Systemy Czasu Rzeczywistego

„FPGA”

laboratorium: 03

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)

Początek zadania dotyczy spełniania kryteriów czasowych przez projekt. W przypadku układów pracujących synchronicznie z sygnałem zegarowym, niejednokrotnie mamy ograniczenia na częstotliwość zegara. Gdy nasz program potrzebuje dużo czasu na wykonanie fragmentu logiki zbyt duża częstotliwość zegara może powodować błędne działanie układu.

Celem jest zaprojektowanie 30-bitowego licznika pracującego z częstotliwością 250 MHz (zegar jest inkrementowany co 4 nanosekundy). Licznik taki umożliwi pomiar czasu pomiędzy dwoma zdarzeniami odległymi o mniej niż 4 sekundy z dokładnością 8 nanosekund.

Minimalny czas pojedynczego cyklu zegara jest wyliczany automatycznie. Można go obejrzeć przechodząc do Design summary, a następnie wybierając „Static timing” (Rysunek 1). Najważniejsze dane znajdują się na końcu raportu, przykładowo:

```
Data Sheet report:
 -----------------
 All values displayed in nanoseconds (ns)

 Clock to Setup on destination clock clk
 ---------------+---------+---------+---------+---------+
                | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
 Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
```



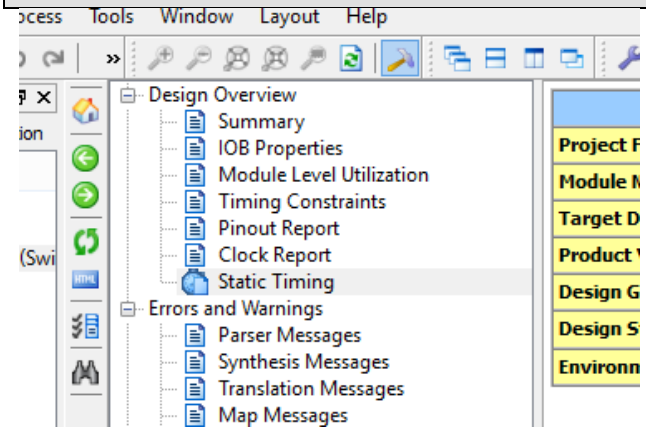
```
---------------+---------+---------+---------+---------+
clk            |    4.670|         |         |         |
---------------+---------+---------+---------+---------+


Timing summary:
---------------

Timing errors: 9  Score: 2678  (Setup/Max: 2678, Hold: 0)

Constraints cover 465 paths, 0 nets, and 60 connections

Design statistics:
   Minimum period:   4.670ns{1}   (Maximum frequency: 214.133MHz)
```



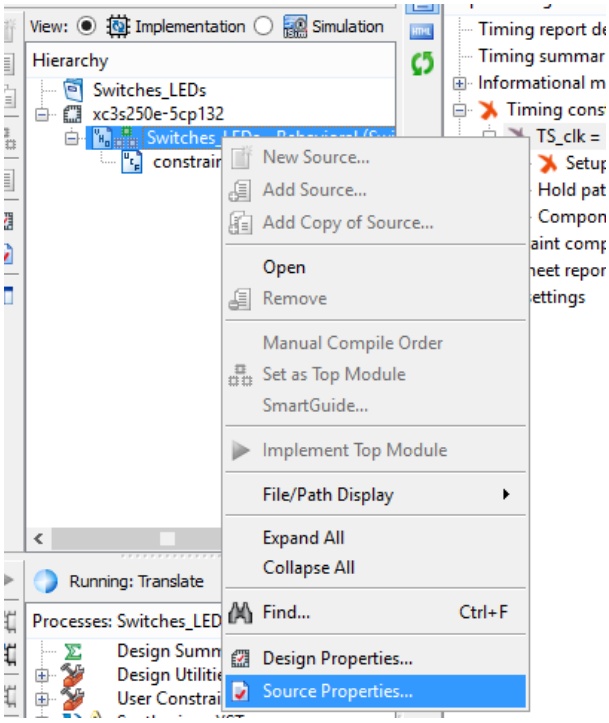
Rysunek 1: Raport dotyczący ograniczeń czasowych

Zapisz w raporcie minimalny okres (*period*) oraz maksymalną częstotliwość. Wykonaj to dla dwóch dostępnych prędkości FPGA (*speed grade*). Ustawienie to można zmienić wybierając opcję jak na Rysunek 2.

Możliwe jest nałożenie ograniczeń czasowych na projekt. Na przykład, aby zażądać pracę z zegarem poniżej 4 nanosekund możemy umieścić poniższy kod w pliku z ograniczeniami:

```
NET "clk" TNM_NET = clk;
TIMESPEC TS_clk = PERIOD "clk" 4 ns HIGH 50%;
```

Proszę sprawdzić, czy dodanie takiego ograniczenia zmienia minimalny okres wyliczany w raporcie i zamieścić wyniki w raporcie (dla obu ustawień *speed grade*).



Rysunek 2: Zmiana właściwości źródła

Jedną z możliwości zwiększenia dopuszczalnej częstotliwości jest podział licznika na dwie części po 15 bitów:

```
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
        );
end Switches_LEDs;

architecture Behavioral of Switches_LEDs is
   signal counter : STD_LOGIC_VECTOR(29 downto 0) := (others => '0');
   signal incHighNext : STD_LOGIC := '0';
begin

 LEDs <= counter(29 downto 22);

clk_proc: process(clk)
 begin
    if rising_edge(clk) then
      counter(29 downto 15) <= counter(29 downto 15)+incHighNext;
```



```
    if counter(14 downto 0) = "111111111111110" then
      incHighNext <= '1';
    else
      incHighNext <= '0';
    end if;

    counter(14 downto 0) <= counter(14 downto 0)+1;
  end if;
 end process;
end Behavioral;
```

Proszę sprawdzić o ile zmniejszył się minimalny okres. Proszę zmodyfikować program tak, aby licznik był podzielony na dwie nierówne części: 20 i 10 oraz (w drugiej wersji) 10 i 20 bitów.

3.2 Zadanie 2. Na ocenę 4.0 (db)

Przedmiotem tego zadania będzie dodanie kolejnego modułu do projektu. Używana terminologia:

- Encja definiuje wewnętrzny widok interfejsu modułu:

```
entity mymodule is
  Port ( input1 : in  STD_LOGIC_VECTOR (3 downto 0);
       output1 : out  STD_LOGIC_VECTOR (3 downto 0));
end mymodule;
```

- Architektura definiuje jak dany komponent działa – opisuje wewnętrzne sygnały i logikę działania:

```
architecture Behavioral of mymodule is
begin
  output1 <= input1;
end Behavioral;
```

- Komponent definiuje zewnętrzny widok interfejsu modułu i pojawia się w modułach wykorzystujących dany komponent (komponenty mogą być umieszczane w deklaracjach architektury):

```
COMPONENT mymodule
PORT(
  input1 : IN std_logic_vector(3 downto 0);
  output1 : OUT std_logic_vector(3 downto 0));
END COMPONENT;
```

- Instancja opisuje podłączenie komponentu w module (instancje można umieszczać w ciele architektury):

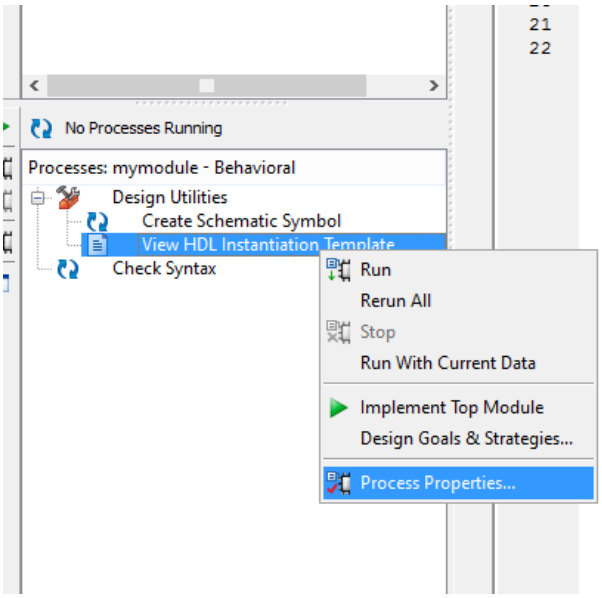
```
  Inst_mymodule: mymodule PORT MAP(
    input1 =>  input_signal1,
    output1 => output_signal1
```



```
);
```

Postępując zgodnie z instrukcjami z pierwszego laboratorium z FPGA utwórz nowy projekt z modułem Switches_LEDs z przełącznikami i diodami. Dodaj moduł o nazwie mymodule mający pojedyncze wejście (o nazwie input1) i pojedyncze wyjście (o nazwie output1), oba będące czteroelementowymi magistralami.

W hierarchii modułów zaznacz nowo utworzony moduł. W oknie poniżej odnajdź opcję „View HDL Instantiation Template” (patrz Rysunek 3). W otwartym oknie zmień język na VHDL. Po dwukrotnym wciśnięciu „View HDL Instantiation Template” szablon instancjacji modułu zostanie wyświetlony. Szablon ten może zostać wykorzystany do umieszczania modułu w innych modułach.



Rysunek 3: Ustawianie opcji modułu

Utwórz teraz nowy moduł o nazwie counter30 – trzydziestobitowy licznik. Powinien posiadać następujące sygnały zewnętrzne:

- clk : in STD_LOGIC
- enable : in STD_LOGIC
- count : out STD_LOGIC_VECTOR(29 downto 0)

Jeśli enable jest ustawione na 1, co każdy takt zegara clk sygnał ustawiany na count ma się zwiększać o 1. Jeśli enable jest ustawione na 0, to count powinien być ustawiany na 0. Sygnał zegarowy ma mieć częstotliwość 50 MHz.

Umieść dwie instancje modułu counter30 w module Switches_LEDs. Podłącz sygnał zegarowy, sygnały enable połącz z przełącznikami 0 i 1, zaś po cztery najstarsze bity wyświetlaj na czterech sąsiednich diodach LED. Po przygotowaniu projektu sprawdź, czy działa on w układzie FPGA.

Zamieść przygotowany kod (z komentarzami) w sprawozdaniu. Dołącz schematy RTL i technologiczny.

3.3 Zadanie 3. Na ocenę 5.0 (bdb)

Jaki najdłuższy licznik można zrobić zachowując warunek, aby maksymalna częstotliwość pracy była nie mniejsza niż 200 MHz? Na ocenę 4,5 należy uzyskać licznik przynajmniej 130-bitowy, zaś na ocenę 5,0 przynajmniej 150-bitowy.

