Chapitre 4 :
Architecture des circuits logiques programmables

Justine Philippe

ISEN | école d'ingénieurs

Sommaire

- Introduction
- Types d'architectures
- Familles de circuits logiques programmables
- Déroulement d'une conception

Sommaire

- Introduction
- Types d'architectures
- Familles de circuits logiques programmables
- Déroulement d'une conception

Concept

Qu'est-ce qu'un circuit logique programmable ?

- **Circuit intégré numérique**, qui réalise une ou plusieurs fonctions logiques (ex : $r_s = a.b + r_e.(a \oplus b)$)
- La fonction logique réalisée n'est pas définie lors de la fabrication du circuit, mais elle est déterminée par **programmation de l'architecture du circuit**, par l'utilisateur.
- Afin de comprendre en quoi consiste cette programmation, il est nécessaire de connaître et de comprendre l'architecture de ces circuits.

Architectures pour l'implémentation de la logique combinatoire

Tout circuit logique combinatoire peut être décrit de diverses façons :

- Table de vérité
- Tableau de Karnaugh

r_e \ ab	00	01	11	10
0	0	0	1	0
1	0	0	1	1

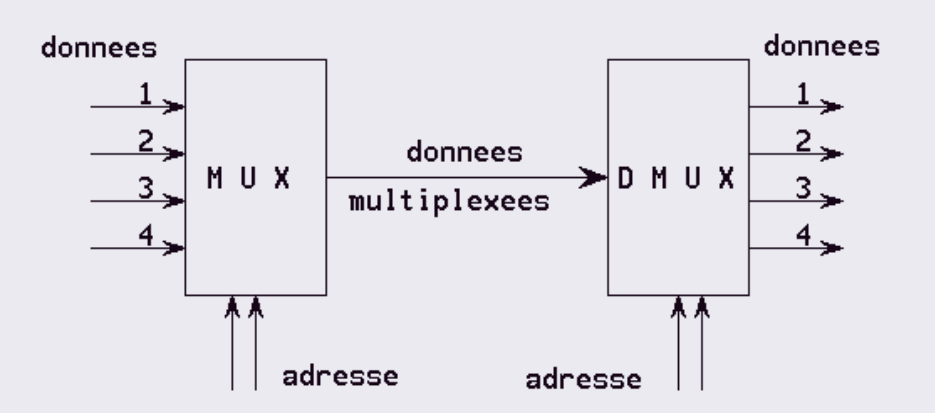
- Equation logique

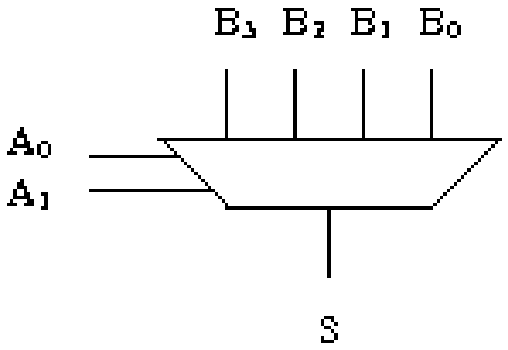
1ère forme canonique :	r_s	=	$a.b.\overline{r_e} + a.b.r_e + a.\overline{b}.r_e$
		=	$a.b + a.r_e$
2ème forme canonique	r_s	=	$(a+b+r_e).(a+b+r_e).$ $(a+b+\overline{r_e}).(a+\overline{b}+\overline{r_e})$
		=	$a + (\overline{a}+b+r_e)$
"sortie de multiplexeur"	r_s	=	$\overline{a}.\overline{b}.0 + \overline{a}.b.r_e + a\overline{b}.r_e + a.b.1$

Sommaire

- Introduction
- Types d’architectures
- Familles de circuits logiques programmables
- Déroulement d’une conception

Architecture de type multiplexeur (MUX)



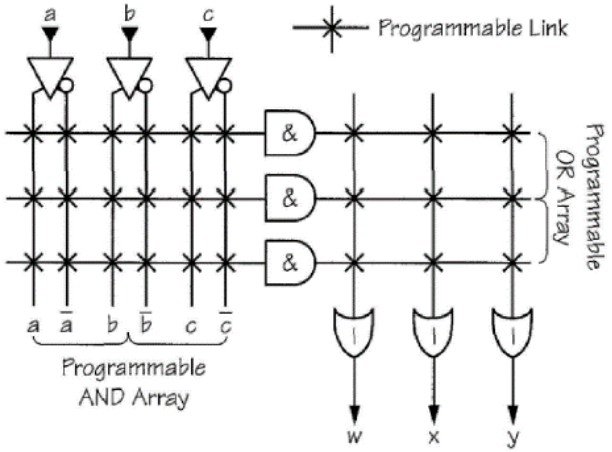


$$S = \overline{A_0}.\overline{A_1}.B_0 + \overline{A_0}.A_1.B_1 + A_0.\overline{A_1}.B_2 + A_0.A_1.B_3$$

Architecture de type PLA *(Programmable Logic Array)*

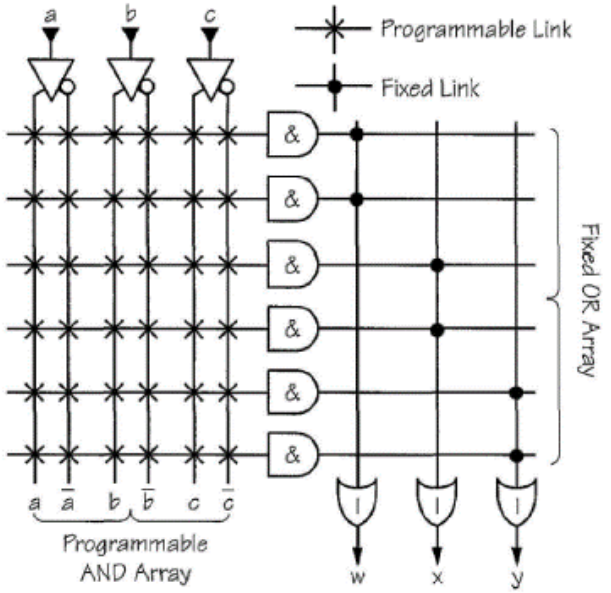
Architecture correspondant à une implémentation sous la 1ère forme canonique.

- 1ère couche : Inversion (ou pas) des variables d'entrée
- 2ème couche : Réseau de « ET » programmable
- 3ème couche : Réseau de « OU » programmable



Architecture de type PLD *(Programmable Logic Device)*

Idem que les PLA, en moins flexible : seul le réseau de « ET » est programmable. Le réseau de « OU » est fixe (non programmable).



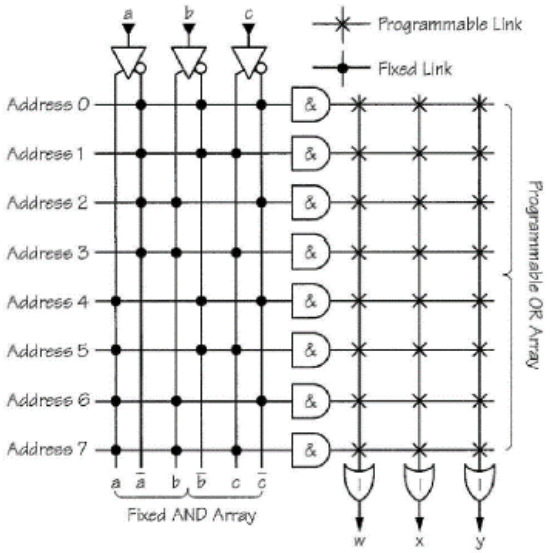
Architectures similaires :

- PAL *(Programmable Array Logic)*
- GAL *(Gate Array Logic)*
- CPLD (*Complex* PLD)
- EPLD (*Erasable* PLD)

Architecture de type LUT *(Look-Up Table)*

Et pourquoi pas l'inverse ?

- Réseau de « ET » fixe
- Réseau de « OU » programmable

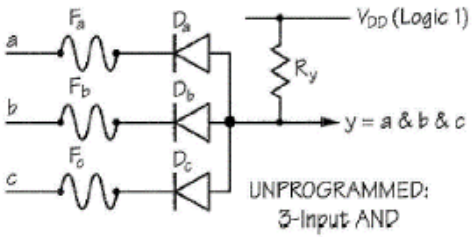


Architectures similaires :

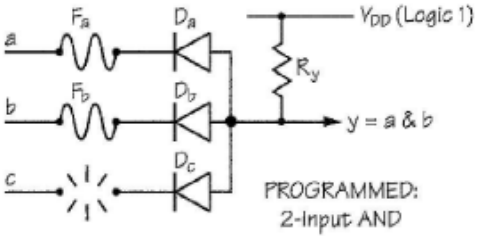
- PROM *(Programmable Read-Only Memory)*
- EPROM (*Erasable* PROM)
- EEPROM (*Electrically Erasable* PROM)
- FPGA *(Field-Programmable Gate Array)*

Principe de réalisation d'une connexion programmable

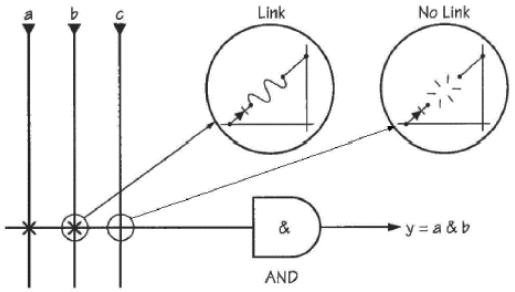
Porte ET non programmée :



Porte ET programmée :



Convention de représentation :



Quelles technologies ?

- Fusible, Antifusible (OTP)
- EPROM (UV-PROM), EEPROM, FLASH
- SRAM, etc…

Implémentation de la logique séquentielle

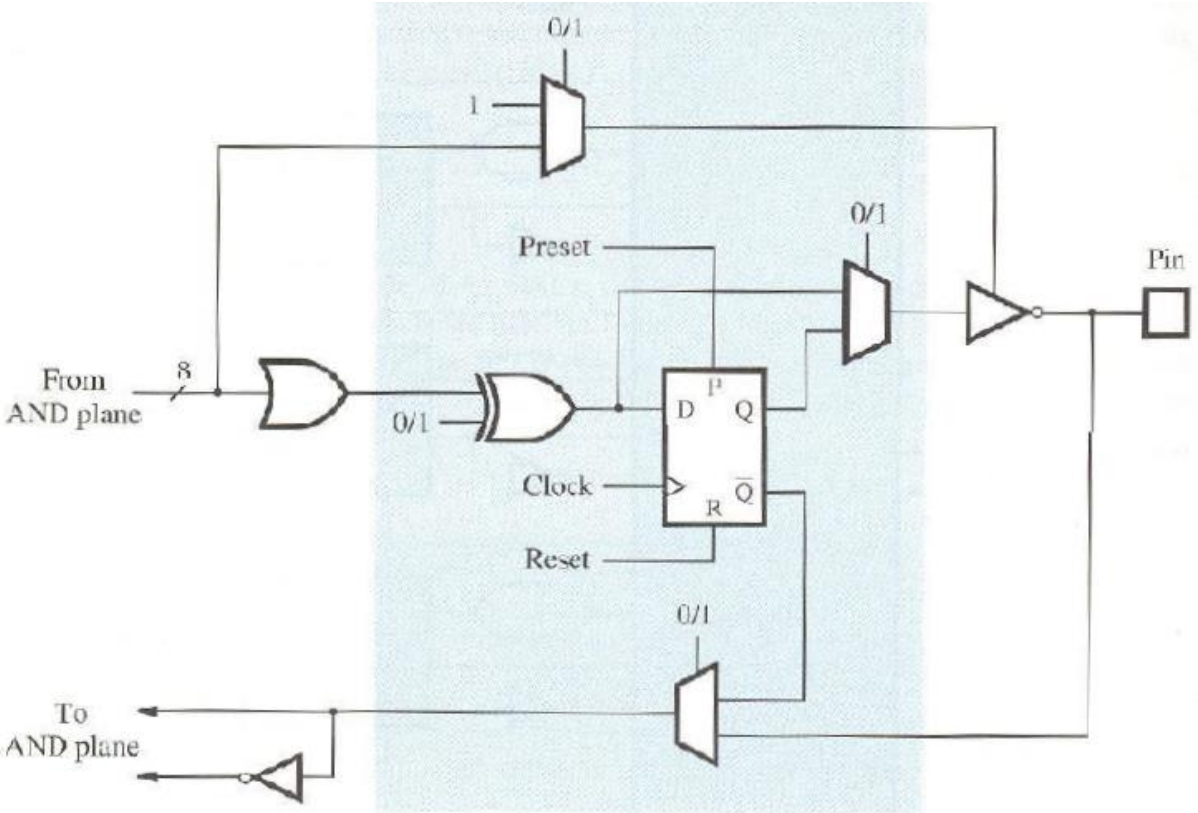
Les architectures précédentes permettent d'implémenter des fonctions logiques combinatoires.

Qu'en est-il pour la logique séquentielle ?

- **Logique séquentielle asynchrone :** Il suffit de faire un bouclage de la sortie du module combinatoire vers une de ses entrées.

- **Logique séquentielle synchrone :** Ajout d'une bascule de synchronisation en sortie du module combinatoire (le plus souvent une bascule D) → *cellule de sortie, ou macrocellule.*

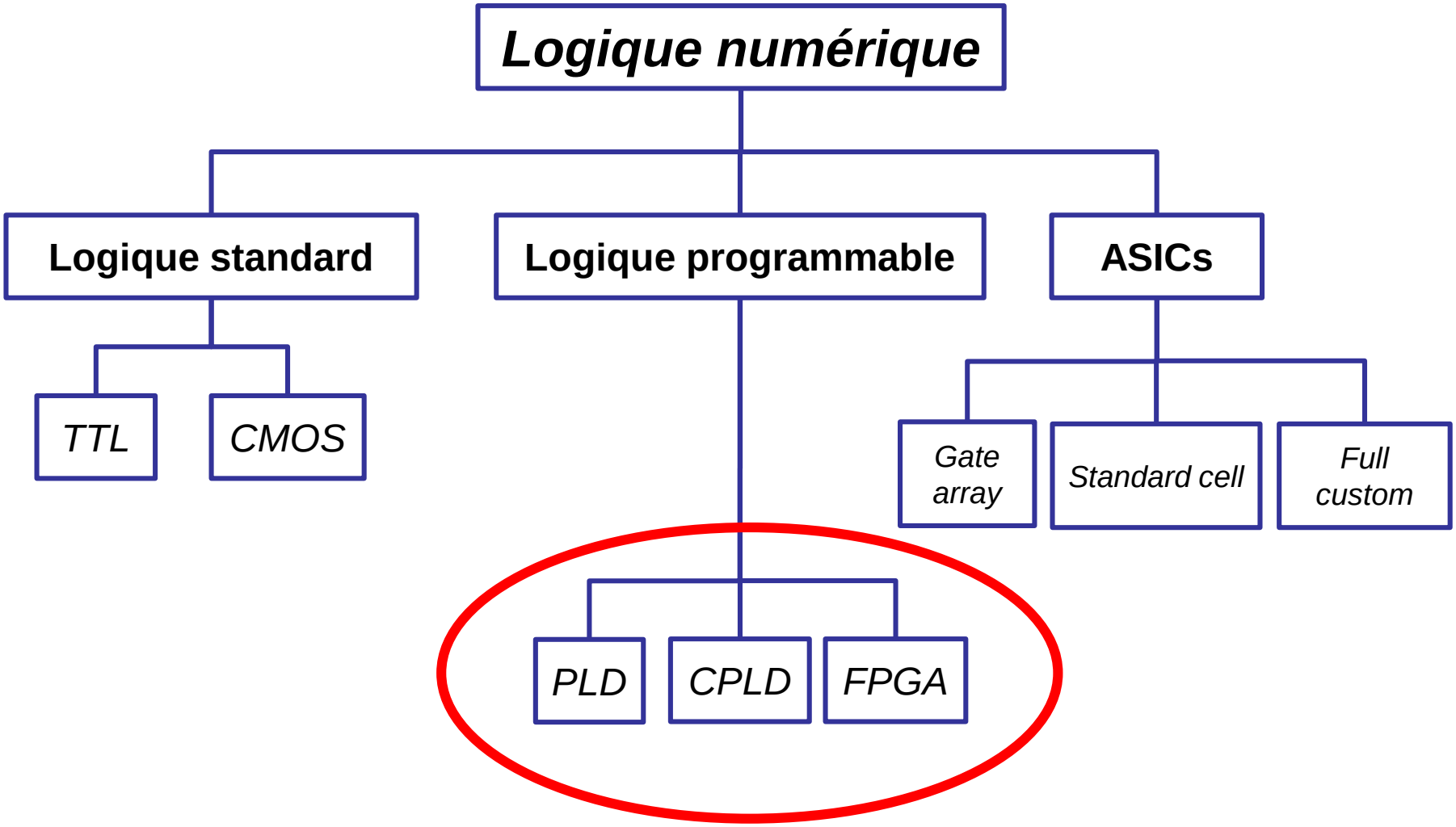
Cellule de sortie – Exemple du circuit PAL22V10



Sommaire

- Introduction
- Types d'architectures
- Familles de circuits logiques programmables
- Déroulement d'une conception

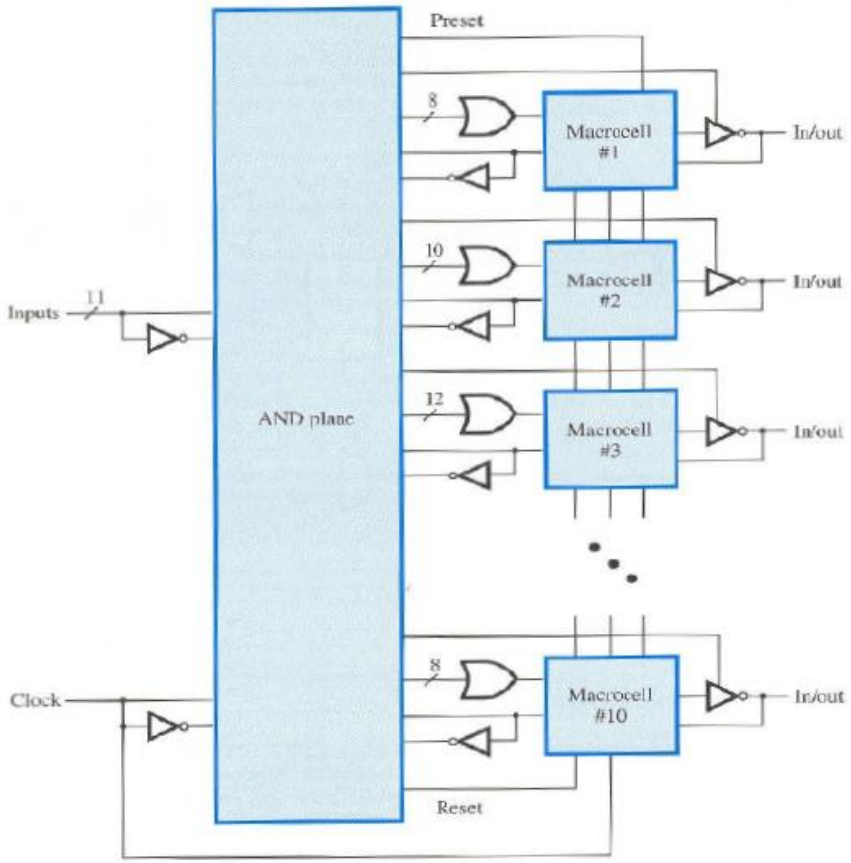
Familles de circuits logiques

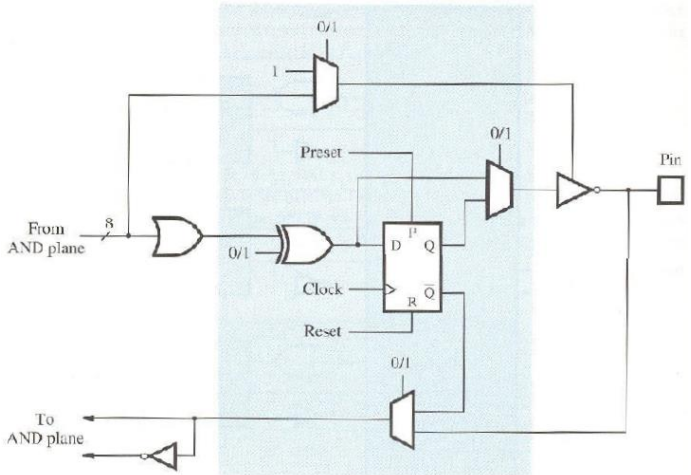


Circuits PLD *(Programmable Logic Device)*

- Introduits dans les années 1970.
- Complexité allant de quelques centaines de portes logiques à quelques dizaines de milliers.
- Architectures d'implémentation :
  - Couche d'inversion des variables d'entrée,
  - Réseau de ET programmable,
  - Réseau de OU fixe,
  - (Macro)cellule de sortie.
- Peu utilisés aujourd'hui, sauf pour des applications simples demandant beaucoup de rapidité, MAIS principe architectural repris dans les composants CPLD.

Exemple du PAL22V10





Circuits CPLD et FPGA

Date	Microprocesseur Intel	Nombre de transistors
1971	4004	2 300
1978	8086	29 000
1982	80286	275 000
1989	80486	1,16 million
1993	Pentium	3,1 millions
1995	Pentium Pro	5,5 millions
1997	Pentium II	27 millions
2001	Pentium 4	42 millions
2004	Pentium Extreme Edition	169 millions
2006	Core 2 Duo	291 millions
2006	Core 2 Quad	582 millions
2007	Dual-Core Itanium 2	1,7 milliard
2011	Core Xeon	2,27 milliards

Ces circuits sont apparus suite à l'amélioration des performances technologiques en électronique : de plus en plus de transistors par cm² de silicium !

Circuits CPLD et FPGA

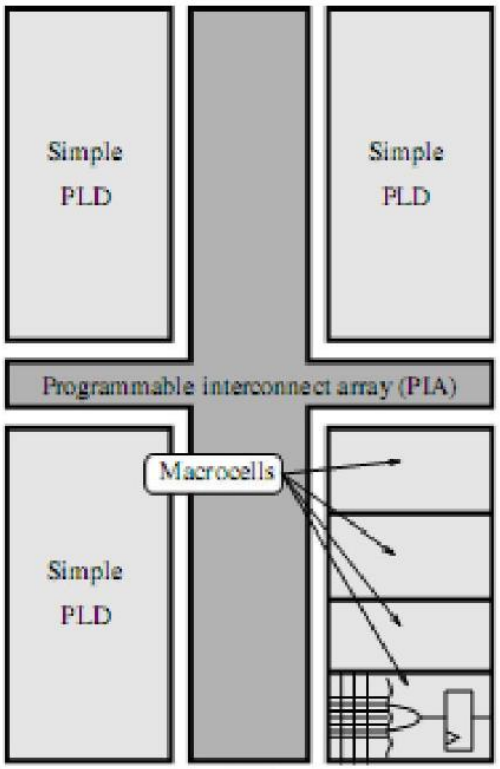
Deux familles complètement différentes :

- **CPLD (Complex PLD) :** Comme son nom l'indique, consiste en un regroupement de plusieurs PLD au sein d'un seul composant, par le biais d'une matrice d'interconnexion centralisée → *Granularité élevée.*

- **FPGA (Field Programmable Gate Array) :** Consiste en un regroupement d'une multitude de petits modules logiques programmables, interconnectés entre eux par le biais d'une matrice d'interconnexion diffuse → *Granularité moyenne (le terme « granularité fine » est réservé aux composants ASIC).*

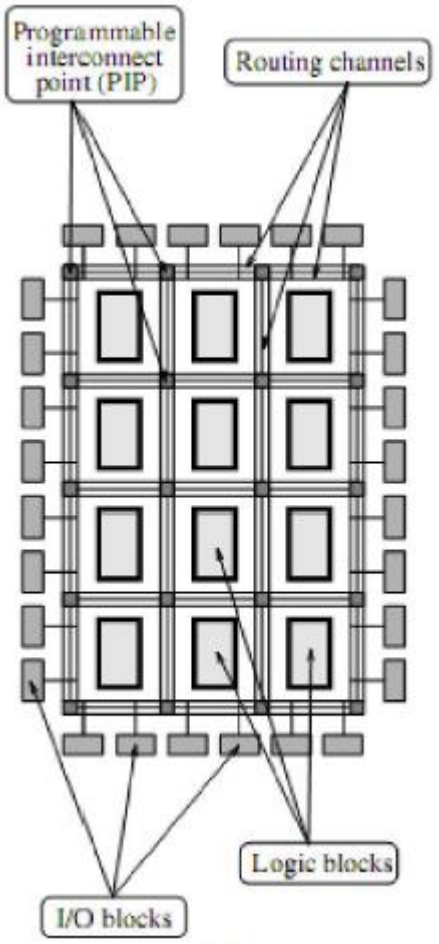
Circuits CPLD

- **Blocs logiques élémentaires :**
  - PLDs simples (type PAL)
  - Complexité de chaque bloc : en moyenne, 8 à 10 entrées et 3 à 4 sorties, 20 termes produits.
- **Blocs d'entrée/sortie** (non représenté sur le schéma) :
  - Blocs dédiés aux entrées d'horloge, de *reset* et de *set*
  - Blocs génériques d'entrée/sortie : configurable en entrée ou en sortie
- **Matrice d'interconnexion programmable :**
  - Larges bus → délais courts
  - Routage « systématique » → délais prédictibles



Circuits FPGA

- **Blocs logiques élémentaires :**
  - LUT ou MUX, suivi d'une bascule
  - Exemple du composant Virtex (Xilinx) : LUT à 4 ou 5 bits d'entrée, 1 ou 2 sorties
  - Exemple du composant ACT-2 (Actel) : MUX
- **Blocs d'entrée/sortie** → idem CPLD, avec plus d'options (ex : PLL)
- **Matrice d'interconnexion programmable :**
  - Compromis flexibilité/exploitation du silicium
  - Hiérarchie : trajets rapides pour traverser le composant, connexions plus lentes entre blocs voisins → délais plus difficilement prédictibles, et surtout *fonction du taux d'utilisation du composant.*
- **Autres ressources :** mémoires, fonctions DSP (ex : multiplicateur « câblé » 18x18), processeurs embarqués *(soft-core,* ou *hard-core).*



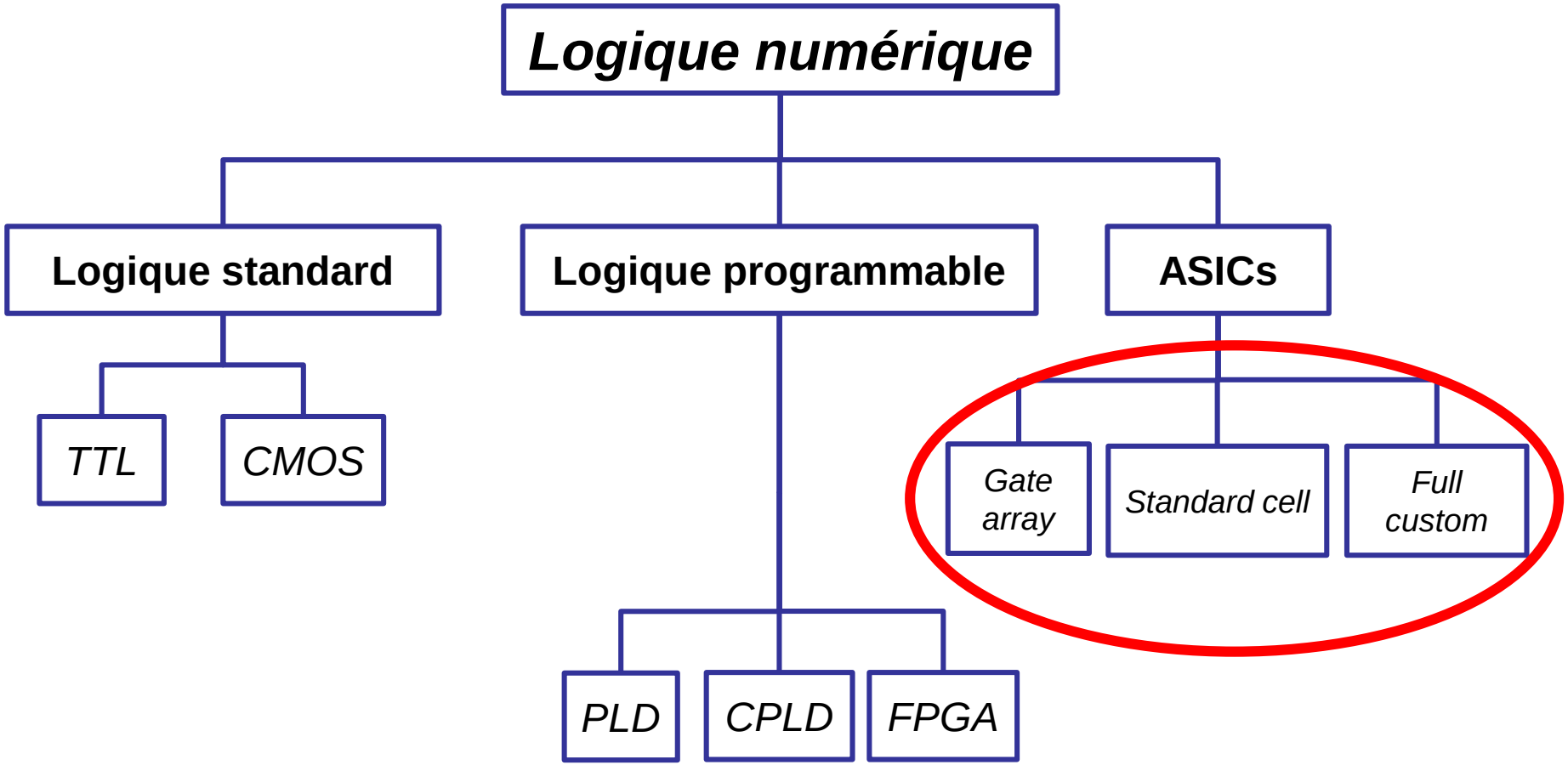
Composant FPGA vs composant DSP

- **DSP :**
  - Architecture matérielle permettant d'effectuer une opération MAC *(Multiplicate And Accumulate)* en un cycle d'horloge.
  - +++ pour l'implémentation d'algorithmes de traitement du signal, qui utilisent largement cette fonction.
  - Mais : un FPGA permet d'implémenter plusieurs opérations MAC en parallèle.

- **FPGA : L'implémentation dans un FPGA a été prouvée plus performante pour :**
  - L'algorithme CORDIC *(COordinate Rotation Digital Computer)* : calcul de fonctions (exp, log, trigonométrique, racine carrée, multiplication, division) sans multiplicateur *hard*.
  - Algorithme NTT *(Number Theory Transform)* → cryptage, signature numérique.
  - Algorithme de corrections/détection d'erreur.

- **Mode de répartition généralement utilisé :**
  - Algorithmes compliqués (ex : plusieurs instructions *if-then-else*) ayant une exigence temporelle moyenne : DSP.
  - Applications *front-end* (ex : filtres numériques, FFT, CORDIC…) : FPGA.

Familles de circuits logiques



Composants ASICs

ASIC : *Application Specific Integrated Circuit*

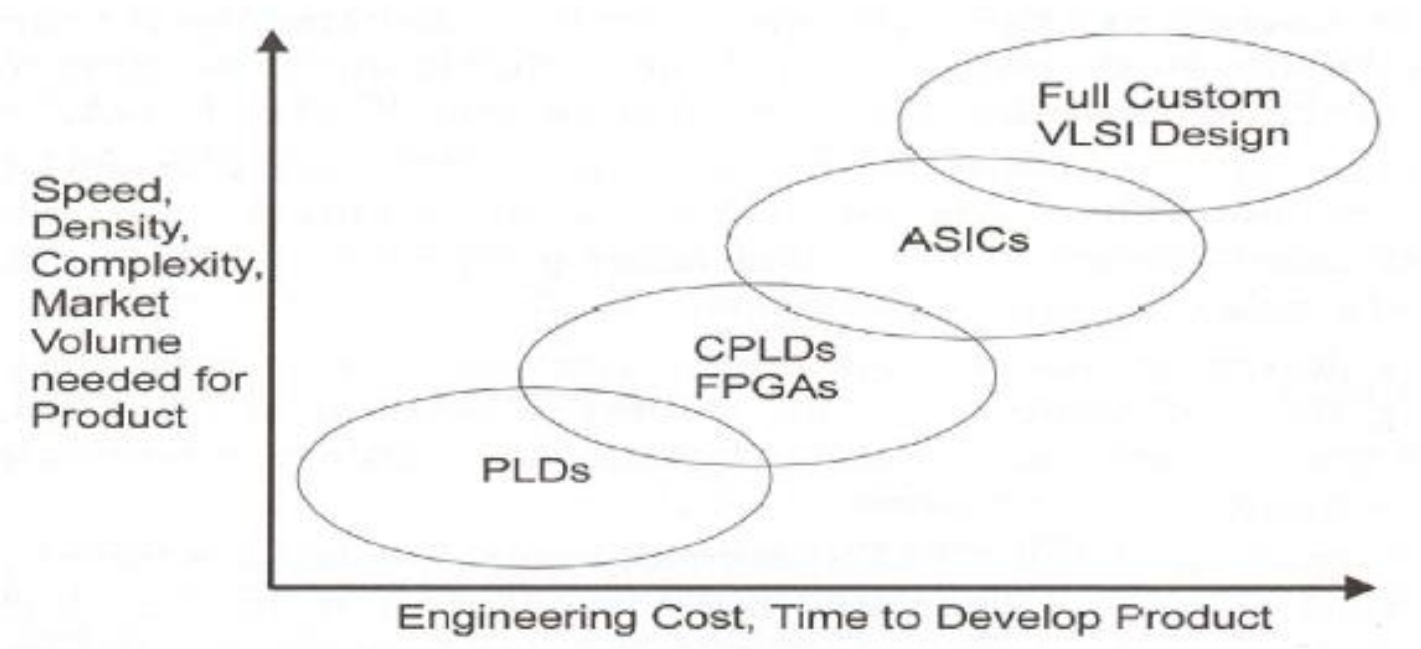
- Fonctionnalité de l'ASIC définie au moment de sa **fabrication**.
- MAIS : cette fonctionnalité est définie par l'utilisateur (et pas par le fabricant du circuit) :
  - Dimensionnement et placement des blocs logiques sur le silicium, routage.
  - Meilleures performances que les composants programmables.
  - Très long à mettre au point, très coûteux → réservé aux grandes séries

Catégories d'ASICs

- 3 catégories de composants ASICs
- Une catégorie correspond à un compromis performance/complexité, temps de développement

ASIC à la demande (Full custom)	ASIC prédiffusé (Gate array)	ASIC précaractérisé (Standard cell)
Liberté totale dans la conception : • Dimensionnement et placement des transistors • Routage	Nébuleuse de transistors et de résistances à interconnecter → conception = routage	Bibliothèque de fonctions disponibles → fonctions simples et avancées (mémoires, fonctions de communications, processeurs, IP…)
Très performant, très long à mettre au point, très coûteux	Il existe des catalogues de ressources (portes, circuits logiques de base)	Catalogues de ressources (portes, circuits logiques de base)
Applications : mémoires, microprocesseurs, microcontrôleurs, DSP	Développement relativement rapide, pas d'utilisation optimale du silicium	Développement rapide, pas d'utilisation optimale du silicium

Performances relatives des circuits

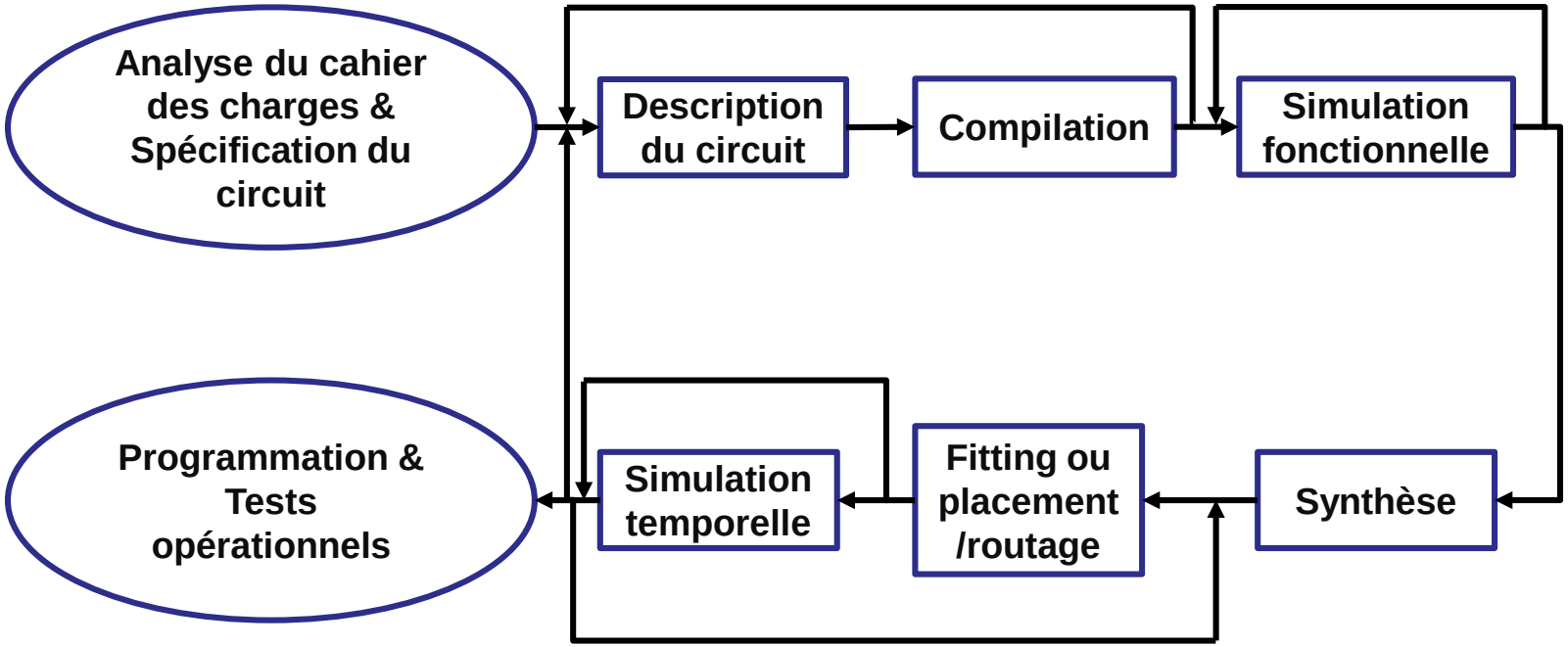


Sommaire

- Introduction
- Types d'architectures
- Familles de circuits logiques programmables
- Déroulement d'une conception

Déroulement d'une conception

Quelles sont les différentes étapes à réaliser afin de concevoir un circuit logique à partir d'un composant programmable de type PLD, CPLD ou FPGA ?



Fin du chapitre 4