Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Système numérique

Résumé Chapitre 2 Types VHDL

Auteurs :
Marc Roten

Professeur :
Nicolas Schroeter

20 novembre 2018

Table des matières

1 Les types

VHDL définit quatre classes de types:

- Types scalaires (scalar): **énuméré** (enumerated)
  entier (integer)
  réel (real)
  physique (physical)
- Types composites: **tableau** (array)
  enregistrement (record)
- Type accès (access):
  - pointeur pour accéder à des objets d'un type donné
- Type fichier (file):
  - séquence de valeurs d'un type donné stockées dans un fichier

Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg
bleu = types synthétisables
SN-1 : Types VHDL

FIGURE 1 – Les différents types

1.1 Scalaire enuméré

Sert à la simplification du code.

- Type énuméré: liste ordonnée de valeurs.
- Types énumérés prédéfinis du paquetage standard :

```
type bit is ('0', '1'); --caractères
type boolean is (false, true); --identificateurs
type character is les 256 caractères ISO 8859-1;
type severity_level is (NOTE,WARNING,ERROR,FAILURE);
```

- Exemples de définition:

```
type logic is ('0', '1', 'Z', 'X');
type etats is (idle, treat, done); --machines d'états
type mixed is ('0', idle, 'c', done); -- car + ident.
```

FIGURE 2 – Utilisation des Enum

1.1.1 focus stdLogic

- Type std_logic: type de base à utiliser pour les signaux logiques défini dans le paquetage 1164.

```
type std_ulogic is ( 'U', -- état non initialisé
                     'X', -- état logique indéfini fort
                     '0', -- état logique 0 fort
                     '1', -- état logique 1 fort
                     'Z', -- état à haute impédance
                     'W', -- état logique indéfini faible
                     'L', -- état logique 0 faible
                     'H', -- état logique 1 faible
                     '-', -- état logique indifférent );

subtype std_logic is resolved std_ulogic;
```

FIGURE 3 – stdLogic

1.2 Scalaire entier

- Types prédéfinis du paquetage standard:

```
type INTEGER is range -2'147'483'648 to 2'147'483'647;
subtype NATURAL is INTEGER range 0 to INTEGER'HIGH;
subtype POSITIVE is INTEGER range 1 to INTEGER'HIGH;
```

FIGURE 4 – Types prédéfinis

- Exemples de définition:

```
type short is range -128 to 127;
subtype nat4 is natural range 0 to 15;
subtype offset is nat4 range 14 to 15;
```

FIGURE 5 – Types définis par le programmeur

1.3 Scalaire réel

1.4 Scalaire physique

- Type time est défini par son intervalle de valeurs, son unité de base et ses sous-unités:

```
type time is range -2147483647 to 2147483647
-- pour exprimer le temps en femtosecondes
```

FIGURE 6 – Scalaire physique

1.5 Composite Array

```
ARCHITECTURE BEV OF RAM IS                                    VHDLBASIC
    TYPE MEM IS ARRAY (255 DOWNTO 0) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL MEMORY : MEM;
    SIGNAL ADDR : INTEGER RANGE 0 TO 255;
    BEGIN
    PROCESS(ADDRESS, DATAIN, W_R)
    BEGIN
        ADDR<=CONV_INTEGER(ADDRESS);
        IF(W_R='0')THEN
            MEMORY(ADDR)<=DATAIN;
            ELSIF(W_R='1')THEN
                DATAOUT<=MEMORY(ADDR);
                ELSE
                DATAOUT<="ZZZZZZZZ";
            END IF;
END PROCESS;
```

FIGURE 7 – Définition tableau

1.5.1 Exemple divers

```
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);
```

```
--------- 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
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)
```

FIGURE 8 – Exemple divers

2 Attribut Scalaires

- Attributs selon l'ordre de déclaration dans l'énuméré:
  - LEFT: élément le plus à gauche de l'énuméré
  - RIGHT: élément le plus à droite de l'énuméré
  - LEFTOF(value): élément à gauche de la valeur
  - RIGHTOF(value): élément à droite de la valeur
  - POS(value): position dans l'énuméré, en débutant par 0 pour l'élément le plus à gauche
  - VAL(position): valeur à la position de l'intervalle
- Attributs selon l'ensemble des valeurs:
  - HIGH: valeur la plus grande dans l'énuméré
  - LOW: valeur la plus petite dans l'énuméré
  - PRED(value): valeur précédente
  - SUCC(value): valeur suivante

FIGURE 9 – Attributs Scalaire physique

3 Attribut types composites

- Les attributs suivants sont définis pour n'importe quel type de tableau:
  - LEFT: l'indice le plus à gauche de l'intervalle
  - RIGHT: l'indice le plus à droite de l'intervalle
  - HIGH: l'indice le plus grand dans l'intervalle
  - LOW: l'indice le plus petit dans l'intervalle
  - RANGE: sous-type des indices, intervalle entre l'attribut LEFT et RIGHT
  - REVERSE_RANGE: intervalle inverse de RANGE
  - LENGTH: nombre d'éléments du tableau

NB :

- Les valeurs d'indice retournées sont du type de l'intervalle (range).
- LENGTH retourne une valeur du type INTEGER

FIGURE 10 – AttributsTypes composites tableau

3.1 Agrégats

Les agrégats permettent de définir des valeurs pour les types composites tableau et enregistrement.

3.1.1 Agrégat simple

- Les agrégats permettent de définir des valeurs pour les types composites tableau et enregistrement.
- Exemples:

```
signal a : std_logic_vector(3 downto 0);
-- affectation de "1000"
Named notation: indice noté explicitement
a <= (3 => '1', 2 => '0', 1 => '0', 0 => '0');
a <= (3 => '1', 2 | 1 | 0 => '0'); -- choix
a <= (3 => '1', 2 downto 0 => '0'); -- subrange
a <= (3 => '1', others => '0'); -- others doit toujours
                                -- être le dernier élément
Positional notation: indices implicites.
a <= ('1', '0', '0', '0');
```

NB:

- VHDL interdit de mélanger les 2 notations lors d'une déclaration.
- Il est conseillé d'utiliser la «named notation» pour augmenter la lisibilité et limiter le risque d'erreurs.

Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg
SN-1 : Types VHDL 32

FIGURE 11 – Faire un agrégat

3.1.2 Agrégats moultes et variés

```
type symbol is ('a', 't', 'd', 'h', digit, cr, other);
type state is range 0 to 3;
type transition_matrix is array (state, symbol) of state;
constant named_state : transition_matrix :=
    ( 0 => ('a' => 1, others => 0),
      1 => ('t' => 2, others => 3),
      2 => ('d' => 3, 'h' => 2, others => 1),
      3 => (digit => 0, others => 2));

constant positional_state : transition_matrix :=
   ((1,0,0,0,0,0,0),
    (3,2,3,3,3,3,3),
    (1,1,3,2,1,1,1),
    (2,2,2,2,0,2,2));
```

FIGURE 12 – Agrégats divers

4 Les opérateurs

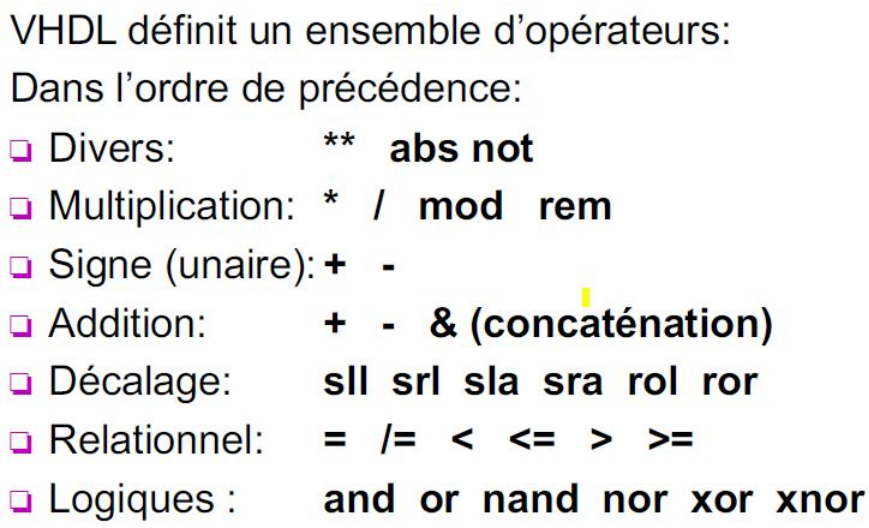

FIGURE 13 – Les opérateurs en VHDL

5 Types supportés par chaque classe Opérateur

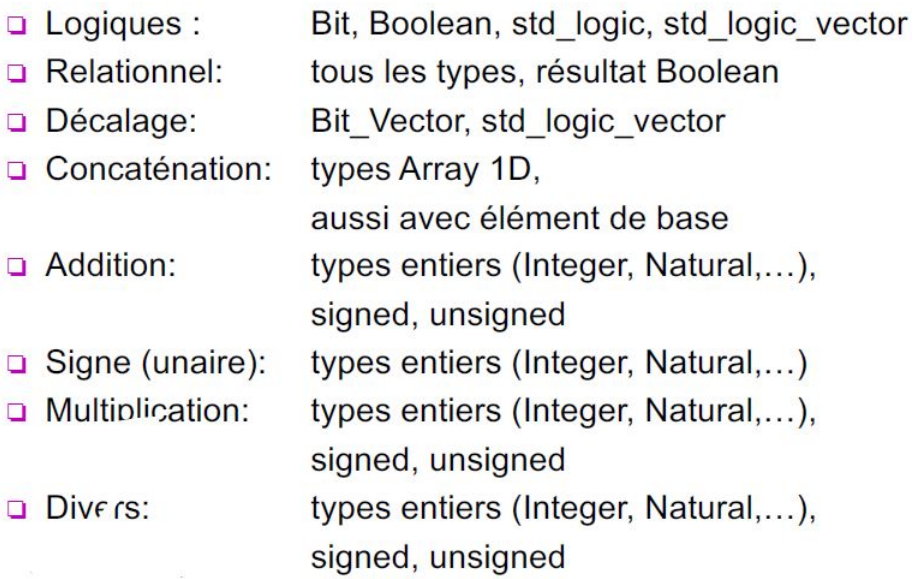

FIGURE 14 – Les opérateurs en VHDL

6 Exemple d'affectation

- Une simple affection (sans voir la déclaration) peut suggérer plusieurs types pour un même signal

```
x0 <= '0';            -- bit, std_logic, or std_ulogic value '0'
x1 <= "00011111";     -- bit_vector, std_logic_vector,
                      -- std_ulogic_vector, signed, or unsigned
x2 <= "101111"        -- binary representation of decimal 47
x3 <= B"101111"       -- binary representation of decimal 47
x4 <= O"57"           -- octal representation of decimal 47
x5 <= X"2F"           -- hexadecimal representation of decimal 47
n <= 1200;            -- integer
IF ready THEN...      -- Boolean, executed if ready=TRUE
y <= 1.2E-5;          -- real, not synthesizable
q <= d after 10 ns; -- physical, not synthesizable
```

FIGURE 15 – Affectation Simple