Introduction à VHDL

Sujets de ce thème

- VHDL: origines et avantages
- Modéliser un circuit numérique simple en VHDL
- Casse, espaces, commentaires et littéraux
- Identificateurs
- Types et opérateurs

Langages de description matérielle

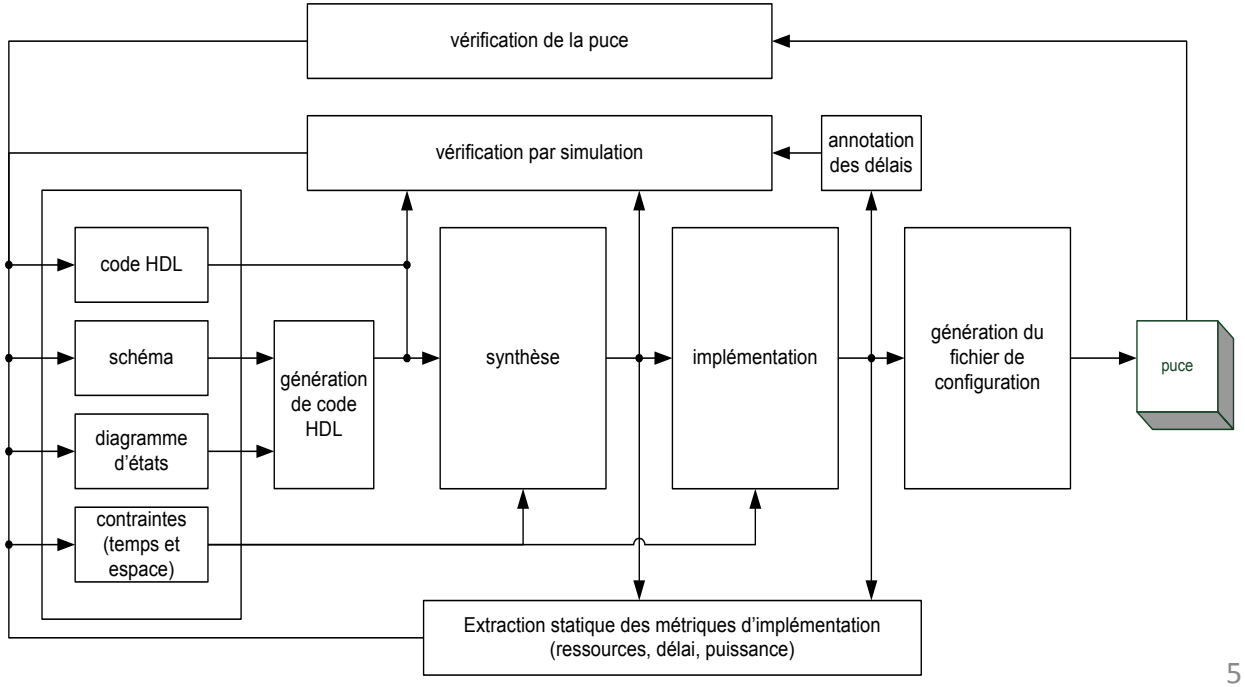
- Les langages de description matérielle (*Hardware Description Language* – HDL) ont vu le jour au début des années 1980 lorsque la complexité des circuits à concevoir a rendu impossible l'utilisation exclusive de schémas.
- Les schémas sont peu adéquats pour décrire un circuit logique.
  - Ils sont limités aux circuits les plus simples.
  - Il est difficile de dessiner et modifier un circuit complexe avec un schéma de portes logiques.
  - Un simple changement dans une équation booléenne du circuit peut se répercuter par une grande quantité de connexions à corriger.
  - Il est difficile d'utiliser des variables en guise de paramètres d'un circuit représenté par un schéma.
- Les HDL peuvent servir à trois choses :
  - la modélisation de circuits (surtout numériques);
  - la description de circuits en vue de leur synthèse (i.e. leur réalisation matérielle); et,
  - la documentation de circuits.
- Avantages des HDL par rapport aux schémas :
  - les HDL permettent de décrire des systèmes complexes complets;
  - les HDL favorisent la décomposition en modules paramétrables;
  - les HDL facilitent l'établissement de spécifications et d'interfaces clairement définies;
  - les HDL normalisent l'échange d'informations.

VHDL, Verilog, SystemC

- Les deux HDL les plus populaires sont Verilog et VHDL.
- Verilog ressemble un peu à C, et VHDL ressemble à ADA.
- Les deux langages sont relativement faciles à apprendre, mais difficiles à maîtriser.
- VHDL est plus vaste, bien que plusieurs des particularités pour lesquelles Verilog n'a pas d'équivalent soient rarement utilisées.
- Quand on connaît l'un des deux langages, il est relativement aisé de passer à l'autre.
- Un troisième langage, SystemC, est populaire pour la modélisation à plus haut niveau.

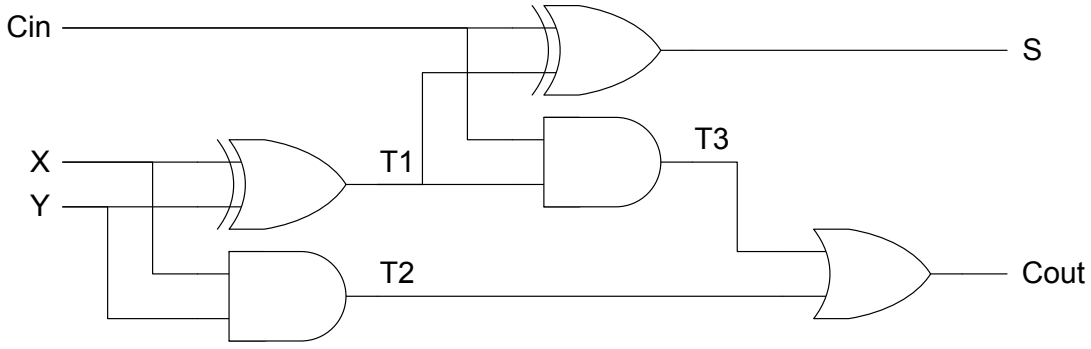
VHDL (*Very high speed integrated circuit Hardware Description Language)*

- VHDL est un langage de programmation complet. Le langage a été développé pour le compte du gouvernement américain pour documenter la conception d'ASIC. Il est fortement inspiré du langage ADA.
- Rapidement, des simulateurs de VHDL sont apparus, puis des synthétiseurs capables de traduire un programme VHDL en une liste d'interconnexions entre des portes logiques (*netlist*) pouvant être réalisée sur un ASIC.
- Le langage VHDL est normalisé par l'IEEE. La première norme remonte à 1987. Des mises à jour ont eu lieu en 1993, 2000, 2002 et 2008. La norme plus récente est présentement mal supportée par les outils de conception.



Modèle VHDL d'un circuit combinatoire simple

- Librarie: définition de types, fonctions, etc.
- Entité: interface avec le monde extérieur
  - Définit les ports, leur type et leur direction
- Architecture: partie déclarative et corps
  - Définit le comportement du module
- Principe de la concurrence
  - L'ordre est sans importance dans le corps de l'architecture.




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity add3bits is

port (
   Cin : in std_logic;
   X : in std_logic;
   Y : in std_logic;
   Cout : out std_logic;
   S : out std_logic
);

end add3bits;

architecture flotDeDonnees of add3bits is

signal T1 : std_logic;
signal T2 : std_logic;
signal T3 : std_logic;

begin

   S <= T1 xor Cin;
   Cout <= T3 or T2;
   T1 <= X xor Y;
   T2 <= X and Y;
   T3 <= Cin and T1;

end flotDeDonnees;
```

VHDL: casse, espaces, commentaires et littéraux

- VHDL n'est pas sensible à la casse.
- Il n'y a pas de différence entre les espaces, tabulations et retours de chariot, présents seuls ou en groupe.
- Tout texte placé après deux tirets et jusqu'à la fin d'une ligne est un commentaire, ce qui est semblable au '//' de C.
  Il n'y a pas de commentaires en bloc (/* … */) en VHDL.

Littéraux:

- Un littéral composé d'un caractère unique est placé entre apostrophes : 'a', '5'.
- Un littéral composé d'une chaîne de caractères est placé entre guillemets : "bonjour", "123ABC".
- Un littéral numérique peut être spécifié avec sa base (de 2 à 16), avec le format base#chiffres[.chiffres]#[exposant]. Par exemple, les nombres suivants ont tous la même valeur : 10#33# = 10#3.3#E1 = 2#10001# = 16#21# = 7#45#.
- Un littéral composé de bits peut être exprimé en base 2, 8 ou 16, avec les spécificateurs de base B, O et X, respectivement : B"11111111", O"377", X"FF".

VHDL: identificateurs

- Un identificateur de base légal est composé de lettres, chiffres et/ou du soulignement.
- Le premier caractère doit être une lettre, le dernier ne peut pas être le soulignement, et on ne peut pas utiliser deux soulignements de suite.
- Un identificateur ne peut pas être l'un des mots réservés du langage.

Les mots réservés de VHDL (selon la norme 1076-2002) sont :

```
abs, access, after, alias, all, and, architecture, array, assert, attribute, begin, block, body,
buffer, bus, case, component, configuration, constant, disconnect, downto, else, elsif, end, entity,
exit, file, for, function, generate, generic, group, guarded, if, impure, in, inertial, inout, is,
label, library, linkage, literal, loop, map, mod, nand, new, next, nor, not, null, of, on, open, or,
others, out, package, port, postponed, procedural, procedure, process, protected, pure, range,
record, reference, register, reject, rem, report, return, rol, ror, select, severity, signal, shared,
sla, sll, sra, srl, subtype, then, to, transport, type, unaffected, units, until, use, variable,
wait, when, while, with, xnor, xor
```

VHDL: types prédéfinis et de base

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

VHDL: opérateurs

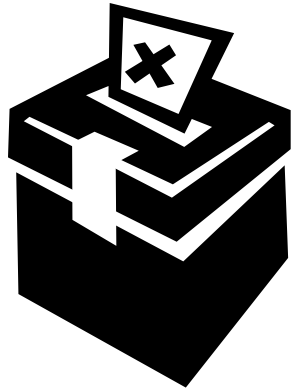
<table>
<tr><th>catégorie</th><th>opérateurs</th><th>type de l'opérande de gauche</th><th>type de l'opérande de droite</th><th>type de l'expression</th></tr>
<tr><td>logique</td><td>and, or, nand, nor, xor, xnor, not</td><td colspan="3">bit, boolean</td></tr>
<tr><td>relation</td><td>=, /=, <, <=, >, >=</td><td colspan="2">scalaire ou tableau</td><td>boolean</td></tr>
<tr><td>décalage</td><td>sll (déc. logique gauche),
srl (déc. logique droite),
sla (déc. arithmétique gauche),
sra (déc. arithmétique droit),
rol (rotation gauche),
ror (rotation droite)</td><td>tableau de bit ou boolean</td><td>integer</td><td>comme l'opérande de gauche</td></tr>
<tr><td rowspan="2">arithmétique</td><td>+, -, *, /,
abs (valeur absolue),
mod (modulo),
rem (reste)</td><td rowspan="2">type numérique</td><td>type numérique</td><td rowspan="2">type numérique</td></tr>
<tr><td>** (exponentiation)</td><td>integer</td></tr>
<tr><td>concaténation</td><td>&</td><td colspan="2">tableau ou type énuméré</td><td>tableau</td></tr>
</table>

Retour: le problème du vote

Un comité composé de quatre personnes a besoin d'un mécanisme de vote secret pour les amendements sur la constitution du comité.

Un amendement est approuvé si au moins 3 personnes votent pour.

Concevoir un circuit logique qui accepte 4 entrées représentant les votes. La sortie du circuit doit indiquer si l'amendement est accepté.



Retour: le problème du vote – modèle VHDL, version 1

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

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

entity vote is
   port (
      lesvotes: in std_logic_vector(3 downto 0);
      approbation : out std_logic
   );
end vote;

-- table de vérité réduite
architecture flotdonnees1 of vote is
begin
   with lesvotes select
      approbation <=
            '1' when "0111",
            '1' when "1011",
            '1' when "1101",
            '1' when "1110",
            '1' when "1111",
            '0' when others;
end flotdonnees1;
```

Retour: le problème du vote – modèle VHDL, versions 2 et 3

$$F = A'BCD + AB'CD + ABC'D + ABCD' + ABCD$$
$$= BCD + ACD + ABD + ABC$$

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

entity vote is
   port (
      lesvotes: in std_logic_vector(3 downto 0);
      approbation : out std_logic
   );
end vote;
```

```
-- équation non réduite
architecture flotdonnees2 of vote is

signal A, B, C, D : std_logic;

begin
   (A, B, C, D) <= lesvotes; -- pour simplifier l'écriture
   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;
```

```
-- équation réduite
architecture flotdonnees3 of vote is

signal A, B, C, D : std_logic;

begin
   (A, B, C, D) <= lesvotes; -- pour simplifier l'écriture
   approbation <=
      (B and C and D)
      or (A and C and D)
      or (A and B and D)
      or (A and B and C);
end flotdonnees3;
```

Vous devriez maintenant être capable de …

- Décrire les origines des HDL et de VHDL. (B2)
- Donner les avantages des HDL pour décrire des circuits numériques. (B1)
- Appliquer la structure de base de la description d'un module en VHDL (librairie et package, entité et architecture, ports, déclaration et utilisation de signaux, assignations concurrentes). (B3)
- Appliquer les règles de base de VHDL concernant la casse, les espaces, littéraux, commentaires et identificateurs. (B3)
- Énumérer, décrire et utiliser les types prédéfinis, les valeurs du type `std_logic` et les opérateurs de VHDL. (B1, B2, 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.