Układy cyfrowe i systemy wbudowane
Laboratorium 6
Licznik synchroniczny sterowany

Autorzy: Daria Jeżowska, 252731
Kacper Śleziak 252703
Prowadzący: 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.

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 wartść `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 lab6 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 lab6;

architecture Behavioral of lab6 is
        signal Number: UNSIGNED(7 downto 0) := "00000000";

begin
        process(Clk, Load)
        begin
                num_out <= STD_LOGIC_VECTOR(Number);
                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;
end Behavioral;
```

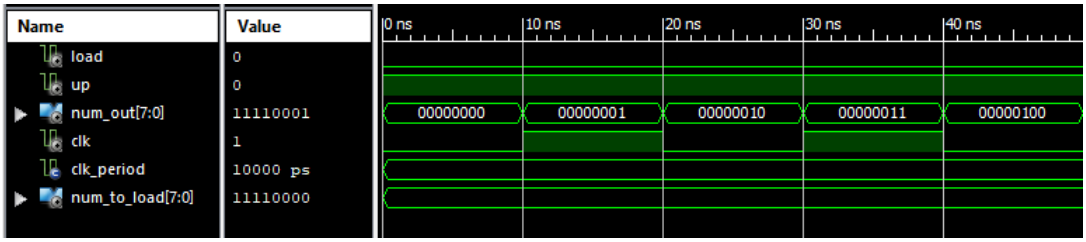
3 Testbench

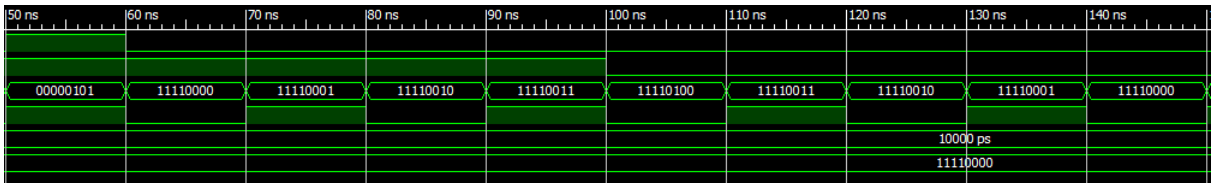
Aby przetestować czy powyższy kod działa poprawnie należało stworzyć *testbench*. Pełen cykl zegaru `clk` trwa 20ns. Wartość liczby do załadowania, `num_to_load`, zostala ustawiona na `11110000`. Domyślnie `up` jest jedynkę, jednak zostaje zmienione na 0 po upływie 100ns symulacji. Liczba `num_to_load`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

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY lab6tb IS
END lab6tb;

ARCHITECTURE behavior OF lab6tb IS
    COMPONENT lab6
    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;

   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 := '1';
   signal num_out : std_logic_vector(7 downto 0);
   constant clk_period : time := 10 ns;

BEGIN
   uut: lab6 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 <= '0' after 100ns;
        load <= '0', '1' after 50ns, '0' after 60ns;
END;
```

4 Symulacja

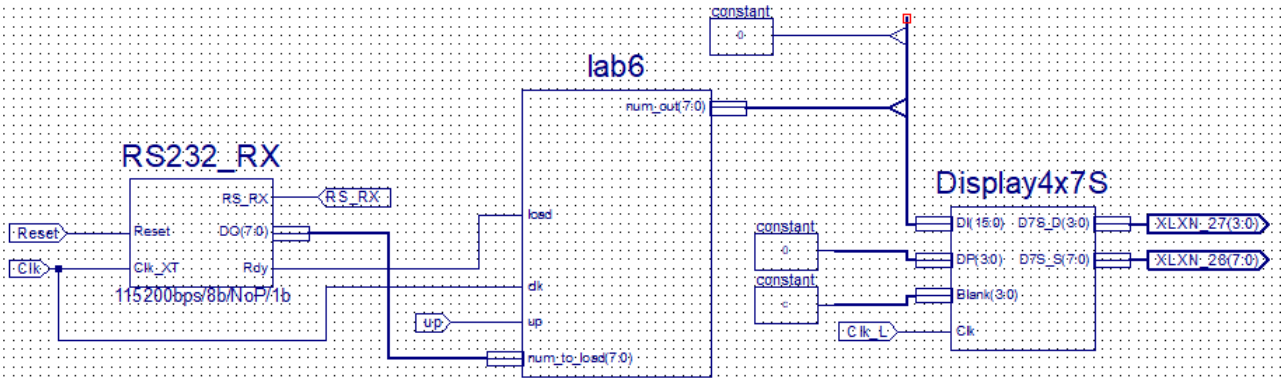
Symulacja przebiegła zgodnie z oczekiwaniami. Licznik zaczyna liczyć od zera w górę, ponieważ `up` ma wartość 1. Na drugim zrzucie ekranu widać, że wartość `load` zmieniła się z zera na jedynkę oraz liczba widoczna na wyjściu zmieniła się na `11110000` i licznik dalej liczy w górę. W setnej nanosekundzie symulacji `up` zmienia wartość na 0, a zliczanie następuje od tego momentu do dołu.





5 Schemat

Na schemacie użyliśmy modułu `RS232_RX` oraz `Display4x7S` pobranych ze strony dr. inż. Sugiera. Moduł `lab6` jest licznikiem. Wyjścia `DO(7:0)` oraz `Rdy` z pierwszego modułu odpowiadają kolejno liczbie do załadowania oraz załadowaniem (`num_to_load` oraz `load`). Natomiast, aby wyświetlić `num_out` na wyświetlaczu stworzona została magistrala, która jako pierwsze 8 bitów zawiera jedynki, a pozostałe są właśnie `num_out`.



6 Wnioski

Niestety z powodu nieobecności na ostatnich zajęciach stacjonarnych dwóch osób z grupy na zajęciach nie mogliśmy przetestować projektu na płytce i teraz także jest to niekoniecznie możliwe.
Zadanie to wymagało przygotowania się i doczytania o działaniu modłów RS232 i wybraniu odpowiedniego oraz analogicznie dla Display4x7S. Sam w sobie licznik nie sprawiał większych problemów - po zajęciach z Logiki układów cyfrowych oraz dotychczasowych zajęć z Układów cyfrowych i systemów wbudowanych wiedzieliśmy jak odpowiednio napisać kod, aby symulacja była satysfakcjonująca.