Wprowadzenie:

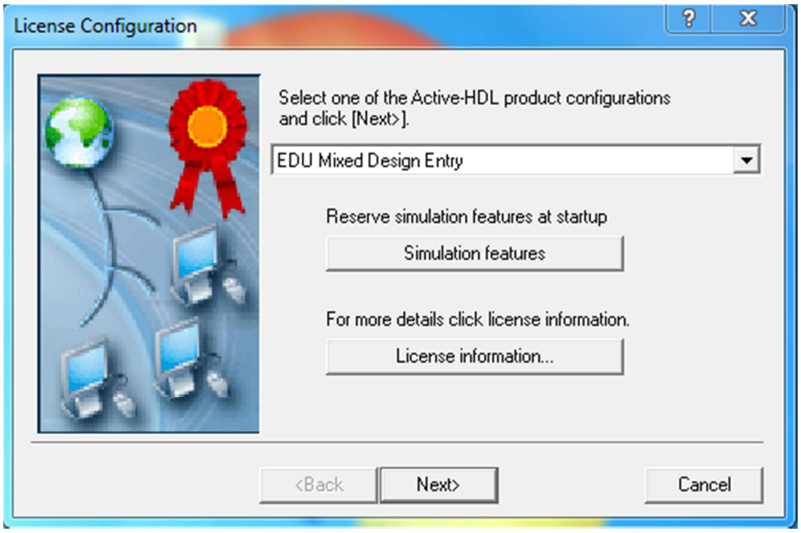
Ten tutorial uczy jak tworzyć projekty wykorzystujące edytor schematów Block Diagram Editor. Zakładamy, że student jest zaznajomiony z podstawową obsługą programu Active-HDL. W tym tutorialu użyjemy języka VHDL do zamodelowania pełnego sumatora (1-bitowego, z wejściem i wyjściem przeniesienia). Następnie, z pomocą edytora schematów, zaprojektujemy trzybitowy sumator z przeniesieniem szeregowym. Na kolejnym schemacie połączymy sumator z dekoderem wyświetlacza siedmiosegmentowego. Przy czym, dekoder będzie zrealizowany w języku VHDL.

Tworzymy nowy projekt

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

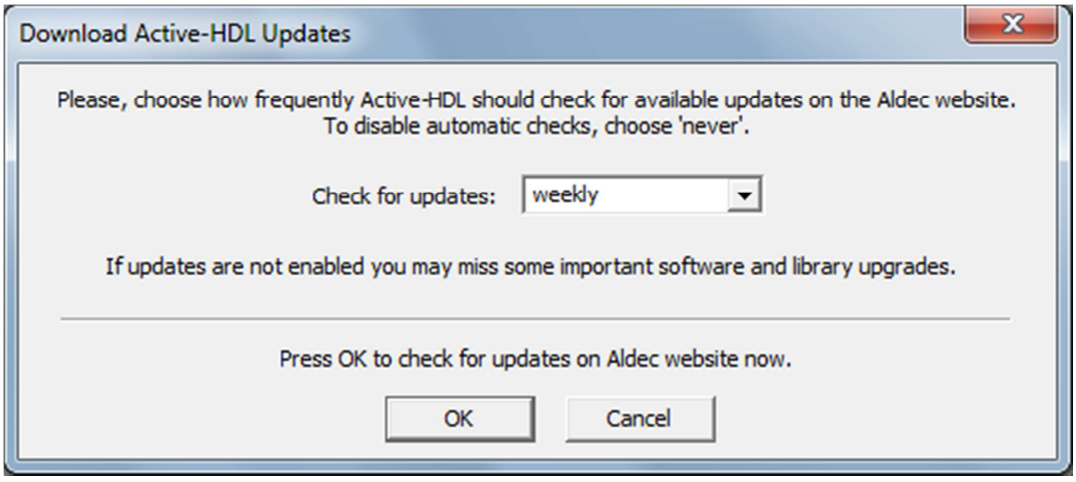
Rys. 1. Ikona na pulpicie – skrót do Aldec Active-HDL.

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



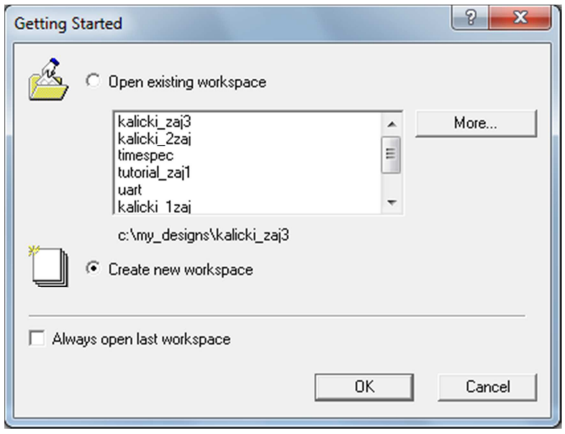
Rys. 2. Okno konfiguracji licencji.

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



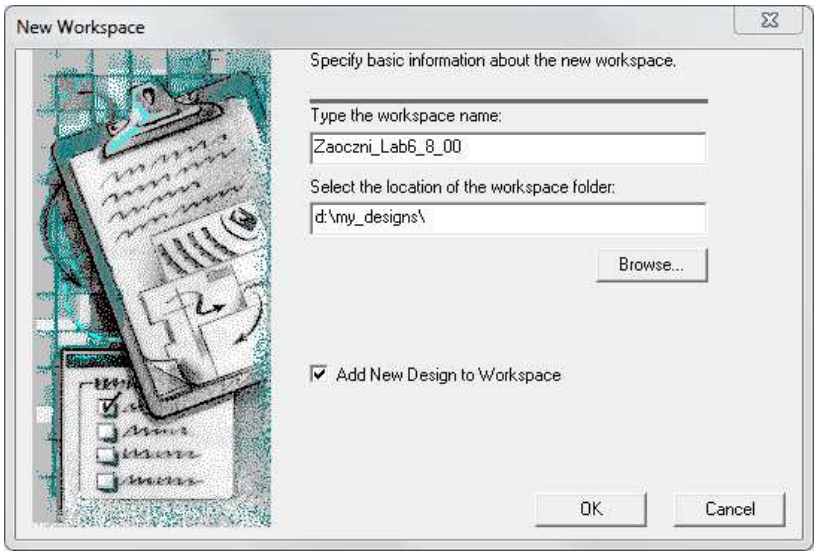
Rys. 3. Okno aktualizacji programu.

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



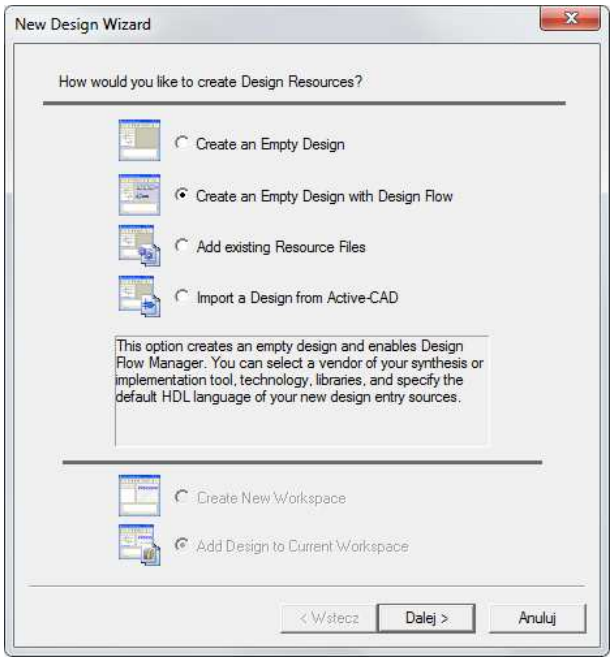
Rys. 4. Wybór przestrzeni roboczej.

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



Rys. 5. Nadanie nazwy przestrzeni roboczej.

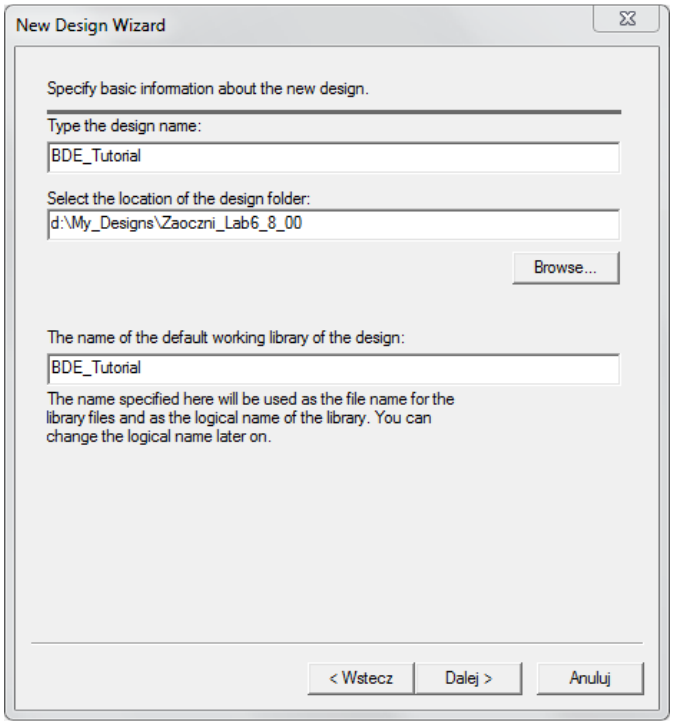
6. W kolejnym oknie wybrać „Create an Empty Design with Design Flow” i kliknąć „Dalej”.



Rys. 6. Wybór rodzaju projektu. Design Flow umożliwia uruchomienie syntezy i implementacji projektu.

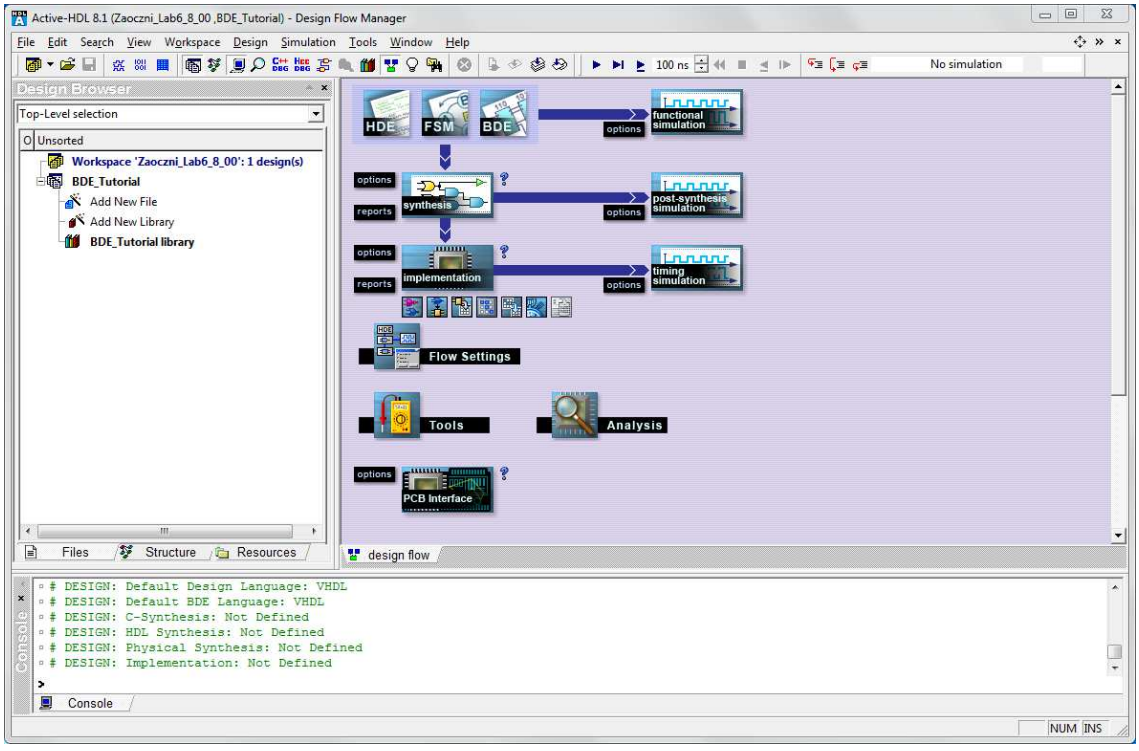
7. Sprawdzić czy w ustawieniach Design Flow są wskazane narzędzia „Synthesis tool” oraz „Implementation tool” (zapytać prowadzącego która wersja Xilinx ISE jest zainstalowana w laboratorium).
   Sprawdzić czy wybrana jest poprawna rodzina układów FPGA (zależy od wykorzystywanego zestawu ewaluacyjnego).
   Block Diagram Configuration: Default HDL Language
   Default HDL Language: VHDL
   W przypadku różnic kliknąć „Flow Settings” i dokonać odpowiednich zmian. Jeżeli wszystko się zgadza, kliknąć „Dalej”.

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



Rys. 7. Nadanie nazwy projektowi.

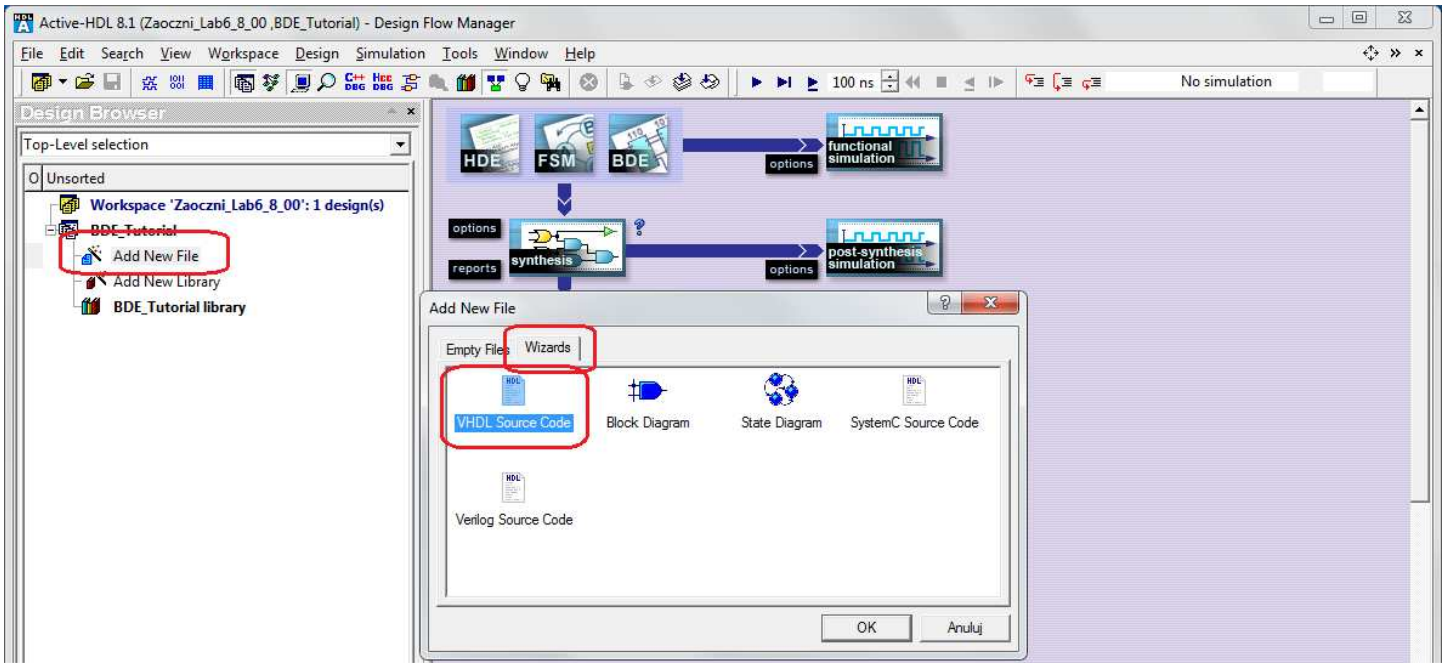
9. W kolejnym oknie wybrać „Zakończ”. Powinniśmy na ekranie otrzymać widok jak na poniższym rysunku.



Rys. 8. Główne okno programu po założeniu projektu.

Tworzymy nowy plik VHDL

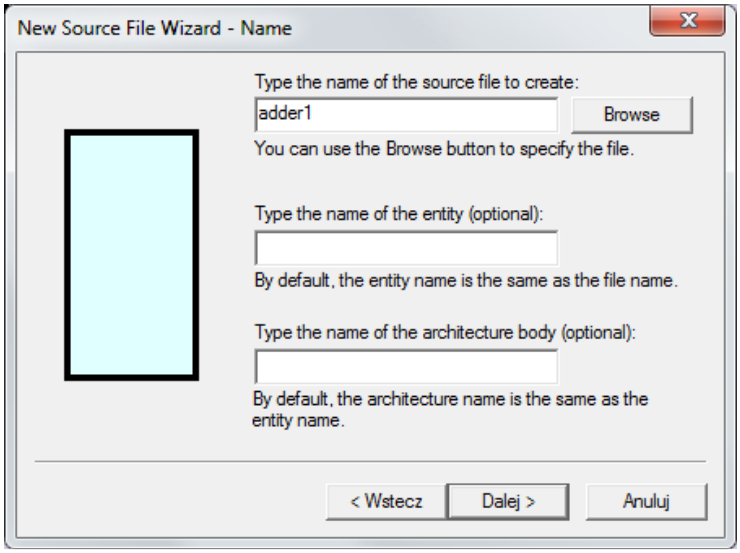
10. W Design Browser dwukrotni kliknij na Add New File. Pojawi się okno dialogowe umożliwiające dołożenie pliku do projektu.
    Zmień zakładkę na Wizards i wskaż VHDL Source Code. Kliknij OK.



Rys. 9. Użycie kreatora do stworzenia pliku VHDL.

11. W pierwszym oknie kreatora zostaw domyślnie zaznaczoną opcję „Add the generated file to the design” i kliknij Dalej.

12. W kolejnym oknie podaj nazwę pliku źródłowego do stworzenia. Wpisz adder1 jak pokazano na rysunku poniżej i kliknij Dalej.

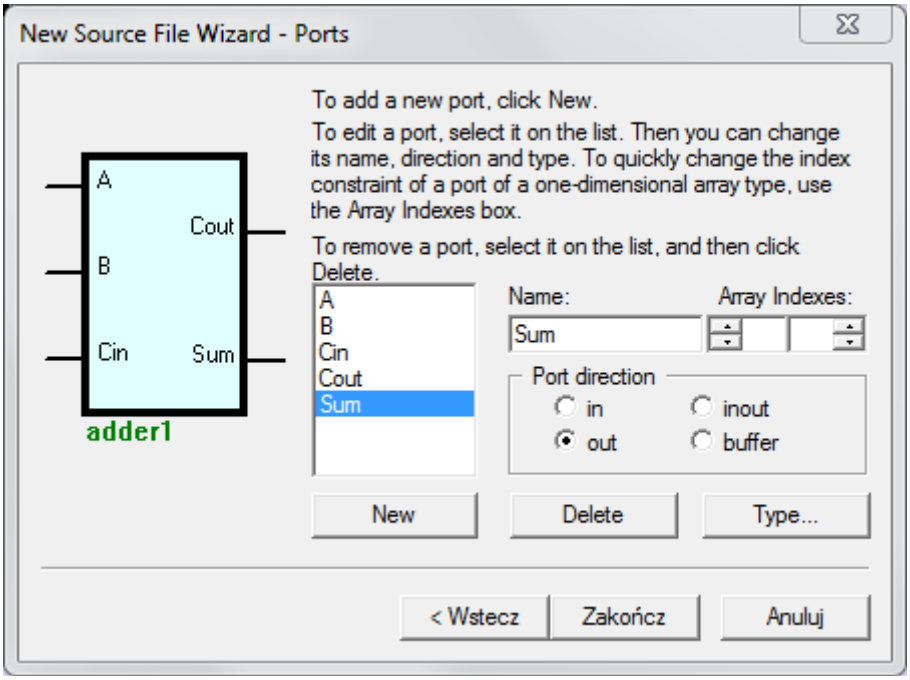


Rys. 10. Nadanie nazwy plikowi VHDL. Nazwy Entity i Architecture będą identyczne jak nazwa pliku.

13. Kolejne okno pozwala na dołożenie portów. 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 pięć sygnałów zgodnie z poniższą tabelą.

Name	Array Indexes	Port direction	Type
A	(puste)	In	STD_LOGIC
B	(puste)	In	STD_LOGIC
Cin	(puste)	In	STD_LOGIC
Cout	(puste)	Out	STD_LOGIC
Sum	(puste)	Out	STD_LOGIC

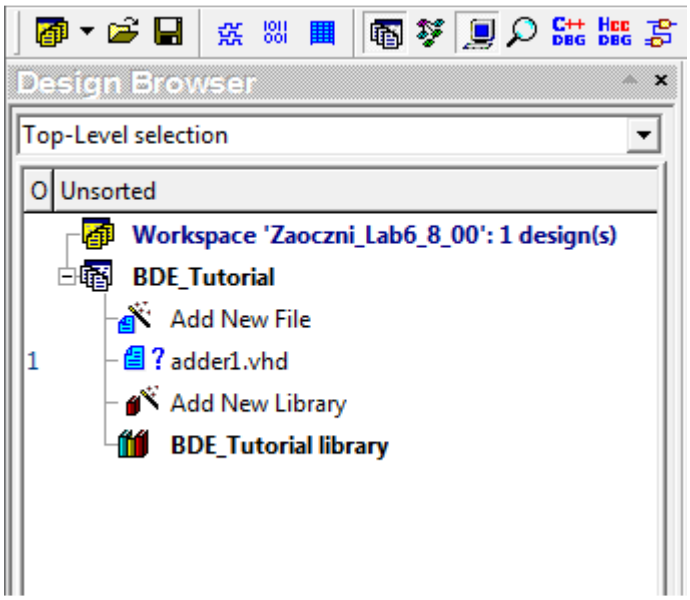
Aby zmodyfikować wcześniej wprowadzony sygnał należy zaznaczyć jego nazwę i dokonać odpowiednich zmian. Błędne / niepotrzebne sygnały można skasować przyciskiem „Delete” na klawiaturze.
W lewej części okna mamy graficzną reprezentację projektowanego elementu. Stworzone wejścia układają się na lewej krawędzi bloczka, z kolei wyjścia układu pojawiają się na prawej krawędzi. Magistrale oznaczane są grubszym „przewodem”.



Rys. 11. Konfiguracja portów.

14. Następnie kliknij „Zakończ”.

W oknie Design Browser pojawi się nowy plik – adder1.vhd. Plik ten zawiera kompletne entity oraz pustą architekturę. W kolejnych punktach stworzymy kod „Data Flow” opisujący działanie jednobitowego, pełnego sumatora.



Rys. 12. Nowy plik w Design Browser.

15. W Design Browser, dwukrotnie kliknij na pliku adder1.vhd aby wyświetlić jego zawartość. Przepisz kod zgodnie z poniższym rysunkiem.

```
Sum <= A xor B xor Cin;
Cout <= (Cin and A) or (Cin and B) or (A and B);
```

```
24
25  library IEEE;
26  use IEEE.STD_LOGIC_1164.all;
27
28  entity adder1 is
29       port(
30            A : in STD_LOGIC;
31            B : in STD_LOGIC;
32            Cin : in STD_LOGIC;
33            Cout : out STD_LOGIC;
34            Sum : out STD_LOGIC
35            );
36  end adder1;
37
38  --}} End of automatically maintained section
39
40  architecture adder1 of adder1 is
41  begin
42
43       -- enter your statements here --
44  Sum <= A xor B xor Cin;
45  Cout <= (Cin and A) or (Cin and B) or (A and B);
46
47  end adder1;
```

Rys. 13. Edycja kodu źródłowego.

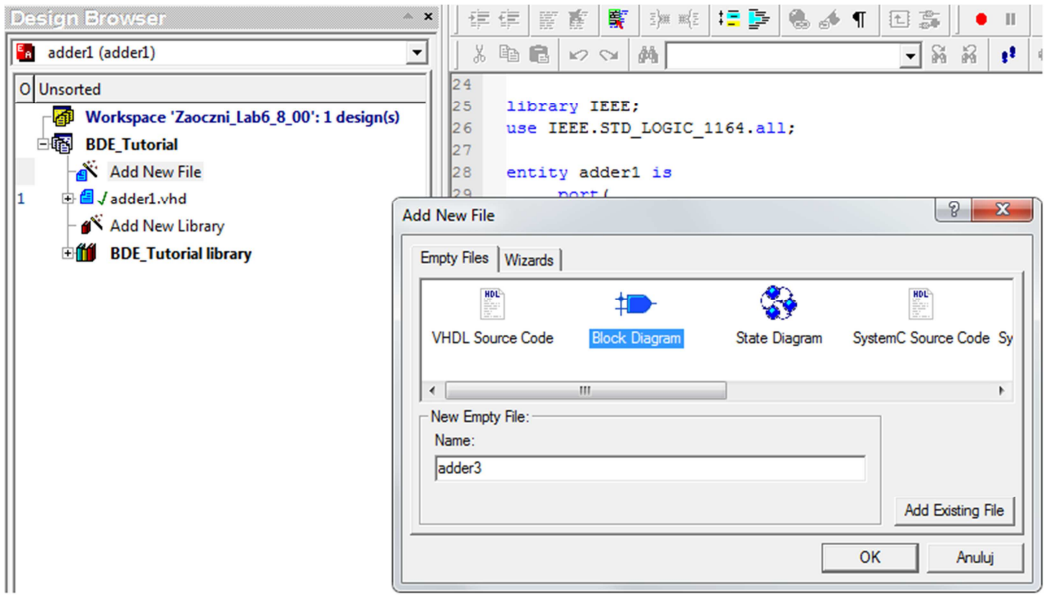
16. Zapisz zmiany w pliku.

17. Wybierz z głównego menu Design -> Compile.

Tworzymy nowy schemat blokowy

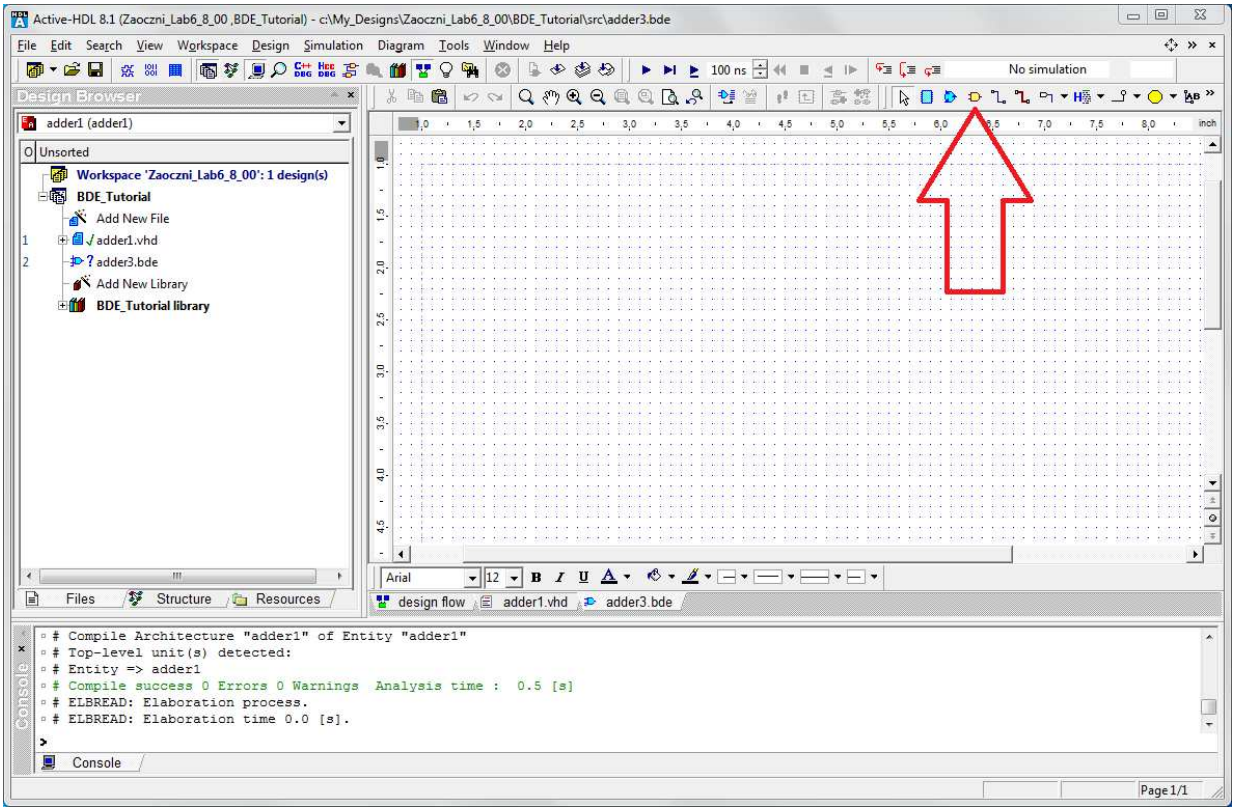
18. Ponownie, dwukrotnie kliknij Add New File w Design Browser.

19. Pozostań na zakładce Empty Files. Wskaż Block diagram. W polu Name wpisz adder3.



Rys. 14. Tworzenie schematu blokowego.

20. Kliknij OK. Otworzy się okno z pustym schematem blokowym.

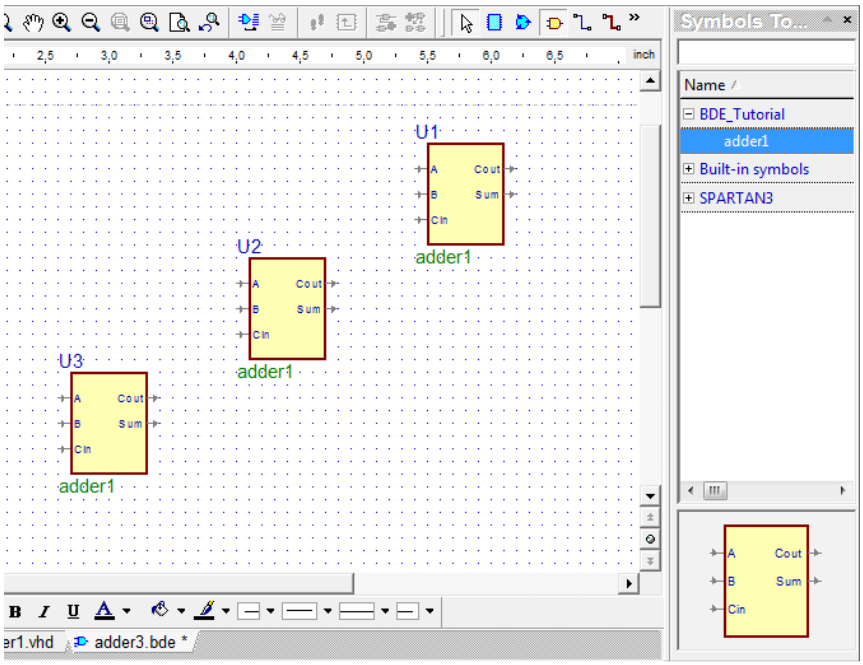


Rys. 15. Puste okno edytora schematów blokowych.

21. Kliknij żółtą ikonę bramki AND wskazana na Rysunku 14. Otworzy się biblioteka elementów / symboli. Dostępne elementy są pogrupowane.
    W „Built-in symbols” są elementy których NIE używamy!
    Jest też „biblioteka” elementów o nazwie takiej samej jak nasz projekt. Jeżeli skompilujemy dowolny plik źródłowy w projekcie (kod VHDL / kod Verilog / schemat blokowy) to będzie on dostępny w Symbols Toolbox.

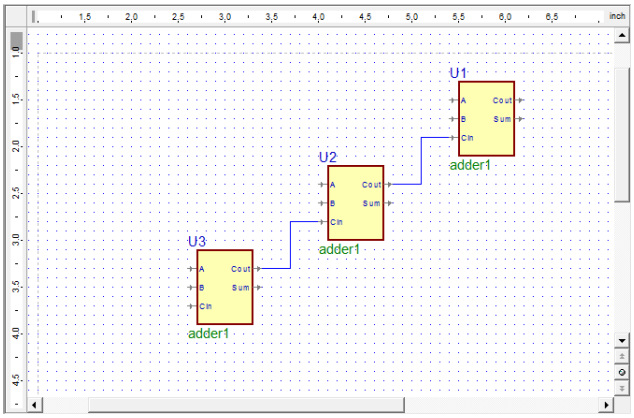
22. Kliknij na nazwie *adder1* w Symbols Toolbox, w dolnej części panelu pojawi się jego ikona.

23. Trzy razy przeciągnij na schemat symbol *adder1*.



Rys. 16. Umieszczenie elementu na schemacie blokowym.

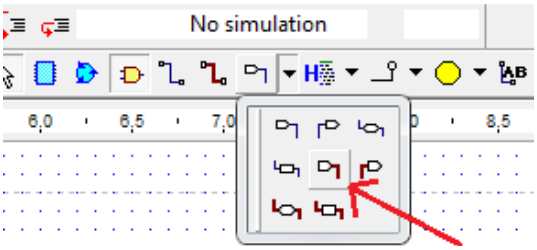
24. Narysuj połączenia między wyjściami Cout oraz wejściami Cin.



Rys. 17. Rysowanie połączeń.

25. Wybierz Bus Input Terminal tak jak pokazano poniżej.

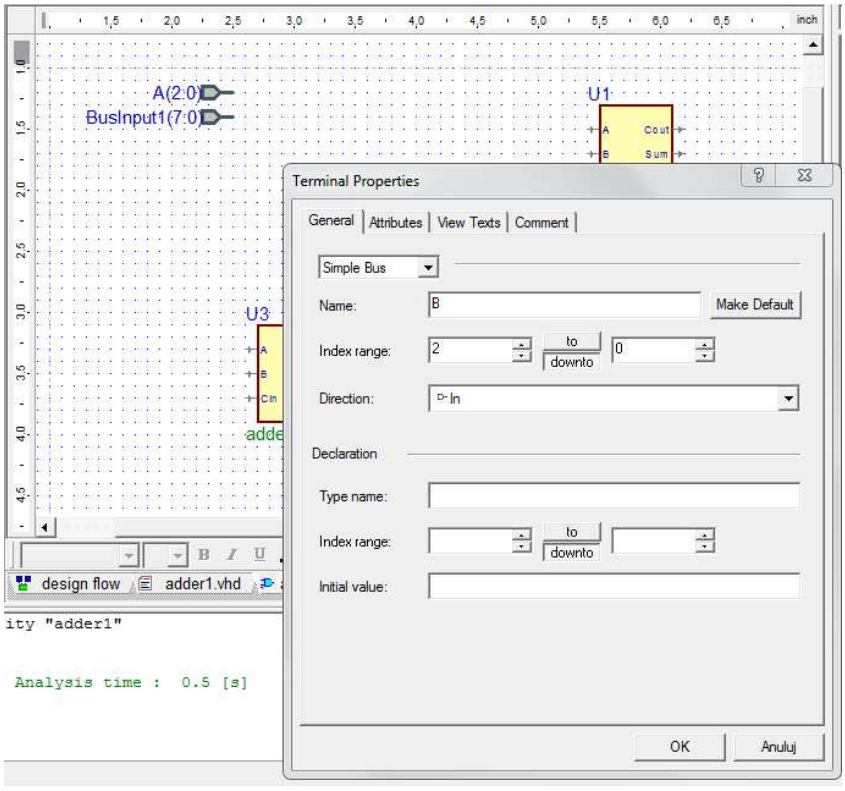
Używając tego narzędzia można wprowadzić na schemacie porty wejścia/wyjścia. W rozwijanym menu są różne ikony w zależności od kierunku sygnału (in / out / inout) oraz rodzaju portu (pojedynczy sygnał / magistrala).



Rys. 18. Narzędzia do wstawiania porów na schemacie.

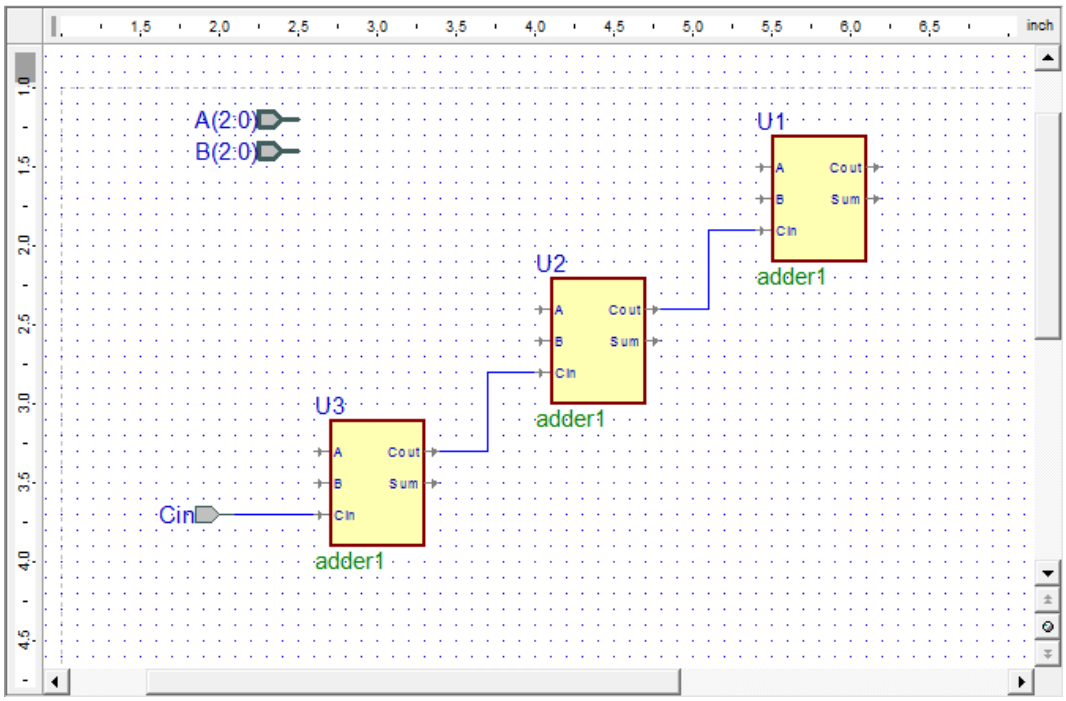
26. Umieść na schemacie porty dwóch magistral wejściowych.
    Naciśnij klawisz Esc aby zakończyć wstawianie portów i przełączyć się do trybu zaznaczania.

27. Kliknij dwukrotnie na porcie aby wyświetlić jego właściwości.
    Zmień nazwy portów na A i B. Zmień szerokość magistral na 2 downto 0.



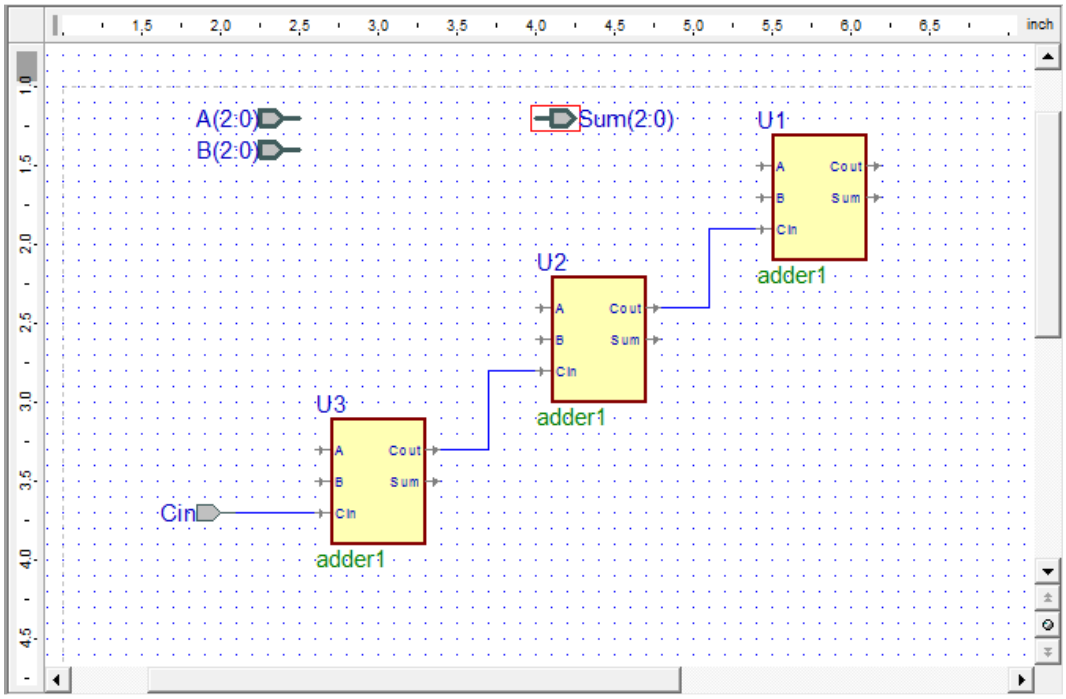
Rys. 19. Edycja nazw i szerokości portów na schemacie.

28. Wstaw Input Terminal (pin wejściowy dla pojedynczego przewodu).
    Zmień jego nazwę na Cin i podłącz go do wejścia Cin lewego, dolnego, pełnego sumatora.



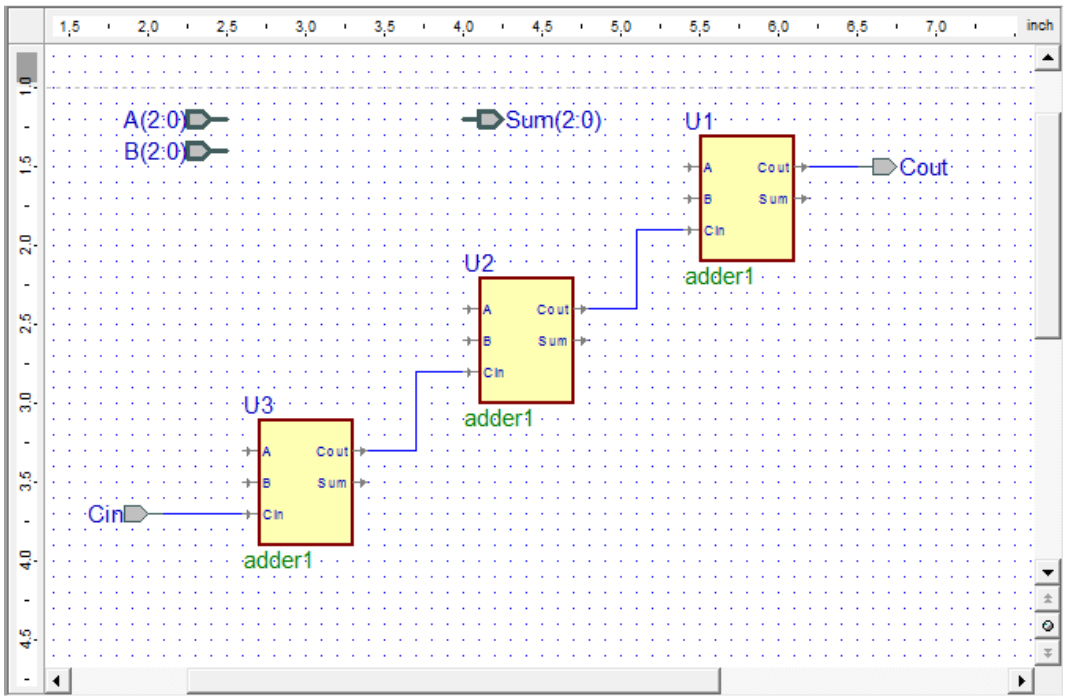
Rys. 20. Dołożenie wejścia Cin na schemacie.

29. Umieść na schemacie Bus Output Terminal (pin wyjściowy dla magistrali).
    Zmień jego nazwę na Sum a szerokość magistrali na 2 downto 0.



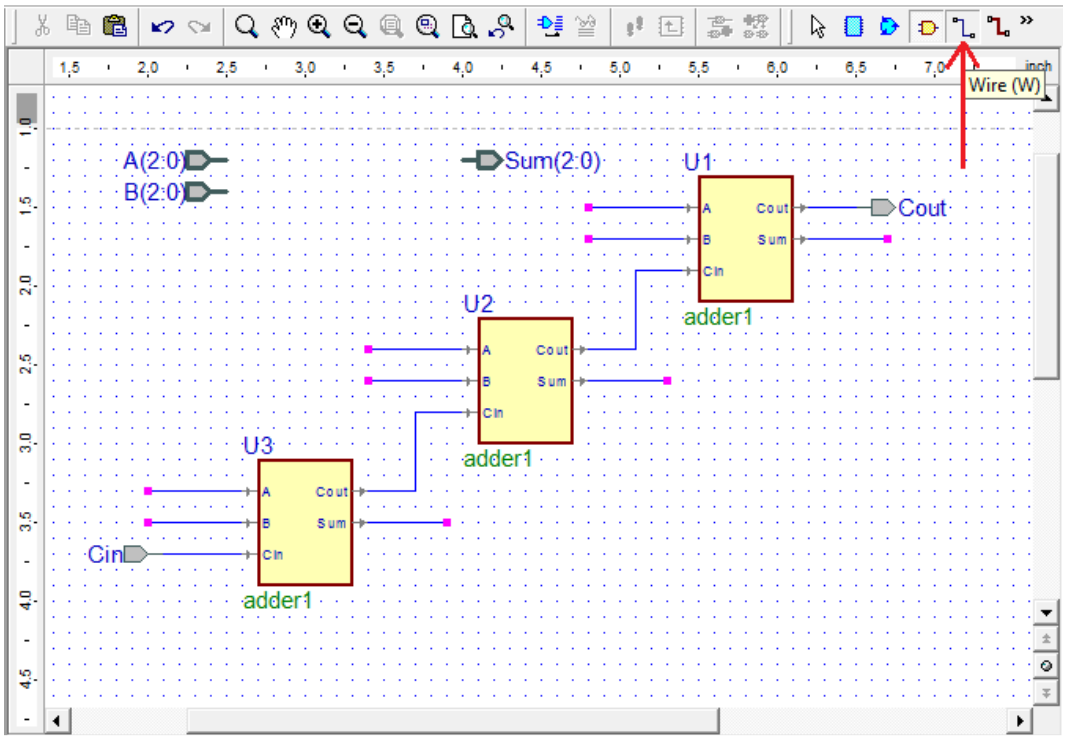
Rys. 21. Dołożenie portu wyjściowego Sum(2:0).

30. Wstaw Output Terminal (pin wyjściowy dla pojedynczego przewodu).
    Zmień jego nazwę na Cout i podłącz go do wyjścia Cout prawego, górnego, pełnego sumatora.



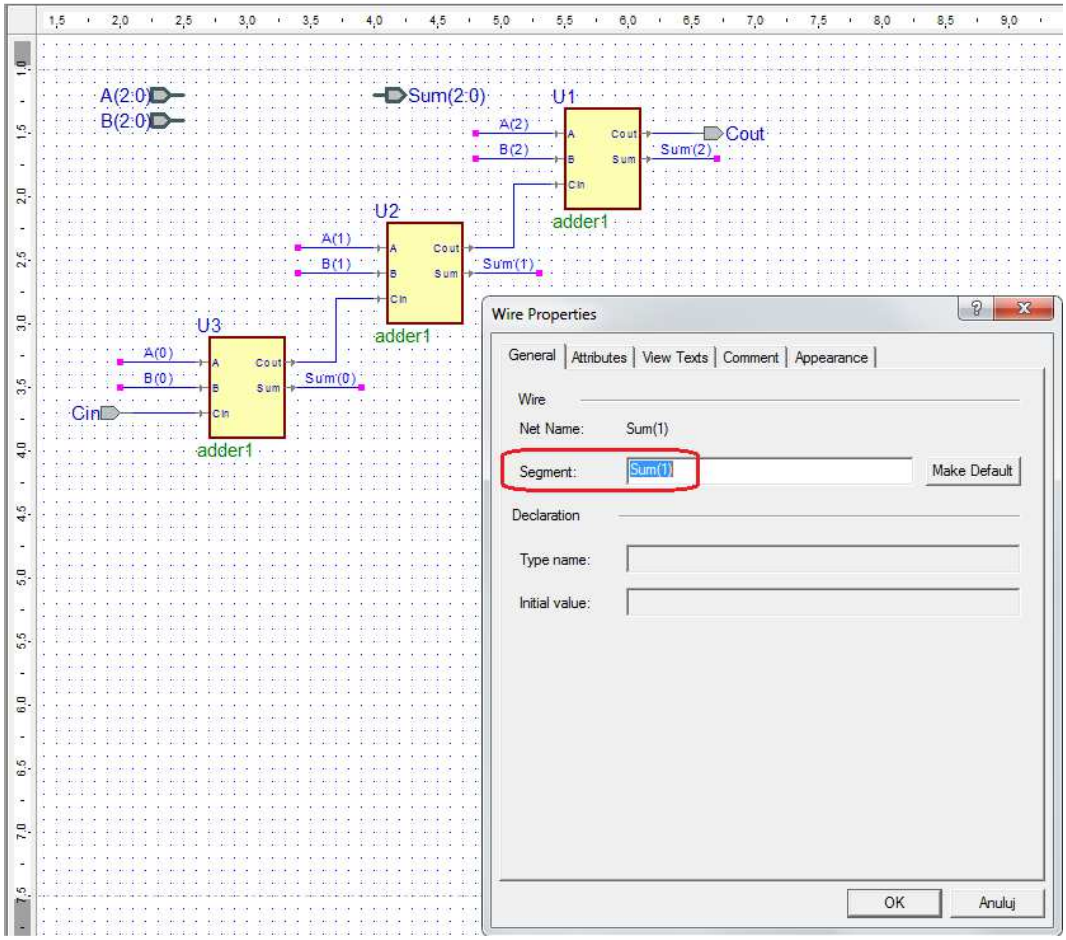
Rys. 22. Dołożenie wyjścia Cout na schemacie.

31. Narysuj przewody tak jak na rysunku 23. Użyj narzędzia Wire dostępnego na prawo od ikony Symbols Toolbox.
    Kliknij dwukrotnie aby zakończyć rysowanie przewodu.



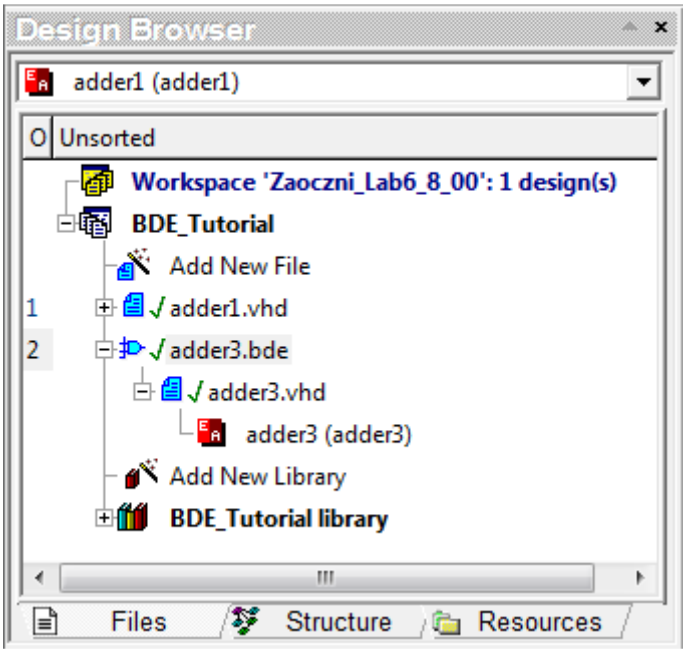
Rys. 23. Przygotowanie przewodów/połączeń.

32. Użyj etykiet A(0), A(1), A(2), B(0), B(1), B(2), Sum(0), Sum(1), Sum(2) aby połączyć schemat.
    Dwukrotnie kliknij w przewód aby wyświetlić jego właściwości. W polu Segment wpisz nazwę etykiety.



Rys. 24. Tworzenie połączeń z wykorzystaniem etykiet.

33. Wybierz z menu Design -> Compile.

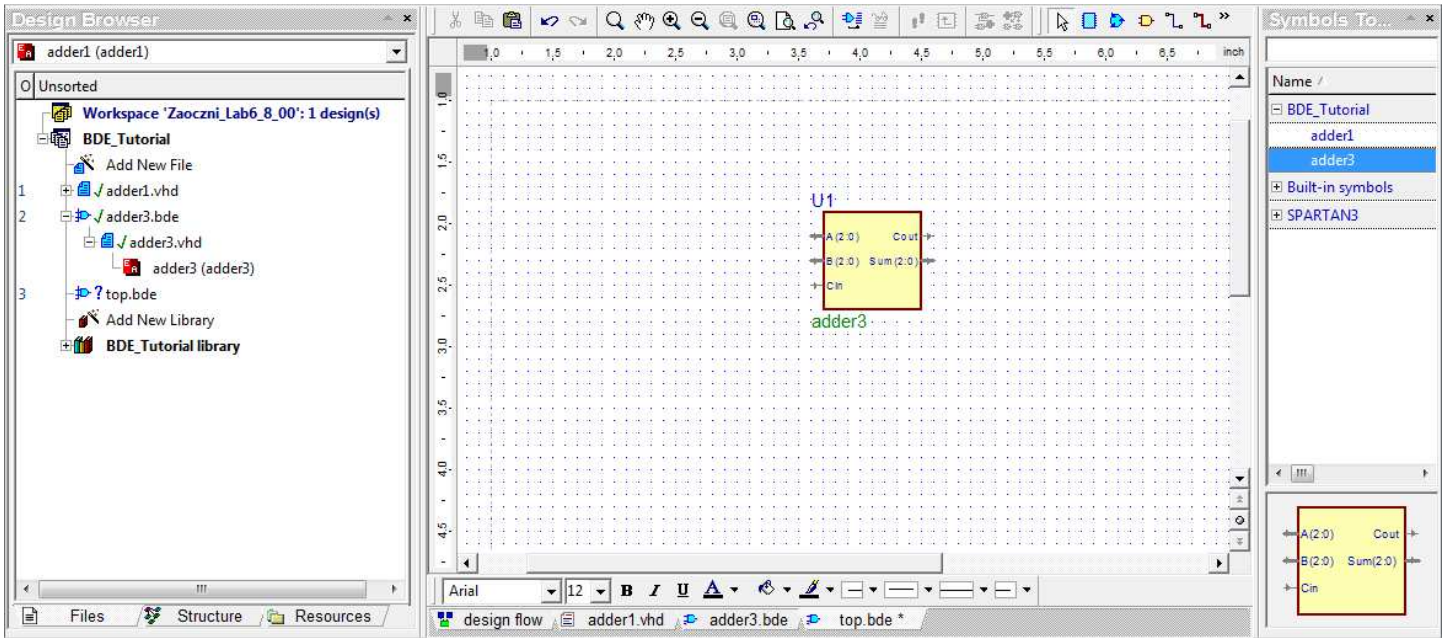


Rys. 25. Zawartość Design Browser po kompilacji schematu.

Tworzymy schemat Top

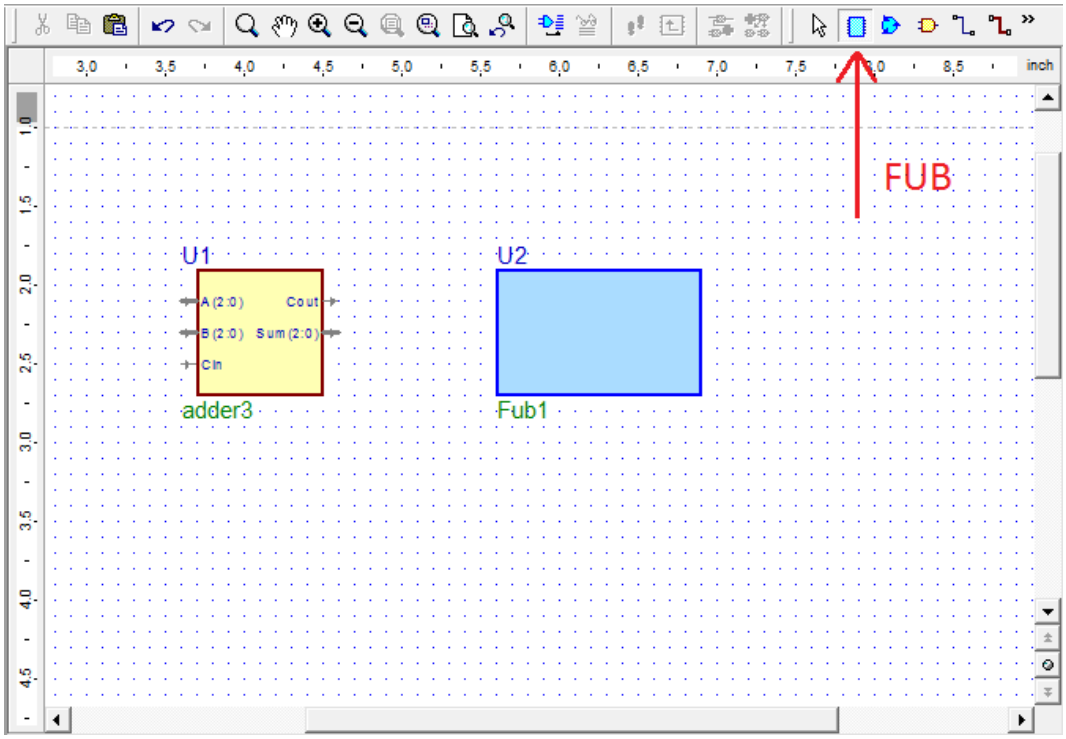
34. Postępując podobnie jak w punktach 18 – 20 stwórz kolejny schemat blokowy o nazwie **top**.
    Umieścimy na nim sumator 3-bitowy oraz dekoder wyświetlacza 7-segmentowego.

35. Umieść na schemacie „top” symbol 3-bitowego sumatora odszukany w Symbols Toolbox.



Rys. 26. Edycja schematu top.

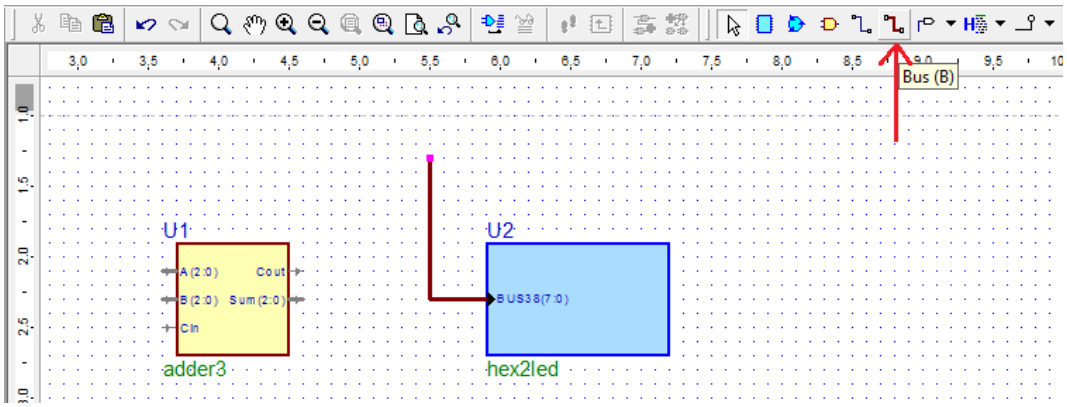
36. Na prawo od sumatora narysuj FUB.
    Naciśnij klawisz Esc aby zakończyć wstawianie FUB i przełączyć się do trybu zaznaczania.



Rys. 27. Schemat top po wstawieniu FUB.

37. Kliknij prawym przyciskiem myszy na FUB i wybierz Properties.
    W polu „Fub name:” zmień nazwę z **Fub1** na **hex2led**.

38. Wybierz narzędzie Bus i narysuj magistralę tak jak pokazano na Rysunku 28.
    Rysowanie magistrali zakończ na elemencie FUB tak by na jego krawędzi powstał pin wejściowy.

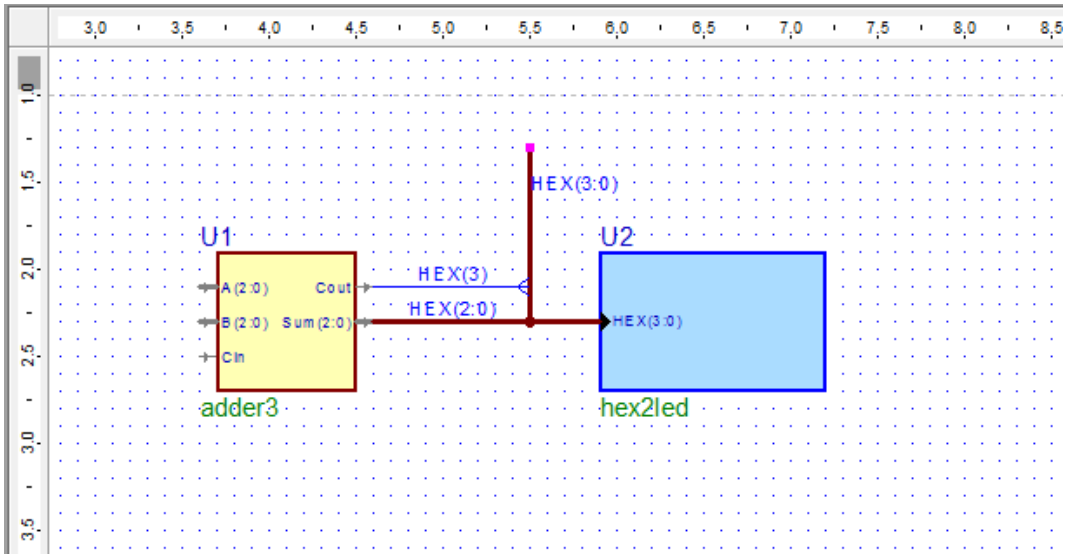


Rys. 28. Rysowanie magistrali.

39. Dwukrotni kliknij na magistrali aby wyświetlić jej właściwości.
    W polu „Segment” wpisz HEX(3:0) i kliknij OK aby zatwierdzić zmiany.

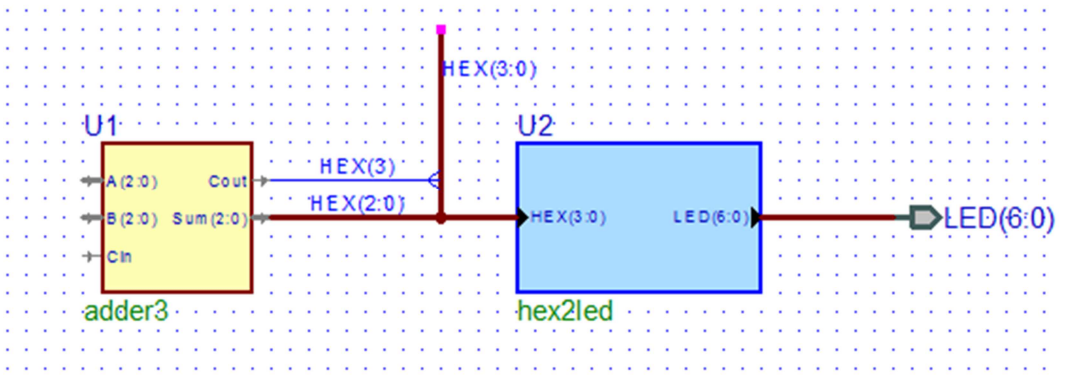
Nazwa pinu na elemencie FUB automatycznie przyjęła nazwę dołączonej magistrali.

40. Wykorzystując narzędzia Wire (pojedynczy przewód), Bus (magistrale) oraz etykiety, połącz sumator z FUB.



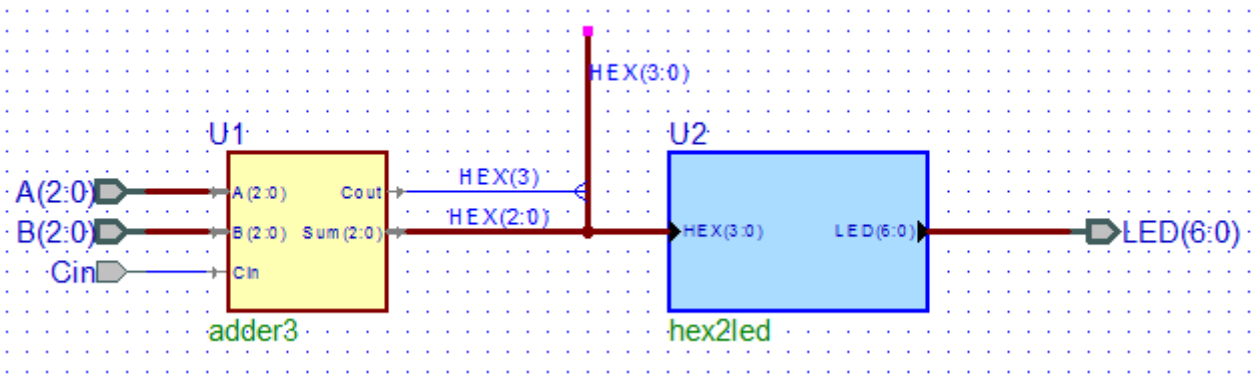
Rys. 29. Połączenie sumatora z FUB.

41. Umieść na schemacie Bus Output Terminal (pin wyjściowy dla magistrali).
    Zmień jego nazwę na LED a szerokość magistrali na 6 downto 0.
    Narysuj magistralę od FUB do pinu LED.



Rys. 30. Pin wyjściowy LED(6:0).

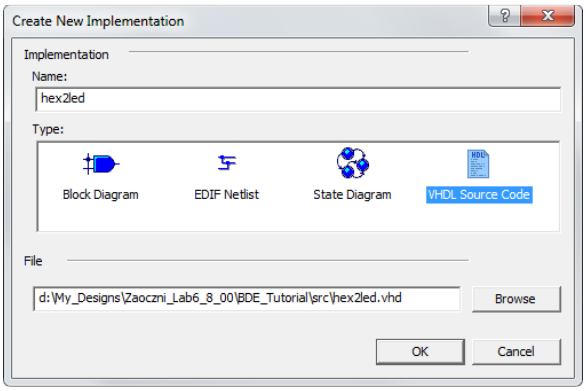
42. Umieść na schemacie Bus Input Terminal dla pinów A(2:0) oraz B(2:0) a także Input Terminal dla pinu Cin. Połącz je z symbolem sumatora.



Rys. 31. Piny wejściowe dołączone do 3-bitowego sumatora.

43. Kliknij FUB prawym przyciskiem myszy i wybierz Push z menu kontekstowego.

44. W oknie Create New Implementation kliknij na VHDL Source Code by wskazać że działanie tego komponentu zostanie zaimplementowane w języku VHDL.



Rys. 32. Wybór sposobu implementacji elementu FUB.

45. Uzupełnij kod dekodera wyświetlacza siedmiosegmentowego.

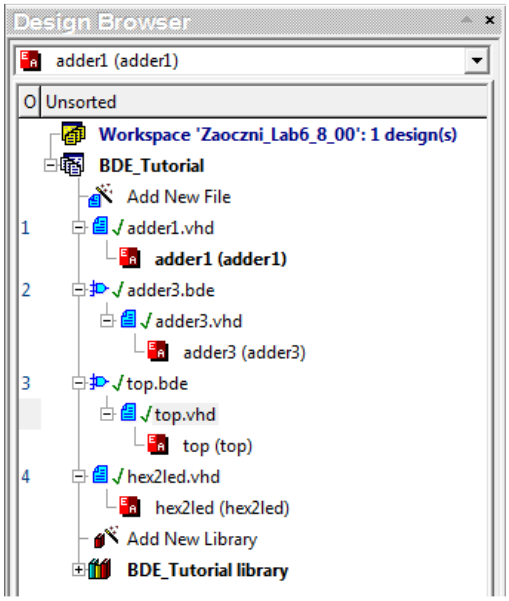
W tym celu:
- umieść kursor w linii 41 pliku hex2led.vhd;
- na głównym pasku menu wybierz Tools -> Language Assistant;
- w oknie Language Assistant rozwiń Synthesis templates i odszukaj HEX2LED Converter;
- kliknij prawym przyciskiem myszy na HEX2LED Converter i wybierz Use;
- zamknij Language Assistant.

Na Rysunku 33 przedstawiono zawartość pliku hex2led.vhd po wprowadzonych zmianach.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity hex2led is
     port(
          HEX : in STD_LOGIC_VECTOR(3 downto 0);
          LED : out STD_LOGIC_VECTOR(6 downto 0)
          );
end hex2led;

--}} End of automatically maintained section

architecture hex2led of hex2led is
begin

     -- enter your statements here --
        --HEX-to-seven-segment decoder
        --  HEX:    in  STD_LOGIC_VECTOR (3 downto 0);
        --  LED:    out STD_LOGIC_VECTOR (6 downto 0);
        --
        -- segment encoding
        --      0
        --     ---
        --  5 |   | 1
        --     ---   <- 6
        --  4 |   | 2
        --     ---
        --      3

        with HEX select
        LED<=   "1111001" when "0001",   --1
                "0100100" when "0010",   --2
                "0110000" when "0011",   --3
                "0011001" when "0100",   --4
                "0010010" when "0101",   --5
                "0000010" when "0110",   --6
                "1111000" when "0111",   --7
                "0000000" when "1000",   --8
                "0010000" when "1001",   --9
                "0001000" when "1010",   --A
                "0000011" when "1011",   --b
                "1000110" when "1100",   --C
                "0100001" when "1101",   --d
                "0000110" when "1110",   --E
                "0001110" when "1111",   --F
                "1000000" when others;   --0
end hex2led;
```

Rys. 33. Implementacja dekodera wyświetlacza siedmiosegmentowego w języku VHDL.

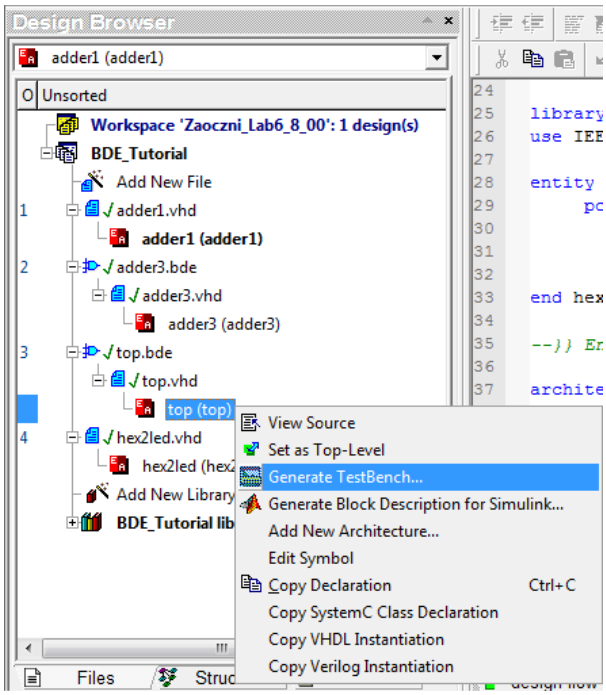
46. Skompiluj cały projekt. Wybierz z głównego menu programu Design -> Compile All.
    Jeśli kompilacja zakończy się poprawnie to w Design Browser będzie można zobaczyć następującą strukturę projektu.



Rys. 34. Zawartość okna Design Browser po kompilacji całego projektu.

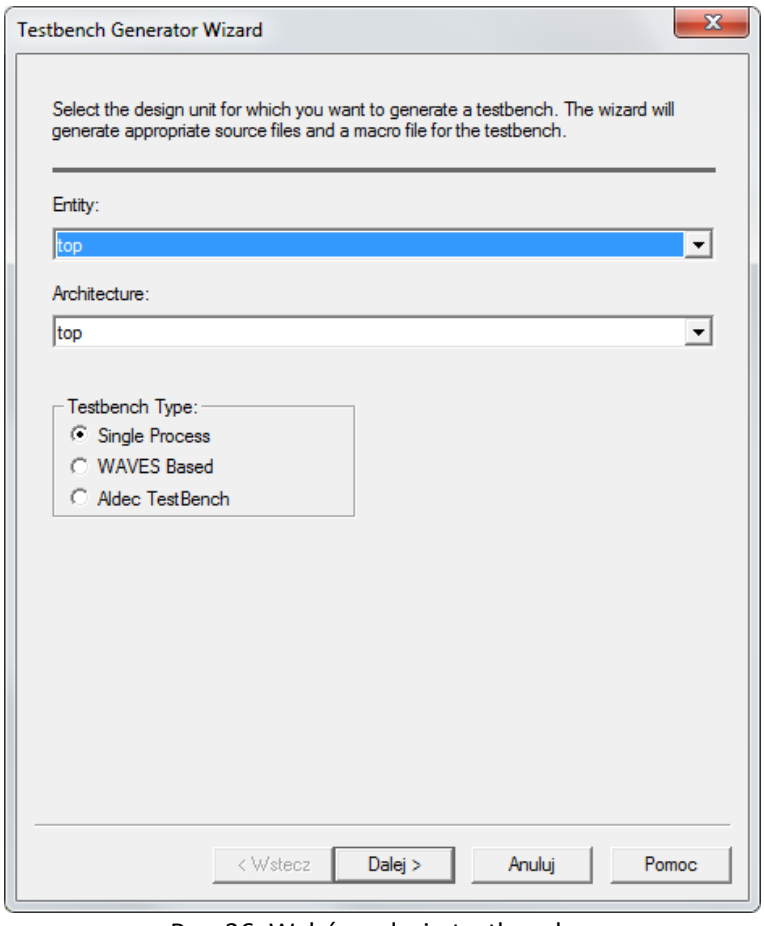
Generowanie Testbenchu

47. W Design Browser kliknij prawym przyciskiem myszy na parze top(top) i wybierz Generate TestBench z menu kontekstowego.



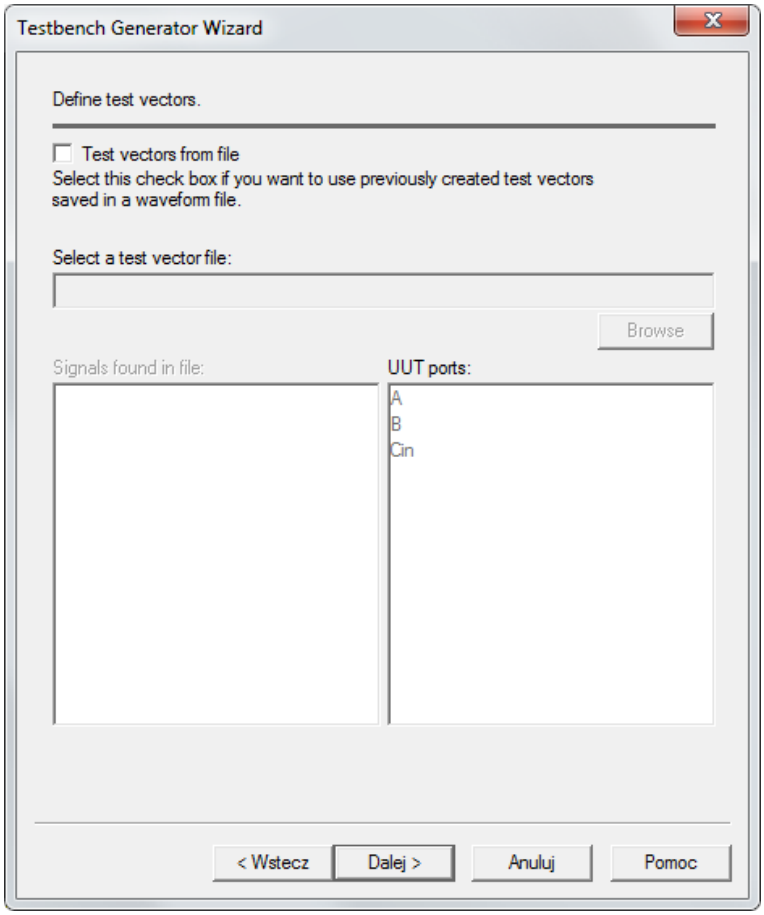
Rys. 35. Generowanie Testbench dla wskazanej pary entity(architecture).

48. W pierwszym oknie wybierz Single Process dla Testbench Type.



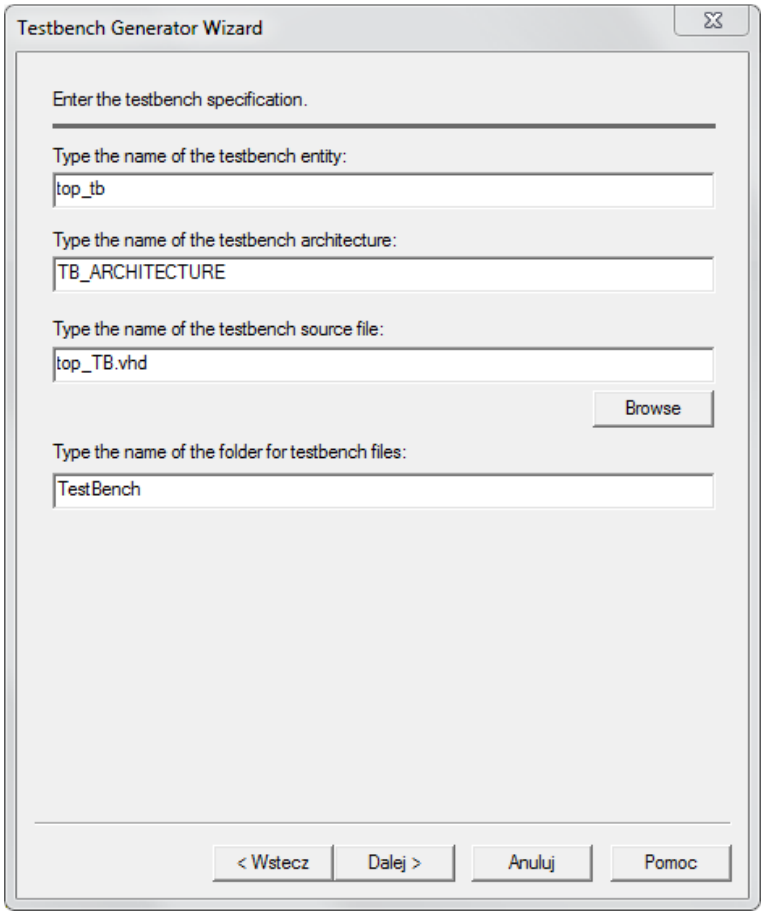
Rys. 36. Wybór rodzaju testbenchu.

49. Nie będziemy wczytywać wektorów testowych z żadnego pliku, więc w kolejnym oknie wybierz Next.



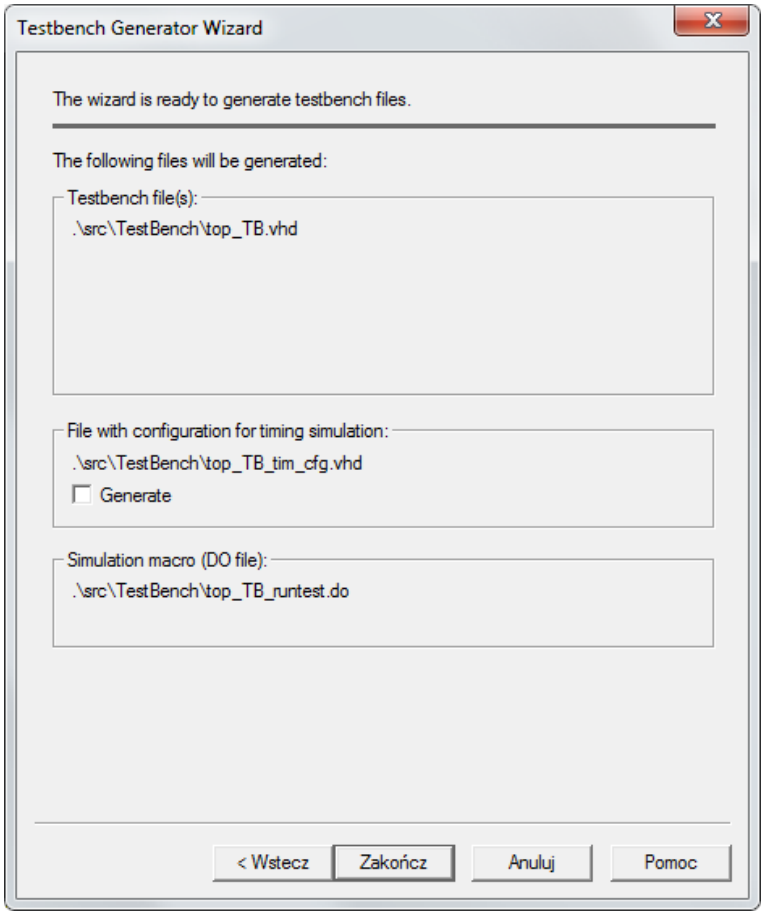
Rys. 37. Pomijamy import wektorów testowych.

50. W kolejnym oknie pozostaw niezmienione, domyślne nazwy dla entity i architecture. Kliknij Dalej.



Rys. 38. Okno w którym można wskazać m.in. nazwy entity i architecture dla generowanego testbenchu.

51. Nie wprowadzaj zmian w kolejnym oknie i zakończ generowanie testbenchu.



Rys. 39. Ostatnie okno narzędzia do generowania Testbenchu.

52. Dołóż wymuszenia w wygenerowanym Testbenchu.

W tym celu:
- wyświetl zawartość pliku top_TB.vhd (plik znajduje się w Design Browser, w folderze TestBench);
- po linii `--Add your stimulus here...` wstaw poniższy kod:

```
Cin <= '0';
A<="010", "111" after 200 ns, "011" after 400 ns;
B<="011", "001" after 200 ns, "001" after 400 ns;
```

-skompiluj plik testbencha.

```
30      -- Unit Under Test port map
31      UUT : top
32          port map (
33              Cin => Cin,
34              A => A,
35              B => B,
36              LED => LED
37          );
38
39      -- Add your stimulus here ...
40      Cin <= '0';
41      A<="010", "111" after 200 ns, "011" after 400 ns;
42      B<="011", "001" after 200 ns, "001" after 400 ns;
43
44  end TB_ARCHITECTURE;
45
46  configuration TESTBENCH_FOR_top of top_tb is
47      for TB_ARCHITECTURE
48          for UUT : top
49              use entity work.top(top);
50          end for;
```

Rys. 40. Edycja kodu VHDL w pliku top_TB.vhd.

53. W folderze TestBench w Design Browser, oprócz pliku VHDL z kodem testbencha znajduje się makro o nazwie top_TB_runtest.do, które służy do automatycznej kompilacji projektu i uruchamiania symulacji. Wyświetl zawartość makra i dołóż na końcu:

run 600 ns

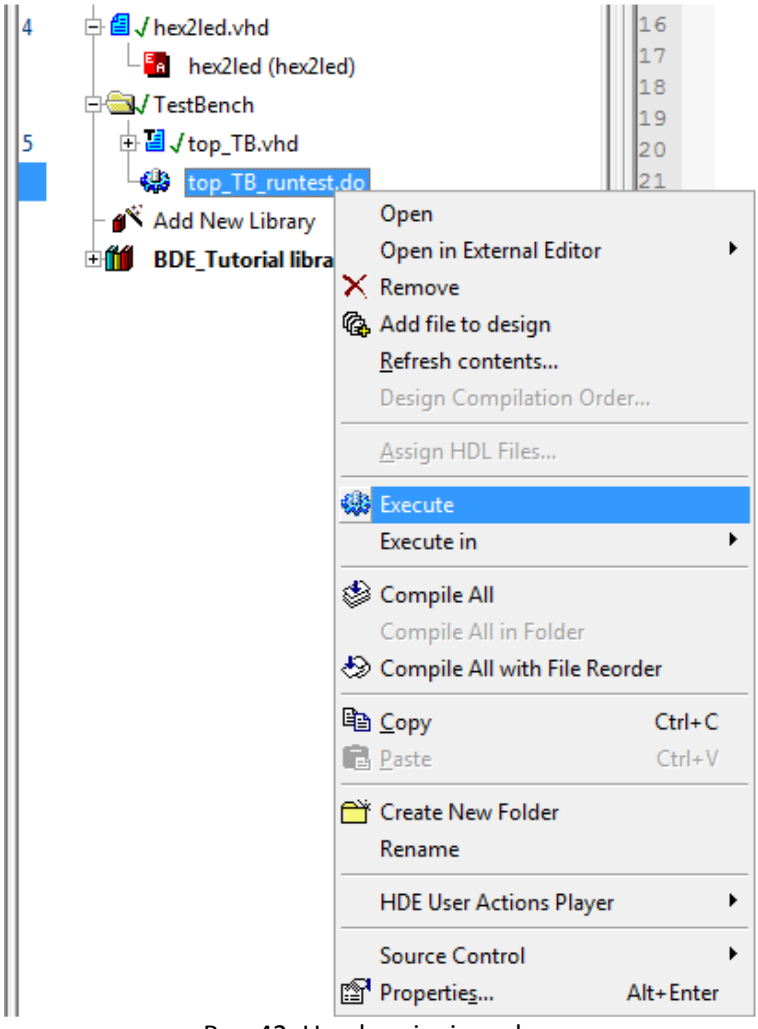
```
1   SetActiveLib -work
2   comp -include "$DSN\compile\top.vhd"
3   comp -include "$DSN\src\TestBench\top_TB.vhd"
4   asim TESTBENCH_FOR_top
5   wave
6   wave -noreg Cin
7   wave -noreg A
8   wave -noreg B
9   wave -noreg LED
10  # The following lines can be used for timing simulation
11  # acom <backannotated_vhdl_file_name>
12  # comp -include "$DSN\src\TestBench\top_TB_tim_cfg.vhd"
13  # asim TIMING_FOR_top
14
15  run 600 ns
```

Rys. 41. Edycja makra służącego do uruchomienia symulacji.

54. Zapisz zmiany wprowadzone w treści makra.

Symulacja Projektu

55. W Design Browser kliknij prawym przyciskiem myszy na top_TB_runtest.do i wybierz Execute z menu kontekstowego.

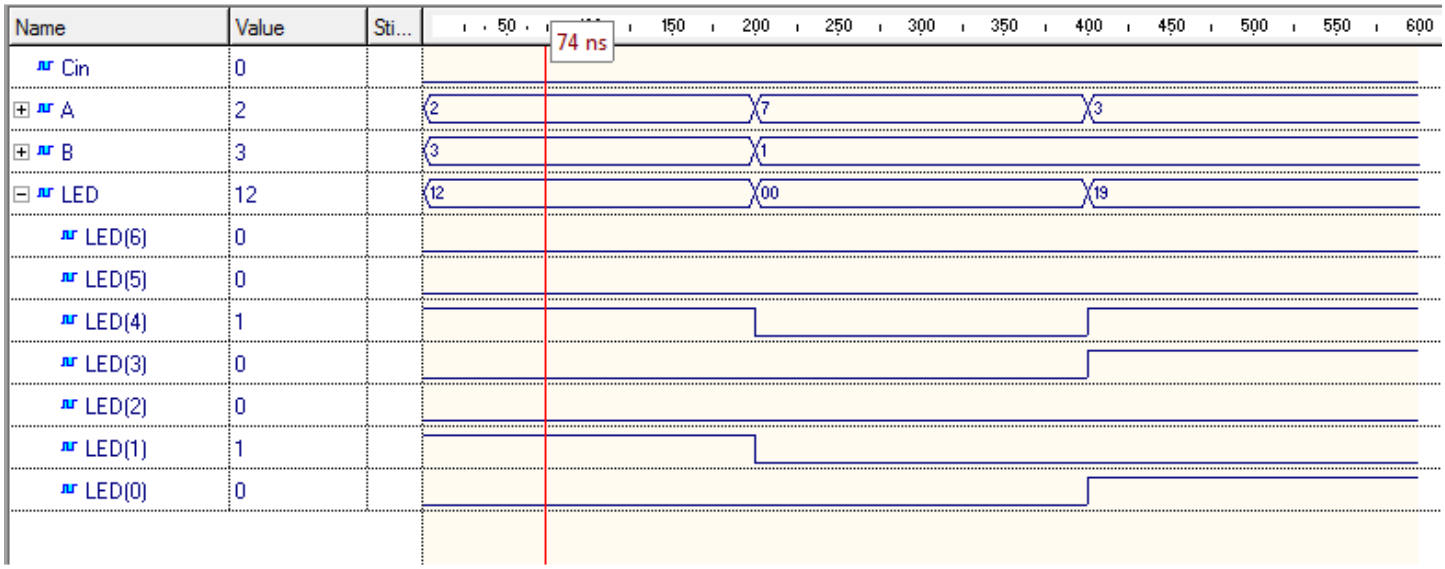


Rys. 42. Uruchomienie makra.

56. Zweryfikuj poprawne działanie układu.

Magistrala LED to sygnały sterujące wyświetlaczem siedmiosegmentowym (to nie jest wynik dodawania w kodzie naturalnym binarnym).

Segment wyświetlacza jest zapalany poziomem niskim (wynika to z rodzaju użytego wyświetlacza – wspólna Anoda).



Rys. 43. Wynik symulacji projektu.