LABORATORIUM - ELEKTRONIKA
Układy kombinacyjne

1. Cel ćwiczenia

Celem ćwiczenia jest zaprojektowanie i praktyczna realizacja prostego układu kombinacyjnego. Student uczy się projektować układy kombinacyjne wykorzystując różne bramki logiczne i oceniać stopień złożoności układu cyfrowego według różnych kryteriów.

Instrukcja została napisana w taki sposób, by student potrafił zbudować układ logiczny zarówno w sposób klasyczny (czyli wykorzystując niskiej skali integracji układy scalone serii 74xx(x) lub 40xx(x)), jak i z wykorzystaniem najnowocześniejszych programowalnych układów logiczne (FPGA – ang. *Field Programmable Gate Array*). W trakcie ćwiczenia układ jest realizowany tym drugim sposobem - z użyciem układu FPGA z rodziny Spartan 3 firmy Xilinx.

2. Trochę podstaw

2.1. Synteza układu kombinacyjnego na bramkach NAND - wersja pełna, czyli bez minimalizacji.

Załóżmy, że dany jest układ cyfrowy, opisany następującą tabelą stanów:

l.p.	A	B	C	OUT
0	0	0	0	**1**
1	0	0	1	**1**
2	0	1	0	**0**
3	0	1	1	**x**
4	1	0	0	**x**
5	1	0	1	**0**
6	1	1	0	**1**
7	1	1	1	**0**

ⓘ *Znak X pojawiający się w tabeli oznacza iż określona kombinacja wejść nie ma prawa w układzie wystąpić lub że z pewnych powodów nie interesuje nas wyjście układu dla danej kombinacji wejść.*

Na podstawie danej tabeli stanów uzyskujemy opis kanoniczny projektowanego układu w postaci sumacyjnej:

$$OUT = \sum(0,1,6\ (3,4))$$

i opis kanoniczny w zapisie Boola (w tym przypadku z postaci sumacyjnej):

$$OUT = \overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C},$$

z którego możemy bezpośrednio uzyskać schemat logiczny układu przy wykorzystaniu bramek NOT, AND i OR (patrz punkt 3.1 instrukcji). **Z tych bramek logicznych najczęściej realizuje się układy logiczne na układach programowalnych (FPGA/CPLD).** W przypadku gdy układ kombinacyjny chcemy zbudować na układach scalonych niskiej skali integracji, to zwykle układ projektuje się tak, by korzystać z jednego rodzaju bramek, najczęściej NAND i ewentualnie NOT.

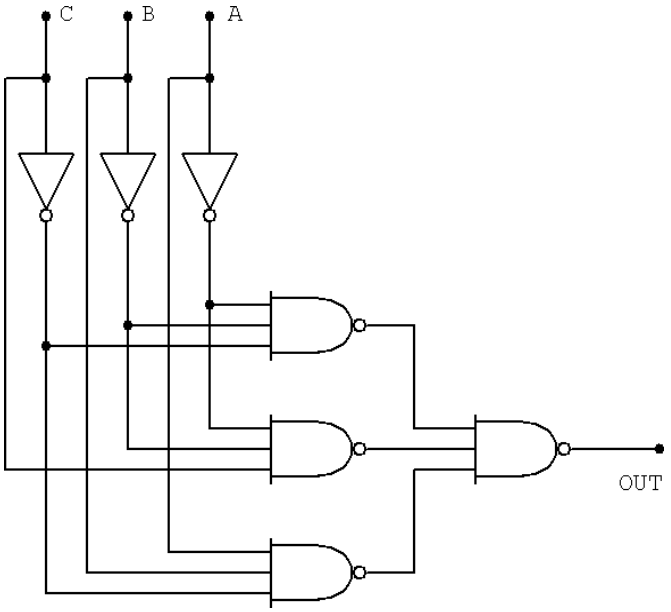
Aby przekształcić powyższą postać funkcji do postaci bezpośrednio odwzorowywalnej przy użyciu bramek NAND, należy uzyskaną wcześniej postać funkcji podwójnie zanegować i następnie skorzystać z praw de Morgana. Mają one postać:

- $\overline{a_1} + \overline{a_2} = \overline{a_1 \cdot a_2}$,
- $\overline{a_1} \cdot \overline{a_2} = \overline{a_1 + a_2}$.

W rozpatrywanym przypadku przekształcenie to wygląda następująco:

$$OUT = \overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} = \overline{\overline{\overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C}}} = \overline{\overline{\overline{A} \cdot \overline{B} \cdot \overline{C}} \cdot \overline{\overline{A} \cdot \overline{B} \cdot C} \cdot \overline{A \cdot B \cdot \overline{C}}}$$

Jak widać, aby zrealizować powyższą postać funkcji potrzeba 4 bramek NAND 3-wejściowych i 3 bramek NOT (Rysunek 1).



Rysunek 1. Realizacja podstawowej funkcji opisującej układ przy użyciu bramek NAND (i NOT).

Należy pamiętać o tym, że układy scalone zawierające bramki NAND są produkowane w następujących konfiguracjach:

- 4 bramki NAND 2-wejściowe (np. US 7400 w technologii TTL lub 4011 w technologii CMOS),
- 3 bramki NAND 3-wejściowe (np. US 7410 w technologii TTL lub 4023 w technologii CMOS),
- 2 bramki NAND 4-wejściowe (np. US 7420 w technologii TTL lub 4012 w technologii CMOS),
- 1 bramka NAND 8-wejściowa (np. US 7430 w technologii TTL lub 4068 w technologii CMOS).

Bramki negujące można znaleźć w układzie scalonym zawierającym 6 bramek NOT (np. US 7404 w technologii TTL lub 4069 w technologii CMOS).

Warto zawsze sprawdzić tzw. złożoność układową zaprojektowanej postaci funkcji. Przez złożoność układową rozumie się:

- LP - liczbę połączeń (liczba wejść wszystkich użytych elementów),
- LB - liczbę użytych elementów (bramek logicznych),
- LS - liczbę układów scalonych potrzebnych do realizacji układu.

Oczywistym jest, że dąży się do tego aby złożoność (jakkolwiek określona) była jak najmniejsza.

W powyższym przypadku:

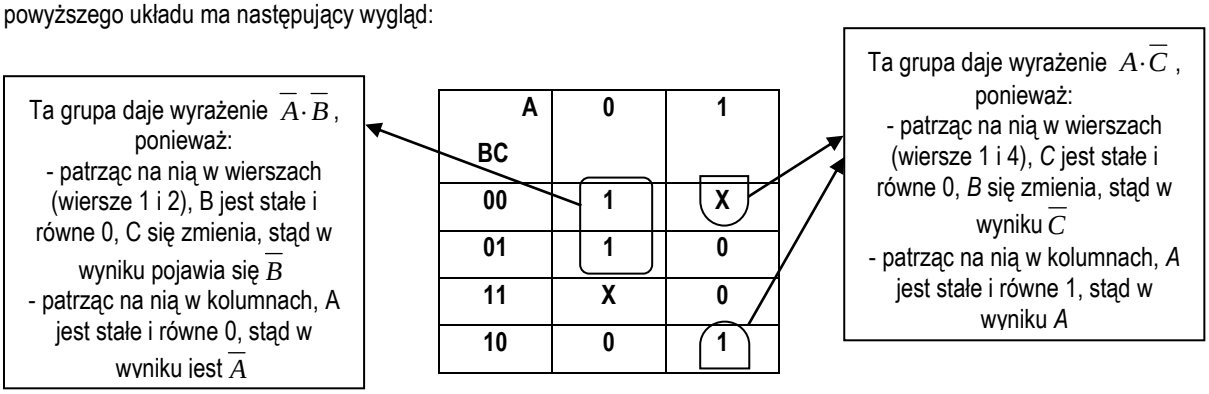
- LP = 15
- LB = 7 (3 bramki NOT i 4 bramki NAND, razem 7)
- LS = 3 (jeden układ 7400/4011 z którego wykorzystamy 3 (z 6 dostępnych) bramki NOT i np. dwa układy 7410/4023, z których weźmiemy 3-wejściowe bramki NAND)

Należy pamiętać, że mając do dyspozycji bramkę NAND o większej niż potrzebna liczbie wejść, można jej użyć jako bramki o mniejszej ich ilości, do wejść nieużywanych należy wtedy podłączyć na stałe stan wysoki (logiczna jedynka). Dlatego też np. realizując układ z Rysunku 1 można zamiast dwóch układów 7410, użyć dwa układy 7420, zawierające bramki 4-wejściowe, lub też po jednym 7410 i 7420. Wybór sposobu realizacji zależy najczęściej od dostępności poszczególnych US.

Oczywiście w przypadku realizacji układu w strukturze FPGA/CPLD również pożądana jest niska złożoność projektu, różnica jest jedynie taka, że struktura układu programowalnego zawiera znacznie większą liczbę bramek, więc liczba układów scalonych potrzebnych do realizacji projektu niezmiernie rzadko przekracza jeden. Na przykład układ FPGA wykorzystywany na ćwiczeniu zawiera zasoby logiczne odpowiadające 200.000 bramek logicznych.

2.2. Synteza układu kombinacyjnego na bramkach NAND - wersja zminimalizowana

W celu uproszczenia układu należy go zminimalizować np. popularną metodą graficzną (metodą tablic Karnaugh), która dla powyższego układu ma następujący wygląd:



Tworzenie tablicy Karnaugh sprowadza się do przepisania do niej stanów wyjść z tabeli stanów układu.

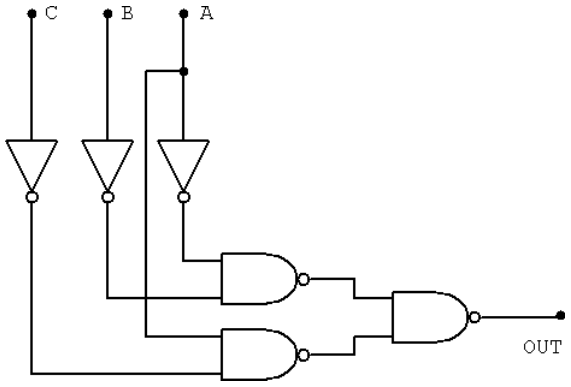
Zasady tworzenia grup przy minimalizacji względem jedynek:

- *łączymy w grupy wszystkie jedynki (pojedyncza otoczona zerami jedynka to też grupa!),*
- *X-y mogą być uznane za 1 lub 0, jak nam jest wygodniej, w przykładzie powyżej jeden X został uznany za 1 i wykorzystany w minimalizacji, drugi za 0.*
- *grupa powinna być „zwarta", ale może się symetrycznie „zawijać" (podobnie jak grupa złożona z 1 i X w powyższym przykładzie),*
- *grupę mogą stworzyć następujące ilości czynników: 1, 2, 4, 8, 16… (czyli potęgi dwójki), przy czym pożądane jest by grupy były jak największe.*

Minimalna (a także przekształcona do postaci realizowalnej na bramkach NAND) postać tej funkcji wygląda więc następująco:

$$OUT = \overline{A}\cdot\overline{B} + A\cdot\overline{C} = \overline{\overline{A}\cdot\overline{B} + A\cdot\overline{C}} = \overline{\overline{A}\cdot\overline{B}\cdot\overline{A\cdot\overline{C}}}$$

Jak widać, aby zrealizować powyższą postać funkcji potrzeba 3 bramek NAND 2-wejściowych i 3 bramek NOT (Rysunek 2).



Rysunek 2. Realizacja przykładowego układu w postaci zminimalizowanej przy użyciu bramek NAND (i NOT).

Zminimalizowany układ ma zdecydowanie mniejszą złożoność układową: **LP=9, LB=6, LS=2**
(poprzedni układ: LP=15, LB=7, LS=3).

PRZEBIEG ĆWICZENIA

CZĘŚĆ A – projekt układu.

3. Na podstawie zadanej przez prowadzącego w postaci tabeli stanów funkcji logicznej (OUT_Z) należy (cała wiedza potrzebna do realizacji niżej postawionych celów znajduje się w instrukcji):

3.1. Uzyskać postać kanoniczną funkcji – zamieścić w protokole (punkt 2).

3.2. Zminimalizować funkcję – wynik (czyli zminimalizowaną postać funkcji) zamieścić w protokole (punkt 3).

3.3. Zaprojektować na papierze uzyskaną minimalną wersję układu na dowolnych bramkach logicznych, projekt zamieścić w protokole (punkt 4).

CZĘŚĆ B – implementacja projektu w strukturze FPGA

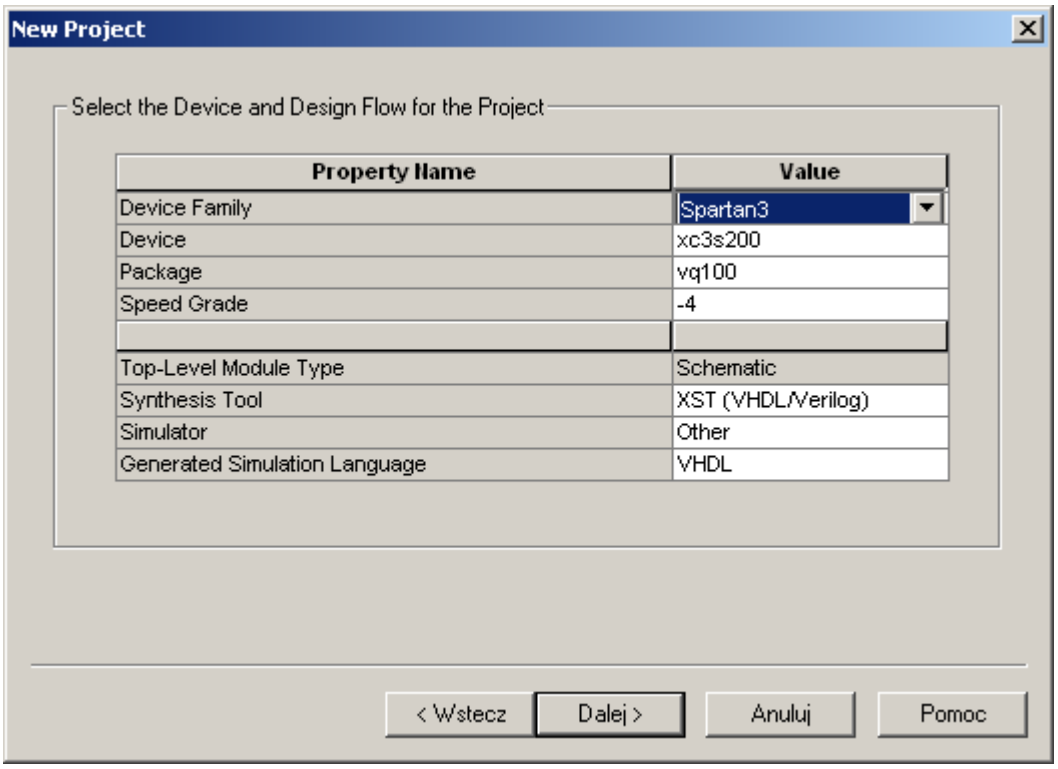
(UWAGA: kolejne podpunkty należy realizować bardzo uważnie, ponieważ błędy popełnione na tym etapie mogą być później nie do cofnięcia i trzeba będzie zaczynać od początku).

4. Na podstawie stworzonego w punkcie 1 projektu:

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

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

4.3. 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):

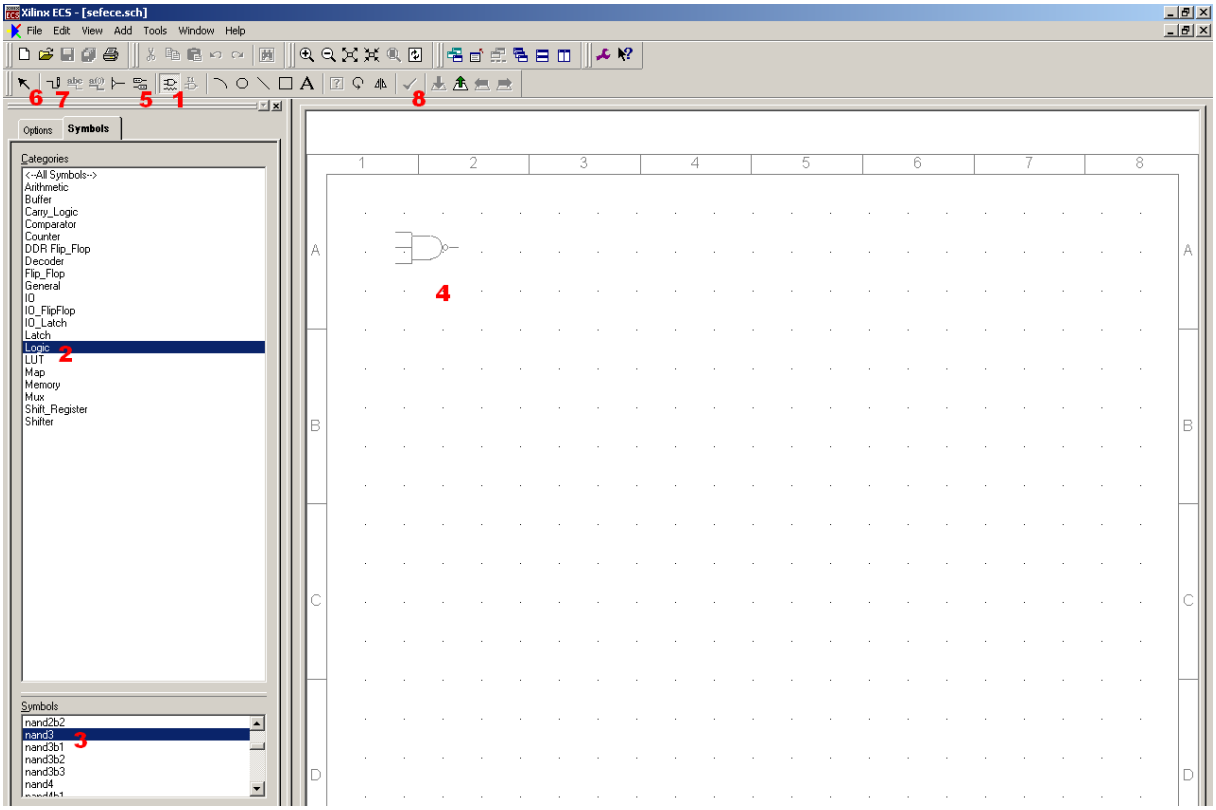


4.4. W kolejnym oknie stworzyć nowe źródło (***New Source***) do projektu. Nadać mu dowolną nazwę (***File Name***) i wybrać typ – **Schematic**.

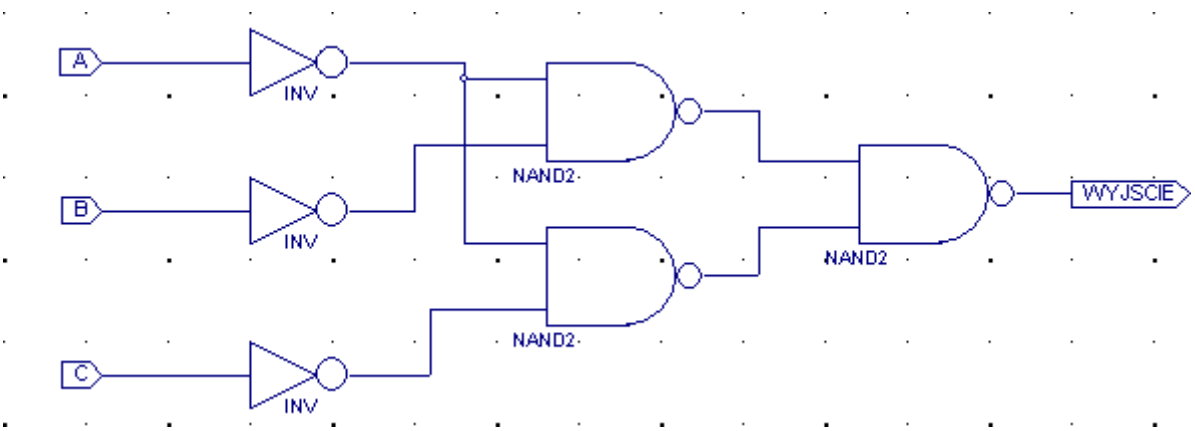
4.5. Dwa kolejne okna można przeklikać bez zmian. Na tym kończy się wstępna konfiguracja projektu. Na ekranie powinno pojawić się okno projektowania układu. W tym oknie budujemy swój projekt:

4.5.1. Najpierw należy wstawić potrzebne bramki logiczne, z menu u góry wybrać pozycję **Add Symbol (1)**, po lewej pojawi się lista dostępnych kategorii, stamtąd wybrać pozycję **Logic(2)**, a wtedy na dole **(3)** pojawi się lista dostępnych bramek logicznych, z których należy wybrać i wstawić do schematu bramki potrzebne do projektu.

4.5.2. Po wstawieniu wszystkich potrzebnych bramek z menu należy wybrać pozycję **Add I/O Marker (5)** i wstawić wejścia i wyjścia układu (w przypadku przykładowego projektu będą to 3 wejścia (**Add an input marker**) i jedno wyjście (**Add an output marker**)). Nazwy wejść można zmieniać dwuklikiem.



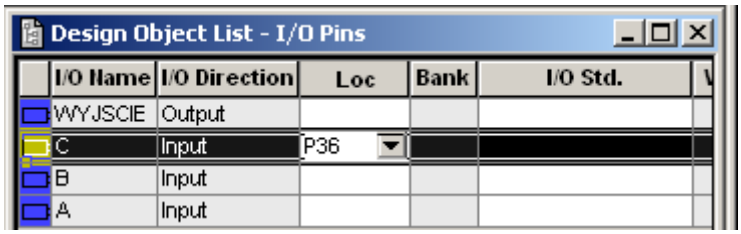
4.5.3. Gdy wstawione będą już bramki, wejścia i wyjścia, można jeszcze ewentualnie uporządkować ułożenie układu (**Select (6)**) i wykonać wszystkie potrzebne połączenia (**Add Wire(7)**). Gotowy układ (przykład na rysunku poniżej) można jeszcze sprawdzić pod względem formalnym (**Check Schematic(8)**).



4.5.4. Następnym krokiem jest przypisanie wejść i wyjść wykonanego układu do nóżek układu scalonego. W tym celu należy wrócić do okna **Xilinx - Project Nawigator** i w oknie po lewej stronie dwukliknąć w opcję **Assign Package Pins**. Powinno otworzyć się okno Xilinx PACE i topologia wykorzystywanego układu FPGA. Po lewej stronie (**Design Object List**) widać wcześniej zaprojektowane wejścia/wyjścia układu.

Nóżki układu przypisać zgodnie z załączoną notą zestawu ZL6PLD. Jako wejścia wykorzystać wybrane z przycisków klawiatury (strona 6 noty), jako wyjścia wybrane diody LED (strona 5 noty).

Piny najłatwiej przypisać wpisując tam bezpośrednio nóżkę układu scalonego, której chcemy przypisać daną funkcję, przykładowo chcąc przypisać nóżce 36 układu scalonego funkcję wejścia C, należy w kolumnie Loc wpisać P36 w odpowiednim wierszu (jak na rysunku poniżej).



- Po przypisaniu pinów można przystąpić do implementacji projektu w strukturze FPGA.

  W tym celu należy kolejno:

  - w oknie „Project Navigator" kliknąć w plik nadrzędny projektu;
  - w menu poniżej, na liście „Implement Design" znajduje się pozycja „Configure Device", dwukliknąć w nią LPM;
  - kolejne okna przeklikać z domyślnymi ustawieniami;
  - jeśli automatycznie nie otworzy się okno wyboru pliku, dwukliknąć LPM w symbol układu scalonego i wybrać plik z rozszerzeniem *.jed lub *.bit, który powinien mieć nazwę jak plik nadrzędny projektu;
  - następnie kliknąć PPM w symbol układu scalonego i wybrać „Program…",
  - po odczekaniu na implementację pliku w strukturze programowalnej, należy zgłosić prowadzącemu gotowość do testów.

5. Jeżeli układ realizuje zadaną funkcję, ćwiczenie jest zaliczone.

6. Ewentualnie nasuwające się wnioski/przemyślenia można zamieścić na odwrocie protokołu.

7. Literatura

[1] Borzdyński J, „Kurs CPLD", Elektronika Praktyczna 02-04/2009.

[2] Filipkowski A., „Układy elektroniczne analogowe i cyfrowe", WN-T, Warszawa 1978

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

[4] Górecki P., „Układy cyfrowe, pierwsze kroki", Wydawnictwo BTC, Warszawa 2004

[5] Kalisz J., „Cyfrowe układy scalone w technice systemowej", WMON, Warszawa 1997

[6] Morris Mano M., Kime Ch.R., „Podstawy projektowania układów logicznych i komputerów", WN-T, Warszawa 2007

Opracowanie ćwiczenia: Seweryn Lipiński