ARCHITECTURE LOGIQUE PROGRAMMABLE

Cne. Brinsi Riadh

Académie militaire F.J

br.riadh@gmail.com

11 novembre 2015

Le langage VHDL (1/2)

c'est un langage de conception, simulation et synthèse de circuits numériques (du pluss simple au plus compliqué)

- Langage unique pour :
  - la spécification, la documentation
  - la vérification (preuve formelle), la simulation
  - la synthèse : le langage sert d'entrée à des outils intelligents qui permettent la réalisation de circuits intégrés (customs, ASICs) ou programmables (PALs, PLDs, CPLDs, FPGAs)
- Interprétable par l'homme et la machine
- Indépendant des process et des technologies, indépendant des systèmes-hôtes

Le langage VHDL (2/2)

Ce n'est pas un langage informatique classique : il sert à décrire un matériel et à créer un circuit.

- blocs d'instructions exécutés simultanément
- prise en compte de la réalité : possibilité de spécifier des retards, de préciser ce qui se passe en cas de conflits de bus, ...

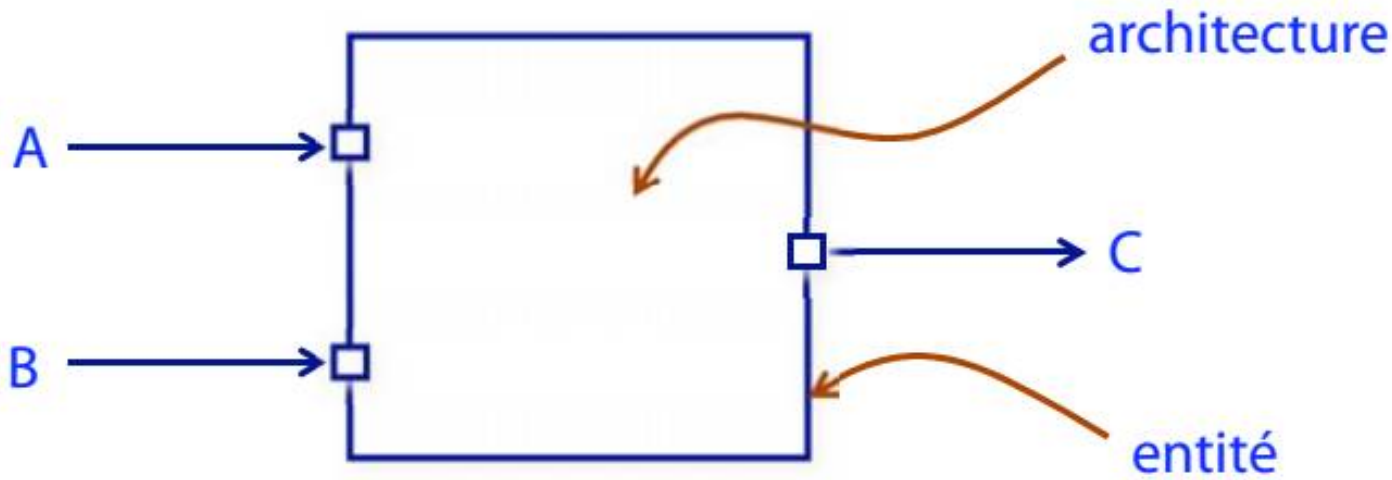
Les mots réservés

abs	disconnect	label	package	then
acces	downto	library	port	to
after		linkage	procedure	transport
alias	else	loop	process	type
all	elsif			
and	end	map	range	units
architecture	entity		record	until
array	exit	nand	register	use
assertnew			rem	
attribute	file	next	report	variable
	for	nor	return	
begin	function	not		wait
blocknull			select	when
body	generate		severity	with
buffer	generic	of	signal	while
bus	guarded	on	subtype	with
		open		
case	if	or		
component	in	others		
configuration	inout	out		
constant	is			

Entité et architecture

Une description vhdl est composée de deux parties : une entité et une architecture

- Au plus haut niveau d'abstraction, un système digital est vu comme une "boîte noire", dont on connaît l'interface avec l'extérieur mais dont on ignore le contenu c'est l'entité (**entity**)
- Une entité doit toujours être associée avec au moins une description de son contenu, de son implémentation : c'est l' **architecture**



Structure d'un programme VHDL

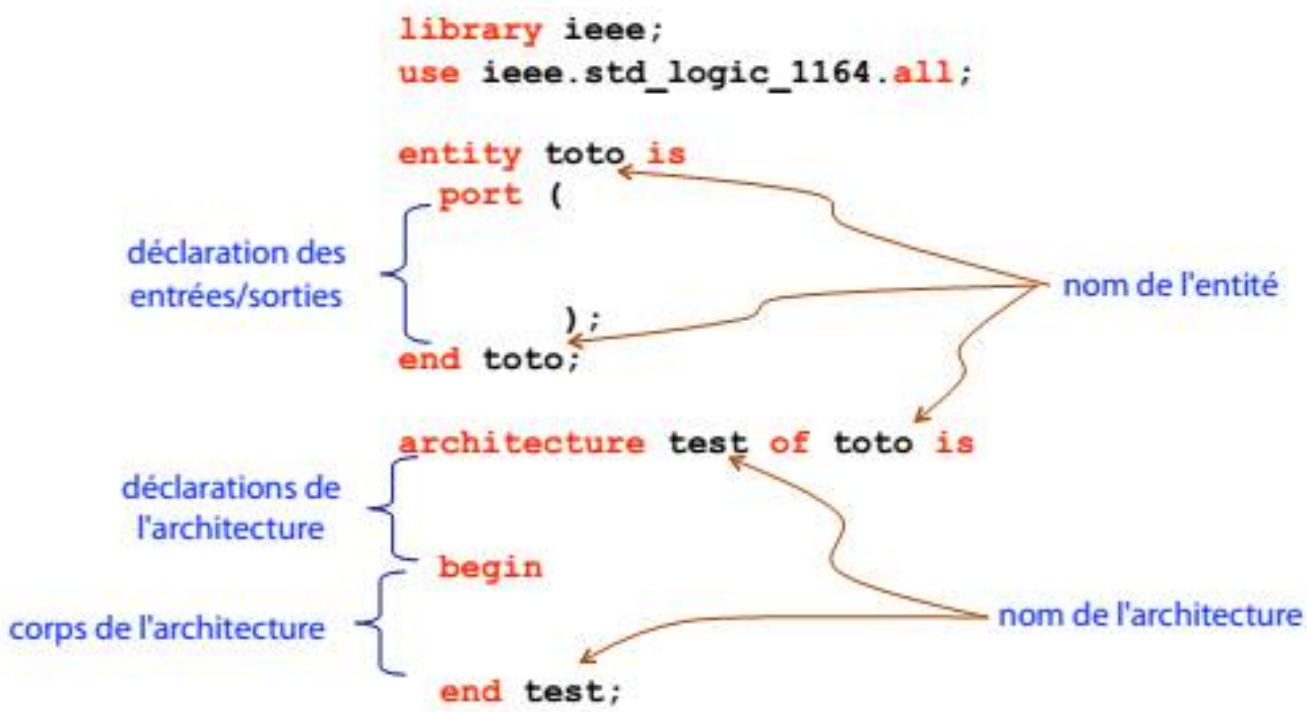


Figure: Structure d'un programme VHDL

L'entité (1/2)

- Les signaux d'entrée/sortie de l'entité sont des PORTS
- un PORT est défini par :
  - un nom :
    - Majuscules/minuscules sont équivalents
    - Le premier caractère doit être une lettre
    - Le dernier caractère ne peut être un "underscore"
    - Deux "underscores"successifs sont interdits
  - un mode (sens) décrit la direction du transfert d'une donnée à travers un port
    - in
    - out
    - inout (signaux bidirectionnels)
    - buffer (un noeud interne utilisé pour feedback)

L'entité(2/2)

- un type : Il y a plusieurs types de données disponibles
  - Boolean, std_logic, std_logic_vector, integer
  - La bibliothèque standard IEEE fournit également std_ulogic et std_logic et des tableaux de ces deux types
  - On doit signaler à VHDL d'utiliser cette bibliothèque

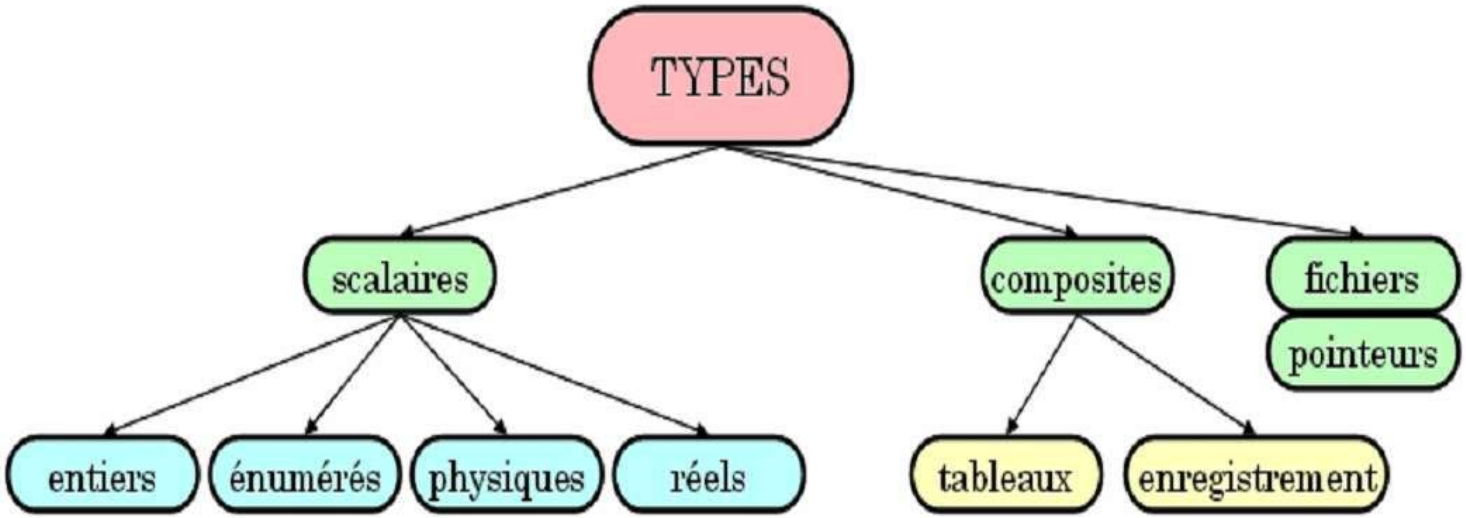
```
library ieee;
use iee.std_logic_1164.all;
```

Les types (1/2)

- Tout objet VHDL doit être associé à un type (objet = signal, constante ou variable)
- Un type définit :
  - L'ensemble des valeurs possibles
  - L'ensemble des opérateurs disponibles
- Organisation des types en VHDL :
  - Types prédéfinis (integer, bit, bit_vector, boolean,etc...)
  - Types complémentaires
  - IEEE1164 (std_logic, std_logic_vector)
  - Types spécifiques définis par les outils des fournisseurs
  - Types utilisateur (type énuméré, sous-type)

Les types (2/2)

Type		Classe
boolean	→	type énuméré
bit	→	type énuméré
character	→	type énuméré
integer	→	type numérique
natural	→	sous-type numérique
positive	→	sous-type numérique
string	→	chaîne de caractères
bit_vector	→	array of bit
time	→	physique



Bibliothèque IEEE-1164 standard logic

- 0 : Niveau logique bas à basse impédance (mise à la masse via une faible impédance)
- 1 : Niveau logique haut à basse impédance (mise à Vcc via une faible impédance)
- Z : Niveau logique flottant (entrée déconnectée)
- L : Niveau logique bas à haute impédance (mise à la masse via une résistance de pull-down)
- H : Niveau logique haut à haute impédance (mise à Vcc via une résistance de pull-up)
- W : Niveau logique inconnu à haute impédance (pouvant être 'L', 'Z' ou 'H')
- X : Niveau logique inconnu (pouvant être '0', 'L', 'Z', 'H' ou '1')
- U : Non défini
- - : N'importe quel niveau logique (renvoie toujours true lors d'une comparaison avec les 8 autres niveaux logiques)

Architecture

- Une architecture décrit le contenu d'une entité
- VHDL a trois styles d'architecture qui peuvent être combinée dans le corps d'une architecture
  - Comportemental
  - Flot de données
  - Structurel
- Le même circuit peut être décrit en utilisant n'importe lequel des trois styles

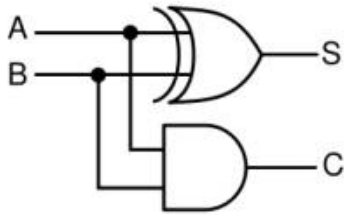
Flot de données

- Description fonctionnelle, des équations booléennes
- MUX 4-1

```
entity MUXis port(
     E0   :in std_logic;
     E1   :in std_logic;
     E2   :in std_logic;
     E3   :in std_logic;
     SEL0 :in std_logic;
     SEL1 :in std_logic;
     S    :out std_logic);
 end MUX;
 architecture FLOT_MUX of MUX is
 begin
     S <= ((not SEL0) and (not SEL1) and E0)or
            (SEL0 and( not SEL1) and E1)or
            ((not SEL0) and SEL1 and E2)or
            (SEL0 and SEL1 and E3);
 end FLOT_MUX;
```

Flot de données

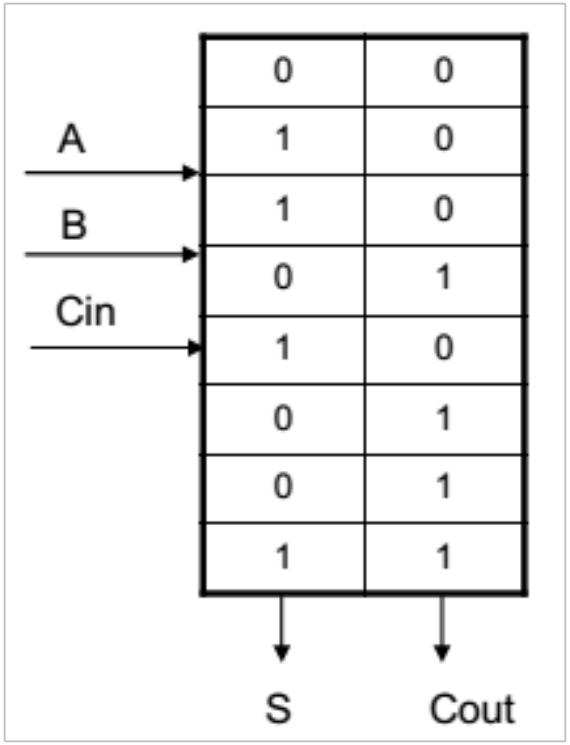
Exercice 1 : Demi addittionneur et additionneur en déscription flot de données



Input		Out
A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Input		Out
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Inputs			Outputs	
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Flot de données

- Usage de forme conditionnelles lors des affectations dans le cas du MUX 4-1

```
entity MUXis port(
      E0   :in std_logic;
      E1   :in std_logic;
      E2   :in std_logic;
      E3   :in std_logic;
      SEL0 :in std_logic;
      SEL1 :in std_logic;
      S    :out std_logic);
 end MUX;
```

Flot de données

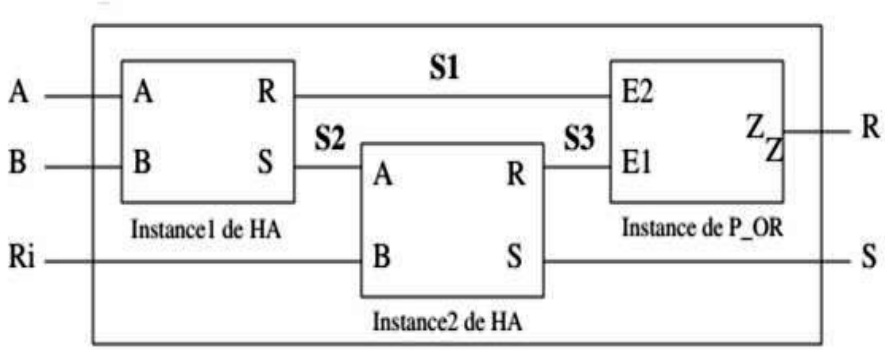
```
when expression else
  architecture FLOT_MUX of MUX is
  begin
  S<=E0 when(SEL0='0' and
       SEL1='0') else
       E1 when(SEL0='1' and
       EL1='0') else
       else
       ...
       '-';
  end FLOT_MUX;
```

Flot de données

```
with expression select
  architecture FLOT_MUX of MUX is
      begin
      with SEL1&SEL0 select
          S<=  E0 when "00",
               E1 when "01",
               E2 when "10",
               E3 when "11",
              '-' when others;
   end FLOT_MUX;
```

Structurel

- Il s'agit de l'assemblage de composants déjà décrits
- On utilise desinstancesde ces boîtes noires.
  Instancier= placer le composant dans son environnement.
- Exemple : Full adder = association de 2 HA et 1 OR



Exercice 2 : additionneur en déscription structurelle

Comportemental

- Une descripton comportementale fournit un algorithme qui modélise le fonctionnement du circuit
- Une déclaration de processus contient un algorithme
  - Elle commence par une étiquette (optionnel), le mot clé "process" et une liste des signaux actifs (sensivity list)
  - La liste des signaux actifs indique quels signaux provoqueront l'exécution du processus

```
[NOM_DU_PROCESS :]process(liste de sensibilité)
begin
-- instructions séquentielles !!
end process [NOM_DU_PROCESS];
```

IF ... THEN ... ELSE

```
if cond_bool_1 then
    instructions_1;
 elsif cond_bool_2 then
    instructions_2;
 [else instructions_3;]
 end if;
```

Exercice 3 : Mux 4 to 1

CASE ... IS WHEN

```
case expression is
   when valeur_1 =>
       instructions_1;
   when valeur_2 =>
       instructions_2;
   [when others=>
       instructions_3;]
  end case;
```

Exercice 4 : Mux 4 to 1

Exemple comparateur

```
 library ieee;
use ieee.std_logic_1164.all;
entity eqcomp4 is port(
    a: in std_logic_vector(3 downto 0);
    b: in std_logic_vector(3 downto 0);
    equals: out std_logic);
end eqcomp4;
architecture behavioral of eqcomp4 is
begin
    comp: process (a, b)
            begin
            if a = b then
                equals <= '1';
            else
                equals <= '0';
            end if;
          end process comp;
end behavioral;
```

Exercice

Exercice 5 : Bascule flip-flop(FF) avec Clock enable

Exercice 6 : Registre à décalage 8 bits à base de FF