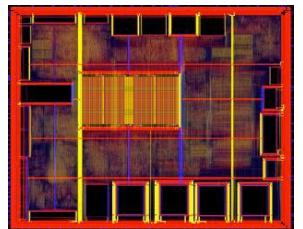




# SoC (System on Chip) Systèmes sur puce

Nadia Khouja Saad  
Hanen Ben Fradj

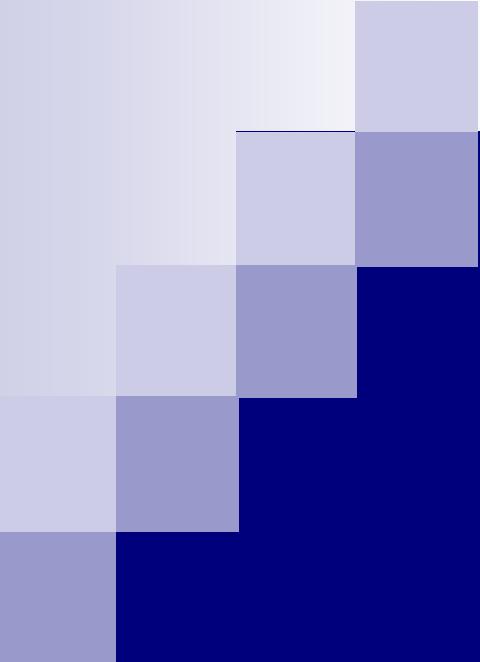


**2<sup>ème</sup> Année Parcours Systèmes Embarqués**

Année Universitaire 2008-2009

# Plan du cours

1. **Chapitre 1 : Introduction aux Systèmes sur Puce**
2. **Chapitre 2 : Etude de cibles numériques matérielles : Les FPGA**
3. **Chapitre 3 : Etude de cibles numériques logicielles : Les microcontrôleurs**



# Chapitre 1 : Introduction aux SoCs

Nadia Khouja Saad

**2ème Année Parcours Systèmes Embarqués**

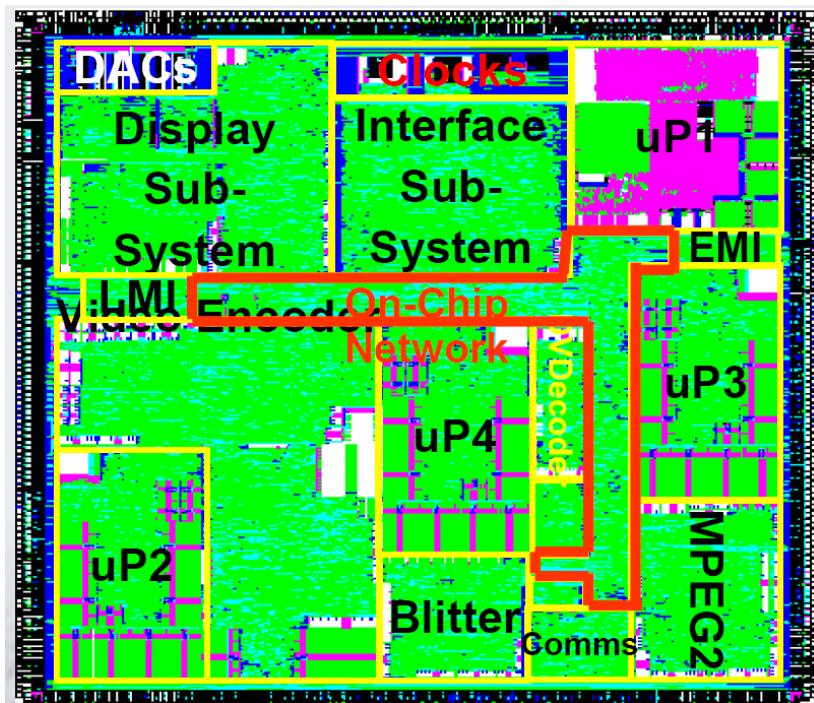
Année Universitaire 2008-2009

# Plan du Chapitre 1

1. **Les systèmes sur puce** : définition, architecture et contraintes de conception
2. **Les cibles numériques logicielles** : définition, architecture et contraintes de conception
3. **Les cibles numériques matérielles** : définition, architectures, cas d'utilisation
4. **Les cibles numériques mixtes** : définition, architectures, cas d'utilisation

# Système mono-puce : c'est quoi ?

- Système mono-puce = System on a Chip (SoC)
  - Système électronique complet intégré dans une puce



- Circuit STM8000 (STMicro)
- Décodeur multi-standards pour lecteurs DVD (audio+vidéo)
- Intègre processseurs, RAM, coprocesseurs dédiés, etc.
- Circuit mixte = **numérique + analogique**

# Système mono-puce : c'est quoi ?

## Définitions:

- **Un système embarqué** est un système électronique intégré dans un système ( voiture, avion, ..... ) qui sert à exécuter une tâche particulière ( contrôle, communication, ....).
- **Un système enfoui** peut être défini comme un système électronique et informatique autonome ne possédant pas des entrées/sorties standards comme un clavier ou un écran d'ordinateur.
- Tous système numérique autonome constitué de parties organisées pour assurer une fonction ou un ensemble de fonctions dans son environnement.
- N'importe quel système numérique qui n'est pas un PC/station de travail/serveur.
- Calculateurs dédiés à quelques applications/fonctions enfouis dans un appareil. Généralement pas de clavier ni d'affichage.

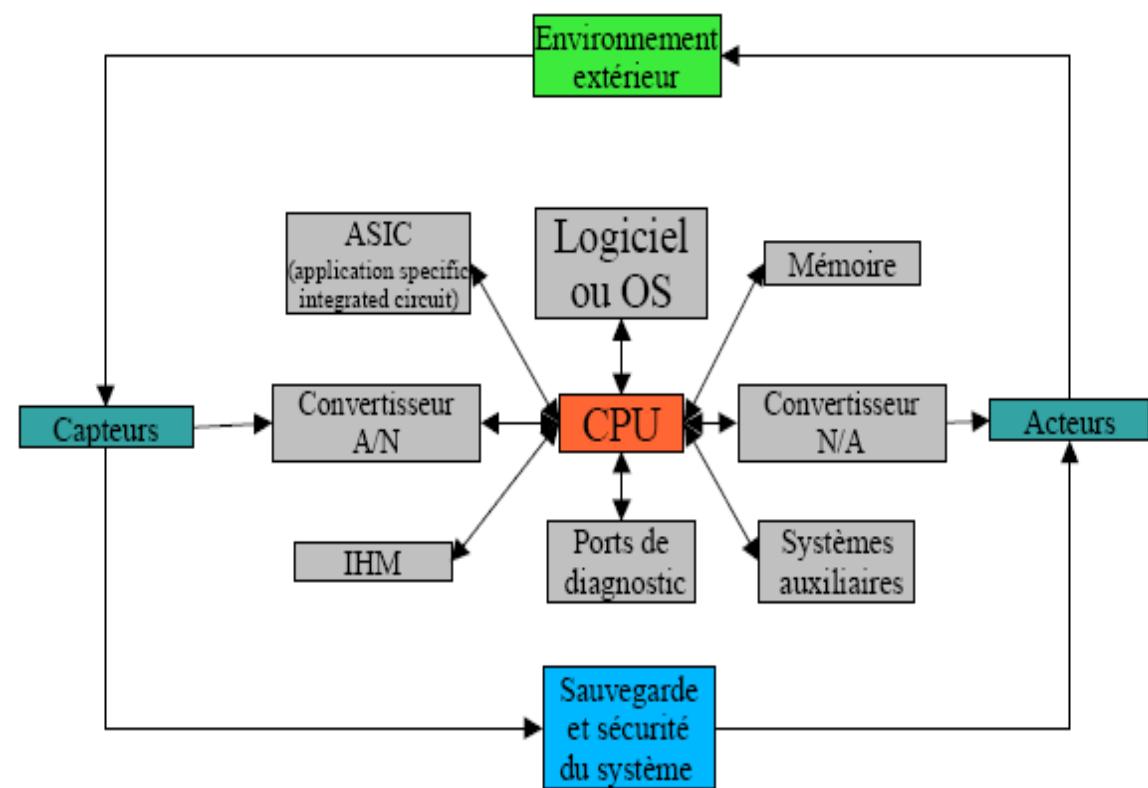
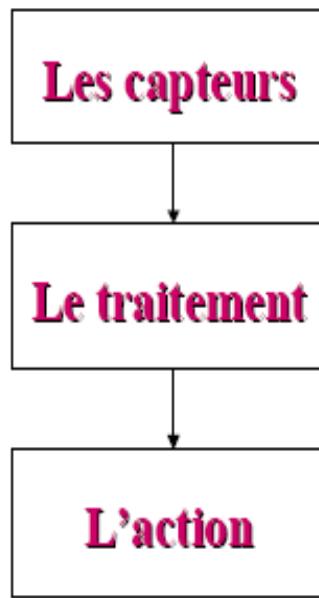
# Système mono-puce : c'est quoi ?

Un **système embarqué** :

- Est un système numérique.
- Utilise généralement un processeur.
- Exécute un logiciel dédié pour réaliser une fonctionnalité précise.
- Remplace souvent des composants électromécaniques.
- N'a pas réellement de clavier standard (BP, clavier matriciel...).
- L'affichage est limité (écran LCD ...) ou n'existe pas du tout.
- N'est pas un PC.

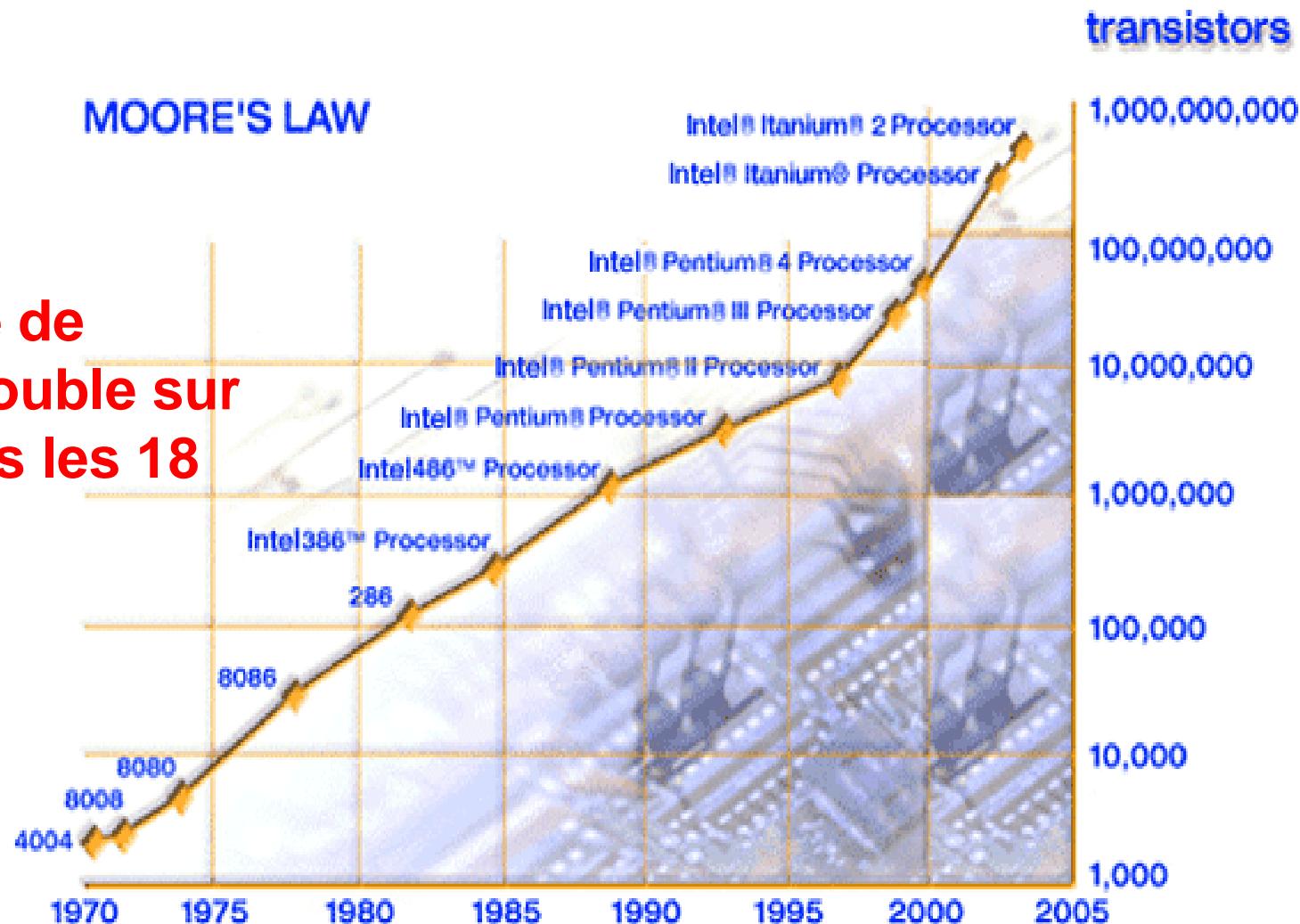
# Caractéristique et organisation

La manière de vivre d'un système embarqué



# Système mono-puce : pourquoi ?

→ Le nombre de transistors double sur une puce tous les 18 mois



# Pourquoi on s'intéressent au SE?



*«Les systèmes embarqués nous entourent et nous envahissent littéralement, fidèles au poste et prêts à nous rendre service. Ils sont donc partout, discrets, efficaces dédiés à ce à quoi ils sont destinés. Omniprésents, ils le sont déjà et le seront de plus en plus»*



Patrice Kadionik de l'ENSEIRB

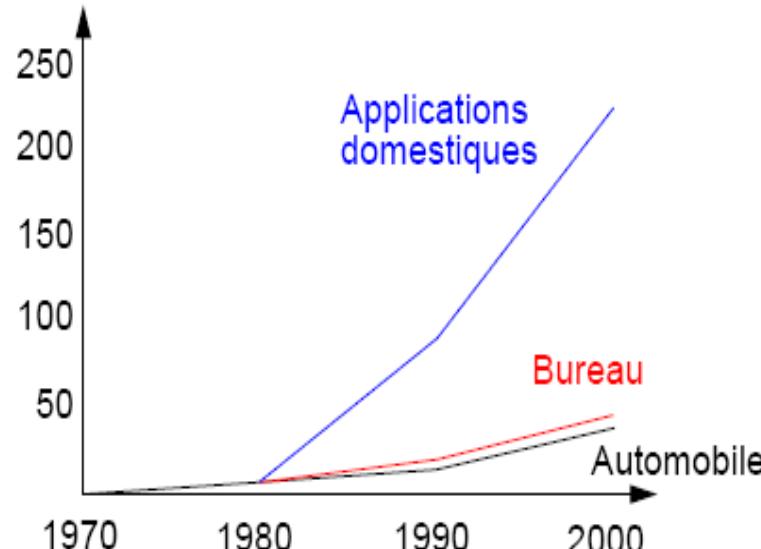
Introduction aux systèmes sur puce



# Pourquoi on s'intéressent au SE?

Nombre moyen de processeurs enfouis

98% des processeurs vendus en 1998 équipent des systèmes enfouis



[G. De Micheli]

## Applications domestiques

Téléphone/Fax  
Répondeurs  
Sécurité  
Téléphone mobile  
Portes garage  
TV, set top box  
Jeux vdeo  
Camescopes  
Magnetoscopes  
Instruments musique  
Jouets  
Réfrigérateurs/Fours

## Bureau

Téléphones  
PC  
Fax  
Sécurité  
Imprimantes  
Photocopieurs  
PDA

## Automobile

Calculateur de bord  
Airbags  
ABS  
Sécurité  
Transmission  
Climatisation  
GPS  
Injection

Estimations pour l'année 2003

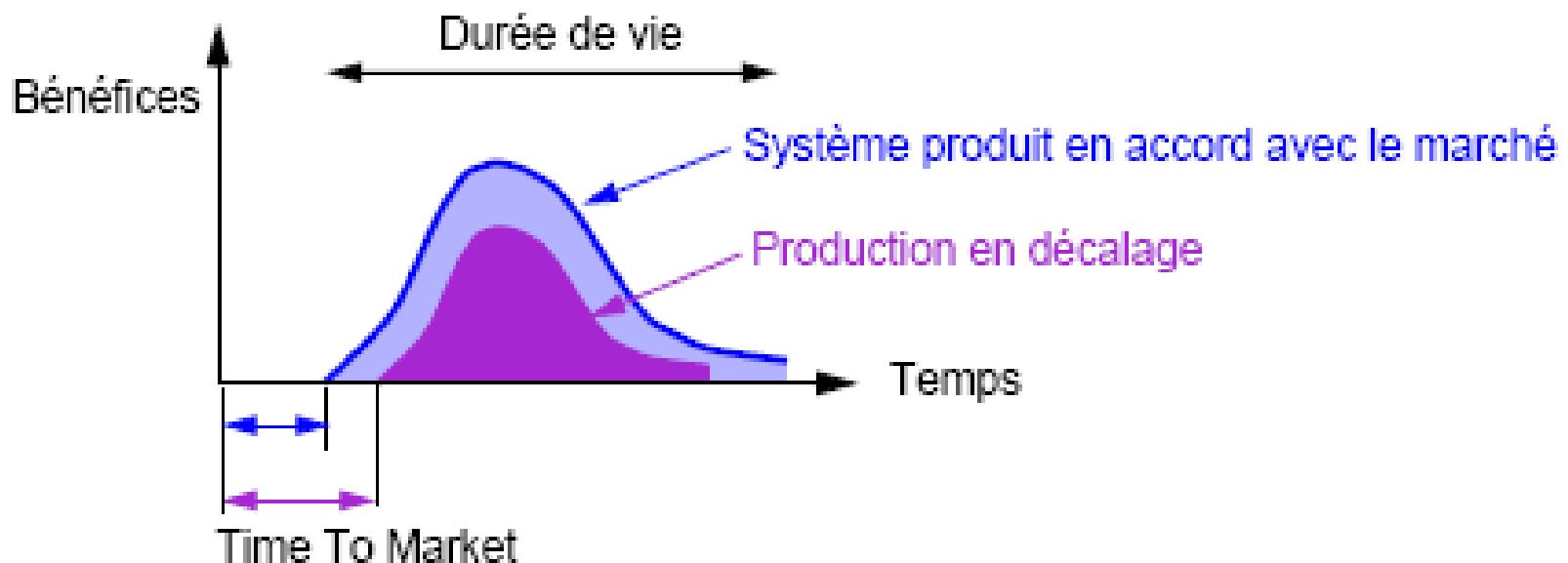
Introduction aux systèmes sur puce

1 milliard de téléphones portables  
200 millions de PC

# Contraintes de conception des systèmes embarqués

- **Performance:** puissance de calcul MIPS (Méga Instruction par seconde), temps d'exécution
- **Fiabilité:** probabilité pour que le système fonctionne correctement si c'était satisfait à  $t = 0$
- **Surface et encombrement:** GSM...
- **Consommation énergétique:** lié à l'autonomie des batteries (PDA, téléphone mobile...)
- **Sûreté:** aucun dommage (automobile, avionique...)
- **Coût et temps de développement:** faible coût

# Contraintes de conception des systèmes embarqués : TTM



Les produits ont une durée de plus en plus faible

Réduire le «time to market»

Réutilisation pour concevoir d'autres produits (rentabiliser)

# Contraintes de conception des systèmes embarqués

- Un système embarqué doit répondre à des contraintes **temps réel**:
  - Un système temps réel doit réagir à un stimuli dans un intervalle de temps dépendant de l'environnement.
  - Un système temps réel qui produit une bonne réponse mais trop tard est défaillant.
  - Contrainte de temps réel: **dure (hard) ou souple (soft)**
    - Une contrainte temps réel est appelé **dure** si le non respect de cette contrainte peut mener à des situations critiques voir catastrophiques.
    - Les autres contraintes sont appelées **soft**.

# Evolution des systèmes embarqués

## 1. Evolution technologique:

La miniaturisation des transistors a permis d'augmenter considérablement la capacité d'intégration dans les systèmes embarqués :

$7,2 \cdot 10^9$  transistors par  $\text{cm}^2$  est envisagée pour 2020

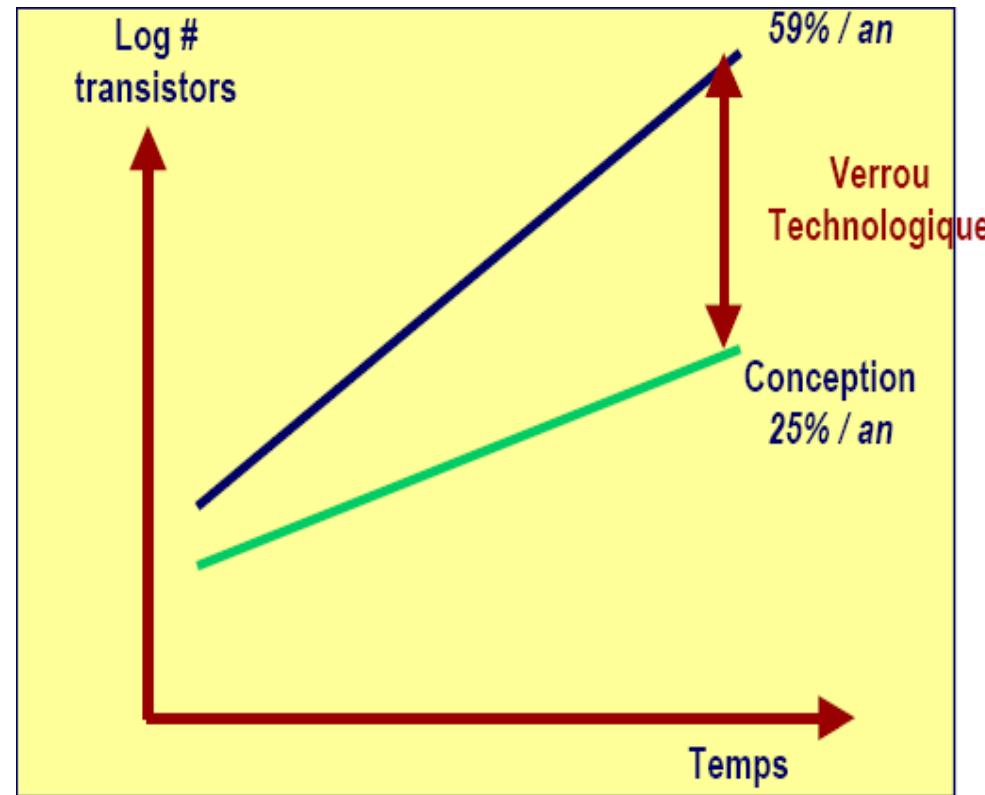
## 2. Evolution applicative:

Des applications plus complexes avec un nombre important et varié de fonctionnalités:

- Les téléphones portables intègrent de plus en plus de fonctionnalités et ils ont des tailles toujours réduites et sont alimentés par batterie.

# Evolution technologique/Conception

Le processus technologique autorise un accroissement de la complexité de 59% par an → Loi de MOORE



L'efficacité des concepteurs n'augmente "que de" 25% par an

→ Les outils de Codesign sont nécessaires pour permettre une exploration rapide de plusieurs architectures (RISC, FPGA, ASIC, IP, DSP, Analogique ...)

# Une grande variété de composants de calculs -1

- **ASIC -Application Specific Integrate Circuit**
  - Fonction complète ou coprocesseur (accélérateurs matérielle)
- **ASIP -Application Specific Instruction-set Processor**
  - Processeur spécialisé pour quelques applications
  - Jeu d'instructions et architecture adaptée à l'application
  - Meilleurs rapports MIPS/mW et MIPS/mm<sup>2</sup> que RISC et DSP
  - Mais coût de développement du compilateur
- **ASSP -Application Specific Standard Product**
  - Composant qui réalise une fonction spécifique
- **DSP, VLIW**
  - Processeurs dédiés au traitement de signal, application multimédia

# Une grande variété de composants de calculs -2

- Microcontrôleurs
- RISC/ Superscalaires
  - ARM, PowerPC, Sparc, ...
- FPGA -Field Programmable Gate Array
  - Reconfigurable
  - Accélérateur de calculs (coprocesseur)

# Différentes cibles utilisables pour concevoir un système embarqué

## ■ **Les cibles logicielles**

- Les processeurs généralistes
- Les DSPs
- Les Microcontrôleurs

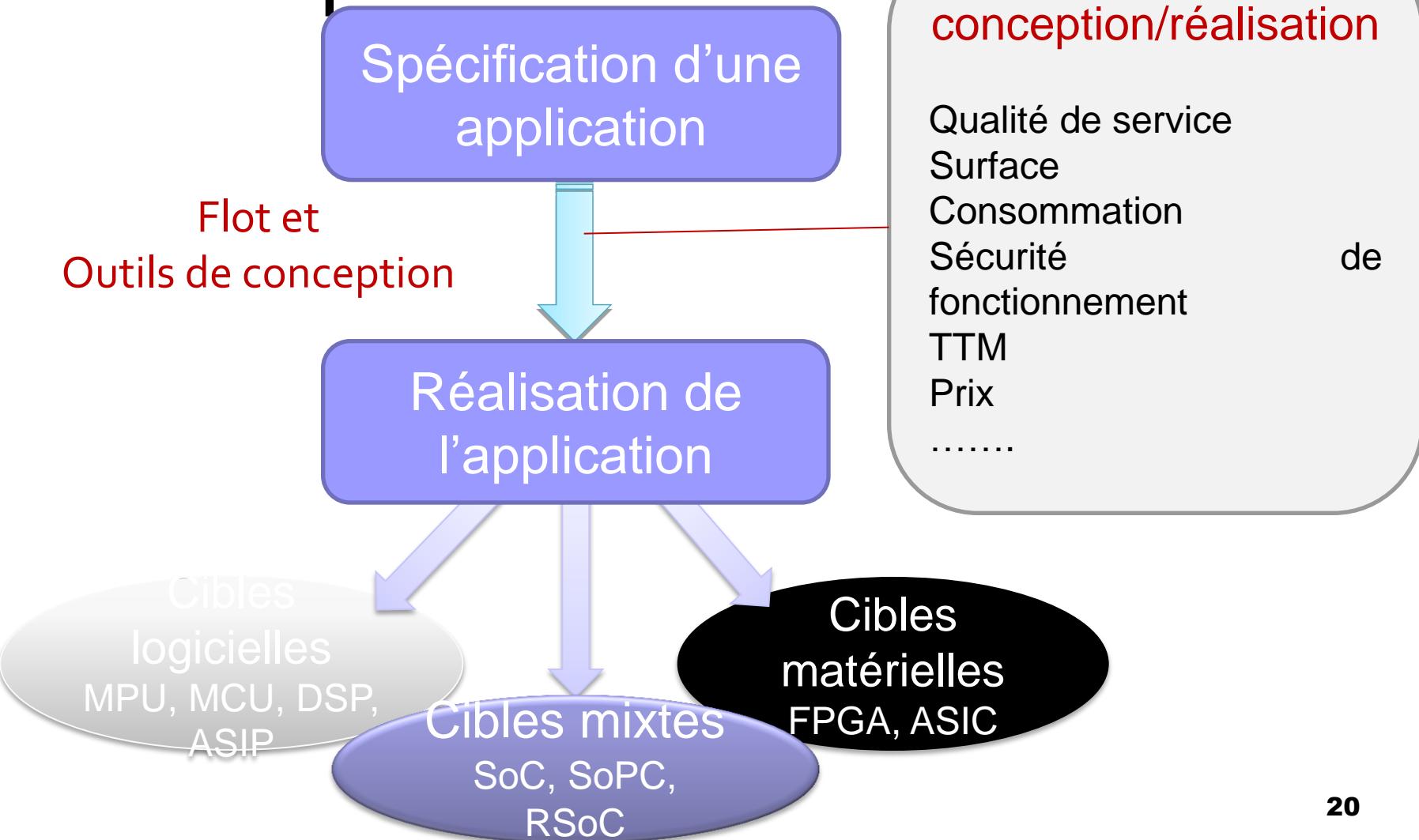
## ■ **Les cibles matérielles**

- Les ASICs: Application Specific Integrate Circuit
- Les FPGAs: Field Programmable Gate Array

## ■ **Les cibles mixtes :**

- Les SOCs(système sur puce) : L'intégration de plusieurs unités **sur une même puce**

# Conception des systèmes embarqués

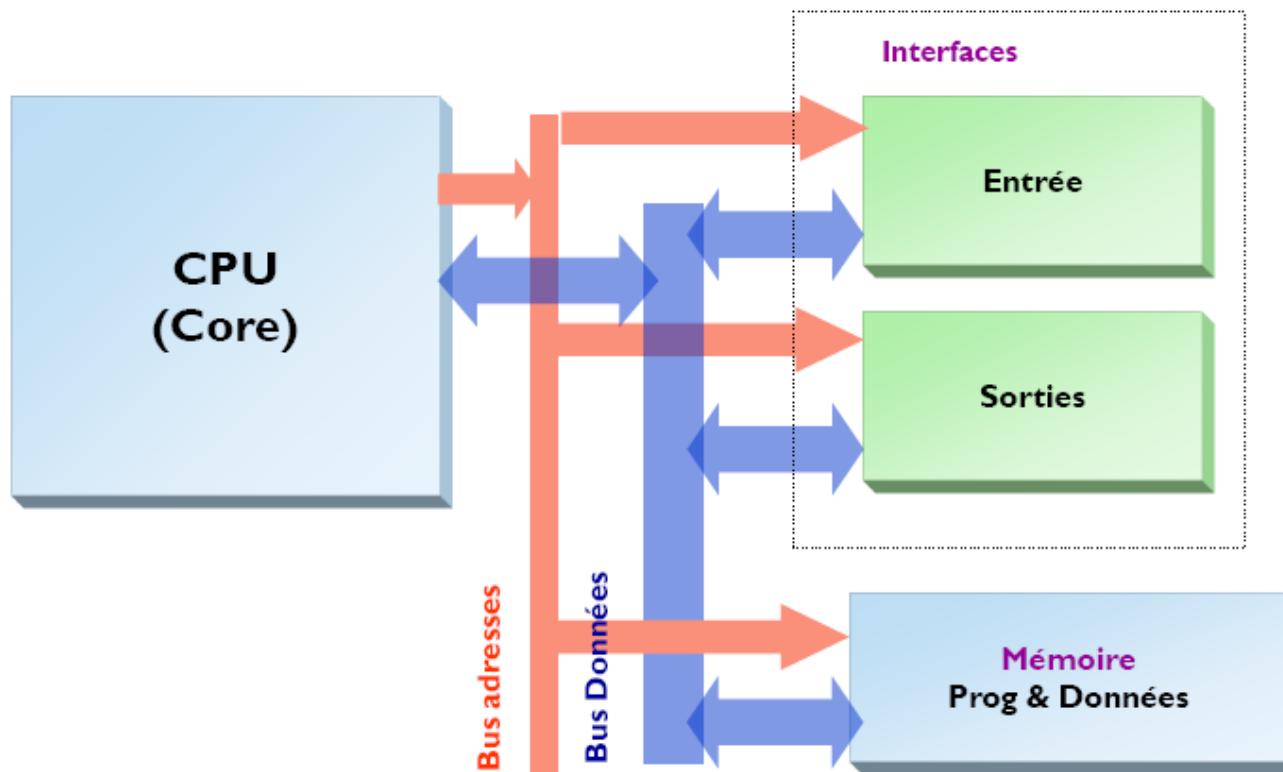


# Cible logicielle et matérielle

- Cibles logicielles : sont des cibles programmables cad qu'on peut modifier l'application dédiée juste en modifiant le code
- Cibles matérielles : sont des cibles programmées dédiées et conçues pour des tâches bien déterminées et qu'on ne peut pas modifier leur traitements

# Processseurs généralistes

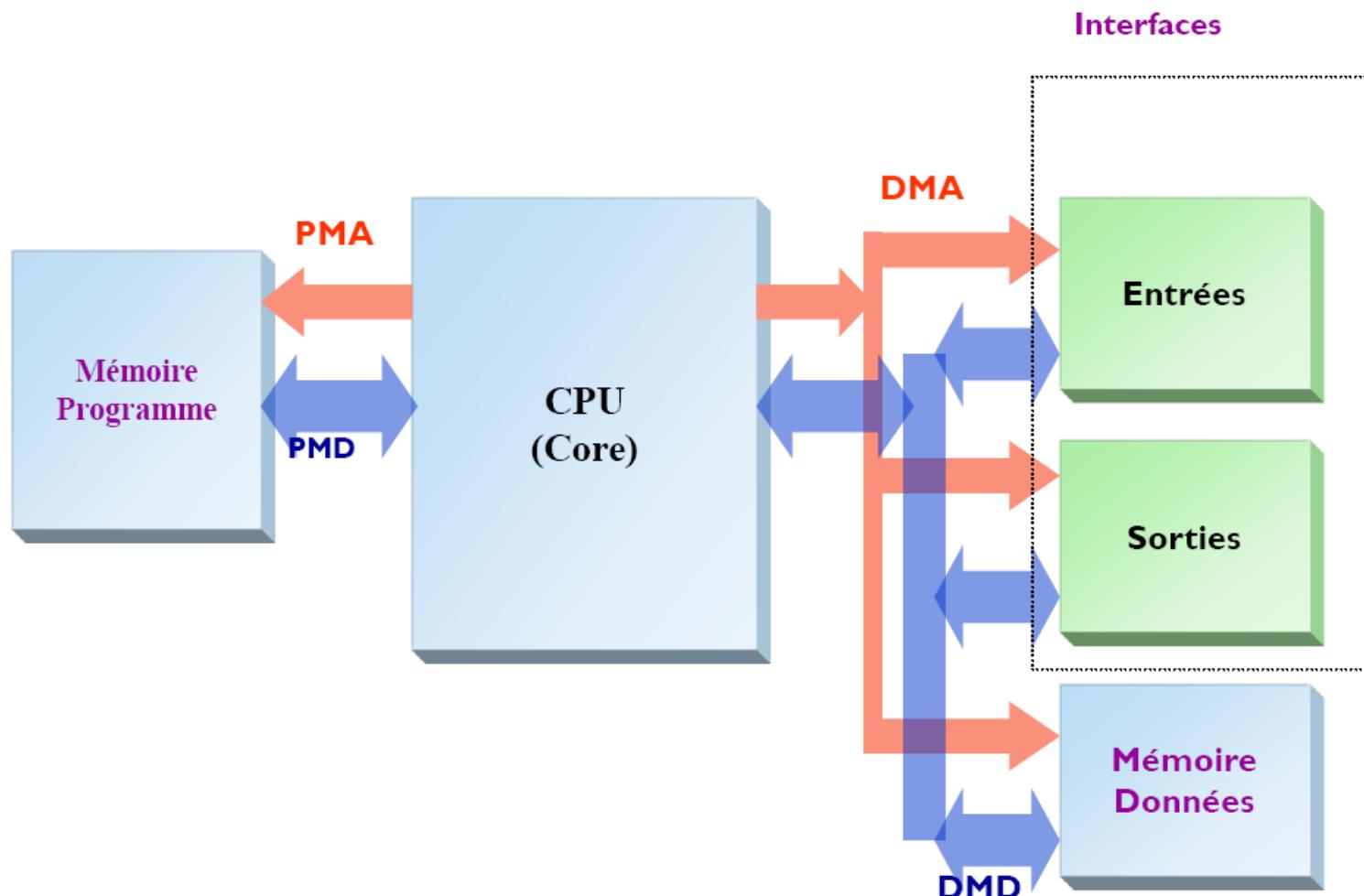
- Architecture de Von Neumann



# Architecture de Von Neumann

- Mémoire de donnée et mémoire de programmes **partagée**
- L'exécution d'une instruction peut se faire en plusieurs cycles processeur
  - Recherche de l'instruction (*Instruction fetch*)
  - Recherche de l'opérande 1 (*data fetch*)
  - Recherche de l'opérande 2 (*data fetch*)
- Performances de calcul limitées
  - Non appropriée aux opérations de traitement du signal

# Architecture Harvard

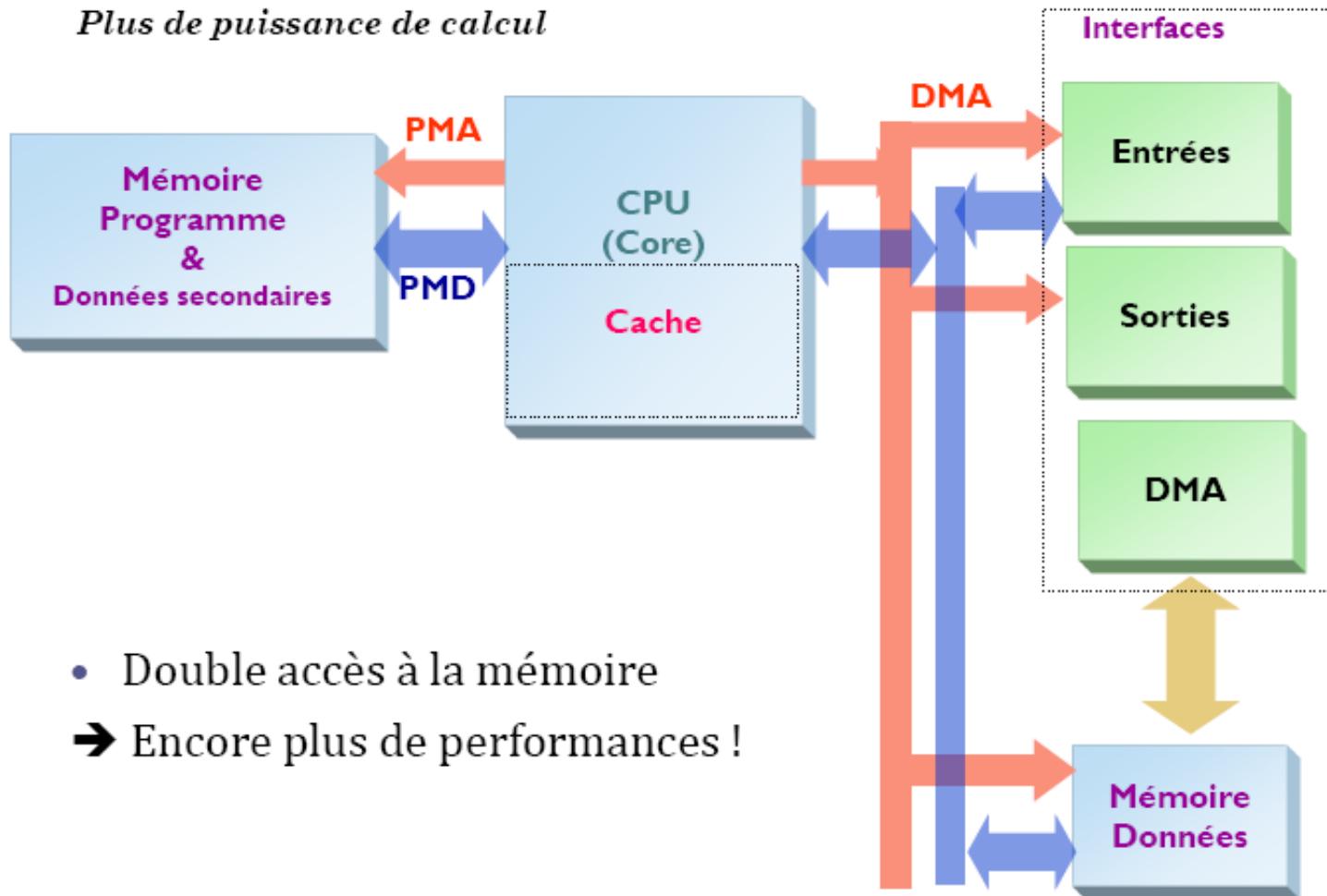


# Architecture Harvard

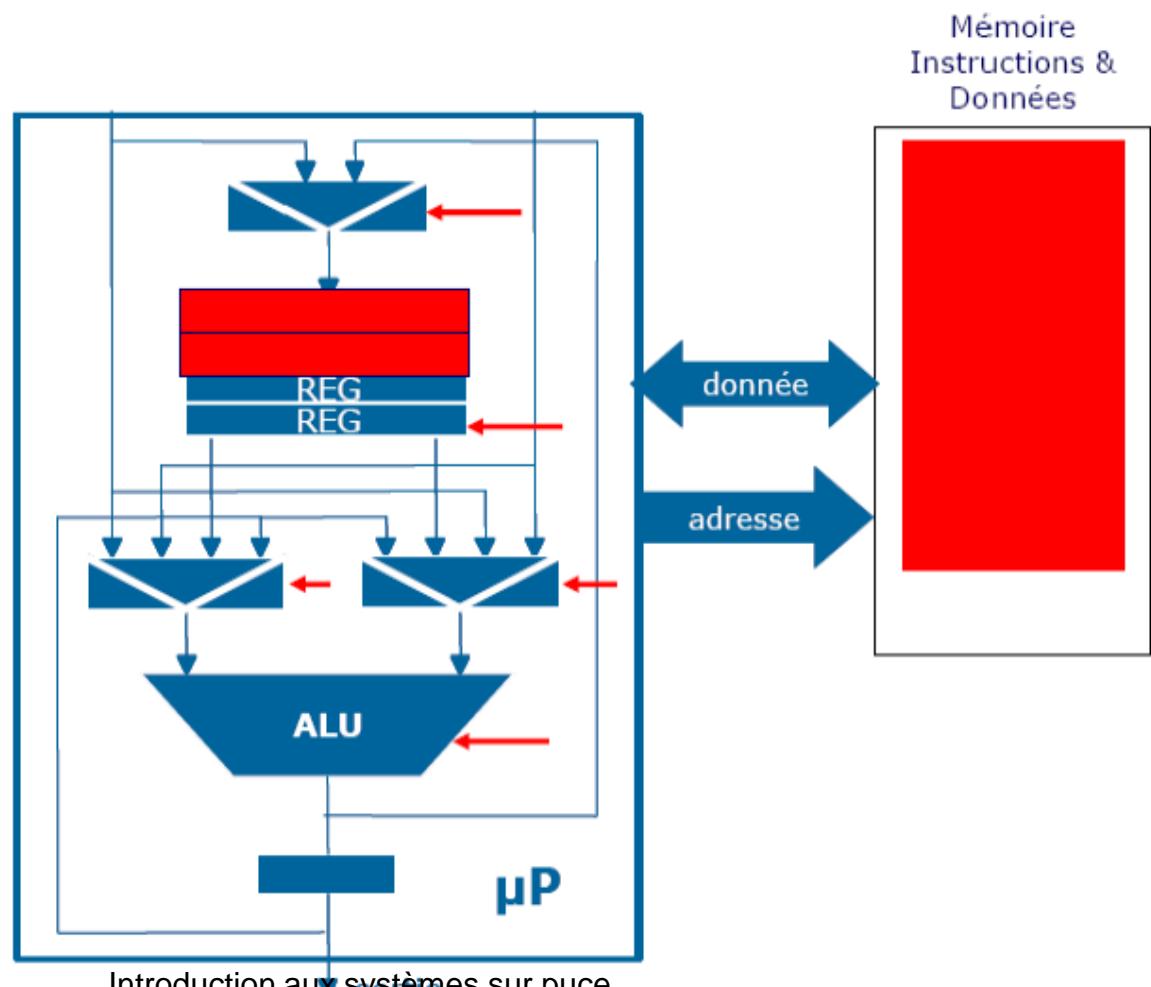
- **Séparation** entre la mémoire de donnée et la mémoire de programme
- Chaque mémoire comporte ses bus propres à elle
- Recherche de l'instruction et de la donnée en 1 cycle d'horloge
- Le CPU (core) comporte un chemin de donnée plus organisé
- Puissance de calcul meilleure

# Architecture Harvard modifiée

*Plus de puissance de calcul*



# Les microprocesseurs standards



# Caractéristiques générales des microprocesseurs

## ■ Performances

- Temps par tâche = I  $\times$  C  $\times$  T
- I : nombre d'instructions par tâche
- C : nombre de cycles machine par instructions
- T : temps d'un cycle machine (dépend de la technologie et de l'efficacité de l'ALU)

## ■ 3 types de processeurs

- **CISC** (Complex Instruction Set Computer) : I faible, C grand
- **RISC** (Reduce Instruction Set Computer) : I élevé, C faible
- **VLIW** (Very Large Instruction Word) : I réduit car macro-instruction RISC, C faible

# Les architectures CISC

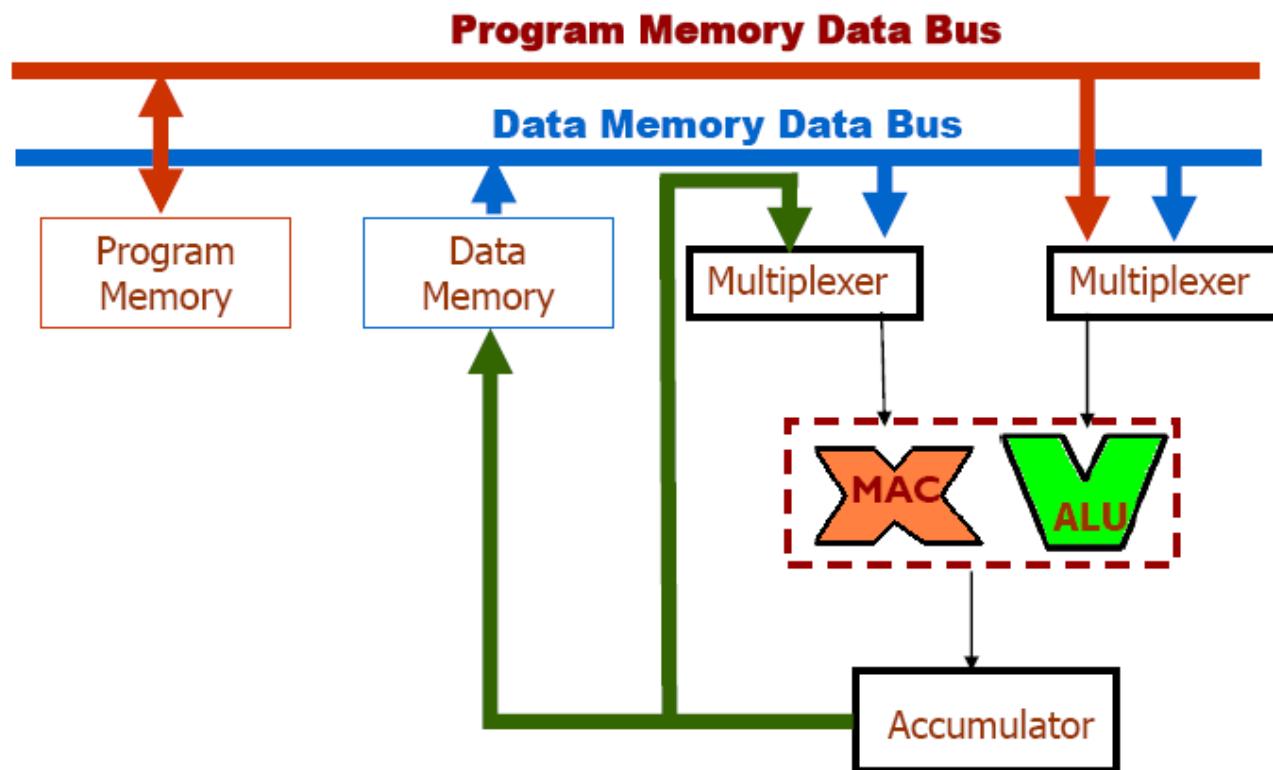
- Ancienne Architecture des processeurs
- Architecture présentant un jeu d'instructions complexe
  - Plusieurs opérations peuvent être codés par une même instruction
- Plusieurs modes d'adressage
- Nécessite moins de mémoire par rapport à une architecture RISC
- **Exemple :**
  - Motorola 680x0 ,
  - S/360 d'IBM,
  - Intel Pentium
  - Intel Pentium Pro, Pentium II III et 4 : PeusdoCISC(cœur de RISC mais vue comme un CISC) permet de garder la compatibilité ascendante des processeurs (x86)

# Les architectures RISC

- Architecture présentant un jeu d'instruction relativement réduit
  - Une seule opération /instruction
  - Taille fixe pour les instructions
- Modes d'adressage simples
- Ont permis une augmentation de la fréquence
- Présente un nombre important de registres généraux
- Les seules instructions ayant besoin d'accès à la mémoire sont les instruction de chargement et de rangement
- **Exemple :**
  - PowerPC
  - ARM
  - SPARC
  - MIPS.

# Digital Signal Processing (DSP)

processeurs dédiés au traitement de signal, application multimédia



# Digital Signal Processing (DSP)

## ■ Caractéristiques

- Architecture RISC complexe, superscalaire(plusieurs unités de traitements), pipeline
- Architecture Harvard et Super Harvard (nombreux bancs mémoire)
- Instructions complexes mais jeux d'instructions réduit
- Exemple : Texas Instrument C6x

## ■ Avantages

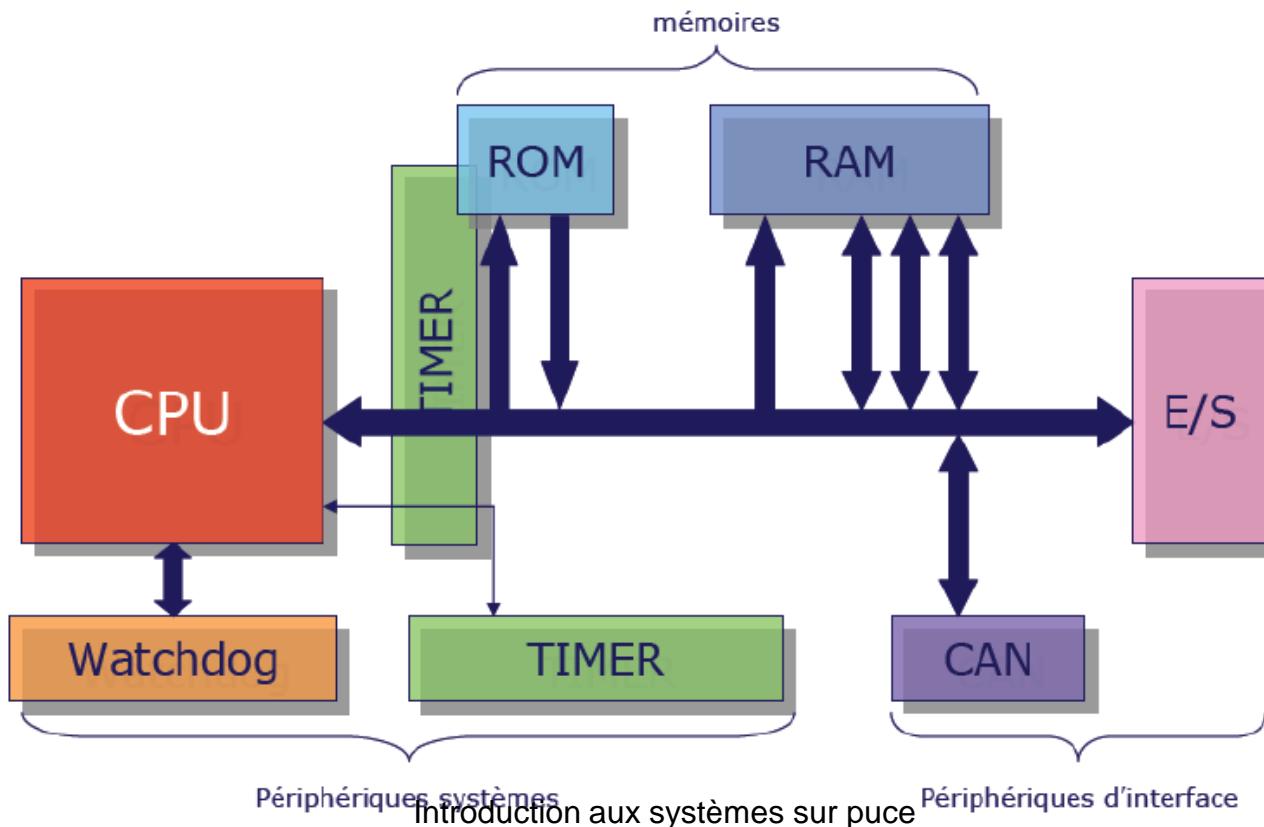
- Très économique : pas besoin d'acheter des périphériques
- Spécialisés traitement du signal
- Peuvent mélanger calcul flottant et virgule fixe

## ■ Inconvénients

- Coûts élevés
- Consommation d'énergie élevée

# Les Microcontrôleurs

un circuit intégré rassemblant dans un même boîtier un **microprocesseur**, plusieurs types de **mémoires** et des **périphériques** de (Entrées-Sorties).



# Les Microcontrôleurs

## ■ Caractéristiques

- Architecture simple, jeux d'instructions réduit
- Basé sur des architectures de processeurs connus
- Exemple : 68HC11, PIC de Microchip, STM32 de ST

## ■ Avantages

- Très économique : pas besoin d'acheter des périphériques Spécialisé
- Simple d'utilisation

## ■ Inconvénients

- Pas temps réel
- Peu performant
- Spécialisé: ne convient pas à tous les domaines d'application

# Cibles logicielles

## ■ Avantages

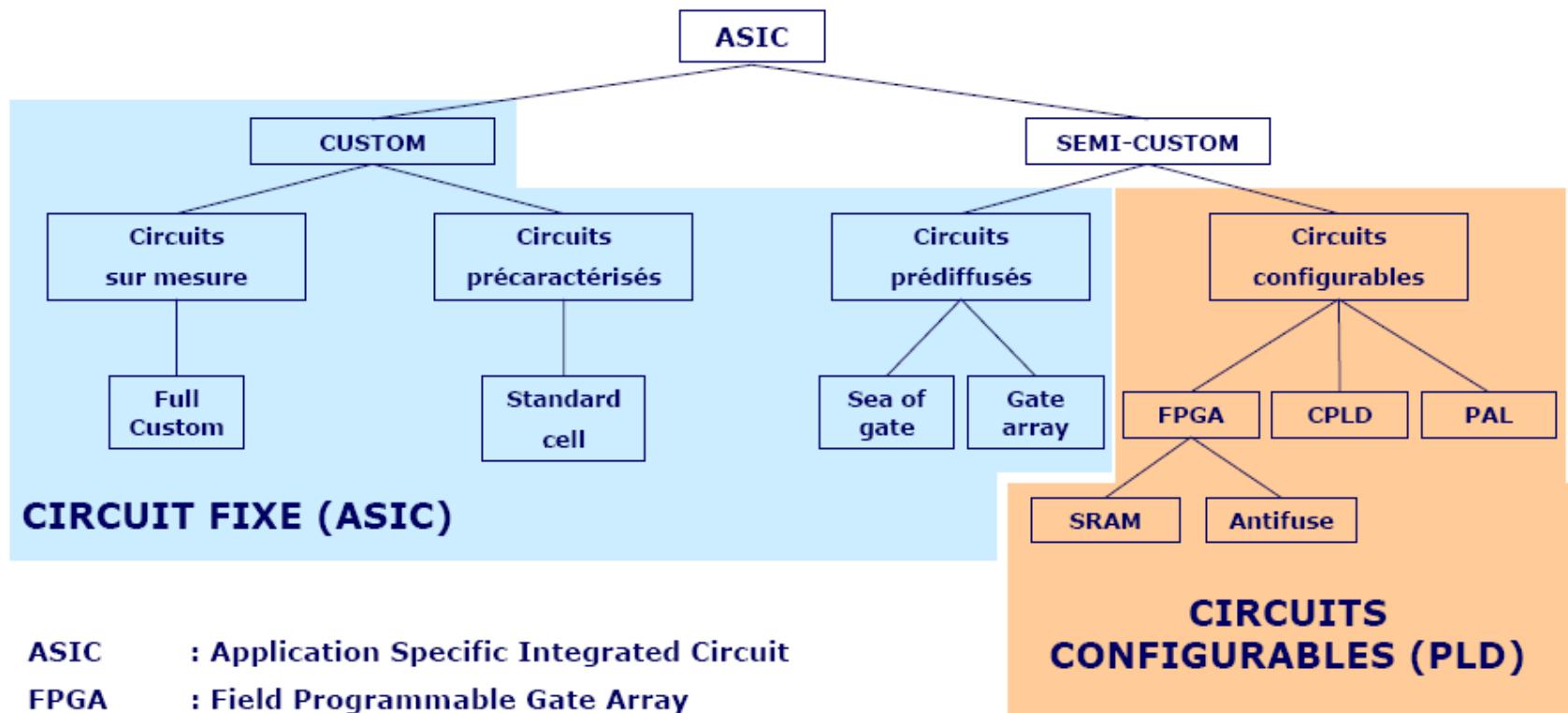
- Flexibilité : il suffit de modifier le programme pour modifier l'application
- Simple à mettre en oeuvre grâce à la programmation de haut niveau (langage C) (possibilité de grande abstraction par rapport au matériel)
- Temps de conception courts et coûts de conception faible
- Prix de reviens faible
- Composants spécialisés traitements du signal : DSP

# Cibles logiciels

## ■ Inconvénients

- Faibles performances: à cause d'une architecture séquentielle, une opération à la fois, ou quelques unes dans le cas superscalaire et des trop nombreux accès à la mémoire (instructions + données)
- Le passage par un compilateur peut dégrader les performances
- Complexité du langage machine et de l'assembleur

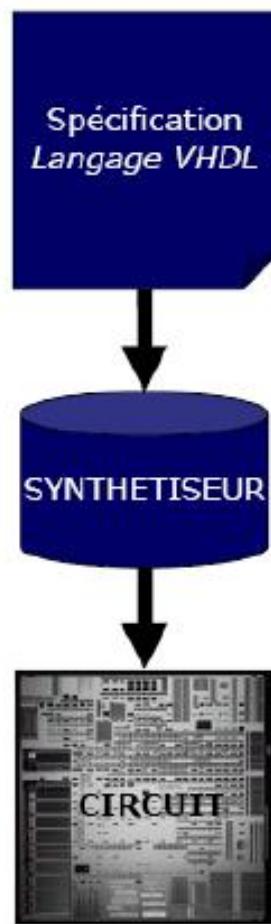
# Les cibles matérielles numériques



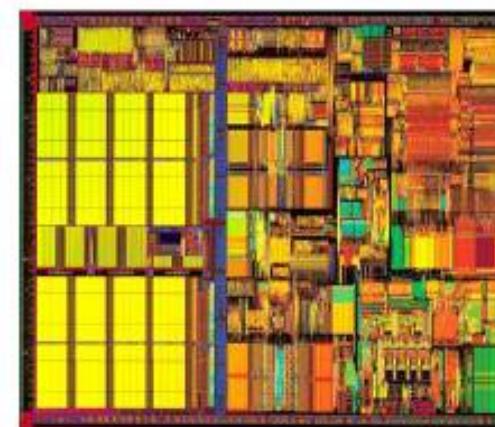
- ASIC** : Application Specific Integrated Circuit
- FPGA** : Field Programmable Gate Array
- CPLD** : Complex Programmable Logic Device
- PAL** : Programmable Array Logic
- GAL** : Generic Array Logic = PAL
- SRAM** : Static Random Access Memory



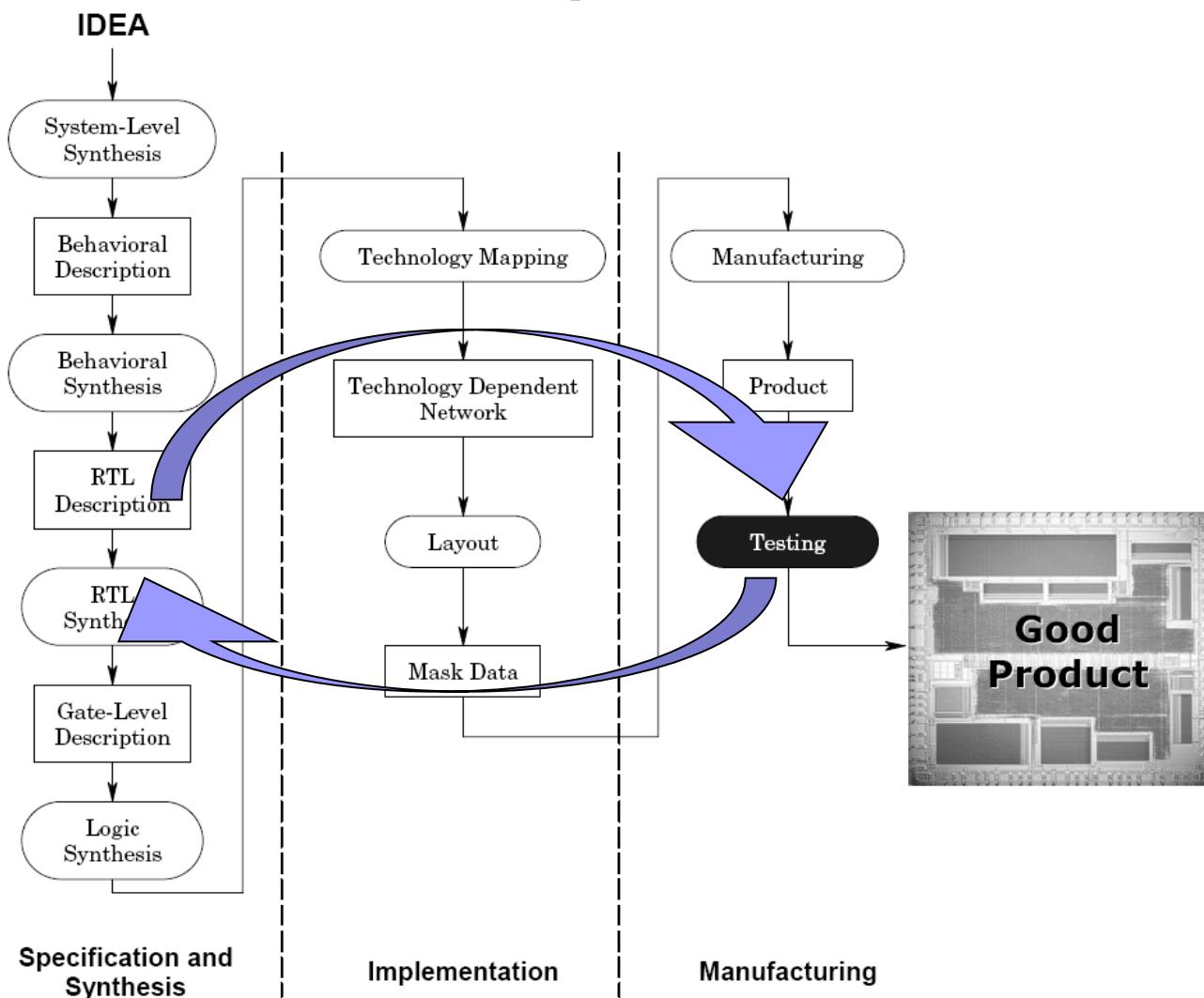
# Les cibles matérielles spécialisées



- **ASIC : Application Specific Integrated Circuit**
  - Numérique, analogique ou mixte (télécommunication)
  - Spécialisé pour une application
  - Réalisation complexe (de la spécification haut niveau à la synthèse physique)
  - Extrêmement performant ; dédié + réalisation parallèle + technologie de pointe
  - Circuit = cahier des charges



# Flot de conception ASIC



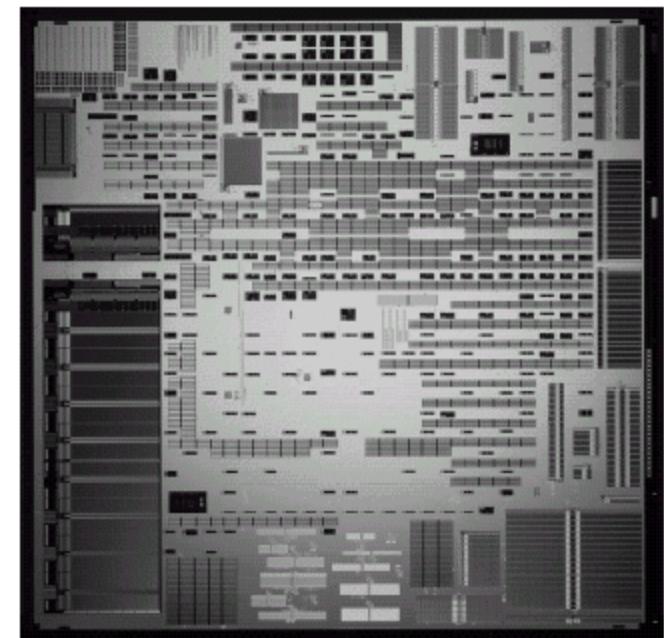
# Les ASIC

## AVANTAGES

- hautes intégrations
- hautes performances (vitesse, low-power)
- coûts faibles pour de gros volumes de production
- personnalisation
- sécurité industrielle

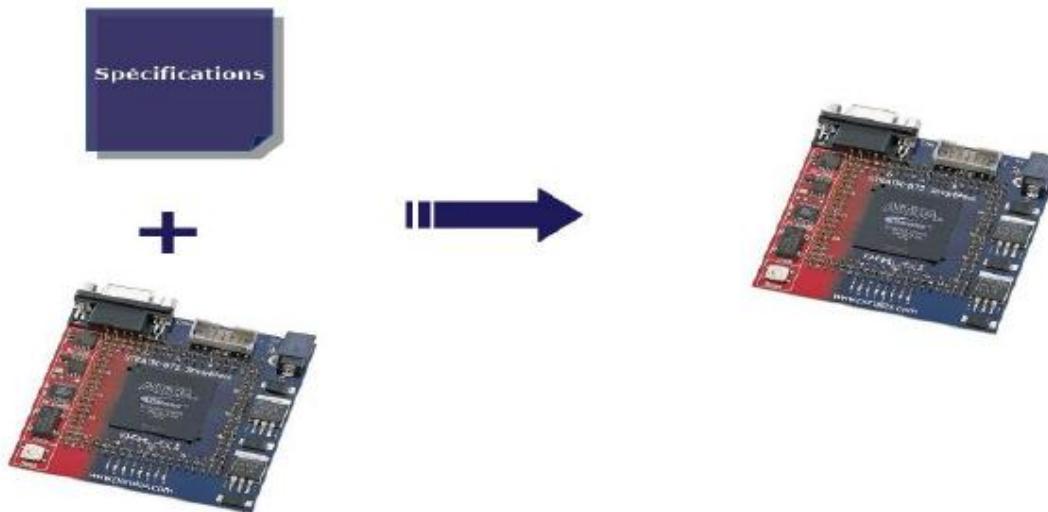
## INCONVENIENTS

- prix du 1er exemplaire
- pas d'erreur possible
- non-flexible
- time-to-market élevé
- fabrication réservée aux spécialistes (fondeur)



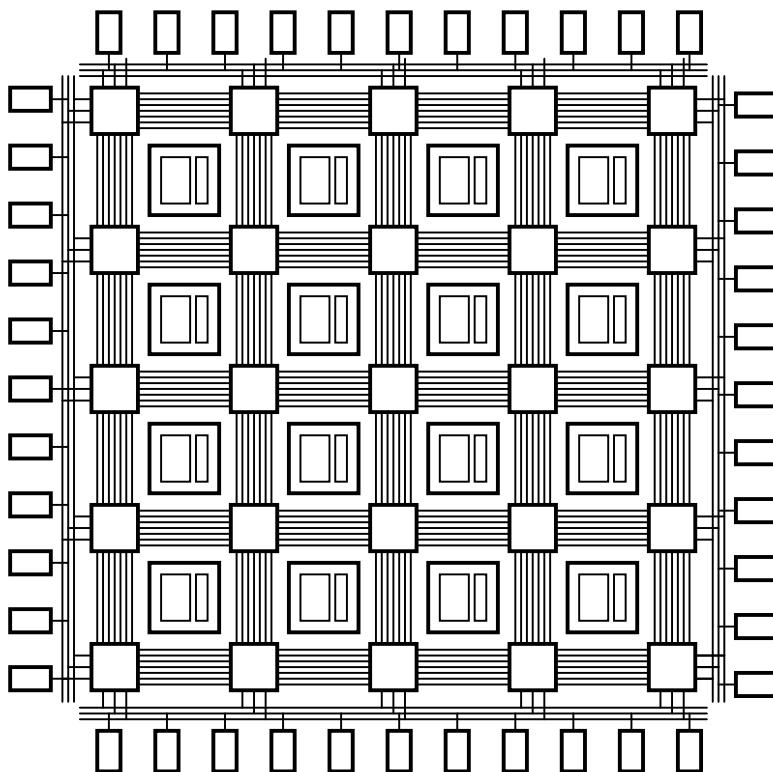
# Les circuits configurables

- **FPGA** : Field Programmable Gate Array
- **CPLD** : Complex Programmable Logic Device
- **PAL** : Programmable Array Logic
- **GAL**: Generic Array Logic= PAL
- **SRAM**: Static Random Acess Memory

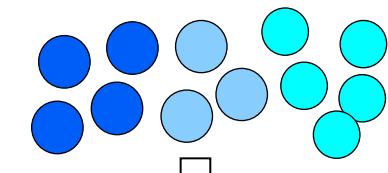


# La logique programmable : les FPGA

- Matrice de blocs logiques reliés par un réseau d'interconnexion programmable



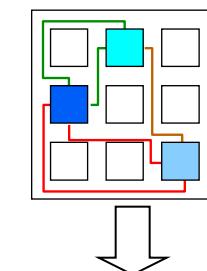
Fonctions logiques



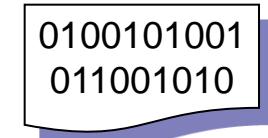
Blocs logiques élémentaires



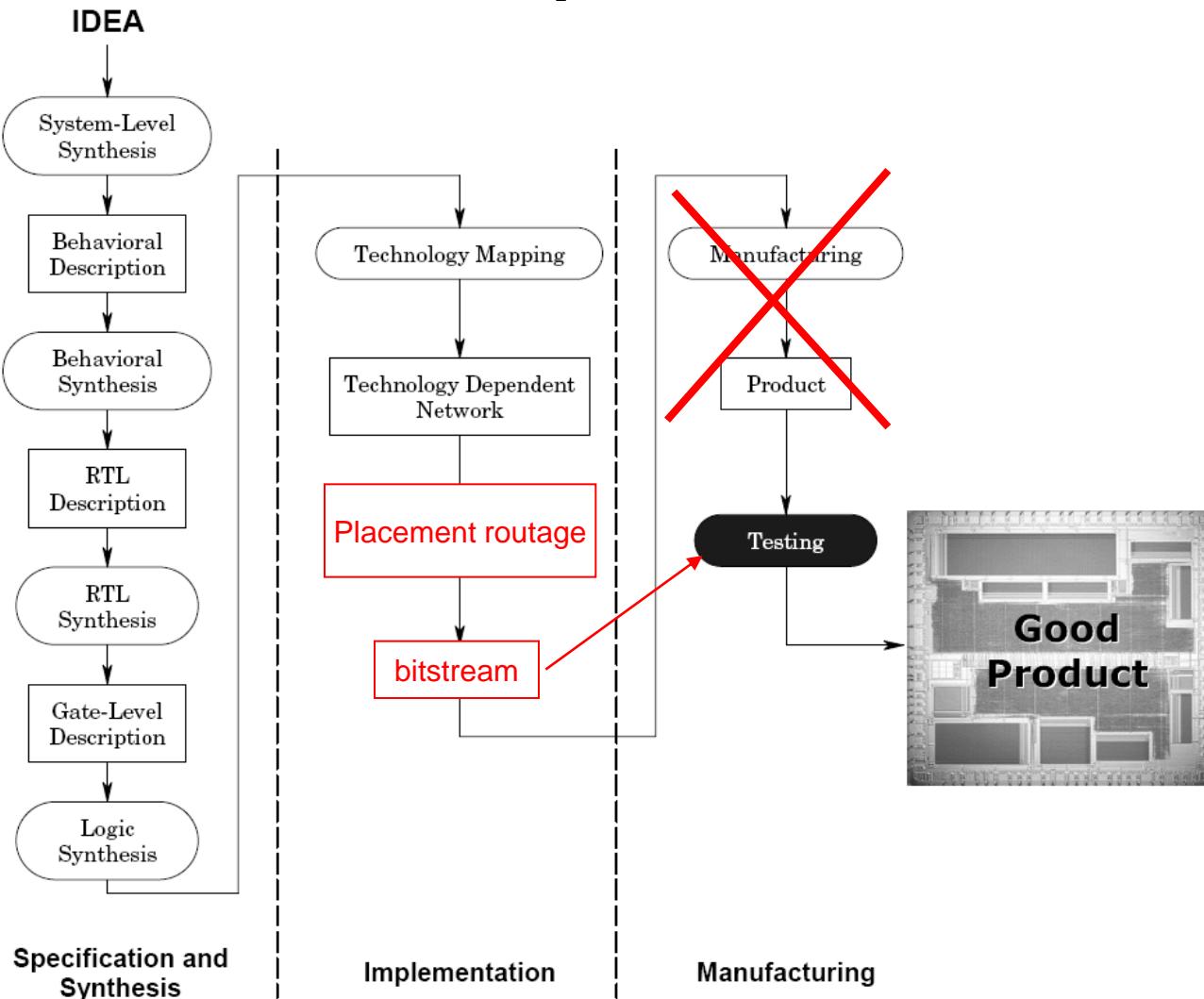
Circuit placé-routé



Fichier de configuration (bitstream)



# Flot de conception FPGA



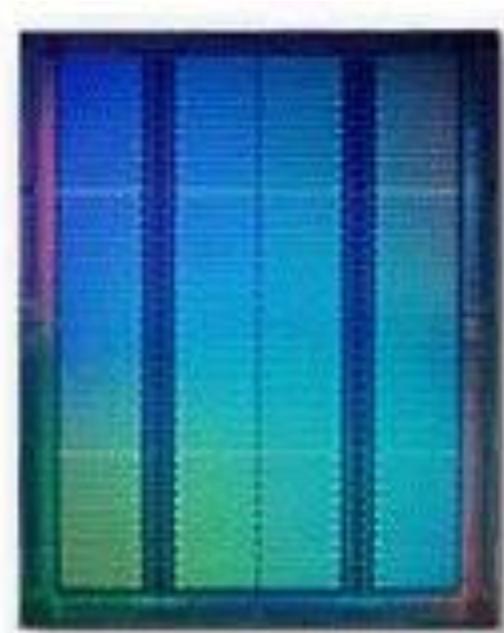
# Les FPGA

## AVANTAGES

- possibilité de prototypage
- time-to-market faible
- adaptabilité aux futurs évolutions grâce à la reconfiguration
- flexibilité

## INCONVENIENTS

- intégration limité par les ressources de routage
- performances
- prix à l'unité élevé pour de grosses productions



# ASIC ou FPGA ?

- Le concepteur a un choix large de cibles matérielles
  - Il doit bien maîtriser les contraintes de son applications pour choisir la bonne cible
  - Il doit si nécessaire s'appuyer sur des estimations (performances, prix, etc.)
- Les FPGA et les ASIC sont des composants clairement différents
  - Les FPGA sont flexibles et peu performants
  - Les ASIC sont performants mais peu flexibles
  - Mais ce sont des cibles matérielles différents des cibles logicielles ( $\mu$ P, DSP)
- Les FPGA sont des circuits de plus en plus utilisés aujourd'hui
  - Rapport prix/performances intéressants
  - Large choix
  - Non plus seulement réservés au prototypage d'ASIC

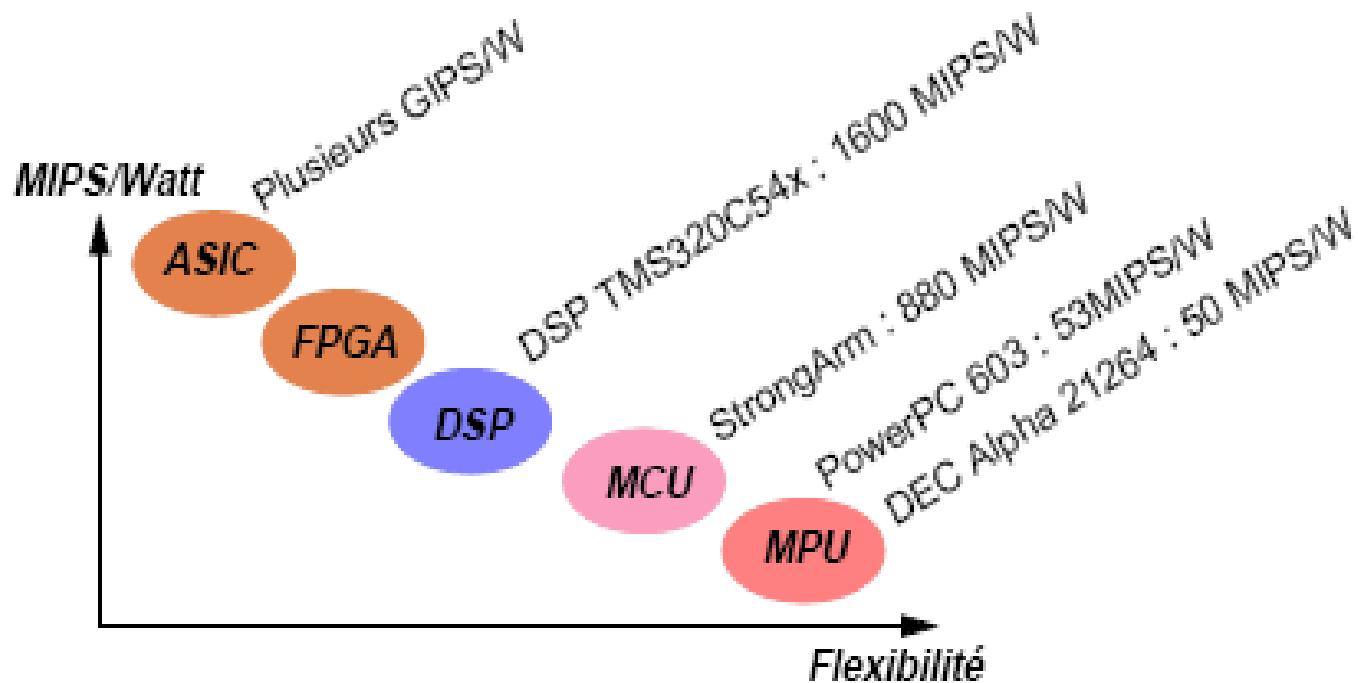
# ASIC ou FPGA ?

- Le choix entre FPGA ou ASIC, se fait en fonction du cahier des charges de l'application :
  - temps de mise sur le marché et durée de vie courte  
→FPGA
  - très petit nombre de circuits  
→ FPGA
  - optimisation des performances  
→ ASIC

# Intérêt des cibles matérielles

- **Performances** : utilisation d'architectures optimales :
  - Pipeline
  - opérations câblées, en mémoire (FPGA)
  - optimisation bits à bits
  - Taille des opérandes optimales
- **Protection industrielle**
- **Outils de conception puissants:**
  - Langages HDL
  - Bibliothèques de macro-fonctions paramétrables
  - compilation (synthèse logique + placement routage)

# Comparaison: performance, consommation, flexibilité



Processeur Intel XScale

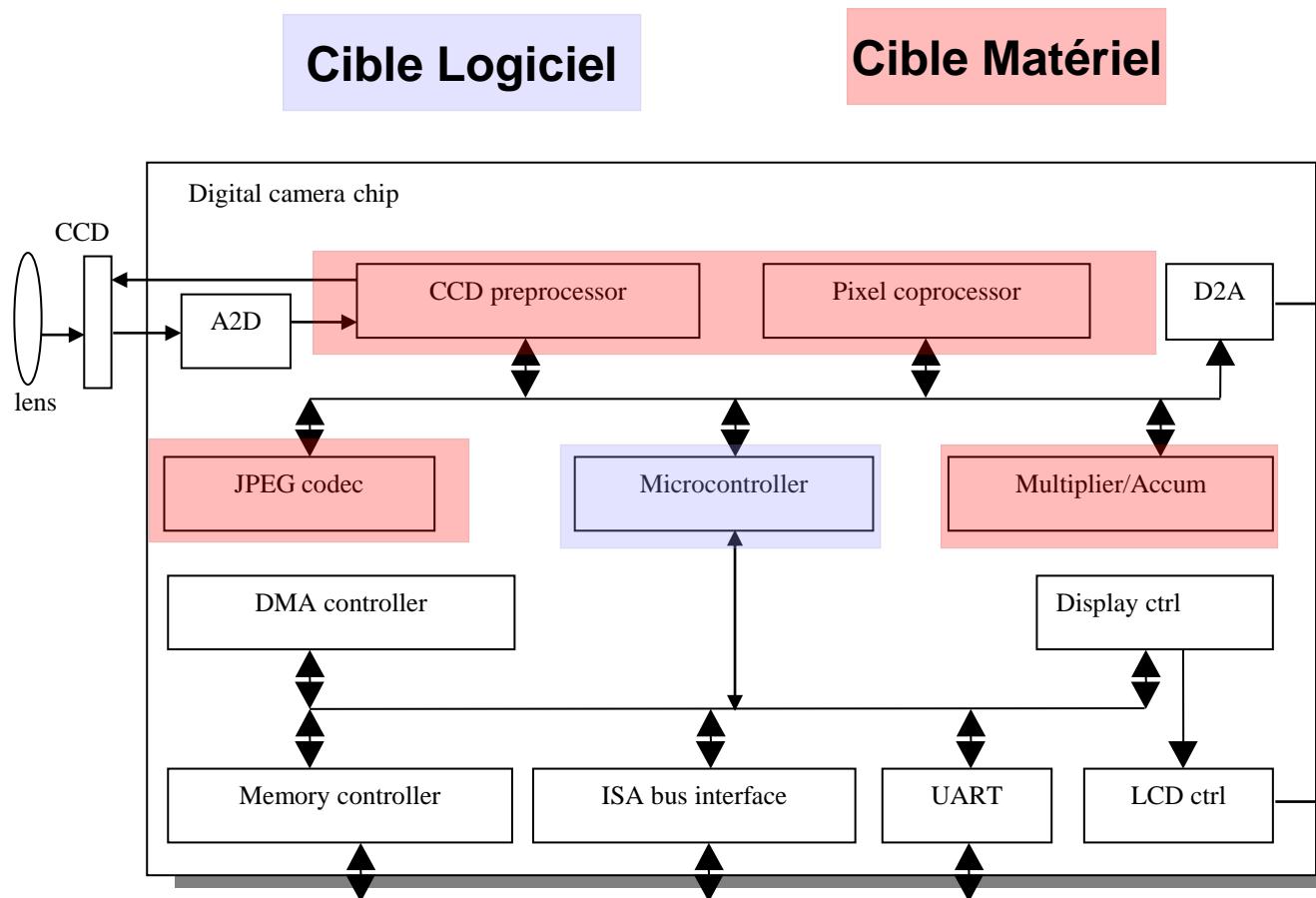
{	800 MHz : 1,65V	1100 MIPS/W
	400 MHz : 1 V	1250 MIPS/W
	150 MHz : 0,75 V	4500 MIPS/W



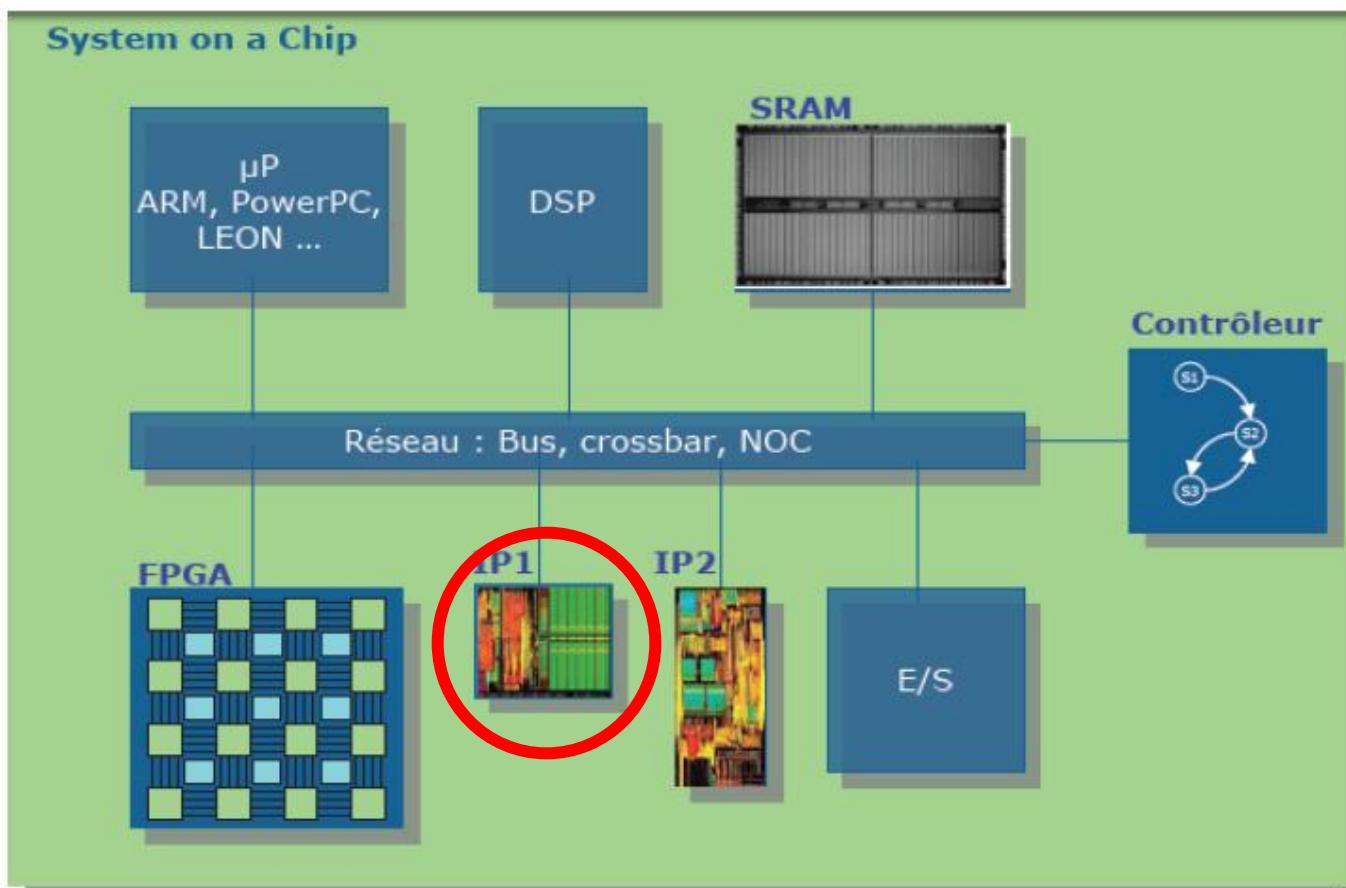
# Cible mixte : Les Systems on Chip

- Système **complet** mis en œuvre sur la même puce
  - SoC pour les télécoms : un microprocesseur, un DSP, RAM, ROM, ...
- Utilisation du parallélisme
  - Accroissement des performances, maîtrise de la complexité et de la consommation

# Exemple : Appareil photos numérique



# Cible mixte :Les Systems on Chip



# Conception modulaire : Notions d'IP (Intellectual Properties)

- IP: se sont des blocs prêt à être utilisé dans des applications et selon le besoins.
- Ces blocs permettent un gain en coût et en temps de conception. Au lieu de concevoir tous les composants, on peut se servir de qlq IP mais tout en s'assurant de leurs intégrité avec nos propres composants.

# Conception modulaire : Notions d'IP (Intellectual Properties)

- **Blocs fonctionnels complexes réutilisables**
  - Hard: déjà implanté, dépendant de la technologies, fortement optimisé
  - Soft : dans un langage de description matériel (VHDL, Verilog...), paramétrables
- Normalisation des interfaces (OCP OpenCore Protocol)
- Environnement de développement (co-design, co-specif, co-verif)
- Performances moyennes (peu optimisé)

# Utilisation d'IP

## ■ Bloc réutilisable (IP)

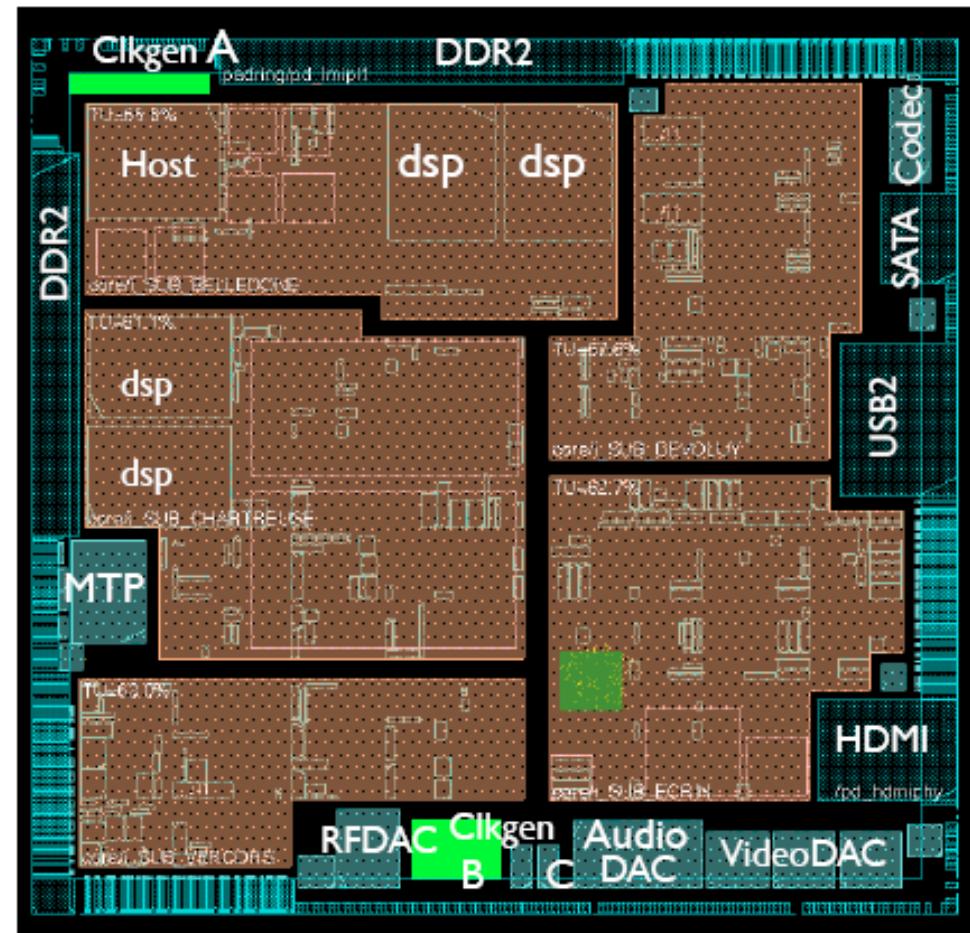
- connaître les fonctionnalités
- estimer les performances dans un système
- être sûr du bon fonctionnement de l'IP
- intégrer cet IP dans le système
- valider le système

# Notion d'IP (IP-reuse)

Communications	Bus Interface	Digital Signal Processing	Processor, Peripheral
ADPCM (u-law, a-law)	PCI Target	Color Space Converter	Nios™ Processor
ATM Controller	PCI Master-Target	Correlator	Tensilica X-tensa Processor
CRC	PCI-X	Digital Modulator	PalmChip Bus
Ethernet MAC (10/100/Gigabit)	CAN Bus	Discrete Cosine Transform	SDRAM Controller
HDLC Protocol Core	IIC Master & Slave	Fast Fourier Transform	DDR-SDRAM Controller
IMA Controller	IEEE 1394	FIR Compiler	QDR-SDRAM Controller
SONET/SDH Framer	PowerPC Bus Arbiter	IIR Filter	8237 DMA Controller
T3/E3 Framer	PowerPC Bus Master	Image Processing Library	8255 Peripheral Interface
Packet Over SONET Processor	PowerPC Bus Slave	NCO	8259 Interrupt Controller
Telephony Tone Generator	USB Function Controller	Reed Solomon Encoder/Decoder	8254 Timer/Counter
Utopia Master & Slave	USB Host Controller	Interleaver/Deinterleaver	
POS-PHY Interface		Viterbi Decoder	
		Turbo Decoder	
			8051, 6502, Z80

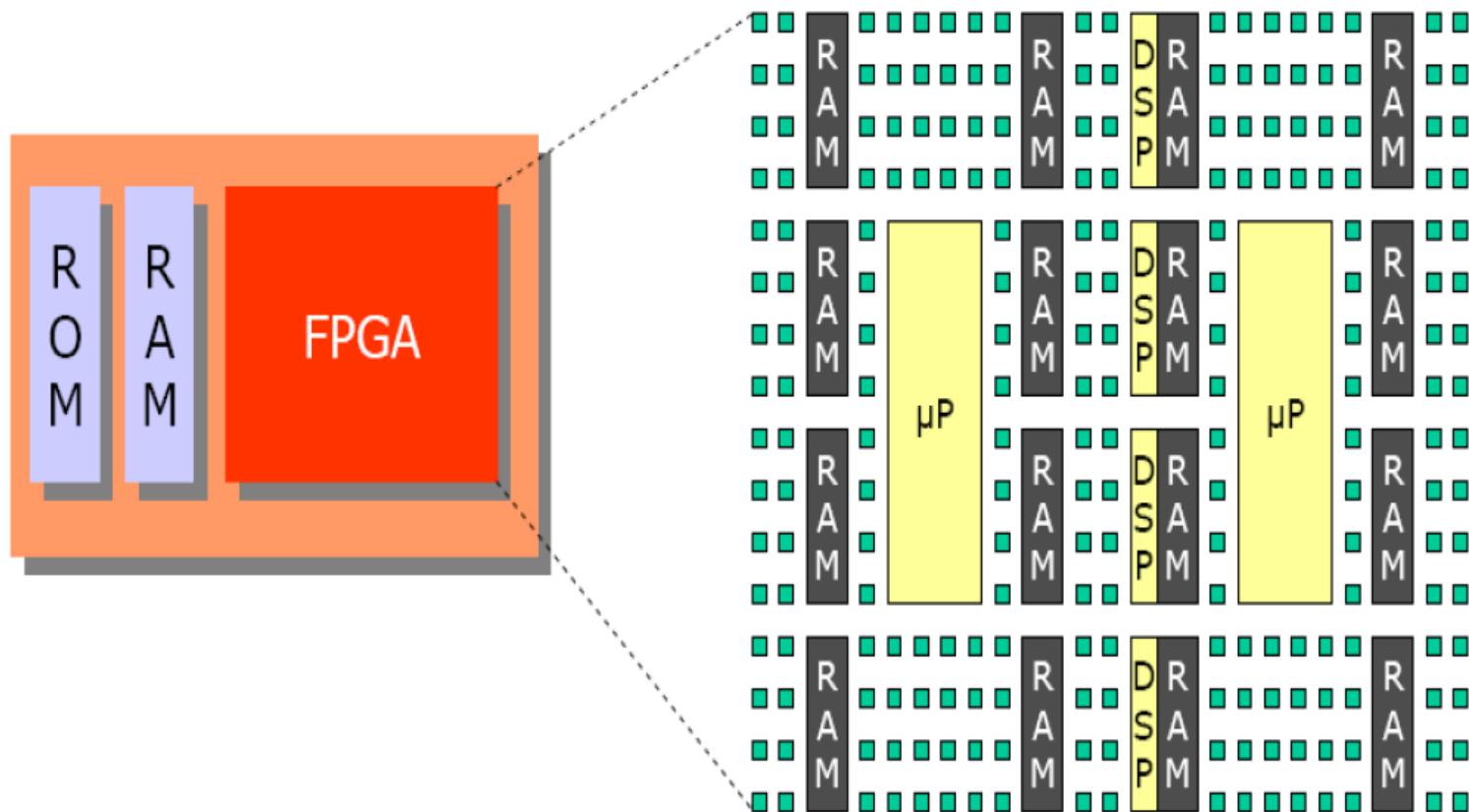
# Exemple : circuit de décodage vidéo de ST (STi7200)

- Circuit dédié au décodage de la TV numérique HD
  - 150 millions de transistors
  - 4 processeurs ( 2 DSP pour la vidéo, 1 DSP pour l'audio et 1 généraliste pour la configuration)
  - 36 Softs IPs et 2 Hard Ips
  - 16 IP analogiques, 19 IOL ibs



# Les Cibles mixtes: System on programmable chip (SOPC)

- Les fonctionnalités peuvent être implantées dans des composants logiques programmables de type FPGA. On parle alors de système **SoPC**



# Les processeurs pour le SoPC

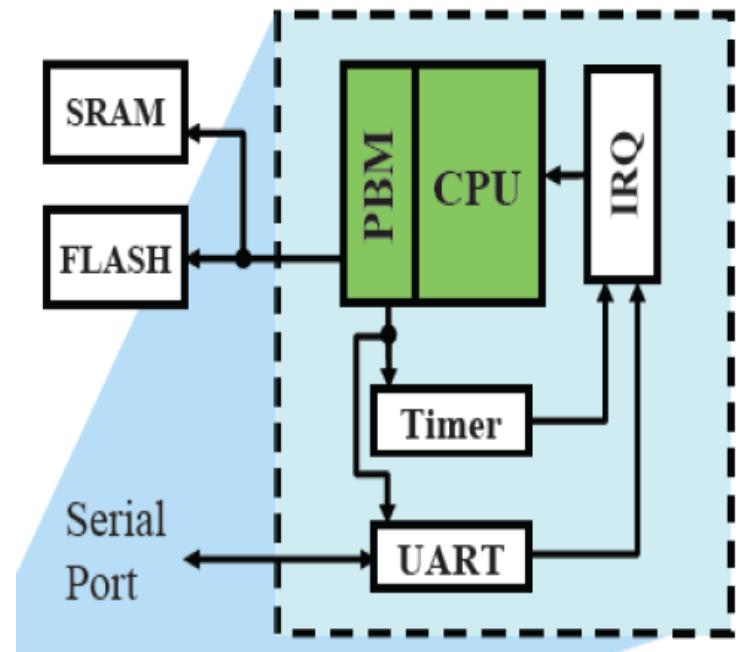
- Le choix d'un processeur pour le SoPC peut se faire sur différents critères :
  - **Processeur hardcore** : pour ses performances au détriment de la flexibilité.
  - **Processeur softcore** : pour sa flexibilité de mise à jour au détriment de performances moindres que le précédent. La portabilité vers n'importe quel circuit FPGA est assurée en étant donc circuit FPGA indépendant. Il est aussi possible de migrer vers un circuit de type ASIC en cas d'une production en grande série.

# Processeur embarqués : hardcore/softcore

- lorsque l'on conçoit un système numérique complexe, on met un œuvre généralement un processeur embarqué. Ce processeur embarqué est :
- Soit un bloc IP : on parle de processeur *softcore*.
- Soit déjà implanté dans le circuit électronique en « dur » : on parle de processeur *hardcore*.
  - Le processeur de ce type est généralement plus performant que le processeur du type précédent.

# Approche Altera : IP soft Le ‘NIOS’

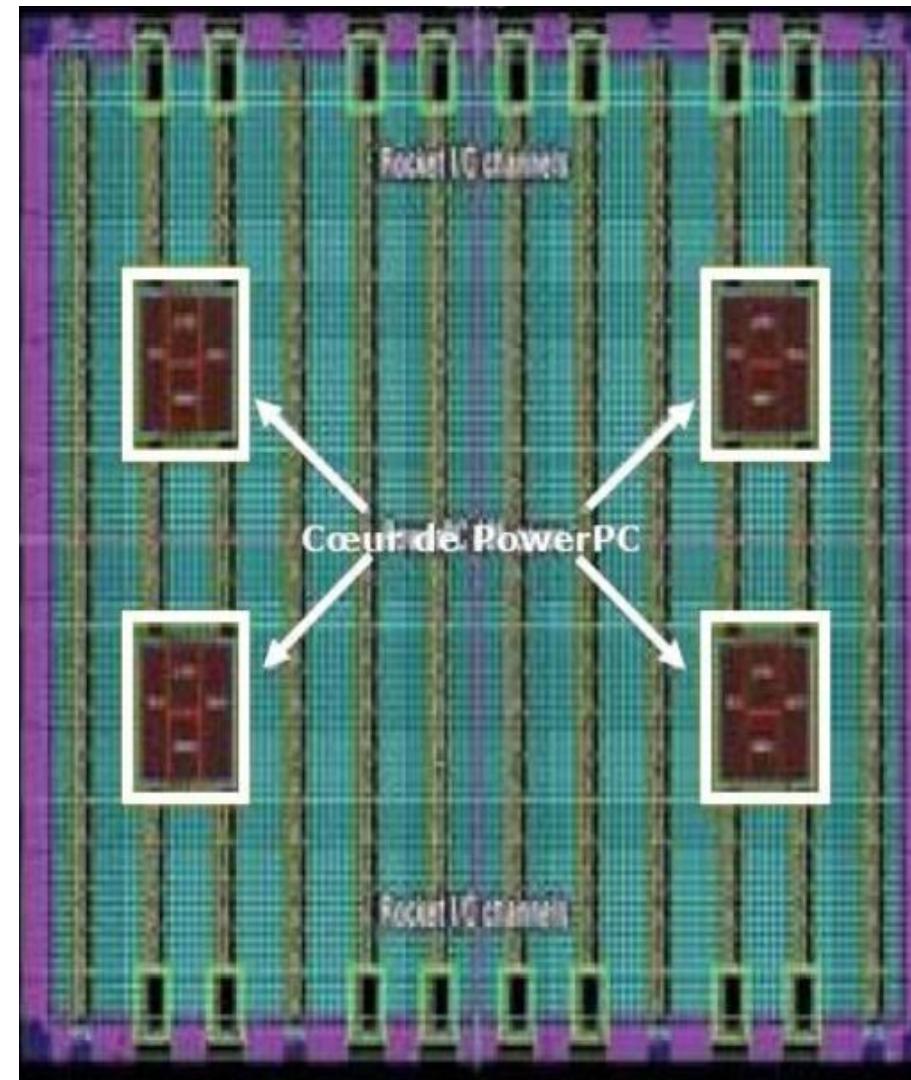
- NIOS : cœur de processeur RISC générique optimisé
- Caractéristiques:
  - données sur 16 ou 32 bits, 128, 256 ou 512 registres
  - registres à décalage rapide (1, 3, 7, 15 ou 31 bits/clock)
  - possibilités de lui adjoindre des périphériques (UART, RAM, ROM)



# Approche Xilinx

## 1. Hardcore : XilinxVIRTEX II PRO (XC2VP):

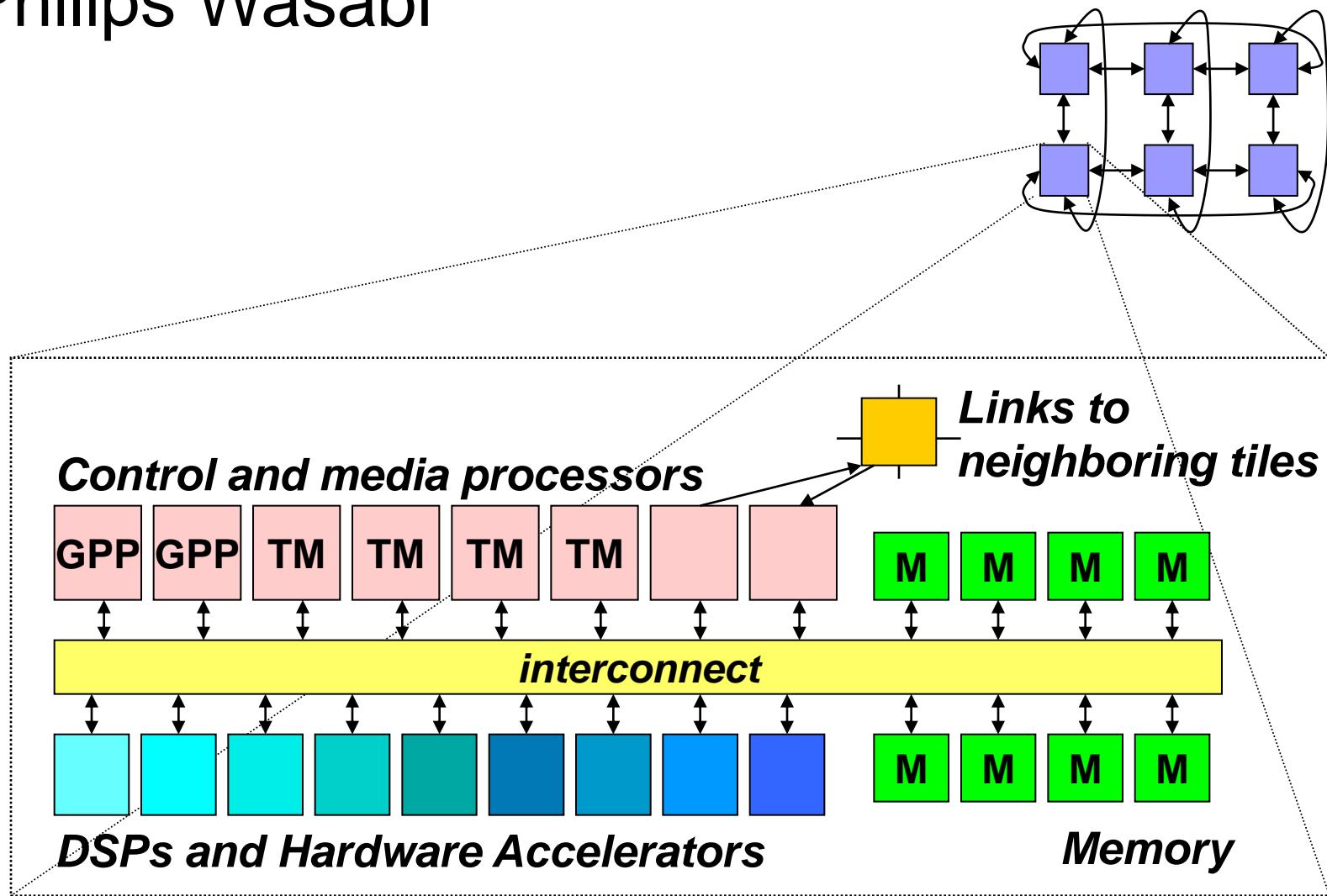
- une matrice configurable
  - 1 500 000 de portes
  - De 216 Kbits à 8 Mbits de mémoires
  - De 204 à 1164 I/Os
- 1, 2 (ou 4) cœurs de processeur PowerPC 405 (32 Bits) à 400MHz (**hard**)
  - 16 Koctets de cache instructions
  - 16 K octets de cache données
- Prix: ~ 1 500 \$ max



## 2. softcore

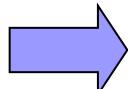
- MicroBlaze, picoblaze

# Philips Wasabi



# Intérêt des cibles mixtes

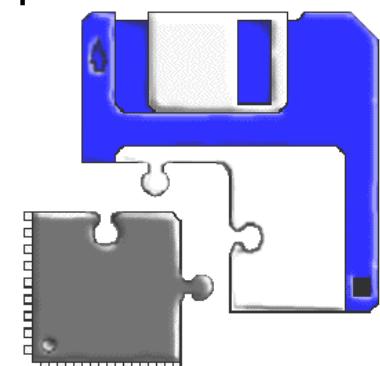
- Bénéficier des avantages des cibles logicielles
  - ?
  - ?
- Bénéficier des avantages des cibles matérielles
  - ?
  - ?



Nécessité d'outil de co-conception Logicielle et Matérielle : Codsing hard and Soft

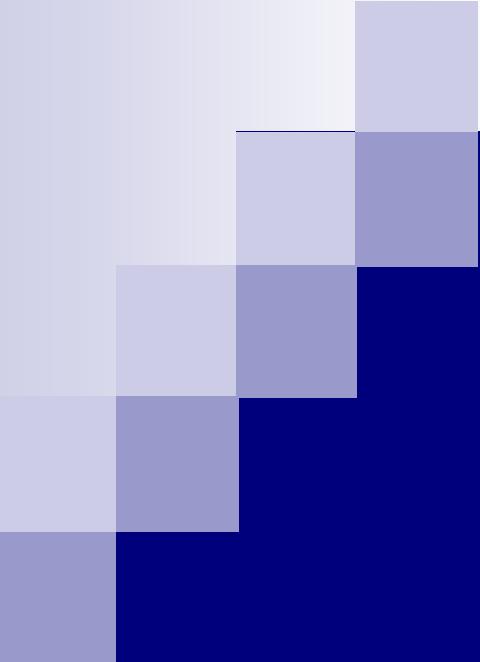
# Le Codesign

- **Définition :** Les méthodes de CoDesign sont des méthodes de développement simultané (de manière concurrente) des parties HW et SW (specification, design, vérification)
- **SW = microprocesseur**                            **HW = FPGA ou ASIC**
- **Buts :**
  - Gérer au mieux l'hétérogénéité de la nature des fonctions qui composent le système (du logiciel à l'architecture reconfigurable)
  - Comparer les différents choix de partitionnement
  - Définir les interfaces entre le SW et le HW
  - Valider le système complet (co-vérification et co-simulation)



# Suite du cours

- Chapitre 2: Cible matérielle FPGA cible
- Chapitre 3: logicielle Microcontrôleur



# Chapitre 2 : Cible matérielle ASIC- FPGA

Nadia Khouja Saad

# Plan du Chapitre 2

1. Introduction : technologie CMOS
2. Niveau d'abstraction pour la modélisation des circuits Intégrés
3. Classification des cibles matérielles
4. Les cibles FPGAs

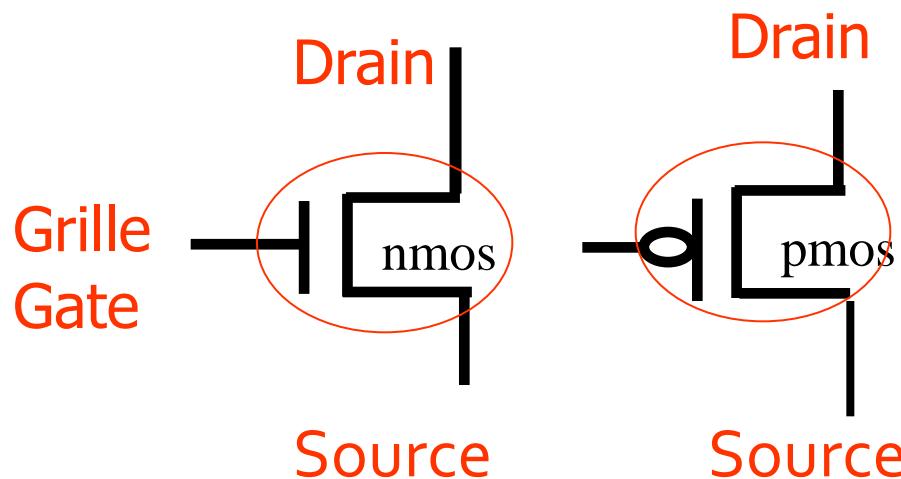
# Introduction

Cibles matérielles sont rendues possibles par les progrès réalisés en technologie d'intégration

- **SSI** : Small Scale Integration (**dizaine** de transistors)
- **MSI** : Medium Scale Integration (**centaine** de transistors)
- **LSI** : Large Scale Integration (**Milliers** de transistors)
  - ➔ Besoin d'outils de conception + automatisation ➔ Outils CAD
- **VLSI** : Very Large Scale Integration (**Centaine de milliers** de transistors)
  - ➔ Outils CAD indispensables pour gérer la complexité de la conception et de la fabrication du Circuit Intégré (CI).

# Les transistors: la technologie de base des CI

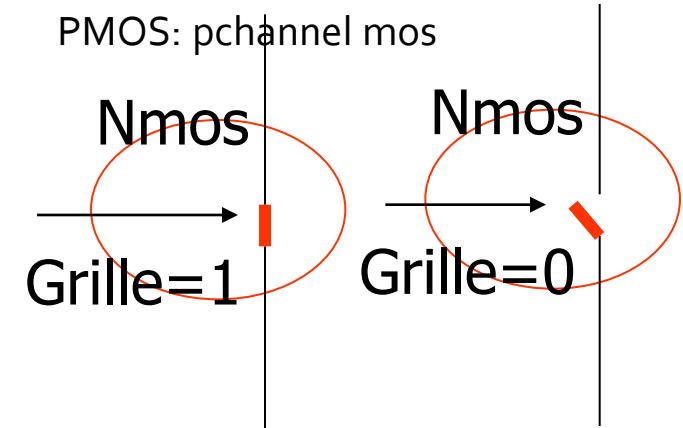
- Les CI (ou chips) représentent la réalisation courante de tout système informatique
- Un CI : un ensemble de transistors + connections



MOS: metal oxyde semiconductor

NMOS: nchannel mos

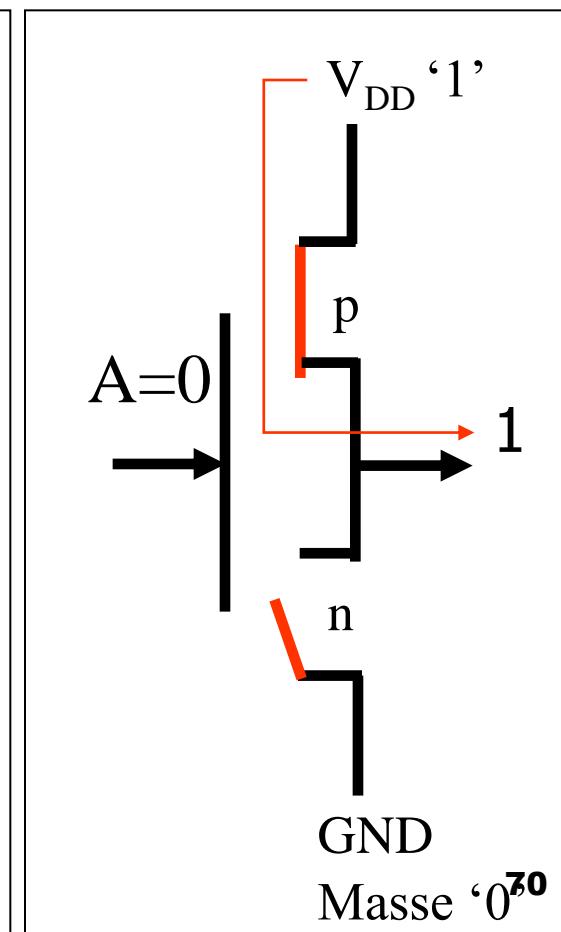
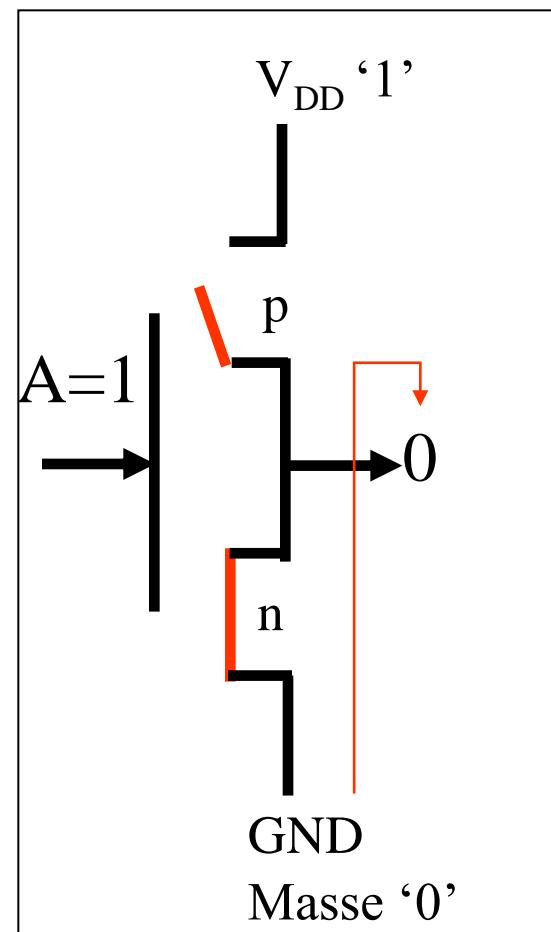
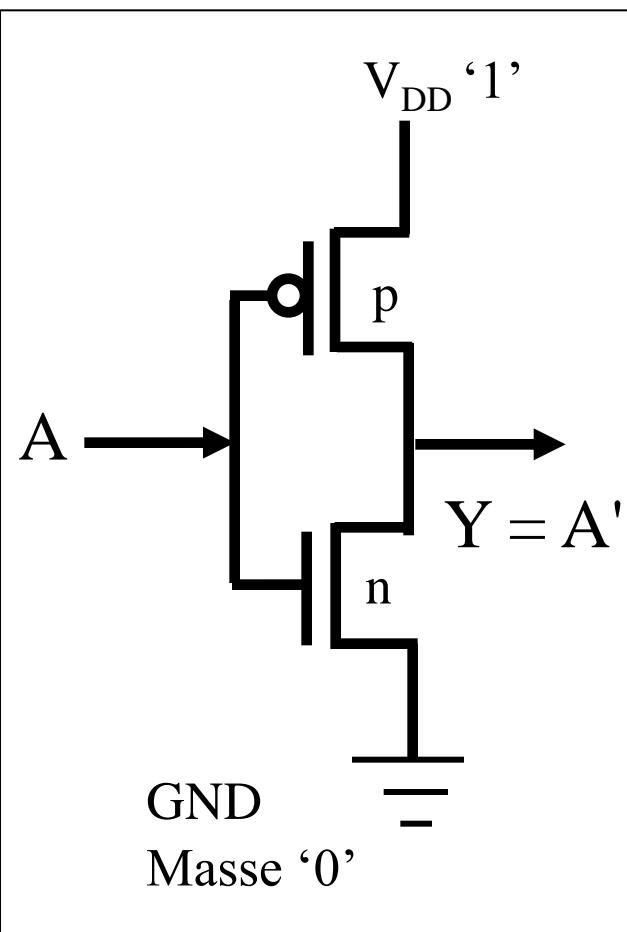
PMOS: pchannel mos



Sur Nmos : Si Grille ==1, alors Drain et Source connectées  
Sur Pmos : Si Grille ==0, alors Drain et Source connectées

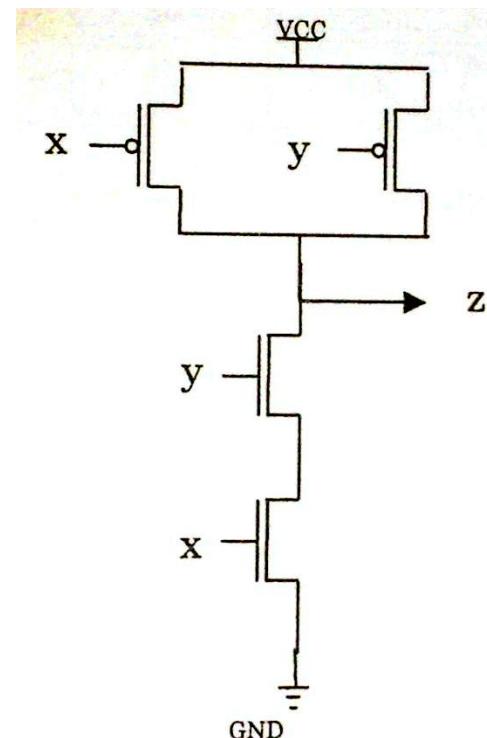
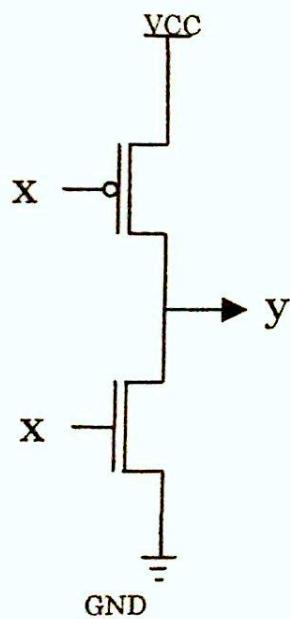
# Un inverseur (NOT) dans la technologie CMOS

**CMOS** : complementary mos (circuit comportant des Pmos et des Nmos)



# Exercice 1

De quel type de porte s'agit-il ?



$$z = \text{NAND}(x, y)$$

x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

NOT

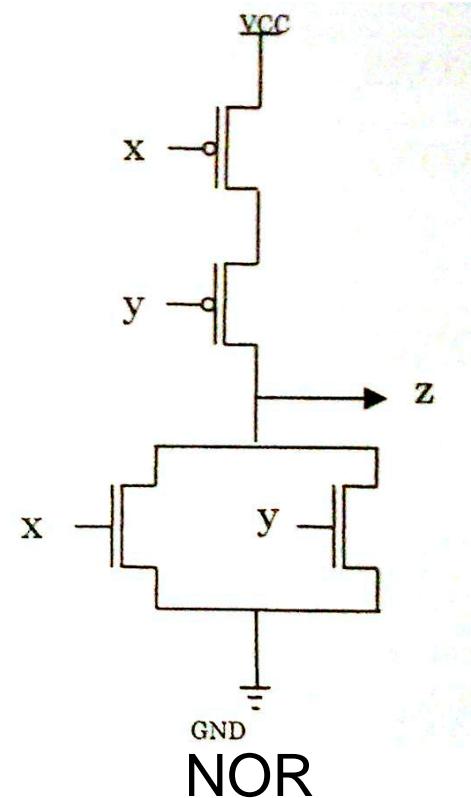
NAND  
Introduction aux systèmes sur puce

# Exercice 2

Comment réaliser une porte NOR en technologie CMOS

$$z = \text{NOR}(x, y)$$

x	y	z
0	0	1
0	1	0
1	0	0
1	1	0



# Niveaux de modélisation des CI

- Les transistors représentent un niveau de modélisation des CI, appelé aussi niveau électrique
- D'autres modèles existent: + détaillés (- abstrait)
  - Dessin de masque (niveau physique)
- D'autres modèles existent: + abstrait (- détaillés)

# Niveaux de modélisation des CI

- Les propriétés d'un système peuvent être spécifiés selon trois domaines:
  - **Comportemental:** Information fonctionnelle du système  
Que fait-il?
  - **Structural:** Information schématique.  
Les différents sous-systèmes et comment sont-ils liés?
  - **Physique:** Information géométrique.  
Taille, forme et placement physique?
- Pour chaque domaine il y a plusieurs niveaux d'abstraction.

# Description d'un système

## ■ **Description structurelle :**

- en explicitant de quoi ce système est fait. On décrit la structure interne d'un bloc. Par exemple : un *microprocesseur contient un ensemble de registres, une ALU, un UC, des caches, et le tout est connecté de telle et telle façon*".
- Les sous-blocs (registres, ALU) peuvent eux-aussi éventuellement être modélisés à partir de sous-blocs, etc, jusqu'à arriver à des blocs de base ultra-simples (bascules D).

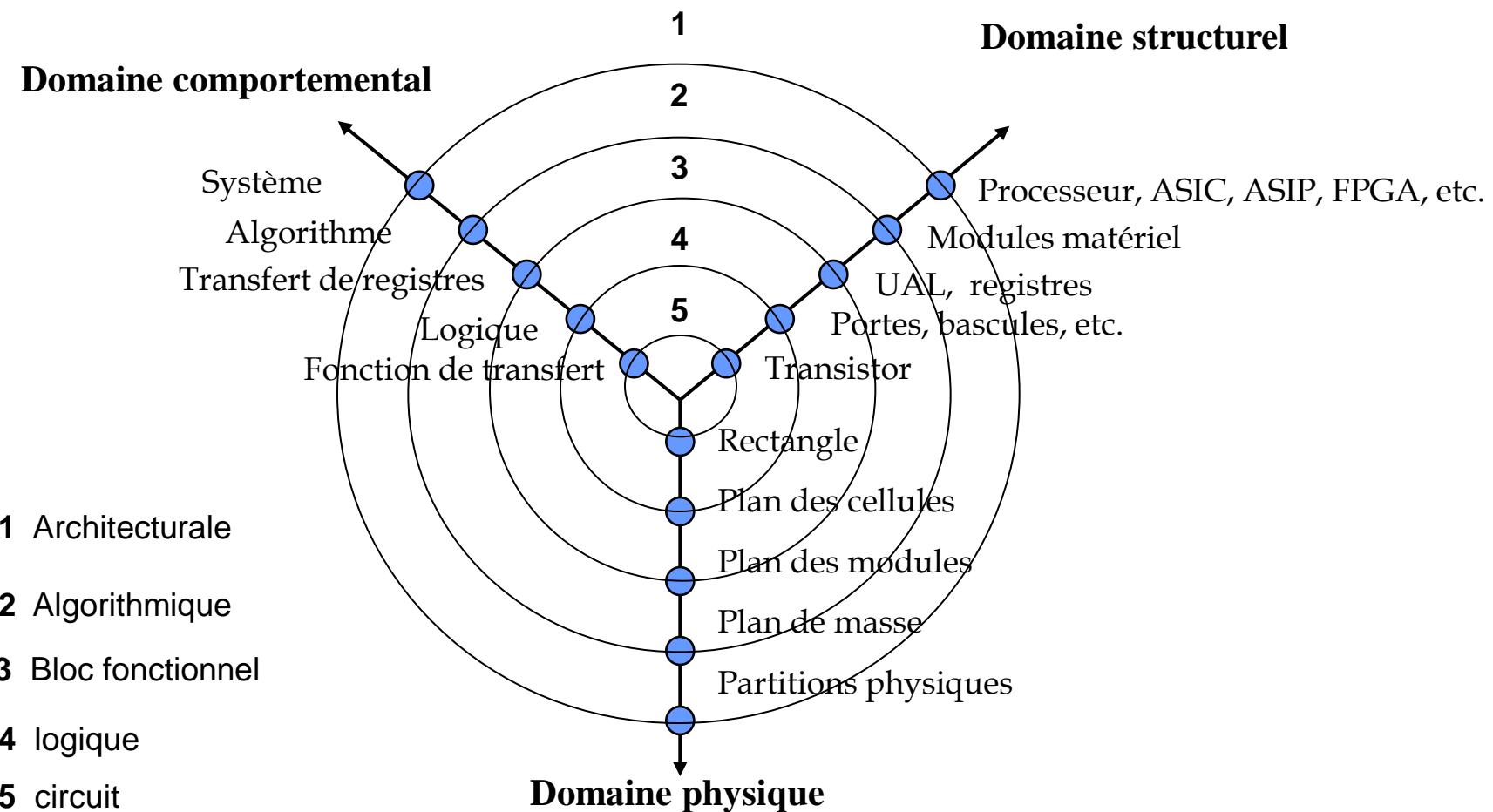
## ■ **Description comportementale:**

- en décrivant son comportement : on se moque de ce qu'il y a dedans, seul ce qui compte c'est ce qu'il fait. Un bloc est donc une boîte noire, donc on décrit le fonctionnement à l'aide d'une ou plusieurs fonctions.

# Description d'un système

- En pratique, on commence par une description comportementale (car les spécification d'un système sont souvent données sous forme algorithmique).
- Puis, à force de simulations et de mesures de performances, on essaye d'identifier des grands blocs. La description devient alors mixte : une partie structurelle donnant la liste des blocs et la façon dont ils sont reliés entre eux, alors que les blocs eux sont encore représentés sous forme comportementale.
- On ré-applique ensuite ce procédé récursivement à chaque bloc (décomposition en sous-blocs, sous-sous-blocs), jusqu'à obtenir une description presque totalement structurelle à base de bascules D et fonctions logiques simples.
- Des outils se chargent alors de traduire tout ça automatiquement en portes logiques.

# Diagramme en Y de Gajski

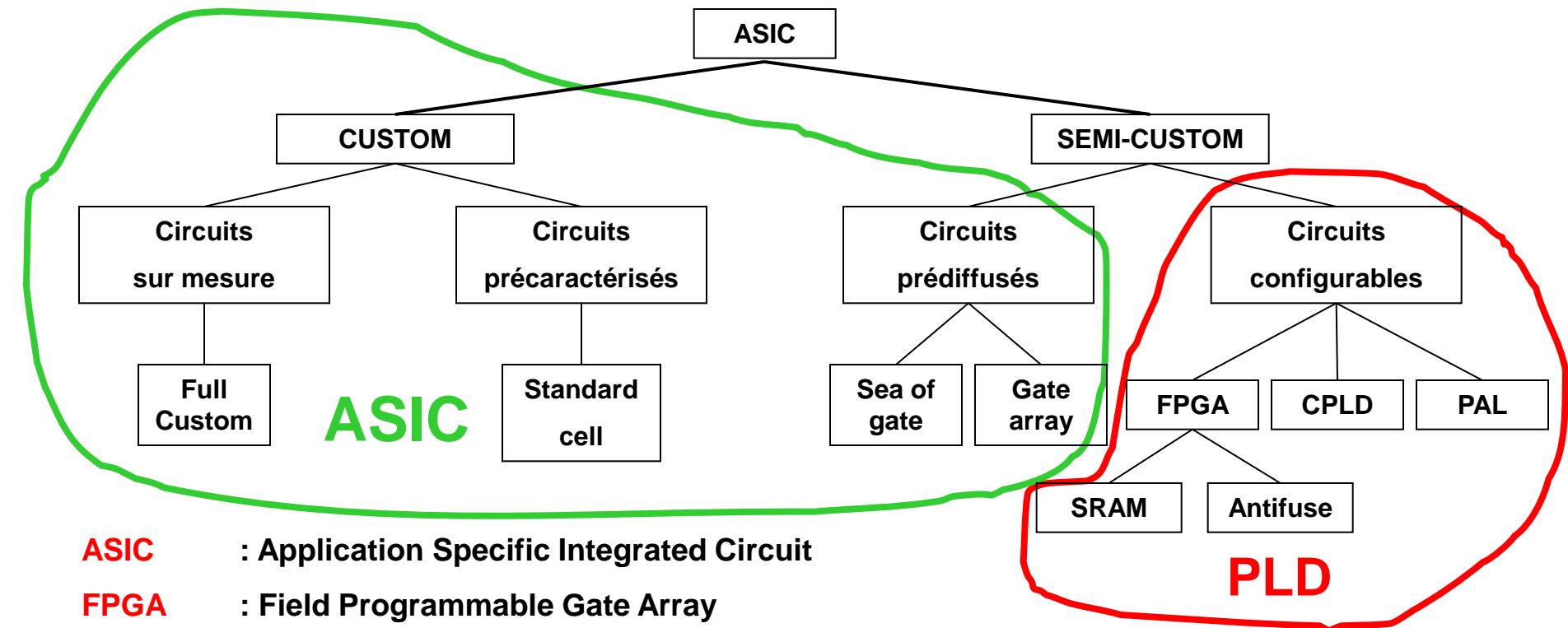


→ Chaque cercle correspond à un niveau d'abstraction : Abstraction croissante de l'intérieur vers l'extérieur.

# Conception des circuits VLSI

- Circuit VLSI (**centaines de milliers** de transistors)
  - ➔ Automatisation du procédé de fabrication
  - ➔ Vérification basée sur la simulation (au lieu des cartes)
  - ➔ Utilisation de langages de description du matériel (HDL)

# Les composants (HARD)



**ASIC** : Application Specific Integrated Circuit

**FPGA** : Field Programmable Gate Array

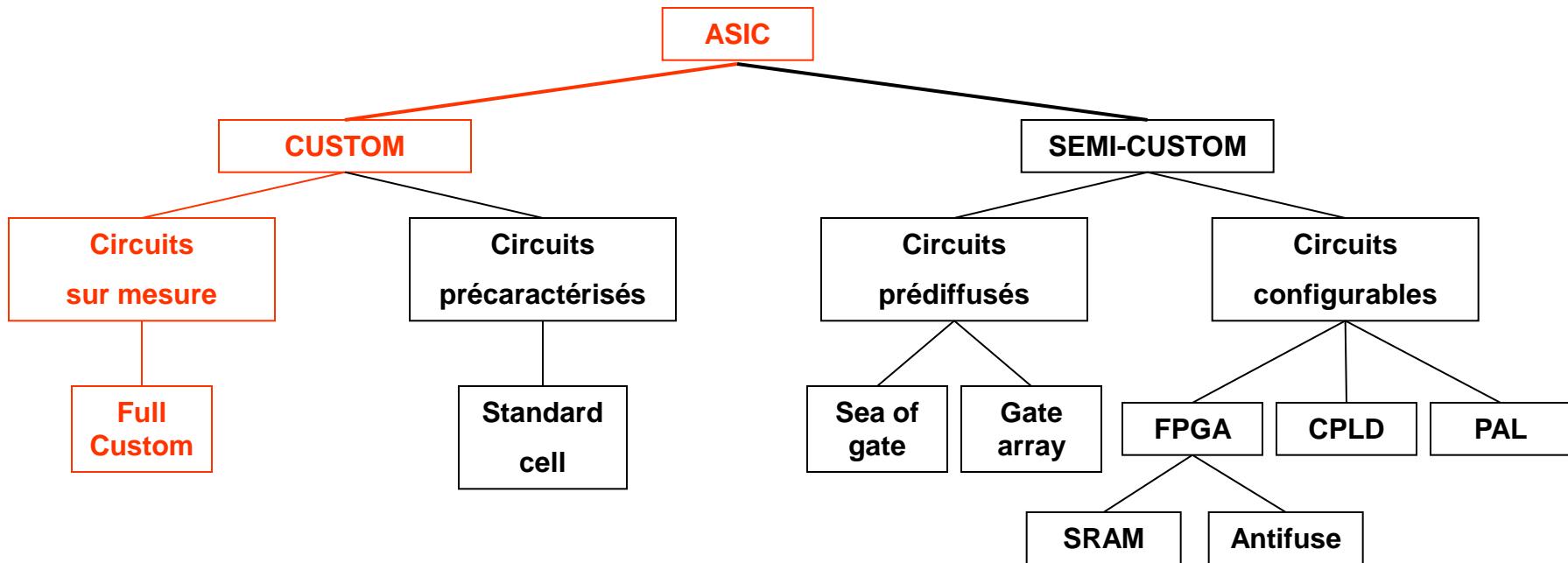
**CPLD** : Complex Programmable Logic Device

**PAL** : Programmable Array Logic

**GAL** : Generic Array Logic = PAL

**SRAM** : Static Random Access Memory

# ASIC: Full Custom



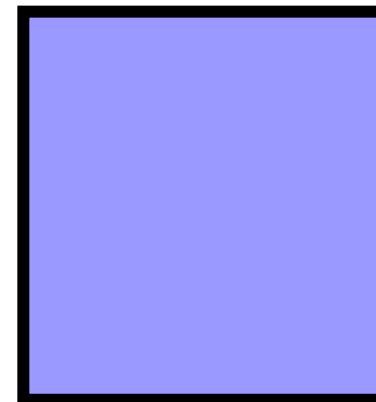
**ASIC** : Application Specific Integrated Circuit

# ASIC: Full Custom

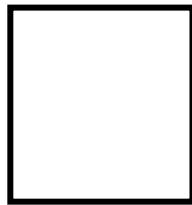
Au départ



Au final



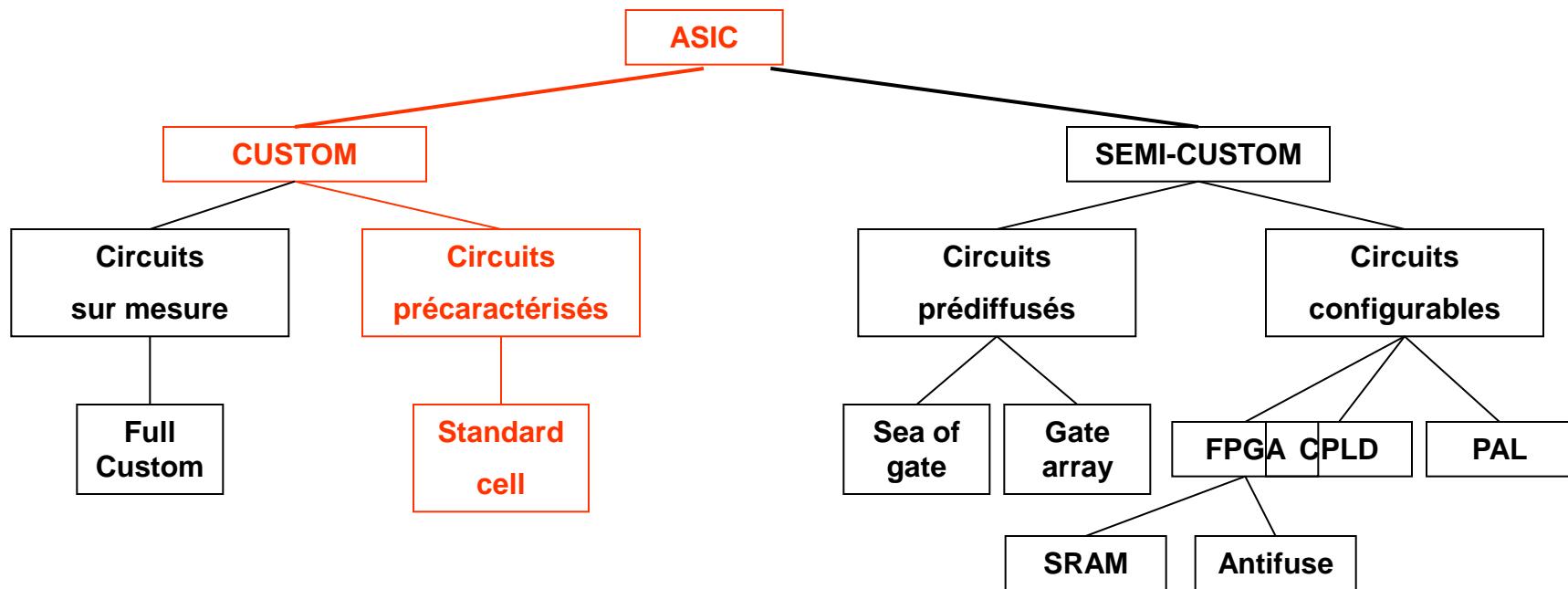
+



## ■ Approche « *full custom* »

- Conception au niveau transistor
- Permet des circuits mixtes analogique/numérique
- Effort de conception très important
- Surface réduite, performance très importantes

# ASIC: Standard Cell



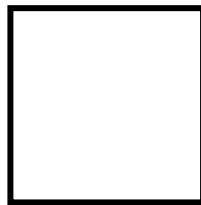
**ASIC** : Application Specific Integrated Circuit

# ASIC: Standard Cell

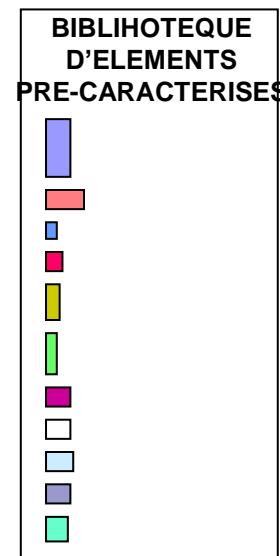
Au départ



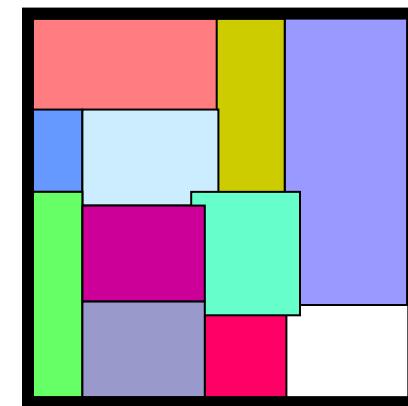
+



+



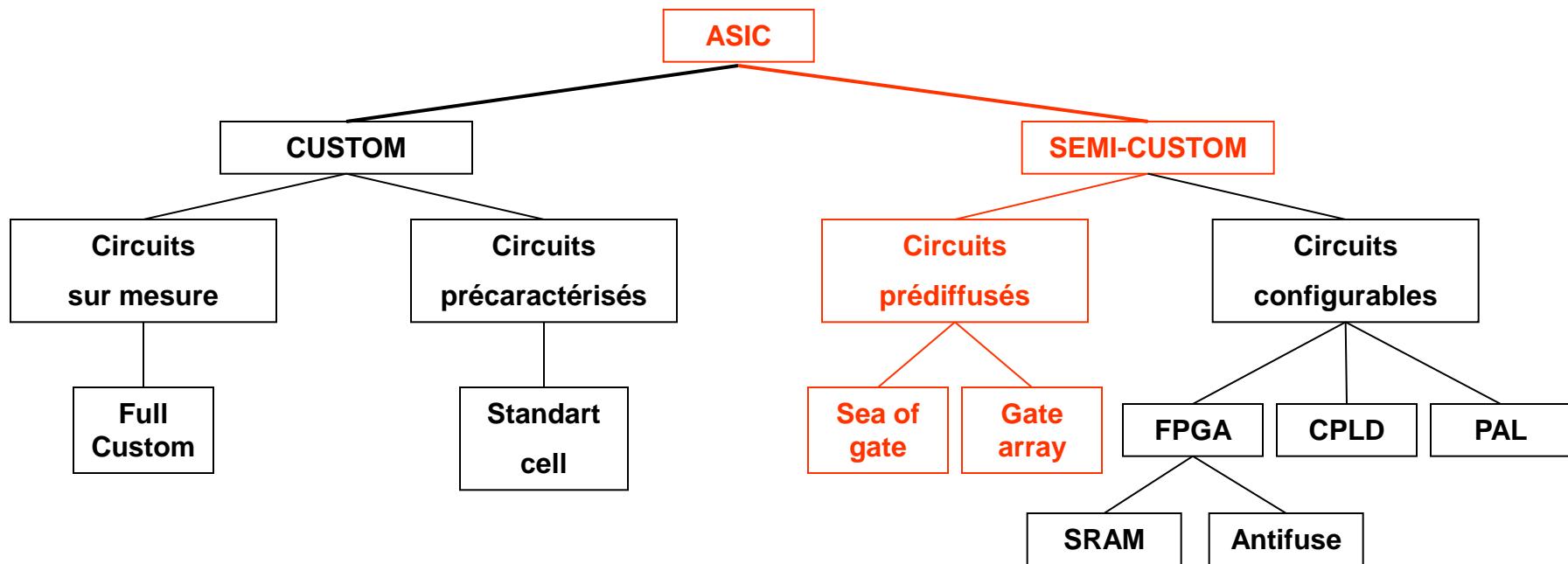
Au final



## ■ Approche « *standard cell* »

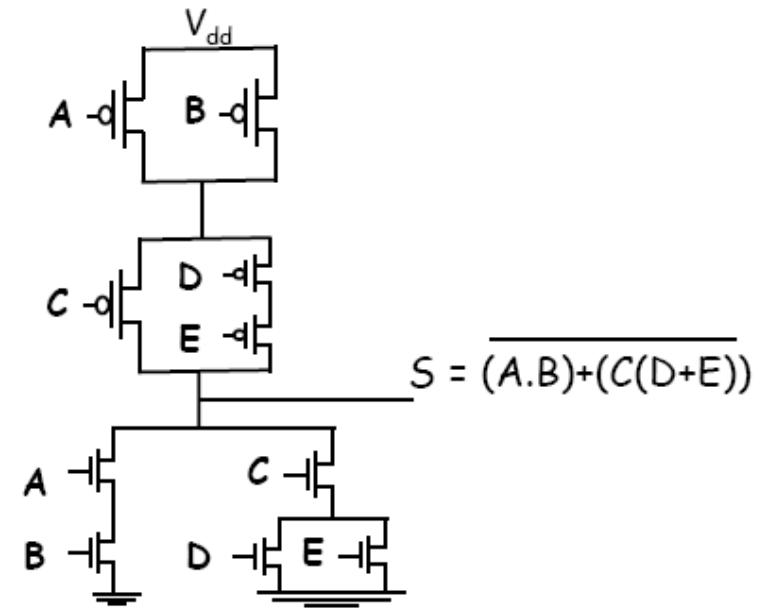
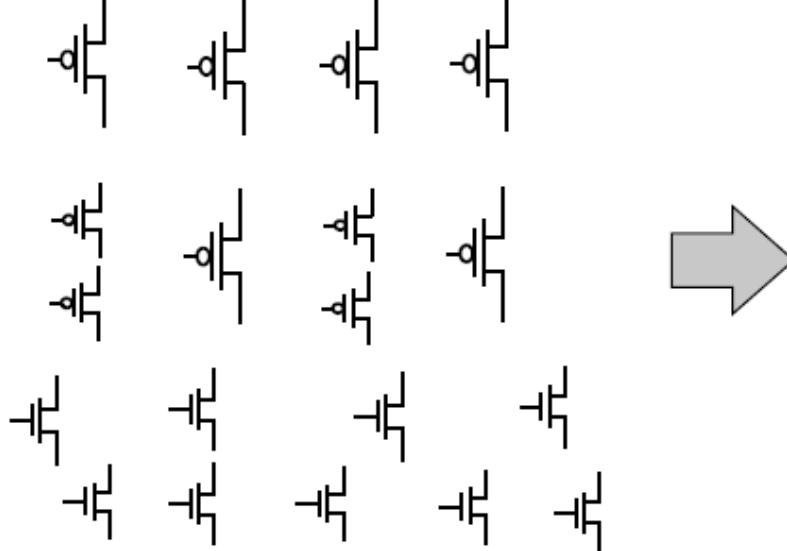
- Utilise des librairies de cellules primitives
  - Portes AND, OR, registres, SRAM, etc.
- Effort de conception réduit, performances souvent proches du full-custom

# ASIC:Circuits prédiffusés



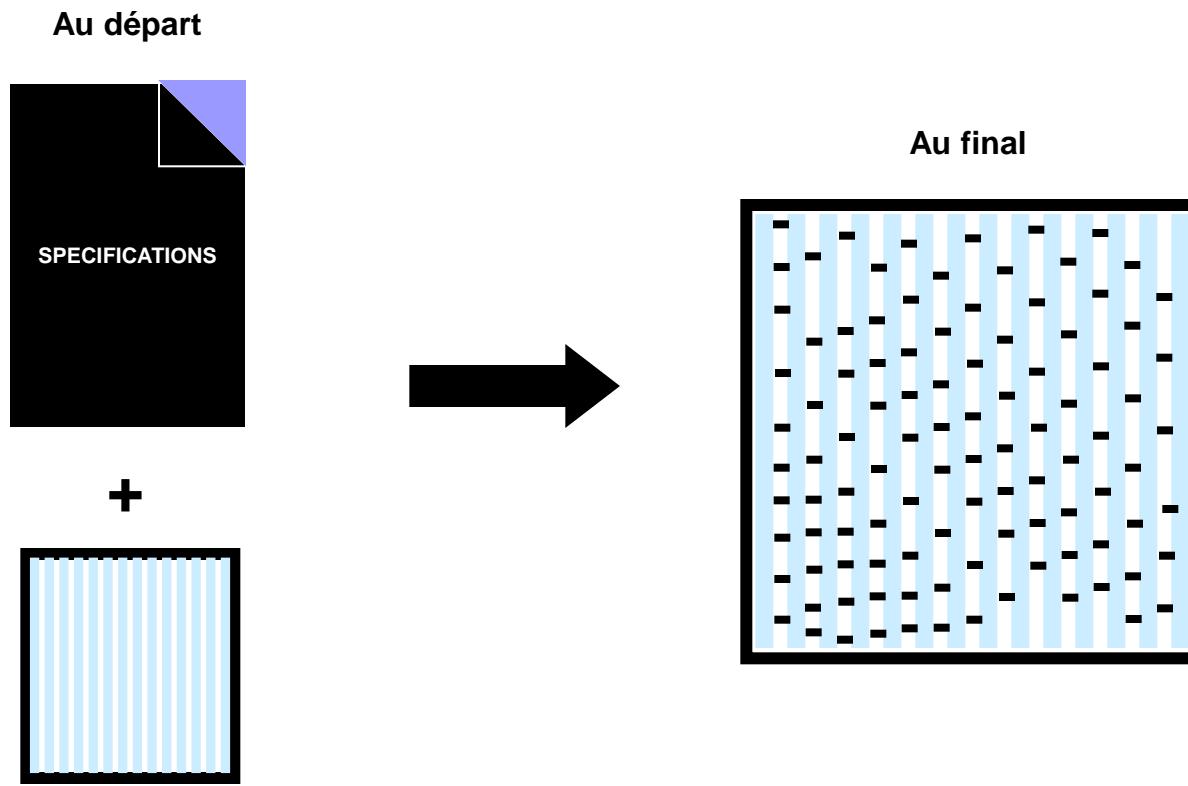
**ASIC** : Application Specific Integrated Circuit

# ASIC: circuits prédiffusés



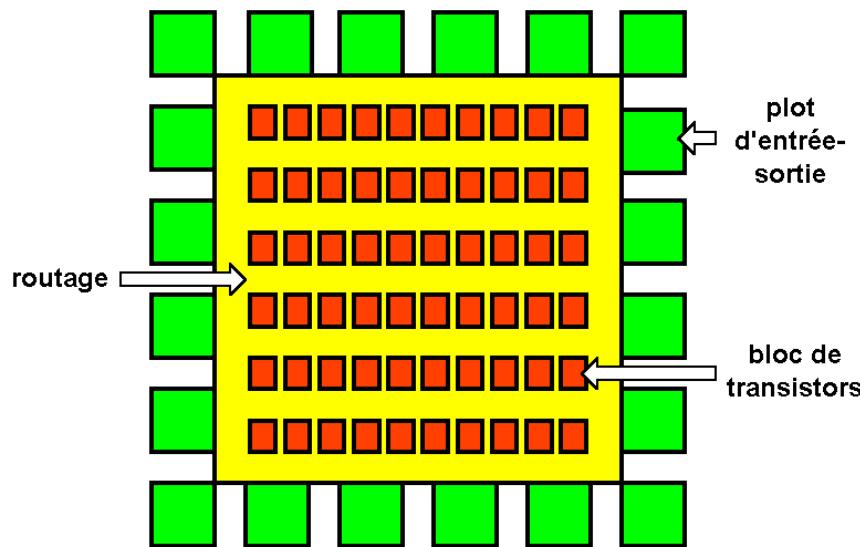
- ❑ Réseaux de transistors déjà implantés dans le silicium, mais non interconnectés.
- ❑ C'est l'interconnexion des transistors qui personnalise le circuit en fonction de l'application visée.

# ASIC: Gate Array

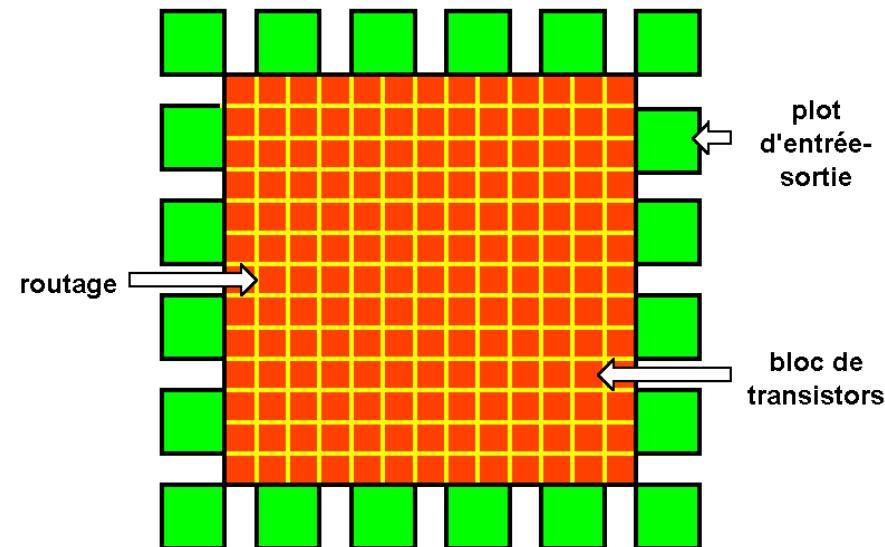


- Bandes de cellules de hauteur fixe séparées par des canaux de routage
- Il est possible aussi d'utiliser des bibliothèques de composants pré-caractérisés

# ASIC: Sea of Gates



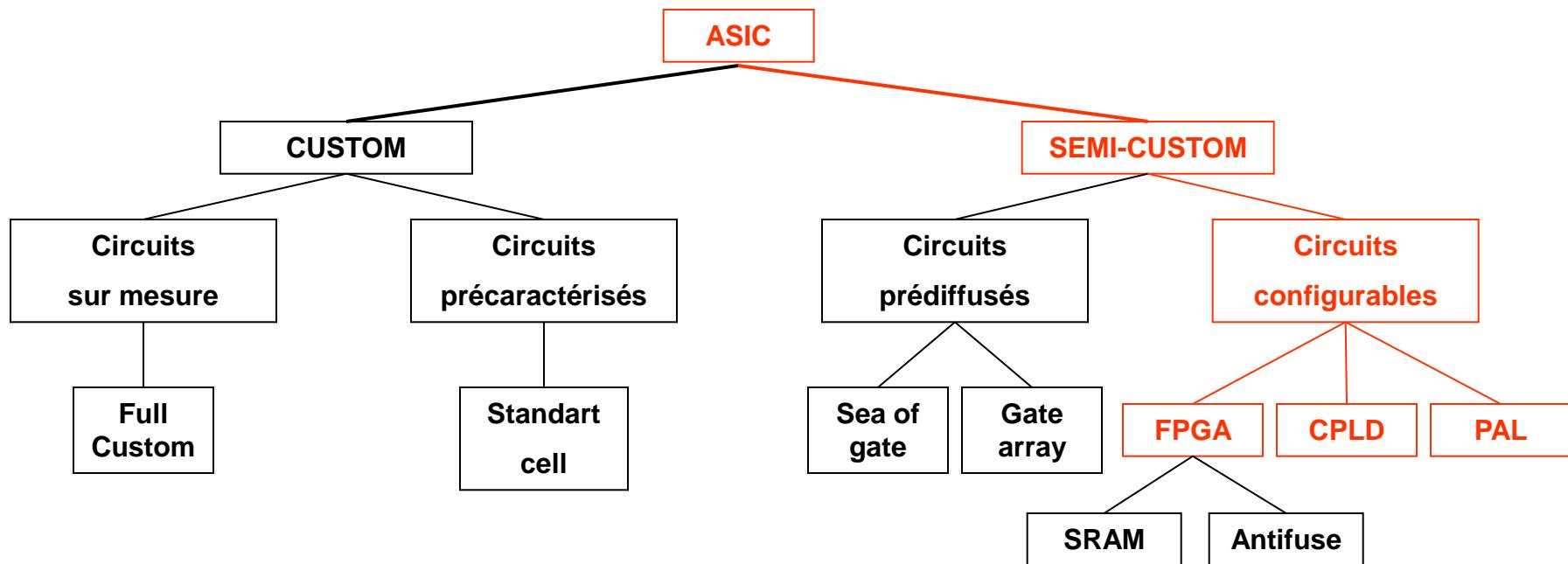
Gate array



Sea of gates

- ❑ circuit composé de blocs contigus de transistors
- ❑ les connexions passent entre les blocs ou par-dessus

# ASIC: les circuits configurables



# Les circuits configurables: définition

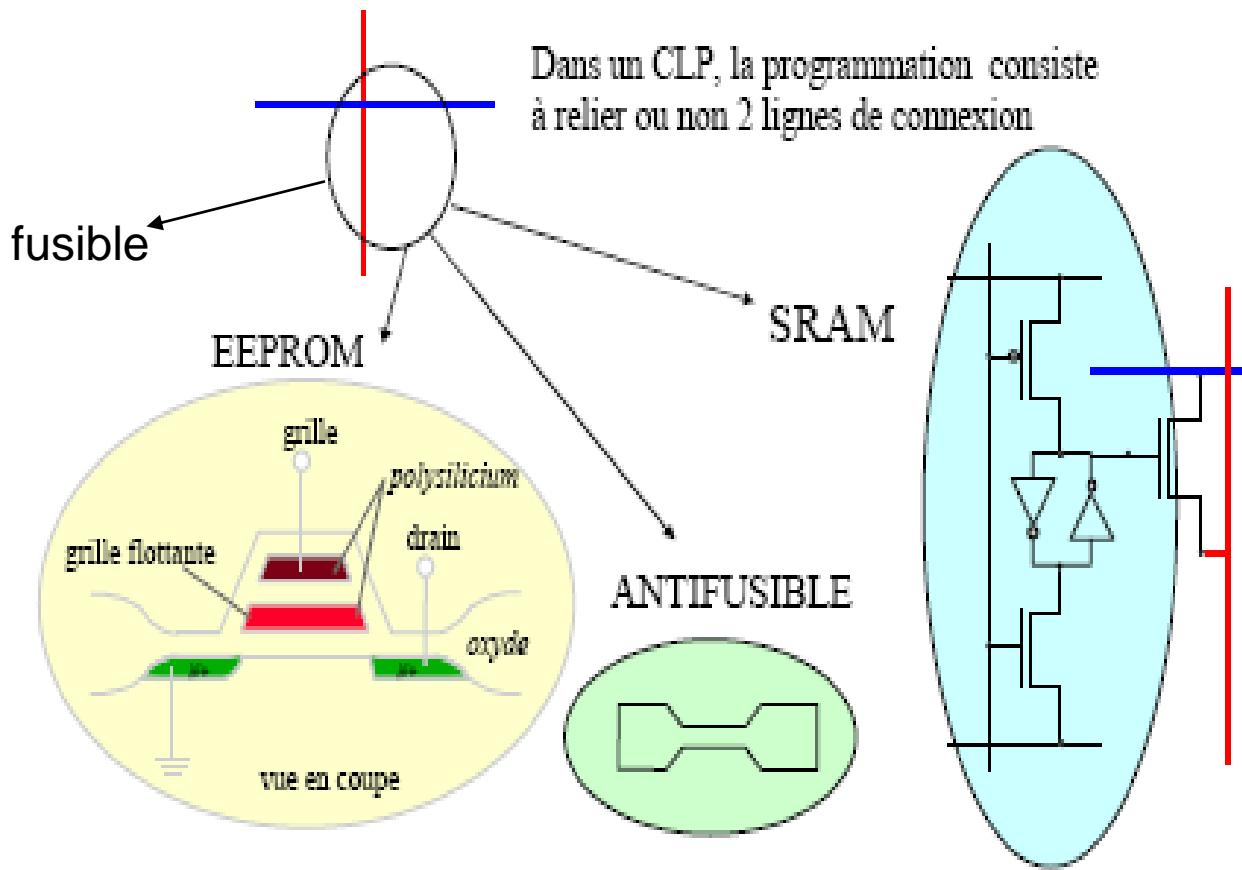
Composants standard programmables électriquement:

- Une seule fois (fusibles, antifusibles)  
ou
- Plusieurs fois: reprogrammables

# Les circuits configurables: principe des architectures

- Ensemble de **ressources logiques** (portes, bascules,...etc) qui peuvent être **interconnectées de différentes façons**.
- Réalisation de fonctions booléennes sous forme d'une somme de produits (PAL, PLA)
- Un réseau de bloc logiques configurables (FPGA)

# Les circuits configurables: technologie de programmation



Le choix se fait selon des critères:

- La vitesse de programmation
- La densité d'intégration
- La facilité de mise en œuvre ( programmation sur site, reprogrammation)

# Les circuits configurables: technologie de programmation

- Programmation une seule fois (PAL)
  - Fusibles (métal)
  - Antifusable (CAPACITE MOS),
- Nombre de configurations limité(EEPROM)
  - Transistor MOS à grille flottante
- Configuration à chaque mise sous tension (FPGA)
  - SRAM

# Technologie de programmation

## ■ Fusible :

- destruction d'un fusible par passage d'un courant supérieur à celui de l'alimentation.

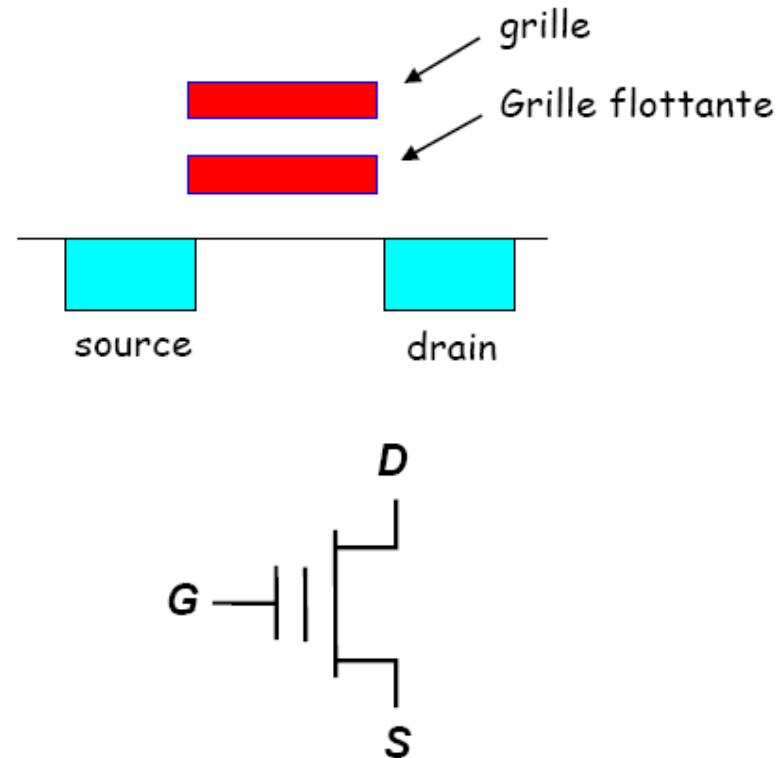
## ■ Antifusible:

- en appliquant une tension importante (16V pendant 1ms) à une zone isolante entre deux zones semi-conductrice fortement dopée, ce dernier diffuse dans cette zone et la rend conductrice.
- Chaque cellule occupe  $1,8\mu\text{m}^2$  ( $700\mu\text{m}^2$  pour un fusible)
- Hormis la non reprogrammabilité, c'est la meilleure technologie (vitesse et surtout une très haute densité d'intégration)

# Technologie de programmation:

## Transistor MOS à grille flottante (EPROM)

- L'apparition du transistor MOS à grille flottante a permis de rendre le composant bloqué ou passant sans l'application permanente d'une tension de commande.
- L'application d'un potentiel sur la grille supérieure provoque le passage d'une partie des électrons du canal à travers la mince couche d'oxyde, ce qui charge la grille flottante. Lors de la lecture, une tension appliquée sur la grille supérieure est complètement masquée par la charge négative emmagasinée sur la grille flottante. Cela équivaut à un transistor toujours bloqué.
- Pour passer d'un 0 (transistor bloqué) à un 1 (transistor passant), on applique une tension sur la source. Cela a pour effet d'attirer les électrons hors de la grille flottante.



# Technologie de programmation:

## Transistor MOS à grille flottante (EPROM)

- UV-EPROM : n'est pas utilisé dans les PLD
- EEPROM
  - Reprogrammation d'une façon sélective (par partie)
  - Une cellule nécessite 6 transistors pour sa réalisation (75 à 100  $\mu\text{m}^2$  en CMOS 0,6 $\mu\text{m}$  → une surface importante réduit la densité d'intégrations)
  - Le nombre de programmations est limité (100 CMOS 0,6 $\mu\text{m}$ ) à cause de la dégradation des isolants
  - La programmation ou l'effacement d'une cellule dure qlq millisecondes
- Flash EPROM
  - N'autorise pas une programmation sélective
  - Une cellule nécessite 2 transistors → une densité d'intégration importante: trois à 4 fois plus que EEPROM mais 10 fois moins que l'antifusible
  - Le nombre de cycle de programmation est plus grands que EEPROM
  - Temps de programmation de qlq dizaine de us et temps d'effacement de qlq ms
  - Inconvénient: nécessite un tension supplémentaire (de 12V) de programmation et d'effacement
  - Programmation in-situ

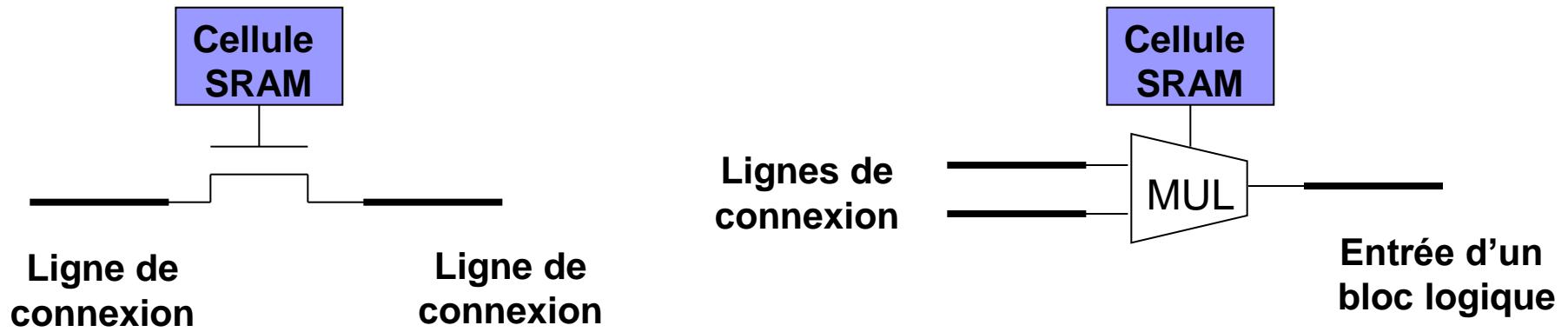
# Technologie de programmation

## Technologie SRAM (FPGA)

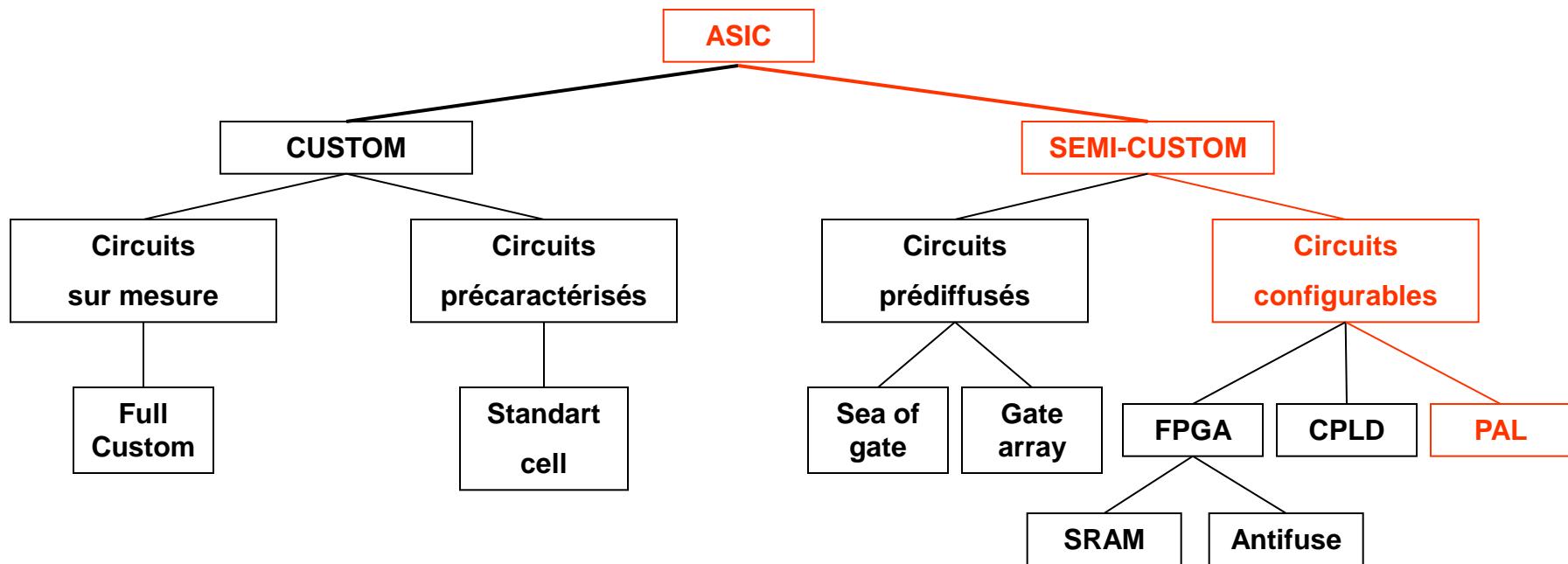
- **Technologie CMOS standard**
- Portes de transmission ou multiplexeurs commandés par des cellules SRAM
- Les mémoires SRAM permettent de configurer les interconnexions et de programmer les cellules
- Programmation illimitée → Programmation à chaque mise sous tension (à partir d'une mémoire externe EEPROM)
- 6 transistors permet un accès sélectif et rapide (qlq ns)
- → Densité d'intégration plus faible (flash EEPROM) mais elle bénéficie de l'évolution technologique importante ( contrairement au EEPROM et flash EEPROM)

# Technologie de programmation

## Technologie SRAM (FPGA)



# PAL

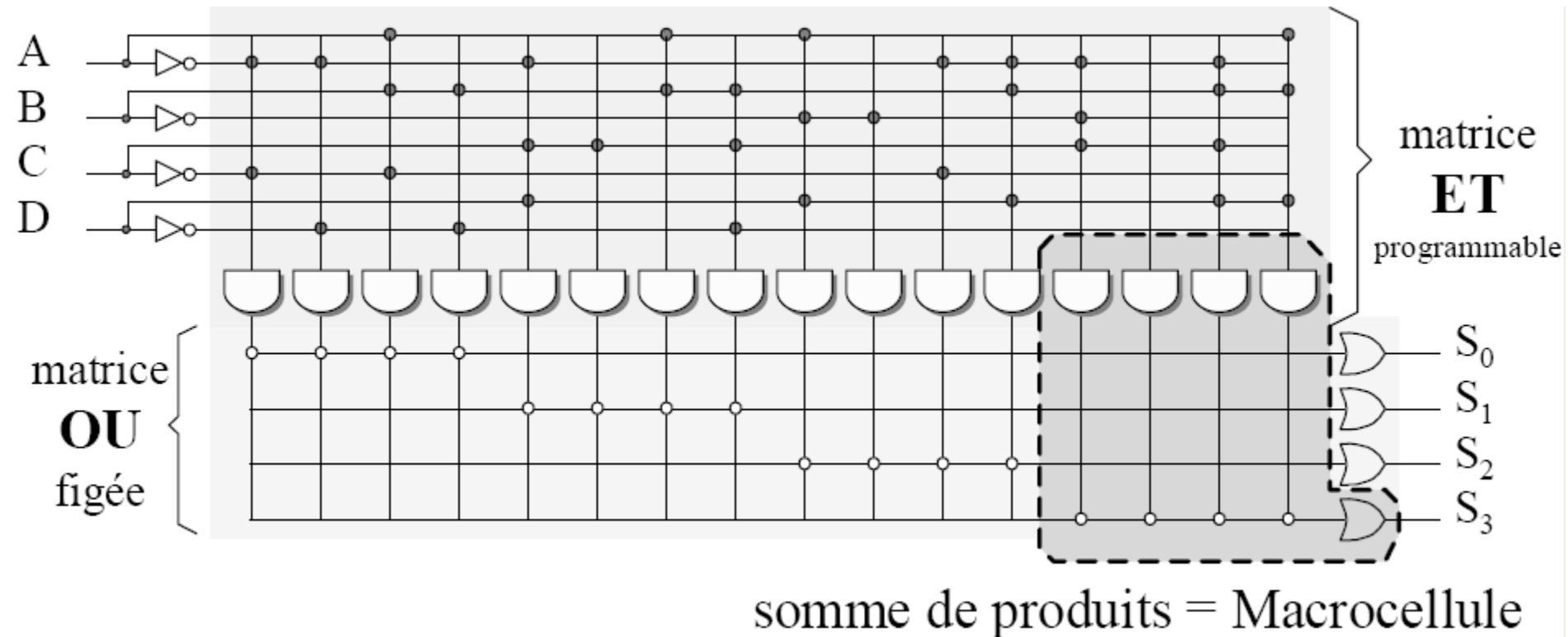


**PAL** : Programmable Array Logic

# PAL

- PAL est le concept de base des PLD (programmable Logic Device)
- PAL: la disposition des transistors et des connections est fixée, mais on peut détruire ou remettre une connexion.
- Principe : Qcq soit la complexité d'une fonction logique, elle peut être écrite sous la forme d'une somme de produits: matrice de portes AND et porte OR

# PAL



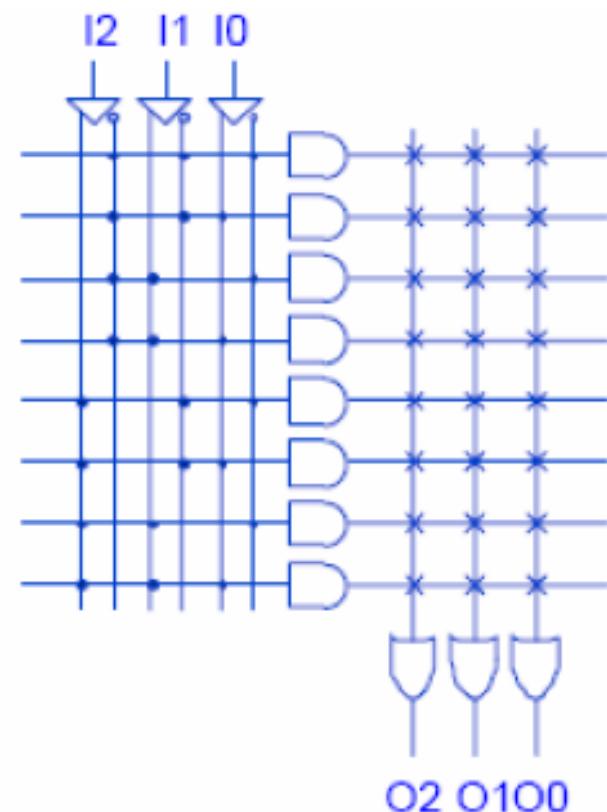
- liaison non programmable
- liaison programmable

# Circuits logiques programmables

- Selon le caractère programmable des matrices AND et OR, il existe :
  - PAL ( programmable array logic):
    - Matrice de ET programmable, matrice OU figée
  - PROM ( programmable read-only memory)
    - Matrice de ET figée, matrice OU programmable
  - PLA ( programmable logic array)
    - Matrice de ET programmable, matrice OU programmable

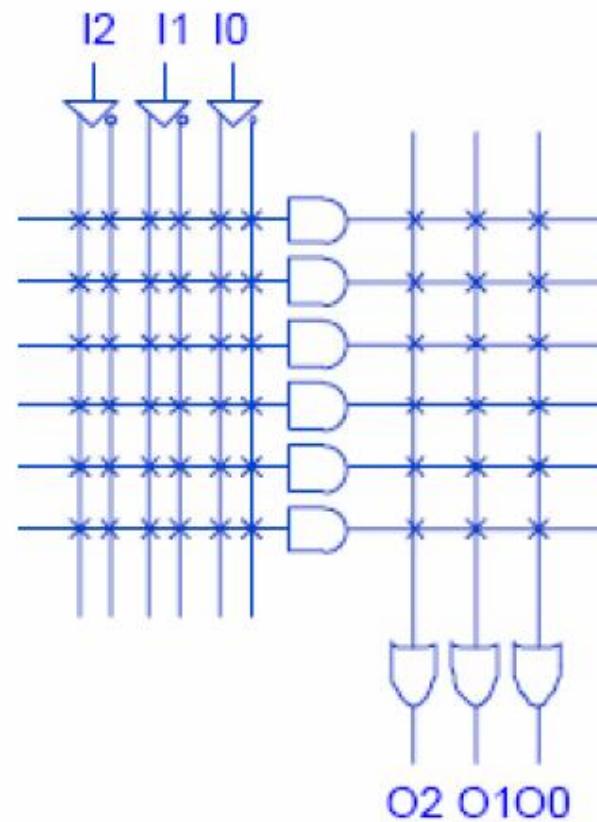
# PROM

- La matrice AND est fixe et la matrice OR est programmable.
- C'est une mémoire:
  - la matrice AND sert de décodeur d'adresse;
  - pour chaque valeur d'adresse, la PROM produit une valeur qui lui a été programmée



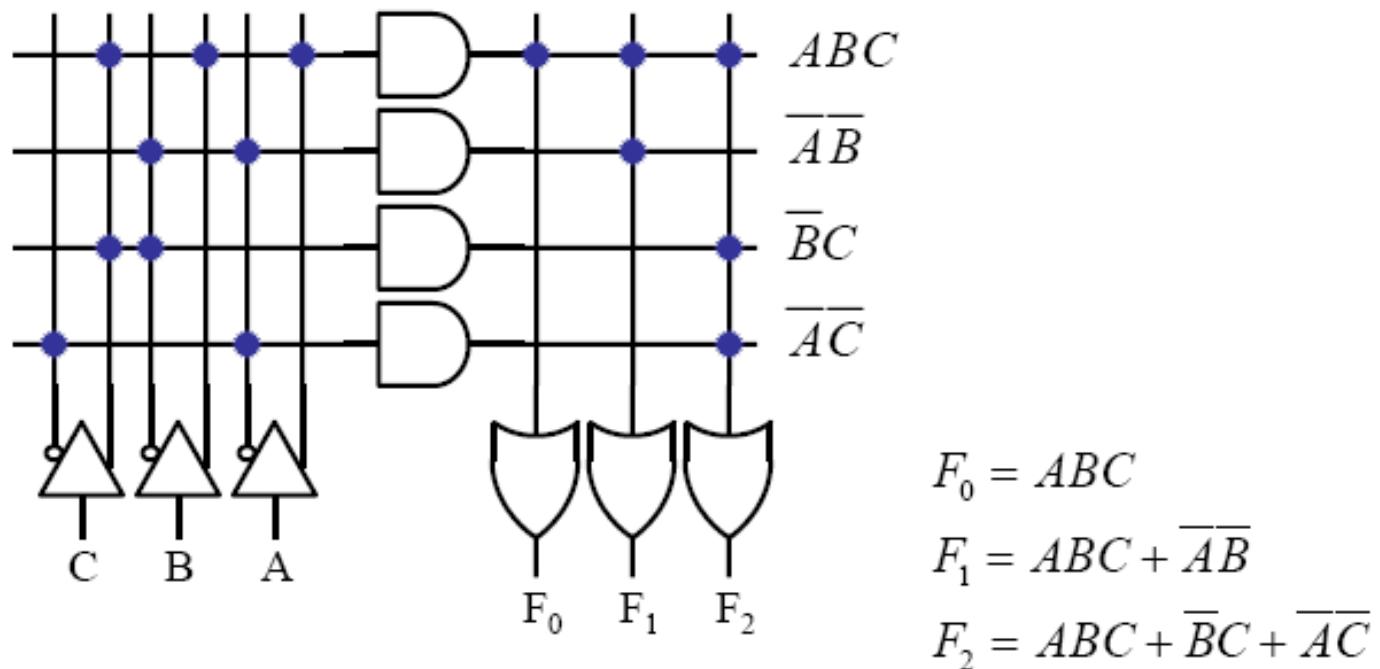
# PLA

- Les deux matrices sont programmables



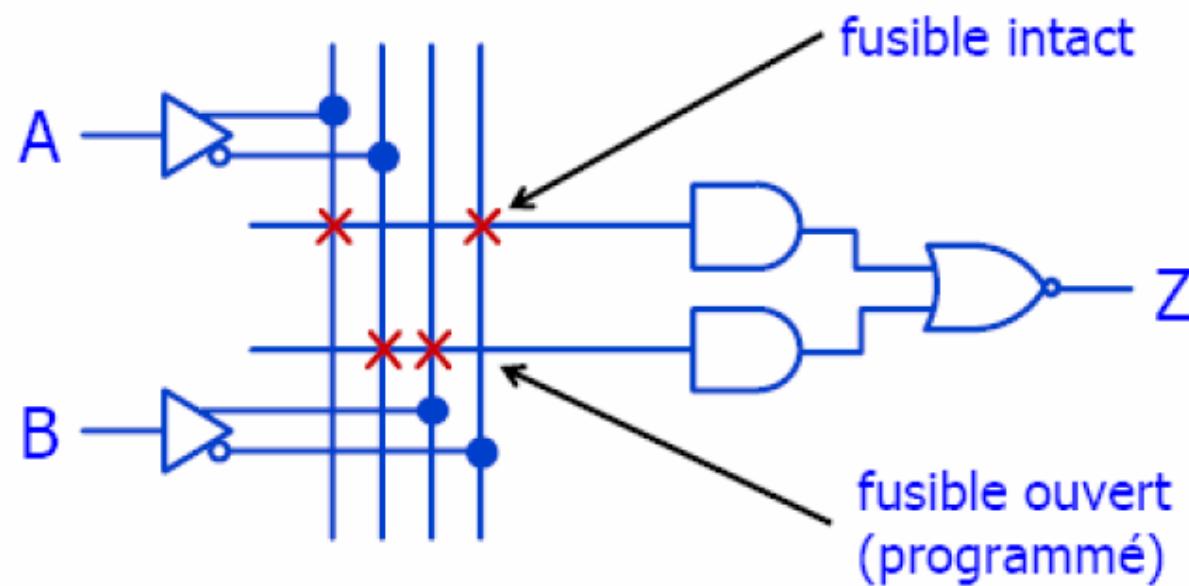
# PLA: exemple

Plans AND-OR : Somme de produits



# PAL: exemple

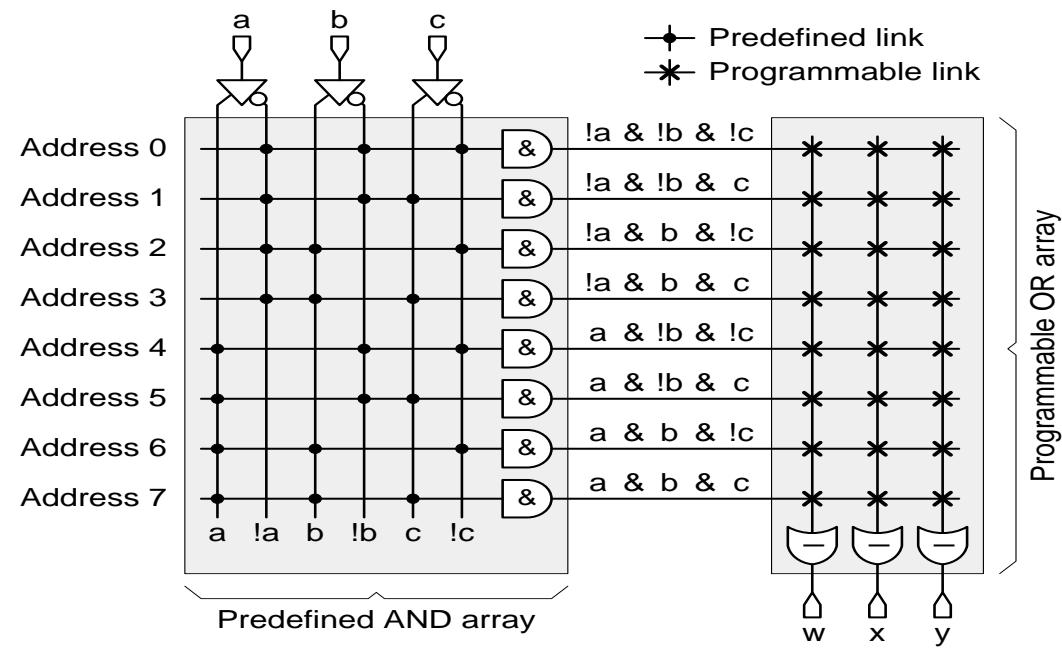
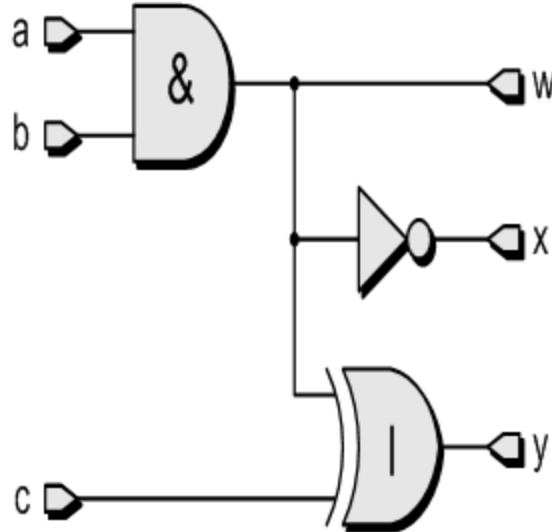
$$Z = \overline{AB} + \overline{AB}$$



# Les circuits configurables:exercice

Donnez le câblage du circuit ci-dessous en utilisant :

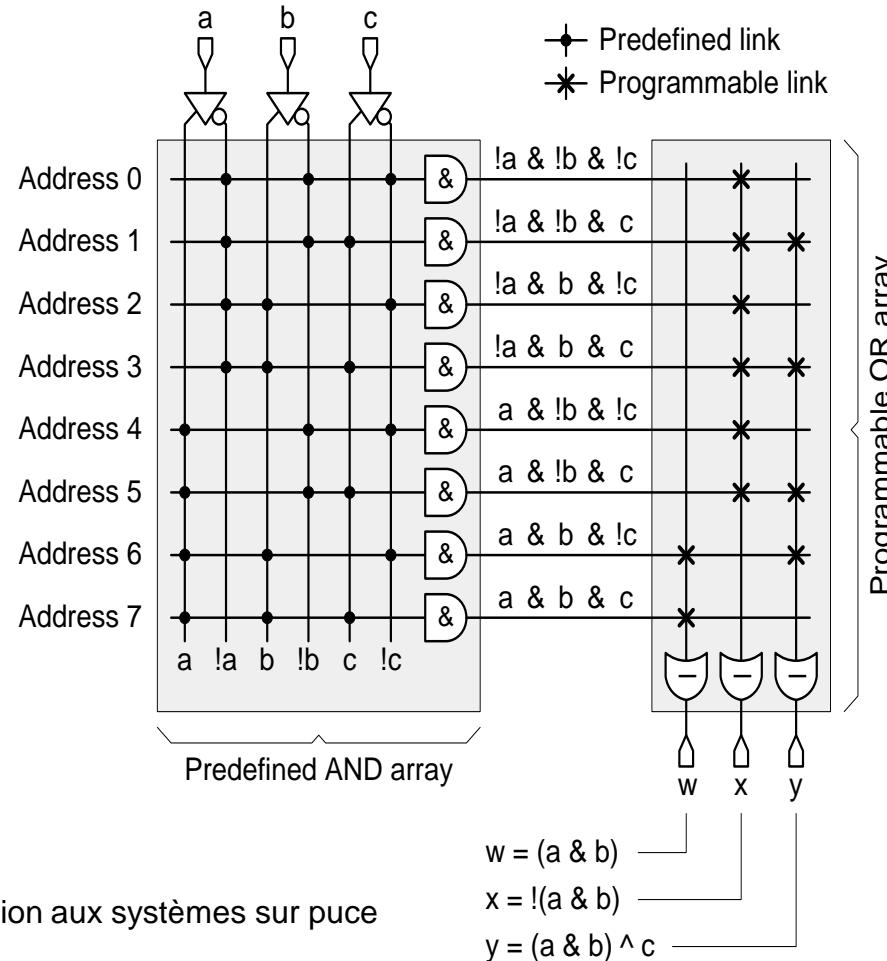
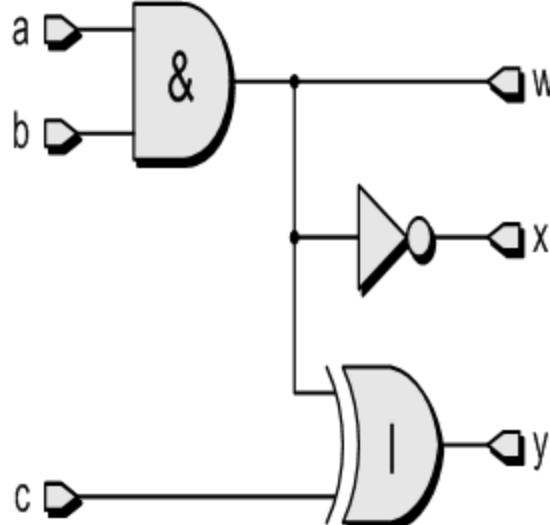
1. PAL
2. PROM



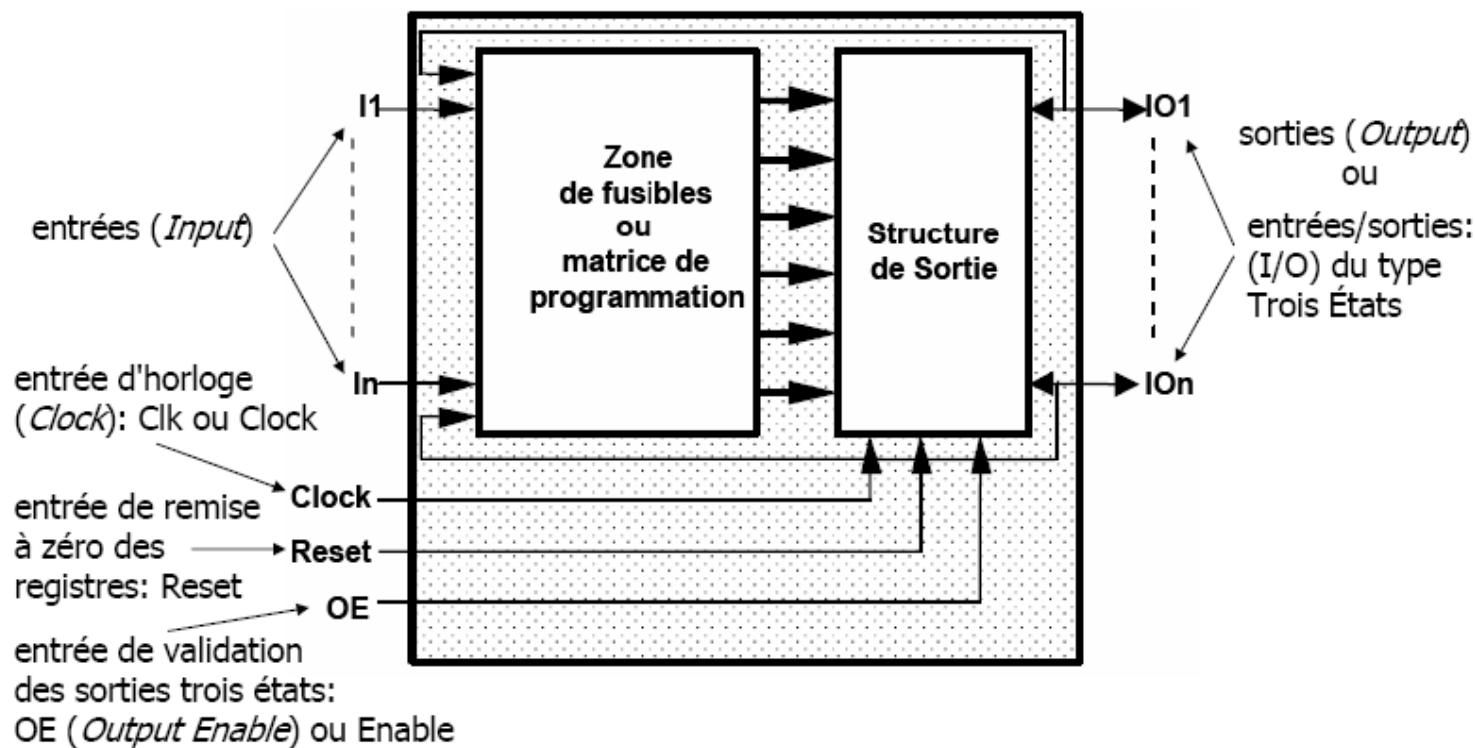
# Les circuits configurables: exercice

Donnez le câblage du circuit ci-dessous en utilisant :

1. PAL
2. PROM



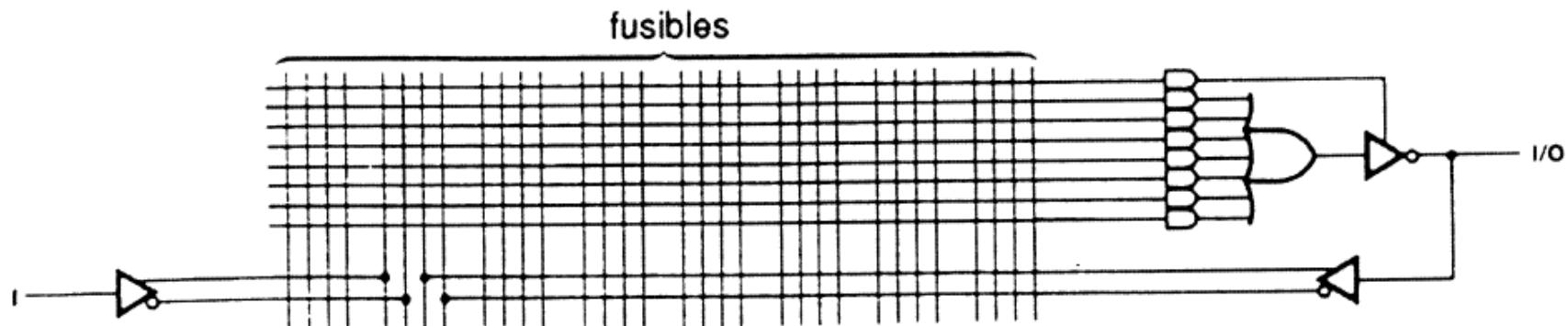
# Les circuits configurables: SPLD(simple programmable logique Device)



# SPLD: Classification selon la structure de sortie

- Trois types de structures de base:
  - Combinatoire
  - Séquentielle
  - Versatile

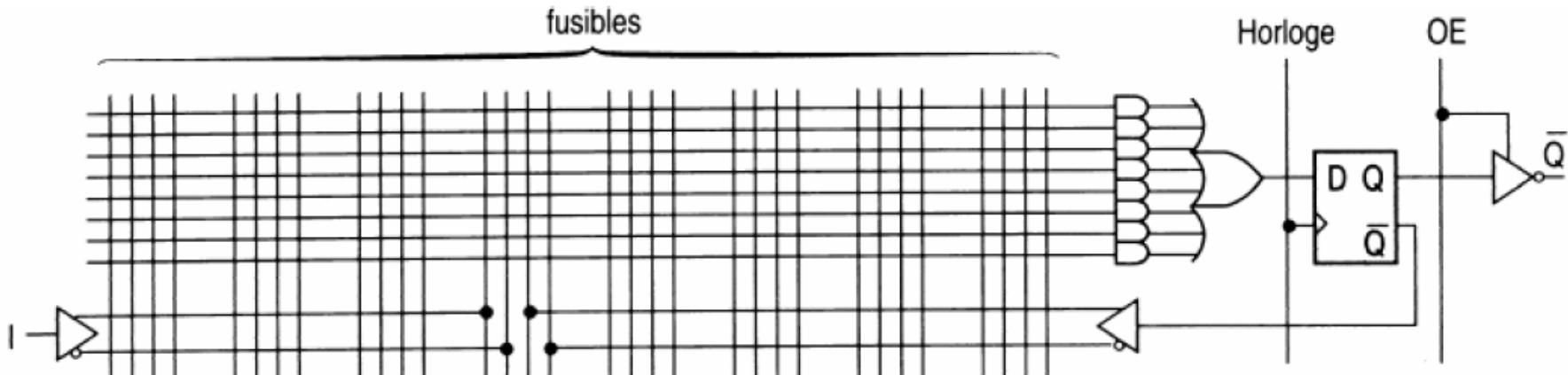
# SPLD:Combinatoire



- Certaines broches peuvent être utilisées aussi bien en entrée qu'en sortie grâce à un système de logique 3 états. La commande de cette dernière est configurée au moment de la programmation.
- La structure de sortie permet aussi de réinjecter les sorties en entrée (Feed-back).

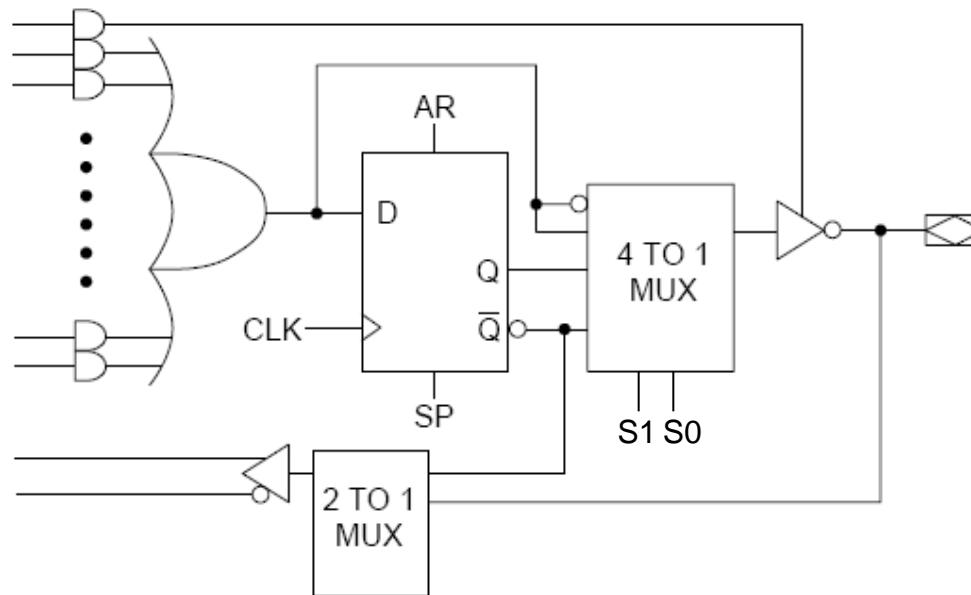
# SPLD: Séquentielle

- Ces circuits sont composés de bascule D. Les sorties des bascules sont de type trois états contrôlées par un signal de validation Enable ou OE, et une horloge est commune à toutes les bascules (clock).

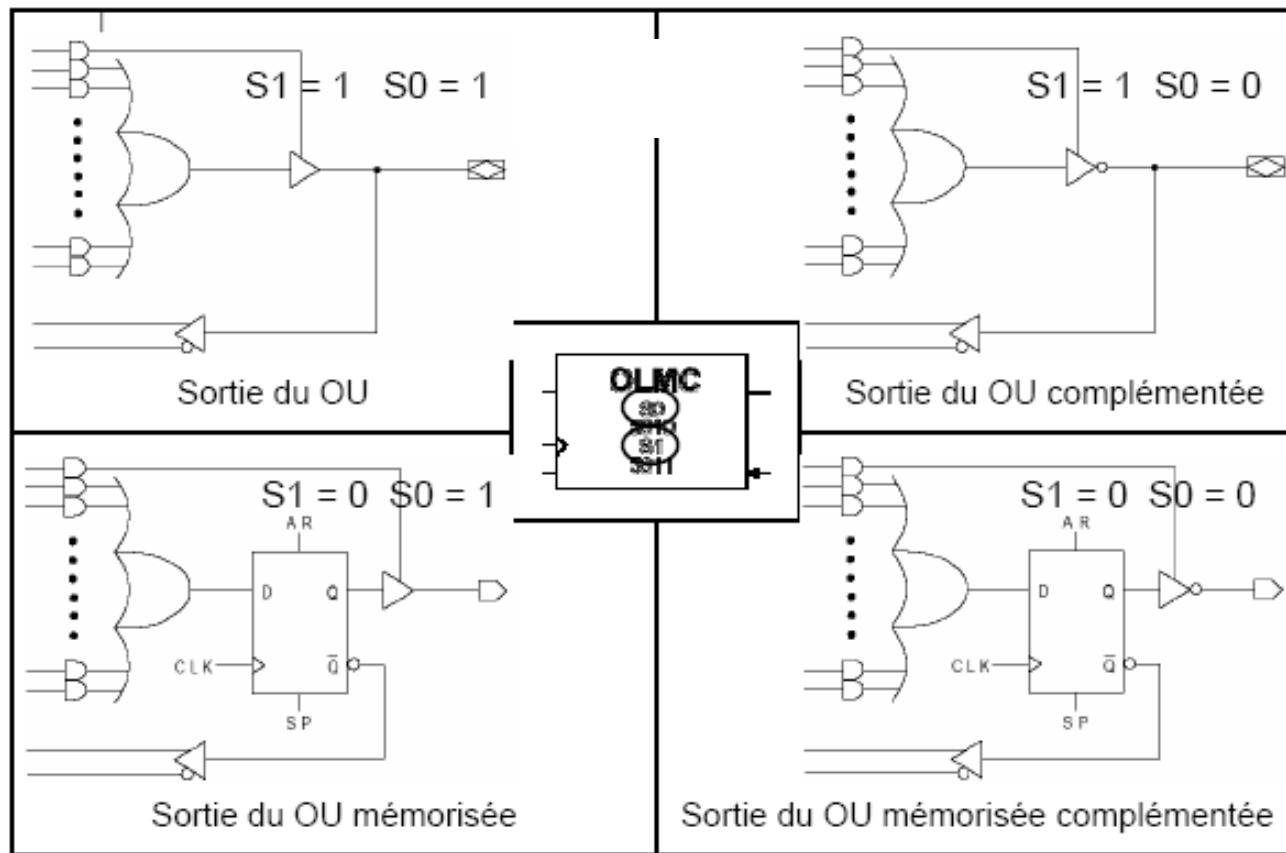


# SPLD: Versatile

- Ce type de structure représente les *P.A.L.* les plus évoluées, car ces structures proposent quatre configurations possibles: combinatoires et séquentielles



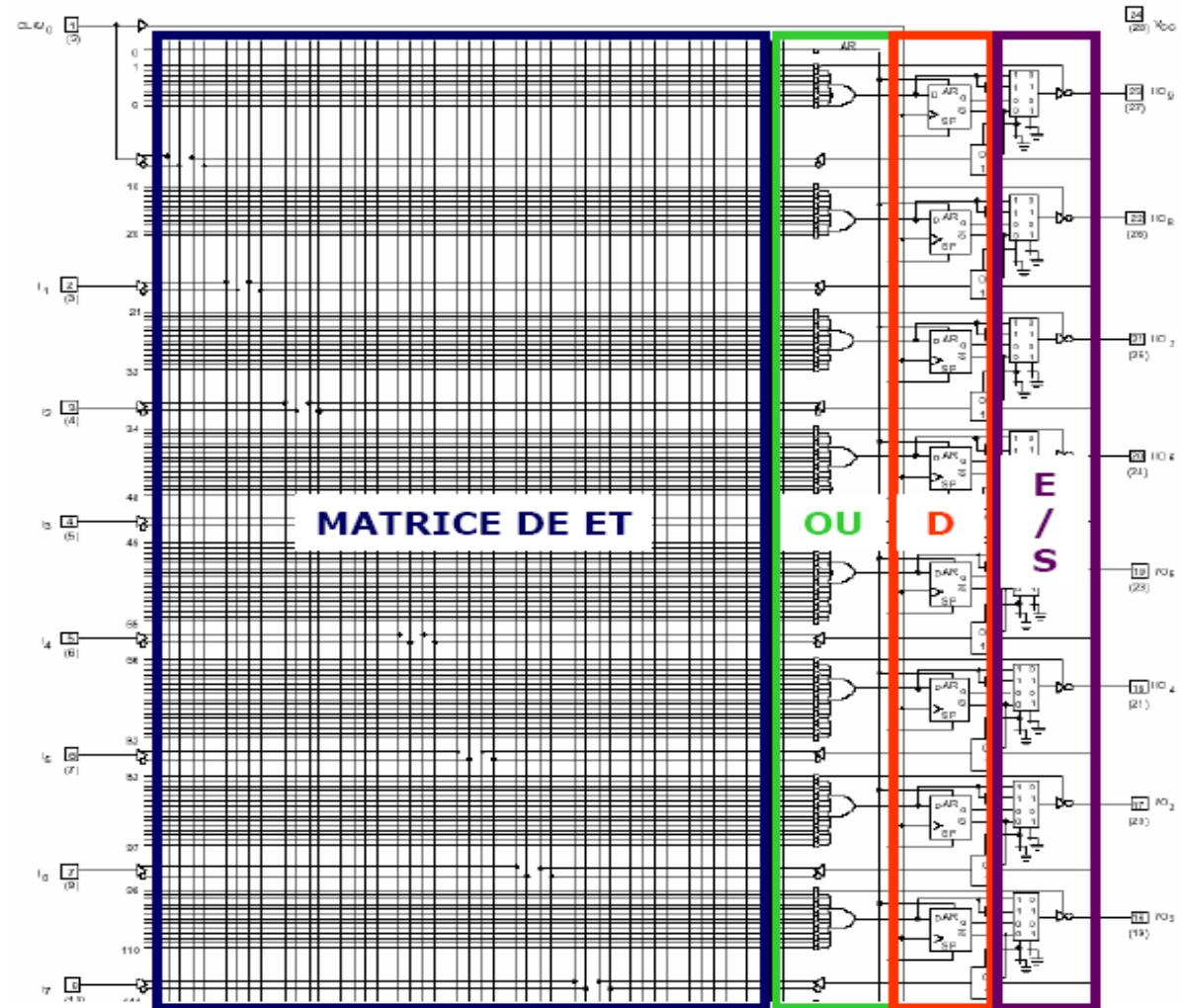
# SPLD: Versatile



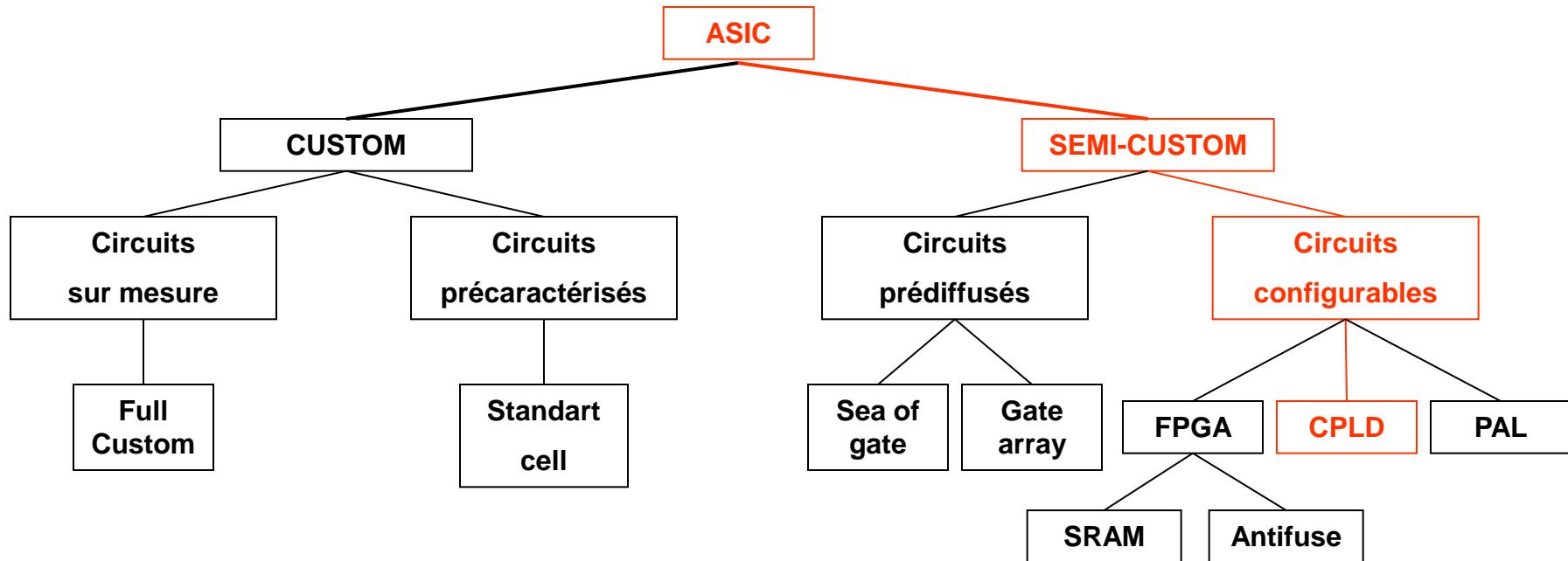
# Les circuits configurables: SPLD(simple programmable logique Device)

Matrice de ET réalisant tous les produits possibles (maxtermes) connectée aux sorties par des OU

Grande surface de Si utilisée. Ces circuits ne sont plus utilisés aujourd'hui



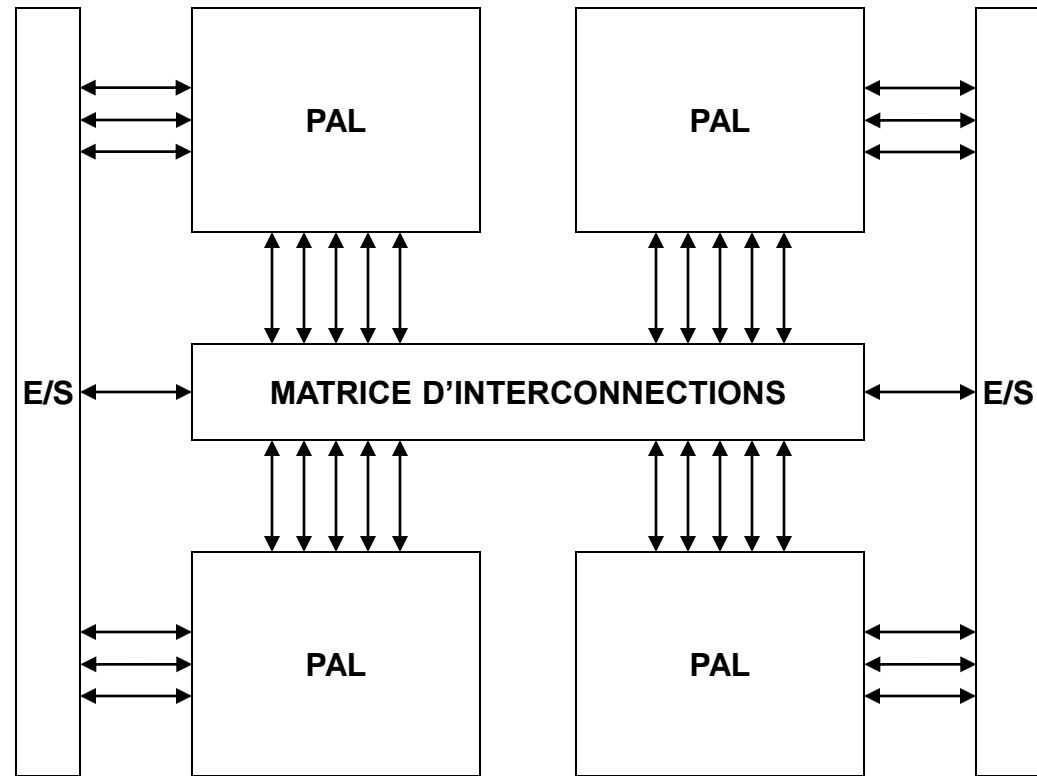
# CPLD



**CPLD** : Complex Programmable Logic Device

# CPLD

- Les CPLDs regroupent plusieurs PALs interconnectés par un réseau de connexions programmables.
- Les CPLDs sont les prémisses des premiers FPGAs.
- Ces circuits ne sont plus utilisés aujourd'hui car remplacés par les FPGAs.



# CPLD

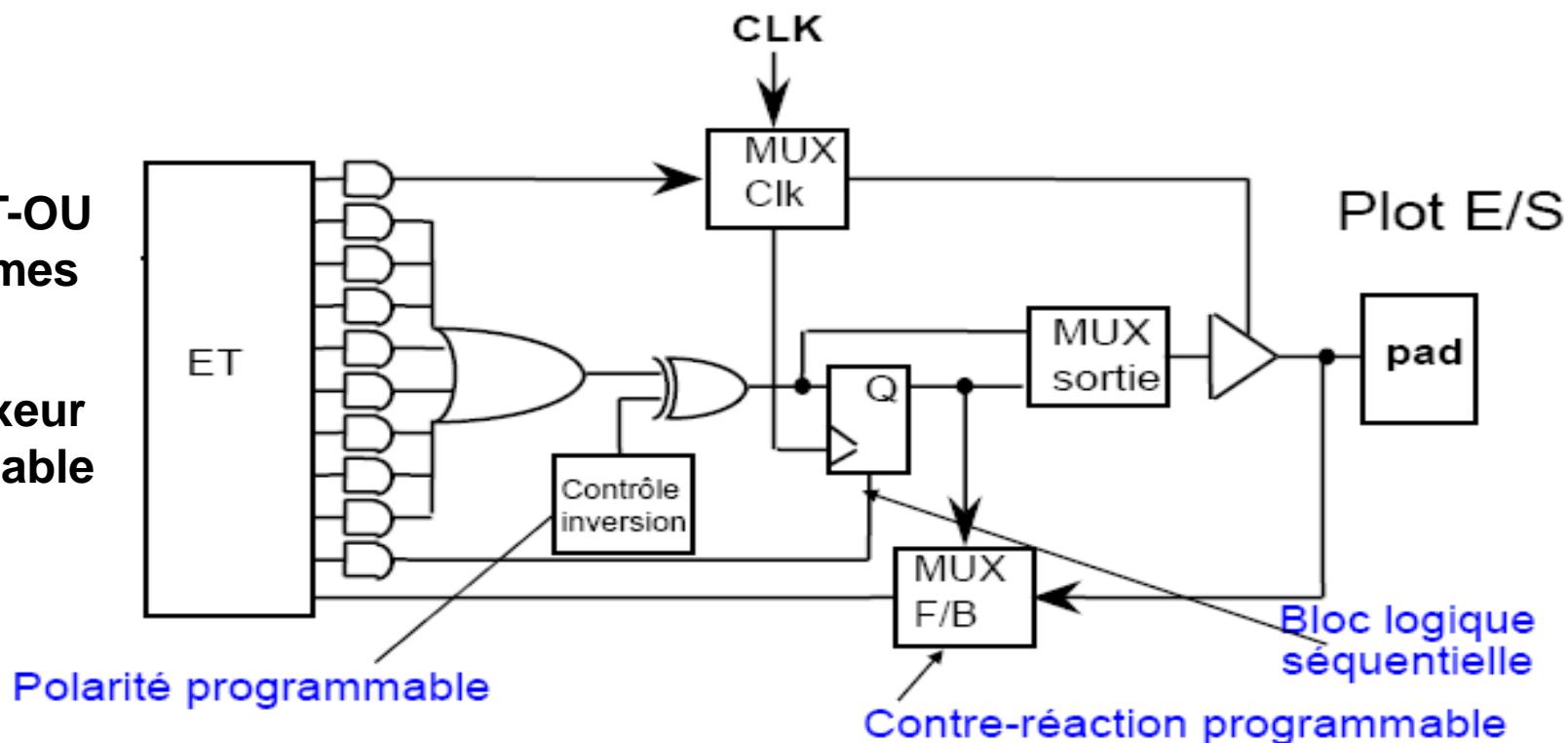
- la technologie de programmation est généralement EEPROM ou Flash EPROM)
- Un seul point de connexion relie entre eux les blocs logiques. Les temps de propagations des signaux sont constants et prédictibles (avant routage) → ce n'est pas le cas des FPGA (voir plus tard)

# CPLD/ EPLD(Erasable)

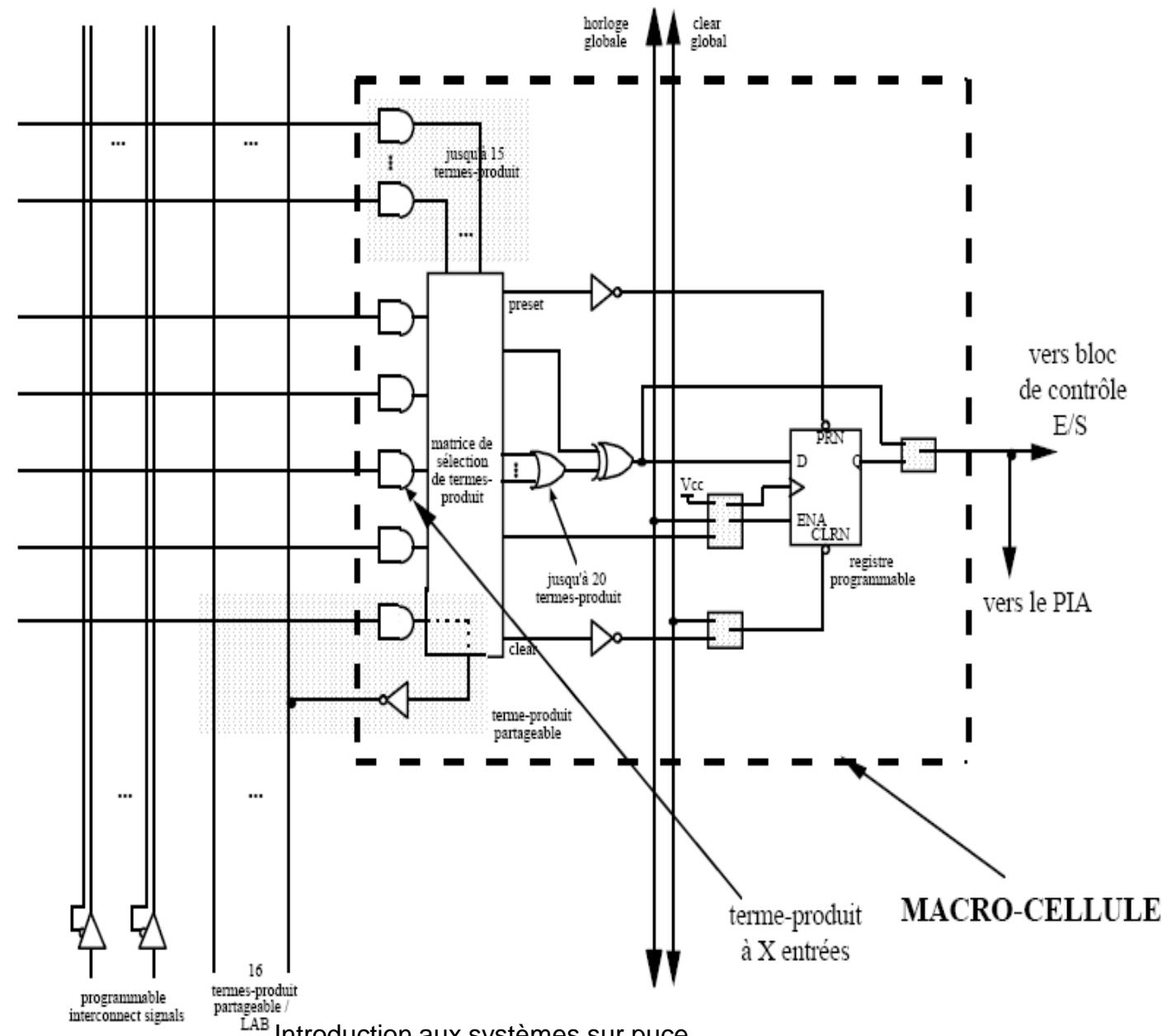
Altera MAX 7000 Macrocell structure

Bloc de base Altera = MACROCELL

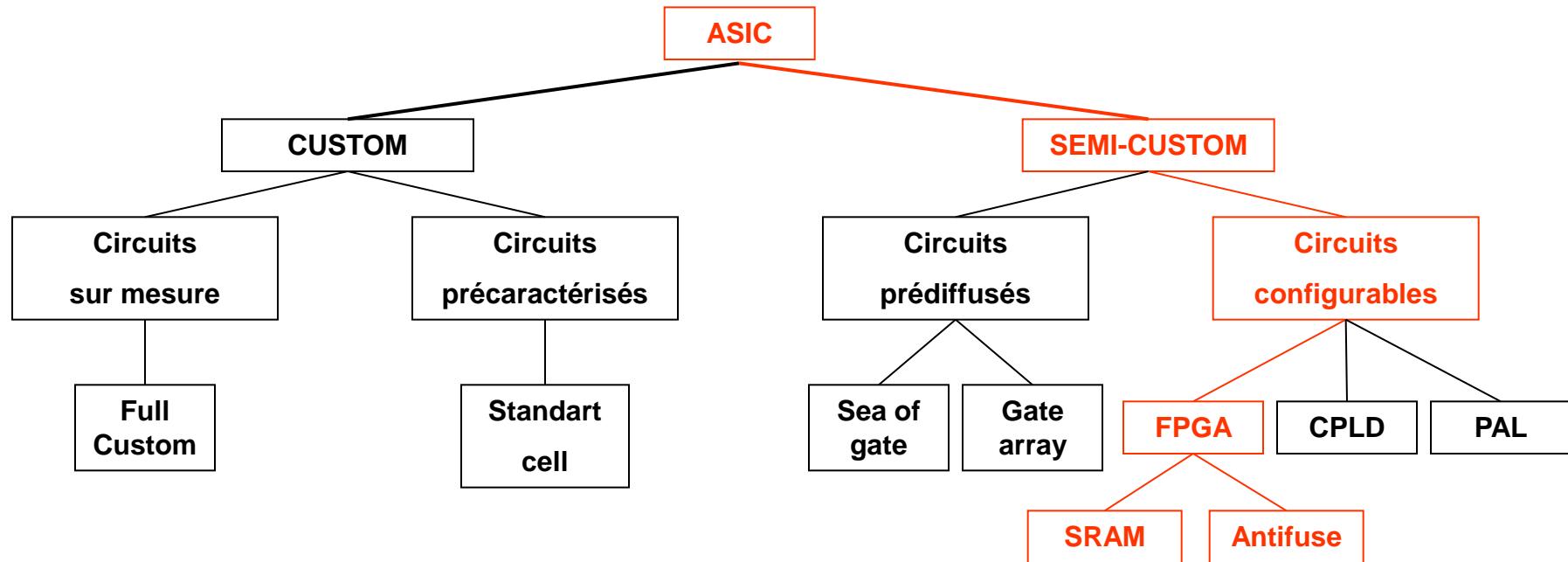
Réseau ET-OU  
avec 8 termes  
produits  
+ Multiplexeur  
programmable



# Altera MAX 7000 Macrocell structure



# FPGA



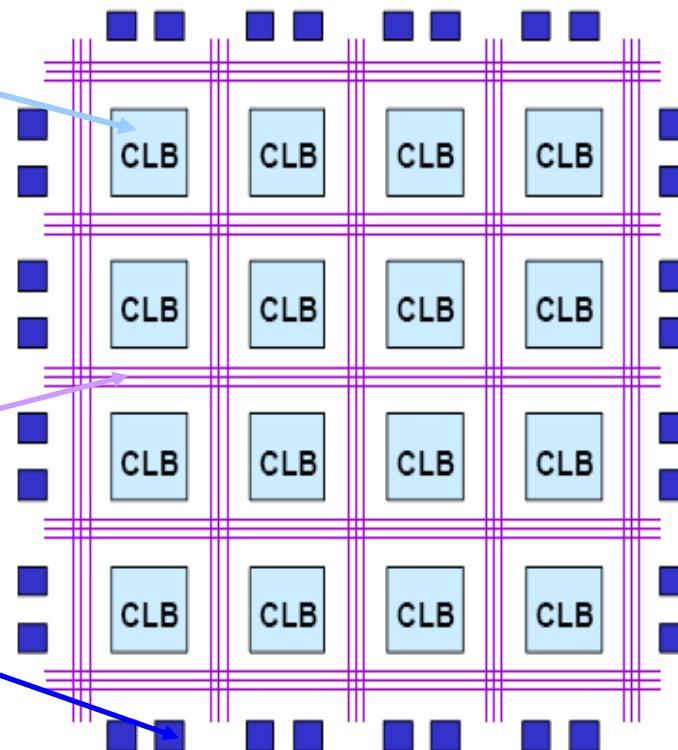
## FPGA : Field Programmable Gate Array

Les blocs logique sont plus nombreux et plus simples que les CPLD mais cette fois les interconnexions entre les blocs logiques ne sont plus centralisées

# FPGA : Architecture

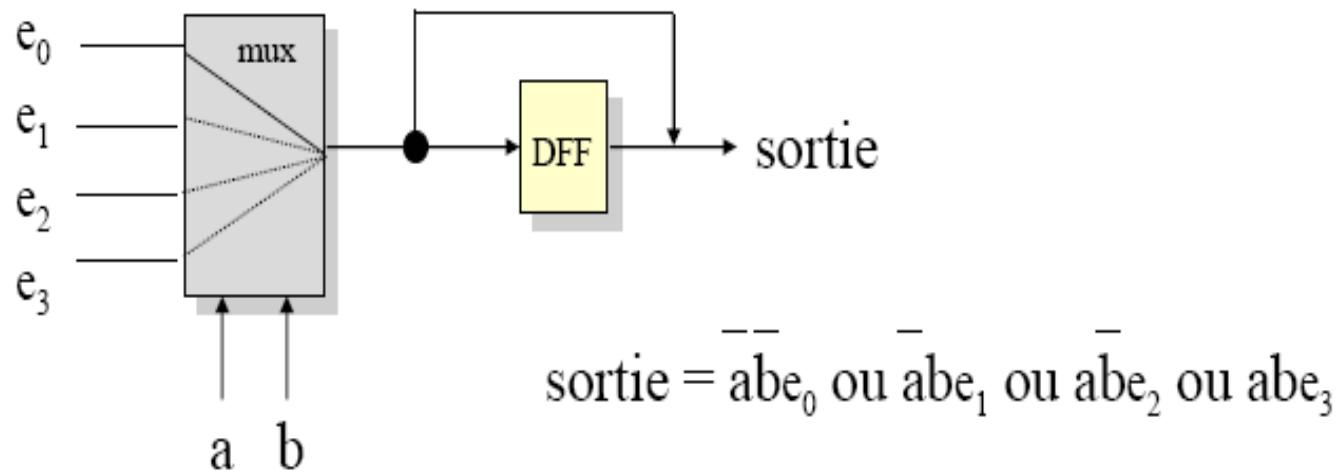
Un FPGA est à la base :

- **Un réseau de blocs de logique programmable** (*Configurable Logic Block -CLB*), chaque bloc pouvant réaliser des fonctions complexes de plusieurs variables, et comportant des éléments à mémoire
- **Un réseau d'interconnexions programmables** entre les blocs
- Des blocs spéciaux d'entrée et de sortie avec le monde extérieur (*Input/Output Block –IOB*).



# CLB à base de multiplexeurs

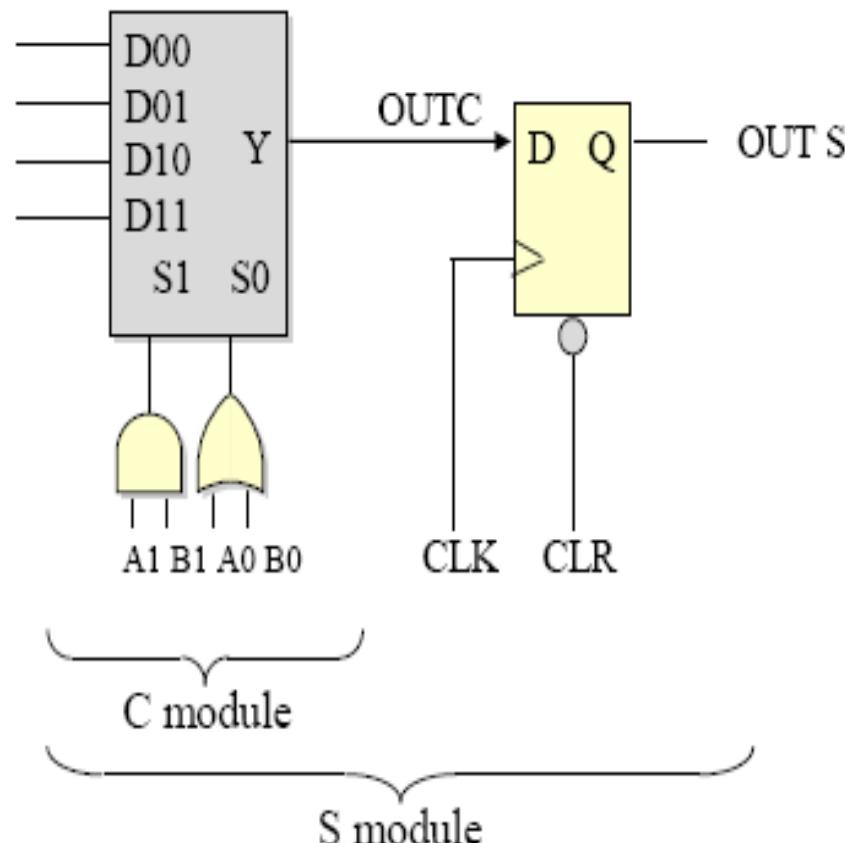
- Utilisée pour la technologie **ANTIFUSIBLE** équivaut à une LUT câblée



intérêt : surface réduite

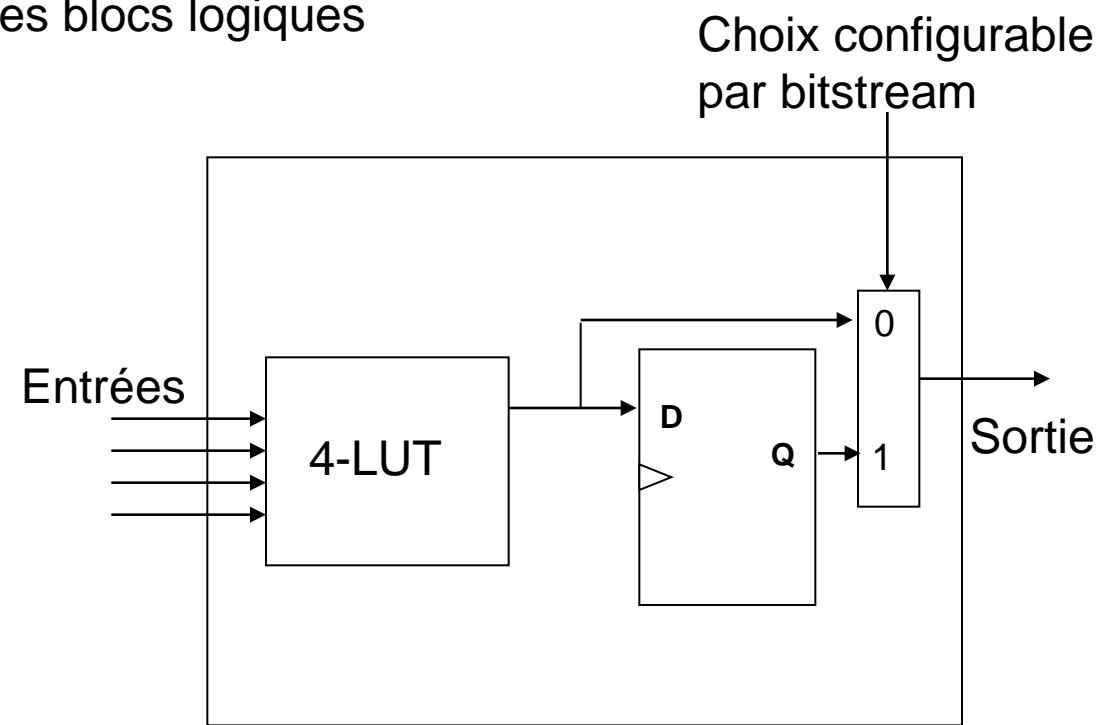
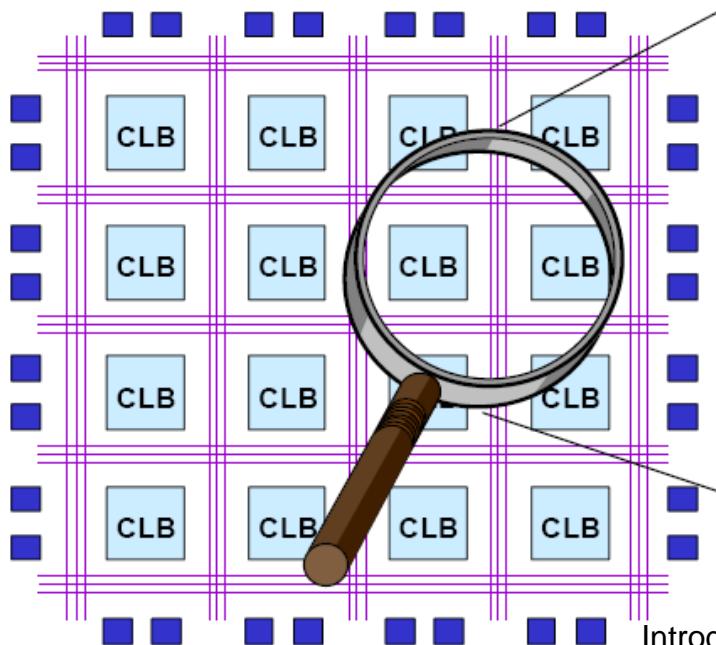
# CLB à base de multiplexeurs : Exemple

## ■ Cellule ACTEL ACT3



# FPGA: CLB à base de LUT

- Arrangement Matriciel de blocs logiques avec configuration des :
  1. La fonction de chaque bloc
  2. Interconnexions entre les blocs logiques



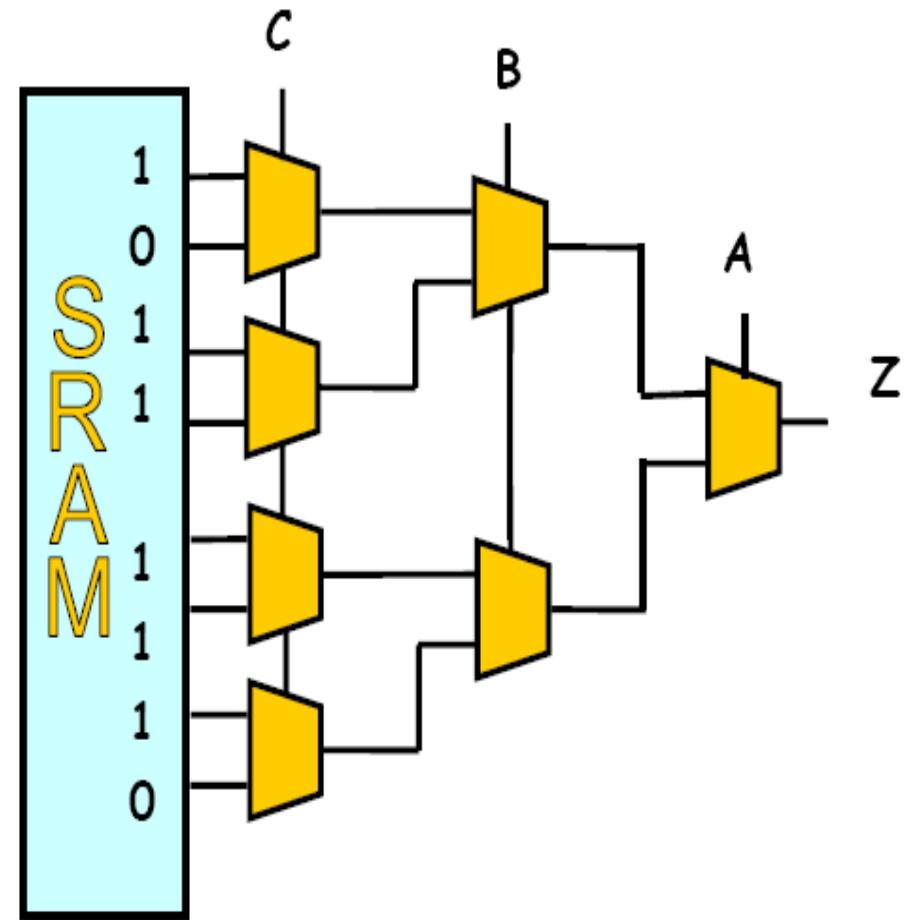
# FPGAs : CLB à base de Look Up Table

Réalisation d'une LUT

A	B	C	Z
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

LUT 3

« Look Up Table »

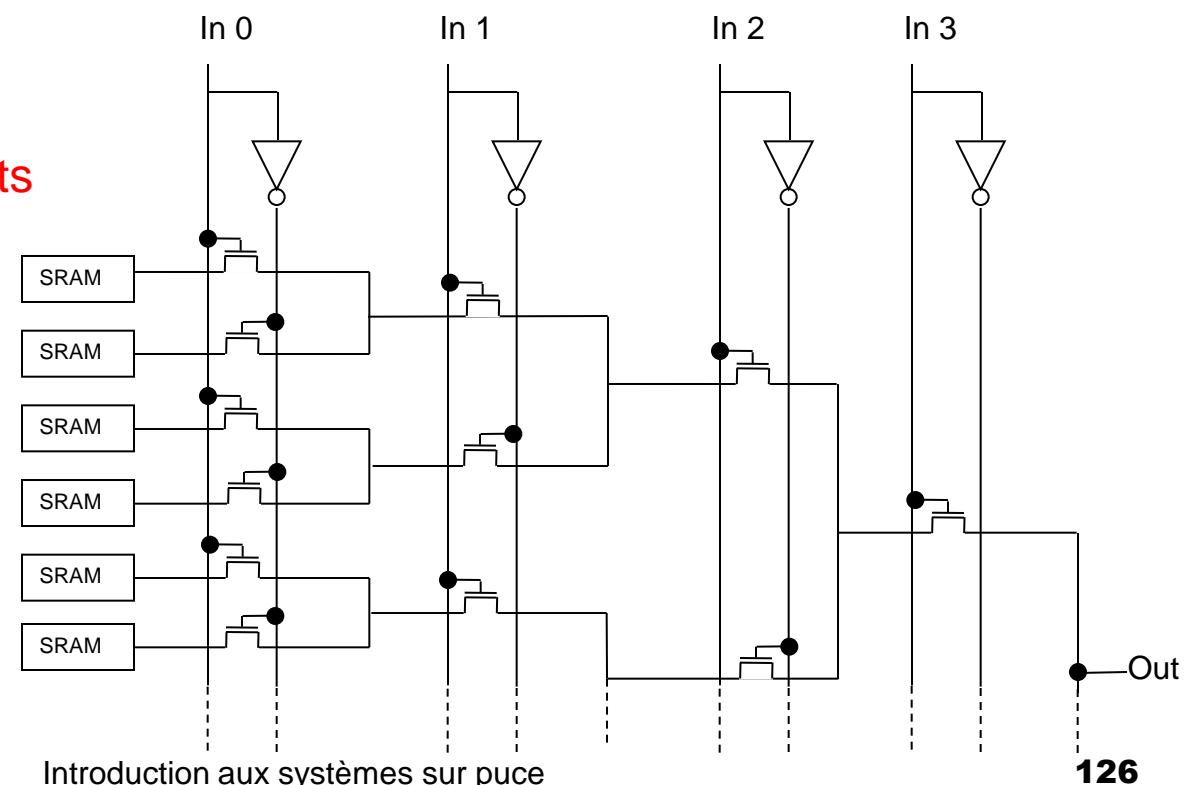
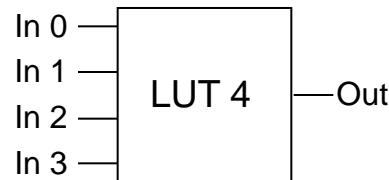


# FPGA :Les Look Up Tables

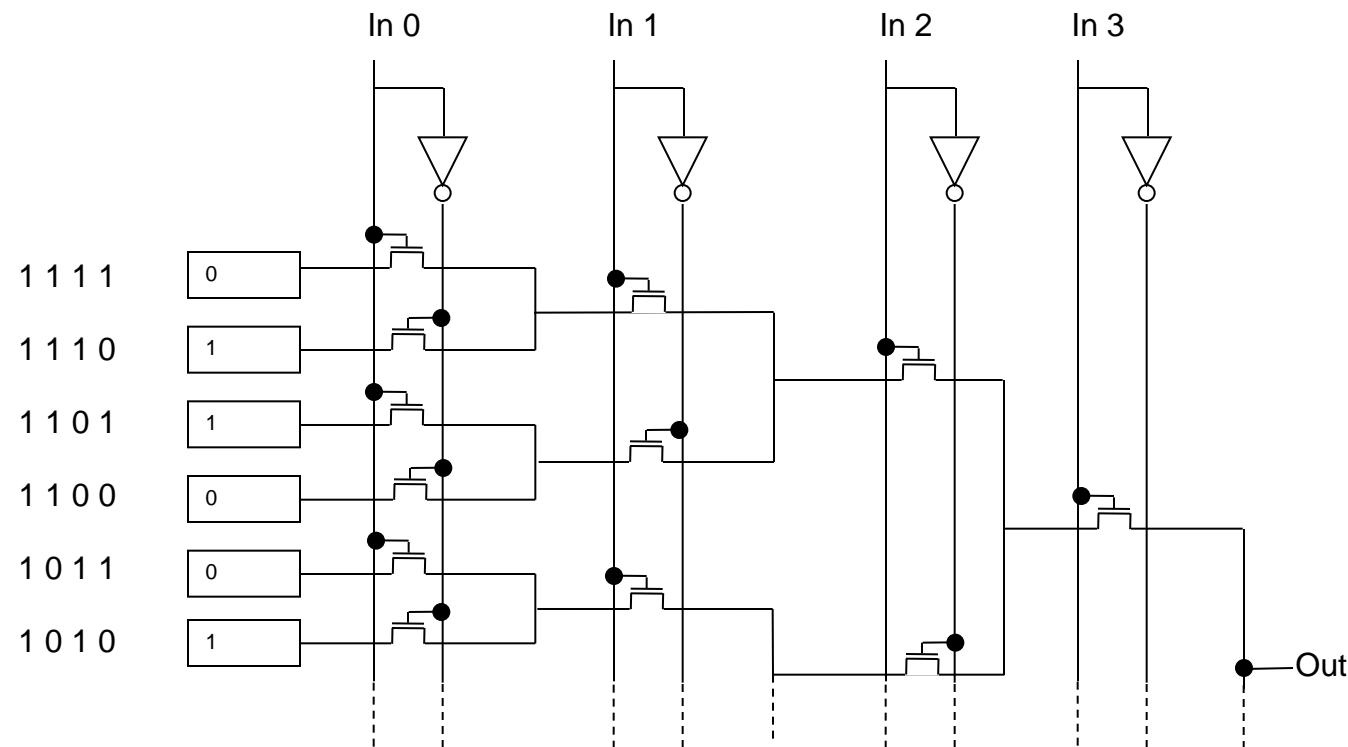
Ce sont de petits éléments de mémorisation, qui reflètent la table de vérité d'une fonction logique.

LUT = Table de scrutation

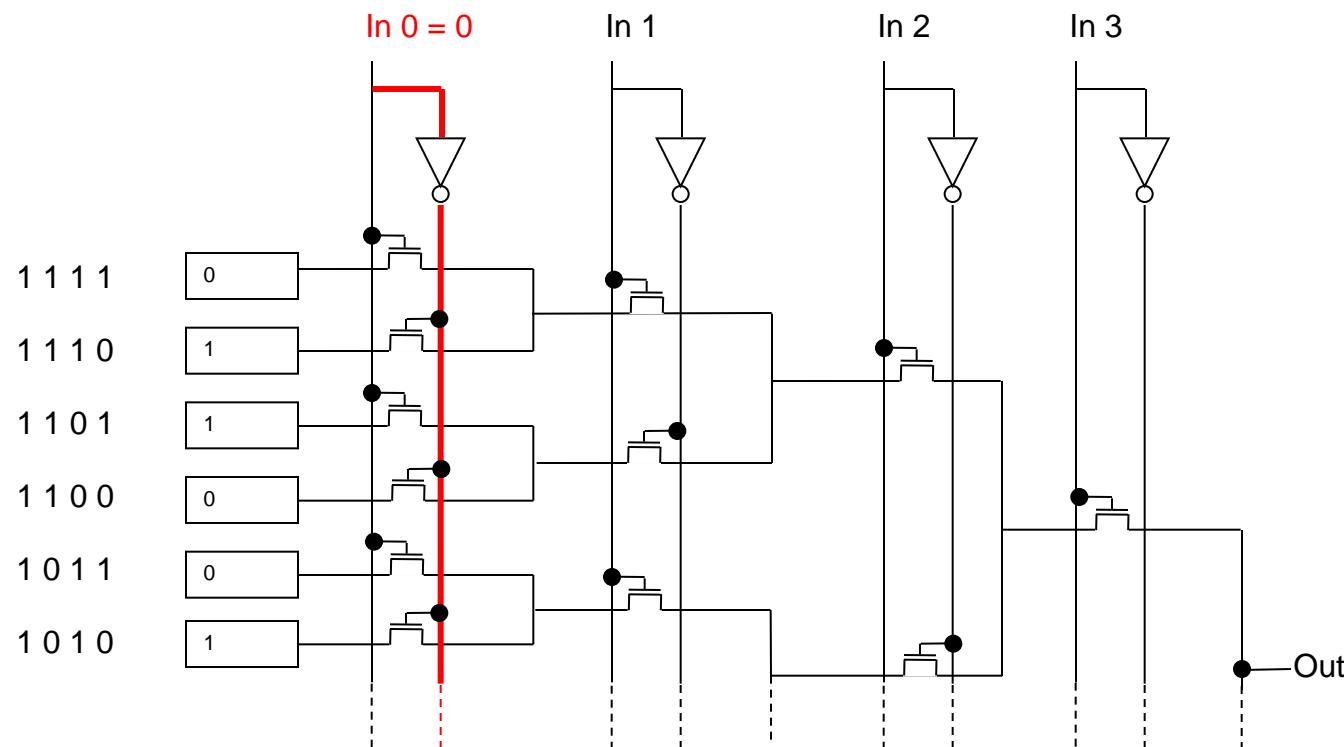
LUT 4 entrées = RAM 2octets



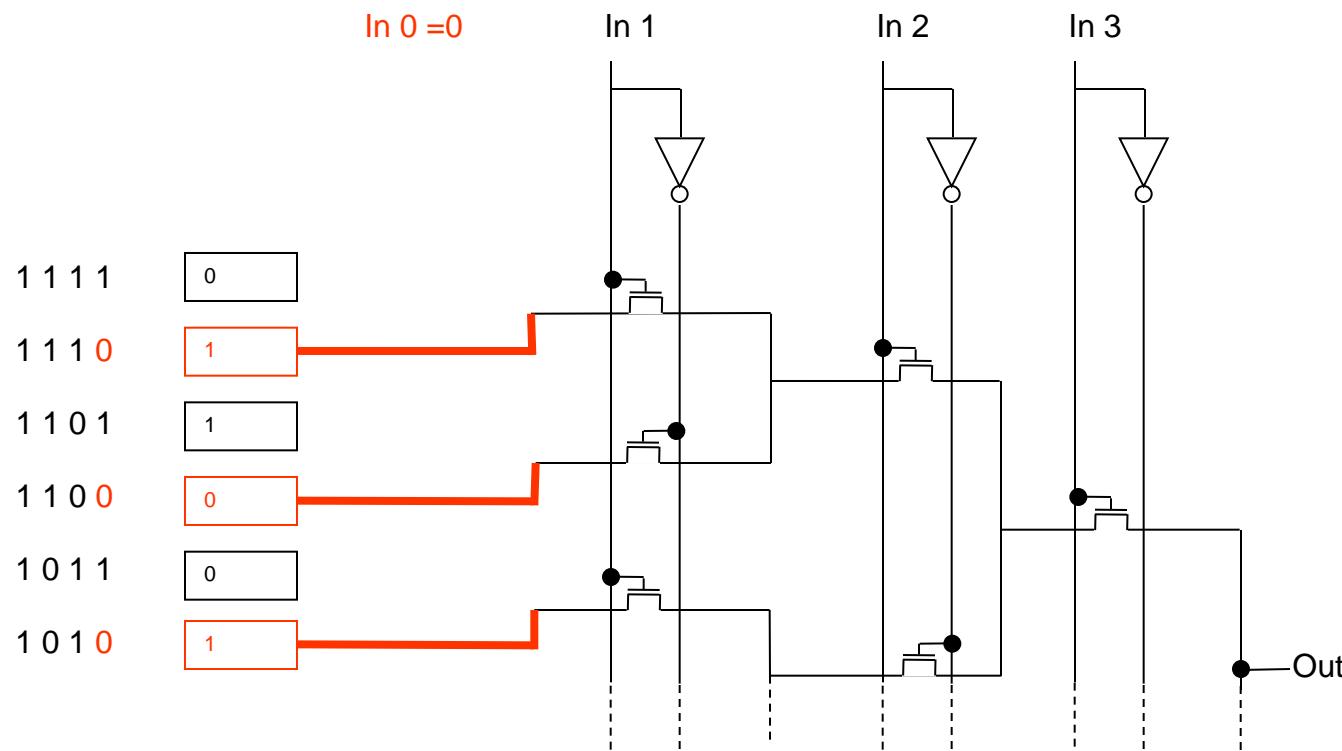
# FPGA : Les Look Up Tables



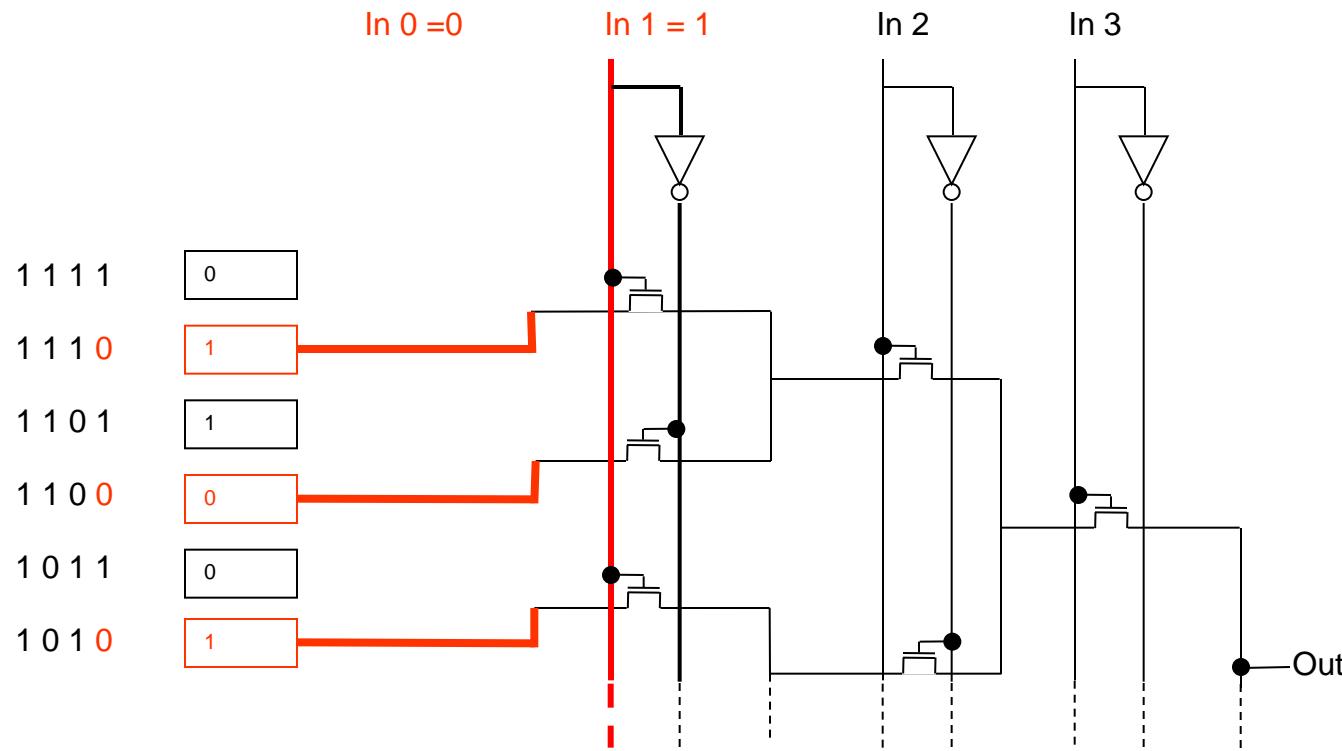
# FPGA :Les Look Up Tables



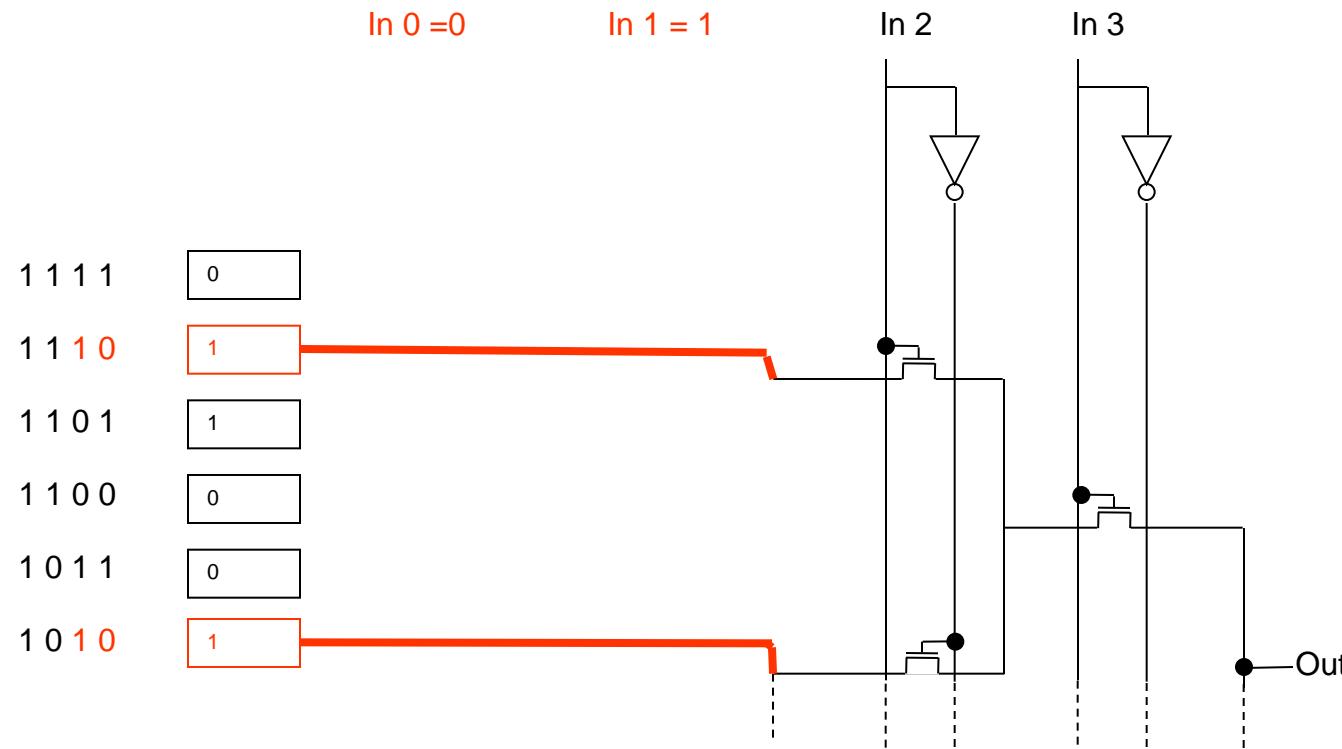
# FPGA : Les Look Up Tables



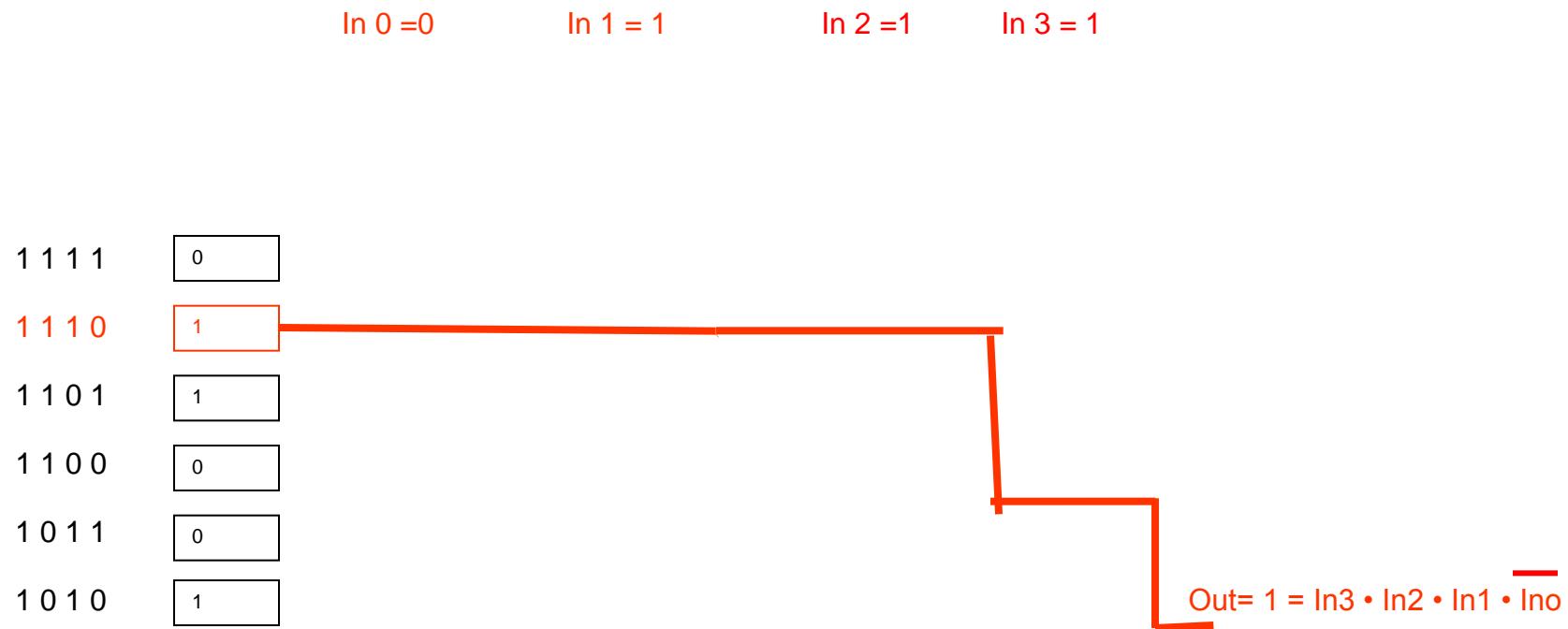
# FPGA : Les Look Up Tables



# FPGA : Les Look Up Tables

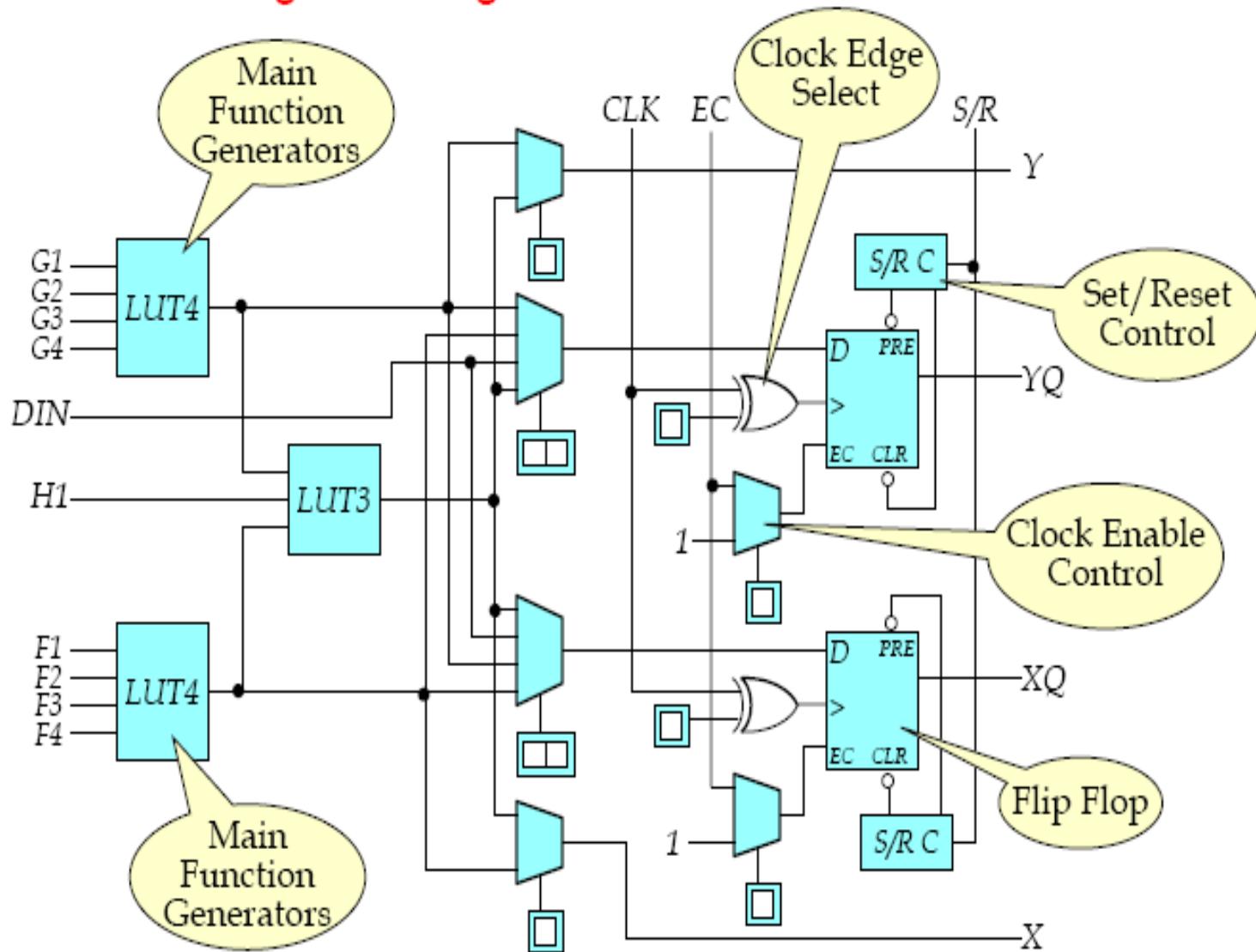


# FPGA: Les Look Up Tables

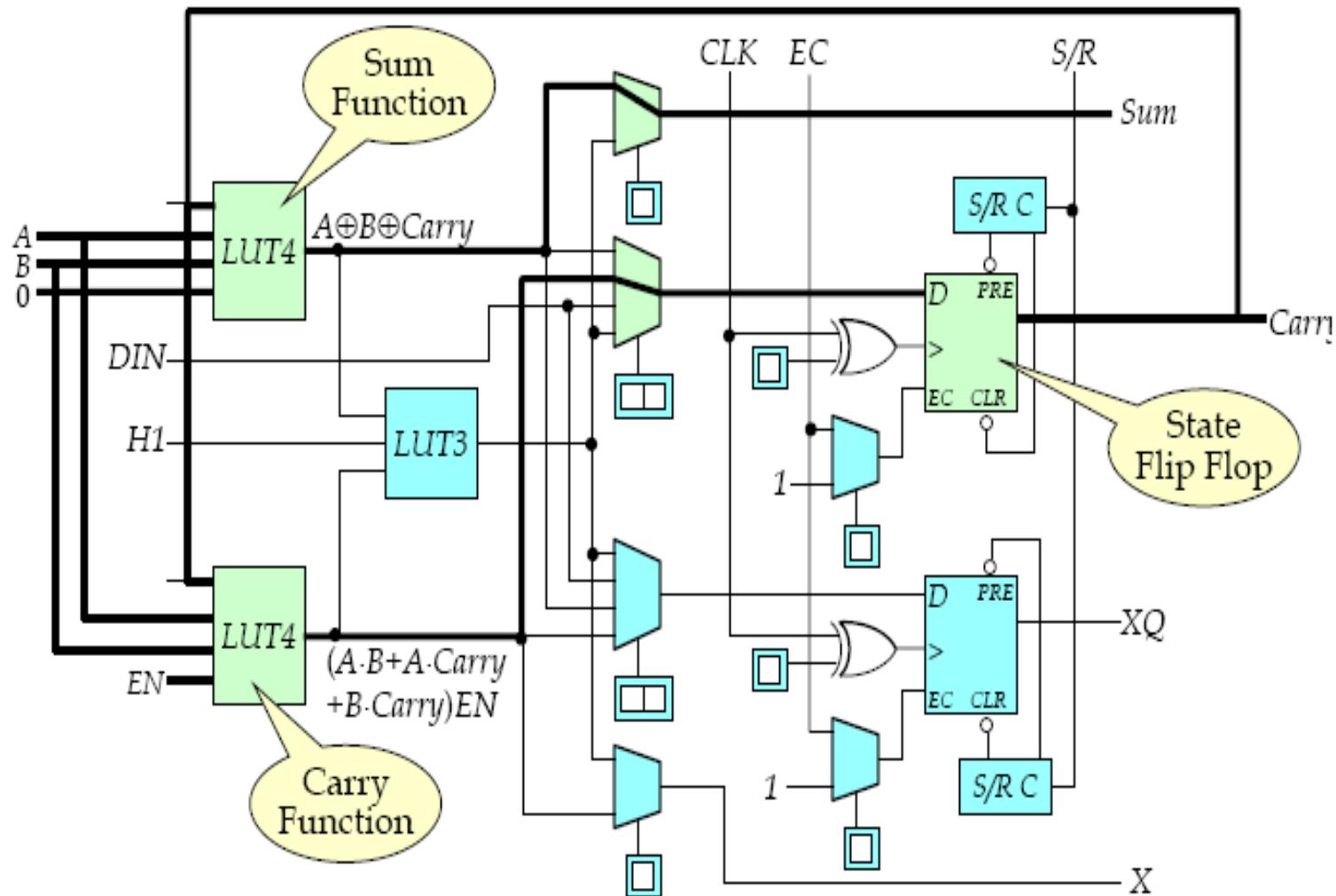


# FPGA : CLB

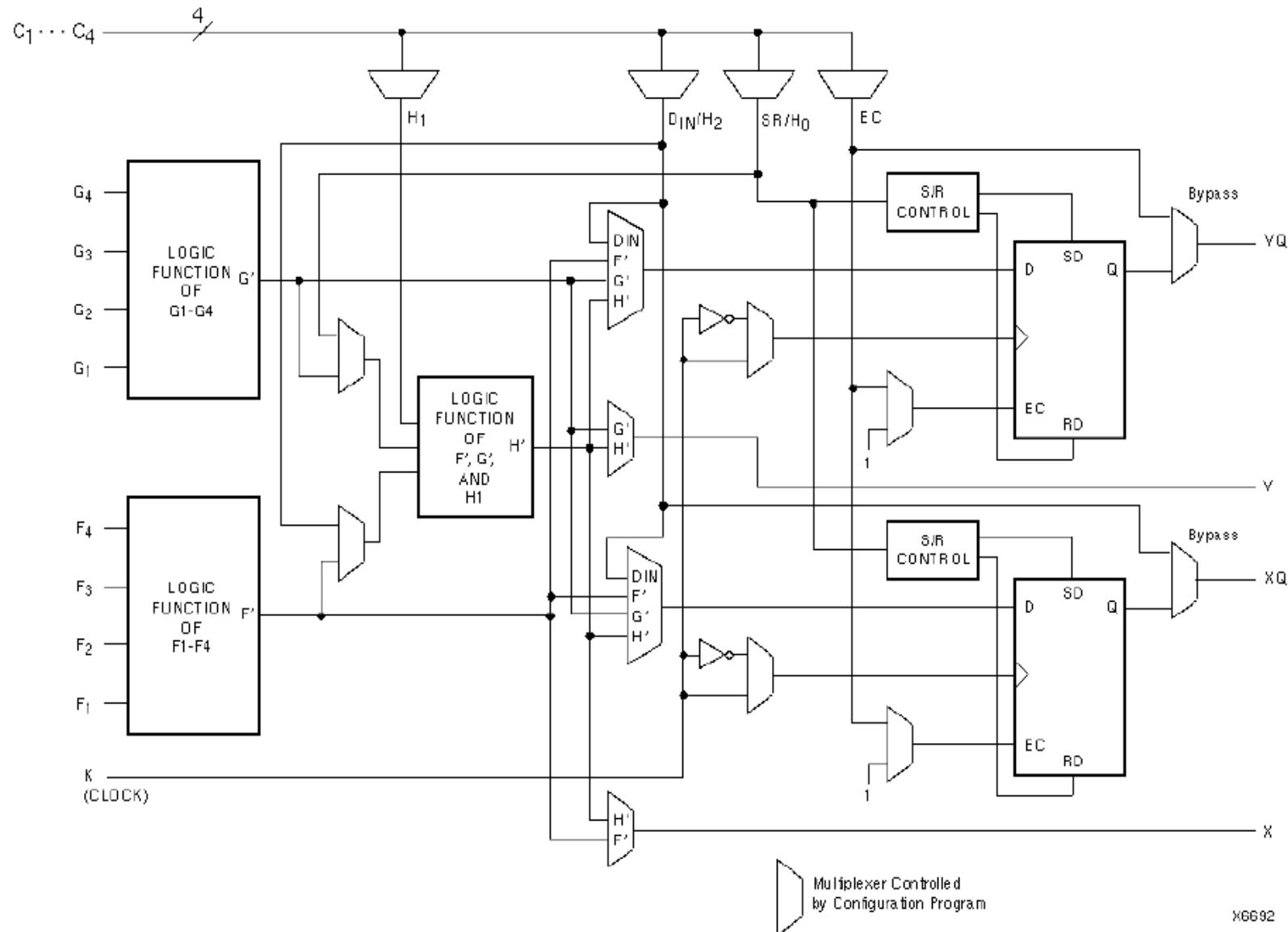
FPGAs : Configurable logic block



# Exemple: implémentation d'un additionneur série



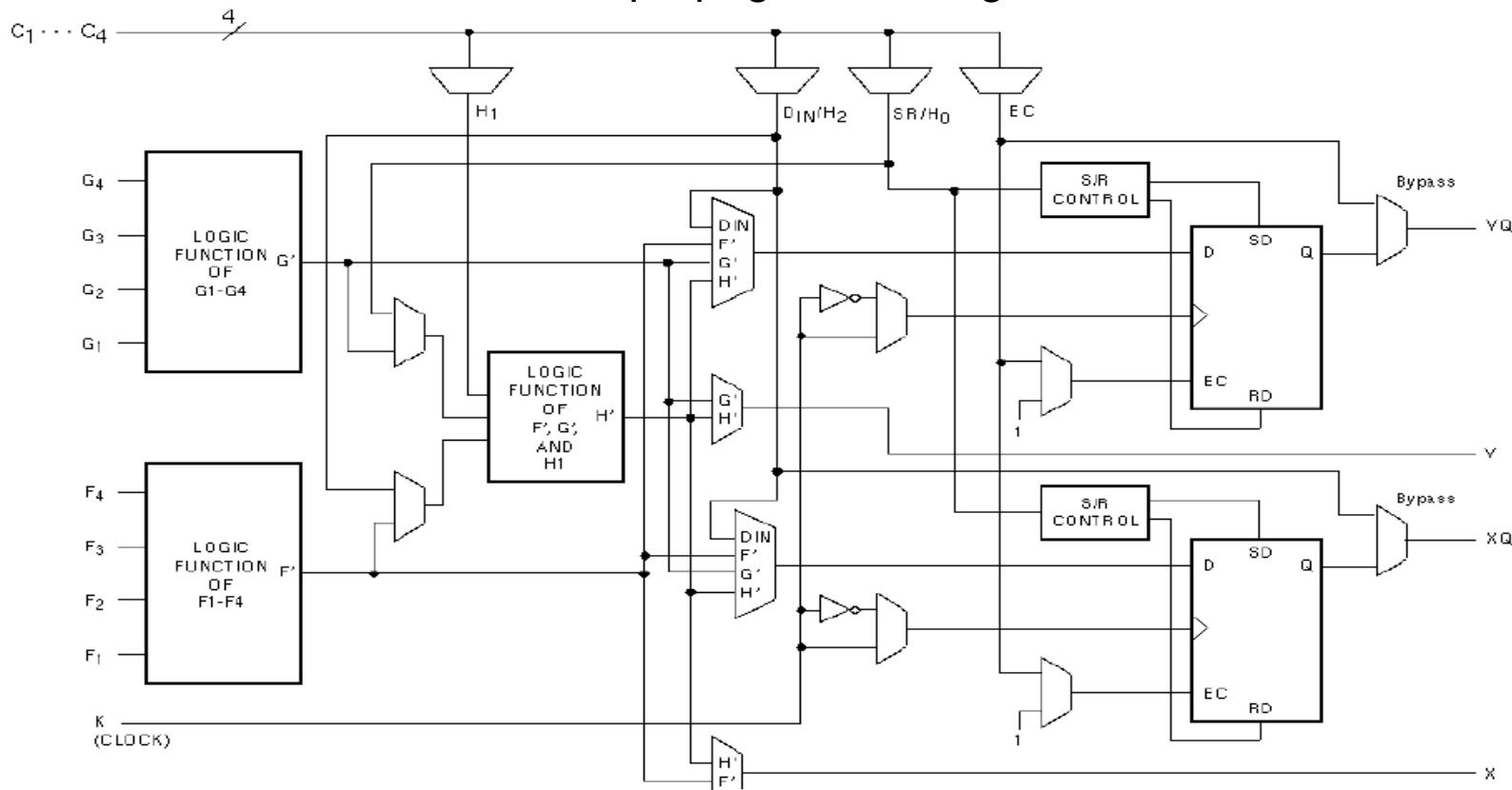
# FPGA : Xilinx CLB XC4000



Multiplexer Controlled  
by Configuration Program

# Exercice:

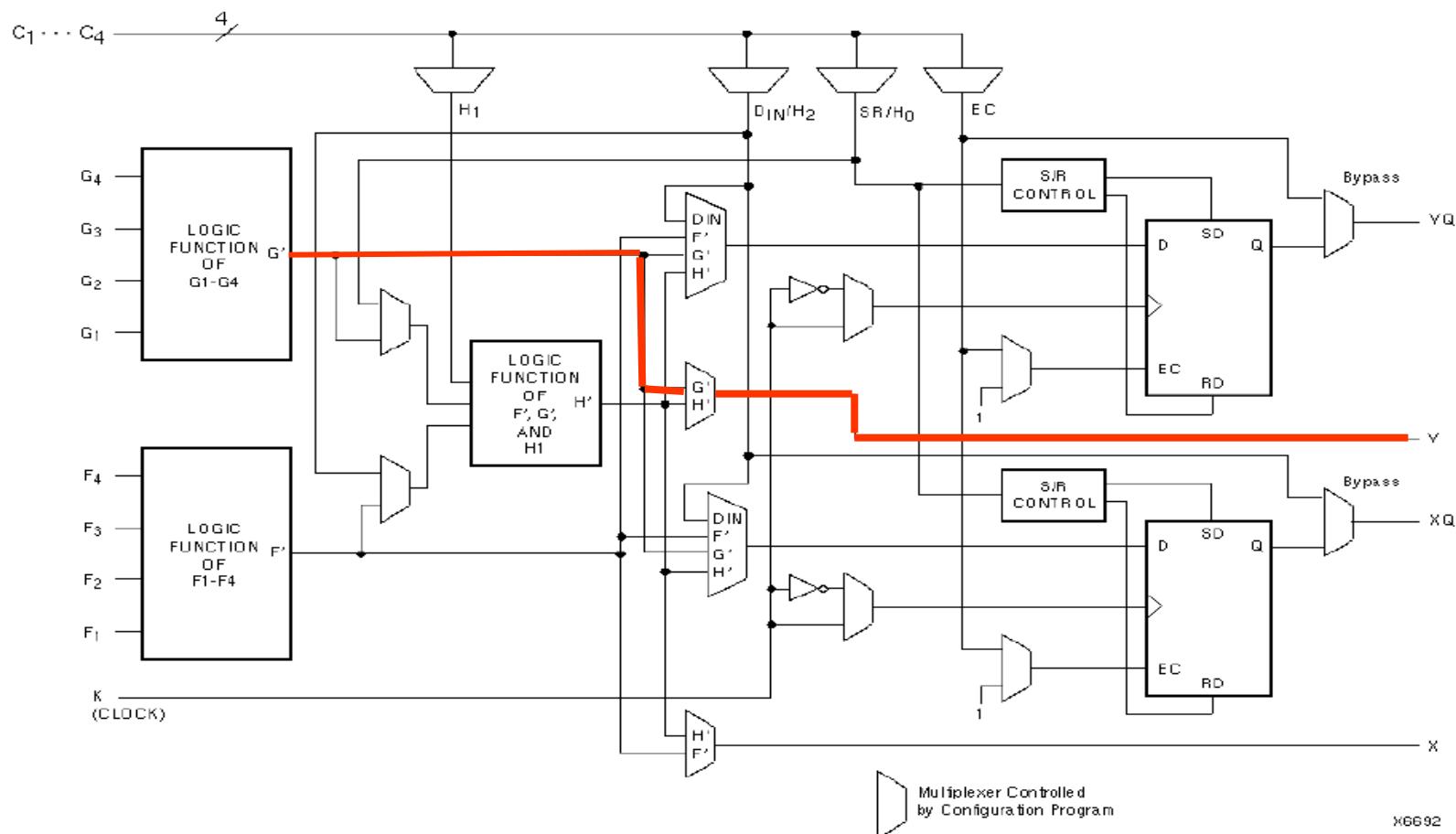
1. Comment peut on-réaliser une fonction **combinatoire** à 4 variables avec le CLB de xilinx? Donner le chemin de propagation du signal de sortie
2. Comment peut on-réaliser une fonction **séquentielle** à 5 variables avec le CLB de xilinx? Donner le chemin de propagation du signal de sortie



Multiplexer Controlled  
by Configuration Program

# Exercice:

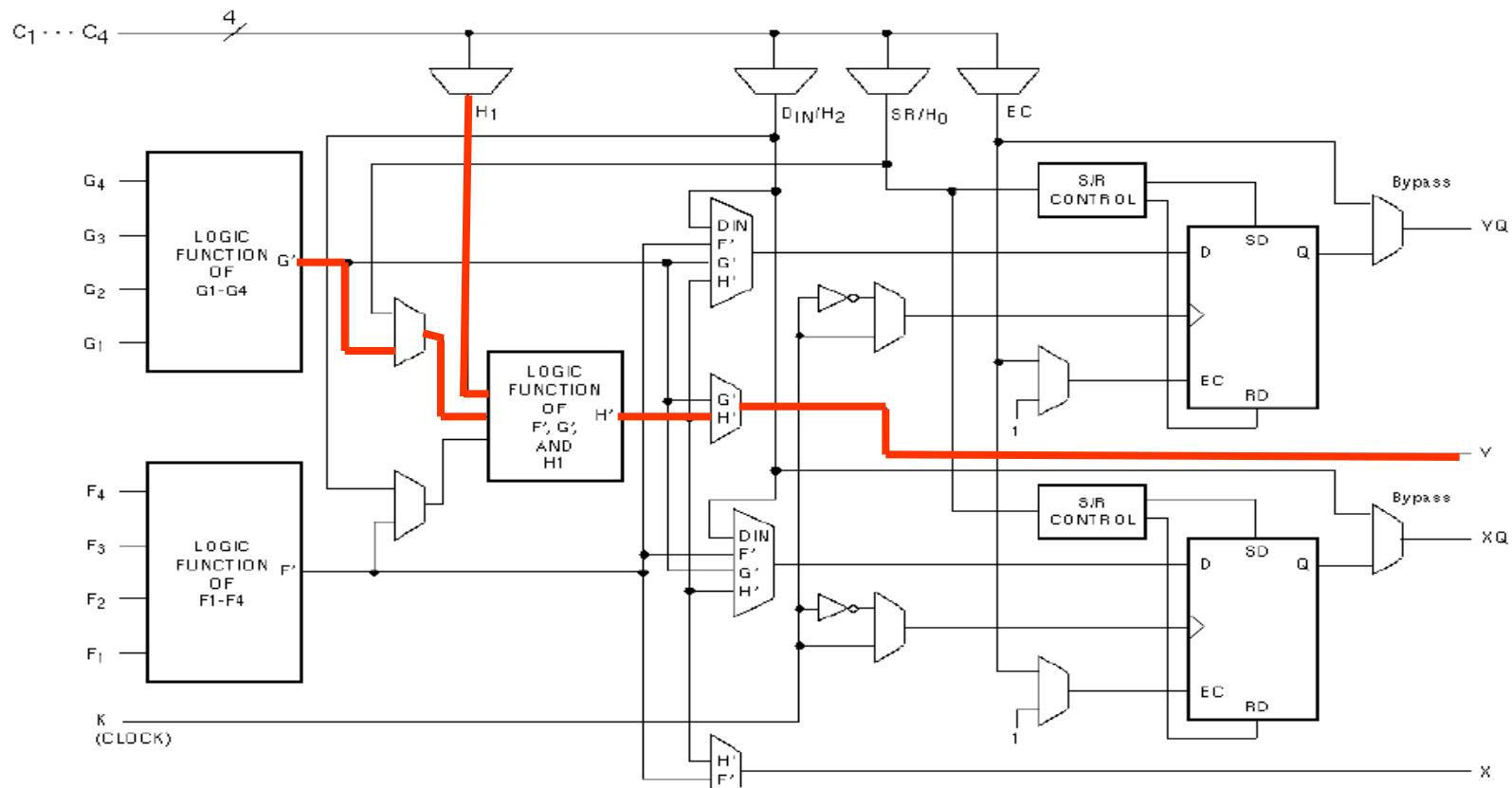
1. Comment peut-on réaliser une fonction combinatoire à 4 variables avec le CLB de xilinx? Donner le chemin de propagation du signal de sortie



# Exercice:

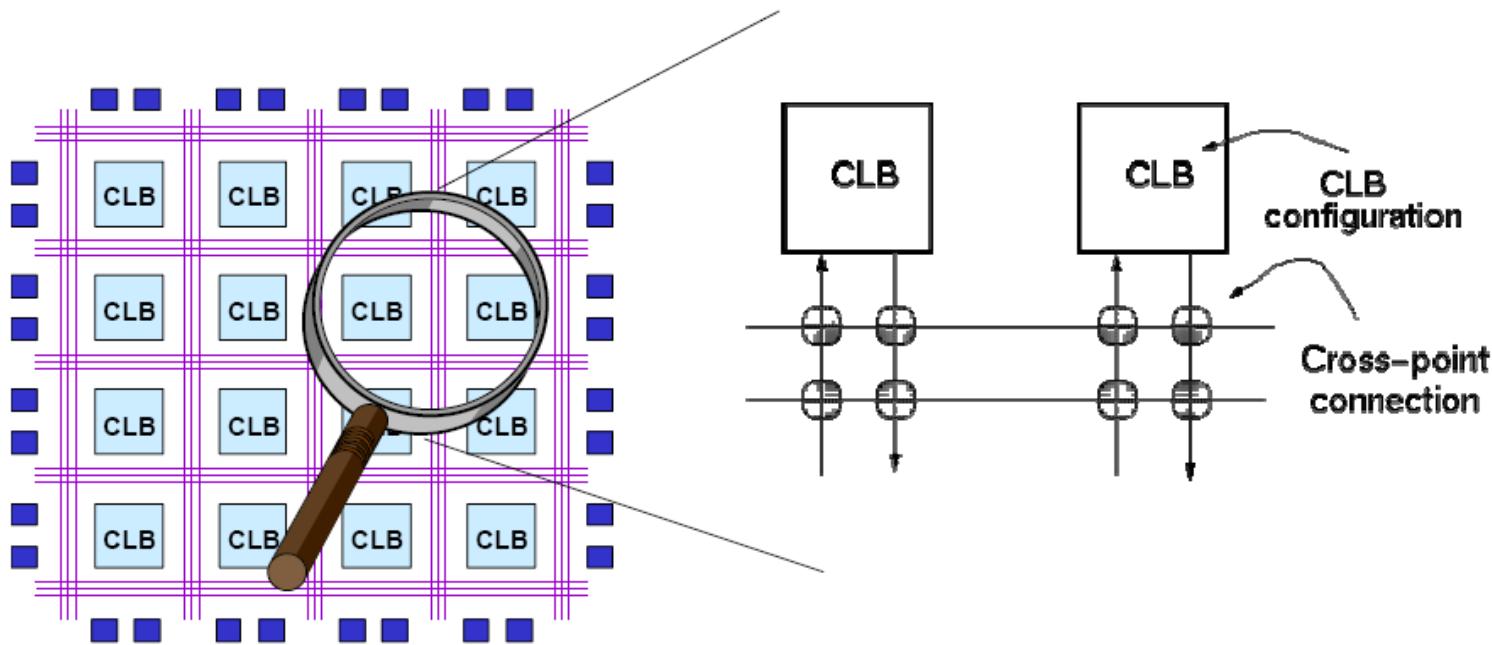


2. Comment peut on réaliser une fonction **séquentielle à 5 variables** avec le CLB de xilinx? Donner le chemin de propagation du signal de sortie



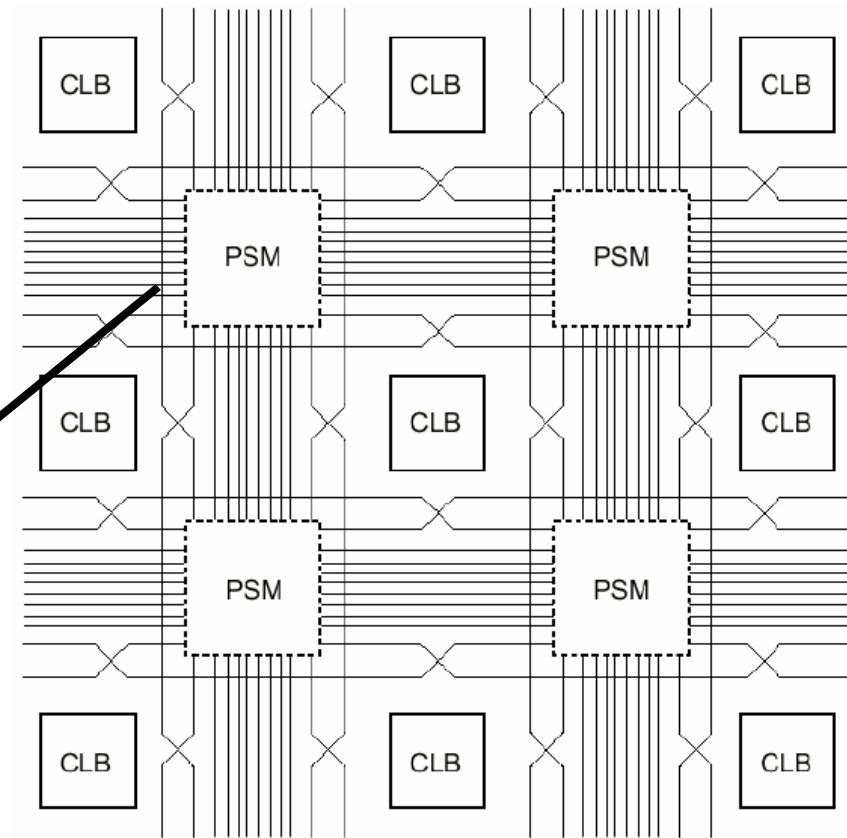
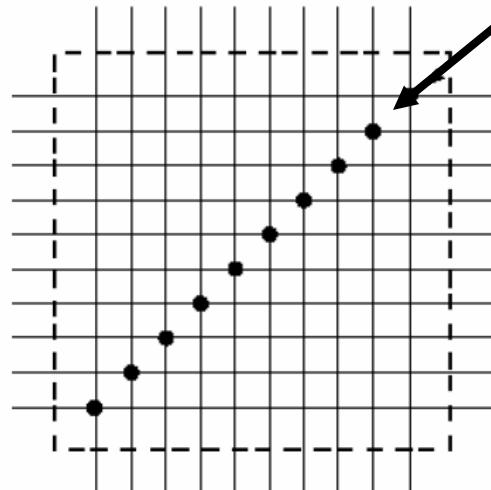
# FPGA: configuration

- Arrangement Matriciel de blocs logiques avec configuration des :
  1. La fonction de chaque bloc
  2. Interconnexions entre les blocs logiques

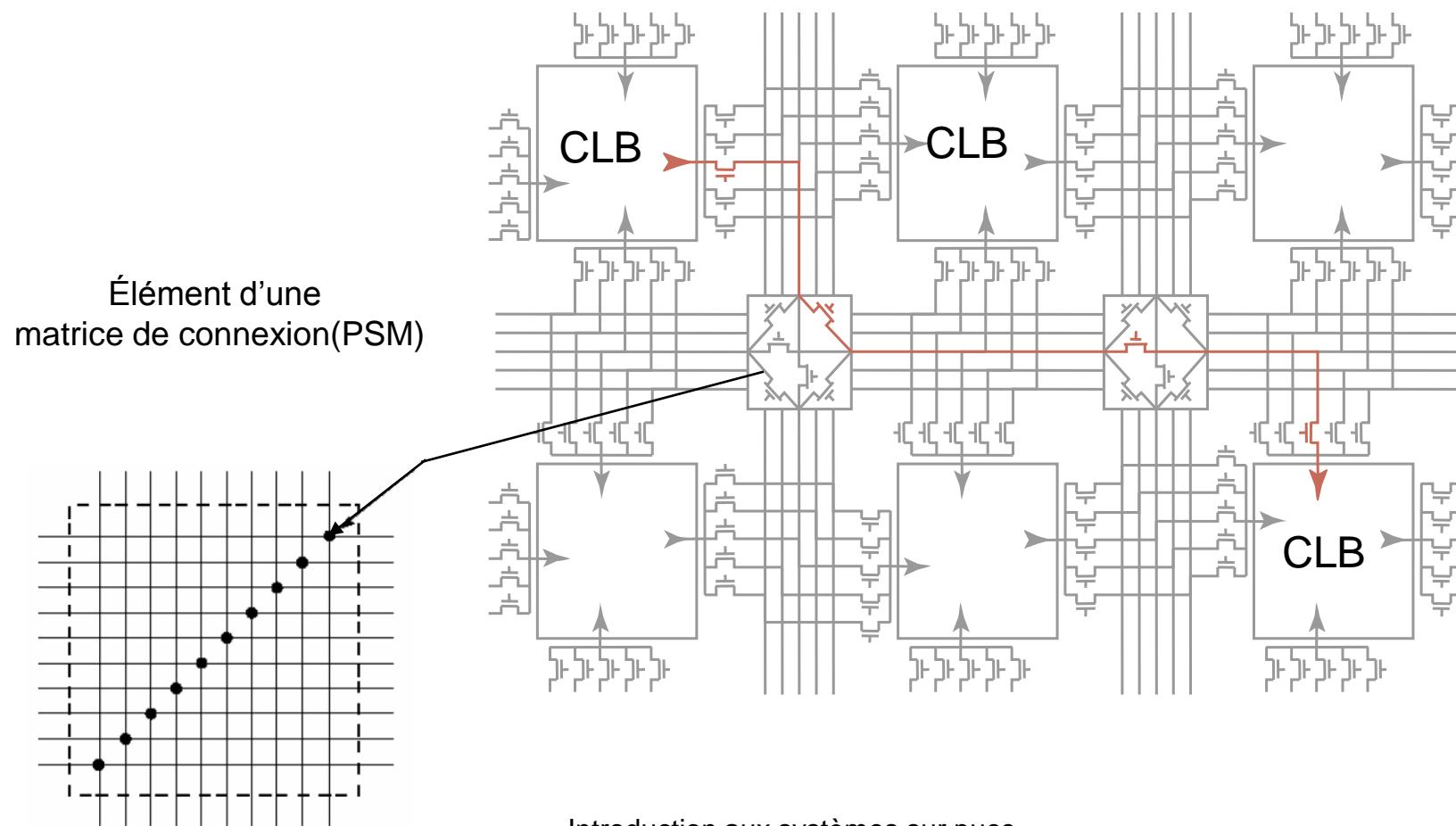


# FPGA: interconnexions programmables

Matrice de connexions  
Programmables  
(Programmable Switch Matrix)



# FPGA: interconnexions programmables



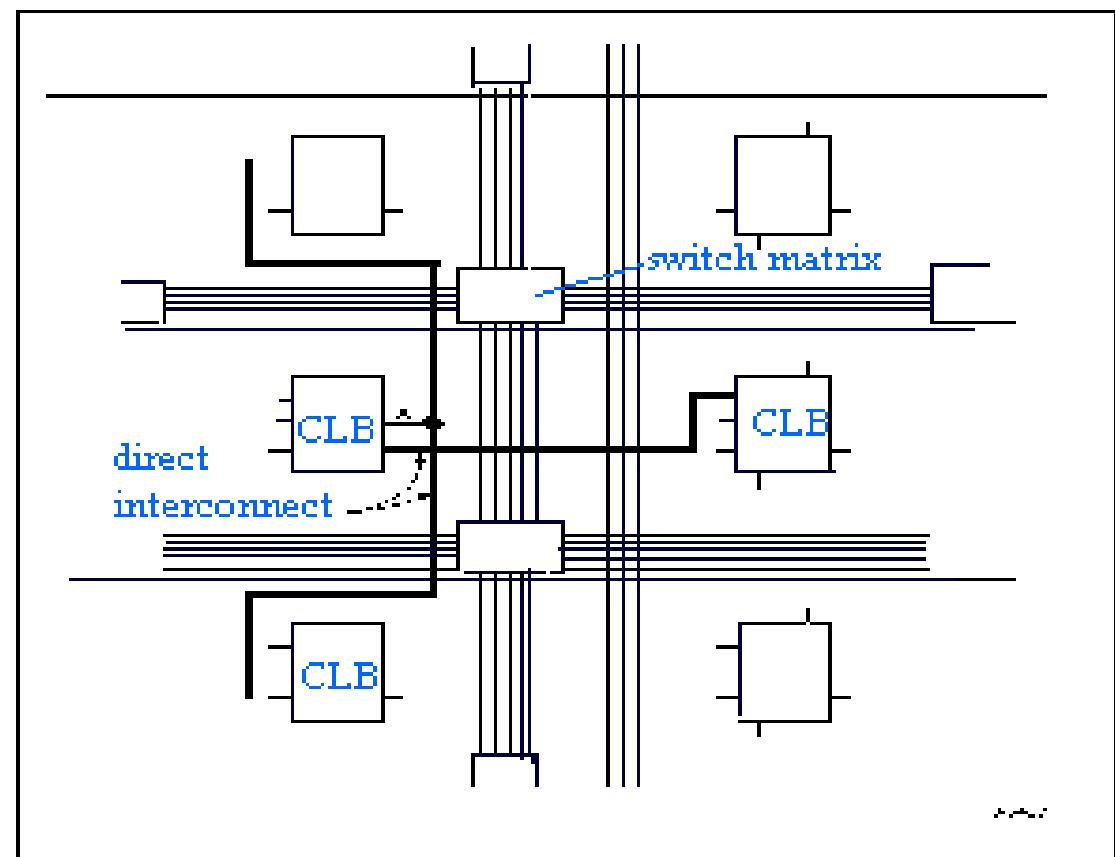
# FPGA: types de connexions

- **Directes:** relient deux CLBs adjacents
- **Simples:** relient deux PSMs (Programmable Switch Matrix) adjacentes
- **Doubles:** relient deux PSMs séparées par une PSM
- **Quadruples:** relient deux PSMs séparées par 3 PSMs
- **Longs:** traversent la puce de part en part, soit horizontalement ou verticalement (ils peuvent toutefois être séparés en deux moities de longueur égale et indépendantes)

# FPGA: types de connexions

## Connexions directes :

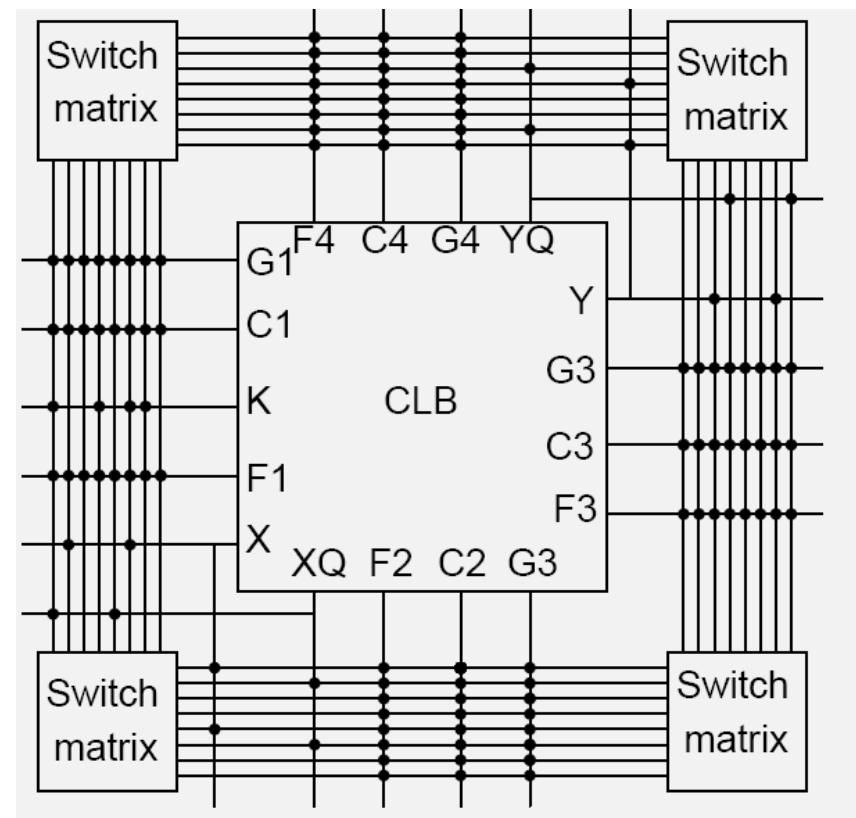
- relient deux CLB adjacents.
- Ne passent pas par une matrice de connexion (PSM)



# FPGA: types de connexions

## Connexions Simples:

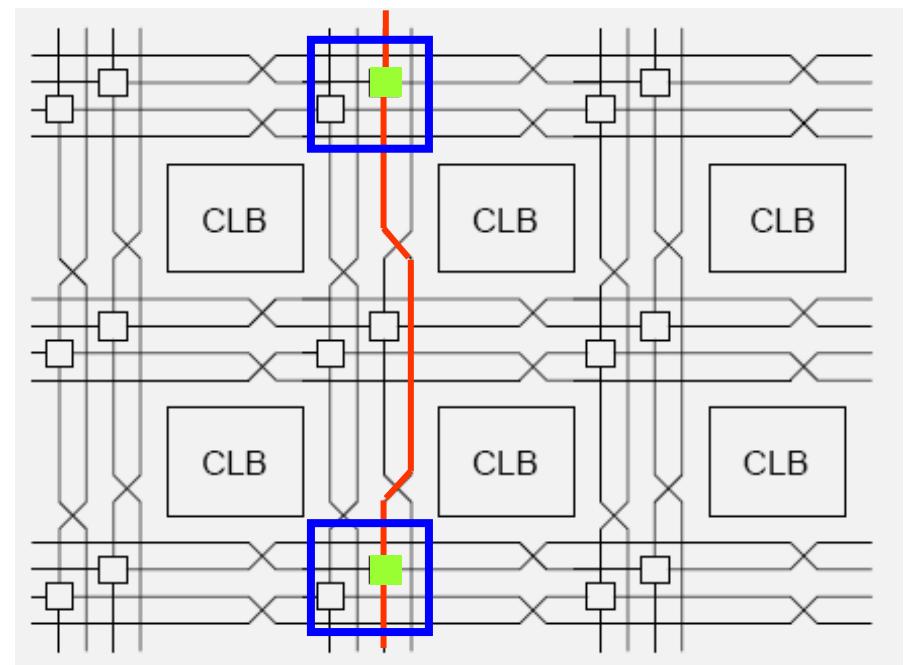
- Relient deux PSM adjacentes.
- Offrent une grande flexibilité de parcours.
- Le passage d'une connexion simple au travers d'une PSM entraîne un délai.
- Ce type de connexion se prête surtout aux liens de nature locale.



# FPGA: types de connexions

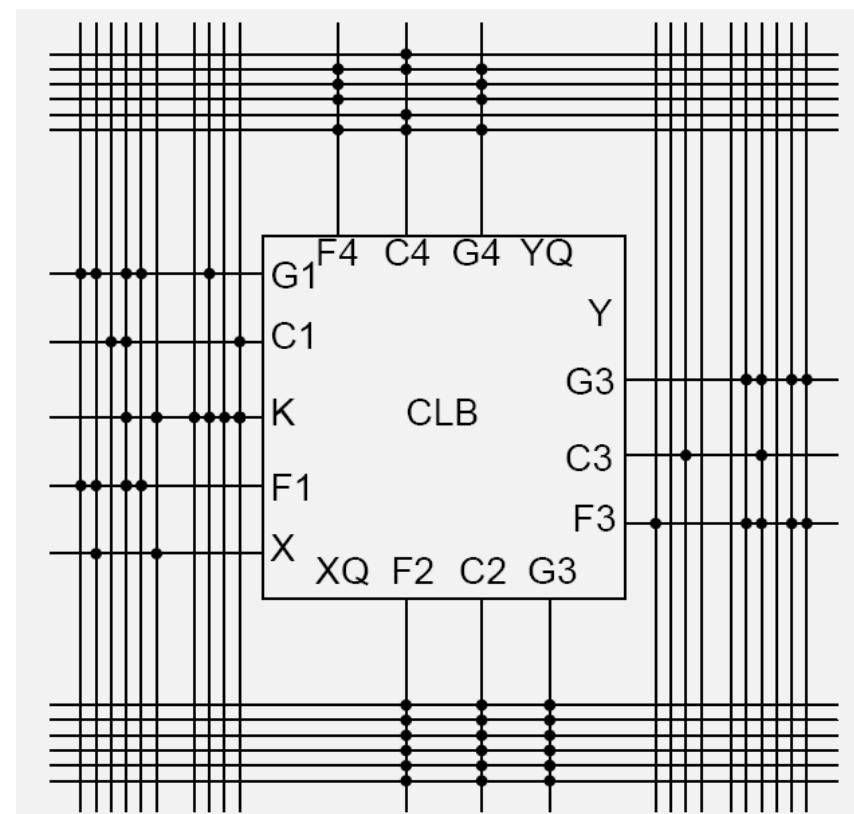
## ■ Connexions doubles:

- relient deux PSMs séparées par une PSM:
- Offrent une plus grande rapidité que les connexions simples pour les liens de longueur intermédiaire
- Maintiennent une bonne flexibilité de parcours.
- De la même manière, les connexions quadruples permettent des liens rapides un peu plus longs mais entraînent une certaine pénalité dans la flexibilité du parcours.

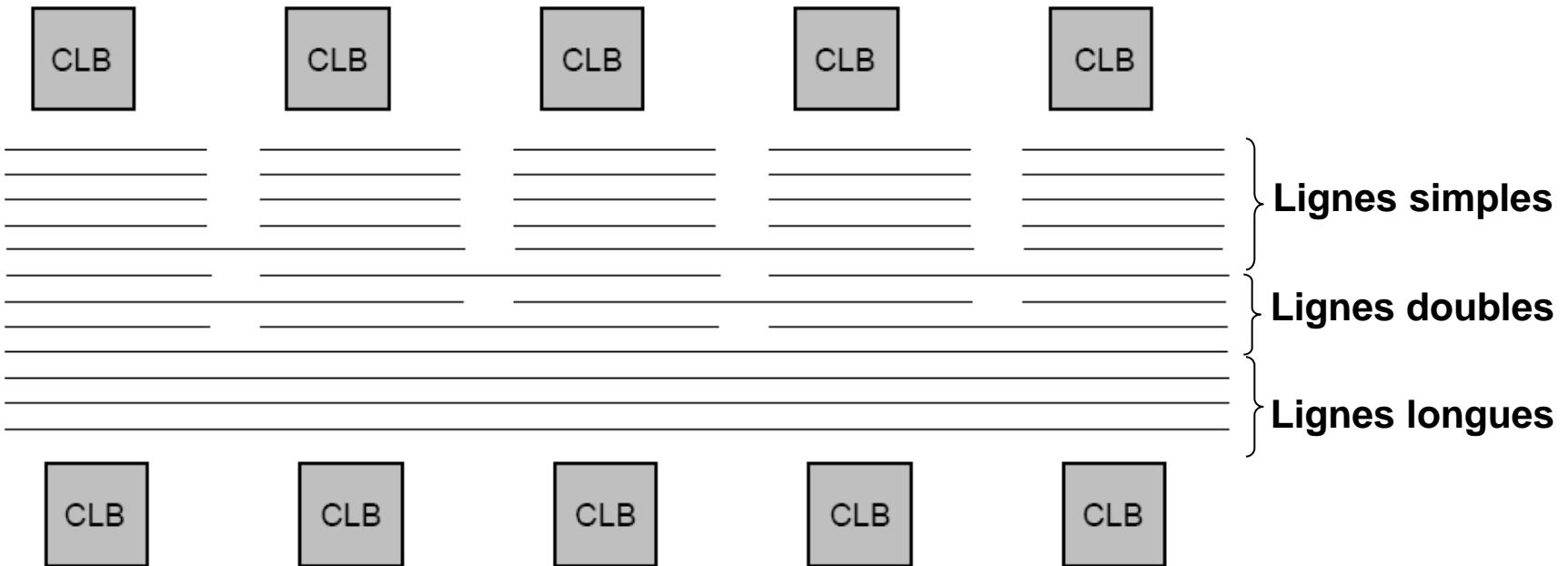


# FPGA: types de connexions

- **Les longues lignes** sont de longs segments métallisés parcourant toute la longueur et la largeur du composant, elles permettent éventuellement de transmettre avec un minimum de retard les signaux entre les différents éléments dans le but d'assurer un synchronisme aussi parfait que possible. De plus, ces longues lignes permettent d'éviter la multiplicité des points d'interconnexion.

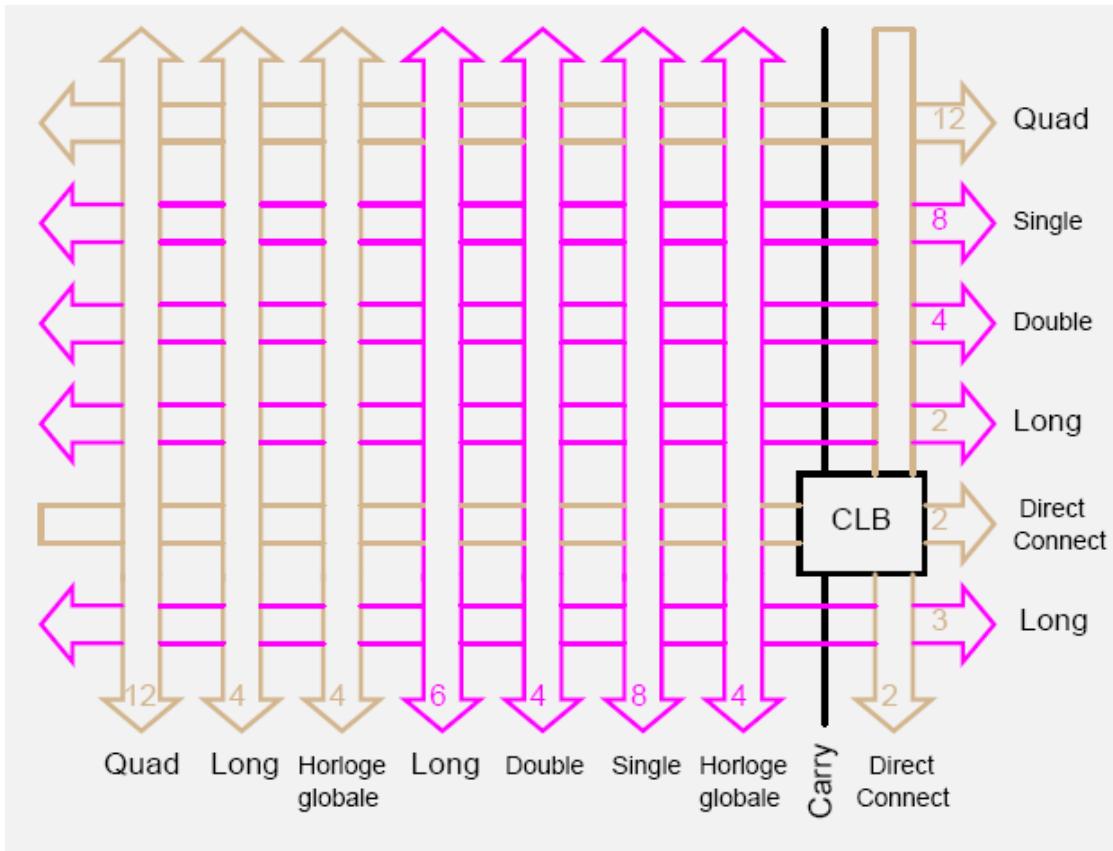


# FPGA: types de connexions



# FPGA: types de connexions

Introduction aux systèmes sur puce



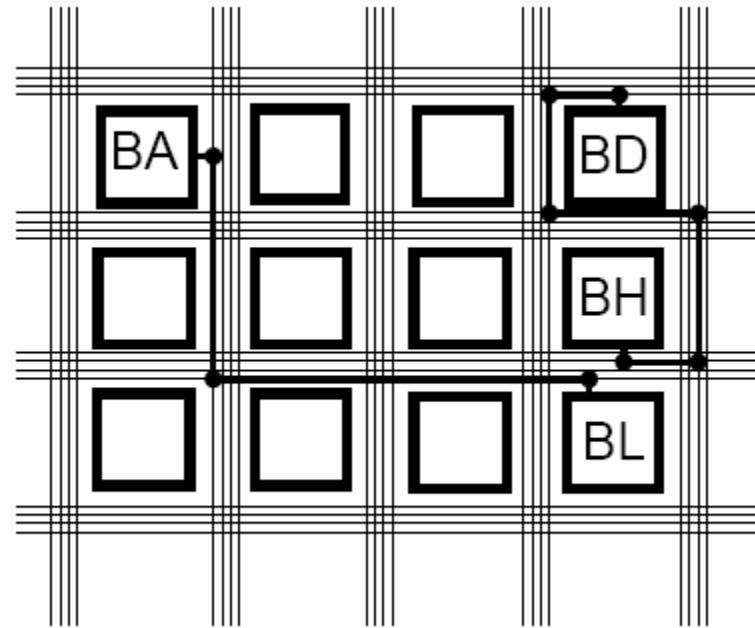
Xilinx XC4000X

# FPGAs : Interconnexions programmables

- Le passage d'un bloc logique à un autre se fera par un nombre de point de connexion (responsable des temps de propagation) fonction de la position relative des deux blocs logique et de l'état d'encombrement de la matrice.
- Ces délais ne sont donc pas prédictibles (contrairement au CPLD) avant le placement routage
- De la phase de placement des blocs logique dépendront donc beaucoup les performances du circuit en terme de vitesse

# FPGAs : Interconnexions programmables

- une liaison entre deux blocs logiques (BA et BL) éloignés, mais passant par peu de points de connexion, donc introduisant un faible retard,
- une liaison entre deux blocs proches (BD et BH) mais passant par de nombreux points de connexions, donc introduisant un retard important.



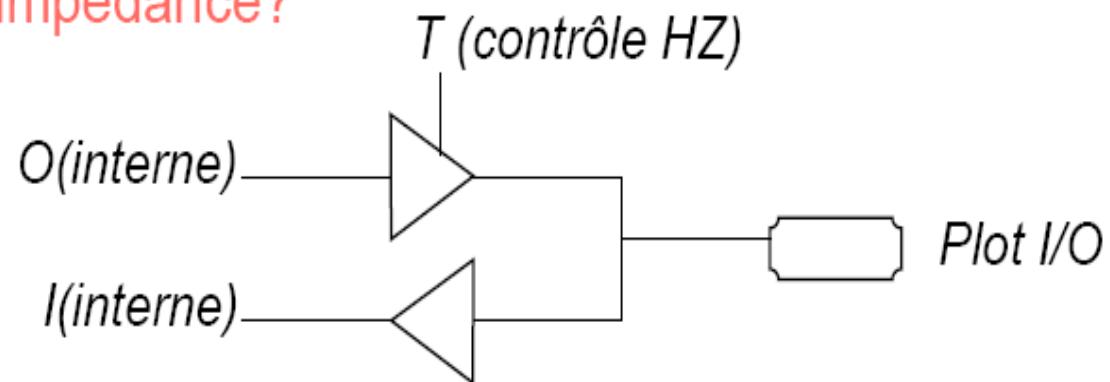
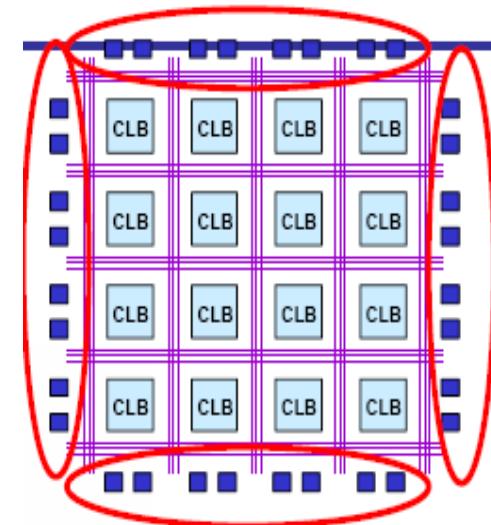
L'utilisation optimale des potentialités d'intégration d'un FPGA conduira à un routage délicat et pénalisant en termes de vitesse.

Il convient de noter que ces retards sont dus à l'interaction de la résistance de la connexion et de la capacité parasite ; cela n'a rien à voir avec un retard dû à la propagation d'un signal sur une ligne tel qu'on le voit en haute fréquence.

# I/O Blocs

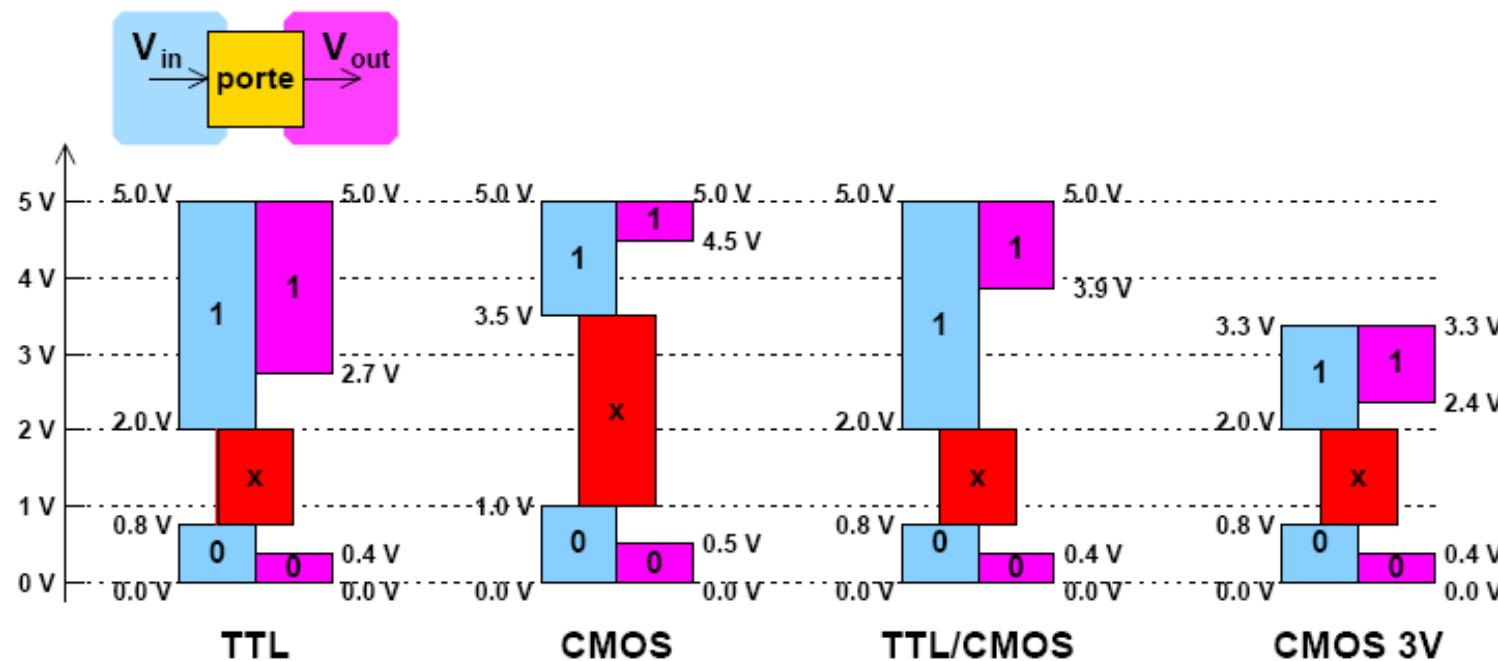
- Les blocs d'entrée/sortie (IOBs) contrôlent le flot de données entre les broches du FPGA et la logique interne utilisateur; les I/O sont:
  - Bidirectionnelles (entrée ou sortie)
  - 3 états (haute impédance)
  - À retards programmables
  - Multi-standards (niveaux logiques)

Bidirectionnel - haute impédance?



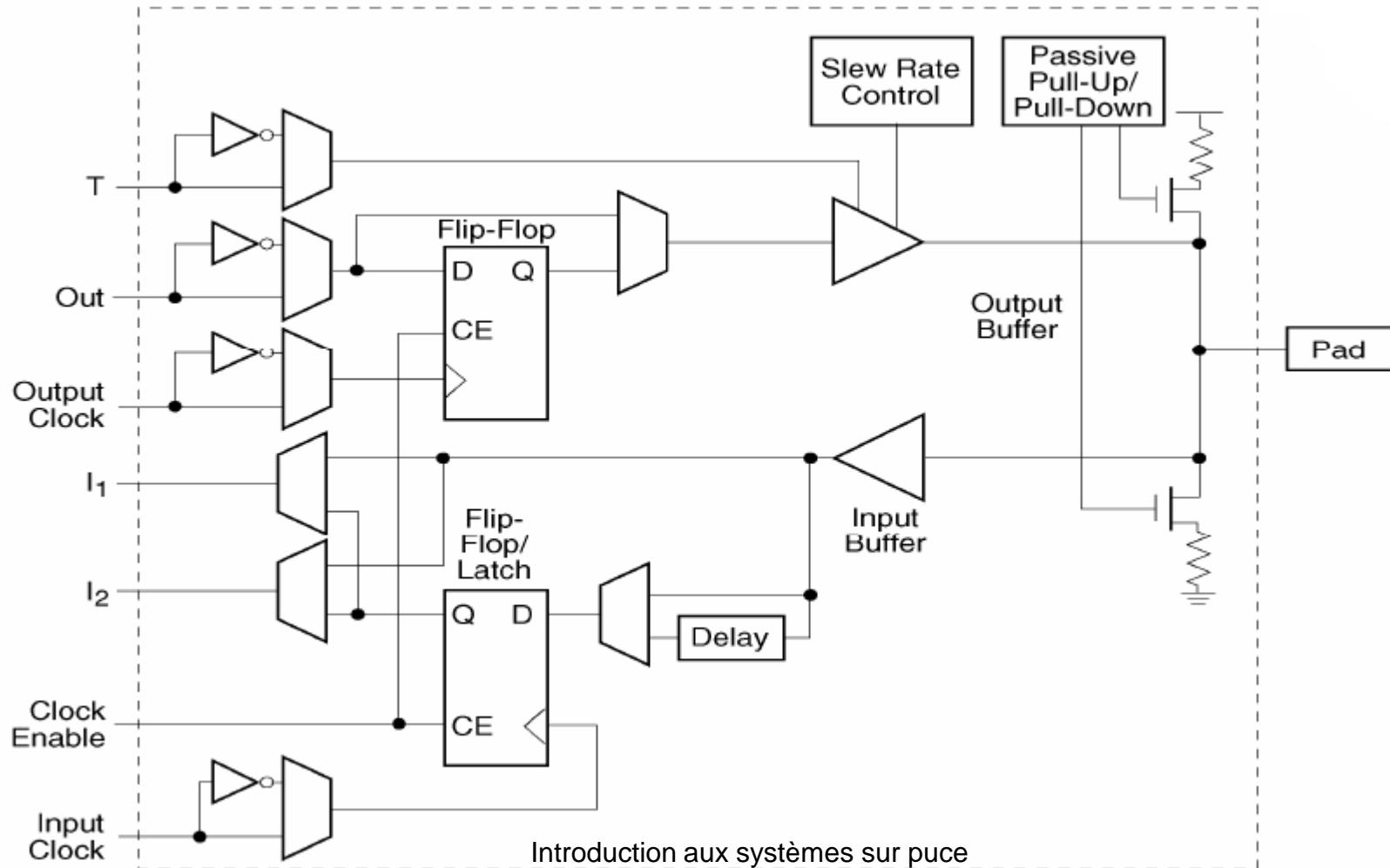
# I/O Blocs

Problème de faire communiquer les composants entre eux : il y a plusieurs codages en tension des niveaux 1 et 0 (tension d'alimentation VDD, type de marges):  
Les buffers permettent de résoudre ce problème

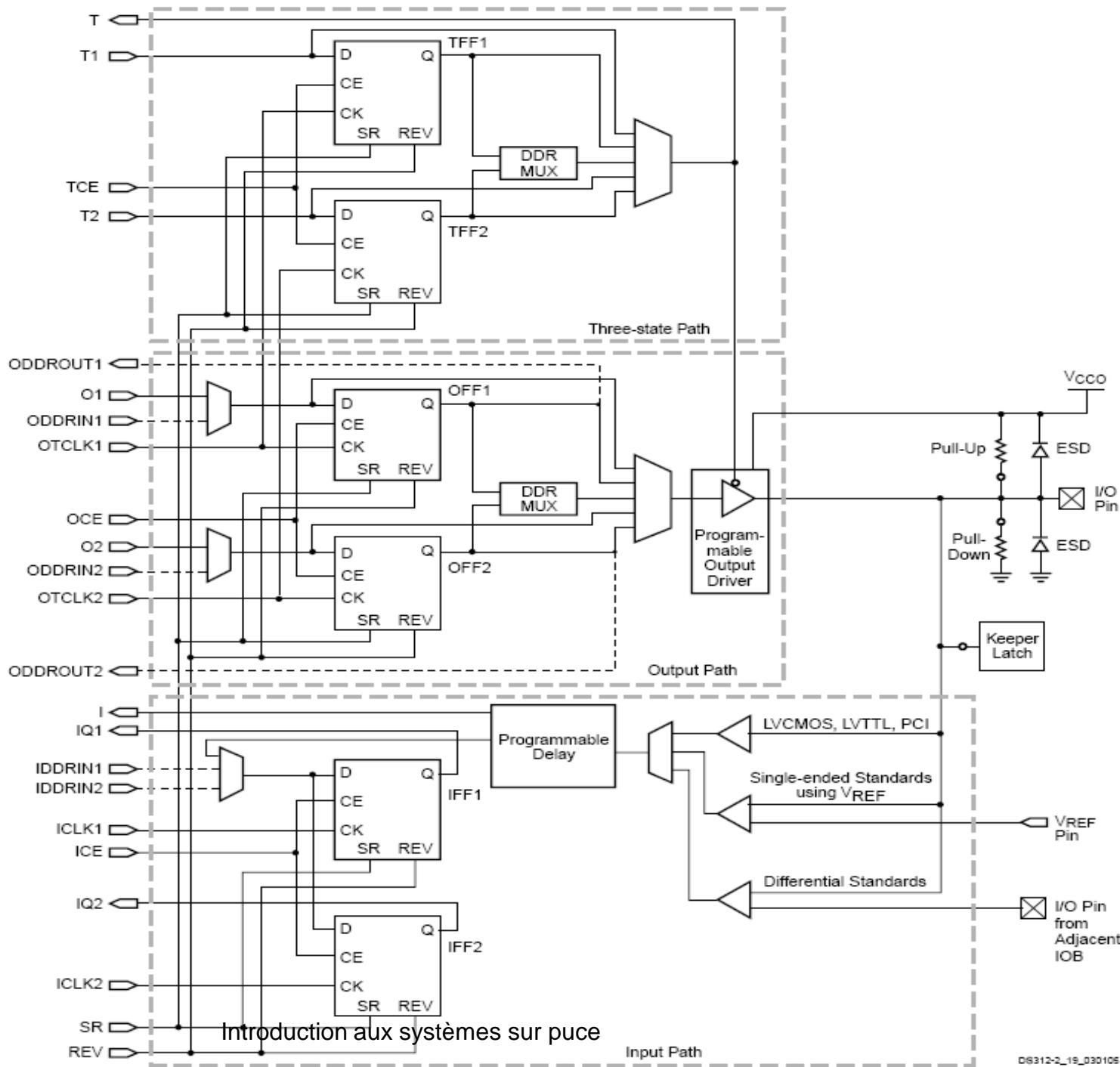


# FPGA : Les entrées/sorties

FPGAs : Xilinx IOB (Input Output Block)



# Sparta n 3E



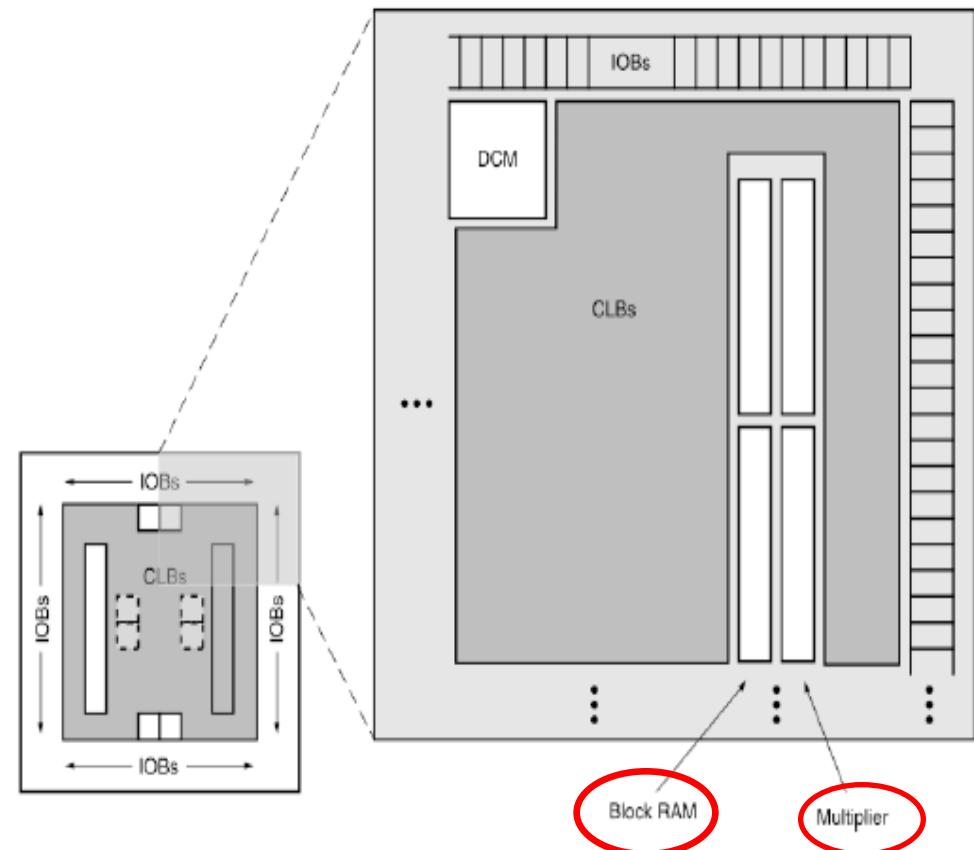
# FPGA: et encore!!!

Xilinx Spartan 3E:

- 1) Configurable Logic Blocks (CLBs)
- 2) Input/output blocks (IOBs)
- 3) Block RAM
- 4) Multiplier blocks
- 5) Digital Clock Manager(DCM)

Circuit XC300S500E  
(plate-formes projet):

- 500.000 gates
- 1164 CLB
- 400.000 bits RAM
- 232 I/O utilisateur
- 4 horloges globales



# La technologie actuelle des FPGA

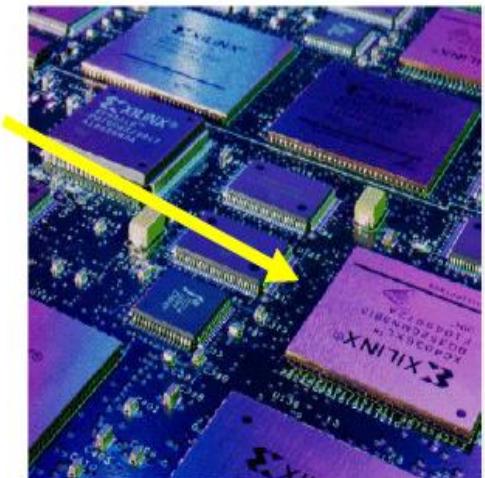
- Jusqu'à 200 000 cellules logiques élémentaires
  - Cellule = Look-up-Table + registres
- **Blocs mémoire configurables**
  - Blocs de 18kbits, au total jusqu'à 1Mo
- **Structures dédiées pour les traitements arithmétique**
  - 512 multiplicateurs 18x18 bits embarqués
- Processseurs embarqués
  - Jusqu'à 4 PowerPC en « dur »
  - Soft-Core : processeurs utilisant les cellules logique
- Liaisons séries haut débit
  - Jusqu'à 10 Gbits
  - MAC Ethernet intégrée

# FPGA: Les modes de configuration

- Le **bitstream** décrit la configuration de tous les éléments configurables du circuits
- Un transfert de bitstream est nécessaire lors de « *la mise sous tension* » et à chaque reconfiguration »



10010010011110010110

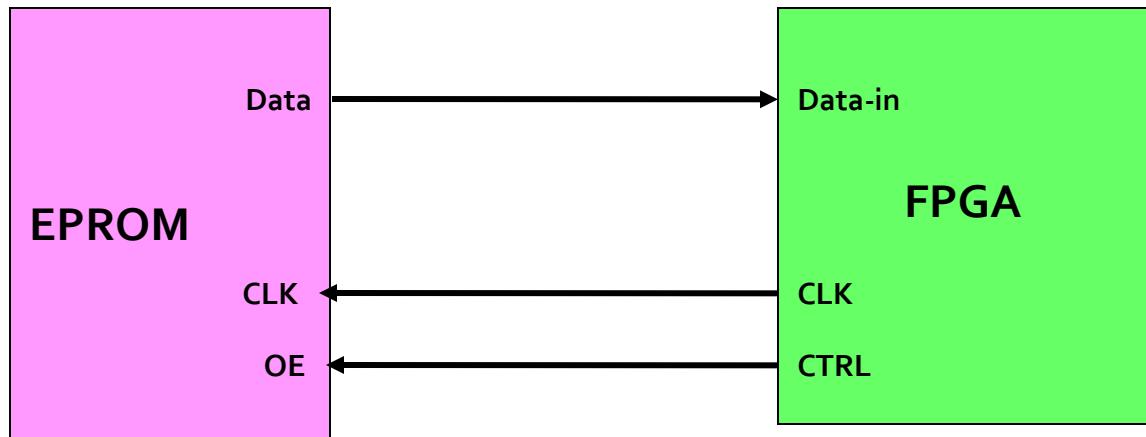


# FPGA: Les modes de configuration

- On rencontre couramment 6 techniques de configuration :
  - **Master** mode série ou parallèle
  - **Slave** mode série ou parallèle
  - **Peripheral** mode série ou parallèle
- L'envois des données de configurations peut se faire en série ou en parallèle
- La sélection de la technique de configuration se fait grâce à des bits de configuration du FPGA
- Dans tous les cas la configuration se fait via un fichier de configuration binaire : **le Bitstream**
- Les entrées sorties utilisées pendant la configuration sont aussi des I/O du circuits utilisables en fonctionnement

# Master mode

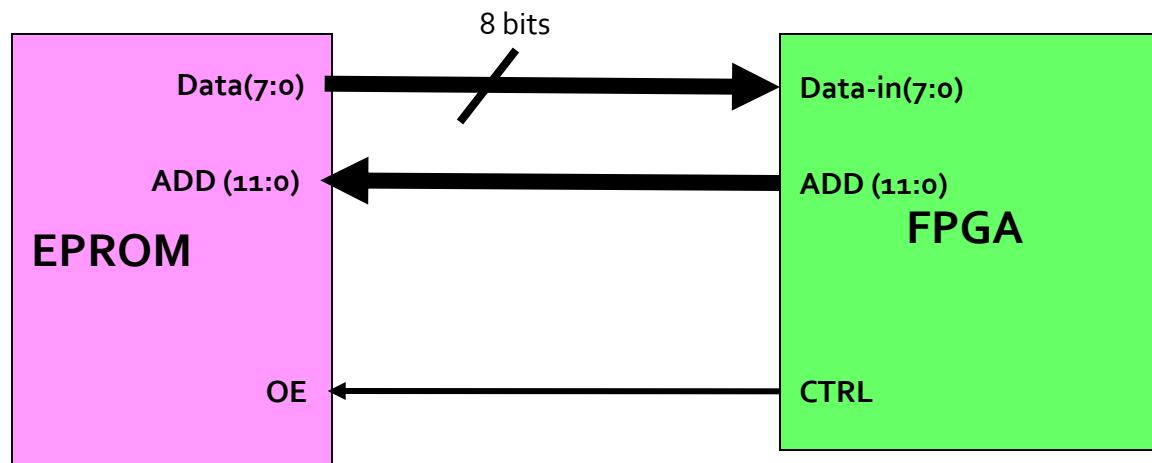
Dans ce cas le FPGA est maître de sa configuration



MODE SERIE

# Master mode

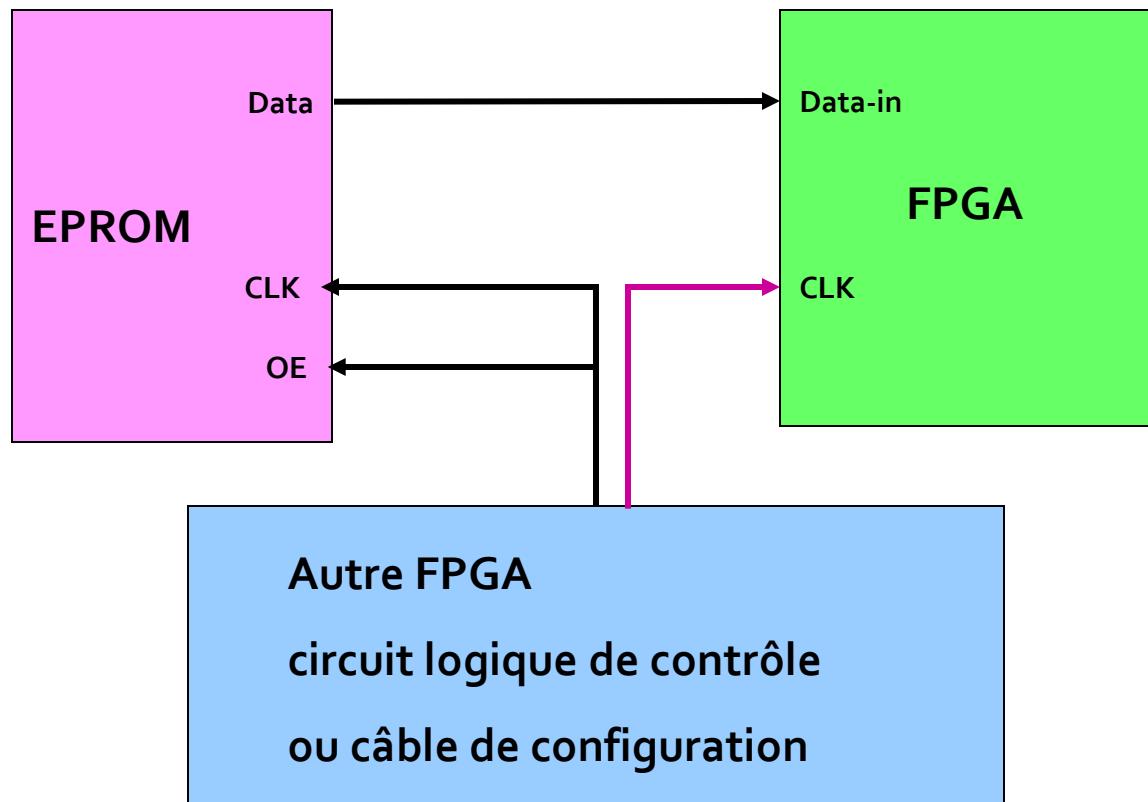
Dans ce cas le FPGA est maître de sa configuration



MODE PARALLELE

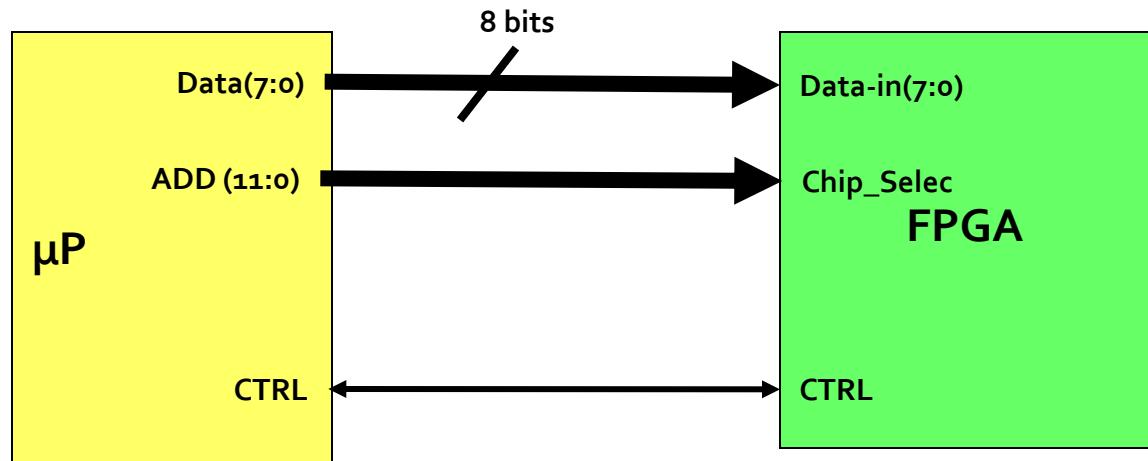
# Slave mode

Dans ce cas le FPGA est esclave, il subit sa configuration



# Peripheral mode

Le FPGA est vu comme un périphérique du microprocesseur



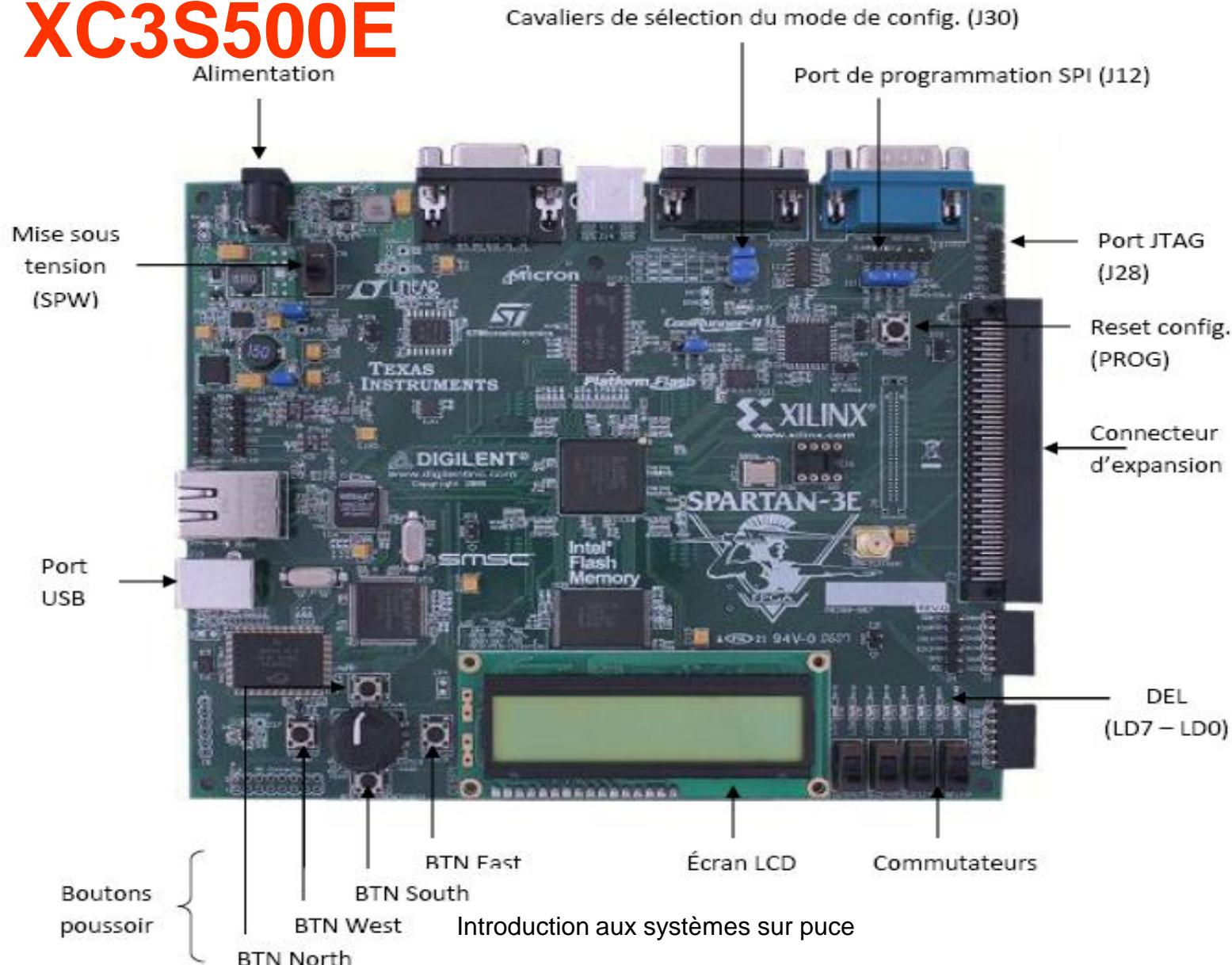
Le microprocesseur peut être un cœur de processeur  
embarqué dans le circuit

# Cas d'étude : Xilinx Spartan 3E

- Plusieurs versions:

# Carte digilent à base de Spartan

## XC3S500E





# Carte Spartan XC3S500E : Composants

La carte Spartan 3E regroupe :

- un FPGA **XC3S500E** Spartan 3E,
- un CPLD Coolrunner II,
- une mémoire PROM de 4 Mbit,
- une mémoire Flash sérielle de 16 Mbit,
- une mémoire Flash parallèle de 128 Mbit,
- une mémoire DDR SDRAM de 512 Mbit,
- un écran LCD,
- un port PS/2,
- un port VGA,
- une prise Ethernet 10/100,
- deux ports RS-232,
- un port de configuration USB,
- **deux convertisseurs de données**,
- un oscillateur à 50 MHz,
- un connecteur d'expansion Hirose FX2 permettant 40 entrées/sorties génériques,
- 8 DEL,
- des boutons poussoir et des commutateurs.

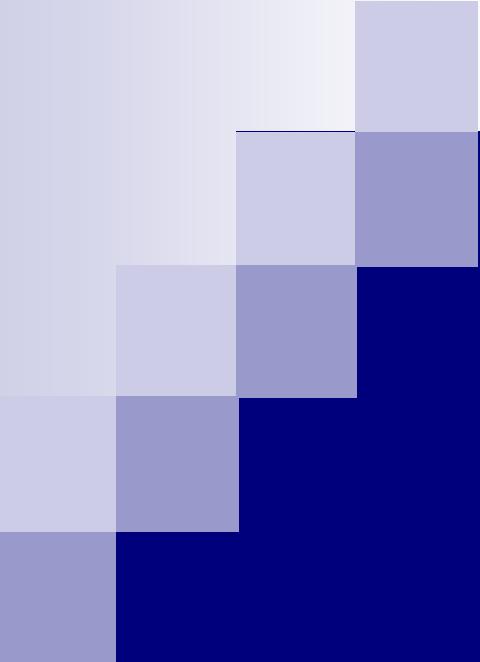


# Carte Spartan 3E: modes de configuration

- La carte supporte **trois modes** de configuration à la mise sous tension,
  - soit le mode **Master-Slave** utilisant le PROM 4 Mbits (1MBits sur version ISI),
  - un mode **SPI** utilisant la mémoire Flash série
  - et un mode **BPI** utilisant la mémoire Flash parallèle.
- La carte peut aussi être programmée directement à l'aide d'un port **JTAG** lorsqu'elle est sous tension.

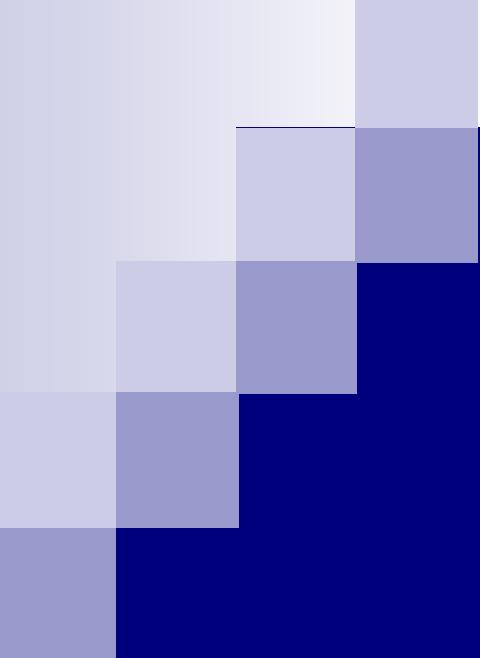
# Flot de conceptions des FPGA

- Voir Module Synthèse VHDL d'architecture



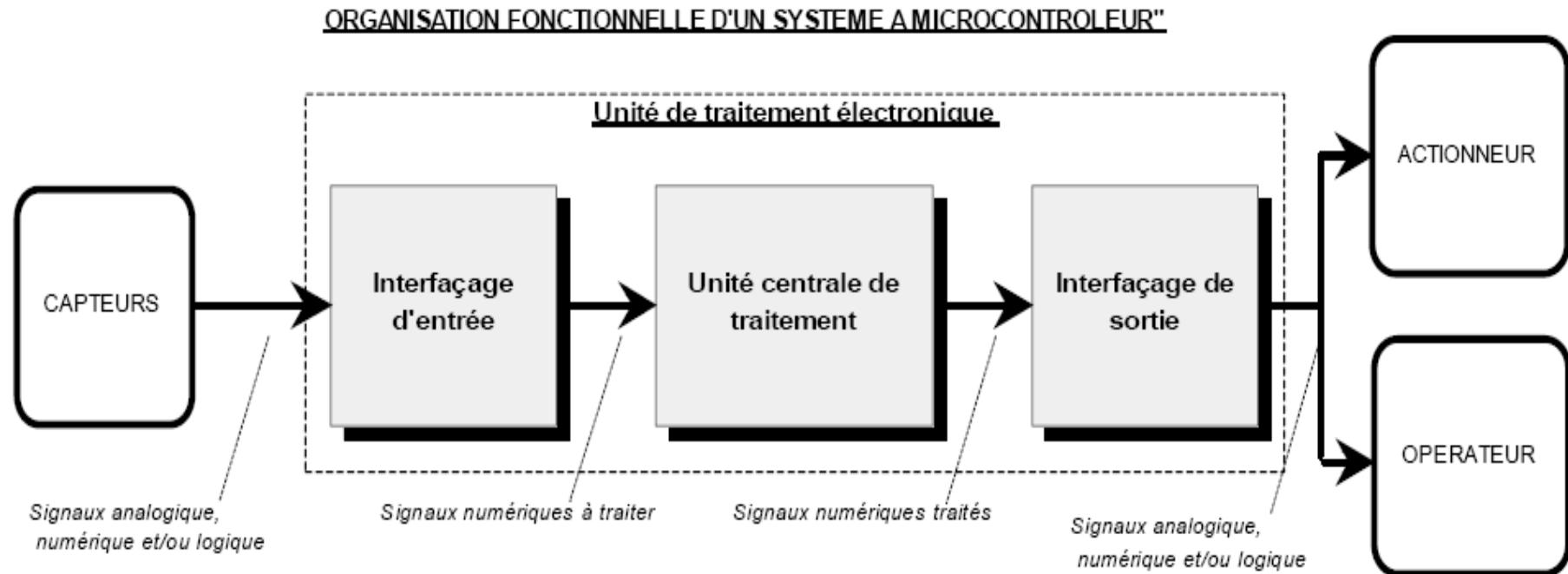
# Chapitre 3 : Cible logicielle- les microcontrôleurs

Hanen ben Fradj



# 1. Fonctionnement et définition

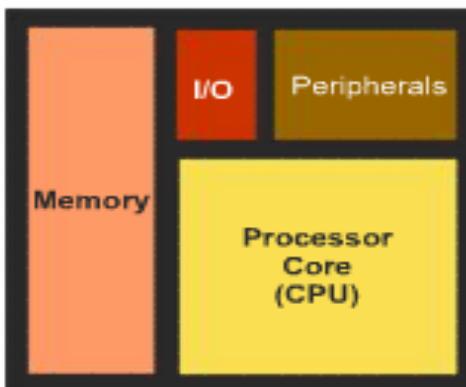
# Organisation fonctionnelle d'un système à microcontrôleur



# Définition

- C'est une **unité de traitement de l'information** de type microprocesseur à laquelle on a ajouté des périphériques internes permettant de réaliser des montages sans nécessiter l'ajout de composants externes.
- Ce sont des systèmes minimum sur une seule puce. Ils contiennent : un CPU, de la RAM, de la ROM et des ports d'Entrée/Sorties (parallèles, séries, etc..).

- **Microcontroller**
  - Single-chip device
  - System decisions based on external signals
  - Controls the behavior of a system
- **Always contains:**
  - Processor core
  - Memory (RAM, ROM)
  - Input/Output (I/O) capability
    - Serial I/O interfaces
  - Various on-chip peripherals such as:
    - Timers
    - Analog to digital convertors
    - Pulse width modulators



# Définition

- Ils comportent aussi des fonctions spécifiques comme des compteurs programmables pour effectuer des mesures de durées, des CAN voir des CNA pour s'insérer au sein de chaînes d'acquisition, des interfaces pour réseaux de terrain, etc ...
- Il est adapté pour répondre au mieux aux besoin des applications embarquées (appareil électroménagers, chaîne d'acquisition, lecteur carte à puce, etc...). Il est par contre généralement moins puissant en terme de rapidité, de taille de données traitables ou de taille de mémoire adressable qu'un microprocesseur.

# Définition

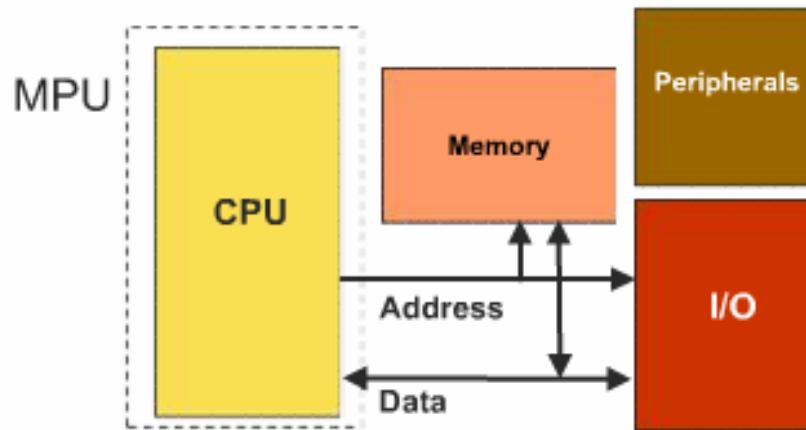
## Avantages :

- . Encombrement réduit,
- . Circuit imprimé peu complexe,
- . Faible consommation,
- . Coût réduit.

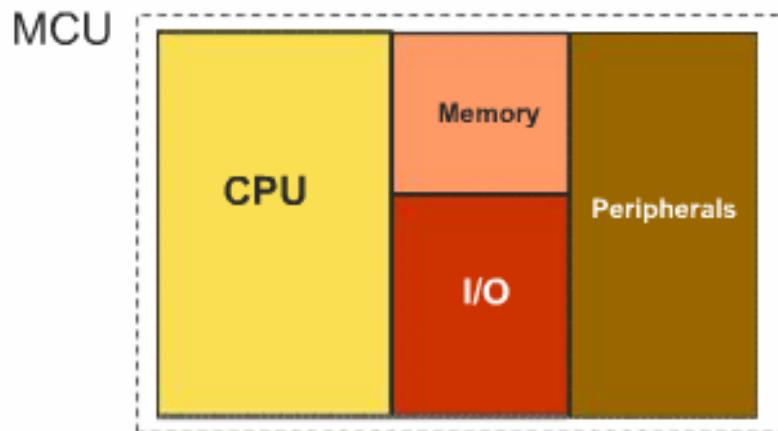
## Inconvénient :

- . Système de développement onéreux,
- . Programmation nécessitant un matériel adapté.

# Différence entre microprocesseur et microcontrôleur

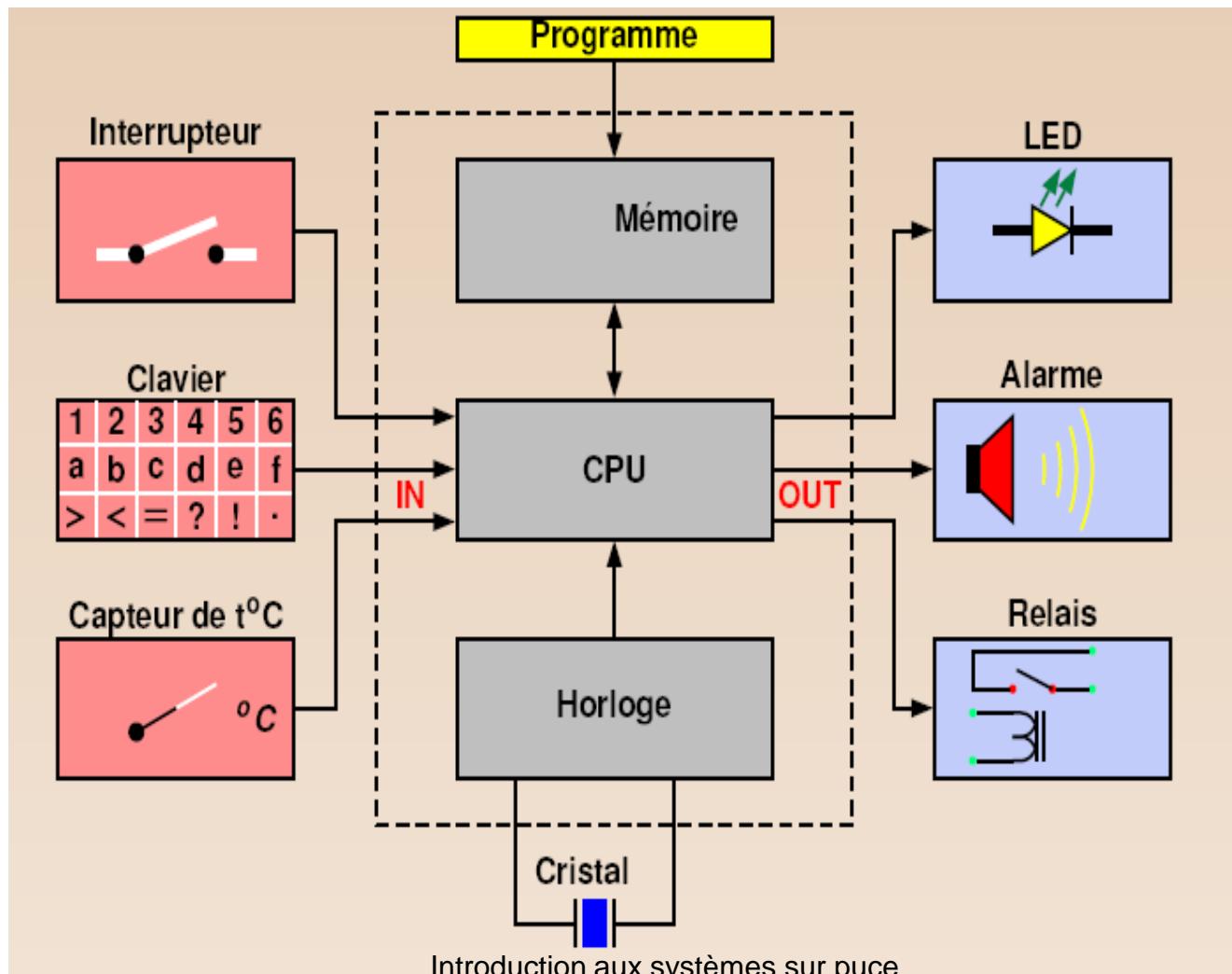


- Microprocessors (MPU): CPUs that connect to external memory and peripherals

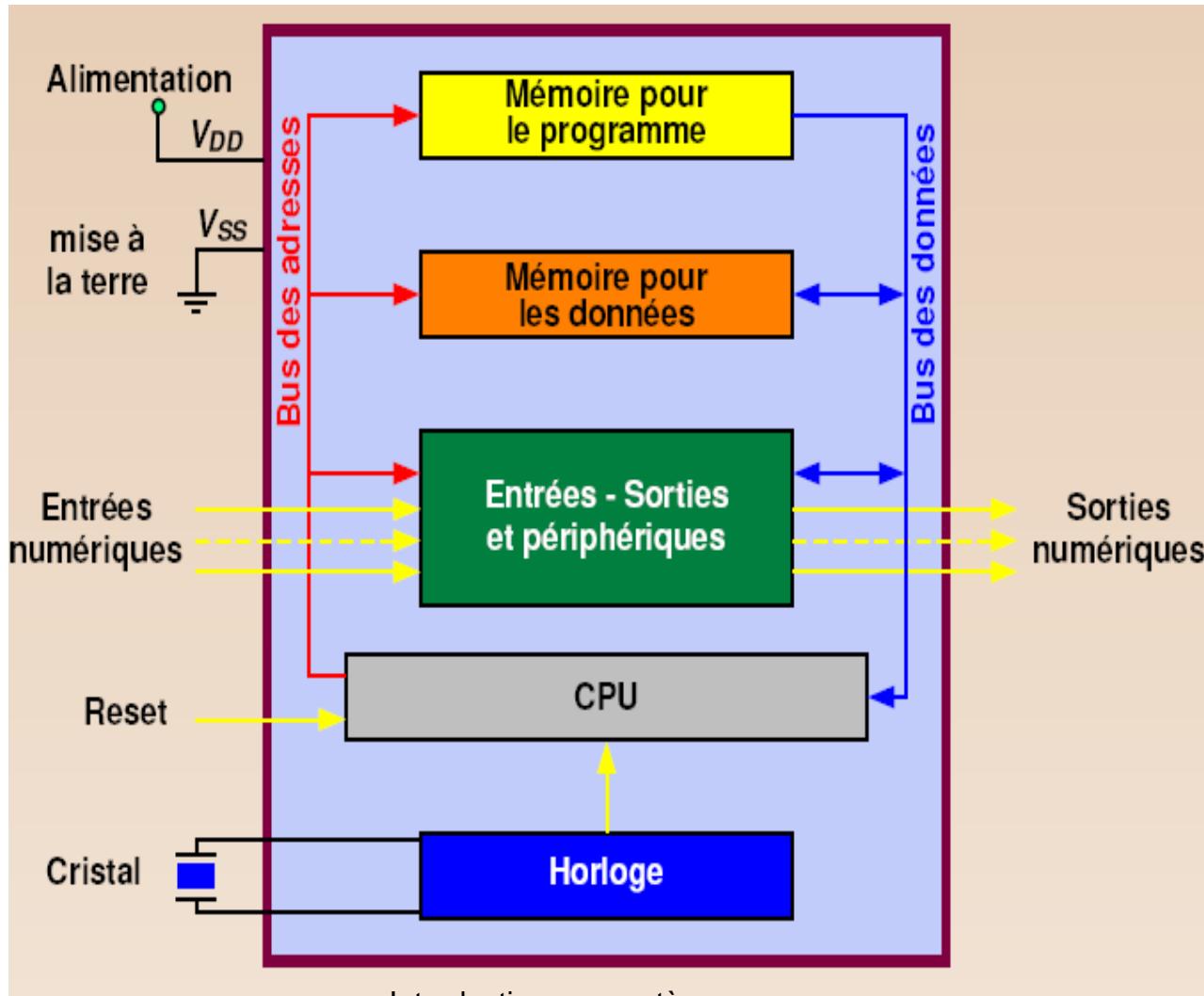


- Microcontrollers (MCU): have CPU core with memory, I/O and peripherals integrated on-chip

# fonctionnement



# Exemple: M68HC05

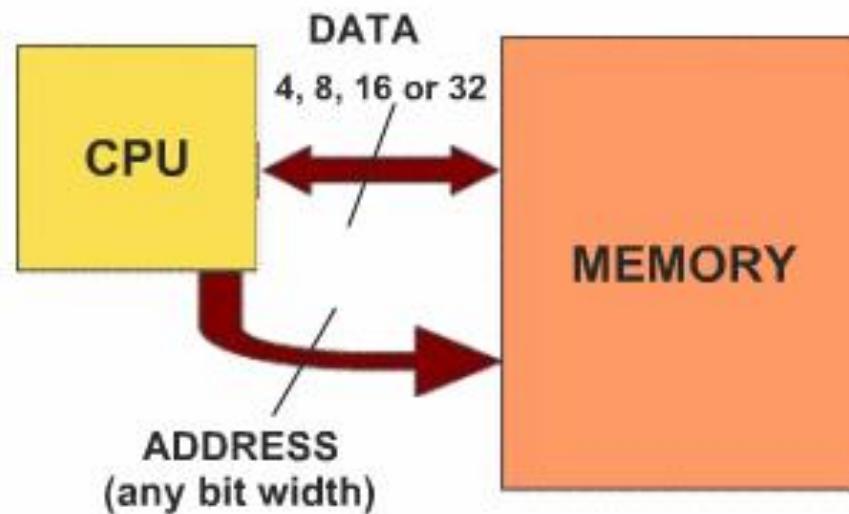




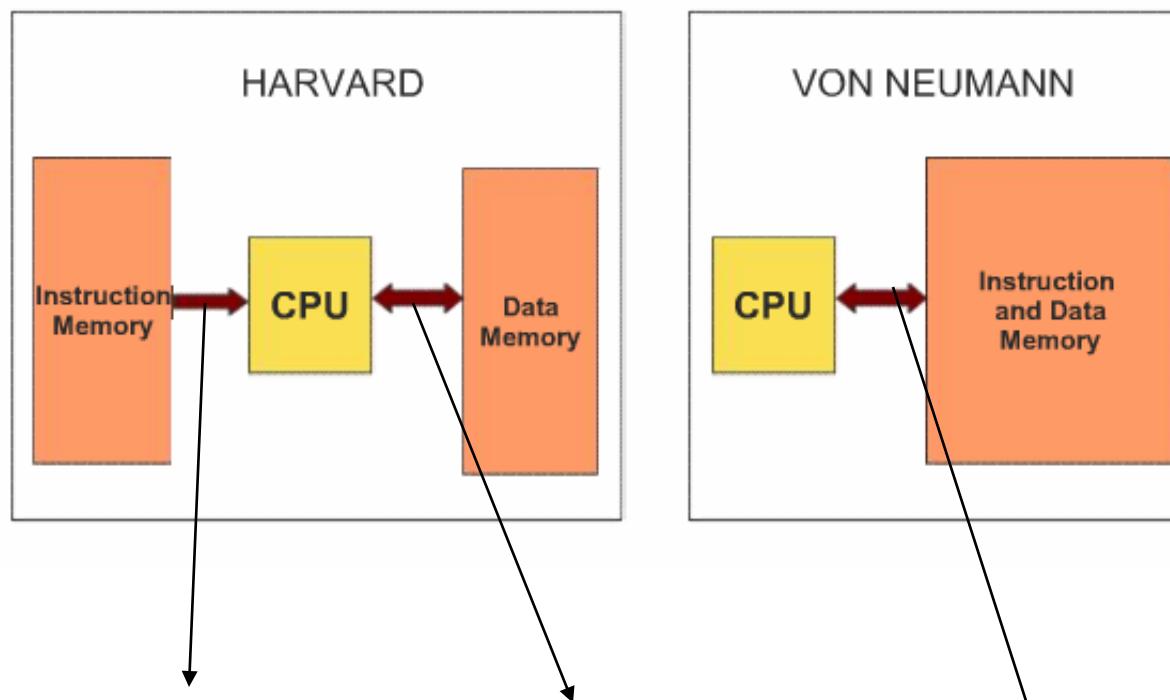
## 2. Architecture

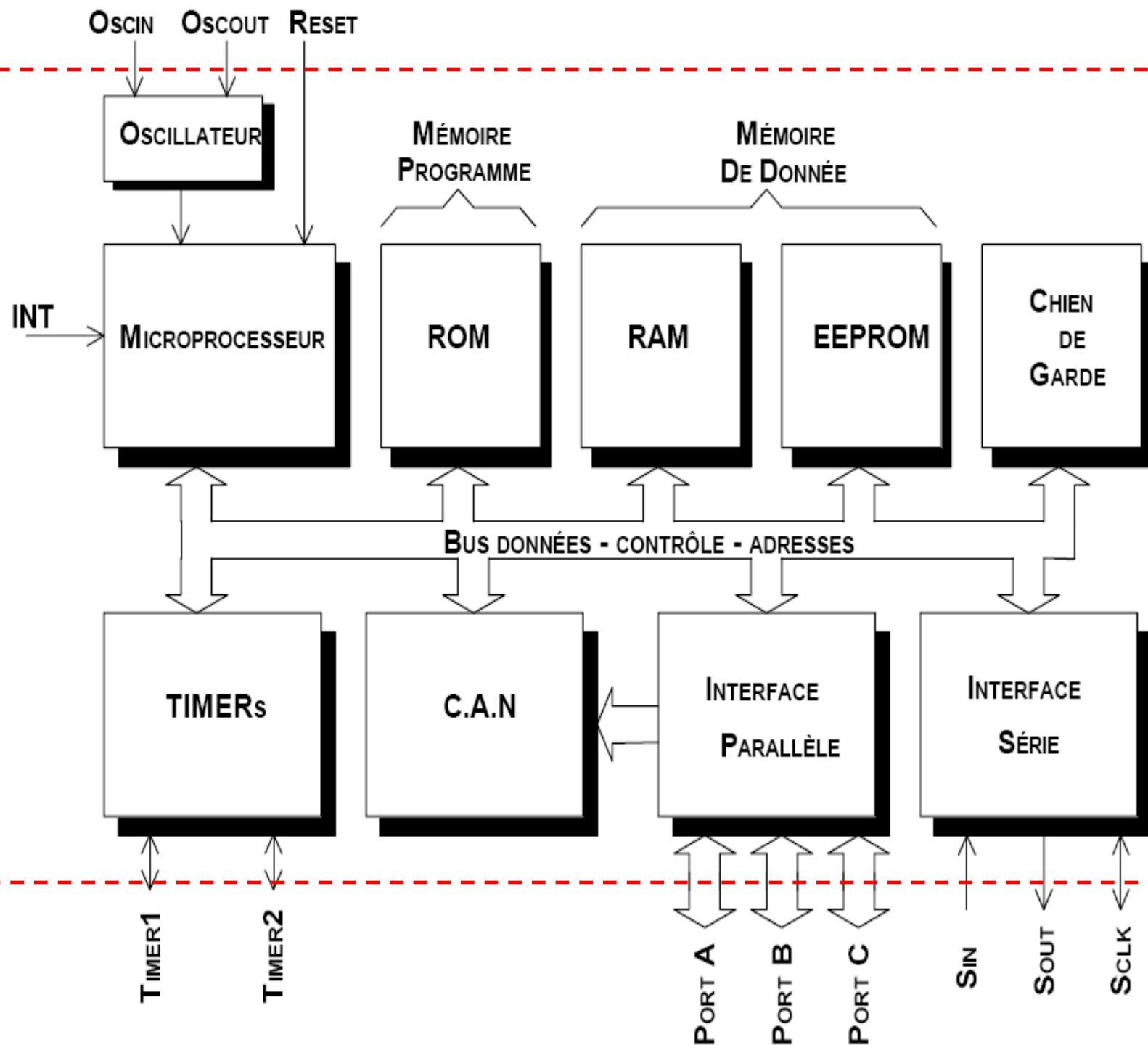
# Bit définition

- 4, 8, 16, or 32 are current options.



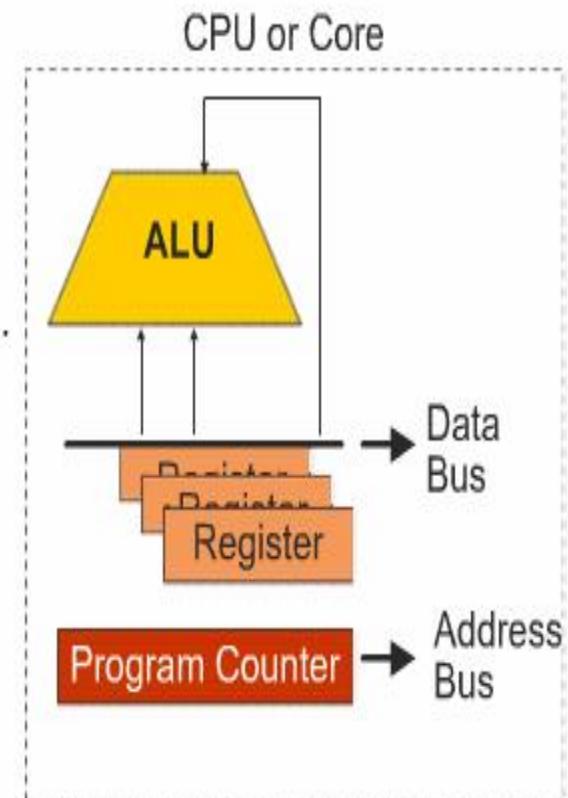
# Harvard ou Von Neumann





# 1. Le microprocesseur

- **RISC ou CISC:** Plus on réduit le nombre d'instructions, plus facile et plus rapide en est le décodage, et plus vite le composant fonctionne. Cependant, il faut plus d'instructions pour réaliser une opération complexe.
- **Jeu d'instructions**
- Codage des instructions et **Mode d'adressage**

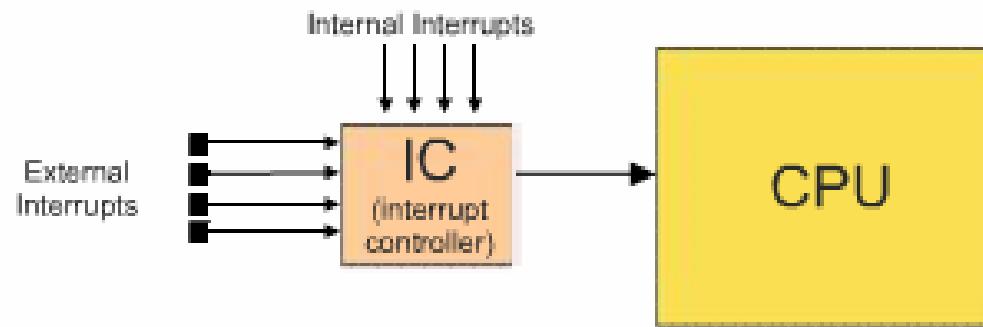


# 1. Le microprocesseur: Contrôleur d' interruptions

- Le microcontrôleur, dans son environnement, est destiné à traiter des informations en “**temps réel** ”.
- L’application est couplée au monde extérieur, par l’échange fréquent de messages et de signaux à des instants prévus.
- Il est dans l’obligation de changer d’état en fonction des priorités relatives de l’opération en cours et de celle qui lui est demandé. Il interrompt ou non le déroulement normal du programme en fonction d’une demande externe.
- Celles-ci sont vues du microcontrôleur comme des demandes d’interruption.
- deux types d’interruption :
  - **NMI (No Maskable Interrupt)** : interruption non - masquable,
  - **IRQ (Interrupt Request)** : interruption masquable.

# 1. Le microprocesseur: Contrôleur d' interruptions

- Du matériel qui gère les signaux d'interruption



- Quelque soit l'entrée d'interruption activée, le microprocesseur réalise des tâches identiques :
  - dans tout les cas, le programme principal est interrompu ;
  - le processeur doit sauvegarder le contenu du PC dans la pile ;
  - le processeur exécute une séquence privilégiée, reflet du type de traitement d'interruption ;

## 2. Mémoire programme

- Ce dispositif contient les instructions du programme que doit exécuter le microprocesseur. Ce type de mémoire (appelée mémoire morte), est uniquement accessible en lecture. Sa programmation nécessite une procédure particulière et un matériel adéquat.
- Il en existe différents types selon leur mode de programmation :
  - De la PROM programmable électriquement une seule fois par le développeur (appelée aussi OTPROM),
  - De la EPROM programmable électriquement et effaçable aux U-V (appelée aussi UV PROM),
  - De la EEPROM programmable et effaçable électriquement.
  - De la mémoire flash (programmable à une vitesse plus rapide, densité élevée)

# 3. Mémoires de données

Ce dispositif permet de mémoriser temporairement les données générées par le microprocesseur pendant les différentes phases du traitement numérique (résultats d'opérations, états des capteurs...). Ces mémoires sont accessibles en écriture et en lecture.

On en trouve 2 types :

- **De la mémoire vive (RAM) volatile** : (données perdues en cas de coupure de l'alimentation) ayant un temps de lecture et écriture assez court(quelques ns),
- **De la mémoire morte (EEPROM) non-volatile** : (données conservées en cas de coupure de l'alimentation) ayant un temps d'écriture assez élevé (quelques ms) par rapport au temps de lecture qui est assez faible (quelques ns).

# 4. L'interface parallèle GPIO (General Purpose Input Output)

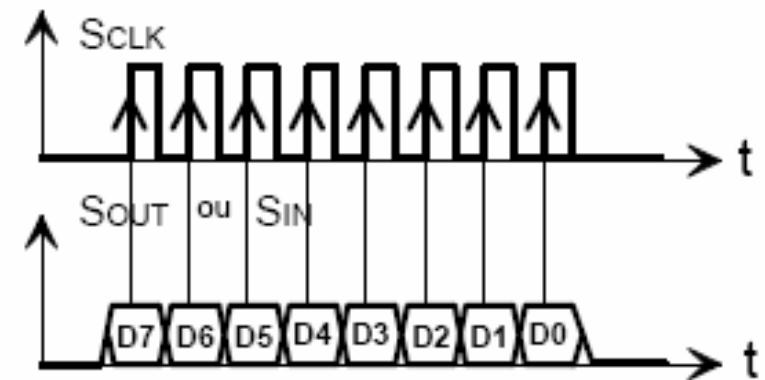
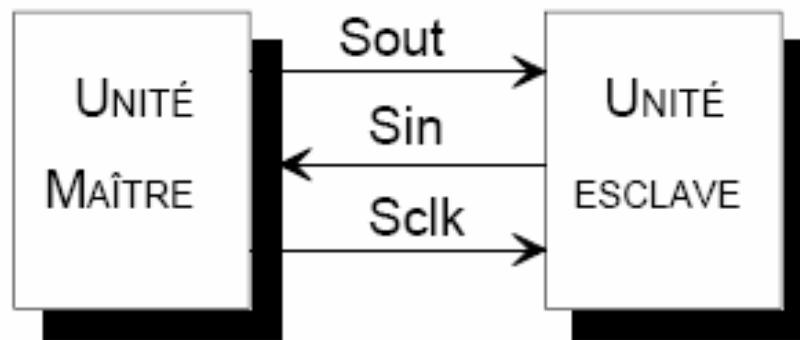
- Ce type d'interface, répartie sur plusieurs ports (maximum 8 bits), permet de prendre en compte des états logiques appliqués en entrée (état de capteurs) ou de générer des signaux binaires en sortie (commande d'actionneurs).
- Les broches de ces ports peuvent donc être configurées en entrée ou en sortie, avec différentes options (résistances de rappel, sorties collecteurs ouverts, interruption...).
- La configuration ainsi que l'état logique de ces broches est obtenue par des opérations d'écriture ou de lecture dans différents registres associés à chaque port.
- On trouve généralement :
  - **Un registre de direction** pour une configuration en entrée ou en sortie,
  - **Un registre de donnée** recopiant les états logiques de chaque broche de port,
  - **Un registre d'option** permettant plusieurs configurations en entrée ou en sortie.

# 5. L'interface série: USART (Universal Synchronous Asynchronous Receiver Transmitter)

- Ce type d'interface permet au microcontrôleur de communiquer avec d'autres systèmes à base de microprocesseur.
- Les données envoyées ou reçues se présentent sous la forme d'une succession temporelle (sur un seul bit) de valeurs binaires images d'un mots.
- Il y a 2 types de liaison série : **synchrone et asynchrone**.

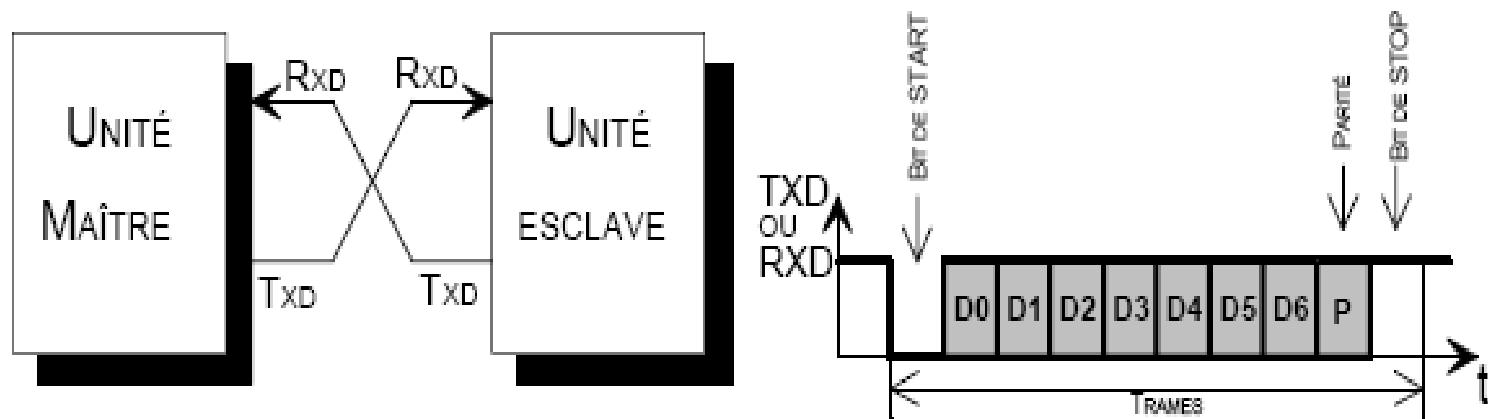
# 5. L'interface série : Liaison série synchrone

Dans ce dispositif la transmission est synchronisée par un signal d'horloge émis par l'unité **maitre**



# 5. L'interface série: Liaison série asynchrone

- Ce dispositif ne possède pas de signal d'horloge de synchronisation.
- Les unités en liaison possèdent chacune une horloge interne cadencée à la même fréquence.
- Lorsqu'une unité veut émettre un mot binaire, elle génère un front descendant sur sa ligne émettrice. A la fin de l'émission de ce mot, la ligne repasse au niveau haut.
- La donnée à transmettre peut contenir un bit supplémentaire appelé "parité" et servant à la correction d'erreurs



# 5. L'interface série

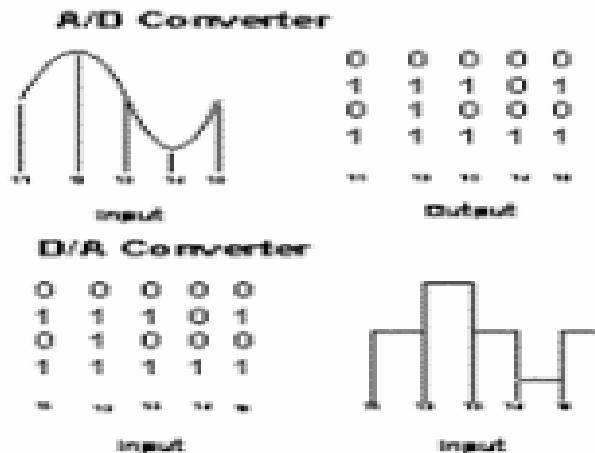
- Plusieurs normes de communication série:
  - USB (Universal serial Bus)
  - CAN (Controller Area Network)
  - SPI (Serial Peripheral Interface)
  - I2C (Inter Integrated Circuit)
  - UART (Universal Asynchronous Receiver Transmitter)
  - LIN (Local Interconnect Network)
  - Ethernet
  - Wireless interfaces
    - Zigbee
    - bluetooth

# 5. L'interface série : LA NORME RS232

- **Longueur des mots** : 7 bits (ex : caractère ascii) ou 8 bits
- **La vitesse de transmission** : elle est défini en bits par seconde ou bauds. Elle peut prendre des valeurs allant de 110 à 115 200 bds.
- **Parité** : le mot transmis peut être suivi ou non d'un bit de parité qui sert à détecter les erreurs éventuelles de transmission.
- **Bit de start** : la ligne au repos est à l'état logique 1 pour indiquer qu'un mot va être transmis la ligne passe à l'état bas avant de commencer le transfert. Ce bit permet de synchroniser l'horloge du récepteur.
- **Bit de stop** : après la transmission, la ligne est positionnée au repos pendant 1, 2 ou 1,5 périodes d'horloge selon le nombre de bits de stop.
- **Niveau de tension** : Un “0” logique est matérialisé par une tension comprise entre 3 et 25V, un “1” par une tension comprise entre -25 et -3 V. Des circuits spécialisés comme le MAX 232 réalise la conversion à partir de niveau TTL.

# 6. Convertisseur Analogique Numérique (CAN / ADC)

- Le ADC intégré dans les microcontrôleurs est généralement du type “Approximations successives”.
- Il possède plusieurs entrées multiplexées accessibles via les broches des ports de l’interface parallèle.
- Le ADC possède normalement 2 registres :
  - Un registre de données contenant le résultat de la conversion,
  - Un registre de contrôle permettant de lancer et de surveiller la conversion.



# 7. TIMER

Le **Timer** permet de réaliser les fonctions suivantes :

1. Génération d'un signal périodique
2. Génération d'une impulsion calibrée,
3. Temporisation: compter les cycles d'horloge du uc lui-même. Dans ce cas, comme l'horloge est fixe, il s'agit en réalité de compter du temps.
4. Comptage d'événements: compter les impulsions reçues sur une pin d'entrée.

→ Plusieurs registres associés au Timer permettent de configurer les différents modes décrits précédemment.

# 8. Le chien de garde (Watchdog)

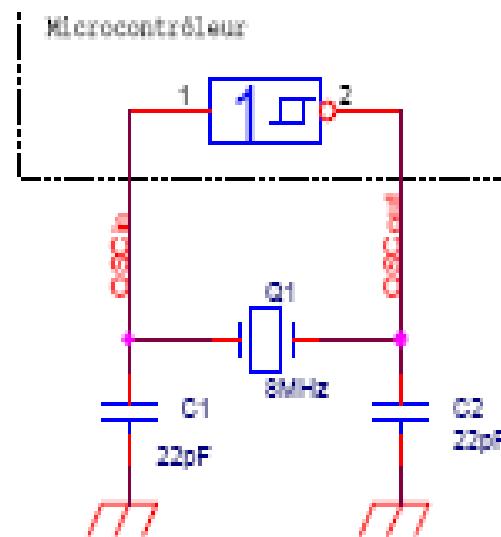
- Ce dispositif est un système **anti-plantage** du microcontrôleur. Il s'assure qu'il n'y ait pas d'exécution prolongé d'une même suite d'instructions.
- il s'agit en général d'un compteur qui est régulièrement remis à zéro.
- Si le compteur dépasse une valeur donnée (*timeout*) alors on procède à un *reset* (redémarrage) du système.
- Le chien de garde consiste souvent en un registre qui est mis à jour via une interruption régulière. Il peut également consister en une routine d'interruption qui doit effectuer certaines tâches de maintenance avant de redonner la main au programme principal. Si une routine entre dans une boucle infinie, le compteur du chien de garde ne sera plus remis à zéro et un *reset* est ordonné.
- Le chien de garde permet aussi d'effectuer un redémarrage si aucune instruction n'est prévue à cet effet. Il suffit alors d'écrire une valeur dépassant la capacité du compteur directement dans le registre : le chien de garde lancera le *reset*.

# 9. Direct Memory Access (DMA)

- Transfert rapide de données entre
    - Mémoire Périphériques
    - Périphérique périphérique
    - Mémoire et mémoire
- sans l'intervention du processeur
- Permet au processeur de réaliser d'autres actions pendant le transfert.

# LES SIGNAUX D'HORLOGE.

- Le signal d'horloge permet de cadencer le fonctionnement du microcontrôleur.
- Ce dernier intègre généralement une porte Trigger de Schmitt afin de réaliser un oscillateur. Pour l'obtenir on place un quartz entre les deux broches “OscIn” et “OscOut” comme l'indique le schéma suivant :





### **3. DÉVELOPPEMENT DU PROGRAMME ET MISE AU POINT.**

# Développement de code

## Programs May Be Written Two Ways

### Assembly Language

Code words standing for individual instructions in the set

```
LDAA #x40  
DEC A  
BNE 1:
```

### High-Level Language (C/C++)

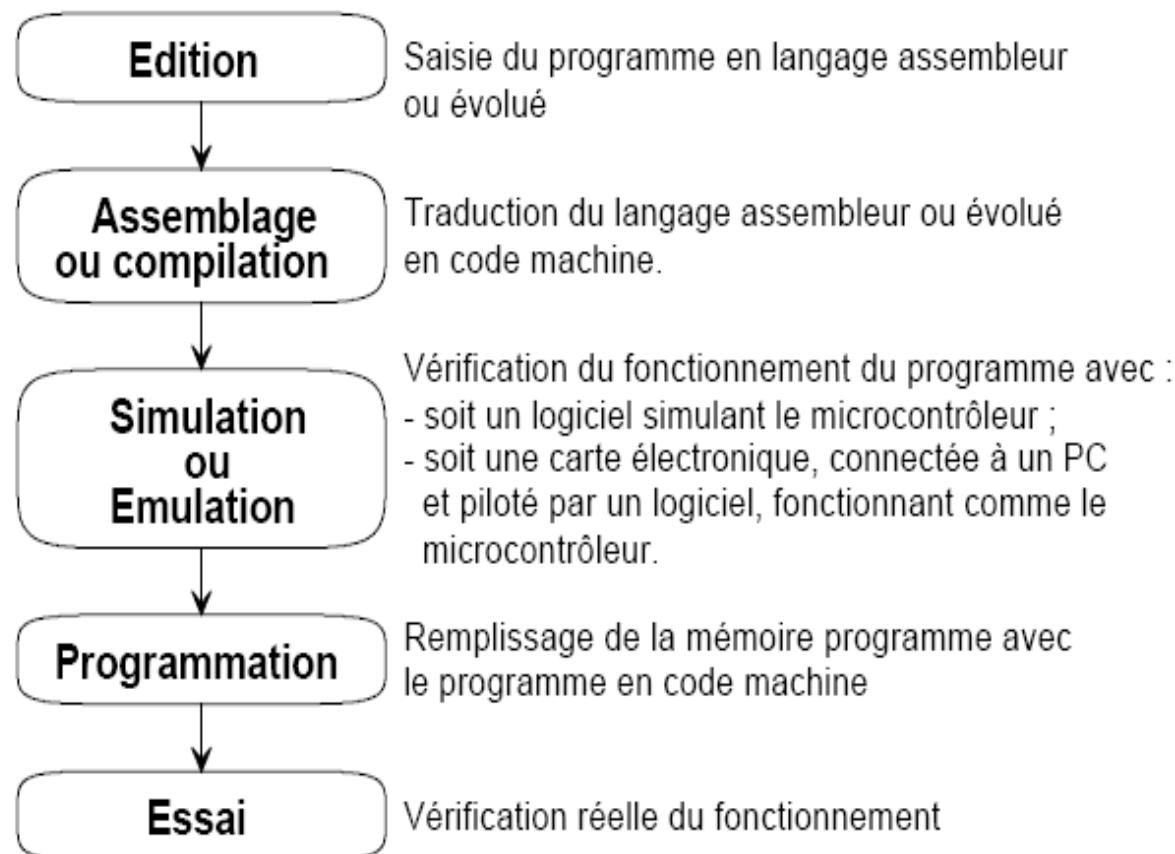
English-like words that correspond to a series of instructions

```
for(int i = 0; i < 10; i++)  
{  
    cout << "Count: " << i << endl;  
}
```

## Ou un mélange

# Développement et mise au point

Que ce soit dans un langage assembleur ou évolué, l'écriture du programme ainsi que sa mise au point doivent suivre le diagramme suivant:



# Simulation ou émulation

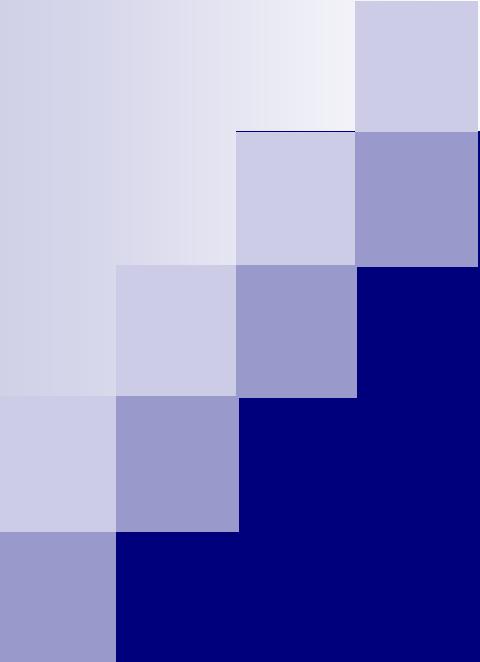
**L'émulateur** : est un dispositif (assez onéreux) qui permet de remplacer le microcontrôleur afin d'effectuer la vérification et la mise au point du programme.

- C'est une carte électronique comportant, d'un coté un connecteur compatible avec le boîtier du microcontrôleur qu'il émule, et de l'autre, une connexion de type RS232 (ou autre) reliée à micro-ordinateur.
- La mise au point peut alors se faire en pas à pas ou avec des points d'arrêt permettant ainsi de cerner très rapidement les "bugs" de certaines routines.

**Le simulateur** : beaucoup moins coûteux, permet la même chose mais de façon beaucoup moins efficace car il ne tient pas ou très peu compte de l'environnement du microcontrôleur

# Exemple de microcontrôleur

RÉFÉRENCE	FABRICANT	VITESSE	RAM	ROM / EPROM / FLASH	EEPROM	E / S Logiques	TIMER	ENTRÉES ANALOGIQUES	Particularité
8051	Intel	12 Mhz	128 o	4 Ko	X	32	2	0	
16C71	Microchip	20 Mhz	36 o	1Kx14	X	13	1	4	RISC
6805 S2	Motorola	4 MHz	64	1 Ko	X	16	2	8	
68HC11 A1	Motorola	8 MHz	256 o	X	512	22	1	8	Etendu
AT90S 8515	Atmel	20 MHz	512 o	4 Ko	512	32	3	8	RISC
ST 6265	Thomson	8 MHz	128 o	4 Ko	64 o	21	2	13	



## 4. Exemple de la famille PIC de **MICROCHIP**

# Architecture

- Les PIC se conforment à l'architecture **Harvard** :
  - La plupart des instructions occupent un mot de la mémoire de programme. La taille de ces mots dépend du modèle de PIC,
  - la mémoire de données est organisée en octets.
- Les PIC sont des processeurs **RISC**,
- Un cycle d'instruction d'un PIC dure 4 temps d'horloge. La plupart des instructions durent un cycle, sauf les sauts qui durent deux cycles. On atteint donc des vitesses élevées.

Avec un quartz de 4 MHz (ou l'horloge interne), on obtient donc 1 000 000 de cycles/seconde, or, comme le PIC exécute pratiquement 1 instruction par cycle, hormis les sauts, cela donne une puissance de l'ordre de **1 MIPS (1 million d'instructions par seconde)**.

- Les PIC peuvent être cadencés à 20 MHz (série PIC16), 40 MHz (série PIC18), voire 48 MHz (exemple : PIC18F2550 — PIC avec USB) et 64 MHz (exemple : PIC18F25K20 — PIC en 3,3 V).

# Les familles du PIC

Les modèles de PIC courants sont repérés par une référence de la forme :

- 2 chiffres : famille du PIC (10, 12, 16, 17 et 18).
- 1 lettre : type de mémoire de programme (C ou F).
  - Le **F** : indique en général qu'il s'agit d'une mémoire flash et donc effaçable électroniquement.
  - Le **C** : indique en général que la mémoire ne peut être effacée que par exposition aux ultra-violets (exception pour le PIC16C84 qui utilise une mémoire EEPROM donc effaçable électriquement).
- Un **L** peut être ajouté devant pour indiquer qu'il s'agit d'un modèle basse tension (exemple : 2 V à 5,5 V si LF — 4,2 V à 5,5 V si F).
- un groupe de lettres pour indiquer le boîtier et la gamme de température.
- *Par exemple, le PIC18LF4682-I/P : est un microcontrôleur de la famille PIC18, basse tension (L), à mémoire flash (F), modèle 4682, gamme de température industrielle (I) et boîtier DIL40.*

# Les familles du PIC

## ■ **PIC16:** Composants de milieu de gamme.

- C'est la famille la plus fournie.
- Jeu d'instruction de 35 instructions.
- Les PIC 16 peuvent être cadencés à 20 MHz

## ■ **PIC18 :**

- Cette famille a un jeu d'instruction plus complet puisqu'il comprend de l'ordre de 75 instructions. Cette palette d'instructions étendue lui permet de faire fonctionner du code C compilé de manière nettement plus efficace que les familles précédentes.
- On peut les utiliser avec un quartz oscillant jusqu'à 64 MHz.

# PIC16F84

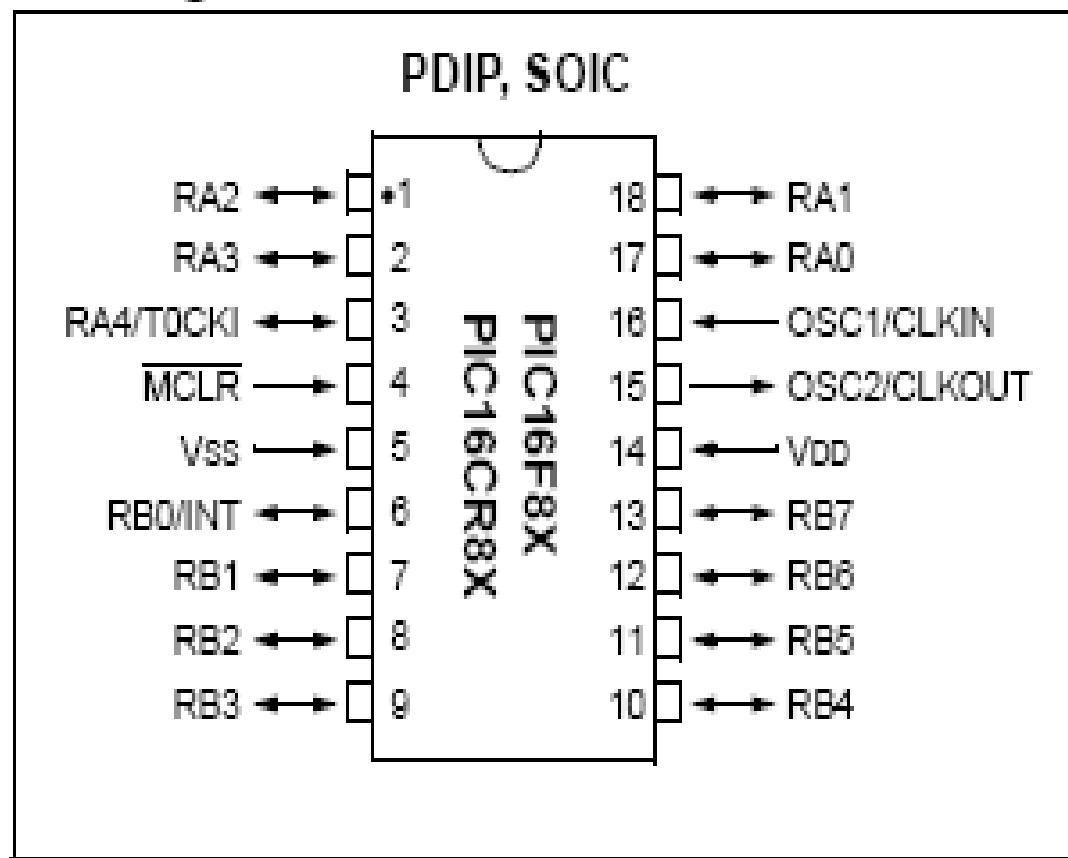
Ses principales caractéristiques:

- microcontrôleurs **8 bits**
- seulement **35 mots d'instruction**
- instruction sur 14 bits
- vitesse d'horloge (existe en version 4, 10 ou 20 Mhz)
- 4 sources d'interruption (?)
- 1000 cycles effacement/écriture possible de la mémoire programme flash
- 1K mots mémoire programme Flash
- 68 octets de données RAM
- 64 octets de données EEPROM
- 13 pins I/O avec contrôle individuel de direction
- TMR0 8bit timer/compteur

Le tout sur un pavé de 18 pins.

# PIC16F84

## Pin Diagrams



# Programmation

- Les PIC disposent de plusieurs technologies de mémoire de programme : **ROM, EPROM, EEPROM, UV PROM, flash**. Certains PIC sont dépourvus de mémoire programme interne, c'est le cas des PIC18C601 et PIC18C801 où le programme doit être contenu dans une mémoire externe.

La programmation du PIC peut se faire de différentes façons :

1. Par l'intermédiaire d'un programmateur dédié (par exemple : PRO MATE ou PICSTART Plus de la société [Microchip](#))
2. Par programmation *in-situ*. Il suffit alors de câbler correctement le microcontrôleur sur la carte fille pour qu'une simple liaison série suffise. Il existe pour cela plusieurs solutions libres (logiciel + interface à faire soi-même) ou commerciales (par exemple : PICkit 2 ou ICD2 de [Microchip](#)).

# MPLAB - Plate forme de développement

Pour écrire un programme PIC il faut :

- un éditeur de texte
- un assembleur
- un compilateur
- un simulateur pour tester le programme sur le micro

Tout cela est mis gracieusement à disposition par Microchip de façon **libre**. Cela s'appelle **MPLAB**

# Le programmateur

- Une fois le programme compilé, il faut le transférer dans la mémoire du microcontrôleur. Pour cela il vous faut :
- **Une petite interface matérielle**
- **Un logiciel:** qui assure le transfert des données entre le PC et le microcontrôleur (il est fourni avec le programmateur). Comme exemple **ICPROG** , c'est le meilleur logiciel de programmation de PIC à l'heure actuelle, et il a l'avantage d'être disponible en **freeware**.

# Exemple d'applications: commande d'une Led

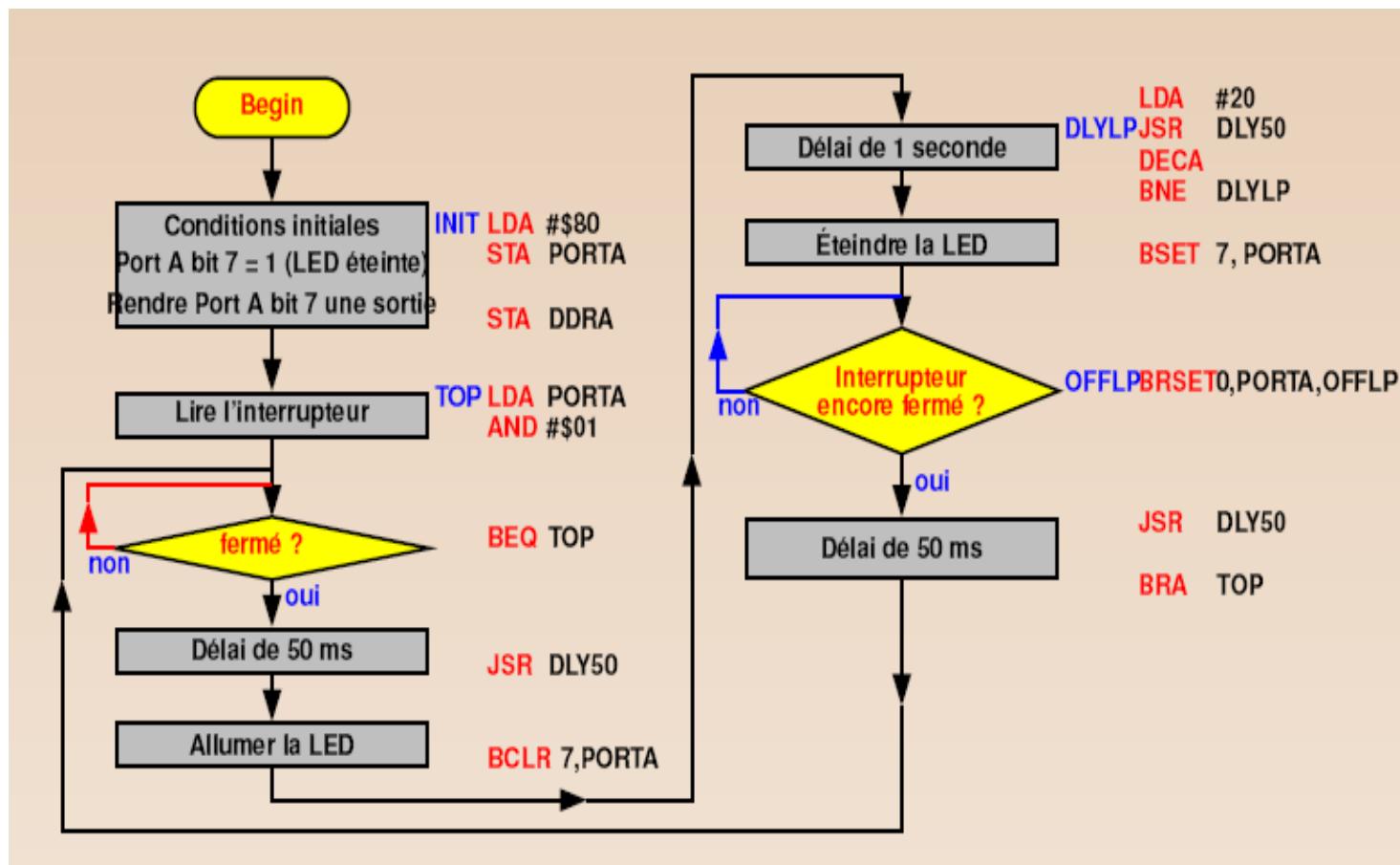
Ecrire un programme qui lit l'état d'un interrupteur branché à une broche d'entrée d'un microcontrôleur.

Lorsque l'interrupteur est fermé, le programme doit allumer une LED branchée à une broche à la sortie pour environ 1 seconde et ensuite s'éteindre. La LED ne se réallumera pas avant que l'interrupteur ne s'ouvre et se ferme une autre fois.

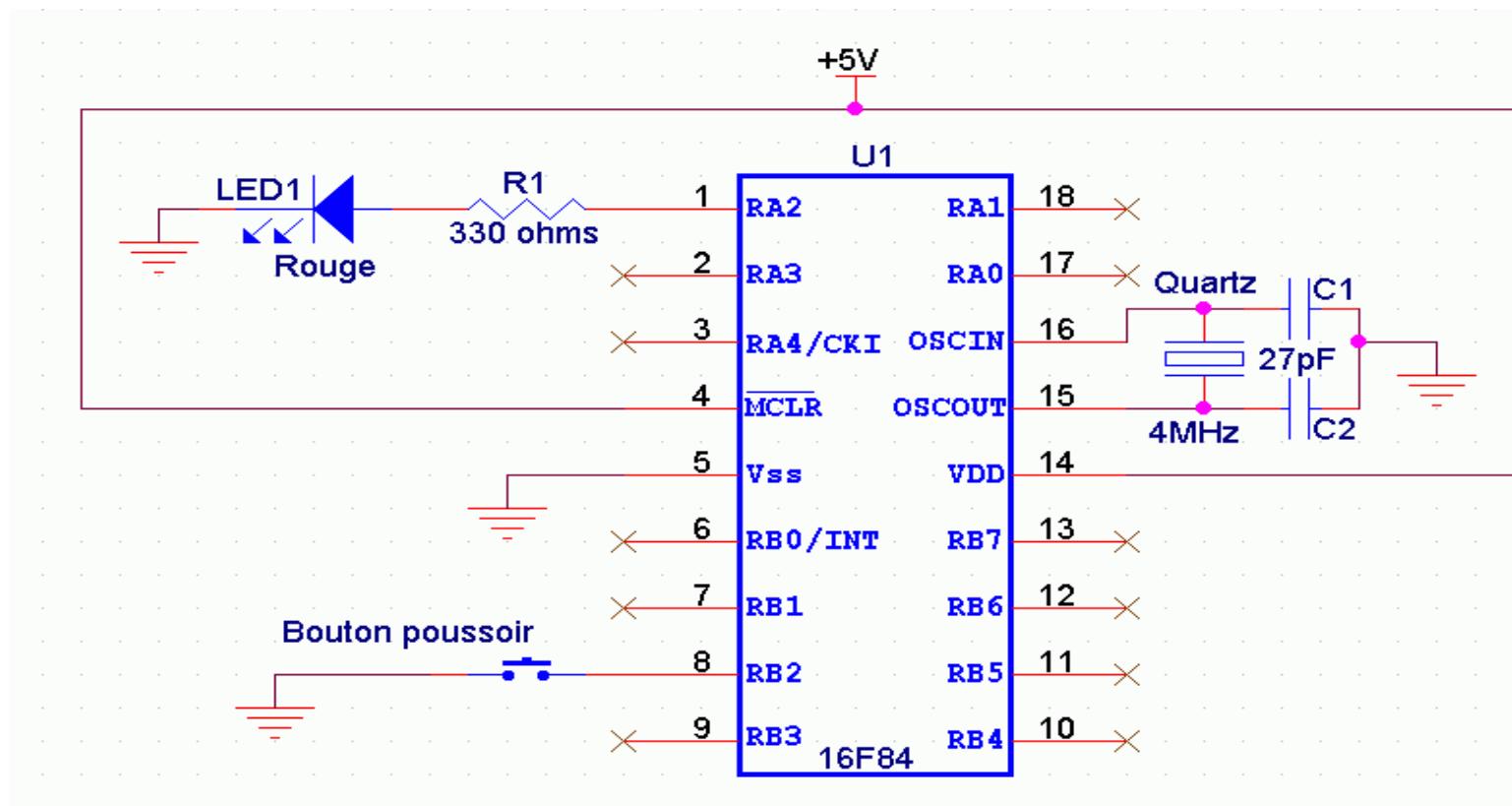
## Objectifs du problème

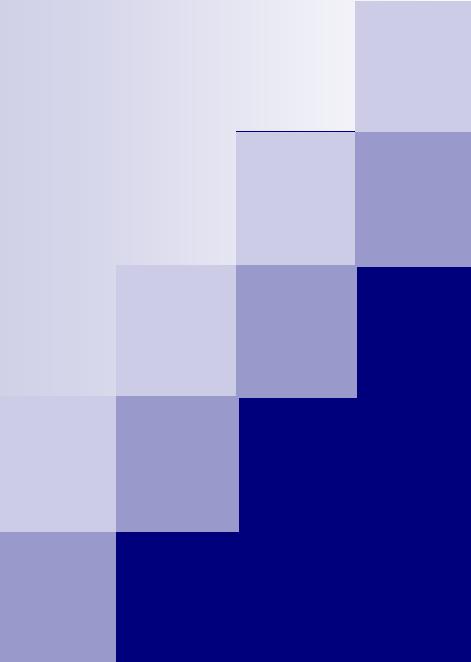
- montrer comment un programme peut détecter des signaux d'entrée comme l'ouverture et la fermeture d'un interrupteur
- c'est un exemple de programme pour contrôler un signal de sortie
- le fait que la LED reste allumée pendant 1 seconde montre comment un programme peut être utilisé pour contrôler la mesure en temps réel

# Commande d'une LED



# Exemple d'applications: clignotement de Led





# 4. Le microcontrôleur STM 32

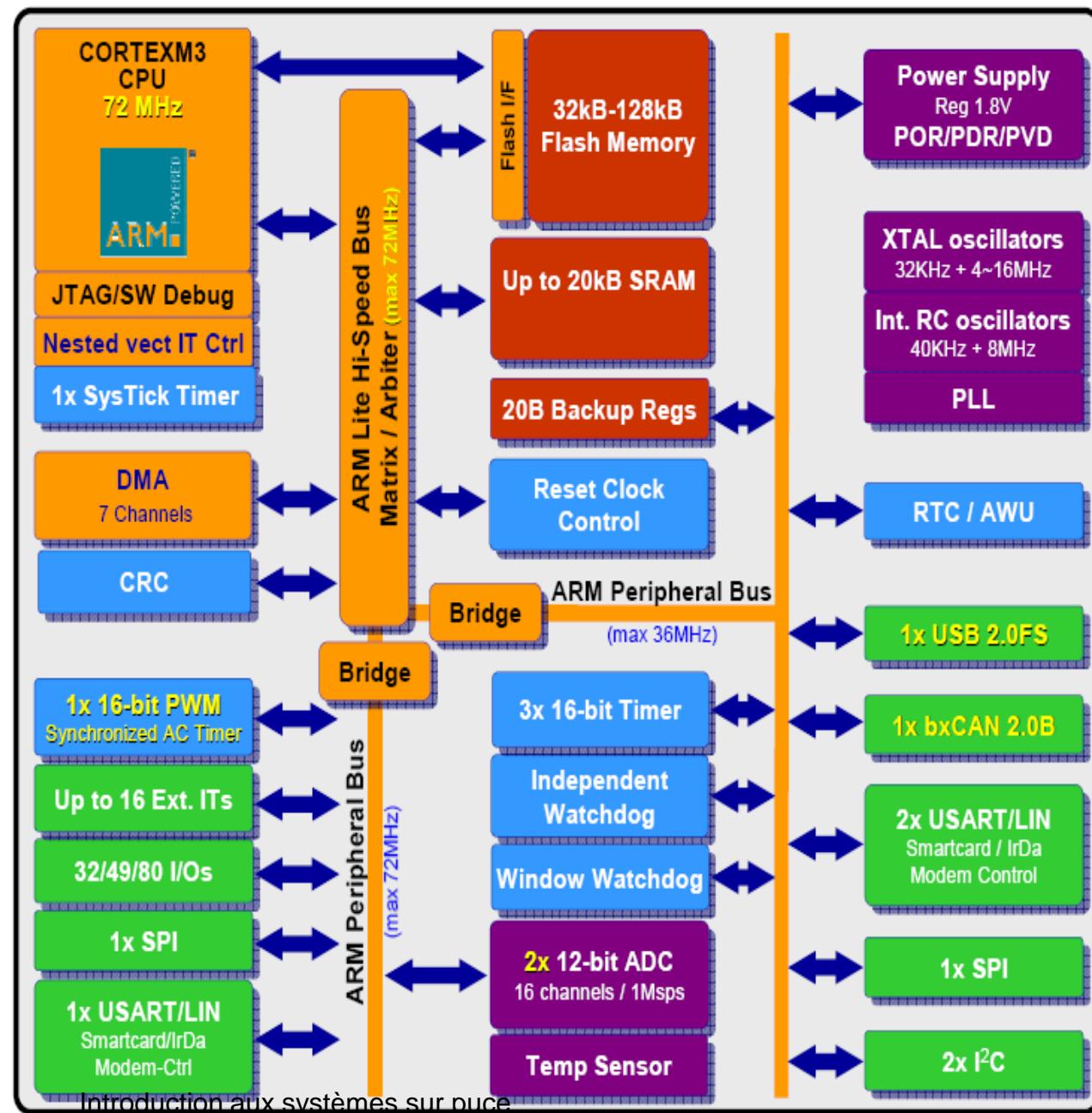
Sources:

STEcole\_training.pdf

RM0008 Reference manual.pdf

Sur le site : [www.st.com](http://www.st.com)

# Architecture interne

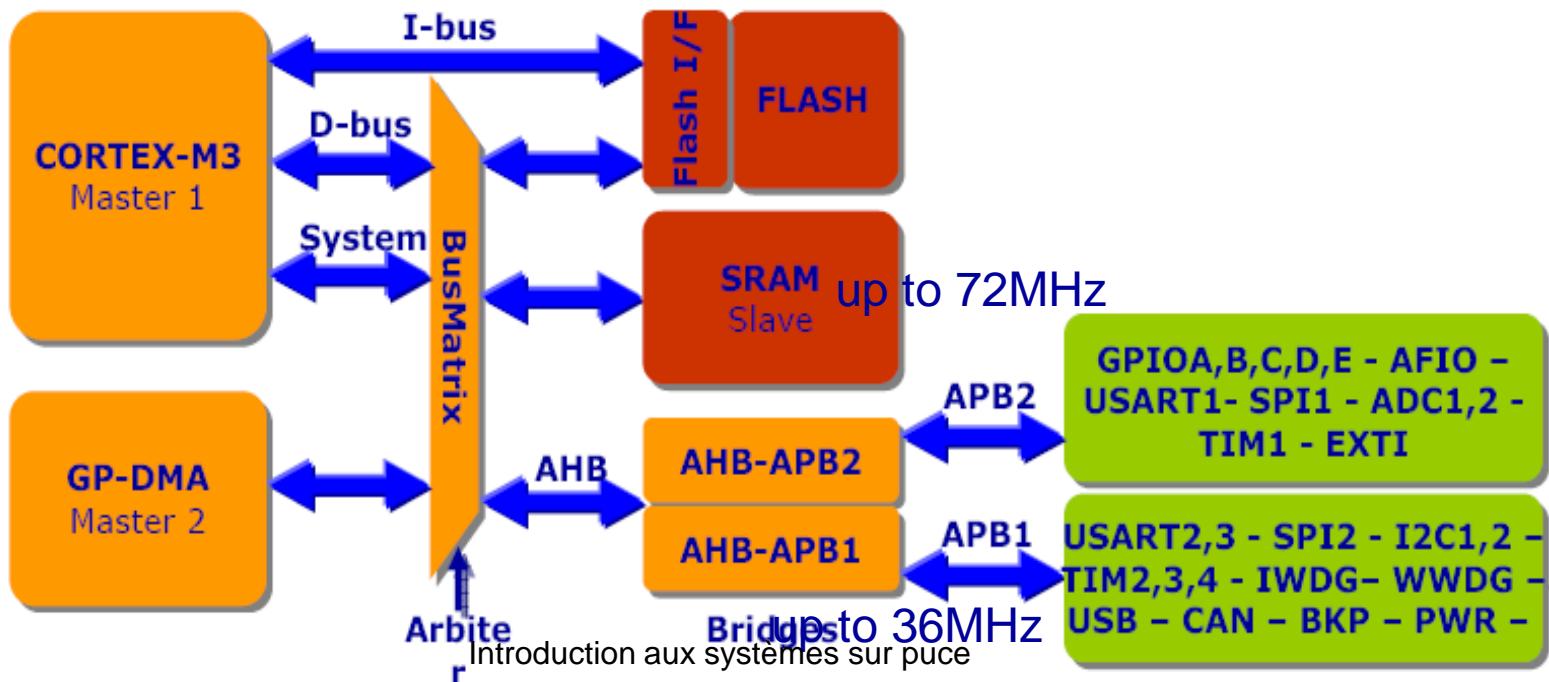


# Architecture interne

- ARM 32-bit Cortex-M3 CPU
- Embedded Memories :
  - FLASH: up 128 Kbytes
  - SRAM: up 20 Kbytes
- CRC calculation unit
- 7 Channels DMA
- Power Supply with internal regulator and low power modes :
  - 2V to 3.6V supply
  - 4 Low Power Modes with Auto Wake-up
- Up to 72 MHz frequency managed & monitored by the Clock Control
- Rich set of peripherals & IOs
  - Embedded low power RTC with  $V_{BAT}$  capability
  - Dual Watchdog Architecture
  - 5 Timers w/ advanced control features(including Cortex SysTick)
  - 9 communications Interfaces
  - Up to 80 I/Os (100 pin package) w/ 16 external interrupts/event
  - Up to 2x12-bits 1Msps ADC w/ up to 16 channels and Embedded temperature sensor w/ +/-1.5°linearity with  $T^\circ$

# System architecture

- **I-Bus** (Instruction Bus): dédié principalement pour le transfert des instructions (code) entre Cortex et la mémoire Flash.
- **D-Bus & System Bus**: connectés directement à la matrice de bus qui sert de lien avec les bus d'accès à la SRAM, le système DMA (Direct Memory Access) ainsi que le **AHB** (Advanced High Bus) .
- **APB1 & 2** (Advanced Peripheral Bus 1 & 2): Chacun des deux bus est connecté au AHB travers un pont (respectivement AHB-APB1 et AHB-APB2) et servent à véhiculer les données vers et depuis une partie des périphériques.



# Processeur CortexM3

- Un processeur 32 bits: Il contient un chemin de données (data path) 32 bits, un banc de registres 32 bits et des interfaces mémoires de 32 bits.
- Il se base sur un pipeline à 3 étages (Fetch, Decode et Execute)
- possède une architecture Harvard, c'est-à-dire qu'il contient deux bus distincts pour le transfert des données et des instructions. Ceci permet d'accéder en même temps aux instructions et aux données, et par conséquent une augmentation de la performance puisque l'accès aux données n'affecte pas le pipeline.
- l'espace d'adressage du Cortex-M3 est unique et peut atteindre la taille de 4 GO

0xFFFFFFFF

*Zone Constructeur*

Firmware : code nécessaire agissant à très bas niveau (Driver) et permettant aux applications d'utiliser le matériel d'une façon transparente

0xE0100000

Espace utilisé pour adresser les périphériques du Cortex-M3 : Contrôleur d'interruption, le Timer système, le bloc de contrôle, etc.

0xE00FFFFF

*Périphériques système  
(1 MO)*

0xE0000000

*Périphériques  
Externes (1 GO)*

0xA0000000

*RAM  
Externe (1 GO)*

0x9FFFFFFF

*Périphériques  
(0,5 GO)*

0x60000000

*SRAM  
(0,5 GO)*

0x5FFFFFFF

Espace utilisé pour adresser les périphériques ajoutés par le constructeur du microcontrôleur intégrant le Cortex-M3.

0x40000000

Espace utilisé pour stocker les données relatives au code.

0x3FFFFFFF

*CODE  
(0,5 GO)*

0x20000000

Espace utilisé pour stocker du code exécutable. On peut également stocker des données.

0x1FFFFFFF

# Gestion des interruptions (NVIC)

- Le NVIC est capable de gérer jusqu'à 240 interruptions générées par des périphériques externes au Cortex-M3 et dont le niveau de priorité peut être dynamiquement fixé parmi 256 niveaux possibles.
- En plus de ces interruptions, le NVIC gère un certain nombre d'exceptions déclenchées par des périphériques internes du Cortex-M3 (Bus AHB, MPU, Timer SysTick, etc..) ou par des fautes au niveau du programme même.

Nr	Type d'exception	Niveau de priorité par défaut	Description
1	Reset	-3 (maximum, figé)	Déclenchée quand un signal de reset est détecté.
2	NMI	-2 (figé)	Il s'agit de l'entrée Non Masquable Interrupt qui est généralement connectée au WatchDog.
3	Hard Fault	-1 (figé)	Erreur de matériel pour lequel aucune routine n'a été prévue
4	MemManage Fault	0 (paramétrable)	Déclenchée en cas d'une tentative d'accès à une zone mémoire non autorisée.
5	Bus Fault	1 (paramétrable)	Générée quand l'interface du bus AHB reçoit une erreur.
6	Usage Fault	2 (paramétrable)	Déclenchée par une erreur au niveau d'un programme (exple : division par 0).
7-10	Réservé		
11	SVCall	3 (paramétrable)	Dans le cas d'un appel à un service système.
12	Debug Monitor	4 (paramétrable)	Déclenchée quand un point de break (ou watch) est atteint ou bien quand le débogage est activé.
13	Réservé		
14	PendSV	5 (paramétrable)	
15	SYSTICK	6 (paramétrable)	Relative au Timer interne SYSTICK
16...	Interrupt # 0	7 (paramétrable)	Il s'agit d'événements déclenchés par les autres périphériques externes ajoutés par le fabricant du microcontrôleur
.....	.....	.....	
256	Interrupt # 240	247 (paramétrable)	

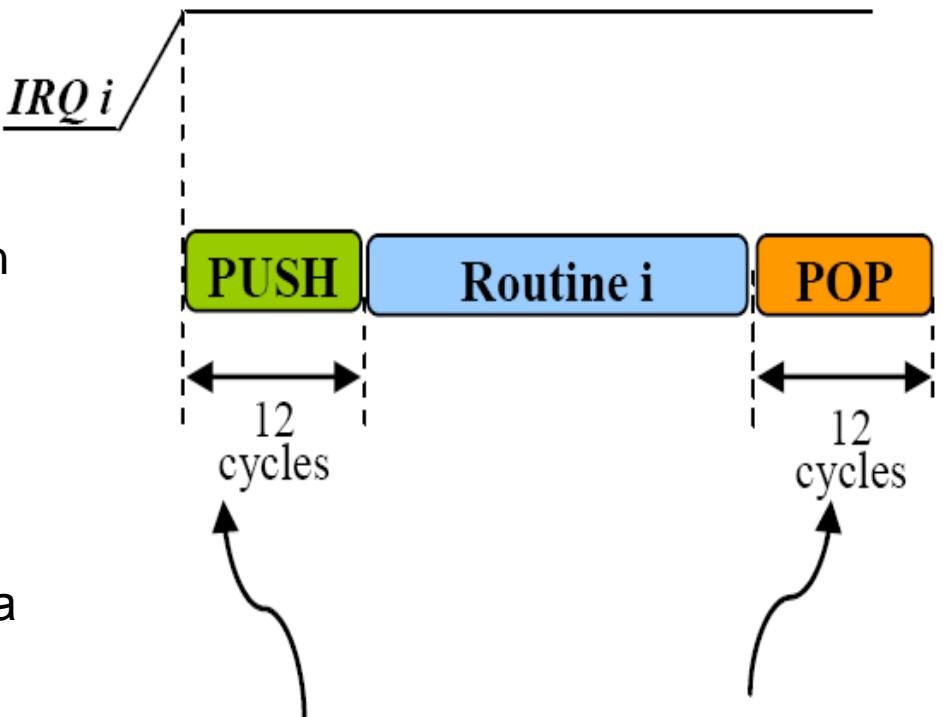
# Gestion des interruptions (NVIC)

- Quand une exception autorisée est déclenchée, le CPU commence par récupérer l'adresse de la routine à exécuter à partir d'un emplacement mémoire spécifique connu d'avance
- L'ensemble des adresses des routines relatives aux différentes sources d'interruptions forme **la table de vecteurs d'interruptions.**
- Dans le Cortex-M3, la table de vecteurs commence à partir de l'adresse 0 de la zone code :

Adresse	Vecteur
0x00	Adresse de début de la pile principale
0x04	Reset
0x08	NMI
0x0C	Hard fault
0x10	MemManage Fault
0x14	Bus Fault
0x18	Usage Fault
0x1C – 0x2B	
0x2C	SVCall
0x30	Debug Monitor
0x34	Réservé
0x38	PendSV
0x3C	SYSTICK
0x40	IRQ0
.....	.....

# Déroulement d'une interruption

- Quand une interruption autorisée est déclenchée par un périphérique, le NVIC force le processeur à la servir.
- Il commence par sauvegarder un certain nombre de registres dans la pile, et parallèlement, il récupère l'adresse de la routine relative à l'interruption.
- Le temps total qui s'écoule du moment où l'interruption se déclenche jusqu'à l'exécution de la première instruction de la routine est de seulement 12 cycles.
- Une fois l'exécution de la routine est terminée, les contenus des registres sauvegardés sont récupérés à partir de la pile ainsi que l'adresse de l'instruction suivante du programme principal.



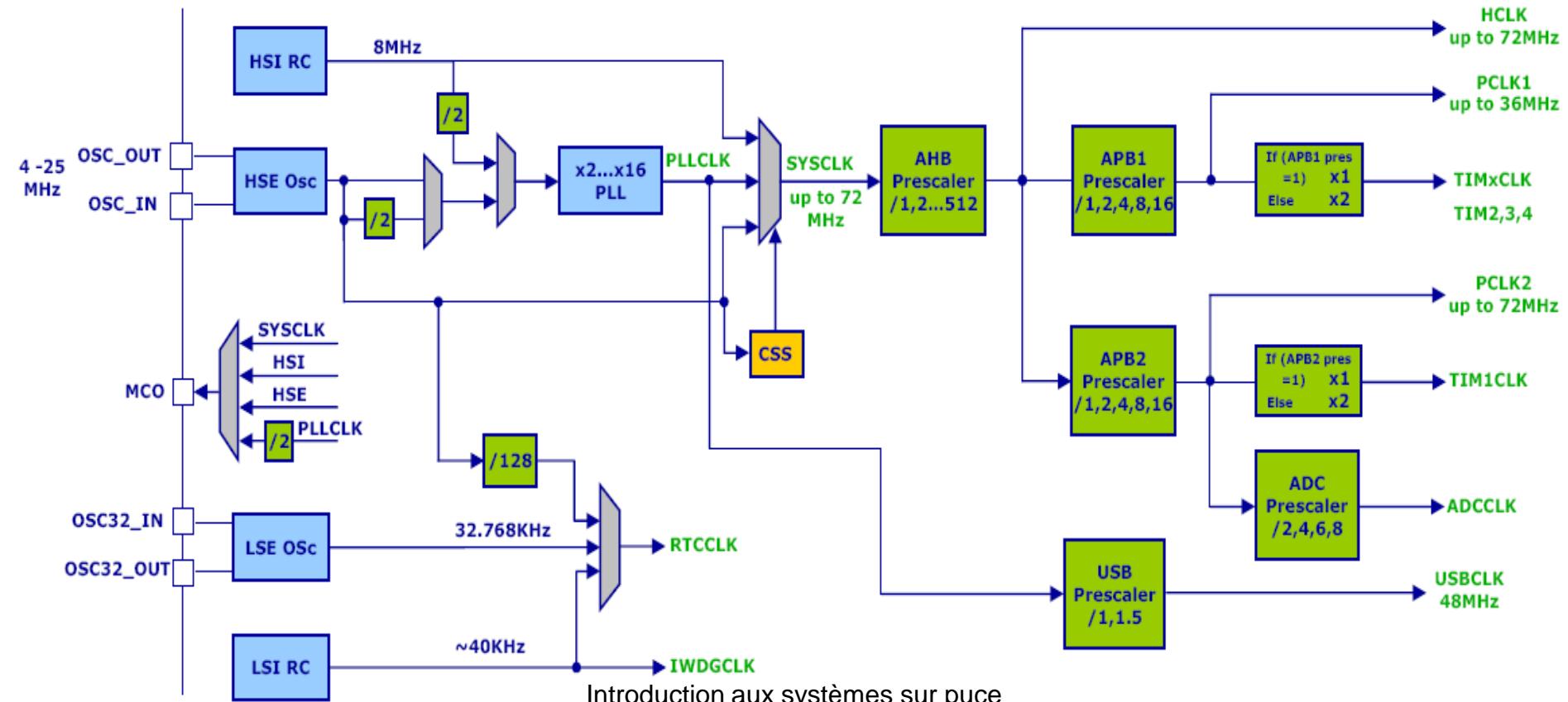
R0-R3 ; R12  
R15 (Program Counter)  
R14 (Link register)  
PSR (Registre d'état)

# Arbre d'horloge du STM32

- Les différents bus, composants et périphériques fonctionnent avec des fréquences d'horloge synthétisées à partir de l'un des deux signaux horloge :
- **HSI (High Speed Internal)** : Il s'agit d'un signal de fréquence 8 Mhz généré par un oscillateur interne intégré au niveau du µc. C'est cette fréquence qui est utilisée par défaut après une opération de reset. Elle présente l'inconvénient de manque de précision et de stabilité.
- **HSE (High Speed External)** : Il s'agit d'un signal de fréquence comprise entre 4 et 25 Mhz, généré par un dispositif (à base d'un cristal) placé à l'extérieur du µc.

# Arbre d'horloge du STM32

- La synthèse des différentes fréquences à partir de l'un des deux signaux :



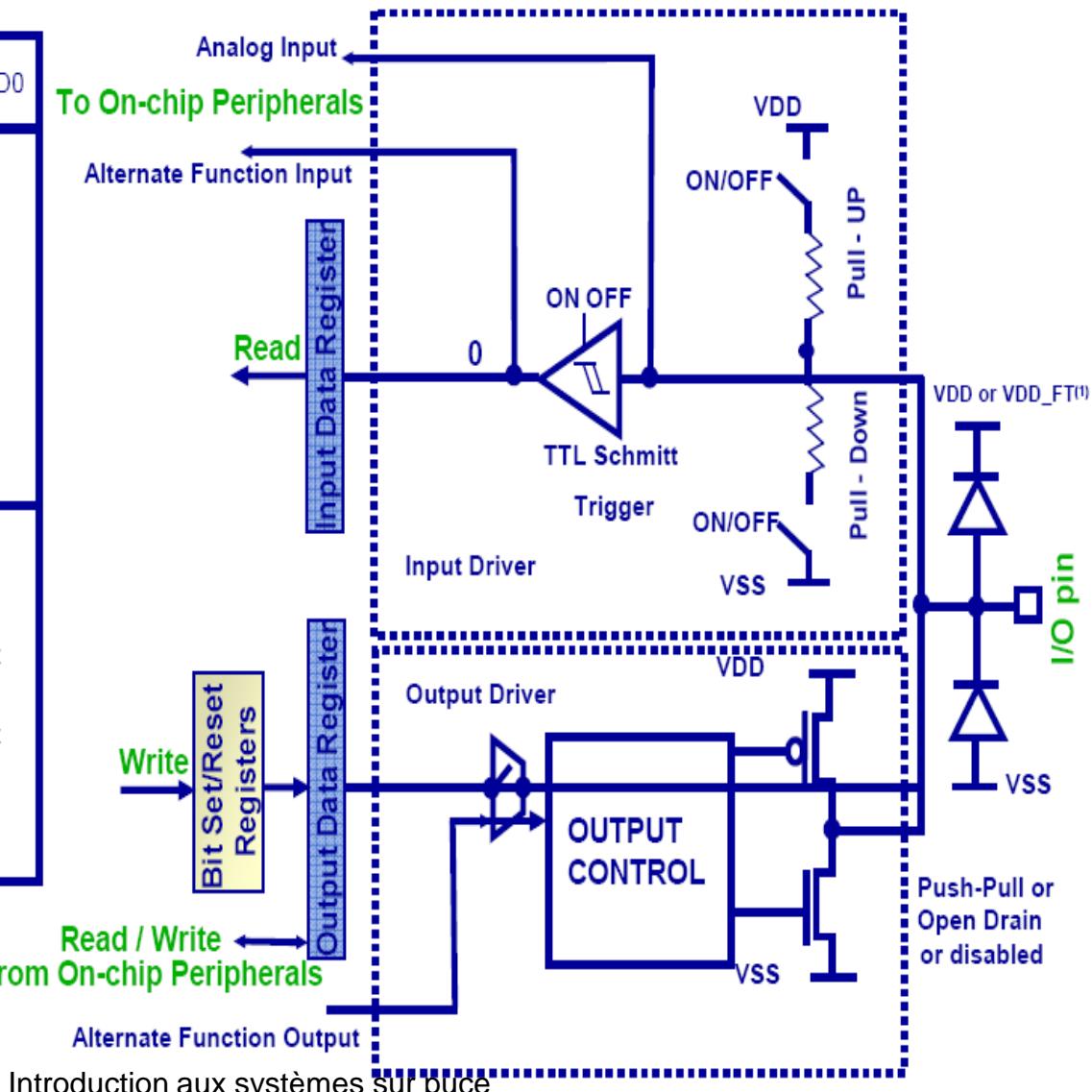
# General Purpose Input Output (GPIO)

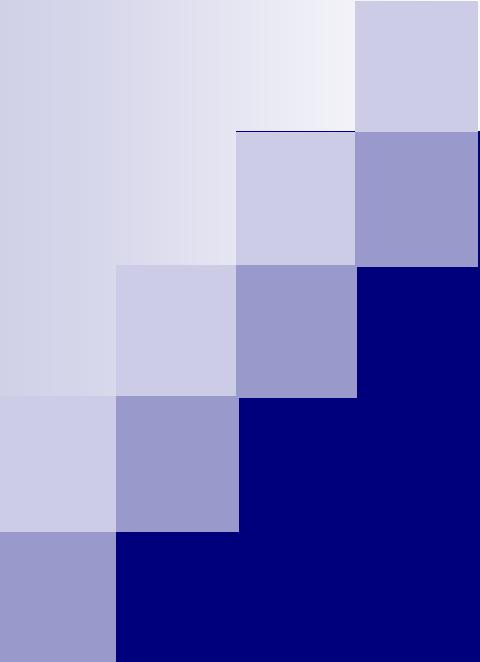
- Tous les I/O sont réparties sur 5 ports (GPIOA...GPIOE)
- Chaque general purpose I/O port possède:
  - 2 registres (32 bits) de configurations
  - 2 registres (32 bits) de données
  - 1 registre (32 bits) set/reset
  - 1 registre (16bits) reset
  - 1 registre (32bits) locking registre
- Chaque port bit d'un GPIO peut être configuré en un de ces mode (Input floating, Input pull-up, Input-pull-down, Analog, Output open-drain, Output push-pull, Alternate function push-pull, Alternate function open-drain)
- Alternate Functions pins (like USARTx, TIMx, I2Cx, SPIx, CAN, USB...)
- Configurable Output Speed up to 50 MHz

# GPIO

Configuration Mode	CNF1	CNF0	MOD1	MOD0	
Analog Input	0	0			
Input Floating (Reset State)	0	1			00
Input Pull-Up <sup>(2)</sup>	1	0			
Input Pull-Down <sup>(2)</sup>					
Output Push-Pull	0	0			01: 10 MHz 10: 2 MHz 11: 50 MHz
Output Open-Drain	0	1			
AF Push-Pull	1	0			
AF Open-Drain	1	1			

- (1) VDD for standard I/Os and VDD\_FT is a potential specific to five-volt tolerant I/Os and different from VDD.
- (2) Input Pull-Up and Input Pull-Down are differentiated by the PxODR.y bit field.



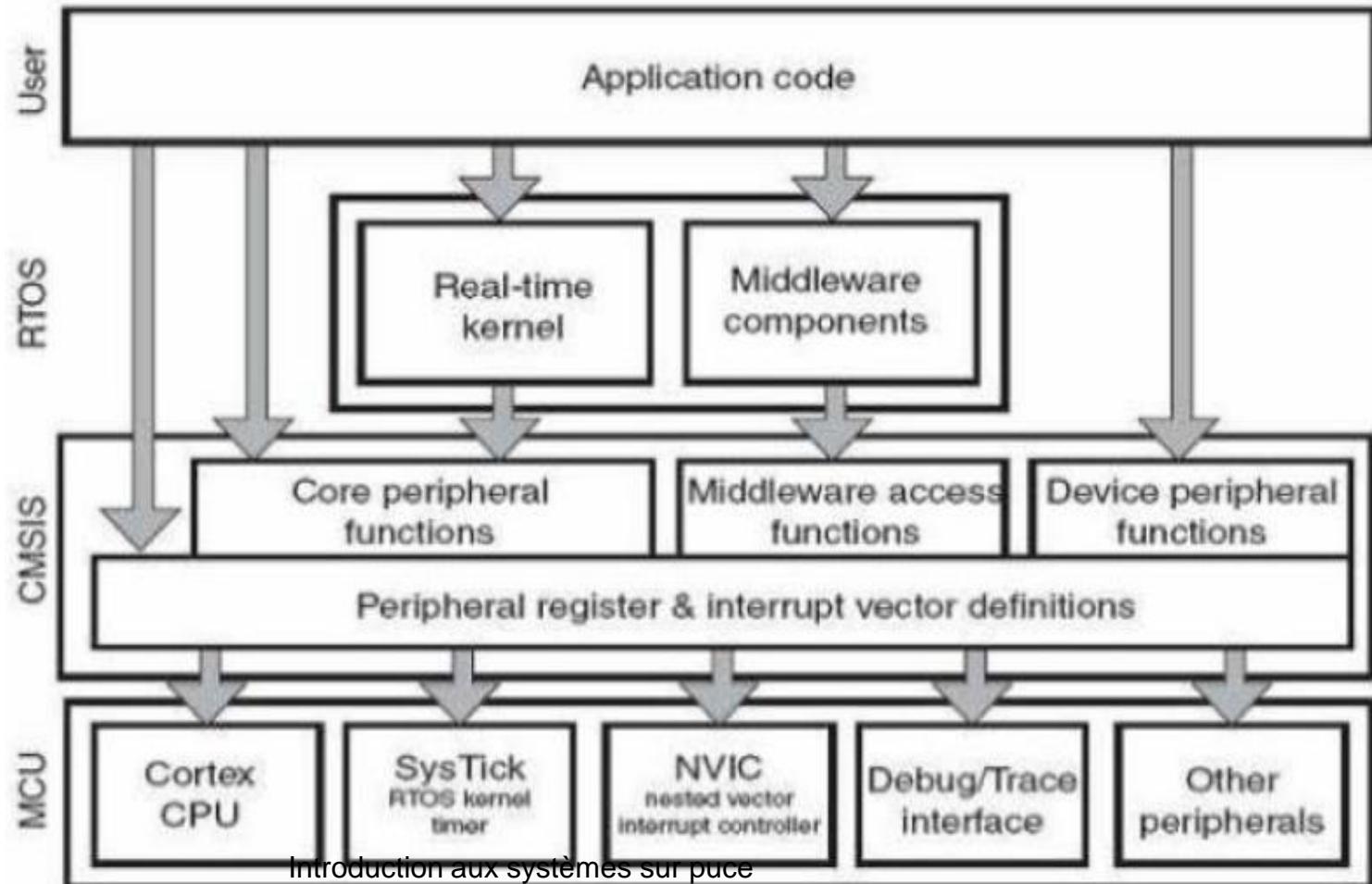


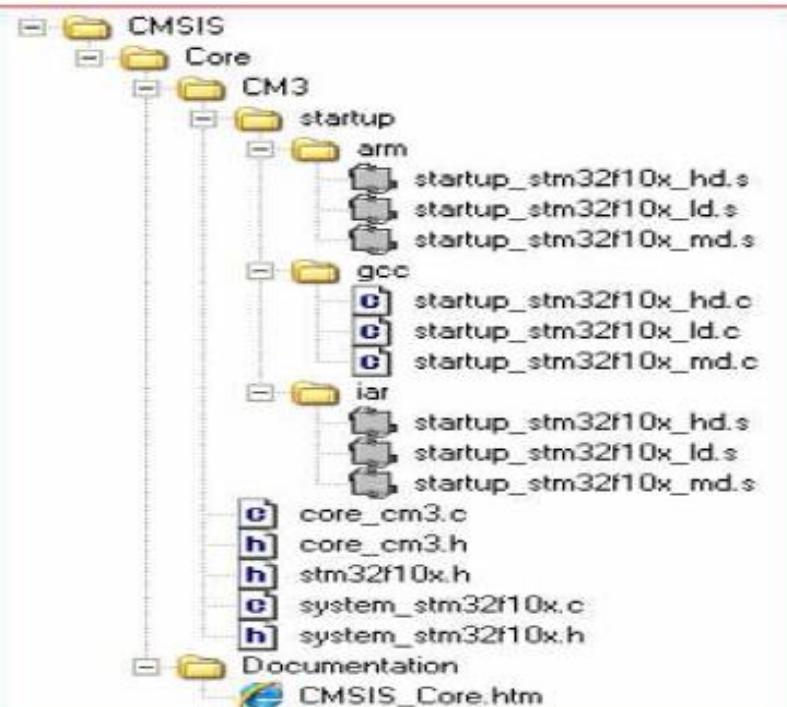
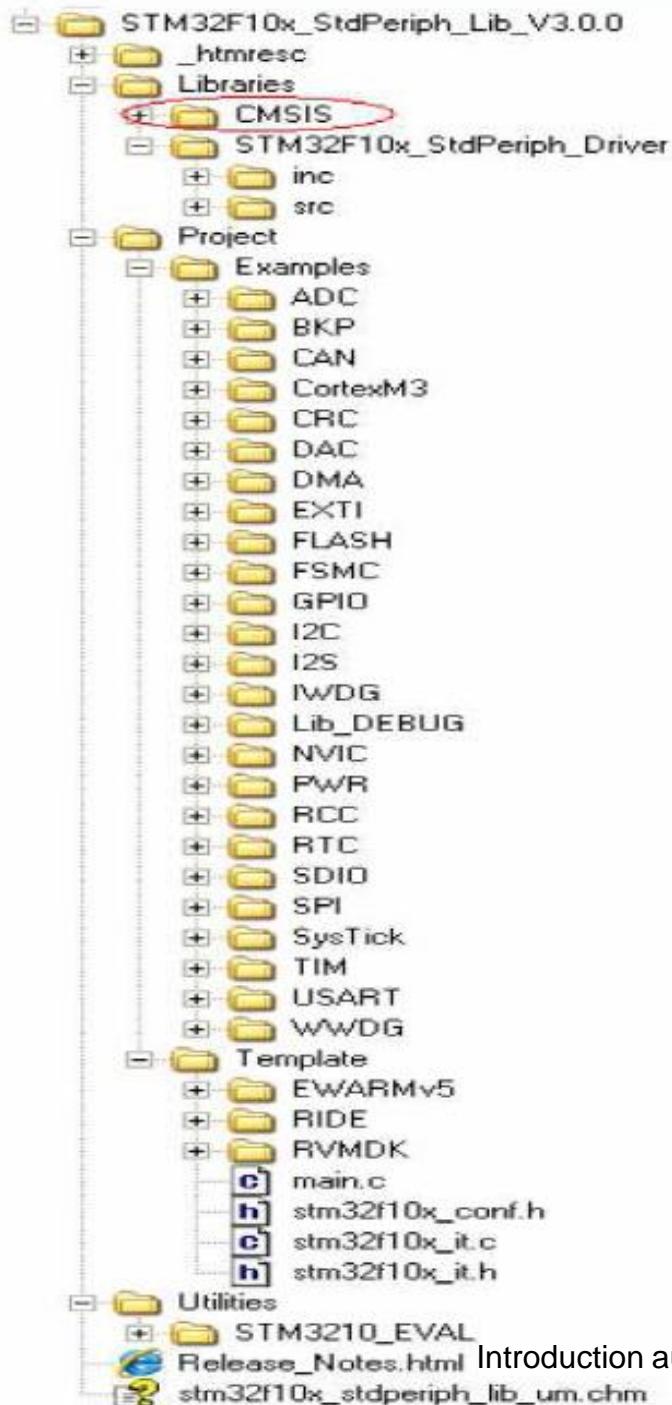
# Programmation du STM32

## 1. Le Standard de programmation en ‘C’ : CMSIS 1.0

# CMSIS : Cortex Microcontroller Software Interface Standard

- Il s'agit en effet d'un ensemble de modules (drivers) qui permettent aux applications d'accéder au matériel d'une façon transparente





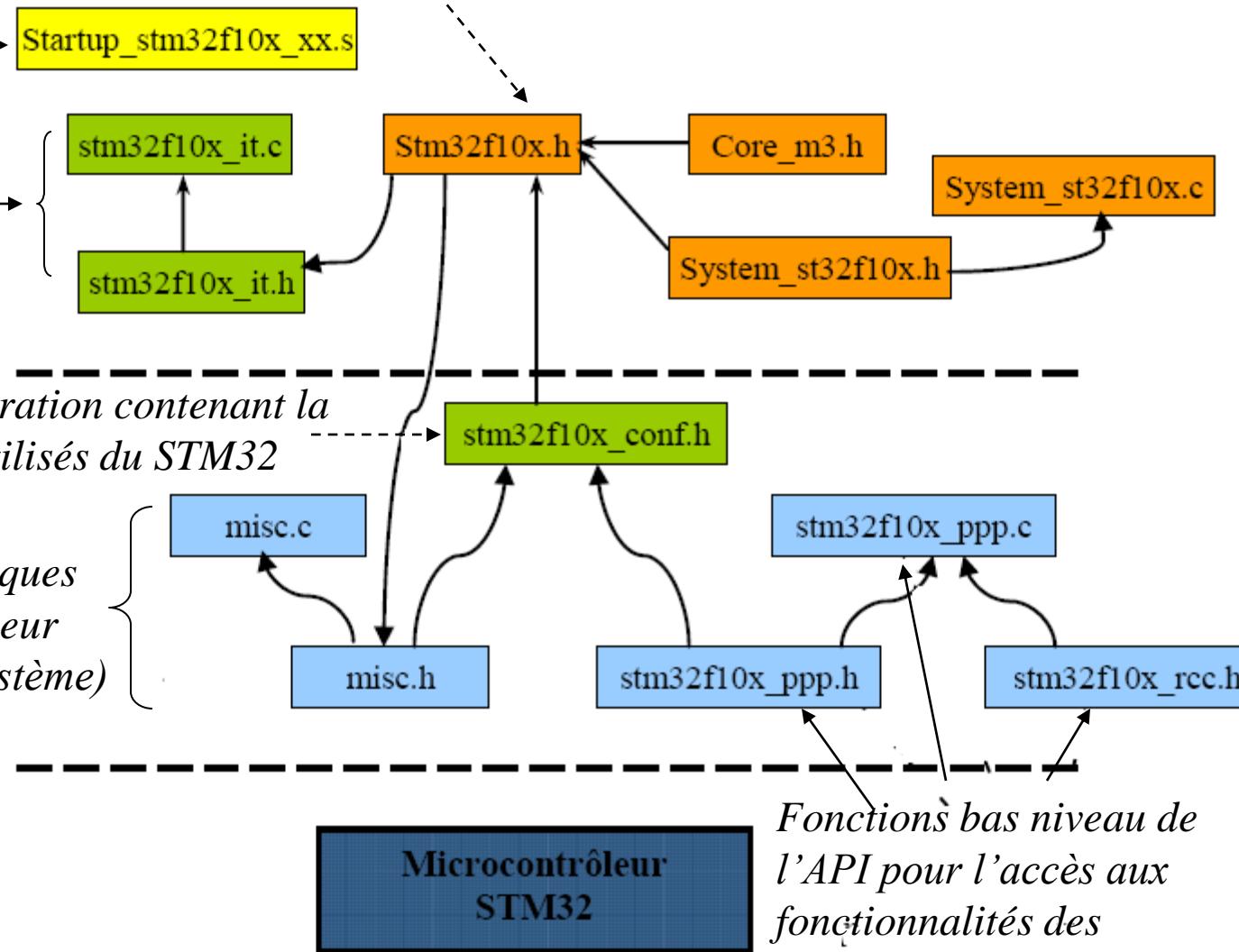
-Liste des interruptions du STM32 spécifiques au noyau Cortex-M3.

-Organisation de la mémoire STM32 et définition des adresses physiques des registres.

(optimisation de la configuration → l'initialisation des paramètres du STM32

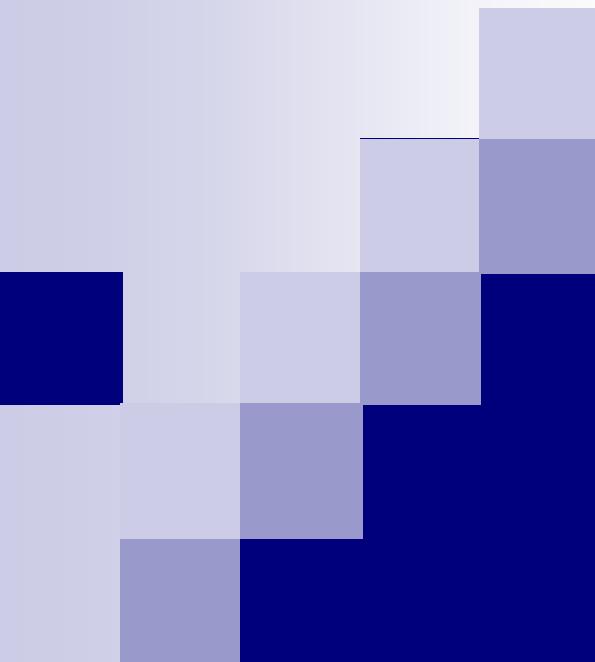
Routines relatives aux exceptions du Cortex-M3

L'application développée par l'utilisateur



- **stm32f10x\_ppp.c et .h** n'est pas uniquement utilisé pour désigner un seul fichier mais elle s'étend à un certain nombre de fichiers.
- Chaque fichier étant relatif à l'un des périphériques du microcontrôleur (ppp étant remplacée par l'abréviation du périphérique).

<i>Nom de fichier</i>	<i>Périphérique</i>
stm32f10x_adc	Convertisseur Analogique/Numérique
stm32f10x_bkp	Sauvegarde de données
stm32f10x_can	Bus CAN
stm32f10x_crc	Calcul des codes de détection d'erreurs
stm32f10x_dac	Convertisseur Analogique/Numérique
stm32f10x_dma	Accès direct à la mémoire
stm32f10x_exti	Interruptions externes
stm32f10x_gpio	Entrées/sorties parallèles
stm32f10x_iwdg	Chien de garde
stm32f10x_pwr	Gestion du mode d'alimentation
stm32f10x_rcc	Contrôle du reset et de la distribution du signal d'horloge
stm32f10x_rtc	Horloge temps réel
stm32f10x_tim	Les Timers (compteurs).
stm32f10x_usart	L'interface de communication série synchrone et asynchrone
misc	Fonctions haut niveau du contrôleur d'interruptions et du timer système (SysTick)



## 2. Développement d'application dans l'environnement Ride7

Ride7 is the RAISONANCE Development Environment.  
Integrated Designed for the development of STRx (STR7 and STR9 ARM families) and STM32 (Cortex-M3 family) projects from beginning to end.

Ride7 for ARM can be used with a range of development tools including the GNU toolchain, the software-based simulator, the RLink USB to JTAG dongle from RAISONANCE, and (for STRx only) the JTAGjet USB to JTAG dongle from Signum Systems.



