

Programowanie układów FPGA – wykład V

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

- Część wykonawcza ciała jednostki projektowej
 - część deklaracyjna i wykonawcza instrukcji procesu
 - instrukcja warunkowa wyboru instrukcji
 - instrukcja selektywnego wyboru przypisania
 - instrukcje pętli oraz instrukcje sterowania pętlą
 - instrukcje powielania
- Parametryzowany sprzęt jednostki projektowej



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 instrukcje w procesie

Podstawowa składnia warunkowego wyboru instrukcji:

```
[etykieta :] if warunek1 then instrukcja { instrukcja }
{ elsif warunek then instrukcja { instrukcja } }
[ else instrukcja_defaultowa {instrukcja_defaultowa} ]
end if [etykieta];
```

- kilku-wariantowy wybór sekwencji instrukcji przypisania:

```
if sel=1 then
  S1 <= argA + 3 ; S2 <= argB ;
elsif sel=2 then
  S1 <= argB + 5 ; S3 <= argB ;
else
  S3 <= 7 ;
end if ;
```

Podstawowe elementy standardu VHDL

Wybrane instrukcje w procesie

Podstawowa składnia warunkowego wyboru instrukcji:

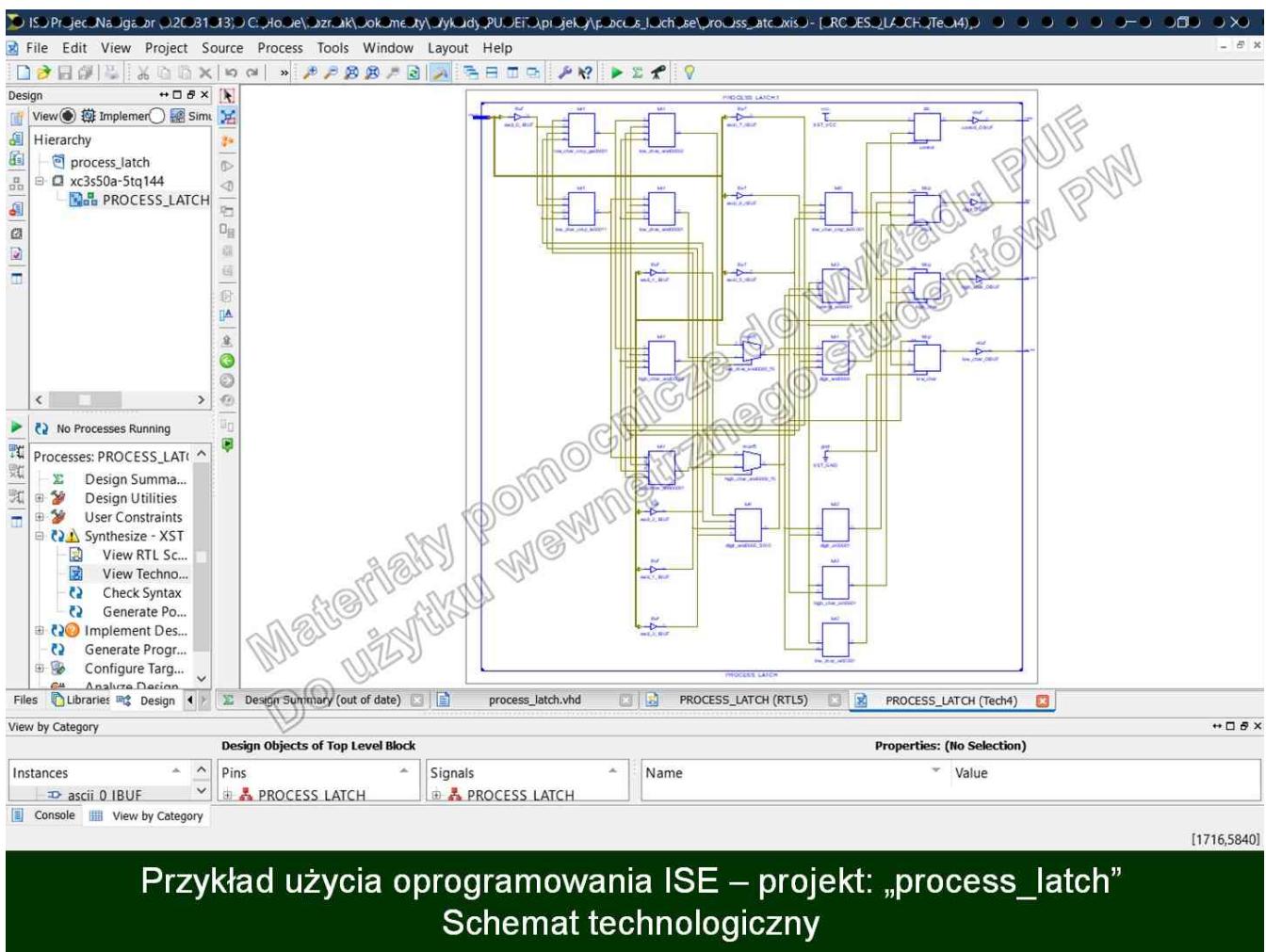
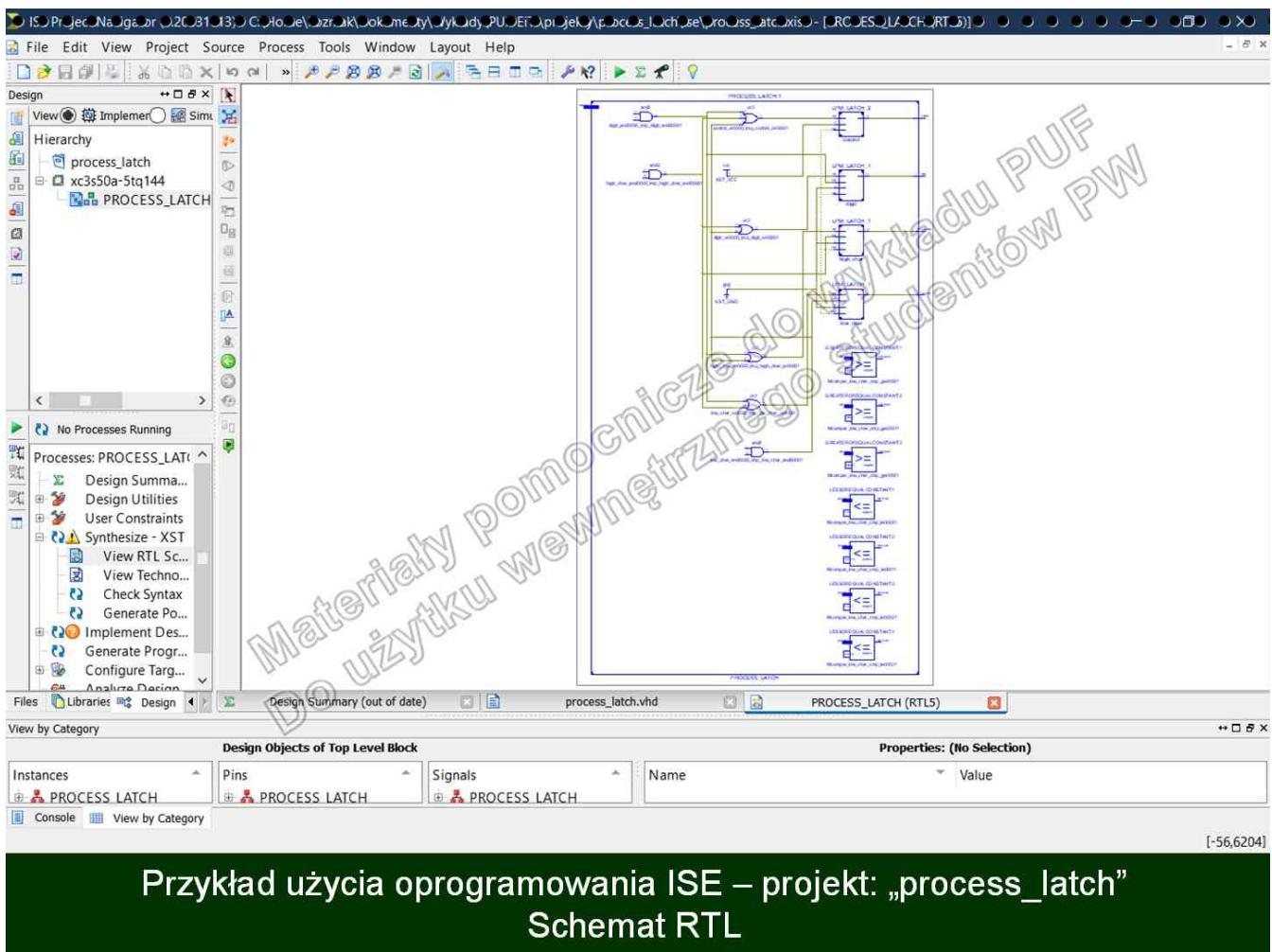
```
[etykieta :] if warunek1 then instrukcja { instrukcja }
{ elsif warunek then instrukcja { instrukcja } }
[ else instrukcja_defaultowa { instrukcja_defaultowa } ]
end if [etykieta];
```

- kilku-wariantowy wybór sekwencji instrukcji przypisania:

```
if sel1=1 then
  S1 <= argA + 3 ; S2 <= argB ;
elsif sel2=2 then
  S1 <= argB + 5 ; S3 <= argB ;
end if ;
```

```
1 entity PROCESS_LATCH is
2   port ( ascii      : in character;
3         low_char   : out bit;
4         high_char : out bit;
5         digit     : out bit;
6         control   : out bit
7       );
8 end PROCESS_LATCH;
9
10 architecture cials of PROCESS_LATCH is
11 begin
12
13   process (ascii) is
14   begin
15     if ascii>='a' and ascii<='z' then low_char <= '1'; high_char <= '0'; digit <= '0'; control <= '0'; -- wybór m:
16     elsif ascii>'A' and ascii<='Z' then low_char = '0'; high_char <= '1'; digit <= '0'; control <= '0'; -- wybór du:
17     elsif ascii>='0' and ascii<='9' then low_char = '0'; high_char <= '0'; digit <= '1'; control <= '0'; -- wybór l:
18     elsif ascii>=NUL and ascii<=USP then low_char <= '0'; high_char <= '0'; digit <= '0'; control <= '1'; -- wybór z:
19   end if;
20   end process;
21
22
23 end architecture cials;
24
25
26
27
28
29
30
31
32
33
34
35
36
```

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



```

1 entity PROCESS_LATCH_TB is -- pusty szkielet projektu symulacji
2 end PROCESS_LATCH_TB;
3
4 architecture behavioural of PROCESS_LATCH_TB is -- cialo architektoniczne projektu
5
6 signal ascii :character; -- symulowane wejście 'ascii'
7 signal low_char :bit; -- obserwowane wyjście 'low_char'
8 signal high_char :bit; -- obserwowane wyjście 'high_char'
9 signal digit :bit; -- obserwowane wyjście 'digit'
10 signal control :bit; -- obserwowane wyjście 'control'
11
12 signal low_char_err :bit; -- obserwowane wyjście bledu 'low_char'
13 signal high_char_err :bit; -- obserwowane wyjście bledu 'high_char'
14 signal digit_err :bit; -- obserwowane wyjście bledu 'digit'
15 signal control_err :bit; -- obserwowane wyjście bledu 'control'
16
17 begin -- czesc wykonawcza ciala projektu
18
19 process is -- proces bezwarunkowy
20 begin -- czesc wykonawcza procesu
21 ascii <= 'W'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='W' i oczekanie 10 ns
22 ascii <= 'Y'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='Y' i oczekanie 10 ns
23 ascii <= 'K'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='K' i oczekanie 10 ns
24 ascii <= 'L'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='L' i oczekanie 10 ns
25 ascii <= 'A'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='A' i oczekanie 10 ns
26 ascii <= 'D'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='D' i oczekanie 10 ns
27 ascii <= ' '; wait for 10 ns; -- ustawienie sygnalu 'ascii'=' ' i oczekanie 10 ns
28 ascii <= '5'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='5' i oczekanie 10 ns
29 ascii <= LF; wait for 10 ns; -- ustawienie sygnalu 'ascii'='LF' i oczekanie 10 ns
30 ascii <= CR; wait for 10 ns; -- ustawienie sygnalu 'ascii'='CR' i oczekanie 10 ns
31 end process;
32

```

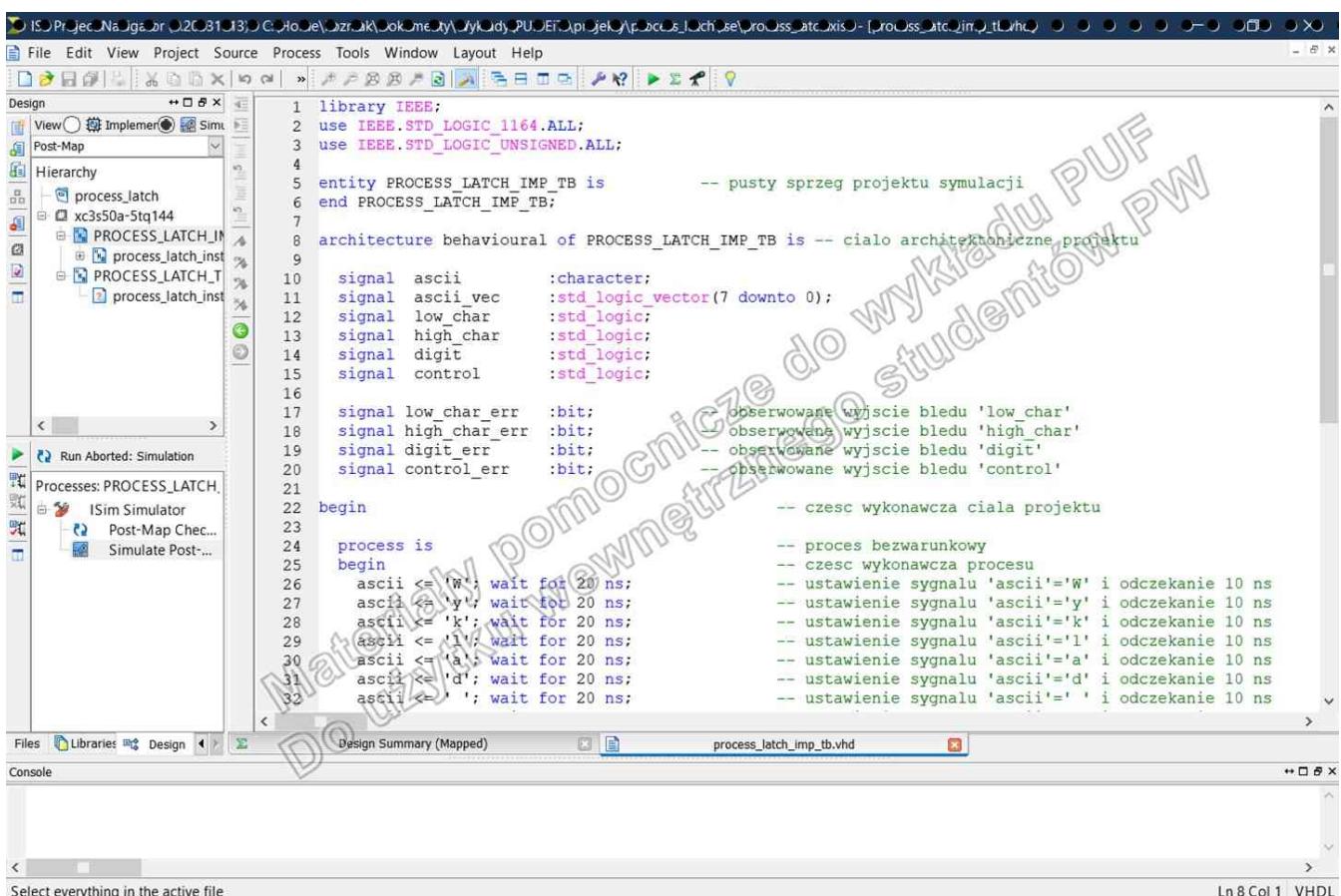
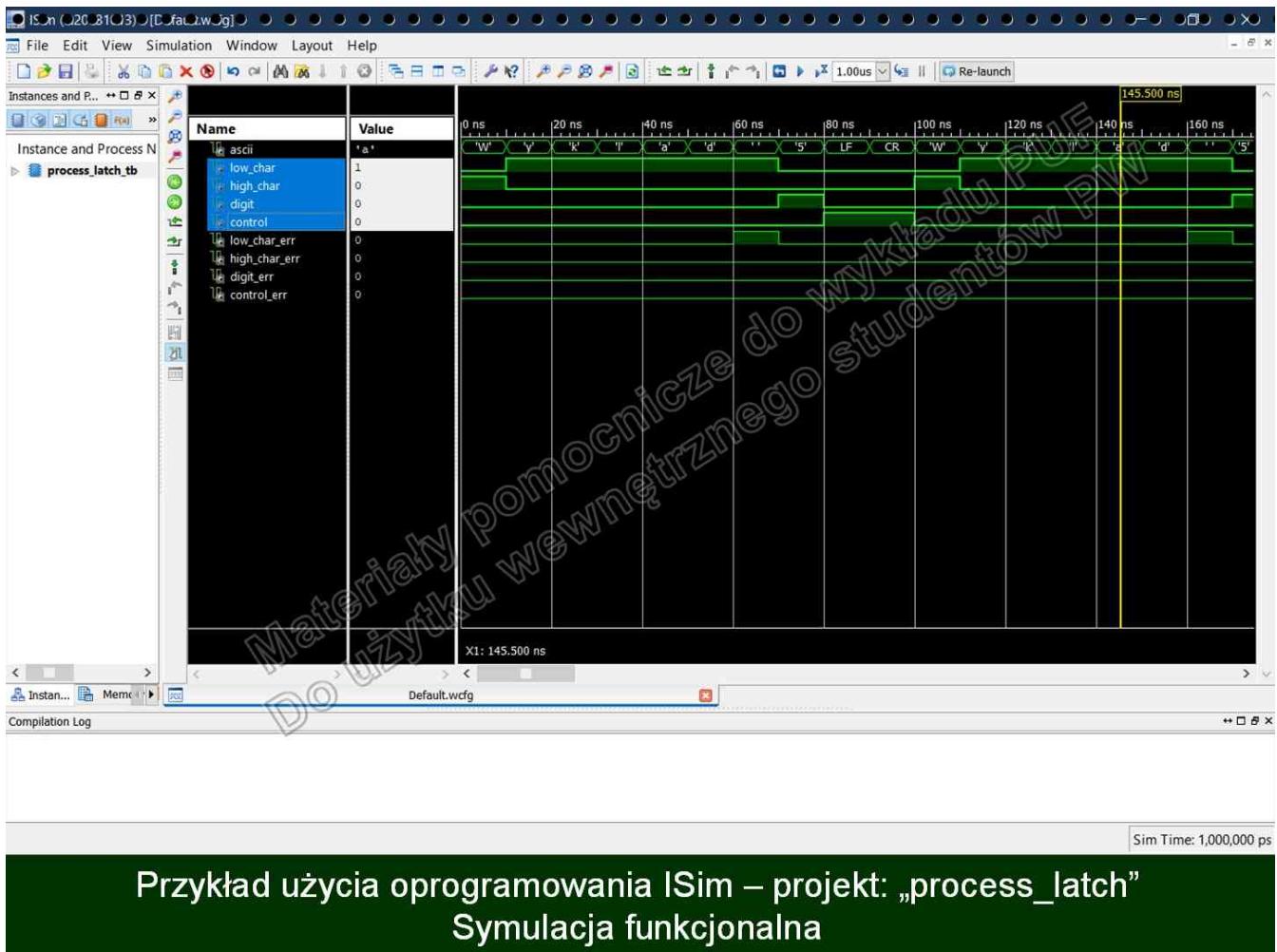
Przykład użycia oprogramowania ISE – projekt: „process_latch” Plik źródłowy „process_latch_TB.vhd” dla symulacji funkcjonalnej (fragment 1)

```

17 begin -- czesc wykonawcza ciala projektu
18
19 process is -- proces bezwarunkowy
20 begin -- czesc wykonawcza procesu
21 ascii <= 'W'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='W' i oczekanie 10 ns
22 ascii <= 'Y'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='Y' i oczekanie 10 ns
23 ascii <= 'K'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='K' i oczekanie 10 ns
24 ascii <= 'L'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='L' i oczekanie 10 ns
25 ascii <= 'A'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='A' i oczekanie 10 ns
26 ascii <= 'D'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='D' i oczekanie 10 ns
27 ascii <= ' '; wait for 10 ns; -- ustawienie sygnalu 'ascii'=' ' i oczekanie 10 ns
28 ascii <= '5'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='5' i oczekanie 10 ns
29 ascii <= LF; wait for 10 ns; -- ustawienie sygnalu 'ascii'='LF' i oczekanie 10 ns
30 ascii <= CR; wait for 10 ns; -- ustawienie sygnalu 'ascii'='CR' i oczekanie 10 ns
31 end process;
32
33 process_latch_inst: entity work.PROCESS_LATCH(cialo) -- instancja projektu 'PROCESS_LATCH'
34 port map (
35     ascii      => ascii,
36     low_char   => low_char,
37     high_char  => high_char,
38     digit      => digit,
39     control    => control
40 );
41
42 low_char_err <= '1' when (ascii>='A' and ascii<='Z') /= (low_char='1') else '0'; -- test malej litery
43 high_char_err <= '1' when (ascii>='A' and ascii<='Z') /= (high_char='1') else '0'; -- test duzej litery
44 digit_err   <= '1' when (ascii>='0' and ascii<='9') /= (digit='1') else '0'; -- test cyfry
45 control_err <= '1' when (ascii>=NUL and ascii<=USP) /= (control='1') else '0'; -- test znaku kontrolnego
46
47 end behavioural; -- zakonczenie ciala architektonicznego
48

```

Przykład użycia oprogramowania ISE – projekt: „process_latch” Plik źródłowy „process_latch_TB.vhd” dla symulacji funkcjonalnej (fragment 2)



Przykład użycia oprogramowania ISE – projekt: „process_latch”
Plik źródłowy „process_latch_TB.vhd” dla symulacji implementacji (fragment 1)

IS Project Navigator (20.3) | Home | Device Selection | Projects | process_latch_tb.vhd - [process_latch_im...]

File Edit View Project Source Process Tools Window Layout Help

Design View Implementer Sim Post-Map Hierarchy

```

29      ascii <= '1'; wait for 20 ns;      -- ustawienie sygnalu 'ascii'='1' i oczekanie 10 ns
30      ascii <= 'a'; wait for 20 ns;      -- ustawienie sygnalu 'ascii'='a' i oczekanie 10 ns
31      ascii <= 'd'; wait for 20 ns;      -- ustawienie sygnalu 'ascii'='d' i oczekanie 10 ns
32      ascii <= ' ' ; wait for 20 ns;     -- ustawienie sygnalu 'ascii'=' ' i oczekanie 10 ns
33      ascii <= '5'; wait for 20 ns;      -- ustawienie sygnalu 'ascii'='5' i oczekanie 10 ns
34      ascii <= LF; wait for 20 ns;      -- ustawienie sygnalu 'ascii'='LF' i oczekanie 10 ns
35      ascii <= CR; wait for 20 ns;      -- ustawienie sygnalu 'ascii'='CR' i oczekanie 10 ns
36  end process;
37
38  process (ascii) is
39    variable pos :integer;
40  begin
41    pos := character'pos(ascii);
42    ascii_vec <= (ascii_vec'range => '0') + pos;
43  end process;
44
45  processLatch_inst: entity work.PROCESS_LATCH(Structure) -- instancja projektu 'PROCESS_LATCH'
46    port map (
47      ascii      => ascii_vec,          -- mabowanie portow
48      low_char   => low_char,         -- przypisanie sygnalu 'ascii' do portu 'ascii'
49      high_char  => high_char,        -- przypisanie sygnalu 'low_char' do portu 'low_char'
50      digit      => digit,           -- przypisanie sygnalu 'high_char' do portu 'high_char'
51      control    => control,          -- przypisanie sygnalu 'digit' do portu 'digit'
52      );                           -- przypisanie sygnalu 'control' do portu 'control'
53
54      low_char_err <= '1' when (ascii='a' and ascii<='z') /= (low_char='1') else '0'; -- test malej lite
55      high_char_err <= '1' when (ascii='A' and ascii<='Z') /= (high_char='1') else '0'; -- test duzej lite
56      digit_err   <= '1' when (ascii='0' and ascii<='9') /= (digit='1') else '0'; -- test cyfry
57      control_err <= '1' when (ascii=NUL and ascii<=USP) /= (control='1') else '0'; -- test znaku kont
58
59  end behavioral;                  -- zakoñczenie ciala architektonicznego
60

```

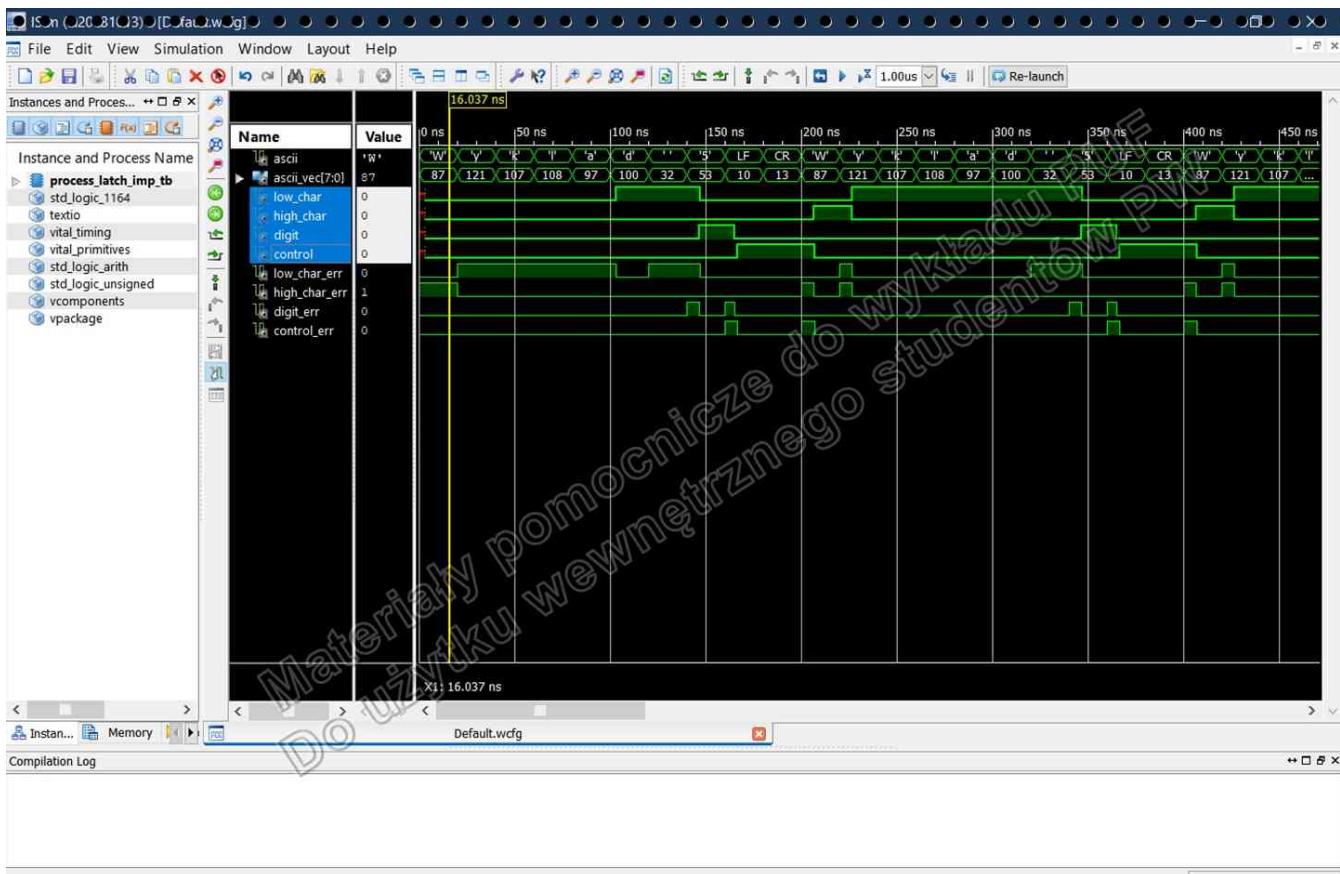
Run Aborted: Simulation
Processes: PROCESS_LATCH.
ISim Simulator
Post-Map Check...
Simulate Post-...

Files Libraries Design Design Summary (Mapped) process_latch_imp_tb.vhd

Console

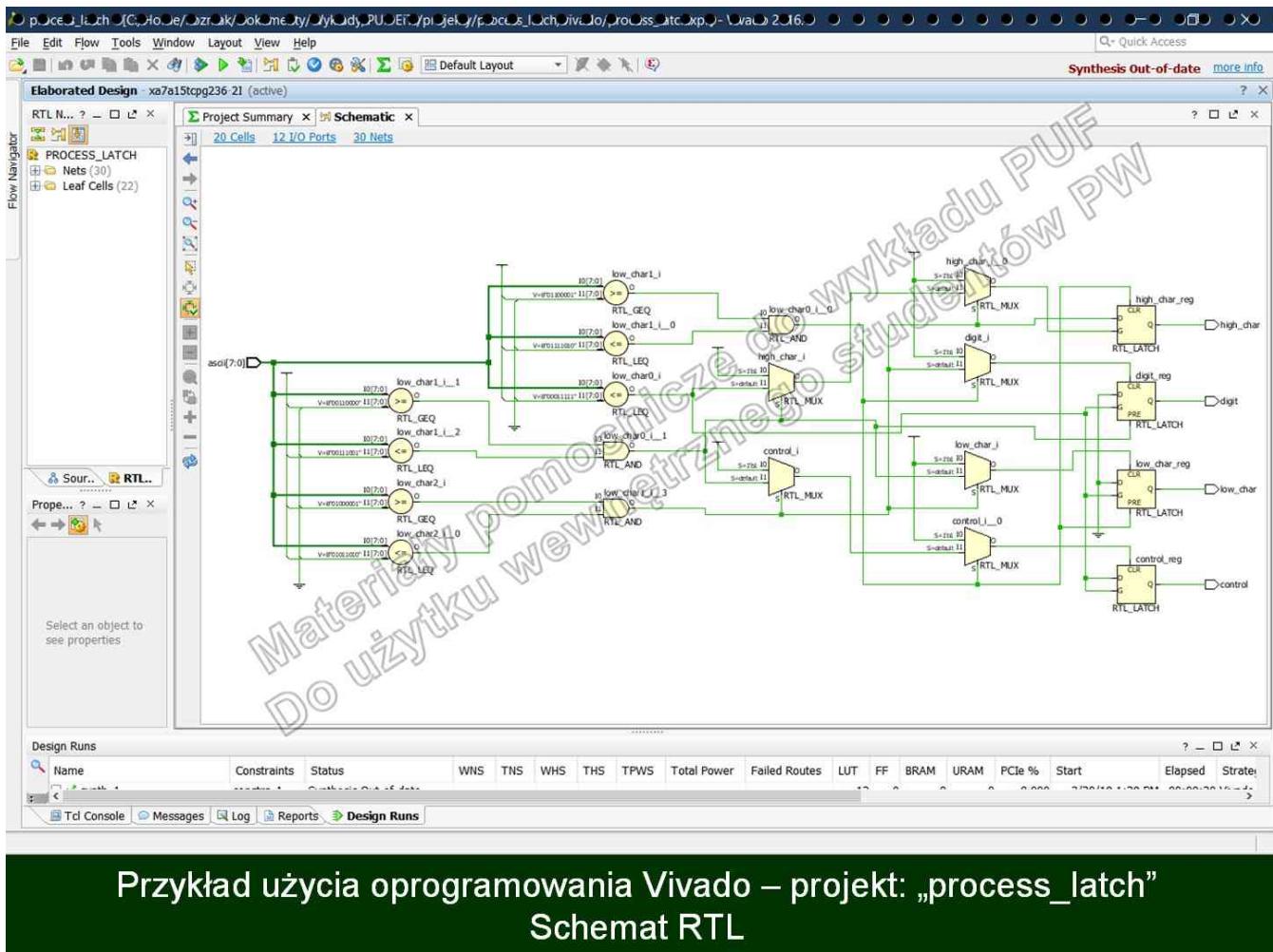
Przykład użycia oprogramowania ISE – projekt: „process_latch”

Plik źródłowy „process_latch_TB.vhd” dla symulacji implementacji (fragment 2)

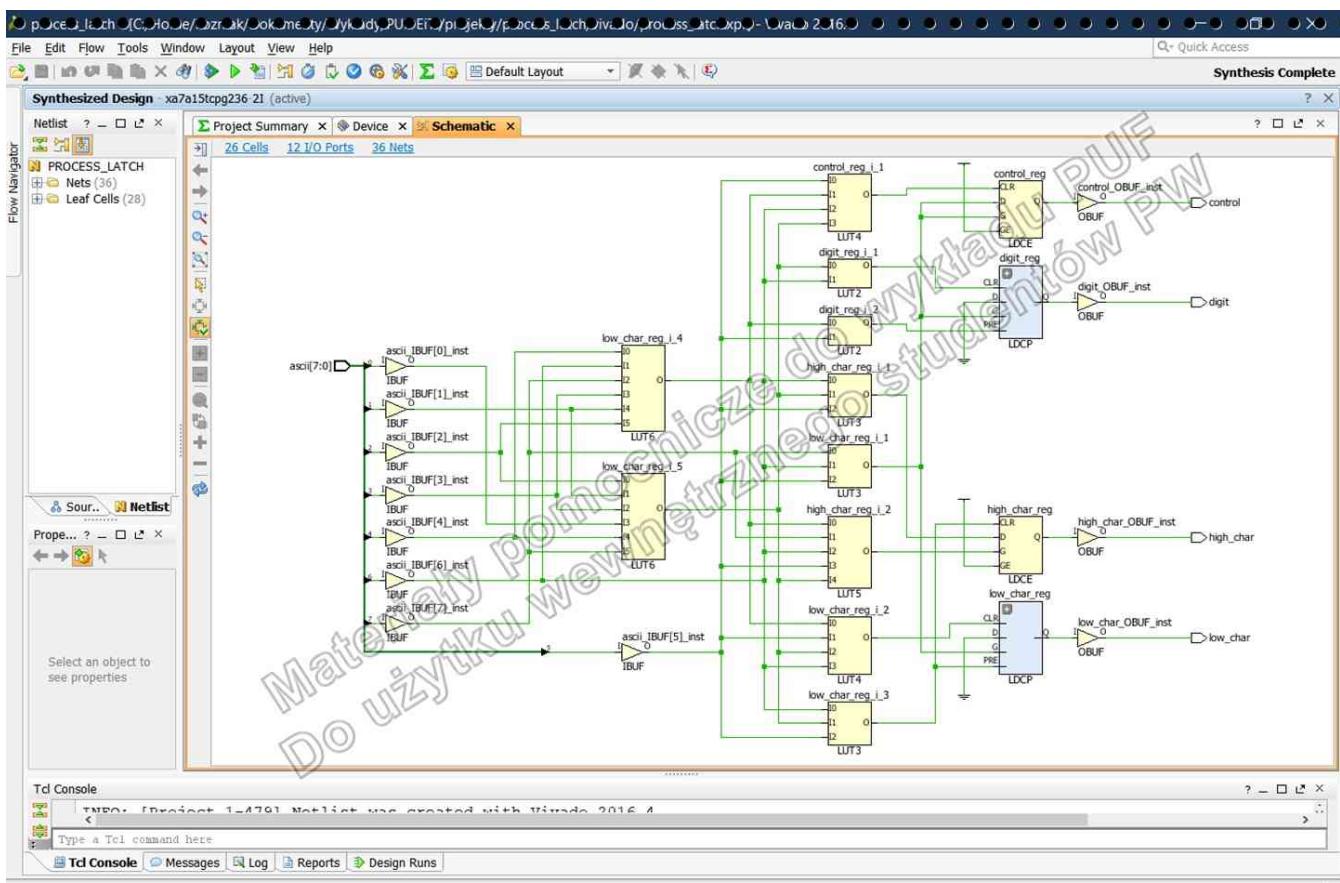


Przykład użycia oprogramowania ISim – projekt: „process_latch”

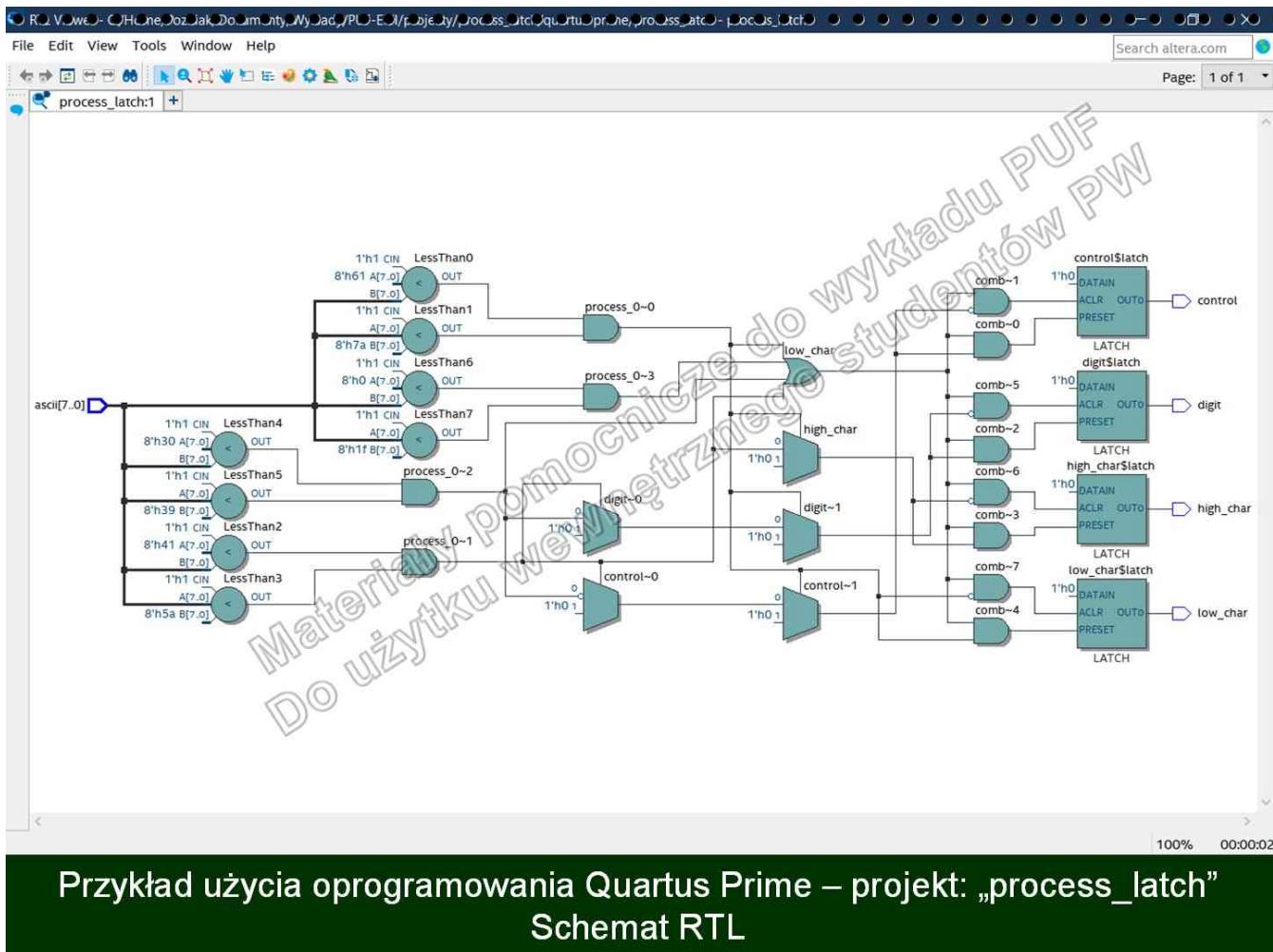
Symulacja czasowa po implementacji



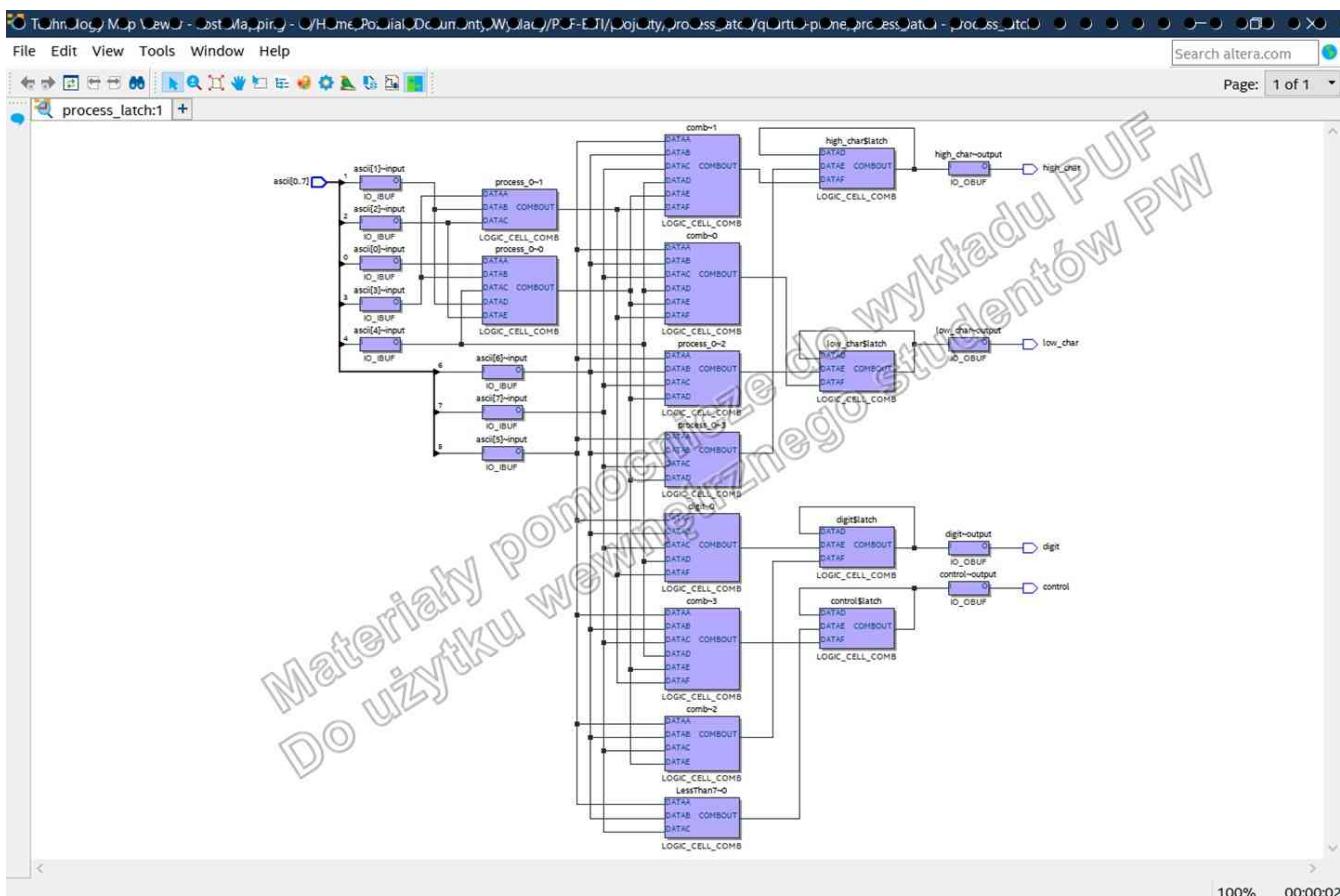
Przykład użycia oprogramowania Vivado – projekt: „process_latch”
Schemat RTL



Przykład użycia oprogramowania Vivado – projekt: „process_latch”
Schemat technologiczny



Przykład użycia oprogramowania Quartus Prime – projekt: „process_latch”
Schemat RTL



Przykład użycia oprogramowania Quartus Prime – projekt: „process_latch”
Schemat technologiczny

Podstawowe elementy standardu VHDL

Wybrane instrukcje w procesie

Podstawowa składnia selektywnego wyboru instrukcji:

```
[etykieta :] case wyrażenie_wyboru is
{ when selekcja { | selekcja} => instrukcja { instrukcja } }
[ when default => instrukcja {instrukcja} ]
end case [etykieta] ;
```

W **wyrażeniu_wyboru** i w **selekcji** nie mogą wystąpić **zmienne**
które ulegną modyfikacji w czasie wykonywania **instrukcji**

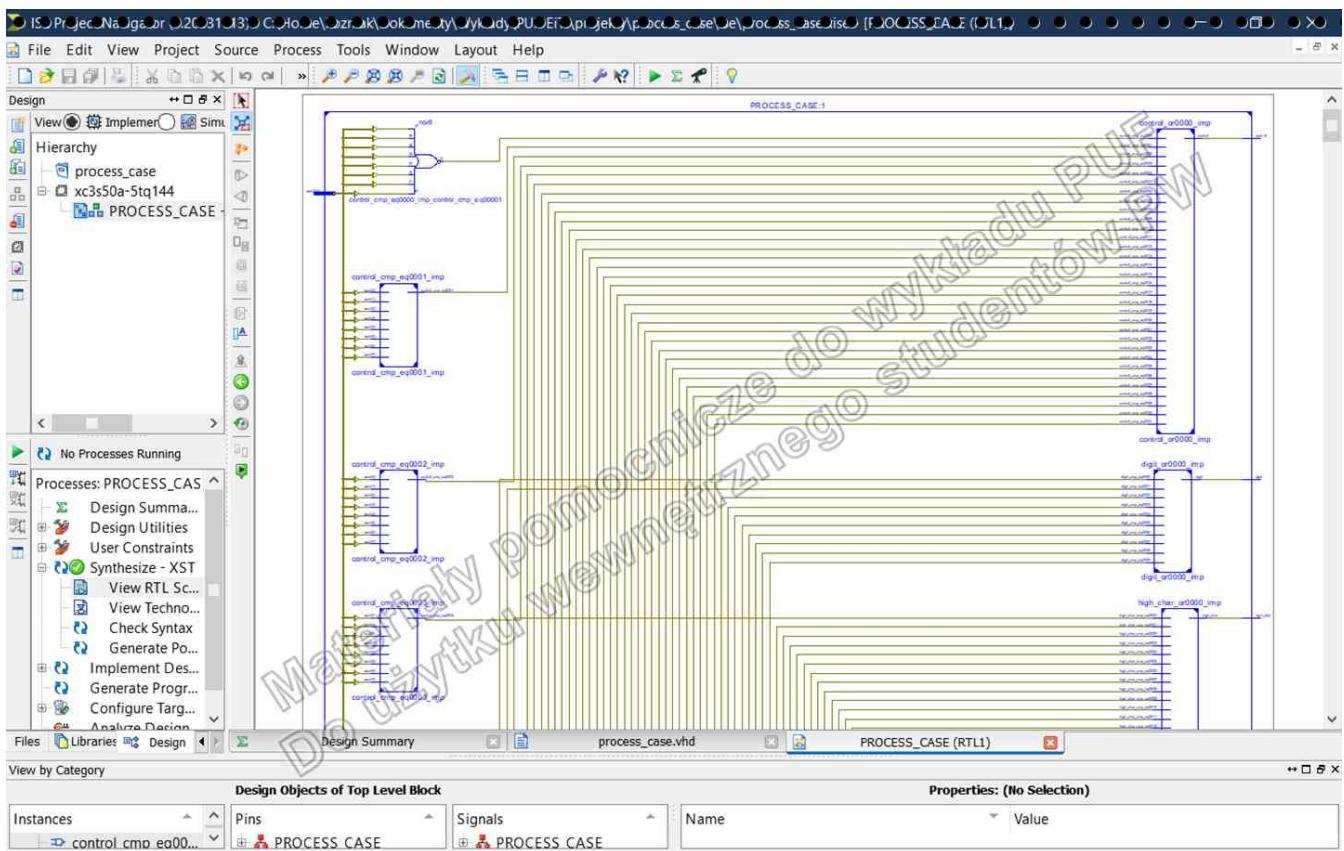
- podstawowe rodzaje selekcji wyboru:

- wyrażenie proste (wartość wyrażenia = wyrażenie_wyboru)
przykłady: 1 lub x lub x+1 lub '0' lub "001"
- zakres dyskretny (element zakresu = wartość wyrażenia_wyboru)
przykłady: 1 to 3 lub x downto 0

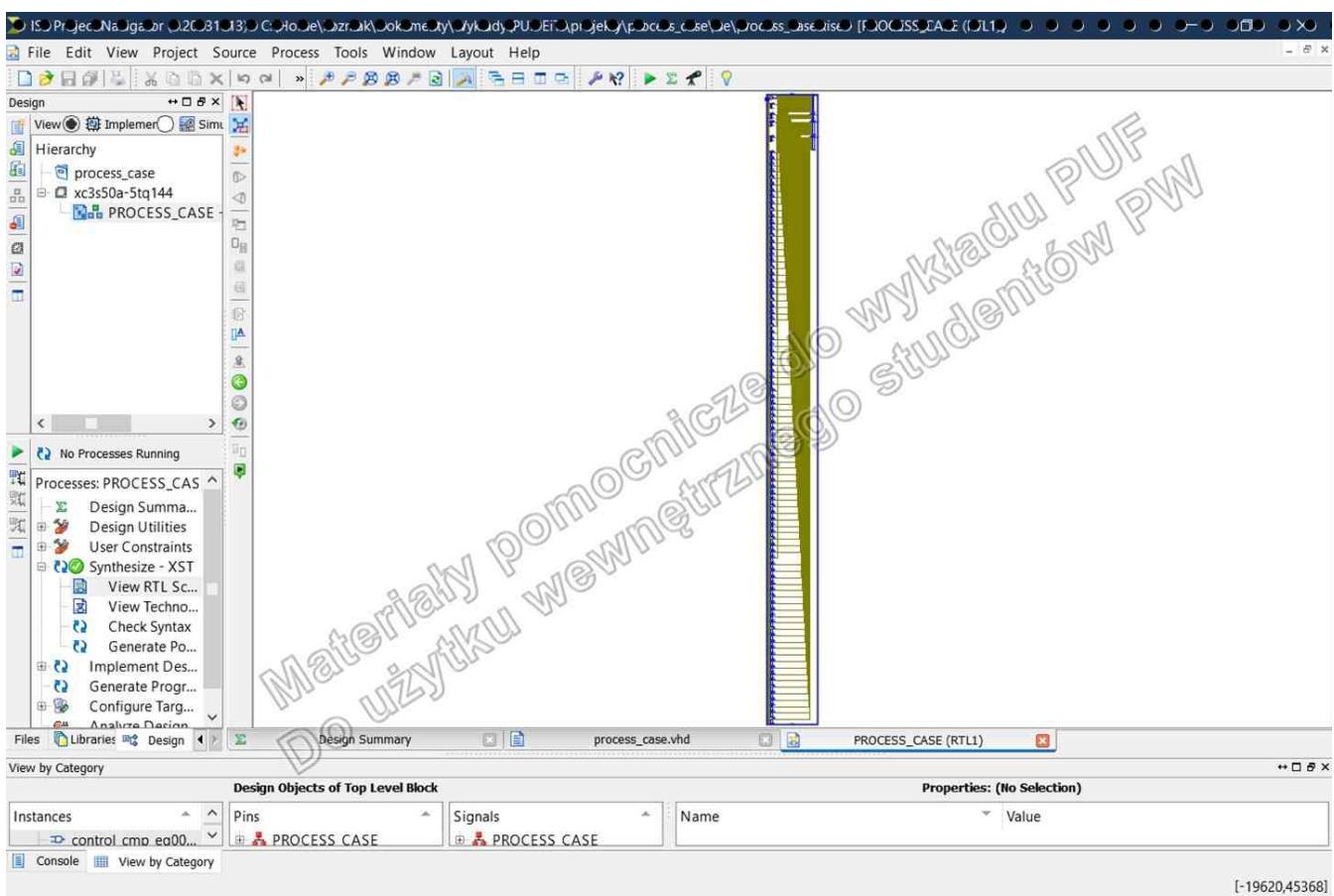


```
1 entity PROCESS_CASE is
2     port ( ascii : in character;
3             low_char : out bit;
4             high_char : out bit;
5             digit : out bit;
6             control : out bit
7         );
8 end PROCESS_CASE;
9
10 architecture cialo of PROCESS_CASE is
11
12 begin
13
14     process (ascii) is
15     begin
16         low_char <= '0';
17         high_char <= '0';
18         digit <= '0';
19         control <= '0';
20         case ascii is
21             when 'a' to 'z' => low_char <= '1';
22             when 'A' to 'Z' => high_char <= '1';
23             when '0' to '9' => digit <= '1';
24             when NUL to USP => control <= '1';
25             when others => null;
26         end case;
27     end process;
28
29 end architecture cialo;
```

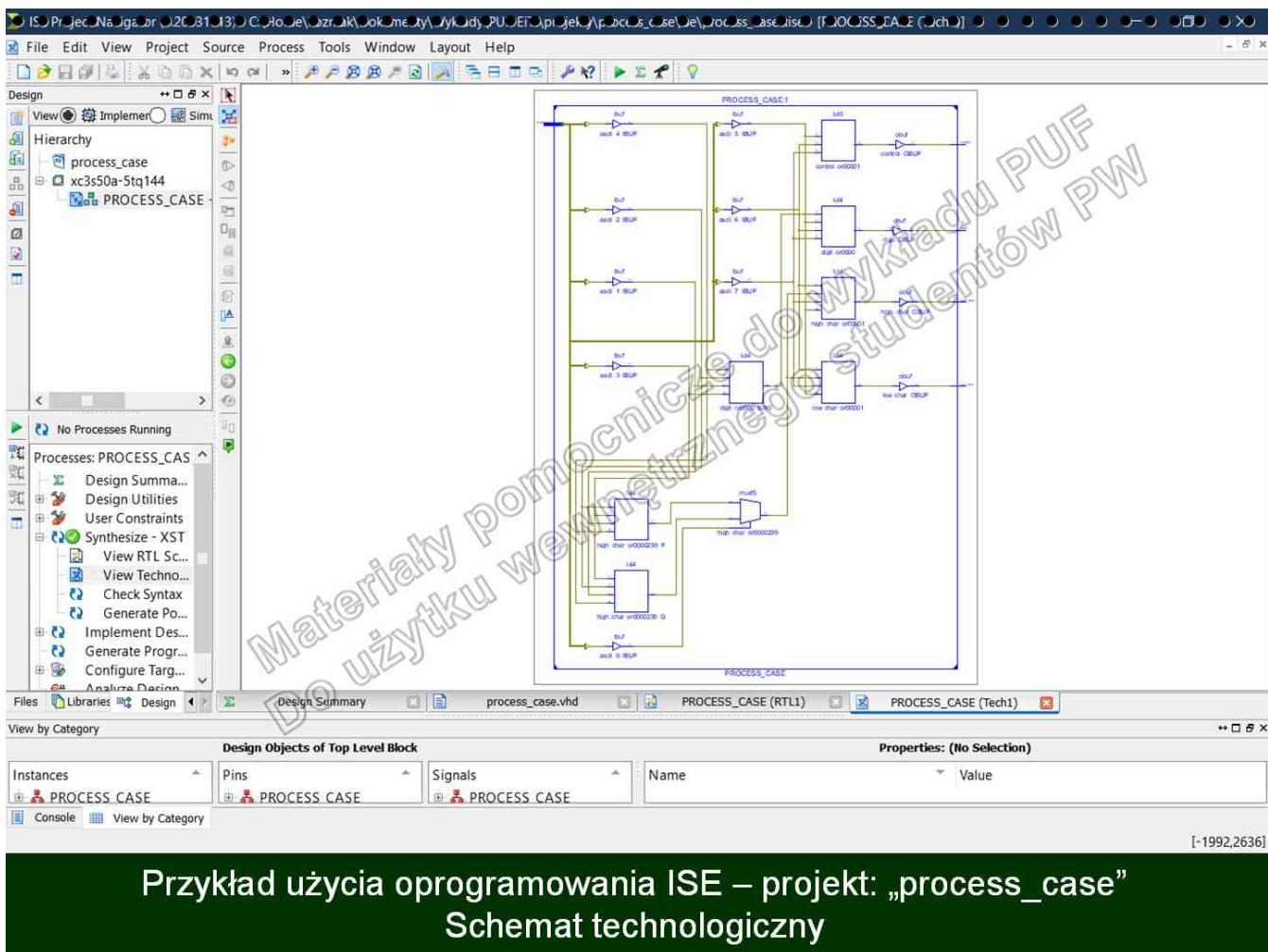
Przykład użycia oprogramowania ISE – projekt: „process_case”
Plik źródłowy „process_case.vhd”



Przykład użycia oprogramowania ISE – projekt: „process_case”
Schemat RTL (fragment)



Przykład użycia oprogramowania ISE – projekt: „process_case”
Schemat RTL (cały)



ASCII/ISO & IEEE CODE CHART										
B7	0	0	0	0	1	0	1	0	1	
B6	0	0	0	1	0	0	1	0	1	
B5										
BITS				CONTROL			NUMBERS SYMBOLS		UPPER CASE	
B4 B3 B2 B1										
0	0	0	0	NUL	20	DLE	40	SP	60	
0	0	0	1	0	10	16	20	32	30	
0	0	0	1	GTL	21	LLO	41	!	1	
1	1	1	1	SOH	DC1		31	49	49	
0	0	1	0	STX	DC2	"	42	62	2	
2	2	2	2	DC3		2	42	66	B	
3	3	3	3	ETX	DC4	#	43	63	C	
0	0	1	1	SDC	DCL	3	35	51	43	
4	4	4	4	EOT	DC4	\$	64	64	D	
5	PPC	25	PPU	ENQ	NAK	%	45	65	4	
0	1	0	1	FNO		5	105	F	105	
								II	e	
									u	

Tablica kodowania znaków ASCII (fragment)

Przykład użycia oprogramowania ISE – projekt: „process_case”

Plik źródłowy „process_case_TB.vhd” (fragment 1)

```

1 entity PROCESS_CASE_TB is -- pusty szkielet projektu symulacji
2 end PROCESS_CASE_TB;
3
4 architecture behavioural of PROCESS_CASE_TB is -- cialo architektoniczne projektu
5
6 signal ascii :character; -- symulowane wejście 'ascii'
7 signal low_char :bit; -- obserwowane wyjście 'low_char'
8 signal high_char :bit; -- obserwowane wyjście 'high_char'
9 signal digit :bit; -- obserwowane wyjście 'digit'
10 signal control :bit; -- obserwowane wyjście 'control'
11
12 signal low_char_err :bit; -- obserwowane wyjście bledu 'low_char'
13 signal high_char_err :bit; -- obserwowane wyjście bledu 'high_char'
14 signal digit_err :bit; -- obserwowane wyjście bledu 'digit'
15 signal control_err :bit; -- obserwowane wyjście bledu 'control'
16 begin -- czesc wykonawcza ciala projektu
17
18 process is -- proces bezwarunkowy
19 begin -- czesc wykonawcza procesu
20   ascii <= 'W'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='W' i oczekanie 10 ns
21   ascii <= 'y'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='y' i oczekanie 10 ns
22   ascii <= 'k'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='k' i oczekanie 10 ns
23   ascii <= 'l'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='l' i oczekanie 10 ns
24   ascii <= 'a'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='a' i oczekanie 10 ns
25   ascii <= 'd'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='d' i oczekanie 10 ns
26   ascii <= ' '; wait for 10 ns; -- ustawienie sygnalu 'ascii'=' ' i oczekanie 10 ns
27   ascii <= '5'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='5' i oczekanie 10 ns
28   ascii <= LF; wait for 10 ns; -- ustawienie sygnalu 'ascii'='LF' i oczekanie 10 ns
29   ascii <= CR; wait for 10 ns; -- ustawienie sygnalu 'ascii'='CR' i oczekanie 10 ns
30 end process;
31
32 process case_inst: entity work.PROCESS_CASE(cialo) -- instancja projektu 'PROCESS_CASE'

```

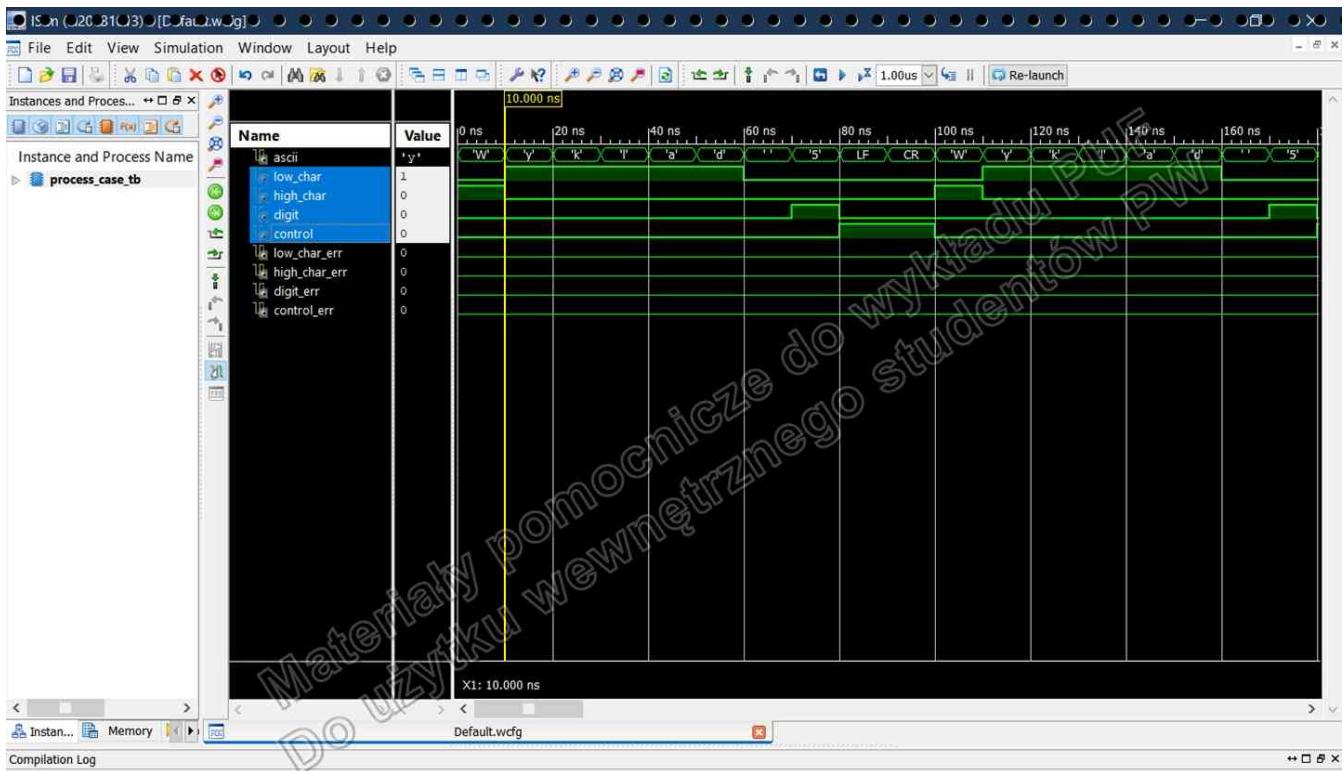
Przykład użycia oprogramowania ISE – projekt: „process_case”

Plik źródłowy „process_case_TB.vhd” (fragment 2)

```

16 begin -- czesc wykonawcza ciala projektu
17
18 process is -- proces bezwarunkowy
19 begin -- czesc wykonawcza procesu
20   ascii <= 'W'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='W' i oczekanie 10 ns
21   ascii <= 'y'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='y' i oczekanie 10 ns
22   ascii <= 'k'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='k' i oczekanie 10 ns
23   ascii <= 'l'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='l' i oczekanie 10 ns
24   ascii <= 'a'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='a' i oczekanie 10 ns
25   ascii <= 'd'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='d' i oczekanie 10 ns
26   ascii <= ' '; wait for 10 ns; -- ustawienie sygnalu 'ascii'=' ' i oczekanie 10 ns
27   ascii <= '5'; wait for 10 ns; -- ustawienie sygnalu 'ascii'='5' i oczekanie 10 ns
28   ascii <= LF; wait for 10 ns; -- ustawienie sygnalu 'ascii'='LF' i oczekanie 10 ns
29   ascii <= CR; wait for 10 ns; -- ustawienie sygnalu 'ascii'='CR' i oczekanie 10 ns
30 end process;
31
32 process case_inst: entity work.PROCESS_CASE(cialo) -- instancja projektu 'PROCESS_CASE'
33 begin -- mabowanie portow
34   port map (
35     ascii      => ascii,
36     low_char   => low_char,
37     high_char  => high_char,
38     digit      => digit,
39     control    => control
40   );
41   low_char_err <= '1' when (ascii>='a' and ascii<='z') /= (low_char='1') else '0'; -- test malej litery
42   high_char_err <= '1' when (ascii>='A' and ascii<='Z') /= (high_char='1') else '0'; -- test duzej litery
43   digit_err <= '1' when (ascii>='0' and ascii<='9') /= (digit='1') else '0'; -- test cyfry
44   control_err <= '1' when (ascii>=NUL and ascii<=USP) /= (control='1') else '0'; -- test znaku kontrolnego
45
46 end behavioural; -- zakonczenie ciala architektonicznego
47

```



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

Podstawowe elementy standardu VHDL

Wybrane instrukcje w procesie

Podstawowa składnia pętli dyskretnej:

```
[etykieta :] for identyfikator in zakres_dyskretny loop
    instrukcja { instrukcja }
end loop [etykieta];
```

W zakresie_dyskretnym nie mogą wystąpić zmienne które ulegną zmianie w czasie wykonywania instrukcji

- podstawowe rodzaje zakresu dyskretnego:
 - wyrażenie proste (identyfikator = od lewej do prawej wartości) przykłady: 1 to 3 lub x downto 0
 - zakres typu (identyfikator = od lewej do prawej wartości typu) przykłady: character lub boolean lub wektor'range

Materiały pomocnicze do wykładowego PUF
Do użytku wewnętrznego PW

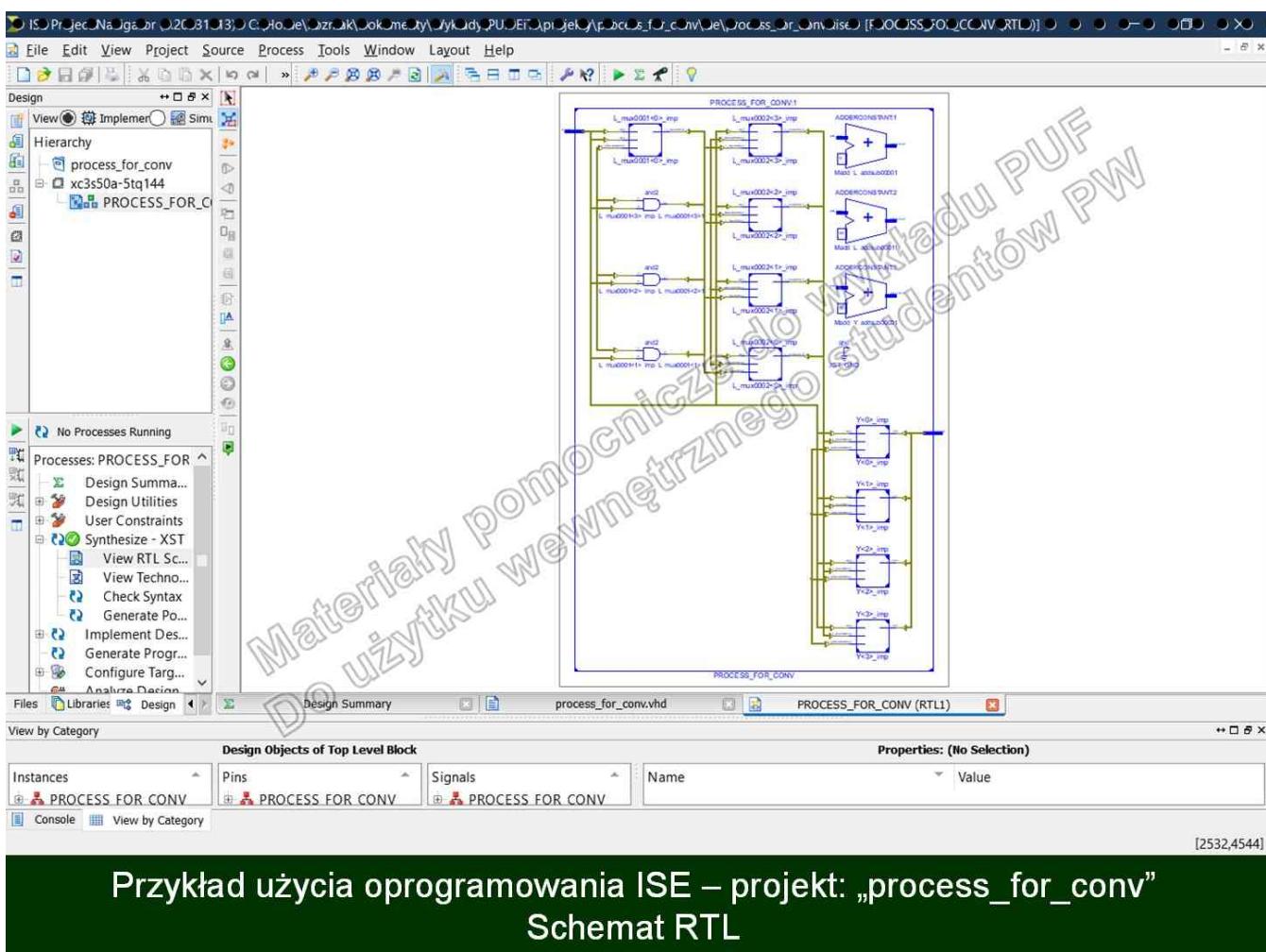
Przykład użycia oprogramowania ISE – projekt: „process_for_conv”
Plik źródłowy „process_for_conv.vhd”

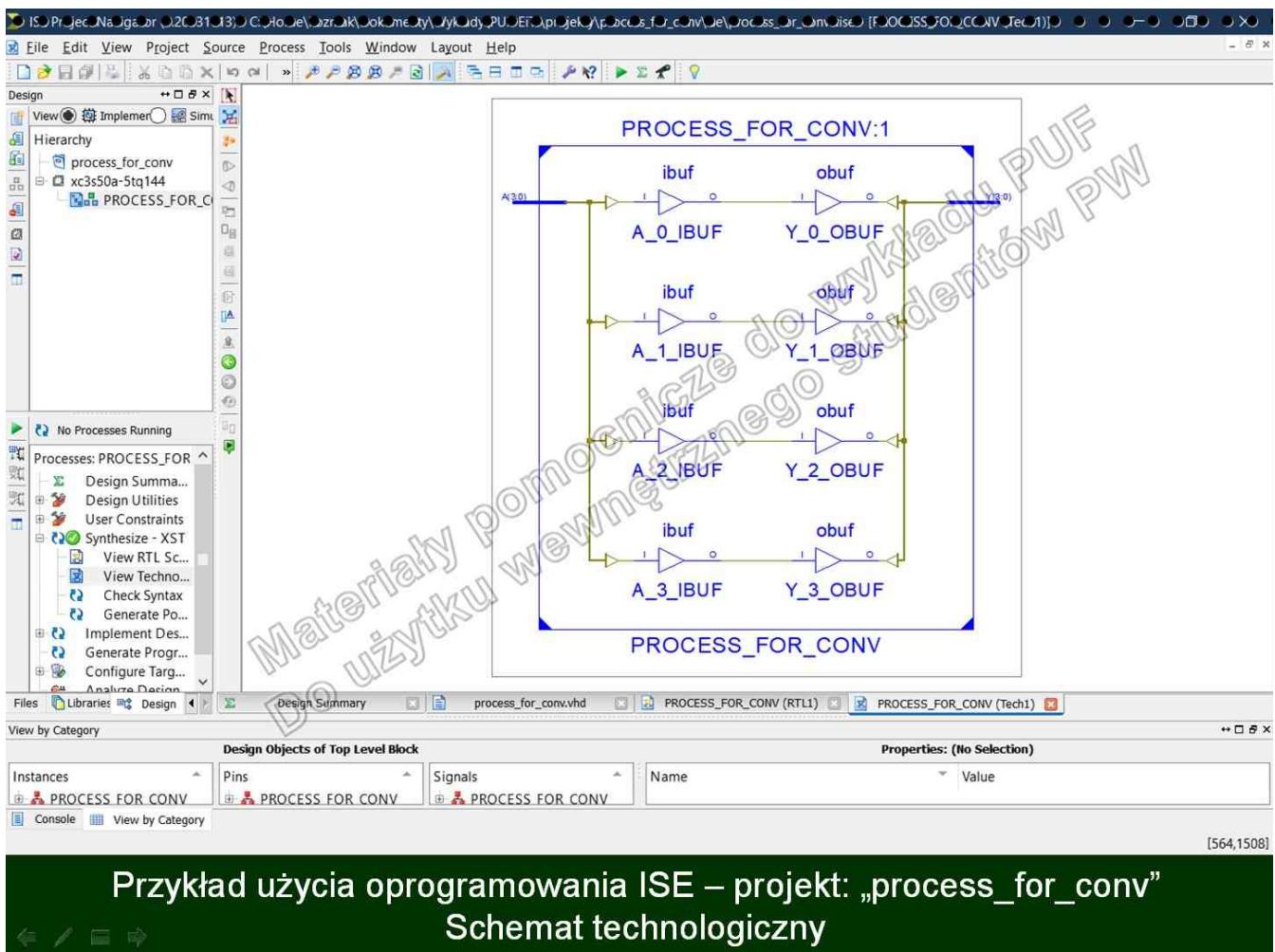
```

1  entity PROCESS_FOR_CONV is
2      port ( A : in bit_vector(3 downto 0);
3              Y : out natural range 0 to 15
4          );
5  end PROCESS_FOR_CONV;
6
7  architecture cialo of PROCESS_FOR_CONV is
8
9  begin
10
11     process (A) is
12         variable L : natural range 0 to 15;
13     begin
14         L := 0;
15         for P in 0 to 3 loop
16             if (A(P)='1') then
17                 L := L + 2**P;
18             end if;
19         end loop;
20         Y <= L;
21     end process;
22
23 end architecture cialo;
24

```

-- deklaracja sprzedu PROCESS_FOR_CONV
-- deklaracja portu wejściowego 'A'
-- deklaracja portu wyjściowego 'Y'
-- zakończenie deklaracji listy portów
-- zakończenie deklaracji nagłówka
-- deklaracja ciała 'cialo' architektury
-- początek części wykonawczej
-- lista czułości procesu
-- utworzenie zmiennej 'L'
-- czasie wykonawcza procesu
-- ustawienie wartości początkowej 'L' na 0
-- pętla od 0 do 3 dla identyfikatora 'P'
-- zadanie, czy bit o indeksie 'P' ma wartość '1'
-- zwiększenie o 2^P licznika 'L'
-- zakończenie warunku
-- zakończenie pętli
-- przypisanie stanu 'L' do sygnału 'Y'
-- zakończenie procesu
-- zakończenie deklaracji ciała 'cialo'





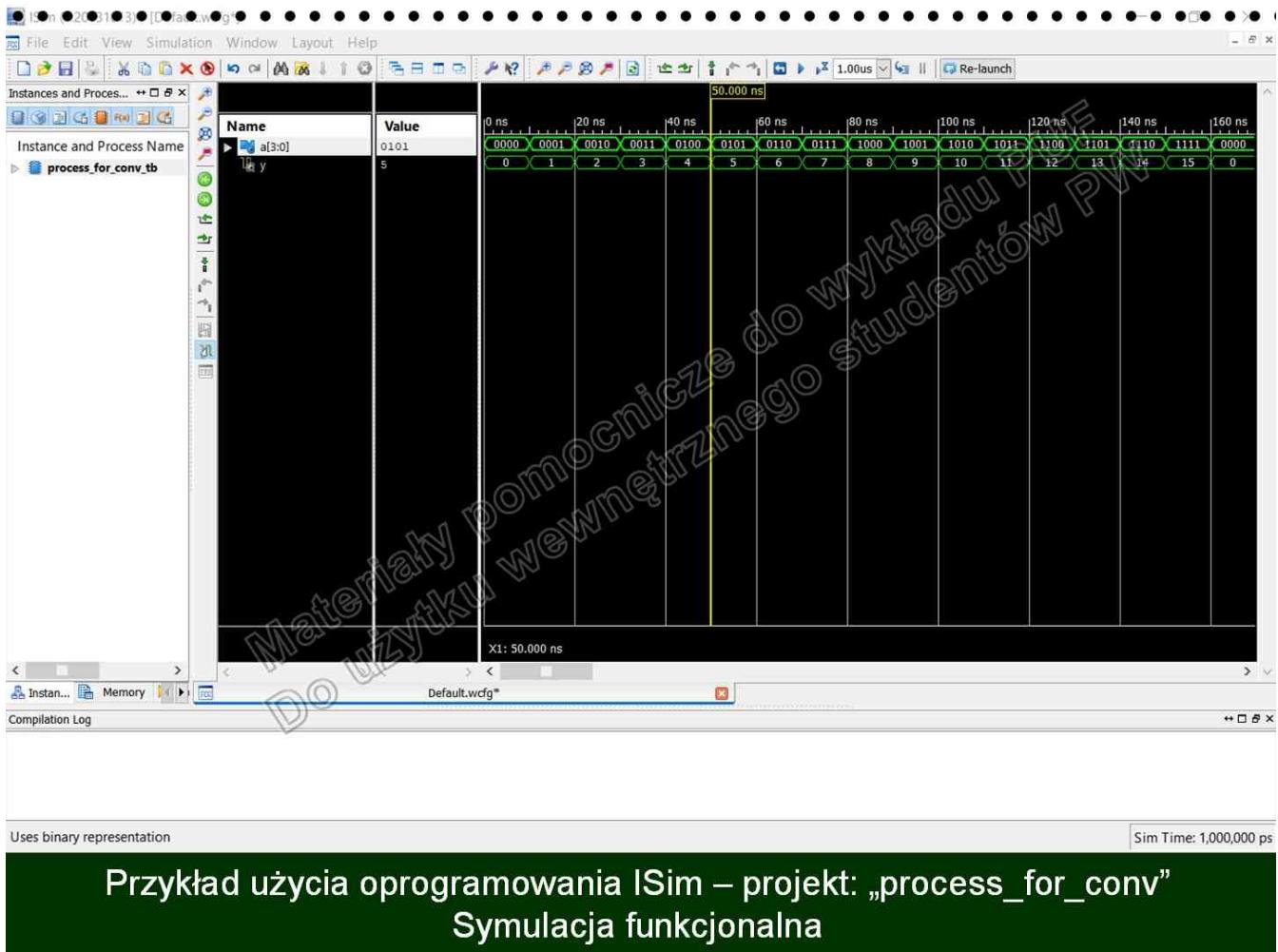
Przykład użycia oprogramowania ISE – projekt: „process_for_conv”
Schemat technologiczny

```

1 entity PROCESS_FOR_CONV_TB is -- pusty szkielet projektu symulacji
2 end PROCESS_FOR_CONV_TB;
3
4 architecture behavioural of PROCESS_FOR_CONV_TB is -- cialo architektoniczne projektu
5
6   signal A      :bit_vector(3 downto 0);    -- symulowane wejscie A
7   signal Y      :natural range 0 to 4;       -- obserwowane wyjście Y
8
9 begin
10
11   process is
12     begin
13     A <= "0000"; wait for 10 ns;           -- ustawienie sygnalu 'A'=0 i oczekanie 10 ns
14     A <= "0001"; wait for 10 ns;           -- ustawienie sygnalu 'A'=1 i oczekanie 10 ns
15     A <= "0010"; wait for 10 ns;           -- ustawienie sygnalu 'A'=2 i oczekanie 10 ns
16     A <= "0011"; wait for 10 ns;           -- ustawienie sygnalu 'A'=3 i oczekanie 10 ns
17     A <= "0100"; wait for 10 ns;           -- ustawienie sygnalu 'A'=4 i oczekanie 10 ns
18     A <= "0101"; wait for 10 ns;           -- ustawienie sygnalu 'A'=5 i oczekanie 10 ns
19     A <= "0110"; wait for 10 ns;           -- ustawienie sygnalu 'A'=6 i oczekanie 10 ns
20     A <= "0111"; wait for 10 ns;           -- ustawienie sygnalu 'A'=7 i oczekanie 10 ns
21     A <= "1000"; wait for 10 ns;           -- ustawienie sygnalu 'A'=8 i oczekanie 10 ns
22     A <= "1001"; wait for 10 ns;           -- ustawienie sygnalu 'A'=9 i oczekanie 10 ns
23     A <= "1010"; wait for 10 ns;           -- ustawienie sygnalu 'A'=10 i oczekanie 10 ns
24     A <= "1011"; wait for 10 ns;           -- ustawienie sygnalu 'A'=11 i oczekanie 10 ns
25     A <= "1100"; wait for 10 ns;           -- ustawienie sygnalu 'A'=12 i oczekanie 10 ns
26     A <= "1101"; wait for 10 ns;           -- ustawienie sygnalu 'A'=13 i oczekanie 10 ns
27     A <= "1110"; wait for 10 ns;           -- ustawienie sygnalu 'A'=14 i oczekanie 10 ns
28     A <= "1111"; wait for 10 ns;           -- ustawienie sygnalu 'A'=15 i oczekanie 10 ns
29   end process;                         -- zakoñczenie procesu
30
31
32   process for_conv_inst: entity work.PROCESS_FOR_CONV(cialo) -- instancja projektu 'PROCESS_FOR_CONV'
33   port map (A => A, Y => Y);          -- przypisanie portom sygnalow
34
35 end behavioural;                      -- zakoñczenie ciala architektonicznego
36

```

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



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

Podstawowe elementy standardu VHDL

Wybrane instrukcje w procesie

Instrukcja wyjścia:

[etykieta :] exit [etykieta_pętli :] [when warunek] ;

- podstawowe rodzaje użycia instrukcji:
 - użycie bezwarunkowe
przykłady: `exit;` lub `if a>0 then exit; end` (warunek zewnętrzny)
 - użycie warunkowe
przykłady: `exit when a>0;`

Podstawowe elementy standardu VHDL

Wybrane instrukcje w procesie

Instrukcja wyjścia:

[etykieta :] exit [etykieta_pętli :] [when warunek] ;

Instrukcja następnego kroku:

[etykieta :] next [etykieta_pętli :] [when warunek] ;

- podstawowe rodzaje użycia instrukcji:

- użycie bezwarunkowe

przykłady: next; lub if a>0 then next; end (warunek zewnętrzny)

- użycie warunkowe

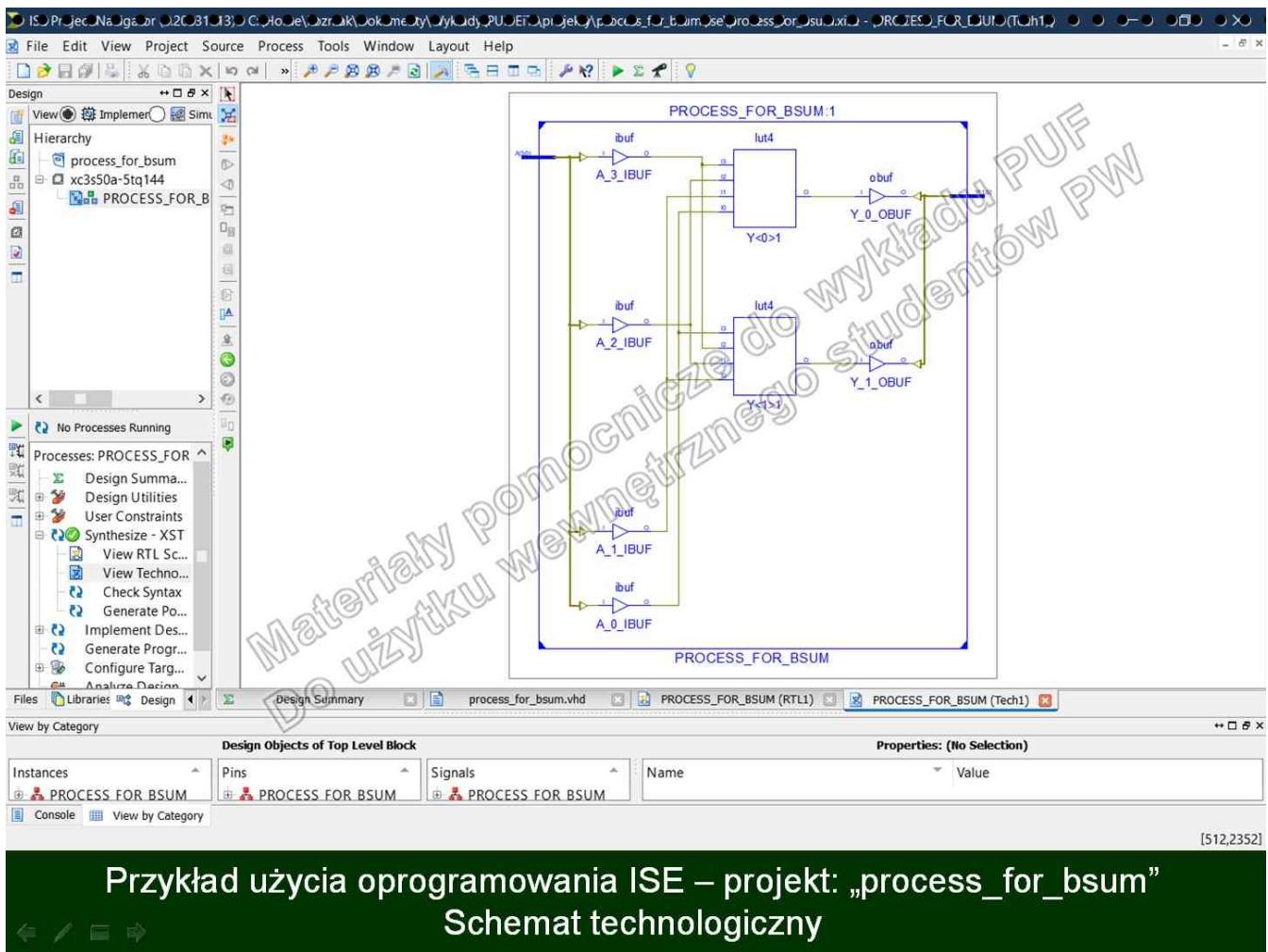
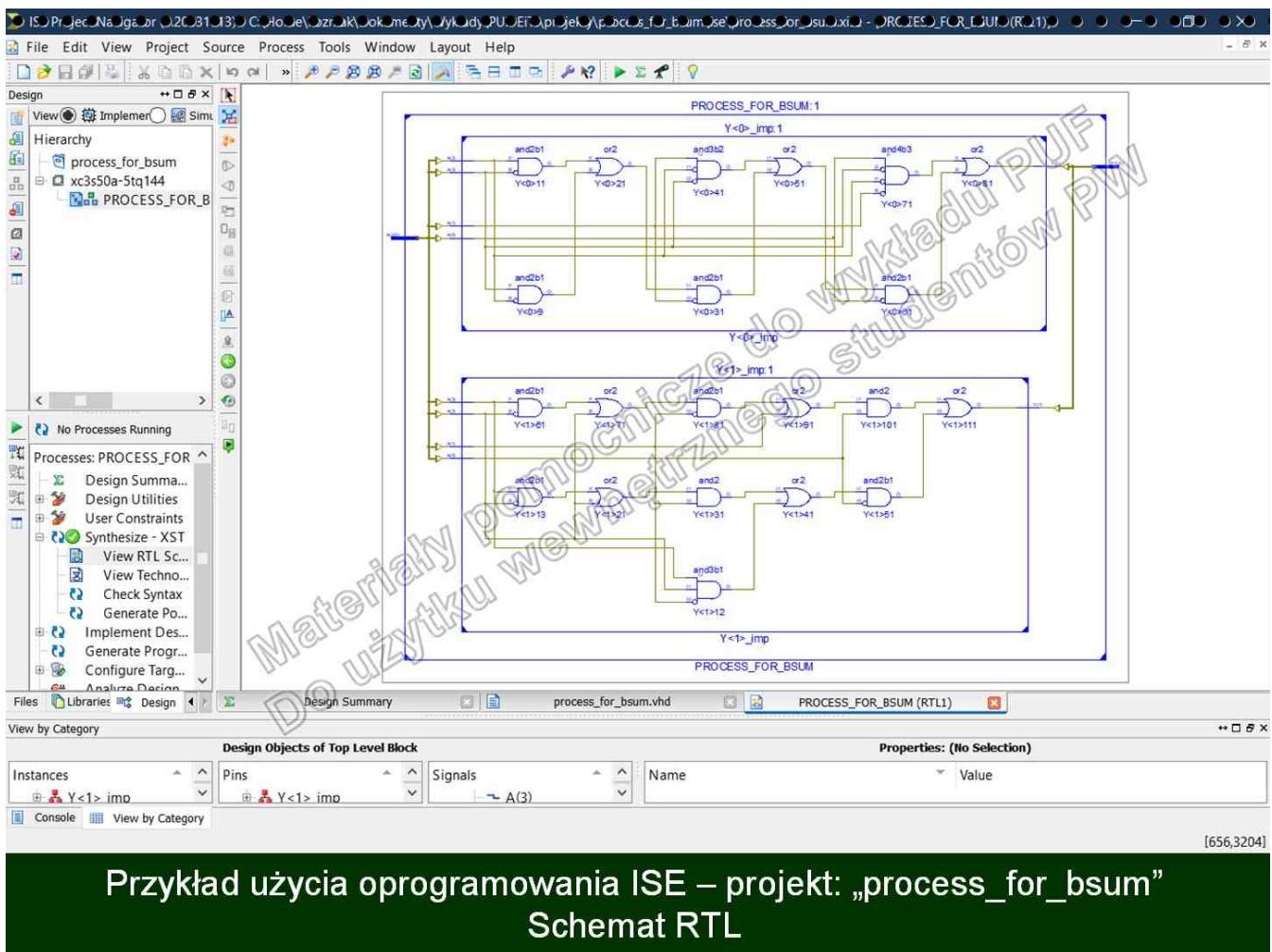
przykłady: next when a>0;



```
1 entity PROCESS_FOR_BSUM is
2     port ( A : in bit_vector(3 downto 0);
3             Y : out natural range 0 to 2
4         );
5 end PROCESS_FOR_BSUM;
6
7 architecture cialo of PROCESS_FOR_BSUM is
8
9 begin
10
11     process (A) is
12         variable L : natural range 0 to 2;
13     begin
14         L := 0;
15         for P in 0 to A'length-1 loop
16             next when A(P)='0';
17             L := L + 1;
18             exit when L=2;
19         end loop;
20         Y <= L;
21     end process;
22
23 end architecture cialo;
24
```

-- deklaracja sprawu PROCESZ FOR BSUM
-- deklaracja portu wejściowego 'A'
-- deklaracja portu wyjściowego 'Y'
-- zakończenie deklaracji listy portów
-- zakończenie deklaracji nagłówka
-- deklaracja ciała 'cialo' architektury
-- początek części wykonawczej
-- lista czułości procesu
-- utworzenie zmiennej 'L'
-- część wykonawcza procesu
-- ustalenie 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 procesu
-- zakończenie deklaracji ciała 'cialo'

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



IS Project Navigator J20.B1.J3.JC.Hol7.Dzr.SK.Jok.memy.Juk.Jdy.PUJET.Jpr.jek.ypr.access_fs_tbim.lse_jro.ass_for_jsu.xls - Process_for_bsum_tb.vhd

File Edit View Project Source Process Tools Window Layout Help

Design View Implement Sim

Hierarchy

```

1 entity PROCESS_FOR_BSUM_TB is -- pusty szkielet projektu symulacji
2 end PROCESS_FOR_BSUM_TB;
3
4 architecture behavioural of PROCESS_FOR_BSUM_TB is -- cialo architektoniczne projektu
5
6 signal A : bit_vector(3 downto 0); -- symulowane wejście A
7 signal Y : natural range 0 to 2; -- obserwowane wyjście Y
8
9 begin
10
11 process is
12 begin
13   A <= "0000"; wait for 10 ns; -- czesc wykonawcza procesu
14   A <= "0001"; wait for 10 ns;
15   A <= "0010"; wait for 10 ns;
16   A <= "0011"; wait for 10 ns;
17   A <= "0100"; wait for 10 ns;
18   A <= "0101"; wait for 10 ns;
19   A <= "0110"; wait for 10 ns;
20   A <= "0111"; wait for 10 ns;
21   A <= "1000"; wait for 10 ns;
22   A <= "1001"; wait for 10 ns;
23   A <= "1010"; wait for 10 ns;
24   A <= "1011"; wait for 10 ns;
25   A <= "1100"; wait for 10 ns;
26   A <= "1101"; wait for 10 ns;
27   A <= "1110"; wait for 10 ns;
28   A <= "1111"; wait for 10 ns;
29 end process; -- ustawienie sygnalu 'A'=0 i oczekanie 10 ns
30
31 process for_bsuma_inst: entity work.PROCESS_FOR_BSUM(cialo) -- instancja projektu 'PROCESS_FOR_BSUM'
32 port map (A => A, Y => Y); -- przypisanie portom sygnałów
33
34 end behavioural; -- zakończenie ciała architektonicznego
35
36

```

No Processes Running

Processes: PROCESS_FOR_BSUM

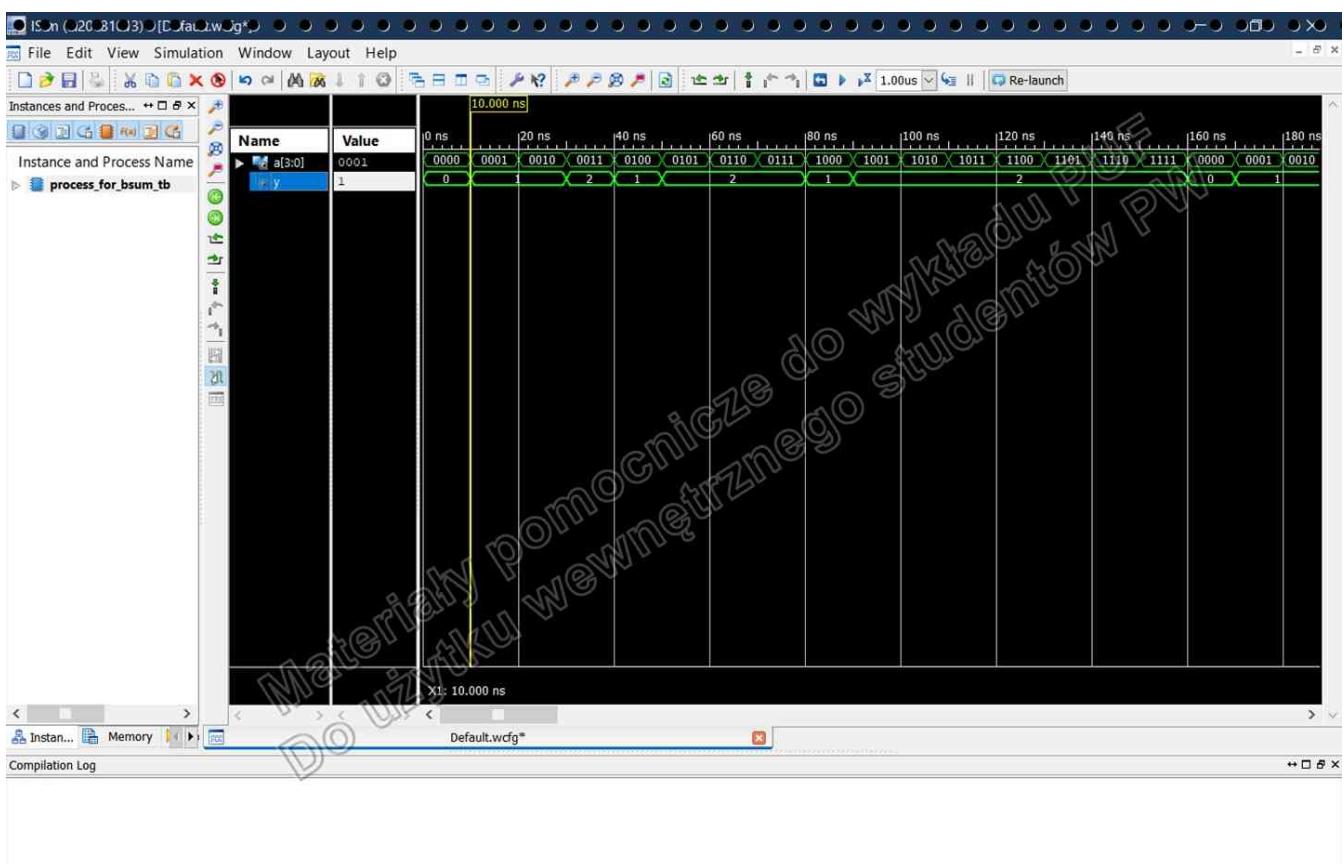
- ISim Simulator
- Behavioral Ch...
- Simulate Behav...

Design Summary (out of date) process_for_bsum.vhd PROCESS_FOR_BSUM (RTL1) PROCESS_FOR_BSUM (Tech1) process_for_bsum_tb.vhd

Ln 4 Col 1 VHDL

Przykład użycia oprogramowania ISE – projekt: „process_for_bsum”

Plik źródłowy „process_for_bsum_TB.vhd”



Przykład użycia oprogramowania ISim – projekt: „process_for_bsum”

Symulacja funkcjonalna

Podstawowe elementy standardu VHDL

Wybrane instrukcje w procesie

Podstawowa składnia pętli warunkowej:

[etykieta :] while wyrażenie_logiczne loop
instrukcja { instrukcja }
end loop [etykieta] ;

Pętla warunkowa jest powtarzana gdy **wyrażenie_logiczne** jest spełnione (tzn. zwraca wartość logiczną TRUE)

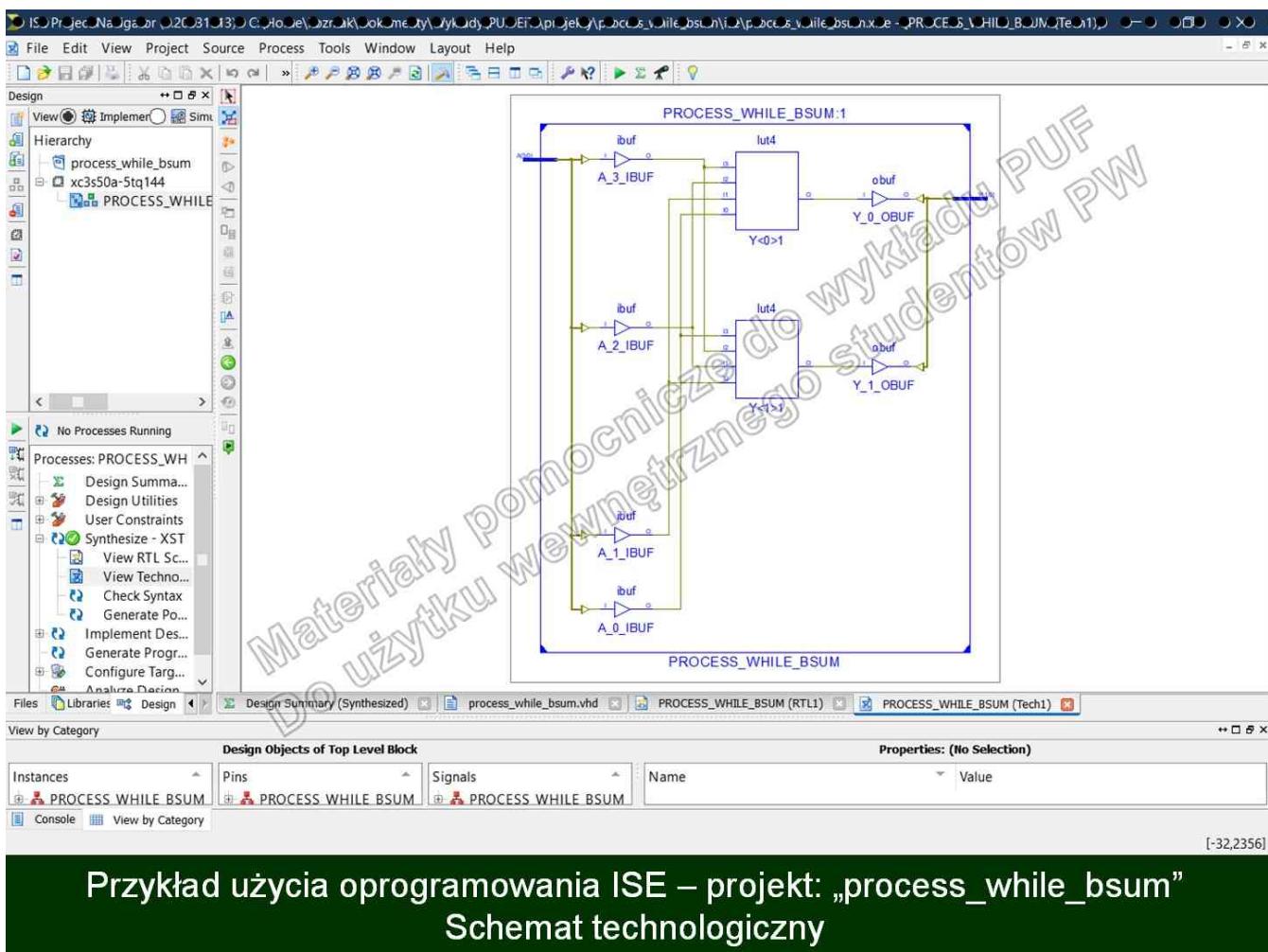
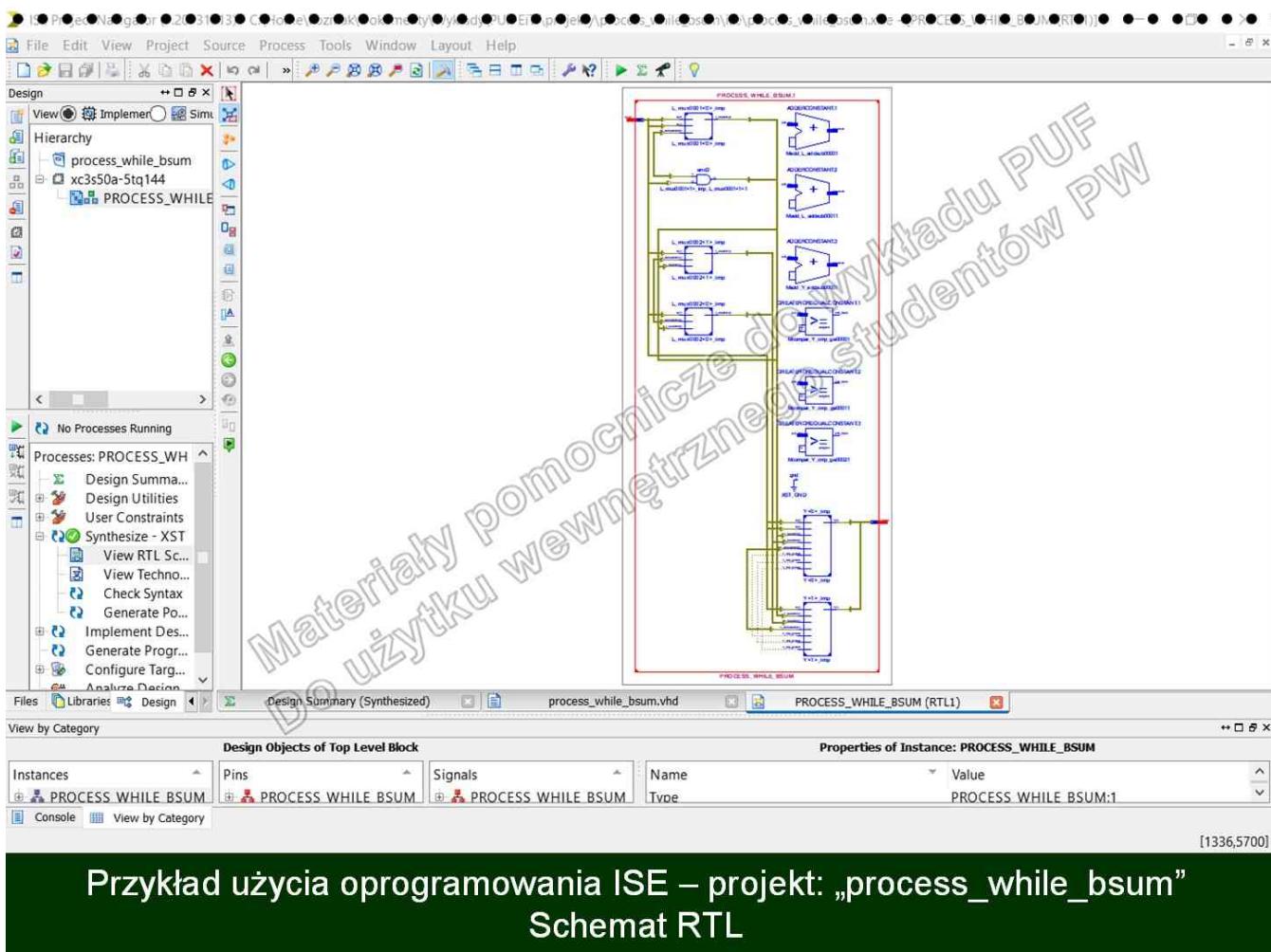
- podstawowe rodzaje zakresu dyskretnego:
 - wyrażenie proste (wymagane wartości FALSE lub TRUE)
przykłady: x lub $x > 0$ lub $x > y - 1$ lub $a > b$ or $c < d$
 - wyrażenie stałe (pętla pusta lub nieskończona)
przykłady: FALSE lub TRUE lub $1 = 1$



```
1 entity PROCESS WHILE_BSUM is
2     port ( A : in bit_vector(3 downto 0);
3             Y : out natural range 0 to 2
4         );
5 end PROCESS WHILE_BSUM;
6
7 architecture cialo of PROCESS WHILE_BSUM is
8
9 begin
10
11     process (A) is
12         variable P : natural range 0 to 4;
13         variable L : natural range 0 to 2;
14     begin
15         P := 0;
16         L := 0;
17         while P<4 and L<2 loop
18             if (A(P)=1') then
19                 L := L + 1;
20             end if;
21             P := P + 1;
22         end loop;
23         Y <= L;
24     end process;
25
26 end architecture cialo;
27
```

-- deklaracja sprzedu PROCESS WHILE_BSUM'
-- 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 zmiennej 'P'
-- utworzenie zmiennej 'L'
-- cześć wykonawcza procesu
-- ustawienie wartości początkowej 'P'
-- ustawienie wartości początkowej 'L'
-- pętla od 0 do 3 dla identyfikatora 'P'
-- badanie, czy bit o indeksie 'P' ma wartość '1'
-- zwiększenie o 1 licznika bitów '1'
-- zakończenie warunku
-- zwiększenie o 1 licznika bitów '1'
-- zakończenie pętli
-- przypisanie stanu 'L' do sygnału 'Y'
-- zakończenie procesu
-- zakończenie deklaracji ciała 'cialo'

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



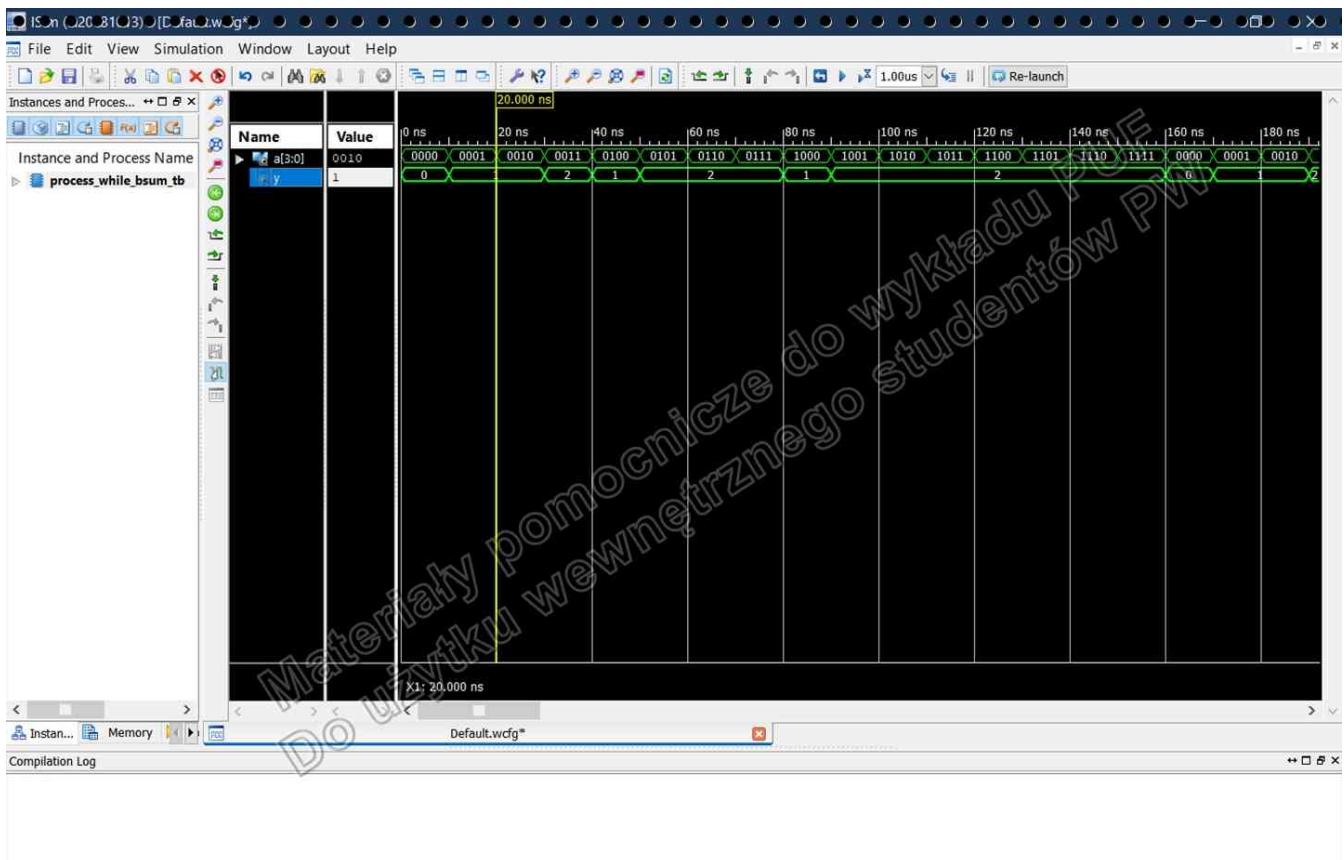
Przykład użycia oprogramowania ISE – projekt: „process_while_bsum”

Plik źródłowy „process_while_bsum_TB.vhd”

```

1 entity PROCESS_WHILE_BSUM_TB is -- pusty szkielet projektu symulacji
2 end PROCESS_WHILE_BSUM_TB;
3
4 architecture behavioural of PROCESS_WHILE_BSUM_TB is -- ciało architektoniczne projektu
5
6 signal A : bit_vector(3 downto 0); -- symulowane wejście A
7 signal Y : natural range 0 to 4; -- obserwowane wyjście Y
8
9 begin
10    process is
11        begin
12            A <= "0000"; wait for 10 ns;
13            A <= "0001"; wait for 10 ns;
14            A <= "0010"; wait for 10 ns;
15            A <= "0011"; wait for 10 ns;
16            A <= "0100"; wait for 10 ns;
17            A <= "0101"; wait for 10 ns;
18            A <= "0110"; wait for 10 ns;
19            A <= "0111"; wait for 10 ns;
20            A <= "1000"; wait for 10 ns;
21            A <= "1001"; wait for 10 ns;
22            A <= "1010"; wait for 10 ns;
23            A <= "1011"; wait for 10 ns;
24            A <= "1100"; wait for 10 ns;
25            A <= "1101"; wait for 10 ns;
26            A <= "1110"; wait for 10 ns;
27            A <= "1111"; wait for 10 ns;
28        end process;
29
30    process while_bsum_inst: entity work.PROCESS_WHILE_BSUM(ciało) -- instancja projektu 'PROCESS_FOR2A'
31        port map(
32            A => A,
33            Y => Y
34        );
35    end behavioural;
36

```



Przykład użycia oprogramowania ISim – projekt: „process_while_bsum”

Symulacja funkcjonalna

Podstawowe elementy standardu VHDL

Wybrane instrukcje w ciele jednostki projektowej

Podstawowa składnia pętli powielania:

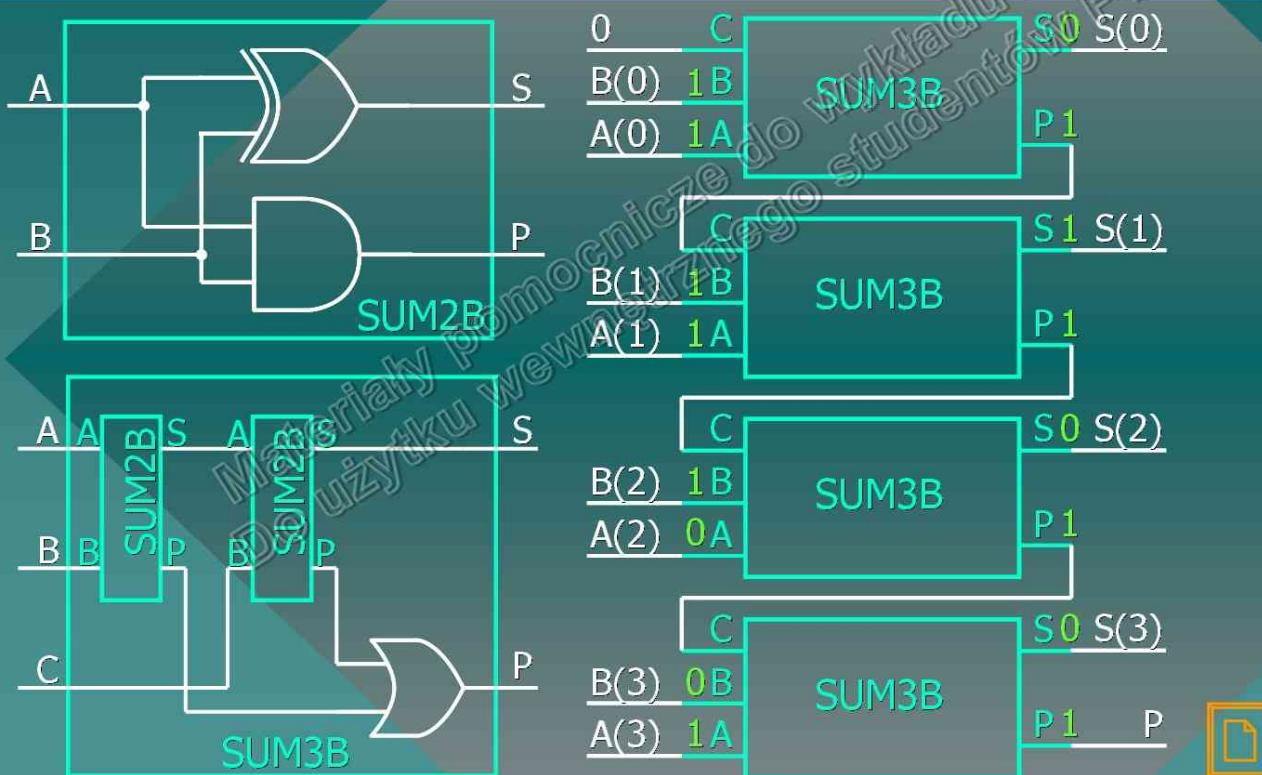
```
etykieta : for identyfikator in zakres_dyskretny generate
    [część_deklaracyjna]
        begin ]
            część_wykonawcza
    end generate [etykieta] ;
```

Podstawowa składnia generacji warunkowej:

etykieta : if warunek generate
[część_deklaracyjna
begin]
część_wykonawcza
end generate [etykieta] ;

Podstawowe elementy standardu VHDL

Przykład realizacji sumatora dwóch liczb 4-bitowych



Project Navigator 203133 C:\Users\Koci\Documents\My Documents\ISE\Projects\sum2x4b_gen - SUM2X4B_GEN

File Edit View Project Source Process Tools Window Layout Help

Design View Implement Sim

Hierarchy

- sum2x4b_gen
 - xc3s50a-5tq144
 - SUM2X4B_GENER
 - SUM3B_inst - SU
 - SUM2B_inst1
 - SUM2B_inst2

```

1 -- biblioteka STD jest wlaczana automatycznie do projektu
2
3 entity SUM2X4B_GENER is
4   port ( A : in bit_vector(3 downto 0);
5         B : in bit_vector(3 downto 0);
6         S : out bit_vector(3 downto 0);
7         P : out bit
8       );
9 end SUM2X4B_GENER;
10
11 architecture cialo of SUM2X4B_GENER is
12
13   signal K :bit_vector(3 downto 0);
14
15 begin
16
17   lg: for i in 0 to 3 generate
18     signal KI :bit := '0';
19   begin
20
21     i2: if (i>0) generate
22       KI <= K(i-1);
23     end generate;
24
25     SUM3B_inst: entity work.SUM3B
26       port map (A(i), B(i), KI, S(i), K(i));
27
28   end generate;
29
30   P <= K(3);
31
32 end architecture cialo;

```

-- deklaracja sprzegu 'SUM2X4B_GENER'
-- deklaracja portu wejsciowego 'A'
-- deklaracja portu wejsciowego 'B'
-- deklaracja portu wyjsciowego 'S'
-- deklaracja portu wyjsciowego 'P'
-- zakończenie deklaracji listy portów
-- zakończenie deklaracji sprzegu
-- deklaracja ciała 'cialo' architektury
-- deklaracja sygnału 'K'
-- początek części wykonawczej
-- czterokrotna pętla powielania
-- deklaracja sygnału w pętli powielania
-- część wykonawcza pętli powielania
-- generacja warunkowa gdy 'i>0'
-- przypisanie sygnału przeniesienia
-- zakończenie generacji warunkowej
-- dolaczenie projektu 'SUM3B'
-- podłączenie wejścia i wyjścia projektu
-- zakończenie pętli powielania
-- instrukcja przypisania sygnału do portu
-- zakończenie deklaracji ciała 'cialo'

Processes: SUM2X4B_GEN

- Design Summary
- Design Utilities
- User Constraints
- Synthesize - XST
 - View RTL Sc...
 - View Techno...
 - Check Syntax
 - Generate Po...
 - Implement Des...
 - Generate Prog...
 - Configure Targ...
 - Analyze Design

Files Libraries Design

Design Summary sum2x4b_generator.vhd

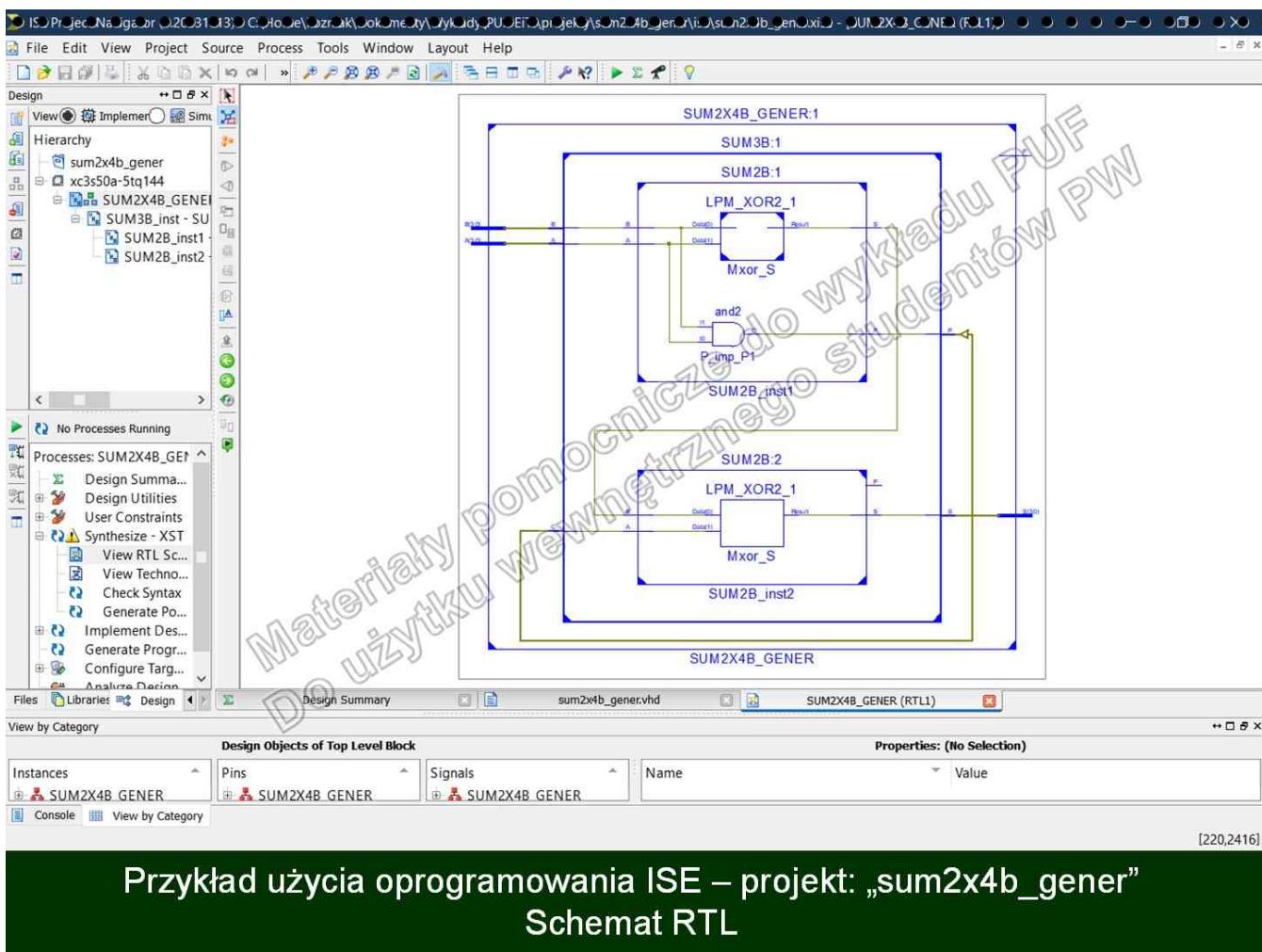
Console

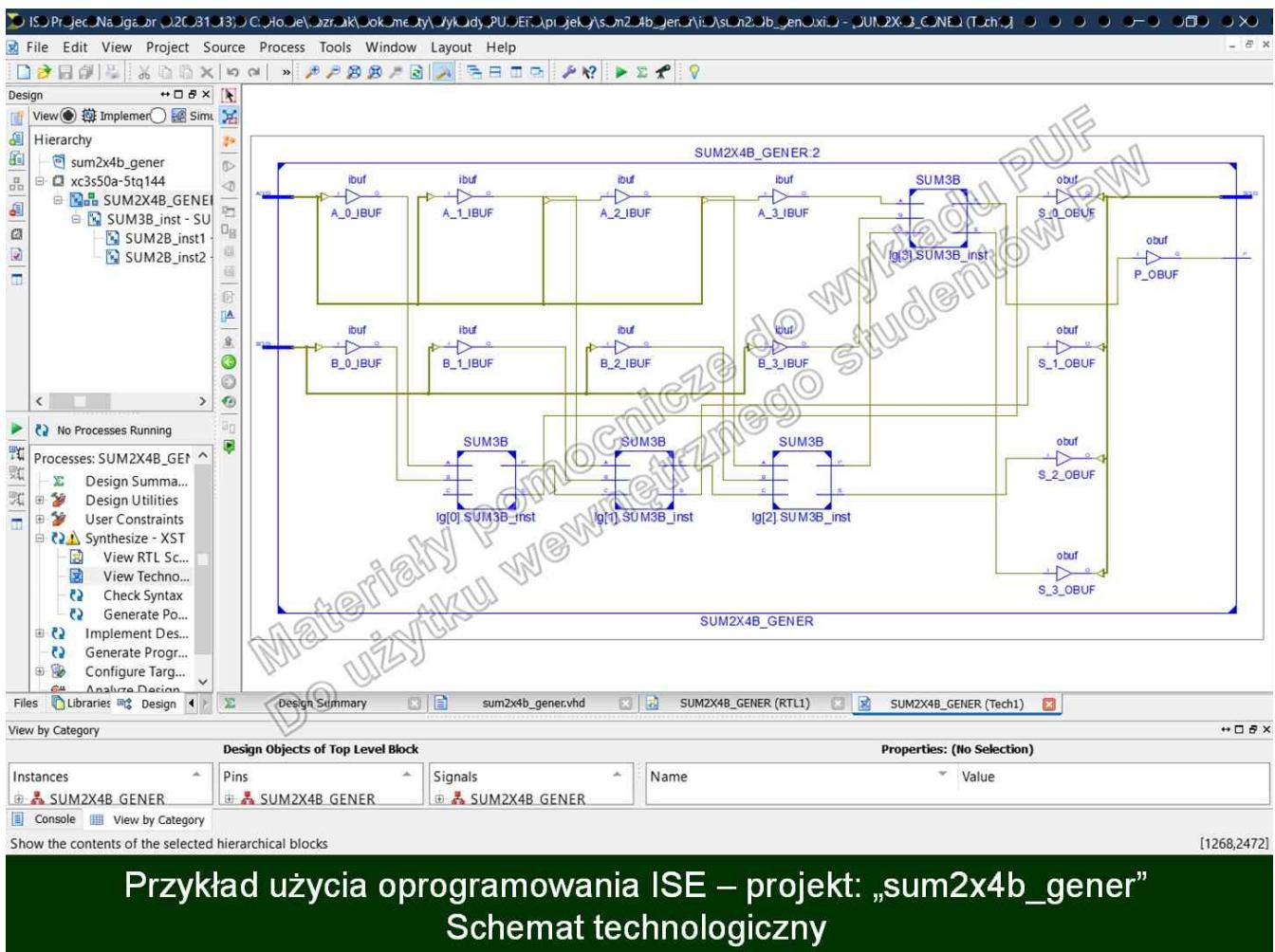
Started : "Launching ISE Text Editor to edit sum2x4b_generator.vhd".

Ln 11 Col 1 VHDL

Przykład użycia oprogramowania ISE – projekt: „sum2x4b_gener”

Plik źródłowy „sum2x4b_generator.vhd”





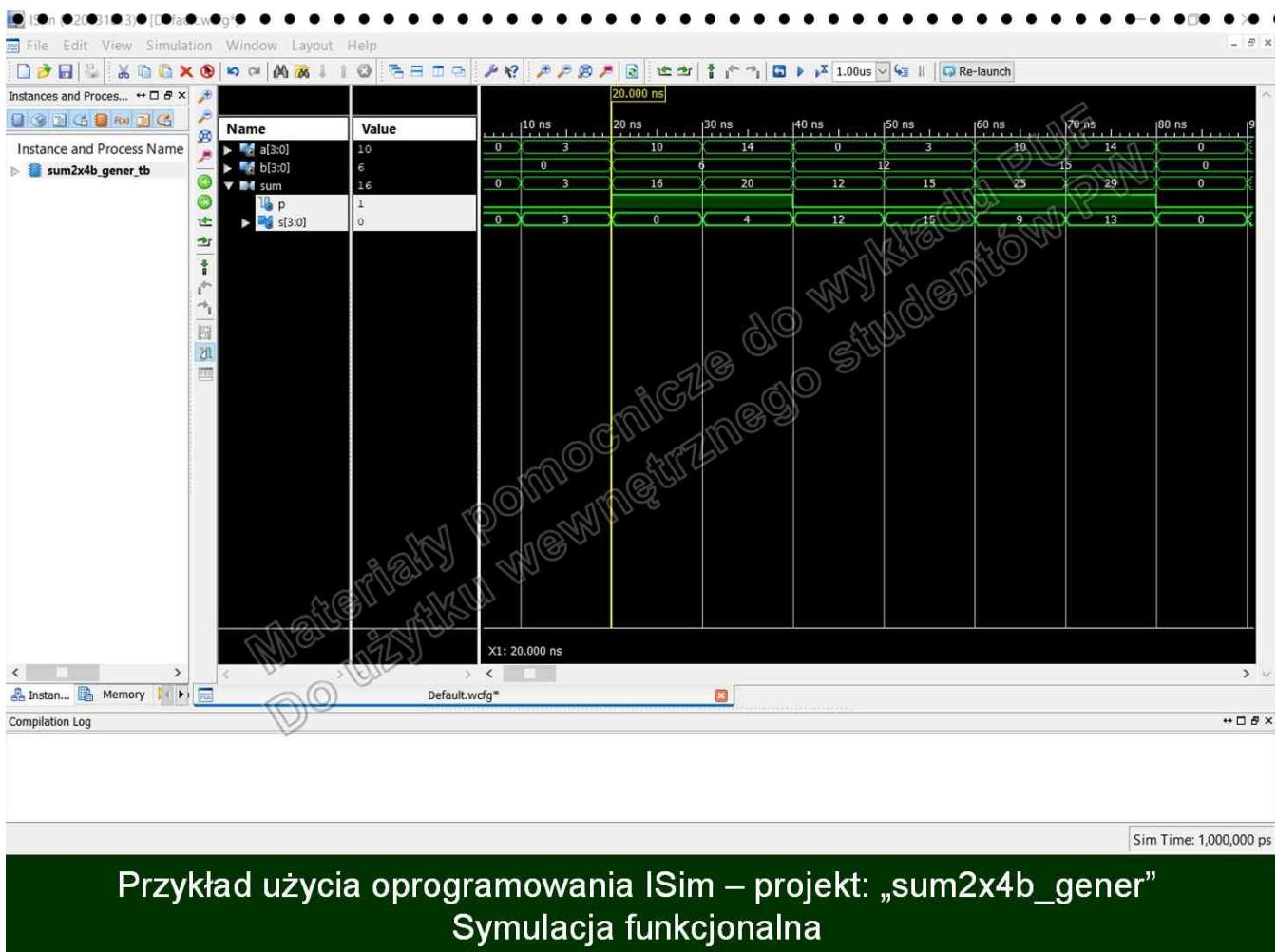
```

1 entity SUM2X4B_GENER_TB is          -- pusty szkielet projektu symulacji
2 end SUM2X4B_GENER_TB;
3
4 architecture behavioural of SUM2X4B_GENER_TB is -- cialo architektoniczne projektu
5
6   signal A : bit_vector(3 downto 0);      -- symulowane wejście A
7   signal B : bit_vector(3 downto 0);      -- symulowane wejście B
8   signal S : bit_vector(3 downto 0);      -- obserwowane wyjście S
9   signal P : bit;                        -- obserwowane wyjście P
10
11 begin
12
13   process is
14     begin
15       A <= "0000";
16       wait for 10 ns;
17       A <= "0011";
18       wait for 10 ns;
19       A <= "1010";
20       wait for 10 ns;
21       A <= "1110";
22       wait for 10 ns;
23     end process;
24
25   process is
26     begin
27       B <= "0000";
28       wait for 20 ns;
29       B <= "0110";
30       wait for 20 ns;
31       B <= "1100";
32       wait for 20 ns;
33       B <= "1111";
34       wait for 20 ns;
35     end process;
36
37   SUM2X4B_GENER_inst: entity work.SUM2X4B_GENER(cialo) -- instancja projektu 'SUM2X4B_GENER'
38   port map(
39     A => A,                                -- przypisanie sygnalu 'A' do portu 'A'
40     B => B,                                -- przypisanie sygnalu 'B' do portu 'B'
41     S => S,                                -- przypisanie sygnalu 'S' do portu 'S'
42     P => P                                -- przypisanie sygnalu 'P' do portu 'P'
43   );
44 end behavioural;

```

Materiały Pomocnicze do wykładu PUF
Drukuj użytkowi wewnętrznego Studentów PW

Przykład użycia oprogramowania ISE – projekt: „sum2x4b_gener”
Plik źródłowy „sum2x4b_TB.vhd”



Podstawowe elementy standardu VHDL

Parametryzowany sprzęt jednostki projektowej

Składnia definicji sprzęgu z listą parametrów stałych i portów:

```
entity nazwa_sprzęgu is
  [generic ( deklaracja_stałej {; deklaracja_stałej} ) ;]
  [port ( deklaracja_portu {; deklaracja_portu} ) ;]
end [entity] [nazwa_sprzęgu];
```

Składnia deklaracji stałej:

```
[constant] nazwa1 { , nazwa2 } : [ in ] typ [:= wyrażenie]
```

Przykłady definicji sprzęgu z listą parametrów stałych:

```
entity sprzeg is
  generic ( constant stala : in integer := 2 );
  port ( wejscie : in bit_vector(stala1-1 downto 0);
         wyjscie : out boolean );
end entity sprzeg ;
```

Podstawowe elementy standardu VHDL

Parametryzowany sprzęt jednostki projektowej

Składnia definicji sprzęgu z listą parametrów stałych i portów:

```
entity nazwa_sprzęgu is
[ generic ( deklaracja_stałej {; deklaracja_stałej} ) ; ]
[ port ( deklaracja_portu {; deklaracja_portu} ) ; ]
end [entity] [nazwa_sprzęgu] ;
```

Składnia deklaracji stałej:

```
[ constant ] nazwa1 { , nazwa2 } : [ in ] typ [:= wyrażenie]
```

Przykłady definicji sprzęgu z listą parametrów stałych:

```
entity sprzeg is
generic (stala1, stala2 : integer);
port (wejście : in bit_vector(stala1-1 downto 0) ;
      wyjście : out bit_vector(stala2-1 downto 0) ) ;
end entity sprzeg ;
```

Podstawowe elementy standardu VHDL

Parametryzowany sprzęt jednostki projektowej

Składnia definicji sprzęgu z listą parametrów stałych i portów:

```
entity nazwa_sprzęgu is
[ generic ( deklaracja_stałej {; deklaracja_stałej} ) ; ]
[ port ( deklaracja_portu {; deklaracja_portu} ) ; ]
end [entity] [nazwa_sprzęgu] ;
```

Składnia deklaracji stałej:

```
[ constant ] nazwa1 { , nazwa2 } : [ in ] typ [:= wyrażenie]
```

Składnia instancji bezpośredniej jednostki projektowej z portami:

```
etykieta : entity work.nazwa_sprzęgu [ ( nazwa_ciała ) ]
generic map (połączenie_stalej {, połączenie_stalej} )
port map (połączenie_portu {, połączenie_portu} );
```

Przykład użycia oprogramowania ISE – projekt: „sum2_param”

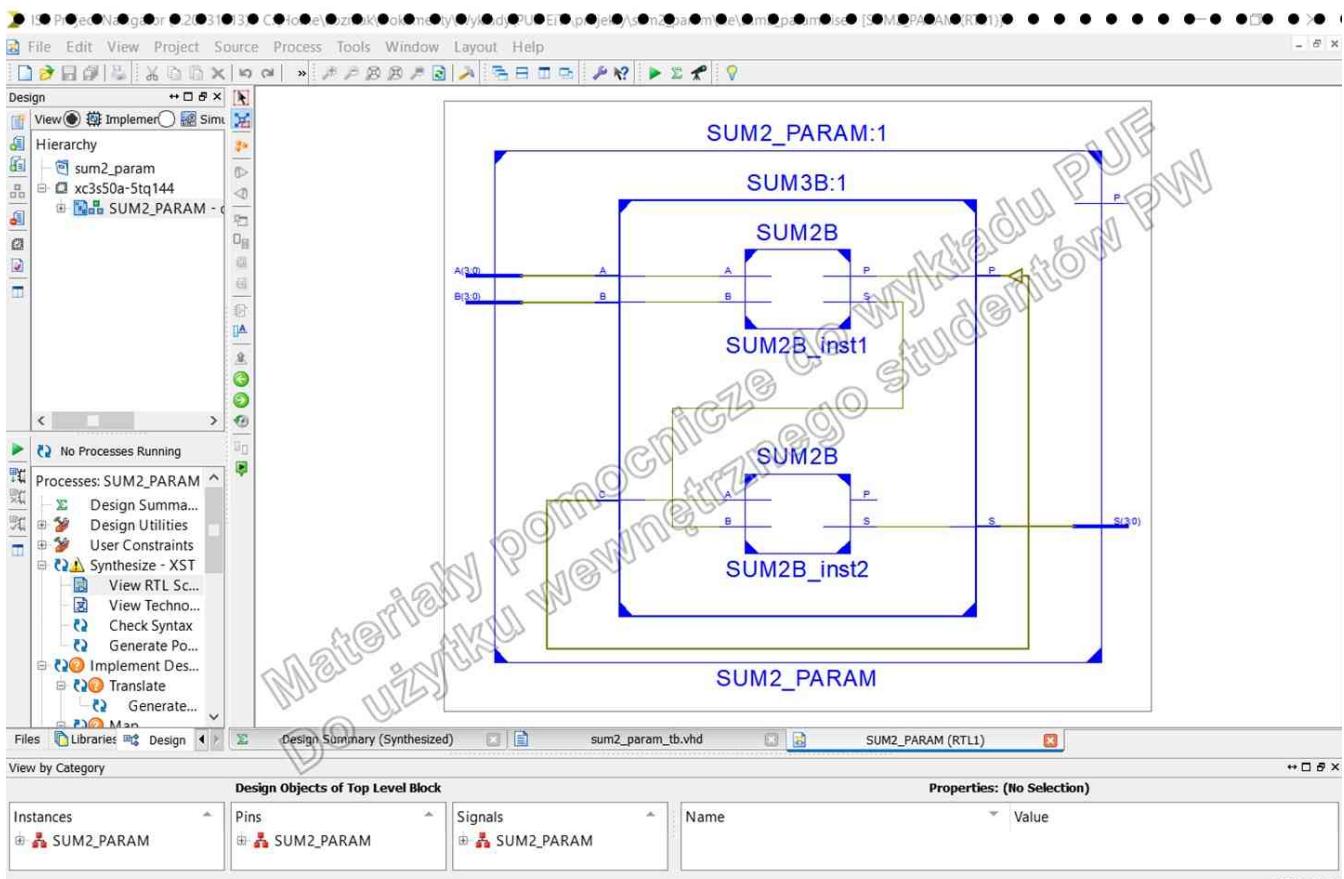
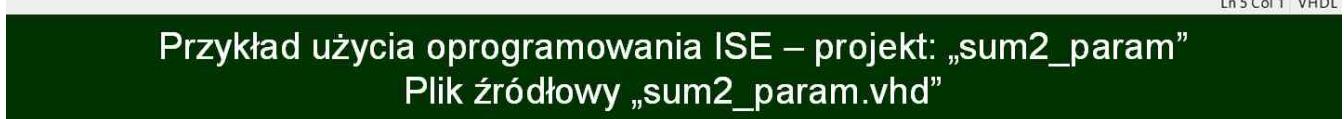
Plik źródłowy „sum2_param.vhd”

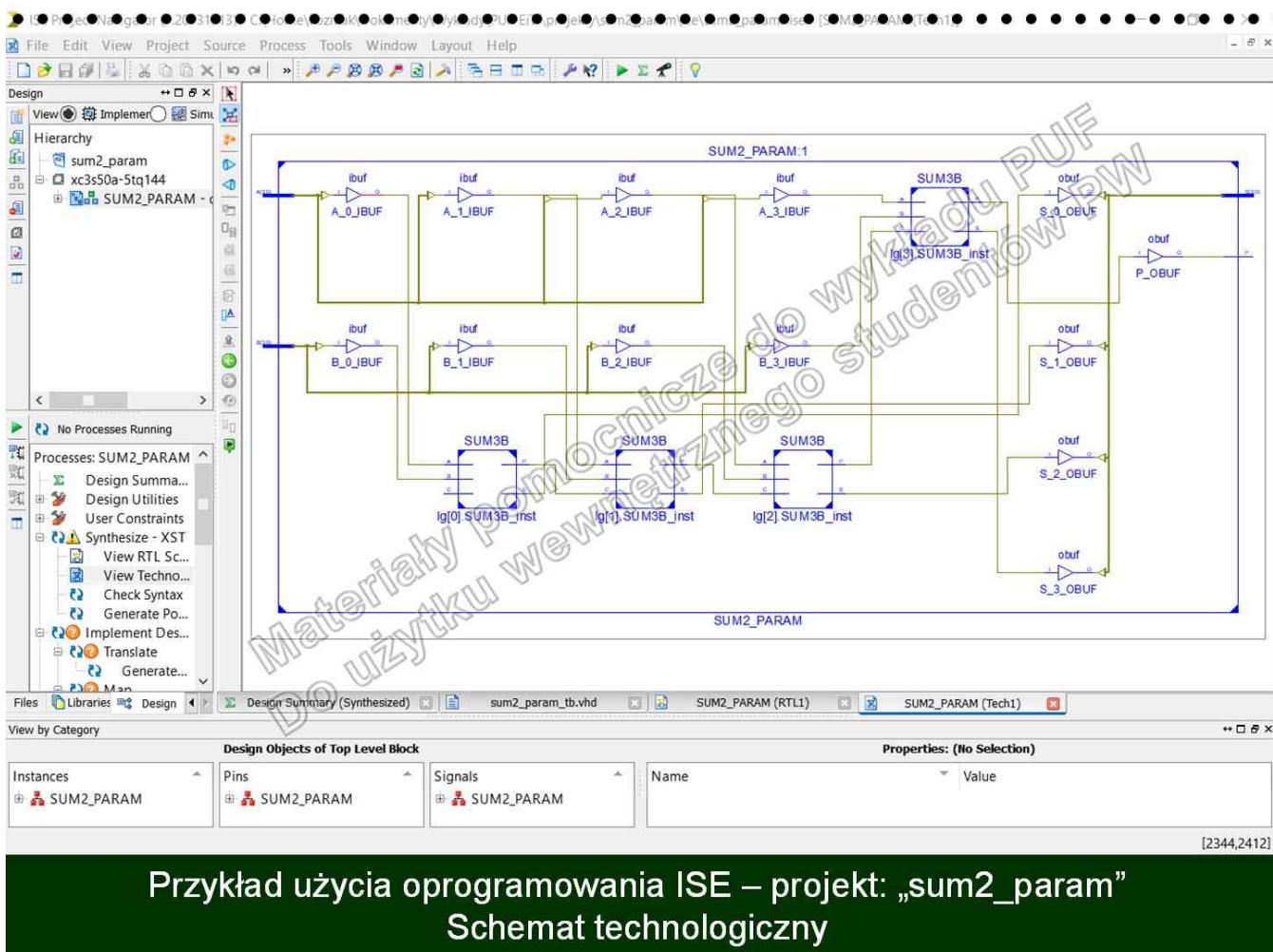
```

1 entity SUM2_PARAM_TB is
2 generic (N : integer := 8);
3 end SUM2_PARAM_TB;
4
5 architecture behavioural of SUM2_PARAM_TB is -- cialo architektoniczne projektu
6
7 signal A : bit_vector(N-1 downto 0) := (others => '0'); -- symulowane wejście A
8 signal B : bit_vector(N-1 downto 0) := (others => '0'); -- symulowane wejście B
9 signal S : bit_vector(N-1 downto 0); -- obserwowane wyjście 'S'
10 signal P : bit; -- obserwowane wyjście P
11
12 begin
13
14 al: for i in 1 to N-1 generate
15 process is
16 begin
17 wait for (i+1)*10 ns;
18 A(i) <= not(A(i));
19 end process;
20 end generate;
21
22 bl: for i in 0 to N-1 generate
23 process is
24 begin
25 wait for N*(i+1)*10 ns;
26 B(i) <= not(B(i));
27 end process;
28 end generate;
29
30 SUM2_PARAM inst: entity work.SUM2_PARAM(cialo) -- instancja projektu 'SUM2_PARAM'
31 generic map (N => N)
32 port map (
33 A => A,
34 B => B,
35 S => S,
36 P => P
37 );
38 end behavioural;

```

-- sprzeg projektu symulacji
-- deklaracja parametru 'N'
-- N-krotna petla powielania
-- proces bezwarunkowy
-- czesc wykonawcza procesu
-- oczekanie '(i+1)'-krotne 10 ns
-- przypisanie sygnalowi 'A' wartosci
-- zakoñczenie procesu
-- N-krotna petla powielania
-- proces bezwarunkowy
-- czesc wykonawcza procesu
-- oczekanie 'N*(i+1)'-krotne 10 ns
-- przypisanie sygnalowi 'A' wartosci
-- zakoñczenie procesu
-- instancja projektu 'SUM2_PARAM'
-- mapowanie parametru
-- mapowanie portow
-- przypisanie sygnalu 'A' do portu 'A'
-- przypisanie sygnalu 'B' do portu 'B'
-- przypisanie sygnalu 'S' do portu 'S'
-- przypisanie sygnalu 'P' do portu 'P'
-- zakoñczenie ciala architektonicznego





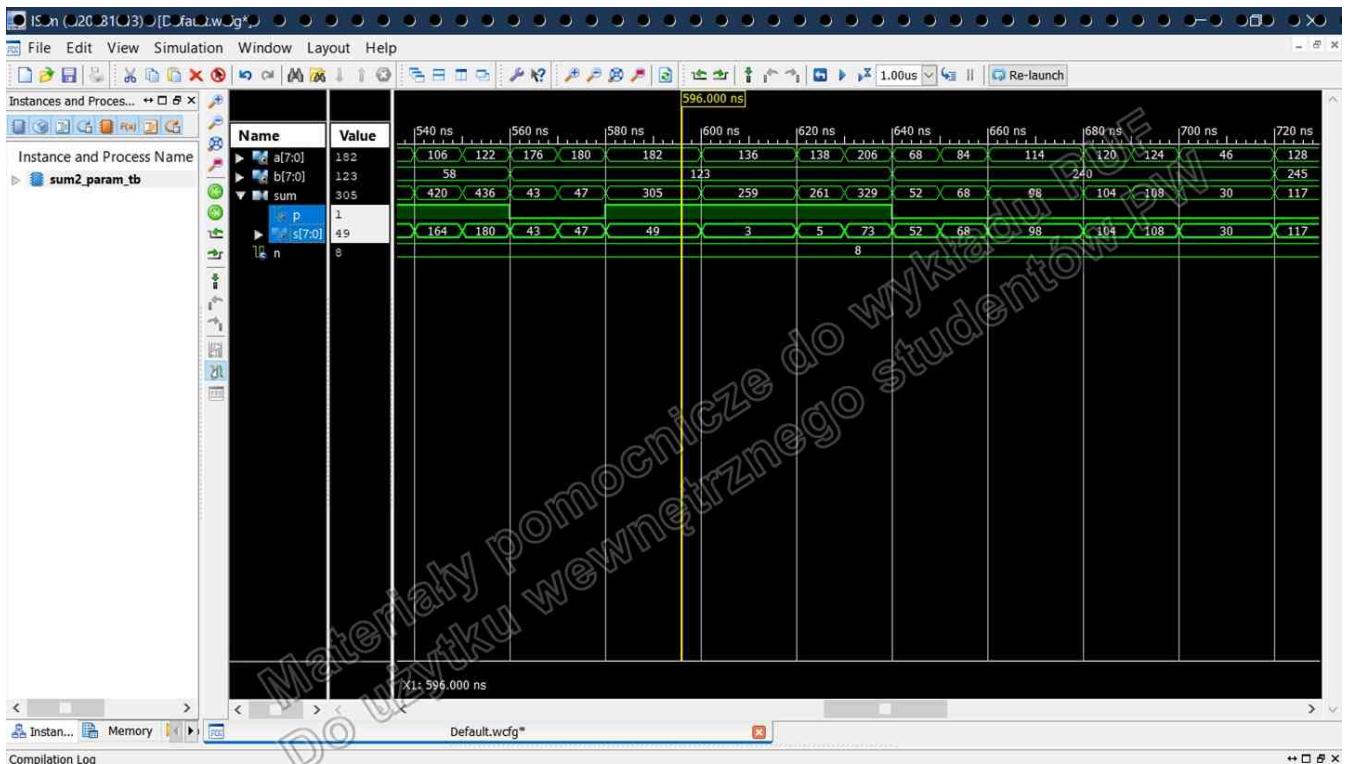
```

1 entity SUM2_PARAM_TB is
2     generic (N : integer := 8); -- sprzed projektu symulacji
3 end SUM2_PARAM_TB; -- deklaracja parametru 'N'
4
5 architecture behavioural of SUM2_PARAM_TB is -- cialo architektoniczne projektu
6
7     signal A : bit_vector(N-1 downto 0) := (others => '0'); -- symulowane wejscie A
8     signal B : bit_vector(N-1 downto 0) := (others => '0'); -- symulowane wejscie B
9     signal S : bit_vector(N-1 downto 0); -- obserwowane wyjoscie S
10    signal P : bit; -- obserwowane wyjoscie P
11
12 begin
13
14     al: for i in 1 to N-1 generate
15         process is
16             begin
17                 wait for (i+1)*10 ns;
18                 A(i) <= not(A(i));
19             end process;
20         end generate;
21
22     bl: for i in 0 to N-1 generate
23         process is
24             begin
25                 wait for N*(i+1)*10 ns;
26                 B(i) <= not(B(i));
27             end process;
28         end generate;
29
30     SUM2_PARAM_inst: entity work.SUM2_PARAM(cialo) -- instancja projektu 'SUM2_PARAM'
31         generic map (N => N)
32         port map (
33             A => A,
34             B => B,
35             S => S,
36             P => P
37         );
38 end behavioural;

```

Materiały Pomocnicze do wykazu PUF do użytku wewnętrznego Studentów PW

Przykład użycia oprogramowania ISE – projekt: „sum2_param”
Plik źródłowy „sum2_param_TB.vhd”



Przykład użycia oprogramowania ISim – projekt: „sum2_param”
Symulacja funkcjonalna