Arithmetic in VHDL

Operators available in VHDL-93 (without including any package)

Operator	Description	Type of *a*	Type of *b*	Type of result
a ** b	Exponentiation	integer	integer	integer
abs a	Absolute value	integer		integer
not a	Negation	boolean, bit, bit_vector		boolean, bit, bit_vector
a * b	Multiplication	integer	integer	integer
a / b	Division			
a **mod** b	Modulo			
a **rem** b	Remainder			
+ a	Identity	integer		integer
- a	Negation			
a + b	Addition	integer	integer	integer
a - b	Subtraction			
a & b	Concatenation	Array 1-D, elements	Array 1-D, elements	Array 1-D
a = b	Equals	Any	Same as a	boolean
a /= b	Not equals			

Integer: $-(2^{31}-1)$ to $+(2^{31}-1)$ **Natural:** 0 to $+(2^{31}-1)$

Chu, Pong P. "RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability"

Operators available in VHDL-93 (without including any package)

Operator	Description	Type of *a*	Type of *b*	Type of result
a **sll** b	Logic left shift	bit_vector	integer	bit_vector
a **srl** b	Logic right shift			
a **sla** b	Arith. left shift			
a **sra** b	Arith. right shift			
a **rol** b	Left rotation			
a **ror** b	Right rotation			
a < b	Less than	Scalar or 1-D array	Same as a	boolean
a <= b	Less or equals than			
a > b	Grater than			
a >= b	Greater or equals than			
a **and** b	and	boolean, bit, bit_vector	Same as a	Same as a
a **or** b	or			
a **xor** b	xor			
a **nand** b	nand			
a **nor** b	nor			
a **xnor** b	xnor			

Chu, Pong P. "RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability"

Operators and functions in the package IEEE *std_logic_1164*

Operator	Type of *a*	Type of *b*	Type of result
not a	std_logic_vector std_logic	-	Same as op. a
a **and** b a **or** b a **xor** b a **nand** b a **nor** b a **xnor** b	std_logic_vector std_logic	Same as op. a	Same as op. a

Function	Type of *a*	Type of result
to_bit(a)	std_logic	bit
to_stdulogic(a)	bit	std_logic
to_bitvector(a)	std_logic_vector	bit_vector
to_stdlogicvector(a)	bit_vector	std_logic_vector

Arithmetic in VHDL

- Xilinx has a non standard library for signed/unsigned arithmetic:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
-- use ieee.std_logic_signed.all;
```

Choose only one of these two packages

```
entity adder4b is
    Port( A, B : in  STD_LOGIC_VECTOR (3 downto 0);
          C    : out STD_LOGIC_VECTOR (3 downto 0) );
end adder4b;

architecture arch_adder4b of adder4b is
begin
    C <= A + B;
end arch_adder4b;
```

Arithmetic in VHDL

- However, these packages are not IEEE-compliant. They were created in the 80's by Synopsys and included "de facto" by most of the synthesis tools at that time.
- Some years later, IEEE promoted the following standardized package, which is recommended since then:

```
library ieee;
use ieee.numeric_std.all;
```

- The package **`ieee.numeric_std.all`** is recommended because 100% compatibility is guaranteed for any synthesis tool.

Arithmetic in VHDL

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

entity adder is
  port(a:   in  std_logic_vector(7 downto 0);
       b:   in  std_logic_vector(7 downto 0);
       ci:  in  std_logic;
       sum: out std_logic_vector(7 downto 0);
       co:  out std_logic);
end adder;

architecture rtl of adder is
  signal ci_i: unsigned(0 downto 0);
  signal a_i, b_i, sum_i: unsigned(8 downto 0);
begin
  a_i <= unsigned("0" & a);
  b_i <= unsigned("0" & b);
  ci_i <= "" & ci;
  sum_i <= a_i + b_i + ci_i;

  sum <= std_logic_vector(sum_i(7 downto 0));
  co <= sum_i(8);
end rtl;
```

- **`numeric_std`** forces to give an individual interpretation for each number (either signed or unsigned).
- This example uses **`unsigned`** numbers. However, **`signed`** numbers can be also used in this code at the same time.
- With **`ieee.std_logic_arith`** package, it is not possible to use both types in the same .vhd code.
- **`numeric_std`** does not allow the operation a+b on **`std_logic_vector`**'s.

This converts from a simple type to a vector type. 4 types of vectors exist in VHDL: bit_vector, std_logic_vector, unsigned and signed.

```
ci_i <= unsigned'("" & ci);
```

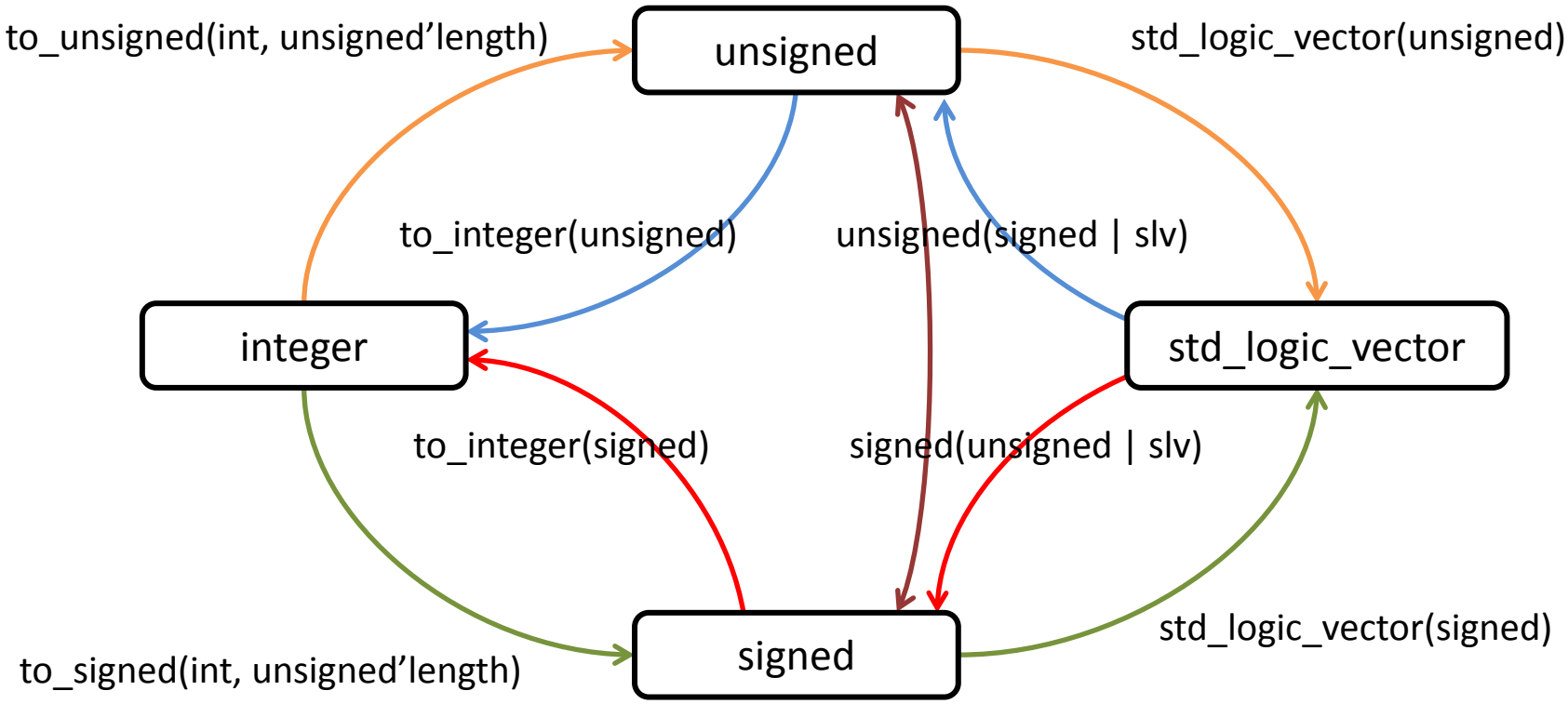
Operators in the package IEEE *numeric_std*

Operator	Description	Type of *a*	Type of *b*	Type of result
abs a - a	Absol. value Negation	signed	-	signed
a * b a / b a **mod** b a **rem** b a + b a - b	Arithmetic operations	unsigned, unsigned, natural signed, signed, integer	unsigned, natural unsigned signed, integer signed	unsigned unsigned signed signed
a = b a /= b a < b a <= b a > b a >= b	Relational operations	unsigned, unsigned, natural signed, signed, integer	unsigned, natural unsigned signed, integer signed	boolean boolean boolean boolean

Chu, Pong P. "RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability"

Arithmetic in VHDL

- In order to convert signals from one type to another, the following functions can be used:



Casting / conversion functions

Data type (FROM)	Data type (TO)	Conversion function / casting
unsigned, signed	std_logic_vector	std_logic_vector(a)
signed, std_logic_vector	unsigned	unsigned(a)
unsigned, std_logic_vector	signed	signed(a)
unsigned, signed	integer	to_integer(a)
natural	unsigned	to_unsigned(a, size)
integer	signed	to_signed(a, size)

Chu, Pong P. “RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability”