Systemy Czasu Rzeczywistego

„FPGA”

laboratorium: 01

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

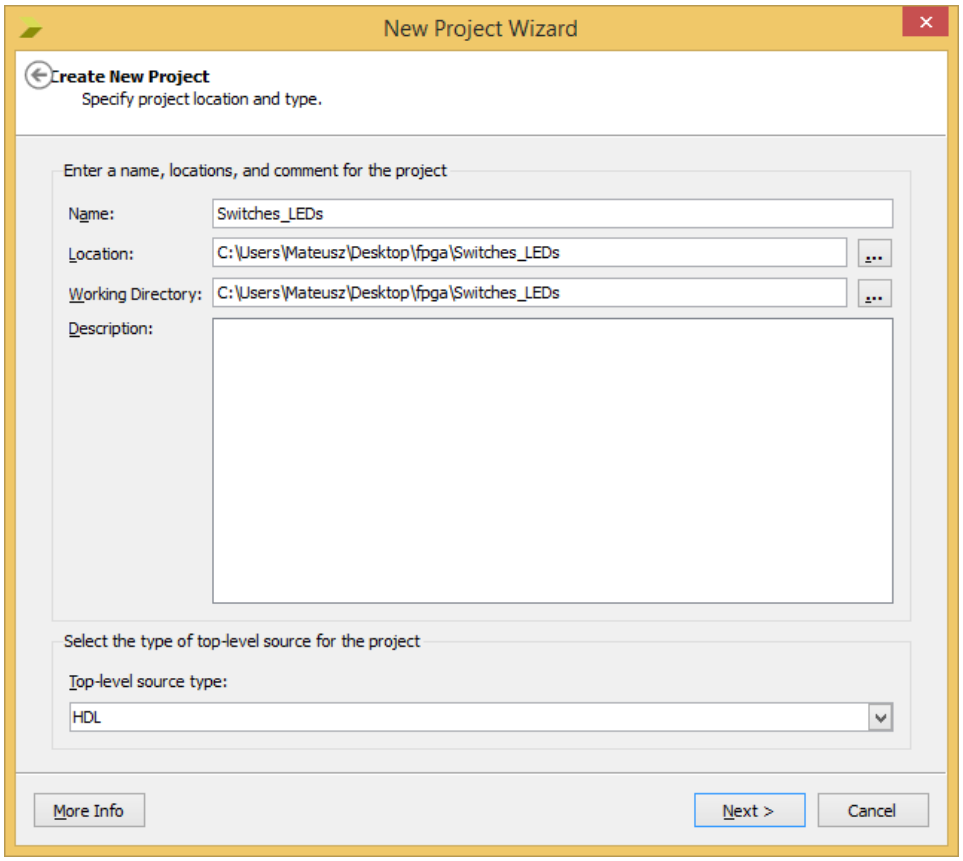
3 Przebieg laboratorium

3.1 Zadanie 1. Na ocenę 3.0 (dst)

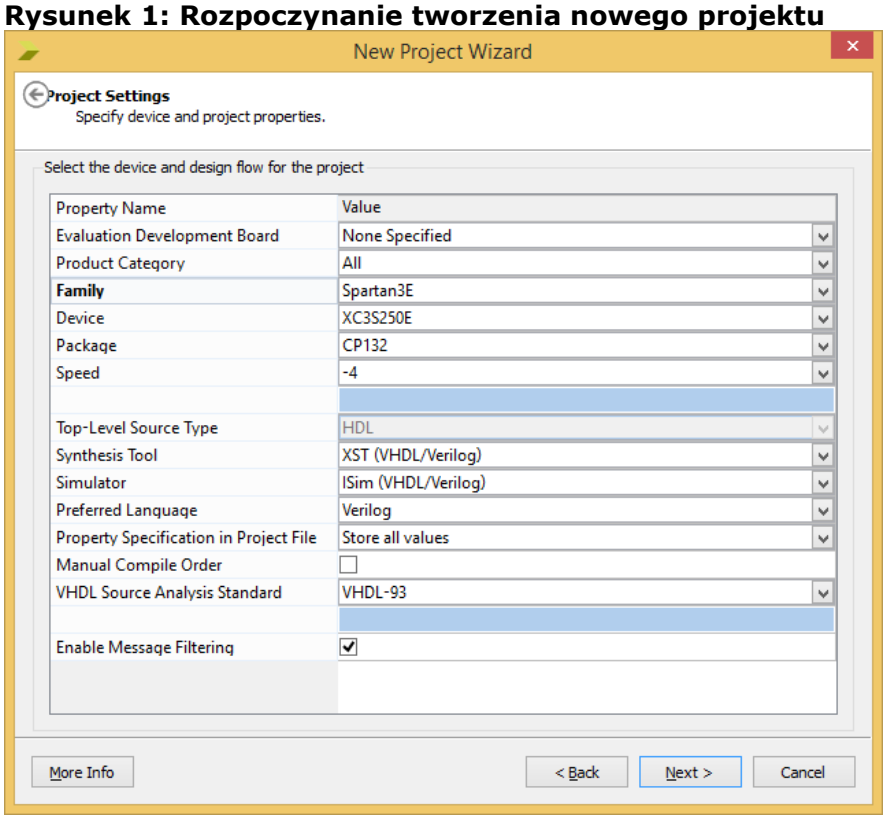
Otwórz program ISE Design Suite 14.7. Rozpocznij tworzenie nowego projektu. Nazwij go Switches_LEDs. Jako położenie w systemie plików należy wskazać specjalnie utworzony na pulpicie folder o nazwie „fpga”. Ekran nowego projektu powinien wyglądać jak na Rysunek 1.

Następnie wciśnij „dalej” i ustaw szczegóły nowego projektu. Wybierz rodzinę z której pochodzi układ FPGA (Spartan3E) oraz konkretne urządzenie (XC3S250E), rodzaj obudowy i prędkość (Rysunek 2). Następnie zakończ tworzenie nowego projektu.

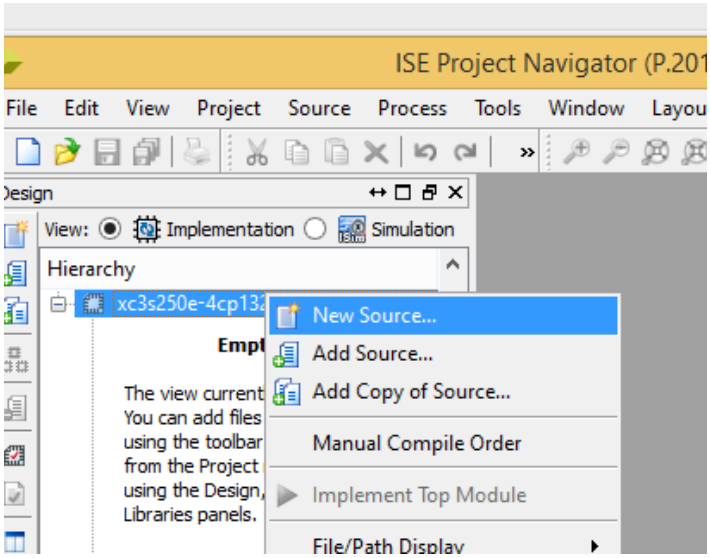
Projekt rozpoczniemy od dodania pierwszego pliku źródłowego. Obrazuje to Rysunek 3. Należy wybrać typ źródła „VHDL Module”, a następnie wpisać nazwę pliku „Switches_LEDs”. Ustawienia portów (wejść i wyjść modułu) prezentuje Rysunek 4.



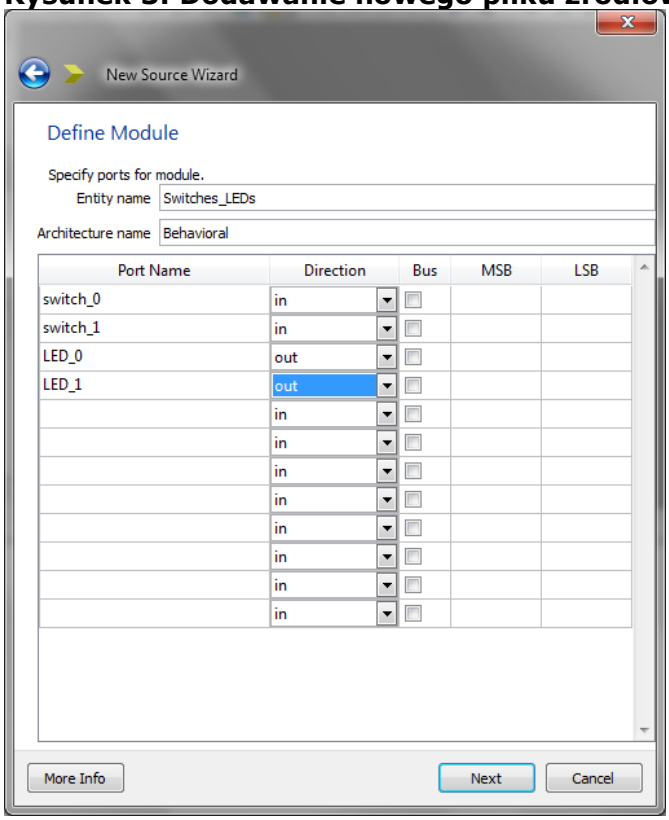
Rysunek 1: Rozpoczynanie tworzenia nowego projektu



Rysunek 2: Ustawienia projektu



Rysunek 3: Dodawanie nowego pliku źródłowego



Rysunek 4: Ustawienia portów

Teraz należy przejść do edycji pliku źródłowego. Domyślną treść pliku należy zastąpić następującą:

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

entity Switches_LEDs is
  Port ( switch_0 : in  STD_LOGIC;
       switch_1 : in  STD_LOGIC;
```



```
        LED_0 : out  STD_LOGIC;
        LED_1 : out  STD_LOGIC);
end Switches_LEDs;

architecture Behavioral of Switches_LEDs is
begin

end Behavioral;
```

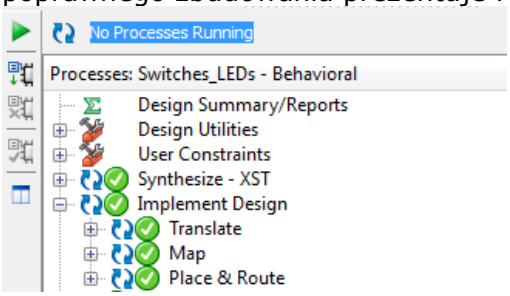
Powyższy kod jest deklaracją encji o nazwie Switches_LEDs oraz pustym opisem behawioralnym zadeklarowanej encji. Encja Switches_LEDs ma dwa porty wejściowe i dwa wyjściowe. Typ STD_LOGIC określa wartości, jakie może przyjmować port: logiczne zero, logiczną jedynkę lub jeden z kilku innych stanów.

Dodaj teraz implementację architektury encji Switches_LEDS. LED_0 ma być jedynką gdy switch_0 jest jedynką oraz LED_1 ma być jedynką gdy switch_1 jest jedynką. Realizuje to poniższy kod (należy go umieścić między begin i end opisu architektury):

```
 LED_0 <= switch_0;
 LED_1 <= switch_1;
```

Operator <= jest odpowiednikiem operatora przypisania.

W tym momencie możliwe jest już zbudowanie projektu. W tym celu należy nacisnąć zielony trójkąt na górnym menu opisany „Implement top module”. Wynik poprawnego zbudowania prezentuje Rysunek 5.



Rysunek 5: Poprawne zbudowanie projektu

Projekt nie jest jednak kompletny, gdyż nie podaliśmy w jaki sposób nasz moduł ma się łączyć z diodami i przełącznikami na płytce. Jest to opisywane przez pliki ograniczeń.

Dodaj nowy plik, tym razem typu „Implementation constraints file” i nazwij go constraints. Wklej do niego następującą zawartość:

```
NET switch_1 LOC = "L3";
NET switch_0 LOC = "P11";
NET LED_1    LOC = "M11";
NET LED_0    LOC = "M5";
```

Nazwy wpisywane po znaku równości są specyficzne dla konkretnej płytki z układem FPGA.

Ostateczny plik programowania układu FPGA jest tworzony po dwukrotnym kliknięciu na „Generate programming file” (w menu widocznym na Rysunek 5). Uwaga: aby to menu było widoczne, w spisie plików projektu musi być zaznaczony plik Switches_LEDs.vhd.

Środowisko ISE pozwala na obejrzenie wygenerowanego programu na kilka sposobów. Rozwiń menu Synthesize – XST oraz Place & Route. W sprawozdaniu krótko

opisz i zamieść zrzuty ekranu z widoków „RTL schematic”, „Technology schematic” oraz „Routed design”. Do schematów RTL i Technology wybierz wszystkie elementy z projektu. Pomocne mogą być strony internetowe:

- http://www.xilinx.com/itp/xilinx10/isehelp/ise_p_viewing_rtl_schematic_xst.htm
- http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/ise_p_viewing_a_technology_schematic_xst.htm
- http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/pp_p_process_view_edit_routed_design.htm

Kolejnym elementem laboratorium jest uruchomienie programu na karcie Basys2. Należy podłączyć kartę dołączonym kablem USB, ustawić przełącznik POWER (niedaleko gniazda USB) na ON i włączyć program Digilent Adept. W prawym górnym rogu powinna być wybrana opcja „Basys2”. W zakładce „Config” należy wcisnąć górny przycisk „Browse” i w katalogu z projektem (katalog był wybierany przy tworzeniu projektu) wybrać plik switches_leds.bit. Reprezentuje on bitstream służący do programowania układów FPGA. Urządzenie zostanie zaprogramowane po naciśnięciu przycisku Program (należy zignorować ostrzeżenie o zegarze JTAG).

Po poprawnym zaprogramowaniu urządzenia dwa z przełączników powinny włączać i wyłączać znajdujące się obok nich diody LED. W sprawozdaniu proszę zaznaczyć, które to były przyciski.

Ostatnim etapem zadania będzie wykorzystanie magistral. Manipulowanie pojedynczymi sygnałami jest niewygodne w wielu zastosowaniach, dlatego grupuje się w magistrale przesyłające wiele sygnałów jednocześnie. Wykorzystuje się do tego typ STD_LOGIC_VECTOR. Zmień program Switches_LEDs tak, aby wykorzystywał magistrale jak w poniższym przykładzie:

```
entity Switches_LEDs is
  Port ( switches : in  STD_LOGIC_VECTOR(1 downto 0);
         LEDs     : out STD_LOGIC_VECTOR(1 downto 0));
end Switches_LEDs;

architecture Behavioral of Switches_LEDs is
begin
  LEDs <= switches;
end Behavioral;
```

Dostęp do pojedynczego sygnału odbywa się z użyciem nawiasów okrągłych:

```
LEDs(0) <= switches(1)
```

W efekcie konieczne jest przerobienie pliku z ograniczeniami. Poniższy kod umożliwi dostęp do ośmiu przełączników i diod LED:

```
NET LEDs(7) LOC = "G1";
NET LEDs(6) LOC = "P4";
NET LEDs(5) LOC = "N4";
NET LEDs(4) LOC = "N5";
NET LEDs(3) LOC = "P6";
NET LEDs(2) LOC = "P7";
NET LEDs(1) LOC = "M11";
NET LEDs(0) LOC = "M5";
```



```
NET switches(7) LOC = "N3";
NET switches(6) LOC = "E2";
NET switches(5) LOC = "F3";
NET switches(4) LOC = "G3";
NET switches(3) LOC = "B4";
NET switches(2) LOC = "K3";
NET switches(1) LOC = "L3";
NET switches(0) LOC = "P11";
```

W języku VHDL operacje bitowe zapisuje się przy użyciu odpowiednich operatorów. Są to AND, OR, XOR, NOT, NAND i NOR. Dla przykładu, zapalenie diody 1 wtedy i tylko wtedy gdy oba przełączniki są włączone wyglądałoby następująco:

```
LED_1 <= switch_0 AND switch_1
```

Zmodyfikuj program tak, aby poszczególne diody prezentowały:

- LEDs(0): Czy oba przełączniki (0 i 1) są włączone.
- LEDs(1): Czy oba przełączniki (0 i 1) są wyłączone.
- LEDs(2): Czy którykolwiek z przełączników (0, 1, 2, 3) jest włączony.
- LEDs(3): Czy przełącznik 0 jest wyłączony.
- LEDs(4): Czy włączony jest tylko jeden z przełączników (0, 1).
- LEDs(5): Czy włączony jest tylko jeden z przełączników (0, 1, 2).

Z pozostałych dwóch diod jedna ma być zawsze zapalona (LEDs(6)), a druga zawsze zgaszona (LEDs(7)).

Połączenie dwóch sygnałów w magistralę (lub dwóch magistral) służy operator ‘&’ (ampersand). Sygnał (magistralę) o stałej wartości można zapisać jako:

```
sygnal <= ‘0’
magistrala <= "010”
```

3.2 Zadanie 2. Na ocenę 4.0 (db)

Porównaj schematy technologiczne dla dwóch par równoważnych wyrażeń. Niech jedną z tych par będzie:

```
LED_1 <= switch_0 AND switch_1
LED_1 <= NOT( NOT(switch_0) OR (NOT switch_1))
```

Zaimplementuj operację OR przy użyciu wyłącznie operatora NOR oraz operację XOR tylko z użyciem operacji OR, AND i NOT. Sprawdź, czy odpowiednie programy poprawnie wykonują się w układzie FPGA.

Przygotuj program, który (korzystając z magistral) będzie wyświetlał koniunkcję(na LEDs(3 downto 0)) i alternatywę (na LEDs(7 downto 4)) stanów przełączników switch(i) i switch(i+4).

3.3 Zadanie 3. Na ocenę 5.0 (bdb)

Przygotuj program zapalający tyle **kolejnych** diod, ile jest włączonych przełączników. Można ograniczyć się do czterech przełączników i diod.

