جامعة نينوى
كلية هندسة الإلكترونيات

Circuit Design with VHDL 3

Textbook: Volnei A. Pedroni

Submitted By: Hussein Aideen

VHDL> Data Conversion

- VHDL does not allow direct operations between data of different types.
- it is necessary to convert data from one type to another.

- If the data are closely related: std_logic_1164 of the ieee library provides straightforward conversion functions.

```
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
```

VHDL> Data Conversion

- Data conversion functions: *std_logic_arith* package of the *ieee* library.

keyword	**Input data type**	**Output data type**
conv_integer(p)	INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC	INTEGER
conv_unsigned(p, b)	INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC	UNSIGNED * Where b is number of bits.
conv_signed(p, b):	INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC	SIGNED
conv_std_logic_vector(p, b)	INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC	STD_LOGIC_VECTOR

VHDL> Data Conversion

- Data conversion functions: std_logic_signed or std_logic_unsigned package of the *ieee* library.

keyword	**Input data type**	**Output data type**
unsigned(p)	STD_LOGIC_VECTOR	UNSIGNED
signed(p):	STD_LOGIC_VECTOR	SIGNED

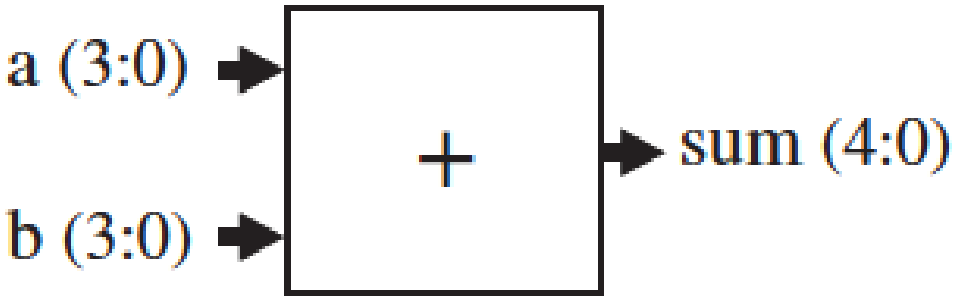
VHDL> Data Conversion

- Example:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
...
SIGNAL a: IN UNSIGNED (7 DOWNTO 0);
SIGNAL b: IN UNSIGNED (7 DOWNTO 0);
SIGNAL y: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
y <= CONV_STD_LOGIC_VECTOR ((a+b), 8);
-- Legal operation: a+b is converted from UNSIGNED to an
-- 8-bit STD_LOGIC_VECTOR value, then assigned to y.
```

VHDL> Examples:

- A 4-bit adder:



VHDL> Examples:

- A 4-bit adder:

```
----- Solution 1: in/out=SIGNED -----
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
-------------------------------------
ENTITY adder1 IS
PORT ( a, b : IN SIGNED (3 DOWNTO 0);
sum : OUT SIGNED (4 DOWNTO 0));
END adder1;
-------------------------------------
ARCHITECTURE adder1 OF adder1 IS
BEGIN
sum <= a + b;
END adder1;
-------------------------------------
```

```
----- Solution 2: out=INTEGER -----
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
-------------------------------------
ENTITY adder2 IS
PORT ( a, b : IN SIGNED (3 DOWNTO 0
sum : OUT INTEGER RANGE -16 TO 15);
END adder2;
-------------------------------------
ARCHITECTURE adder2 OF adder2 IS
BEGIN
sum <= CONV_INTEGER(a + b);
END adder2;
-------------------------------------
```

VHDL>Static and non-static data

- CONSTANT:
  - establish default values
  - can be declared in a PACKAGE, ENTITY, or ARCHITECTURE.

```
CONSTANT name : type := value;
```

```
CONSTANT set_bit : BIT := '1';
CONSTANT datamemory : memory := (('0','0','0','0'),
                                 ('0','0','0','1'),
                                 ('0','0','1','1'));
```

VHDL>Static and non-static data

- GENERIC:
  - specifying a generic parameter (that is, a static parameter).
  - code more flexibility and reusability.
  - must be declared in the ENTITY.

```
GENERIC (parameter_name : parameter_type := parameter_value);
```

```
ENTITY adder2 IS
GENERIC (n : INTEGER := 8);
PORT ( a, b : IN SIGNED (3 D
sum : OUT INTEGER RANGE -16
```

VHDL>Static and non-static data

- SIGNAL:
  - pass values in and out the circuit, as well as between its internal units.
  - circuit interconnects (wires).

```
SIGNAL name : type [range] [:= initial_value];
```

```
SIGNAL control: BIT := '0';
SIGNAL count: INTEGER RANGE 0 TO 100;
SIGNAL y: STD_LOGIC_VECTOR (7 DOWNTO 0);
```

VHDL>Static and non-static data

- VARIABLE:
  - represents only local information.
  - It can only be used inside a sequential code (PROCESS for example).

```
VARIABLE name : type [range] [:= init_value];
```

```
variable control: BIT := '0';
variable count: INTEGER RANGE 0 TO 100;
variable y: STD_LOGIC_VECTOR (7 DOWNTO 0);
```

VHDL>Static and non-static data

Table 7.1
Comparison between SIGNAL and VARIABLE.

	SIGNAL	VARIABLE
Assignment	<=	:=
Utility	Represents circuit interconnects (wires)	Represents local information
Scope	Can be global (seen by entire code)	Local (visible only inside the corresponding PROCESS, FUNCTION, or PROCEDURE)
Behavior	Update is not immediate in sequential code (new value generally only available at the conclusion of the PROCESS, FUNCTION, or PROCEDURE)	Updated immediately (new value can be used in the next line of code)
Usage	In a PACKAGE, ENTITY, or ARCHITECTURE. In an ENTITY, all PORTS are SIGNALS by default	Only in sequential code, that is, in a PROCESS, FUNCTION, or PROCEDURE

VHDL> Operators

- VHDL provides several kinds of pre-defined operators:
  - Assignment operators
  - Logical operators
  - Arithmetic operators
  - Relational operators
  - Shift operators
  - Concatenation operators

VHDL> Operators

- Assignment operators

Operator	using
<=	SIGNAL.
:=	VARIABLE, CONSTANT, GENERIC, initial values.
=>	vector elements or with OTHERS.

VHDL> Operators

- Assignment operators

```
SIGNAL x : STD_LOGIC;
VARIABLE y : STD_LOGIC_VECTOR(3 DOWNTO 0); -- Leftmost bit is MSB
SIGNAL w: STD_LOGIC_VECTOR(0 TO 7); -- Rightmost bit is -- MSB
-------------------------------------------------------------
x <= '1'; -- '1' is assigned to SIGNAL x using "<="
y := "0000"; -- "0000" is assigned to VARIABLE y using ":="
w <= "10000000"; -- LSB is '1', the others are '0'
w <= (0 =>'1', OTHERS =>'0'); -- LSB is '1', the others are '0'
```