

# Środowiska Xilinx ISE i ISim

## Instrukcja laboratoryjna

*dr inż. Jarosław Sugier*



**Wersja  
14.7**

## Spis treści

1	Środowisko ISE.....	2
1.1	Project Navigator.....	2
1.2	Uruchamianie procesów .....	3
1.3	Częste problemy .....	4
1.3.1	Skojarzenia View Association .....	4
1.3.2	Nazwy plików i ścieżek .....	4
2	Opis układu w postaci schematowej .....	5
2.1	Utworzenie nowego projektu .....	5
2.2	Edycja schematu.....	5
2.2.1	Symbole, połączenia i porty WE/WY .....	5
2.2.2	Połączenia a magistrale .....	6
2.2.3	Nazwy połączeń .....	7
3	Symulacja projektu.....	7
3.1	Utworzenie pliku VHDL oraz definiowanie pobudzeń testowych .....	7
3.1.1	Przypisania sygnałów.....	9
3.1.2	Magistrale .....	9
3.1.3	Sygnały okresowe .....	9
3.2	Uruchomienie symulatora ISim .....	9
3.2.1	Wykresy czasowe.....	10
3.2.2	Śledzenie sygnałów wewnętrznych .....	11
3.2.3	Projekty wielomodułowe .....	11
4	Implementacja projektu oraz konfiguracja układu .....	12
4.1	Określenie lokalizacji sygnałów WE/WY.....	12
4.2	Zaprogramowanie układu.....	13
5	Opis układu w języku VHDL .....	14
5.1	Tworzenie modułów VHDL w projekcie.....	14
5.2	Submoduły i projekty heirarchiczne .....	15
	Źródła .....	15

# 1 Środowisko ISE

Zintegrowane środowisko Xilinx ISE służy do wykonania wszystkich operacji związanych z opracowaniem projektu układu cyfrowego oraz jego implementacją w układzie CPLD lub FPGA. Uruchamiane jest ikoną *Xilinx ISE Design Suite* (pulpit Windows lub pasek szybkiego uruchamiania), bądź z menu Start o odpowiedniej nazwie.

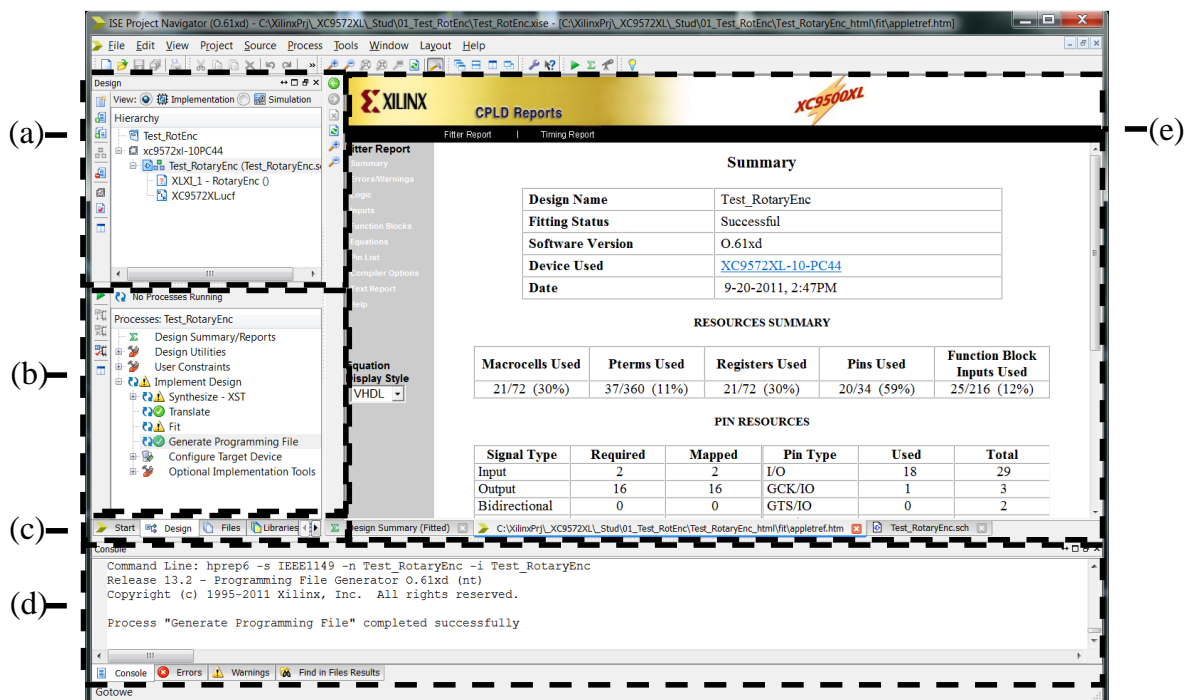
## 1.1 Project Navigator

Podstawowym komponentem środowiska ISE jest aplikacja *Project Navigator*, w której, uruchamiając inne składniki pakietu, można edytować pliki źródłowe, uruchamiać procesy odpowiadające za kolejne kroki syntezy i implementacji oraz zaprogramować układ dołączony poprzez interfejs JTAG.

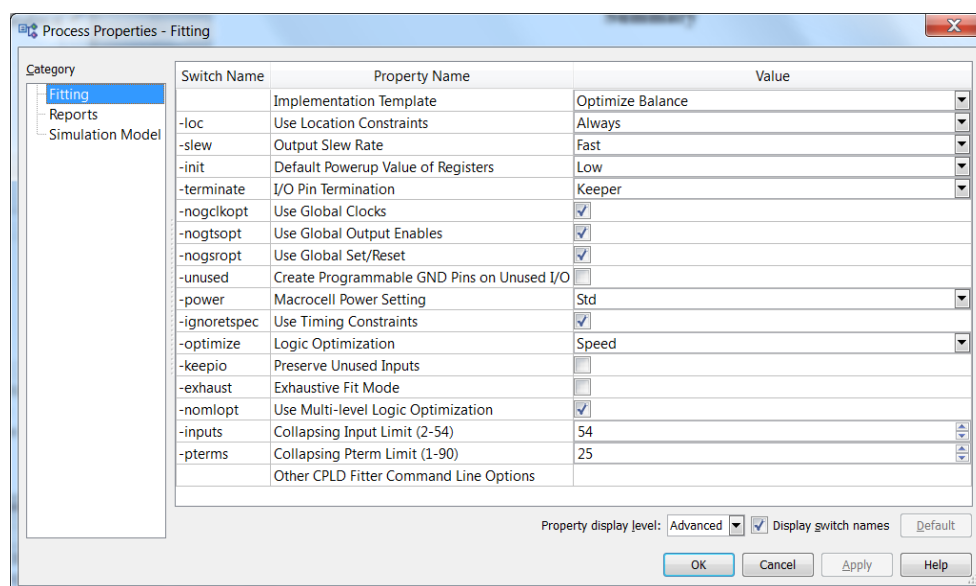
Widoczne na rys. 1 okno Nawigatora dzieli się na 4 główne obszary (układ przy wybranej zakładce *Design* w obszarze (c)):

- (a) drzewo plików źródłowych *Hierarchy* z hierarchią elementów źródłowych projektu;
- (b) drzewo procesów *Processes* z operacjami dostępnymi dla elementu źródłowego aktualnie wybranego w oknie (a);
- (c) zakładki wyboru dla obszarów (a) oraz (b)
- (d) konsola z komunikatami generowanymi przez procesy uruchomione w obszarze (b);
- (e) obszar roboczy, używany do edycji plików projektu, wizualizacji wyników itp.

Pole wyboru *View* na szczycie okna plików źródłowych (a) decyduje o tym, jakiego typu składniki projektu są wyświetlane w hierarchii. Dla projektów CPLD możliwe są do wyboru



Rys. 1 Okno główne Nawigatora projektu przy wyborze zakładki *Design*: (a) elementy źródłowe projektu, (b) okno procesów, (c) zakładki, (d) konsola, (e) obszar roboczy.



Rys. 2 Okno właściwości procesu *Implement Design* → *Fit*.

dwie opcje: *Implementation* oraz *Simulation*, przy czym dla opcji *Simulation* dodatkowo w rozwijanym oknie wybiera się jej typ: 1) behawioralna (bez opóźnień czasowych) lub 2) dokładna symulacja czasowa z uwzględnieniem wszystkich opóźnień projektu zaimplementowanego w rzeczywistym układzie (tzw. *post-fit*). W przypadku układów FPGA proces implementacji jest bardziej złożony i typów symulacji czasowej jest więcej.

☞ Do przywrócenia domyślnego rozkładu okien Nawigatora służy polecenie w menu *Layout* → *Load Default Layout* (bardzo pomocne, gdy układ okien został zmieniony i wymknął się spod kontroli).

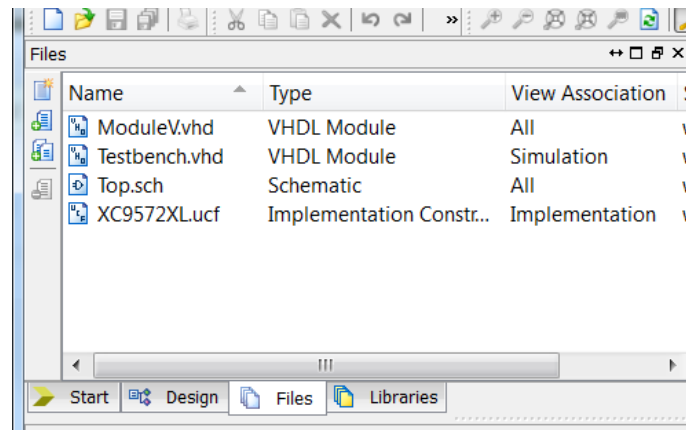
W każdym programie środowiska ISE polecenie menu *Help* → *Software Manuals* otwiera stronę WWW z odnośnikami do pełnej dokumentacji elektronicznej. Informacje na temat aktualnie uruchomionej aplikacji dostępne są w standardowym systemie pomocy Windows uruchamianym poleceniem *Help* → *Help Topics*.

## 1.2 Uruchamianie procesów

Zawartość okna procesów (b) na Rys. 1 zależy od tego, jakiego rodzaju element (schemat, moduł VHDL, wektor testowy itp.) jest aktualnie wybrany w oknie plików źródłowych (a). Większość operacji podczas pracy z Nawigatorem wykonuje się wybierając najpierw odpowiedni plik źródłowy w oknie (a) (co może wymagać przełączenia pola wyboru *View*, które znajduje się na jego szczycie) i następnie uruchamiając odpowiedni proces w oknie (b).

Proces zaznaczony w oknie (b) najprościej uruchomić przez jego dwukrotne kliknięcie. Środowisko ISE domyślnie wykonuje go wówczas w trybie *automake*, tzn. przed jego wywołaniem aktualizuje pliki wynikowe kroków wcześniejszych, o ile jest to potrzebne (decyduje porównanie daty utworzenia plików wynikowych oraz źródłowych). Kliknięcie na proces prawym klawiszem otwiera menu kontekstowe, w którym mogą być dostępne inne tryby jego uruchomienia, np. *Rerun All*, *Run with current data*, itp.

Często przed uruchomieniem procesu konieczne jest ustawienie jego specyficznych parametrów. Wykonuje się to poprzez okno właściwości procesu (przykład na rys. 2), otwierane poleceniem *Process Properties* z menu kontekstowego (kliknięcie na proces prawym klawiszem myszy).



Rys. 3 Różne typy plików źródłowych i ich poprawne skojarzenia View Association.

- ☞ Polecenie menu Project → Cleanup Project Files... usuwa istniejące (już wygenerowane) pliki wynikowe procesów syntezy / implementacji i jest zalecane jako metoda re-inicjalizacji projektu potrzebna niekiedy np. po wprowadzeniu wielu zmian, które nie zawsze są poprawnie wykrywane przez mechanizm automake. Podobny efekt może przynieść polecenie Rerun All (z menu kontekstowego procesu), które powoduje bezwarunkowe powtórzenie wszystkich poprzednich kroków projektowych.
- ☞ Okno źródłowe (a) oraz procesów (b) znajdują się w zakładce Design; uruchomienie w obszarze roboczym np. edytora schematów powoduje niekiedy automatyczne przełączenie zakładek.

## 1.3 Częste problemy

### 1.3.1 Skojarzenia View Association

Pliki źródłowe w projekcie posiadają tzw. skojarzenia View Association (rys. 3), które są podawane w zakładce Files znajdującej się obok Design. Dla poprawnej pracy środowiska powinny być zachowane następujące skojarzenia:

- schematy oraz moduły VHDL do syntezy – skojarzenie All;
- pliki VHDL z pobudzeniami testowymi – Simulation;
- pliki UCF z ograniczeniami projektowymi – Implementation.

Pojawiające się niekiedy związane z tym błędy w pracy ISE to np. niewidoczny w hierarchii projektu plik ze schematem (gdy ma błędne skojarzenie Simulation) lub bezsensowna próba implementacji pliku VHDL z pobudzeniami testowymi (gdy taki plik ma skojarzenie All i staje się elementem szczytowym hierarchii).

### 1.3.2 Nazwy plików i ścieżek

Nazwy niektórych plików źródłowych są używane jako identyfikatory w kodzie VHDL i należy tworzyć je tak, aby spełniały odpowiednie warunki: muszą rozpoczynać się od litery oraz nie mogą zawierać żadnych znaków innych niż litery, cyfry oraz znak podkreślenia. Błędy tego rodzaju w nazwach plików nie zawsze są sygnalizowane przez środowisko, a objawiają się dopiero złą pracą niektórych jego narzędzi podczas dalszych etapów implementacji lub symulacji.

Ponadto, w ścieżce katalogu z projektem ISE nie powinny występować odstępów (w szczególności więc nie należy umieszczać projektów na pulpicie Windows). Zalecana lokalizacja folderów z projektami to „C:\XilinxPrj\”. Folder „C:\Użytkownicy\” w polskiej wersji systemu także może być powodem problemów (litera ‘ż’).

## 2 Opis układu w postaci schematowej

Rozdział ten omawia główne kroki przy projektowaniu układu, którego opis ma być zadany w postaci graficznej jako schemat logiczny.

### 2.1 Utworzenie nowego projektu

1° Po uruchomieniu środowiska wybierz polecenie *File* → *New Project...*. W pierwszym oknie *Create New Project* podaj jego nazwę, lokalizację oraz rodzaj głównego pliku źródłowego *Top-Level Source Type*; w tym przypadku wybierz *Schematic*. Wyświetlany katalog roboczy (*Working directory*) powinien być identyczny jak lokalizacja (*Location*).

☞ *Foldery Working directory oraz Location powinny zawsze wskazywać tę samą lokalizację. Sprawdzanie tego warunku (poleceniem Project → Design properties) powinno być wykonywane szczególnie przy kopiowaniu folderu projektu np. między różnymi komputerami.*

2° W drugim oknie *Project Settings* wybierz kategorię oraz rodzinę układów, konkretne urządzenie, typ obudowy oraz gradację szybkości (np. płyta CPLD zawiera układ XC9572XL w obudowie PC44 o gradacji szybkości -10). Inne istotne parametry:

Synthesis Tool = XST (VHDL/Verilog)

Simulator = ISim (VHDL/Verilog)



Preferred Language = VHDL

Pozostałe opcje pozostaw bez zmian i potwierdź utworzenie projektu w kolejnym oknie.


### 2.2 Edycja schematu

Dodaj do pustego projektu nowy plik źródłowy typu schemat: menu *Project* → *New Source...*, rodzaj pliku *Schematic*, wpisz nazwę pliku, pozostaw opcję *Add to project* włączoną, naciśnij *Next* i potwierdź wpisane parametry w kolejnym oknie.

Po utworzeniu nowego pliku wg powyższych kroków środowisko automatycznie otwiera w obszarze roboczym Nawigatora edytor schematów ECS. Dla schematu już istniejącego edytor ten można uruchomić klikając dwukrotnie plik *.sch* w drzewie elementów źródłowych projektu (zakładka *Design*).

Domyślnie edytor jest wyświetlany jako podokno w obszarze roboczym Nawigatora, ale możliwa jest także jego praca niezależna (rys. 4). Przełączenie pomiędzy tymi dwoma trybami pracy umożliwiają klawisze  /  (menu *Window* → *Float / Dock*).

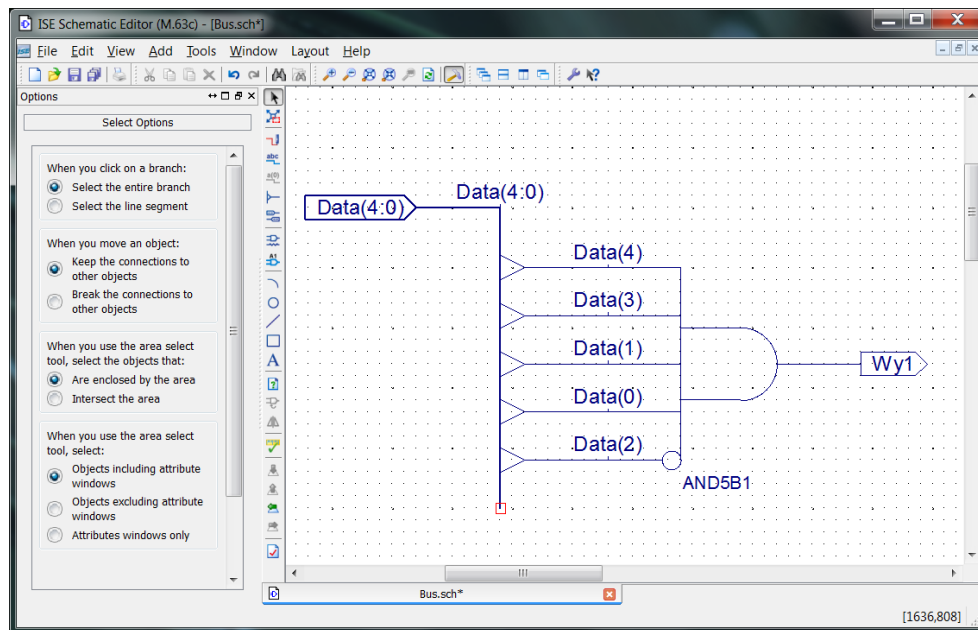
#### 2.2.1 Symbole, połączenia i porty WE/WY

Podczas edycji schematu należy umieszczać symbole elementów logicznych wybrane w zakładce *Symbols* widocznej po lewej stronie obszaru roboczego (klawisz  na pionowym pasku narzędzi przy lewej krawędzi schematu).

☞ *Źródłami sygnałów logicznej 1 oraz 0 są elementy vcc oraz gnd; bramka negacji nosi nazwę inv.*

Szczegółowych informacji nt. edytora ECS można szukać w jego systemie pomocy *Help* → *Help Topics*. Opisy wszystkich elementów logicznych (bramki, przerzutniki itp.) zawarte są w dokumencie *Libraries Guide* dostępnym w systemie pomocy. Opisy poszczególnych elementów można także wywołać klawiszem *Symbol Info* na dole zakładki *Symbols*.





Rys. 4 Edytor schematów ECS (i przykład definicji magistrali wielobitowej).

Do rysowania połączeń służy polecenie menu *Add* → *Wire* (🔌). Utworzone w ten sposób połączenie ma w kodzie VHDL nazwę *XLXN\_nm*, którą – np. w celu odszukania sygnału podczas symulacji – można zmienić poleceniem *Add* → *Net Name* (🔍).

Porty WE/WY projektu należy tworzyć jako znaczniki WE/WY poleceniem *Add* → *I/O Marker* (🔌). Nazwa portu jest zawsze identyczna z nazwą dochodzącego do niego połączenia (zmiana nazwy jednego pociąga za sobą zmianę nazwy drugiego).

Dwukrotne kliknięcie dowolnego elementu na schemacie (kursor w trybie *Select*, 🖱️) otwiera okno z jego atrybutami. Poprzez atrybuty połączenia z portem WE/WY można nadać mu nazwę własną lub zmienić jego kierunek (WE lub WY).


☞ Podczas pracy z edytorem schematów w zakładce *Options* widocznej na lewo od obszaru roboczego dostępne są różne przydatne opcje aktualnie wybranego narzędzia, np. zaznaczanie całości połączenia lub tylko pojedynczych jego odcinków (konieczne przy usuwaniu fragmentów narysowanego połączenia), automatyczne lub ręczne trasowanie ścieżki połączeń podczas ich dodawania (Autorouter dla 🔌), itp.

☞ Przydatne skróty klawiszowe: *F8 / F7* = *Zoom In/Out*, *F5* = *Refresh*.

Po zakończeniu edycji schematu należy sprawdzić jego poprawność poleceniem *Tools* → *Check Schematic* (klawisz 🟢); jeśli znajdowane są błędy, raportowane symbole lub połączenia można szybko odnaleźć na schemacie poleceniem *Edit* → *Find* (Ctrl+F).

## 2.2.2 Połączenia a magistrale

Rysunek 4 ilustruje zasadę konstruowania magistral sygnałowych: połączenie staje się magistralą (tzn. wektorem sygnałów binarnych), jeśli w jego nazwie zostanie podany zakres indeksu (*k:l*), np. *Data(4:0)*. Dostęp do poszczególnych składowych magistrali następuje poprzez odwołanie do konkretnej wartości indeksu w nazwie połączenia do niej poprowadzonego, np. *Data(0)*. Zakres indeksu nie musi być malejący ani kończyć się na 0, ale należy trzymać się takiego standardu.

Elementem graficznym przedstawiającym dołączenie linii do magistrali jest tzw. odczep (*bus tap*, ). Odczep można umieścić na schemacie tylko poprzez dołączenie do już istniejącej magistrali, tzn. należy najpierw narysować połączenie, następnie zmienić jego nazwę tak, aby zawierała zakres indeksu (czyli utworzyć magistralę) i dopiero potem dorysować do niego odczepy.

☞ *Jeśli odczep nie daje się dołączyć do magistrali, upewnij się, że w opcjach narzędzia puste są pola Selected Bus Name oraz Net Name.*

### 2.2.3 Nazwy połączeń

Nazwy portów i połączeń na schemacie będą nazwami sygnałów VHDL i muszą być poprawnymi identyfikatorami tego języka: muszą zaczynać się od litery, nie mogą być słowami kluczowymi, np. *in*, *out*, itp. Błędy tego rodzaju nie zawsze są wykrywane przez polecenie *Check Schematic*.

Dwa połączenia o tych samych nazwach, nawet jeśli wizualnie oddzielne na schemacie, są uważane za zwarte i jest to często powodem błędów. Sytuacja taka zazwyczaj powstaje po usunięciu tylko jednego segmentu dużego rozbudowanego połączenia: pozostałe po nim dwie części mają wciąż tę samą nazwę i są uważane za zwarte. Zmiana nazwy jednej części problemu nie rozwiązuje (nazwa drugiej części zostanie również zmieniona) i należy wówczas użyć polecenia *Edit → Rename (Bus / Net)* z wybraną opcją *Rename the branch*.

## 3 Symulacja projektu

Symulacja przygotowanego projektu wymaga:

- 1) utworzenia pliku VHDL z wektorami pobudzeń dla wszystkich portów WE (tzw. plik *testbench*),
- 2) wywołania symulatora ISim, który obliczy oraz przedstawi graficznie odpowiedzi wygenerowane przez układ na portach WY.

Język VHDL będzie omawiany na wykładzie, natomiast podane niżej informacje wstępne, choć fragmentaryczne, są wystarczające do pisania prostych wektorów pobudzeń na początkowych zajęciach.

### 3.1 Utworzenie pliku VHDL oraz definiowanie pobudzeń testowych

System ISE automatycznie generuje odpowiedni szablon VHDL dla wskazanego schematu (zob. przykład na rys. 5), zadaniem użytkownika jest tylko dodać do niego polecenia definiujące przebiegi czasowe sygnałów WE.

- 1° Wybierz polecenie *Project → New Source...*, wybierz rodzaj *VHDL Test Bench* oraz podaj nazwę pliku; w następnym oknie zaznacz plik schematowy jako obiekt symulacji.

☞ *Podana nazwa pliku TestBench musi być inna niż nazwa pliku ze schematem!*

- 2° Po zatwierdzeniu parametrów zostanie utworzony plik o rozszerzeniu *.vhd* z automatycznie utworzoną jednostką testową (ENTITY) oraz architekturą (ARCHITECTURE). Wygenerowane nazwy jednostki oraz architektury nie są w tym momencie istotne i należy pozostawić je bez zmian, uzupełnienia natomiast wymagać będzie treść architektury zawarta pomiędzy liniami *BEGIN ... END*.



```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.numeric_std.ALL;
4  LIBRARY UNISIM;
5  USE UNISIM.vcomponents.ALL;
6
7  ENTITY Top_sch_tbw IS
8  END Top_sch_tbw;
9
10 ARCHITECTURE behavioral OF Top_sch_tbw IS
11
12     COMPONENT Top
13     PORT ( We2      : IN STD_LOGIC;
14           We1      : IN STD_LOGIC;
15           Wy       : OUT  STD_LOGIC);
16     END COMPONENT;
17
18     SIGNAL We2      : STD_LOGIC;
19     SIGNAL We1      : STD_LOGIC;
20     SIGNAL Wy       : STD_LOGIC;
21
22 BEGIN
23
24     UUT: Top PORT MAP (
25         We2 => We2,
26         We1 => We1,
27         Wy  => Wy
28     );
29
30     WE1 <= '0', '1' after 100 ns, '0' after 300 ns;
31     WE2 <= '0', '1' after 200 ns, '0' after 400 ns;
32
33
34 END;

```

Rys. 5 Szablon kodu VHDL automatycznie wygenerowany dla prostego schematu, uzupełniony o instrukcje przypisania pobudzeń do portów wejściowych (pominięto linie komentarzy rozpoczynające się znakami '--'). Linie 1÷5 to deklaracje bibliotek, 7÷8 – definicja jednostki, a 10÷34 – definicja architektury, w tym:  
 12÷16 – deklaracja testowanego komponentu, 18÷20 – definicje sygnałów odpowiadających portom komponentu, 24÷28 – instrukcja instancji testowanego komponentu z przypisaniem sygnałów do portów, 30÷31 – dopisane instrukcje przypisać sygnałów WE.

- 3° Przed modyfikacją treści architektury należy jeszcze zwrócić uwagę na jej część deklaracyjną przed słowem BEGIN. Powinny znajdować się w niej automatycznie wygenerowane (zob. Rys. 5): deklaracja symulowanej jednostki (COMPONENT *nazwa\_pliku*) z portami WE/WY jak na schemacie oraz definicje odpowiadających im sygnałów (SIGNAL...). Identyczne nazwy sygnałów oraz portów komponentu nie powodują konfliktów w języku VHDL; ponadto duże i małe litery nie są rozróżniane w identyfikatorach, więc tego typu różnice mogą wystąpić w porównaniu do nazw występujących na schemacie.

☞ *Tworzenie pliku powiedzie się tylko dla poprawnie skonstruowanego schematu. Jeśli plik vhd nie powstaje lub ma błędną treść (np. nie zawiera wszystkich portów znajdujących się na schemacie), należy wrócić do edycji pliku sch, sprawdzić jego poprawność poleceniem Tools → Check Schematic i usunąć błędy.*

- 4° W treści architektury (po słowie BEGIN) powinna znajdować się instrukcja instancji zadeklarowanego wyżej komponentu UUT: (...) ;. Należy pozostawić ją bez zmian, natomiast dalsze linie (Clock proces definition, Stimulus proces, ...) usunąć – zostaną zastąpione poleceniami własnymi.
- 5° W miejscu usuniętych linii należy dopisać własne pobudzenia tych i tylko tych sygnałów, które odpowiadają portom WE. Ostateczna struktura kodu powinna przypominać tę z Rys. 5.

### 3.1.1 Przypisania sygnałów

Pobudzenia testowe najprościej wyrazić instrukcjami przypisań współbieżnych ' $\leq$ ', których uproszczoną składnię wystarczająco tłumaczą następujące przykłady:

```
we1 <= '0', '1' after 100 ns, '0' after 300 ns;  
we2 <= '0', '1' after 200 ns, '0' after 400 ns;
```

Dla każdego portu WE należy podać dokładnie jedną instrukcję przypisania. W przykładach powyższych we1 i we2 są nazwami portów (jak na schemacie), '0' oraz '1' są przypisywanymi wartościami – stałymi logicznymi, natomiast po słowach after podawane są momenty czasu, w których dane przypisanie ma mieć miejsce. Efekt powyższych dwóch instrukcji można zobaczyć na rys. 6: oba sygnały są początkowo inicjalizowane zerem logicznym, po czym na każdym z nich pojawia się trwający 200 ns impuls jedynkowy. Różne przesunięcia czasowe impulsu dają efekt dwubitowego kodu Gray'a.

☞ *Podawane w tej instrukcji wartości czasu są momentami wykonania przypisań, a nie długościami ich trwania; momenty przypisani muszą tworzyć ciąg rosnący.*

☞ *Oznaczenia jednostek czasu w języku VHDL: fs, ps, ns, us, ms, sec.*

### 3.1.2 Magistrale

Sygnałom, które są wektorami (tzn. odpowiadają portom magistralowym), przypisuje się stałe ujęte w znaki "...", np.:

```
-- In section with signal definitions:  
SIGNAL BUS4b : STD_LOGIC_VECTOR( 3 DOWNT0 0 );  
  
(...)  
  
-- Concurrent assignment in architecture body  
BUS4b <= "0000", "0001" after 100 ns, "0011" after 300 ns;
```

Stałe wektorowe, których długość jest wielokrotnością 4, można zapisywać w notacji heksadecymalnej z przedrostkiem X, np.: "0011" = X"3", "11110011" = X"F3", itp. (ale "011"  $\neq$  X"3", bo X"3" = "0011").

### 3.1.3 Sygnały okresowe

Gdy zachodzi potrzeba generacji fali prostokątnej o wypełnieniu 50% i stałym okresie, należy:

a) dopisać inicjalizację wartości początkowej sygnału do jego definicji w części deklaracyjnej architektury:

```
SIGNAL Clk : STD_LOGIC := '0'; --dopisano tekst podkreślony
```

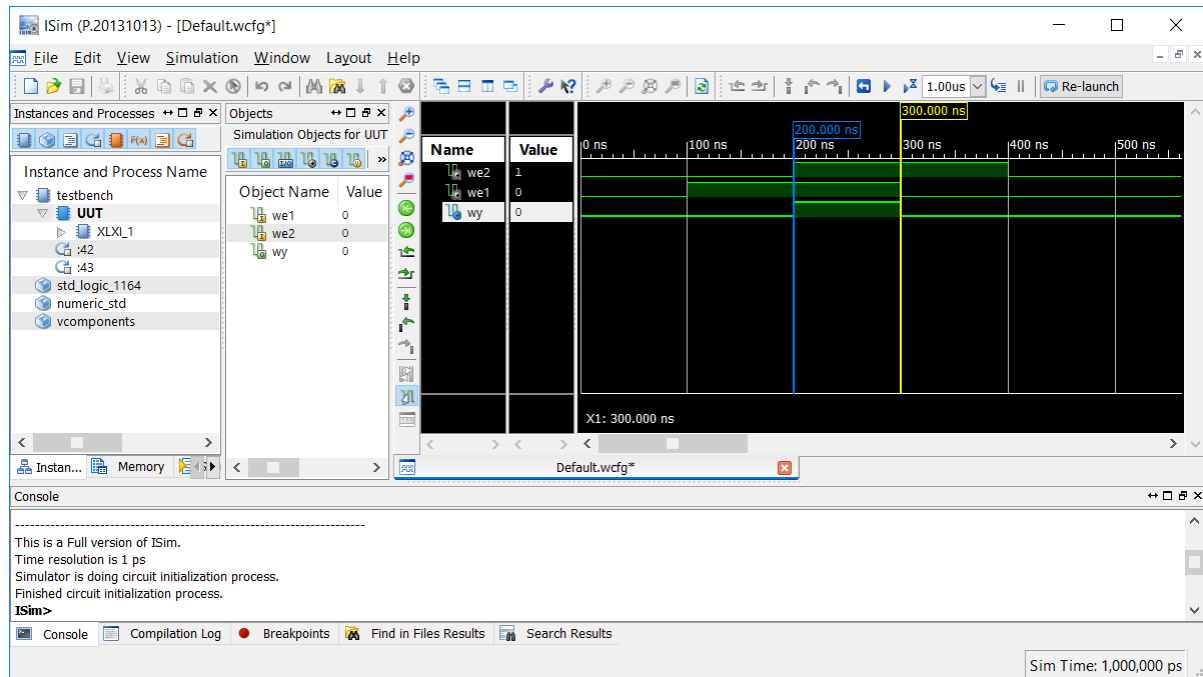
b) w treści architektury zastosować przypisanie sygnału jak w poniższym przykładzie:

```
Clk <= not Clk after 500 ns;
```

gdzie wartość 500 ns jest zadaną wartością półokresu, a Clk jest przykładową nazwą portu.

## 3.2 Uruchomienie symulatora ISim

W oknie plików źródłowych Nawigatora (rys. 1a) wybierz opcję *View: Simulation* oraz zaznacz utworzony plik VHDL z pobudzeniami. Uruchom proces *ISim Simulator* → *Simulate*



Rys. 6 Okno z przebiegami czasowymi symulatora ISim.

*Behavioral Model* lub *ISim Simulator* → *Simulate Post-Fit Model*, w zależności od wybranego typu symulacji (Rys.1, rozwijana lista wyboru na szczycie obszaru (a)).

☞ W przypadku wybrania symulacji czasowej post-fit przed uruchomieniem symulatora może okazać się konieczna synteza i implementacja układu; jeśli nie zakończy się powodzeniem, zamiast oczekiwanej pracy z aplikacją ISim trzeba będzie wrócić do poprawiania błędów wykrytych w źródłach projektu.

☞ Przy uruchamianiu symulatora w oknie Hierarchy musi być zaznaczony plik .vhd z pobudzeniami, a nie podporządkowany mu schemat UUT (również widoczny w drzewie źródłowym bezpośrednio pod TBW).

Symulator ISim pracuje we własnym oknie z szeregiem specyficznych obszarów roboczych; przykład obliczonej symulacji behawioralnej pokazuje Rys. 6. Przywrócenie domyślnego rozkładu okien wykonuje polecenie menu *Layout* → *Restore Default Layout*.

### 3.2.1 Wykresy czasowe

Przy analizie wykresów podstawą jest ustawienie odpowiedniego powiększenia skali czasowej poleceniami *View* → *Zoom* →

...In	F8	
...Out	F7	
...To Full View	F6	

Ponadto przy wyznaczaniu opóźnień pomocne są polecenia ustawiające precyzyjnie kursor w momentach zmian sygnałów oraz dodające markery w aktualnej pozycji kursora:

<i>View</i> → <i>Cursor</i> → <i>Previous Transition</i>	<i>Left</i>	
<i>View</i> → <i>Cursor</i> → <i>Next Transition</i>	<i>Right</i>	
<i>Edit</i> → <i>Markers</i> → <i>Add Marker</i>		

Na Rys. 6 widać ustawienie markera oraz kursora dla impulsu sygnału wy.

Nawigator ISE domyślnie wywołuje symulator z symulacją trwającą 1000 ns. Kolejne przebiegi wydłużające czas symulacji już wykonanej można uzyskać wpisując w panelu *Console* na dole okna ISim polecenie `run <time>`, np. `run 10 us`. Polecenie *Simulation* → *Restart* (Ctrl+Shift+F5 lub wpisanie `restart` w panelu *Console*) wraca do chwili zero.

### 3.2.2 Śledzenie sygnałów wewnętrznych

Podczas symulacji behawioralnej możliwe jest śledzenie nie tylko portów badanego modułu (które są zawsze domyślnie dodawane do okna przebiegów przy uruchomieniu symulatora), ale też wszystkich jego sygnałów wewnętrznych - połączeń na schemacie czy sygnałów zdefiniowanych w architekturze VHDL.

Aby dodać nowy sygnał do okna przebiegu:

- 1° w panelu *Instance and Process Name* przy lewej krawędzi okna ISim rozwiń znajdującą się na szczycie nazwę architektury z pliku z pobudzeniami (na Rys. 6 – *testbench*) oraz zaznacz moduł UUT (*Unit Under Test*, nazwa instancji testowanego komponentu w pliku *testbench* - por. linia 24 Rys. 5),
- 2° w sąsiednim panelu *Simulation Objects* pojawi się pełna lista portów oraz sygnałów wewnętrznych wybranego modułu; zaznacz potrzebne sygnały oraz przeciągnij je do okna przebiegów.

☞ Aby można było rozpoznać w symulatorze sygnały wewnętrzne warto nadać połączeniom na schemacie nazwy znaczące w miejsce domyślnie generowanych przez ISE nazw automatycznych typu `XLXN_nn`.

☞ W symulacji typu *Post-fit* sygnałami wewnętrznymi, których zmiany także można analizować, są sygnały biegnące pomiędzy elementami występującymi w rzeczywistym układzie CPLD lub FPGA (bramkami, przerzutnikami, buforami, itp.). Podczas implementacji projektu ich nazwy są tworzone jako pochodne nazw portów i połączeń występujących na schemacie.

Domyślnie symulator pamięta historię zmian tylko sygnałów aktualnie wyświetlanych w oknie przebiegów i dlatego po dodaniu nowego sygnału jego wykres jest pusty. Należy wówczas ponownie wykonać symulację poleceniami `restart` i `run`, jak opisano to powyżej.

☞ Aby obejść tę niedogodność można włączyć rejestrowanie historii zmian wszystkich sygnałów poleceniem `„wave log -r /”` wpisanym ręcznie w oknie *Console* zaraz po uruchomieniu symulatora.

### 3.2.3 Projekty wielomodułowe

W projektach, które składają się z wielu modułów źródłowych (np. schemat szczytowy + podschematy lub zagnieżdżone instancje modułów VHDL), można przeprowadzić symulację behawioralną dla dowolnego modułu w hierarchii, tzn. testować wybraną część projektu: tworząc plik z pobudzeniami należy wskazać plik źródłowy, który ma być symulowany, i określić pobudzenia dla jego wejść.

Symulacja czasowa (*post-fit* dla CPLD, *post-route* dla FPGA) ma sens wyłącznie dla modułu będącego na szczycie hierarchii plików źródłowych, tj. dla całości projektu zaimplementowanego w układzie. Ustawienie szczytu hierarchii wykonuje się poleceniem *Set as Top Module*, które jest dostępne w menu kontekstowym modułu widocznego w drzewie

plików źródłowych (jeśli jest ono nieaktywne, dany moduł już znajduje się na szczycie hierarchii).

## 4 Implementacja projektu oraz konfiguracja układu

### 4.1 Określenie lokalizacji sygnałów WE/WY

Przypisanie sygnałów WE/WY projektu do konkretnych wyprowadzeń obudowy układu programowalnego jest fragmentem szerszego zagadnienia, którym jest definiowanie tzw. ograniczeń projektowych (*User Constraints*). Sformułowane ograniczenia są przechowywane w pliku tekstowym o rozszerzeniu UCF, który jest jednym z (ważnych!) elementów źródłowych projektu i jest widoczny w Nawigatorze jako element w drzewie plików wejściowych dla *View = Implementation*.

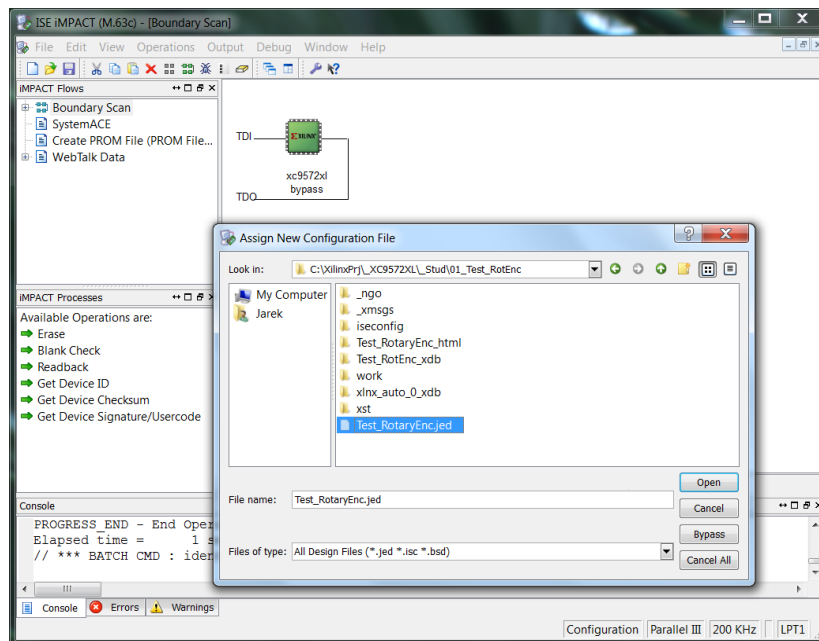
Podczas pracy z płytą ZL-9572 należy korzystać z pliku UCF dostępnego na stronie WWW, który zawiera poprawne definicje przypisań wszystkich wyprowadzeń. Plik ten należy skopiować do folderu projektu, dodać go poleceniem „Project → Add Source” i następnie usunąć w nim znaki komentarza (#) w liniach opisujących sygnały rzeczywiście używane w projekcie.

W przypadku ogólnym, aby utworzyć nowy plik UCF oraz przypisać wyprowadzenia należy ręcznie dodać nowe źródło UCF do projektu (menu *Project → New Source → Implementation Constraints File*), następnie otworzyć go do edycji (proces *User Constraints → Edit Constraints (Text)*) i samodzielnie wpisać odpowiednie linie z poleceniami LOC (*Location*):

```
#Format: NET "NazwaPortuSCH" LOC = "Wyprowadzenie";  
NET "We1" LOC = "P13";  
NET "We2" LOC = "P12";  
NET "Wy" LOC = "P24";  
...
```

Numerory wyprowadzeń zależą od typu obudowy używanego układu. Dla obudowy typu PLCC numeracja wyprowadzeń jest liniowa i symbole mają postać „Pn” (P1, P2, ...); dla obudów z dwuwymiarową matrycą wyprowadzeń ( np. obudowy *Ball Grid Array*, BGA, używane w układach FPGA), symbole mają format „szachowy”, np. A1, C35 itp. W każdym przypadku należy sprawdzić w dokumentacji płyty, do którego wyprowadzenia obudowy powinien być dołączony każdy port występujący na schemacie i opisać to odpowiednio w pliku UCF.

- ☞ *Jeśli w projekcie pozostają porty, dla których nie podano poleceń LOC w pliku UCF, program automatycznie przypisze im lokalizację wg własnych kryteriów. Implementacja będzie wówczas przebiegała bez żadnych komunikatów o błędach lub ostrzeżeniach, ale są niewielkie szanse na to, że sygnały trafią w odpowiednie miejsca i projekt będzie działać po zaprogramowaniu w sprzęcie; zazwyczaj efektem jest np. zapalanie losowych diod. Wniosek: wszystkie porty projektu powinny być uwzględnione w pliku UCF.*
- ☞ *Obecność w pliku UCF lokacji sygnału, który nie występuje w projekcie, jest błędem przerywającym proces implementacji.*



Rys. 7 Poprawnie wykryty układ w aplikacji iMPACT z oknem wyboru pliku konfiguracyjnego.

## 4.2 Zaprogramowanie układu

1° Dla zaznaczonego głównego pliku schematowego uruchom proces *Implement Design* → *Generate Programming File*. Jeśli zakończy się powodzeniem, w katalogu roboczym projektu powstanie plik o rozszerzeniu .jed gotowy do przesłania do układu CPLD.

☞ *Implementacja projektów CPLD składa się z procesów Synthesize, Translate oraz Fit. Jeśli którykolwiek nie zakończy się pomyślnie, należy odszukać w oknie konsoli komunikaty o błędach i wrócić do ich usunięcia. Najczęściej błędy procesu Synthesize dotyczą kodu VHDL lub schematu, błędy Translate – pliku UCF, błędy Fit – problemów z przypisaniem wyprowadzeń, niedostatecznymi zasobami w wybranym układzie itp.*

2° Podłącz sprzęt do zasilania oraz poprzez kabel JTAG do komputera i uruchom aplikację iMPACT (proces *Configure Target Device* dla szczytowego elementu hierarchii źródłowej).

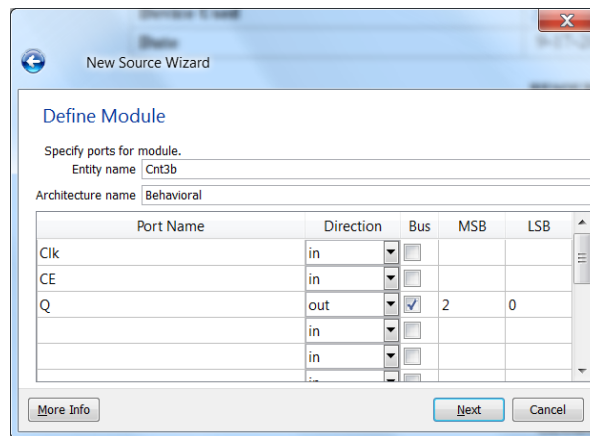
3° Jeśli się pojawi, zamknij okno *New iMPACT Project* bez tworzenia projektu oraz dwukrotnie kliknij opcję *Boundary Scan* w panelu *iMPACT Flows* (lewy górny róg okna głównego). Następnie wybierz polecenie *File* → *Initialize Chain* (Ctrl+I).

☞ *W tym momencie program iMPACT musi automatycznie wykryć i poprawnie zidentyfikować dołączony układ CPLD, tak jak pokazuje to rys. 7. Jeśli tak się nie stanie, oznacza to problem ze sprzętem: sprawdź jeszcze raz podłączenie kabli, zasilanie płyty, itp.*

4° W oknie wyboru pliku konfiguracyjnego (rys. 7) wskaż plik .jed wygenerowany w katalogu głównym projektu, następnie zaakceptuj bez zmian zaproponowane ustawienia *Device Programming Properties*.

5° W tym momencie aplikacja iMPACT jest gotowa do zaprogramowania układu: kliknij prawym klawiszem myszy na jego symbol oraz wybierz polecenie *Program...*; jeśli





Rys. 8 Definicja portów podczas tworzenia nowego modułu VHDL.

transmisja poprzez kabel JTAG odbędzie się pomyślnie, pojawi się niebieski komunikat *Programming succeeded*.

☞ W przypadku problemów z transmisją można wykonać test łącza JTAG poleceniem Debug → IDCODE Looping.

- 6° Układ CPLD został skonfigurowany zgodnie z przygotowanym projektem i rozpoczął pracę – sprawdź jej poprawność na płycie laboratoryjnej.

## 5 Opis układu w języku VHDL

### 5.1 Tworzenie modułów VHDL w projekcie

Język VHDL jest podstawowym formatem opisu danych w środowisku ISE dla ścieżki projektowej *XST VHDL* i nie wymaga żadnych zewnętrznych edytorów. Aby utworzyć nowy moduł w języku VHDL i dodać go do projektu:

- 1° W nawigаторze wybierz polecenie *Project → New Source...*; wybierz rodzaj *VHDL Module*, podaj nazwę pliku oraz naciśnij klawisz *Next*.
- 2° W oknie *Define Module* (rys. 8) podaj nazwy jednostki i architektury oraz opisz porty jednostki. Dla każdego portu wpisz jego nazwę, wybierz kierunek oraz, w przypadku magistral, w polach MSB / LSB podaj zakres indeksu sygnałów składowych.
 

☞ Ze względu na możliwości syntezy nie należy stosować innych trybów pracy portów jak dwa podstawowe: *in* oraz *out*.
- 3° Po zatwierdzeniu podanych parametrów nowopowstały plik *.vhd* zostanie automatycznie dodany do projektu i otworzony do edycji. Początkowa treść pliku będzie zawierała definicję jednostki (z podanymi wcześniej portami) oraz definicję jednej, pustej, architektury, jak na rys. 9.
- 4° Szkielet kodu należy uzupełnić wpisując instrukcje VHDL jako treść architektury.

☞ Przy opisie podstawowych układów kombinacyjnych oraz sekwencyjnych należy wzorować się na przykładach zawartych w dokumencie „XST User Guide...”, rozdział „XST HDL Coding Techniques”. Pomocne mogą być też szablony, których bibliotekę otwiera w oknie Nawigatora polecenie Edit → Language Templates... (💡; patrz gałąź VHLD → Synthesis Constructs → Coding Examples).

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

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Cnt3b is
    Port ( Clk : in  STD_LOGIC;
          CE : in  STD_LOGIC;
          Q : out STD_LOGIC_VECTOR (2 downto 0));
end Cnt3b;

architecture Behavioral of Cnt3b is

begin

end Behavioral;
```

Rys. 9 Pusty szablon VHDL wygenerowany dla parametrów jak na rys. 8.

Po wskazaniu modułu VHDL jako głównego pliku projektu, pozostałe kroki projektowe (symulacja, przypisanie wyprowadzeń, implementacja oraz zaprogramowanie układu) wykonuje się identycznie, jak przedstawiono to w rozdziałach 3 i 3.2.3 dla schematów.

## 5.2 Submoduły i projekty heirarchiczne

Dla każdego modułu VHDL można utworzyć jego symbol schematowy uruchamiając dla niego proces *Design Utilities* → *Create Schematic Symbol*. Po umieszczeniu takiego symbolu na schemacie powstaje hierarchia różnych plików źródłowych (SCH na szczycie + VHDL jako submoduł).

Domyślnie każdy symbol jest tworzony jako prostokąt z dołączonymi wyprowadzeniami, które odpowiadają portom jednostki VHDL: po lewej stronie porty WE, po prawej WY. Aby go zmienić należy poddać symbol edycji: w edytorze schematów ECS po jego zaznaczeniu dostępne jest polecenie *Edit* → *Symbol*.

☞ *Podana powyżej procedura tworzenia symboli może być zastosowana również do schematów. Schematy mogą być także submodułami, a liczba poziomów w hierarchii plików źródłowych oraz jej struktura (VHDL vs. schematy) może być dowolna.*

## Źródła

- 1) *ISE DS Software Manuals – PDF Collection: Libraries Guides*, dok. pakietu ISE.
- 2) *ISE DS Software Manuals – PDF Collection: XST User Guide*, dok. pakietu ISE.
- 3) Pomoc on-line aplikacji pakietu ISE.
- 4) *ISE Quick Start Tutorial*, Xilinx Inc. [www.xilinx.com](http://www.xilinx.com).
- 5) *Programmable Logic Design Quick Start Guide* (ug500.pdf), Xilinx Inc. [www.xilinx.com](http://www.xilinx.com)