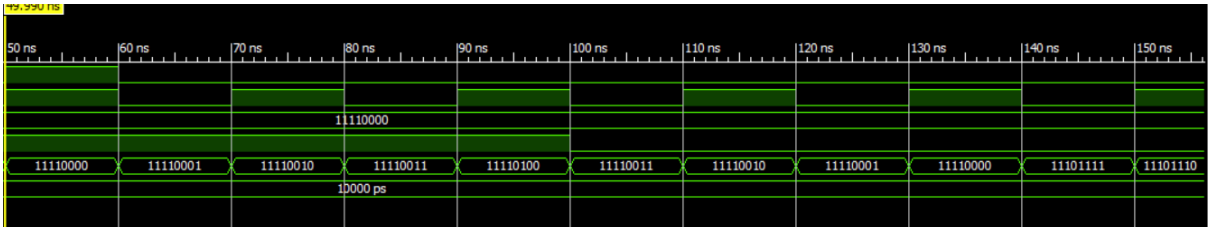
Aby przetestować czy powyższy kod działa poprawnie należało stworzyć testbench. Pełen cykl zegaru clk trwa 20ns. Wartość liczby do załadowania,num_to_load, została ustawiona na 11110000. Domyślnie up jest jedynkę, jednak zostaje zmienione na 0 po upływie 100ns symulacji. Liczba num_to_loadM zostanie załadowana po 50ns, gdy load zmieni wartość na 1.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY lab7_test_bench IS
END lab7_test_bench;


ARCHITECTURE behavior OF lab7_test_bench IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT lab7
    PORT(
         load : IN  std_logic;
         clk : IN  std_logic;
         num_to_load : IN  std_logic_vector(7 downto 0);
         up : IN  std_logic;
         num_out : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal load : std_logic := '0';
   signal clk : std_logic := '0';
   signal num_to_load : std_logic_vector(7 downto 0) := (others => '0');
   signal up : std_logic := '0';

   --Outputs
   signal num_out : std_logic_vector(7 downto 0);

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: lab7 PORT MAP (
          load => load,
          clk => clk,
          num_to_load => num_to_load,
          up => up,
          num_out => num_out
        );

   clk <= not Clk after 10ns;
   num_to_load <= "11110000";
   up <= '1', '0' after 100ns;
   load <= '0', '1' after 50ns, '0' after 60ns;


END;
```

4 Testbench

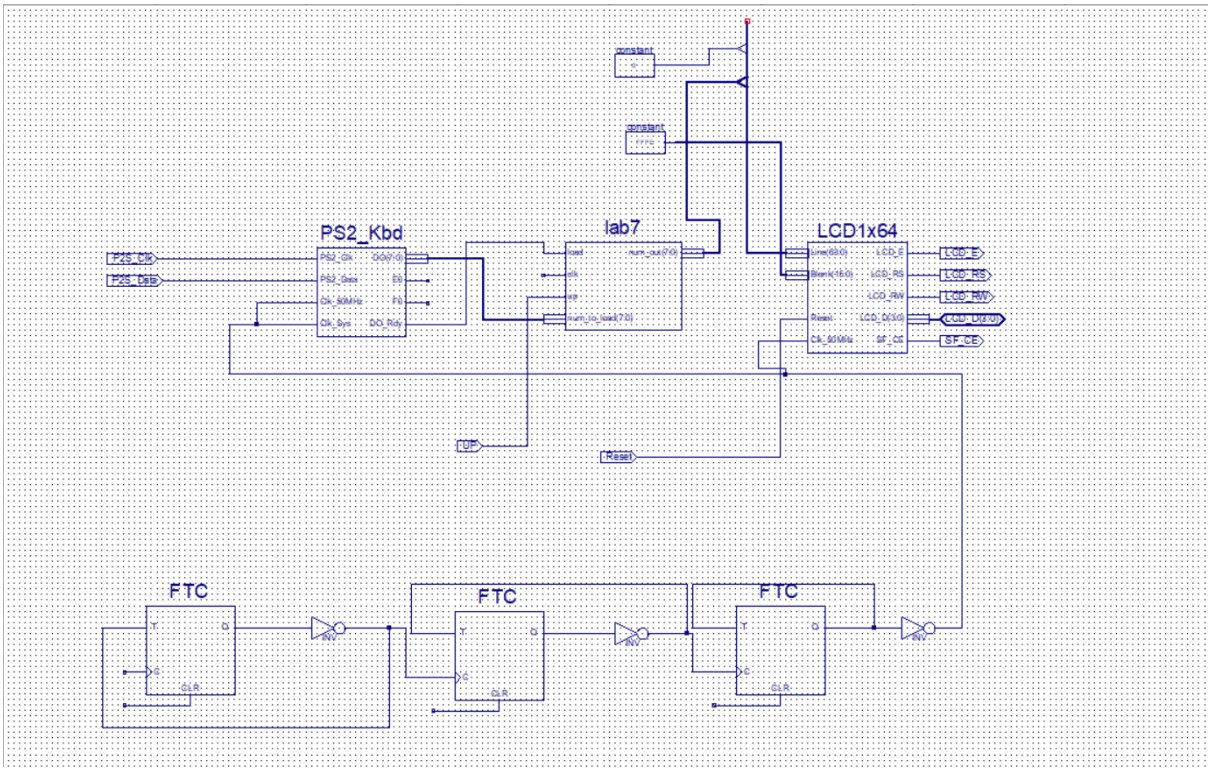
Symulacja przebiegła zgodnie z oczekiwaniami. Licznik zaczyna liczyć od maksymalnej liczby, a później wchodzi do zera, następnie zliczający w górę, ponieważ up ma wartość 1. Na drugim zrzucie ekranu widać, że w 110ns wartość load zmieniła się z 1 na 0, a wraz ze zmianą tej wartości licznik zaczął liczyć w dół.





5 Schemat

Względem poprzedniego laboratorium zamieniliśmy moduł RS232_RX na PS2_Kbd oraz Display4x7s na LCD1X64, oba modułu zostały pobrane ze strony dr. inż. Sugiera. Moduł o nazwie lab7 jest zaprojektowanym przez nas licznikiem. Wyjścia DO(7:0) oraz Do_Rdy z pierwszego modułu odpowiadają kolejno liczbie do załadowania oraz załadowaniem (num_to_load oraz load). Do wyświetlenia num_out została stworzona magistrala.



6 Wnioski

Z racji przejścia na zajęcia zdalne nie udało nam się przetestować powyższego układu na płytce. Wykonanie zadania odbyło się bez większych problemów.