Exemple VHDL

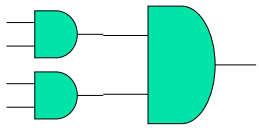
- Cahier des charges
  - on voudrait construire un circuit de commande d'un distributeur de café.
  - 4 électrovalves : eau, café, sucre et lait
  - 5 boutons
    - café sucré
    - café sans sucre
    - lait sucré
    - lait sans sucre
    - chocolat au lait (présent sur le chassis extérieur)

Table de vérité

Choix/ Val	Eau	Café	Sucre	Lait	Alar
C sucré	1	1	1	0	0
C non sucré	1	1	0	0	0
Lait sucré	1	1	1	1	0
Lait non su	1	1	O	1	0
Chocolat	0	0	0	0	1

Implantation

- Méthode classique
  - écrire les équations
  - optimiser
  - dessiner le circuit
  - choisir les composants



- Méthode actuelle
  - Rédiger le VHDL correspondant au cahier des charges
  - synthèse
  - choix technologique
  - réalisation

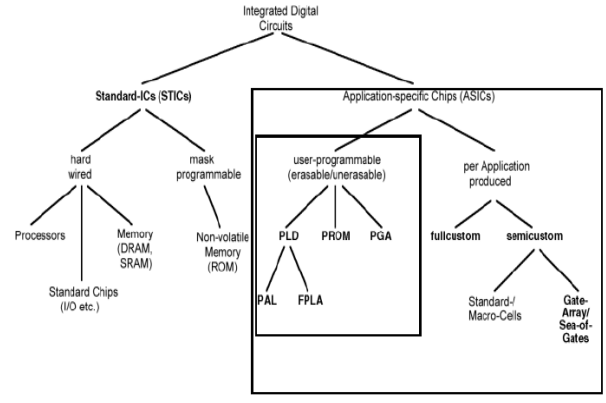
Equations puis VHDL source

- Valves
  - eau =
  - café =
  - sucre =
  - lait =

Table de vérité, équations et VHDL au tableau

Cycle de développement

- Partant du VHDL, arriver aux données technologiques
- Quelle que soit la cible de réalisation, le cycle de développement est pratiquement le même



Deux approches sont possibles pour les circuits spécifiques :

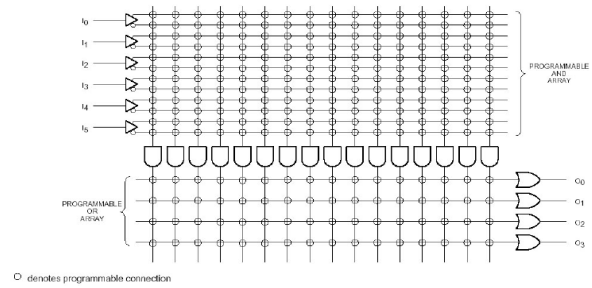
- circuits programmables ;
- circuits demandant le passage en fonderie de silicium.

Les circuits demandant de passer en fonderie ne sont rentables que pour des quantités très élevées (millions de pièces)

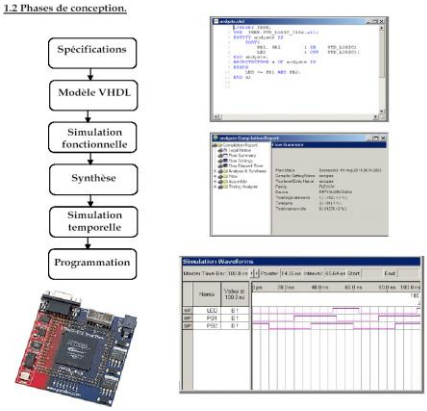
Les circuits programmables peuvent être utilisés pour des petites séries de circuits. Mis à part la densité d'intégration plus faible, un des problèmes majeur est la maîtrise des délais de propagation à l'intérieur du circuit.

1.6 Architecture interne des circuits programmables.

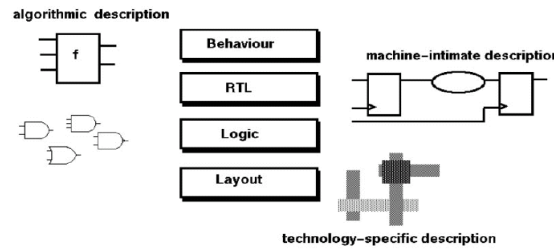
1.6.1 Programmable Logic Array (PLA)

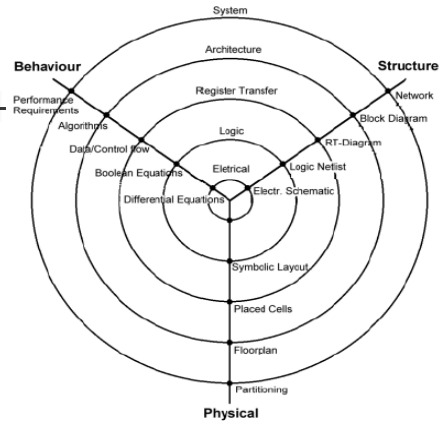


1.2 Phases de conception.



1.4 Niveaux de description.





Aspects avancés

VHDL COMPORTEMENTAL VERSUS STRUCTUREL

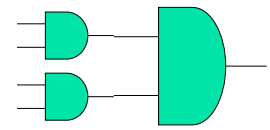
VHDL comportemental

- Dans tous les exemples précédents
  - description comportementale
    - on décrit le fonctionnement effectif de l 'opérateur

```
-- opérateur ET à 4 entrées
ENTITY et4 IS
PORT (e1,e2, e3, e4 : IN BIT ;
                s : OUT BIT ) ;
END et2 ;
ARCHITECTURE titi1 of et2 IS
BEGIN
        s <= a AND b AND c AND d ;
END titi1 ;
```


VHDL structurel

```
-- en supposant que  ET à 2 entrées déjà défini
ENTITY
même chose END et4
ARCHITECTURE titi2 of et4 IS
signal intern1, intern2 : bit ;
BEGIN
i1 : et2 PORT MAP (e1, e2, intern1) ;
-- attention à l 'ordre des
-- signaux entre les parenthèses
i2 : et2 PORT MAP (e3, e4, intern2) ;
i3 : et2 PORT MAP (intern1, intern2, s) ;
-- i1, i2, i3 les noms de composant étapes
-- qu 'on n 'a pas à déclarer
```



Comportemental vs structurel

Comportemental

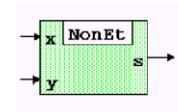
```
-- opérateur ET à 4 entrées
ENTITY et4 IS
PORT (a,b, c, d : IN BIT ;
                s : OUT BIT ) ;
END et2 ;
ARCHITECTURE titi1 of et2 IS
BEGIN
        s <= a AND b AND c AND d ;
END titi1 ;
```

- *Structurel*
- ENTITY

```
même chose END et4
ARCHITECTURE titi2 of et4 IS
signal intern1, intern2 : bit ;
BEGIN
i1 : et2 PORT MAP (a, b, intern1) ;
-- attention à l 'ordre des
-- signaux entre les parenthèses
i2 : et2 PORT MAP (c, d, intern2) ;
i3 : et2 PORT MAP (intern1, intern2, s) ;
-- i1, i2, i3 les noms de composant
   étapes
-- qu 'on n 'a pas à déclarer
```


Description comportementale

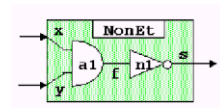
s <= not(x and y);



Description structurelle

a1: et PORT MAP (x,y,f);

n1: inv PORT MAP (f,s);



- **Behaviour (comportement) : décrit ce qui se passe (diagrammes temporels, d'états…, relations entre les entrées et les sorties)**
- **Structure : décrit de manière symbolique quels sont les composants et leurs interconnexions.**
- **Physical : indique où les éléments sont localisés.**

Pour chaque type de description, plusieurs niveaux d'abstraction sont possibles.

Le langage VHDL permet une description comportementale et structurelle

- **Description comportementale abstraite : algorithme.**

  Exemple : un compteur est un circuit dont les sorties sont incrémentées à chaque flanc montant de l'horloge.

- **Description comportementale de bas niveau : équations logiques.**

  Exemple : un additionneur est un circuit dont la sortie est égale au « ou exclusif » des entrées.

Choix description Comportementale ou Structurelle

	Comportementale	**Structurelle**
Méthode	Décrire la fonction en utilisant les opérateurs de base + opérateurs de la bibliothèque (système et utilisateur)	Décrire en utilisant des opérateurs encapsulés (nombre déterminé de lignes E/S) bibliothèque système et utilisateur
Contraintes	peu	plus importantes
Lisibilité	plutôt bonne sous réserve du 'style '	limitée lecture d'un schéma, sans graphique
Synthèse	Globale, plus efficace, plus optimisée, mais peu de garanties sur les délais de propagation	plus déterministe, pas de liberté d'optimisation au synthétiseur, délais de propagation garantis, possibilité d'implanter les opérateurs de base
Utilisation		• traduction de schéma électrique • garantir les délais de propa

Types logique standard

type std_logic

valeurs : '0', '1', 'Z', 'X'

-- exemple de déclarations de broches et de signaux

```
PORT(... e:IN std_logic;... s:OUT std_logic);
SIGNAL s1, s2 : std_logic;
```

type std_logic_vector

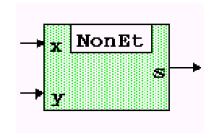
-- exemple de déclarations de broches et de signaux

```
PORT(... A:IN std_logic_vector(7 DOWNTO 0);...);
SIGNAL reg : std_logic_vector(3 DOWNTO 0);

accès     reg<=('0','0','0','1');
          ou encore reg<="0001"; reg(2)<='0'; reg(3)<=A(6);
```



```
-- usage de la bibliothèque standard
LIBRARY IEEE; USE IEEE.std_logic_1164.all ;
ENTITY NonEt IS   -- broches d'entrée et de sortie
        PORT (x,y: IN std_logic; s: OUT std_logic);
END;
ARCHITECTURE comportement OF NonEt IS
        -- description de la fonction réalisée
        s <= not(x and y);
END comportement;
```



Les valeurs logiques portées par les signaux (broches ou fils internes) sont définies par les types logiques standard de la bibliothèque IEEE :

std_logic : bit simple, qui peut prendre 4 valeurs notées '0', '1', 'Z' et 'X' qui représentent respectivement le bit 0, le bit 1, la déconnexion (signal non forcé, ou encore laissé en "haute impédance" selon la terminologie électronique), et une valeur indéterminée.

std_logic_vector : vecteur de bits. Chaque élément est de type std_logic. Les bornes d'indice d'un vecteur sont indiquées à sa déclaration, sous la forme (7 DOWNTO 0) ou encore (0 TO 7) pour un vecteur de 8 bits indicé de 0 à 7. La différence entre les deux formes ne concerne que les notations littérales de valeurs, la première plaçant le bit 7 à gauche et la deuxième à droite.

L'accès à l'élément d'indice i d'un tableau tab se note : tab(i)

On peut dénoter une valeur de tableau en énumérant ses composante : (v1,v2,v3,v4)

Une notation littérale de la forme "0001" est acceptée pour le type std_logic_vector.

Quelques opérateurs

opérations sur bits et vecteur de bits : types std_logic et std_logic_vector
opérations logiques : and, or, not, xor
concaténation : &

exemple : "1101"&"011" vaut "1101011"

opérations booléennes : type boolean
valeurs : false, true
opérations logiques : and, or, not, xor

opérations arithmétiques : type integer
opérations arithmétiques : +, -, *, /

comparaisons : opérandes std_logic, integer...
résultat boolean

comparaisons : =,/=, <, <=, >, >=

Les types std_logic et std_logic_vector disposent des opérations usuelles sur bits : and (et), or (ou), not (non), xor (ou exclusif).

Pour des vecteurs de bits elles signifient les opérations bit à bit.

L'opération de concaténation, notée &, permet de construire des vecteurs à partie de vecteurs plus petits ou de simples bits.

Pour exprimer des conditions logiques, le langage possède le type boolean (booléen).

Ses valeurs sont false (faux) et true (vrai). Le type booléen dispose des opérations logiques usuelles : and, or, not, xor.

Les nombres entiers sont offerts par le type integer, doté des opérations arithmétiquesusuelles.

Les opérations de comparaison, notées =, /=, <, <=, >, >=, rendent un résultat booléen. Elles sont définies pour tous les types scalaires (std_logic ,entiers, caractères, flottant...).