Résumé des packages

VHDL standards

NE PAS UTILISER LES LIBRAIRIES BARRÉES → PAS STANDARD !
Indiquées comme référence seulement

R. Beuchat
[0.9]

package std.standard

Librairies utilisées

library **std ;**
use **std.standard.all ;**

Utilisation

Package standard utilisé par défaut
Il définit les types de base utilisés en VHDL, 1992 Language Reference Manual.

Types

Boolean

```
    type boolean is (false,true);
```

Bit / Bit_vector

```
    type bit is ('0', '1');
    type bit_vector is array (natural range <>) of bit;
```

Character

```
    type character is (
          nul, soh, stx, etx, eot, enq, ack, bel,
          bs,  ht,  lf,  vt,  ff,  cr,  so,  si,
          dle, dc1, dc2, dc3, dc4, nak, syn, etb,
          can, em,  sub, esc, fsp, gsp, rsp, usp,

          ' ', '!', '"', '#', '$', '%', '&', ''',
          '(', ')', '*', '+', ',', '-', '.', '/',
          '0', '1', '2', '3', '4', '5', '6', '7',
          '8', '9', ':', ';', '<', '=', '>', '?',
          '@', 'A', 'B', 'C', 'D', 'E', 'F', 'G',
          'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O',
          'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
          'X', 'Y', 'Z', '[', '\', ']', '^', '_',
          '`', 'a', 'b', 'c', 'd', 'e', 'f', 'g',
          'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
          'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
          'x', 'y', 'z', '{', '|', '}', '~', del,

          c128, c129, c130, c131, c132, c133, c134, c135,
          c136, c137, c138, c139, c140, c141, c142, c143,
          c144, c145, c146, c147, c148, c149, c150, c151,
          c152, c153, c154, c155, c156, c157, c158, c159,

          -- the character code for 160 is there (NBSP), but prints as no char

          ' ', '¡', '¢', '£', '¤', '¥', '¦', '§',
          '¨', '©', 'ª', '«', '¬', '-', '®', '¯',
          '°', '±', '²', '³', '´', 'µ', '¶', '·',
          '¸', '¹', 'º', '»', '¼', '½', '¾', '¿',

          'À', 'Á', 'Â', 'Ã', 'Ä', 'Å', 'Æ', 'Ç',
          'È', 'É', 'Ê', 'Ë', 'Ì', 'Í', 'Î', 'Ï',
          'Ð', 'Ñ', 'Ò', 'Ó', 'Ô', 'Õ', 'Ö', '×',
          'Ø', 'Ù', 'Ú', 'Û', 'Ü', 'Ý', 'Þ', 'ß',

          'à', 'á', 'â', 'ã', 'ä', 'å', 'æ', 'ç',
          'è', 'é', 'ê', 'ë', 'ì', 'í', 'î', 'ï',
          'ð', 'ñ', 'ò', 'ó', 'ô', 'õ', 'ö', '÷',
          'ø', 'ù', 'ú', 'û', 'ü', 'ý', 'þ', 'ÿ' );
```

Severity_level

```
type severity_level is (note, warning, error, failure);
```

Integer / natural / positive

```
type integer is range -2147483648 to 2147483647;
subtype natural is integer range 0 to integer'high;
subtype positive is integer range 1 to integer'high;
```

Real

```
type real is range -1.0E308 to 1.0E308;
```

Time / delay_length

```
type time is range -2147483647 to 2147483647
     units
           fs;
           ps = 1000 fs;
           ns = 1000 ps;
           us = 1000 ns;
           ms = 1000 us;
           sec = 1000 ms;
           min = 60 sec;
           hr = 60 min;
     end units;
```

```
subtype delay_length is time range 0 fs to time'high;

impure function now return delay_length;
```

String

```
type string is array (positive range <>) of character;
```

File

```
type file_open_kind is (
     read_mode,
     write_mode,
     append_mode);

type file_open_status is (
     open_ok,
     status_error,
     name_error,
     mode_error);

attribute foreign : string;
```

package ieee.STD_LOGIC_1164

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

Utilisation

- Système logique à 9 états standardisé par l'IEEE
- Fichier défini initialement pour la simulation uniquement, actuellement utilisable pour la synthèse aussi
- Ce paquetage définit les **fonctions loqiques (and, nand, or, nor, xor, (*xnor*), not**) pour : (note : Xnor n'est pas encore accepté par le standard)

- **std_ulogic**
- **std_logic_vector**
- **std_ulogic_vector**

std_logic_1164 type définitions

```
TYPE std_ulogic IS (
        'U', -- Uninitialized
        'X', -- Forcing Unknown
        '0', -- Forcing 0
        '1', -- Forcing 1
        'Z', -- High Impedance
        'W', -- Weak Unknown
        'L', -- Weak 0
        'H', -- Weak 1
        '-' -- Don't care
        );
```

Directives pour la synthèse Exemplar de std_ulogic

- Déclare l'attribut d'encodage de type et indique la valeur pour le type **std_ulogic**

```
ATTRIBUTE logic_type_encoding : string ;
ATTRIBUTE logic_type_encoding of std_ulogic:type is
--    ('U','X','0','1','Z','W','L','H','-')
      ('X','X','0','1','Z','X','0','1','X') ;
```

unconstrained array of std_ulogic for use with the resolution function

```
TYPE std_ulogic_vector IS ARRAY ( NATURAL RANGE <> ) OF std_ulogic;
```

fonction de resolution

```
FUNCTION resolved ( s : std_ulogic_vector ) RETURN std_ulogic;
```

type logique standard

```
SUBTYPE std_logic IS resolved std_ulogic;
TYPE std_logic_vector IS ARRAY ( NATURAL RANGE <>) OF std_logic;
```

subtypes communs

```
SUBTYPE X01 IS resolved std_ulogic RANGE 'X' TO '1'; -- ('X','0','1')
SUBTYPE X01Z IS resolved std_ulogic RANGE 'X' TO 'Z'; -- ('X','0','1','Z')
SUBTYPE UX01 IS resolved std_ulogic RANGE 'U' TO '1'; -- ('U','X','0','1')
SUBTYPE UX01Z IS resolved std_ulogic RANGE 'U' TO 'Z'; -- ('U','X','0','1','Z')
```

Fonctions logiques (and, nand, …)

Opérations	Type IN left	Type IN right	RETURN
And Nand Or Nor Xor *Xnor*	std_ulogic	std_ulogic	UX01
	std_ulogic_vector	std_ulogic_vector	std_ulogic_vector
	std_logic_vector	std_logic_vector	std_logic_vector
Not	std_ulogic		UX01
	std_ulogic_vector		std_ulogic_vector
	std_logic_vector		std_logic_vector

Xnor pas encore standardisé !

```
FUNCTION "and"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "and"  ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;
FUNCTION "and"  ( l, r : std_logic_vector  ) RETURN std_logic_vector;
```

Conversion de type

Function	In	Param	Return
To_bit	Std_ulogic	xmap : BIT := '0'	Bit
To_bitvector	Std_logic_vector	xmap : BIT := '0'	Bit_vector
	Std_ulogic_vector		
To_StdULogic	Bit		Std_ulogic
To_StdLogicVector	Bit_vector		Std_logic_vector
	Std_ulogic_vector		
To_StdULogicVector	Bit_ vector		Std_ulogic_vector
	Std_logic_vector		

Xmap → valeur par défaut

```
FUNCTION To_bit ( s : std_ulogic; xmap : BIT := '0') RETURN BIT;
FUNCTION  To_bitvector  (  s  :  std_logic_vector  ;  xmap  :  BIT  :=  '0')  RETURN
   BIT_VECTOR;
FUNCTION  To_bitvector  (  s  :  std_ulogic_vector;  xmap  :  BIT  :=  '0')  RETURN
   BIT_VECTOR;
FUNCTION To_StdULogic ( b : BIT ) RETURN std_ulogic;
FUNCTION To_StdLogicVector ( b : BIT_VECTOR ) RETURN std_logic_vector;
FUNCTION To_StdLogicVector ( s : std_ulogic_vector ) RETURN std_logic_vector;
FUNCTION To_StdULogicVector ( b : BIT_VECTOR ) RETURN std_ulogic_vector;
FUNCTION To_StdULogicVector ( s : std_logic_vector ) RETURN std_ulogic_vector;
```

Conversion de type et de force

Function	In	Return
To_X01 To_X01Z To_UX01	Std_logic_vector	Std_logic_vector
	Std_ulogic_vector	Std_ulogic_vector
	Bit_vector	Std_logic_vector
		Std_ulogic_vector
To_X01	Std_ulogic Bit	X01
To_X01Z		X01Z
To_UX01		UX01

FUNCTION **To_X01** (s : ***std_logic_vector***) RETURN std_logic_vector;
FUNCTION **To_X01** (s : ***std_ulogic_vector***) RETURN std_ulogic_vector;
FUNCTION **To_X01** (s : ***std_ulogic***) RETURN X01;
FUNCTION **To_X01** (b : ***BIT_VECTOR***) RETURN std_logic_vector;
FUNCTION **To_X01** (b : ***BIT_VECTOR***) RETURN std_ulogic_vector;
FUNCTION **To_X01** (b : ***BIT***) RETURN X01;

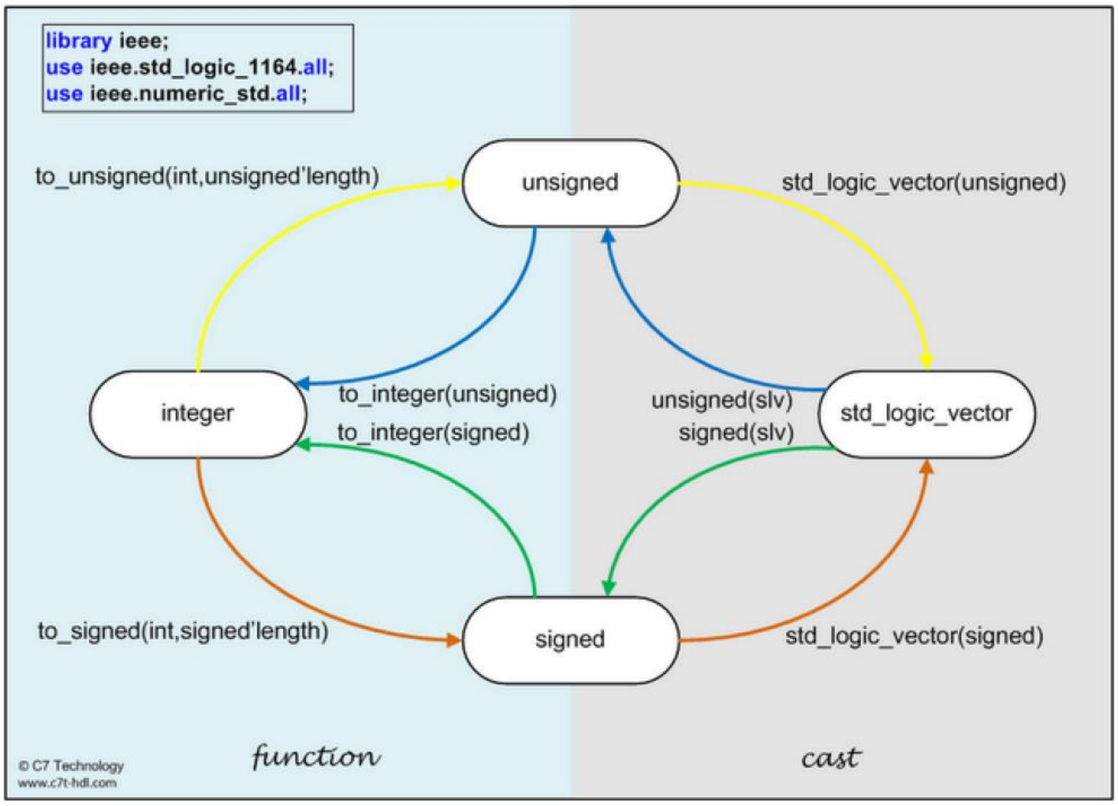
FUNCTION **To_X01Z** (s : **std_logic_vector**) RETURN std_logic_vector;
FUNCTION **To_X01Z** (s : **std_ulogic_vector**) RETURN std_ulogic_vector;
FUNCTION **To_X01Z** (s : ***std_ulogic***) RETURN X01Z;
FUNCTION **To_X01Z** (b : ***BIT_VECTOR***) RETURN std_logic_vector;
FUNCTION **To_X01Z** (b : ***BIT_VECTOR***) RETURN std_ulogic_vector;
FUNCTION **To_X01Z** (b : ***BIT***) RETURN X01Z;

FUNCTION **To_UX01** (s : ***std_logic_vector***) RETURN std_logic_vector;
FUNCTION **To_UX01** (s : ***std_ulogic_vector***) RETURN std_ulogic_vector;
FUNCTION **To_UX01** (s : ***std_ulogic***) RETURN UX01;
FUNCTION **To_UX01** (b : ***BIT_VECTOR***) RETURN std_logic_vector;
FUNCTION **To_UX01** (b : ***BIT_VECTOR***) RETURN std_ulogic_vector;
FUNCTION **To_UX01** (b : ***BIT***) RETURN UX01;

-- edge detection

FUNCTION **rising_edge** (s : **std_ulogic**) RETURN BOOLEAN;
FUNCTION **falling_edge** (s : **std_ulogic**) RETURN BOOLEAN;

FUNCTION **Is_X** (s : **std_ulogic_vector**) RETURN BOOLEAN;
FUNCTION **Is_X** (s : **std_logic_vector**) RETURN BOOLEAN;
FUNCTION **Is_X** (s : **std_ulogic**) RETURN BOOLEAN;



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

package ieee.NUMERIC_STD
package ieee.NUMERIC_BIT

Libraries used

library **IEEE**;
use **IEEE.STD_LOGIC_1164**.all;

use **IEEE.NUMERIC_STD**.all;

OR

use **IEEE.NUMERIC_BIT**.all;

Numeric array type definitions

For package ieee.NUMERIC_STD:

type ***UNSIGNED*** is array (NATURAL range <>) of **STD_LOGIC**;
type ***SIGNED*** is array (NATURAL range <>) of **STD_LOGIC**

For package ieee.NUMERIC_BIT:

type ***UNSIGNED*** is array (NATURAL range <>) of **BIT**;
type ***SIGNED*** is array (NATURAL range <>) of **BIT**;

Utilisation

Those package defined types, numerical and logical functions for synthesis and simulation tools. Two numerical types are defined:

- **UNSIGNED**: represent an unsigned number as a vector of std_logic or bit
- **SIGNED**: represent a signed number as a vector of std_logic or bit

This package has some conversion function and clk edge detection

Type conversions

Function	In	Param	Return
TO_INTEGER	Unsigned		Natural
	Signed		Integer
TO_UNSIGNED	Natural	Size: natural	Unsigned
TO_SIGNED	Integer	Size: natural	Signed
RESIZE	Signed	Size: natural	Signed
	Unsigned		Unsigned

size : largeur du vecteur de retour

RESIZE Functions

```
function RESIZE (ARG: SIGNED; NEW_SIZE: NATURAL) return SIGNED;
-- Result subtype: SIGNED(NEW_SIZE-1 downto 0)
-- Result: Resizes the SIGNED vector ARG to the specified size.
-- To create a larger vector, the new [leftmost] bit positions are filled with the sign
   bit (ARG'LEFT). When truncating, the sign bit is retained along with the rightmost
   part.
function RESIZE (ARG: UNSIGNED; NEW_SIZE: NATURAL) return UNSIGNED;
-- Result subtype: UNSIGNED(NEW_SIZE-1 downto 0)
-- Result: Resizes the UNSIGNED vector ARG to the specified size.
-- To create a larger vector, the new [leftmost] bit positions are filled with '0'. When
   truncating, the leftmost bits are dropped.
```

Conversion Functions

```
function TO_INTEGER (ARG: UNSIGNED) return NATURAL;
-- Result subtype: NATURAL. Value cannot be negative since parameter is an UNSIGNED vector.
-- Result: Converts the UNSIGNED vector to an INTEGER.
function TO_INTEGER (ARG: SIGNED) return INTEGER;
-- Result: Converts a SIGNED vector to an INTEGER.
function TO_UNSIGNED (ARG, SIZE: NATURAL) return UNSIGNED;
-- Result subtype: UNSIGNED(SIZE-1 downto 0)
-- Result: Converts a non-negative INTEGER to an UNSIGNED vector with the specified size.
function TO_SIGNED (ARG: INTEGER; SIZE: NATURAL) return SIGNED;
-- Result subtype: SIGNED(SIZE-1 downto 0)
-- Result: Converts an INTEGER to a SIGNED vector of the specified size.
```

Unary Arithmetic (-, ABS)

Opérations	Type IN	RETURN
Abs **-**	Signed	Signed

Arithmétique (+, -, *, /, REM, MOD)

Opérations	Type IN left	Type IN right	RETURN
+	Unsigned	Unsigned	
-	Unsigned	Natural	Unsigned
*****	Natural	Unsigned	
/	Signed	Signed	
rem	Signed	Integer	Signed
mod	Integer	Signed	

Comparaison, Integer/Natural/Signed/Unsigned OF bit

Opérations	Type IN left	Type IN right	RETURN
< **<=** **>** **>=** **=** **/=**	Unsigned	Unsigned	Boolean
	Signed	Signed	
	Natural	Unsigned	
	Unsigned	Natural	
	Integer	Signed	
	Signed	Integer	

Décalage et rotation(SHIFT, ROTATE), Signed/Unsigned

Opérations	Argument	Count	RETURN
SHIFT_LEFT **SHIFT_RIGHT** **ROTATE_LEFT** **ROTATE_RIGHT**	Unsigned	Count: natural	Unsigned
	Signed		Signed

Shift and Rotate (sll, srl, rol, ror), non compatible VHDL 1076-1987

Opérations	Argument	Count	RETURN
Sll **Srl** **Rol** **Ror**	Unsigned	Count: integer	Unsigned
	Signed		Signed

Logic Functions (and, nand, …)

Opérations	Type IN left	Type IN right	RETURN
And **Nand** **Or** **Nor** **Xor** ***Xnor***	Unsigned	Unsigned	Unsigned
	Signed	Signed	Signed
Not	Unsigned		Unsigned
	Signed		Signed

XNOR not standardised

Clock edge detection (Numeric_Bit only, already available in IEEE.STD_LOGIC_1164 for std_logic)

```
function RISING_EDGE (signal S: BIT) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Returns TRUE if an event is detected on signal S and the
  --         value changed from a '0' to a '1'.

function FALLING_EDGE (signal S: BIT) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Returns TRUE if an event is detected on signal S and the
  --         value changed from a '1' to a '0'.
```

Translation Functions (Numeric_Std only)

Function	In	Param	Return
TO_01	Unsigned	Xmap: std_logic := '0'	Unsigned
	Signed		Signed

```
function TO_01 (S: UNSIGNED; XMAP: STD_LOGIC := '0') return UNSIGNED;
-- Result subtype: UNSIGNED(S'RANGE)
-- Result: Termwise, 'H' is translated to '1', and 'L' is translated to '0'. If
   a value other than '0'|'1'|'H'|'L' is found,
-- the array is set to (others => XMAP), and a warning is issued.

function TO_01 (S: SIGNED; XMAP: STD_LOGIC := '0') return SIGNED;
-- Result subtype: SIGNED(S'RANGE)
-- Result: Termwise, 'H' is translated to '1', and 'L' is translated to '0'. If
   a value other than '0'|'1'|'H'|'L' is found,
-- the array is set to (others => XMAP), and a warning is issued.
```

Type verification for compatibility (Numeric_Std only)

The STD_MATCH function are added for testing validity of '0' and '1', soit '0', '1', 'L' andt 'H', and parameters size that need to be the same and >0.

Opérations	Type IN left	Type IN right	RETURN
STD_MATCH	Std_ulogic	Std_ulogic	Boolean
	Unsigned	Unsigned	
	Signed	Signed	
	Std_logic_vector	Std_logic_vector	
	Std_ulogic_vector	Std_ulogic_vector	

```
function STD_MATCH (L, R: STD_ULOGIC) return BOOLEAN;
function STD_MATCH (L, R: UNSIGNED) return BOOLEAN;
function STD_MATCH (L, R: SIGNED) return BOOLEAN;
function STD_MATCH (L, R: STD_LOGIC_VECTOR) return BOOLEAN;
function STD_MATCH (L, R: STD_ULOGIC_VECTOR) return BOOLEAN;
```

~~package ieee.STD_LOGIC_ARITH~~
(Mentor Graphics)

Librairies utilisées

library **IEEE**;
use **IEEE.STD_LOGIC_1164**.all;
use **IEEE.STD_LOGIC_arith**.all; -- package from Mentor Graphics

type definitions

type **UNSIGNED** is array (NATURAL range <>) of STD_LOGIC;
type **SIGNED** is array (NATURAL range <>) of STD_LOGIC;
subtype **SMALL_INT** is INTEGER range 0 to 1;

FUNCTION ***std_ulogic_wired_or*** (input : std_ulogic_vector) RETURN std_ulogic;
-- a wired OR operation is performed on the inputs to determine the resolved value
FUNCTION ***std_ulogic_wired_and*** (input : std_ulogic_vector) RETURN std_ulogic;
-- a wired AND operation is performed on the inputs to determine the resolved value

Conversions de type, extensions

Function	In	Param	Return
To_Integer	Std_ulogic_vector	x : integer := 0	Integer
	Std_logic_vector		
	signed		
	unsigned		natural
	Std_logic		
To_StdUlogicVector	Integer	size : natural	Std_ulogic_vector
To_StdlogicVector			Std_logic_vector
To_Stdlogic	boolean		Std_logic
CONV_INTEGER	Std_ulogic_vector	x : integer := 0	Integer
	Std_logic_vector		
	signed		
	unsigned		natural
	Std_logic		
CONV_UNSIGNED ***To_unsigned***	natural	Size: natural	Unsigned
CONV_SIGNED	Integer	Size:natural	Signed
To_Signed			
Zero_extend	Std_ulogic_vector	Size: natural	Std_ulogic_vector
	Std_logic_vector		Std_logic_vector
	signed		signed
	unsigned		unsigned
	Std_logic		Std_logic

size : largeur du vecteur de retour

Arithmétique (+, -)

Opérations	Type IN left	Type IN right	RETURN
+ -	Std_ulogic_vector	Std_ulogic_vector	Std_ulogic_vector
	Std_logic_vector	Std_logic_vector	Std_logic_vector
	signed	signed	signed
	unsigned	unsigned	unsigned
	Std_logic	Std_logic	Std_logic

Arithmétique unaire (+, -, ABS)

Opérations	Type IN	RETURN
+	Std_ulogic_vector	Std_ulogic_vector
	Std_logic_vector	Std_logic_vector
	signed	signed
	unsigned	unsigned
-	Signed	Signed
ABS	Signed	Signed

Multiplication, Signed/Unsigned

Opérations	Type IN left	Type IN right	RETURN
*	Std_ulogic_vector	Std_ulogic_vector	Std_ulogic_vector
	Std_logic_vector	Std_logic_vector	Std_logic_vector
	signed	signed	signed
	unsigned	unsigned	unsigned

Vectorized Overloaded Arithmetic Operators, not supported for synthesis.
The following operators are not supported for synthesis.

Opérations	Type IN left	Type IN right	RETURN
/ **MOD** **REM** **	Std_ulogic_vector	Std_ulogic_vector	Std_ulogic_vector
	Std_logic_vector	Std_logic_vector	Std_logic_vector
	unsigned	unsigned	signed
/	signed	signed	unsigned

Comparison, Signed/Unsigned Overloaded

Opérations	Type IN left	Type IN right	RETURN
Eq Ne Lt Gt Le Ge	Std_logic	Std_logic	Boolean
	Std_ulogic_vector	Std_ulogic_vector	
	Std_logic_vector	Std_logic_vector	
Eq Ne Lt Gt Le Ge = /= < > <= >=	unsigned	Unsigned	Boolean
	Signed	Signed	

Décalage/rotation

Opérations	Argument	Count	RETURN
Sla Sra Sll Srl Rol Ror	Std_ulogic_vector	natural	Std_ulogic_vector
	Std_logic_vector	natural	Std_logic_vector
	signed	natural	signed
	unsigned	natural	unsigned

Logical functions

Opérations	Type IN left	Type IN right	RETURN
And Nand Or Nor Xor Not xnor	signed	signed	signed
	unsigned	unsigned	unsigned
xnor	Std_ulogic_vector	Std_ulogic_vector	Std_ulogic_vector
	Std_logic_vector	Std_logic_vector	Std_logic_vector

~~package ieee.STD_LOGIC_UNSIGNED~~ (Synopsys Inc.)

~~package ieee.STD_LOGIC_SIGNED~~ (Synopsys Inc.)

Utilisation

Ce package est utilisé pour la synthèse de type **SIGNED/UNSIGNED** avec Std_logic_vector.
définis dans **IEEE.std_logic_arith**.all; (Synopsys, Inc.)

Librairies utilisées

library **IEEE**;	use **IEEE.std_logic_unsigned**.all;
use **IEEE.std_logic_1164**.all;	*ou*
use **IEEE.std_logic_arith**.all;	use **IEEE.std_logic_signed**.all;

Arithmétique (+,-)

Opérations	Type IN left	Type IN right	RETURN
+ -	Std_logic_vector	Std_logic_vector	Std_logic_vector
	Std_logic_vector	Integer	
	Integer	Std_logic_vector	
	Std_logic_vector	Std_logic	
	Std_logic	Std_logic_vector	

```
function "+"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR
function "+"(L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
function "+"(L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
function "+"(L: STD_LOGIC_VECTOR; R: STD_LOGIC) return STD_LOGIC_VECTOR;
function "+"(L: STD_LOGIC; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
```

Arithmétique unaire (+) STD_LOGIC_VECTOR

Opérations	Type IN	RETURN
+	Std_logic_vector	Std_logic_vector

```
function "+"(L: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
```

Multiplication STD_LOGIC_VECTOR

Opérations	Type IN left	Type IN right	RETURN
*	Std_logic_vector	Std_logic_vector	Std_logic_vector

```
function "*"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
```

Comparaison, Integer/Signed/Unsigned

Opérations	Type IN left	Type IN right	RETURN
< <= > >= = /=	Std_logic_vector	Std_logic_vector	Boolean
	Std_logic_vector	Integer	
	Integer	Std_logic_vector	

```
function "<"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
function "<"(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
function "<"(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;
```

~~package ieee.NUMERIC_UNSIGNED~~

Utilisation

Ce package est utilisé pour la synthèse de type NUMERIC UNSIGNED de std_logic_vector.
Synopsys, Inc.

Librairies utilisées

library **IEEE**;
use **IEEE.STD_LOGIC_1164**.all;
use **IEEE.NUMERIC_STD**.all;

Fonctions de Conversion/ RESIZE

Function	In	Param	Return
TO_INTEGER	Std_logic_vector		Integer
RESIZE	Std_logic_vector	New_size: natural	Std_logic_vector

function ***TO_INTEGER*** (ARG: STD_LOGIC_VECTOR) return INTEGER;
function ***RESIZE*** (ARG: STD_LOGIC_VECTOR; NEW_SIZE: NATURAL) return STD_LOGIC_VECTOR;
-- Result subtype: STD_LOGIC_VECTOR(NEW_SIZE-1 downto 0)
-- Result: ReSIZEs the STD_LOGIC_VECTOR vector ARG to the specified SIZE.
-- To create a larger vector, **the new [leftmost] bit positions are filled with '0'**.

Arithmétique (+, -, *, /, REM, MOD) NON SIGNÉE

Opérations	Type IN left	Type IN right	RETURN
+ - * / rem mod	Std_logic_vector	Std_logic_vector	Std_logic_vector
	Integer	Std_logic_vector	
	Std_logic_vector	Integer	

Les paramètres peuvent être de longueur différentes

```
function "+" (L, R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
  -- Result: UNSIGNED add of two STD_LOGIC_VECTOR vectors that may be of different lengths.

  function "+" (L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(R'LENGTH-1 downto 0).
  -- Result: Adds an INTEGER, L(may be positive or negative), to a STD_LOGIC_VECTOR
  --         R which is assumed to be UNSIGNED.

  function "+" (L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(L'LENGTH-1 downto 0).
  -- Result: Adds a STD_LOGIC_VECTOR vector assumed UNSIGNED, L, to an INTEGER, R.

  function "-" (L, R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
  -- Result: UNSIGNED subtraction of two STD_LOGIC_VECTOR vectors that may be of different
   lengths.

  function "-" (L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(R'LENGTH-1 downto 0).
  -- Result: Subtracts a UNSIGNED STD_LOGIC_VECTOR, R, from an INTEGER, L.

  function "-" (L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(L'LENGTH-1 downto 0).
  -- Result: Subtracts an INTEGER, R, from a UNSIGNED STD_LOGIC_VECTOR vector, L.

  function "*" (L, R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR((L'LENGTH+R'LENGTH-1) downto 0)
  -- Result: Multiplies two STD_LOGIC_VECTOR vectors that may possibly be of
  --         different lengths.  The inputs and outputs are assumed to be UNSIGNED.

  function "*" (L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR((L'LENGTH+L'LENGTH-1) downto 0)
  -- Result: Multiplies a STD_LOGIC_VECTOR vector, L, with an INTEGER, R. R is
  --         converted to a UNSIGNED vector of SIZE L'LENGTH before
  --         multiplication.  The multiplication is UNSIGNED.

  function "*" (L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR((R'LENGTH+R'LENGTH-1) downto 0)
  -- Result: Multiplies a STD_LOGIC_VECTOR vector, R, with an INTEGER, L. L is
  --         converted to a UNSIGNED vector of SIZE R'LENGTH before
  --         multiplication.  The multiplication is UNSIGNED.

  function "/" (L, R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(L'LENGTH-1 downto 0)
  -- Result: Divides an STD_LOGIC_VECTOR vector, L, by another STD_LOGIC_VECTOR vector,
   R.
  --         This is an unsigned divide.

  function "/" (L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(L'LENGTH-1 downto 0)
  -- Result: Divides a STD_LOGIC_VECTOR vector, L, by an INTEGER, R.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to L'LENGTH.
  --         This is an unsigned divide.

  function "/" (L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(R'LENGTH-1 downto 0)
  -- Result: Divides an INTEGER, L, by a STD_LOGIC_VECTOR vector, R.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to R'LENGTH.
  --         This is an unsigned divide.

  function "rem" (L, R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L and R are STD_LOGIC_VECTOR vectors.
  --         This is an unsigned operation.

  function "rem" (L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(L'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L is STD_LOGIC_VECTOR vector and R is an INTEGER.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to L'LENGTH.
  --         This is an unsigned operation.

  function "rem" (L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where R is STD_LOGIC_VECTOR vector and L is an INTEGER.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to R'LENGTH.
  --         This is an unsigned operation.

  function "mod" (L, R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L and R are STD_LOGIC_VECTOR vectors.
  --         This is an unsigned operation.

  function "mod" (L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(L'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L is a STD_LOGIC_VECTOR vector and
  --         R is an INTEGER.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to L'LENGTH.
  --         This is an unsigned operation.

  function "mod" (L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
  -- Result subtype: STD_LOGIC_VECTOR(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L is an INTEGER and
  --         R is a STD_LOGIC_VECTOR vector.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to R'LENGTH.
  --         This is an unsigned operation.
```

~~package ieee.NUMERIC_SIGNED~~

Utilisation

Ce package est utilisé pour la synthèse de type NUMERIC SIGNED de std_logic_vector.
Synopsys, Inc.

Librairies utilisées

library **IEEE**;
use **IEEE.STD_LOGIC_1164**.all;
use **IEEE.NUMERIC_STD**.all;

Fonctions de Conversion/ RESIZE

Function	In	Param	Return
TO_INTEGER	Std_logic_vector		Integer
RESIZE	Std_logic_vector	New_size: natural	Std_logic_vector

function ***TO_INTEGER*** (ARG: STD_LOGIC_VECTOR) return INTEGER;

function ***RESIZE*** (ARG: STD_LOGIC_VECTOR; NEW_SIZE: NATURAL) return STD_LOGIC_VECTOR;

-- Result subtype: STD_LOGIC_VECTOR(NEW_SIZE-1 downto 0)

-- Result: ReSIZEs the STD_LOGIC_VECTOR vector ARG to the specified SIZE.

-- To create a larger vector, the new [leftmost] bit positions are filled with the sign bit (ARG'LEFT). When truncating, **the sign bit is retained along with the rightmost part**.

Arithmétique (+, -, *, /, REM, MOD) SIGNÉE

Opérations	Type IN left	Type IN right	RETURN
+ - * / rem mod	Std_logic_vector	Std_logic_vector	Std_logic_vector
	Integer	Std_logic_vector	
	Std_logic_vector	Integer	

Les paramètres peuvent être de longueurs différentes

~~package ieee.NUMERIC_EXTRA~~ (Mentor Graphics)

Librairies utilisées

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
```

Fonctions de Conversion

Function	Arg	Return
TO_STD_LOGIC	Boolean	Std_logic

function ***TO_STD_LOGIC***(arg : BOOLEAN) return STD_LOGIC;

Fonctions logiques de réduction

Opérations	Arg	Return
And_reduce **Nand_reduce** **Or_reduce** **Nor_reduce** **Xor_reduce** **Xnor_reduce**	Std_logic_vector	Std_logic

Effectue l'opération spécifiée entre tous les bits du vecteur d'entrée

Resumé des Librairies

Librairies / **types fonctions conversions**	std.standard	Ieee.std_logic_1164	~~Ieee.std_logic_arith~~	~~Ieee.std_logic_signed~~ ~~ieee.std_logic_unsigned~~	Ieee.numeric_std	Ieee.numeric_signed ieee.numeric_unsigned	Ieee.numeric_bit
bit	x						
std_logic/std_logic_vector		x	x	x			
integer	x		x	x	x	x	x
signed/unsigned of bit							x
signed/unsigned of std_logic			x		x	x	
logic		x	x		x		x
shift/rotate			x		x		x
arithm.			x	x	x	x	x
comparaison			x	x	x		x
rising/falling_edge		x					x
Conversions :							
To_bit		x					
To_bitvector		x					
To_StdULogic		x					
To_StdLogicVector		x					
To_StdULogicVector		x					
To_Integer			x		x	x	x
To_Stdlogic			x				
To_StdULogicVector			x				
To_StdLogicVector			x				
To_Unsigned			x		x		x
To_Signed			x		x		x
CONV_UNSIGNED			x				
CONV_SIGNED			x				
Zero_extend			x				
RESIZE					x	x	x