

Programowanie układów FPGA – wykład VII

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

- Instrukcje monitorowania
- Instrukcje operacji na pliku
- Wybrane metody parametryzacji projektu
- Parametryzowany odbiornik szeregowy
- Parametryzowany nadajnik szeregowy
- Synchronizowany interfejs komunikacyjny

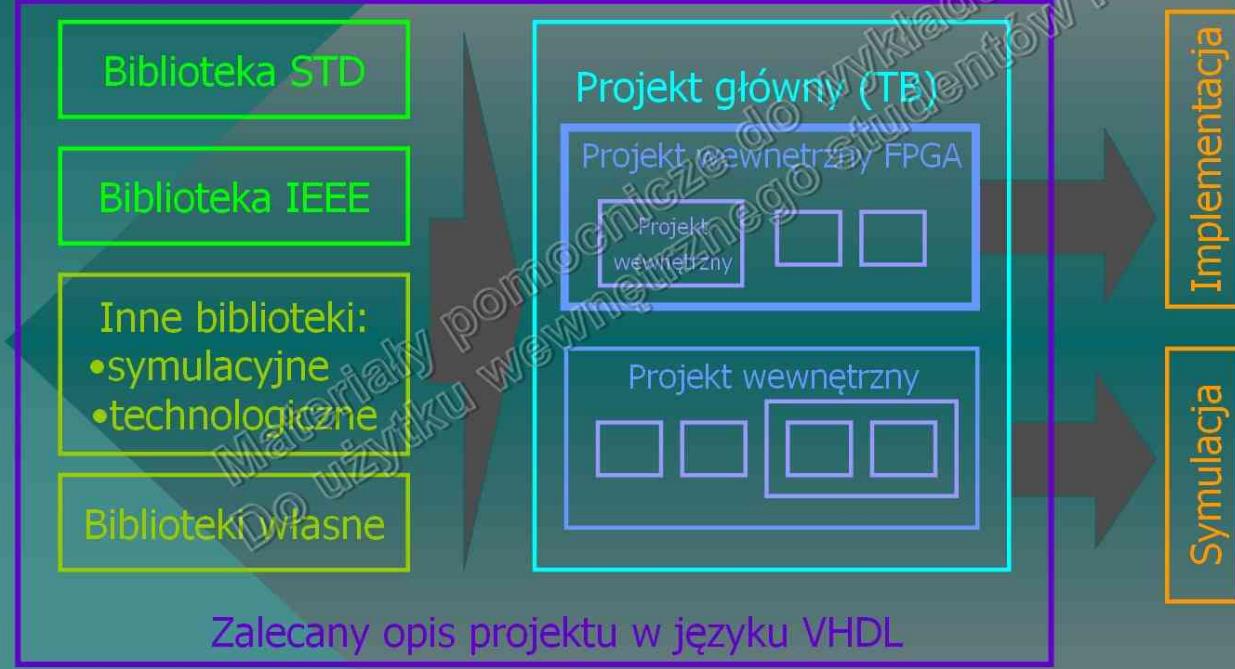
Podstawowe elementy standardu VHDL

Podstawowa struktura opisu projektu dla FPGA



Podstawowe elementy standardu VHDL

Podstawowa struktura opisu projektu dla FPGA



Instrukcje monitorowania

Instrukcja założenia

Składnia instrukcji założenia:

[etykieta :] assert warunek [report opis] [severity poziom] ;

typy składowych instrukcji założenia:

- warunek: wyrażenie logiczne zwracające TRUE lub FALSE
- opis: ciąg znakowy typu STRING;
- poziom: poziom sygnalizacji typu SEVERITY_LEVEL;

```
type SEVERITY_LEVEL is (
    NOTE,          -- informacja
    WARNING,       -- ostrzeżenie
    ERROR,        -- błąd
    FAILURE        -- błąd krytyczny
);
```

Instrukcje monitorowania

Instrukcja założenia

Składnia instrukcji założenia:

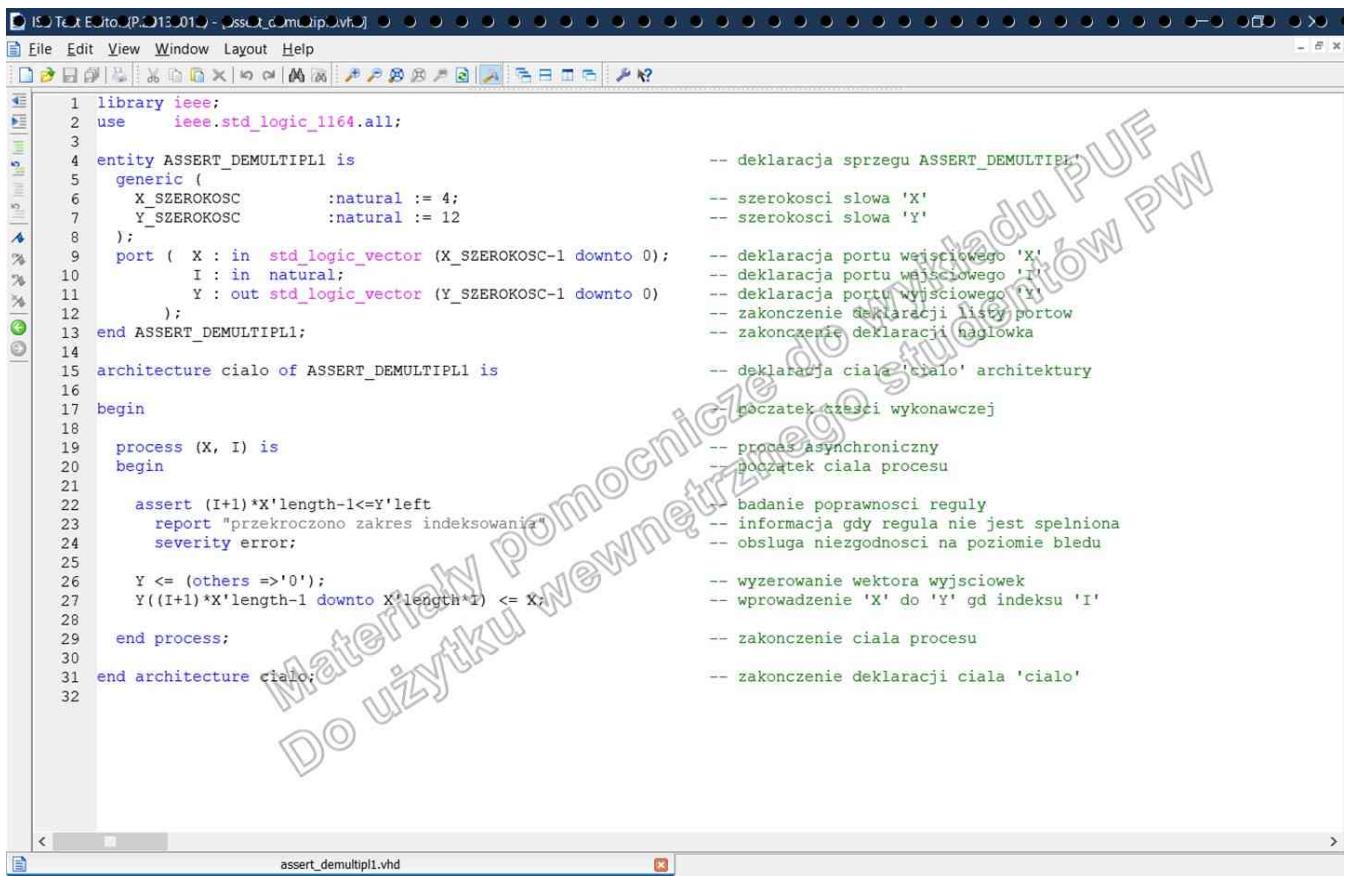
[etykieta :] assert warunek [report opis] [severity poziom];

typy składowych instrukcji założenia:

- warunek: wyrażenie logiczne zwracające TRUE lub FALSE
- opis: ciąg znakowy typu STRING;
- poziom: poziom sygnalizacji typu SEVERITY_LEVEL;

Instrukcja założenia jest wykonywana wtedy, gdy
warunek nie jest spełniony, tzn. zwraca wartość FALSE

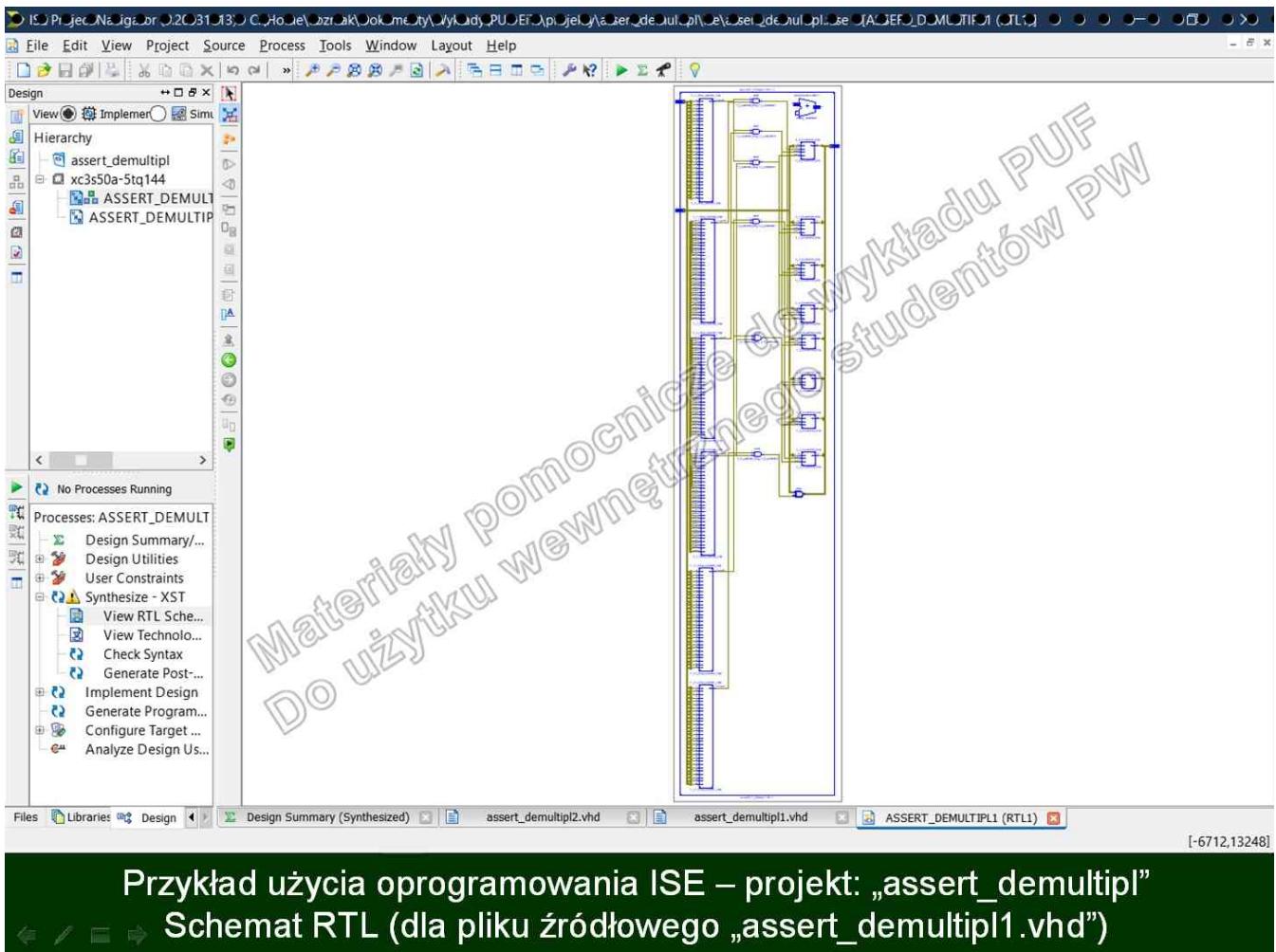
Na konsoli jest wypisywana zawartość składowej *opis*
w trybie odpowiadającym przypisaniu składowej *poziom*



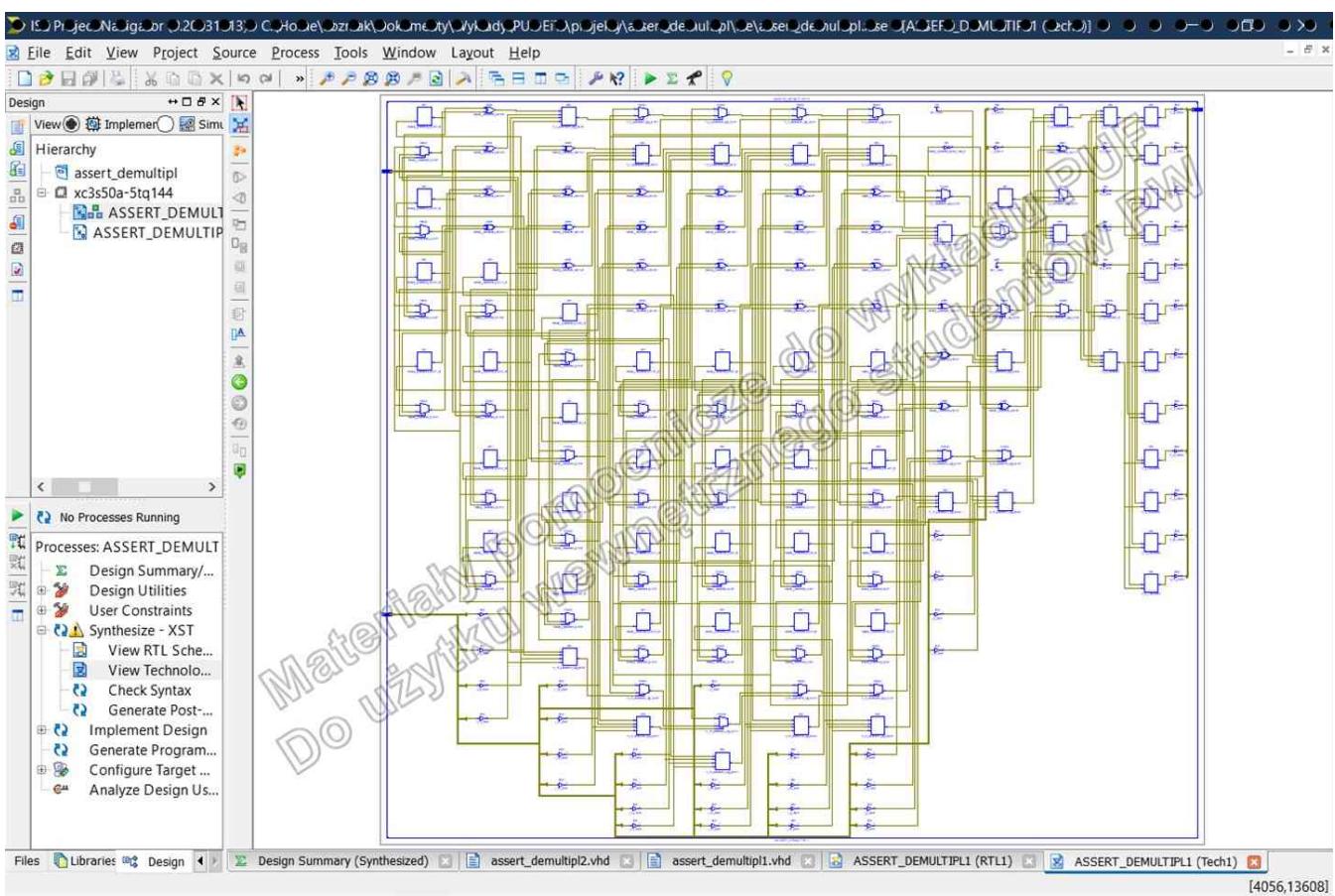
```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity ASSERT_DEMULTIPL1 is
5     generic (
6         X_SZEROKOSC      :natural := 4;
7         Y_SZEROKOSC      :natural := 12
8     );
9     port ( X : in std_logic_vector (X_SZEROKOSC-1 downto 0);
10            I : in natural;
11            Y : out std_logic_vector (Y_SZEROKOSC-1 downto 0)
12        );
13 end ASSERT_DEMULTIPL1;
14
15 architecture cialo of ASSERT_DEMULTIPL1 is
16
17 begin
18
19 process (X, I) is
20 begin
21
22     assert (I+1)*X'length-1 <= Y'left
23         report "przekroczeno zakres indeksowania"
24         severity error;
25
26     Y <= (others =>'0';
27     Y((I+1)*X'length-1 downto X'length*I) <= X;
28
29 end process;
30
31 end architecture cialo;
32
```

-- deklaracja sprzegu ASSERT_DEMULTIPL1
-- szerokosci slowa 'X'
-- szerokosci slowa 'Y'
-- deklaracja portu wejsciowego 'X'
-- deklaracja portu wejsciowego 'I'
-- deklaracja portu wyjsciowego 'Y'
-- zakonczenie deklaracji listy portow
-- zakonczenie deklaracji naglowka
-- deklaracja ciala 'cialo' architektury
-- poczatek czesci wykonawczej
-- proces asynchroniczny
-- poczatek ciala procesu
-- badanie poprawnosci reguly
-- informacja gdy regula nie jest spełniona
-- obsługa niezgodnosci na poziomie bledu
-- wyzerowanie wektora wyjsciowek
-- wprowadzenie 'X' do 'Y' gd indeksu 'I'
-- zakonczenie ciala procesu
-- zakonczenie deklaracji ciala 'cialo'

Przykład użycia oprogramowania ISE – projekt: „assert_demultipl”
Plik źródłowy „assert_demultipl1.vhd”



Przykład użycia oprogramowania ISE – projekt: „assert_demultipl”
Schemat RTL (dla pliku źródłowego „assert_demultip1.vhd”)



Przykład użycia oprogramowania ISE – projekt: „assert_demultipl”
Schemat technologiczny (dla pliku źródłowego „assert_demultip1.vhd”)

Przykład użycia oprogramowania ISE – projekt: „assert_demultipl”

Plik źródłowy „assert_demultipl_TB.vhd” (entity „assert_demultipl1_TB”)

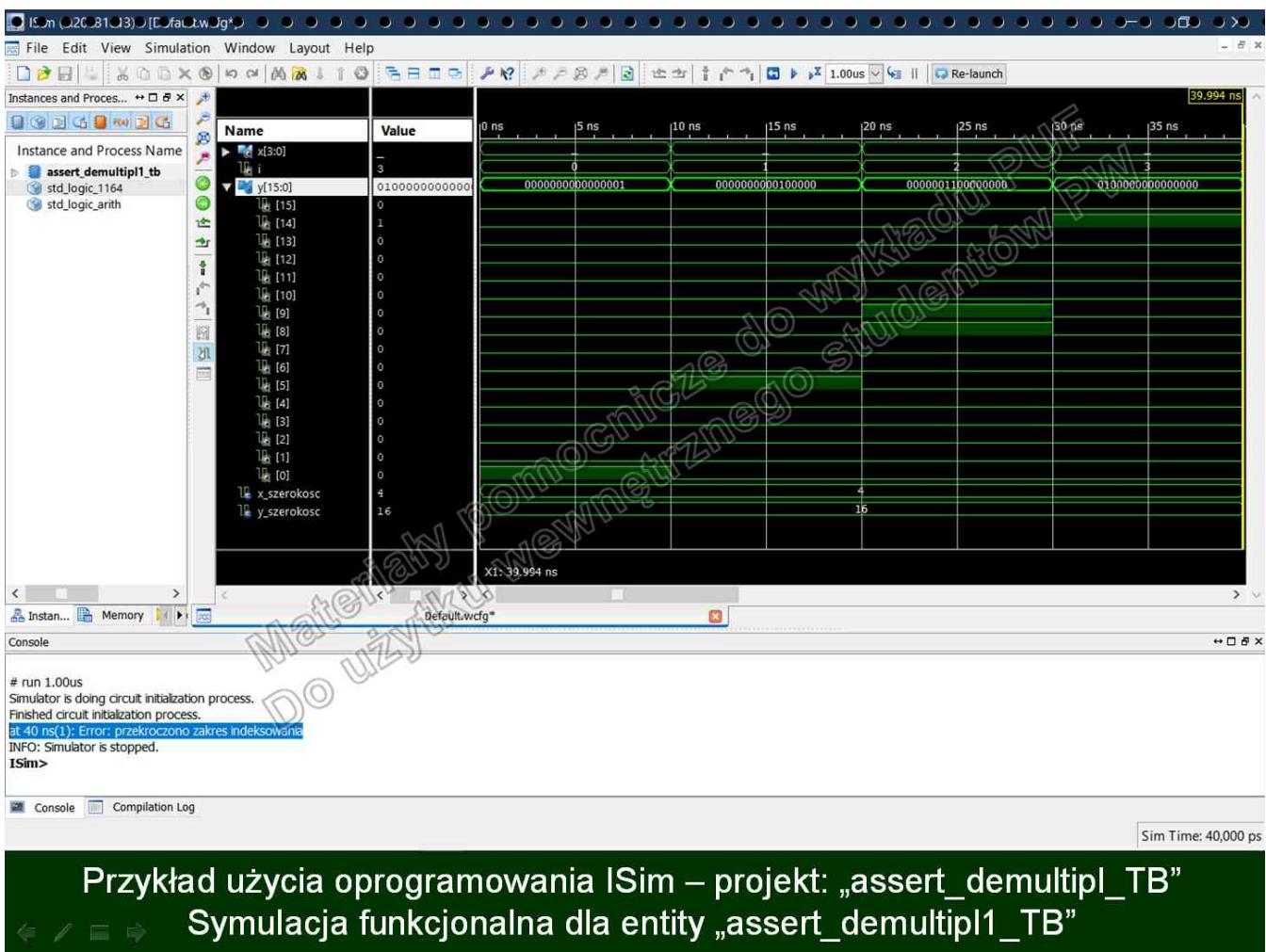
```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity ASSERT_DEMULTIPL1_TB is
    generic (
        X_SZEROKOSC : natural := 4;
        Y_SZEROKOSC : natural := 16
    );
end ASSERT_DEMULTIPL1_TB;

architecture behavioural of ASSERT_DEMULTIPL1_TB is
begin
    signal X : std_logic_vector (X_SZEROKOSC-1 downto 0);
    signal I : natural;
    signal Y : std_logic_vector (Y_SZEROKOSC-1 downto 0);
    begin
        process is
        begin
            for k in 0 to 10 loop
                I <= k;
                X <= CONV_STD_LOGIC_VECTOR(k+1,X'length);
                wait for 10ns;
            end loop;
            wait;
        end process;
        assert_demultipl1_inst: entity work ASSERT_DEMULTIPL1(ciało)
            generic map (
                X_SZEROKOSC => X_SZEROKOSC,
                Y_SZEROKOSC => Y_SZEROKOSC
            )
            port map (
                X => X,
                I => I,
                Y => Y
            );
    end behavioural;

```



ISE Text Editor (P.2013.10.3) - assert_demultipl2.vhd

```

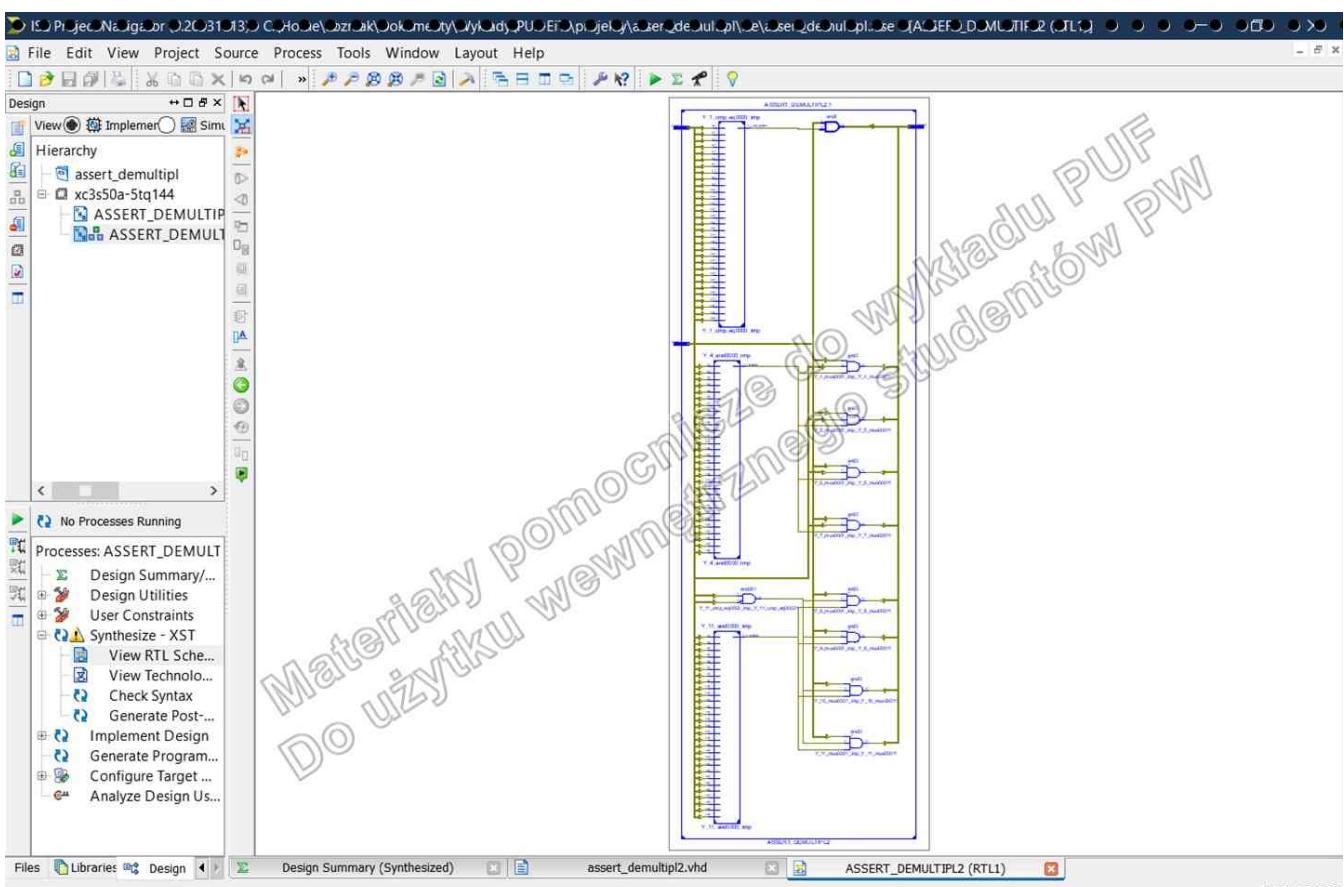
1 library ieee;
2 use      ieee.std_logic_1164.all;
3
4 entity ASSERT_DEMULTIPL2 is
5     generic (
6         X_SZEROKOSC      :natural := 4;
7         Y_SZEROKOSC      :natural := 12
8     );
9     port ( X : in  std_logic_vector (X_SZEROKOSC-1 downto 0);
10            I : in  natural;
11            Y : out std_logic_vector (Y_SZEROKOSC-1 downto 0)
12        );
13 end ASSERT_DEMULTIPL2;
14
15 architecture cialo of ASSERT_DEMULTIPL2 is
16
17 begin
18
19     process (X, I) is
20     begin
21
22         Y <= (others =>'0');
23         for k in 0 to Y'length/X'length-1 loop
24             if (k=I) then
25                 Y((k+1)*X'length-1 downto X'length*k) <= X;
26                 exit;
27             end if;
28             assert k<Y'length/X'length-1
29                 report "indeks I=" & integer'image(I) & " jest za duzy"
30                         severity warning;
31         end loop;
32         -- Y((I+1)*X'length-1 downto X'length*I) <= X;
33
34     end process;
35
36 end architecture cialo;
37

```

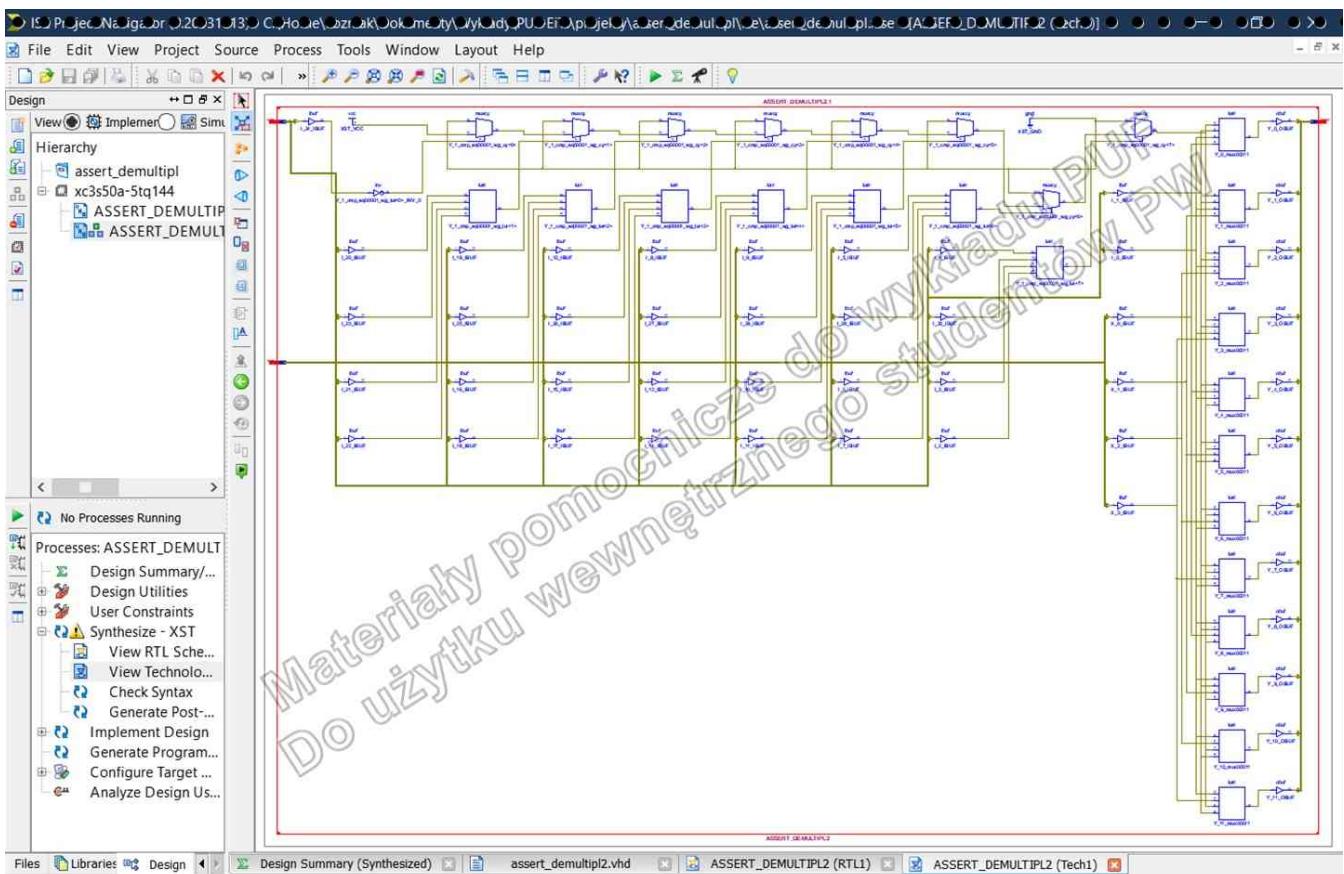
-- deklaracja sprzegu ASSERT_DEMULTIPL2
-- szerokosc slowa 'X'
-- szerokosc slowa 'Y'
-- deklaracja portu wejsciowego 'X'
-- deklaracja portu wejsciowego 'I'
-- deklaracja portu wyjsciowego 'Y'
-- zakoñczenie deklaracji listy portow
-- zakoñczenie deklaracji naglowka
-- deklaracja ciala 'cialo' architektury
-- poczatek czesci wykonawczej
-- proces asynchroniczny
-- poczatek ciala procesu
-- wyzerowanie wektora wyjsciowek
-- wprowadzenie 'X' do 'Y' gd indeksu 'I'
-- badanie wykonywania petli
-- informacja gdy regula nie jest spełniona
-- obsługa niezgodnosci na poziomie ostrzezenia
-- wprowadzenie 'X' do 'Y' gd indeksu 'I'
-- zakoñczenie ciala procesu
-- zakoñczenie deklaracji ciala 'cialo'

assert_demultipl2.vhd

Przykład użycia oprogramowania ISE – projekt: „assert_demultipl”
 Plik źródłowy „assert_demultipl2.vhd”



Przykład użycia oprogramowania ISE – projekt: „assert_demultipl”
 Schemat RTL (dla pliku źródłowego „assert_demultipl2.vhd”)



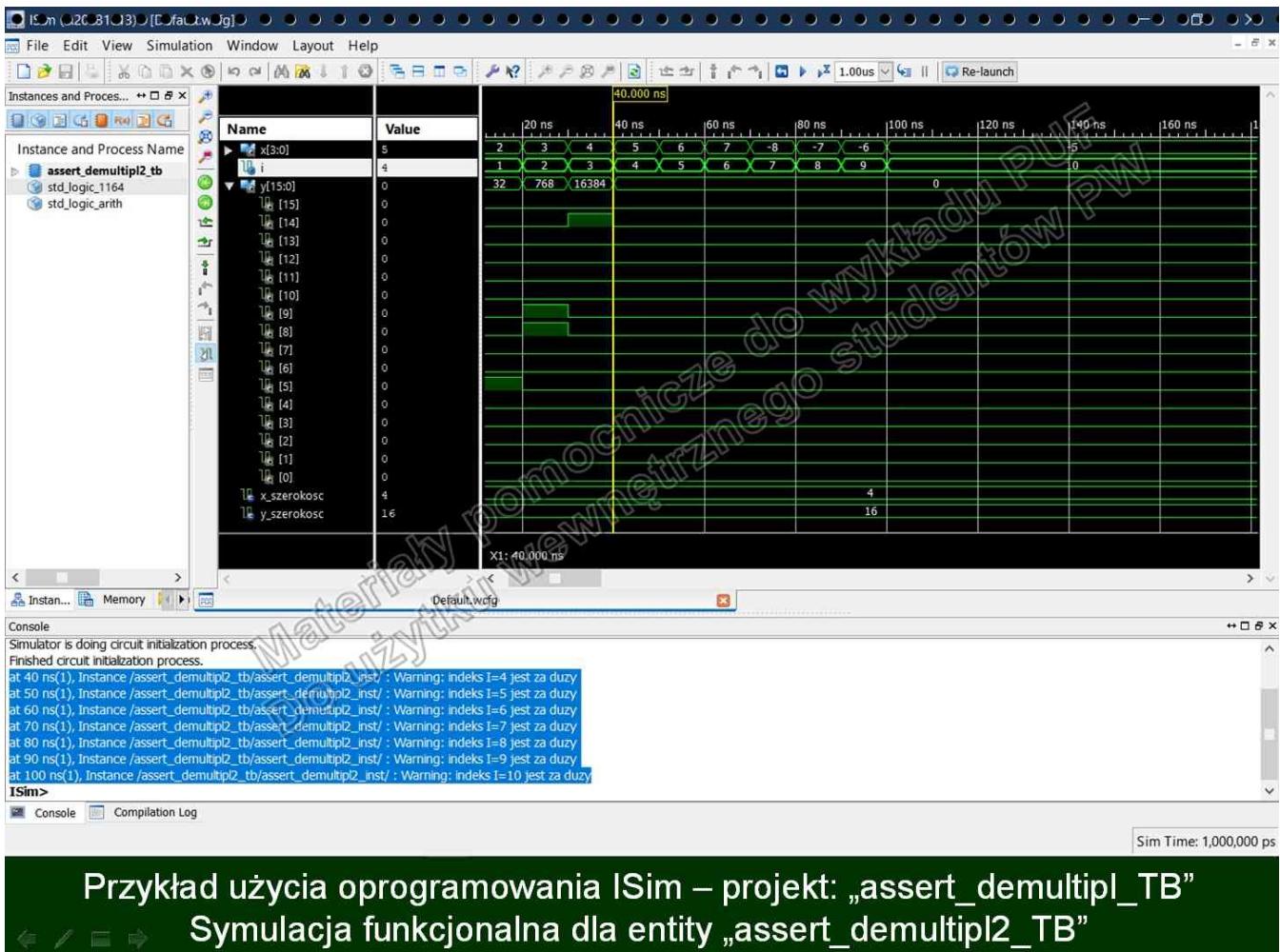
Przykład użycia oprogramowania ISE – projekt: „assert_demultipl”
Schemat technologiczny (dla pliku źródłowego „assert_demultipl2.vhd”)

```

48
49 entity ASSERT_DEMULTIPL2_TB is
50 generic (
51   X_SZEROKOSC      :natural := 4;
52   Y_SZEROKOSC      :natural := 16
53 );
54 end ASSERT_DEMULTIPL2_TB;
55
56 architecture behavioural of ASSERT_DEMULTIPL2_TB is
57
58   signal X :std_logic_vector (X_SZEROKOSC-1 downto 0);
59   signal I :natural;
60   signal Y :std_logic_vector (Y_SZEROKOSC-1 downto 0);
61
62 begin
63
64   process is
65   begin
66     for k in 0 to 10 loop
67       I <= k;
68       X <= CONV_STD_LOGIC_VECTOR(k+1,X'length);
69       wait for 10ns;
70     end loop;
71     wait;
72   end process;
73
74   assert demultipl2_inst: entity work ASSERT_DEMULTIPL2(cialo)
75     generic map (
76       X_SZEROKOSC => X_SZEROKOSC,
77       Y_SZEROKOSC => Y_SZEROKOSC
78     )
79     port map (
80       X    => X,
81       I    => I,
82       Y    => Y
83     );
84
85 end behavioural;
86

```

Przykład użycia oprogramowania ISE – projekt: „assert_demultipl”
Plik źródłowy „assert_demultipl_TB.vhd” (entity „assert_demultipl2_TB”)



Instrukcje monitorowania Instrukcja raportowania

Składnia instrukcji raportowania:

[etykieta :] report opis [severity poziom] ;

typy składowych instrukcji założenia:

- **opis:** ciąg znakowy typu STRING;
- **poziom:** poziom sygnalizacji typu SEVERITY_LEVEL;

```
type SEVERITY_LEVEL is (
    NOTE,          -- informacja
    WARNING,       -- ostrzeżenie
    ERROR,         -- błąd
    FAILURE        -- błąd krytyczny
);
```

Instrukcja raportowania jest równoważna instrukcji założenia dla przypadku gdy warunek posiada na stałe wartość FALSE

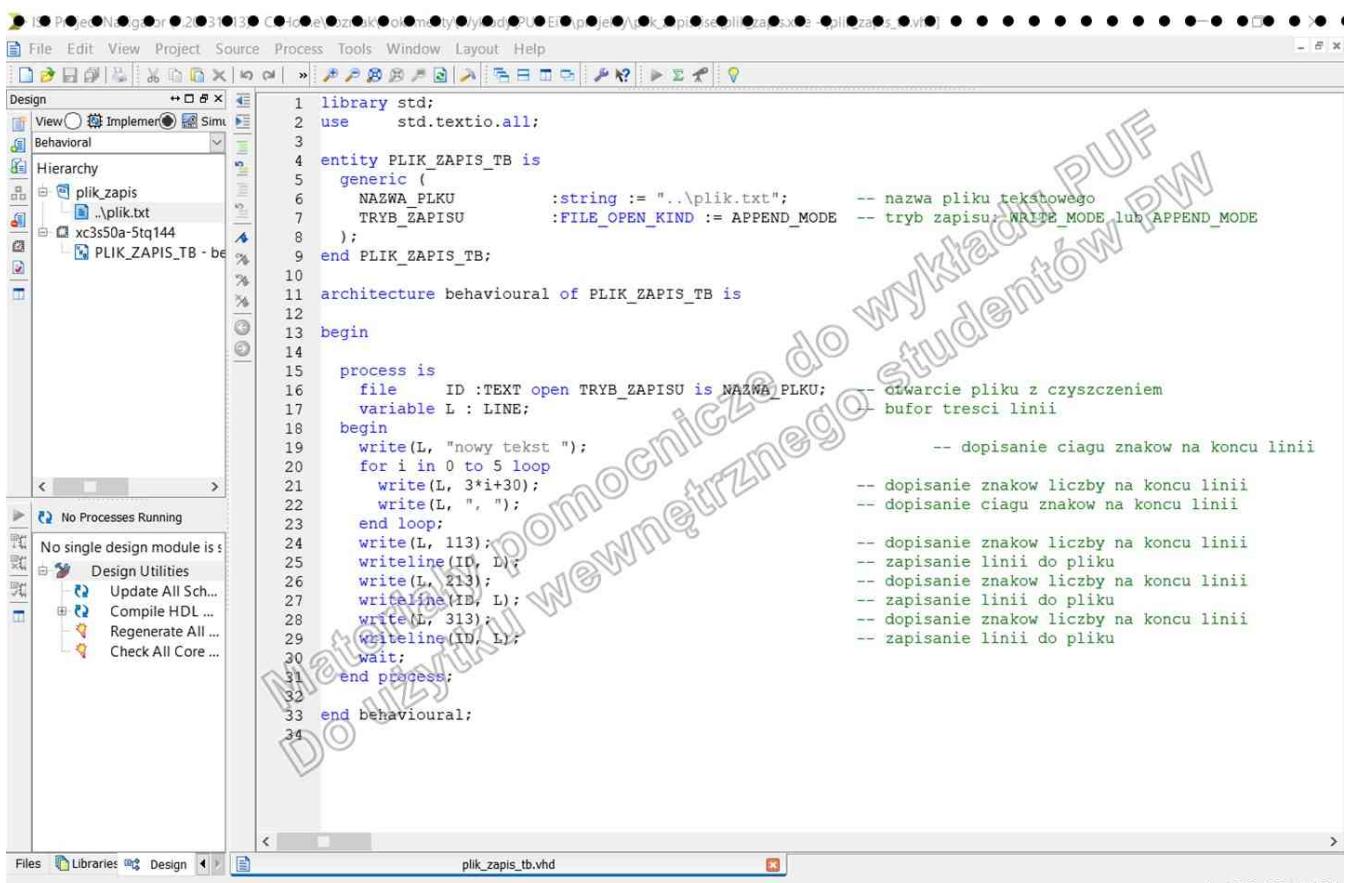
Instrukcje operacji na pliku tekstowym (pakiet TEXTIO)

Instrukcje zapisywania danych

- typ FILE: określa ścieżkę+nazwę i sposób otwarcia pliku

```
file identyfikator : TEXT open WRITE_MODE is sciezka_nazwa_pliku;  
                                APPEND_MODE
```

- typ LINE: zawiera ciąg znakowy pojedynczej linii;
- procedury: umożliwiają utworzenie linii oraz jej zapis
 - write: dopisuje na końcu linii nowy element danego typu
 - writeline: zapisuje utworzoną linie do pliku tekstopowego



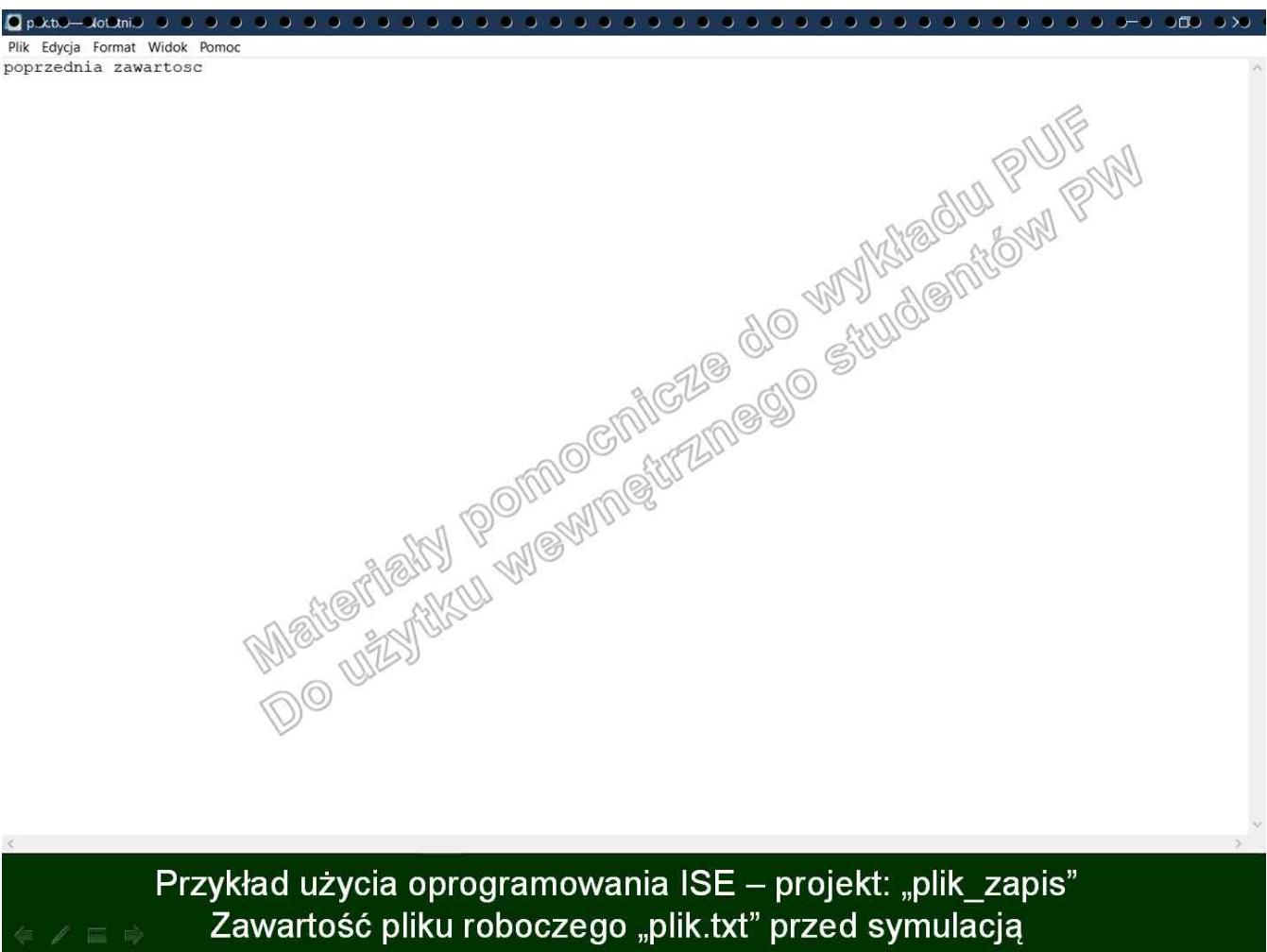
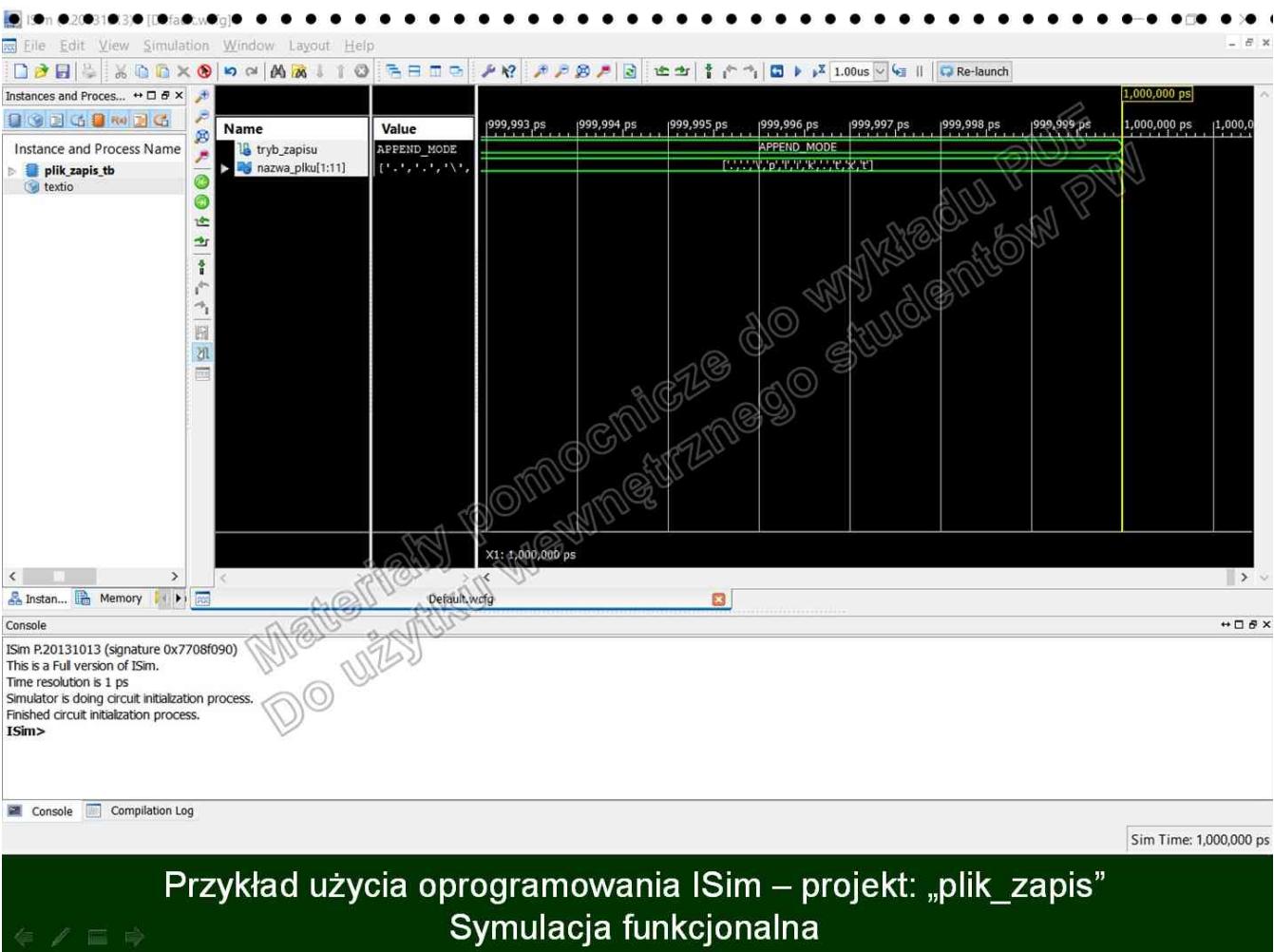
```
library std;
use std.textio.all;

entity PLIK_ZAPIS_TB is
    generic (
        NAZWA_PLKU      :string := "..\plik.txt"; -- nazwa pliku tekstopowego
        TRYB_ZAPISU     :FILE_OPEN_KIND := APPEND_MODE -- tryb zapisu - WRITE_MODE lub APPEND_MODE
    );
end PLIK_ZAPIS_TB;

architecture behavioural of PLIK_ZAPIS_TB is
begin
begin
process is
    file ID :TEXT open TRYB_ZAPISU is NAZWA_PLKU; -- otwarcie pliku z czyszczeniem
    variable L : LINE; -- bufor tresci linii
begin
    write(L, "nowy tekst ");
    for i in 0 to 5 loop
        write(L, 3*i+30);
        write(L, " ");
    end loop;
    write(L, 113);
    writeline(ID, L);
    write(L, 213);
    writeline(ID, L);
    write(L, 313);
    writeline(ID, L);
    wait;
end process;
end behavioural;
```

Przykład użycia oprogramowania ISE – projekt: „plik_zapis”

Plik źródłowy „plik_zapis_TB.vhd”



p.kt@... Dot. Oni... Plik Edycja Format Widok Pomoc
poprzednia zawartosc
nowy tekst 30, 33, 36, 39, 42, 45, 113
213
313

Przykład użycia oprogramowania ISE – projekt: „plik_zapis” Zawartość pliku roboczego „plik.txt” po symulacji (tryb APPEND)

Instrukcje operacji na pliku tekstowym (pakiet TEXTIO)

Instrukcje odczytywania danych

- typ FILE: określa ścieżkę+nazwę i sposób otwarcia pliku
`file identyfikator : TEXT open READ_MODE is sciezka_nazwa_pliku;`
- typ LINE: zawiera ciąg znakowy pojedynczej linii;
- procedury: umożliwiają utworzenie linii oraz jej odczyt
 - readline: odczytuje całą linie z pliku tekstowego
 - read: interpretuje w linii kolejny element danego typu
 - endfile: sprawdza czy osiągnięto koniec pliku



```
library std;
use std.textio.all;

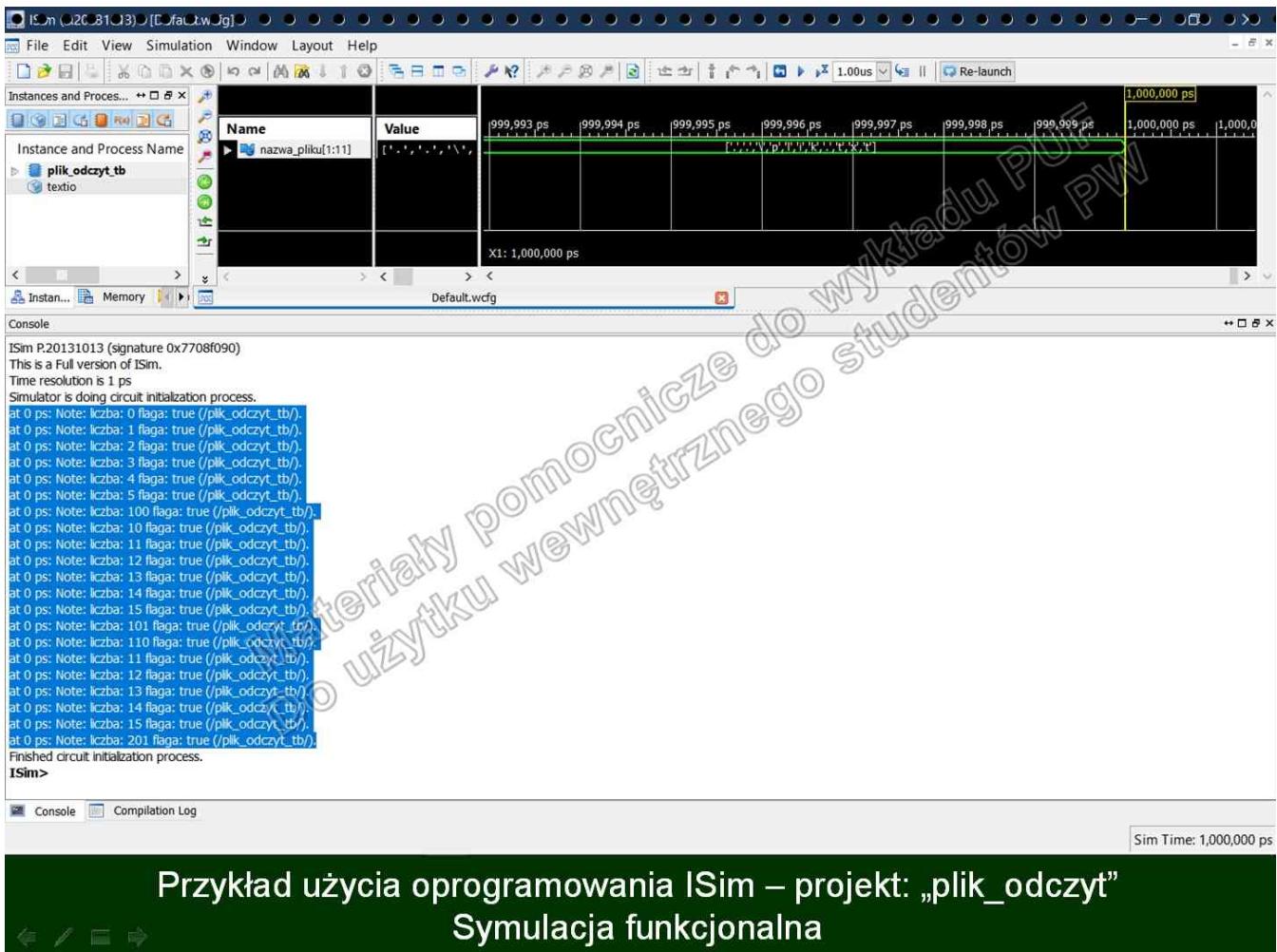
entity PLIK_ODCZYT_TB is
    generic (
        NAZWA_PLIKU : string := "..\plik.txt" -- nazwa pliku tekstowego
    );
end PLIK_ODCZYT_TB;

architecture behaviouralal of PLIK_ODCZYT_TB is
begin
    process is
        file ID : TEXT open READ_MODE is NAZWA_PLIKU; -- otwarcie pliku w trybie odczytu
        variable L : LINE; -- bufor tresci linii
        variable N : natural; -- bufor liczby
        variable F : boolean; -- bufor flagi poprawnosci
    begin
        while (endfile(ID)=FALSE) loop
            readline(ID, L); -- odczytanie linii z pliku
            for i in 0 to 6 loop
                read(L, N, F); -- odczytanie kolejnych znakow liczby
                report "liczba: " & integer'image(N) & " flaga: " & boolean'image(F);
            end loop;
            end loop;
            wait;
        end process;
    end behaviouralal;
```

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

```
0 1 2 3 4 5 100
10 11 12 13 14 15 101
110 11 12 13 14 15 201
```

Przykład użycia oprogramowania ISE – projekt: „plik_odczyt”
Zawartość pliku roboczego „plik.txt”



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

Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:
 - definicji pakietu (zasięg w zakresie użycia pakietu)

```
package pakiet is
  constant STALA :natural := 24;
end package pakiet;
```

Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)

```
package pakiet is  
end package pakiet;  
package body pakiet is  
    constant STALA :natural := 24;  
end package body pakiet;
```

Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)

```
entity sprzeg is  
end entity sprzeg;  
architecture cialo of sprzeg is  
    constant STALA :natural := 24;  
begin  
end architecture cialo ;
```

Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)

```
process ( sygnal ) is
    constant STALA :natural := 24;
begin
end process ;
```

Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

```
function fun ( arg : bit ) return bit is
    constant STALA :natural := 24;
begin
end function fun ;
```

Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

- Wybrane miejsca podejmowania stałych w:

- definicjach typów/podtypów (np. rozmiary wektorów, zakresy)

```
type tablica is array (0 to STALA1) of bit_vector(STALA2 downto 0);
```



Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

- Wybrane miejsca podejmowania stałych w:

- definicjach typów/podtypów (np. rozmiary wektorów, zakresy)
- definicji zmiennych/sygnalów (np. rozmiary wektorów, zakresy)

```
signal sygnał : bit_vector(STALA1 downto 0);
```

```
variable zmienna : bit_vector(STALA1 downto 0) := (others => '0');
```



Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

- Wybrane miejsca podejmowania stałych w:

- definicjach typów/podtypów (np. rozmiary wektorów, zakresy)
- definicji zmiennych/sygnalów (np. rozmiary wektorów, zakresy)
- definicji portów sprzęgu (np. rozmiary wektorów, zakresy)

```
entity sprzeg is
  port (
    wejscie :in bit_vector(STALA1 downto 0);
    wyjscie :out bit_vector(STALA2 downto 0)
  );
end entity sprzeg;
```

Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

- Wybrane miejsca podejmowania stałych w:

- definicjach typów/podtypów (np. rozmiary wektorów, zakresy)
- definicji zmiennych/sygnalów (np. rozmiary wektorów, zakresy)
- definicji portów sprzęgu (np. rozmiary wektorów, zakresy)
- wyrażeniach (arytmetycznych, logicznych, relacji)

```
sygnal <= zmienna + STALA;
if sygnal = STALA then
  zmienna := STALA * zmienna ;
end if;
```

Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

- Wybrane miejsca podejmowania stałych w:

- definicjach typów/podtypów (np. rozmiary wektorów, zakresy)
- definicji zmiennych/sygnalów (np. rozmiary wektorów, zakresy)
- definicji portów sprzęgu (np. rozmiary wektorów, zakresy)
- wyrażeniach (arytmetycznych, logicznych, relacji)

- Wybrane metody przekazywania stałych poprzez:

- użycie atrybutów (udostępnienie aktualnych atrybutów typu)



```
1 library ieee; -- Klauzula dostępu do biblioteki 'IEEE'
2 use ieee.std_logic_1164.all; -- dodanie całego pakietu 'STD LOGIC_1164'
3
4 entity PARAM_ATRYB is -- deklaracja sprzęgu 'PARAM_ATRYB'
5   port (
6     W : in std_logic_vector(0 to 3); -- wejście danych 'W'
7     S : out natural range 0 to 4; -- wyjście danych 'S'
8   );
9 end PARAM_ATRYB;
10
11 architecture cialo of PARAM_ATRYB is -- deklaracja ciała 'cialo' architektury
12
13 begin -- początek części wykonawczej
14
15 process (W) is -- lista części的过程
16   variable N : natural range 0 to W'length; -- deklaracja zmiennej 'N'
17 begin -- część wykonawcza procesu
18   N := 0; -- podstawienie licznika jedynek N=0
19   for i in W'range loop -- pętla zmiennej 'i' po zakresie 'W'
20     if W(i)'=1' then -- sprawdzenie, czy i-ty bit 'W' zawiera '1'
21       N := N + 1; -- zwiększenie licznika jedynek 'N' o 1
22     end if; -- zakończenie instrukcji wyboru
23   end loop; -- zakończenie pętli
24   S <= N; -- podstawienie pod sygnał 'S' wartości 'N'
25 end process; -- zakończenie procesu
26
27 end cialo; -- zakończenie ciała architektonicznego
28
29
```

Design View Implement Sim

Hierarchy

- param_atryb
 - xc3s50a-5tq144
 - PARAM_ATRYB

No Processes Running

Processes: PARAM_ATRYB ->

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

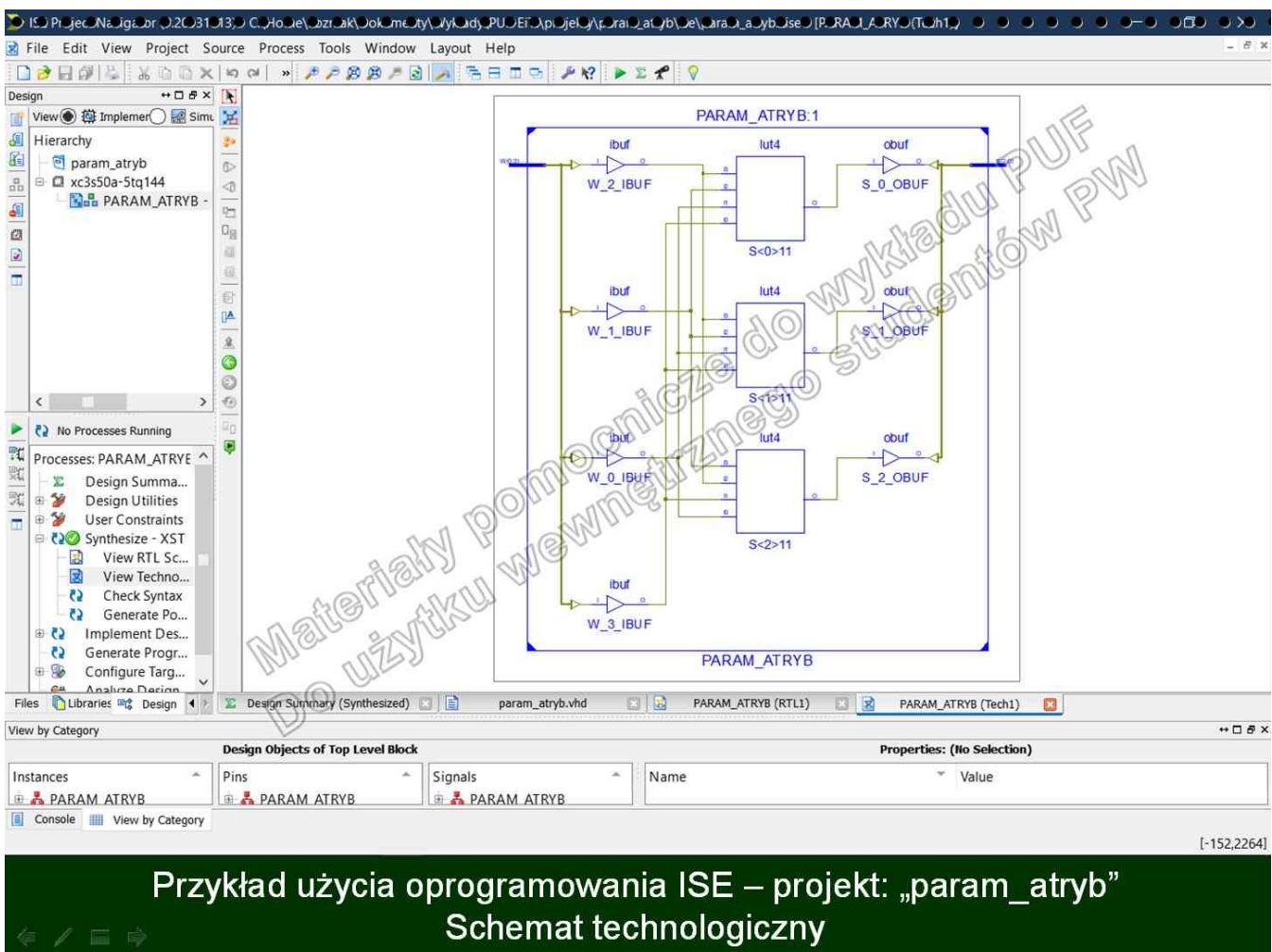
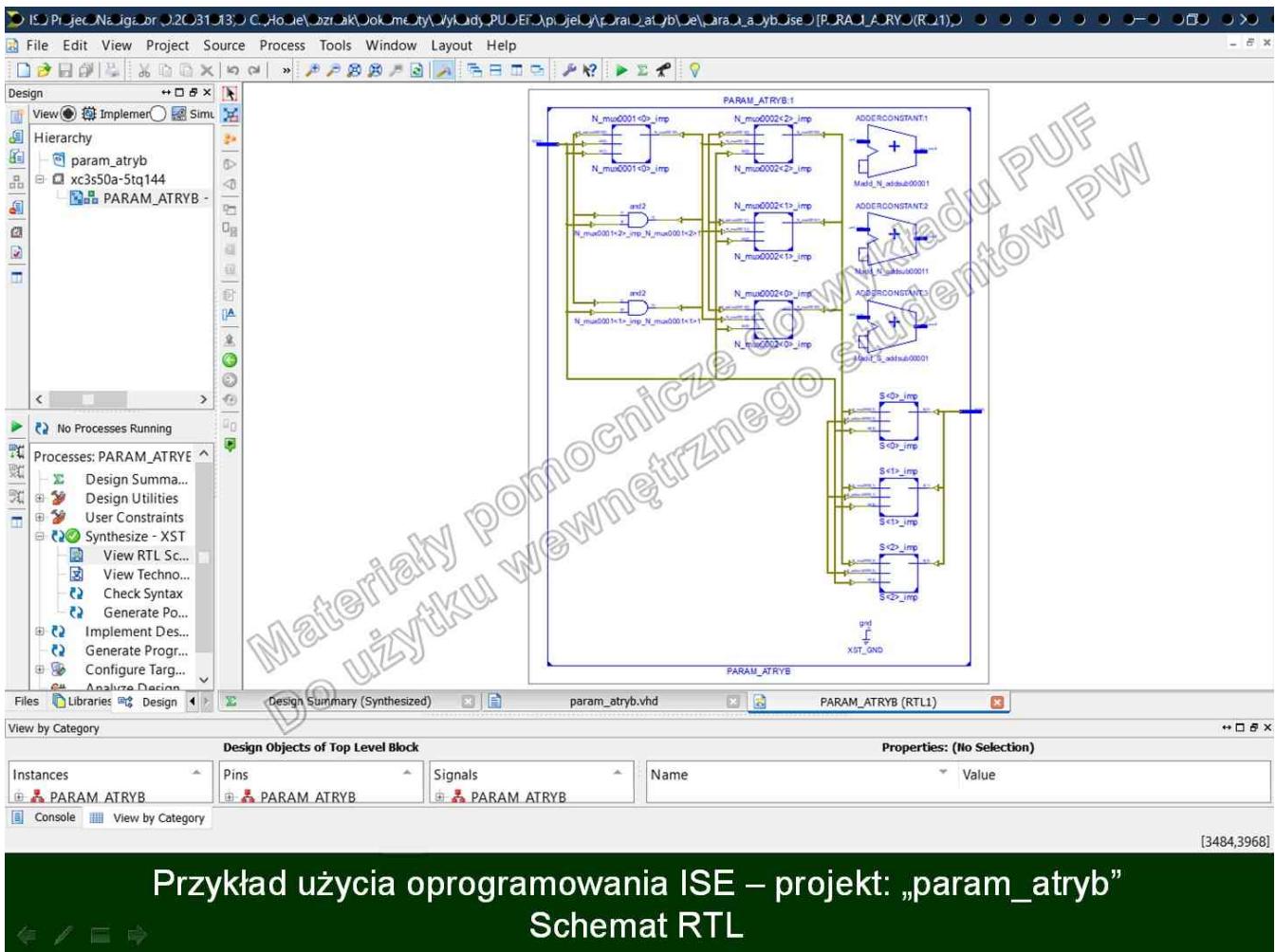
Design Summary (Synthesized) param_atryb.vhd

Console

```
Process "Synthesize - XST" completed successfully
```

Przykład użycia oprogramowania ISE – projekt: „param_atryb”

Plik źródłowy „param_atryb.vhd”



Do użytku pomocykicze do wykładu PUT
Do użytku pomocykicze do wykładu PUT

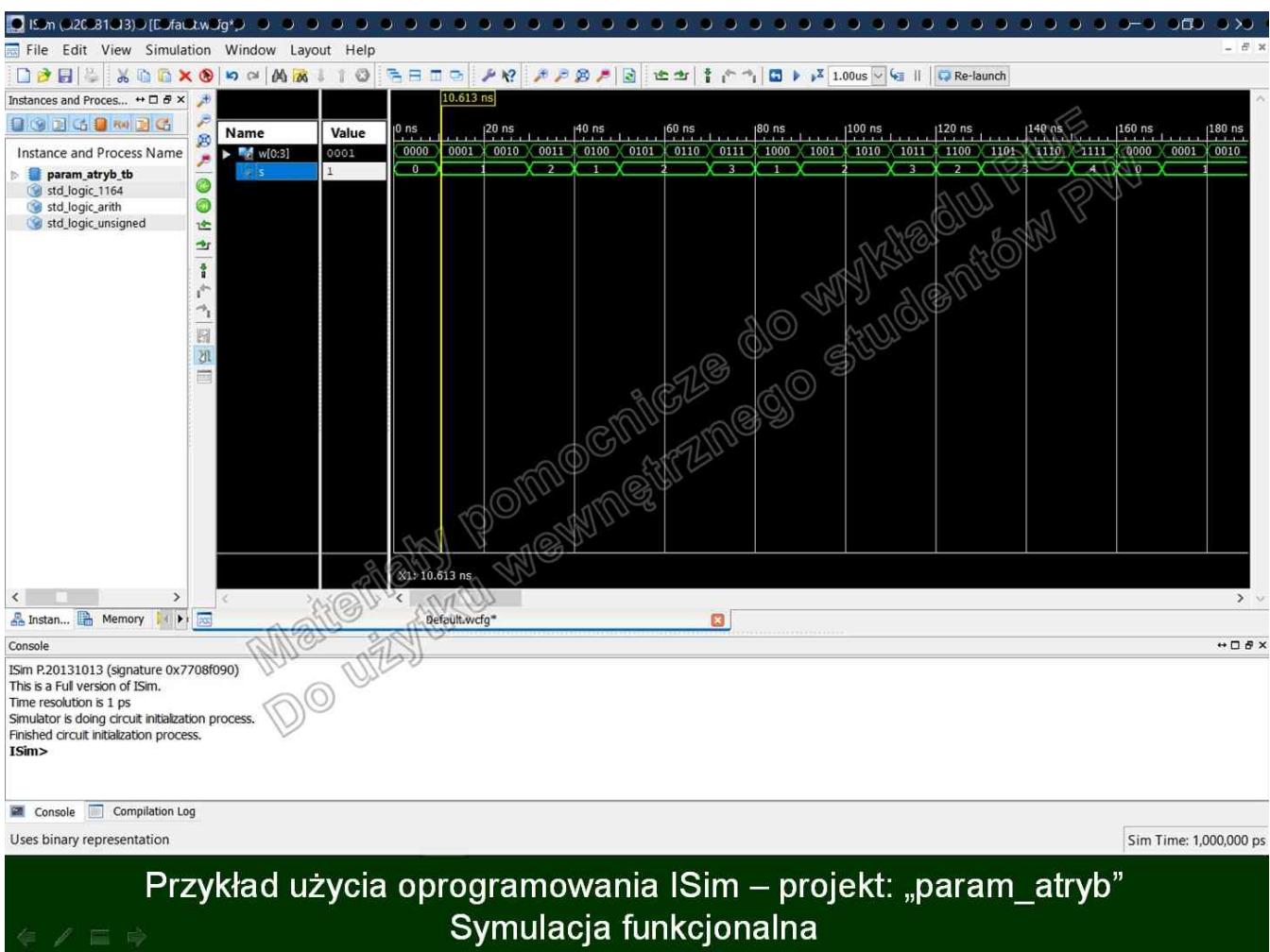
```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4
5 entity PARAM_ATRYB_TB is
6 end PARAM_ATRYB_TB;
7
8 architecture behavioural of PARAM_ATRYB_TB is
9
10 signal W :std_logic_vector(0 to 3);
11 signal S :natural range 0 to 4;
12
13 begin
14
15 process is
16 begin
17 W <= (others =>'0');
18 for i in 0 to 2**W'length-1 loop
19 wait for 10 ns;
20 W <= W + 1;
21 end loop;
22 end process;
23
24 param_atryb_inst: entity work.PARAM_ATRYB(cialo)
25 port map(
26 W => W,
27 S => S
28 );
29
30 end behavioural;
31
32

```

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

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



Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

- Wybrane miejsca podejmowania stałych w:

- definicjach typów/podtypów (np. rozmiary wektorów, zakresy)
- definicji zmiennych/sygnalów (np. rozmiary wektorów, zakresy)
- definicji portów sprzęgu (np. rozmiary wektorów, zakresy)
- wyrażeniach (arytmetycznych, logicznych, relacji)

- Wybrane metody przekazywania stałych poprzez:

- użycie atrybutów (udostępnienie aktualnych atrybutów typu)
- wywołanie podprogramu (przekazanie aktualnych argumentów)



ISE Text Editor (P-2015-10-fs) - (param_fun.vhd)

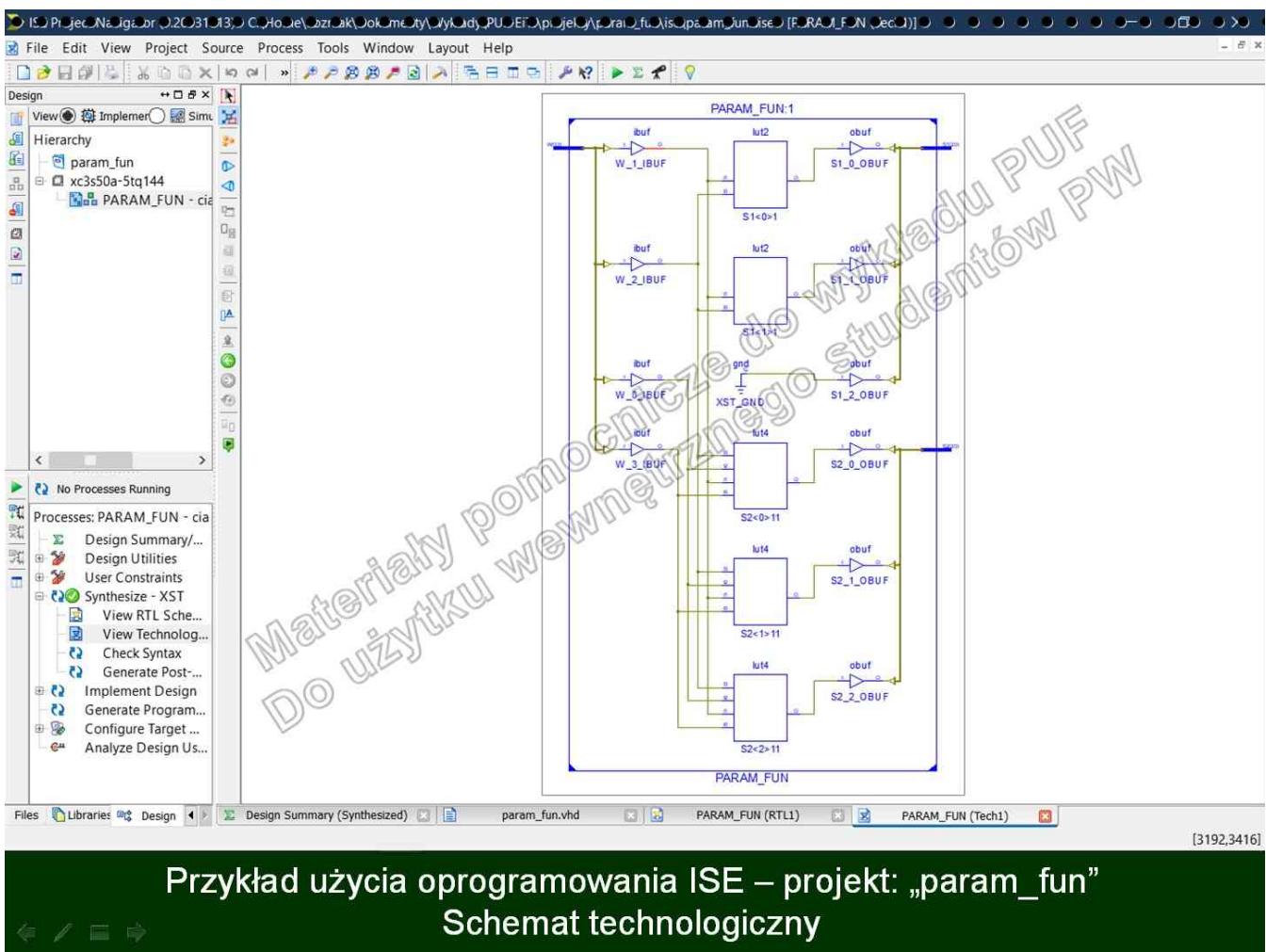
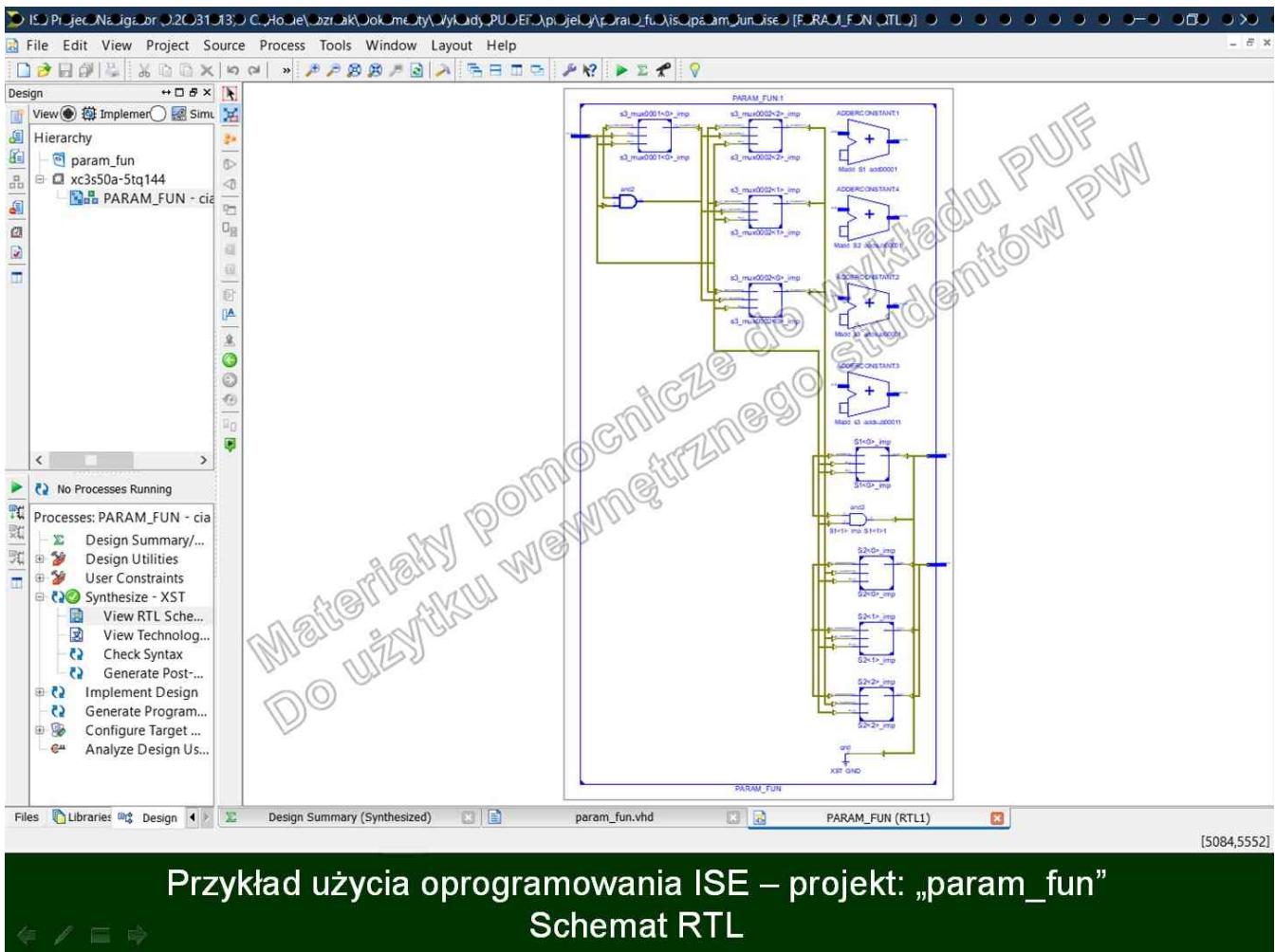
File Edit View Window Layout Help

```
library ieee; -- klauzula dostępu do biblioteki 'IEEE'
use ieee.std_logic_1164.all; -- dołączanie całego pakietu 'STD_LOGIC_1164'

entity PARAM_FUN is -- deklaracja sprzągu 'PARAM_FUN'
  port (
    W      : in std_logic_vector(0 to 3); -- wyjście danych 'W'
    S1     : out natural range 0 to 4; -- wyjście danych 'S1'
    S2     : out natural range 0 to 4 -- wyjście danych 'S2'
  );
end PARAM_FUN;
architecture cialo of PARAM_FUN is -- deklaracja ciała 'cialo' architektury
begin
  function sum(a :std_logic_vector) return natural is -- deklaracja funkcji 'sum' z argumentem 'a'
    variable s :natural range 0 to a'length; -- utworzenie zmiennej 's' w funkcji
    begin -- czesc wykonawcza funkcji
      s := 0; -- wykonywanie sumy jedynek 's'
      for i in a'range loop -- petla zmiennej 'i' po zakresie 'a'
        if a(i)='1' then -- sprawdzenie, czy i-ty bit 'a' zawiera '1'
          s := s + 1; -- zwiększenie sumy jedynek 's' o 1
        end if;
      end loop;
      return(s); -- zakończenie instrukcji wyboru
    end function; -- zakończenie petli
    signal S :std_logic_vector(W'high downto W'low); -- utworzenie sygnalu 'S' wzgledem wektora 'W'
begin
  S1 <= sum(W(W'left+1 to W'right-1)); -- poczatek czesci wykonawczej
  S <= W; -- wywołanie funkcji 'sum' dla czesci wektora 'W'
  S2 <= sum(S); -- podstawienie 'W' do sygnalu 'S'
  end cialo; -- wywołanie funkcji 'sum' dla całego wektora 'S'
end;
```

param_fun.vhd

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



Do wykładowej pomocy technicznej do wykładu PUF

```

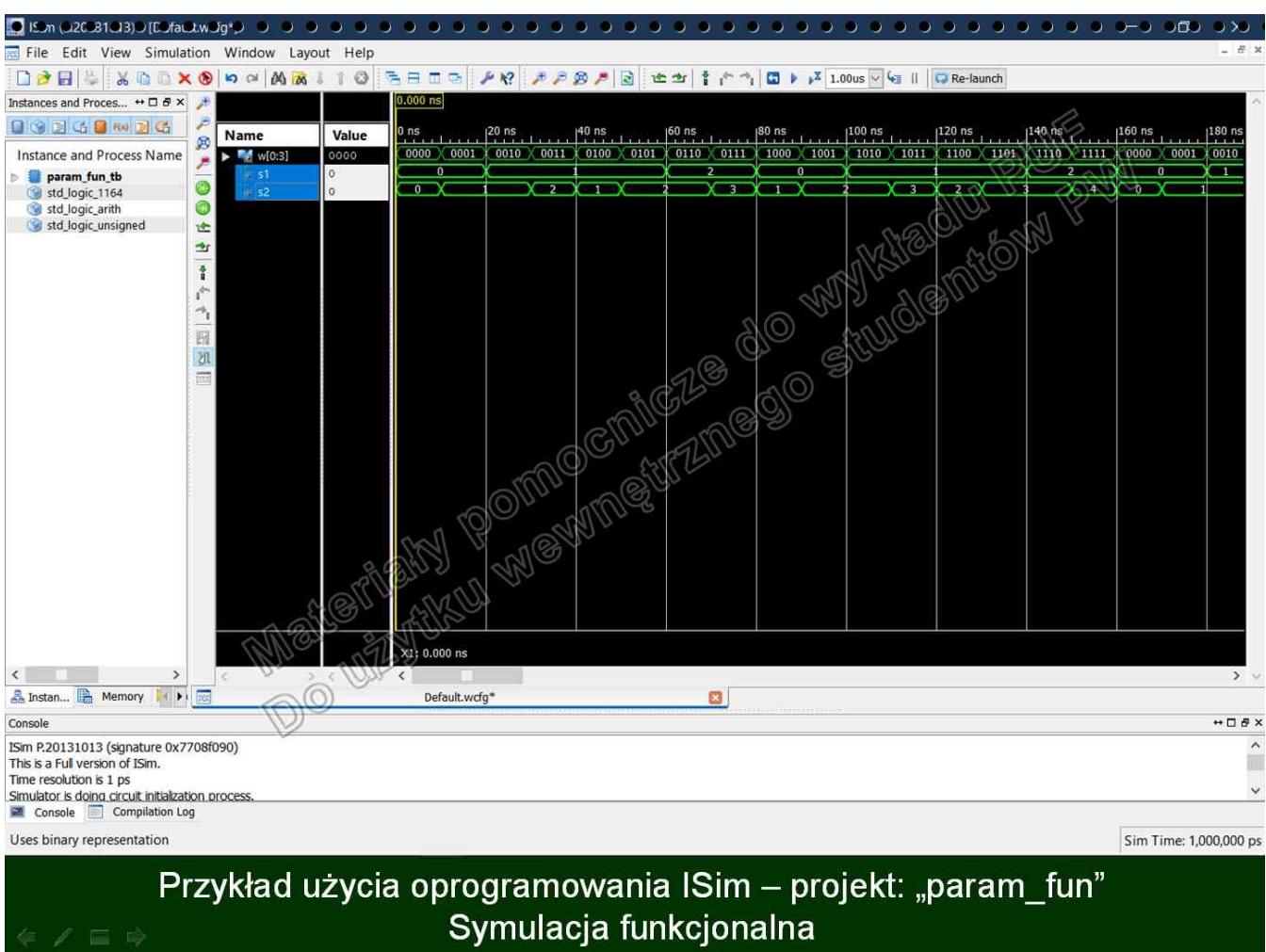
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4
5 entity PARAM_FUN_TB is
6 end PARAM_FUN_TB;
7
8 architecture behavioural of PARAM_FUN_TB is
9
10 signal W :std_logic_vector(0 to 3);
11 signal S1 :natural range 0 to 4;
12 signal S2 :natural range 0 to 4;
13
14 begin
15
16 process is
17 begin
18   W <= (others =>'0');
19   for i in 0 to 2**W'length-1 loop
20     wait for 10 ns;
21     W <= W + 1;
22   end loop;
23 end process;
24
25 param_fun_inst: entity work.PARAM_FUN(cialo)
26   port map
27     W => W,
28     S1 => S1,
29     S2 => S2
30   ;
31
32 end behavioural;

```

-- klauzula dostepu do biblioteki 'IEEE'
-- dolaczenie calego pakietu 'STD LOGIC 1164'
-- dolaczenie calego pakietu 'STD LOGIC_UNSIGNED'
-- pusty sprzeg projektu symulacji
-- cialo architektoniczne projektu
-- symulowane wejście danych 'W'
-- obserwowane wyjście danych 'S1'
-- obserwowane wyjście danych 'S2'
-- początek części wykonawczej
-- proces bezwarunkowy
-- część wykonawcza procesu
-- wyzerowanie wektora 'W'
-- petla po wartosciach wektora 'W'
-- odczekanie 20 ns
-- zwiększenie wektora 'W' o 1
-- zakonczenie petli
-- zakonczenie procesu
-- instancja projektu 'PARAM_FUN'
-- mapowanie portow
-- przypisanie sygnalu 'W' do portu 'W'
-- przypisanie sygnalu 'S1' do portu 'S1'
-- przypisanie sygnalu 'S2' do portu 'S2'
-- zakonczenie ciala architektonicznego

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

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



Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

- Wybrane miejsca podejmowania stałych w:

- definicjach typów/podtypów (np. rozmiary wektorów, zakresy)
- definicji zmiennych/sygnalów (np. rozmiary wektorów, zakresy)
- definicji portów sprzęgu (np. rozmiary wektorów, zakresy)
- wyrażeniach (arytmetycznych, logicznych, relacji)

- Wybrane metody przekazywania stałych poprzez:

- użycie atrybutów (udostępnienie aktualnych atrybutów typu)
- wywołanie podprogramu (przekazanie aktualnych argumentów)
- włączenie pakietu (udostępnienie stałych z definicji pakietu)



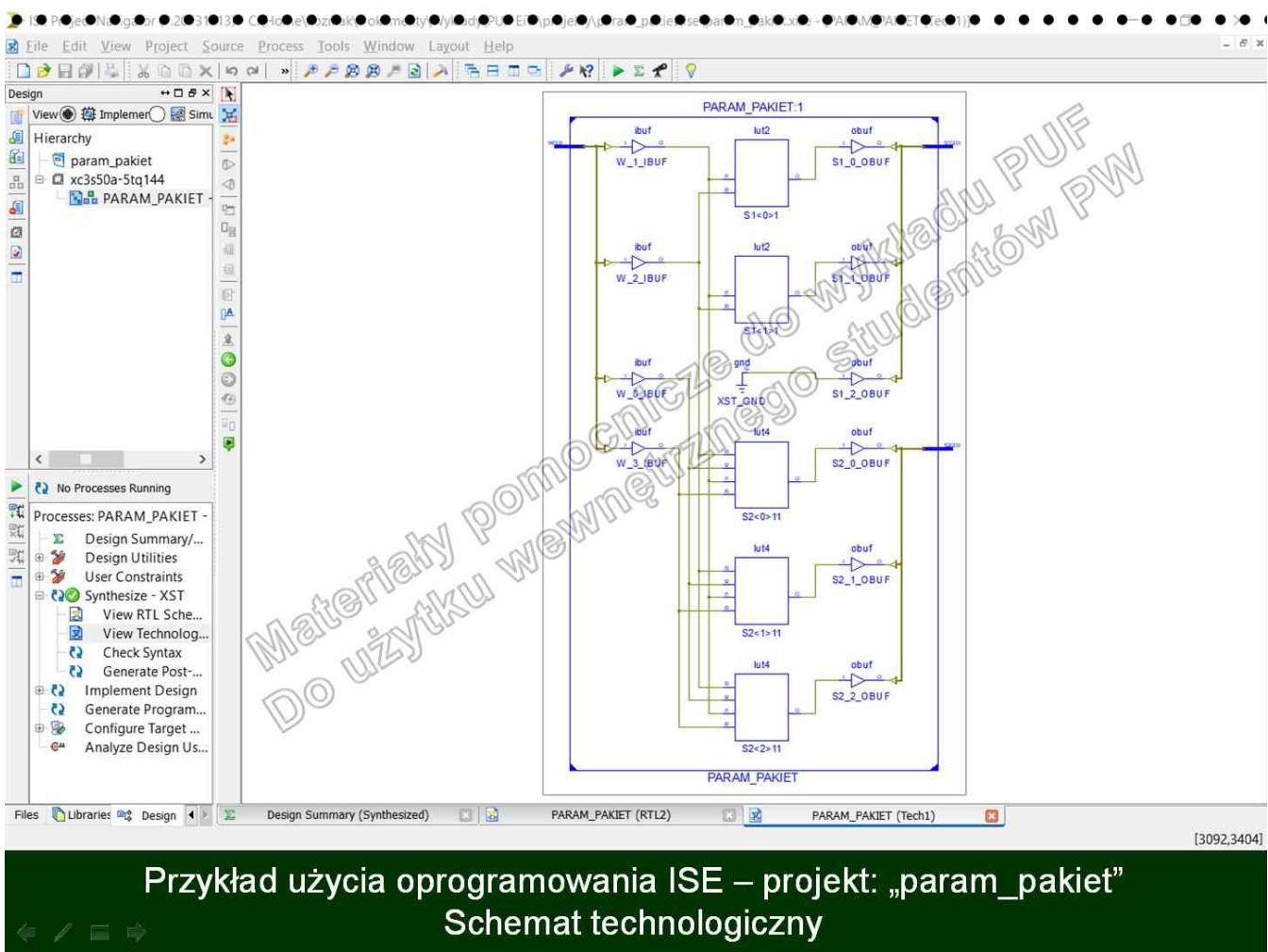
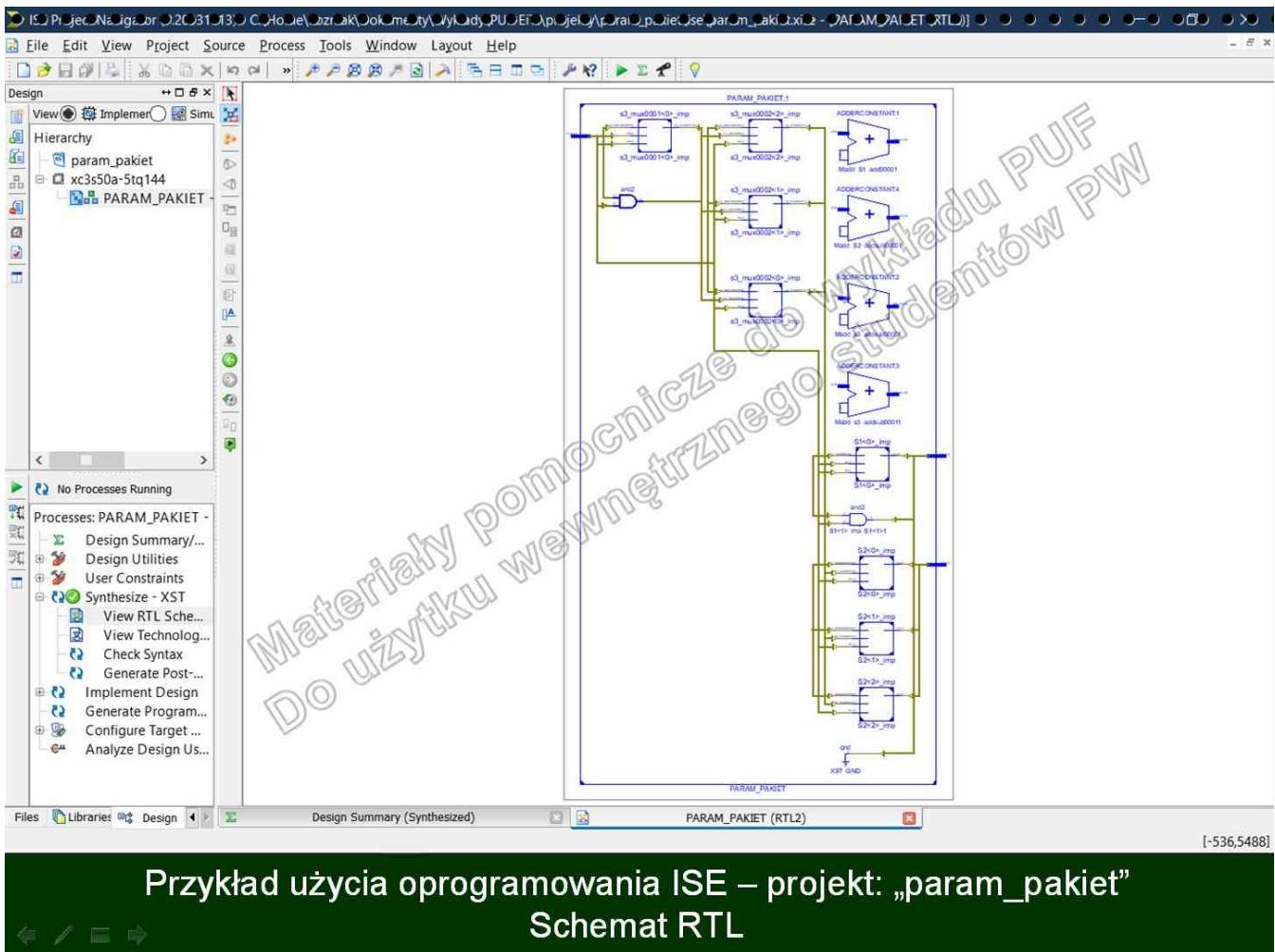
ISE Text Editor (P-2015-10-f5) - (param_pakiet.vhd*)

File Edit View Window Layout Help

```
1 -- biblioteka STD jest włączana automatycznie do projektu
2 package pakiet is
3     constant ROZMIAR :natural := 4;
4 end package pakiet;
5
6 library ieee;
7 use ieee.std_logic_1164.all;
8 library work;
9 use work.pakiet.all;
10
11 entity PARAM_PAKIET is
12     port (
13         W      :in std_logic_vector(0 to ROZMIAR-1); -- wyjście danych 'W'
14         S1    :out natural range 0 to ROZMIAR;        -- wyjście danych 'S1'
15         S2    :out natural range 0 to ROZMIAR         -- wyjście danych 'S2'
16     );
17 end PARAM_PAKIET;
18
19 architecture cialo of PARAM_PAKIET is
20     function sum(a :std_logic_vector) return natural is
21         variable s :natural range 0 to a'length;
22     begin
23         s := 0;
24         for i in a'range loop
25             if a(i)='1' then
26                 s := s + 1;
27             end if;
28         end loop;
29         return(s);
30     end function;
31
32     signal S :std_logic_vector(W'high downto W'low);
33 begin
34
35     S1 <= sum(W(W'left+1 to W'right-1));
36     S  <= W;
37     S2 <= sum(S);
38
39 end cialo;
```

param_pakiet.vhd*

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



Przykład użycia oprogramowania ISE – projekt: „param_pakiet”

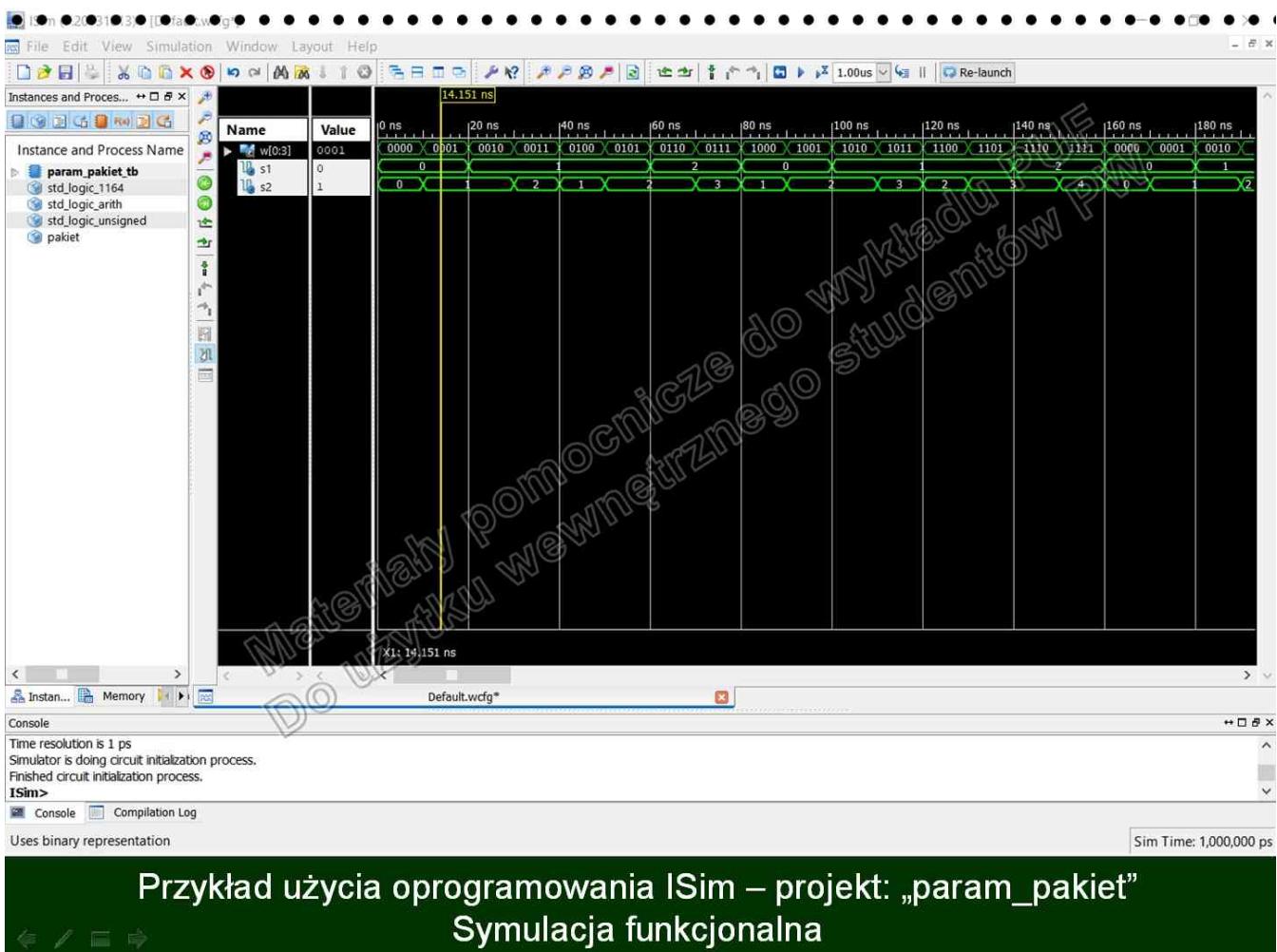
Plik źródłowy „param_pakiet_TB.vhd”

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 library work;
5 use work.pakiet.all;
6
7 entity PARAM_PAKIET_TB is
8 end PARAM_PAKIET_TB;
9
10 architecture behavioural of PARAM_PAKIET_TB is
11
12 signal W      :std_logic_vector(0 to ROZMIAR-1);
13 signal S1     :natural range 0 to ROZMIAR;
14 signal S2     :natural range 0 to ROZMIAR;
15
16 begin
17
18 process is
19 begin
20   W <= (others =>'0');
21   for i in 0 to 2**W'length-1 loop
22     wait for 10 ns;
23     W <= W + 1;
24   end loop;
25 end process;
26
27 param_pakiet_inst: entity work.PARAM_PAKIET(cialo)
28   port map (
29     W => W,
30     S1 => S1,
31     S2 => S2
32 );
33
34 end behavioural;
35
36

```

-- klauzula dostępu do biblioteki 'IEEE'
-- dodanie całego pakietu 'STD_LOGIC_1164'
-- dodanie całego pakietu 'STD_LOGIC_UNSIGNED'
-- opcjonalna klauzula dostępu do biblioteki 'work'
-- dodanie całego pakietu 'pakiet'
-- pusty przed projekt symulacji
-- ciało architektoniczne projektu
-- symulowane wejście danych 'W'
-- obserwowane wyjście danych 'S1'
-- obserwowane wyjście danych 'S2'
-- początek części wykonawczej
-- proces bezwarunkowy
-- część wykonawcza procesu
-- wyczekanie wektora 'W'
-- pętla po wartościach wektora 'W'
-- zwiększenie wektora 'W' o 1
-- zakończenie pętli
-- zakończenie procesu
-- instancja projektu 'PARAM_PAKIET'
-- mapowanie portów
-- przypisanie sygnału 'W' do portu 'W'
-- przypisanie sygnału 'S1' do portu 'S1'
-- przypisanie sygnału 'S2' do portu 'S2'
-- zakończenie ciała architektonicznego



Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

- Wybrane miejsca podejmowania stałych w:

- definicjach typów/podtypów (np. rozmiary wektorów, zakresy)
- definicji zmiennych/sygnalów (np. rozmiary wektorów, zakresy)
- definicji portów sprzęgu (np. rozmiary wektorów, zakresy)
- wyrażeniach (arytmetycznych, logicznych, relacji)

- Wybrane metody przekazywania stałych poprzez:

- użycie atrybutów (udostępnienie aktualnych atrybutów typu)
- wywołanie podprogramu (przekazanie aktualnych argumentów)
- włączenie pakietu (udostępnienie stałych z definicji pakietu)
- instancję bezpośrednią (przekazanie aktualnych argumentów)



The screenshot shows the Xilinx ISE Design Suite interface. On the left, there's a hierarchical tree view under 'Design' showing a project named 'param_inst' containing an entity 'PARAM_INST'. The main window displays the VHDL code for this entity. The code defines an entity 'PARAM_INST' with a single port containing three signals: W (input std_logic_vector), S1 (output natural), and S2 (output natural). It also contains an architecture 'cialo' of 'PARAM_INST' which includes a function 'sum' that calculates the sum of bits in 'W' and assigns it to 'S1'. The code is annotated with comments explaining each part.

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity PARAM_INST is
5     port (
6         W      :in std_logic_vector;
7         S1    :out natural;
8         S2    :out natural
9     );
10 end PARAM_INST;
11
12 architecture cialo of PARAM_INST is
13
14     function sum(a :std_logic_vector) return natural is
15         variable s :natural range 0 to a'length;
16     begin
17         s := 0;
18         for i in a'range loop
19             if a(i)'=1' then
20                 s := s + 1;
21             end if;
22         end loop;
23         return(s);
24     end function;
25
26     signal S :std_logic_vector(W'high downto W'low);
27
28 begin
29
30     S1 <= sum(W(W'left+1 to W'right-1));
31     S <= W;
32     S2 <= sum(S);
33
34 end cialo;
35
36
```

-- klauzula dostępu do biblioteki 'IEEE'
-- dodanie całego pakietu 'STD_LOGIC_1164'
-- deklaracja sprzęgu 'PARAM_INST'
-- deklaracja portów
-- wyjście danych 'W'
-- wyjście danych 'S1'
-- wyjście danych 'S2'
-- deklaracja ciała 'cialo' architektury
-- deklaracja funkcji 'sum' z argumentem 'a'
-- utworzenie zmiennej 's' w funkcji
-- część wykonawcza funkcji
-- wyczerpanie sumy jedynek 's'
-- pętla zmiennej 'i' po zakresie 'a'
-- sprawdzenie, czy i-ty bit 'a' zawiera '1'
-- zwiększenie sumy jedynek 's' o 1
-- zakończenie instrukcji wyboru
-- zakończenie pętli
-- zwrócenie wartości sumy jedynek 's'
-- zakończenie funkcji
-- utworzenie sygnału 'S' względem wektora 'W'
-- początek części wykonawczej
-- wywołanie funkcji 'sum' dla części wektora 'W'
-- podstawienie 'W' do sygnału 'S'
-- wywołanie funkcji 'sum' dla całego wektora 'S'
-- zakończenie ciała architektonicznego

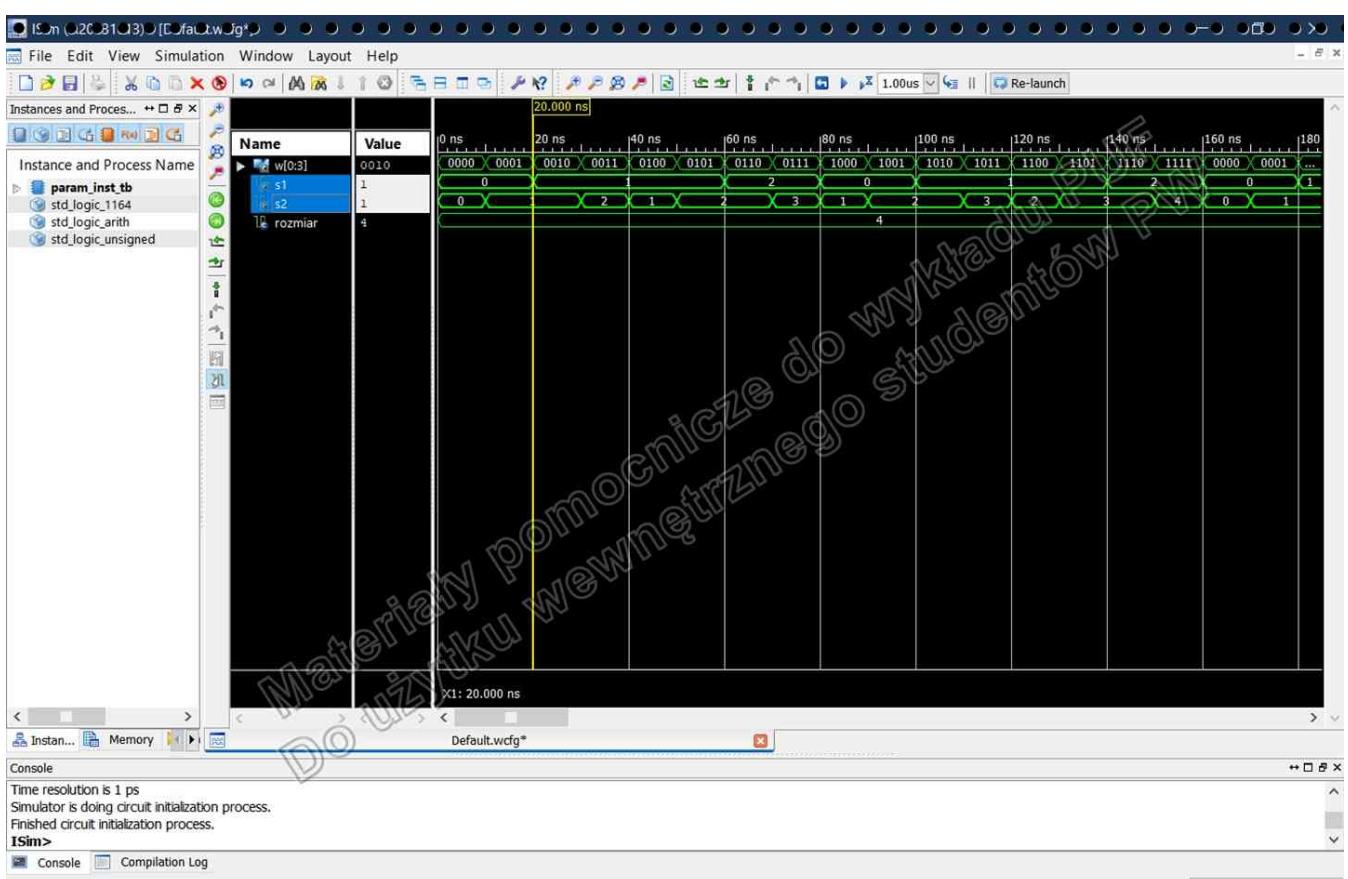
Przykład użycia oprogramowania ISE – projekt: „param_inst”

Plik źródłowy „param_inst.vhd”

The screenshot shows the Quartus II software interface with the following details:

- File Menu:** File, Edit, View, Project, Source, Process, Tools, Window, Layout, Help.
- Toolbars:** Standard toolbar with icons for file operations, zoom, and search.
- Design View:** Shows the project hierarchy:
 - param_inst
 - xc3s50a-5tq144
 - PARAM_INST_TB
- Editor Area:** Displays the VHDL code for the testbench. The code defines an entity PARAM_INST_TB with an architecture behavioural. It includes a constant ROZMIAR, three signal declarations (W, S1, S2), a process loop that increments W from 0 to ROZMIAR-1, and a port map for connecting the testbench to the project's entity.
- Right Margin:** Contains comments explaining the code line by line.
- Status Bar:** Shows "Design Summary" and file names "param_inst.vhd" and "param_inst_tb.vhd".
- Bottom Bar:** Shows "Ln 8 Col 1 VHDL".

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



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

Podstawowe elementy standardu VHDL

Wybrane metody parametryzacji projektu

- Wybrane miejsca tworzenia stałych w:

- definicji pakietu (zasięg w zakresie użycia pakietu)
- definicji ciała pakietu (zasięg w ciele pakietu)
- definicji architektury (zasięg w ciele jednostki projektowej)
- definicji procesu (zasięg w ciele procesu)
- definicji podprogramu (zasięg w ciele podprogramu)

- Wybrane miejsca podejmowania stałych w:

- definicjach typów/podtypów (np. rozmiary wektorów, zakresy)
- definicji zmiennych/sygnalów (np. rozmiary wektorów, zakresy)
- definicji portów sprzęgu (np. rozmiary wektorów, zakresy)
- wyrażeniach (arytmetycznych, logicznych, relacji)

- Wybrane metody przekazywania stałych poprzez:

- użycie atrybutów (udostępnienie aktualnych atrybutów typu)
- wywołanie podprogramu (przekazanie aktualnych argumentów)
- włączenie pakietu (udostępnienie stałych z definicji pakietu)
- instancję bezpośrednią (przekazanie aktualnych argumentów)
- instancję parametryzowaną (przekazanie aktualnych parametrów)



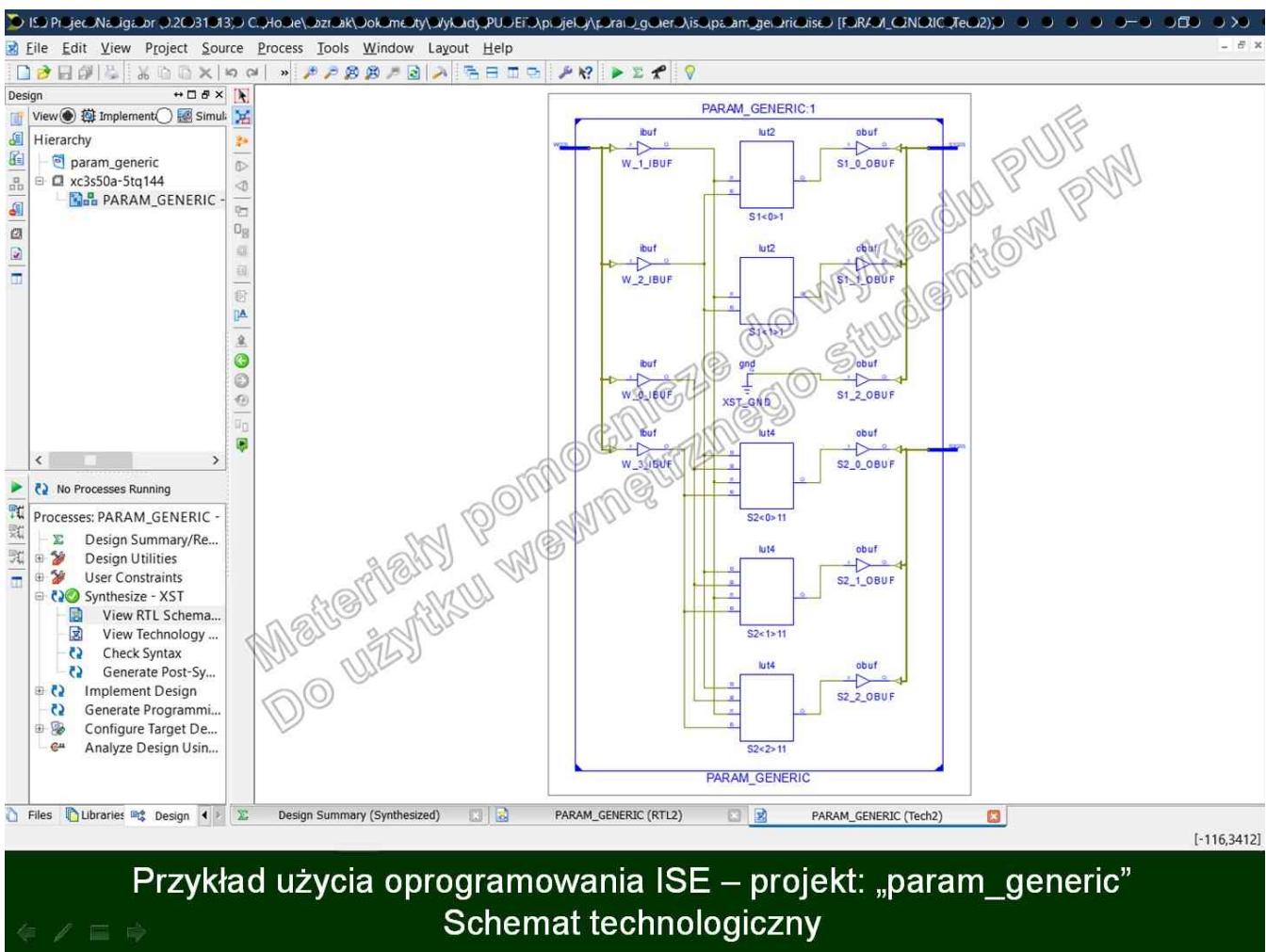
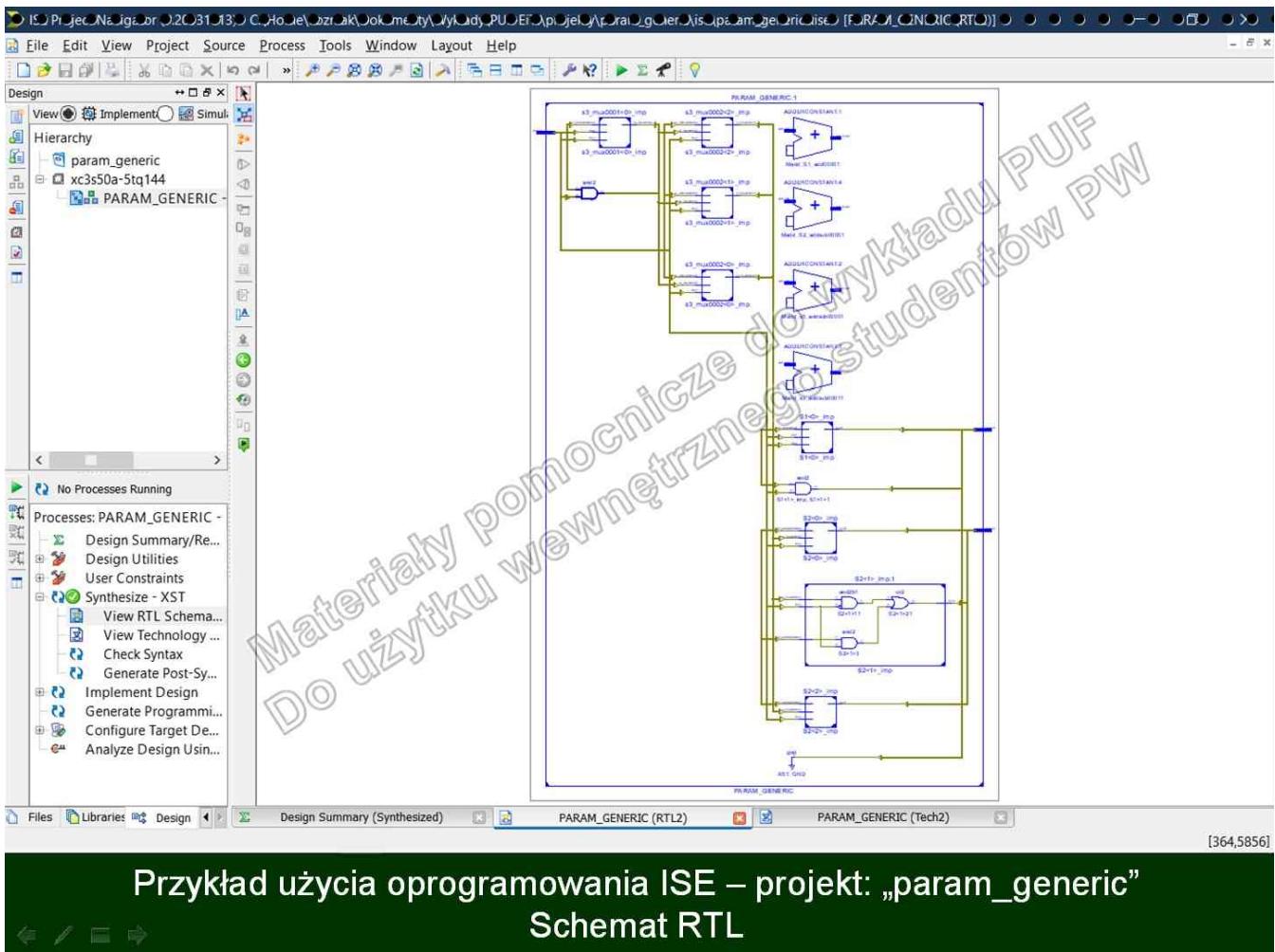
```
library ieee;
use ieee.std_logic_1164.all;

entity PARAM_GENERIC is
    generic (
        ROZMIAR : natural := 4
    );
    port (
        W      : in std_logic_vector(0 to ROZMIAR-1);
        S1    : out natural range 0 to ROZMIAR;
        S2    : out natural range 0 to ROZMIAR
    );
end PARAM_GENERIC;

architecture cialo of PARAM_GENERIC is
begin
    function sum(a : std_logic_vector) return natural is
        variable s : natural range 0 to a'length;
    begin
        s := 0;
        for i in a'range loop
            if a(i)'=1' then
                s := s + 1;
            end if;
        end loop;
        return(s);
    end function;
    signal S : std_logic_vector(W'high downto W'low);
begin
    S1 <= sum(W(W'left+1 to W'right-1));
    S  <= W;
    S2 <= sum(S);
end cialo;
```

Przykład użycia oprogramowania ISE – projekt: „param_generic”

Plik źródłowy „param_generic.vhd”



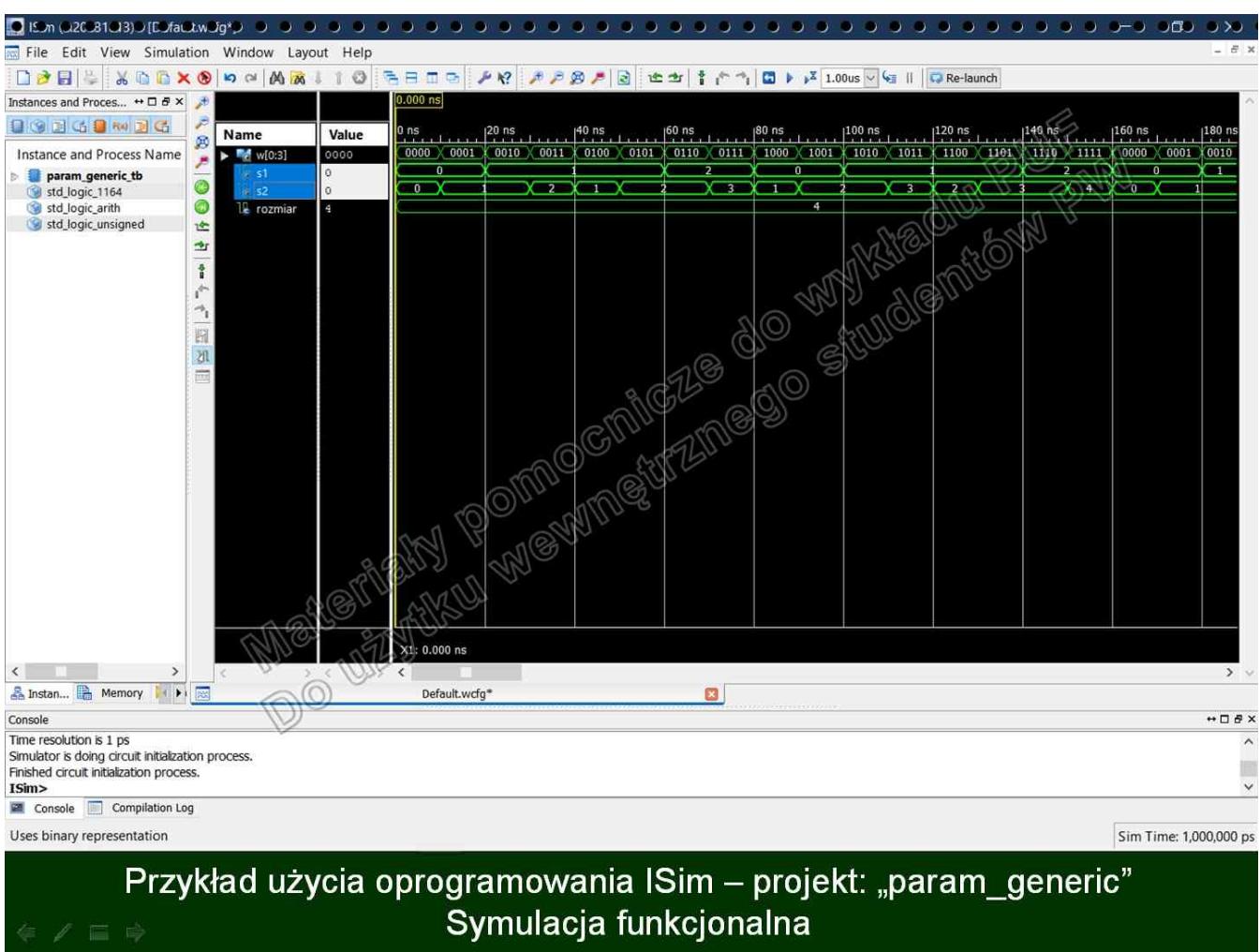
Do użytku wewnętrznego studentów PW

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4
5 entity PARAM_GENERIC_TB is
6 end PARAM_GENERIC_TB;
7
8 architecture behavioural of PARAM_GENERIC_TB is
9
10 constant ROZMIAR :natural := 4;
11 signal W      :std_logic_vector(0 to ROZMIAR-1);
12 signal S1     :natural range 0 to ROZMIAR;
13 signal S2     :natural range 0 to ROZMIAR;
14
15 begin
16
17   process is
18   begin
19     W <= (others =>'0');
20     for i in 0 to 2**W'length-1 loop
21       wait for 10 ns;
22       W <= W + 1;
23     end loop;
24   end process;
25
26   param_generic inst: entity work.PARAM_GENERIC(cialo)
27     generic map (
28       ROZMIAR => ROZMIAR
29     )
30     port map
31     W => W,
32     S1 => S1,
33     S2 => S2
34   );
35
36 end behavioural;
37
38

```

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



Parametryzowany odbiornik szeregowy

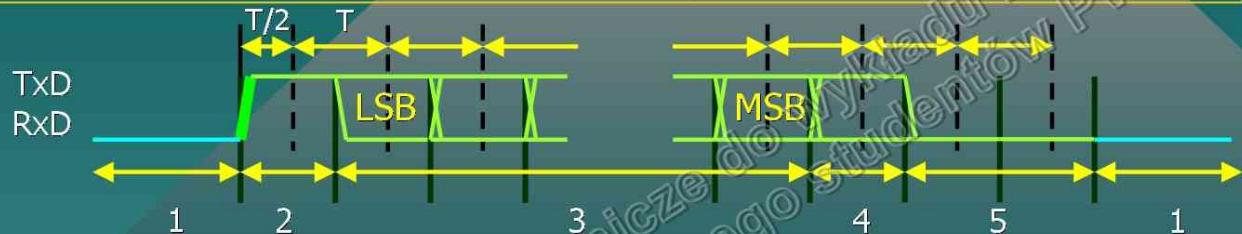
Format transmitowanej ramki danych



- 1: stan bezczynności, sygnał ustawiono na „0”
- 2: bit START o wartości „1” – pierwsze zbocze inicjuje transmisję
- 3-5: bity danej od najmłodszego do najstarszego (stany „0” lub „1”)
- 6: opcjonalny bit kontrolny (stan „0” lub „1”)
- 7,8: bit(y) STOP o wartości „0” – przejście do stanu bezczynności

Parametryzowany odbiornik szeregowy

Format transmitowanej ramki danych



- 1: stan CZEKANIE – oczekивание на начало бита старта
- 2: stan START – прием бита старта
- 3: stan DANA – прием количества B_SLOWA (5-8) битов слова
- 4: stan PARZYSTOSC – прием количества B_PARZYSTOSC (0-1) битов теста
- 5: stan STOP – прием количества B_STOP (1-2) битов остановки

wyznaczenie okresu T trwania jednego bitu danych:

- F_ZEGARA - parametr określający częstotliwość zegara w [Hz]
- L_BODOW – parametr określający liczbę bitów w jednostce czasu

stąd: **T = F_ZEGARA / L_BODOW** - liczba taktów zegara



IS Project Navigator 2.20.31.13 [C:\Holej\Zdziark\Jok_metry\Wykady\PUJET\projekty\serial_rx\serial_rx.vhd]

File Edit View Project Source Process Tools Window Layout Help

Design View Implement Sim

Hierarchy

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 use ieee.std_logic_misc.all;
5
6 entity SERIAL_RX is
7     generic (
8         F_ZEGARA           :positive := 20_000_000;          -- częstotliwość zegara w [Hz]
9         L_MIN_BODOW        :positive := 110;                -- minimalna predkosc nadawania w
10        B_SLOWA             :natural range 5 to 8 := 8;      -- liczba bitow slowa danych (5-8)
11        B_PARZYSTOSCI      :natural range 0 to 1 := 1;      -- liczba bitow parzystosci (0-1)
12        B_STOPOW            :natural range 0 to 2 := 2;      -- liczba bitow stopu (1-2)
13        N_RX                :boolean := FALSE;              -- negacja logiczna sygnalu szeregowego
14        N_SLOWO              :boolean := FALSE;              -- negacja logiczna slowa danych
15    );
16    port (
17        R                  :in  std_logic;                   -- sygnal resetowania
18        C                  :in  std_logic;                   -- zegar taktujacy
19        T_BODU              :in  natural range 0 to F_ZEGARA/L_MIN_BODOW;  -- liczba taktow zegara dla jednej
20        RX                 :out std_logic;                  -- odebranej parzystosci
21        SLOWO               :out std_logic_vector(B_SLOWA-1 downto 0);  -- odebrane slowo danych
22        GOTOWE              :out std_logic;                  -- flaga potwierdzenia odbioru
23        BLAD                :out std_logic;                  -- flaga wykrycia bledu w odbiorze
24    );
25 end SERIAL_RX;
26
27 architecture behavioural of SERIAL_RX is
28
29     signal wejście   :std_logic_vector(0 to 1);          -- podwojny rejestr sygnalu RX
30
31     type ETAP       is (CZEKANIE, START, DANA, PARZYSTOSC, STOP);  -- lista etapow pracy odbiornika
32     signal stan     :ETAP;                                -- rejestr maszyny stanow odbiornika
33
34     constant T_MAX_BODU :positive := F_ZEGARA/L_MIN_BODOW;  -- czas jednego bodu - liczba tak
35     signal l_czasu   :natural range 0 to T_MAX_BODU-1;  -- licznik czasu jednego bodu
36     signal l_bitow   :natural range 0 to B_SLOWA-1;      -- licznik odebranych bitow danych
37
38     signal bufor    :std_logic_vector(SLOWO'range);      -- rejestr kolejno odebranych bitow
39     signal problem   :std_logic;                            -- rejestr (flaga) wykrytego bledu

```

No Processes Running

Processes: SERIAL_RX_TB - t

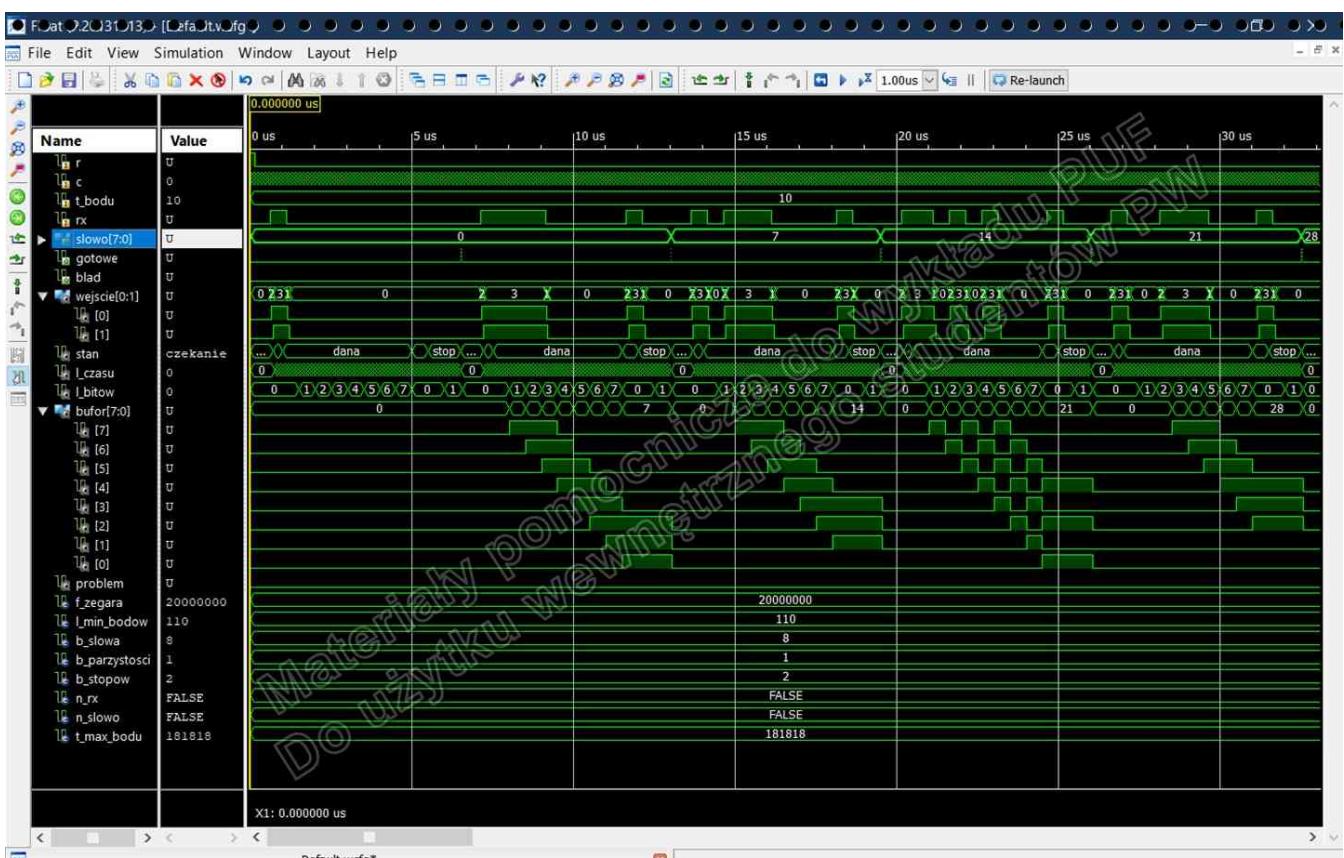
- ISim Simulator
- Behavioral Che...
- Simulate Behav...

Files Libraries Design Design Summary (Synthesized) serial_rx.vhd serial_rx_tb.vhd

Ln 1 Col 1 VHDL

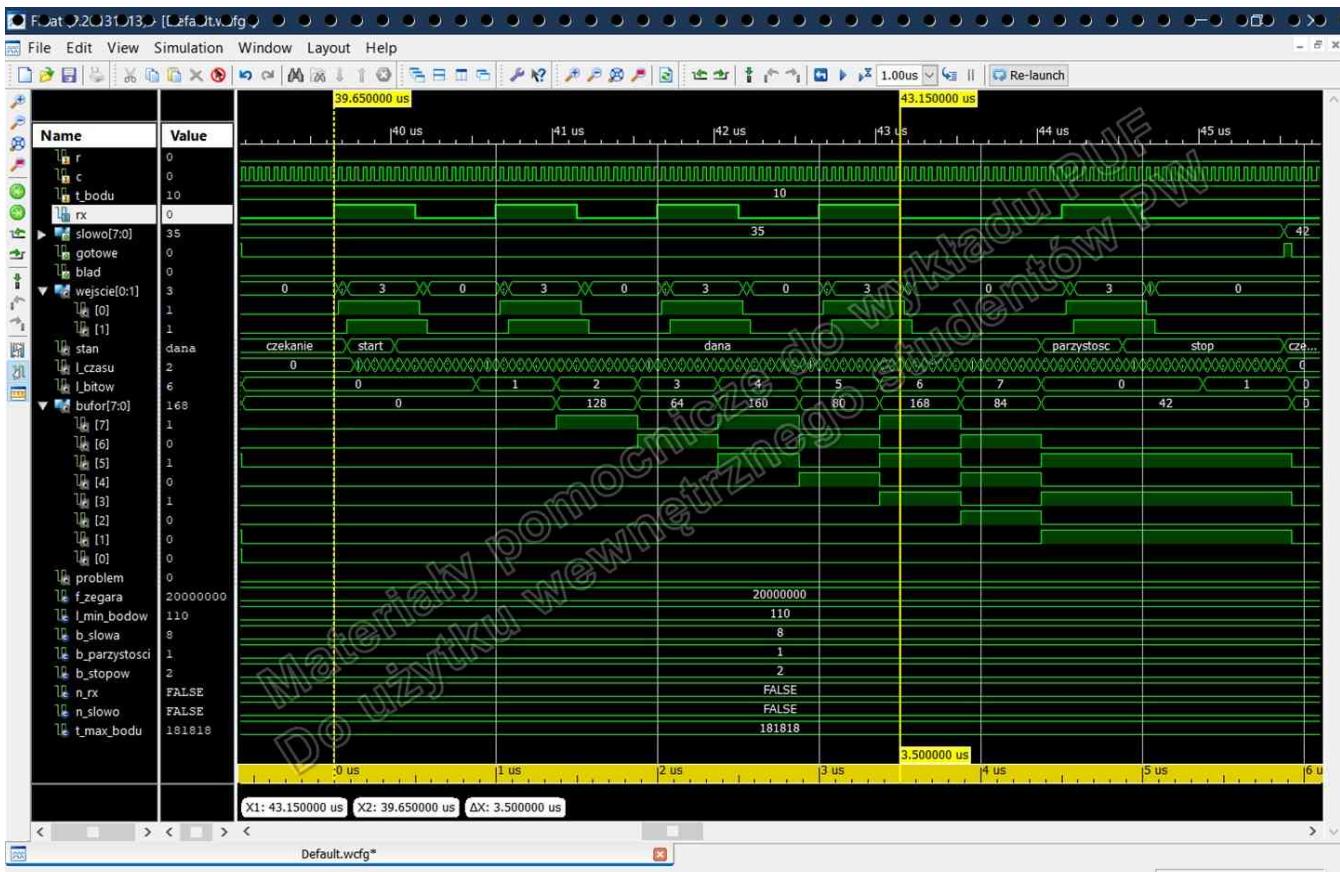
Przykład użycia oprogramowania ISE – projekt: „serial_rx”

Plik źródłowy „serial_rx.vhd” (fragment z entity)

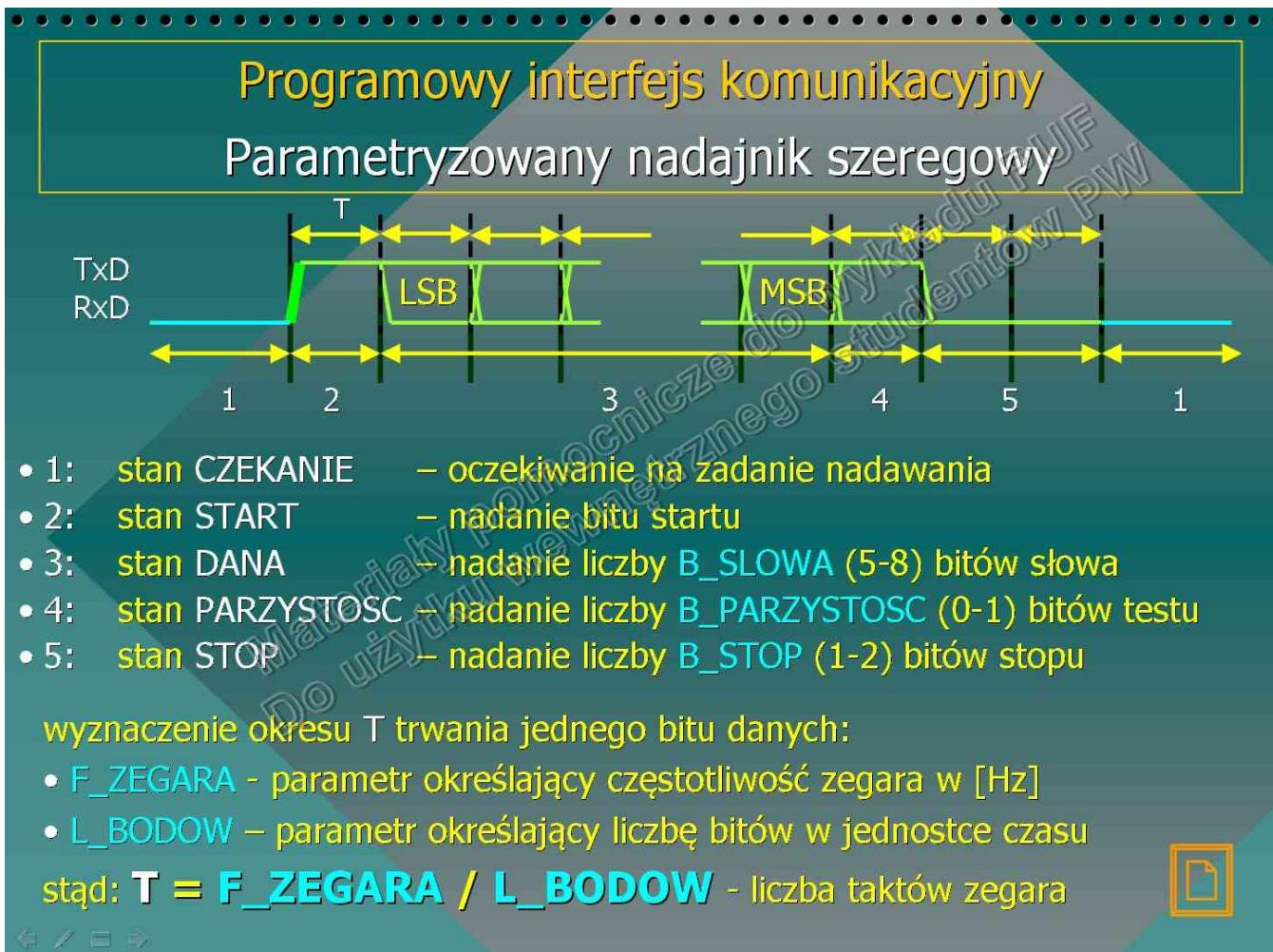


Przykład użycia oprogramowania ISim – projekt: „serial_rx”

Symulacja funkcjonalna (duża skala czasu - odebranie kilku kolejnych słów)



Przykład użycia oprogramowania ISim – projekt: „serial_rx”
 Symulacja funkcjonalna (mała skala czasu - odebranie jednego słowa)



Przykład użycia oprogramowania ISE – projekt: „serial_tx”

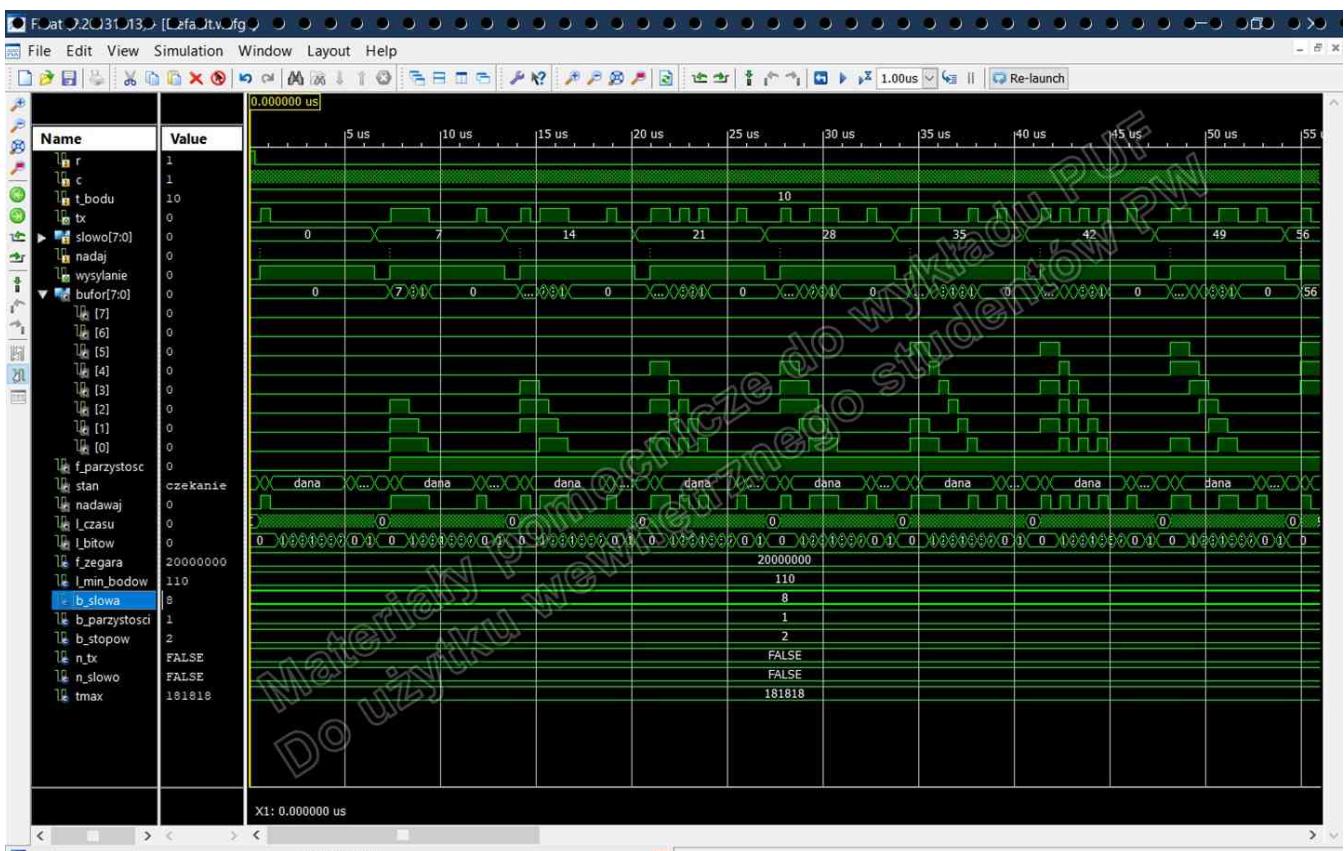
Plik źródłowy „serial_tx.vhd” (fragment z entity)

```

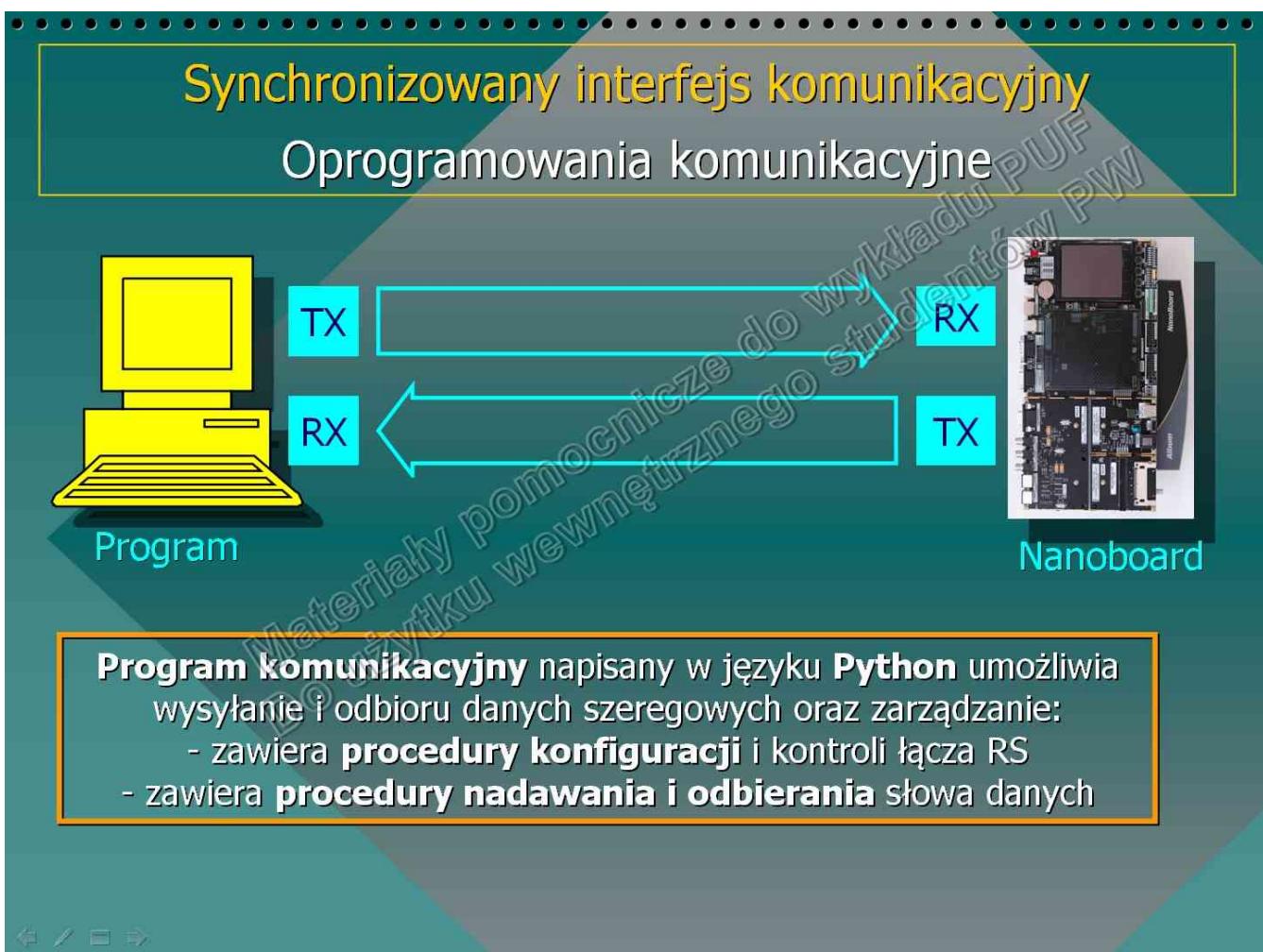
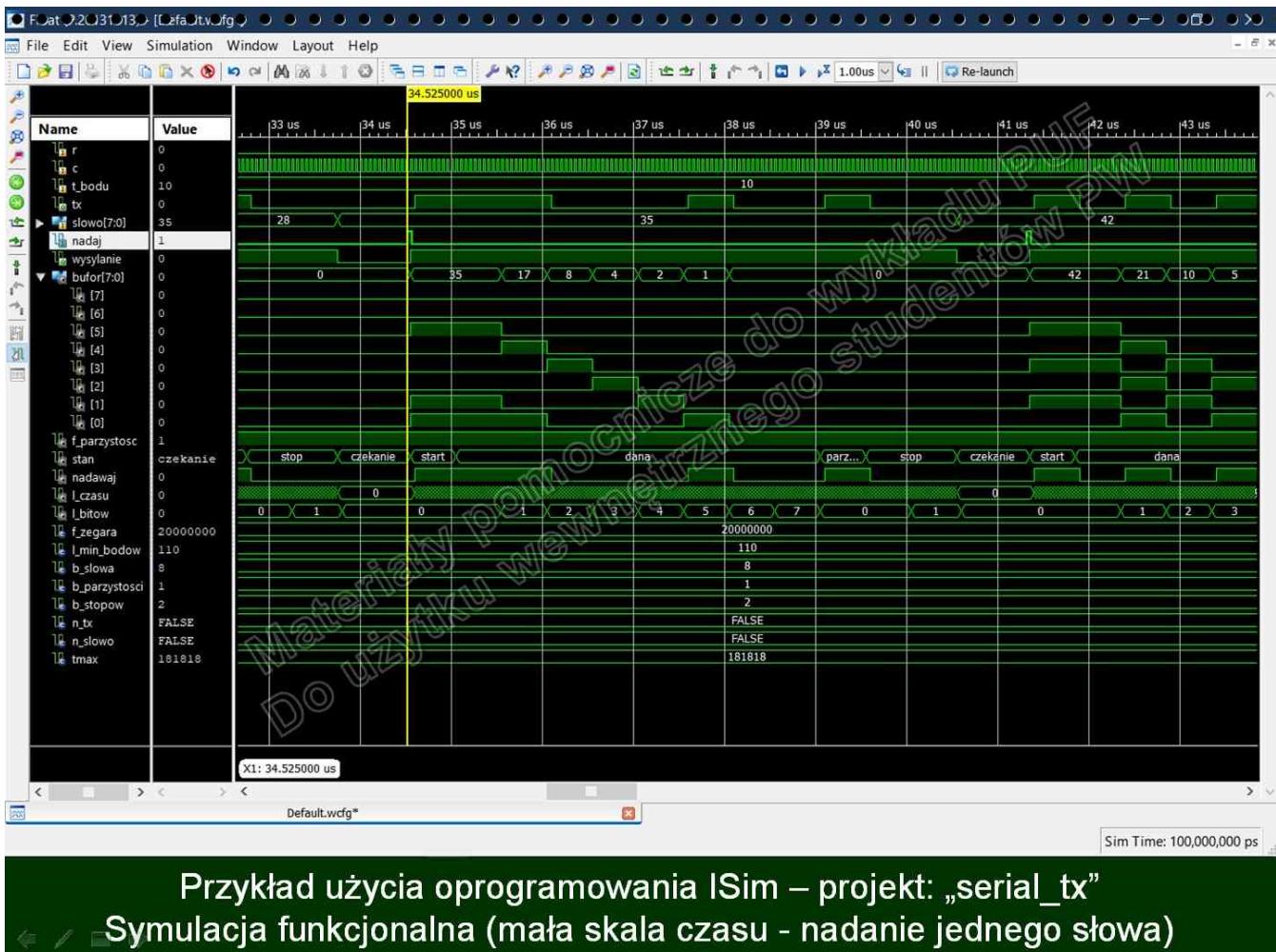
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 use ieee.std_logic_misc.all;
5
6 entity SERIAL_TX is
7     generic (
8         F_ZEGARA           :positive := 20_000_000;
9         L_MIN_BODOW        :positive := 110;
10        B_SLOWA             :positive range 5 to 8 := 8;
11        B_PARZYSTOSCI      :natural range 0 to 1 := 1;
12        B_STOPOW            :natural range 0 to 2 := 2;
13        N_TX                :boolean := FALSE;
14        N_SLOWO              :boolean := FALSE
15    );
16    port (
17        R                  :in std_logic;                                -- sygnał resetowania
18        C                  :in std_logic;                                -- zegar taktujacy
19        T_BODU              :in natural range 0 to F_ZEGARA/L_MIN_BODOW; -- liczba taktow zegara dla jednego bitu
20        TX                 :out std_logic;                               -- wysylany sygнал szeregowy
21        SLOWO              :in std_logic_vector(B_SLOWA-1 downto 0); -- wyslane slowo danych
22        NADAJ               :in std_logic;                                -- flaga zadania nadania
23        WYSYLANIE           :out std_logic;                               -- flaga potwierdzenia wysylani
24    );
25 end SERIAL_TX;
26
27 architecture behavioural of SERIAL_TX is
28
29    signal bufor          :std_logic_vector(SLOWO'range);           -- rejestr kolejno odebranych bitow danych
30    signal f_parzystosc   :std_logic;                                 -- flaga parzystosci
31
32    type ETAP              is (CZEKANIE, START, DANA, PARZYSTOSC, STOP); -- lista etapow pracy odbiornika
33    signal stan             :ETAP;                                     -- rejestr maszyny stanow odbiornika
34
35    signal nadawaj          :std_logic;                                -- wysylany sygnal szeregowy
36
37    constant TMAX           :positive := F_ZEGARA/L_MIN_BODOW;       -- czas jednego bodu - liczba taktow zegara
38    signal l_czasu          :natural range 0 to TMAX-1;             -- licznik czasu jednego bodu
39    signal l_bitow           :natural range 0 to B_SLOWA-1;           -- licznik odebranych bitow danych lub stan
40
41

```

Ln 27 Col 1 VHDL

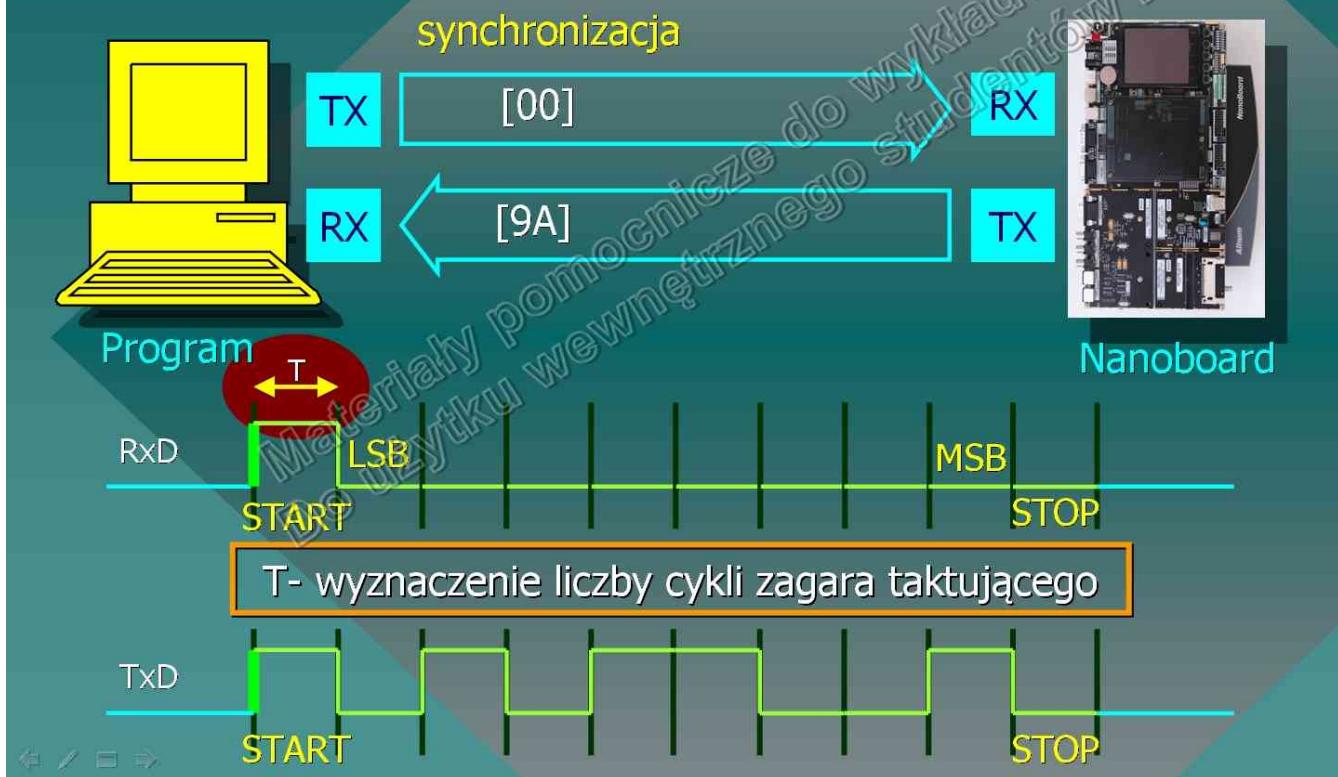


Sim Time: 100,000,000 ps



Synchronizowany interfejs komunikacyjny

Przykład synchronizacji łącza szeregowego



Synchronizowany interfejs komunikacyjny

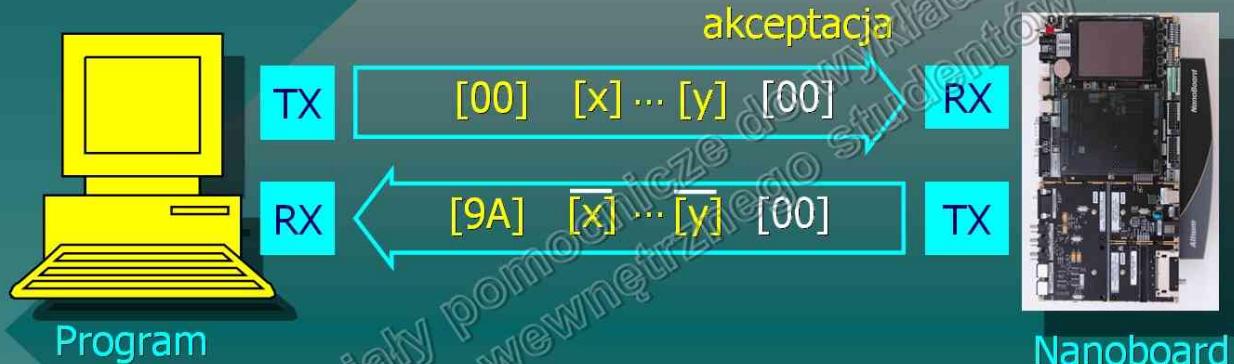
Przykład testowania łącza szeregowego



Testowanie polega na wysłaniu dowolnej liczby słów testowych poza wartością 0 – następuje zwrot wartości zanegowanej

Synchronizowany interfejs komunikacyjny

Przykład finalizacji testowania łącza szeregowego



Akceptacja synchronizacji polega na wysłaniu i odebraniu wartości 0 – od tego momentu łącze obsługuje dane programu

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_misc.all;

entity SERIAL_INTERF is
generic (
    F_ZEGARA :natural := 20000000;
    L_MIN_BODOW :natural := 110;
    B_SLOWA :natural := 8;
    B_PARZYSTOSCI :natural := 1;
    B_STOPOW :natural := 2;
    N_SERIAL :boolean := FALSE;
    N_SLOWO :boolean := FALSE
);
port (
    R :in std_logic; -- sygнал resetowania
    C :in std_logic; -- zegar taktujacy
    RX :in std_logic; -- odbierany sygнал szeregowy
    TX :out std_logic; -- wysylany sygнал szeregowy
    RX_ODEBRANE :out std_logic; -- flaga potwierdzenia odbioru
    RX_SLOWO :out std_logic_vector(B_SLOWA-1 downto 0); -- odebrane slowo danych
    TX_WOLNY :out std_logic; -- flaga gotowosci do nadawania
    TX_NADAJ :in std_logic; -- flaga zadania nadania
    TX_SLOWO :in std_logic_vector(B_SLOWA-1 downto 0); -- wysylane slowo danych
    SYNCH_GOTOWA :out std_logic; -- wysylany sygнал gotowosci
    BLAD_ODBIORU :out std_logic; -- wysylany sygнал bledu odbioru
);
end SERIAL_INTERF;
architecture behavioural of SERIAL_INTERF is
begin
    signal wejscie :std_logic_vector(0 to 1); -- podwojny rejestr sygnalu RX
    constant SLOWO_ZERO :std_logic_vector(B_SLOWA-1 downto 0) := (others => '0'); -- slowo z ustawiona wa
    signal srx_slowo :std_logic_vector(B_SLOWA-1 downto 0); -- odebrane slowo danych
    signal srx_gotowe :std_logic; -- flaga potwierdzenia odbioru
    signal srx_bled :std_logic; -- flaga wskazania bledu w odbiorze
end;
```

Przykład użycia oprogramowania ISE – projekt: „serial_interf”
Plik źródłowy „serial_interf.vhd” (fragment z entity)

