1. Uruchomić "Aldec Actve-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".

License Configuration		? X
	Select one of the Active-HDL product configurations and click [Next>]. EDU Mixed Design Entry	3
	Reserve simulation features at startup	
	For more details click license information.	
	<back next=""></back>	Cancel

Rys. 2. Okno konfiguracji licencji.

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

Download Active-HDL Updates
Please, choose how frequently Active-HDL should check for available updates on the Aldec website. To disable automatic checks, choose 'never'.
Check for updates: weekly
If updates are not enabled you may miss some important software and library upgrades.
Press OK to check for updates on Aldec website now.
OK Cancel

Rys. 3. Okno aktualizacji programu.

4. W oknie wybrać "Create new workspace" i kliknąć "OK".

Getting	Started		? ×
	C Open existing workspace		
	kalicki_zaj3 kalicki_2zaj timespec tutorial_zaj1 uart kalicki 1zaj	*	More
0	c:\my_designs\kalicki_zaj3		
	ays open last workspace		
		OK	Cancel

Rys. 4. Wybór przestrzeni roboczej.

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

New Workspace		×
	Specify basic information about the new workspace.	
	Type the workspace name:	
Centra	Zaoczni_Lab1_8_00	
A Contraction	Select the location of the workspace folder:	
A COLOR	c:\my_designs\	
	Browse ▼ Add New Design to Workspace OK C	Cancel

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".

New Desig	n Wizard
Specify	additional information about the new design.
	C-Synthesis tool: <none></none>
	Synthesis tool: Xilinx ISE/WebPack 14.4 XST VHDL/Verilog
	Physical Synthesis tool: <none></none>
	Implementation tool: Xilinx ISE/WebPack 14.4
	Default Family: Xilinx14x SPARTAN3 Flow Settings
	Block Diagram Configuration: Default HDL Language 🗨
	Default HDL Language: VHDL
	< Wstecz Dalej > Anuluj

Rys. 6. Ustawienia syntezy i implementacji.

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

nective basic information about the pe	w design		
Type the design name:	w ucaight.		
tutorial 1			
, Select the location of the design folder			
c:\My_Designs\Zaoczni_Lab1_8_00			
			Browse
The name of the default working library	of the design:		
tutorial 1			
The name specified here will be used a ibrary files and as the logical name of the change the logical name later on.	is the file name fi he library. You ca	or the an	

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.

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



Rys. 9. Pierwsze okno kreatora plików vhdl.

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

New Source File Wizard -	Name	23
	Type the name of the source file to create:	
	Counter E	Browse
	You can use the Browse button to specify the	file.
	Type the name of the entity (optional):	
	Counter	
	By default, the entity name is the same as the f	ile name.
	Type the name of the architecture body (option	nal):
	Counter	
	By default, the architecture name is the same a entity name.	as the
	< Wstecz Dalej >	Anuluj

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	Туре
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".

New Source File Wizard -	Ports
CLK Q[3:0] RESET Counter	To add a new port, click New. To edit a port, select it on the list. Then you can change its name, direction and type. To quickly change the index constraint of a port of a one-dimensional array type, use the Array Indexes box. To remove a port, select it on the list, and then click Delete. CLK O[5:0] RESET Port direction C in C inout © out C buffer
	New Delete Type           Vew         Vew           < Wstecz         Zakończ

Rys. 11. Konfiguracja portów entity.

 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 przeglądanego 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 vhdl.

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 skrótu: Pojawi się nowe okno:

P Language Assistant							23
VHDL	- 🗅 徾 🚔 🖬	$\times$ D <sub>k</sub>	R	\$ «	×		
Templates Code Auto Complete Language templates MATLAB Interface Simulation templates Synthesis templates Training Tutorial User templates Utility templates							▲ ▼
						NUM	INS //

Rys. 15. Okno Language Assistant.

- b) Rozwiń pole "Tutorial".
- c) Zaznacz element "Counter". W prawej części okna pojawi się kod vhdl 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.



 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:

- a) Wybierz z menu File -> New -> VHDL Source.
- b) Jako nazwę pliku podaj "Decoder". (Domyślnie nazwa entity jest taka sama jak nazwa pliku. Domyślnie nazwa architecture jest taka sama jak nazwa entity).

c	) Dodai komponentowi portv	/ svgnały tak jak w noniższej tabelce
U)		/ Sygnaly lak jak w pullizszej labelle.

Name	Array Indexes	Port direction	Туре
Dataln <sup>*</sup>	3 - 0	In	STD_LOGIC_VECTOR
Output	9 - 0	Out	STD_LOGIC_VECTOR

<sup>\*</sup> I jak Irena; to nie jest małe I jak Leszek ;D



Rys. 18. Porty komponentu Decoder.

d) W rezultacie powinniśmy otrzymać efekt przedstawiony na rysunku poniżej.

Active-HDL 9.3 (Zaoczni_Lab1_8_00 ,	utorial1) - d:\Elektronika\ActiveHDL\Zaoczni_Lab1_8_00\tutorial1\src\Decoder.vhd	
<u>File Edit Search View Workspace</u>	<u>D</u> esign <u>S</u> imulation <u>T</u> ools <u>W</u> indow <u>H</u> elp	÷, ≫ ×
🗗 🕶 🖬 🐹 🐰 🔳 🗗 💱	🔵 🔎 🚟 🌫 🛝 🏙 🚏 🖓 ቬ 🙀 🕺 😓 🗇 🆃 🏷   🕨 👌 100 ns 🗄 🕂 🗉 🔺 📭 🦕 📭 📮	
Design Browser 💦 🔹 🗙	] 律律 業盤 戰 評略 福厚 後 ▲ ¶ 西部   ● Ⅱ Ⅱ ▶ ▶	
Counter (Counter)	j X 🗈 🖻 ∽ ∽ 👭 🔽 🔽 🖌 🖌 🖌 🖌	
O Unsorted Workspace 'Zaoczni_Lab1 Display tutorial1	<pre>24 25 library IEEE; 26 use IEEE.STD_LOGIC_1164.all; 27</pre>	•
Add New File	28 entity Decoder is 29 port( 30 DataIn : in STD LOGIC VECTOR(3 downto 0);	
2 - S f Decoder.vhd - S Add New Library	31 Output : out STD_LOGIC_VECTOR(9 downto 0) 32 );	
⊞ <b>un</b> tutorial1 library	<pre>33 end Decoder; 34 35}} End of automatically maintained section 36</pre>	
	37 architecture Decoder of Decoder is 38 begin 39	
	40 enter your statements here 41 42 end Decoder:	1 0 7
Files / Struc/	design flow A Counter.vhd A decoder.vhd	•

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 następnej 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(
               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
                  Output <= "100000000" when "0000",
43
44
                          "0100000000" when "0001",
                          "0010000000" when "0010",
45
                          "0001000000" when "0011",
46
47
                          "0000100000" when "0100",
                          "0000010000" when "0101",
48
                          "0000001000" when "0110",
49
                          "0000000100" when "0111",
51
                          "0000000010" when "1000",
52
                          "0000000001" when "1001",
                          "0000000000" when others;
53
54
     end Decoder;
📱 design flow 💒 counter.vhd 💒 decoder.vhd *
```

Rys. 20. Poprawna zawartość pliku Decoder.vhd.

- 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 dekoder a następnie je ze sobą połączymy. Tym razem nie będziemy korzystać z kreatora tworzenia nowych plików vhd 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.

Design Browser 🔨 🖈	「年」 藍 藍   籔   謙   純   ほ 🍃   🍓 🏕 ¶   田 ( 露   ] ● 🗉   🗉   🗣 🕨
Counter (Counter)	B B 🗠 ∽ ∽ 👭 🚽 🚽 🖓 👘 🔍 Q Q Q 🤞 🔺 🔧 🔧
Counter (Counter)	Image: Incomposition of the second secon
25 26	Inserts template into the document

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

```
Dołączamy bibliotekę
     library IEEE;
3
     use IEEE.std logic 1164.all;
                                        i jej pakiety
4
     use IEEE.std logic unsigned.all;
6
     entity Top is
7
        port (
8
            CLK : in STD LOGIC;
9
            RESET : in STD LOGIC;
            Output : out STD LOGIC VECTOR (9 downto 0)
11
        );
12
     end Top;
13
14
     architecture Structure of Top is
15
16
        component Counter is
                                       Deklarujemy
17
            port (
18
                CLK : in STD LOGIC;
                                       komponent Counter
19
                RESET : in STD LOGIC;
                Q : out STD_LOGIC_VECTOR (3 downto 0)
21
            );
22
        end component;
23
        component Decoder is Deklarujemy komponent Decoder
24
25
            port (
26
                DataIn : in STD LOGIC VECTOR (3 downto 0);
27
                Output : out STD LOGIC VECTOR (9 downto 0)
28
            );
29
        end component;
31
        signal Internal : STD LOGIC VECTOR (3 downto 0);
32
33
    begin
34
        port map (CLK, RESET, Internal), Łączymy komponenty ze sobą
        CNT : Counter
36
37
                                      korzystając z funkcji port map
38
        DEC : Decoder
                                     i notacji pozycyjnej
39
        port map (Internal, Output);
40
41
    end structure:
42
```

```
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.



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 rozwiń 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 osiągane 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)

Freiefices	
Category:	
Access to Design Objects	
Environment	
Appearance I Enable access to bits of vector in Verligg nets (+accb)	
Eila Extanciono	
Windows	
Console	_
B⊡-Design □ Read (+r)	
Advanced options	
Compilation Path (+p+<>)	
ti · VHDL Compiler	
E · Verilog Compiler	
Linting	
Path (+p+<>)	
Assertions	
Debugger	
- Access to Design Objects Unit (+m+<>)	
Advanced Dataflow Path (+n+<>)	
Riviera-PRO	
Generation	
- VHDL Case	1
Ji Veriloo Standard V Denouit On Califer Apply	

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.)

Category: Waveform Viewer/Editor	Preferences	×
Waveform Viewer/Editor	Category:	
B. Verlog Compiler Unting     ALINT Simulation     Verlog     Assertions     Debugger     Access to Design Objects     Memory Management     Advanced Dataflow     Riviera PRO     Generation     Verlog Standard     B. File Headers     B. Copy Instantiation     Verlog Standard     B. File Headers     B. Copy Instantiation     Verlog DI Application     Editors     B. State Diagram Editor     B. State Diagram Editor	E- Verilog Compiler     Linting     ALINT     Simulation     Verilog     Assertions     Debugger     Ascess to Design Objects     Memory Management     Advanced Dataflow     Riviera-PRO     Generation     Verilog Standard     B-File Headers     E- Copy Instantiation     Verilog PLI Application     Editors     B-HDL Editor     Block Diagram Editor     H-State Diagram Editor     Source Control     Source Control	Waveform Viewer/Editor         □ Efault waveform viewer/editor:         □ Standard Waveform Viewer/Editor         □ Use one CPU core for simulation kernel and ASDB processing         □ When simulation results are saved to the ASDB database, Active+HDL can automatically distribute simulation tasks and processes related to updating simulation database among available CPU cores. The use of multiple cores increases simulation performance and simultaneously allocates more system resources.         If this option is checked, the use of multiple CPU cores is disabled, which may decrease simulation performance and allocation of system resources.
Team   Default  OK  Cancel  Apply	Team	Default     OK     Cancel     Apply

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) Dołóż sygnały do Standrad List Viewer podobnie jak w podpunkcie f). Powinieneś uzyskać efekt przedstawiony na poniższym rysunku.

<u>File Edit Search View Workspace</u>	<u>D</u> esign <u>S</u> imulation <u>L</u> ist <u>T</u> ools <u>W</u> indow <u>H</u> elp	t, × ×
🖉 🕶 🖨 👷 🐰 🛄 🚳 💱 🖉	💻 🔎 🔛 🗊 🔍 🍏 🚏 🖓 🖺 🥸 🕼 🚸 🖉 😓 🔸 🖢 😓 📗 🕨 🛓 100 ns 🗄 📢 🔳 🛃 1	▶ ਓ⊒ ⋤⊒ ∞
Design Browser 🔷 🔺	¶u ∰ •!	
Top (Structure)	Time Delta Mr /Top/Internal Pr /Top/CLK Pr /Top/RESET - /Top/Output	^
Hierarchy		
E-Top (Structure)		
- P std.standard		
- std. IEXTIO		
-P ieee.std_logic_arith		
P ieee.STD_LOGIC_UN		
<b></b>		
Name Value		
► CLK U		
► RESET U		
🗄 📲 Output UUU		
🗄 🏧 Internal 🛛 U		
Files } ¥Struc/@Reso/	📲 design flow 💒 counter.vhd 💒 decoder.vhd 💒 top.vhd 👔 🛣 lis	it1

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 Wavewform. 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).

Name	Value	Stimulator	0.05	2,0	i 4,0	I.	6,0	1	8,0	1	100	I	120	I	140	Т	160	Т	180
± # Internal	U		Lo ba																
CLK	U																		
► RESET	U																		
🛨 🗝 Output	υυυ																		
Stimulate	ors		I															Σ	3
Signals H	otkeys Prede	fined																	
Set :			-		New		1	B	emov	/e	1								
Signals:			Type:																
		Type	Ci For 0 1	ock f(t) mula	)	1		e - <u>[0f</u> :	3	] <b>→</b>	que	ncy	-[100	)ns J MHz	} 50 %	~	-		
Display	/ paths	Save		alue	-	$\langle$	A	pply			St	ren	gth:	0	vemid	e	Clos	▼ se	

Rys. 34. Przypisywanie wymuszeń do sygnałów.

k) Zaznacz sygnał "RESET" i przypisz mu wymuszenie w postaci "Formula".
 Wklej formułę:
 1 0, 0 10000

i kliknij Apply a następnie Close.

Name	Value	Stimulator	0 ps 20 + 40	ço i 60 i 80 i 100 i 120 i 140 i 160 i 180
표 🏧 Internal	U		0 93	
<mark>► CLK</mark>	0	Clock		
• RESET	1	Formula		
🛨 🗝 Output	υυυ			
Signals H	ors otkeys Prede	fined		
Set :		-	New	N Remove
Name ✓ CLK ✓ RESI	(	Type C <b>lock</b> Formula	Clock	Forces a waveform defined by a textual formula.
			Formula 010 110 Value	repeat above sequence every:
are Display	/ paths	Save		Apply Strength: Override
a.				Close

Rys. 35. Przypisanie wymuszeń do sygnałów – cd.

 Wykonaj kilkanaście kroków symulacji klikając każdorazowo ikonę skrótu "Run For". Nigdy nie uruchamiaj symulacji klikając skrót "Run" (nie mamy zdefiniowanego momentu zakończenia symulacji; symulacja będzie się wykonywać "w nieskończoność" i zawiesi komputer).



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".

A Save As	-			23
Zapisz w:	📗 src	•	+ 🗈 💣 📰 🕇	1
Ostatnie miejsca Pulpit Biblioteki Komputer	Nazwa	vhd vhd	Data modyfikacji 2014-04-15 16:15 2014-04-16 14:49 2014-04-15 22:53	Typ Plik VHD Plik VHD Plik VHD
	•	III		÷.
	Nazwa pliku:	Waveform Editor 1.awf	•	Zapisz
	Zapisz jako typ:	Waveform File (*.*) add new files to design	•	Anuluj

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

2<sup>4</sup>2 XX Y

liczbę wyników klikając 🛱 . Wtedy będzie widoczny tylko wynik ostatniego cyklu delta.

The East Sealer	Ten nonspace	Design Sinna		<u>10013 Millidow II</u> e	<b>י</b> ץ				·
🛛 🗗 🎽 🛣	88 🏛 🚯 🏂	📃 🔎 🚟 🎘 🛯	L 🛍 🔽 🖓	🌇 🐂 🕺 🕒	• 🕹 🕹 🕨	🕨 🕨 🛓 100 ns	∃∢ ∎ ⊴ ⊳	⋐⊒ ⋤⊒ ≫	
Design Browse	r × ×	<b>*</b> ∎ ∰							_
Top (Structure)	-	Time	Delta	🏧 /Top/Internal	/Top/CLK	/Top/RESET	P /Top/Output		*
Hierarchy		900.000 ns	0	9	0	0	001		
	.)	950.000 ns	0	9	1	0	001		
E-1: CNT : coun	nter (co	950.000 ns	1	0	1	0	001		
EC : deco	der (de	950.000 ns	2	0	1	0	200		
- B std.standard		1.000 us	0	0	0	0	200		
std.TEXTIO		1.050 us	0	0	1	0	200		
ieee.std_logic_	_1164	1.050 us	1	1	1	0	200		
eee.std_logic	_arith	1.050 us	2	1	1	0	100		
	IC_UN	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		
Name	Value	1.200 us	0	2	0	0	080		_
	Unavailable								
P RESEI	Unavailable								
- Output	Unavailable								
" Internal	Unavailable								
	•	1	Þ	<					) <b>-</b>
📄 Files / 🌮 Struc	c/@Reso/	🛂 design flo	w 💒 count	er.vhd 💒 decoder.	vhd 💒 top.	/hd 🛛 🛓 🛣 wavefor	rm e ្ត្រឹង list1		

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 zmień 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".

stbench Generator W	/izard		-	×
Select the design unit generate appropriate	for which you war source files and a	nt to generate a macro file for th	a testbench. The ne testbench.	wizard will
Entity:				
top				•
Architecture				
structure				•
⊂ Testbench Type:				
	< Wstecz	Dalej >	Anuluj	Pomoc

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.)

stbench Generator Wizard	25
Define test vectors.	
✓ Test vectors from file Select this check box if you want to use saved in a waveform file.	e previously created test vectors
Select a test vector file:	
D:\Elektronika\ActiveHDL\Zaoczni_La	ab1_8_00\tutorial1\src\Waveform Editor 1.awf
	Browse
Signals found in file:	UUT ports:
CLK RESET	CLK RESET
< Wstecz	Dalej > Anuluj Pomoc

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".

Testbench Generator Wizard
Enter the testbench specification.
Type the name of the testbench entity:
top_tb
Type the name of the testbench architecture:
TB_ARCHITECTURE
Type the name of the testbench source file:
top_TB.vhd
Browse
Type the name of the folder for testbench files:
TestBench
< Wstecz Dalej > Anuluj Pomoc

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 testbencha klikając "Zakończ".

Testbench Generator Wizard			×
The wizard is ready to gener	rate testbench files.		
The following files will be ge	nerated:		
Testbench file(s):			
.\src\TestBench\top_TB	.vhd		
Dia with a sufficient to a fact			
Src\TestBench\top_TB	iming simulation:		
Generate	orgo		
- Simulation macro (DO file)	:		
.\src\TestBench\top_TB	_runtest.do		
< W	/stecz Zakończ	Anuluj	Pomoc

Rys. 44. Generowanie testbench – cd.

W Design Browser zostanie uwożony folder zawierający plik vhdl 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).

Active-HDL 9.3 (Zaoczni_Lab1_8_00 ,tu	itorial1) - Waveform Editor 2 *			
<u>File Edit Search View Workspace</u>	Design Simulation Wave	form <u>T</u> ools <u>W</u> indow <u>H</u> elp		ć∱ ≫ ×
🛛 🕶 🚔 🔛 👯 📖 🔳 🚳 💱 🖉	I 🖓 🔛 🕃 🔍 🛍 🛂	🗠 🍪 🔶 🖉 🚳 🖬	🕨 🍋 🕨 100 ns 🗧 📢 🔳 🚽 🕪	ਓ⊒ ⊊≡ ≫
Design Browser 🔷 🔺	X 🖻 🖻 🖉 😒 🔖	🔍 म 🖓 🔍 🔍 🔍 🔩	u 🗤 👭 🚱 🖭 🎿 👫 👎 🦽 %	1 76
C testbench_for_top	Name Value	Sti 100 i 200	ı 300 ı 400 ı 500 ı 600 ı 700 ı	800 i 900 i 1000 i 1200 ns
Hierarchy	™ CLK 0			
+-1 testbench_for_top : t	■ RESET 0			
-P std.standard	🖃 🕊 Output 🛛 080	200 (100 (080	(040 X020 X010 X008 X004 X00	<u>)2 X001 X200 X100 X080</u>
std.TEXTIO	<sup>™</sup> Output(9) 0			
Pieee.std_logic_1164	<sup>™</sup> Output(8) 0			
ieee.std_logic_arith	™ Output(7) 1			
- leee.stb_todic_on	™ Output(6) 0			
	• Output(5) 0			
	<sup>™</sup> Output(4) 0			
	r Output(3) 0			
	Jutput(2) 0			
Name Value	<sup>™</sup> Output(1) 0			
AF CLK 0	<sup>™</sup> Output(0) 0			
AF RESET 0				
r END_SIM true				
				-
		•		H Q H 4
E Files / V Struc/ Reso/	📲 design flow 🚊 coun	ter.vhd 💒 decoder.vhd 💒	top.vhd 👌 🛣 waveform e 🖓 🐘 list1	<u>∧ ⊠</u> waveform e…

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.

Wprawdzie 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".

'tutorial1' Design Settings	×	'tutorial1' Design Settings	×
Category: General Design Structure	VHDL Language Specification Standard version VHDL 1076-2002	Category: General Design Structure	Verilog Include directories:
Status     Status     Status     Status			Verlog libraries: Defined macros:
SDF     VFOL     Verlag     Access to Design Objects     Startions     Generic/Parameters     Trace/Debug     Rvirer aPRO/VSimSA     Coverage/Profiler     - Code Coverage     Si- Expression/Path Coverage     Derorelage		SDF     VHDL     Verlog     Access to Design Objects     Generic/Parameters     Trace/Debug     Riviera-PRO/VSImSA     Coverage/Profiler     Code Coverage     Expression/Path Coverage     Expression/Path Coverage	Standard version:     Verligg 1364-2005     ▼     Strict LRM mode       □ Ignore Timing from Specify Blocks      Strict LRM mode       Optimization level:     Level 2     ✓       Protection level:     Level 0     ▼       Severity level:     Errors + Warnings     ▼       Stop compliation after     0     errors
Todde Coverade	Default OK Cancel Apply	Todale Coverade	Default OK Cancel Apply

Rys. 48. Włączenie debugowania w ustawieniach projektu.

- b) Wyłącz poprzednią symulację (uruchomioną przez wykonanie makra TestBench) wybierając z manu 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.

<u>File Edit Search View Workspace</u>	Design Simulation Waveform	<u>T</u> ools <u>W</u> indow <u>H</u> elp			+>>×>
🗗 🕶 🖶 💥 📖 🔳 🚳 💱 💄	Į 🔎 🔛 🌮 🍋 🍏 🛂 🖓 🔝	🎭 🔕 🔓 🕫 🍪 🚽	▶ ▶ ▶ ▶ 100 ns ÷	📢 🔳 🛃 🕪 🖣 🖓 🖓	»>
Design Browser 🔷 🗙	X 🖻 🖻 🗠 😒 ┣ 🔍	다 🗟 🔍 이 이 🔍 🖏	씨 유 🔊 🖭 🕯	14 11 🗡 🕉 🌾 👘	
C testbench_for_top	Name Value	Sti 0 ps	300 i 400 i 500 i	600 i 700 i 800 i 90	00 i 1000 i 1100 i <sup>ns</sup>
Hierarchy	™ CLK U				
testbench_for_top : t	■ RESET U	1			
-P std.standard	🗆 🕊 Output 🛛 🛛 UUU	200/100 /080 /0	40 X020 X010 X0	008 X004 X002 X001	<u> </u>
-P std.TEXTIO	🅶 Output(9) 🛛 U				
P ieee.std logic_1164	🕶 Output(8) U				
P ieee.STD LOGIC UN	🕶 Output(7) U				
Eles Struc / Beso	💶 design flow 🖉 sounterst		n vind (*) vinverform a		• • • • • •
			p.vnd <u>A sa</u> wavelorm e	e <u>k</u> eie listi <u>kas</u> iwah	velorm e
• # Signal /Top/Internal n	not 🔺 🚺 Name	Туре	Value	Last Value	Last Event Time
found in design. • # Signal /Top/CLK not for	cund Cli	ick here			
in design.					
<pre>signal /Top/RESET not in design</pre>	found				
<pre># Signal /Top/Output not</pre>					
>					
Console /					
Label Hierarchy path	Status				
STIMULUS /top_tb	Ready				
CLOCK_CLK /top_tb	Ready				
line_44 /top_tb/UUT/CNT	Ready				
Line_42 /top_tb/UUT/DEC	Ready				
( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( )					
× Name	Valua	Last Value			<u>•</u>
	Value				
ő					
<u>C</u>					
			• -		

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.

F 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.

•		Name	Туре	Value	Last Value	Last Event	t 🔺
		" /top_tb/CLK	STD_LOGIC	0	υ	Ofs	
e	1	" /top_tb/RESET	STD_LOGIC	0	1	10ns	-
			STD_LOGIC_VECTO	200	000	Ofs	=
		" /top_tb/END_SIM	boolean	false	false	Ofs	
		Click here to add ne					-
	٠					•	

Rys. 51. Zmiana sygnału RESET.

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

	Label	Hierarchy path	Status
	line_44	/top_tb/UUT/CNT	Ready
7	STIMULUS	/top_tb	Wait
3	CTOCK_CTK	/top_tb	Wait
	line42	/top_tb/UUT/DEC	Wait

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

The East Search Mew Workspace	e <u>D</u> esign <u>Simulation</u>					· · · · ·
🖉 🕶 🖬 🐇 🐰 🔳 🚳 💱	📃 🔎 🚟 🍃 🔍 🎁	। 🛂 🖓 🜇 🙀 🐼 😓 🚸 🏟 🎝	👌 📔 🕨 🕨 🕨 100 ns	: 🕂 📢 🔳 🔳	I⊳ ਯ≣ ⊊≣ ∞	
Design Browser 🔷 🔺	· · · · · · · · · · · · · · · · · · ·	🛒 🕬 🕊 🚦 📑 🚱 📣 👖 🚦	1 😹 🛛 🗉 🔳	R 🕨		
top_tb (TB_ARCHITECTURE)	8 B B 10 9	× #	- N N + 4		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
Hierarchy	44 proce	SS (CLK, RESET)				
	45 <b>v</b>	ariable Qint : STD_LOGIC_VE	CTOR (3 downto 0	);		_
-P std.standard	46 begin					
-B std.TEXTIO	47 🗘 1	f RESET = '1' then				
-P ieee.std_logic_1164	49 e	lise				•
-P ieee.std_logic_arith	50	if CLK'event and CLK = '	1' then			±
eee.STD_LOGIC_UN	51	if Qint <9 then				*
	11					▶
Files Struc/ EReso/	📲 design flow 🖉	🛿 counter.vhd 🖉 decoder.vhd 💒	top.vhd 🔬 🛣 wavefo	orm e 🖉 top_	tb.vhd	
1 stan -over		News	T	V-L-	Last Value	Last Friend A
× step -over	- ×	Name	stp. Locto	value		
🚨 🛛 step -over	6		STD_LOGIC	0	1	10
step -over	_ 6		STD_LOGIC	200	-	E
step -over		(the the the TWD STM	SID_LOGIC_VECTO	200	6-2	OFF
	•	- /top_tb/END_SIM	Doolean	Talse	Ialse	UIS
Console /		Click here to add ne				
						,
Label Hierarchy path	Status					
line_44 /top_tb/UUT/CN1	r Ready					
STIMULUS /top_tb	Wait					
CLOCK_CLK /top_tb	Wait					
line_42 /top_tb/UUT/DEC	C Wait					
/top_tb/UU1/CN1/line_44						
Name Type	Value	Last Value				
Image: W <sup>±</sup> Qint     STD_LOGIC_VECTO     0						
<u></u>						
8						
30.7.00 P						

## Rys. 53. Analiza kodu.

22. Analiza projektu z wykorzystaniem okna Dataflow.

Okno Dataflow jest narzędziem pozwalającym na graficzną reprezentację sygnałów wypływających i wpływających do procesów w trakcie symulacji. Przez proces jest tu rozumiana dowolna instrukcja współbieżna modelująca sekwencyjny proces (instrukcja proces, współbieżne przypisanie wartości sygnałom, 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 klikanie elementów w oknie Dataflow. W obu widokach sygnały są wyświetlone 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).

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





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.