Chapitre 2. Programmation en VHDL

- Historique du VHDL.
- Comparaison entre le VHDL et les langages de programmation.
- Différentes descriptions d'une architecture :
  - flot de données,
  - comportementalle,
  - structurelle.

Historique du VHDL

- **VHDL: VHSIC =** VHSIC (*very-high-speed integrated circuits)* **Hardware Description Language**
- Développé dans les années 80 aux États-Unis Défense américaine,
- VHDL est devenu une norme IEEE numéro 1076 en 1987.
- Révisée en 1993 pour supprimer quelques ambiguïtés et améliorer la portabilité du langage, cette norme est vite devenue un standard en matière d'outils de description de fonctions logiques.
- le langage VHDL sert à:
  - ✓ concevoir des ASICs,
  - ✓ programmer des composants programmables de type PLD, CPLD et FPGA,
  - ✓ concevoir des modèles de simulations numériques ou des bancs de tests.

Comparaison entre le VHDL et les langages de programmation

1. Objectif principal :

- **VHDL :** Principalement utilisé pour la description matérielle et la modélisation de circuits électroniques.
- **Langages de programmation généraux (comme C, Java, Python) :** Utilisés pour le développement logiciel et la résolution d'une variété de problèmes.

2. Niveau d'abstraction :

- **VHDL :** Bas niveau d'abstraction, se concentrant sur la description du matériel au niveau de la porte et du registre.
- **Langages de programmation généraux :** Plusieurs niveaux d'abstraction, permettant la programmation à des niveaux plus élevés avec des structures de contrôle, des bibliothèques et des fonctions prédéfinies.

3. Typage :

- **VHDL :** Typage statique fort, avec une spécification rigoureuse des types de données pour les signaux et les variables.
- **Langages de programmation généraux :** Typage statique ou dynamique, avec une flexibilité souvent plus grande en ce qui concerne les types de données.

4. Utilisation :

- **VHDL :** Principalement utilisé pour la conception et la vérification de circuits électroniques, simulation et synthèse.
- **Langages de programmation généraux :** Utilisés pour une gamme étendue d'applications, y compris le développement d'applications logicielles, les applications web, l'automatisation, etc.

5. Parallélisme :

•**VHDL :** Naturellement adapté pour exprimer le parallélisme inhérent aux circuits électroniques.

•**Langages de programmation généraux :** Peuvent utiliser des concepts de parallélisme, mais souvent de manière moins explicite que VHDL.

6. Concurrence :

• **VHDL :** La concurrence est une caractéristique fondamentale, avec la possibilité de décrire plusieurs processus fonctionnant simultanément.

•**Langages de programmation généraux :** La concurrence peut être gérée via des threads, des processus légers, ou d'autres mécanismes, mais c'est généralement moins direct que dans VHDL.

7. Outils associés :

- **VHDL :** Souvent utilisé avec des outils de synthèse pour générer des circuits physiques à partir des descriptions VHDL, ainsi que des simulateurs pour la vérification.
- **Langages de programmation généraux :** Utilise généralement des compilateurs et des interprètes pour exécuter le code sur une machine cible.

En résumé, VHDL est spécifiquement conçu pour la description matérielle et est utilisé dans le domaine de la conception de circuits intégrés, tandis que les langages de programmation généraux sont plus polyvalents et adaptés à une gamme plus large d'applications logicielles.

Structure d’une description VHDL

En VHDL, une structure logique est décrite à l'aide d'une entité et d'une architecture de la façon suivante :

Entité et Architecture

Entité (ENTITY):

L'entité donne les informations concernant les signaux d'entrées et de sorties de la structure ainsi que leurs noms et leurs types.

L'entité est une vue de l'extérieur du composant (interface) .

Syntaxe:

ENTITY ***Nom de l'entité*** ***IS***

Description des entrées et des sorties de la structure en explicitant pour chacune d'entre elles le nom, la direction (IN, OUT et INOUT) et le type.

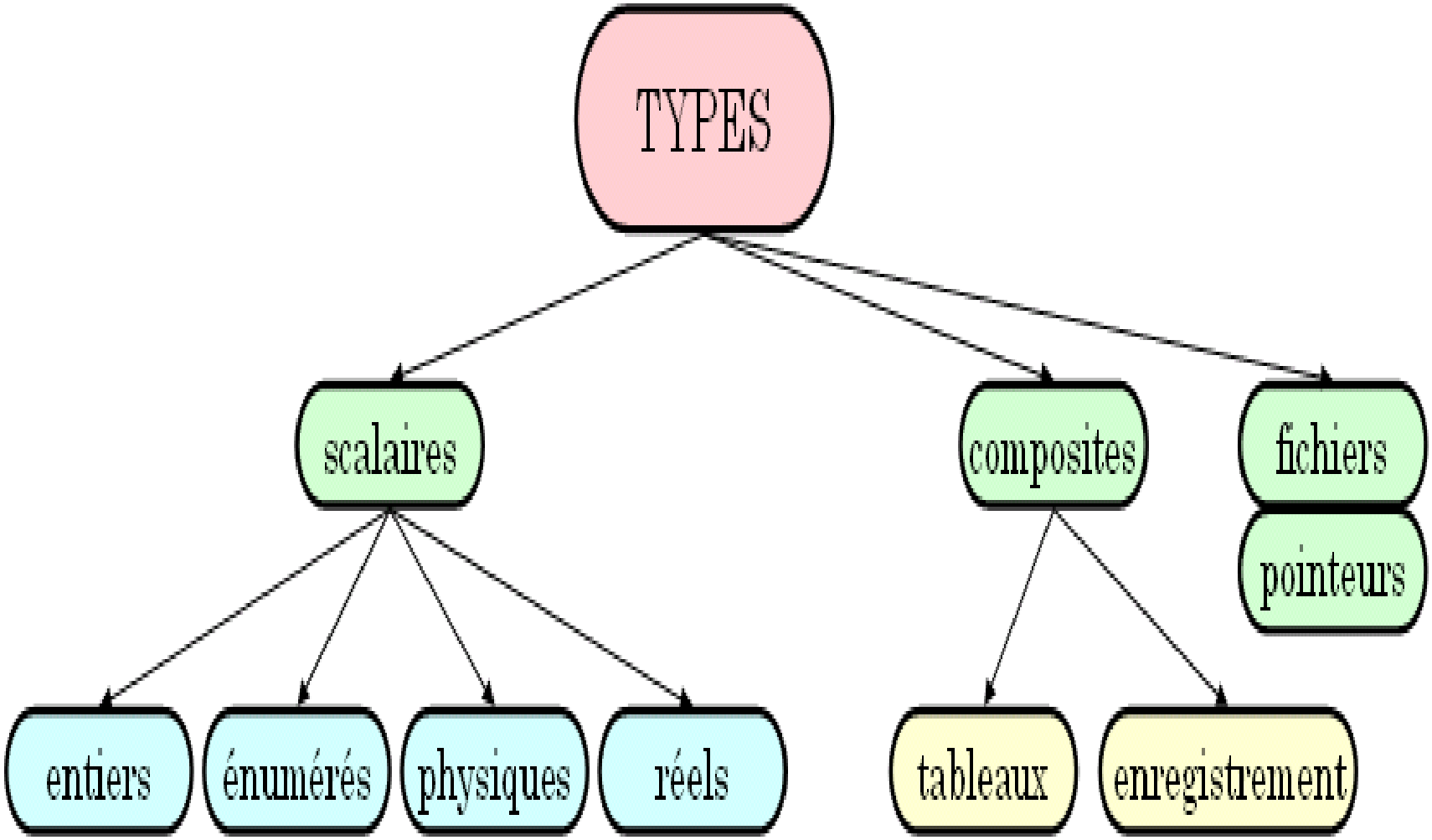
END ***Nom de l'entité ;***

Pour le type des entrées/sorties, on utilise principalement deux :

- std_logic : 1 bit
- std_logic_vector: 1 vecteur de bits

Les types

- VHDL est un langage fortement typé.
- Tous les objets définis en VHDL doivent appartenir à un type avant d'être utilisés.
- Deux objets sont compatibles s’ils ont la même définition de type.
- Un type définit l'ensemble des valeurs que peut prendre un objet ainsi que l'ensemble des opérations disponibles sur cet objet.


TYPES
scalaires
composites
fichiers
pointeurs
entiers
énumérés
physiques
réels
tableaux
enregistrement

Dans le paquetage STANDARD de la bibliothèque STD, plusieurs types énumérés sont définis :

- type boolean is (FALSE, TRUE);
- type bit is ('0', '1');
- type severity_level is (NOTE, WARNING, ERROR, FAILURE);
- type character is (NUL, SOH, STX, ETX,..., '0','1', ...);

Dans le paquetage STD_LOGIC_1164 de la bibliothèque IEEE, le type STD_ULOGIC est défini par :

- type std_ulogic is ('U', 'X, 'O', '1', 'Z', 'W', 'L', 'H', '-');

- Ceci permet d'avoir 9 états significatifs de la logique.

- Ces états illustrent les cas où le signal est soumis à de multiples affectations.

- Dans ce cas chaque valeur a un niveau de priorité.

- La liste suivante donne la signification de ces 9 valeurs en commençant par la valeur de plus grande priorité :

✓Au démarrage les signaux sont dans un état inconnu 'U'.
✓'X' indique un conflit, le signal lui est affecté d'un côté à '1' et d'un autre à '0'.
✓'0' et '1' correspondant aux valeurs booléennes du signal.
✓'Z' correspond à l'état haute 'impédance".
✓'W' est la valeur d'un signal relié à 2 résistances de tirage, une tirant à 0 (pull down) et l'autre à 1 (pull up).
✓'H' et 'L' sont des valeurs d'un signal relié respectivement à une résistance de tirage à 1 et à 0.
✓'-' est un état indifférent. Utile pour décrire les tables de vérité.

- Au niveau de la syntaxe, pour indiquer la valeur d'un bit, scalaire, on met le symbole entre apostrophes : '1'

- dans le cas d'un vecteur, on utilisera des guillemets droits : "11"

- Ces types sont définis dans la bibliothèque ieee.std_logic_1164.all (il faudra le spécifier avant chaque entité VHDL).

Il existe différentes descriptions d'une architecture :

- flot de données,
- comportementalle,
- structurelle.

Architecture:

L'architecture décrit le comportement de l'entité. l'architecture explicitera les liens entre les entrées et les sorties (implémentation).

Syntaxe:

ARCHITECTURE ***Nom de l'architecture OF Nom de l'entité IS***

Zone de déclaration.

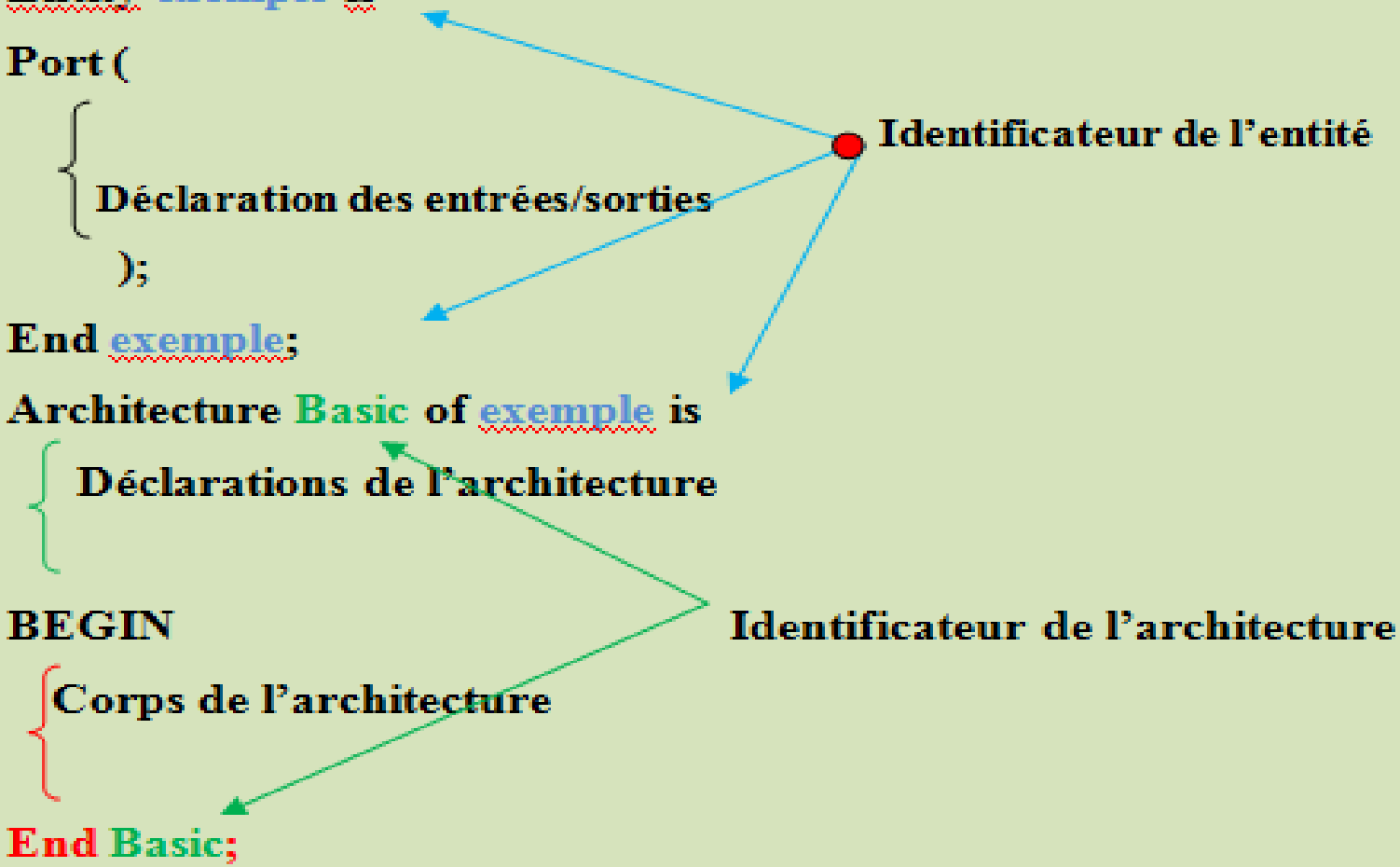
BEGIN

Description de la structure logique.

END ***nom de l'architecture ;***

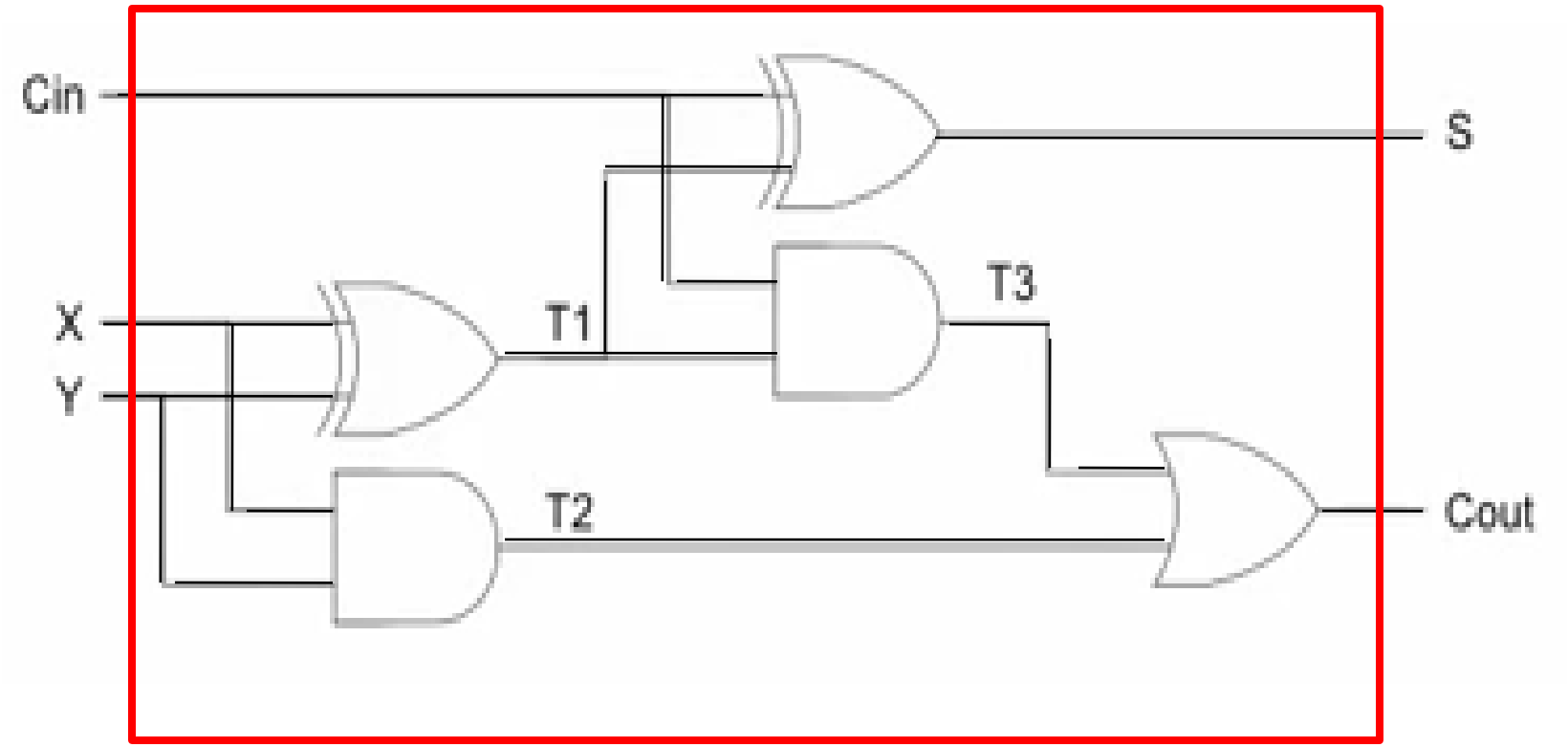
N.B: Il est possible de créer plusieurs architectures pour une même entité. Chacune de ces architectures décrira l'entité de façon différente.

Structure d'un programme VHDL



Description flot de données

Modèle VHDL d'un circuit combinatoire simple



```
library IEEE;
use  IEEE.std_logic_1164.all;
```

Librairie: définition de type, fonctions, …etc.

```
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;
```

Entité: interface avec le monde extérieur
Définit les ports, leurs types et leurs directions

```
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 T3;
end flotdedonnees;
```

Architecture: partie déclarative et corps.

Décrit le comportement du circuit.

Principe de la concurrence: l'ordre n'est pas important dans le corps de l'architecture.