Systemy Czasu Rzeczywistego

„FPGA”

laboratorium: 04

autor: mgr inż. Mateusz Baran

Data utworzenia: 01/10/14

1 Spis treści

2 Wiadomości wstępne

Pierwsza część niniejszej instrukcji zawiera podstawowe wiadomości teoretyczne dotyczące omawianego tematu. Poznanie tych wiadomości umożliwi prawidłowe zrealizowanie praktycznej części laboratorium.

2.1 Niezbędne wiadomości

Przed laboratorium należy zapoznać się z podstawowymi pojęciami dotyczącymi budowy układów FPGA:

- zasoby logiczne FPGA (CLB, slice, pamięć Block RAM, mnożarki),
- zasoby połączeniowe (programowalne połączenia wewnętrze, IOB),
- podstawy języka VHDL.

Podczas laboratorium będzie wykorzystywany układ Digilent Basys2 oparty o układ Xilinx Spartan3E-250.

2.2 Literatura:

[1] Podstawowy opis architektury układów FPGA:
http://www.csd.uoc.gr/~hy220/2009f/lectures/11_basic_fpga_arch.pdf

[2] Szczegóły budowy wykorzystywanych układów FPGA:
http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

[3] Podstawy języka VHDL:
https://wiki.ittc.ku.edu/ittc/images/3/37/EECS_140_VHDL_Tutorial.pdf

3 Przebieg laboratorium

3.1 Zadanie 1. Na ocenę 3.0 (dst)

W zadaniu wykorzystamy instrukcję warunkową CASE (podobną do instrukcji switch z języka C) do sterowania wyświetlaczem siedmiosegmentowym. W języku VHDL instrukcja CASE ma następującą postać:

```
CASE input(2 downto 0) IS
  WHEN "000" =>
    output1 <= '1';
    output2 <= '1';
  WHEN "001" =>
    output1 <= '0';
    output2 <= '1';
  WHEN "110" =>
    output1 <= '1';
    output2 <= '0';
  WHEN OTHERS =>
    output1 <= '0';
    output2 <= '0';
```



```
END CASE;
```

Może ona występować, podobnie jak instrukcja IF, wyłącznie w procesach. Ponadto zawsze musimy pokryć wszystkie możliwe przypadki. Na ogół sprowadza się to do napisania warunku „WHEN OTHERS”. Z uwagi na wiele możliwych wartości sygnałów typu STD_LOGIC jawne wyliczenie wszystkich możliwości bywa mało praktyczne.

W celu wykorzystania wyświetlacza siedmiosegmentowego musimy rozszerzyć nasz plik ograniczeń o odpowiadające mu wpisy:

```
NET "sevenseg<0>" LOC = "L14";
NET "sevenseg<1>" LOC = "H12";
NET "sevenseg<2>" LOC = "N14";
NET "sevenseg<3>" LOC = "N11";
NET "sevenseg<4>" LOC = "P12";
NET "sevenseg<5>" LOC = "L13";
NET "sevenseg<6>" LOC = "M12";
NET "dp" LOC = "N13";

NET "anodes<3>" LOC = "K14";
NET "anodes<2>" LOC = "M13";
NET "anodes<1>" LOC = "J12";
NET "anodes<0>" LOC = "F12";
```

Następnie proszę rozszerzyć poniższy kod o tak, aby wyjście anodes było ustawiane na „1110”, zaś wyjścia sevenseg i dp były sterowane przełącznikami. Proszę zanotować który przełącznik steruje którym segmentem wyświetlacza.

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



entity Switches_LEDs is
   Port ( switches : in  STD_LOGIC_VECTOR(7 downto 0);
          LEDs     : out STD_LOGIC_VECTOR(7 downto 0);
          clk      : in STD_LOGIC;
          sevenseg : out STD_LOGIC_VECTOR(6 downto 0);
          dp : out STD_LOGIC;
          anodes : out STD_LOGIC_VECTOR(3 downto 0)
        );
end Switches_LEDs;

architecture Behavioral of Switches_LEDs is

begin
```



```
end Behavioral;
```

Następnie proszę przygotować dodać 32-bitowy licznik (na podstawie wcześniejszych laboratoriów) inkrementowany co takt zegara. Na wyświetlaczu będziemy wyświetlać bity od 27 do 24.

Poniższy proces umożliwia wyświetlenie czterobitowego sygnału jako cyfry na wyświetlaczu siedmiosegmentowym:

```
num: process(currentDigit)
        begin
                case currentDigit is
                        when "1111" => sevenseg <= "0001110"; --F
                        when "1110" => sevenseg <= "0000110"; --E
                        when "1101" => sevenseg <= "0100001"; --d
                        when "1100" => sevenseg <= "1000110"; --C
                        when "1011" => sevenseg <= "0000011"; --b
                        when "1010" => sevenseg <= "0001000"; --A
                        when "1001" => sevenseg <= "0010000"; --9
                        when "1000" => sevenseg <= "0000000"; --8
                        when "0111" => sevenseg <= "1111000"; --7
                        when "0110" => sevenseg <= "0000010"; --6
                        when "0101" => sevenseg <= "0010010"; --5
-- proszę uzupełnić
                        when "0000" => sevenseg <= "1000000"; --0
                        when others => sevenseg <= "0110110"; --???
                end case;
        end process;
```

Proszę uzupełnić brakujące przypadki oraz zastosować powyższy kod do wyświetlenia wymienionych wcześniej bitów licznika. Co się dzieje gdy brakuje któregoś z przypadków? Proszę zamieścić odpowiedź i kod projektu w sprawozdaniu.

3.2 Zadanie 2. Na ocenę 4.0 (db)

Wcześniejsze zadanie wykorzystuje wyłącznie jedną z czterech części wyświetlacza siedmiosegmentowego. Niestety nie jest możliwe niezależne sterowanie segmentami dla poszczególnych części. Należy jednak pamiętać, że segment nie gaśnie natychmiast po odłączeniu napięcia – dioda świeci jeszcze przez pewien krótki czas. Efekt ten jest wykorzystywany do wyświetlania różnych symboli na poszczególnych częściach wyświetlacza.

Musimy cyklicznie przechodzić przez cztery stany, w każdym dając sygnał dla innej części wyświetlacza. Odpowiednią część będziemy wybierać za pomocą magistrali anodes i magistralę sevenseg będziemy ustawiali zgodnie z wartością, jaka ma być wyświetlana na danej części wyświetlacza. Jest to działanie podobne do mechanizmu stosowanego w monitorach kineskopowych.

Zmianę uaktualnianej cyfry można przeprowadzać w tym samym procesie co inkrementację licznika. Należy pamiętać o dobraniu odpowiedniej częstotliwości odświeżania poprzez uzależnienie odświeżania od właściwych bitów licznika:

```
case counter(??? downto ???) is
```



```
        when "00" => ???
        when "01" => ???
        when "10" => ???
        when "11" => ???
        when others => ???
end case;
```

Proszę zamieścić końcowy kod razem z krótkim komentarzem w sprawozdaniu.

3.3 Zadanie 3. Na ocenę 5.0 (bdb)

Proszę tak zmodyfikować zarządzanie wyświetlaczem siedmiosegmentowym, aby wyświetlały się liczby przy podstawie 10 (zamiast 16). Proszę wyświetlać 12 najstarszych bitów licznika.

Podpowiedź: VHDL nie umożliwia dzielenia przez 10 w taki sam sposób jak wspiera na przykład dodawanie. Proszę nie używać rdzeni IP w celu uzyskania odpowiedniej funkcjonalności, a jedynie poznane dotychczas konstrukcje językowe. Najprostsze rozwiązanie w ogóle nie opiera się o dzielenie.

