Concepts intermédiaires de VHDL

Sujets de ce thème

- Objets: constantes, signaux, variables et fichiers
- Structures de répétition et de sélection
- Attributs de signaux
- Définition de types scalaires et composés
- Fonctions et procédures

Quatre catégories d'objets en VHDL

Il y a quatre catégories d'objets en VHDL :

- `constant` (et `generic`) : peut contenir une valeur unique qui ne change pas; un objet `generic` est une constante spéciale permettant d'appliquer un paramètre à une entité lors de son instanciation;
- `signal` et `port` : peuvent contenir une liste de valeurs dans le temps; un objet `signal` correspond en général à un fil d'un circuit; un `port` d'une entité est implicitement un signal pour cette entité;
- `variable`: peut contenir une valeur temporaire; les objets `variable` sont utiles pour stocker des valeurs intérimaires dans les calculs;
- `file`: un objet dans lequel on peut écrire et lire des valeurs, et qui correspond à un fichier du système d'exploitation.
  (*Nous n'allons pas discuter de ce type d'objet dans cette présentation*).

Lors de la déclaration d'un objet, on spécifie sa catégorie, son identificateur et son type. On peut aussi lui assigner une valeur initiale.

VHDL – objets: 1. `constant` et `generic`

- Une `constant` (et un `generic`) peuvent contenir une valeur unique qui ne change pas.
- Un objet `generic` est une constante spéciale permettant d'appliquer un paramètre à une entité lors de son instanciation.
- Un objet `constant` peut être défini pour une architecture ou un processus.

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

entity vote is
   generic (
      W : positive := 4
   );
   port (
      lesvotes: in std_logic_vector(W - 1 downto 0);
      approbation : out std_logic
   );
end vote;

architecture comportementale3 of vote is
constant nMin : natural := W / 2;
begin
   process(lesvotes)
   variable compte : natural range 0 to lesvotes'length;
   begin
      compte := 0;
      for k in lesvotes'range loop
         if lesvotes(k) = '1' then
            compte := compte + 1;
         end if;
      end loop;
      if compte > nMin then
         approbation <= '1';
      else
         approbation <= '0';
      end if;
   end process;
end comportementale3;
```

VHDL – objets: 2. `signal` et `port`

- Un `signal` est déclaré dans la partie déclarative d'une architecture et est visible partout dans celle-ci.
- Un `signal` représente à peu près un fil ou un groupe de fils dans un circuit.
- Un `port` d'une entité est implicitement un signal dans toutes les architectures de cette entité.
  - Un `port` de direction `in` est comme un signal qui ne peut pas être écrit.
  - Un `port` de direction `out` est comme un signal qui ne peut pas être lu.
- Pour assigner une valeur à un `port` on utilise l'opérateur <=

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

entity vote is
   generic (
      W : positive := 4
   );
   port (
      lesvotes: in std_logic_vector(W - 1 downto 0);
      approbation : out std_logic
   );
end vote;

architecture flotdonnees2 of vote is

signal A, B, C, D : std_logic;

begin

   (A, B, C, D) <= lesvotes;
   approbation <=
      (not(A) and B and C and D)
      or (A and not(B) and C and D)
      or (A and B and not(C) and D)
      or (A and B and C and not(D))
      or (A and B and C and D);
end flotdonnees2;
```

VHDL – objets: 3. `variable`

- Une `variable` peut contenir une valeur temporaire.
- Les `variables` ne peuvent être utilisées qu'à l'intérieur des processus.
- Les objets `variable` sont utiles pour stocker des valeurs intérimaires dans les calculs.
- Pour assigner une valeur à une `variable` on utilise l'opérateur `:=`

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

entity vote is
   generic (
      W : positive := 4
   );
   port (
      lesvotes: in std_logic_vector(W - 1 downto 0);
      approbation : out std_logic
   );
end vote;

architecture comportementale3 of vote is
constant nMin : natural := W / 2;
begin
   process(lesvotes)
   variable compte : natural range 0 to lesvotes'length;
   begin
      compte := 0;
      for k in lesvotes'range loop
         if lesvotes(k) = '1' then
            compte := compte + 1;
         end if;
      end loop;
      if compte > nMin then
         approbation <= '1';
      else
         approbation <= '0';
      end if;
   end process;
end comportementale3;
```

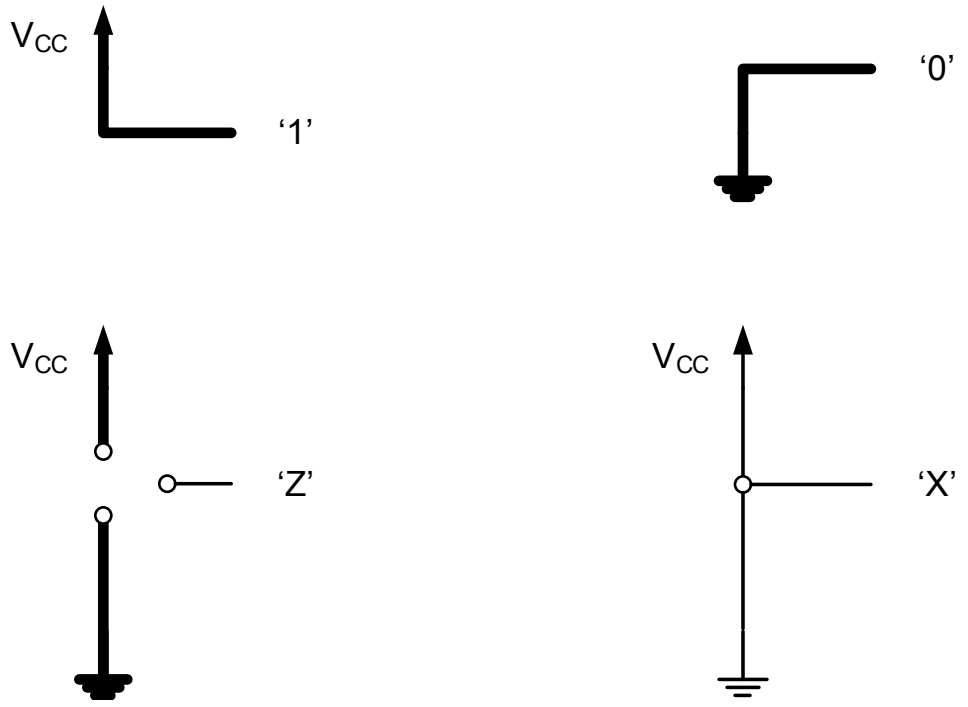
VHDL: types prédéfinis

- Les `generic`, `constant`, `signal`, `port` et `variable` doivent avoir un type.
- VHDL inclut plusieurs types prédéfinis et d'autres qui sont définis dans des packages normalisés.
- On peut aussi définir de nouveaux types.

catégorie	type ou sous-type	source de la définition	valeurs
scalaires	boolean	type prédéfini	FALSE et TRUE
	bit	type prédéfini	'0' et '1'
	character	type prédéfini	256 caractères de la norme ISO 8859-1, avec des abréviations reconnues et certaines qui sont propres à VHDL Les 128 premiers sont les caractères ASCII.
	integer	type prédéfini	plage minimale de $-2^{31} + 1$ à $2^{31} - 1$
	natural	sous-type prédéfini	0 à $2^{31} - 1$
	positive	sous-type prédéfini	1 à $2^{31} - 1$
	real	type prédéfini	typiquement –1.7014111E±308 à 1.7014111E±308
	std_logic	Package std_logic_1164	'U' : valeur inconnue, pas initialisée 'X' : valeur inconnue forcée '0' : 0 forcé '1' : 1 forcé 'Z' : haute impédance (pas connecté) 'W' : inconnu faible 'L' : 0 faible 'H' : 1 faible '-' : peu importe (don't care)
composés	bit_vector	type prédéfini	tableau de bit
	string	type prédéfini	tableau de character
	std_logic_vector	Package std_logic_1164	tableau de std_logic
	unsigned	Package numeric_std	tableau de std_logic, interprété comme un nombre binaire non signé
	signed	Package numeric_std	tableau de std_logic, interprété comme un nombre binaire signé en complément à deux

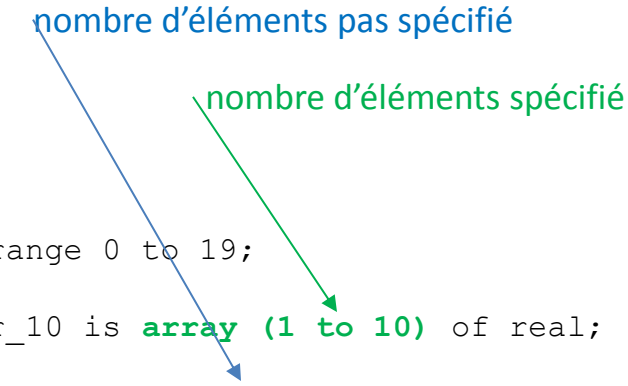
Le type `std_logic`

- Le type `std_logic` est utilisé pour modéliser les valeurs logiques sur un fil électrique.
- Un signal ou une variable de type `std_logic` peut prendre l'une de neuf valeurs:
  '1', '0', 'Z', 'X', 'U', 'W', 'L', 'H' et '-'.
- Une valeur de 'X' représente en général un signal mené par deux sources en même temps.
- La valeur 'U' est donnée au début d'une simulation.
- La valeur '-' représente une valeur sans importance – *don't care.*
- Les valeurs 'W', 'L' et 'H' sont peu utilisées.



Définition de nouveaux types en VHDL

- Les définitions de types sont faites dans la partie déclarative d'une architecture.
- On peut définir de nouveaux types avec le mot clé `type`, et des sous-types avec le mot clé `subtype`.
- On peut entre autres définir des types composés dont les éléments:
  - sont tous du même type et sont numérotés (vecteurs ou matrices, `array`)
  - être de types différents et qui sont nommés (structures, `record`).




```
type entiers_sous_20 is range 0 to 19;

type matrice_reelle_1_par_10 is array (1 to 10) of real;

type tableauSLV3 is array (natural range <>) of
std_logic_vector(2 downto 0);

constant vecteurs : tableauSLV3 :=
("000", "001", "010", "011", "100", "101", "110", "111");

type nom_de_mois is (janvier, février, mars, avril, mai,
juin, juillet, aout, septembre, octobre, novembre,
décembre);

type date is record
jour : integer range 1 to 31;
mois : nom_de_mois;
année : positive range 1 to 3000;
end record;

constant confederation : date := (1, juillet, 1867);
```

Structures de répétition et de sélection

- Les énoncés `for`, `while`, `if-then-else` et `case` ne s'utilisent qu'à l'intérieur des processus.

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

entity vote is
   generic (
      W : positive := 4
   );
   port (
      lesvotes: in std_logic_vector(W - 1 downto 0);
      approbation : out std_logic
   );
end vote;

architecture comportementale3 of vote is
constant nMin : natural := W / 2;
begin
   process(lesvotes)
   variable compte : natural range 0 to lesvotes'length;
   begin
      compte := 0;
      for k in lesvotes'range loop
         if lesvotes(k) = '1' then
            compte := compte + 1;
         end if;
      end loop;
      if compte > nMin then
         approbation <= '1';
      else
         approbation <= '0';
      end if;
   end process;
end comportementale3;
```

Attributs en VHDL

- En VHDL, les attributs permettent d'obtenir de l'information à propos de types, signaux, variables, etc.
- VHDL inclut des attributs prédéfinis, et on peut en définir de nouveaux.
- Les attributs s'utilisent dans des expressions.

Utilisation de l'attribut	Résultat
A'left, A'right	Indice le plus à gauche, le plus à droite du vecteur A
A'low, A'high	Indice le plus bas, le plus haut du vecteur A
A'range	La gamme des indices du vecteur A
A'length	Le nombre d'éléments du vecteur A
T'left, T'right, T'low, T'high	La valeur la plus à gauche, à droite, la plus basse, la plus élevée du type T
T'Image(x)	La représentation textuelle de la valeur x dans le type T.
T'Value(s)	La valeur dans le type T de la chaîne de caractères s.
S'event	Vrai si un événement s'est produit sur le signal S.

```
...
   process(lesvotes)
   variable compte : natural range 0 to lesvotes'length;
   begin
      compte := 0;
      for k in lesvotes'range loop
...
```

Fonctions et procédures en VHDL

- Les fonctions et procédures peuvent être définies dans la partie déclarative d'une architecture ou bien dans un package.
- Les fonctions et les procédures peuvent être surchargées, c'est-à-dire que deux d'entre elles peuvent avoir le même identificateur mais une liste de paramètres différents. (On peut aussi surcharger les opérateurs de VHDL.)

```
function porteET(V: std_logic_vector) return std_logic is
variable resultat : std_logic := '1';
begin
   for k in V'range loop
      resultat := resultat and V(k);
   end loop;
   return resultat;
end;
```

- Une fonction …
  - est un sous-programme qui retourne un résultat unique;
  - retourne une valeur qui peut être d'un type scalaire ou composé;
  - peut accepter des paramètres en entrée, ces paramètres ne peuvent pas être modifiés par la fonction;
  - est appelée dans une expression.
- Une procédure …
  - est un sous-programme avec une liste de paramètres qui peuvent avoir les modes `in`, `out` et `inout`, permettant ainsi à la procédure d'accepter des valeurs en entrée, de retourner des résultats, et de modifier des paramètres qui lui sont passés;
  - est appelée comme un énoncé concurrent.

Vous devriez maintenant être capable de …

- Décrire et utiliser les quatre catégories d'objet en VHDL: constante, signal, variable et fichier. (B2, B3)
- À l'intérieur d'un processus, utiliser les structures de répétition et de sélection. (B3)
- Utiliser les attributs les plus utiles en VHDL: `left`, `right`, `low`, `high`, `range`, `length`, `image`, `value`, `event`. (B3)
- Utiliser la définition de nouveaux types scalaires et composés en VHDL. (B3)
- Utiliser les fonctions et les procédures en VHDL. (B3)

Code	Niveau (http://fr.wikipedia.org/wiki/Taxonomie_de_Bloom)
B1	Connaissance - mémoriser de l'information.
B2	Compréhension – interpréter l'information.
B3	Application – confronter les connaissances à des cas pratiques simples.
B4	Analyse – décomposer un problème, cas pratiques plus complexes.
B5	Synthèse – expression personnelle, cas pratiques plus complexes.