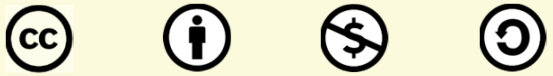
Paquetage numeric_std

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Etienne Messerli

Octobre 2021



Contenu de la présentation

- Présentation du paquetage numeric_std
  - Définit des types pour les nombres non-signés et signés
  - Définit les opérations arithmétiques de bases (+, -)
  - Définit la comparaison entre des nombre entier

Manuel VHDL, synthèse et simulation

Liste des chapitres à lire :

- Chapitre 7. Paquetage *numeric_std* et opérations arithmétiques, page 57 à 62
  - 7-1. Déclaration du paquetage *numeric_std*
  - 7-2. Fonctions définies dans le paquetage *numeric_std*
  - 7-3. Vecteurs et nombres
  - 7-3.1.Passage de *std_logic_vector* à *signed* ou *unsigned*
  - 7-3.2. Conversion d'un *signed* ou *unsigned* en un nombre entier
  - 7-4. Exemple d'additions
  - 7-5. Exemples de comparaison

Paquetage numeric_std

- Bibliothèque IEEE
- Norme IEEE-1076.3, 1997
  - paquetages `numeric_bit` & `numeric_std`
- Déclaration :

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

… paquetage numeric_std …

- But du paquetage :
  - types numériques et fonctions arithmétiques pour la synthèse
- Défini 2 types numériques :

  `unsigned` : vecteur représentant un nombre non signé

  `signed` : vecteur représentant un nombre signé C2

  Remarques :

  il s'agit de nombres **entier** en binaire

  ces 2 types sont basés sur le type `std_logic`

… paquetage numeric_std

- Représentation de nombre entier :
  - non signé
  - signé en représentation en complément à 2
- Paquetage contient :
  - surcharge des opérateurs arithmétiques pour les types `unsigned` et `signed`
  - fonctions de conversion et d'adaptation

Rappel:

- Bit le plus à gauche : bit le plus significatif, MSB (Most Significant Bit)
- Bit le plus à droite : bit le moins significatif, LSB (Least Significant Bit

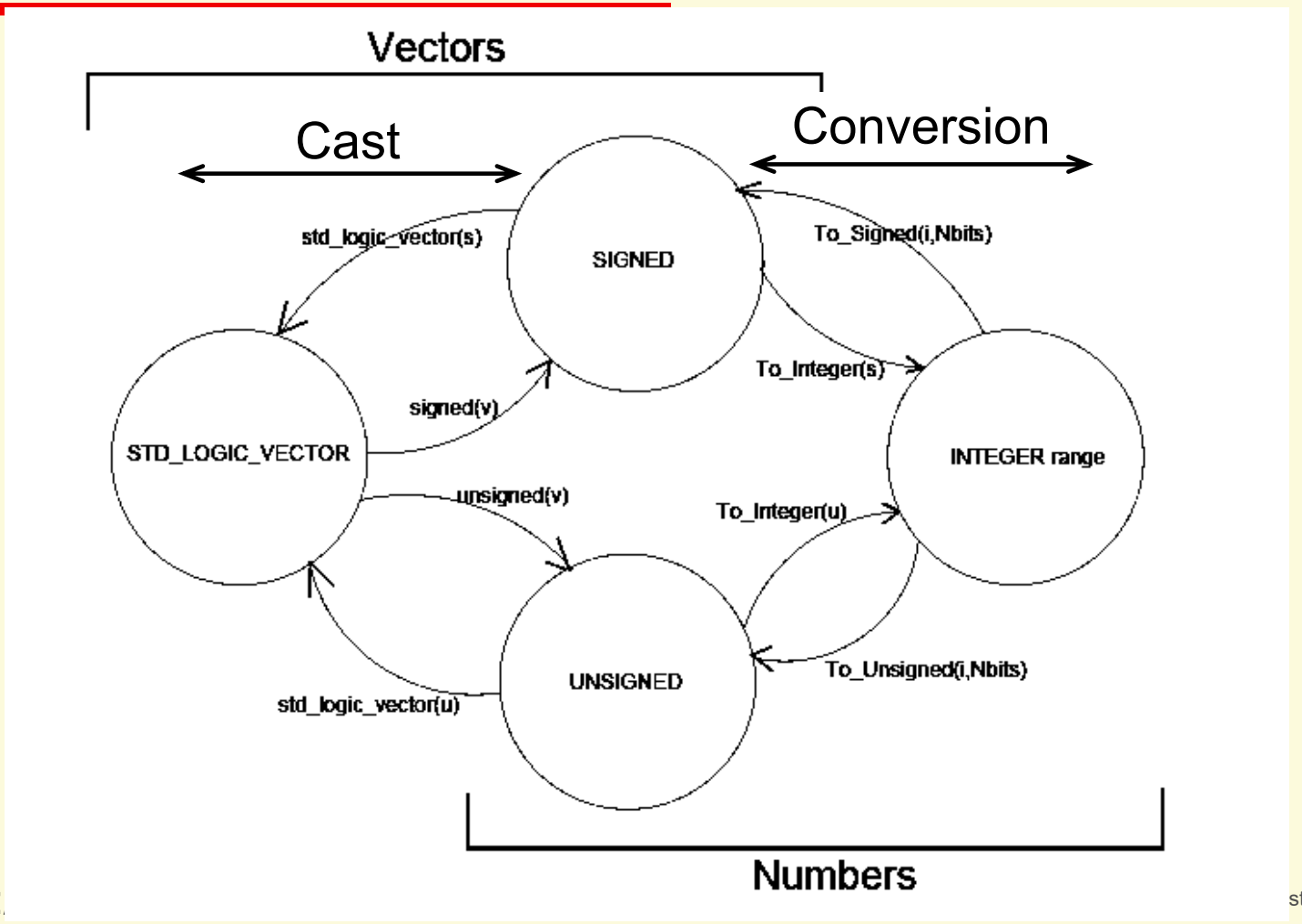
Types unsigned et signed

- Basé sur le type : `std_logic`
- Déclaration dans le paquetage :

```
type unsigned is array (natural range <>) of std_logic;
type signed   is array (natural range <>) of std_logic;
```

- Remarque : `std_logic_vector` <> `unsigned` ou `signed`
  - à chaque type correspond une interprétation différente :
    - vecteur binaire
    - nombre entier en binaire en signé ou non signé

Représentation : vecteurs/nombres



src: http:

Adaptation de type (*cast*)

```
-- Déclaration des signaux :
signal vecteur : std_logic_vector(7 downto 0);
signal nombre  : unsigned(7 downto 0);

--Exemples d'adaptation de type (cast):
nombre  <= unsigned(vecteur);
vecteur <= std_logic_vector(nombre);

--Basé sur le Std_Logic, donc :
vecteur(i) <= nombre(i);   --correct
nombre(i)  <= vecteur(i);  --correct
```

Fonctions de conversion ...

- Fonction `to_integer` :

```
to_integer (ARG: unsigned) return natural
to_integer (ARG: signed) return integer
```

- Fonction `to_unsigned` :

```
to_unsigned (ARG, SIZE: natural) return unsigned
```

- Fonction `to_signed` :

```
to_signed (ARG: integer; SIZE: natural) return signed
```

… fonctions de conversion

```
-- Déclaration des signaux :
signal nbr_entier : natural;
signal nbr_bin    : unsigned(7 downto 0);

--Exemples de conversion :
nbr_entier <= to_integer(nbr_bin);

nbr_bin <= to_unsigned(nbr_entier,8);

-- en utilisant l'attribut : 'length
nbr_bin <=
      to_unsigned(nbr_entier,nbr_bin'length);
```

Fonctions "+" et "-"

- Caractéristiques :

2 opérandes de tailles différentes

résultat aura la taille du plus grand

```
result :max(L'length, R'length)-1 downto 0
```

- Combinaisons des 2 opérandes (L & R):

```
(L, R: unsigned) return unsigned
(L: unsigned; R: natural) return unsigned
(L: natural; R: unsigned) return unsigned
(L, R: signed) return signed
(L: integer; R: signed) return signed
(L: signed; R: integer) return signed
```

Exemples d'addition

```
-- Déclaration des signaux :
signal na, nb   : unsigned(7 downto 0);
signal somme    : unsigned(7 downto 0);

--Exemples d'addition :
somme <= na + nb;
somme <= na + "0001";
somme <= nb + 1;
somme <= na + "00110011";
--Erroné somme <= nb_b + '1';  Pourquoi?
```

Similaire pour le type `signed`

Fonctions de comparaison

- Caractéristiques :

  Opérandes de tailles différentes

  Résultat de type booléen

- Combinaisons des 2 opérandes (L & R):

```
(L, R: unsigned) return boolean
(L: unsigned; R: natural) return boolean
(L: natural; R: unsigned) return boolean
(L, R: signed) return boolean
(L: integer; R: signed) return boolean
(L: signed; R: integer) return boolean
```

Exemples de comparaison ...

```
--déclaration des signaux
signal usgn_a,usgn_b : unsigned(3 downto 0);

--résultat de la comparaison est un booléen
  usgn_a =  "1010"          -- comp. unsigned - unsigned
  usgn_a = 10               -- comp. unsigned - integer
  usgn_a >= usgn_b
  usgn_a /= usgn_b
```

… exemples de comparaison

```
--déclaration des signaux
signal sgn_a, sgn_b : signed(3 downto 0);

--résultat de la comparaison est un booléen
--comparaison correcte avec nbr signé !
  sgn_a < "1100"    -- comp. signed - signed
  sgn_a < -4        -- comp. signed - integer

  sgn_a > sgn_b
```

Laboratoire

- Réalisation de différents additionneurs avec cin, cout et overflow.
  - Voir présentation séparée!

Questions ?