

Programowanie układów FPGA – wykład VI

prof. nzw. dr hab. inż. Krzysztof Poźniak
Wydział Elektroniki i Technik Informacyjnych
Instytut Systemów Elektronicznych
e-mail: pozniak@ise.pw.edu.pl,
pok. 262 GE w kor. IIB, tel: (22) 234-7954
konsultacje: wtorek 14-16

- Podstawowe elementy standardu VHDL
 - Wybrane konstrukcje podprogramów
 - procedury
 - funkcje
 - Wybrane konstrukcje biblioteczne
 - Wybrane elementy biblioteki STD
 - Wybrane elementy biblioteki IEEE

Podstawowe elementy standardu VHDL

Podstawowa struktura opisu projektu dla FPGA



Podstawowe elementy standardu VHDL

Podstawowa struktura opisu projektu dla FPGA



Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ] is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [ procedure ] [ nazwa_proc ] ;
```

Podstawowa składnia deklaracji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ] ;
```

Deklaracja procedury **zapewnia jej udostępnienie**,
a jej definicja może być umieszczona **w innej części programu**

Deklaracja procedury **jest opcjonalna**, ale jej definicja **musi** być
umieszczona w sprzążonej jednostce leksykalnej programu

Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ] is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [ procedure ] [ nazwa_proc ] ;
```

Podstawowa składnia deklaracji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ] ;
```

- Wybrane rodzaje listy argumentów:

- stała: [constant] nazwa1 { , nazwa2 } : [in] typ
- zmienna: variable nazwa1 { , nazwa2 } : [in] | out | inout typ
- sygnał: signal nazwa1 { , nazwa2 } : [in] | out | inout typ

Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ] is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [ procedure ] [ nazwa_proc ] ;
```

Podstawowa składnia deklaracji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ] ;
```

- Wybrane składniki części deklaracyjnej:

- definicja typu/podtypu (podobnie jak w ciele jednostki projektowej)
- definicja stałej (podobnie jak w ciele jednostki projektowej)
- definicja zmiennej (podobnie jak w ciele procesu)
- definicja podprogramu (jest to podprogram zagnieżdzony)

Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ] is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [ procedure ] [ nazwa_proc ];
```

Instrukcje sekwencyjne

Podstawowa składnia deklaracji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ];
```

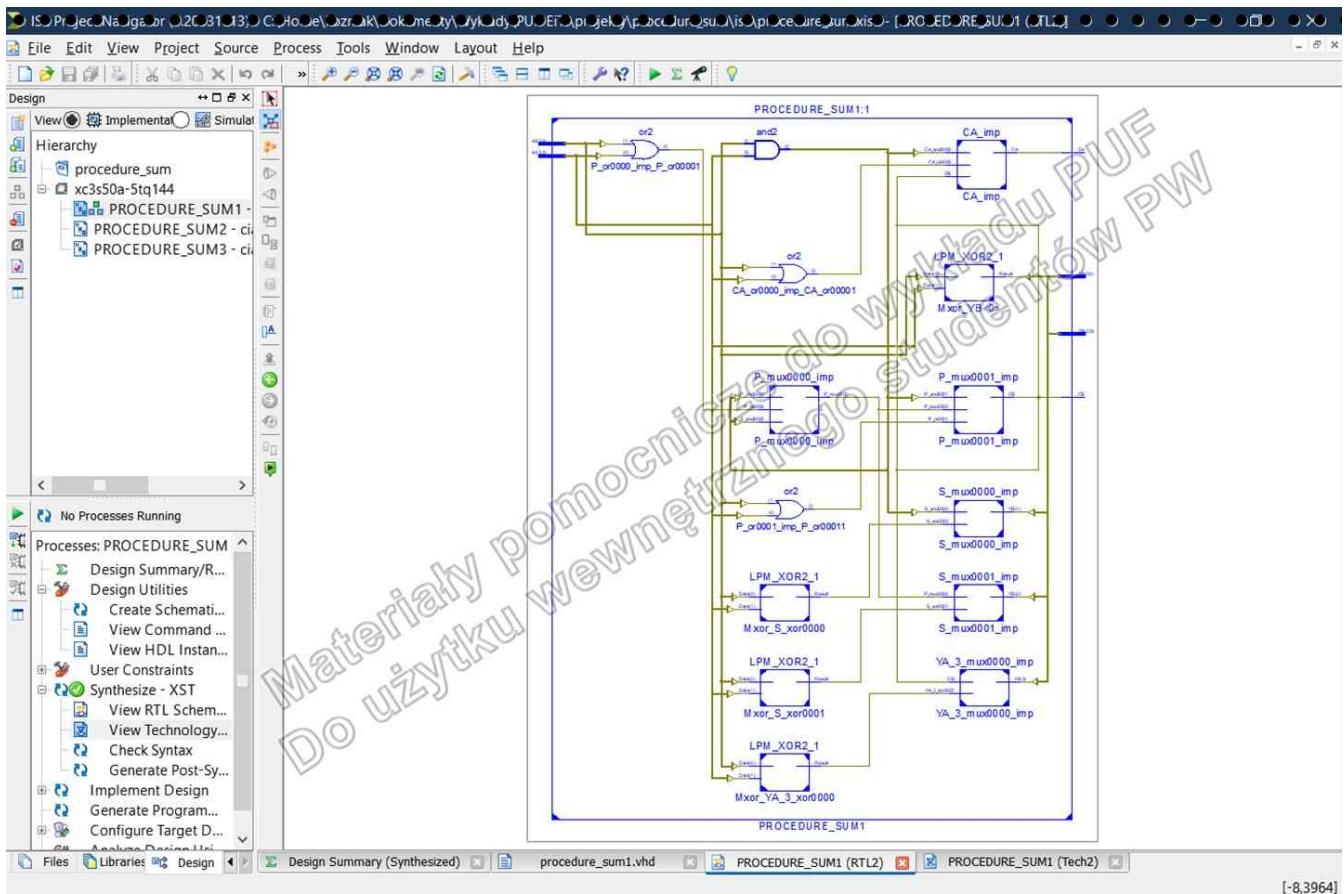
- Wybrane instrukcje części wykonawczej (podobnie jak dla procesu) :
 - instrukcja prostego przypisania sygnału/zmiennej
 - instrukcja warunkowego/selektywnego wyboru instrukcji
 - instrukcja pętli dyskretnej/warunkowej
 - wywołanie podprogramu



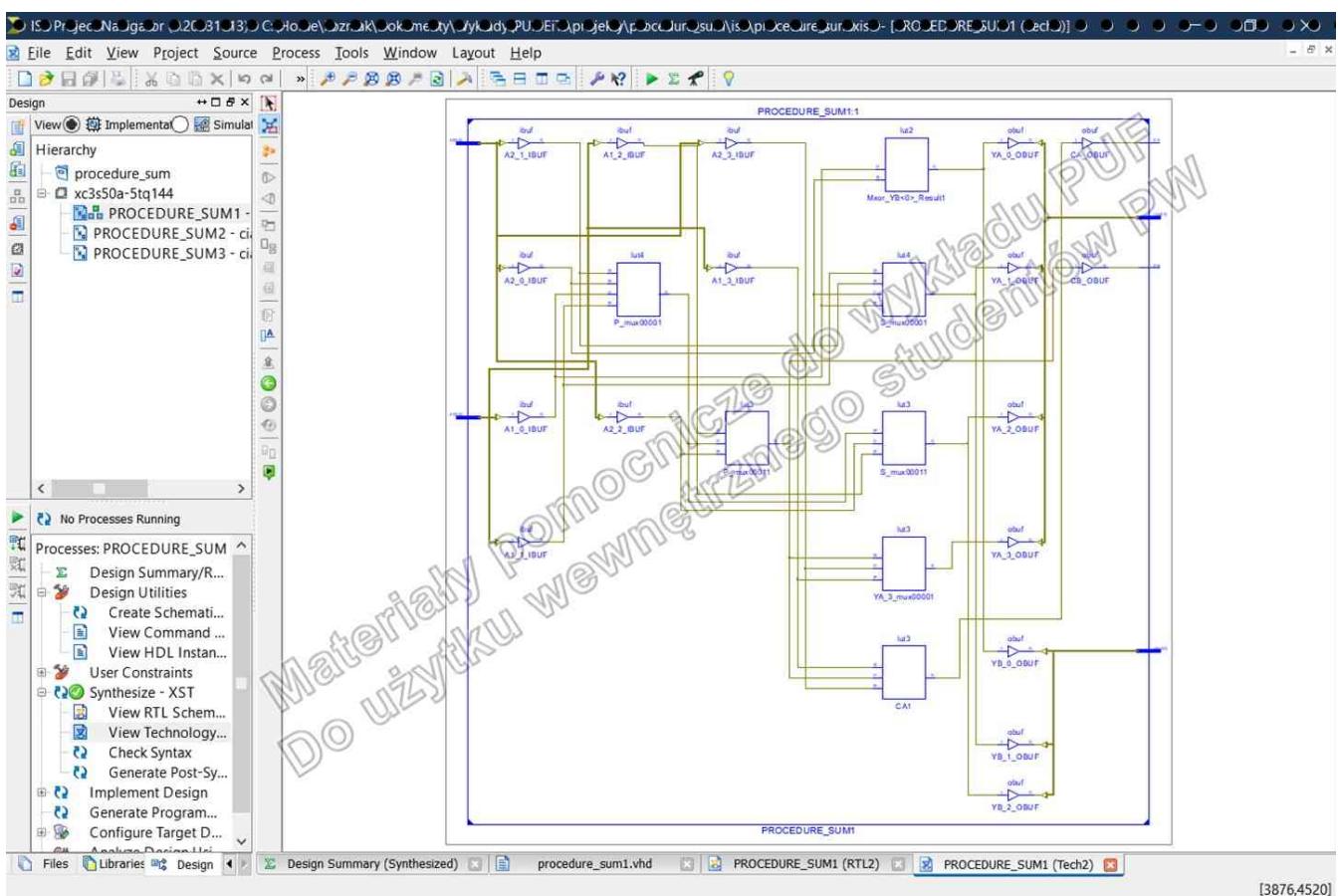
```
1 entity PROCEDURE_SUM1 is
2     port ( A1 : in bit_vector(3 downto 0);
3            A2 : in bit_vector(3 downto 0);
4            YA : out bit_vector(3 downto 0);
5            CA : out bit;
6            YB : out bit_vector(2 downto 0);
7            CB : out bit
8        );
9 end PROCEDURE_SUM1;
10
11 architecture cialo of PROCEDURE_SUM1 is
12
13     procedure sum3b(A, B : bit; S : out bit; P : inout bit) is
14     begin
15         if (P='0') then
16             S := A xor B;
17             P := A and B;
18         else
19             S := A xor B;
20             P := A or B;
21         end if;
22     end procedure sum3b;
23
24     procedure sumator(
25         signal A, B : in bit_vector;
26         signal Y : out bit_vector;
27         signal C : out bit)
28     is
29         variable P, S : bit;
30     begin
31         P := '0';
32         for i in 0 to Y'length loop
33             sum3b(A(i), B(i), S, P);
34             Y(i) <= S;
35         end loop;
36         C <= P;
37     end procedure sumator;
38
39 begin
40
41     sumator(A1, A2, YA, CA);
42     sumator(C=>CB, A=>A1(2 downto 0), Y=>YB, B=>A2(2 downto 0));
43
44 end architecture cialo;
```

Przykład użycia oprogramowania ISE – projekt: „procedure_sum”

Plik źródłowy „procedure_sum1.vhd”



Przykład użycia oprogramowania ISE – projekt: „procedure_sum”
Schemat RTL (dla pliku źródłowego „procedure_sum1.vhd”)



Przykład użycia oprogramowania ISE – projekt: „procedure_sum”
Schemat technologiczny (dla pliku źródłowego „procedure_sum1.vhd”)

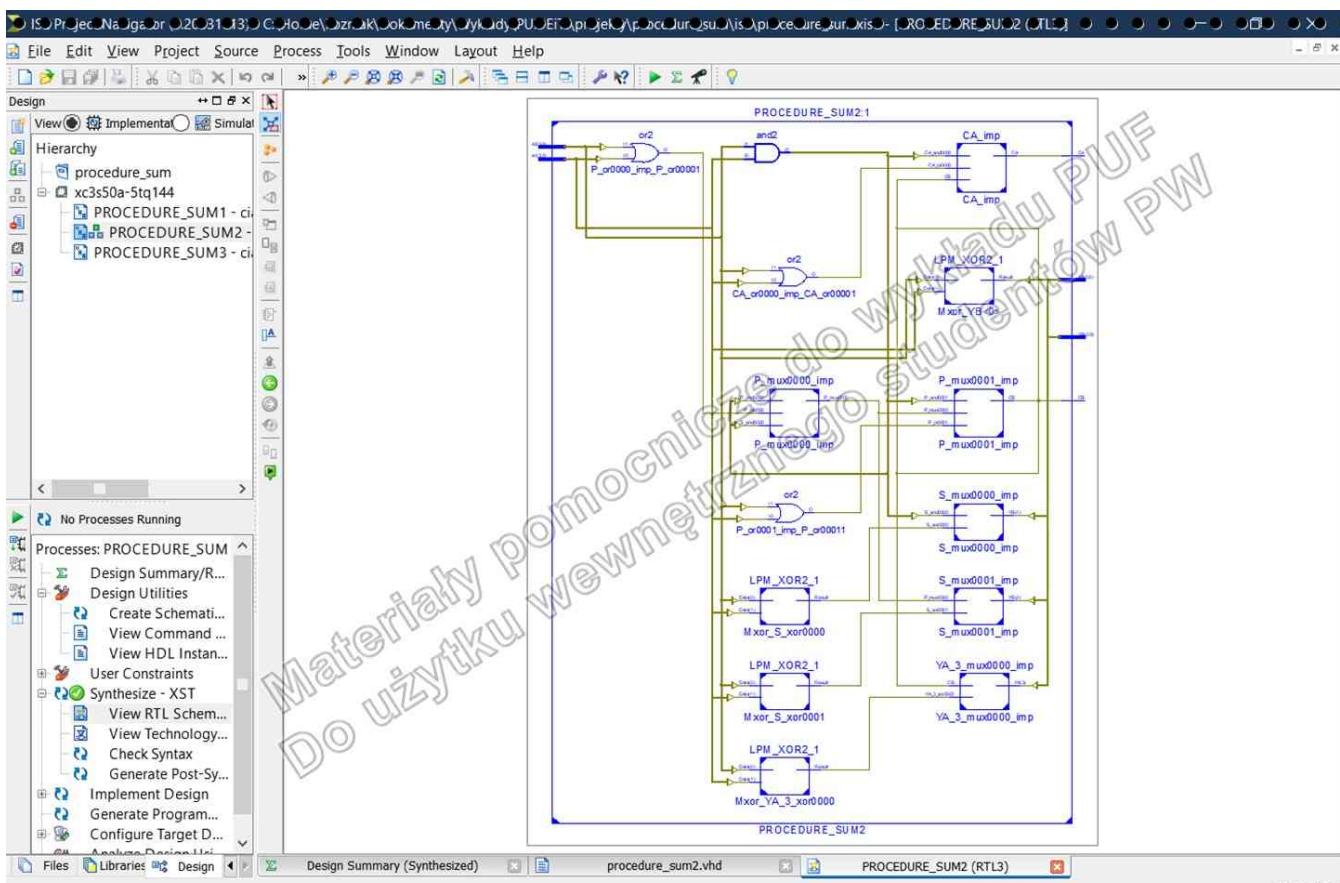
Materiały pomocnicze do wykładu PUF
Do użytku wewnętrznego

```

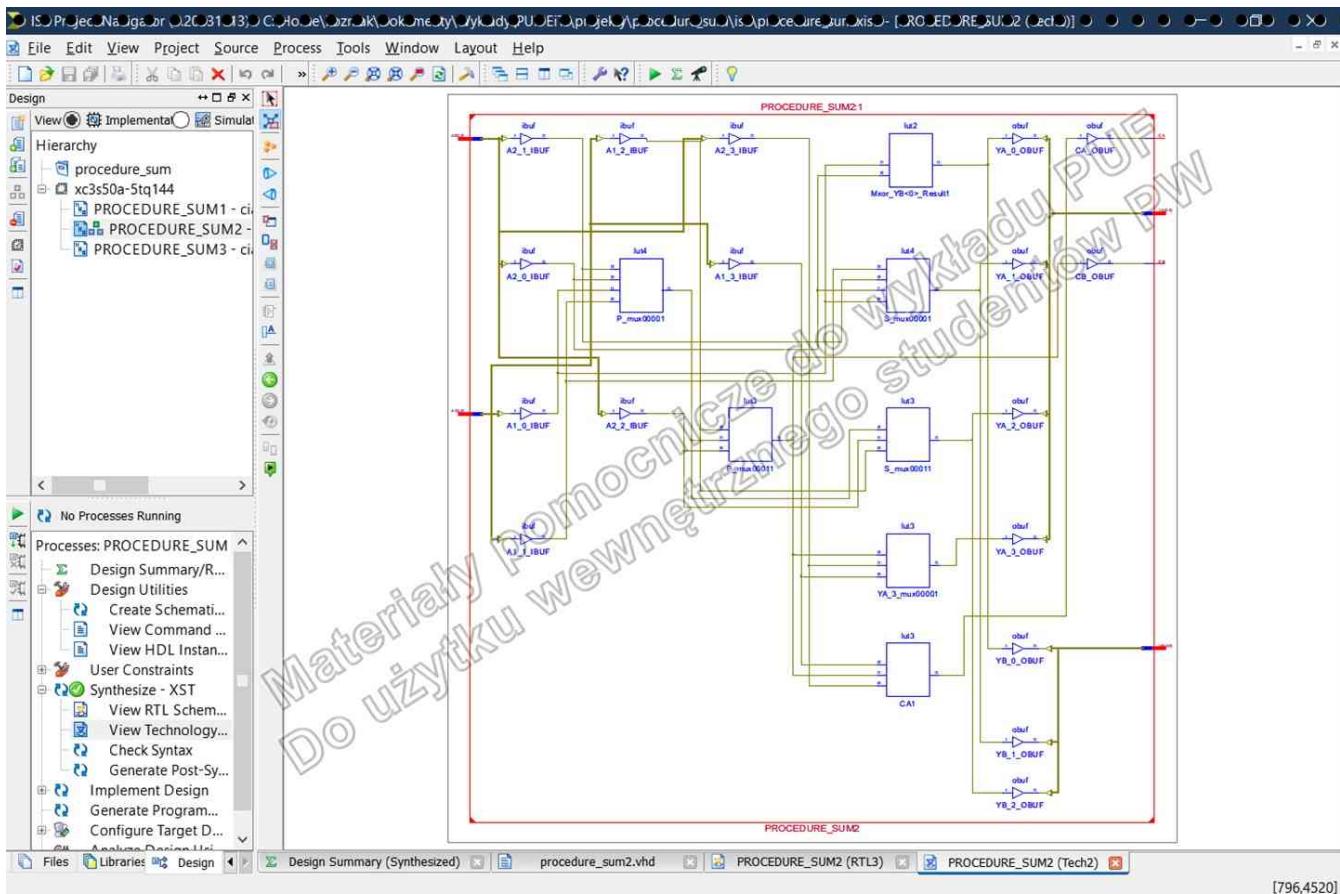
1 entity PROCEDURE_SUM2 is
2 port ( A1 : in bit_vector(3 downto 0);
3          A2 : in bit_vector(3 downto 0);
4          YA : out bit_vector(3 downto 0);
5          CA : out bit;
6          YB : out bit_vector(2 downto 0);
7          CB : out bit
8 );
9 end PROCEDURE_SUM2;
10
11 architecture cialo of PROCEDURE_SUM2 is
12
13 procedure sumator(
14    signal A, B : in bit_vector;
15    signal Y : out bit_vector;
16    signal C : out bit)
17 is
18 variable P, S : bit;
19
20 procedure sum3b(A, B : bit; S : out bit; P : inout bit) is
21 begin
22   if (P='0') then
23     S := A xor B;
24     P := A and B;
25   else
26     S := A xor B;
27     P := A or B;
28   end if;
29 end procedure sum3b;
30
31 begin
32   P := '0';
33   for i in 0 to Y'length-1 loop
34     sum3b(A(i),B(i),S,P);
35     Y(i) <= S;
36   end loop;
37   C <= P;
38 end procedure sumator;
39
40 begin
41   sumator(A1, A2, YA, CA);
42   sumator(C=>CB, A=>A1(2 downto 0), Y=>YB, B=>A2(2 downto 0));
43   sumator(C=>CB, A=>A1(1 downto 0), Y=>YA, B=>A2(1 downto 0));
44 end architecture cialo;
45

```

Przykład użycia oprogramowania ISE – projekt: „procedure_sum” Plik źródłowy „procedure_sum2.vhd”



Przykład użycia oprogramowania ISE – projekt: „procedure_sum” Schemat RTL (dla pliku źródłowego „procedure_sum2.vhd”)



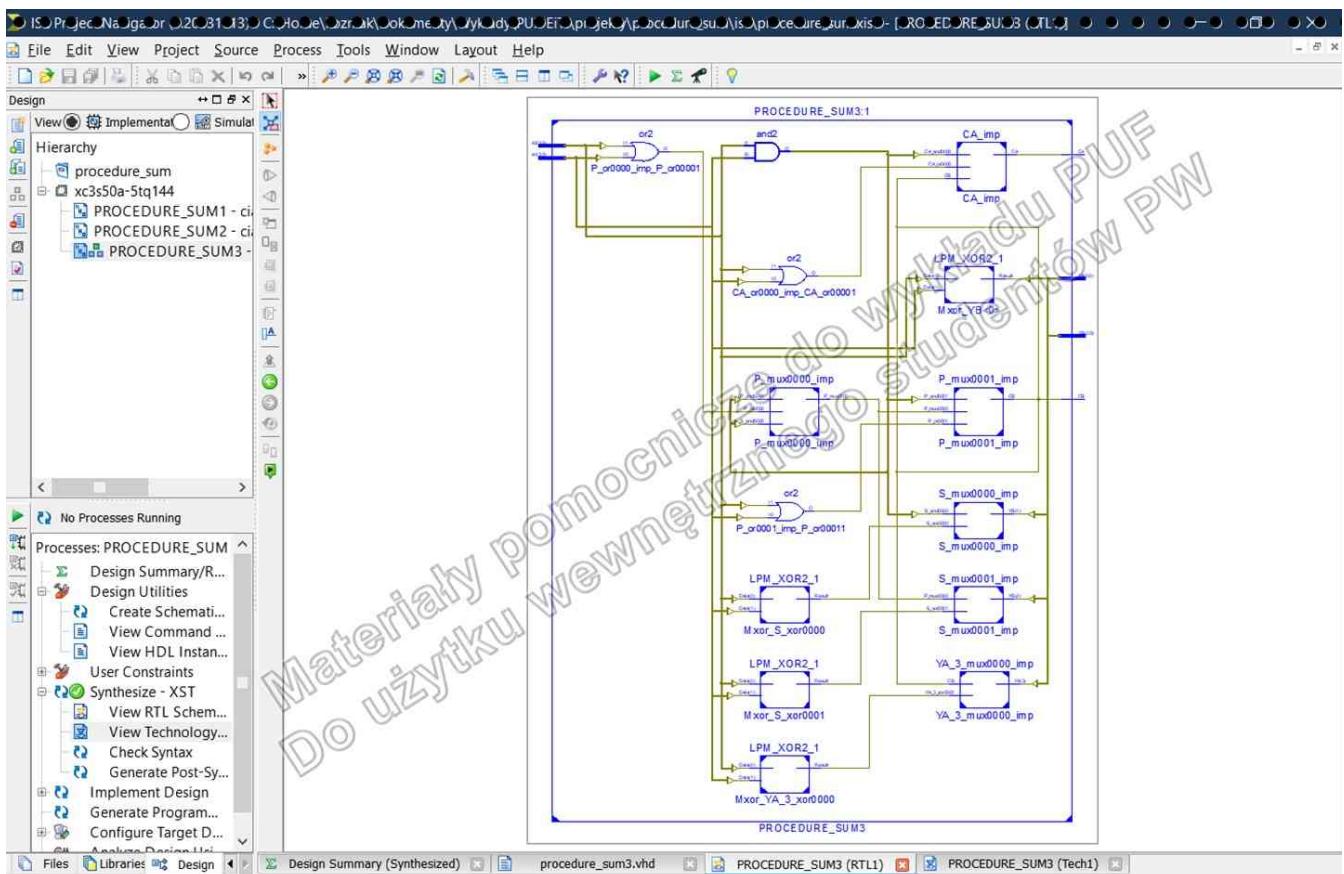
Przykład użycia oprogramowania ISE – projekt: „procedure_sum”
Schemat technologiczny (dla pliku źródłowego „procedure_sum2.vhd”)

```

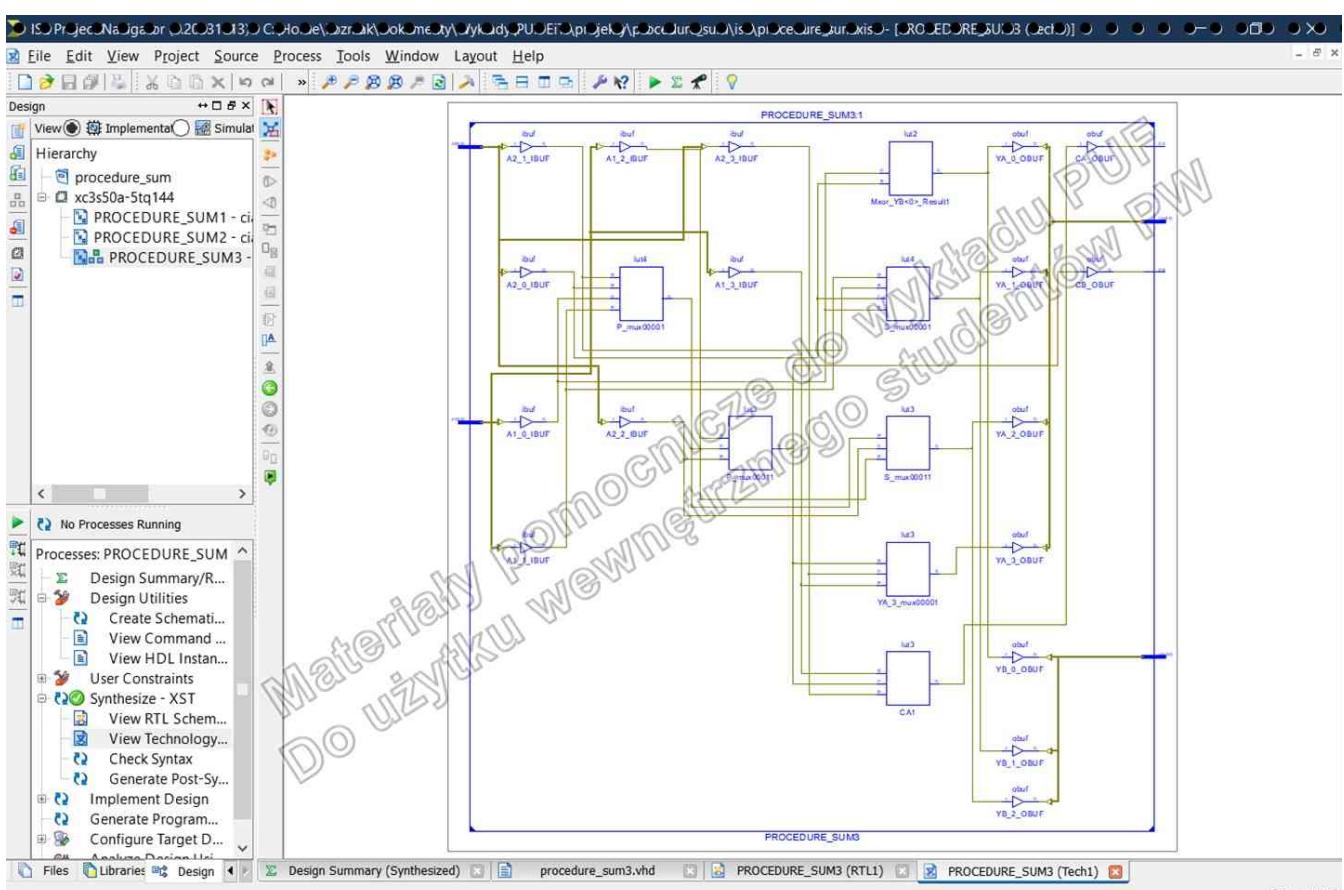
File Edit View Project Source Process Tools Window Layout Help
Design Implementa Simulat
Hierarchy
procedure_sum
  xc3s50a-5tg144
    PROCEDURE_SUM1 - ci
    PROCEDURE_SUM2 - ci
    PROCEDURE_SUM3 - ci
No Processes Running
Processes: PROCEDURE_SUM ^
  Design Summary/R...
  Design Utilities
    Create Schematic...
    View Command ...
    View HDL Instant...
  User Constraints
  Synthesize - XST
    View RTL Scheme...
    View Technology...
    Check Syntax
    Generate Post-Sy...
  Implement Design
  Generate Program...
  Configure Target D...
  Analysis Design...
Design Summary (Synthesized) procedure_sum2.vhd PROCEDURE_SUM2 (RTL3) PROCEDURE_SUM2 (Tech2)
[796,4520]
entity PROCEDURE_SUM3 is
  port ( A1 : in bit_vector(3 downto 0);
         A2 : in bit_vector(3 downto 0);
         YA : out bit_vector(3 downto 0);
         CA : out bit;
         YB : out bit_vector(2 downto 0);
         CB : out bit
       );
end PROCEDURE_SUM3;
architecture cialo of PROCEDURE_SUM3 is
  procedure sumator(
    signal A, B : in bit_vector;
    signal Y : out bit_vector;
    signal C : out bit
  );
  variable P, S : bit;
begin
  if (P='0') then
    S := A xor B;
    P := A and B;
  else
    S := A xor B;
    P := A or B;
  end if;
  end procedure sum3b;
  begin
    P := '0';
    for i in 0 to Y'length-1 loop
      sum3b(A(i),B(i),S,P);
      Y(i) <= S;
    end loop;
    C := P;
  end;
  procedure sumator;
  begin
    sumator(A1, A2, YA, CA);
  end procedure;
begin
  sum;
end process;
sumator(A1(2 downto 0), A2(2 downto 0), YB, CB);
end architecture cialo;

```

Przykład użycia oprogramowania ISE – projekt: „procedure_sum”
Plik źródłowy „procedure_sum3.vhd”



Przykład użycia oprogramowania ISE – projekt: „procedure_sum”
Schemat RTL (dla pliku źródłowego „procedure_sum3.vhd”)



Przykład użycia oprogramowania ISE – projekt: „procedure_sum”
Schemat technologiczny (dla pliku źródłowego „procedure_sum3.vhd”)

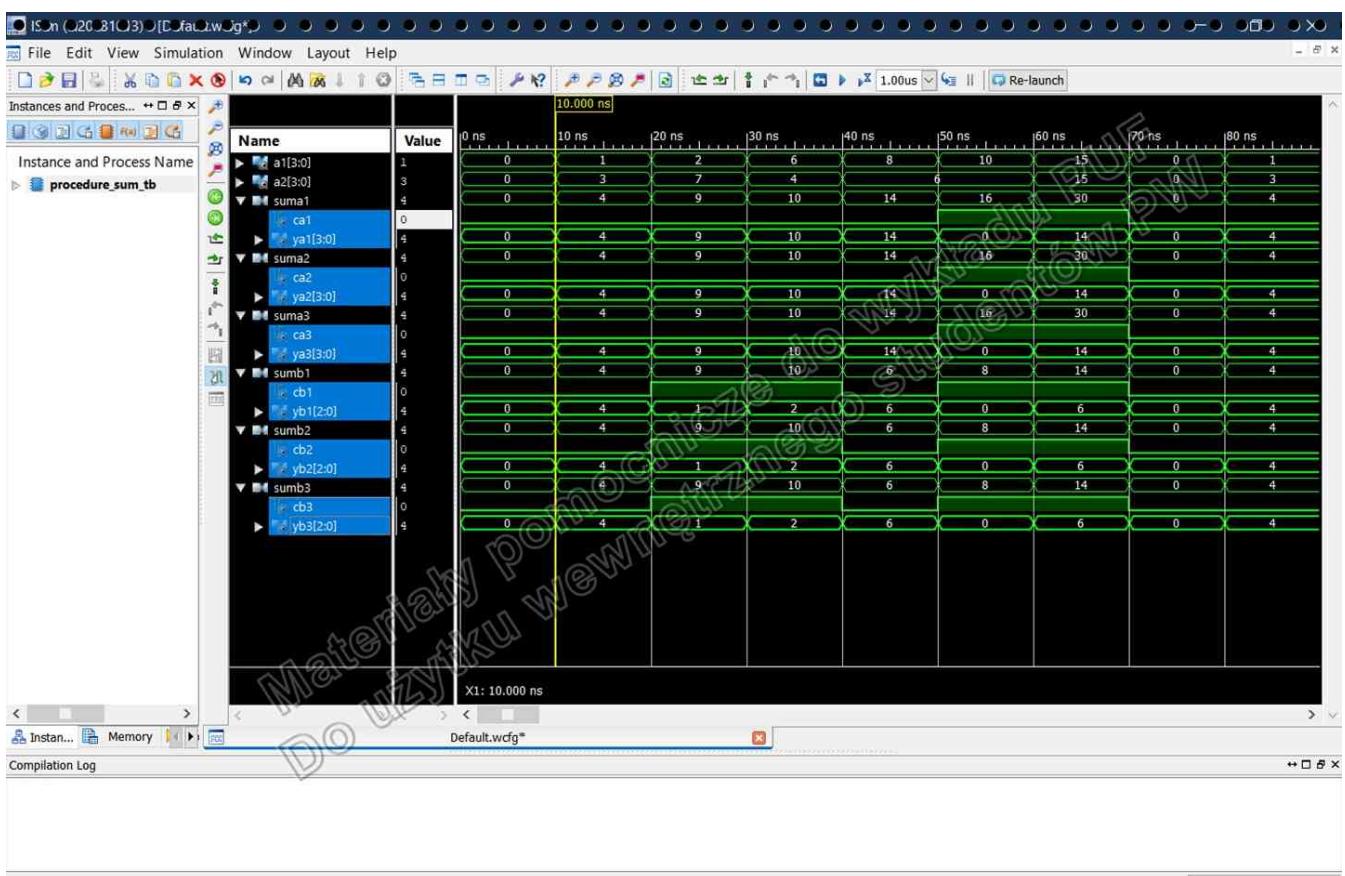
File Edit View Project Source Process Tools Window Layout Help

```

Design View Implementa Behavioral
Hierachy
procedure_sum
  xc3s50a-5tq144
    PROCEDURE_SUM_TB
      PROCEDURE_SUM_TB is
        entity PROCEDURE_SUM_TB is
          end PROCEDURE_SUM_TB;
        architecture behavioural of PROCEDURE_SUM_TB is
          signal A1 : bit_vector(3 downto 0);
          signal A2 : bit_vector(3 downto 0);
          signal YA1, YA2, YA3 : bit_vector(3 downto 0);
          signal YB1, YB2, YB3 : bit_vector(2 downto 0);
          signal CA1, CA2, CA3 : bit;
          signal CB1, CB2, CB3 : bit;
        begin
          process is
            begin
              A1 <= "0000"; A2 <= "0000"; wait for 10ns;
              A1 <= "0001"; A2 <= "0011"; wait for 10ns;
              A1 <= "0010"; A2 <= "0111"; wait for 10ns;
              A1 <= "0110"; A2 <= "0100"; wait for 10ns;
              A1 <= "1000"; A2 <= "0010"; wait for 10ns;
              A1 <= "1010"; A2 <= "0110"; wait for 10ns;
              A1 <= "1111"; A2 <= "1111"; wait for 10ns;
            end process;
          procedure sum1 inst: entity work.PROCEDURE_SUM1(cialo) -- instancja projektu 'PROCEDURE_SUM1'
            port map(A1 => A1, A2 => A2, YA => YA1, CA => CA1, YB => YB1, CB => CB1); -- przypisanie portom sy
          procedure sum2_inst: entity work.PROCEDURE_SUM2(cialo) -- instancja projektu 'PROCEDURE_SUM2'
            port map(A1 => A1, A2 => A2, YA => YA2, CA => CA2, YB => YB2, CB => CB2); -- przypisanie portom sy
          procedure sum3_inst: entity work.PROCEDURE_SUM3(cialo) -- instancja projektu 'PROCEDURE_SUM3'
            port map(A1 => A1, A2 => A2, YA => YA3, CA => CA3, YB => YB3, CB => CB3); -- przypisanie portom sy
        end behavioural;
      end PROCEDURE_SUM_TB;
    end xc3s50a-5tq144;
  end procedure_sum;

```

Przykład użycia oprogramowania ISE – projekt: „procedure_sum” Plik źródłowy „procedure_sum_TB.vhd”



Przykład użycia oprogramowania ISim – projekt: „procedure_sum” Symulacja funkcjonalna

```

1  entity PROCEDURE_OPTREG2 is
2    generic ( ASYN : boolean := FALSE);
3    port ( R : in bit;
4           C : in bit;
5           E : in bit;
6           A : in bit_vector(3 downto 0);
7           Y : out natural range 0 to 2
8         );
9  end PROCEDURE_OPTREG2;
10
11 architecture cialo of PROCEDURE_OPTREG2 is
12 begin
13
14   process (R, C, A) is
15     procedure exec is
16       variable L : natural range 0 to 2;
17     begin
18       L := 0;
19       for P in 0 to A'length-1 loop
20         next when A(P)'=0';
21         L := L + 1;
22         exit when L=2;
23       end loop;
24       Y <= L;
25     end procedure;
26     begin
27       if (not(ASYN)) then
28         exec;
29       elsif (R='1') then
30         Y <= 0;
31       elsif (C'e'vent and C='1') then
32         if (E='1') then
33           exec;
34         end if;
35       end if;
36     end process;
37
38 end architecture cialo;

```

-- deklaracja sprzegu 'PROCEDURE_OPTREG2'
-- deklaracja i ustawienie parametru 'ASYN'
-- deklaracja portu kasującego 'R'
-- deklaracja portu taktującego 'C'
-- deklaracja portu zezwalającego 'E'
-- deklaracja portu wejściowego 'A'
-- deklaracja portu wyjściowego 'Y'
-- zakończenie deklaracji listy portów
-- zakończenie deklaracji nagłówka
-- deklaracja ciała 'cialo' architektury
-- początek części wykonawczej
-- lista czynności procesu
-- utworzenie procedury 'exec'
-- utworzenie zmiennej 'L'
-- część wykonawcza procedury
-- ustawienie wartości początkowej 'L' na 0
-- pętla po kolejnych bitach identyfikatora 'P'
-- warunkowy powrót do pętli
-- zwiększenie o 2^P licznika 'L'
-- warunkowe opuszczenie pętli
-- zakończenie pętli
-- przypisanie stanu 'L' do sygnału 'Y'
-- zakończenie procedury
-- część wykonawcza procesu
-- warunek dla 'ASYN=False'
-- wywołanie procedury 'exec'
-- warunek dla sygnału 'R'
-- przypisanie stałej do wyjścia
-- warunek zbożca narastającego 'C'
-- warunek zezwolenia na zapis
-- wywołanie procedury 'exec'
-- zakończenie instrukcji wyboru
-- zakończenie instrukcji wyboru
-- zakończenie procesu
-- zakończenie deklaracji ciała 'cialo'

Przykład użycia oprogramowania ISE – projekt: „procedure_optreg” Plik źródłowy „procedure_optreg2.vhd” (fragment 1)

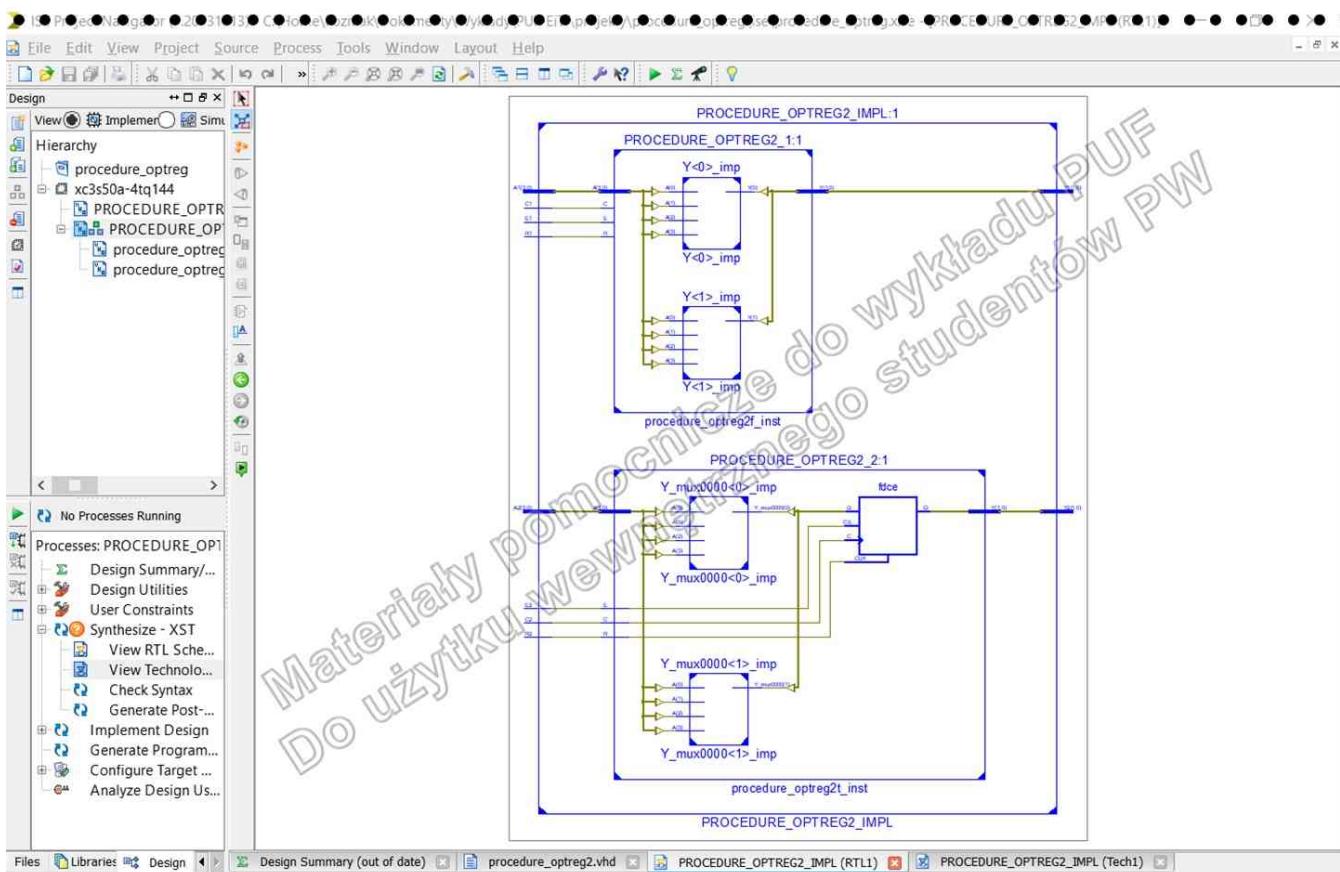
```

24   Y <= L;
25 end procedure;
26 begin
27   if (not(ASYN)) then
28     exec;
29   elsif (R='1') then
30     Y <= 0;
31   elsif (C'e'vent and C='1') then
32     if (E='1') then
33       exec;
34     end if;
35   end if;
36 end process;
37
38 end architecture cialo;
39
40 entity PROCEDURE_OPTREG2_IMPL is
41   port ( R1, R2 : in bit;
42          C1, C2 : in bit;
43          E1, E2 : in bit;
44          A1, A2 : in bit_vector(3 downto 0);
45          Y1, Y2 : out natural range 0 to 2
46        );
47 end PROCEDURE_OPTREG2_IMPL;
48
49 architecture cialo of PROCEDURE_OPTREG2_IMPL is
50 begin
51
52   procedure_optreg2_inst: entity work.PROCEDURE_OPTREG2(cialo) -- instancja projektu 'PROCEDURE_OPTREG2'
53   generic map (ASYN => FALSE) -- mapowanie generic
54   port map (R=>R1, C=>C1, E=>E1, A=>A1, Y=>Y1); -- mapowanie portów
55
56   procedure_optreg2t_inst: entity work.PROCEDURE_OPTREG2(cialo) -- instancja projektu 'PROCEDURE_OPTREG2'
57   generic map (ASYN => TRUE) -- mapowanie generic
58   port map (R=>R2, C=>C2, E=>E2, A=>A2, Y=>Y2); -- mapowanie portów
59
60 end architecture cialo;
61

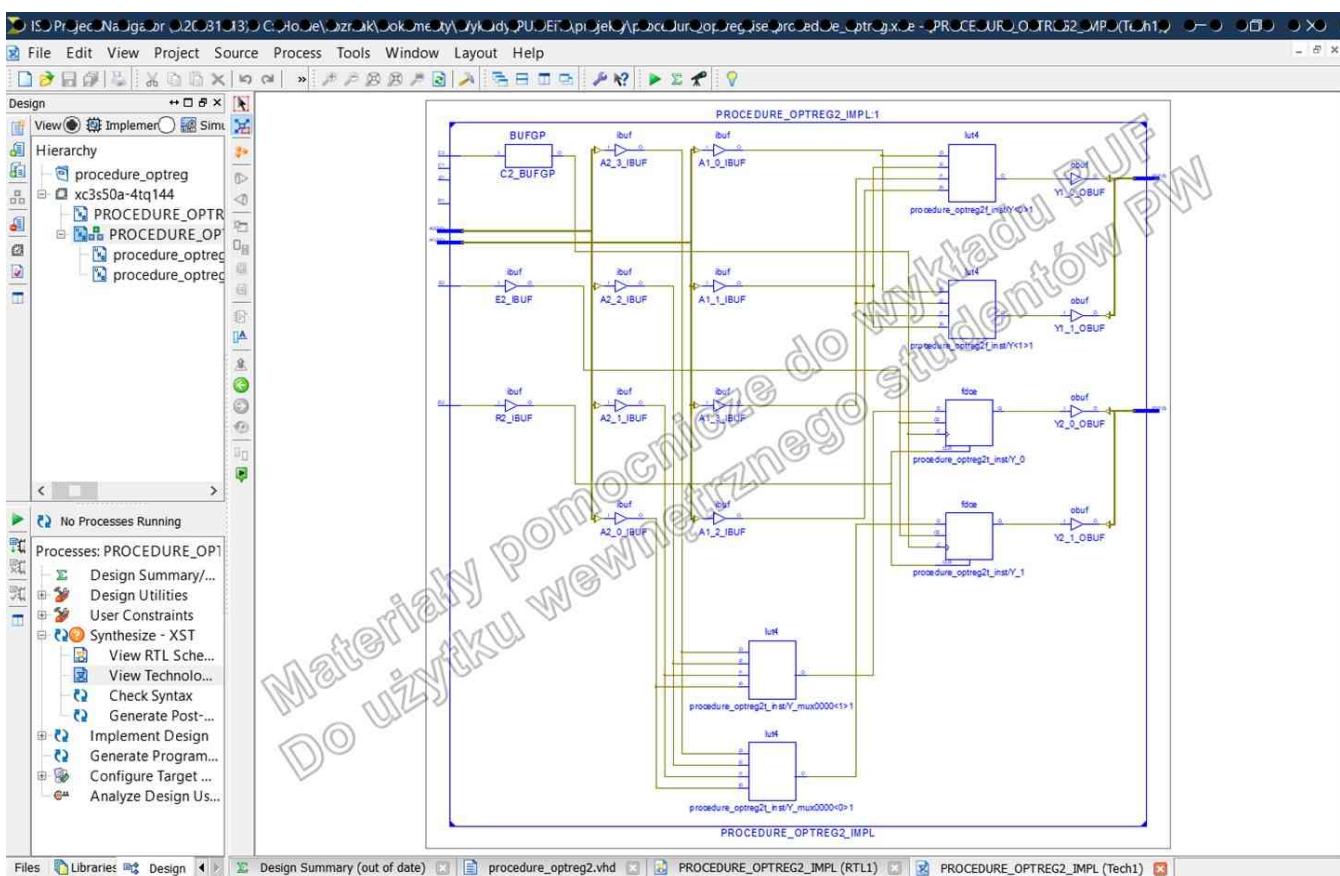
```

-- przypisanie stanu 'L' do sygnału 'Y'
-- zakończenie procedury
-- część wykonawcza procesu
-- warunek dla 'ASYN=False'
-- wywołanie procedury 'exec'
-- warunek dla sygnału 'R'
-- przypisanie stałej do wyjścia
-- warunek zbożca narastającego 'C'
-- warunek zezwolenia na zapis
-- wywołanie procedury 'exec'
-- zakończenie instrukcji wyboru
-- zakończenie instrukcji wyboru
-- zakończenie procesu
-- zakończenie deklaracji ciała 'cialo'
-- deklaracja sprzegu 'PROCEDURE_OPTREG2_IMPL'
-- deklaracja portów kasujących 'R1' i 'R2'
-- deklaracja portów taktujących 'C1' i 'C2'
-- deklaracja portów zezwalających 'E1' i 'E2'
-- deklaracja portów wejściowych 'A1' i 'A2'
-- deklaracja portów wyjściowych 'Y1' i 'Y2'
-- zakończenie deklaracji listy portów
-- zakończenie deklaracji nagłówka
-- deklaracja ciała 'cialo' architektury
-- początek części wykonawczej
-- procedura_optreg2t_inst: entity work.PROCEDURE_OPTREG2(cialo) -- instancja projektu 'PROCEDURE_OPTREG2'
-- generic map (ASYN => TRUE) -- mapowanie generic
-- port map (R=>R2, C=>C2, E=>E2, A=>A2, Y=>Y2); -- mapowanie portów
-- zakończenie deklaracji ciała 'cialo'

Przykład użycia oprogramowania ISE – projekt: „procedure_optreg” Plik źródłowy „procedure_optreg2.vhd” (fragment 2)



Przykład użycia oprogramowania ISE – projekt: „procedure_optreg”
Schemat RTL (dla entity „procedure_optreg2Impl”)



Przykład użycia oprogramowania ISE – projekt: „procedure_optreg”
Schemat technologiczny (dla entity „procedure_optreg2Impl”)

```

1 entity PROCEDURE_OPTREG_TB is -- pusty szkielet projektu symulacji
2 end PROCEDURE_OPTREG_TB;
3
4 architecture behavioural of PROCEDURE_OPTREG_TB is -- cialo architektoniczne projektu
5
6 signal R : bit; -- symulowane wejście kasujące 'R'
7 signal C : bit; -- symulowane wejście taktujące ''
8 signal E : bit; -- symulowane wejście zezwalające 'E'
9 signal A : bit_vector(3 downto 0); -- symulowane wejście 'A'
10 signal Y1, Y2 : natural range 0 to 2; -- obserwowane wyjścia 'Y1' i 'Y2'
11
12 begin -- początek części wykonawczej architektury
13
14 process is -- proces bezwarunkowy
15 begin
16   R <= '1'; wait for 20 ns;
17   R <= '0'; wait for 200 ns;
18 end process;
19
20 process is -- proces bezwarunkowy
21 begin
22   C <= '0'; wait for 5 ns;
23   C <= '1'; wait for 5 ns;
24 end process;
25
26 process is -- proces bezwarunkowy
27 begin
28   E <= '0'; wait for 10 ns;
29   E <= '1'; wait for 20 ns;
30 end process;
31
32 process is -- proces bezwarunkowy
33 begin
34   A <= "0000"; wait for 10 ns;
35   A <= "0001"; wait for 10 ns;
36   A <= "0010"; wait for 10 ns;
37   A <= "0011"; wait for 10 ns;
38   A <= "0100"; wait for 10 ns;
39   A <= "0101"; wait for 10 ns;
40
41   A <= "0000"; wait for 10 ns;
42   A <= "0001"; wait for 10 ns;
43   A <= "0010"; wait for 10 ns;
44   A <= "0011"; wait for 10 ns;
45   A <= "0100"; wait for 10 ns;
46   A <= "0101"; wait for 10 ns;
47   A <= "1000"; wait for 10 ns;
48   A <= "1001"; wait for 10 ns;
49   A <= "1010"; wait for 10 ns;
50
51
52 procedure_optreg_inst: entity work.PROCEDURE_OPTREG2_IMPL(cialo) -- instancja projektu 'PROCEDURE_OPTREG2_IMPL'
53 port map (R, R, C, C, E, E, A, A, Y1, Y2); -- mapowanie portow
54
55 end behavioural; -- zakończenie ciała architektonicznego

```

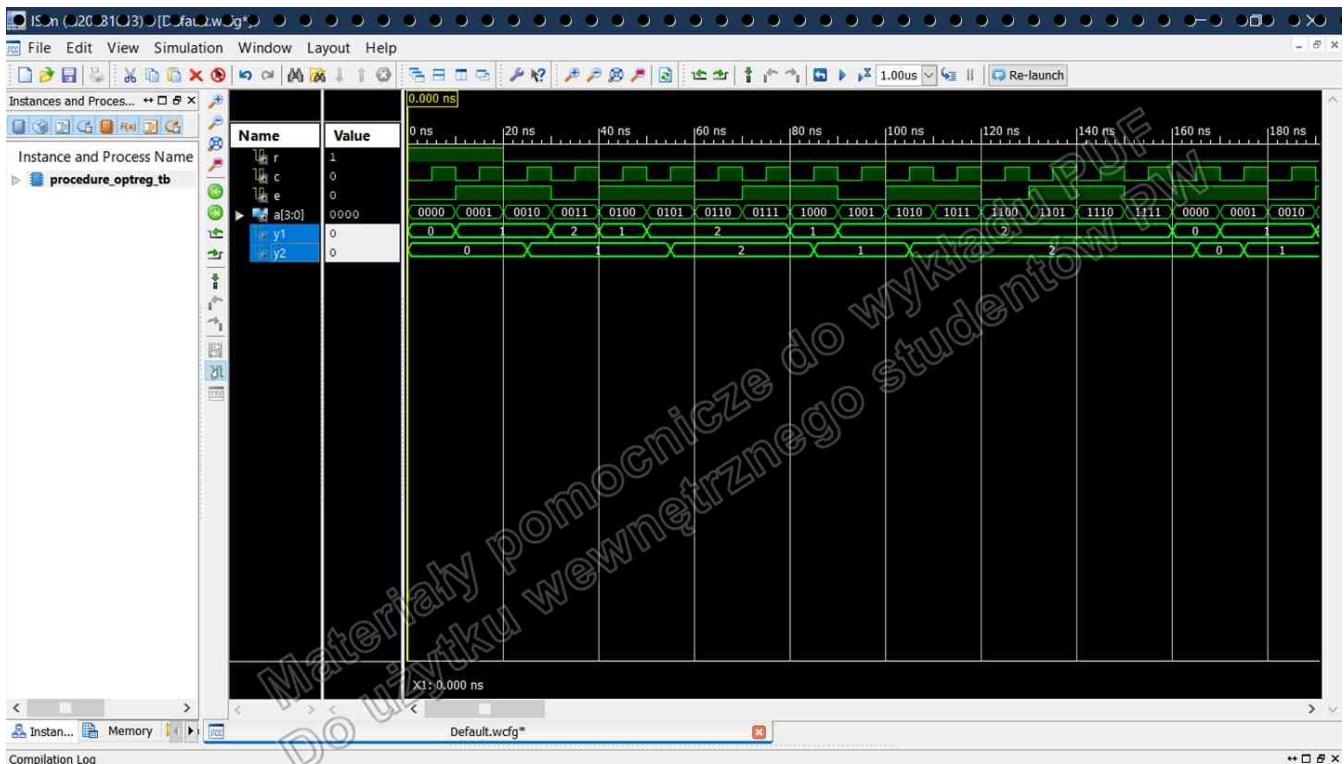
Przykład użycia oprogramowania ISE – projekt: „procedure_optreg” Plik źródłowy „procedure_optreg_TB.vhd” (fragment 1)

```

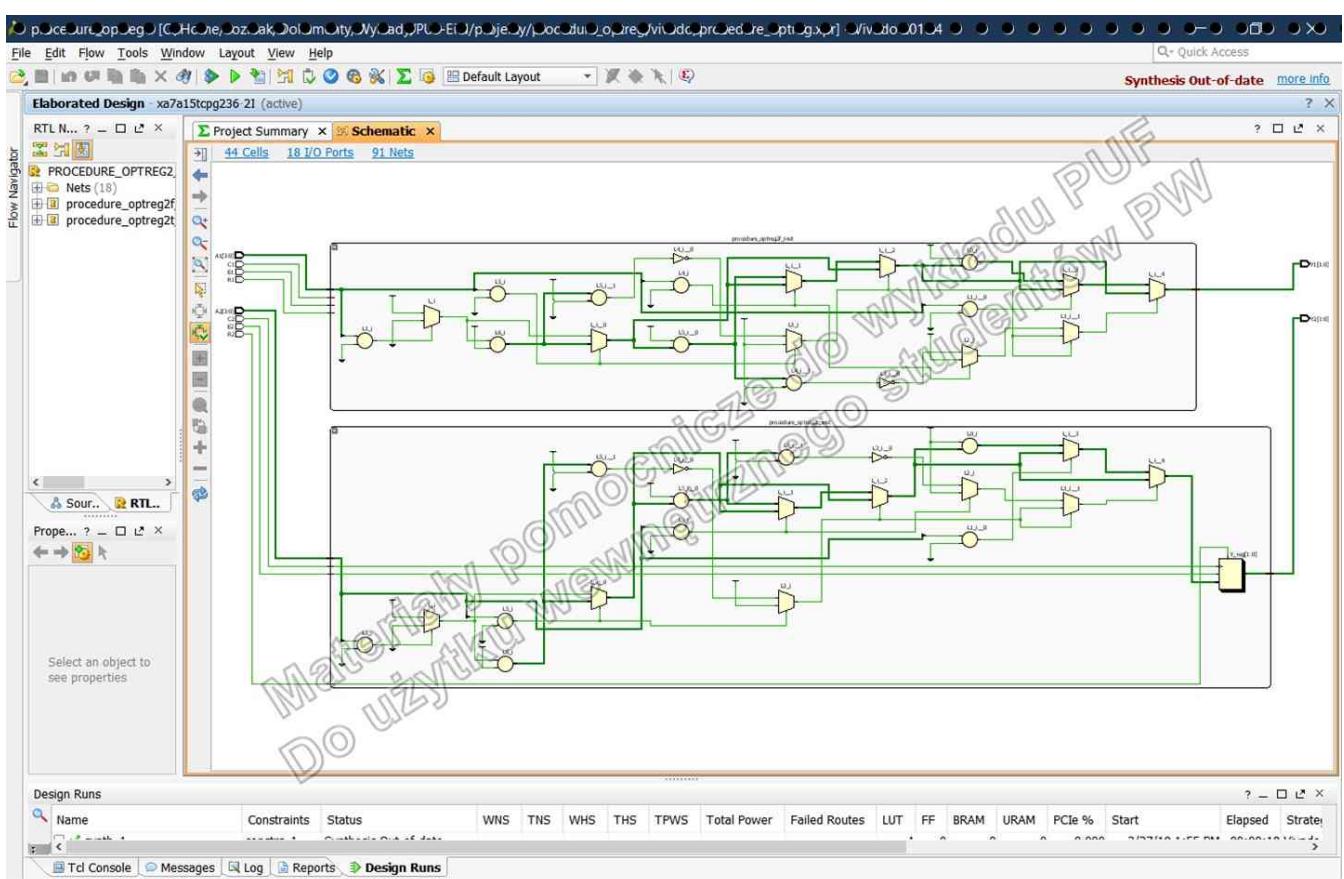
19
20 process is -- proces bezwarunkowy
21 begin
22   C <= '0'; wait for 5 ns;
23   C <= '1'; wait for 5 ns;
24 end process;
25
26 process is -- proces bezwarunkowy
27 begin
28   E <= '0'; wait for 10 ns;
29   E <= '1'; wait for 20 ns;
30 end process;
31
32 process is -- proces bezwarunkowy
33 begin
34   A <= "0000"; wait for 10 ns;
35   A <= "0001"; wait for 10 ns;
36   A <= "0010"; wait for 10 ns;
37   A <= "0011"; wait for 10 ns;
38   A <= "0100"; wait for 10 ns;
39   A <= "0101"; wait for 10 ns;
40   A <= "0110"; wait for 10 ns;
41   A <= "0111"; wait for 10 ns;
42   A <= "1000"; wait for 10 ns;
43   A <= "1001"; wait for 10 ns;
44   A <= "1011"; wait for 10 ns;
45   A <= "1100"; wait for 10 ns;
46   A <= "1101"; wait for 10 ns;
47   A <= "1110"; wait for 10 ns;
48   A <= "1111"; wait for 10 ns;
49
50 end process;
51
52 procedure_optreg_inst: entity work.PROCEDURE_OPTREG2_IMPL(cialo) -- instancja projektu 'PROCEDURE_OPTREG2_IMPL'
53 port map (R, R, C, C, E, E, A, A, Y1, Y2); -- mapowanie portow
54
55 end behavioural; -- zakończenie ciała architektonicznego

```

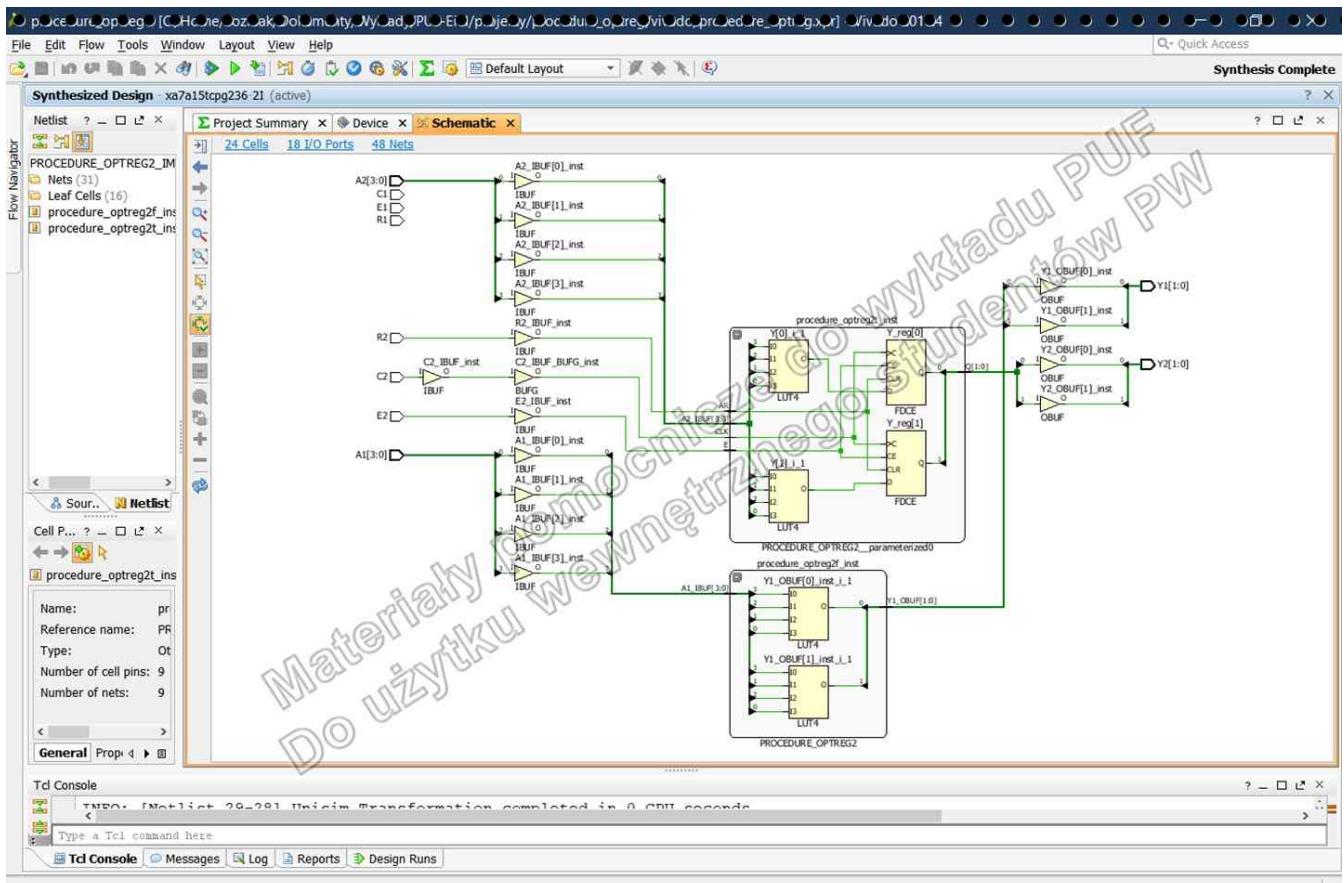
Przykład użycia oprogramowania ISE – projekt: „procedure_optreg” Plik źródłowy „procedure_optreg_TB.vhd” (fragment 2)



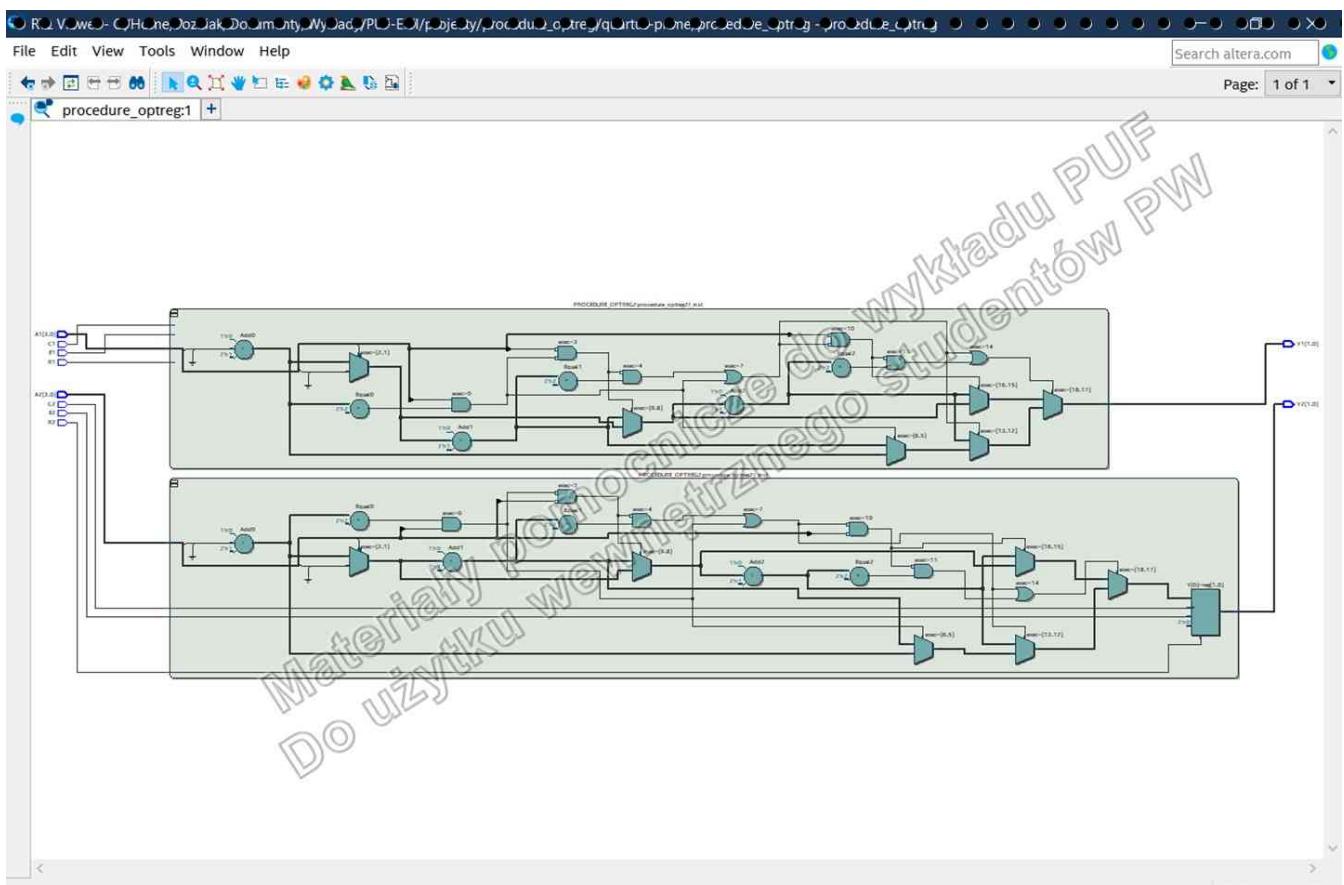
Przykład użycia oprogramowania ISim – projekt: „procedure_optreg” Symulacja funkcjonalna



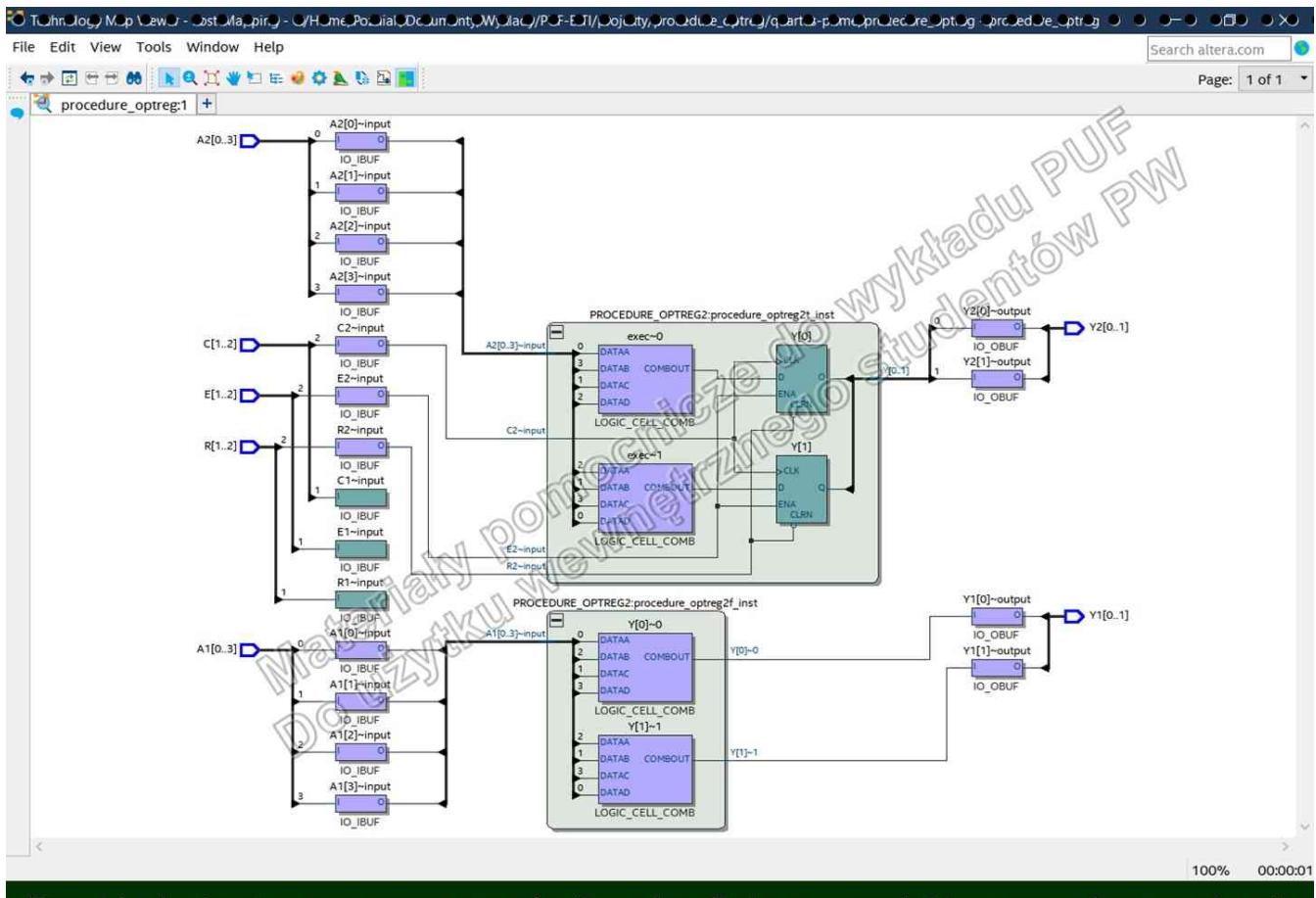
Przykład użycia oprogramowania Vivado – projekt: „procedure_optreg” Schemat RTL (dla entity „procedure_optreg2_impl”)



Przykład użycia oprogramowania Vivado – projekt: „procedure_optreg”
Schemat technologiczny (dla entity „procedure_optreg2Impl”)



Przykład użycia oprogramowania Quartus Prime – projekt: „procedure_optreg”
Schemat RTL (dla entity „procedure_optreg2Impl”)



Przykład użycia oprogramowania Quartus Prime – projekt: „procedure_optreg”
 Schemat technologiczny (dla entity „procedure_optreg2Impl”)

Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ] is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [ procedure ] [ nazwa_proc ];
```

Podstawowa składnia deklaracji procedury:

```
procedure nazwa_proc [ ( lista_arg1 { ; lista_arg2 } ) ];
```

Procedura **może być przeciążona**, czyli tworzyć **rodzinę** procedur o **wspólnej nazwie** procedury, ale o **różnej liczbie argumentów** oraz o **różnych typach** dla tych samych argumentów



```

1 entity PROCEDURE_OVERSUM is
2   port ( A1 : in bit_vector(3 downto 0);
3         A2 : in bit_vector(3 downto 0);
4         A3 : in bit_vector(3 downto 0);
5         YA : out bit_vector(3 downto 0);
6         CA : out bit;
7         YB : out bit_vector(4 downto 0);
8         CB : out bit
9       );
10 end PROCEDURE_OVERSUM;
11
12 architecture cialo of PROCEDURE_OVERSUM is
13 | procedure sum(
14   | constant A, B : in bit_vector;
15   | variable Y : out bit_vector;
16   | variable C : out bit)
17 is
18   variable P, S : bit;
19
20   procedure sum(A, B : bit; S : out bit; P : inout bit) is -- utworzenie procedury 'sum'
21   begin
22     if (P='0') then -- czesc wykonawcza procedury
23       S := A xor B; -- badanie czy bit przeniesienia jest rowny '0'
24       P := A and B; -- sumowanie bitow funkcja 'xor'
25     else -- wyznaczenie bitu przeniesienia funkcja 'and'
26       S := A xor B; -- wariant dla 'P' rownego '1'
27       P := A or B; -- sumowanie bitow funkcja 'xnor'
28     end if; -- wyznaczenie bitu przeniesienia funkcja 'or'
29   end procedure sum;
30
31 begin
32   P := '0'; -- zakonczenie instukcji warunkowej
33   for i in 0 to Y'length-1 loop -- czesc wykonawcza procedury
34     sum(A(i),B(i),S,P); -- ustawienie wartosci poczatkowej 'P' na '0'
35     Y(i) := S; -- wywolanie procedury wewnętrznej 'sum'
36   end loop; -- przypisanie zmiennej 'S' do portu 'Y(i)'
37   C := P; -- zakonczenie petli
38   end procedure sum; -- przypisanie zmiennej 'P' do portu 'C'
39
40 end procedure sum; -- zakonczenie procedury
41
42 procedure sumator(
43   signal A1, A2 : in bit_vector;
44   signal Y : out bit_vector;
45   signal C : out bit)
46 is
47   variable S : bit_vector(Y'range);
48   variable P : bit;
49 begin
50   sum(A1,A2,S,P); -- czesc deklaracyjna procedury
51   Y <= S; -- utworzenie zmiennej 'S' w procedurze
52   C <= P; -- utworzenie zmiennej 'P' w procedurze
53 end procedure sumator;
54
55 procedure sumator(
56   signal A1, A2, A3 : in bit_vector;
57   signal Y : out bit_vector;
58   signal C : out bit)
59 is
60   variable S1, S2, S3 : bit_vector(Y'range);
61   variable P : bit;
62 begin
63   sum(A1,A2,S1(A1'range),S1(A1'length)); -- czesc wykonawcza procedury
64   S3 := '0'&A3; -- wywolanie procedury 'sum'
65   sum(S3,S1,S2,P); -- przypisanie zmiennej 'S3'
66   -- sum("0"&A3,S1,S2,P) -- wywolanie przeciazonej procedury 'sumator'
67   Y <= S2; -- ??? wywolanie przeciazonej procedury 'sumator'
68   C <= P; -- przypisanie zmiennej 'S' do portu 'Y'
69   and procedure sumator; -- przypisanie zmiennej 'P' do portu 'C'
70
71   sumator(A1, A2, YA, CA); -- zakonczenie procedury
72   sumator(A1, A2, A3, YB, CB); -- poczatek czesci wykonawczej
73
74 end architecture cialo; -- wywolanie przeciazonej procedury 'sumator'
75 -- wywolanie przeciazonej procedury 'sumator'
76
77 end PROCEDURE_OVERSUM; -- zakończenie deklaracji ciala 'cialo'

```

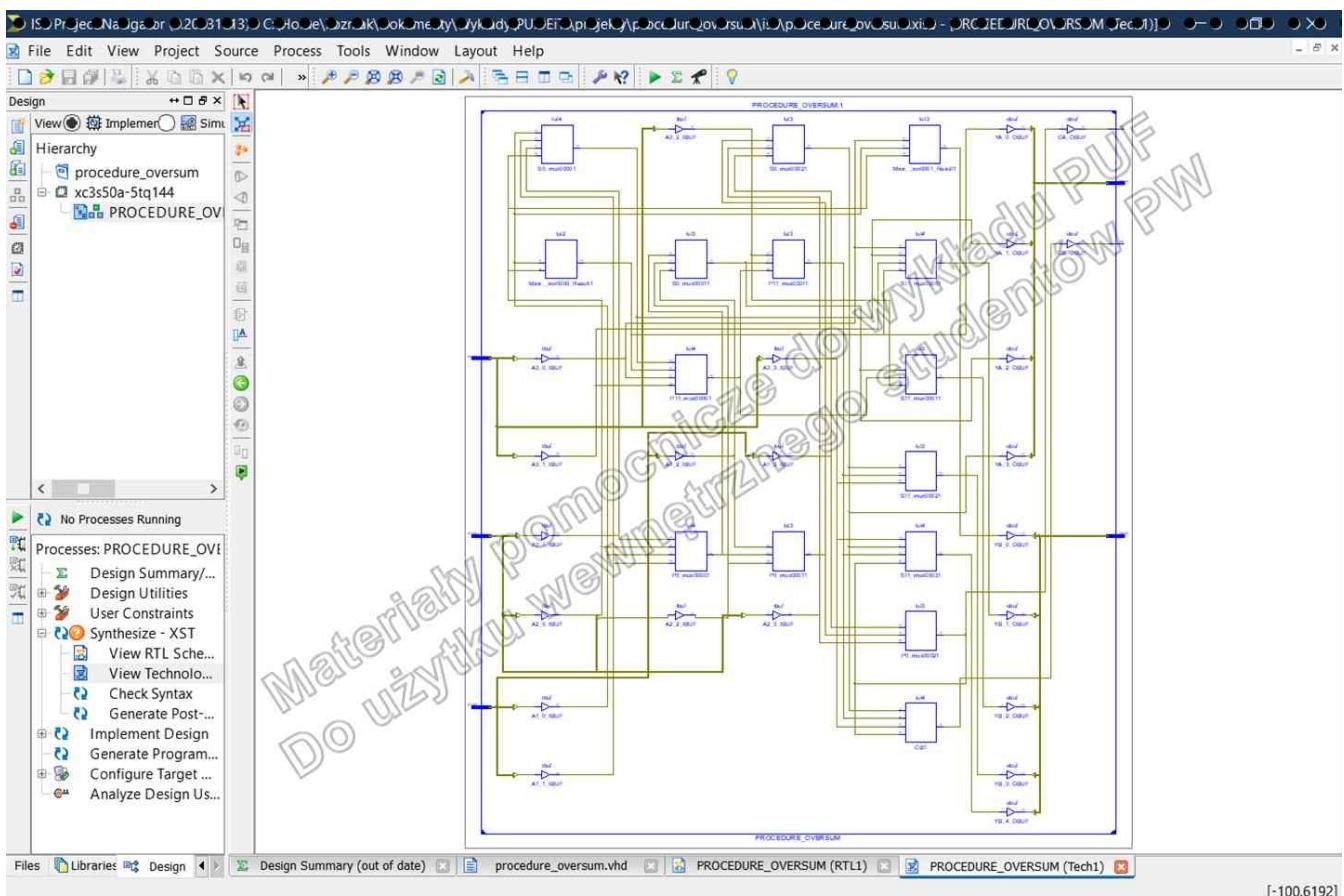
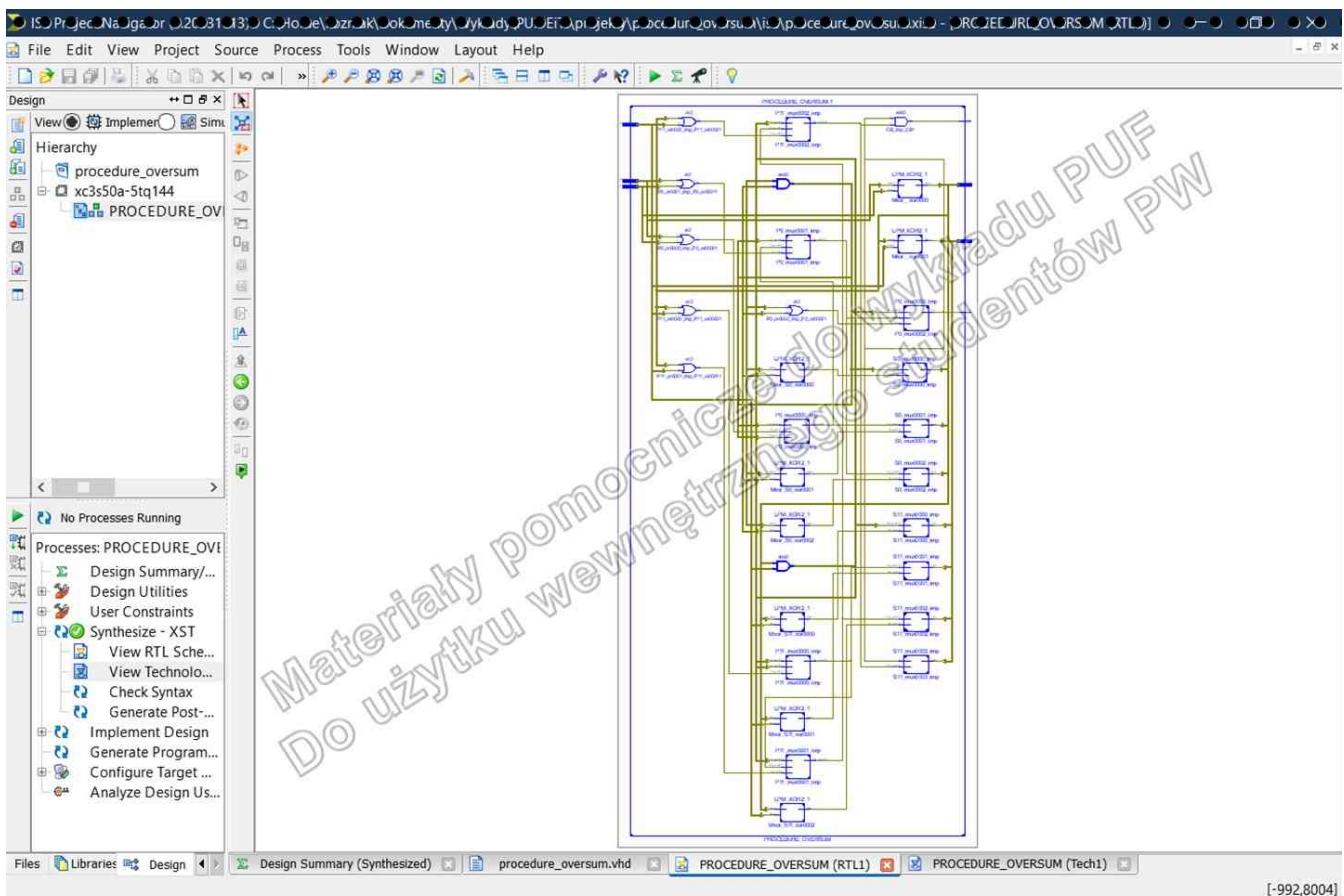
Przykład użycia oprogramowania ISE – projekt: „procedure_oversum”
Plik źródłowy „procedure_oversum.vhd” (fragment 1)

```

38 end procedure sum; -- zakonczenie procedury
39
40 procedure sumator(
41   signal A1, A2 : in bit_vector;
42   signal Y : out bit_vector;
43   signal C : out bit)
44 is
45   variable S : bit_vector(Y'range);
46   variable P : bit;
47 begin
48   sum(A1,A2,S,P); -- deklaracja procedury 'sumator'
49   Y <= S; -- wejsciowe sygnały typu 'bit_vector'
50   C <= P; -- wyjściowy sygnał typu 'bit_vector'
51
52 end procedure sumator;
53
54 procedure sumator(
55   signal A1, A2, A3 : in bit_vector;
56   signal Y : out bit_vector;
57   signal C : out bit)
58 is
59   variable S1, S2, S3 : bit_vector(Y'range);
60   variable P : bit;
61 begin
62   sum(A1,A2,S1(A1'range),S1(A1'length)); -- czesc deklaracyjna procedury
63   S3 := '0'&A3; -- utworzenie zmiennej 'S' w procedurze
64   sum(S3,S1,S2,P); -- utworzenie zmiennej 'P' w procedurze
65   -- sum("0"&A3,S1,S2,P) -- czesc wykonawcza procedury
66   Y <= S2; -- wywolanie przeciazonej procedury 'sumator'
67   C <= P; -- przypisanie zmiennej 'S3'
68   and procedure sumator; -- wywolanie przeciazonej procedury 'sumator'
69
70   sumator(A1, A2, YA, CA); -- ??? wywolanie przeciazonej procedury 'sumator'
71   sumator(A1, A2, A3, YB, CB); -- przypisanie zmiennej 'P' do portu 'C'
72
73 end architecture cialo; -- zakończenie deklaracji ciala 'cialo'
74
75 end PROCEDURE_OVERSUM; -- zakonczenie deklaracji sprawu 'PROCEDURE_OVERSUM'

```

Przykład użycia oprogramowania ISE – projekt: „procedure_oversum”
Plik źródłowy „procedure_oversum.vhd” (fragment 2)



File Edit View Project Source Process Tools Window Layout Help

Design View Implement Sim

Hierarchy

```

1 entity PROCEDURE_OVERSUM_TB is
2 end PROCEDURE_OVERSUM_TB; -- pusty szkielet projektu symulacji

3
4 architecture behavioural of PROCEDURE_OVERSUM_TB is -- cialo architektoniczne projektu
5
6 signal A1 : bit_vector(3 downto 0); -- deklaracja portu wejsciowego 'A1'
7 signal A2 : bit_vector(3 downto 0); -- deklaracja portu wejsciowego 'A2'
8 signal A3 : bit_vector(3 downto 0); -- deklaracja portu wejsciowego 'A3'
9 signal YA : bit_vector(3 downto 0); -- deklaracja portu wyjsciowego 'YA'
10 signal CA : bit; -- deklaracja portu wyjsciowego 'CA'
11 signal YB : bit_vector(4 downto 0); -- deklaracja portu wyjsciowego 'YB'
12 signal CB : bit; -- deklaracja portu wyjsciowego 'CB'

13 begin
14
15 process is -- proces bezwarunkowy
16 begin -- czesc wykonawcza procesu
17
18 A1 <= "0000"; A2 <= "0000"; A3 <= "0000"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2' i oczekanie
19 A1 <= "0001"; A2 <= "0011"; A3 <= "0110"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2' i oczekanie
20 A1 <= "0010"; A2 <= "0111"; A3 <= "1100"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2' i oczekanie
21 A1 <= "0110"; A2 <= "0100"; A3 <= "0001"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2' i oczekanie
22 A1 <= "1000"; A2 <= "0111"; A3 <= "0111"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2' i oczekanie
23 A1 <= "1010"; A2 <= "0110"; A3 <= "1101"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2' i oczekanie
24 A1 <= "1111"; A2 <= "1111"; A3 <= "1111"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2' i oczekanie
25 end process; -- zakończenie procesu
26
27 procedure_oversum inst: entity work.PROCEDURE_OVERSUM(cialo) -- instancja projektu 'PROCEDURE_OVERSUM'
28   port map (A1, A2, A3, YA, CA, YB, CB); -- przypisanie portom sygnalow
29
30 end behavioural; -- zakończenie ciala architektonicznego
31

```

Files Libraries Design Design Summary (out of date) procedure_oversum_tb.vhd

Console

```

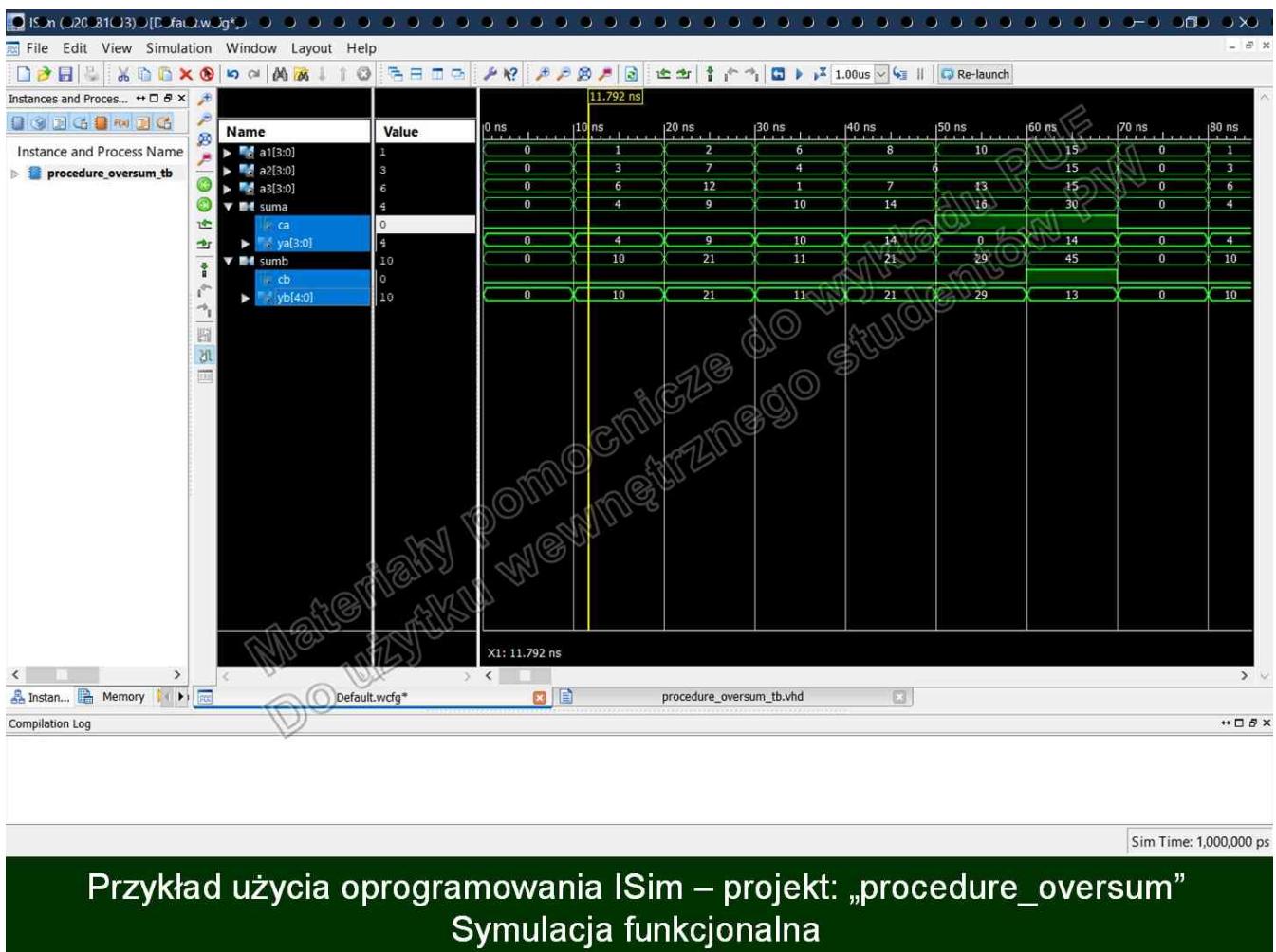
Started : "Launching ISE Text Editor to edit procedure_oversum_tb.vhd".

```

Ln 4 Col 1 VHDL

Przykład użycia oprogramowania ISE – projekt: „procedure_oversum”

Plik źródłowy „procedure_oversum_TB.vhd”



Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [function] [nazwa_fun];
```

Podstawowa składnia deklaracji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ ;
```

Deklaracja funkcji **zapewnia jej udostępnienie**,
a jej definicja może być umieszczona **w innej części programu**

Deklaracja funkcji **jest opcjonalna**, ale jej definicja **musi** być
umieszczona w sprzężonej jednostce leksykalnej programu

Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [function] [nazwa_fun];
```

Podstawowa składnia deklaracji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ ;
```

- Wybrane rodzaje listy argumentów:

- stała: [constant] nazwa1 { , nazwa2 } : [in] typ
- zmienna: jest traktowana identycznie jak stała
- sygnał: signal nazwa1 { , nazwa2 } : [in] typ

Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [function] [nazwa_fun];
```

Podstawowa składnia deklaracji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ ;
```

- Wybrane rodzaje listy argumentów:
 - stała: [constant] nazwa1 { , nazwa2 } : [in] typ
 - zmienna: jest traktowana identycznie jak stała
 - sygnał: signal nazwa1 { , nazwa2 } : [in] typ
- Funkcja zwraca typ uprzednio zdefiniowany (poza funkcją)

Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji funkcji:

```
function nazwa_fun [ ( lista_arg1 {, lista_arg2} ) ] return typ is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [function] [nazwa_fun];
```

Podstawowa składnia deklaracji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ ;
```

- Wybrane składniki części deklaracyjnej (identycznie jak dla procedury):
 - definicja typu/podtypu (podobnie jak w ciele jednostki projektowej)
 - definicja stałej (podobnie jak w ciele jednostki projektowej)
 - definicja zmiennej (podobnie jak w ciele procesu)
 - definicja podprogramu (jest to podprogram zagnieżdżony)

Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [function] [nazwa_fun];
```

Instrukcje sekwencyjne

Podstawowa składnia deklaracji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ ;
```

- Wybrane instrukcje części wykonawczej (tak jak dla procedury) :

- instrukcja prostego przypisania sygnału/zmiennej
- instrukcja warunkowego/selektywnego wyboru instrukcji
- instrukcja pętli dyskretnej/warunkowej
- wywołanie podprogramu
- instrukcja powrotu z funkcji: **return**



```
entity FUNCTION_SUM is
    port ( A1 : in bit_vector(3 downto 0);
           A2 : in bit_vector(3 downto 0);
           Y : out bit_vector(3 downto 0));
end FUNCTION_SUM;
-- deklaracja sprawdzenia 'FUNCTION_SUM'
-- deklaracja portu wejściowego 'A1'
-- deklaracja portu wejściowego 'A2'
-- deklaracja portu wyjściowego 'Y'
-- zakończenie deklaracji nagłówka

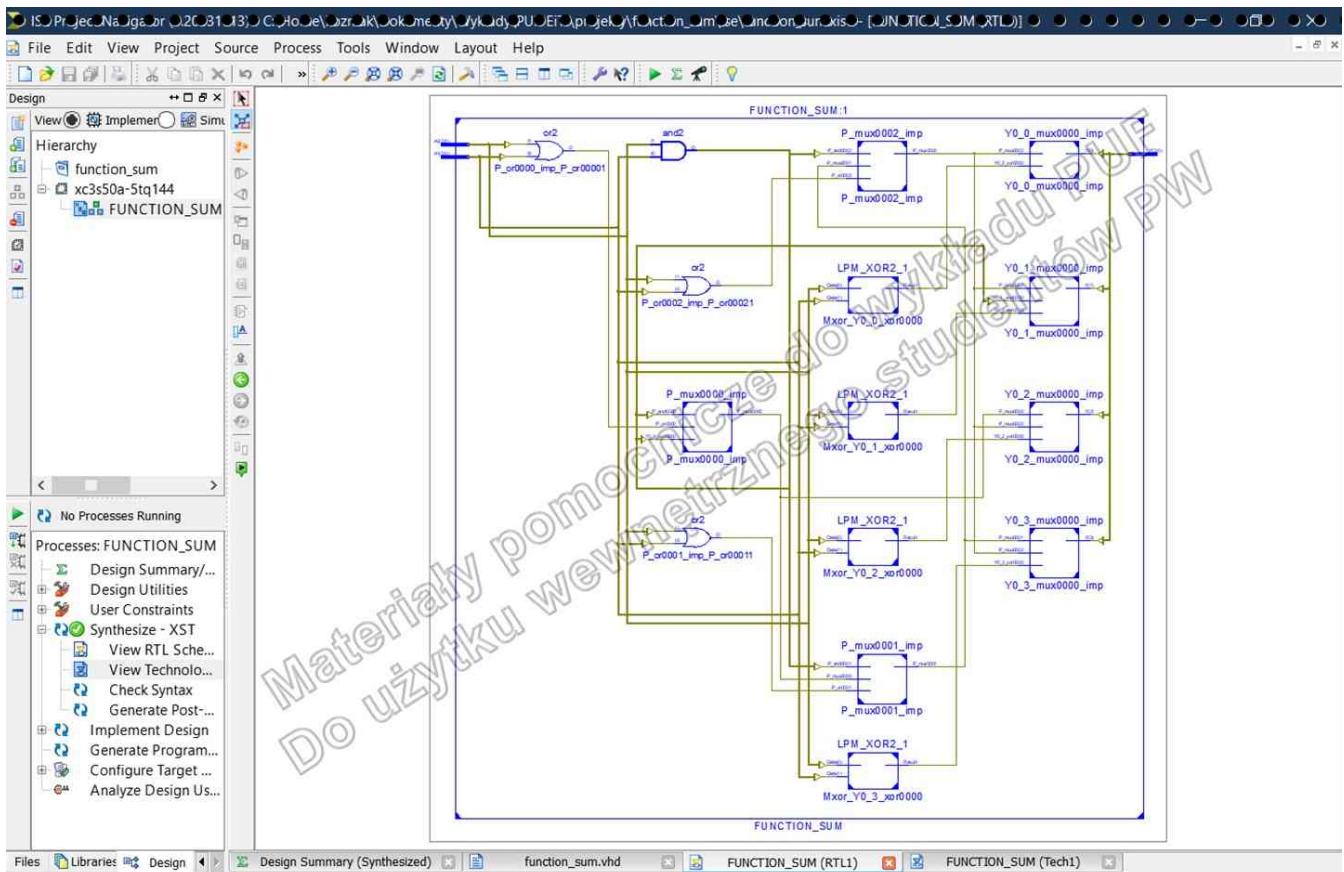
architecture cialo of FUNCTION_SUM is
    -- deklaracja ciała 'cialo' architektury

    function sumator(A, B :bit_vector) return bit_vector is -- utworzenie funkcji 'sumator'
        variable P, S : bit; -- utworzenie zmiennej 'C' i 'S' w procedurze
        variable Y : bit_vector(A'range); -- utworzenie zmiennej 'C' i 'S' w procedurze
        procedure sum(A, B :bit; S :out bit; P :inout bit) is -- utworzenie procedury 'sum'
            begin -- część wykonawcza procedury
                if (P='0') then -- badanie, czy bit przeniesienia jest równy '0'
                    S := A xor B; -- sumowanie bitów funkcja 'xor'
                    P := A and B; -- wyznaczenie bitu przeniesienia funkcja 'and'
                else -- wariant dla 'P' równego '1'
                    S := A xnor B; -- sumowanie bitów funkcja 'xnor'
                    P := A or B; -- wyznaczenie bitu przeniesienia funkcja 'or'
                end if; -- zakończenie instrukcji warunkowej
            end procedure sum; -- zakończenie procedury
            begin -- część wykonawcza procedury
                P := '0'; -- ustalenie wartości początkowej 'P' na '0'
                for i in 0 to Y'length-1 loop -- pętla po kolejnych bitach dla identyfikatora
                    sum(A(i),B(i),S,P); -- wywołanie procedury wewnętrznej 'sum'
                    Y(i) := S; -- przypisanie zmiennej 'S' do portu 'Y(i)'
                end loop; -- zakończenie pętli
                if (P='1') then -- badanie wystąpienia przeniesienia 'P'
                    Y := (others => '1'); -- ustalenie wartości maksymalnej
                end if; -- zakończenie instrukcji warunkowej
                return Y; -- zwrócenie zmiennej 'Y'
            end function sumator; -- zakończenie funkcji

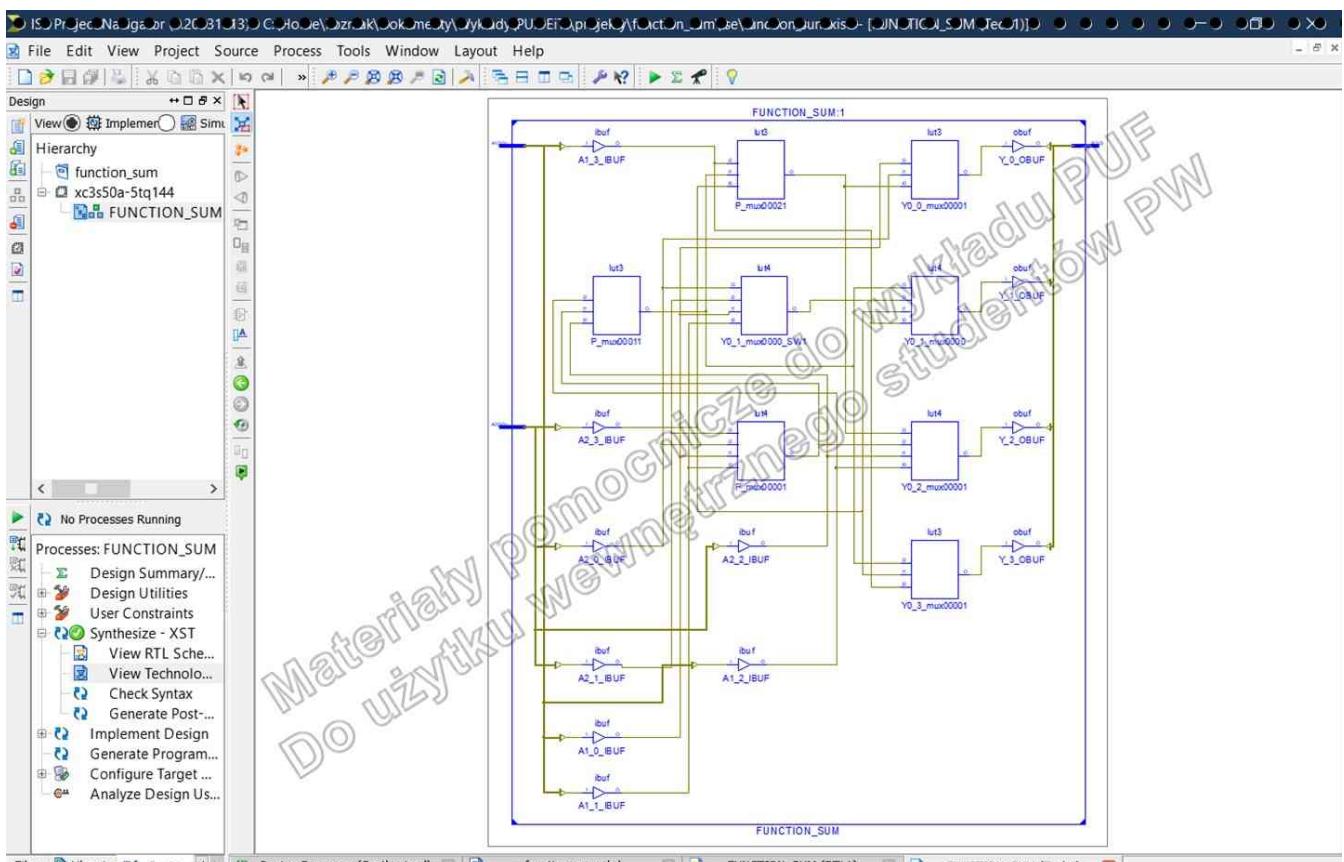
            begin -- początek części wykonawczej
                Y <= sumator(A1, A2); -- wywołanie funkcji 'sumator'
            end architecture cialo; -- zakończenie deklaracji ciała 'cialo'
```

Przykład użycia oprogramowania ISE – projekt: „function_sum”

Plik źródłowy „function_sum.vhd”



Przykład użycia oprogramowania ISE – projekt: „function_sum”
Schemat RTL



Przykład użycia oprogramowania ISE – projekt: „function_sum”
Schemat technologiczny

Materiały Domociche do wykładu
Do użytku wewnętrznego studentów PWSZ

```

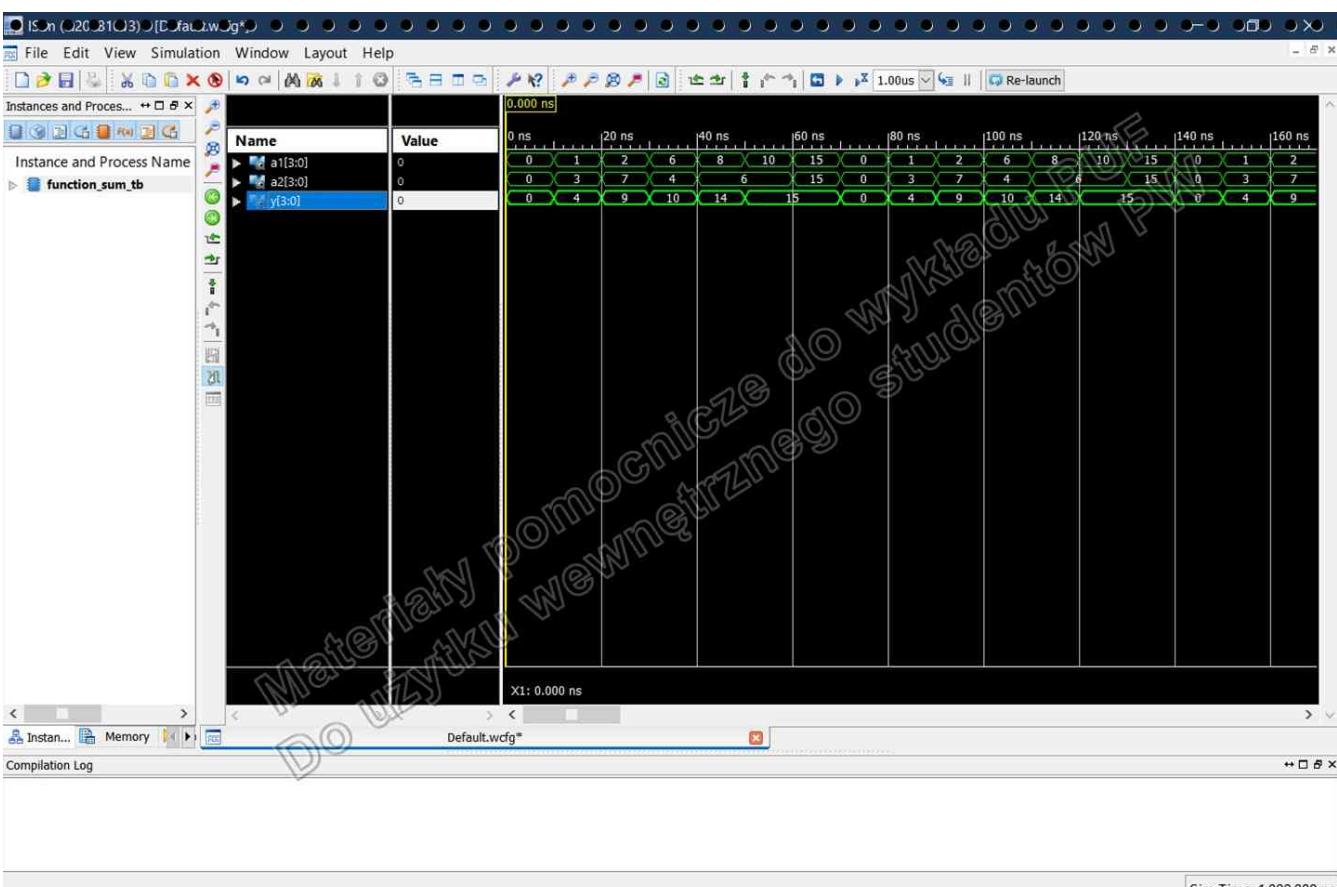
1 entity FUNCTION_SUM_TB is
2 end FUNCTION_SUM_TB;
-- pusty szkielet projektu symulacji

3
4 architecture behavioural of FUNCTION_SUM_TB is -- ciało architektoniczne projektu
5
6 signal A1 : bit_vector(3 downto 0); -- deklaracja portu wejściowego 'A1'
7 signal A2 : bit_vector(3 downto 0); -- deklaracja portu wejściowego 'A2'
8 signal Y : bit_vector(3 downto 0); -- deklaracja portu wyjściowego 'YA'
9
10 begin
11
12 process is
13 begin
14     A1 <= "0000"; A2 <= "0000"; wait for 10ns;
15     A1 <= "0001"; A2 <= "0011"; wait for 10ns;
16     A1 <= "0010"; A2 <= "0111"; wait for 10ns;
17     A1 <= "0110"; A2 <= "0100"; wait for 10ns;
18     A1 <= "1000"; A2 <= "0110"; wait for 10ns;
19     A1 <= "1010"; A2 <= "0110"; wait for 10ns;
20     A1 <= "1111"; A2 <= "1111"; wait for 10ns;
21 end process;
22
23 function_sum_inst: entity work.FUNCTION_SUM(ciało)
24     port map (A1, A2, Y);
25
26 end behavioural;
27

```

Started : "Launching ISE Text Editor to edit function_sum_tb.vhd".

Przykład użycia oprogramowania ISE – projekt: „function_sum” Plik źródłowy „function_sum_TB.vhd”



Przykład użycia oprogramowania ISim – projekt: „function_sum” Symulacja funkcjonalna

Podstawowe elementy standardu VHDL

Wybrane konstrukcje podprogramów

Podstawowa składnia definicji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ is
    [ część_deklaracyjna ]
    begin
        [ część_wykonawcza ]
    end [function] [nazwa_fun];
```

Podstawowa składnia deklaracji funkcji:

```
function nazwa_fun [ ( lista_arg1 {; lista_arg2} ) ] return typ ;
```

Funkcja może być przeciążona, czyli tworzyć rodzinę funkcji o wspólnej nazwie funkcji, ale o różnej liczbie argumentów oraz o różnych typach dla tych samych argumentów

```
1 entity FUNCTION_OVERSUM is
2     port ( A1 : in bit_vector(4 downto 0);
3             A2 : in bit_vector(3 downto 0);
4             A3 : in bit_vector(2 downto 0);
5             Y2 : out bit_vector(4 downto 0);
6             Y3 : out bit_vector(5 downto 0));
7     end FUNCTION_OVERSUM;
8
9 architecture cialo of FUNCTION_OVERSUM is
10
11     function sumator(A, B :bit_vector) return bit_vector is
12
13         function max(A,B :positive) return positive is
14             begin
15                 if (A>=B) then return A; end if;
16                 return B;
17             end function max;
18
19         constant N : positive := max(A'length,B'length);
20         variable AN, BN, Y : bit_vector(N downto 0) := (others => '0');
21
22         function sum(A, B :bit_vector) return bit_vector is
23             variables P, S : bit;
24             variables Y : bit_vector(A'range);
25             procedure sum(A, B :bit; S :out bit; P :inout bit) is
26                 begin
27                     if (P='0') then
28                         S := A xor B;
29                         P := A and B;
30                     else
31                         S := A xor B;
32                         P := A or B;
33                     end if;
34                 end procedure sum;
35             begin
36                 P := '0';
37                 for i in 0 to Y'length-1 loop
38                     sum(A(i),B(i),S,P);
39                     Y(i) := S;
40                 end loop;
41             return Y;
42         end function sum;
```

Annotations in the code:

- Line 1: -- deklaracja sprawu FUNCTION_OVERSUM'
- Line 2: -- deklaracja portu wejściowego 'A1'
- Line 3: -- deklaracja portu wejściowego 'A2'
- Line 4: -- deklaracja portu wejściowego 'A3'
- Line 5: -- deklaracja portu wyjściowego 'Y2'
- Line 6: -- deklaracja portu wyjściowego 'Y3'
- Line 7: -- zakończenie deklaracji nagłówka
- Line 9: -- deklaracja ciała 'cialo' architektury
- Line 11: -- utworzenie przeciążonej funkcji 'sumator'
- Line 13: -- utworzenie wewnętrznej funkcji 'max'
 - cześć wykonawcza procedury
 - badanie i zwrocenie wartości 'A' gdy nie mniejsza
 - zwrocenie wartości 'B'
 - zakończenie funkcji
- Line 19: -- utworzenie stałej 'N' o wartości maksymalnej długości
- Line 20: -- utworzenie zmiennych 'AN', 'BN', 'Y'
- Line 22: -- utworzenie funkcji 'sum'
- Line 23: -- utworzenie zmiennych 'P' i 'S' w procedurze
- Line 24: -- utworzenie zmiennej 'Y' w procedurze
- Line 25: -- utworzenie wewnętrznej procedury 'sum'
- Line 26: -- cześć wykonawcza procedury
- Line 27: -- badanie czy bit przeniesienia jest równy '0'
- Line 28: -- sumowanie bitów funkcja 'xor'
- Line 29: -- wyznaczenie bitu przeniesienia funkcja 'and'
- Line 30: -- wariant dla 'P' równego '1'
- Line 31: -- sumowanie bitów funkcja 'xnor'
- Line 32: -- wyznaczenie bitu przeniesienia funkcja 'or'
- Line 33: -- zakończenie instrukcji warunkowej
- Line 34: -- zakończenie procedury
- Line 35: -- cześć wykonawcza procedury
- Line 36: -- ustawienie wartości początkowej 'P' na '0'
- Line 37: -- pętla po kolejnych bitach dla identyfikatora 'i'
- Line 38: -- wywołanie procedury wewnętrznej 'sum'
- Line 39: -- przypisanie zmiennej 'S' do portu 'Y(i)'
- Line 40: -- zakończenie pętli
- Line 41: -- przypisanie zmiennej 'P' do portu 'C'
- Line 42: -- zakończenie funkcji

Przykład użycia oprogramowania ISE – projekt: „function_oversum”

Plik źródłowy „function_oversum.vhd” (fragment 1)

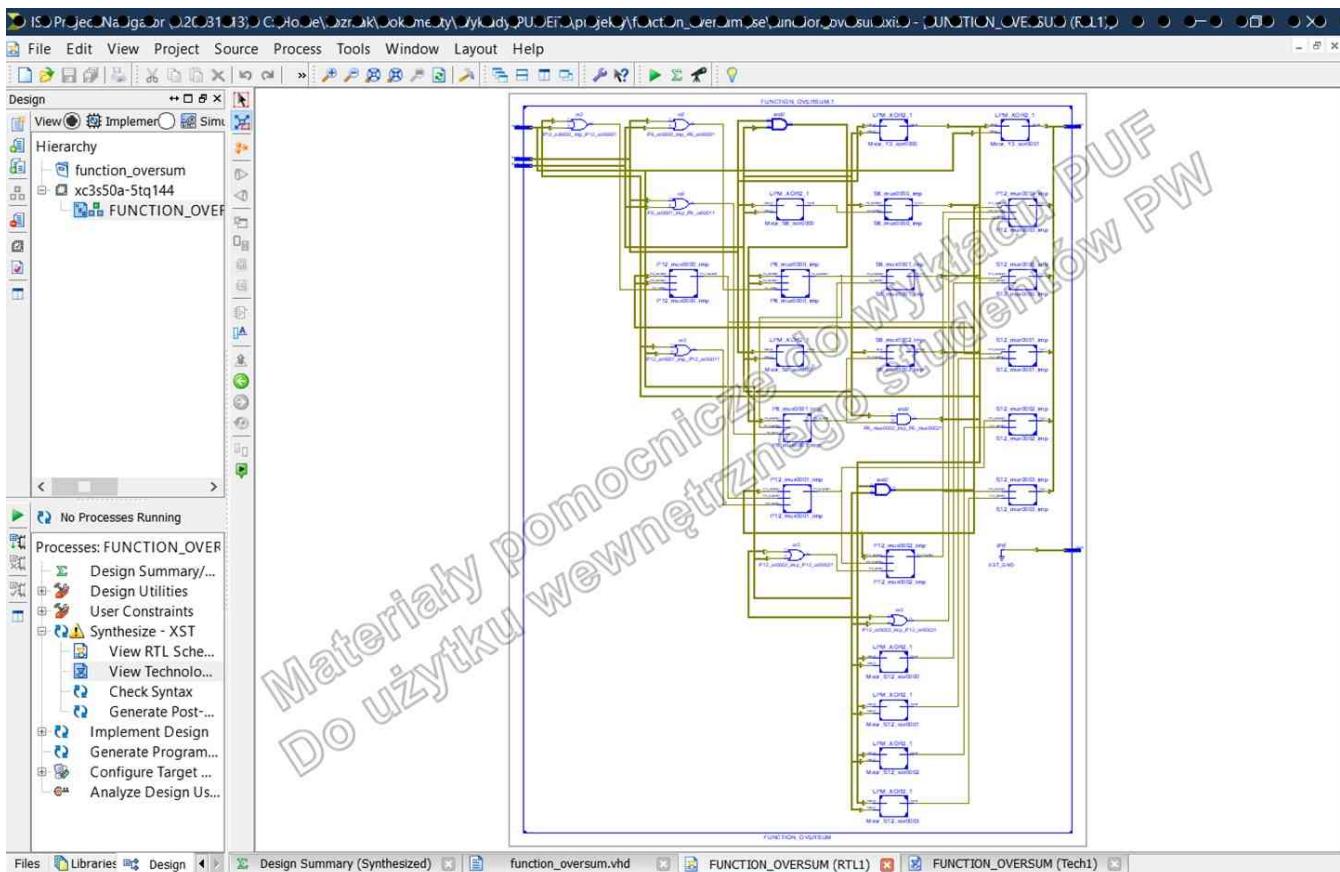
Przykład użycia oprogramowania ISE – projekt: „function_oversum”

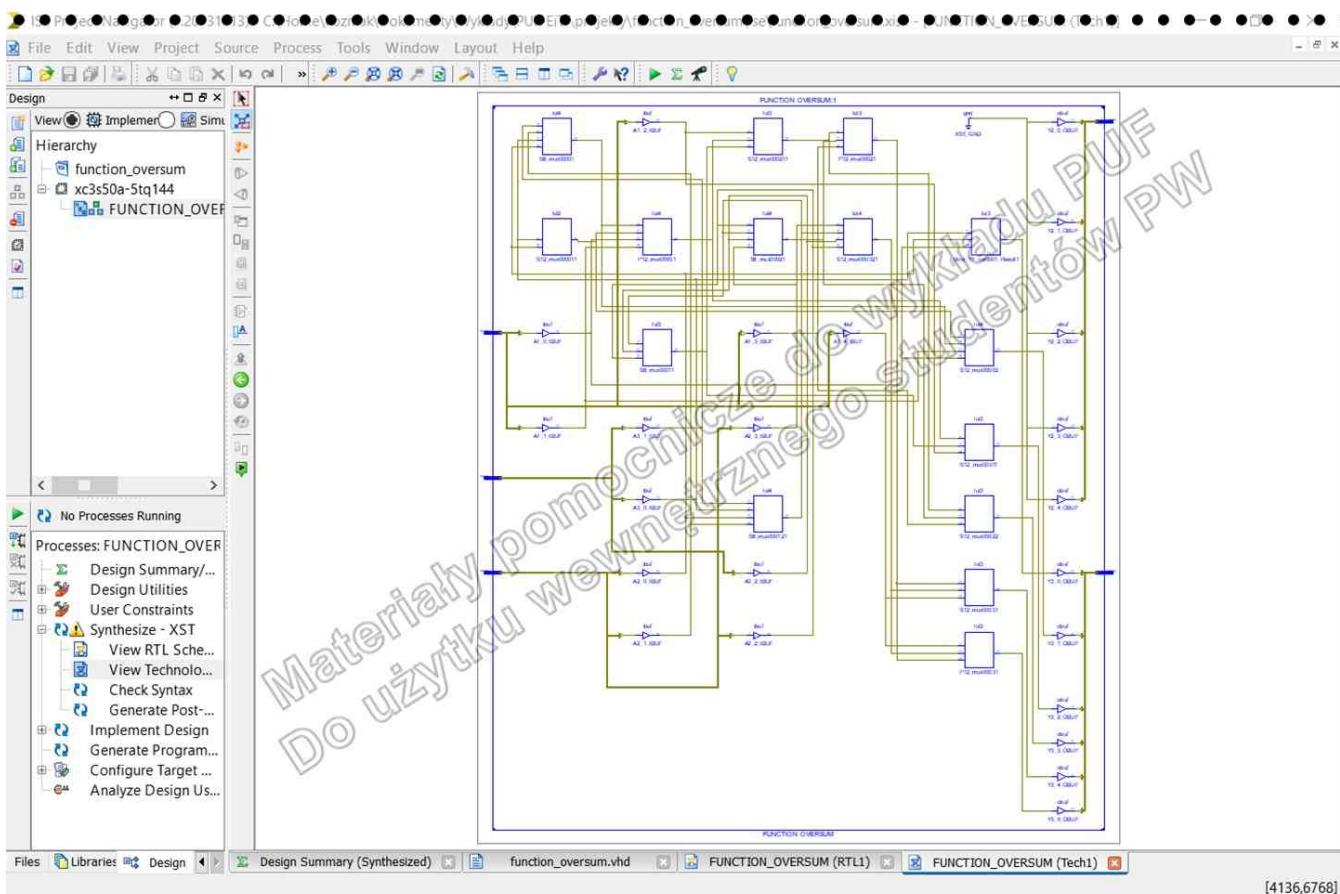
Plik źródłowy „function_oversum.vhd” (fragment 2)

```

40      end loop;
41      return Y;
42  end function sum;
43
44 begin
45     AN(A'range) := A;
46     BN(B'range) := B;
47     return sum(AN, BN);
48 end function sumator;
49
50
51 function sumator(A, B : bit_vector; L: positive) return bit_vector is -- utworzenie przekształconej funkcji 'sumator'
52     constant S : bit_vector := sumator(A, B); -- utworzenie stałej 'S' z wartością zwracaną przez 'sumator'
53     variable Y : bit_vector(L-1 downto 0) := (others =>'1'); -- utworzenie zmiennej 'Y' o wartości maksymalnej
54 begin
55     if (L < S'length) then -- czesc wykonawcza procedury
56         for i in L to S'length-1 loop -- czesciowe przypisanie zmiennej 'Y'
57             if (S(i)='1') then -- czesciowe przypisanie zmiennej 'BN'
58                 return Y; -- wywołanie przekształconej procedury 'sum'
59             end if;
60         end loop; -- zakończenie funkcji
61         return S(L-1 downto 0); -- zakończenie procedury
62     end if;
63     Y := (others => '0'); -- zakończenie instrukcji warunkowej
64     Y(S'range) := S; -- zakończenie instrukcji warunkowej
65     return Y; -- zakończenie procedury
66 end function sumator;
67
68
69 function sumator(A, B, C : bit_vector; L: positive) return bit_vector is -- utworzenie przekształconej funkcji 'sumator'
70 begin -- czesc wykonawcza procedury
71     return sumator(A,sumator(B,C),L); -- zagnieżdżone wywołanie przekształconych funkcji 'sumator'
72 end function sumator;
73
74 begin
75     Y2 <= sumator(A1, A2, Y2'length); -- zakończenie procedury
76     Y3 <= sumator(A1, A2, A3, Y3'length); -- zakończenie procedury
77
78 end architecture cialo; -- zakończenie deklaracji ciała 'cialo'
79
80
81

```





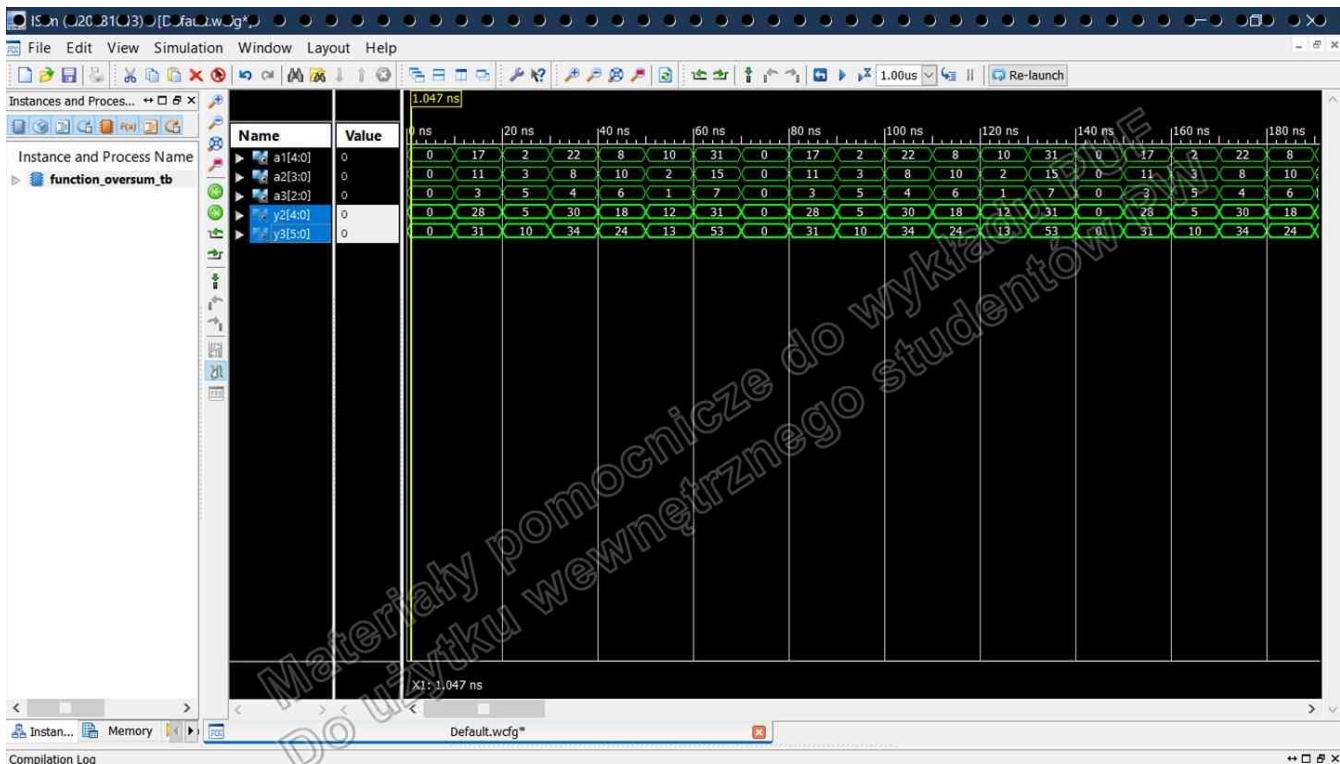
Przykład użycia oprogramowania ISE – projekt: „function_oversum” Schemat technologiczny

```

1 entity FUNCTION_OVERSUM_TB is
2 end FUNCTION_OVERSUM_TB;
3
4 architecture behavioural of FUNCTION_OVERSUM_TB is
5
6   signal A1 : bit_vector(4 downto 0);
7   signal A2 : bit_vector(3 downto 0);
8   signal A3 : bit_vector(2 downto 0);
9   signal Y2 : bit_vector(4 downto 0);
10  signal Y3 : bit_vector(5 downto 0);
11
12 begin
13
14  process is
15    begin
16    A1 <= "00000"; A2 <= "0000"; A3 <= "000"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2', 'A3' i odcz
17    A1 <= "10001"; A2 <= "1011"; A3 <= "111"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2', 'A3' i odcz
18    A1 <= "00010"; A2 <= "0011"; A3 <= "011"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2', 'A3' i odcz
19    A1 <= "10110"; A2 <= "1000"; A3 <= "100"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2', 'A3' i odcz
20    A1 <= "00100"; A2 <= "1010"; A3 <= "110"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2', 'A3' i odcz
21    A1 <= "01010"; A2 <= "0010"; A3 <= "0011"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2', 'A3' i odcz
22    A1 <= "11111"; A2 <= "1111"; A3 <= "1111"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2', 'A3' i odcz
23  end process; -- zakończenie procesu
24
25  function_oversum inst: entity work.FUNCTION_OVERSUM(ciało) -- instancja projektu 'FUNCTION_OVERSUM'
26    port map (A1, A2, A3, Y2, Y3); -- przypisanie portów sygnałów
27
28 end behavioural; -- zakończenie ciała architektonicznego
29

```

Przykład użycia oprogramowania ISE – projekt: „function_oversum” Plik źródłowy „function_oversum_TB.vhd”



Przykład użycia oprogramowania ISim – projekt: „function_oversum” Symulacja funkcjonalna

Podstawowe elementy standardu VHDL Wybrane konstrukcje podprogramów

~~Podstawowa składnia definicji funkcji operatorowej:~~

```
function "operator" ([larg ;] parg | lista_2arg) return typ is
  [ część_deklaracyjna ]
  begin
  [ część_wykonawcza ]
end [ function ] [ "operator" ] ;
```

~~Podstawowa składnia deklaracji funkcji operatorowej:~~

```
function "operator" ([larg ;] parg | lista_2arg) return typ ;
```

Funkcja operatorowa **może używać predefiniowanych operatorów i może zostać użyta w postaci operatora** w relacji do argumentów

Funkcja operatorowa **może być przeciążona**, czyli tworzyć **rodzinę funkcji o wspólnej nazwie** funkcji, ale o **różnych typach** argumentów

```

1 entity FUNCTION_OPERSUM is
2 port ( A1 : in bit_vector(4 downto 0);
3         A2 : in bit_vector(3 downto 0);
4         A3 : in bit_vector(2 downto 0);
5         Y : out bit_vector(4 downto 0));
6 end FUNCTION_OPERSUM;
7
8 architecture cialo of FUNCTION_OPERSUM is
9
10 function "+"(A, B :bit_vector) return bit_vector is
11
12     function max(A,B :positive) return positive is
13     begin
14         if (A>B) then return A; end if;
15         return B;
16     end function max;
17
18     constant N : positive := max(A'length,B'length);
19     variable AN, BN, Y : bit_vector(N-1 downto 0) := (others => '0'); -- utworzenie zmiennych 'AN', 'BN'
20
21     function sum(A, B :bit_vector) return bit_vector is
22         variable P, S : bit;
23         variable Y : bit_vector(A'range);
24         procedure sum(A, B :bit; S :out bit ; P :inout bit) is
25         begin
26             if (P='0') then
27                 S := A xor B;
28                 P := A and B;
29             else
30                 S := A xnor B;
31                 P := A or B;
32             end if;
33
34         begin
35             Y := (others => '0');
36             for i in 0 to Y'length-1 loop
37                 sum(A(i),B(i),S,P);
38                 Y(i) := S;
39             end loop;
40             return Y;
41         end function sum;
42
43         begin
44             AN(A'range) := A;
45             BN(B'range) := B;
46             return sum(AN, BN);
47         end function "+";
48
49     begin
50
51         Y <= A1 + A2 + A3;
52
53     end architecture cialo;

```

Started : "Launching ISE Text Editor to edit function_opersum.vhd".

Ln 10 Col 1 VHDL

Przykład użycia oprogramowania ISE – projekt: „function_opersum”
Plik źródłowy „function_opersum.vhd” (fragment 1)

```

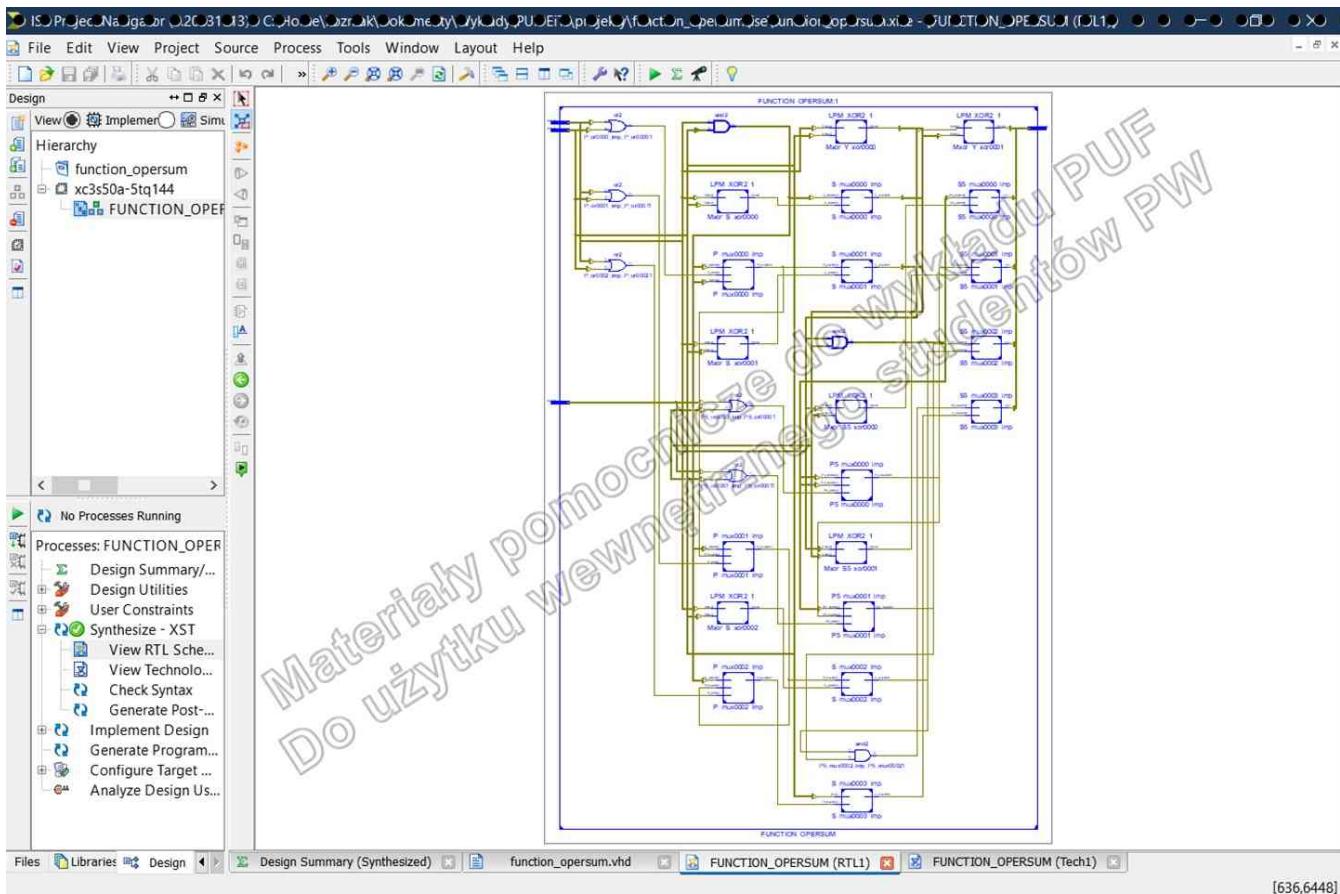
23         variable Y : bit_vector(A'range);
24         procedure sum(A, B :bit; S :out bit ; P :inout bit) is
25         begin
26             if (P='0') then
27                 S := A xor B;
28                 P := A and B;
29             else
30                 S := A xnor B;
31                 P := A or B;
32             end if;
33         end procedure sum;
34
35         begin
36             P := '0';
37             for i in 0 to Y'length-1 loop
38                 sum(A(i),B(i),S,P);
39                 Y(i) := S;
40             end loop;
41             return Y;
42         end function sum;
43
44         begin
45             AN(A'range) := A;
46             BN(B'range) := B;
47             return sum(AN, BN);
48         end function "+";
49
50     begin
51         Y <= A1 + A2 + A3;
52
53     end architecture cialo;

```

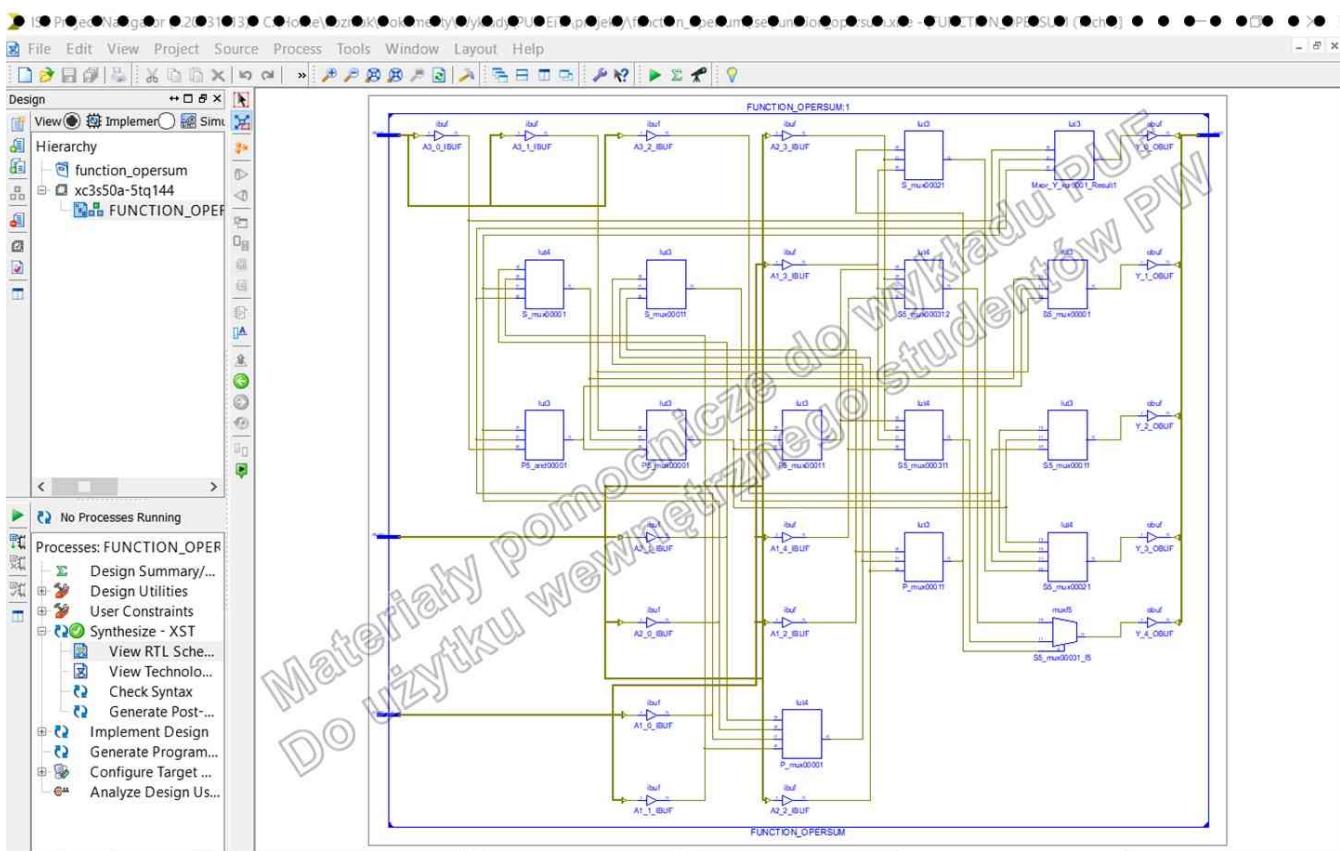
Started : "Launching ISE Text Editor to edit function_opersum.vhd".

Ln 10 Col 1 VHDL

Przykład użycia oprogramowania ISE – projekt: „function_opersum”
Plik źródłowy „function_opersum.vhd” (fragment 2)



Przykład użycia oprogramowania ISE – projekt: „function_opersum”
Schemat RTL



Przykład użycia oprogramowania ISE – projekt: „function_opersum”
Schemat technologiczny

File Edit View Project Process Tools Window Layout Help

Design View Implement Sim

Hierarchy

function_opersum

 FUNCTION_OPERSUM

 function_opersum

```

1 entity FUNCTION_OPERSUM_TB is
2 end FUNCTION_OPERSUM_TB; -- pusty szkielet projektu symulacji
3
4 architecture behavioural of FUNCTION_OPERSUM_TB is -- cialo architektoniczne projektu
5
6   signal A1 : bit_vector(4 downto 0); -- deklaracja portu wejsciowego 'A1'
7   signal A2 : bit_vector(3 downto 0); -- deklaracja portu wejsciowego 'A2'
8   signal A3 : bit_vector(2 downto 0); -- deklaracja portu wejsciowego 'A3'
9   signal Y : bit_vector(4 downto 0); -- deklaracja portu wyjsciowego 'Y'
10
11 begin
12
13   process is -- proces bezwarunkowy
14     begin
15       A1 <= "00000"; A2 <= "000"; A3 <= "000"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2', 'A3' i odcz
16       A1 <= "10001"; A2 <= "1011"; A3 <= "011"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2', 'A3' i odcz
17       A1 <= "00010"; A2 <= "0011"; A3 <= "111"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2', 'A3' i odcz
18       A1 <= "10110"; A2 <= "1000"; A3 <= "110"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2', 'A3' i odcz
19       A1 <= "01000"; A2 <= "1010"; A3 <= "110"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2', 'A3' i odcz
20       A1 <= "01010"; A2 <= "0010"; A3 <= "001"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2', 'A3' i odcz
21       A1 <= "11111"; A2 <= "1111"; A3 <= "1111"; wait for 10ns; -- ustawienie sygnalow 'A1', 'A2', 'A3' i odcz
22     end process; -- zakonczenie procesu
23
24   function_opersum_inst: entity work.FUNCTION_OPERSUM(cialo) -- instancja projektu 'FUNCTION_OPERSUM'
25     port map (A1, A2, A3, Y); -- przypisanie portow sygnalow
26
27 end behavioural; -- zakonczenie ciala architektonicznego
28

```

Files Libraries Design

Design Summary (Synthesized)

function_opersum.vhd

FUNCTION_OPERSUM (RTL1)

FUNCTION_OPERSUM (Tech1)

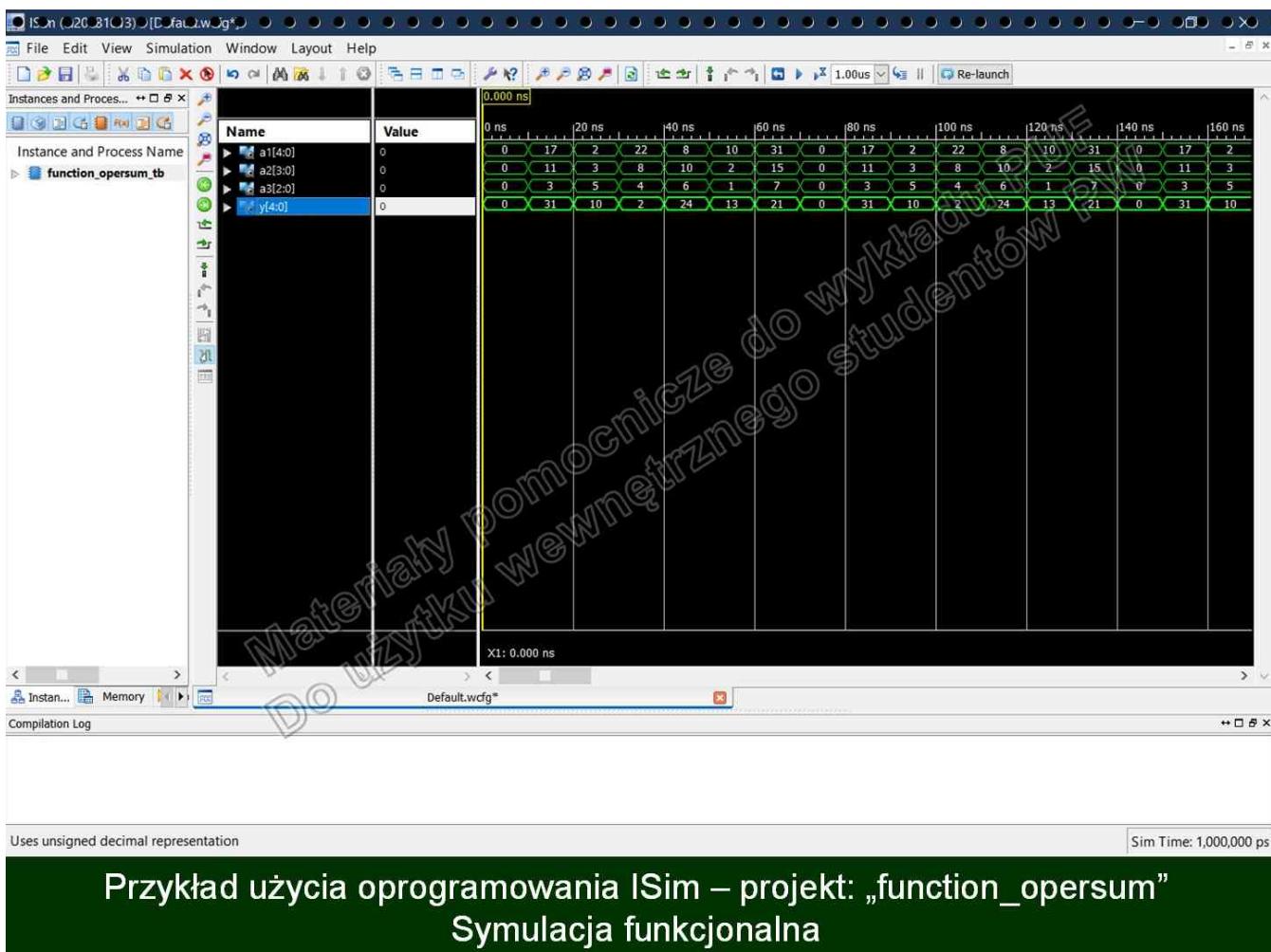
function_opersum_tb.vhd

Console

Started : "Launching ISE Text Editor to edit function_opersum_tb.vhd".

Ln 4 Col 1 | VHDL

Przykład użycia oprogramowania ISE – projekt: „function_opersum” Plik źródłowy „function_opersum_TB.vhd”



Podstawowe elementy standardu VHDL

Wybrane konstrukcje biblioteczne

Podstawowa składnia deklaracji pakietu:

```
package nazwa_pakietu is  
  [ publiczna_część_deklaracyjna ]  
end [package] [nazwa_pakietu] ;
```

- Wybrane składniki publicznej części deklaracyjnej:

- definicja typu/podtypu
- deklaracja stałej
- deklaracja sygnału
- nagłówek podprogramu (procedury lub funkcji)

Podstawowe elementy standardu VHDL

Wybrane konstrukcje biblioteczne

Podstawowa składnia deklaracji pakietu:

```
package nazwa_pakietu is  
  [ publiczna_część_deklaracyjna ]  
end [package] [nazwa_pakietu] ;
```

Podstawowa składnia deklaracji ciała pakietu:

```
package body nazwa_pakietu is  
  [ niepubliczna_część_deklaracyjna ]  
end [package body] [nazwa_pakietu] ;
```

- Wybrane składniki niepublicznej części deklaracyjnej:

- definicja typu/podtypu
- deklaracja stałej
- deklaracja podprogramu (procedury lub funkcji)

Podstawowe elementy standardu VHDL

Wybrane konstrukcje biblioteczne

Podstawowa składnia deklaracji pakietu:

```
package nazwa_pakietu is  
[ publiczna_część_deklaracyjna ]  
end [package ] [nazwa_pakietu ] ;
```

Podstawowa składnia deklaracji ciała pakietu:

```
package body nazwa_pakietu is  
[ niepubliczna_część_deklaracyjna ]  
end [package body ] [nazwa_pakietu ] ;
```

Podstawowa składnia użycia pakietu z biblioteki:

```
library nazwa_biblioteki ;  
{ use nazwa_biblioteki.nazwa_pakietu.all ; }
```

Biblioteka pakietów użytkownika ma nazwę **work** – można ją pominać

```
File Edit View Project Source Process Tools Window Layout Help  
File Files Name Type  
..\\library_sum.vhd VHDL  
..\\library_sum_tb... VHDL  
..\\package_sum.... VHDL  
1 package PACKAGE_SUM is  
2  
3     function sumator(A, B :bit_vector) return bit_vector;  
4     function "+"(A, B :bit_vector) return bit_vector;  
5  
6 end package PACKAGE_SUM;  
7  
8 package body PACKAGE_SUM is  
9  
10    function max(A,B :positive) return positive is  
11        begin  
12            if (A>=B) then return A; end if;  
13            return B;  
14        end function max;  
15  
16    function sumator(A, B :bit_vector) return bit_vector is  
17  
18        constant N : positive := max(A'length,B'length);  
19        variable AN, BN, Y : bit_vector(N-1 downto 0) := (others => '0'); -- utworzenie zmiennych '  
20  
21        function sum(A, B :bit_vector) return bit_vector is  
22            variable P:3 bit;  
23            variable Y : bit vector(A'range);  
24            procedure sum(A, B :bit; S :out bit ; P :inout bit) is  
25                begin  
26                    if (P='0') then  
27                        S := A xor B;  
28                        P := A and B;  
29                    else  
30                        S := A xnor B;  
31                        P := A or B;  
32                    end if;  
33                end procedure sum;  
34                begin  
35                    P := '0';  
-- deklaracja pakietu 'PACKAGE_SUM'  
-- deklaracja funkcji 'sumator'  
-- deklaracja funkcji operatorowej '+'  
-- zakonczenie deklaracji pakietu  
-- deklaracja ciała pakietu 'PACKAGE_SUM'  
-- utworzenie niepublicznej procedury 'max'  
-- zbadanie i zwrocenie wartosci 'A'  
-- zwrocenie wartosci 'B'  
-- zakonczenie procedury  
-- utworzenie publicznej funkcji 'sumator'  
-- utworzenie stalej 'N' o wartosci 3  
-- utworzenie zmiennych 'AN', 'BN', 'Y'  
-- utworzenie funkcji 'sum'  
-- utworzenie zmiennej 'P' i 'Y'  
-- utworzenie wewnętrznej procedury 'sum'  
-- czesc wykonawcza procedury 'sum'  
-- badanie czy bit przeniesieniowy jest '0'  
-- sumowanie bitowe funkcja 'xor'  
-- wyznaczenie bitu przeniesieniowego dla 'P'=1  
-- sumowanie bitowe funkcja 'xnor'  
-- wyznaczenie bitu przeniesieniowego dla 'P'=0  
-- zakonczenie instrukcji warunkowej  
-- zakonczenie procedury  
-- czesc wykonawcza procedury 'sum'  
-- ustawienie wartosci poczatkowej '0'
```

Przykład użycia oprogramowania ISE – projekt: „library_sum”

Plik źródłowy biblioteki „package_sum.vhd” (fragment 1)

```

22 variable P, S : bit;
23 variable Y : bit_vector(A'range);
24 procedure sum(A, B :bit; S :out bit ; P :inout bit) is
25 begin
26     if (P='0') then
27         S := A xor B;
28         P := A and B;
29     else
30         S := A xnor B;
31         P := A or B;
32     end if;
33 end procedure sum;
34 begin
35     P := '0';
36     for i in 0 to Y'length-1 loop
37         sum(A(i),B(i),S,P);
38         Y(i) := S;
39     end loop;
40     return Y;
41 end function sum;
42
43 begin
44     AN(A'range) := A;
45     BN(B'range) := B;
46     return sum(AN,BN);
47 end function sumator;
48
49 function "+"(A, B :bit_vector) return bit_vector is
50 begin
51     return sumator(A,B);
52 end function "+";
53
54 end package body PACKAGE_SUM;
55

```

Annotations for the code:

- Line 22: -- utworzenie zmiennych 'P' i 'S'
- Line 23: -- utworzenie zmiennej 'Y' w programie
- Line 24: -- utworzenie wewnętrznej procedury
- Line 25: -- czesc wykonawcza procedury
- Line 26: -- badanie czy bit przeniesieni
- Line 27: -- sumowanie bitowe funkcja 'xor'
- Line 28: -- wyznaczenie bitu przeniesieni wariancji dla 'P' rownego '1'
- Line 29: -- sumowanie bitowe funkcja 'xnor'
- Line 30: -- wyznaczenie bitu przeniesieni
- Line 31: -- zakonczenie instrukcji warunkowej
- Line 32: -- zakonczenie procedury
- Line 33: -- czesc wykonawcza procedury
- Line 34: -- ustawienie wartosci poczatkowej
- Line 35: -- petla po kolejnych bitach dlugosci wejscia
- Line 36: -- wywolanie procedury wewnętrznej
- Line 37: -- przypisanie zmiennej 'S' do zmiennej 'Y'
- Line 38: -- zakonczenie petli
- Line 39: -- zwrocenie zmiennej 'Y'
- Line 40: -- zakonczenie funkcji
- Line 43: -- czesc wykonawcza procedury
- Line 44: -- czesciowe przypisanie zmiennej 'AN'
- Line 45: -- czesciowe przypisanie zmiennej 'BN'
- Line 46: -- zwrocenie rezultatu funkcji '+'
- Line 47: -- zakonczenie funkcji
- Line 49: -- utworzenie publicznej funkcji
- Line 50: -- czesc wykonawcza procedury
- Line 51: -- zwrocenie rezultatu funkcji
- Line 52: -- zakonczenie funkcji operatorowej
- Line 53: -- zakonczenie deklaracji ciala

Przykład użycia oprogramowania ISE – projekt: „library_sum”

Plik źródłowy biblioteki „package_sum.vhd” (fragment 2)

```

1 library work;
2 use work.PACKAGE_SUM.all;
3
4 entity LIBRARY_SUM is
5     port ( A1 : in bit_vector(3 downto 0);
6             A2 : in bit_vector(3 downto 0);
7             Y : out bit_vector(3 downto 0));
8 end LIBRARY_SUM;
9
10 architecture cialo of LIBRARY_SUM is
11
12 begin
13
14     Y <= A1+sumator(A2,A2);
15
16 end architecture cialo;
17

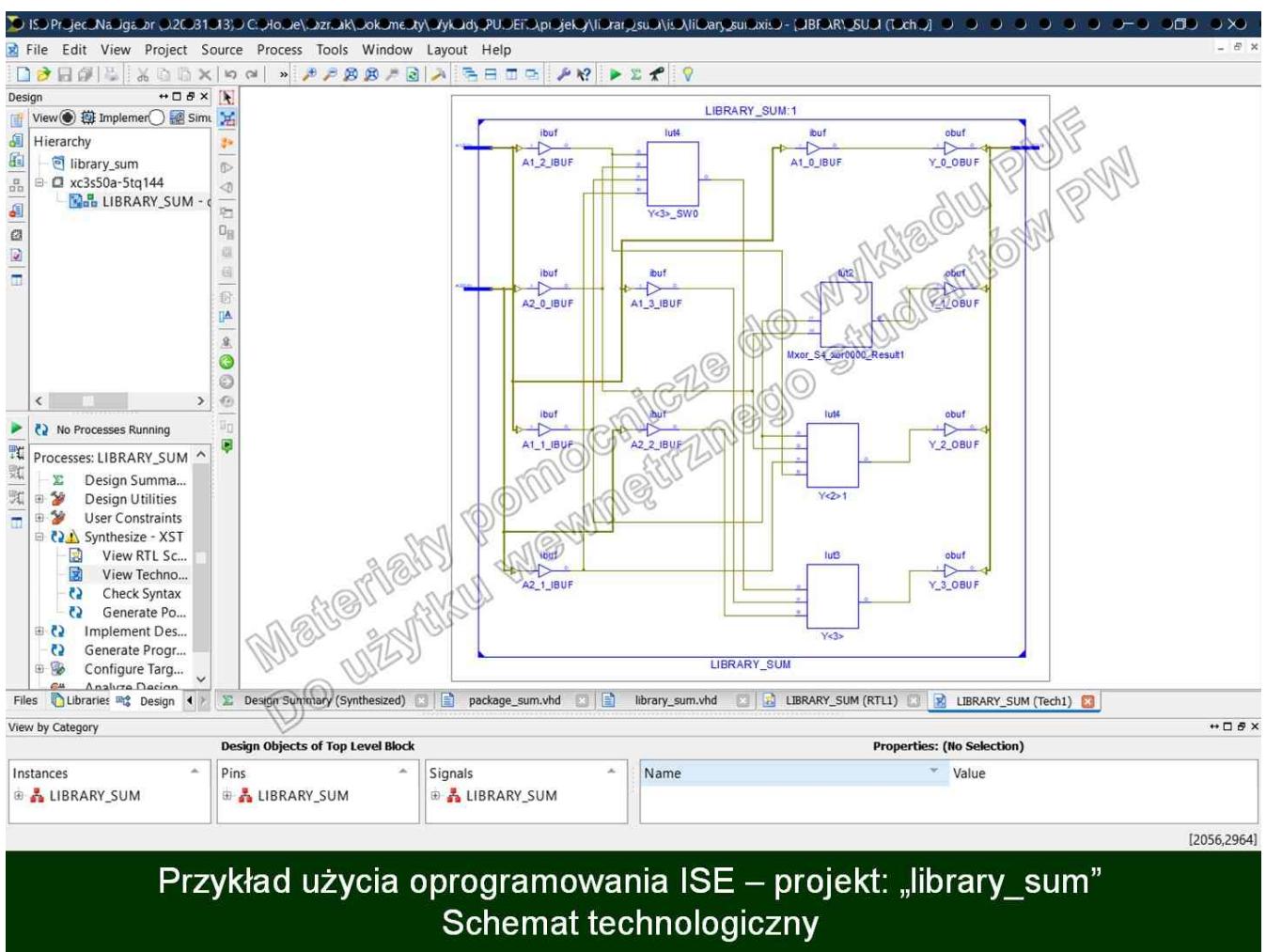
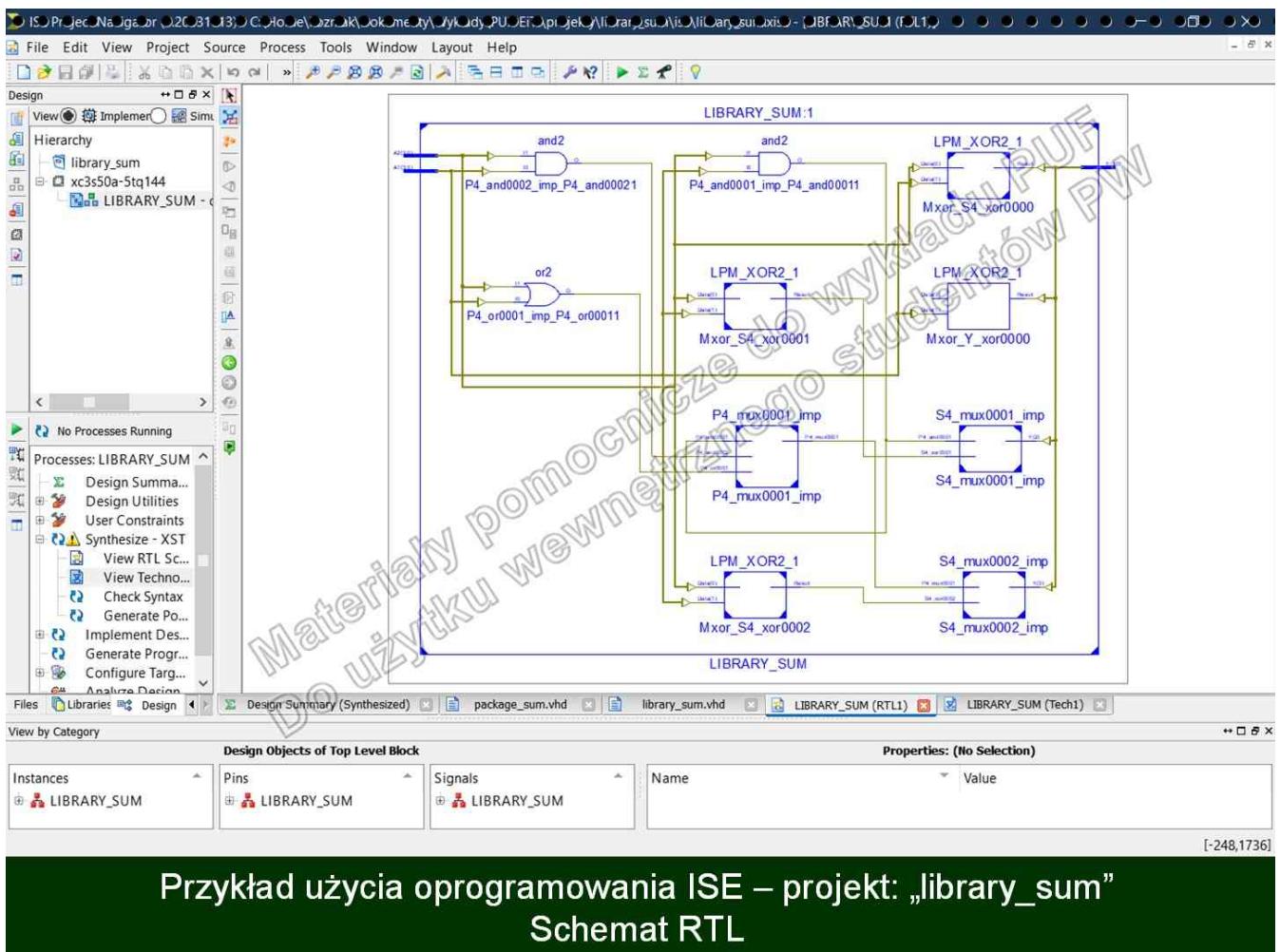
```

Annotations for the code:

- Line 1: -- klaszula dostepu do biblioteki 'work'
- Line 2: -- dolaczanie calego pakietu 'PACKAGE_SUM'
- Line 4: -- deklaracja sprzegu LIBRARY_SUM
- Line 5: -- deklaracja portu wejsciowego 'A1'
- Line 6: -- deklaracja portu wejsciowego 'A2'
- Line 7: -- deklaracja portu wyjsciowego 'Y'
- Line 8: -- zakonczenie deklaracji naglowka
- Line 10: -- deklaracja ciala 'cialo' architektury
- Line 11: -- poczatek czesci wykonawczej
- Line 14: -- wywolanie operatora '+' oraz funkcji 'sumator'
- Line 16: -- zakonczenie deklaracji ciala 'cialo'

Przykład użycia oprogramowania ISE – projekt: „library_sum”

Plik źródłowy „library_sum.vhd”



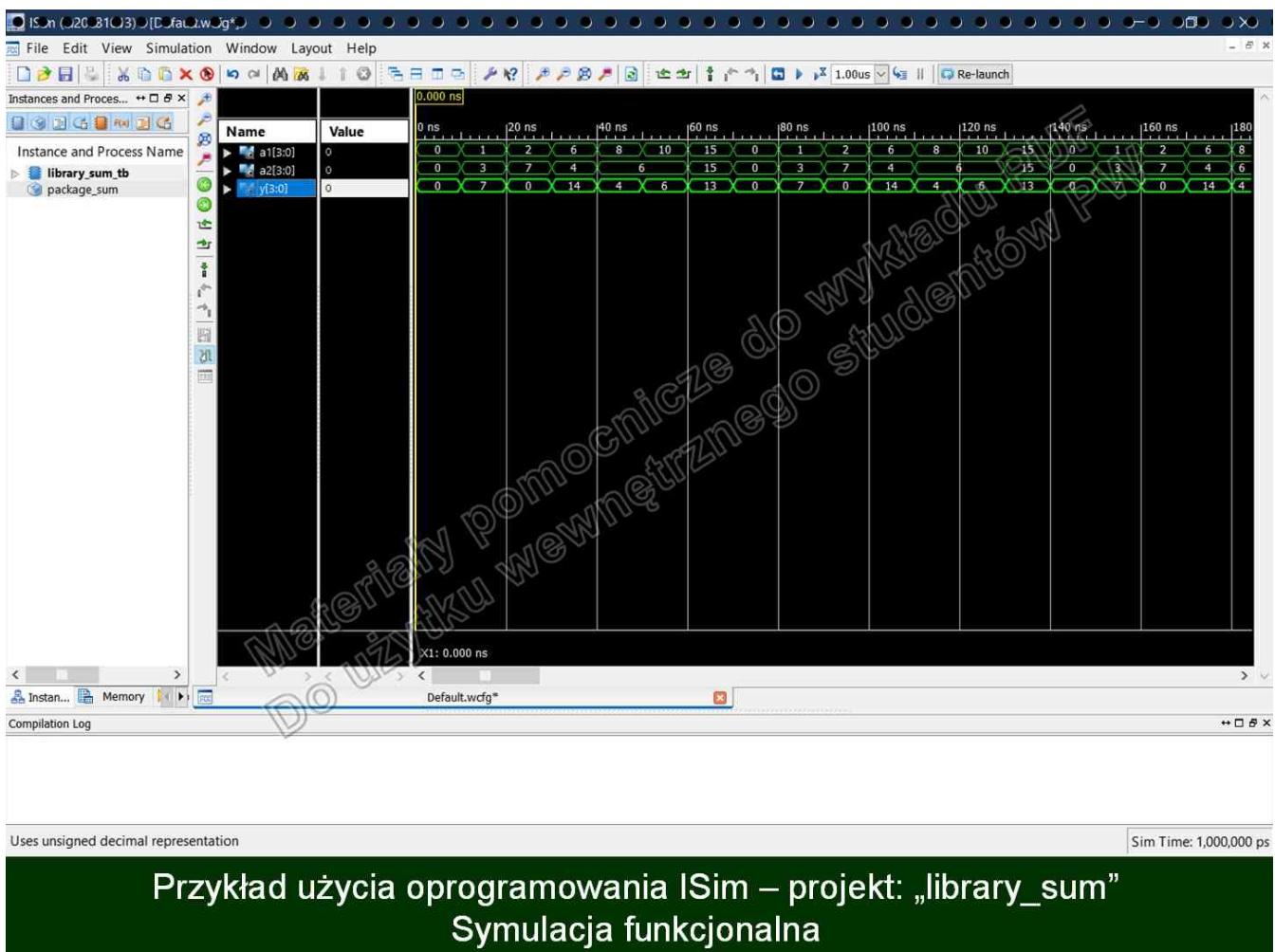
Przykład użycia oprogramowania ISE – projekt: „library_sum”

Plik źródłowy „library_sum_TB.vhd”

```

1 entity LIBRARY_SUM_TB is -- pusty szkielet projektu symulacji
2 end LIBRARY_SUM_TB;
3
4 architecture behavioural of LIBRARY_SUM_TB is -- ciało architektoniczne projektu
5
6 signal A1 : bit_vector(3 downto 0); -- deklaracja portu wejściowego 'A1'
7 signal A2 : bit_vector(3 downto 0); -- deklaracja portu wejściowego 'A2'
8 signal Y : bit_vector(3 downto 0); -- deklaracja portu wyjściowego 'YA'
9
10 begin -- część wykonawcza ciała projektu
11 process is -- proces bezwarunkowy
12 begin -- część wykonawcza procesu
13 A1 <= "0000"; A2 <= "0000"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2' i oczekanie 10 n
14 A1 <= "0001"; A2 <= "0011"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2' i oczekanie 10 n
15 A1 <= "0010"; A2 <= "0111"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2' i oczekanie 10 n
16 A1 <= "0110"; A2 <= "0100"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2' i oczekanie 10 n
17 A1 <= "1000"; A2 <= "0110"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2' i oczekanie 10 n
18 A1 <= "1010"; A2 <= "0111"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2' i oczekanie 10 n
19 A1 <= "1111"; A2 <= "1111"; wait for 10ns; -- ustawienie sygnałów 'A1', 'A2' i oczekanie 10 n
20 end process; -- zakończenie procesu
21
22 library_sum inst: entity work.LIBRARY_SUM(ciało) -- instancja projektu 'LIBRARY_SUM'
23 port map (A1, A2, Y); -- przypisanie portom sygnałów
24
25 end behavioural; -- zakończenie ciała architektonicznego
26
27

```



Podstawowe elementy standardu VHDL

Biblioteka STD – wybrane predefiniowane typy

```
type integer      is range -2147483647 to 2147483647;    -- definicja liczby całkowitej
subtype natural   is integer range 0 to integer'high;       -- definicja liczby naturalnej
subtype positive  is integer range 1 to integer'high;       -- definicja liczby dodatniej
type real         is range -1.0E308 to 1.0E308;           -- definicja liczby rzeczywistej
type boolean      is (false, true);                      -- definicja typu logicznego
type character    is (nul, ..., '0', '1', ..., '@', 'A', ...); -- def. znaków (ISO 8859-2)
type string        is array (positive range <>) of character; -- definicja ciągu znaków
type bit          is ('0', '1');                         -- def. typu sygnału logicznego
type bit_vector   is array (natural range <>) of bit;     -- definicja ciągu bitów (słowo)
type time         is range -9223372036854775807 to 9223372036854775807

units  -- definicja podstawy czasu
  fs;
  ps = 1000 fs;
  ...
  hr = 60 min;
end units;
```

Podstawowe elementy standardu VHDL

Biblioteka STD – wybrane predefiniowane operatory

- Dwuargumentowe operatory relacji:
 - a = b : operator zwraca TRUE gdy a jest takie same jak b
 - a /= b : operator zwraca TRUE gdy a jest różne od b
 - a < b : operator zwraca TRUE gdy a jest mniejsze od b
 - a <= b : operator zwraca TRUE gdy a jest mniejsze lub równe b
 - a > b : operator zwraca TRUE gdy a jest większe od b
 - a >= b : operator zwraca TRUE gdy a jest większe lub równe b
- Dwuargumentowe operatory arytmetyczne:
 - a + b : operacja dodawania a i b
 - a - b : operacja odejmowania b od a
 - a * b : operacja mnożenia a i b
 - a / b : operacja dzielenia a przez b
 - a mod b : operacja dzielenia modulo a przez b
 - a rem b : operacja reszty z dzielenia modulo a przez b
 - a ** b : operacja potęgowania a do b

Podstawowe elementy standardu VHDL

Biblioteka STD – wybrane predefiniowane operatory

- Jednoargumentowe operatory arytmetyczne:

- a : operator zwraca wartość ujemną a
- + a : operator zwraca wartość a
- abs a : operator zwraca bezwzględną wartość a

- Dwuargumentowy operator łączenia:

- a & b : operacja łączenia (sklejania) a i b

- Dwuargumentowe operatory przesuwania:

- a sll n : operacja przesunięcia logicznego a w lewo o n pozycji
- a srl n : operacja przesunięcia logicznego a w prawo o n pozycji
- a sla n : operacja przesunięcia arytmetycz. a w lewo o n pozycji
- a sra n : operacja przesunięcia arytmetycz. a w prawo o n pozycji
- a rol n : operacja obrotu logicznego a w lewo o n pozycji
- a ror n : operacja obrotu logicznego a w prawo o n pozycji

Podstawowe elementy standardu VHDL

Biblioteka IEEE – wybrane pakiety podstawowe

- pakiet **std_logic_1164** – podstawowe typy i operacje logiczne

- type std_logic is ('U',
 'X',
 '0',
 '1',
 'Z',
 'W',
 'L',
 'H',
 '-');
 -- wartość nigdy dotychczas nie została określona
 -- silnie wysterowany sygnał nieokreślonej wartości
 -- silnie wysterowany sygnał logiczny 0
 -- silnie wysterowany sygnał logiczny 1
 -- stan wysokiej impedancji
 -- słabo wysterowany sygnał nieokreślonej wartości
 -- słabo wysterowany sygnał logiczny 0
 -- słabo wysterowany sygnał logiczny 1
 -- wartość sygnału nie ma znaczenia

- type std_logic_vector is

- operacje logiczne: and



--	u	x	0	1	z	w	l	h	--	--
('u',	'u',	'0',	'u',	'u',	'0',	'u',	'u',	--	u
)	'u',	'x',	'0',	'x',	'x',	'0',	'x',	'x',	--	x
('0',	'0',	'0',	'0',	'0',	'0',	'0',	'0',	--	0
)	'u',	'x',	'0',	'1',	'x',	'x',	'0',	'1',	--	1
('u',	'x',	'0',	'x',	'x',	'x',	'0',	'x',	--	z
)	'u',	'x',	'0',	'x',	'x',	'x',	'0',	'x',	--	w
('0',	'0',	'0',	'0',	'0',	'0',	'0',	'0',	--	l
)	'u',	'x',	'0',	'1',	'x',	'x',	'0',	'1',	--	h
('u',	'x',	'0',	'x',	'x',	'x',	'0',	'x',	--	-

Materiały pomocnicze do wykładu PUF
Do użytku wewnętrznego studentów PW

```

1 library ieee;
2 use ieee.std_logic_1164.all;
-- klauzula dostępu do biblioteki 'IEEE'
-- dodanie całego pakietu 'STD_LOGIC_1164'

3
4 entity IEEE_AND is
5 port (
6   I1 : in std_logic;
7   I2 : in std_logic;
8   O : out std_logic
9 );
10 end IEEE_AND;
11
12 architecture cialo of IEEE_AND is
13 begin
14
15   O <= I1 and I2;
16
17 end cialo;
18

```

Design View Implement Sim

Hierarchy

- ieee_and
 - xc3s50a-5tq144
 - IEEE_AND - cialo

No Processes Running

Processes: IEEE_AND - cialo

- Design Summary/...
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Program...
- Configure Target ...
- Analyze Design Us...

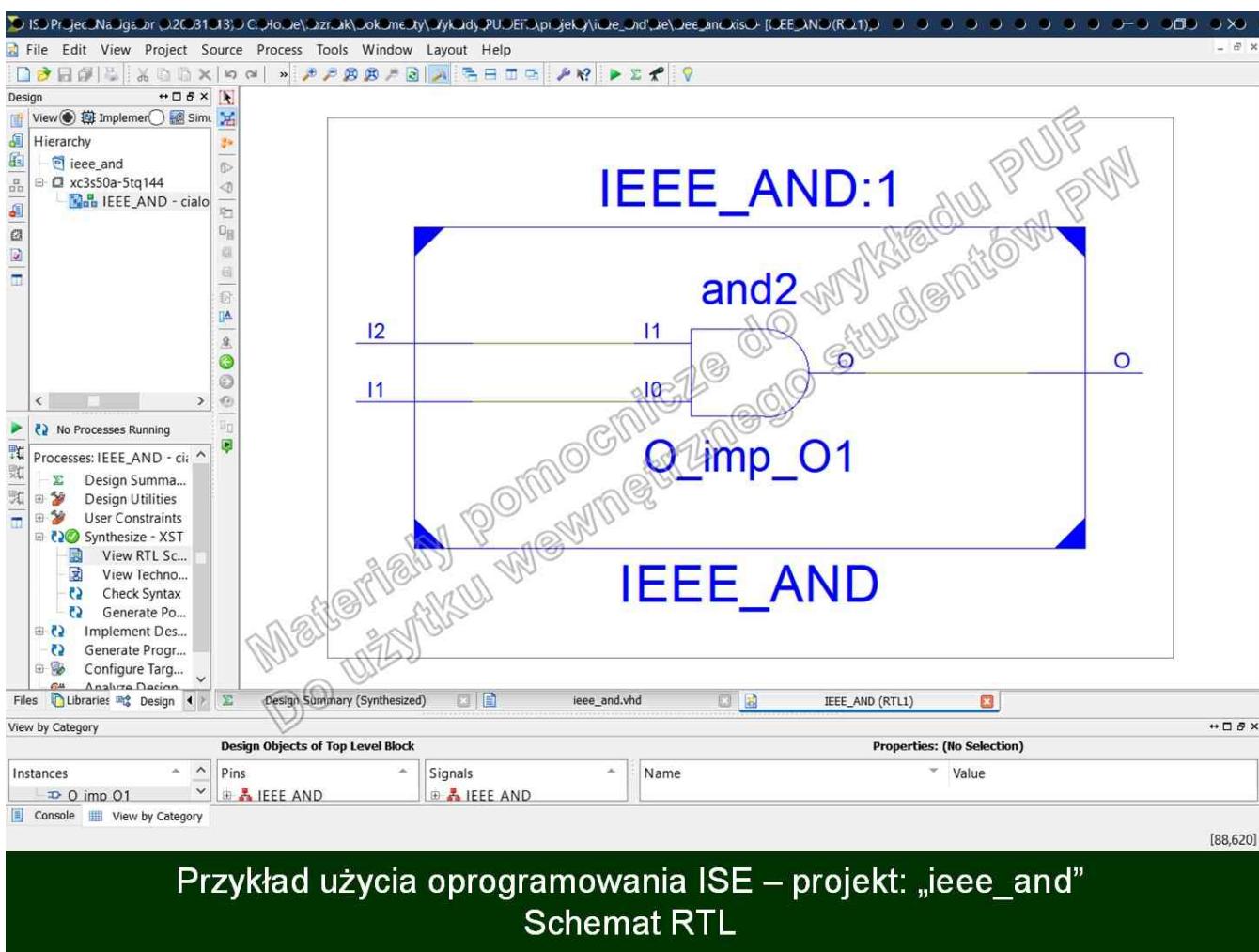
Files Libraries Design Design Summary ieee_and.vhd

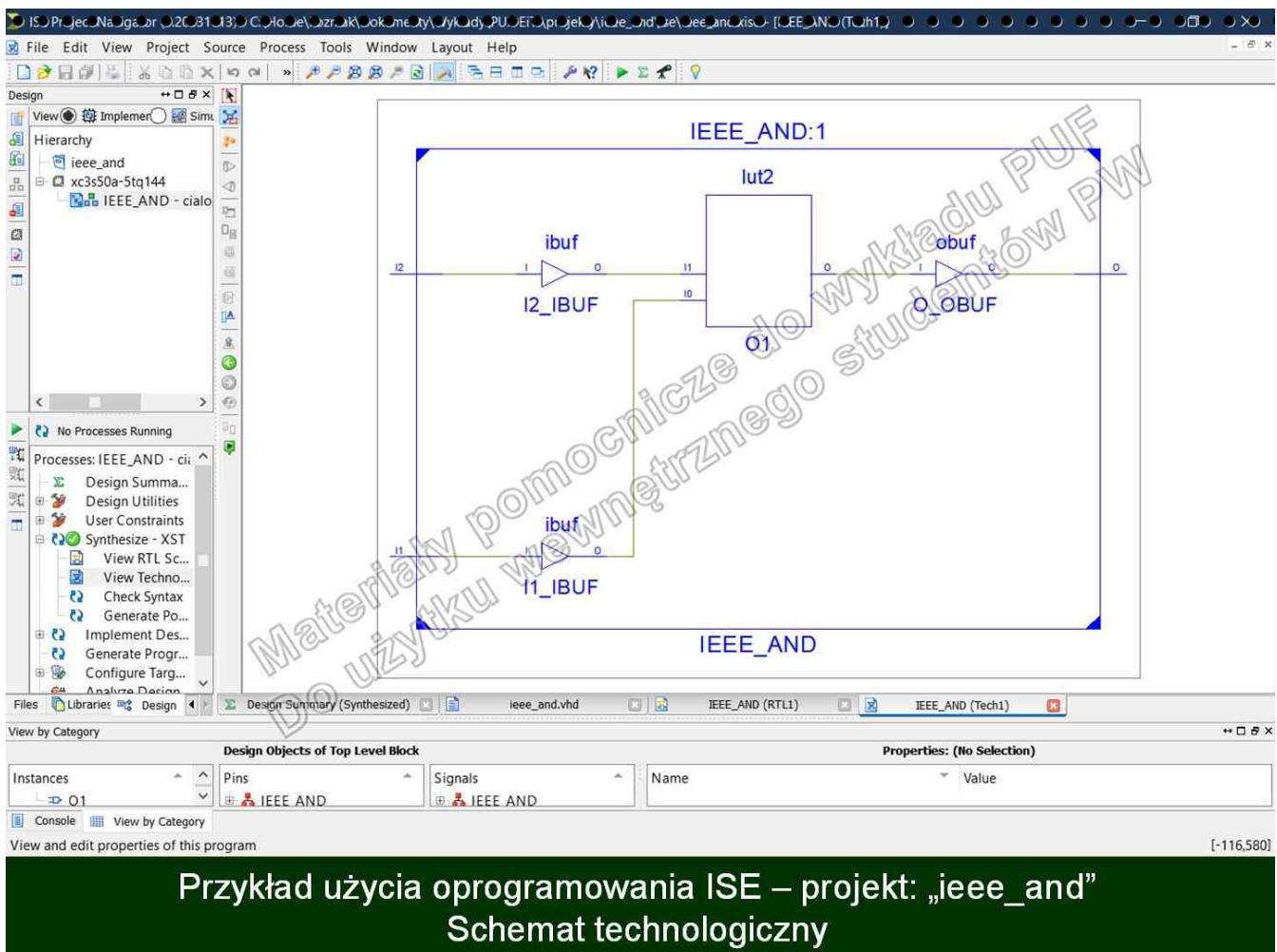
Console

Started : "Launching ISE Text Editor to edit ieee_and.vhd".

Ln 12 Col 1 | VHDL

**Przykład użycia oprogramowania ISE – projekt: „ieee_and”
Plik źródłowy „ieee_and.vhd”**





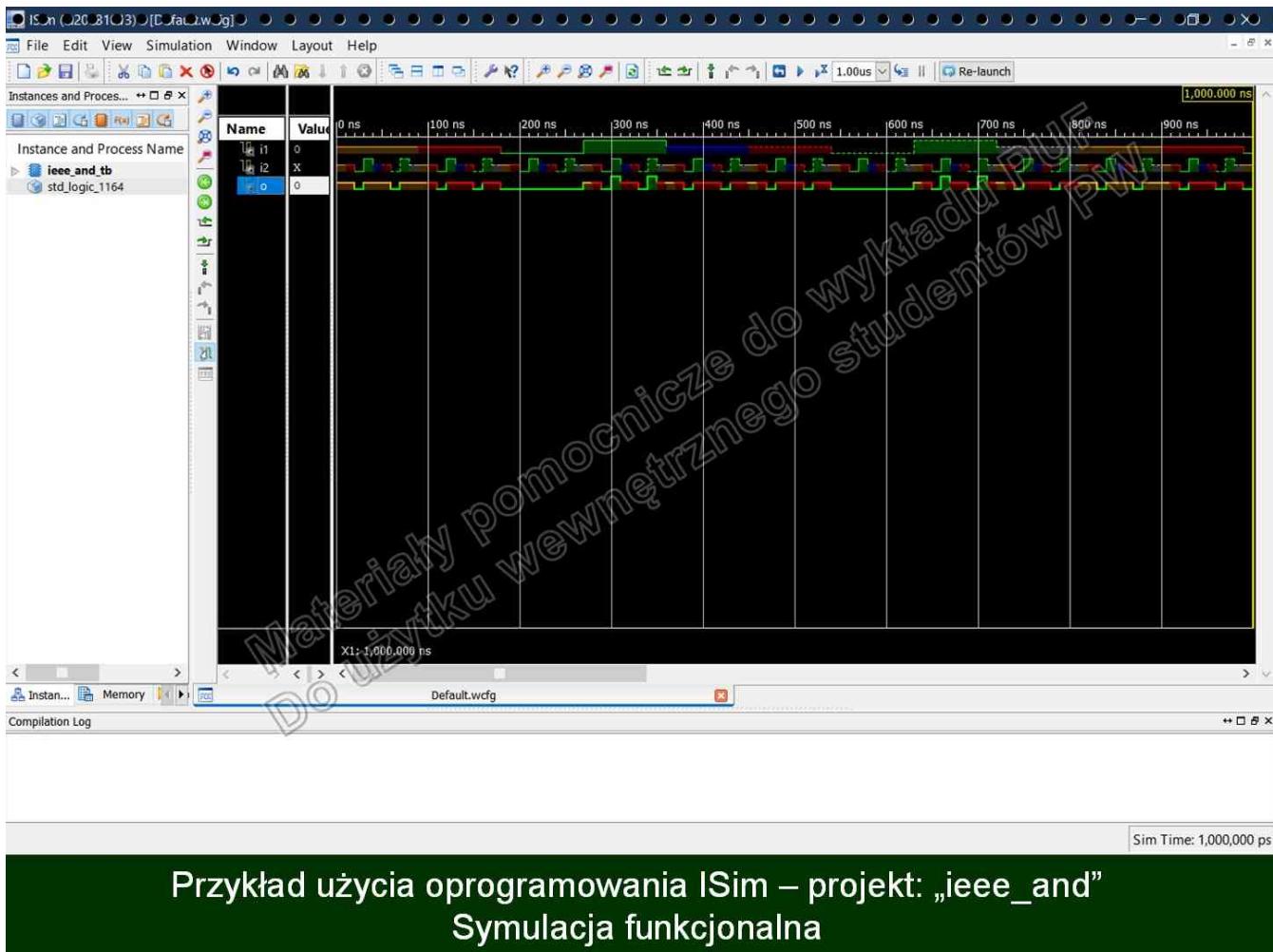
The screenshot shows the Xilinx ISE Design Suite interface with the VHDL code for the 'IEEE_AND_TB' testbench. The code is as follows:

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity IEEE_AND_TB is
5 end IEEE_AND_TB;
6
7 architecture behavioural of IEEE_AND_TB is -- cialo architektoniczne projektu
8
9 signal I1 : std_logic; -- symulowane wejście 'I1'
10 signal I2 : std_logic; -- symulowane wejście 'I2'
11 signal O : std_logic; -- obserwowane wyjście 'O'
12
13 begin
14
15 process is
16 begin
17   for f1 in std_logic loop
18     I1 <= f1;
19     for f2 in std_logic loop
20       I2 <= f2;
21       wait for 10 ns;
22     end loop;
23   end loop;
24 end process;
25
26 ieee_and_inst: entity work.IEEE_AND(cialo) -- instancja projektu 'IEEE_AND'
27 port map (
28   I1 => I1,
29   I2 => I2,
30   O => O
31 );
32 end behavioural;

```

Przykład użycia oprogramowania ISE – projekt: „ieee_and”
Plik źródłowy „ieee_and_TB.vhd”



Podstawowe elementy standardu VHDL

Biblioteka IEEE – wybrane pakiety podstawowe

- pakiet **std_logic_1164** – podstawowe typy i operacje logiczne
 - type std_logic is ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-'); -- wartość nigdy dotychczas nie została określona
-- silnie wysterowany sygnał nieokreślonej wartości
-- silnie wysterowany sygnał logiczny 0
-- silnie wysterowany sygnał logiczny 1
-- stan wysokiej impedancji
-- słabo wysterowany sygnał nieokreślonej wartości
-- słabo wysterowany sygnał logiczny 0
-- słabo wysterowany sygnał logiczny 1
-- wartość sygnału nie ma znaczenia
 - type std_logic_vector is array (natural range <>) of std_logic;
 - operacje logiczne: and , or , xor , nand , nor , xnor , not



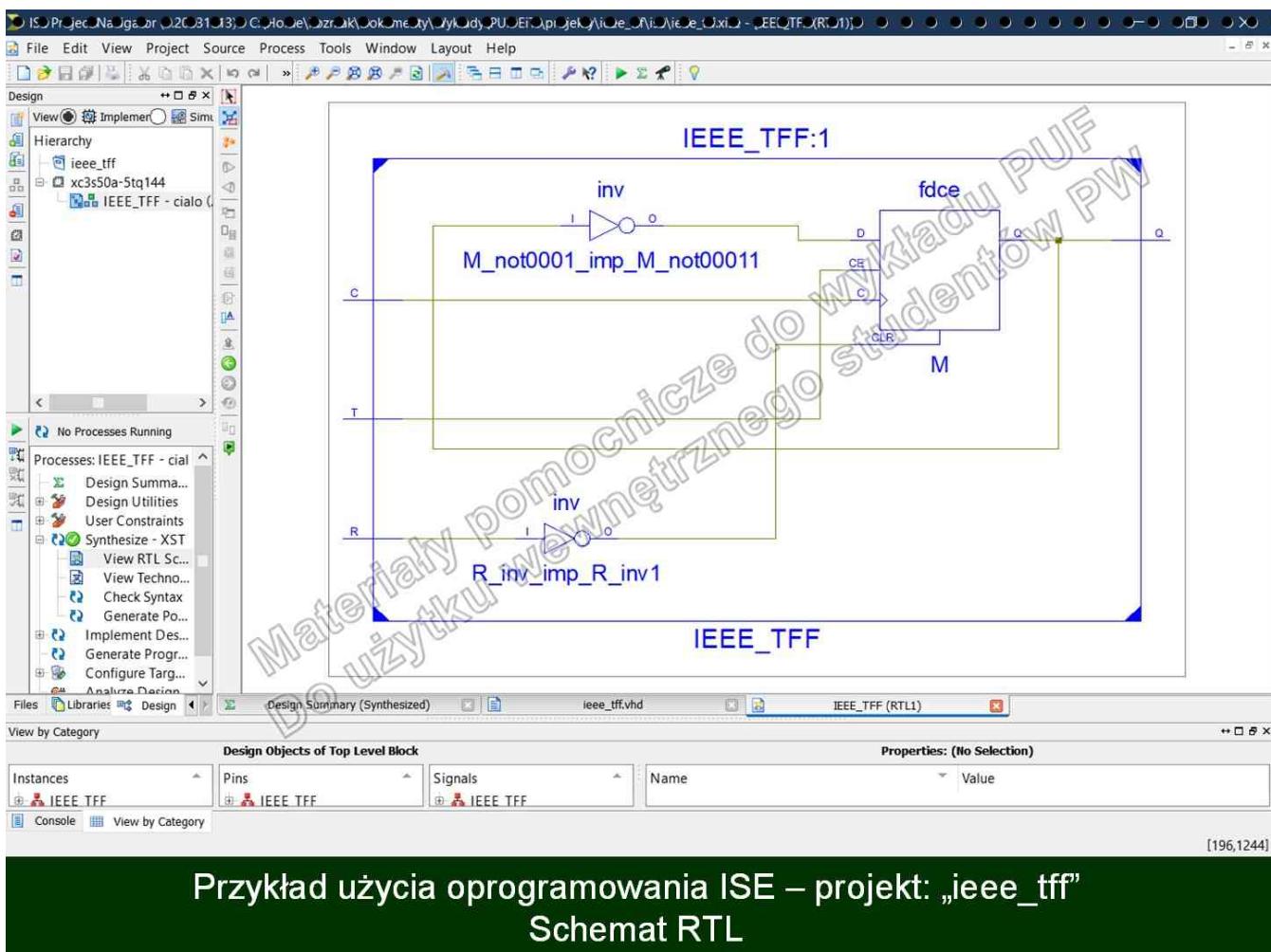
Do użytku wewnętrznego

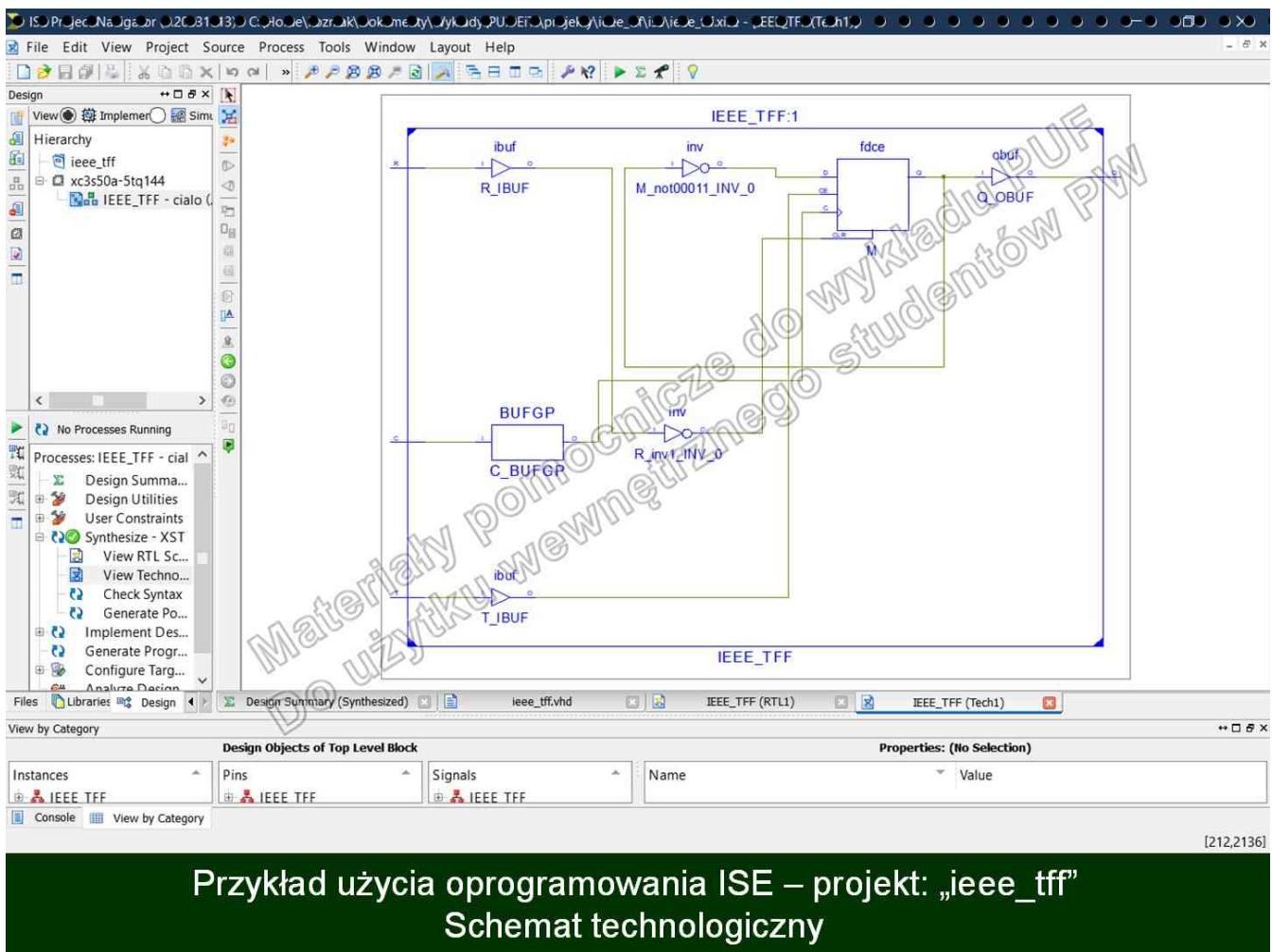
Przykład użycia oprogramowania ISE – projekt: „ieee_tff”
Plik źródłowy „ieee_tff.vhd”

```

1 library ieee; -- klauzula dostępu do biblioteki 'IEEE'
2 use ieee.std_logic_1164.all; -- dodanie całego pakietu 'STD_LOGIC_1164'
3
4 entity IEEE_TFF is -- deklaracja sprzegu IEEE_TFF
5   port (
6     R : in std_logic; -- wejście kasujące 'R'
7     C : in std_logic; -- wejście zegarowe 'C'
8     T : in std_logic; -- wejście sterujące 'T'
9     Q : out std_logic; -- wyjście danych 'Q'
10    );
11 end IEEE_TFF;
12
13 architecture cialo of IEEE_TFF is -- deklaracja ciała 'cialo' architektury
14
15   signal M : std_logic; -- element pamiętający na bazie sygnału 'M'
16
17 begin
18
19   process (R,C) is -- lista części procesu
20   begin
21     if (R='0') then -- czasie wykonawcza procesu
22       M <= '0'; -- warunek kasowania dla sygnału 'R='0'
23     elsif (C'event and C='1') then -- przypisanie stałej '0' do sygnału 'R'
24       if (T='1') then -- warunek dla zboocznej narastającej 'C'
25         M <= not(M); -- przypisanie sygnału zezwolenia ;T='1'
26       end if; -- przypisanie sygnału zanegowanego 'M'
27     end if;
28   end process; -- zakonczenie instrukcji wyboru
29   Q <= M; -- zakonczenie procesu
30
31 end cialo; -- przypisanie do portu 'Q' sygnału 'M'
32
33 end IEEE_TFF; -- zakonczenie deklaracji ciała 'cialo'

```





Przykład użycia oprogramowania ISE – projekt: „ieee_tff” Schemat technologiczny

The screenshot shows the Xilinx ISE Design Suite interface with the VHDL code for the **IEEE_TFF_TB** test bench. The code defines an entity **IEEE_TFF_TB** with four signals: **R**, **C**, **T**, and **Q**. It contains three processes: one for setting **R** and **C**, another for setting **T**, and a third for reading **Q**. An instance of the **IEEE_TFF** component is mapped to port map **ieee_tff_inst**.

```

library ieee;
use ieee.std_logic_1164.all;

entity IEEE_TFF_TB is
end IEEE_TFF_TB;

architecture behavioural of IEEE_TFF_TB is
begin
process is
begin
  C <= '0'; wait for 5 ns;
  C <= '1'; wait for 5 ns;
end process;

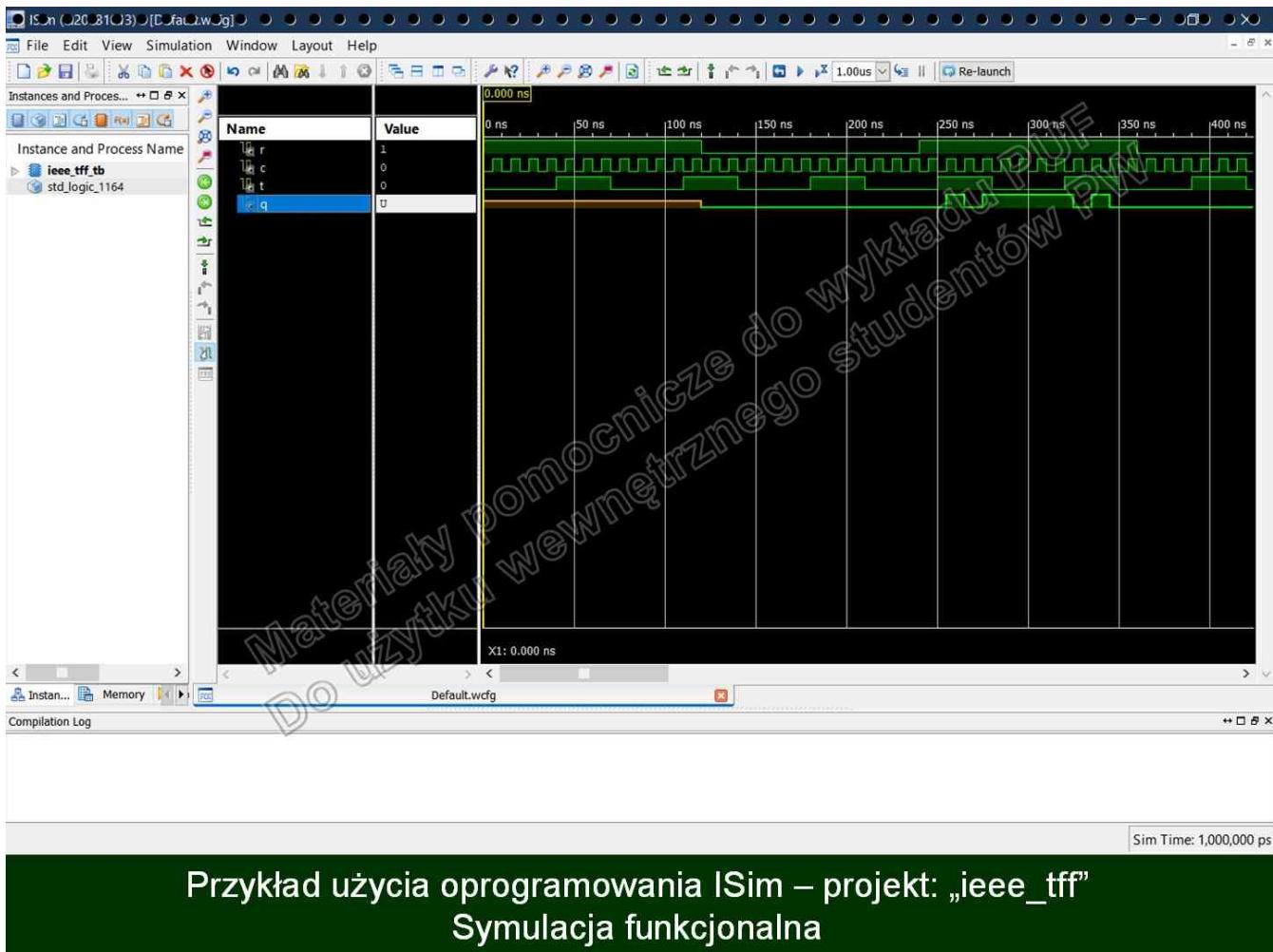
process is
begin
  T <= '0'; wait for 40 ns;
  T <= '1'; wait for 30 ns;
end process;

process is
begin
  R <= '1'; wait for 120 ns;
  R <= '0'; wait for 120 ns;
end process;

ieee_tff_inst: entity work.IEEE_TFF(cialo) port map(
  R => R,
  C => C,
  T => T,
  Q => Q
);
end behavioural;

```

Przykład użycia oprogramowania ISE – projekt: „ieee_tff” Plik źródłowy „ieee_tff_TB.vhd”



Podstawowe elementy standardu VHDL

Biblioteka IEEE – wybrane pakiety podstawowe

- pakiet **std_logic_1164** – podstawowe typy i operacje logiczne
 - type std_logic is ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-'); -- wartość nigdy dotychczas nie została określona
-- silnie wysterowany sygnał nieokreślonej wartości
-- silnie wysterowany sygnał logiczny 0
-- silnie wysterowany sygnał logiczny 1
-- stan wysokiej impedancji
-- słabo wysterowany sygnał nieokreślonej wartości
-- słabo wysterowany sygnał logiczny 0
-- słabo wysterowany sygnał logiczny 1
-- wartość sygnału nie ma znaczenia
 - type std_logic_vector is array (natural range <>) of std_logic;
 - operacje logiczne: and, or, xor, nand, nor, xnor, not
 - funkcje taktujące: rising_edge, falling_edge
 - funkcje konwertujące: To_BitVector, To_StdLogicVector

Podstawowe elementy standardu VHDL

Biblioteka IEEE – wybrane pakiety podstawowe

- pakiet **std_logic_1164** – podstawowe typy i operacje logiczne
- pakiet **std_logic_unsigned** – operacje na naturalnej liczbie binarnej (NB)
 - operacje arytmetyczne: +, -, *
 - operacje relacji: =, /=, >, >=, <, <=
 - funkcje konwertujące: CONV_INTEGER
 - funkcje przesuwające: SHL, SHR

Podstawowe elementy standardu VHDL

Biblioteka IEEE – wybrane pakiety podstawowe

- pakiet **std_logic_1164** – podstawowe typy i operacje logiczne
- pakiet **std_logic_unsigned** – operacje na naturalnej liczbie binarnej (NB)
- pakiet **std_logic_signed** – operacje na liczbie binarnej ze znakiem (U2)
 - operacje arytmetyczne: +, -, *
 - operacje relacji: =, /=, >, >=, <, <=
 - funkcje konwertujące: CONV_INTEGER
 - funkcje przesuwające: SHL, SHR

Podstawowe elementy standardu VHDL

Biblioteka IEEE – wybrane pakiety podstawowe

- pakiet **std_logic_1164** – podstawowe typy i operacje logiczne
- pakiet **std_logic_unsigned** – operacje na naturalnej liczbie binarnej (NB)
- pakiet **std_logic_signed** – operacje na liczbie binarnej ze znakiem (U2)
- pakiet **numeric_std** – podstawowe typy arytmetyczne z operacjami
 - typy arytmetyczne: SIGNED, UNSIGNED – podtypy std_logic_vector
 - operacje arytmetyczne: +, -, *, /, abs, rem, mod
 - operacje relacji: =, /=, >, >=, <, <=

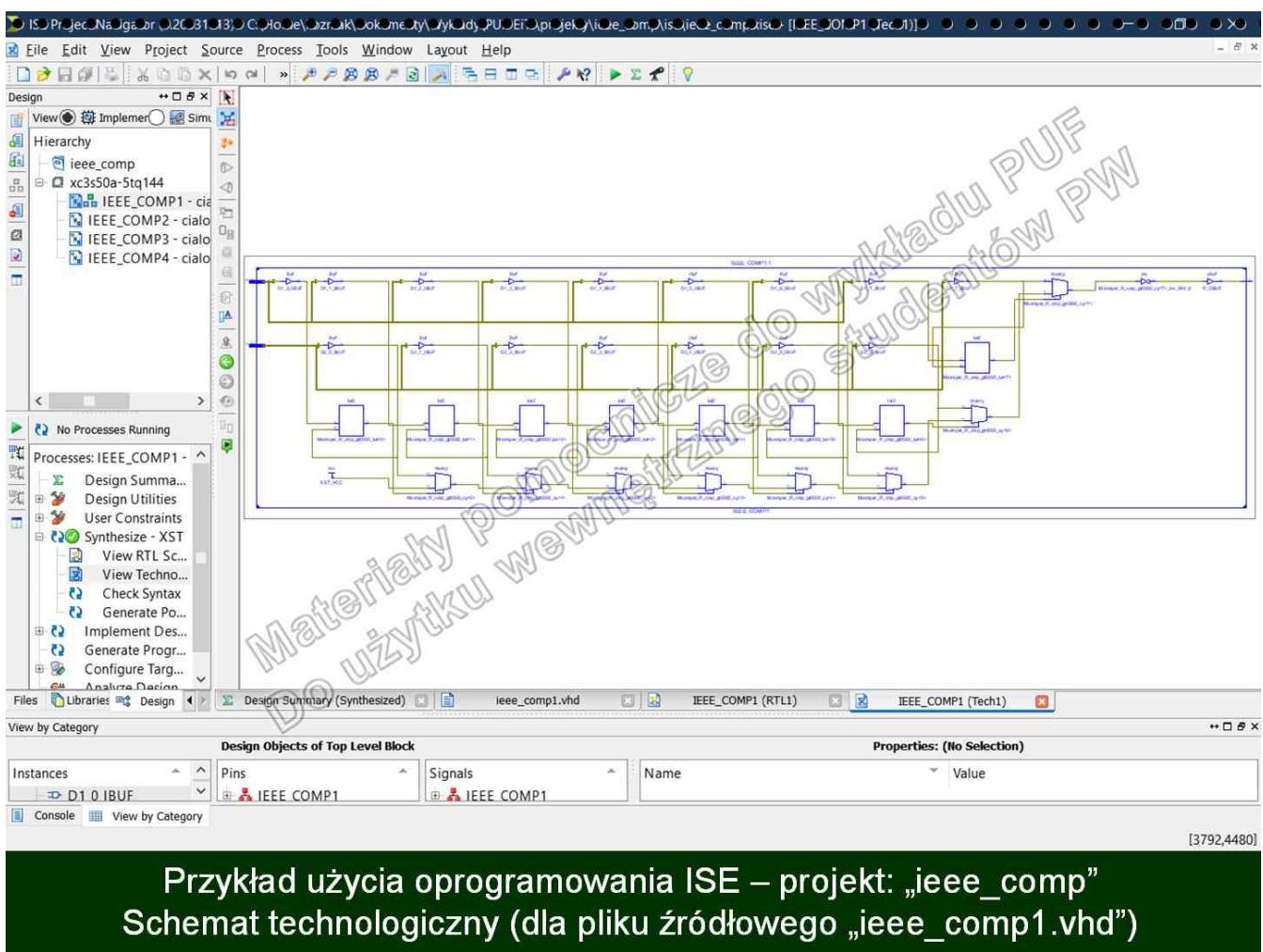
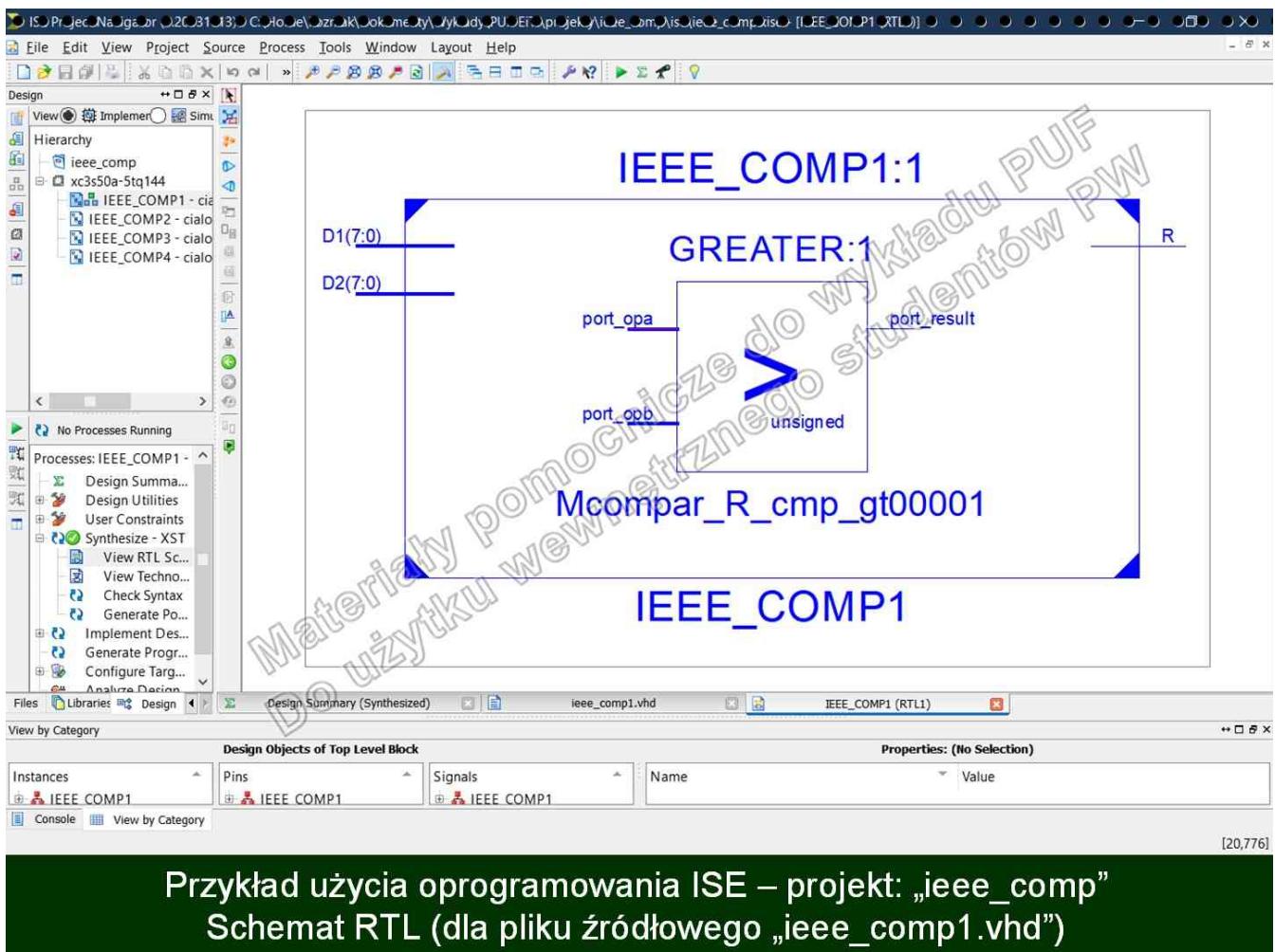


The screenshot shows the Xilinx ISE (Integrated Software Environment) interface. On the left, the 'Hierarchy' panel displays a project structure under 'ieee_comp'. The main workspace shows a VHDL source file named 'ieee_comp1.vhd'. The code is as follows:

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4
5 entity IEEE_COMP1 is
6     port (
7         D1      : in  std_logic_vector(7 downto 0);
8         D2      : in  std_logic_vector(7 downto 0);
9         R       : out std_logic
10    );
11 end IEEE_COMP1;
12
13 architecture cialo of IEEE_COMP1 is
14
15 begin
16
17     R <= '1' when (D1>D2) else '0';
18
19 end cialo;
```

The code is annotated with comments explaining its functionality. The 'Design' menu is open at the top, and the 'Console' window at the bottom shows successful compilation messages.

Przykład użycia oprogramowania ISE – projekt: „ieee_comp”
Plik źródłowy „ieee_comp1.vhd”



Materiały pomocnicze do wykładu PUF
Do użytku wewnętrznego

```

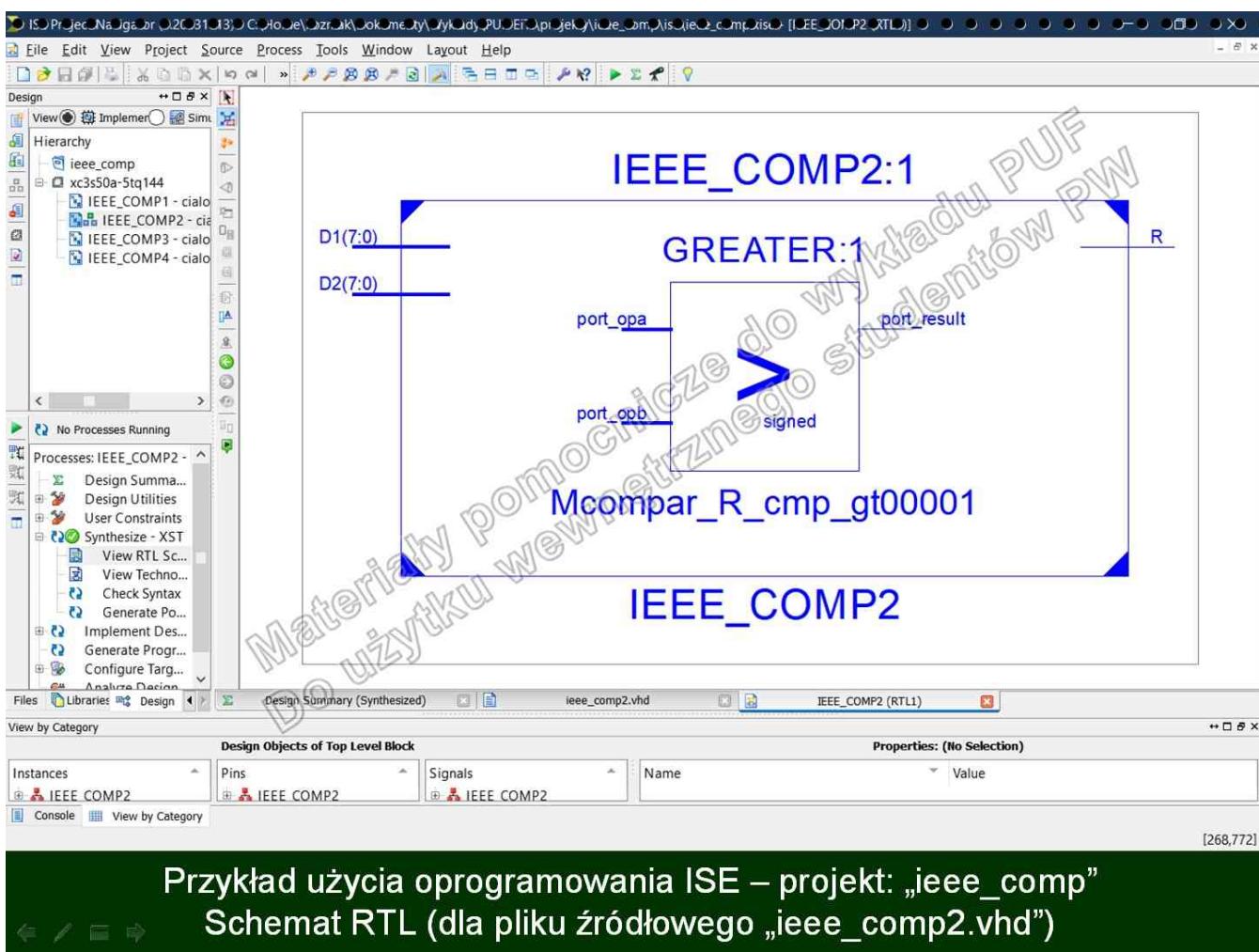
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_signed.all;
4
5 entity IEEE_COMP2 is
6 port (
7    D1 : in std_logic_vector(7 downto 0);
8    D2 : in std_logic_vector(7 downto 0);
9    R  : out std_logic
10 );
11 end IEEE_COMP2;
12
13 architecture cialo of IEEE_COMP2 is
14 begin
15 begin
16    R <= '1' when (D1>D2) else '0';
17 end cialo;
18
19 end;
20
21

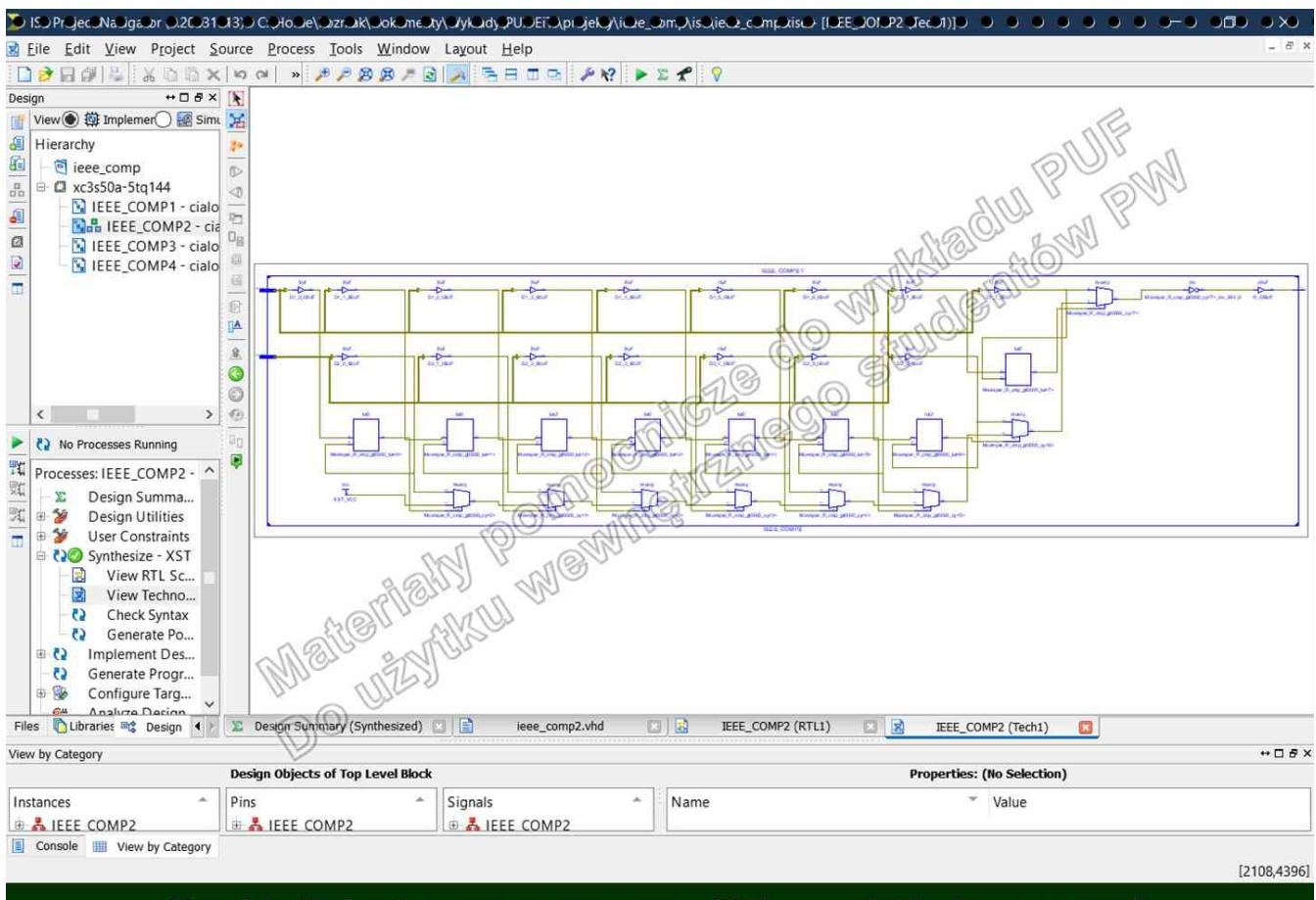
```

Started : "Launching ISE Text Editor to edit ieee_comp2.vhd".

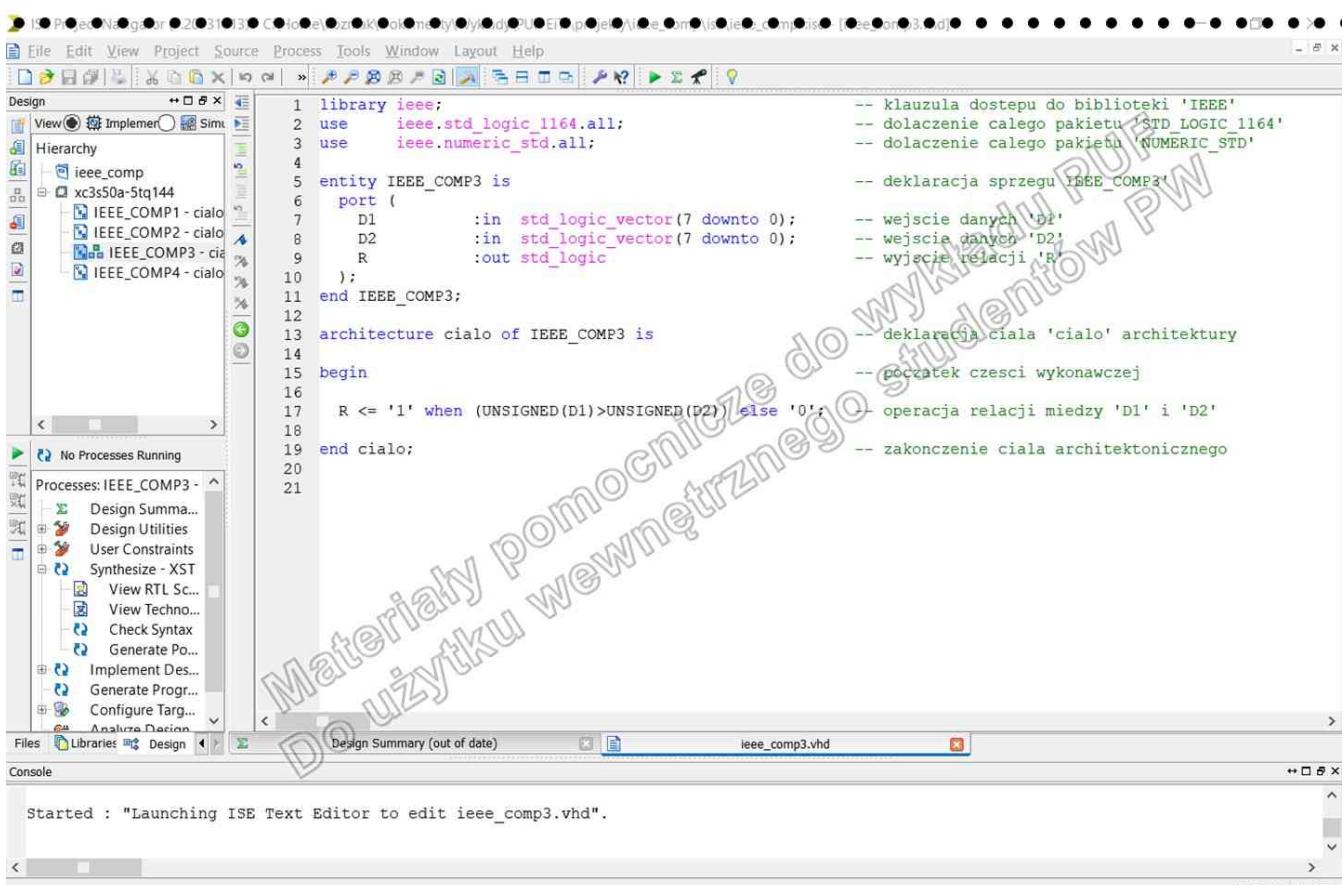
Ln 13 Col 1 | VHDL

**Przykład użycia oprogramowania ISE – projekt: „ieee_comp”
Plik źródłowy „ieee_comp2.vhd”**

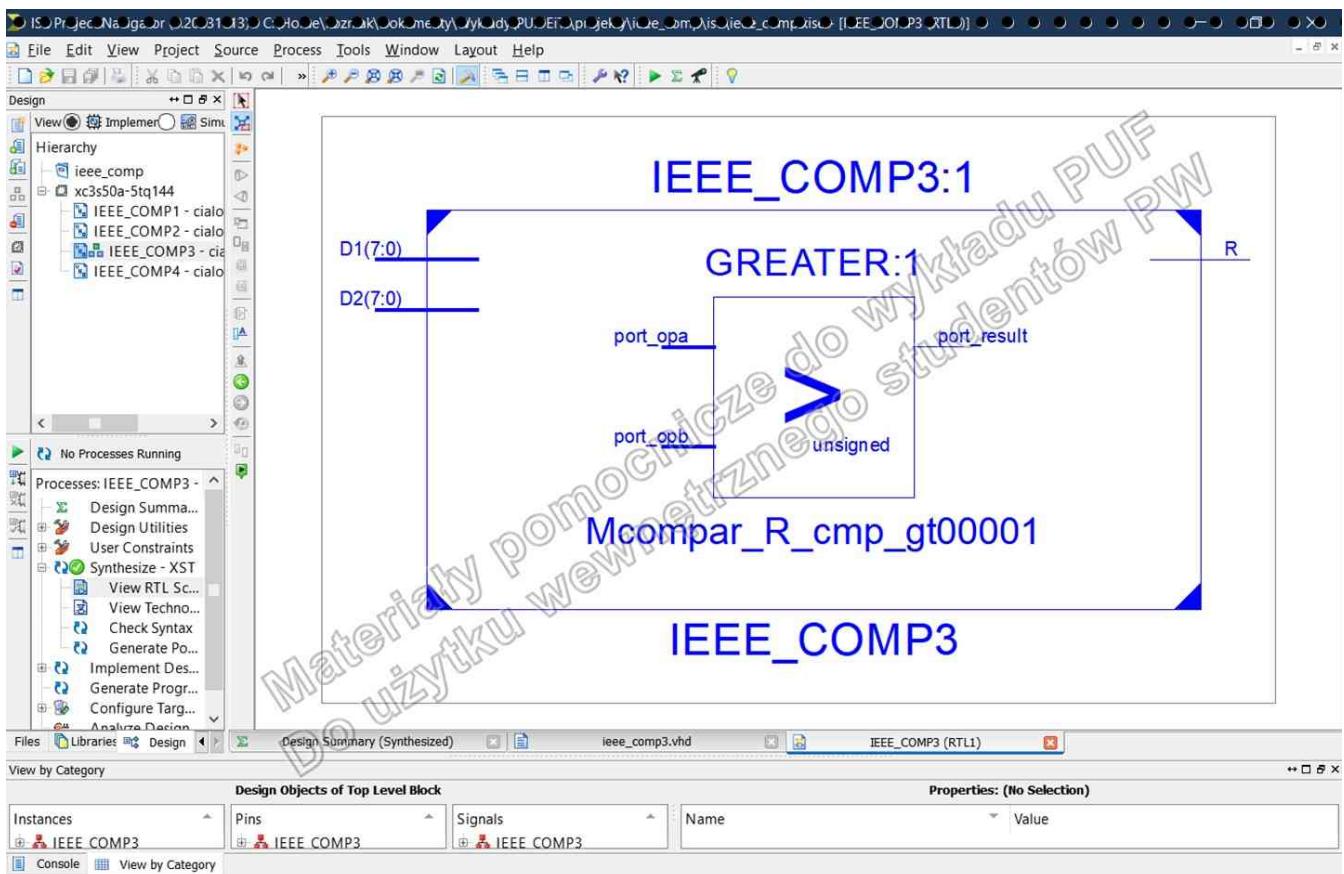




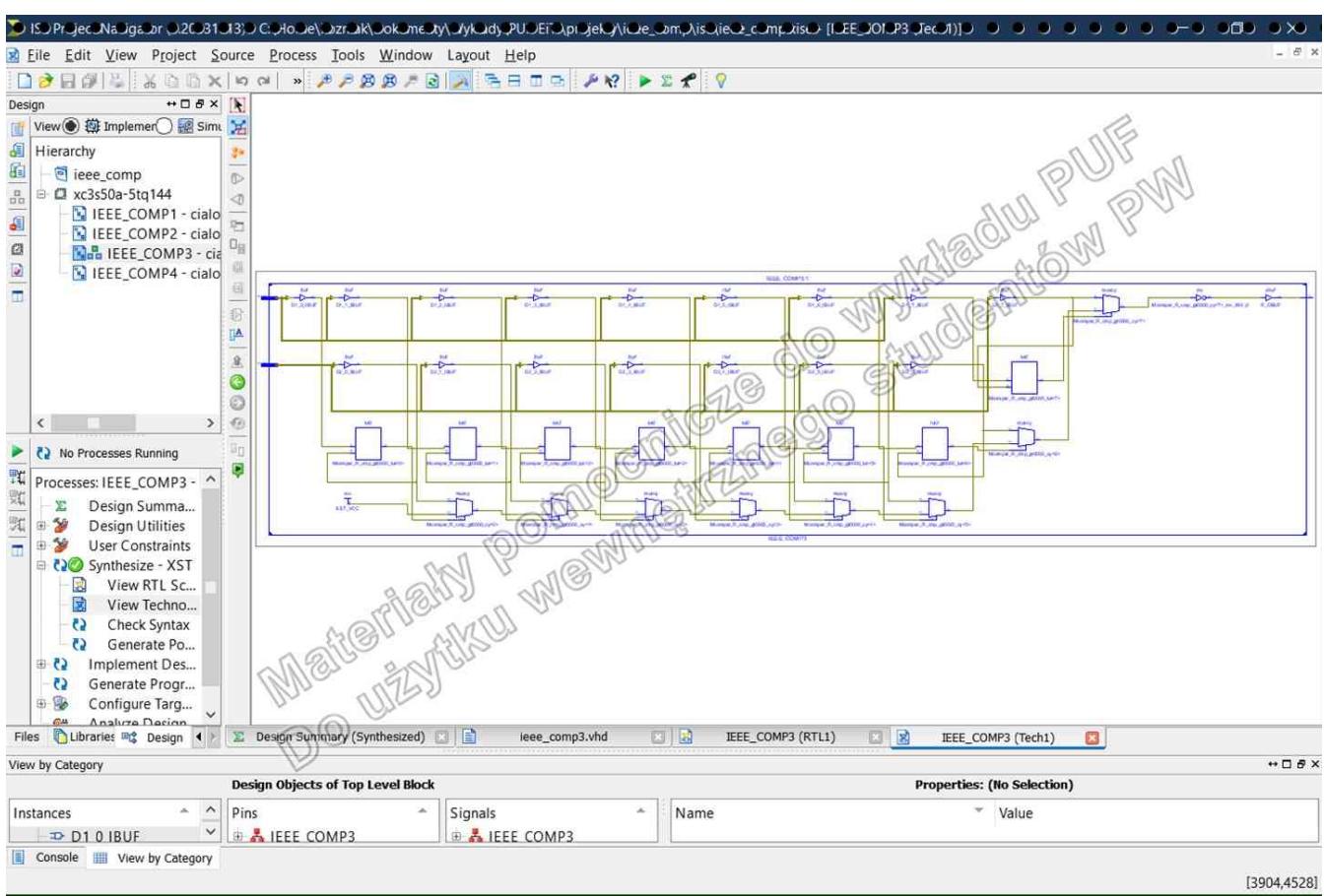
Przykład użycia oprogramowania ISE – projekt: „ieee_comp”
Schemat technologiczny (dla pliku źródłowego „ieee_comp2.vhd”)



Przykład użycia oprogramowania ISE – projekt: „ieee_comp”
Plik źródłowy „ieee_comp3.vhd”



Przykład użycia oprogramowania ISE – projekt: „ieee_comp”
Schemat RTL (dla pliku źródłowego „ieee_comp3.vhd”)



Przykład użycia oprogramowania ISE – projekt: „ieee_comp”
Schemat technologiczny (dla pliku źródłowego „ieee_comp3.vhd”)

Materiały pomocnicze do wykładu PUF
Do użytku wewnętrznego studentów PW

```

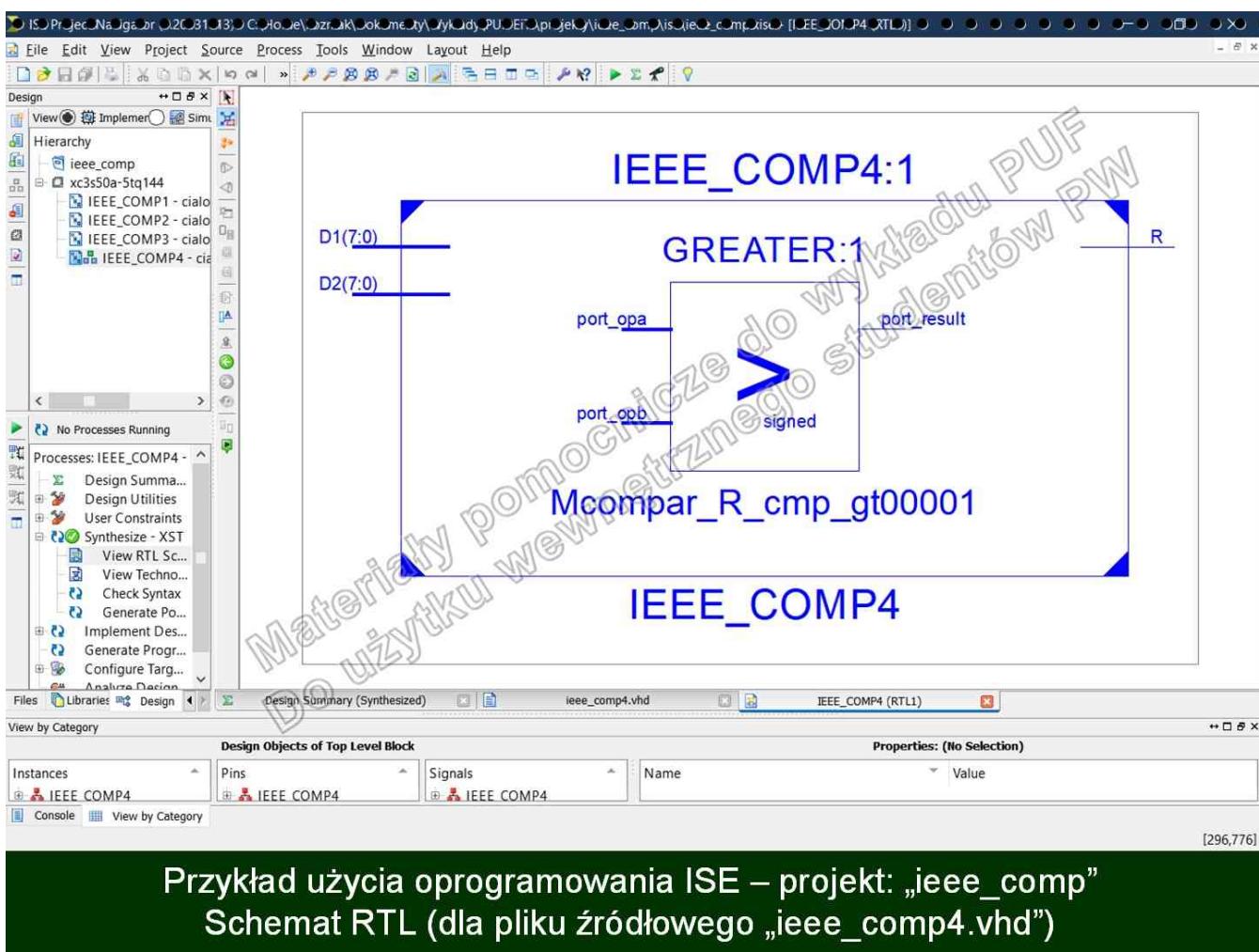
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity IEEE_COMP4 is
6 port (
7    D1      :in std_logic_vector(7 downto 0);
8    D2      :in std_logic_vector(7 downto 0);
9    R       :out std_logic
10 );
11 end IEEE_COMP4;
12
13 architecture cialo of IEEE_COMP4 is
14
15 begin
16
17    R <= '1' when (SIGNED(D1)>SIGNED(D2)) else '0';
18
19 end cialo;
20
21

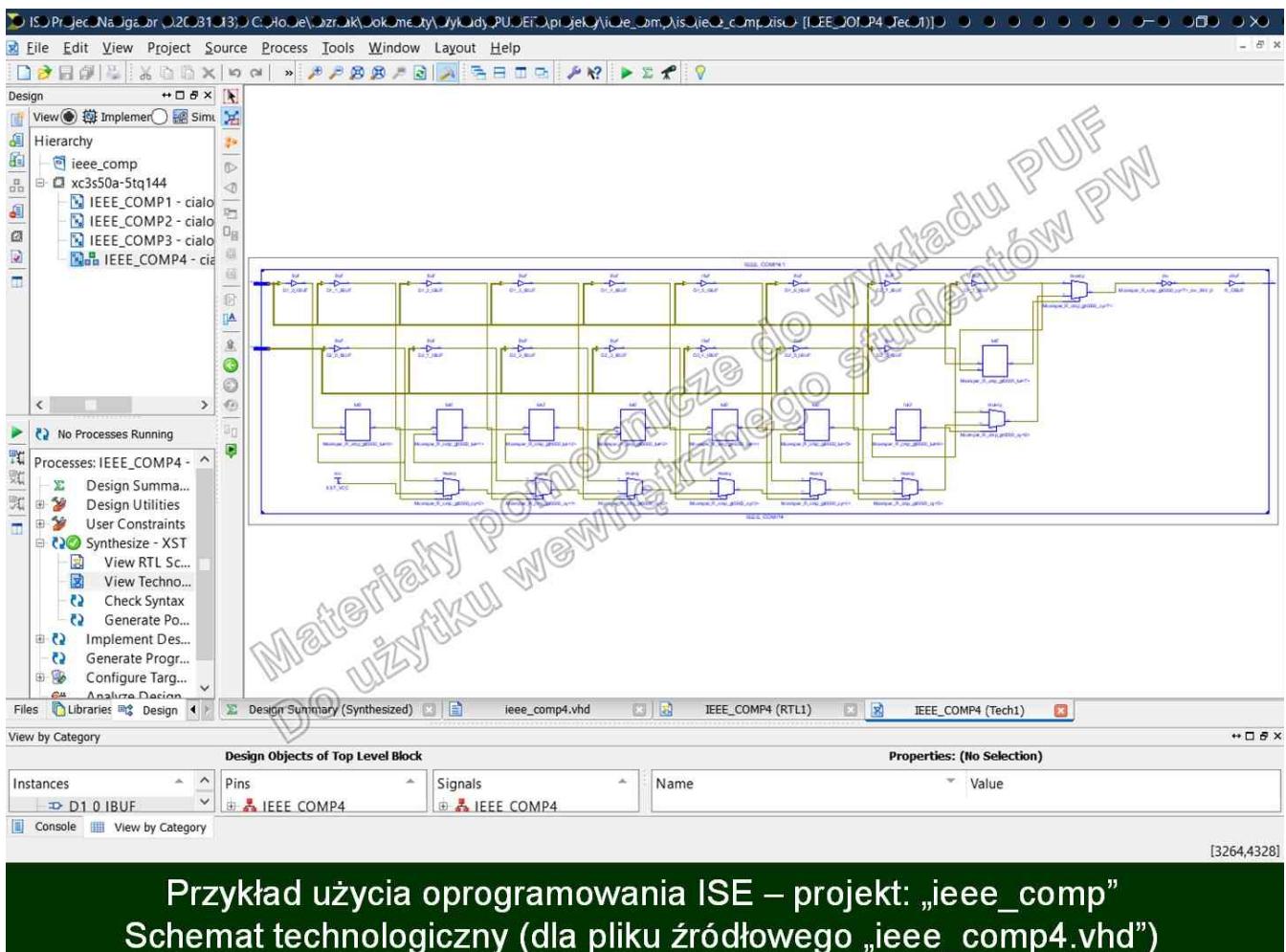
```

Started : "Launching ISE Text Editor to edit ieee_comp4.vhd".

Ln 13 Col 1 | VHDL

**Przykład użycia oprogramowania ISE – projekt: „ieee_comp”
Plik źródłowy „ieee_comp4.vhd”**





Przykład użycia oprogramowania ISE – projekt: „ieee_comp” Schemat technologiczny (dla pliku źródłowego „ieee_comp4.vhd”)

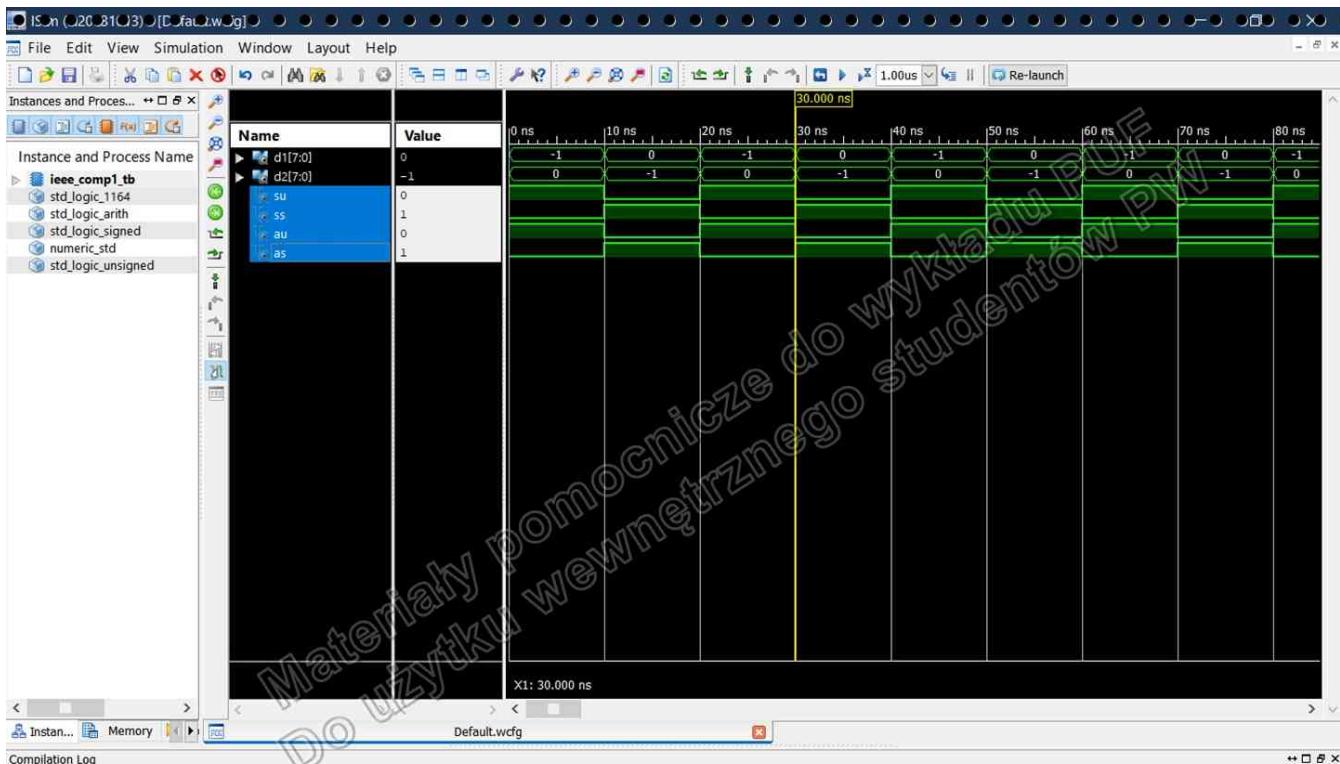
```

File Edit View Project Source Process Tools Window Layout Help
Design View Implement Sim
Hierarchy
  ieee_comp
    xc3s50a-5tq144
      IEEE_COMP1 - cialo
      IEEE_COMP2 - cialo
      IEEE_COMP3 - cialo
      IEEE_COMP4 - cialo
Processes: IEEE_COMP1_TB -
  ISim Simulator
    Behavioral Che...
    Simulate Behav...
Design Objects of Top Level Block
Properties: (No Selection)
Instances Pins Signals Name Value
D1 0 IBUF IEEE COMP4 IEEE COMP4
Console View by Category
[3264,4328]

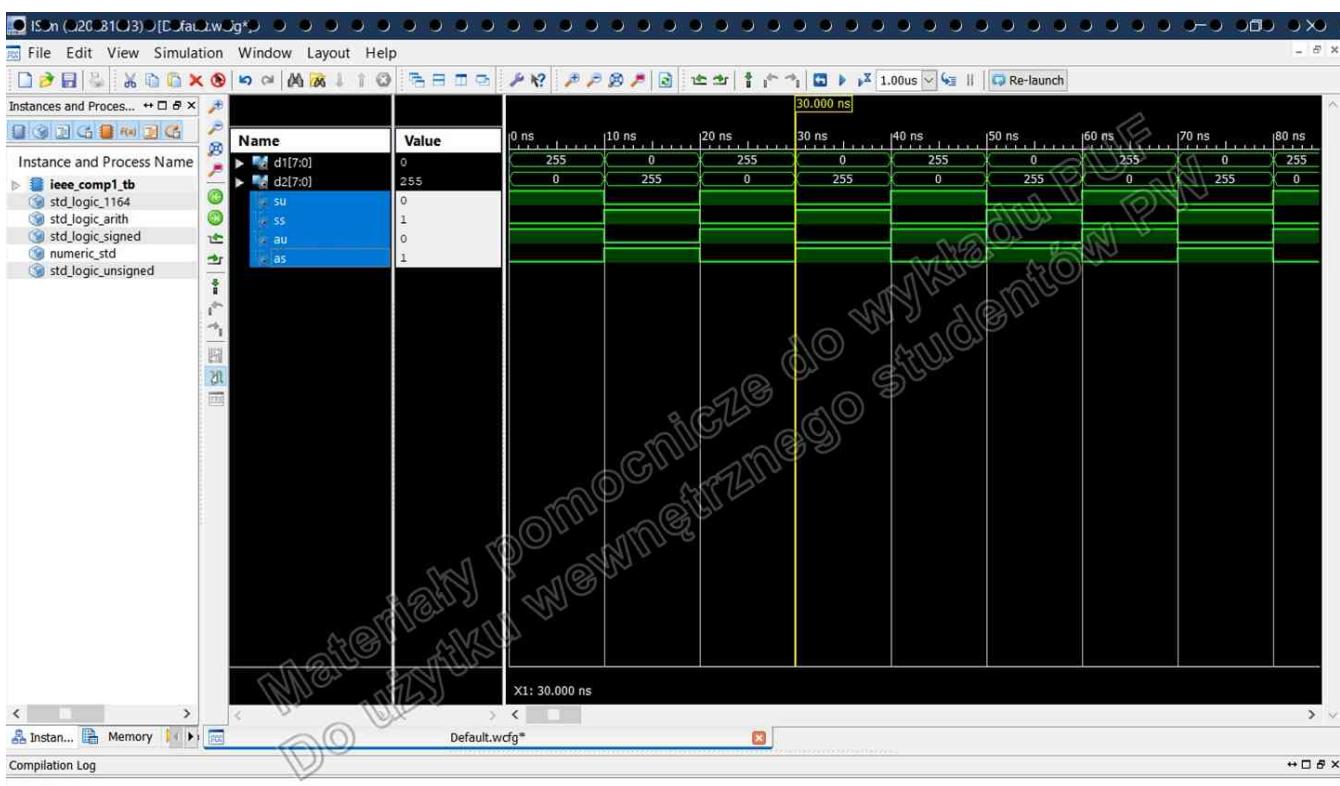
library ieee;
use ieee.std_logic_1164.all;
entity IEEE_COMP1_TB is
end IEEE_COMP1_TB;
architecture behavioural of IEEE_COMP1_TB is
begin
process is
begin
  D1 <= (others => '1');
  D2 <= (others => '0');
  wait for 10 ns;
  D1 <= (others => '0');
  D2 <= (others => '1');
  wait for 10 ns;
end process;
ieee_comp1_inst: entity work.IEEE_COMP1(cialo)
  port map (D1 => D1, D2 => D2, R => SU);
ieee_comp2_inst: entity work.IEEE_COMP2(cialo)
  port map (D1 => D1, D2 => D2, R => SS);
ieee_comp3_inst: entity work.IEEE_COMP3(cialo)
  port map (D1 => D1, D2 => D2, R => AU);
ieee_comp4_inst: entity work.IEEE_COMP4(cialo)
  port map (D1 => D1, D2 => D2, R => AS);
end behavioural;

```

Przykład użycia oprogramowania ISE – projekt: „ieee_comp” Plik źródłowy „ieee_comp_TB.vhd”



Przykład użycia oprogramowania ISim – projekt: „ieee_comp” Symulacja funkcjonalna (interpretacja dla kodowania U2)



Przykład użycia oprogramowania ISim – projekt: „ieee_comp” Symulacja funkcjonalna (interpretacja dla kodowania NB)

Podstawowe elementy standardu VHDL

Biblioteka IEEE – wybrane pakiety podstawowe

- pakiet **std_logic_1164** – podstawowe typy i operacje logiczne
- pakiet **std_logic_unsigned** – operacje na naturalnej liczbie binarnej (NB)
- pakiet **std_logic_signed** – operacje na liczbie binarnej ze znakiem (U2)
- pakiet **numeric_std** – podstawowe typy arytmetyczne z operacjami
 - typy arytmetyczne: SIGNED, UNSIGNED – podtypy std_logic_vector
 - operacje arytmetyczne: +, -, *, /, abs, rem, mod
 - operacje relacji: =, /=, >, >=, <, <=
 - funkcje konwertujące: TO_INTEGER, TO_UNSIGNED, TO_SIGNED,
TO_STDLOGICVECTOR, RESIZE
 - funkcje przesuwające: sll, srl, rol, ror,
SHIFT_LEFT, SHIFT_RIGHT, ROTATE_LEFT, ROTATE_RIGHT

Nie zaleca się stosowania pakietu **std_logic_arith** zamiast **numeric_std**

Podstawowe elementy standardu VHDL

Biblioteka IEEE – wybrane pakiety podstawowe

- pakiet **std_logic_1164** – podstawowe typy i operacje logiczne
- pakiet **std_logic_unsigned** – operacje na naturalnej liczbie binarnej (NB)
- pakiet **std_logic_signed** – operacje na liczbie binarnej ze znakiem (U2)
- pakiet **numeric_std** – podstawowe typy arytmetyczne z operacjami
- pakiet **std_logic_misc** – operacje uzupełniające
 - funkcje logiczne: AND_REDUCE, OR_REDUCE, XOR_REDUCE,
NAND_REDUCE, NOR_REDUCE, XNOR_REDUCE



Materiały pomocnicze do wykładu PUF
Do użytku wewnętrznego studentów PW

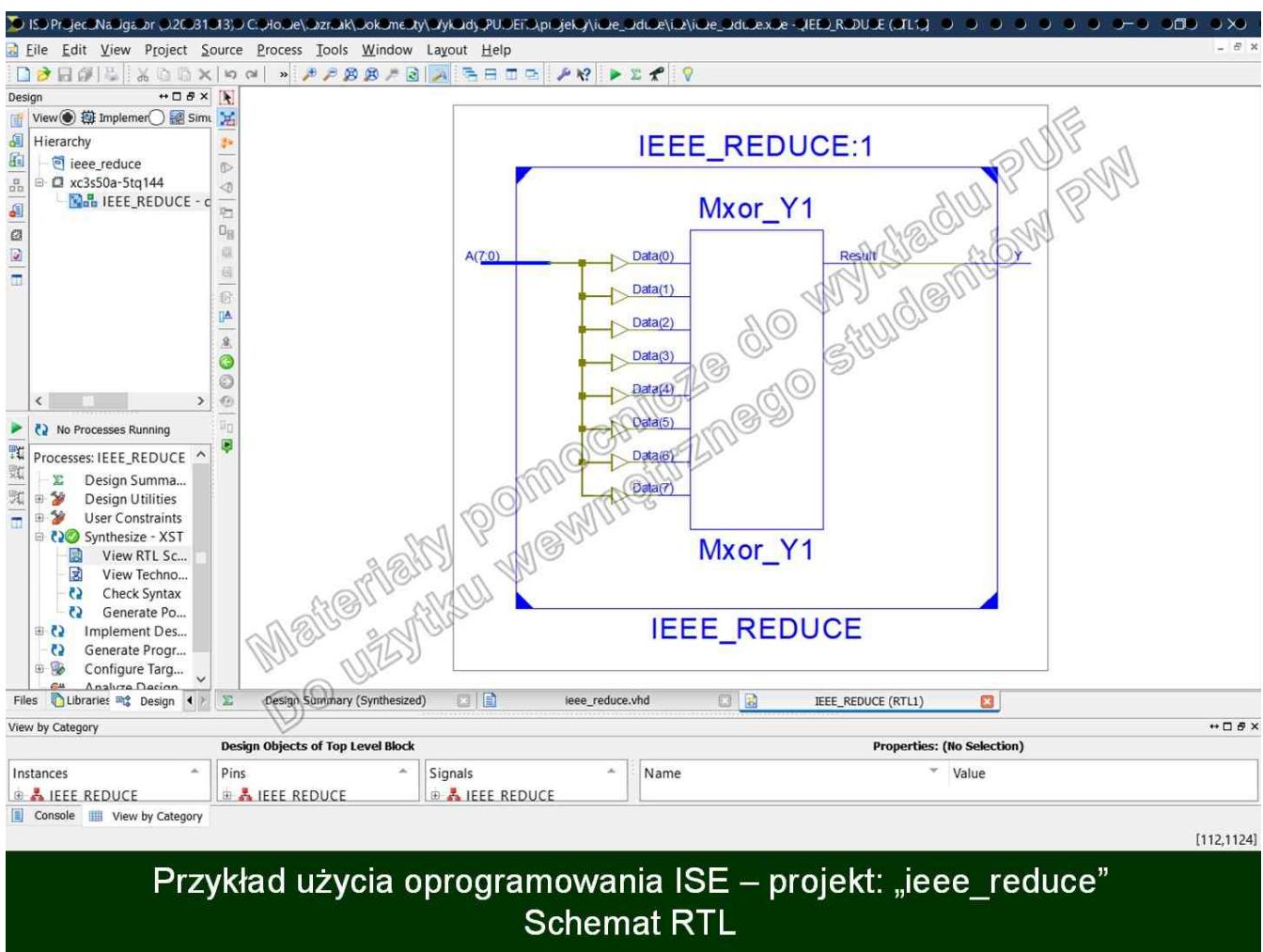
```

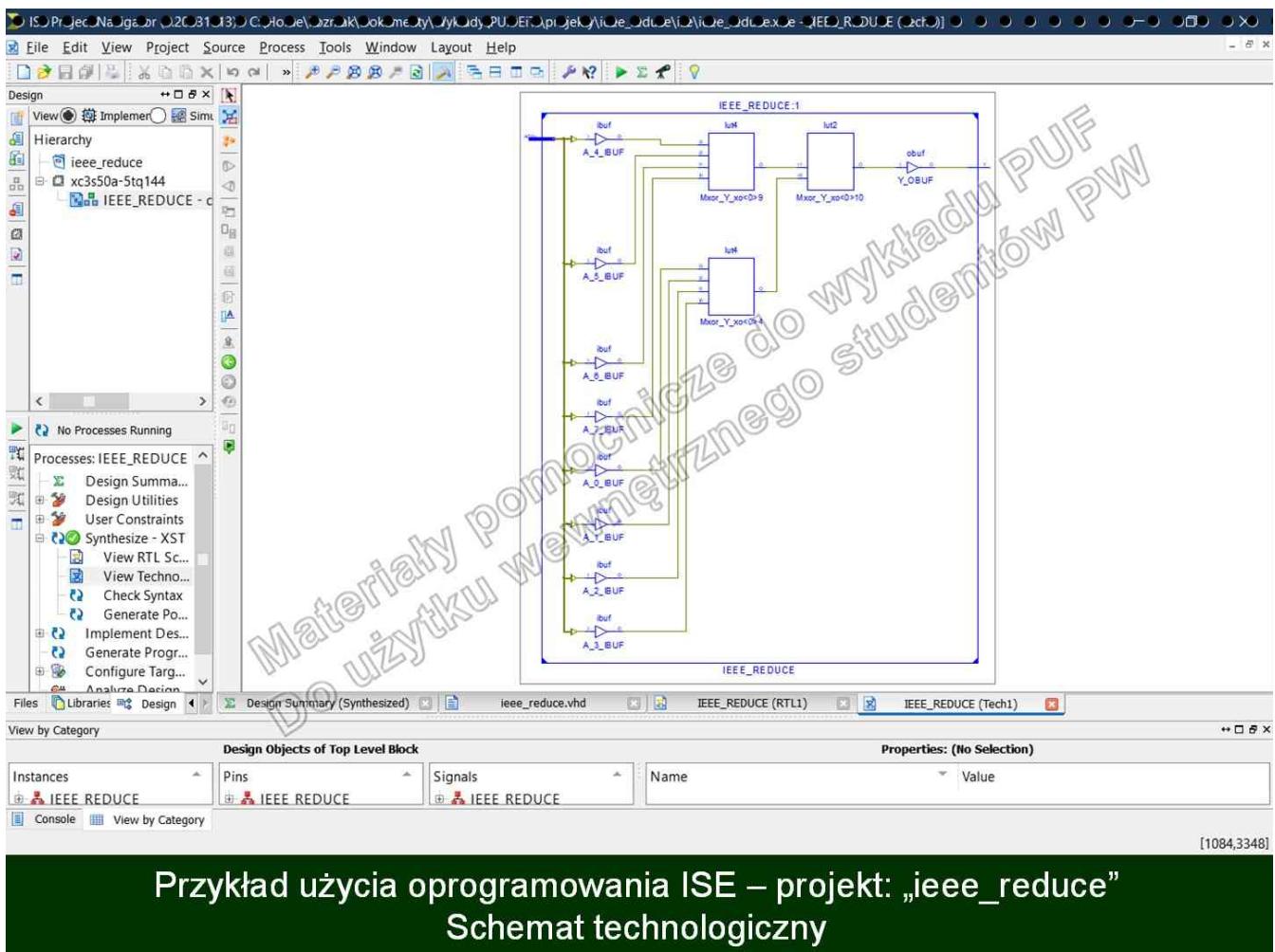
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_misc.all;
4
5 entity IEEE_REDUCE is
6   port ( A : in std_logic_vector(7 downto 0);
7         Y : out std_logic
8       );
9 end IEEE_REDUCE;
10
11 architecture cialo of IEEE_REDUCE is
12
13 begin
14
15   Y <= XOR_REDUCE(A);
16
17 end architecture cialo;
18
  -- klauzula dostepu do biblioteki 'IEEE'
  -- dolaczenie całego pakietu 'STD_LOGIC_1164'
  -- dolaczenie całego pakietu 'STD_LOGIC_MISC'
  -- deklaracja sprzedu 'IEEE_REDUCE'
  -- deklaracja portu wejściowego 'A'
  -- deklaracja portu wyjściowego 'Y'
  -- zakończenie deklaracji listy portów
  -- zakończenie deklaracji sprzedu
  -- deklaracja ciała 'cialo' architektury
  -- początek części wykonawczej architektury
  -- przypisanie do '0' operacji XOR_REDUCE na 'A'
  -- zakończenie deklaracji ciała 'cialo'

```

Started : "Launching ISE Text Editor to edit ieee_reduce.vhd".

Przykład użycia oprogramowania ISE – projekt: „ieee_reduce” Plik źródłowy „ieee_reduce.vhd”



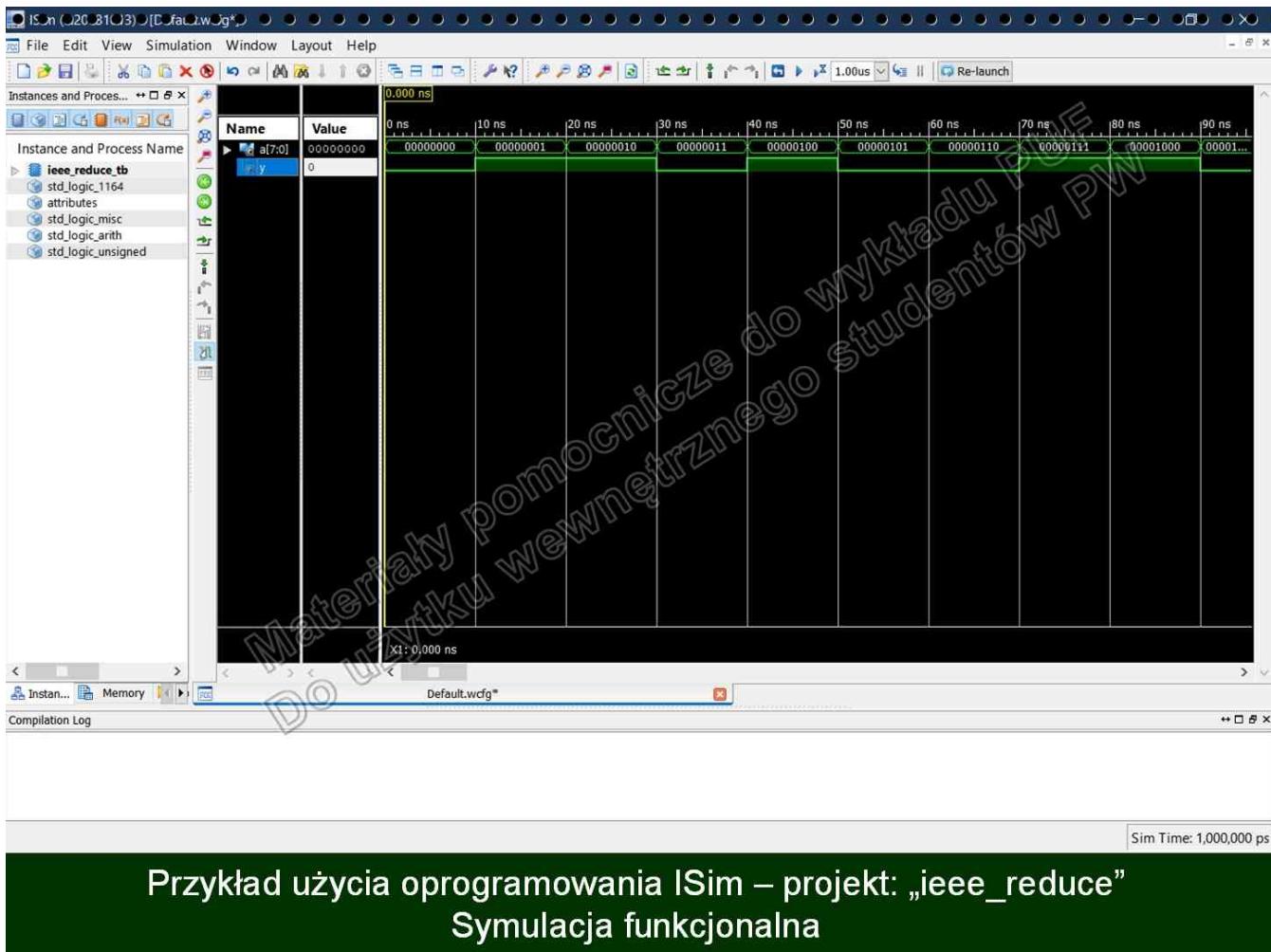


```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4
5 entity IEEE_REDUCE_TB is
6 end IEEE_REDUCE_TB;
7
8 architecture behavioural of IEEE_REDUCE_TB is
9
10 signal A :std_logic_vector(7 downto 0) := (others => '0'); -- symulowane wejście A z inicjalizacją
11 signal Y :std_logic; -- obserwowane wyjście
12
13 begin
14
15 process is
16 begin
17 wait for 10 ns;
18 A <= A + 1;
19 end process;
20
21 IEEE_REDUCE_inst: entity work.IEEE_REDUCE(ciało) -- instancja projektu 'BBUF'
22 port map (
23 A => A,
24 Y => Y
25 );
26 end behavioural;

```

Przykład użycia oprogramowania ISE – projekt: „ieee_reduce”
Plik źródłowy „ieee_reduce_TB.vhd”



Przykład użycia oprogramowania ISim – projekt: „ieee_reduce”
Symulacja funkcjonalna