KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

„Układy reprogramowalne i SoC”
„Język VHDL (część 1)”

Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”

Prezentacja dystrybuowana jest bezpłatnie

O VHDL

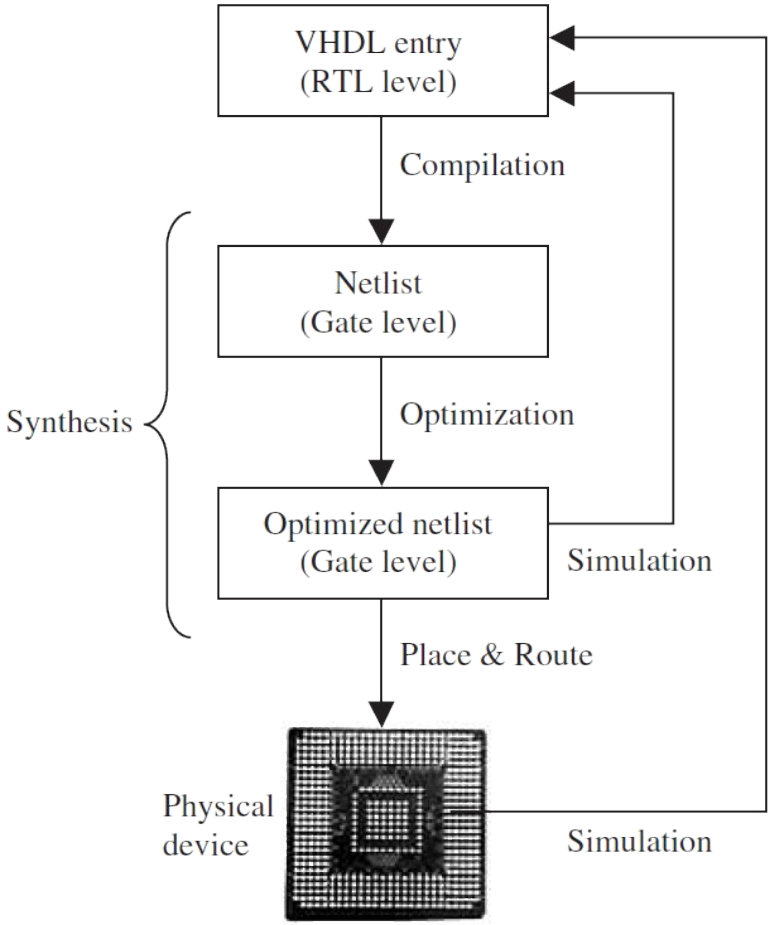
- VHDL jest językiem opisu sprzętu
  - Opisuje zachowanie układu lub systemu elektronicznego
  - Na podstawie tego opisu układ lub system będzie zaimplementowany
- Very High Speed Integrated Circuits Hardware Description Language
  - Opracowany na zamówienie ministerstwa obrony USA
    - VHDL 87
    - VHDL 93
- Pierwszy język opisu sprzętu, który został standardem IEEE
  - IEEE 1076
  - IEEE 1164 opisuje logikę wielowartościową
- VHDL może służyć zarówno do *syntezy* jak i *symulacji*
  - Nie wszystkie konstrukcje VHDL są syntezowalne

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Proces projektowania z użyciem VHDL



Rysunek: Pedroni V. A., "Circuit Design with VHDL"

Narzędzia projektowe

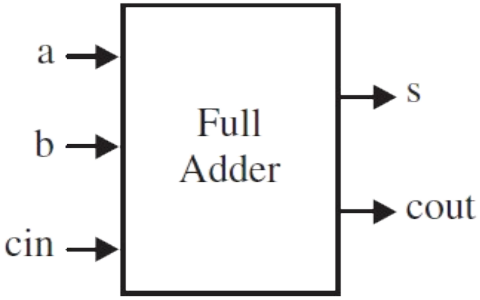
- Electronic Design Automation (EDA)
- Oferowane przez
  - Producentów układów programowalnych
    - Altera Quartus II
    - Xilinx ISE
  - Niezależni producenci oprogramowania
    - Leonardo Spectrum (Mentor Graphics)
    - Synplify (Synopsys)
    - ModelSim (Mentor Graphics)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

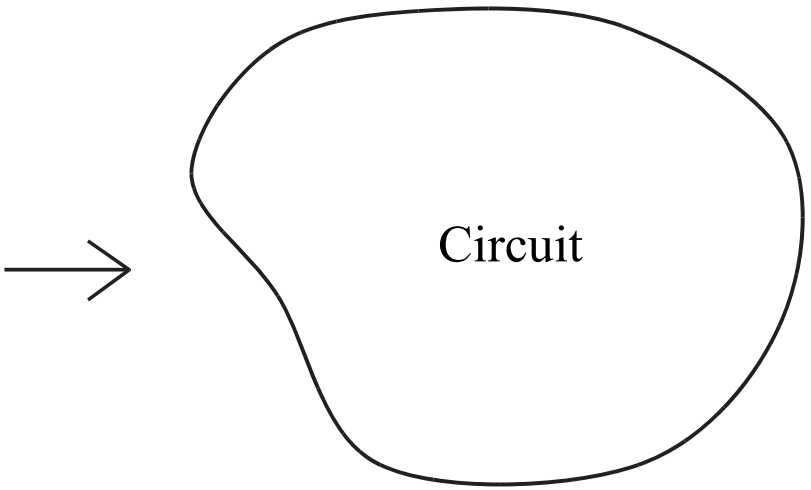
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykład: sumator



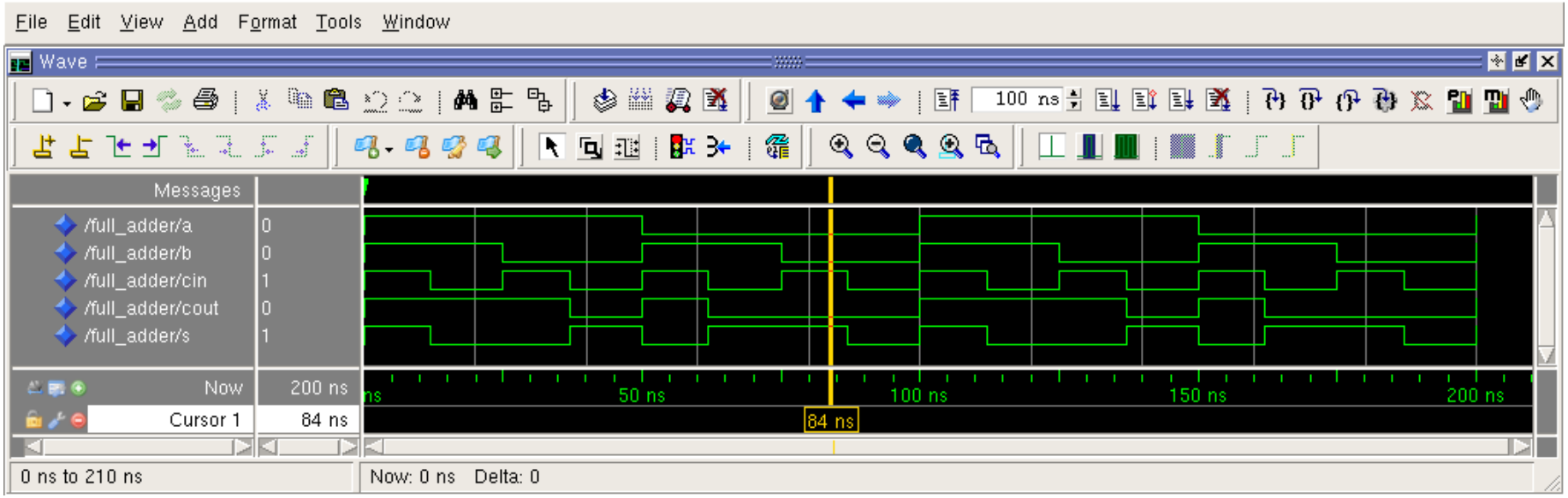
a b	cin	s	cout
0 0	0	0	0
0 1	0	1	0
1 0	0	1	0
1 1	0	0	1
0 0	1	1	0
0 1	1	0	1
1 0	1	0	1
1 1	1	1	1

```
ENTITY full_adder IS
   PORT (a, b, cin: IN BIT;
         s, cout: OUT BIT);
END full_adder;
-------------------------------------
ARCHITECTURE dataflow OF full_adder IS
BEGIN
   s <= a XOR b XOR cin;
   cout <= (a AND b) OR (a AND cin) OR
           (b AND cin);
END dataflow;
```



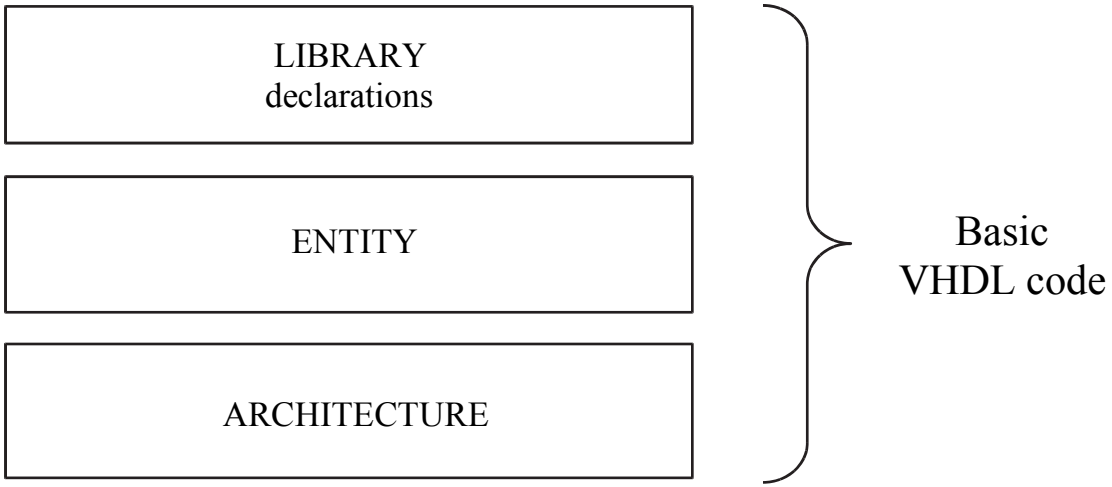
Symulacja

```
cd project_directory
vlib work
vcom adder.vhd
vsim
# ...
# tutaj skonfigurowanie wyświetlania, wymuszeń okresowych itd.
# ...
run 200 ns
```

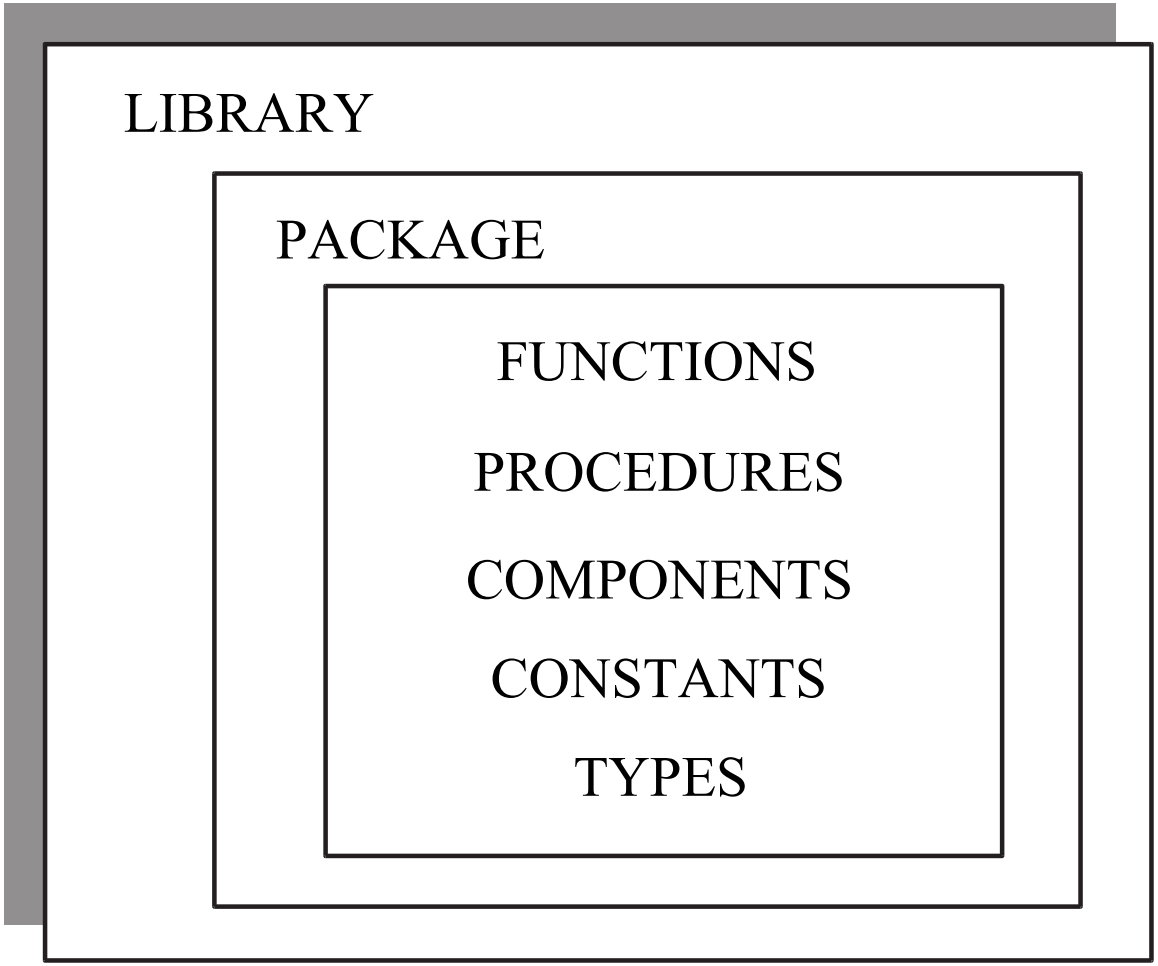


Podstawowe elementy VHDL

- Deklaracje **`LIBRARY`**
  - Zawiera listę bibliotek używanych w projekcie. Na przykład: ieee, std, work, itd.
- **`ENTITY`**
  - Określa wyprowadzenia układu
- **`ARCHITECTURE`**
  - Zawiera kod VHDL, opisujący jak obwód powinien się zachowywać (funkcjonalność obwodu)



Struktura biblioteki



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Deklaracja biblioteki

- Aby zawartość biblioteki była widzialna w projekcie, potrzebne są dwie linie kodu: jedna zawierająca nazwę biblioteki, a druga klauzulę use

```
LIBRARY library_name;
USE library_name.package_name.package_parts;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Najczęściej stosowane biblioteki

- Trzy pakiety, z trzech różnych bibliotek, są przeważnie potrzebne w projekcie:
  - ieee.std_logic_1164 (z biblioteki ieee),
  - standard (z biblioteki std),
  - work (biblioteka work)

```
LIBRARY ieee;                    -- A semi-colon (;) indicates
USE ieee.std_logic_1164.all; -- the end of a statement or

LIBRARY std;                     -- declaration, while a double
USE std.standard.all;            -- dash (--) indicates a comment.

LIBRARY work;
USE work.all;
```

- Bibilioteki std i work są domyślnie włączane, nie ma potrzeby ich jawnego dołączania

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Biblioteka ieee

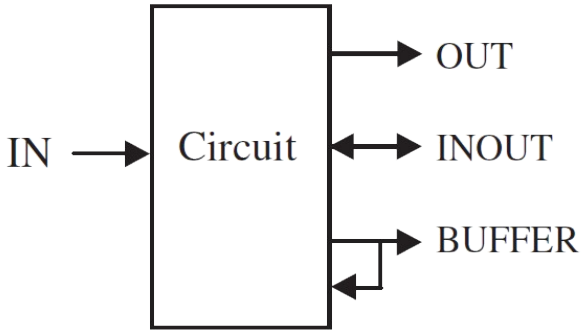
- std_logic_1164: systemy logiki wielowartościowej
  - STD_LOGIC (9 wartości logicznych)
  - STD_ULOGIC (9 wartości logicznych, bez rozwiązywania konfliktów)
- std_logic_arith: defacto standard Synopsys
  - Typy danych SIGNED i UNSIGNED oraz powiązane operatory arytmetyczne i porównań
  - Funkcje konwersji danych między różnymi typami
    - conv_integer(p)
    - conv_unsigned(p, b)
    - conv_signed(p, b)
    - conv_std_logic_vector(p, b).
- std_logic_signed: Zawiera funkcje pozwalające na wykonywanie operacji na typie STD_LOGIC_VECTOR tak jak na typie SIGNED. Defacto standard Synopsys
- std_logic_unsigned: Zawiera funkcje pozwalające na wykonywanie operacji na typie STD_LOGIC_VECTOR tak jak na typie UNSIGNED. Defacto standard Synopsys
- numeric_std: Alternatywa do std_logic_arith. Standard IEEE
- Nie należy używać jednocześnie std_logic_arith i numeric_std

ENTITY

- Deklaracja ENTITY zawiera listę portów wejściowych i wyjściowych układu.

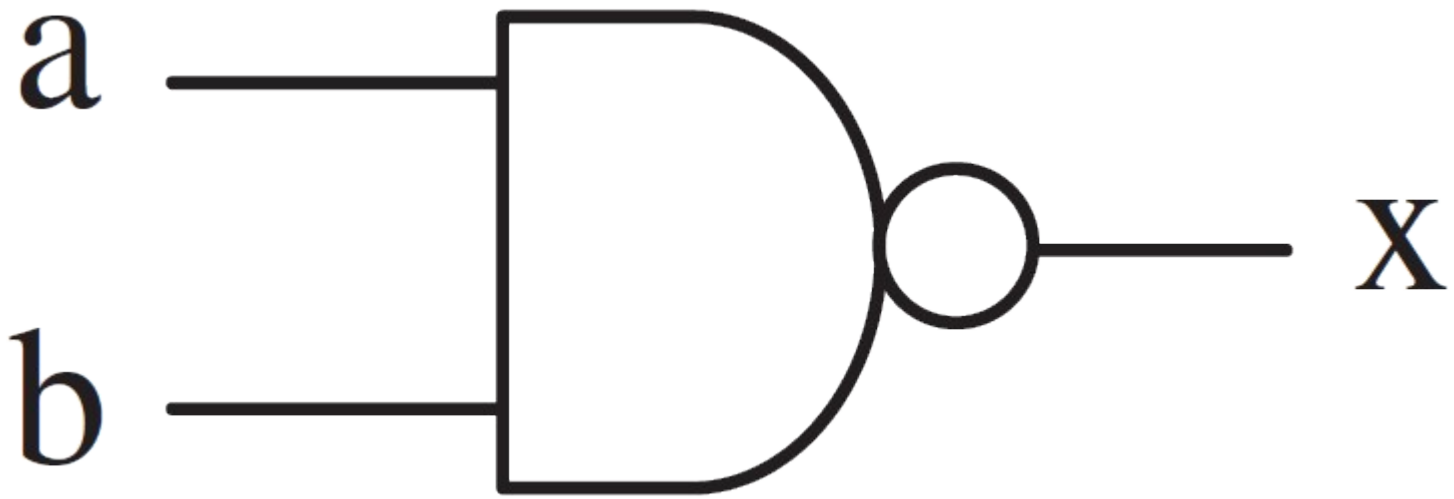
```
ENTITY entity_name IS
    PORT (
        port_name : signal_mode signal_type;
        port_name : signal_mode signal_type;
        ...);
END entity_name;
```

- Możliwe są następujące tryby portów:
  - IN
  - OUT
  - INOUT
  - BUFFER



Przykład ENTITY

```
ENTITY nand_gate IS
        PORT (a, b : IN BIT;
                x : OUT BIT);
END nand_gate;
```



ARCHITECTURE

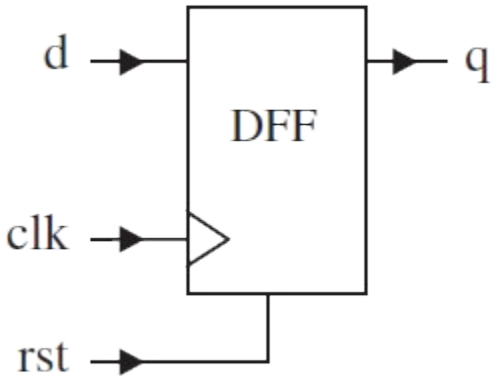
- ARCHITECTURE opisuje, jak układ powinien funkcjonować.

```
ARCHITECTURE architecture_name OF entity_name IS
    [declarations]
BEGIN
    (code)
END architecture_name;
```

```
ARCHITECTURE myarch OF nand_gate IS
BEGIN
    x <= a NAND b;
END myarch;
```

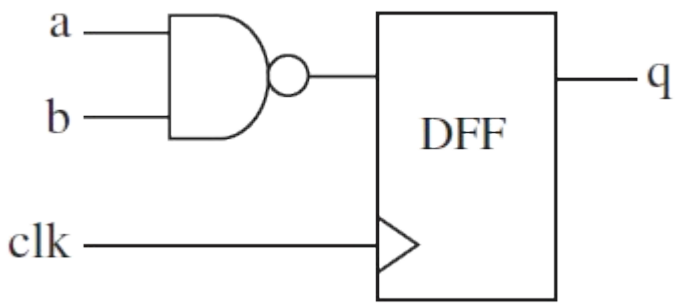
Przerzutnik wyzwalany zboczem z asynchronicznym resetem

```
1 ---------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ---------------------------------------
5 ENTITY dff IS
6    PORT ( d, clk, rst: IN STD_LOGIC;
7           q: OUT STD_LOGIC);
8 END dff;
9 ---------------------------------------
10 ARCHITECTURE behavior OF dff IS
11 BEGIN
12    PROCESS (rst, clk)
13    BEGIN
14       IF (rst='1') THEN
15          q <= '0';
16       ELSIF (clk'EVENT AND clk='1') THEN
17          q <= d;
18       END IF;
19    END PROCESS;
20 END behavior;
21 ---------------------------------------
```



Przerzutnik i bramka NAND

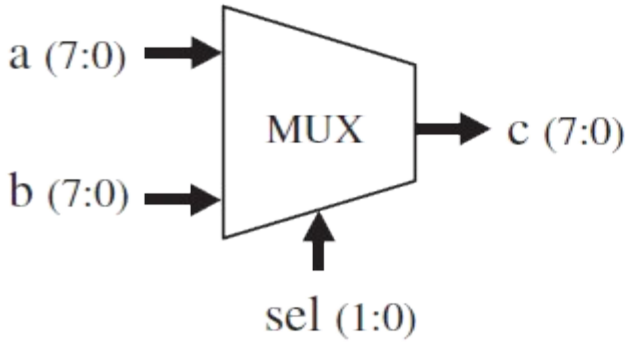
```
1 ---------------------------------------
2 ENTITY example IS
3    PORT ( a, b, clk: IN BIT;
4           q: OUT BIT);
5 END example;
6 ---------------------------------------
7 ARCHITECTURE example OF example IS
8    SIGNAL temp : BIT;
9 BEGIN
10   temp <= a NAND b;
11   PROCESS (clk)
12      BEGIN
13         IF (clk'EVENT AND clk='1') THEN q<=temp;
14         END IF;
15      END PROCESS;
16 END example;
17 ---------------------------------------
```



Pedroni V. A., "Circuit Design with VHDL"

Multiplekser

Pedroni V. A., "Circuit Design with VHDL"

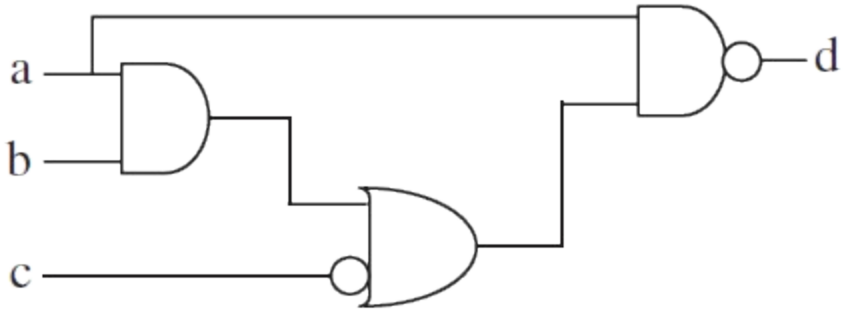


sel	c
00	0
01	a
10	b
11	Z

```
1 -------------------------------------
2 LIBRARY ieee;
3 USE ____________________________ ;
4 -------------------------------------
5 ENTITY mux IS
6    PORT ( __ , __ : ___ STD_LOGIC_VECTOR (7 DOWNTO 0);
7           sel : IN ___________________________ ;
8           ___ : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
9 END _____ ;
10 -------------------------------------
11 ARCHITECTURE example OF _____ IS
12 BEGIN
13    PROCESS (a, b, ____ )
```

```
14    BEGIN
15       IF (sel = "00") THEN
16          c <= "00000000";
17       ELSIF (__________) THEN
18          c <= a;
19          _____ (sel = "10") THEN
20          c <= __;
21       ELSE
22          c <= (OTHERS => '__');
23       END ___ ;
24    END _________ ;
25 END _________ ;
26 -------------------------------------
```

Układ bramek



Pedroni V. A., "Circuit Design with VHDL"

Typy danych

- std.standard:
  - BIT, BOOLEAN, INTEGER i REAL
- ieee.std_logic_1164:
  - STD_LOGIC i STD_ULOGIC
- ieee.std_logic_arith:
  - SIGNED i UNSIGNED
  - conv_integer(p),conv_unsigned(p, b), conv_signed(p, b), i conv_std_logic_vector(p, b).
- ieee.std_logic_signed, ieee.std_logic_unsigned:
  - Funkcje pozwalające na operacje na danych typu STD_LOGIC_VECTOR odpowiednio jak na SIGNED lub UNSIGNED

BIT (i BIT_VECTOR)

- Logika dwuwartościowa

```
SIGNAL x: BIT;
-- x is declared as a one-digit signal of type BIT.
SIGNAL y: BIT_VECTOR (3 DOWNTO 0);
-- y is a 4-bit vector, with the leftmost bit being the MSB.
SIGNAL w: BIT_VECTOR (0 TO 7);
-- w is an 8-bit vector, with the rightmost bit being the MSB.
```

```
x <= '1';
-- x is a single-bit signal (as specified above), whose value is
-- '1'. Notice that single quotes (' ') are used for a single bit.
y <= "0111";
-- y is a 4-bit signal (as specified above), whose value is "0111"
-- (MSB='0'). Notice that double quotes (" ") are used for
-- vectors.
w <= "01110001";
-- w is an 8-bit signal, whose value is "01110001" (MSB='1').
```

STD_LOGIC (i STD_LOGIC_VECTOR)

- Logika dziewięciowartościowa według standardu IEEE 1164

```
'U' Uninitialized
'X' Forcing Unknown
'0' Forcing Low (synthesizable logic '1')
'1' Forcing High (synthesizable logic '0')
'Z' High impedance (synthesizable tri-state buffer)
'W' Weak unknown
'L' Weak low
'H' Weak high
'-' Don't care
```

```
SIGNAL x: STD_LOGIC;
-- x is declared as a one-digit (scalar) signal of type STD_LOGIC.
SIGNAL y: STD_LOGIC_VECTOR (3 DOWNTO 0) := "0001";
-- y is declared as a 4-bit vector, with the leftmost bit being
-- the MSB. The initial value (optional) of y is "0001". Notice
-- that the ":=" operator is used to establish the initial value.
```

Tablica rozwiązywania konfliktów dla typu STD_LOGIC

	U	X	0	1	Z	W	L	H	-
U	U	U	U	U	U	U	U	U	U
X	U	X	X	X	X	X	X	X	X
0	U	X	0	X	0	0	0	0	X
1	U	X	X	1	1	1	1	1	X
Z	U	X	0	1	Z	W	L	H	X
W	U	X	0	1	W	W	W	W	X
L	U	X	0	1	L	W	L	W	X
H	U	X	0	1	H	W	W	H	X
-	U	X	X	X	X	X	X	X	X

STD_ULOGIC

- Brak rozwiązywania konfliktów
- Nie można łączyć bezpośrednio wyjść takiego typu
- Może być użyty do wykrycia błędów projektowych polegających na omyłkowym połączeniu przewodów

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Pozostałe typy

- BOOLEAN: True, False.
- INTEGER: 32-bitowe liczby całkowite (w zakresie -2,147,483,647 do +2,147,483,647).
- NATURAL: Nieujemne liczby całkowite (0 do +2,147,483,647).
- REAL: Liczby rzeczywiste w zakresie od 1.0E38 do +1.0E38. Nie syntezowalne.
- Literały fizyczne: używane do oznaczania wielkości fizycznych jak czas, napiecie itd. Przydatne w symulacjach. Nie syntezowalne.
- Literały znakowe: pojedyncze znaki ASCII lub łańcuchy takich znaków. Nie syntezowalne.
- SIGNED i UNSIGNED: typy danych zdefiniowane w pakiecie ieee.std_logic_arith. Są podobne do STD_LOGIC_VECTOR, ale umozliwiają użycie operacji arytmetycznych, typowych dla danych całkowitych.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykłady stosowania typów

```
x0 <= '0';           -- bit, std_logic, or std_ulogic value '0'
x1 <= "00011111";    -- bit_vector, std_logic_vector,
                     -- std_ulogic_vector, signed, or unsigned
x3 <= "101111";      -- binary representation of decimal 47
x4 <= B"101111";     -- binary representation of decimal 47
x5 <= O"57";         -- octal representation of decimal 47
x6 <= X"2F";         -- hexadecimal representation of decimal 47
n <= 1200;           -- integer
m <= 1_200;          -- integer, underscore allowed
o <= 2#10011110#;    -- base 2 (binary) integer
IF ready THEN...     -- Boolean, executed if ready=TRUE
y <= 1.2E-5;         -- real, not synthesizable
q <= d after 10 ns; -- physical, not synthesizable
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Legalne i nielegalne operacje na danych różnych typów

```
SIGNAL a: BIT;
SIGNAL b: BIT_VECTOR(7 DOWNTO 0);
SIGNAL c: STD_LOGIC;
SIGNAL d: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL e: INTEGER RANGE 0 TO 255;
...
a <= b(5);    -- legal (same scalar type: BIT)
b(0) <= a;    -- legal (same scalar type: BIT)
c <= d(5);    -- legal (same scalar type: STD_LOGIC)
d(0) <= c;    -- legal (same scalar type: STD_LOGIC)
a <= c;       -- illegal (type mismatch: BIT x STD_LOGIC)
b <= d;       -- illegal (type mismatch: BIT_VECTOR x
              -- STD_LOGIC_VECTOR)
e <= b;       -- illegal (type mismatch: INTEGER x BIT_VECTOR)
e <= d;       -- illegal (type mismatch: INTEGER x
              -- STD_LOGIC_VECTOR)
```

Typy danych zdefiniowane przez użytkownika

- Typy całkowite:

```
TYPE integer IS RANGE -2147483647 TO +2147483647;
-- This is indeed the pre-defined type INTEGER.
TYPE natural IS RANGE 0 TO +2147483647;
-- This is indeed the pre-defined type NATURAL.
TYPE my_integer IS RANGE -32 TO 32;
-- A user-defined subset of integers.
TYPE student_grade IS RANGE 0 TO 100;
-- A user-defined subset of integers or naturals.
```

- Typy wyliczeniowe

```
TYPE bit IS ('0', '1');
-- This is indeed the pre-defined type BIT
TYPE my_logic IS ('0', '1', 'Z');
-- A user-defined subset of std_logic.
TYPE bit_vector IS ARRAY (NATURAL RANGE <>) OF BIT;
-- This is indeed the pre-defined type BIT_VECTOR.
-- RANGE <> is used to indicate that the range is unconstrained.
-- NATURAL RANGE <>, on the other hand, indicates that the only
-- restriction is that the range must fall within the NATURAL
-- range.
TYPE state IS (idle, forward, backward, stop);
-- An enumerated data type, typical of finite state machines.
TYPE color IS (red, green, blue, white);
-- Another enumerated data type.
```

Podtypy (SUBTYPE)

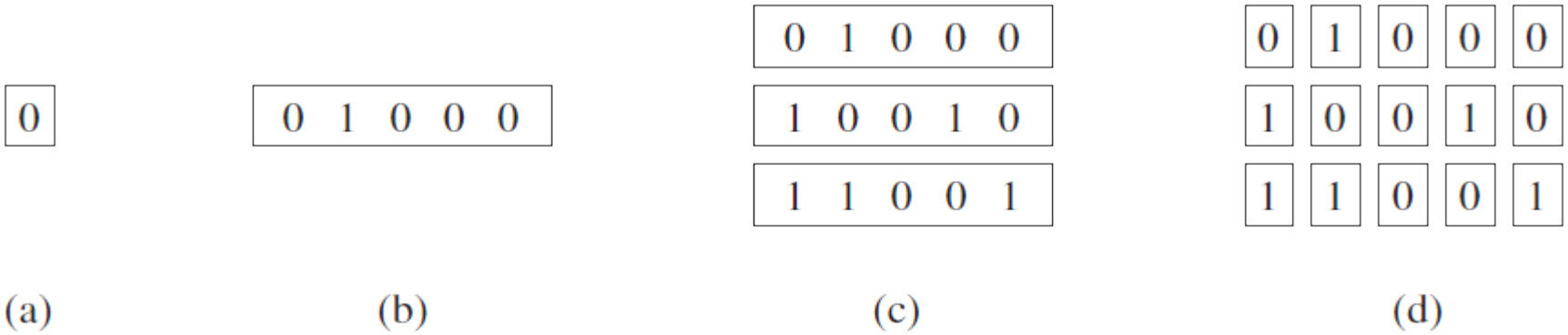
- Podtyp to typ z ograniczonym zakresem
- Dozwolone są operacje między podtypem i jego typem podstawowym

```
SUBTYPE natural IS INTEGER RANGE 0 TO INTEGER'HIGH;
-- As expected, NATURAL is a subtype (subset) of INTEGER.
SUBTYPE my_logic IS STD_LOGIC RANGE '0' TO 'Z';
-- Recall that STD_LOGIC=('X','0','1','Z','W','L','H','-').
-- Therefore, my_logic=('0','1','Z').
SUBTYPE my_color IS color RANGE red TO blue;
-- Since color=(red, green, blue, white), then
-- my_color=(red, green, blue).
SUBTYPE small_integer IS INTEGER RANGE -32 TO 32;
-- A subtype of INTEGER.

SUBTYPE my_logic IS STD_LOGIC RANGE '0' TO '1';
SIGNAL a: BIT;
SIGNAL b: STD_LOGIC;
SIGNAL c: my_logic;
...
b <= a; --illegal (type mismatch: BIT versus STD_LOGIC)
b <= c; --legal (same "base" type: STD_LOGIC)
```

Tablice

- Tablice to kolekcje obiektów tego samego typu
- Mogą być jednowymiarowe (1D) - (b) poniżej, dwuwymiarowe (2D) - (d) lub jednowymiarowe tablic jednowymiarowych (1Dx1D) - (c)



Tablice

- Mogą mieć więcej rozmiarów, ale nie są one wówczas przeważnie syntezowalne
- Predefiniowane typy VHDL obejmują jedynie typy skalarne (pojedynczy bit) i wektorowe (jednowymiarowa tablica bitów)
  - BIT, STD_LOGIC, STD_ULOGIC i BOOLEAN
  - BIT_VECTOR, STD_LOGIC_VECTOR, STD_ULOGIC_VECTOR, INTEGER, SIGNED i UNSIGNED
- Brak predefiniowanych tablic 2D i 1Dx1D, które w razie potrzeby muszą być wyspecyfikowane przez użytkownika, np.:

```
TYPE type_name IS ARRAY (specification) OF data_type;
```

- Tablic można użyć w następujący sposób:

```
SIGNAL signal_name: type_name [:= initial_value];
```

- W powyższy sposób można również zadeklarować CONSTANT lub VARIABLE

Przykład: tablice

- Zdefiniować tablicę zawierającą cztery wektory, zawierające 8 bitów każdy

```
TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC; -- 1D array
TYPE matrix IS ARRAY (0 TO 3) OF row;        -- 1Dx1D array
SIGNAL x: matrix;                            -- 1Dx1D signal
```

- Inna możliwość osiągnięcia tego samego efektu:

```
TYPE matrix IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
```

- Tablica dwuwymiarowa:

```
TYPE matrix2D IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;
-- 2D array
```

- Inicjalizacja:

```
... :="0001";                                  -- for 1D array
... :=('0','0','0','1')                       -- for 1D array
... :=(('0','1','1','1'), ('1','1','1','0')); -- for 1Dx1D or
                                               -- 2D array
```

Legalne i nielegalne przypisania tablic

```
TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC;
-- 1D array
TYPE array1 IS ARRAY (0 TO 3) OF row;
-- 1Dx1D array
TYPE array2 IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
-- 1Dx1D
TYPE array3 IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;
-- 2D array
SIGNAL x: row;
SIGNAL y: array1;
SIGNAL v: array2;
SIGNAL w: array3;
--------- Legal scalar assignments: --------------
-- The scalar (single bit) assignments below are all legal,
-- because the "base" (scalar) type is STD_LOGIC for all signals
-- (x,y,v,w).
x(0) <= y(1)(2); -- notice two pairs of parenthesis
                 -- (y is 1Dx1D)
x(1) <= v(2)(3); -- two pairs of parenthesis (v is 1Dx1D)
x(2) <= w(2,1);  -- a single pair of parenthesis (w is 2D)
y(1)(1) <= x(6);
y(2)(0) <= v(0)(0);
y(0)(0) <= w(3,3);
w(1,1) <= x(7);
w(3,0) <= v(0)(3);
```

Legalne i nielegalne przypisania tablic

```
--------- Vector assignments: ---------------------
x <= y(0);                                -- legal (same data types: ROW)
x <= v(1);                                -- illegal (type mismatch: ROW x
                                          -- STD_LOGIC_VECTOR)
x <= w(2);                                -- illegal (w must have 2D index)
x <= w(2, 2 DOWNTO 0);                    -- illegal (type mismatch: ROW x
                                          -- STD_LOGIC)
v(0) <= w(2, 2 DOWNTO 0);                 -- illegal (mismatch: STD_LOGIC_VECTOR
                                          -- x STD_LOGIC)
v(0) <= w(2);                             -- illegal (w must have 2D index)
y(1) <= v(3);                             -- illegal (type mismatch: ROW x
                                          -- STD_LOGIC_VECTOR)
y(1)(7 DOWNTO 3) <= x(4 DOWNTO 0);        -- legal (same type,
                                          -- same size)
v(1)(7 DOWNTO 3) <= v(2)(4 DOWNTO 0);     -- legal (same type,
                                          -- same size)
w(1, 5 DOWNTO 1) <= v(2)(4 DOWNTO 0);     -- illegal (type mismatch)
```

Rekordy

- Rekordy są podobne do tablic, ale zawierają obiekty różnych typów

```
TYPE birthday IS RECORD
  day: INTEGER RANGE 1 TO 31;
  month: month_name;
END RECORD;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Typy SIGNED i UNSIGNED

- Zdefiniowane w ieee.std_logic_arith oraz ieee.numeric_std

```
SIGNAL x: SIGNED (7 DOWNTO 0);
SIGNAL y: UNSIGNED (0 TO 3);
```

- Składnia typów zbliżona do STD_LOGIC_VECTOR
- Typ UNSIGNED reprezentuje liczbę w naturalnym kodzie binarnym
- Typ SIGNED reprezentuje liczbę w kodzie uzupełnień do dwóch
- Typy przeznaczone do operacji arytmetycznych

Typy SIGNED i UNSIGNED

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all; -- extra package necessary
...
SIGNAL a: IN SIGNED (7 DOWNTO 0);
SIGNAL b: IN SIGNED (7 DOWNTO 0);
SIGNAL x: OUT SIGNED (7 DOWNTO 0);
...
v <= a + b;   -- legal (arithmetic operation OK)
w <= a AND b; -- illegal (logical operation not OK)

LIBRARY ieee;
USE ieee.std_logic_1164.all; -- no extra package required
...
SIGNAL a: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL b: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL x: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
v <= a + b;   -- illegal (arithmetic operation not OK)
w <= a AND b; -- legal (logical operation OK)
```

Operacje arytmetyczne na STD_LOGIC_VECTOR

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all; -- extra package included
...
SIGNAL a: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL b: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL x: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
v <= a + b;   -- legal (arithmetic operation OK), unsigned
w <= a AND b; -- legal (logical operation OK)
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Konwersja danych

- VHDL nie pozwala na bezpośrednie operacje na danych różnych typów
- W tym celu potrzebna jest jawna konwersja danych
- ieee.std_logic_1164 dostarcza funkcje konwersji między danymi pokrewnych typów
- Elementy std_logic_vector, signed i unsigned do std_logic - nie trzeba konwersji

```
TYPE long IS INTEGER RANGE -100 TO 100;
TYPE short IS INTEGER RANGE -10 TO 10;
SIGNAL x : short;
SIGNAL y : long;
...
y <= 2*x + 5;       -- error, type mismatch
y <= long(2*x + 5); -- OK, result converted into type long

SIGNAL uv  : UNSIGNED(31 downto 0);
SIGNAL sv  : SIGNED(31 downto 0);
SIGNAL slv : STD_LOGIC_VECTOR(31 downto 0);

uv(0) = slv(0);                 -- legal
uv <=  slv;                     -- error - type mismatch
uv <=  UNSIGNED(slv);           -- legal
sv <=  SIGNED(slv);             -- legal
slv <= STD_LOGIC_VECTOR(uv);    -- legal
```

Konwersja danych z uzyciem std_logic_arith

- conv_integer(p) : Konwertuje parametr p typu UNSIGNED lub SIGNED do wartości typu INTEGER.
- conv_unsigned(p, b): Konwertuje parametr p typu INTEGER do wartości b-bitowej typu UNSIGNED.
- conv_signed(p, b): Konwertuje parametr p typu INTEGER do wartości b-bitowej typu SIGNED.
- conv_std_logic_vector(p, b): Konwertuje parametr p typu INTEGER do wartości b-bitowej typu STD_LOGIC_VECTOR.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Konwersja danych z uzyciem numeric_std

- to_integer(p): Konwertuje parametr p typu UNSIGNED lub SIGNED do wartości typu INTEGER.
- to_unsigned(p, b): Konwertuje parametr p typu INTEGER do wartości b-bitowej typu UNSIGNED.
- to_signed(p, b): Konwertuje parametr p typu INTEGER do wartości b-bitowej typu SIGNED.

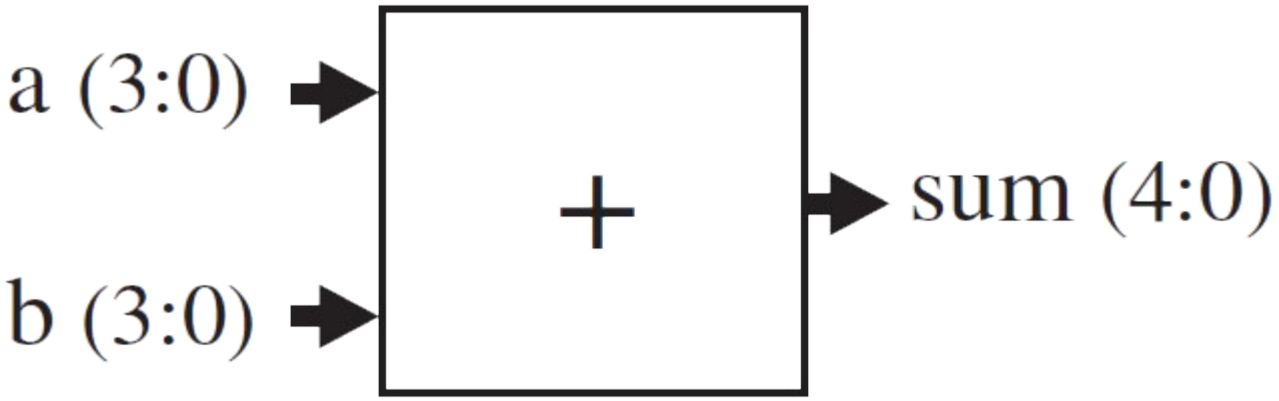
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Problem

- Zaprojektować sumator 4-bitowy:



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

„Układy reprogramowalne i SoC”
„Język VHDL (część 1)”

Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”

Prezentacja dystrybuowana jest bezpłatnie