LABORATORIUM – ELEKTRONIKA

Projektowanie koderów, transkoderów i dekoderów w języku VHDL

1. Cel ćwiczenia

W ćwiczeniu student projektuje i implementuje w strukturze układu FPGA (*Field Programmable Gate Array*) układ przetwarzający kod binarny na kod wyświetlacza siedmiosegmentowego. W trakcie ćwiczenia student poznaje też podstawy programowania układów w języku VHDL (*Very High Speed Integrated Circuits Hardware Description Language*).

2. Wstęp

Przesyłanie, przetwarzanie i obróbka informacji w systemach cyfrowych jest wykonywana w oparciu o reprezentację odpowiednich zmiennych w postaci kodów liczbowych. Wybór kodu do zastosowania w danej aplikacji zależy od potrzeb, szczególnie chodzi o ułatwienie wykonywania operacji na zmiennych w tym układzie oraz zapewnienie ich prawidłowości przy występowaniu zakłóceń i uszkodzeń w układzie. Przy wyborze kodu umożliwiającego osiągnięcie minimum kosztów układu realizującego daną operację (np. arytmetyczną) należy również uwzględnić problemy konwersji tego kodu na inny/inne stosowane w tym systemie. Do tego celu stosuje się właśnie układy koderów, dekoderów i transkoderów, które to układy mają za zadanie przeliczyć dany kod binarny na inny.

Innym typowym zastosowaniem tego typu układów jest przetworzenie informacji w danym kodzie na kod który jest „czytelny z zewnętrz". Typowym układem tego typu jest US7447, który ma za zadanie przeliczać kod binarny na kod wyświetlacza siedmiosegmentowego. Innym znanym układem scalonym podobnego zastosowania jest US7442 przeliczający kod binarny na kod 1 z 10. Układ projektowany w ramach laboratorium ma działać podobnie jak US 7447, czyli będzie on przetwarzał 4-bitowy kod binarny na wejściu na 7-bitowy kod, który po podaniu na wejścia wyświetlacza siedmiosegmentowego będzie wyświetlał liczby w pełnym zakresie 4 bitów, czyli od 0 do 15 (tzn. 0, 1, … , E, F).

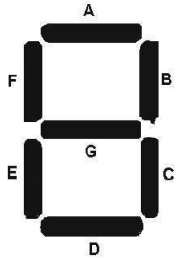
3. Wykonanie ćwiczenia.

CZĘŚĆ A – TABELA STANÓW

3.1. Wypełnić Tabelę 1 w protokole. Jest to tabela stanów projektowanego transkodera, czyli tabela zawierająca wszystkie możliwe kombinacje wejść (od 0000 do 1111, innymi słowy binarnie policzone od 0 do 15) i odpowiadające im stany wyjść. Wykorzystywany na laboratorium wyświetlacz działa w taki sposób, że zapala segment na który podana jest logiczna jedynka.

A zatem, przykładowo:

- pierwszy wiersz tabeli odpowiada kombinacji wejść 0000, czyli po prostu dziesiętnemu 0,
- by wyświetlić na wyświetlaczu 0, należy zaświecić wszystkie segmenty poza G,
- zatem na wyjścia należy podać kombinację: 0111111 (G=0, B=1, C=1, itd.).



Analogicznie należy wypełnić pozostałe wiersze tabeli stanów.

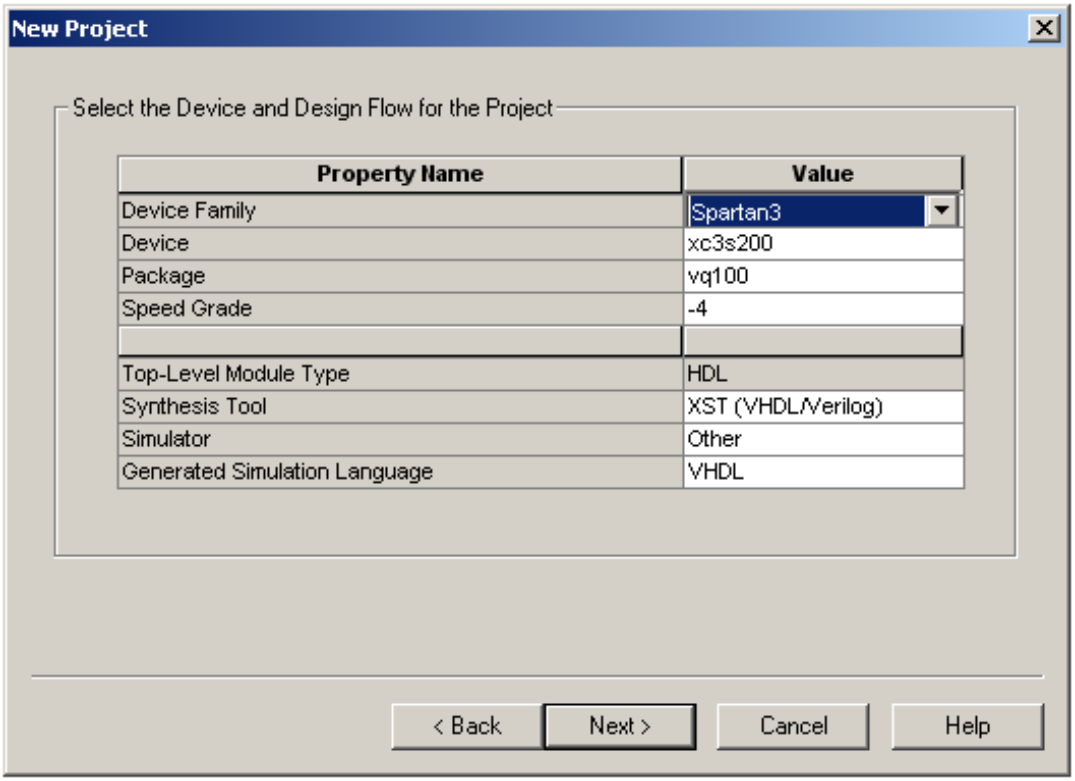
Po wykonaniu tej czynności można przejść do CZĘŚCI B ćwiczenia.

CZĘŚĆ B – implementacja projektu w strukturze FPGA

3.2. Uruchomić aplikację **Xilinx - Project Navigator**. Jest to aplikacja firmy Xilinx umożliwiająca projektowanie, symulację i implementację układów logicznych w strukturach układów CPLD i FPGA.

3.3. Stworzyć nowy projekt (***File -> New Project***). Nadać mu nazwę (**w nazwach NIE używać spacji**), wskazać lokalizację (można użyć domyślnej) i wybrać sposób realizacji projektu (***Top-Level Module Type***) – **HDL**.

3.4. W następnym oknie wybrać rodzaj programowanego układu logicznego. Powinien być to układ wykorzystywany na ćwiczeniu, zgodnie ze screenem (wszystkie widoczne informacje są dostępne na obudowie układu FPGA):



3.5. W kolejnym oknie stworzyć nowe źródło (***New Source***) do projektu. Nadać mu nazwę i wybrać typ – **VHDL Module**.

3.6. Kolejne okno służy do definiowania wejść i wyjść układu. Projektowany układ powinien zawierać jeden 4 bitowy wektor wejść (bo kod wejściowy naszego transkodera jest 4-bitowym kodem binarnym), deklarowany następująco:

Port Name	Direction	MSB	LSB
wejscia	in	3	0

Operacja jak widać sprowadza się do wpisania nazwy sygnału w kolumnie ***Port Name***, wskazania, czy definiujemy wejście (***in***) czy wyjście (***out***) i zadeklarowania wielkości wektora (w tym przypadku od 0 do 3, więc razem 4). Analogicznie do wektora wejść należy zadeklarować 7 bitowy wektor wyjść (bo wyświetlacz ma 7 segmentów) i jeden bit (także wyjściowy) do zasilenia wyświetlacza. Przy deklarowaniu tego ostatniego, jako, że nie jest to wektor ale sygnał, kolumny MSB (*Most Significant Bit*) i LSB (*Least Significant Bit*) powinny pozostać puste.

3.7. Kolejne okna przeklikujemy, zgadzając się na ewentualne pytania programu. Ostatecznie powinien się nam ukazać automatycznie utworzony szkielet programu, wyglądający mniej więcej tak:

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

-- Uncomment the following lines to use the declarations that are
-- provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

entity transkoder is
   Port ( wejscia : in std_logic_vector(3 downto 0);
          wyjscia : out std_logic_vector(6 downto 0);
          zasil : out std_logic);
end transkoder;

architecture Behavioral of transkoder is
-- (1) TUTAJ DEKLARUJE SIĘ SYGNAŁY WEWNĘTRZNE
begin
-- (2) TUTAJ TWORZY SIĘ WŁAŚCIWY PROGRAM
end Behavioral;
```

Jak widać, w programie pojawiły się (zaznaczone na niebiesko) deklarowane wcześniej wejścia i wyjścia układu. We wskazanych na czerwono miejscach tworzy się właściwy program, czyli **(1)** deklaruje ewentualne sygnały działające jedynie wewnątrz programu (tzn. niewyprowadzane na zewnątrz) oraz **(2)** określa jego działanie.

3.8. Przystąpmy więc do tworzenia właściwego programu. W naszym transkoderze nie będzie sygnałów działających jedynie wewnątrz programu, więc nasze działania będą się toczyć jedynie we fragmencie kodu **(2)**.
Opis działania dekodera, możliwy do wykonania na kilka sposobów, sprowadza się do wskazania mu jaką kombinację wyjść ma wystawiać w zależności od kombinacji wejść. Takie zachowanie można mu narzucić na przykład przypisaniem rozpoczynającym się następująco:

```
wyjscia <=
"0111111" when (wejscia = "0000") else
```

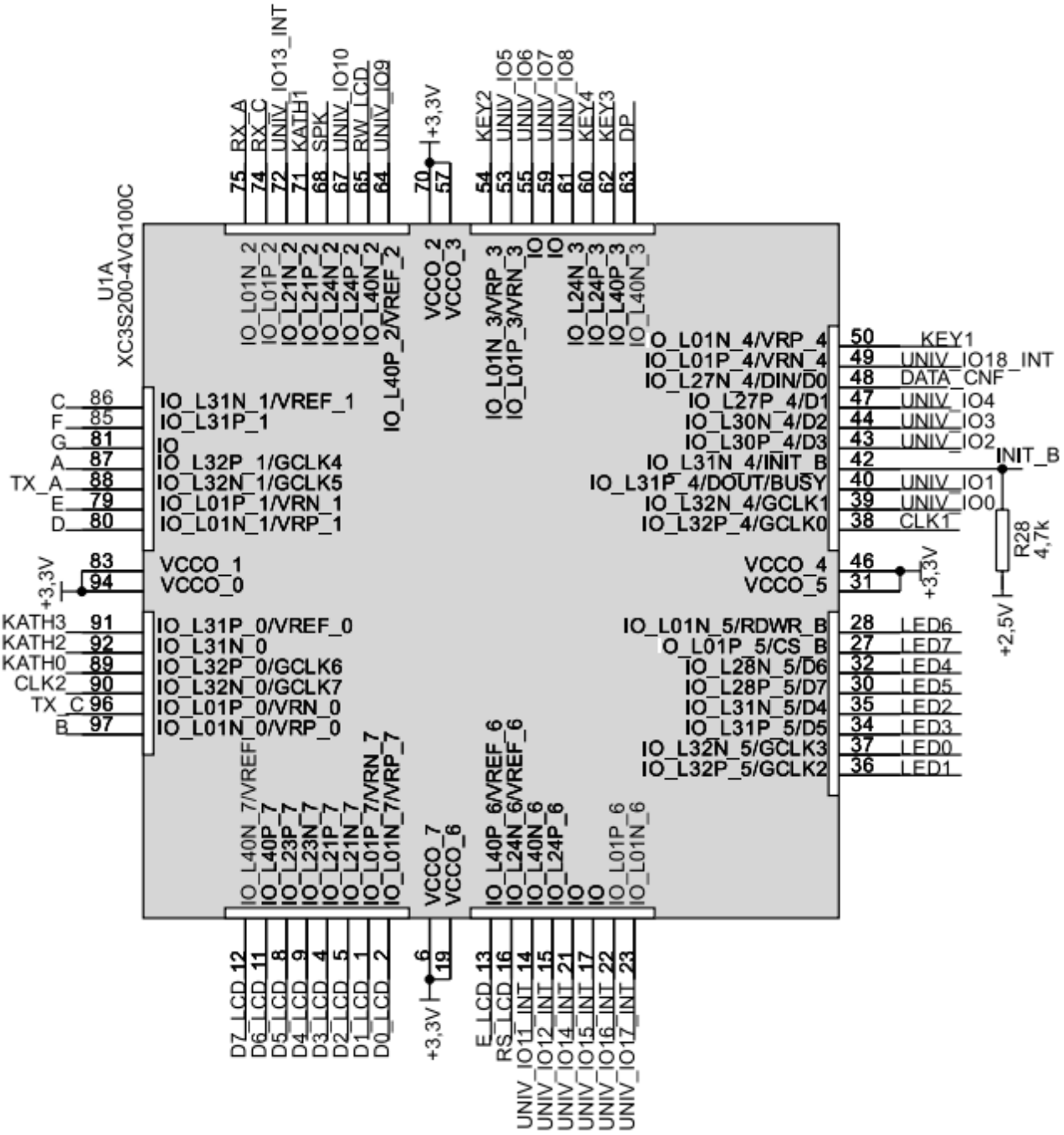
która to kombinacja, zgodnie z wcześniejszym przykładem, powinna na wyświetlaczu siedmiosegmentowym pokazywać 0 (tj. wygaszać jedynie segment G) dla kombinacji wejść 0000.
W kolejnych wierszach wpisujemy pozostałe 15 zależności wyjść od wejść.
Ostatniego wiersza nie kończymy słowem „else" lecz średnikiem.

Ostatnim co pozostało do zrobienia, jest zasilenie wyświetlacza. W tym celu należy wystawić stan wysoki (logiczną jedynkę) na bit zasilający. Wystarczy do tego prosta instrukcja (słowo „zasil" oczywiście jest nadawaną wcześniej nazwą bitu zasilającego wyświetlacz):

```
zasil <= '1';
```

3.9. I to jest koniec właściwego programu. Kolejnym krokiem jest przypisanie wejść i wyjść wykonanego układu do nóżek układu scalonego. W tym celu należy w oknie po lewej stronie dwukliknąć w opcję **Assign Package Pins**. Jeżeli w programie nie ma błędów, to powinno otworzyć się okno Xilinx PACE i topologia wykorzystywanego układu FPGA. Po lewej stronie (**Design Object List**) widać wcześniej deklarowane wejścia/wyjścia układu.

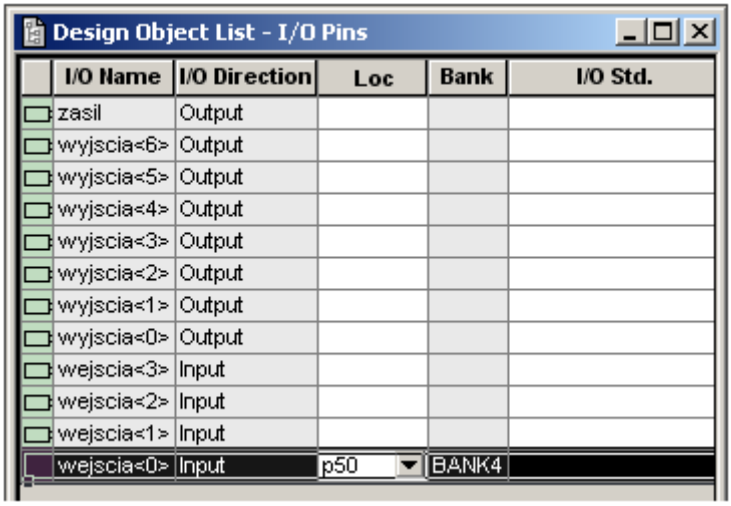
Wykorzystywane piny układu przypisać zgodnie z rysunkiem poniżej:



Oznaczenia są następujące:

- wejścia - przyciski – KEY4, KEY3, KEY 2, KEY1 (gdzie: wejścia<3> to KEY4, wejścia<2> to KEY3 itd.),
- wyjścia – G, F, E, D, C, B, A (gdzie: wyjscia<6> to G, wyjscia<5> to F itd.),
- pin zasilający wyświetlacz – dowolny z pinów KATH0, KATH 2 lub KATH3.

Piny przypisuje się wpisując bezpośrednio nóżkę układu scalonego, której chcemy przypisać daną funkcję. Przykładowo: chcąc przypisać najmniej znaczący bit wejściowy (wejścia<0>) do przycisku KEY1, na rysunku powyżej sprawdzamy, że KEY1 jest podpięty do nóżki (pinu) 50 układu FPGA, więc w odpowiednim miejscu wpisujemy p50, jak na rysunku poniżej, następnie analogicznie przypisujemy wszystkie pozostałe piny.



3.10. Po przypisaniu pinów można przystąpić do implementacji projektu w strukturze FPGA i sprawdzenia poprawności działania transkodera.

W tym celu należy zgłosić prowadzącemu gotowość do zaprogramowania układu.

Jeżeli zaimplementowany w strukturze FPGA program działa zgodnie z założeniami, można przejść do CZĘŚCI C ćwiczenia.

CZĘŚĆ C – projekt i implementacja projektu w strukturze FPGA

4.1. Wykorzystując zawarte w protokole wskazówki, wypełnić Tabelę 2, tworząc układ mnożący dwie liczby 2-bitowe.

4.2. Zaimplementować zaprojektowany układ w strukturze FPGA. Wszystkie czynności wykonać analogicznie jak w CZĘŚCI B ćwiczenia, oczywiście uwzględniając specyfikę nowego projektu (np. fakt, że wyjście nie jest 7-bitowe, lecz 4-bitowe). Jako wyjścia tym razem wykorzystać 4 diody LED (LED0 - LED7 na rysunku ze strony 4 instrukcji), wejściami ponownie mogą być przyciski – KEY4, KEY3, KEY 2, KEY1.

5. Literatura

[1] Borzdyński J.: *Kurs CPLD*, Elektronika dla Wszystkich 02/2009-04/2009

[2] Głocki W., Grabowski L., „Pracownia podstaw techniki cyfrowej”, WSiP, Warszawa 1998

[3] Pawluczuk A.: *Układy programowalne dla początkujących*, btc, Legionowo 2010.

[4] Sasal W., „Układy scalone serii UCA64/UCY74, parametry i zastosowania”, WKiŁ, Warszawa 1985

[5] Skahill K.: *Język VHDL - projektowanie programowalnych układów logicznych*, WN-T, Warszawa 2004

[6] Zwoliński M.: *Projektowanie układów cyfrowych z wykorzystaniem języka VHDL*, WKiŁ, Warszawa 2007

Opracowanie ćwiczenia: Seweryn Lipiński