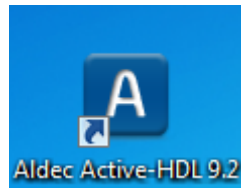
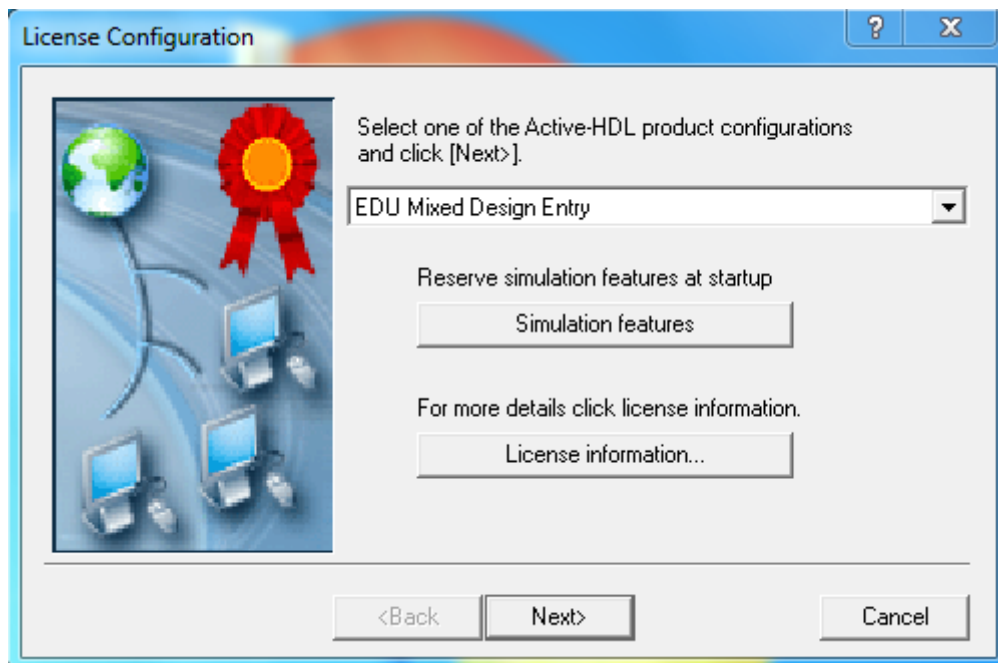


1. Uruchomić „Aldec Active-HDL 9.2” (skrót na pulpicie). Program uruchamia się dosyć długo – cierpliwości.



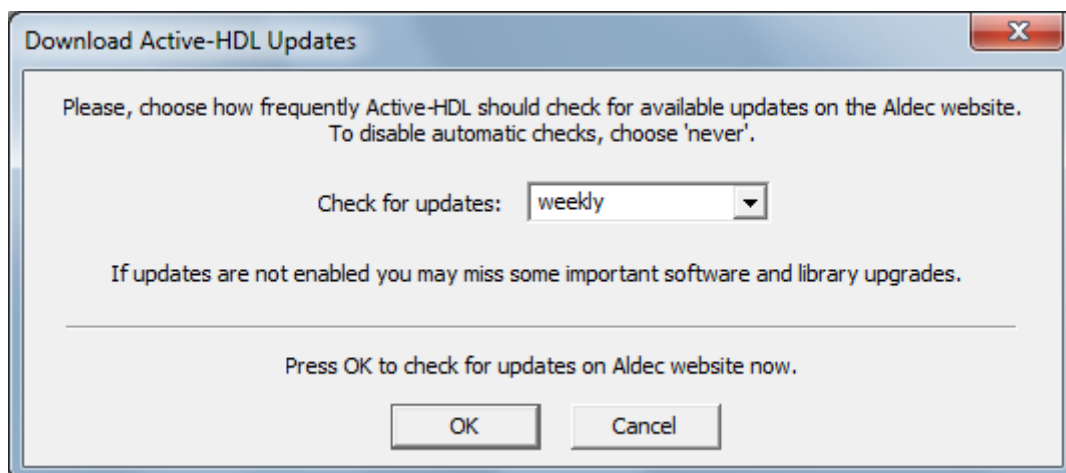
Rys. 1. Ikona na pulpicie – skrót do Aldec Active-HDL.

2. Pojawi się okno z pytaniem o licencję. Kliknąć „Next”.



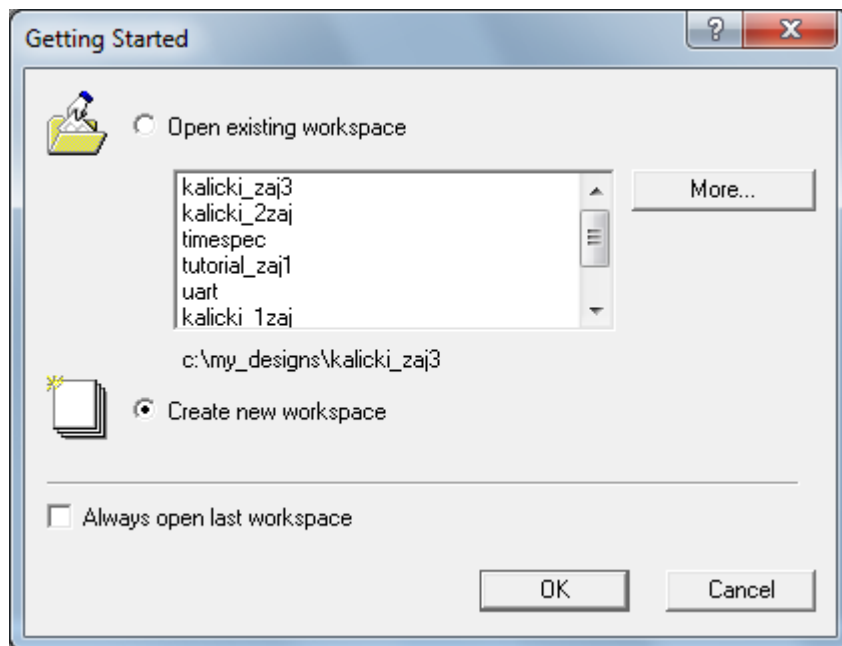
Rys. 2. Okno konfiguracji licencji.

3. Jeżeli program zapyta się o sprawdzenie aktualizacji, to wybrać „Cancel”.



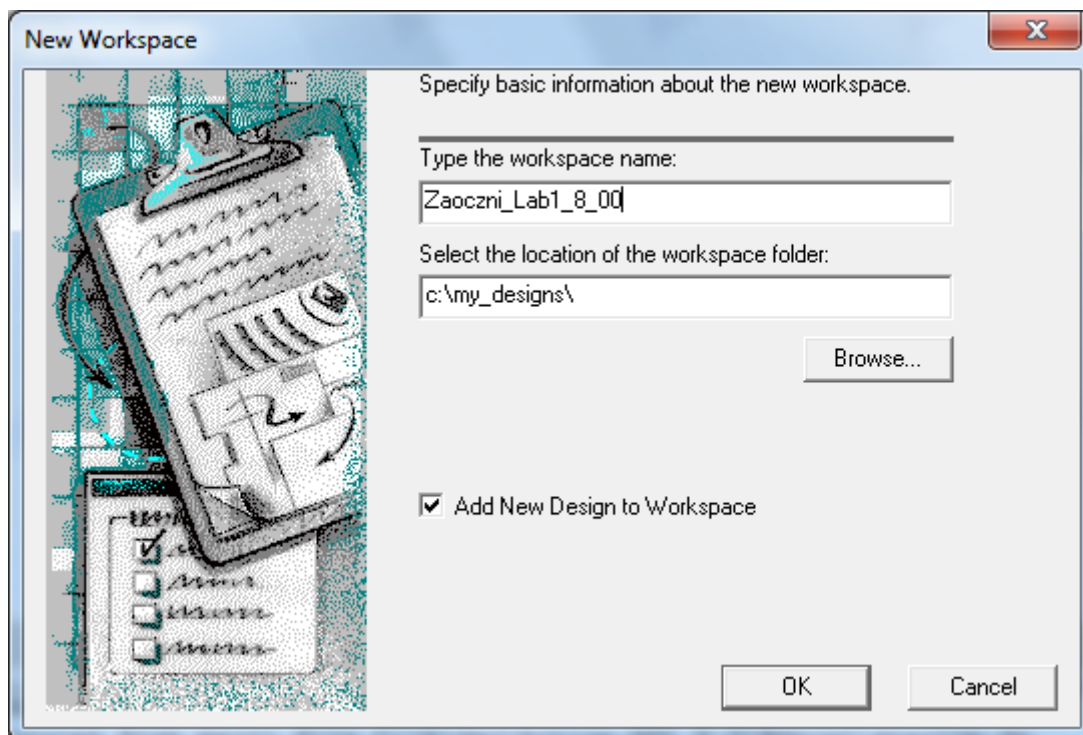
Rys. 3. Okno aktualizacji programu.

4. W oknie wybrać „Create new workspace” i kliknąć „OK”.



Rys. 4. Wybór przestrzeni roboczej.

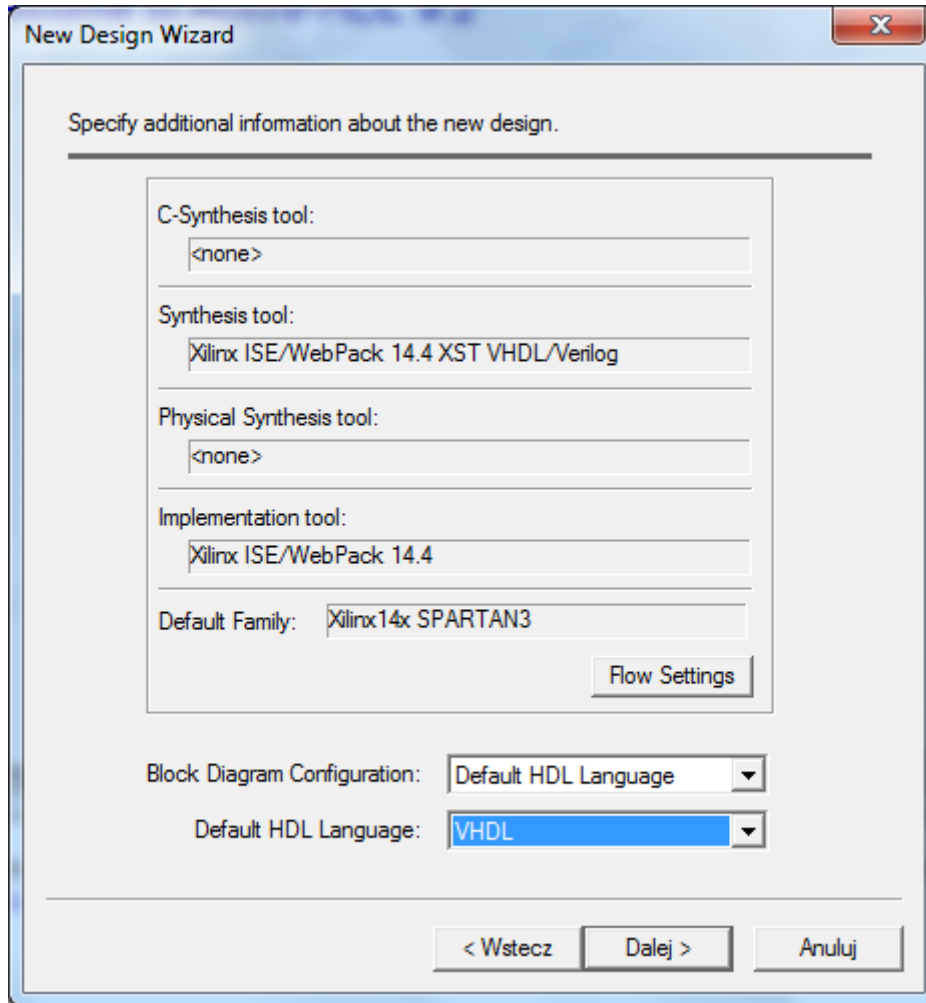
5. Wprowadzić nazwę przestrzeni projektowej (tekst bez odstępów, bez polskich liter oraz bez znaków specjalnych) i kliknąć „OK”.



Rys. 5. Nadanie nazwy przestrzeni roboczej.

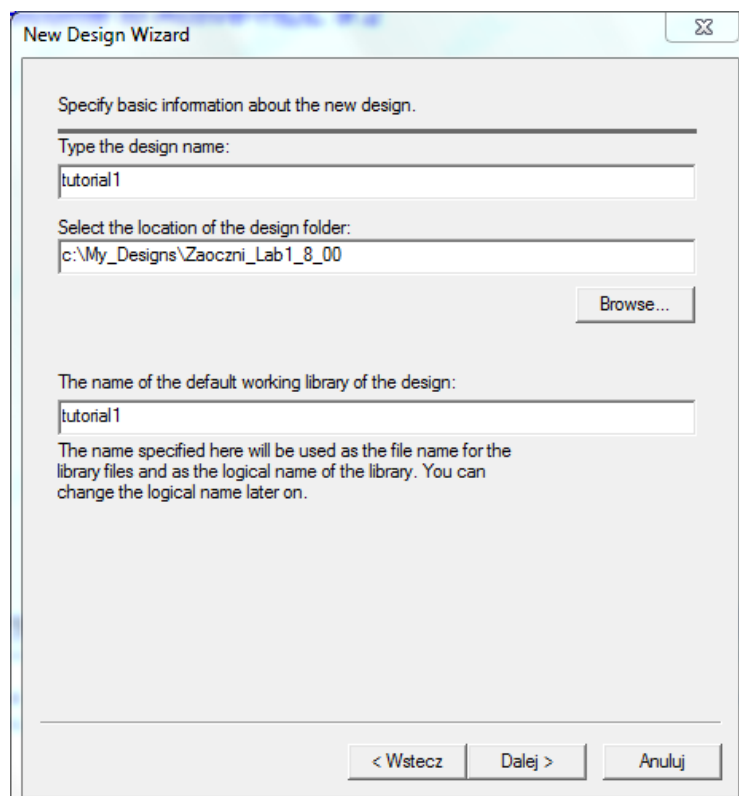
6. W kolejnym oknie wybrać „Create an Empty Design with Design Flow” i kliknąć „Dalej”.

7. Sprawdzić czy ustawienia Design Flow są takie jak na poniższym rysunku. W przypadku różnic kliknąć „Flow Settings” i dokonać odpowiednich zmian. Jeżeli wszystko się zgadza, kliknąć „Dalej”.



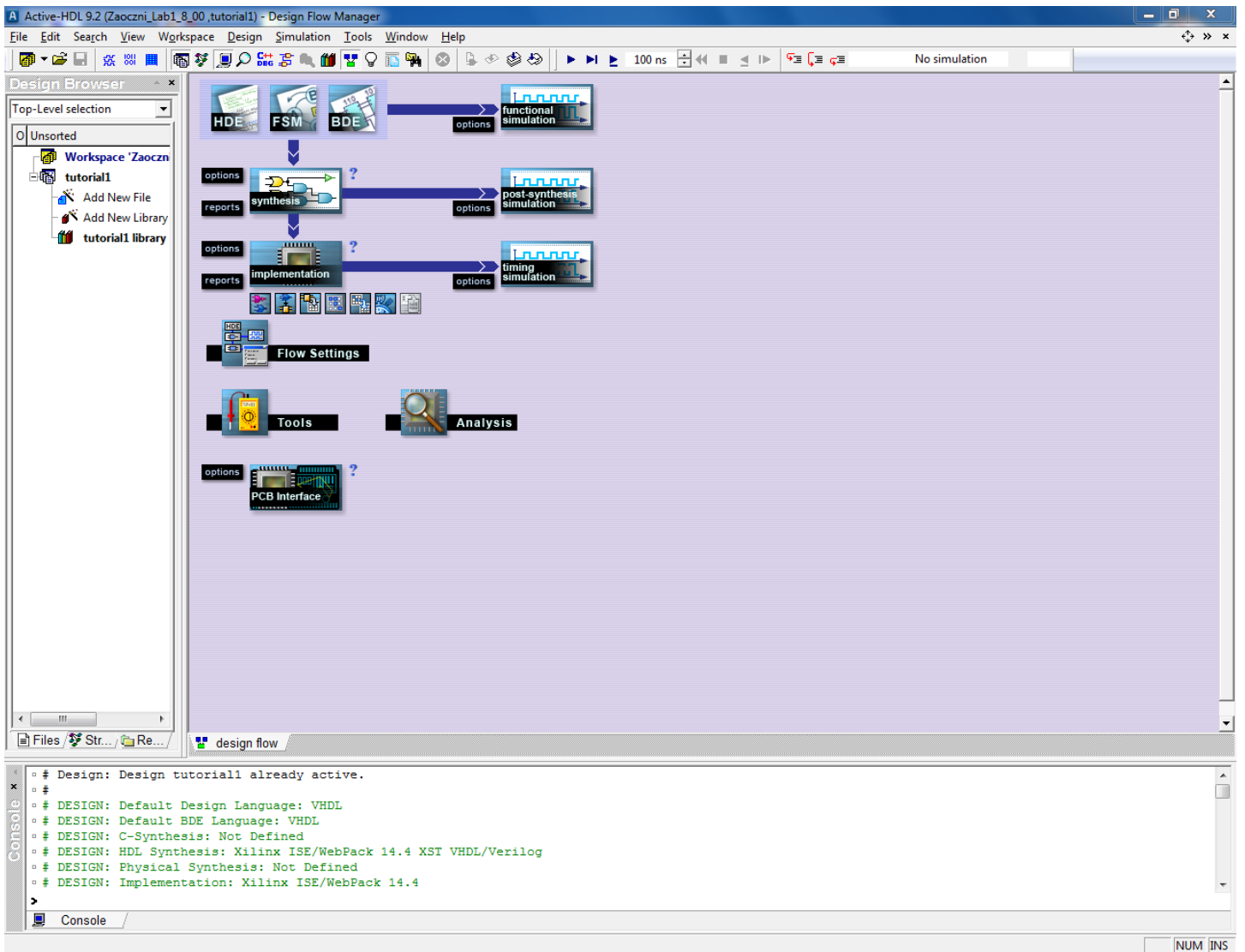
Rys. 6. Ustawienia syntezy i implementacji.

8. Wprowadzić nazwę projektu (tekst bez odstępów, bez polskich liter i znaków specjalnych) i kliknąć „Dalej”.



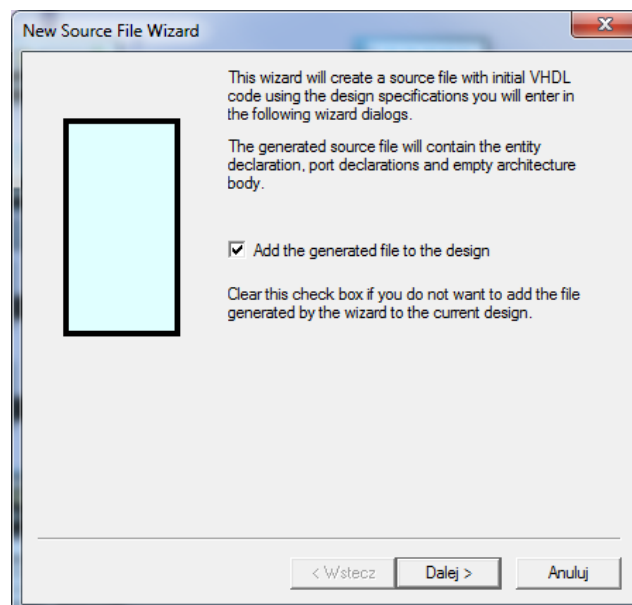
Rys. 7. Nadanie nazwy projektowi.

9. W kolejnym oknie wybrać „Zakończ”. Powinniśmy na ekranie otrzymać widok jak na poniższym rysunku.



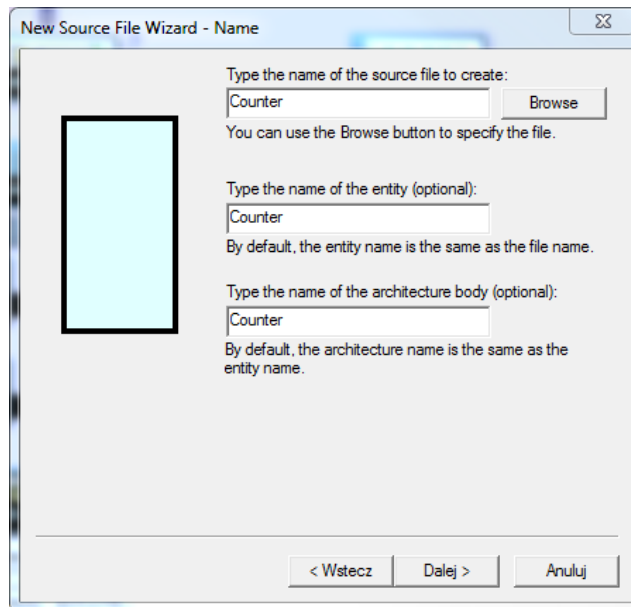
Rys. 8. Główne okno programu po założeniu projektu.

10. Z menu programu wybierz File -> New -> VHDL Source. Otworzy się okno kreatora nowych plików vhd. W pierwszym oknie zostawiamy domyślnie zaznaczoną opcję „Add the generated file to the design” i klikamy „Dalej”.



Rys. 9. Pierwsze okno kreatora plików vhd.

11. Uzupełnij nazwę pliku, nazwę entity oraz architecture tak jak na poniższym rysunku i kliknij „Dalej”.



Rys. 10. Drugie okno kreatora plików vhdl.

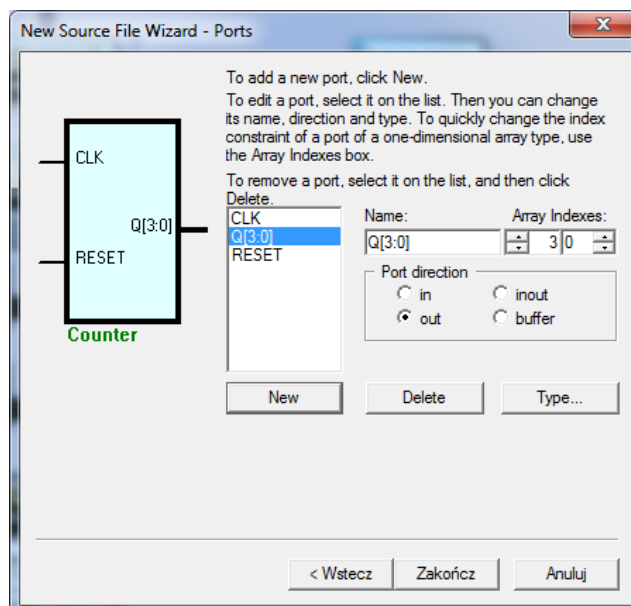
12. Kolejne okno pozwala na dołożenie portów (zdefiniowania sygnałów w entity). Aby dodać nowy port/sygnał należy kliknąć przycisk „New”. W polu „Name” wprowadza się jego nazwę. Pola „Array Indexes” umożliwiają definiowanie magistral. Trzeba też zdefiniować kierunek sygnału (in / out / inout) oraz jego typ (używamy wyłącznie STD\_LOGIC oraz STD\_LOGIC\_VECTOR!).

Proszę dołożyć trzy sygnały zgodnie z poniższą tabelą a następnie kliknąć „Zakończ”.

Name	Array Indexes	Port direction	Type
CLK	(puste)	In	STD_LOGIC
RESET	(puste)	In	STD_LOGIC
Q	3 - 0	Out	STD_LOGIC_VECTOR

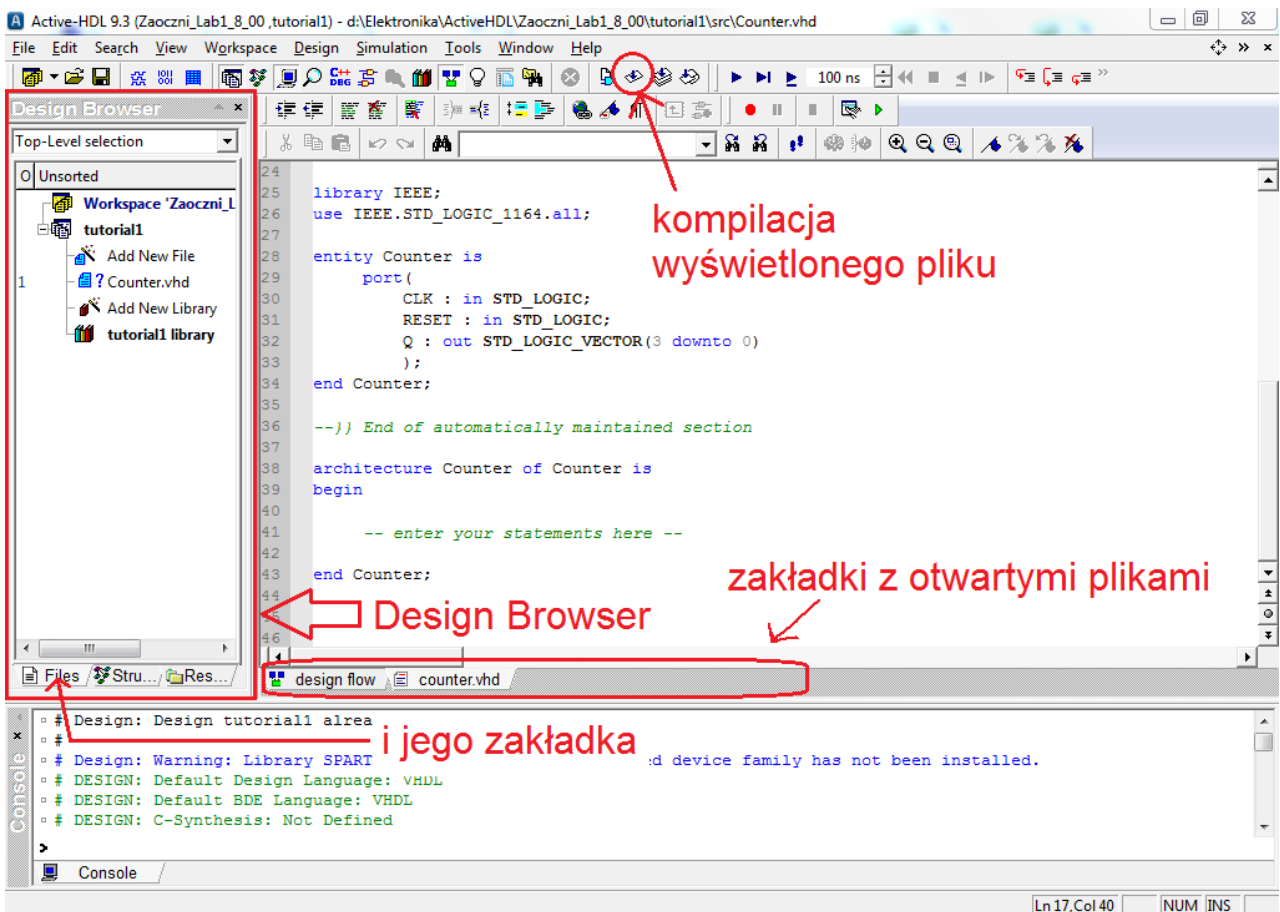
Aby zmodyfikować wcześniej wprowadzony sygnał należy zaznaczyć jego nazwę i dokonać odpowiednich zmian. Błędne / niepotrzebne sygnały można skasować przyciskiem „Delete”.

W lewej części okna mamy graficzną reprezentację projektowanego elementu. Stworzone wejścia układają się na lewej krawędzi bloczka, z kolei wyjścia układu pojawiają się na prawej krawędzi. Magistrale oznaczane są grubszym „przewodem”.



Rys. 11. Konfiguracja portów entity.

13. Po zakończeniu działania kreatora nowego pliku vhd, powinniśmy uzyskać rezultat podobny do poniższego. W lewej części okna jest wyświetlony Design Browser (lista plików w projekcie), prawa część to zawartość aktualnie przeglądane go pliku. Proszę sprawdzić czy mają Państwo identyczną listę sygnałów w entity.



Rys. 12. Główne okno programu Active-HDL po dodaniu do projektu pliku vhd.

14. Skompiluj plik Counter.vhd. Poprawną kompilację pliku symbolizuje ikona  obok nazwy pliku.

Metoda I:

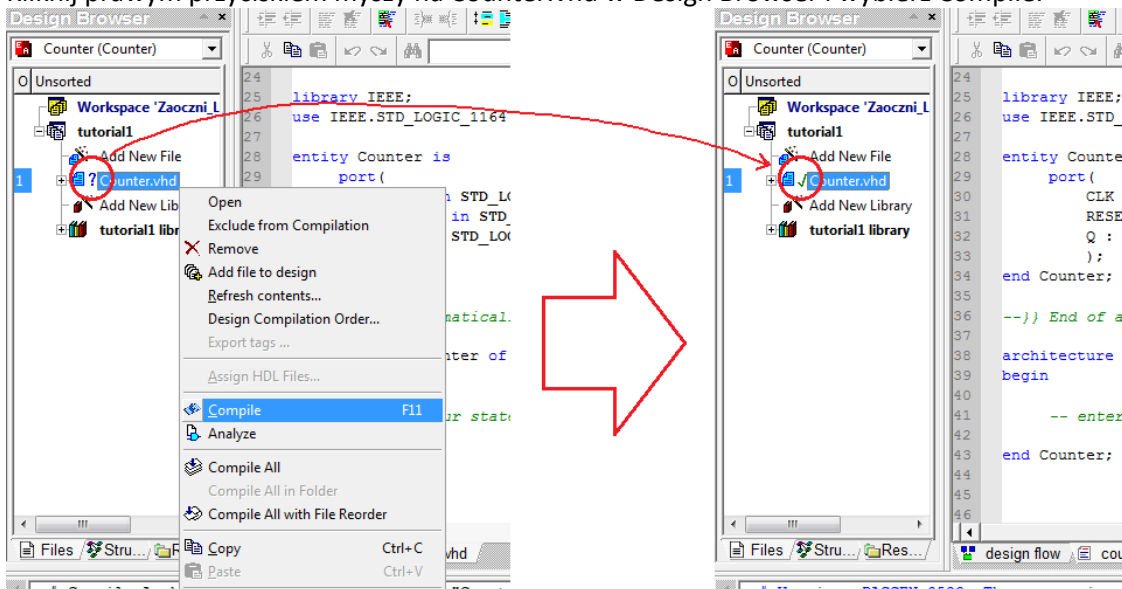
Kliknij na pasku przycisk skrótu zaznaczony na Rysunku 12.

Metoda II:

Wybierz z menu Design -> Compile.

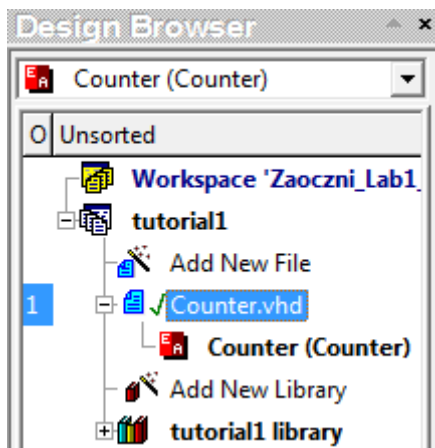
Metoda III:

Kliknij prawym przyciskiem myszy na Counter.vhd w Design Browser i wybierz Compile.



Rys. 13. Jedna z metod kompilacji pliku vhd.

Klikając w plusik można rozwinąć widok struktury projektu. Widoczna jest wtedy para entity-architecture zdefiniowana w danym pliku vhd.



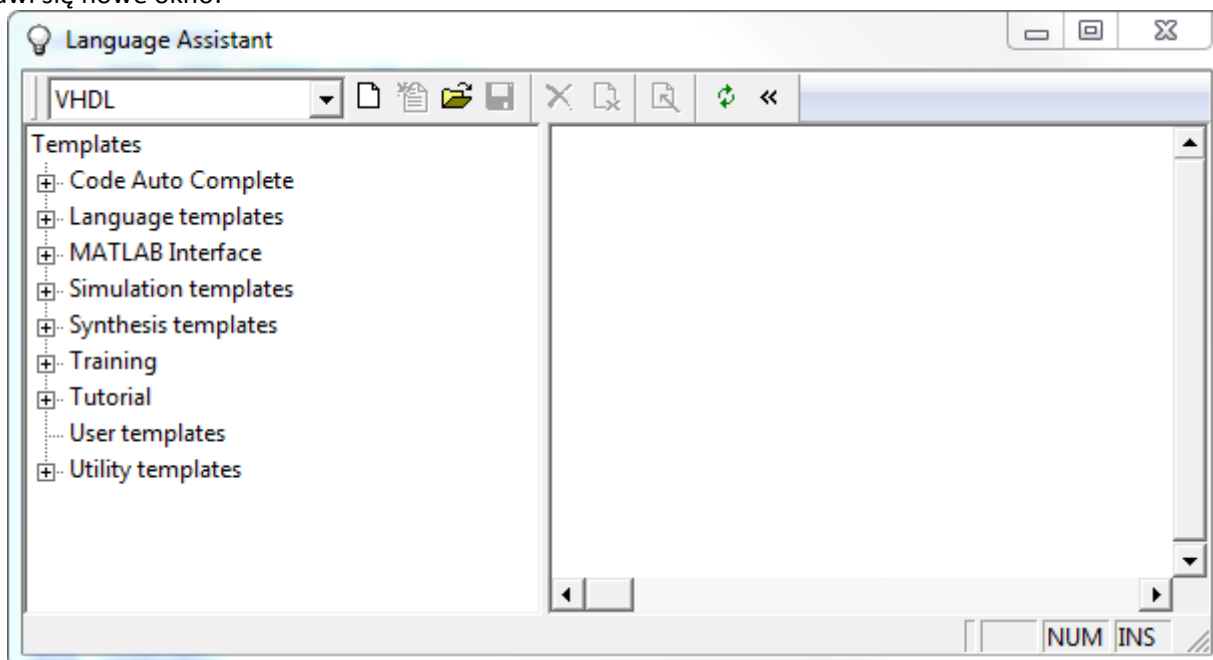
Rys. 14. Design Browser z rozwiniętym widokiem struktury projektu.

15. Edycja kodu opisującego działanie elementu – uzupełnienie architecture. Wykorzystamy gotowiec dostępny w „Language Assistant”.

- a) Wybierz z menu Tools -> Language Assistant lub kliknij na pasku ikonę skrót:



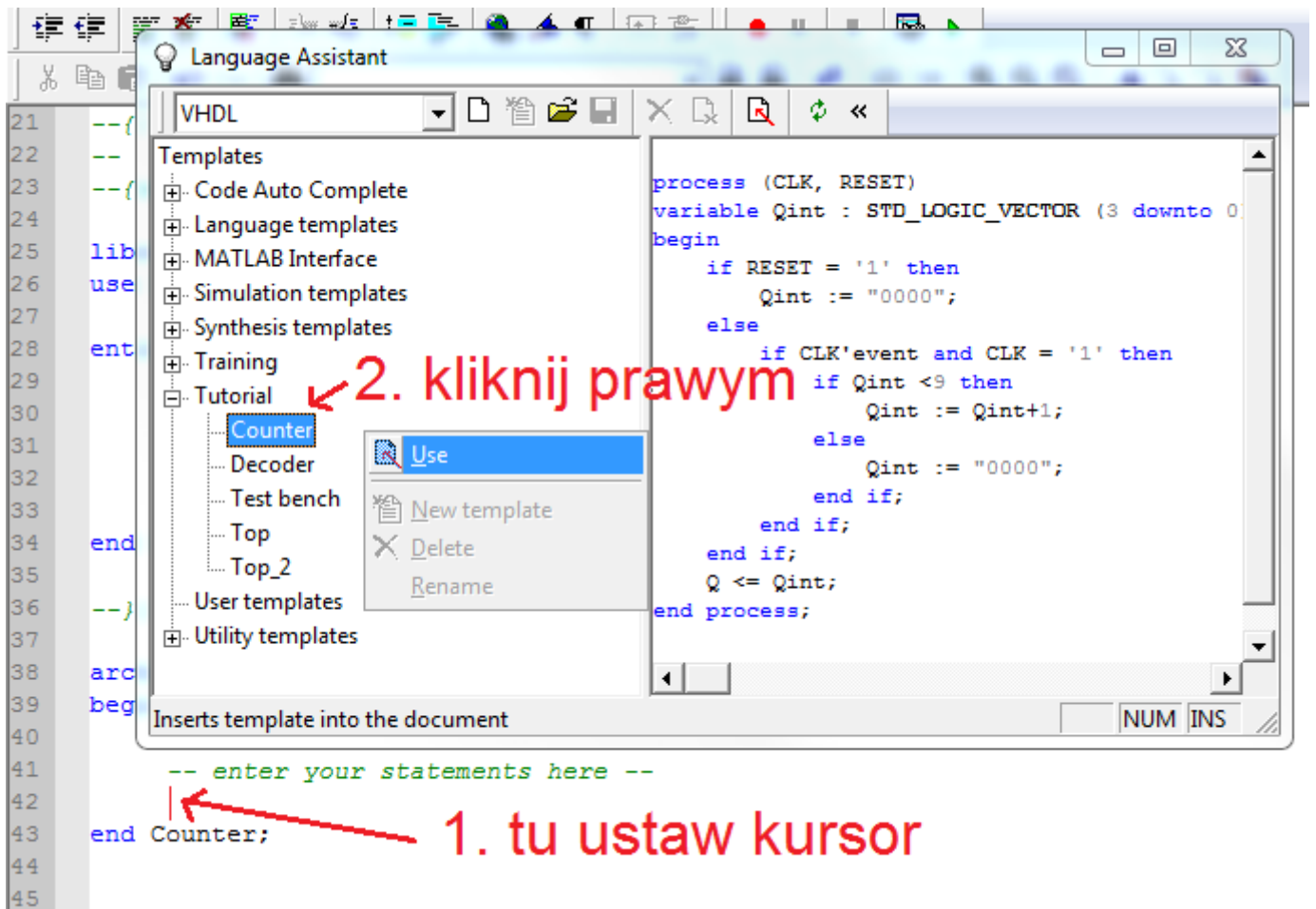
Pojawi się nowe okno:



Rys. 15. Okno Language Assistant.

- b) Rozwiń pole „Tutorial”.
- c) Zaznacz element „Counter”. W prawej części okna pojawi się kod vhd opisujący działanie 4-bitowego licznika modulo 10 z asynchronicznym resetem.
- d) Na chwilę zminimalizuj okno Language Assistant i w pliku „Counter.vhd” ustaw kursor w miejscu w którym zostanie wklejony kod z Language Assistant.  
Chcemy ustawić kursor w kolejnej linii po komentarzu:  
*-- enter your statements here --*  
(jeżeli nie dokonano żadnych zmian w pliku Counter.vhd, to kursor należy ustawić w linii 42)
- e) Przywróć okno Language Assistant, kliknij prawym przyciskiem myszy na “Counter” i wybierz „Use”.  
(poniżej zrzut ekranu)





Rys. 16. Wstawienie kodu vhdl z Language Assistant.

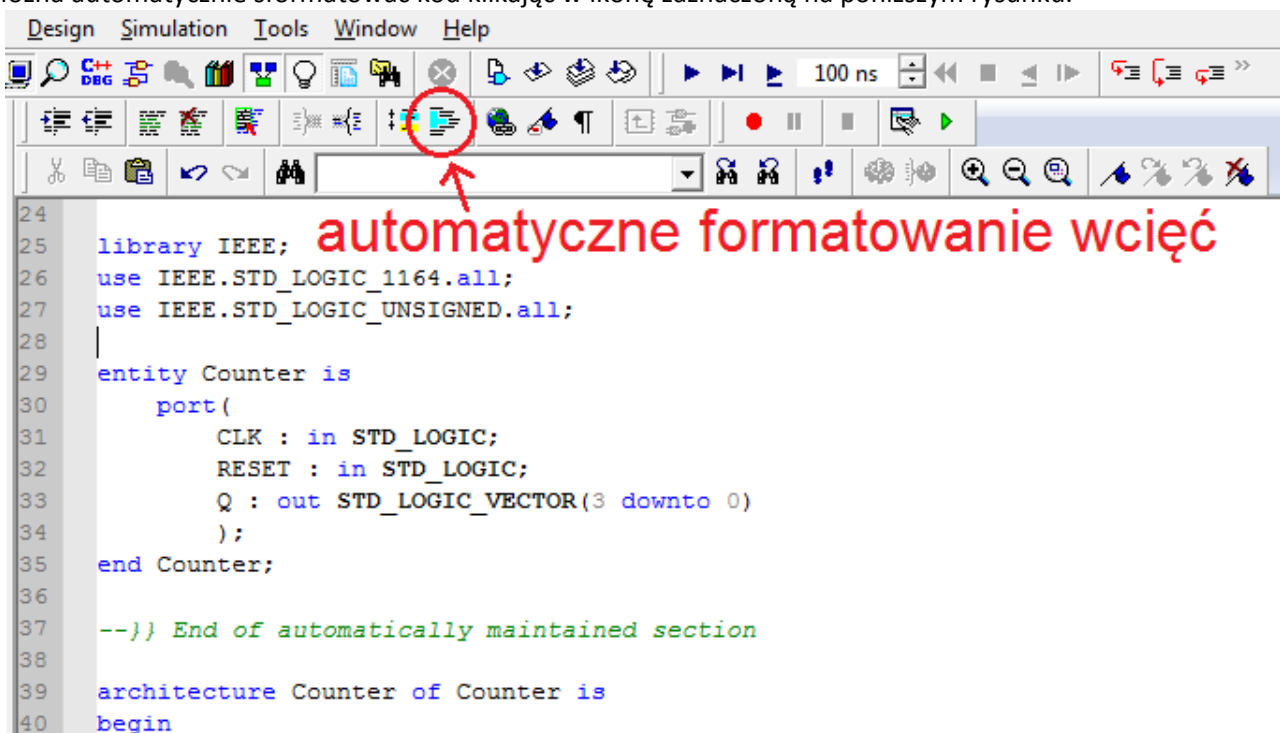
f) W tym momencie Language Assistant może zostać zamknięty lub zminimalizowany (będziemy jeszcze z niego korzystać).

g) Aby kod się skompilował **musimy dołączyć pakiet**. Pakiety dołącza się poza ciałem entity oraz poza ciałem architecture.

Przewijamy w górę plik Counter.vhd. W okolicach linii 25 powinniśmy znaleźć dołączenie biblioteki IEEE. Poniżej wstawiamy kod:

```
use IEEE.STD_LOGIC_UNSIGNED.all;
```

h) Można automatycznie sformatować kod klikając w ikonę zaznaczoną na poniższym rysunku.



Rys. 17. Kod vhdl po dołączeniu pakietu STD\_LOGIC\_UNSIGNED.



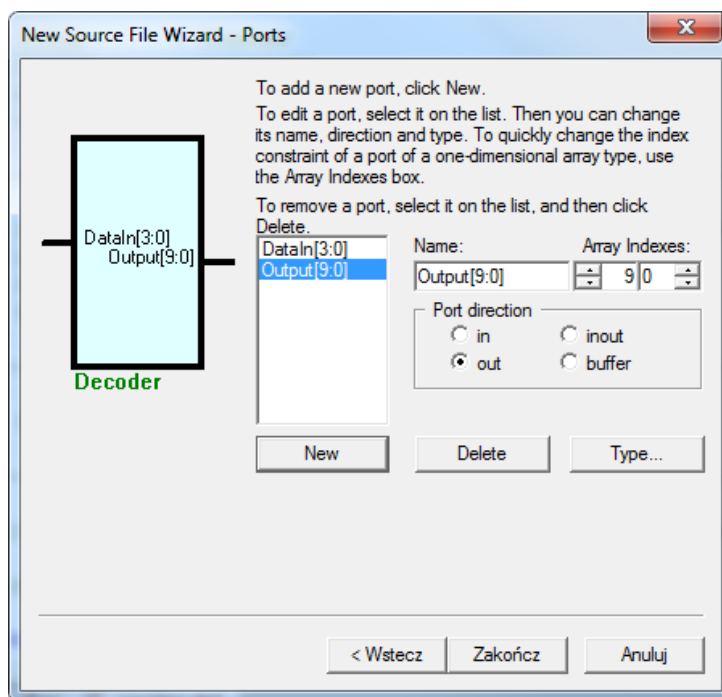
- i) Skompiluj plik podobnie jak w punkcie 14 instrukcji – kompilacja powinna zakończyć się sukcesem. Jeżeli plik źródłowy zawiera błędy, to obok jego nazwy pojawia się czerwony krzyżyk. Dodatkowo, błędna linia kodu jest podświetlona i w konsoli otrzymujemy opis błędu. Jeżeli pojawią się błędy, to je rozwiąż (samodzielnie lub z pomocą prowadzącego).

16. Stwórz nowy plik “Decoder.vhd”. Postępuj podobnie jak w punktach 10 – 12, czyli:

- Wybierz z menu File -> New -> VHDL Source.
- Jako nazwę pliku podaj „Decoder”. (Domyślnie nazwa entity jest taka sama jak nazwa pliku. Domyślnie nazwa architecture jest taka sama jak nazwa entity).
- Dodaj komponentowi porty / sygnały tak jak w poniższej tabelce.

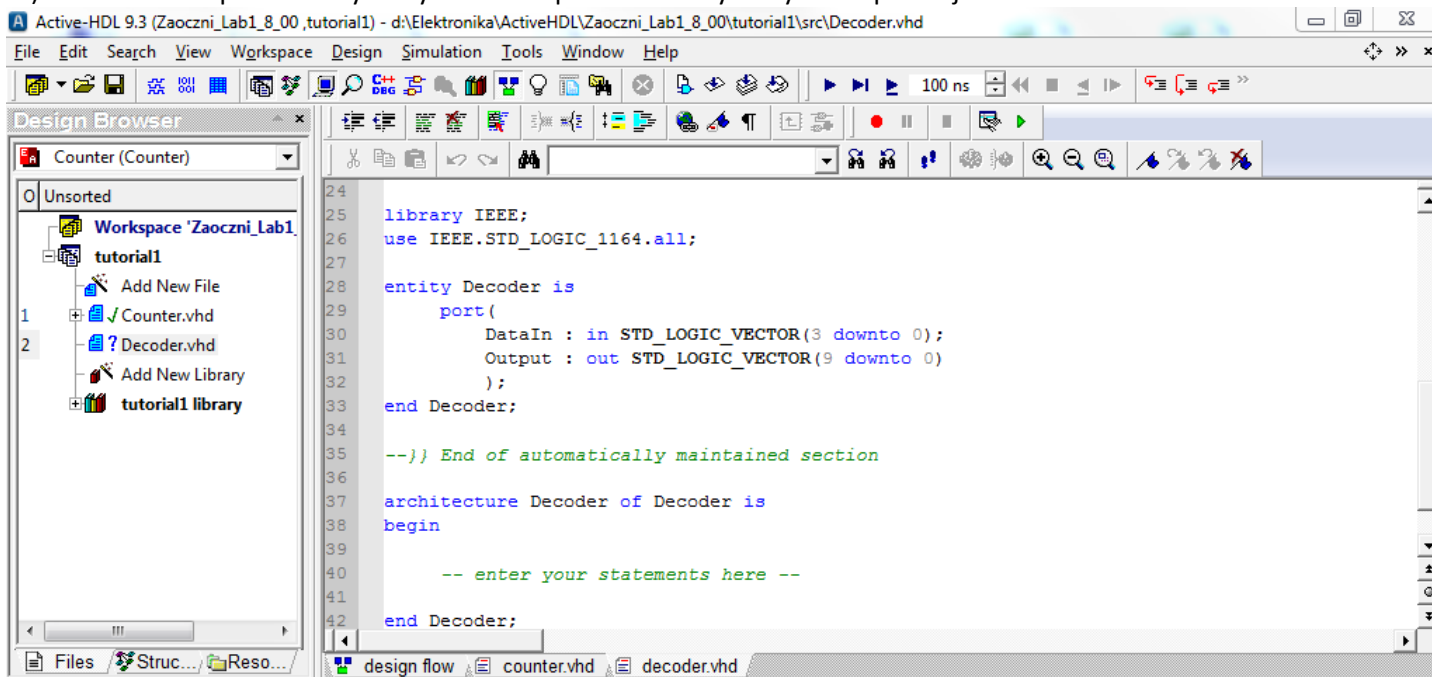
Name	Array Indexes	Port direction	Type
DataIn*	3 - 0	In	STD_LOGIC_VECTOR
Output	9 - 0	Out	STD_LOGIC_VECTOR

\*I jak Irena; to nie jest małe I jak Leszek ;D



Rys. 18. Porty komponentu Decoder.

- W rezultacie powinniśmy otrzymać efekt przedstawiony na rysunku poniżej.

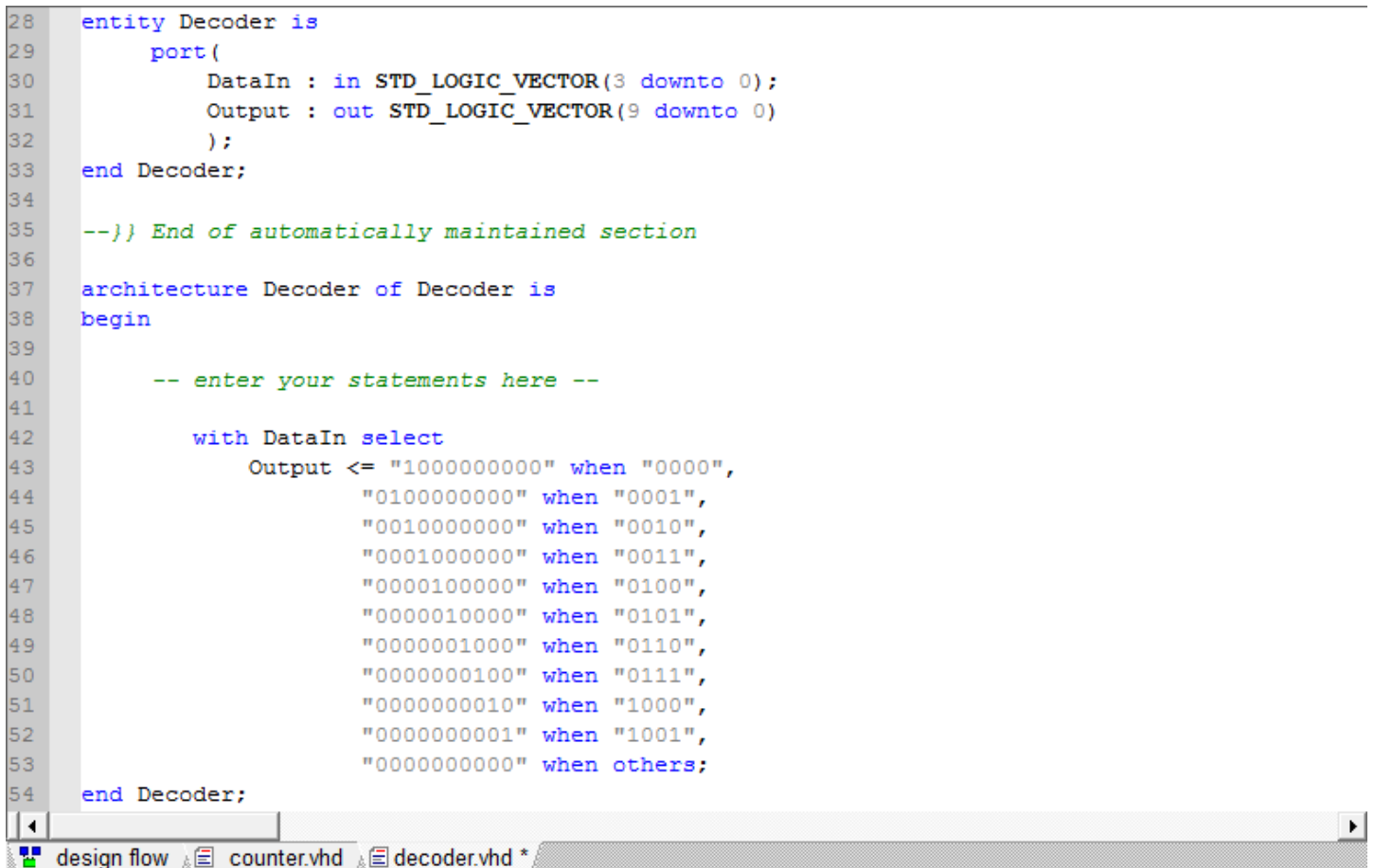


Rys. 19. Zawartość pliku Decoder.vhd.

17. Uzupełniamy architecture dekodera korzystając z gotowego kodu w Language Assistant.

- a) W pliku Decoder.vhd umieść kursor w następczej linii po komentarzu  
*-- enter your statements here --*  
(jeżeli nie dokonano wcześniej żadnych zmian to kursor należy ustawić w linii 41)
- b) Otwórz (lub przywróć) okno Language Assistant podobnie jak w punkcie 15-a).
- c) Rozwiń pole „Tutorial”.
- d) Zaznacz element „Decoder”. W prawej części okna pojawi się kod vhdl opisujący działanie dekodera kodu naturalnego binarnego na kod „1 z 10”. W danej chwili czasu, tylko jeden bit magistrali wyjściowej jest w stanie wysokim a pozostałe bity są wyzerowane. W stanie wysokim jest bit magistrali wyjściowej o numerze podanym na magistrali wejściowej w postaci liczby binarnej.
- e) Kliknij prawym przyciskiem myszy na “Decoder” i wybierz „Use”. Poniżej oczekiwany efekt.

```
28 entity Decoder is
29     port (
30         DataIn : in STD_LOGIC_VECTOR(3 downto 0);
31         Output  : out STD_LOGIC_VECTOR(9 downto 0)
32     );
33 end Decoder;
34
35 --}) End of automatically maintained section
36
37 architecture Decoder of Decoder is
38 begin
39
40     -- enter your statements here --
41
42     with DataIn select
43         Output <= "1000000000" when "0000",
44                 "0100000000" when "0001",
45                 "0010000000" when "0010",
46                 "0001000000" when "0011",
47                 "0000100000" when "0100",
48                 "0000010000" when "0101",
49                 "0000001000" when "0110",
50                 "0000000100" when "0111",
51                 "0000000010" when "1000",
52                 "0000000001" when "1001",
53                 "0000000000" when others;
54 end Decoder;
```

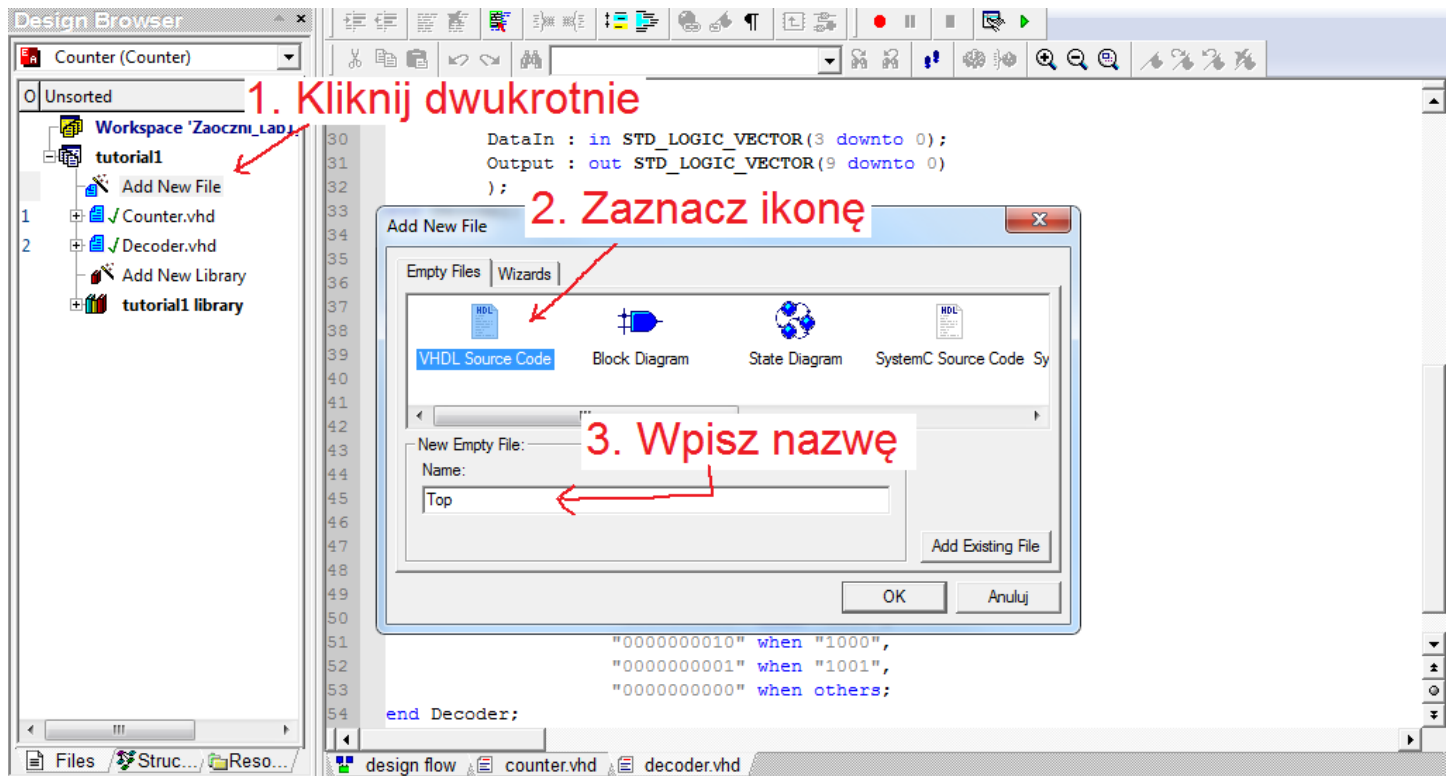


Rys. 20. Poprawna zawartość pliku Decoder.vhd.

- f) W tym momencie Language Assistant może zostać zamknięty lub zminimalizowany (będziemy jeszcze z niego korzystać).
- g) Zapisz i skompiluj plik Decoder.vhd. Popraw ewentualne błędy.

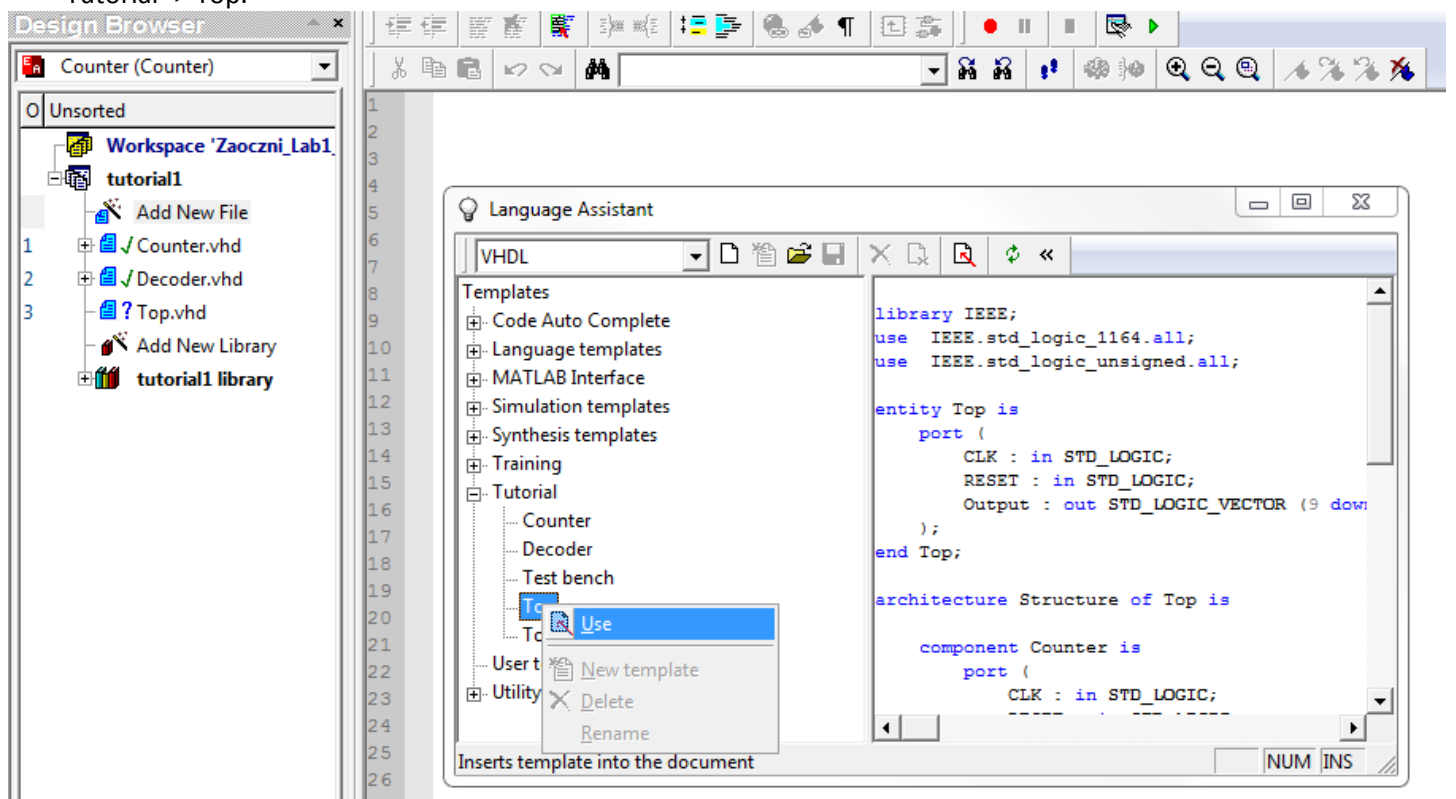
18. Tworzymy kolejny (główny) plik projektu w którym umieścimy licznik i dekodera a następnie je ze sobą połączymy. Tym razem nie będziemy korzystać z kreatora tworzenia nowych plików vhdl – stworzymy zupełnie pusty plik, a następnie uzupełnimy jego treść.

- a) W Design Browser kliknij dwukrotnie lewym przyciskiem myszy „Add New File”.
- b) W nowo otwartym oknie zaznacz ikonę „VHDL Source Code” i wpisz nazwę pliku „Top”. Zamknij okno klikając „OK”.  
(zrzut poniżej)



Rys. 21. Dodawanie nowego, pustego pliku vhd.

- c) Ustaw kursor na początku pierwszego wiersza pliku „Top.vhd” i z Language Assistant wstaw kod znajdujący się w Tutorial -> Top.



Rys. 22. Uzupełnienie pliku Top.vhd kodem z Language Assistant.

```

1
2  library IEEE;
3  use IEEE.std_logic_1164.all;
4  use IEEE.std_logic_unsigned.all;
5
6  entity Top is
7      port (
8          CLK : in STD_LOGIC;
9          RESET : in STD_LOGIC;
10         Output : out STD_LOGIC_VECTOR (9 downto 0)
11     );
12 end Top;
13
14 architecture Structure of Top is
15
16     component Counter is
17         port (
18             CLK : in STD_LOGIC;
19             RESET : in STD_LOGIC;
20             Q : out STD_LOGIC_VECTOR (3 downto 0)
21         );
22     end component;
23
24     component Decoder is
25         port (
26             DataIn : in STD_LOGIC_VECTOR (3 downto 0);
27             Output : out STD_LOGIC_VECTOR (9 downto 0)
28         );
29     end component;
30
31     signal Internal : STD_LOGIC_VECTOR (3 downto 0);
32
33 begin
34
35     CNT : Counter
36     port map (CLK,RESET,Internal);
37
38     DEC : Decoder
39     port map (Internal, Output);
40
41 end structure;
42

```

Dołączamy bibliotekę i jej pakiety

Deklarujemy komponent Counter

Deklarujemy komponent Decoder

Łączymy komponenty ze sobą korzystając z funkcji port map i notacji pozycyjnej

Rys. 23. Zawartość i opis pliku „Top.vhd”.

d) Zapisz plik Top.vhd a następnie skompiluj cały projekt. Sprawdź czy kompilacja zakończyła się sukcesem.

*Metoda I:*

Kliknij na pasku przycisk skrótu zaznaczony na Rysunku 24.



Rys. 24. Ikona skrótu „Compile All”.

*Metoda II:*

Wybierz z menu Design -> Compile All.

*Metoda III:*

Kliknij prawym przyciskiem myszy na dowolny plik w Design Browser i wybierz Compile All.

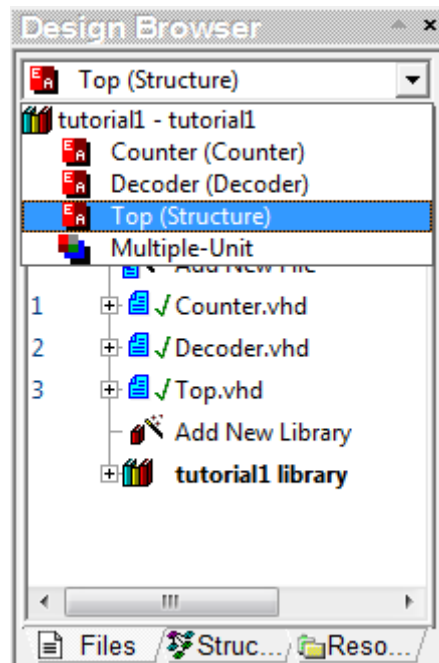
19. Symulacja projektu (w podobny sposób można przesymulować dowolną parę entity-architecture).

- a) Wybieramy co chcemy symulować poprzez wskazanie pary entity-architecture jako „Top-Level”. Chcemy przesymulować działanie komponentu „Top”.

**Aby móc wskazać komponent jako „Top-Level” należy go wcześniej skompilować.**

*Metoda I:*

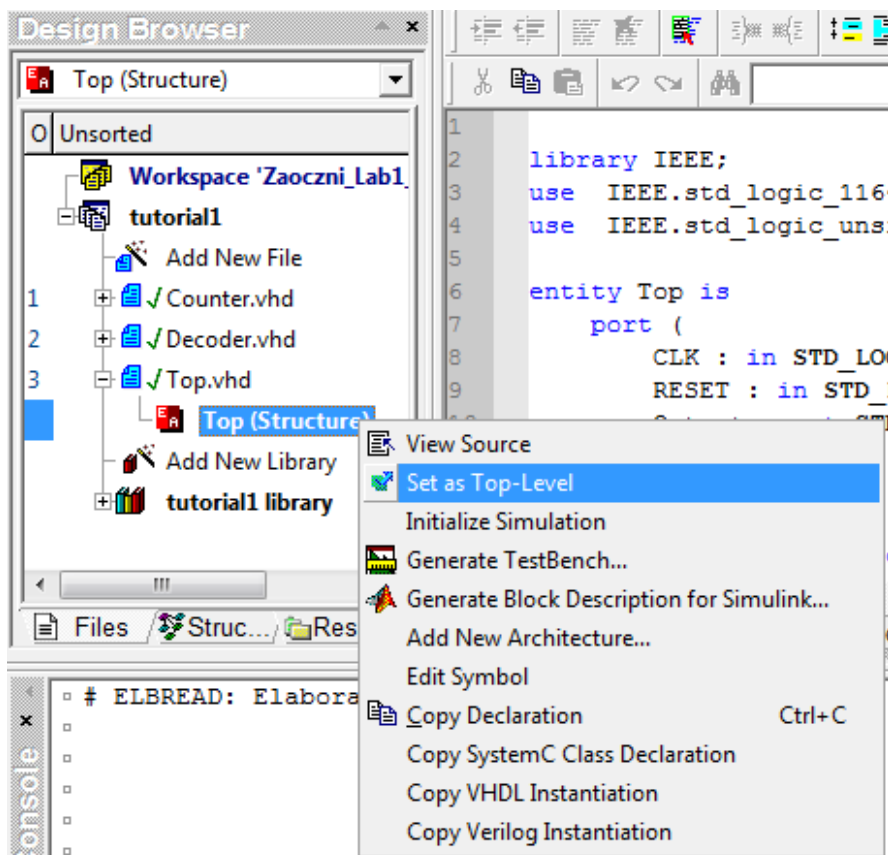
Od samej góry okna Design Browser jest rozwijana lista. Wybieramy w niej „Top (Structure)”.



Rys. 25a. Wskazanie komponentu jako „Top-Level” w Design Browser.

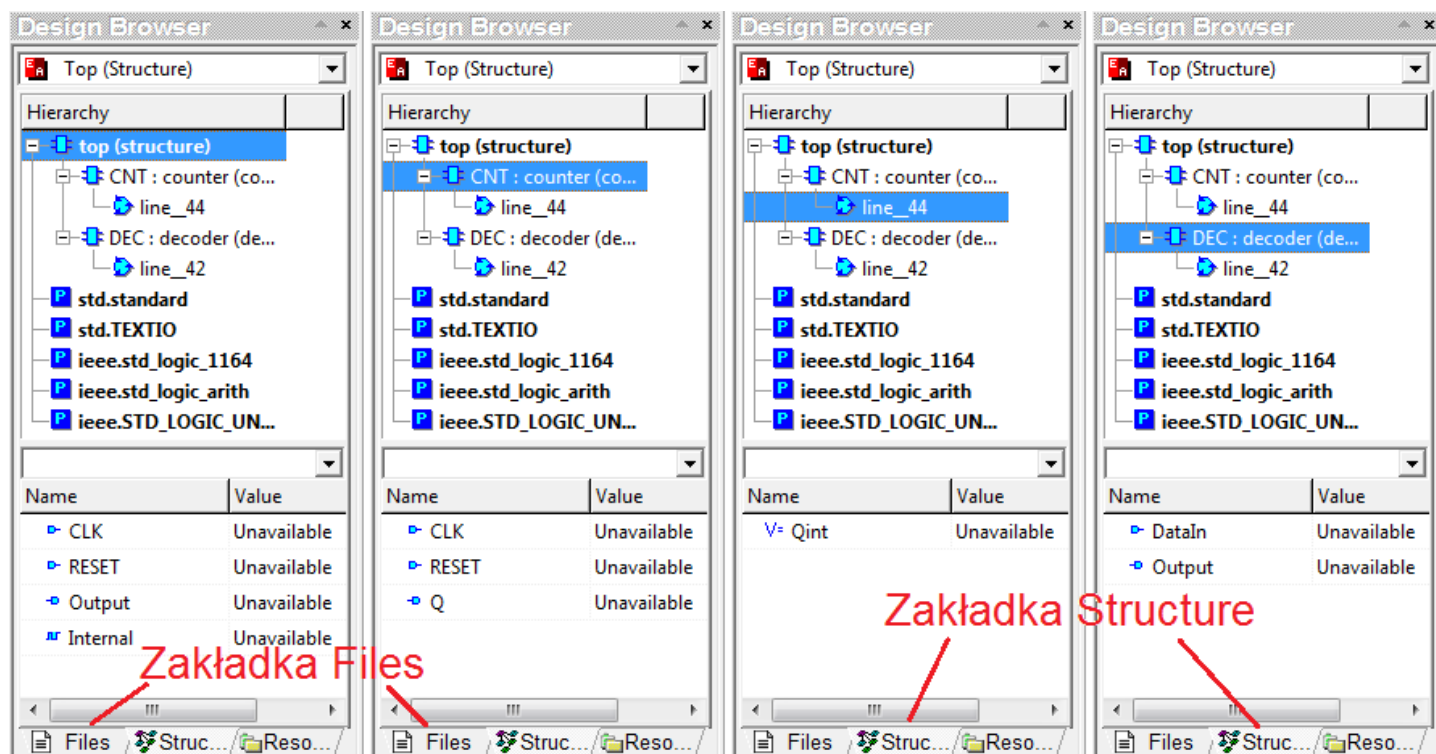
*Metoda II:*

W Design Browser rozwin plusik obok Top.vhd. Pojawi się w drzewie projektu kolejna pozycja którą klikamy prawym przyciskiem myszy i z menu podręcznego wybieramy „Set as Top-Level”.



Rys. 25b. Wskazanie komponentu jako „Top-Level” w Design Browser.

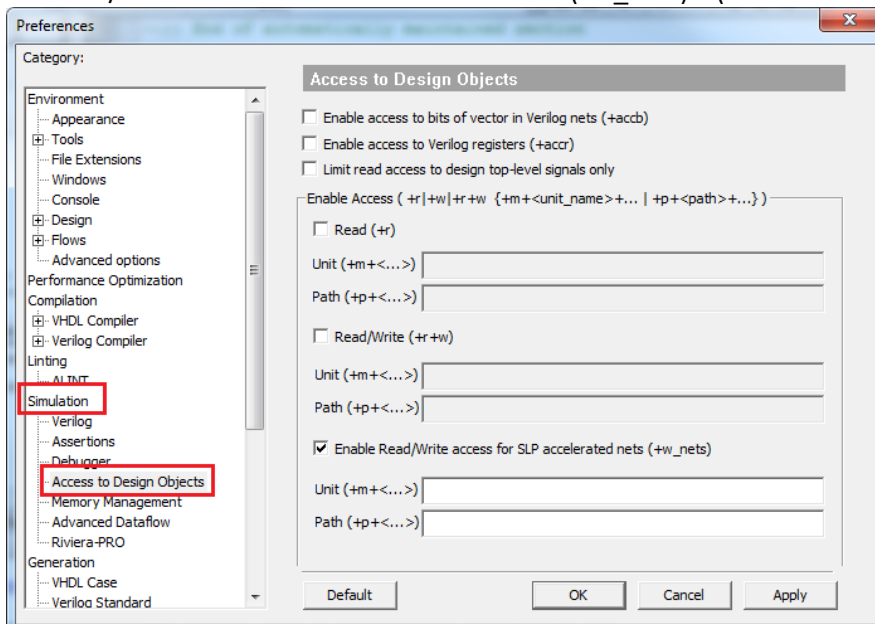
W Design Browser możemy też obejrzeć strukturę projektu. W tym celu należy zmienić zakładkę z „Files” na „Structure”. W naszym przypadku komponent „Top” zawiera w sobie komponent „Counter” oraz komponent „Decoder”. Dodatkowo program podaje numer wiersza w którym znajdują się instrukcje współbieżne (w Counter jest to proces w linii 44, natomiast w Decoder jest to współbieżne przypisanie decyzyjne w linii 42). W dolnej części Design Browser możemy obserwować porty, sygnały i zmienne zdefiniowane na danym poziomie hierarchii (po zaznaczeniu Counter pojawiają się sygnały licznika, po zaznaczeniu procesu w linii 44 pojawia się zmienna tam zdefiniowana, itd.).



Rys. 26. Struktura komponentu wskazanego jako Top-Level.

Domyślnie Active-HDL jest skonfigurowany tak by osiągał maksymalną wydajność w trakcie symulacji. Jest to osiągnięte poprzez wprowadzenie pewnych ograniczeń (np. liczby sygnałów których poziom logiczny jest zapisywany). Aby umożliwić debugowanie i obserwowanie wszystkich sygnałów projektu (co odbywa się kosztem wydajności) konieczne jest wyłączenie tych ograniczeń.

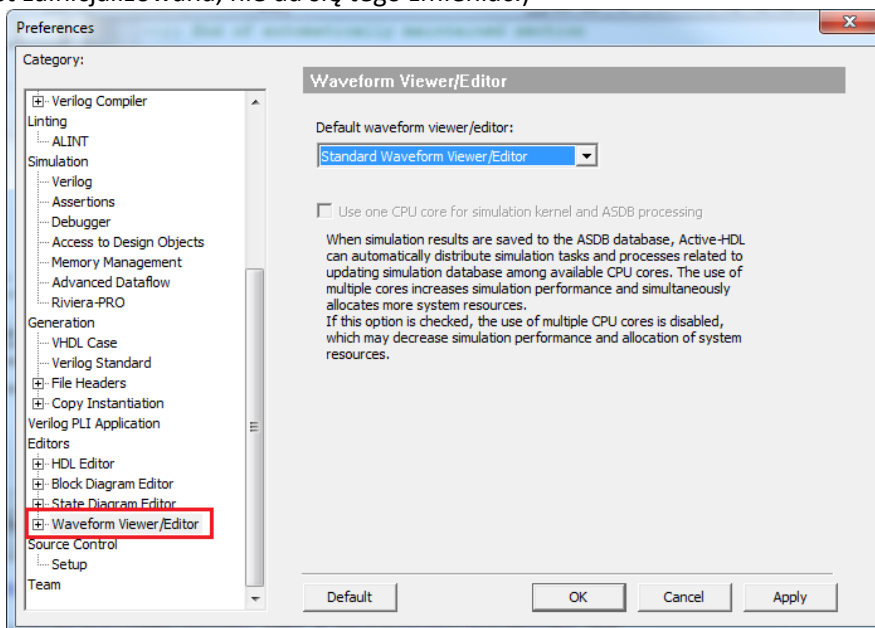
- b) (Jednorazowa) konfiguracja programu Active-HDL. Wybierz z głównego menu Tools -> Preferences...  
 W nowym oknie rozwiń pozycję „Simulation” i wskaż „Access to Design Objects”.  
 Odznacz „Limit read access to design top-level signals only”.  
 Zaznacz za to „Enable Read/Write access for SLP accelerated nets (+w\_nets)”. (zrzut ekranu poniżej)



Rys. 27. Zaawansowane ustawienia symulacji.

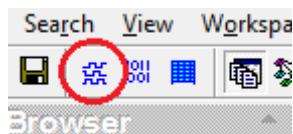


- c) W oknie ustawień zaznacz pozycję „Waveform Viewer/Editor” i upewnij się że jest wybrany „Standard Waveform Viewer/Editor”. Kliknij „OK” aby zamknąć okno ustawień.  
(Gdy symulacja jest zainicjalizowana, nie da się tego zmieniać.)



Rys. 28. Zaawansowane ustawienia symulacji - cd.

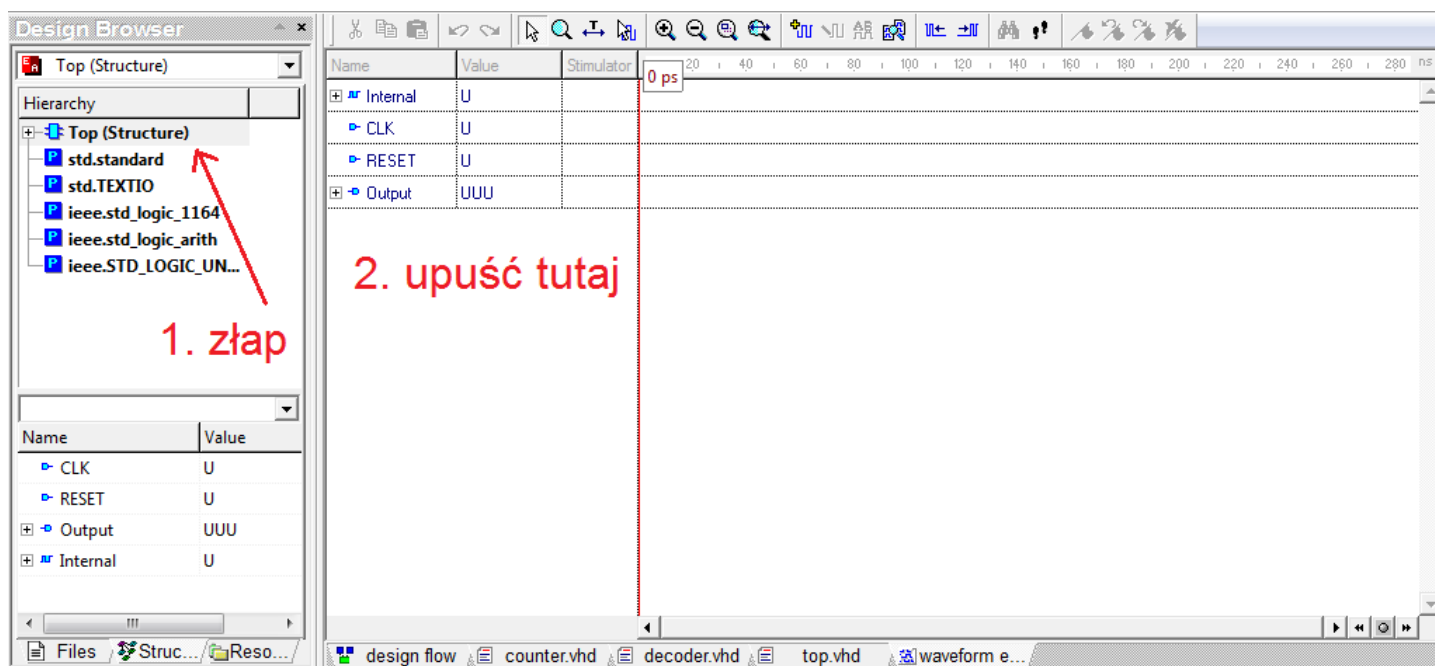
- d) Z menu głównego wybierz Simulation -> Initialize Simulation.  
e) Korzystając z ikony skrótu otwórz New Waveform.



Rys. 29. Ikona skrótu New Waveform.

W prawej części okna programu otworzy się nowa zakładka z możliwością oglądania przebiegów.

- f) W Design Browser, zakładka Structure, zaznacz „Top (Structure)” i trzymając wciśnięty lewy przycisk myszy przenieś go do okna New Waveform (tak jak się przesuwa ikony na pulpicie).



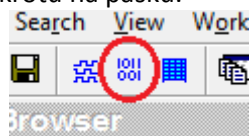
Rys. 30. Okno symulatora z wybranymi sygnałami.

Niepotrzebny sygnał można usunąć zaznaczając go i wciskając klawisz Delete.



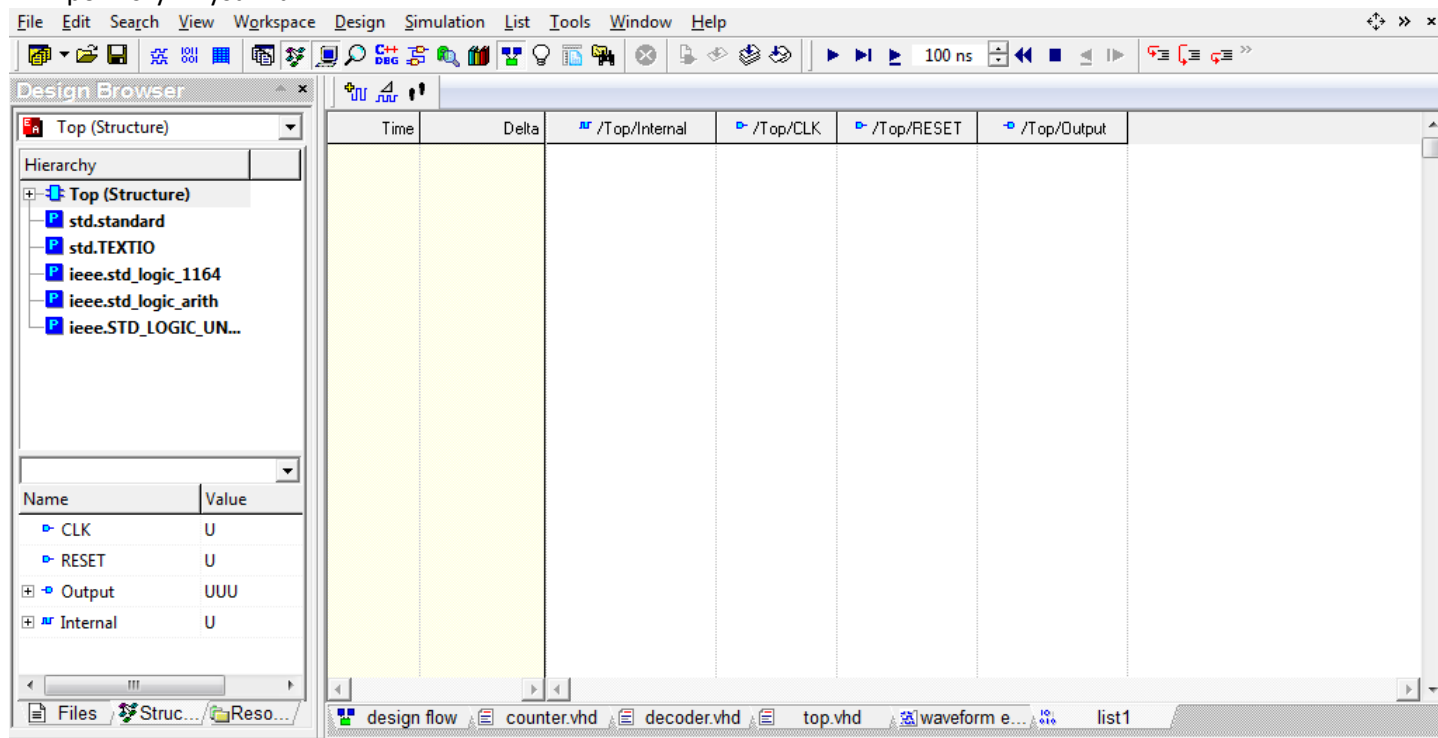
Active-HDL umożliwia również oglądanie wyników symulacji w postaci tabelarycznej z rozdzielczością „delta time” (por. wykład 5, slajd 25). Okno Standard List Viewer umożliwia monitorowanie wartości sygnałów bez możliwości ich zmieniania.

g) Otwórz Standard List Viewer korzystając ze skrótu na pasku.



Rys. 31. Ikona skrótu Standard List Viewer.

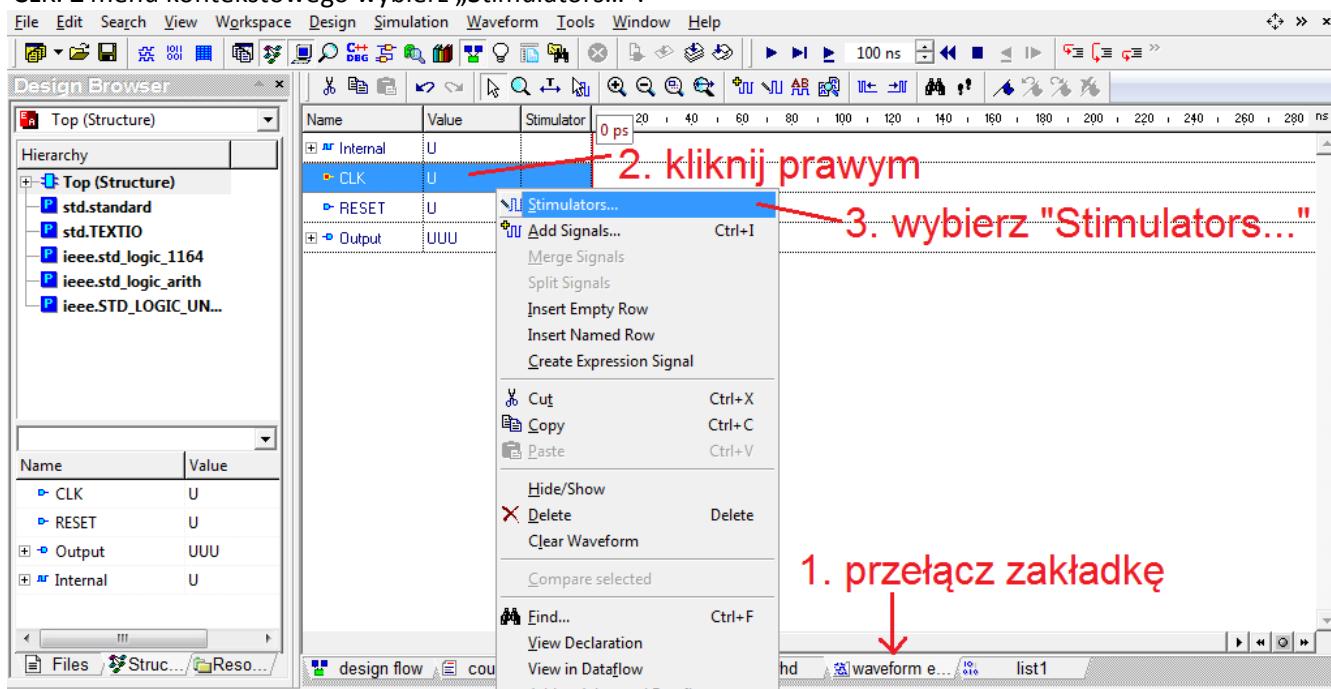
h) Dołącz sygnały do Standard List Viewer podobnie jak w podpunkcie f). Powinieneś uzyskać efekt przedstawiony na poniższym rysunku.



Rys. 32. Standard List Viewer z dołożonymi sygnałami.

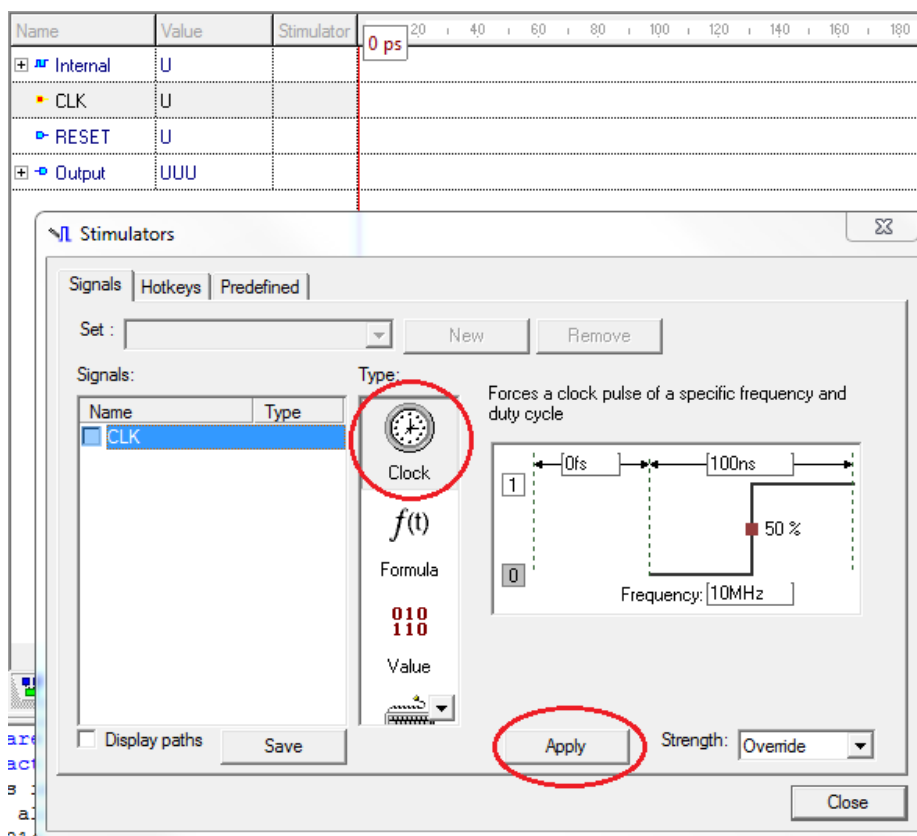
i) Przypisz stymulator w postaci sygnału zegarowego do sygnału CLK.

W tym celu przełącz się z powrotem na zakładkę z Wawewform. Kliknij prawym przyciskiem myszy sygnał o nazwie CLK. Z menu kontekstowego wybierz „Stimulators...”.



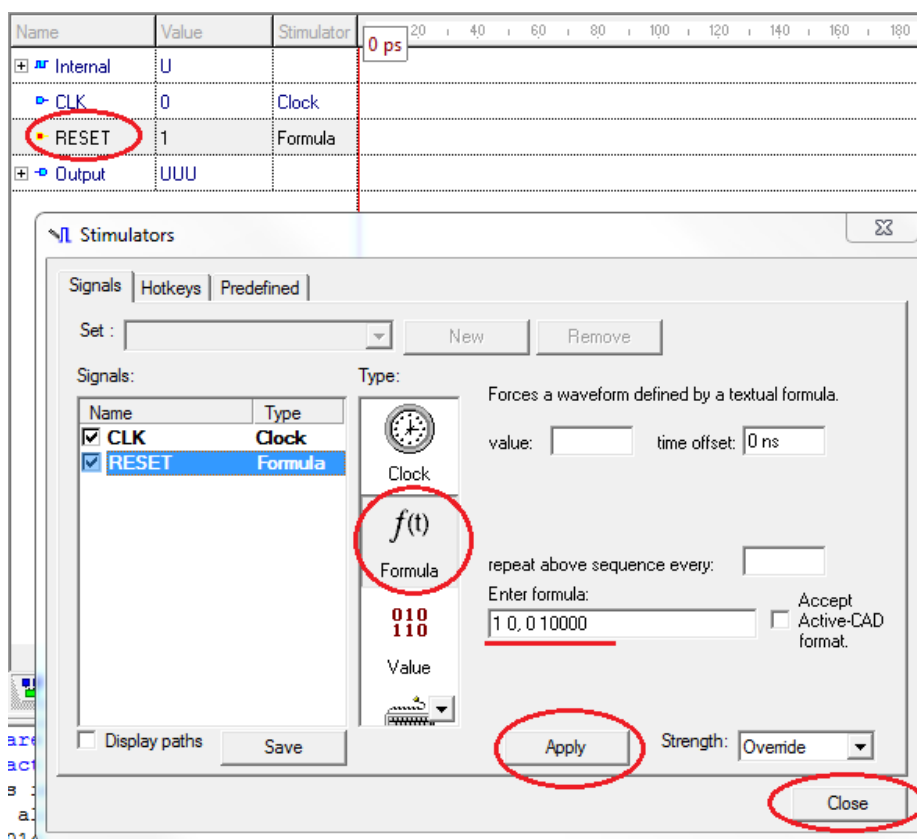
Rys. 33. Otwieranie okna wymuszeń (Stimulators).

- j) Wybierz typ wymuszenia „Clock”, ustaw częstotliwość przebiegu na 10 MHz i kliknij „Apply”. (nie zamykaj jeszcze okna Stimulators).



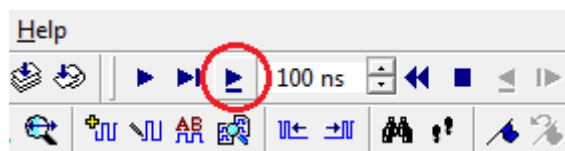
Rys. 34. Przypisywanie wymuszeń do sygnałów.

- k) Zaznacz sygnał „RESET” i przypisz mu wymuszenie w postaci „Formuła”.  
 Wklej formułę:  
 1 0, 0 10000  
 i kliknij Apply a następnie Close.



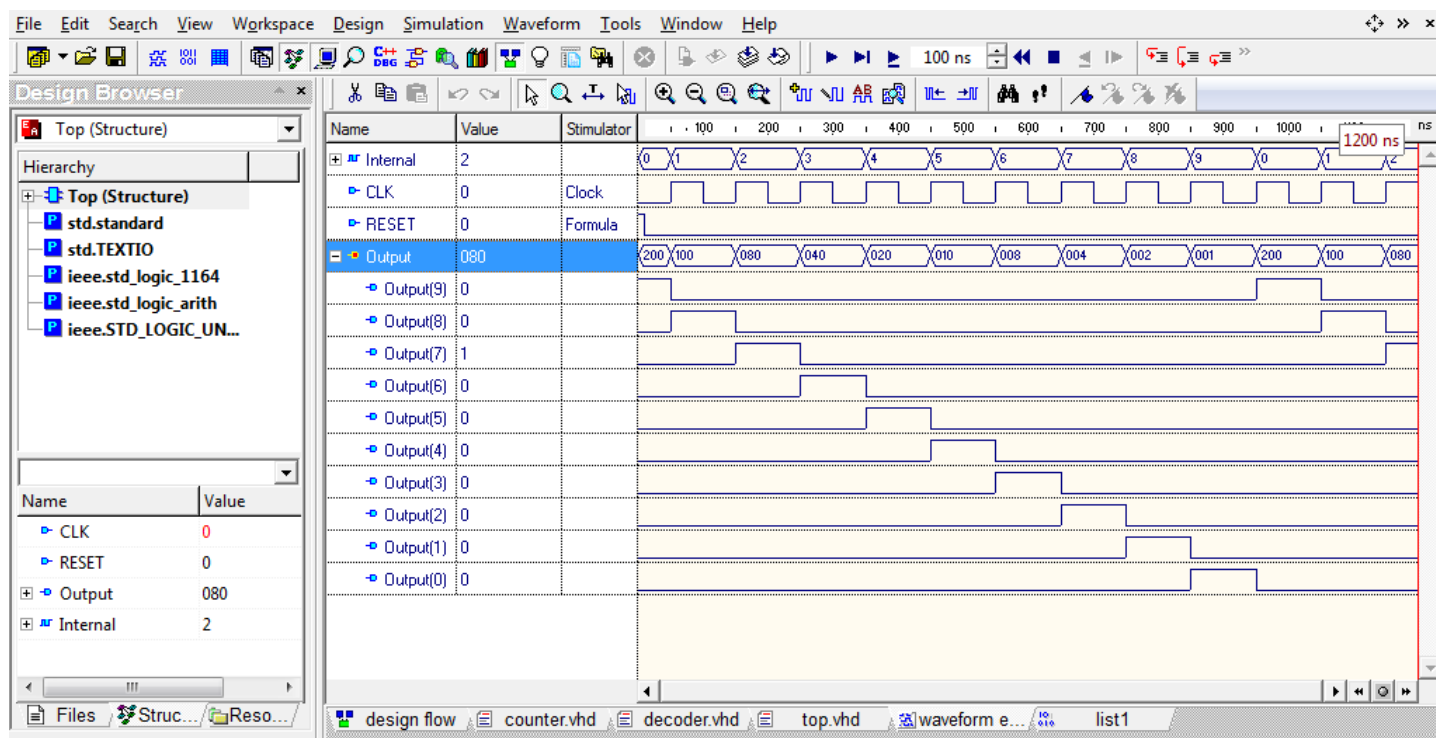
Rys. 35. Przypisanie wymuszeń do sygnałów – cd.

- l) Wykonaj kilkanaście kroków symulacji klikając każdorazowo ikonę skrótów „Run For”.  
**Nigdy nie uruchamiaj symulacji klikając skrót „Run” (nie mamy zdefiniowanego momentu zakończenia symulacji; symulacja będzie się wykonywać „w nieskończoność” i zawiesi komputer).**



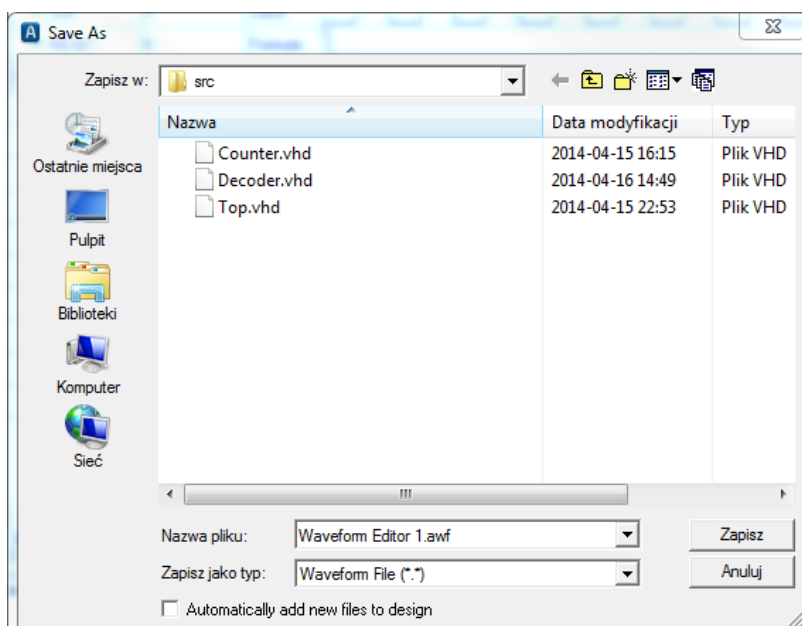
Rys. 36. Przycisk skrótów „Run For”.

Można obejrzeć stan logiczny poszczególnych sygnałów magistrali klikając w „plusik” obok jej nazwy. Sprawdź czy otrzymano identyczny efekt jak na rysunku poniżej.



Rys. 37. Okno symulacji.


- m) Zakończ symulację wybierając z menu programu Simulation -> End Simulation.  
 n) Zapisz waveform wybierając z menu programu File -> Save.  
 Pozostaw domyślną nazwę pliku „Waveform Editor 1.awf”.

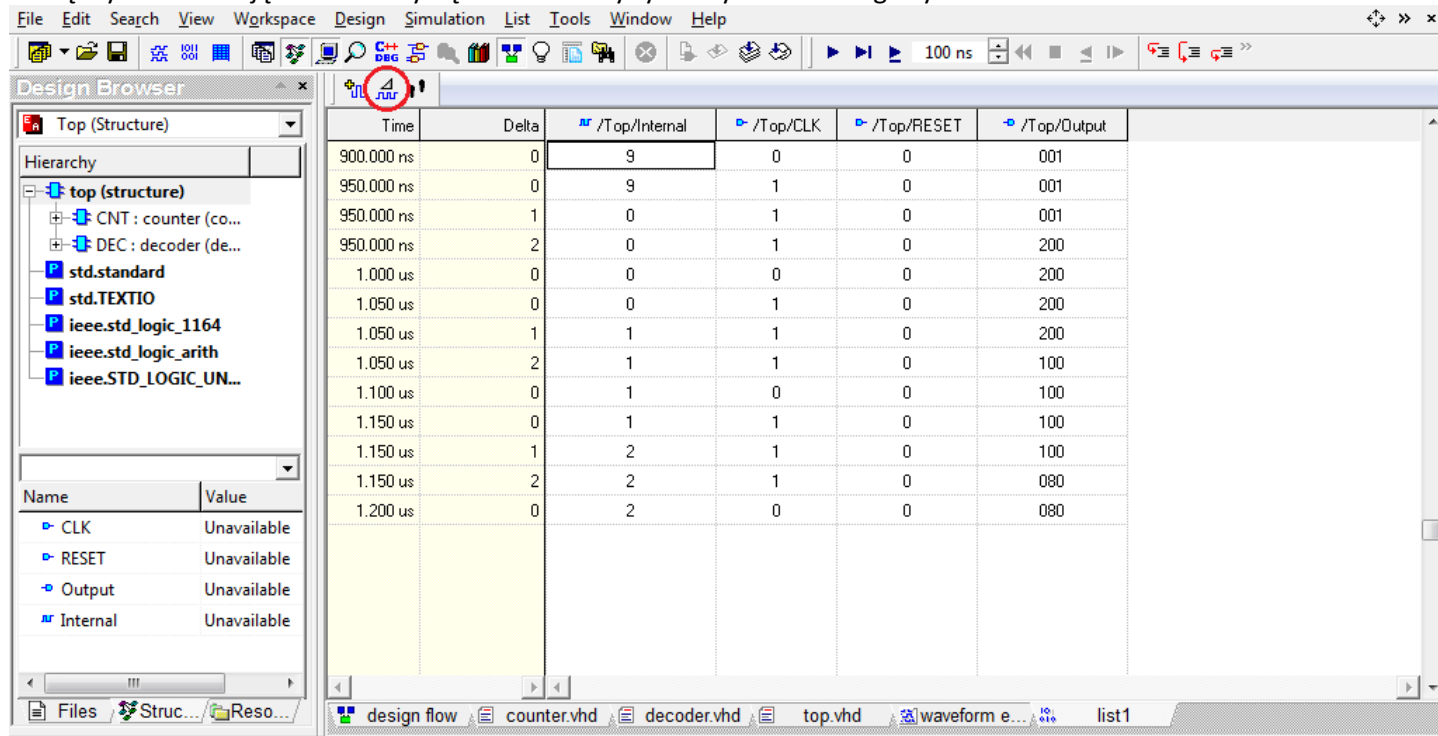


Rys. 38. Zapis wyników symulacji.

o) Przejrzyj wyniki uzyskane w Standard List Viewer.

W tym celu przełącz zakładkę z „waveform” na „list1”.

Tabela podaje moment czasu w którym jakiś sygnał uległ zmianie oraz numer cyklu Delta symulacji. Można ograniczyć liczbę wyników klikając . Wtedy będzie widoczny tylko wynik ostatniego cyklu delta.



Time	Delta	/Top/Internal	/Top/CLK	/Top/RESET	/Top/Output
900.000 ns	0	9	0	0	001
950.000 ns	0	9	1	0	001
950.000 ns	1	0	1	0	001
950.000 ns	2	0	1	0	200
1.000 us	0	0	0	0	200
1.050 us	0	0	1	0	200
1.050 us	1	1	1	0	200
1.050 us	2	1	1	0	100
1.100 us	0	1	0	0	100
1.150 us	0	1	1	0	100
1.150 us	1	2	1	0	100
1.150 us	2	2	1	0	080
1.200 us	0	2	0	0	080

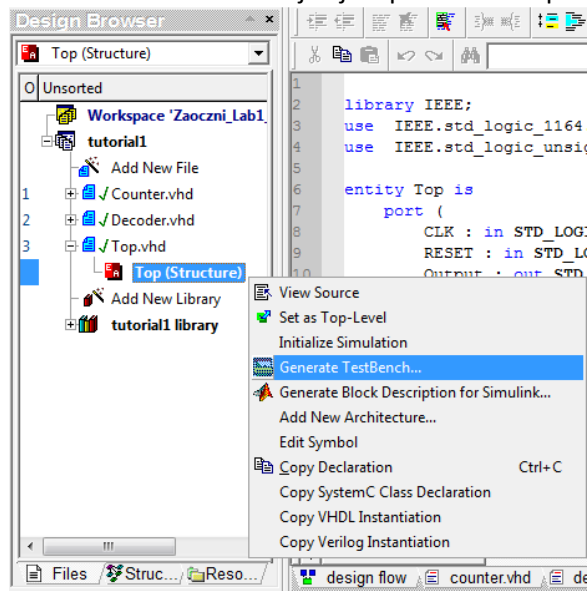
Rys. 39. Wynik symulacji przedstawiony w formie tabeli.

## 20. Generowanie Testbench.

Testbench to kolejny plik vhdl. Instrukcje w nim umieszczone mają na celu przetestowanie prawidłowego działania zaprojektowanego komponentu. Ponieważ Testbench służy wyłącznie do testów (i nie będzie syntezowany), to możemy w nim korzystać ze wszystkich dobrodziejstw języka VHDL (zmiana sygnału z opóźnieniem, dostęp do plików, itd.). Testbench ma na celu ułatwienie symulacji, pozwala na automatyczną realizację wszystkich podpunktów z punktu 19. Jest ponadto wygodniejszy, gdyż w programie mamy dostępne tylko proste wymuszenia, a w testbenchu możemy w łatwy sposób zdefiniować sekwencję zmian sygnałów wejściowych.

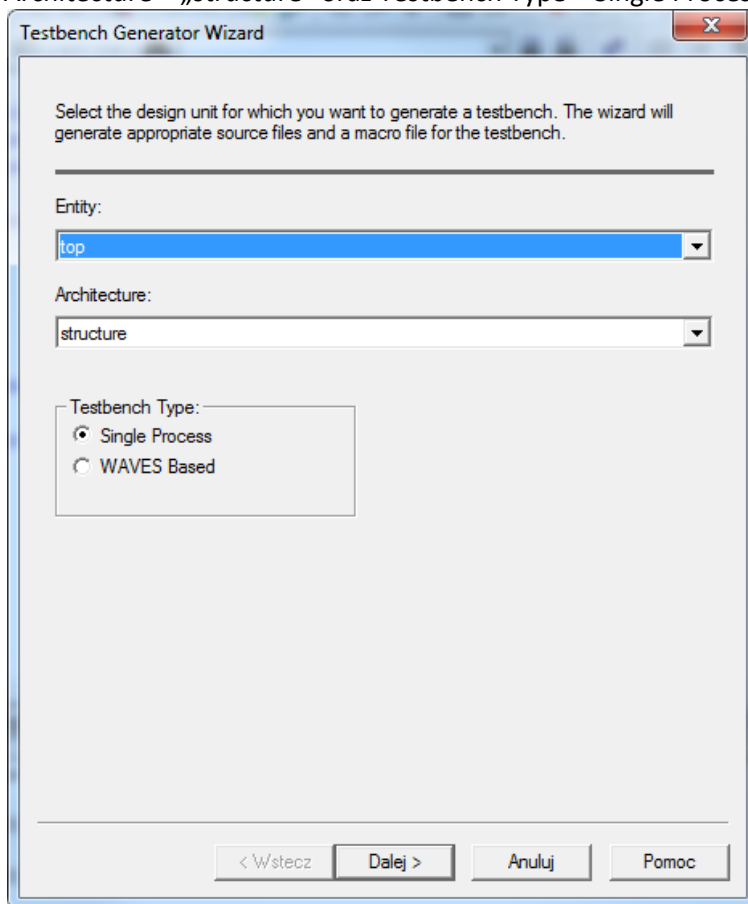
W przypadku testowania skomplikowanego komponentu, testbench jest powiększany o kolejne instrukcje sprawdzające działanie poszczególnych funkcjonalności. Na każdym etapie testowania, wystarczy go uruchomić by wykonać wszystkie dotychczasowe testy.

- a) W Design Browser zmien zakładkę na Files. Rozwiń pozycję Top.vhd. Kliknij prawym przyciskiem myszy parę „Top (Structure)”. Wybierz „Generate TestBench...” tak jak jest pokazane na poniższym rysunku.



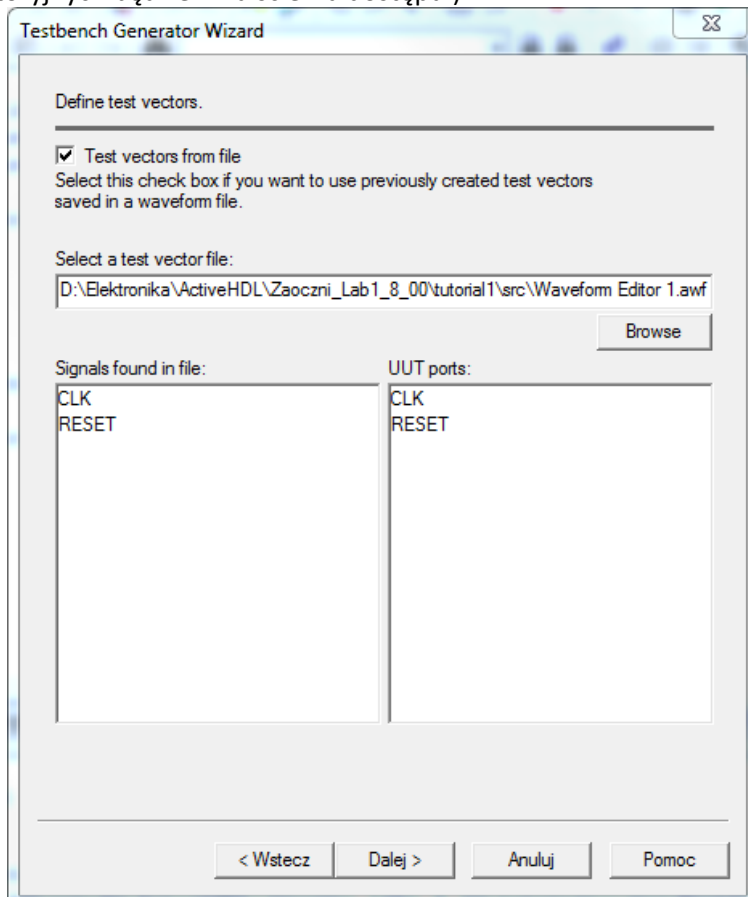
Rys. 40. Uruchomienie kreatora do tworzenia TestBench.

- b) W oknie, które się pojawi, jako pierwsze trzeba wskazać entity i architecture które chcemy symulować. Wybierz Entity – „top”, Architecture – „structure” oraz Testbench Type – Single Process. Kliknij „Dalej”.



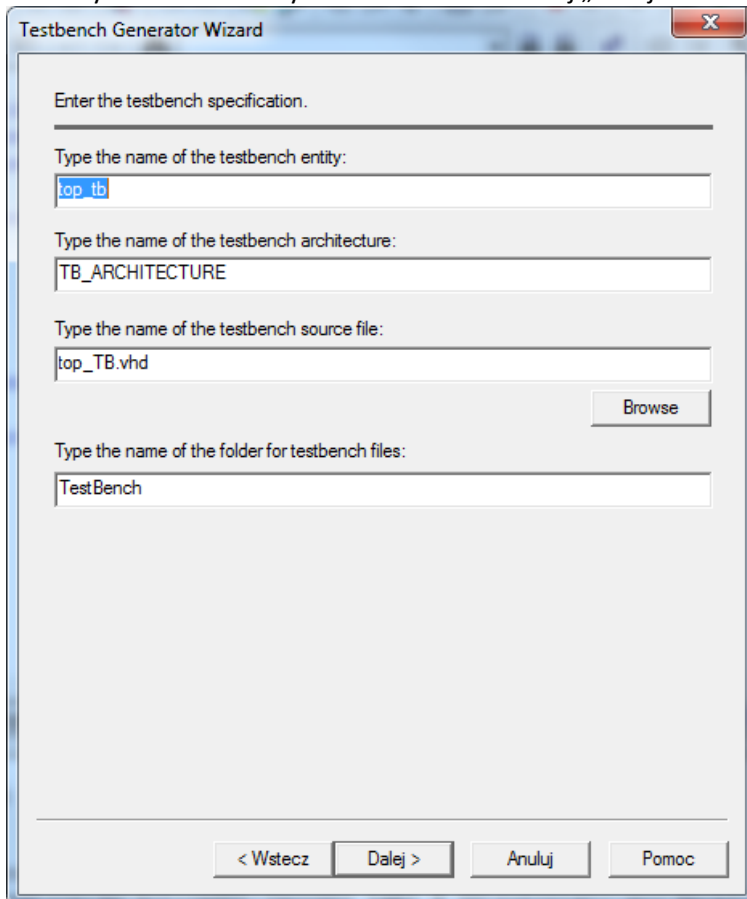
Rys. 41. Wybór testowanej entity i architecture.

- c) Do generacji wektorów testowych wykorzystamy plik „Waveform Editor 1.awf” stworzony w punkcie 19. n). Zaznacz pole „Test vectors from file”. Kliknij „Browse” i wskaż na dysku swój plik „Waveform Editor 1.awf”. (W trakcie zajęć laboratoryjnych będzie inna ścieżka dostępu.)



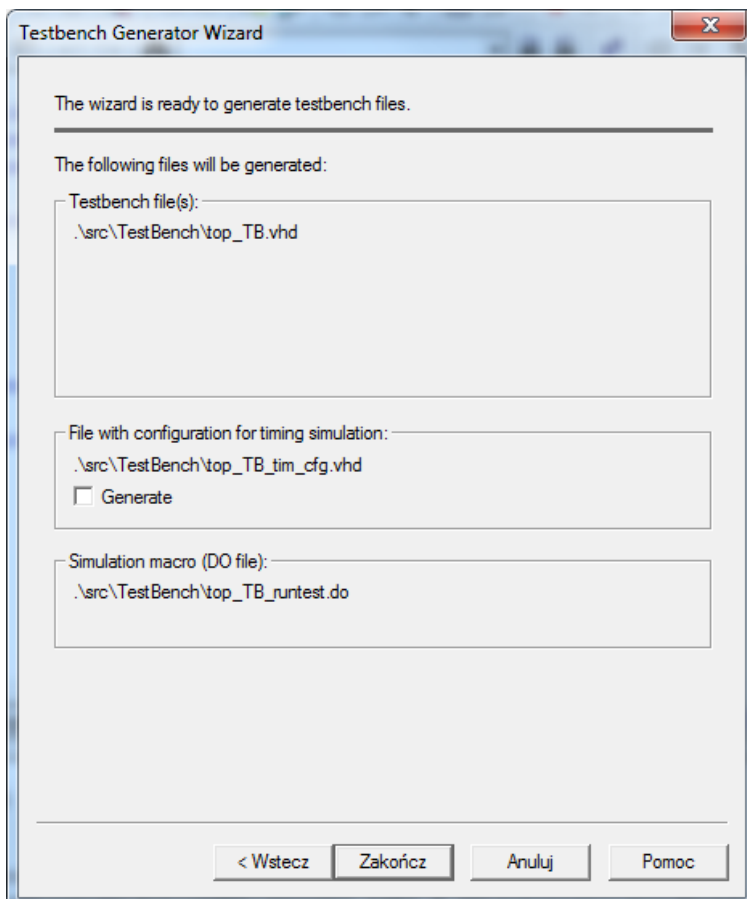
Rys. 42. Import wektorów testowych z zapisanej symulacji.

- d) Kolejne okno umożliwia nadanie nazwy dla entity i architecture generowanego testbenchu a także nazwę i położenie pliku na dysku twardym. Zostaw domyślne ustawienia i kliknij „Dalej”.



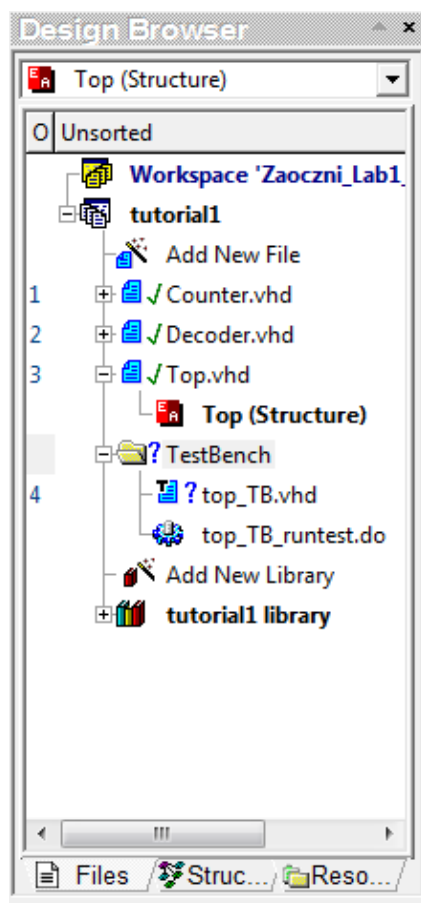
Rys. 43. Nazwy entity i architecture testbenchu oraz nazwa pliku i jego folder.

- e) W ostatnim oknie można włączyć generowanie konfiguracji dla symulacji czasowych. Domyślnie ta opcja jest wyłączona i tak ma pozostać. Zakończ generowanie testbenchu klikając „Zakończ”.



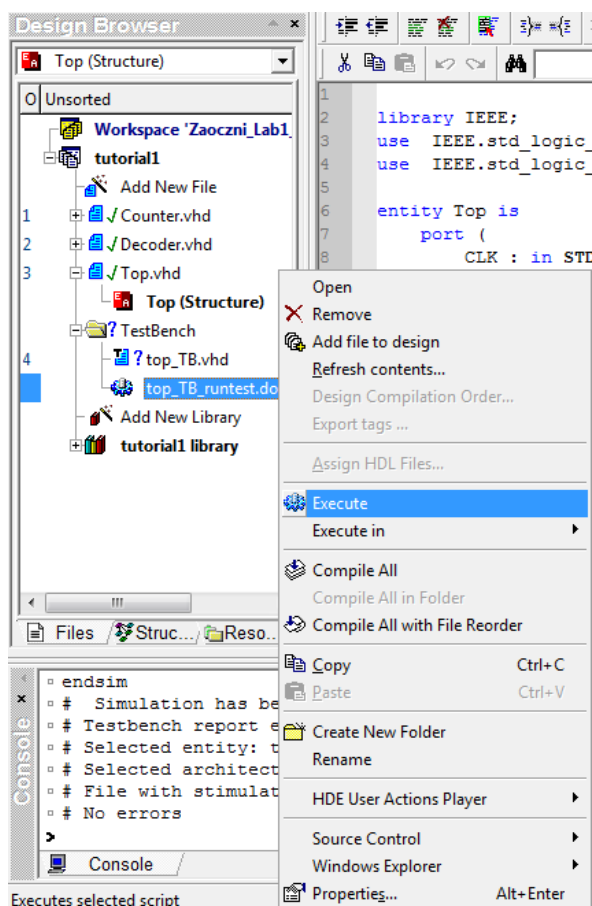
Rys. 44. Generowanie testbench – cd.

W Design Browser zostanie ułożony folder zawierający plik vhd testbencha oraz makro służące do jego uruchomienia. Uruchomienie makra powoduje kompilację testbencha oraz uruchomienie symulacji.



Rys. 45. Folder zawierający kod TestBencha oraz makro do jego uruchomienia.

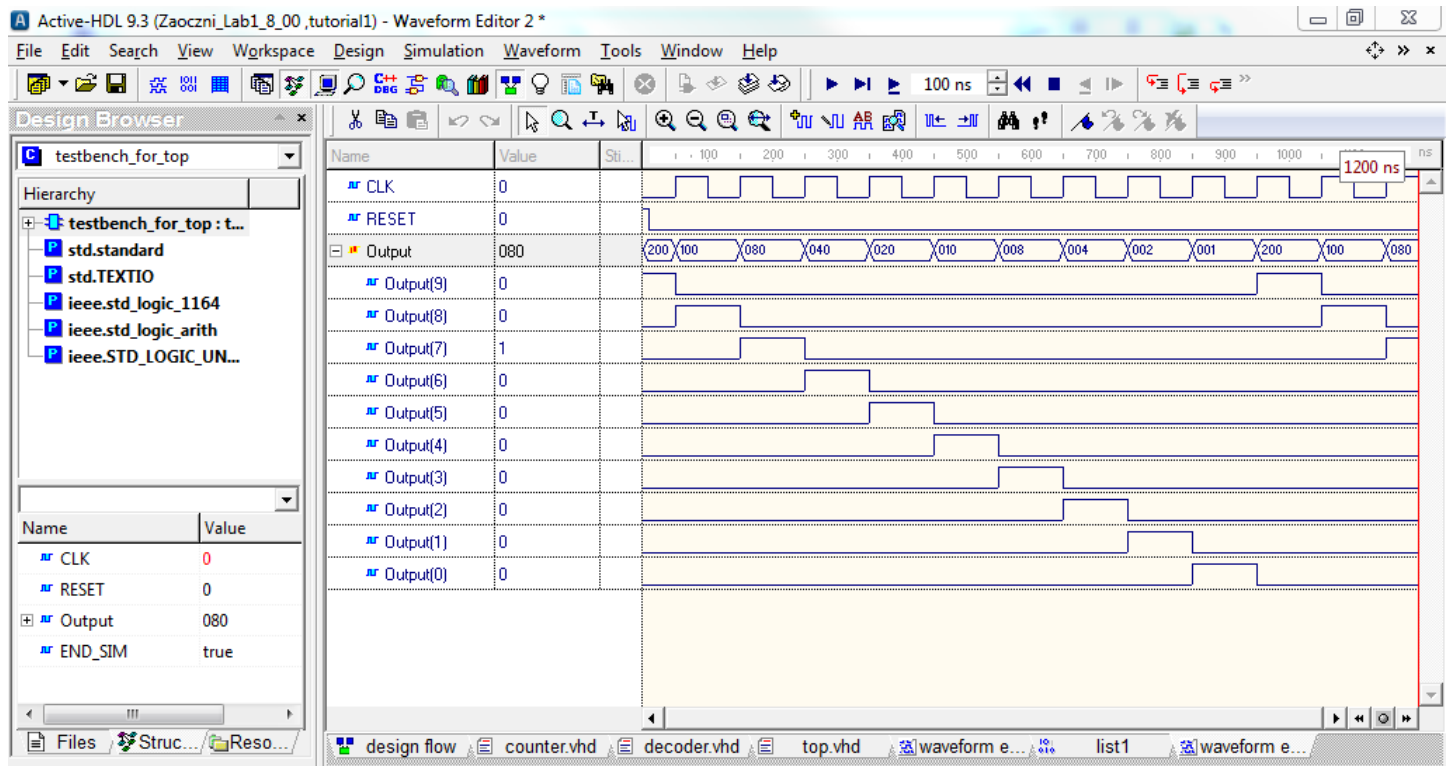
f) Uruchom symulację klikając prawym przyciskiem myszy makro i wybierając z menu podręcznego „Execute”.



Rys. 46. Uruchomienie TestBench.



W prawej części okna programu pojawi się nowa zakładka z przebiegami (niezapisany Waveform Editor 2).



Rys. 47. Wynik uruchomienia TestBench.

## 21. Debugowanie kodu vhdl.

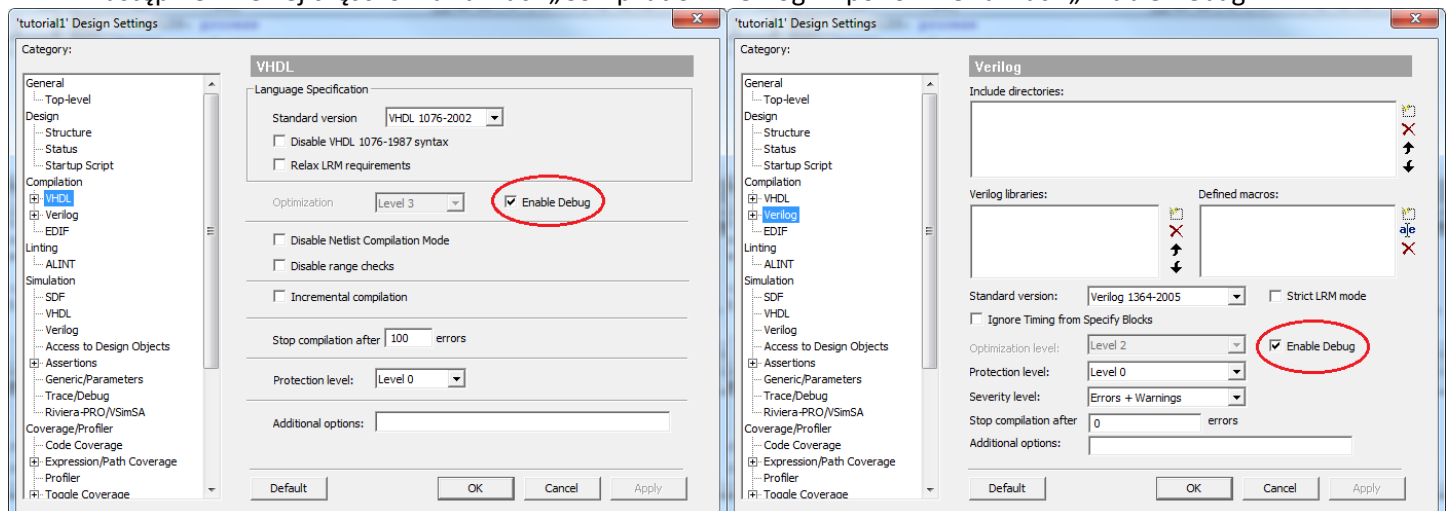
Active-HDL umożliwia krokowe wykonywanie kodu źródłowego w trakcie symulacji. Aktualnie wykonywany kod jest automatycznie otwierany i po użyciu przycisków „Trace Into”, „Trace Over” lub „Trace Out”. Wykonywana instrukcja jest podświetlona w oknie edytora na kolor żółty.

Wprowadź wewnątrz procesu instrukcje są wykonywane jedna po drugiej, to już instrukcje wewnątrz architektury są współbieżne. Ponadto uruchomienie procesu jest powodowane zmianą stanu na sygnale podanym w liście wrażliwości. Ostatecznie debugowanie kodu jest dosyć męczące – na pierwszy rzut oka symulator przeskakuje pomiędzy różnymi plikami vhd wykonując procesy i inne instrukcje w „niezrozumiałej” kolejności.

a) Z menu głównego wybierz Design -> Settings...

W lewej części okna zaznacz „Compilation: VHDL”. W prawej części okna zaznacz „Enable Debug”.

A następnie w lewej części okna zaznacz „Compilation: Verilog”. I ponownie zaznacz „Enable Debug”.

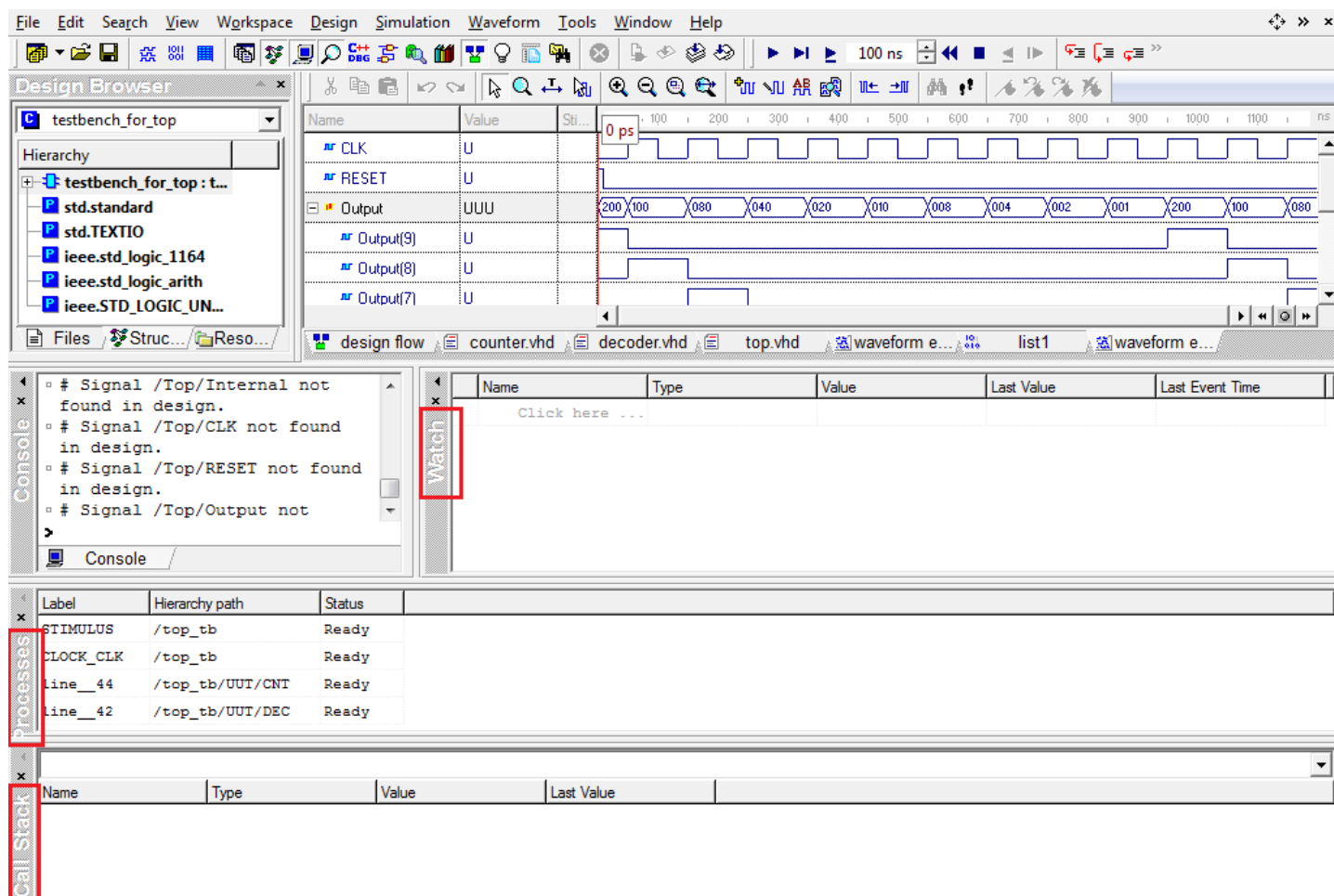


Rys. 48. Włączenie debugowania w ustawieniach projektu.

b) Wyłącz poprzednią symulację (uruchomioną przez wykonanie makra TestBench) wybierając z menu głównego Simulation -> End Simulation.

c) W Design Browser ustaw top\_tb (TB\_ARCHITECTURE) jako Top-Level (podobnie jak w pkt 19. a) ).

- d) Zainicjalizuj ponownie symulację wybierając z menu programu Simulation -> Initialize Simulation.  
 e) Wybierz z menu View -> Watch, Processes oraz Call Stack.  
 Na ekranie powinny pojawić się okna Watch, Processes oraz Call Stack.



Rys. 49. Active-HDL z otwartymi oknami Watch, Processes oraz Call Stack.

Okno Processes wyświetla stan wszystkich procesów w projekcie.

W oknie Watch można oglądać wartość sygnałów i zmiennych.

Okno Call Stack wyświetla listę podprogramów (funkcji i procedur) występujących w aktualnie wykonywanym procesie. Przez proces jest tu rozumiana dowolna instrukcja współbieżna modelująca sekwencyjny proces (instrukcja proces, współbieżne przypisanie wartości sygnałom, współbieżne użycie instrukcji assert, współbieżne wywołanie procedury).

Dla każdego podprogramu okno Call Stack wyświetla:

- Parametry formalne wraz z ich aktualną wartością;
- Zmienne, stałe i pliki zadeklarowane lokalnie w ciele podprogramu wraz z ich aktualną wartością.

W trakcie debugowania można korzystać z komend:

Trace into – wykonuje pojedynczą instrukcję vhdl. Jeżeli tą instrukcją jest wywołanie podprogramu, wykonywanie instrukcji otwiera ciało podprogramu;

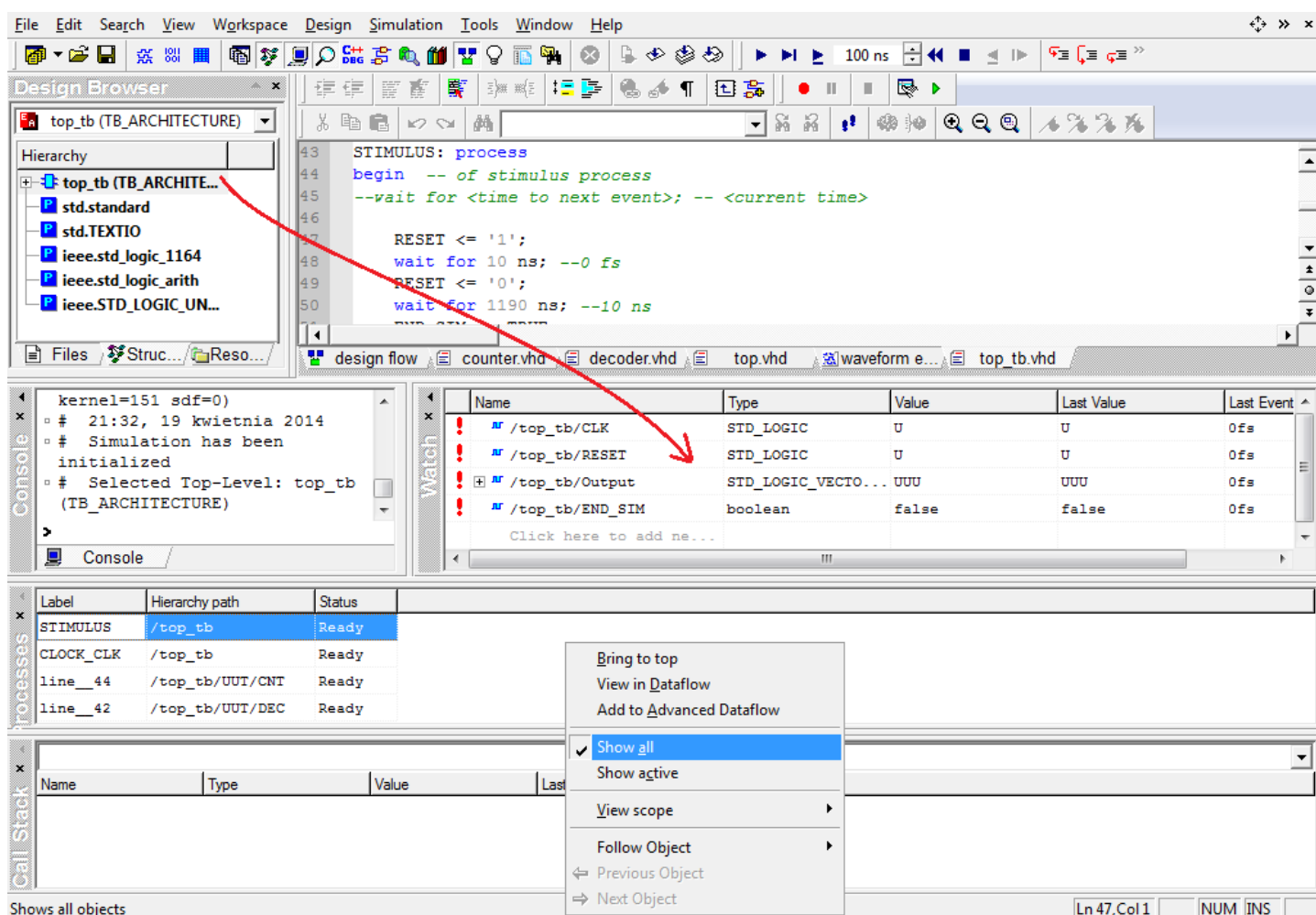
Trace over – wykonuje pojedynczą instrukcję vhdl. Jeżeli tą instrukcją jest wywołanie podprogramu, wszystkie instrukcje ciała podprogramu wykonywane są w jednym kroku.

Trace out – wykonują tyle instrukcji ile jest potrzebnych by zakończyć wykonywanie podprogramu. Jeśli podprogramy są zagnieżdżone, to komenda trace out kończy wykonywanie tylko „najgłębszego” podprogramu.

Używając tych komend można oglądać:

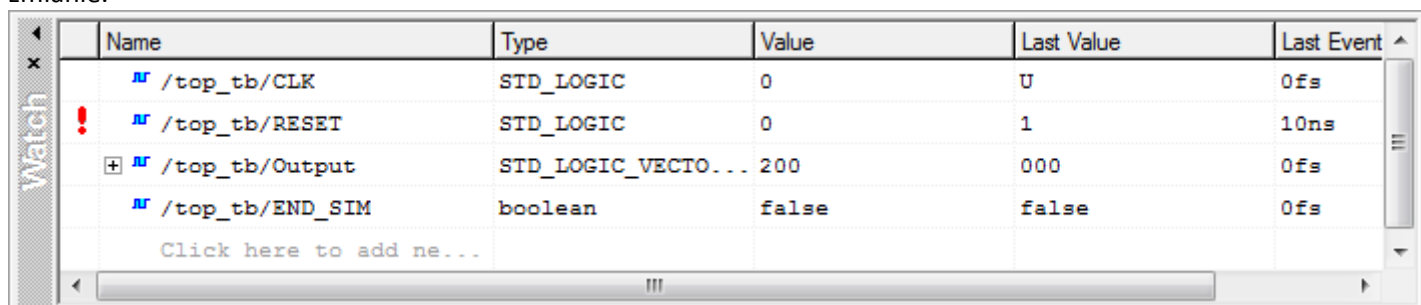
- zmiany wartości sygnałów w oknie Watch;
- aktywność poszczególnych procesów w oknie Processes;
- zmienne, stałe i pliki zadeklarowane lokalnie w ciele podprogramów razem z ich wartościami w oknie Call Stack.

- f) Upewnij się że w Design Browser wybrana jest zakładka Structure. Używając metody „przeciągnij i upuść”, zaznacz top\_tb (TB\_Architecture) w Design Browser i przenieś je do okna Watch.
- g) Kliknij prawym przyciskiem myszy gdziekolwiek w oknie Processes i wybierz z menu podręcznego „Show all”. Dzięki temu, będziemy mogli również obserwować procesy nieaktywne.



Rys. 50. Ustawienia Debuggera.

- h) Wykonaj kilkanaście kroków „Trace Over” (i spróbuj zrozumieć co się dzieje ;D). Jeżeli po pierwszym kliknięciu „Trace Over” otrzymujesz komunikat o zakończeniu symulacji, to wybierz z głównego menu programu Workspace -> Compile Workspace. Następnie Simulation -> Restart Simulation i spróbuj ponownie. Wykrzyknik obok sygnału w oknie Watch informuje że w danym cyklu symulacji / debuggowania sygnał ten uległ zmianie.



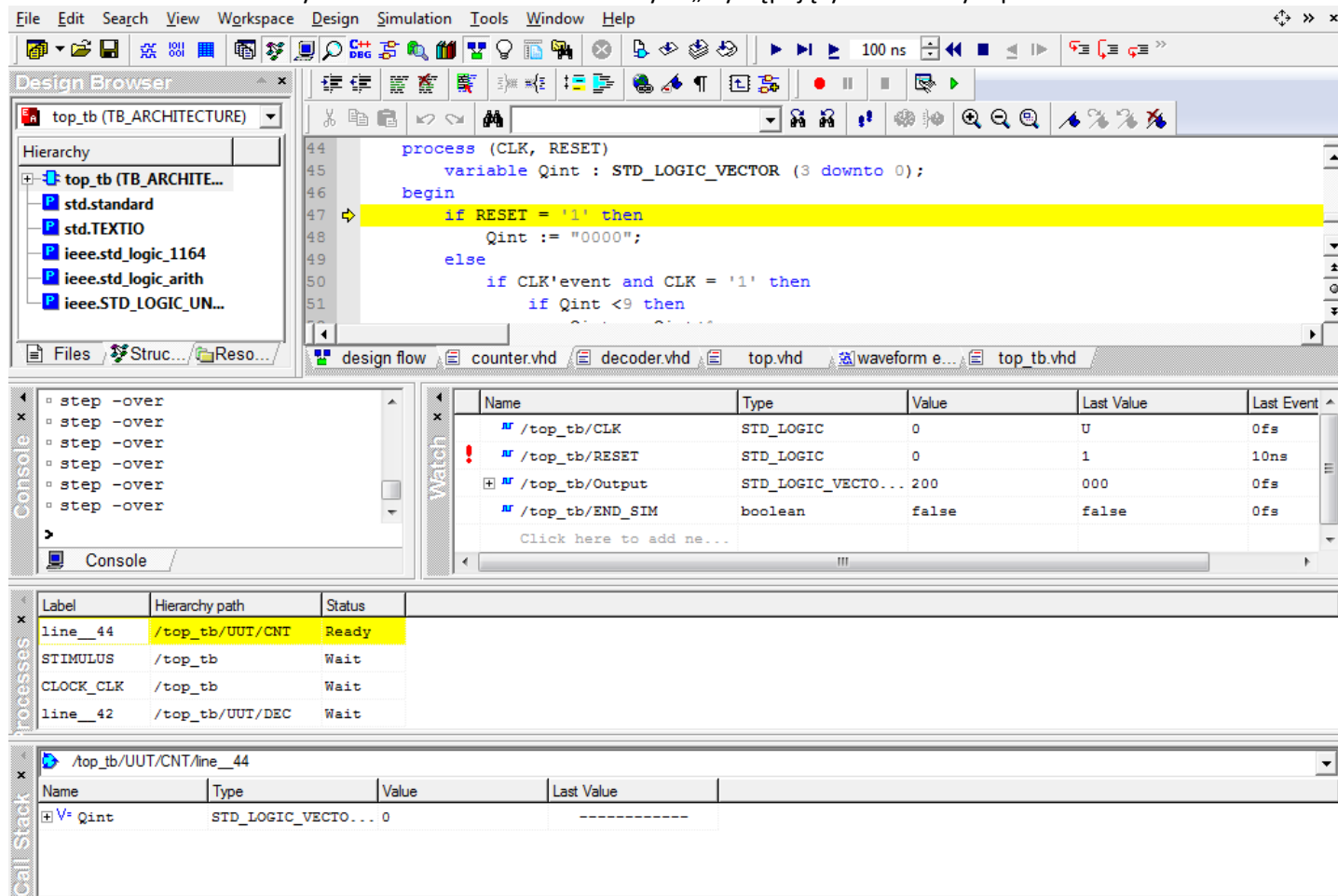
Rys. 51. Zmiana sygnału RESET.

W oknie Process można sprawdzić stan danego proces (ready lub wait). Ready oznacza że proces jest aktualnie aktywny.



Rys. 52. Podświetlanie aktywnego procesu.

W oknie Call Stack możemy obserwować wartości zmiennych „występujących” na danym poziomie hierarchii kodu vhdl.



Rys. 53. Analiza kodu.

## 22. Analiza projektu z wykorzystaniem okna Dataflow.

Okno Dataflow jest narzędziem pozwalającym na graficzną reprezentację sygnałów wpływających i wpływających do procesów w trakcie symulacji. Przez proces jest tu rozumiana dowolna instrukcja współbieżna modelująca sekwencyjny proces (instrukcja proces, współbieżne przypisanie wartości sygnałowi, współbieżne użycie instrukcji assert, współbieżne wywołanie procedury).

Okno posiada dwa różne widoki:

- w środku okna jest proces;
- w środku okna jest sygnał.

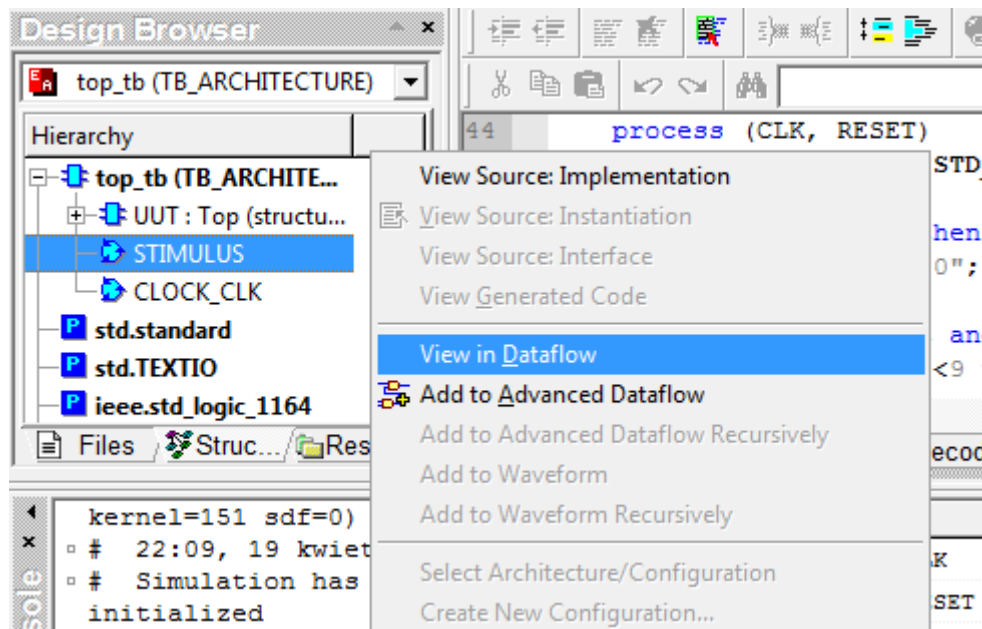
Jeżeli w środku okna jest proces, to jest on reprezentowany przez prostokąt z sygnałami wejściowymi po lewej stronie i sygnałami wyjściowymi po jego prawej stronie. Sygnały wejściowe dostarczają informacji do procesu. Sygnały wyjściowe wyprowadzają dane z procesu.

Jeżeli w środku okna jest wyświetlony sygnał, to jest on reprezentowany przez grubą poziomą linię. Po jej lewej i prawej stronie znajdują się procesy. Proces wyświetlony po lewej stronie nadaje wartość sygnałowi. Proces po prawej stronie czyta wartość sygnału.

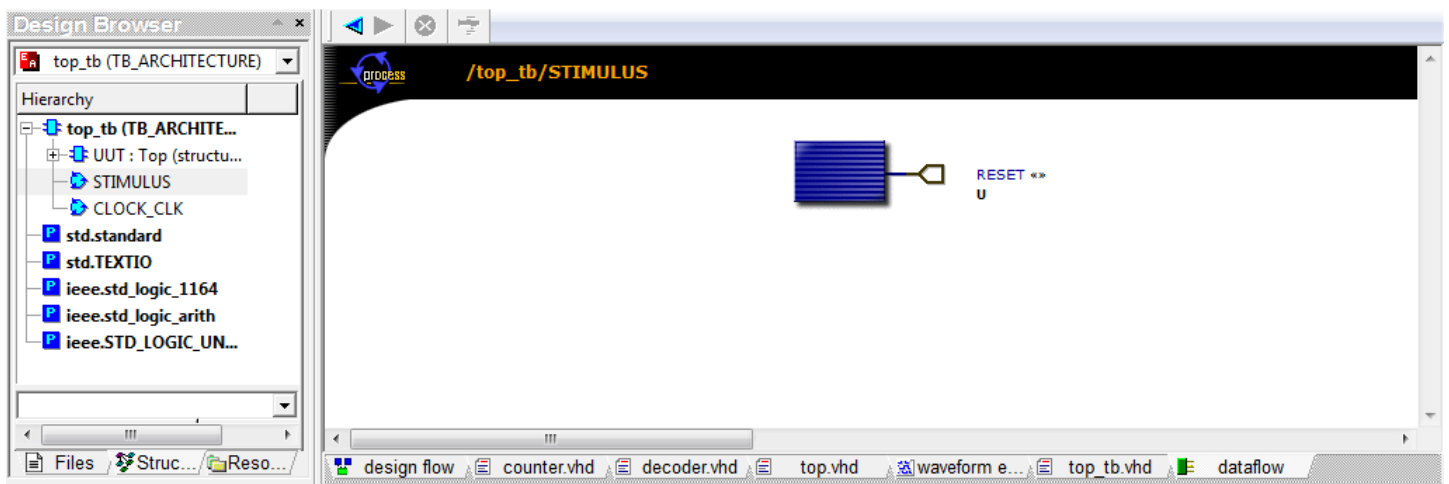
Można przełączać się pomiędzy tymi widokami poprzez kliknięcie elementów w oknie Dataflow.

W obu widokach sygnały są wyświetlane razem z ich nazwą i obecną wartością. Podobnie, w obu widokach procesy są wyświetlane razem z ich etykietą (jeżeli nie została podana jawnie przez użytkownika, to wyświetlana jest nazwa automatycznie wygenerowana przez symulator).

- Zrestartuj symulację wybierając z menu głównego Simulation -> Restart Simulation.
- W Design Browser (zakładka Structure) kliknij prawym przyciskiem myszy proces „STIMULUS” i wybierz „View in Dataflow”. (zrzut poniżej)

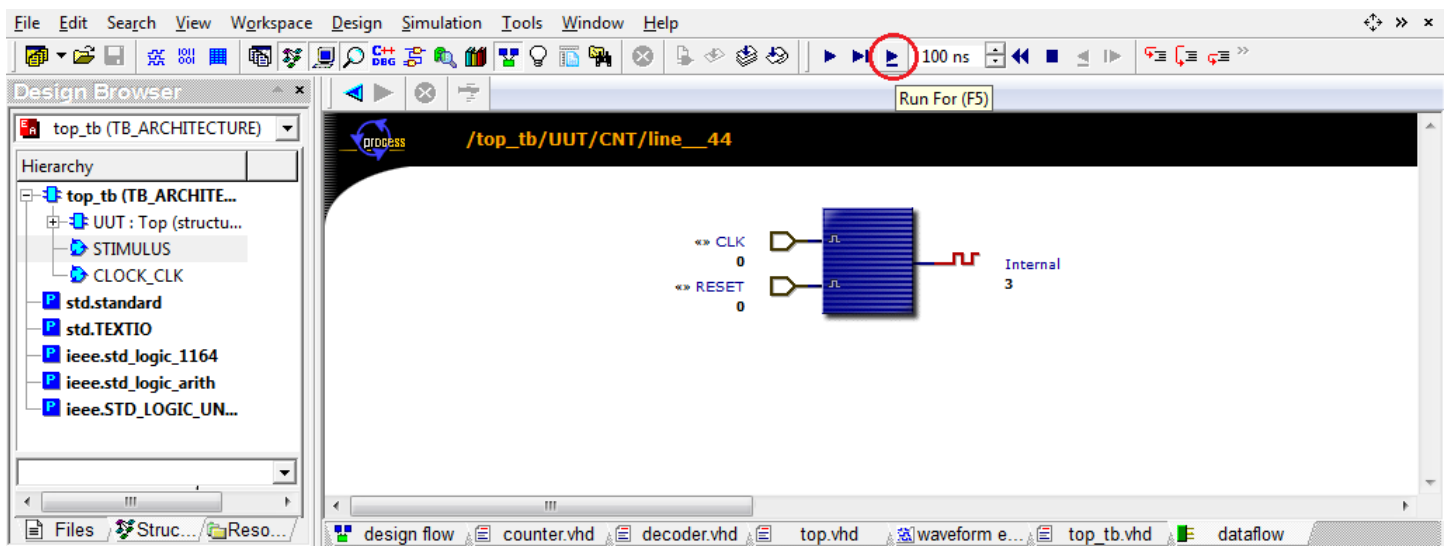


Rys. 54. Otwarcie okna Dataflow.



Rys. 55. Okno Dataflow.

- c) W oknie Dataflow kliknij na nazwie sygnału „RESET”. W tym momencie Dataflow przełączy się na widok z sygnałem w środku.
- d) Kliknij na nazwie procesu „/top\_tb/UUT/CNT/line\_\_44”. Jest to główny proces licznika modulo 10.
- e) Obserwuj zmiany wartości sygnału „Internal” (zdefiniowany w Top.vhd i dołączony do wyjścia Q licznika) powodowane kliknięciem ikony skrótu „Run For” na pasku programu (lub użyj F5 na klawiaturze).



Rys. 56. Zmiany sygnału Internal po każdym kroku symulacji.