Embedded Systems In Health

GB 852 – M1 IBH

Mme S. LAZZOUNI

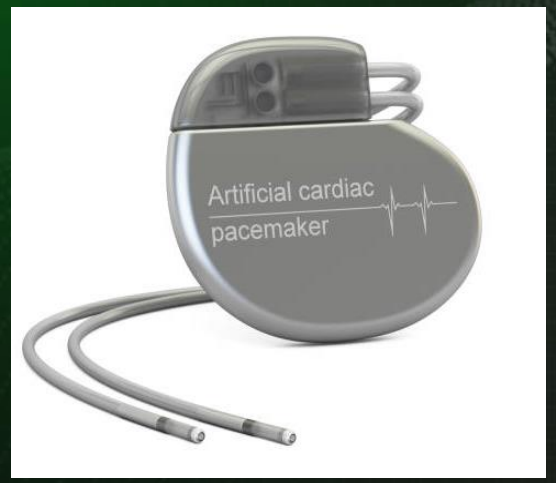
Plan

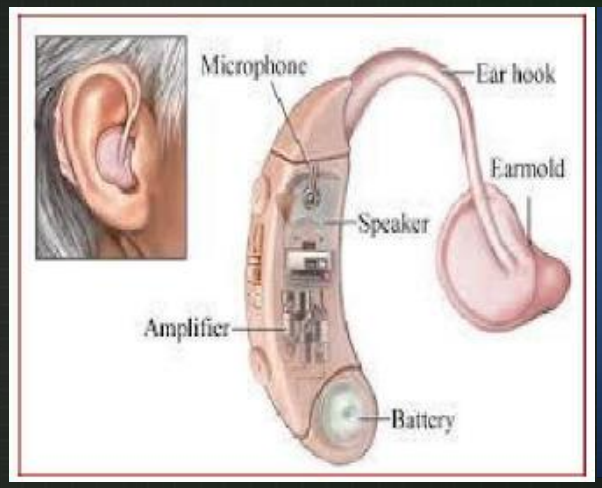
1. Définition et Architecture d'un système embarqué
2. Programmation d'un système embarqué (assembleur/ langage évolué C et Python)
3. Les différentes interfaces d'un système embarqué (GPIO, CAN, PWM, SCI, I2C, SPI, …
4. Driver ou Bibliothèque pour dialoguer avec un périphérique
5. Les systèmes embarqués temps réel et les RTOS
6. Les CPLD et FPGA dans un système embarqué

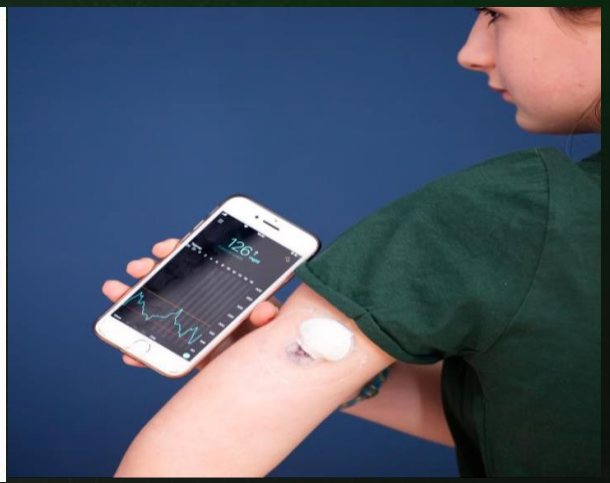
Chapter 1:

Definition and Architecture of an embedded system

Introduction

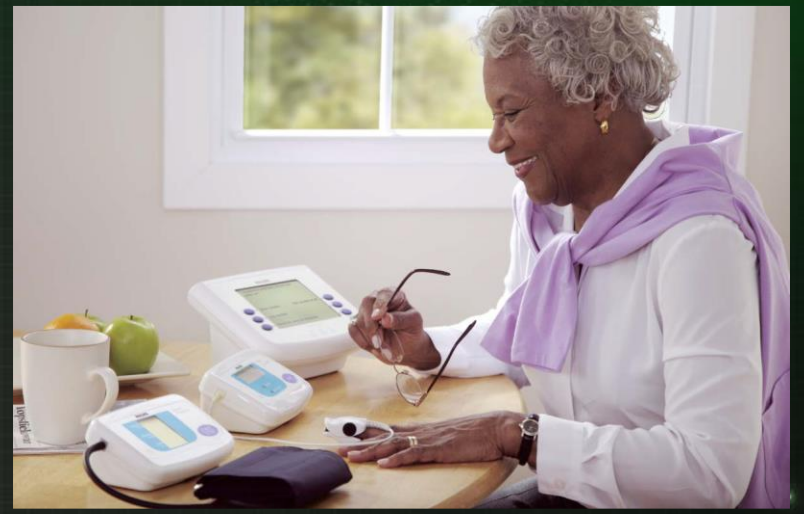


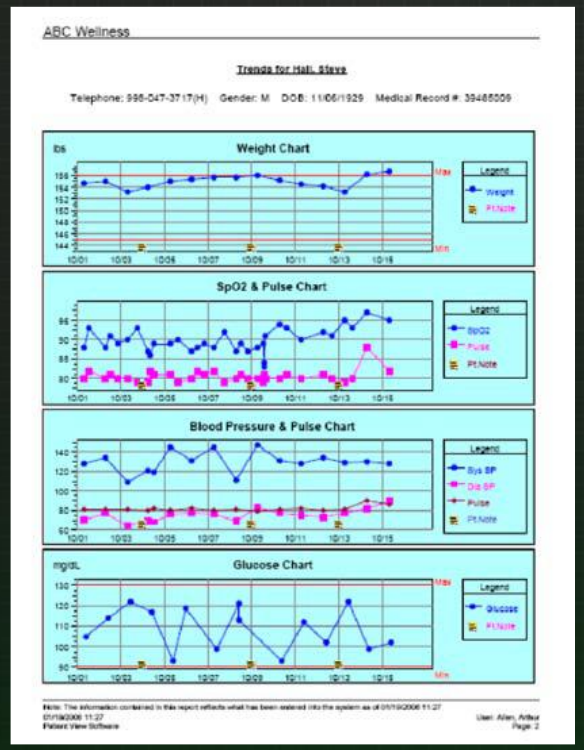


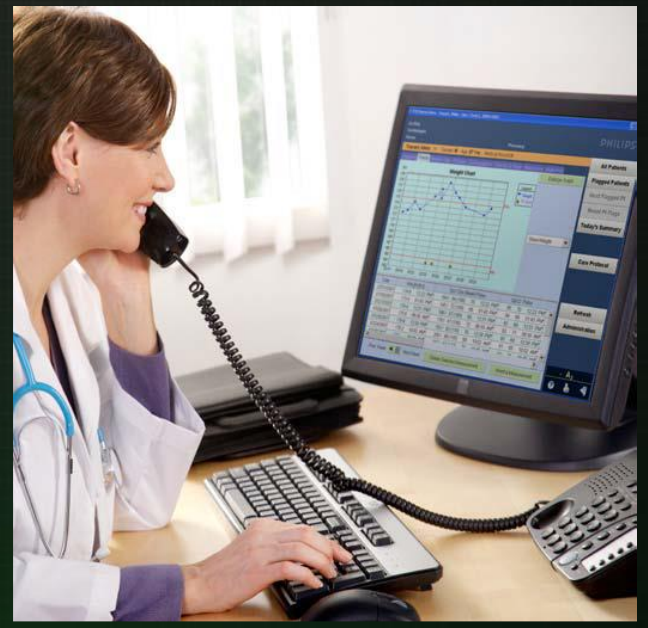


Introduction

Surveillance des patients à distance







Introduction

Les systèmes embarqués nous entourent et nous envahissent. On en croise des dizaines par jour sans le savoir (microonde, GPS, jeux électroniques,…).

Ils sont donc partout, discrets, efficaces et dédiés à ce à quoi ils sont destinés.

Ils sont bourrés d'électronique plus ou moins complexe et d'informatique plus ou moins évoluée.

Dans le domaine de la santé, ils permettent de détecter, de dépister et de traiter d’une manière précoce (Systèmes d’Aide au Diagnostic ou Interventionnels),

Par un suivi des signes vitaux (rythme cardiaque, respiration, glycémie, tumeur,….etc)

Qu’est-ce qu’un système embarqué ?

Un système **embarqué** est défini comme un système électronique et informatique autonome spécialisé dans une tâche bien précise.

Un système embarqué est un système dans lequel un microcontrôleur ou un microprocesseur (un processeur) se situe comme la première source de contrôle; → C’est un système de contrôle automatique.

L'omniprésence des systèmes embarqués dans notre vie est liée à la révolution numérique opérée dans les années 1970 avec l'avènement des processeurs qui sont devenus de plus en plus rapides, puissants et bon marché.

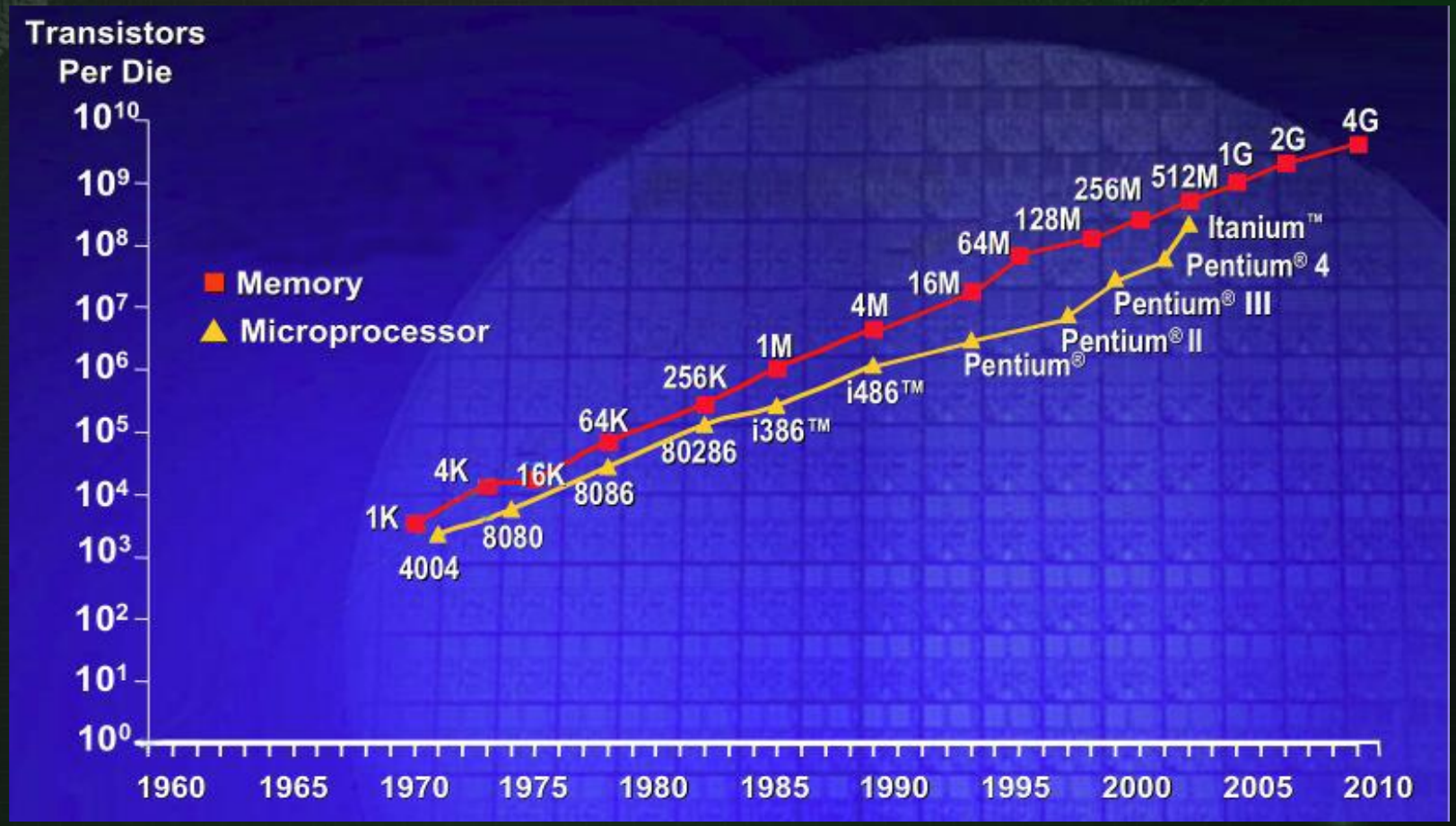


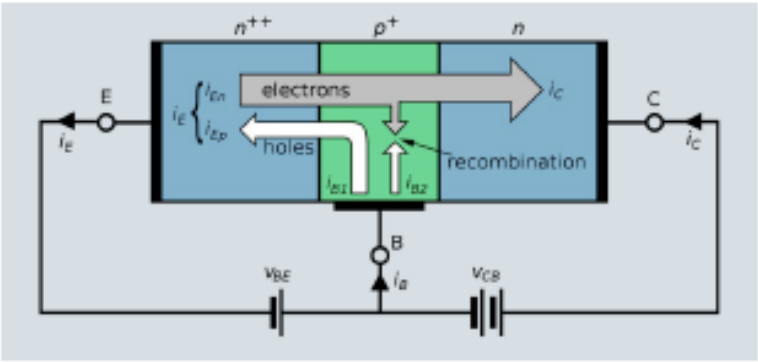
Figure 1.1 : La loi G. Moore pour les processeurs Intel (pour une surface de silicium donnée, on double le nombre de transistors intégrés tous les 18 mois)

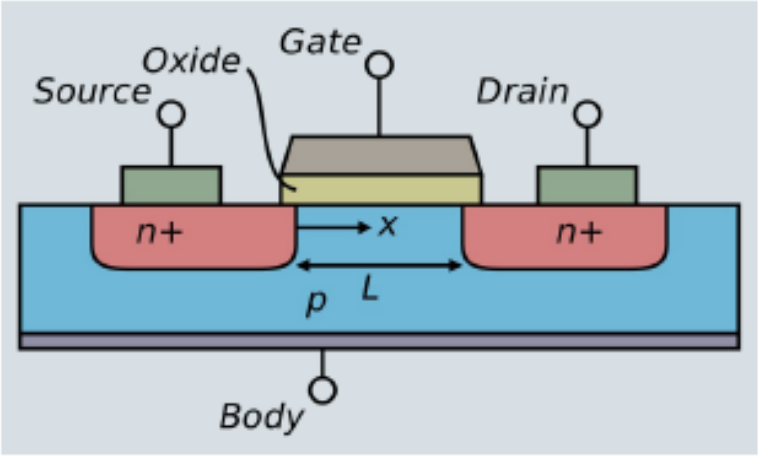
En 1999, il a été vendu pour le marché de l'embarqué :

- 1,3 milliard de processeurs 4 bits.
- 1,4 milliard de processeurs 8 bits.
- 375 millions de processeurs 16 bits.
- 127 millions de processeurs 32 bits.
- 3,2 millions de processeurs 64 bits.

A côté de cela, à cette époque, il a été vendu seulement 108 millions de processeurs (famille x86) pour le marché du PC grand public.

Histoire de l'électronique numérique





1947: Invention du Transistor à Jonction Bipolaire

par Bardeen, Schokley et Brattain (Bell labs), Lauréats du Prix Nobel

1958/1959: Création des Circuits Intégrés

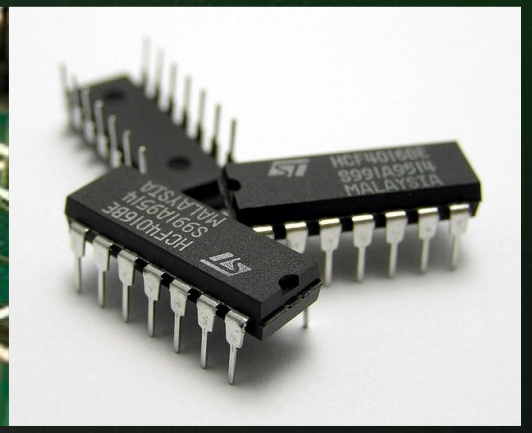
par Texas Instruments, puis Fairchild

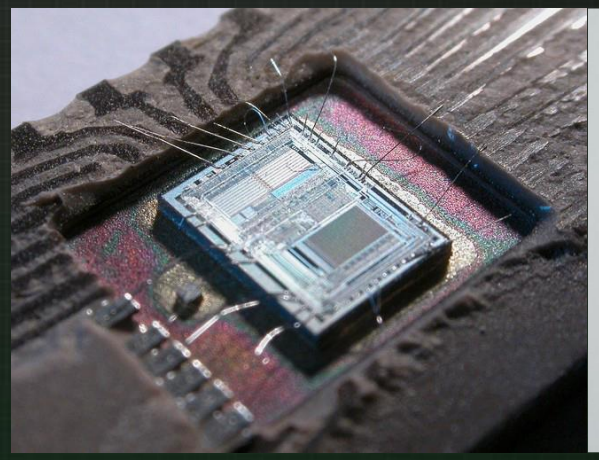
1960: Invention du Transistor à Effet de Champ MOS

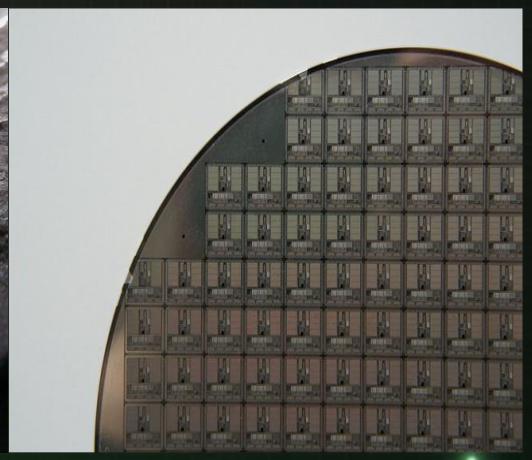
par Mohammed Atalla et Dawon Kahng

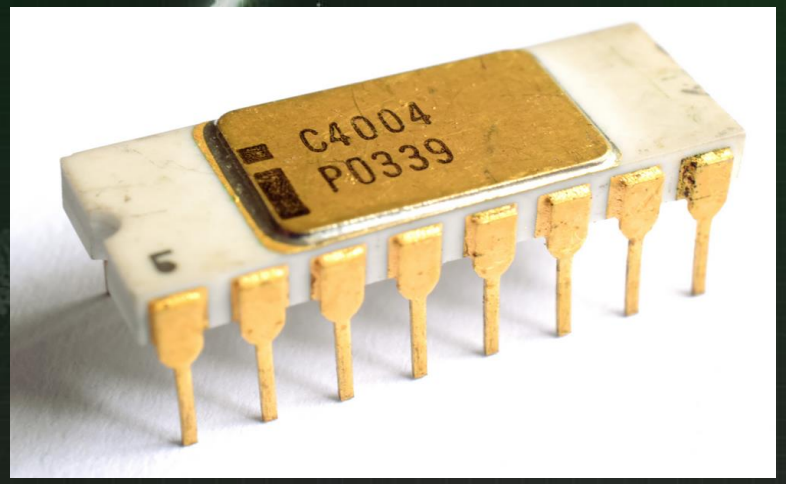
Histoire de l’électronique numérique

First bipolar junction transistor (1947)

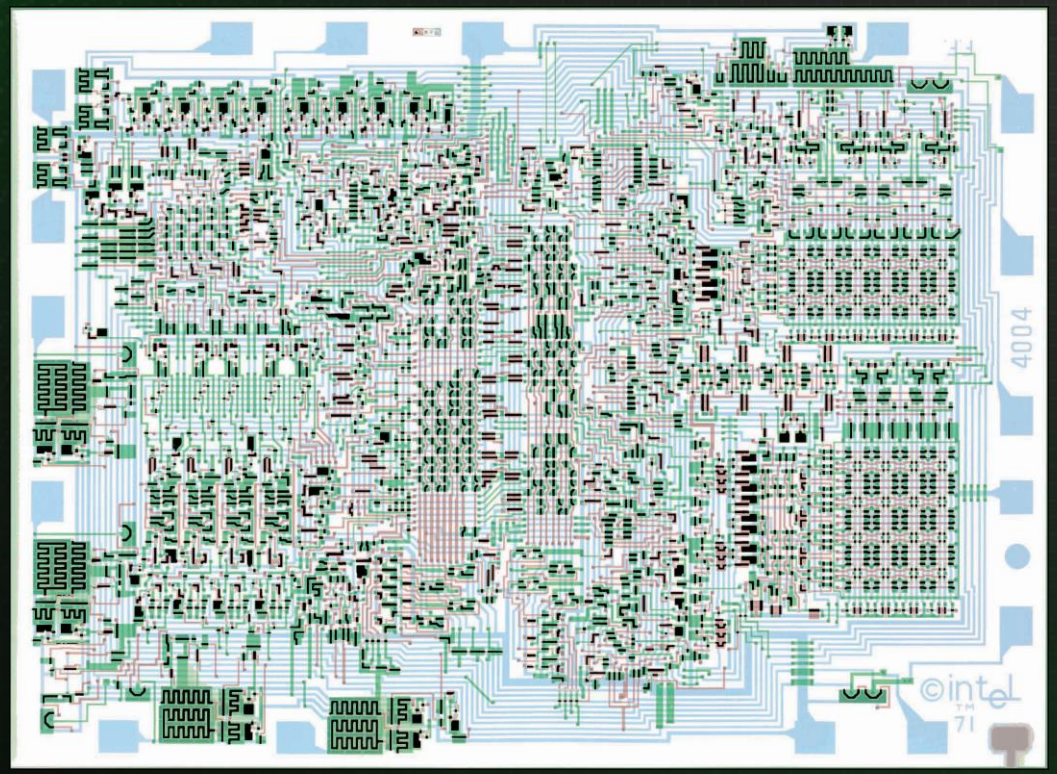


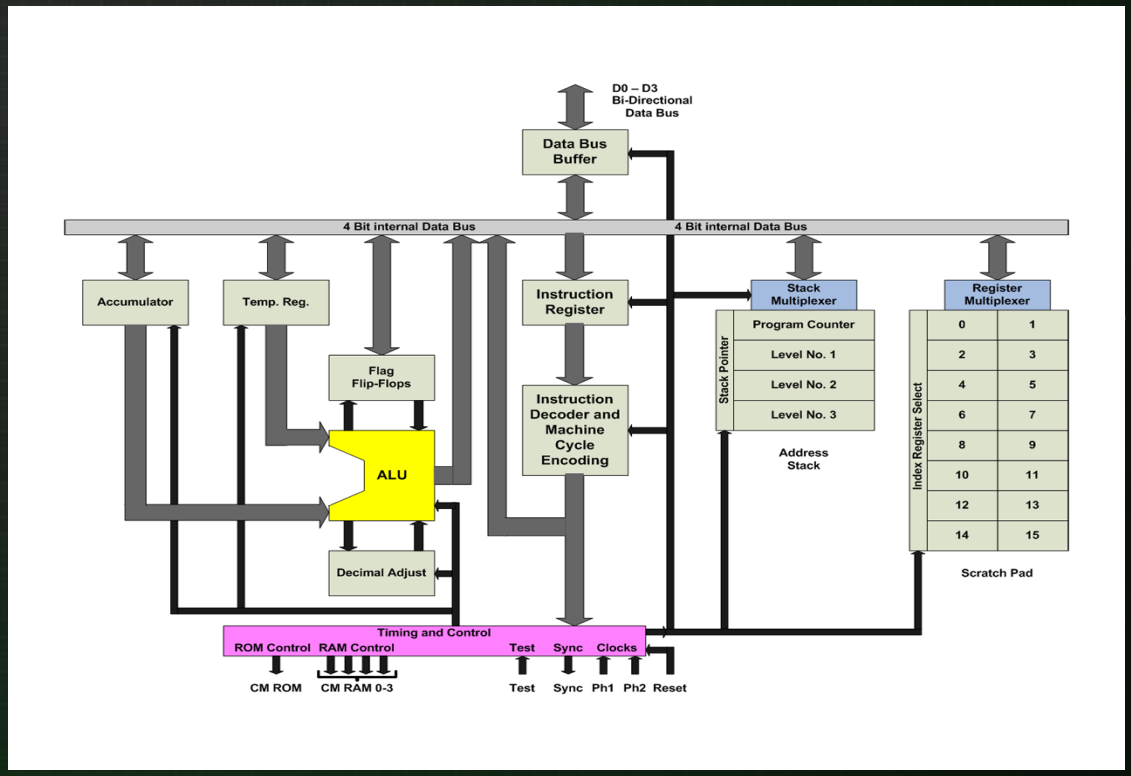






- Le 4004 est le premier microprocesseur (Intel)
- Créé en 1971
- Il contenait 2300 transistors





Evolution des processeurs

40 ANS DE COURSE À L'INNOVATION

1971
PROCESSEUR 4004 D'INTEL

Nombre de transistors : **2.300**
Puissance : **108 kilohertz**
10 microns

1981
PROCESSEUR 8088

Introduit dans les PC d'IBM
Nombre de transistors : **29.000**
Puissance : **5 megahertz**
3 microns

1993
PENTIUM

Nombre de transistors : **3.1 millions**
Puissance : **66 megahertz**
0,8 micron

2006
INTEL CORE 2 DUO

Nombre de transistors :
291 millions
Puissance : **2.93 gigahertz**
65 nanomètres

2012
PROCESSEURS IVY BRIDGE

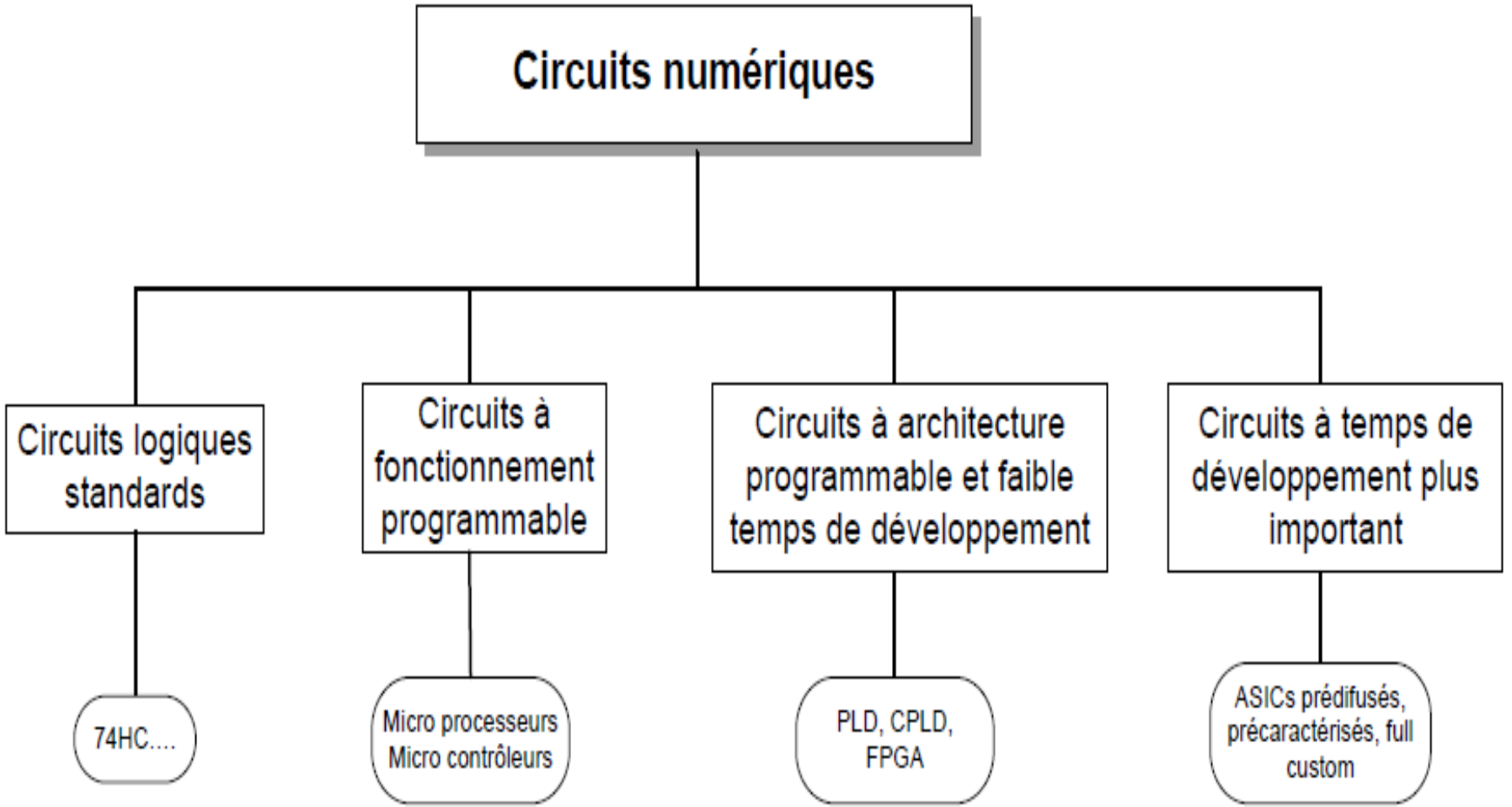
Nombre de transistors :
1.400 millions (3D)
Puissance non communiquée
22 nanomètres

IDÉ / SOURCE ET PHOTOS : INTEL

... etc

Histoire des SE

- Le premier système reconnu comme étant un SE est le AGC (Apollo Guidance Computer), développé par le laboratoire d'Instrumentation de MIT
- AGC a été conçu sur une ROM de 4K de mots et une RAM de 256 mots .
- La fréquence d'horloge du AGC était de 1.024 MHz
- L'unité de l' AGC consistait de 11 instructions et une logique de mots de 16 bits.


Circuits numériques
Circuits logiques standards
Circuits à fonctionnement programmable
Circuits à architecture programmable et faible temps de développement
Circuits à temps de développement plus important
74HC....
Micro processeurs
Micro contrôleurs
PLD, CPLD, FPGA
ASICs prédifusés, précaractérisés, full custom

Définition

Un système embarqué « **Embedded system** » est le produit de l'Electronique et de l'Informatique, conçu spécialement pour réaliser une tâche bien précise.

Son opposé, c'est l'ordinateur, qui réalise des tâches variées.

Il ne possède généralement pas des entrées/sorties standards et classiques comme un clavier ou un écran d'ordinateur.

Le système matériel et l'application sont intimement liés et noyés dans le matériel et ne peuvent être discernables comme dans un PC, on dit que le système est **enfoui** (embedded) dans le matériel.

Caractéristiques d’un système embarqué

Il est conçu pour :

Son faible coût,

Sa faible consommation d’énergie

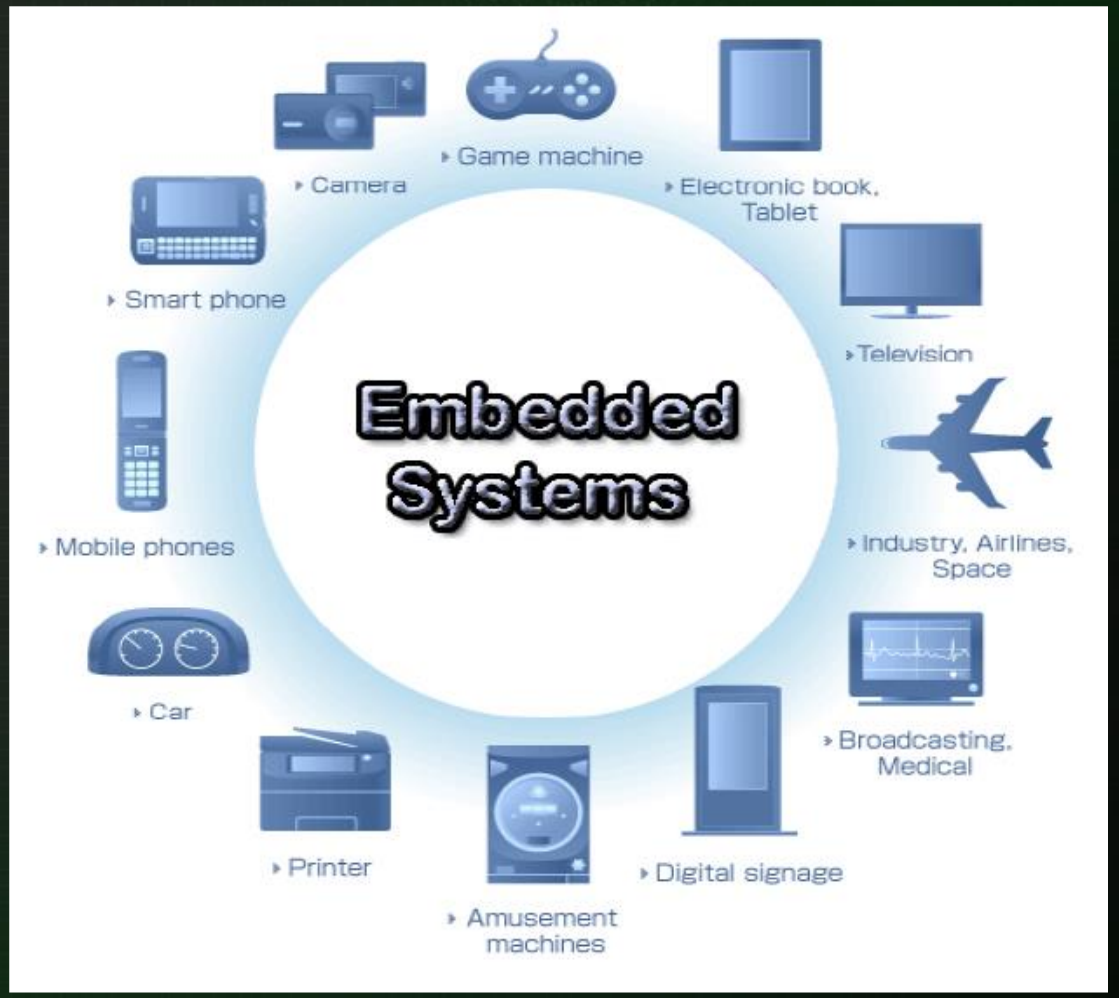
Sa robustesse

C’est un système « dédié » ; il réalise une fonctionnalité précise et n'exécute donc pas une application scientifique ou grand public traditionnelle.

C’est un système à temps réel ; répond rapidement aux évènements internes ou externes.

Domaine d'Application

- Astronautique : fusée, satellite, sonde spatiale…etc.
- Equipement médical.
- Electronique grand public (télévision, four microondes,…).
- Militaire/aérospatial : missile, fusée, satellite,…etc
- Contrôle industriel.
- Automate programmable industriel.
- Télécommunications : Téléphonie, routeurs,…)
- Guichet automatique (GAB)
- Transport Automobile, Aéronautique, ferroviaire…
- Multimédia : Console de jeux vidéo,…



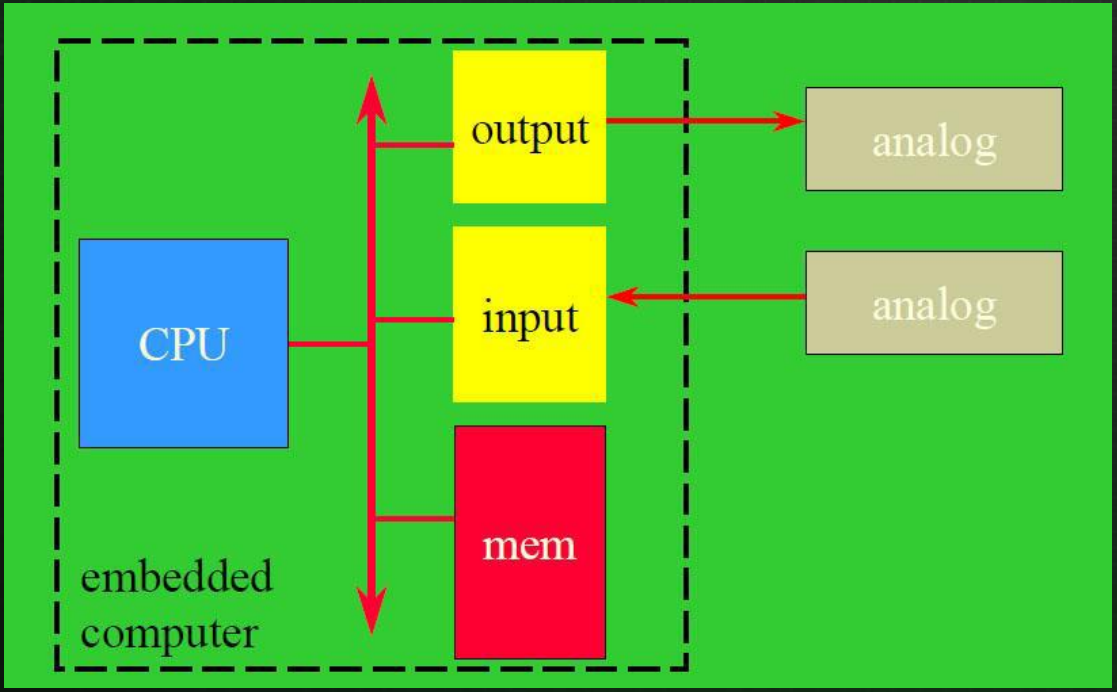
Composition d'un système embarqué

Un système embarqué doit pouvoir :

Communiquer avec l'extérieur : *capteur, actionneur, …*

Exécuter des instructions : *processeur*

Conserver des informations : *mémoire*

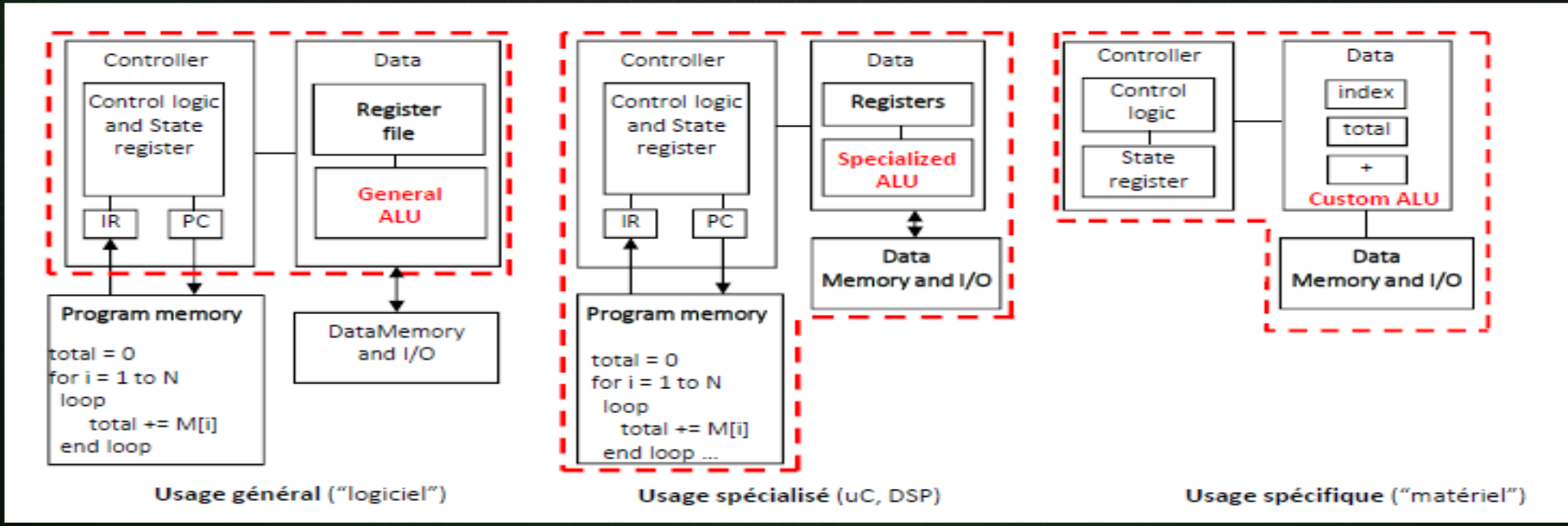


Mais quel processeur utiliser ?

Le choix se fait en fonction des contraintes de design, dont le coût.

Il existe trois types d'unités de calcul :

- Processeur à usage général (gaspillage potentiel de ressources),
- Processeur spécialisé (ressources mieux adaptées), ou
- Processeur dédié (ressources optimisées).



Processeur à usage général

Souvent appelé microprocesseur

•Reprogrammable

•Mémoire de programme

•Grand nombre et variété de registres

•CPU à usage général

Avantages

•Temps de mise en marché et NRE (non-recurring engineering cost) bas (coût de conception initial)

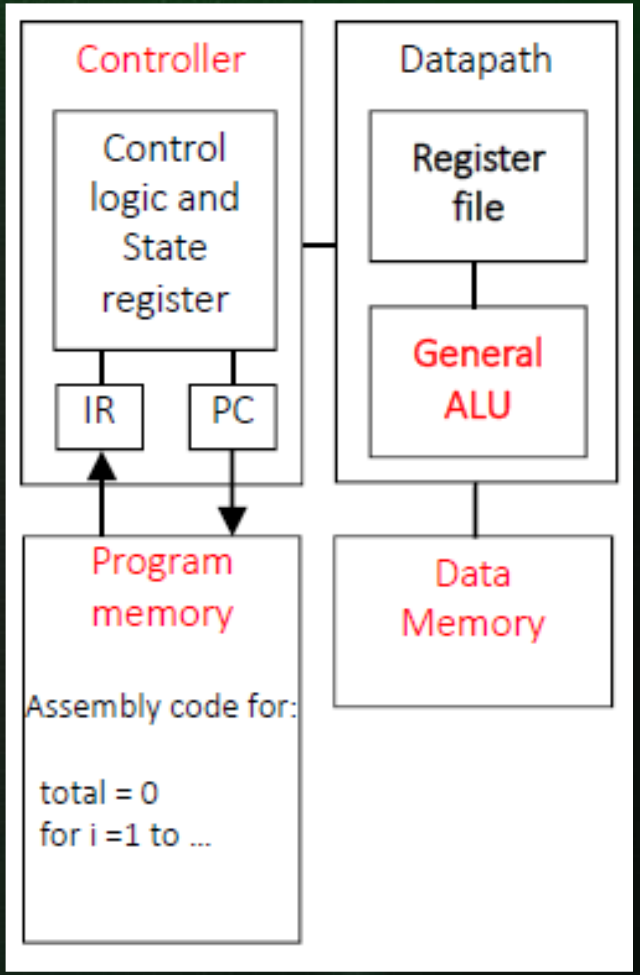
•Grande **flexibilité**

Inconvénients

•Consommation et effets thermiques élevés

•Interfaces complexes

Les famille de processeurs d'Intel(pentium) et AMD (Ryzen) les plus connues



Processeur à usage spécialisé

- Reprogrammable et optimisé pour une classe d'applications
- Compromis entre l'usage général et l'usage dédié ;
- Mémoire de programme intégrée
- Circuits de données optimisés
- Unités fonctionnelles spécialisées

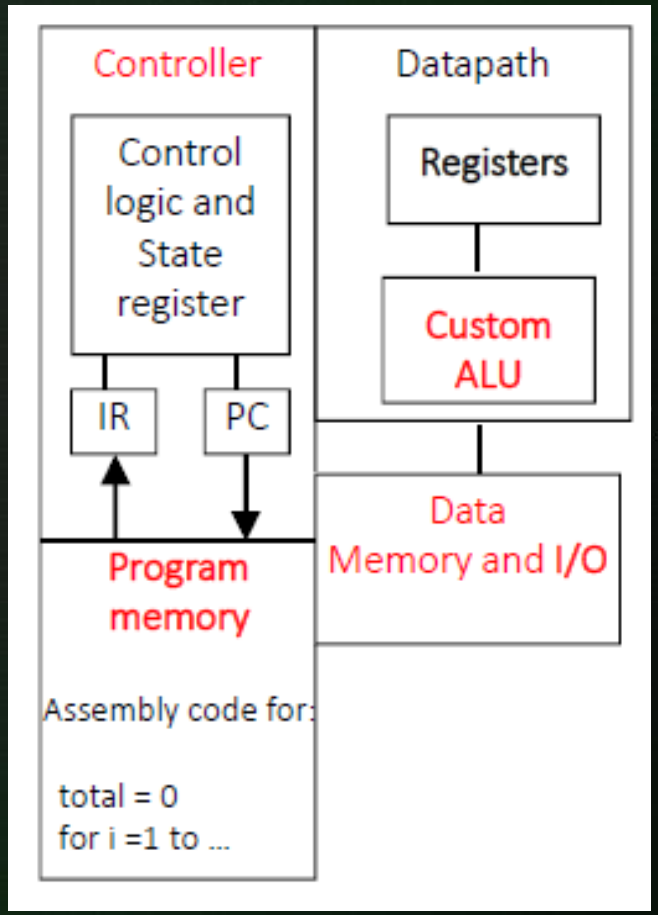
Avantages

- Un peu de flexibilité, bonnes performances,
- dimension et consommation d'énergie réduites
- temps de mise en marché **moyen**

Inconvénients

- NRE **faible à élevé**

exemples : les µC et les DSP(Digital Signal Processor)



Processeur dédié

Coprocesseur, accélérateur ou périphérique

Souvent associé avec un autre type

Contient uniquement les composants requis pour l'application

Non programmable au sens habituel (CLB programmables)

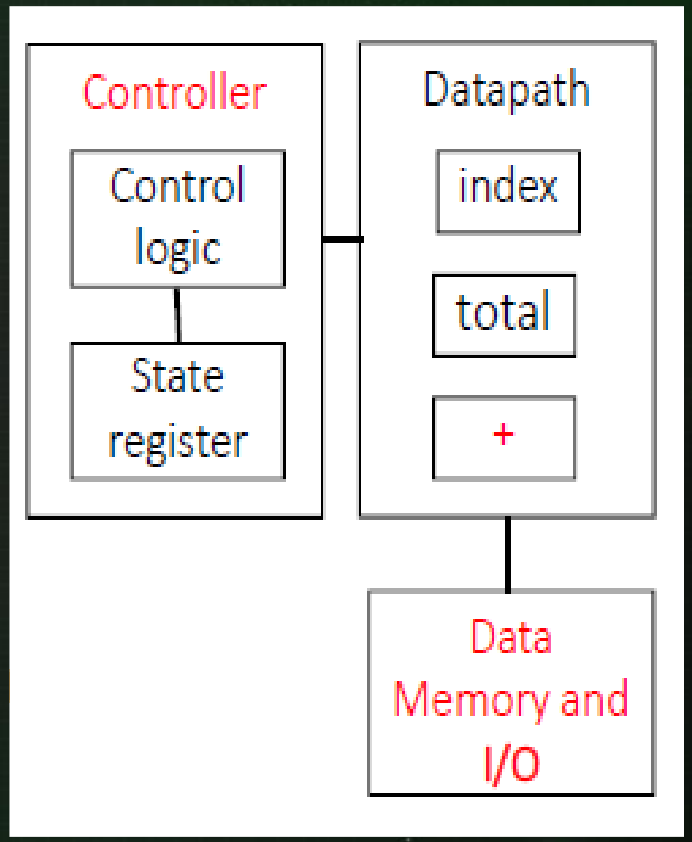
Avantages

- Rapidité,
- consommation d'énergie optimisée pour certains,
- dimensions physiques réduites

Inconvénients

- Temps de mise en marché et NRE **moyens à élevés**
- **Rigidité**

Exemples : ASIC (Application Specific Integrated Circuits), FPGA



La partie Software : jeu d'instructions

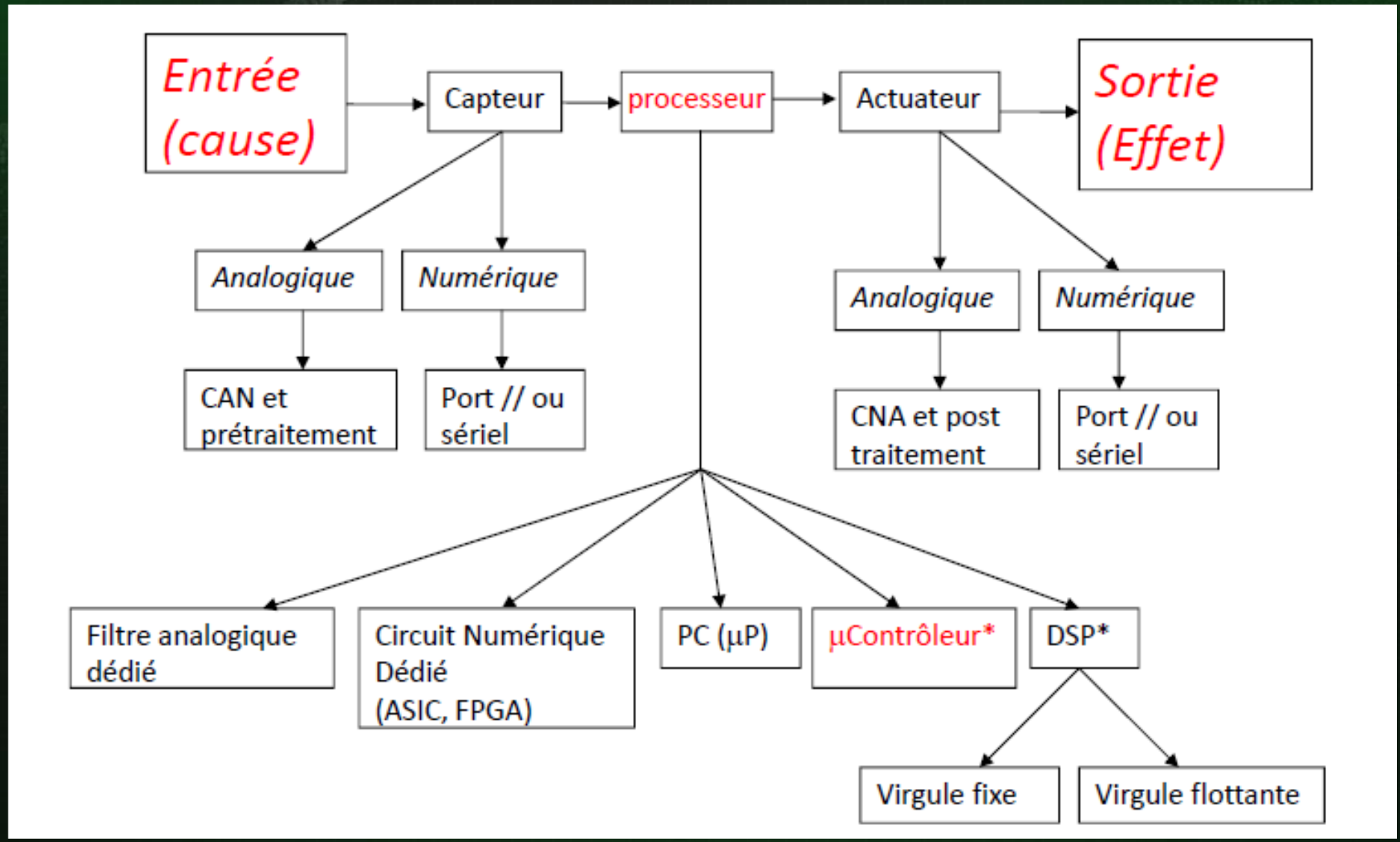
CISC (*complex instruction set*)

- Grand nombre d'instructions
- langages de programmation évolués comme c++, java…
- Exemples : la majorité des processeurs pour ordinateurs de bureau (Pentium,…)

RISC (reduced instruction set)

- Nombre réduit d'instructions
- Opérations simples uniquement
- Exemples : ARM, µC récents.

Diagramme d'un système embarqué



Classification des Systèmes Embarqués

Les systèmes embarqués sont généralement classés en fonction de plusieurs facteurs ; leurs fonctions, leurs performances, leur connectivité, le degré de sécurité, leurs caractéristiques de fonctionnement, leurs domaines d'application et leur consommation d'énergie.

Ces critères peuvent être regroupés comme suit :

1. Selon la génération (on generation)
2. Selon la complexité et la performance (on complexity and performance)
3. Selon le comportement déterministe (on deterministic behavior)
4. Et Selon le type de déclenchement (on triggering)

Classification des SE

Selon la Génération

1. **1G :**

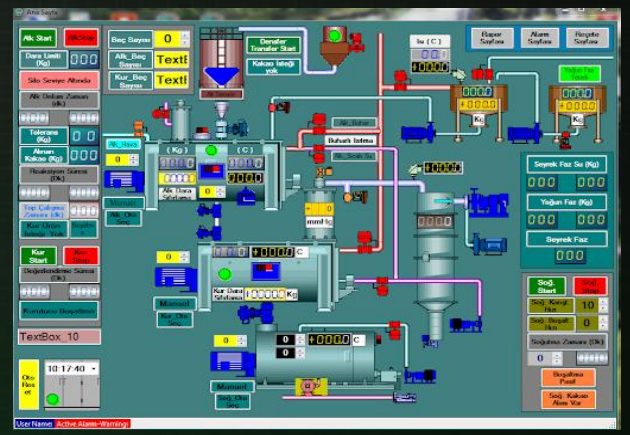
Conçus autour d'un µP ou d'un µC à 8 bits

Exemple : Claviers téléphoniques numériques

2. **2G :**

Conçus autour de µPs à 16 bits ou de µCs à 8/16 bits

Exemple : Système de contrôle et d'acquisition de données (SCADA : supervisory control and Data Acquisition)



Classification des SE

Selon la Génération

3. 3G :

Conçus autour de µPs/µCs de haute performance à 16/32 bits, DSP et ASICs (Application Specific Integrated Circuits).

Exemple : Les Robots

4. 4G :

Conçus autour de µPs/µCs à 64/32 bits et de SoC's

(System on Chips) qui est un système complet embarqué

sur un seul circuit intégré,

Exemple : Les Smart Phones

Classification des SE

Selon la Complexité et la Performance : Cette classification des systèmes embarqués est basée sur les performances et les exigences fonctionnelles.

1. Petite échelle : conçus autour de µPs/µCs 8 ou 16 bits à faible performance et à faible coût. Il convient aux applications simples, Où le temps n'est pas un facteur critique. Il peut ou non contenir un Système d'exploitation.

2. Moyenne échelle : Légèrement complexe en termes de Hardware et de Firmware. Il contient généralement un système d'exploitation (Machines Industrielles).

3. Grande échelle : Hardware et Firmware très complexe. Conçus autour de µp/µc 32 /64 bit de type RISC ou PLDs, ou Processeurs Multicores. Le facteur "temps" est crucial (contient un RTOS).

Classification des SE

Selon le comportement déterministe : Cette classification est applicable Aux systèmes en temps réel. Il existe deux types :

1. Systèmes temps réel souples (Soft real time systems): Ces systèmes n'imposent pas de contraintes temporelles strictes pour les échéances des tâches. Même s'ils nécessitent toujours un délai de réponse à des événements spécifiques
2. Systèmes temps réel durs (Hard real time systems): Ces systèmes exigent le respect strict de leurs contraintes de temps, car le non-respect du temps de réponse peut entraîner de graves conséquences.

Classification des SE

Basée sur le type de déclenchement : Il existe deux types :

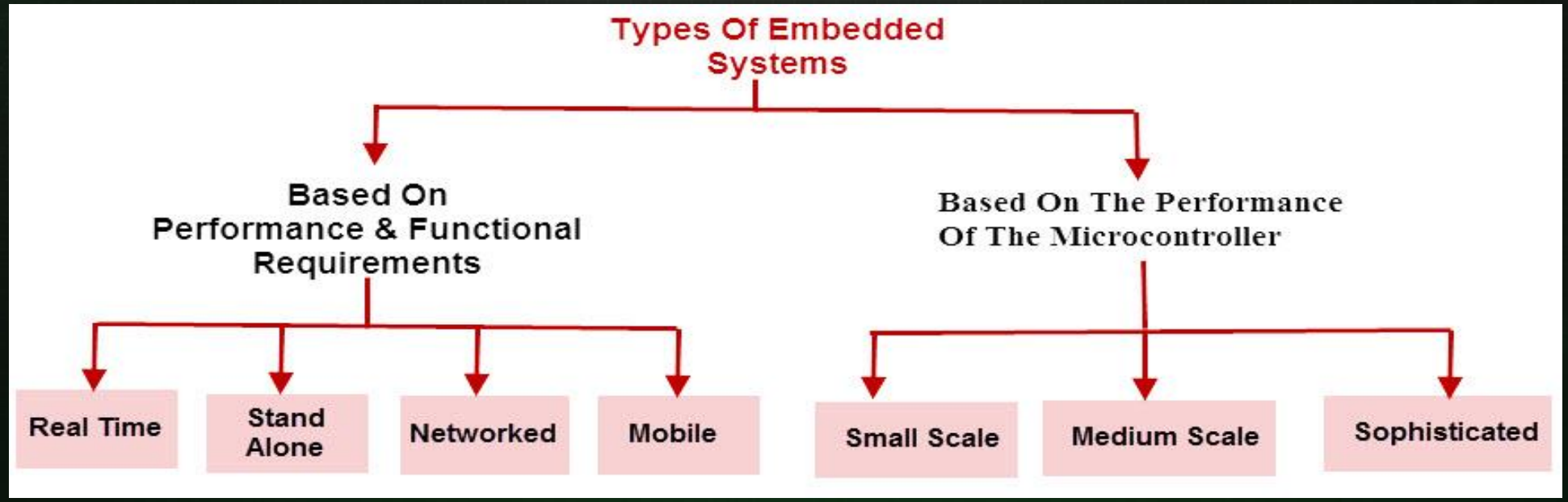
1. Systèmes déclenchés par évènement: Ces systèmes s'appuient sur des événements ou des activités externes spécifiques pour lancer des tâches. Ils réagissent aux changements de variables telles que la température, la pression,…(Système de détection d'intrusion)

2. Systèmes déclenchés par le temps: Ces systèmes sont activés ou lancés à des intervalles prédéterminés ou à un moment précis. Dans ces systèmes, les tâches sont programmées pour être exécutées en fonction d'une heure prédéfinie ou d'un minuteur périodique.(Système de collecte de données de trafic routier)

Types des Systèmes embarqués

Cette classification donne lieu à 4 types de systèmes embarqués qui sont :

1. Les Systèmes autonomes (Standalone Systems)
2. Les Systèmes en temps réel (Real Time Systems)
3. Les Systèmes en réseau (Networked Systems)
4. Les Systemes Mobiles (Mobile Systems)



Les Systèmes embarqués autonomes

Ces systèmes sont autonomes et ne dépendent pas d’autres systèmes.

Ils acceptent les entrées sous forme analogique ou numérique, les traitent et génèrent une sortie.

Exemples : Les lecteurs mp3, les appareils photo numériques, les consoles de jeux vidéo, etc…

Les Systèmes embarqués en Temps Réel

Ces systèmes fournissent le résultat requis dans un délai prédéterminé. Ils respectent des contraintes de temps strictes pour l'exécution des tâches (souples et durs).

(Exemple : Les systèmes de surveillance des aéronefs)

Les Systèmes embarqués en Réseau

Ils se connectent à un réseau pour accéder aux ressources (Un domaine en expansion).

Ce réseau peut être un réseau LAN (Local Area Network), WAN (Wide Area Network) ou Internet, et la connexion peut être avec ou sans fil.

Exemple : Les serveurs Web intégrés, les systèmes de sécurité domestique, …

Les Systèmes embarqués mobiles

Ces systèmes sont compacts et économes en ressources.

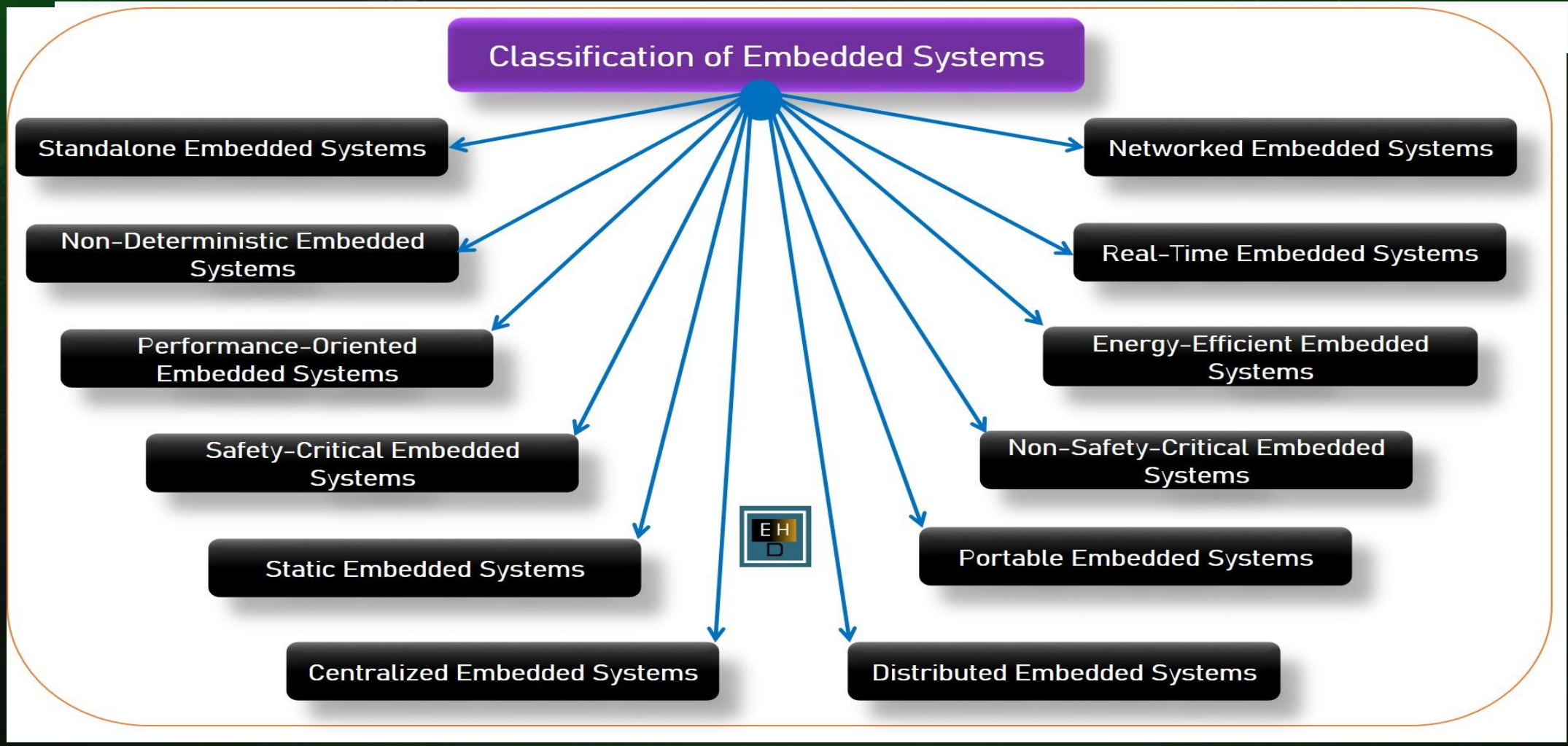
On les trouve couramment dans les appareils portables tels que les téléphones mobiles, les appareils photo numériques, les lecteurs MP3 et les assistants numériques personnels.

Ces systèmes sont conçus pour être faciles à utiliser

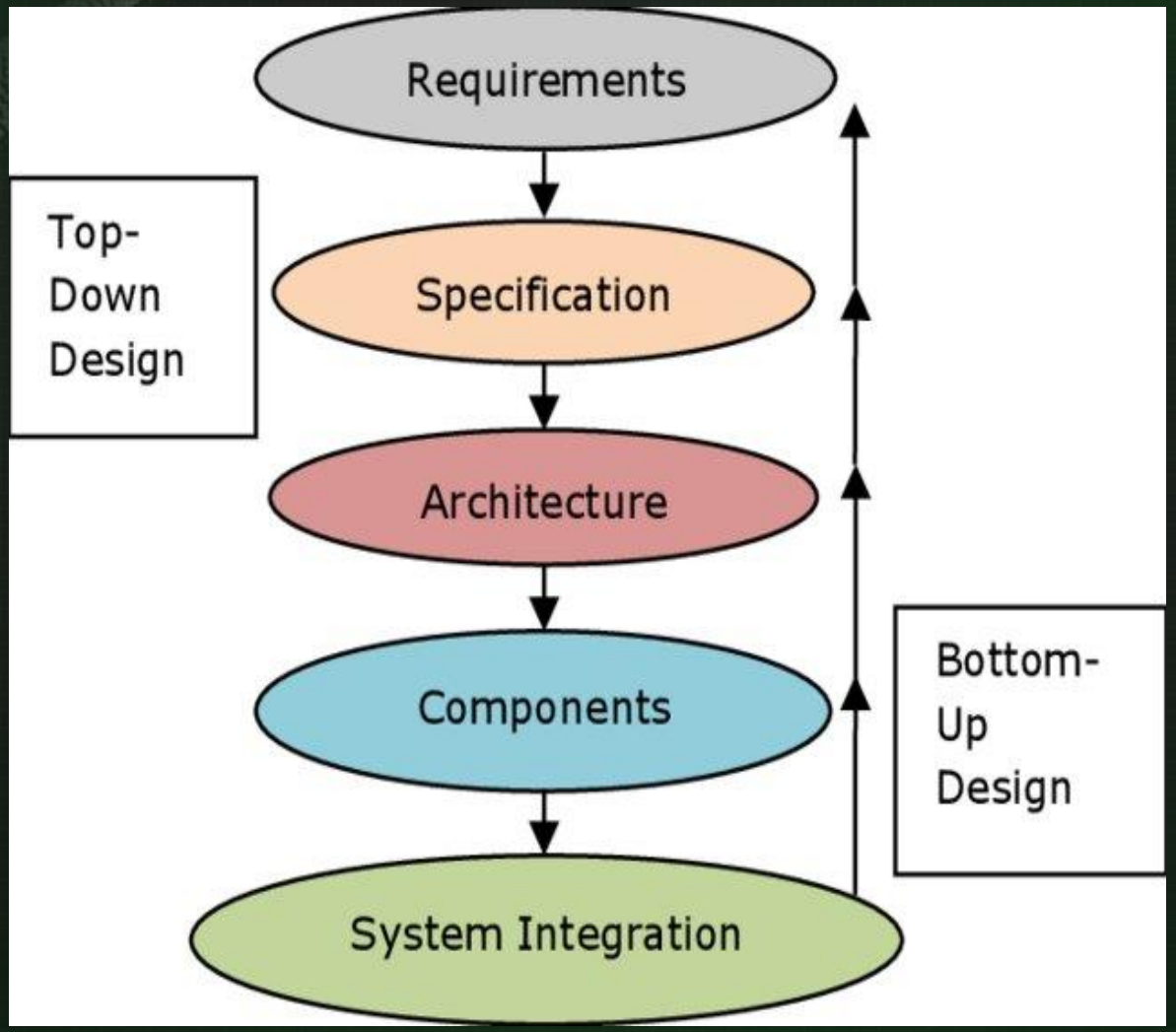
et portables, ce qui les rend idéaux pour les applications

en déplacement.

Types des systèmes embarqués



Les étapes de conception des SE

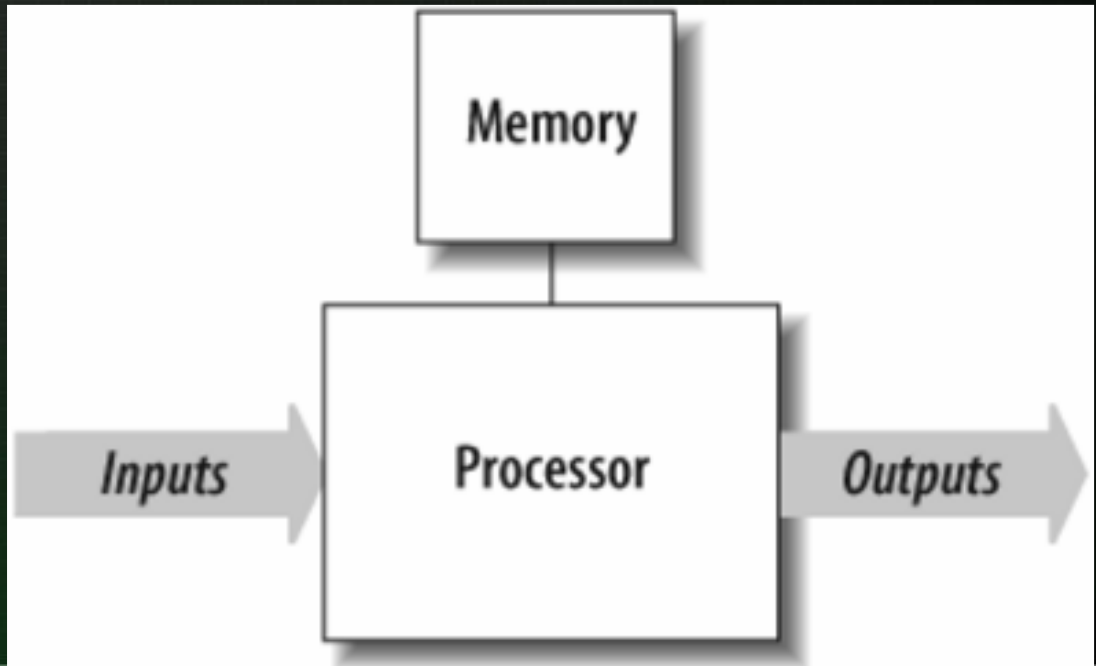


Les étapes de conception des SE

1. Les exigences (Requirements) : Identifier le problème (Que veut le client ?)
2. Les Spécificités (Specification) : Une description plus détaillée du système (les fonctions, les données, les entrées/sorties…)
3. L’architecture : (Hardware & Software)
4. Composants (Components) : La conception détaillée du HW. et du SW (processeur, langage de programmation,…).
5. Intégration

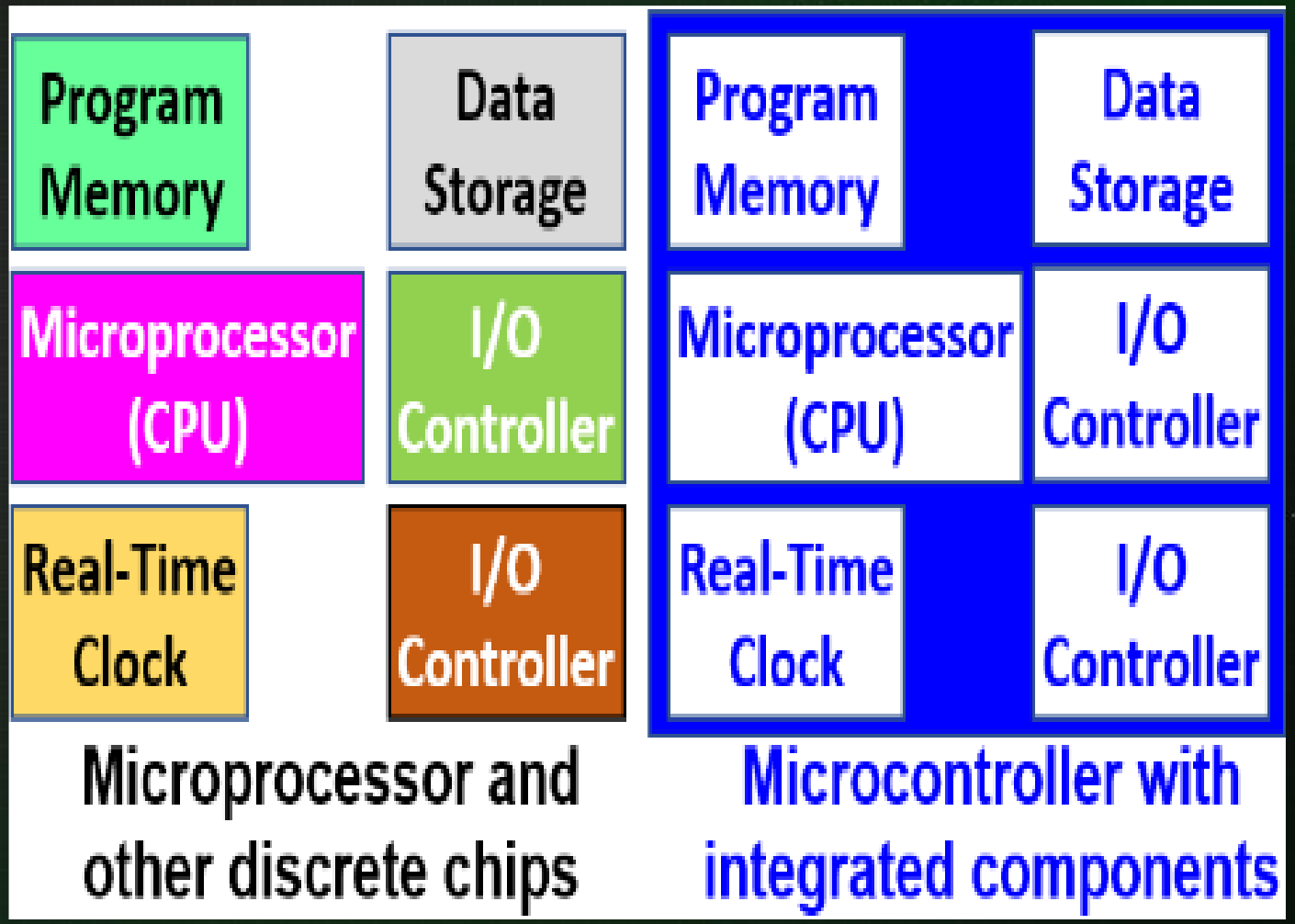
L’Architecture des Systèmes Embarqués

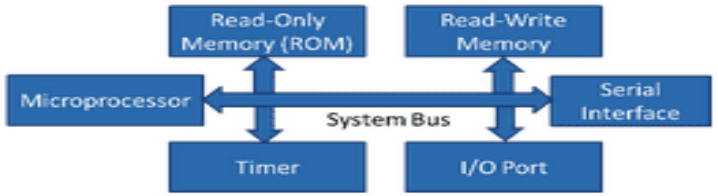
- John Von Neumann est à l'origine d'un modèle de machine universelle de traitement programmé de l’information (1945). Cette architecture sert de base à la plupart des systèmes à processeur actuels. Elle est composée des éléments suivants :
  - Le processeur (Unité centrale de traitement=CPU) ;
  - Une mémoire principale (MP) qui contient les instructions et les données lors de l’exécution d’un programme ;
  - Des interfaces d’E/S pour communiquer avec l’extérieur.



Types de processeurs

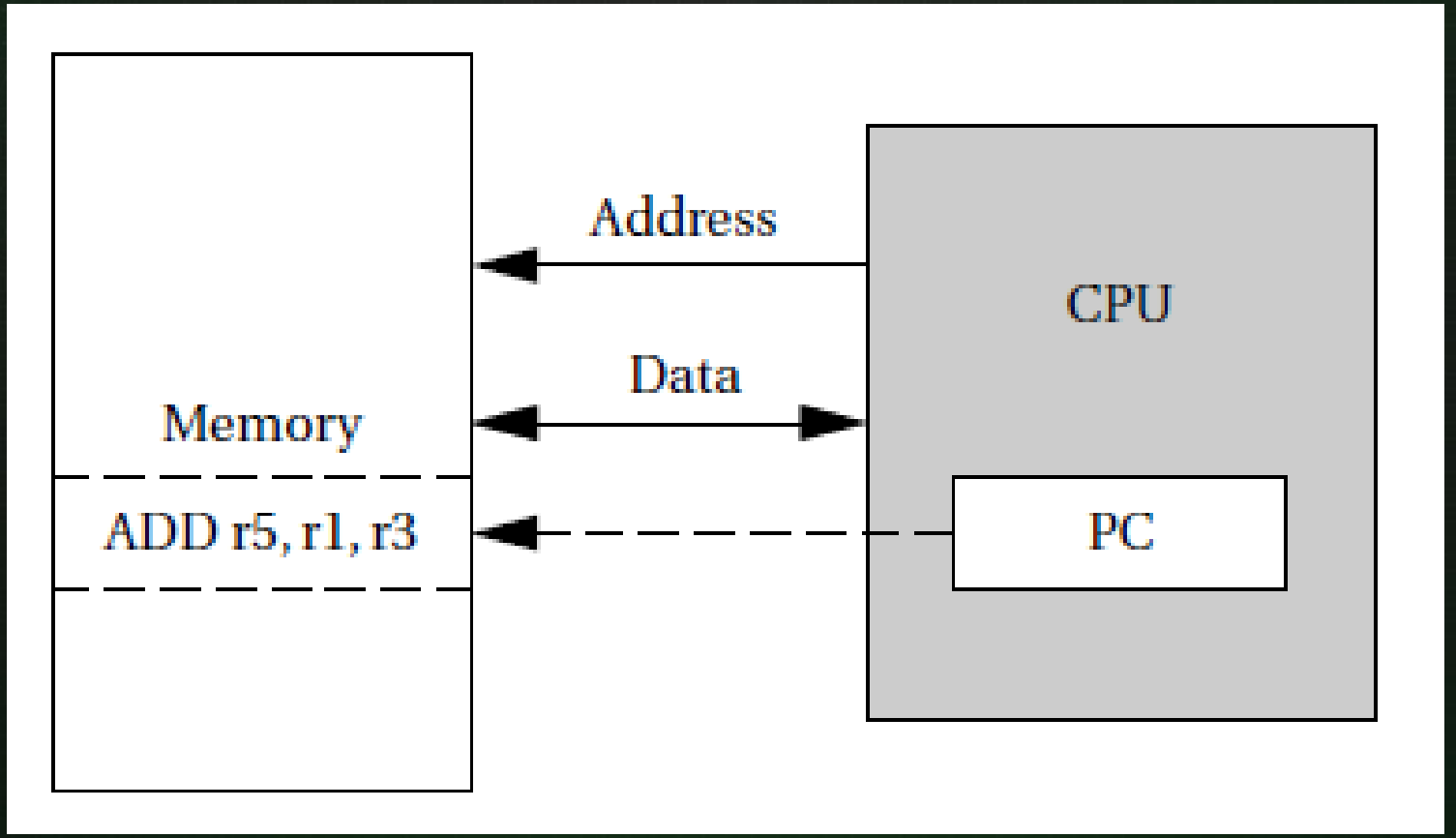
- Microprocesseurs
- Microcontrôleurs
- DSP
- System on Chip (SoC)
- ASIC
- CPLD & FPGA



Microprocessor	Micro Controller
	
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

Architecture de Von Neumann (Princeton)

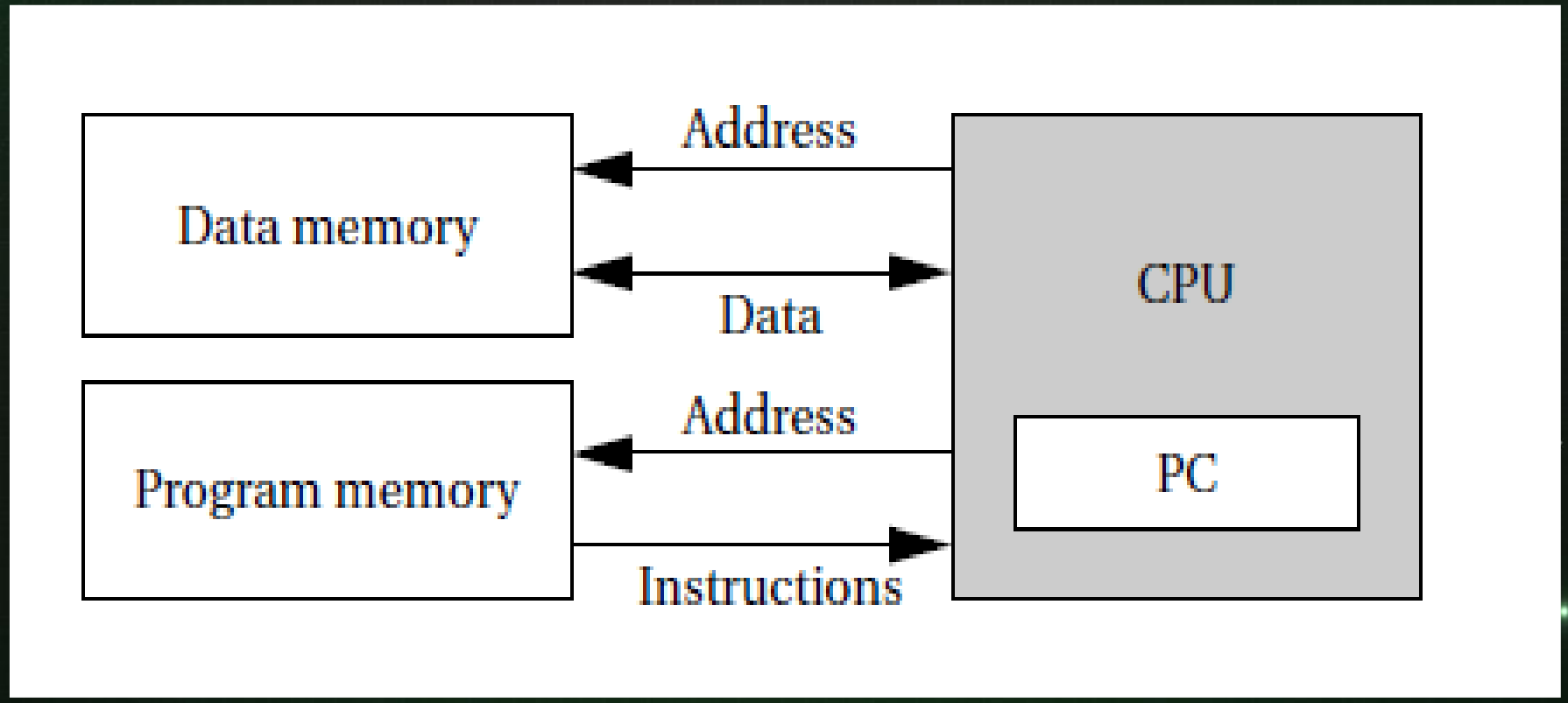
Les instructions et les données partagent les mêmes bus (Utilisée en GPP)



Architecture de Harvard

Le programme et les données sont stockés dans deux modules de mémoire séparés et utilisés

Utilisée en µC et DSP)

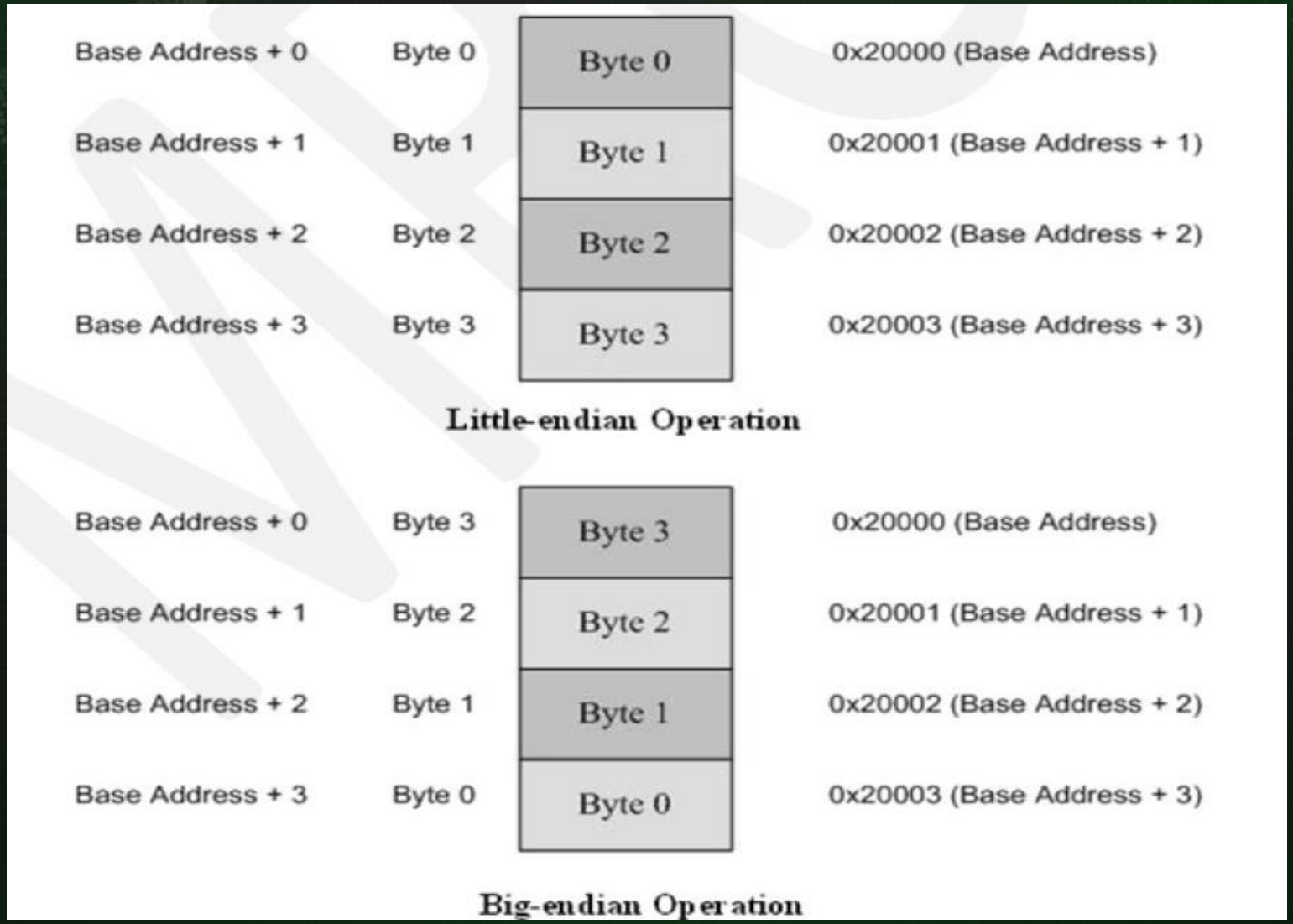


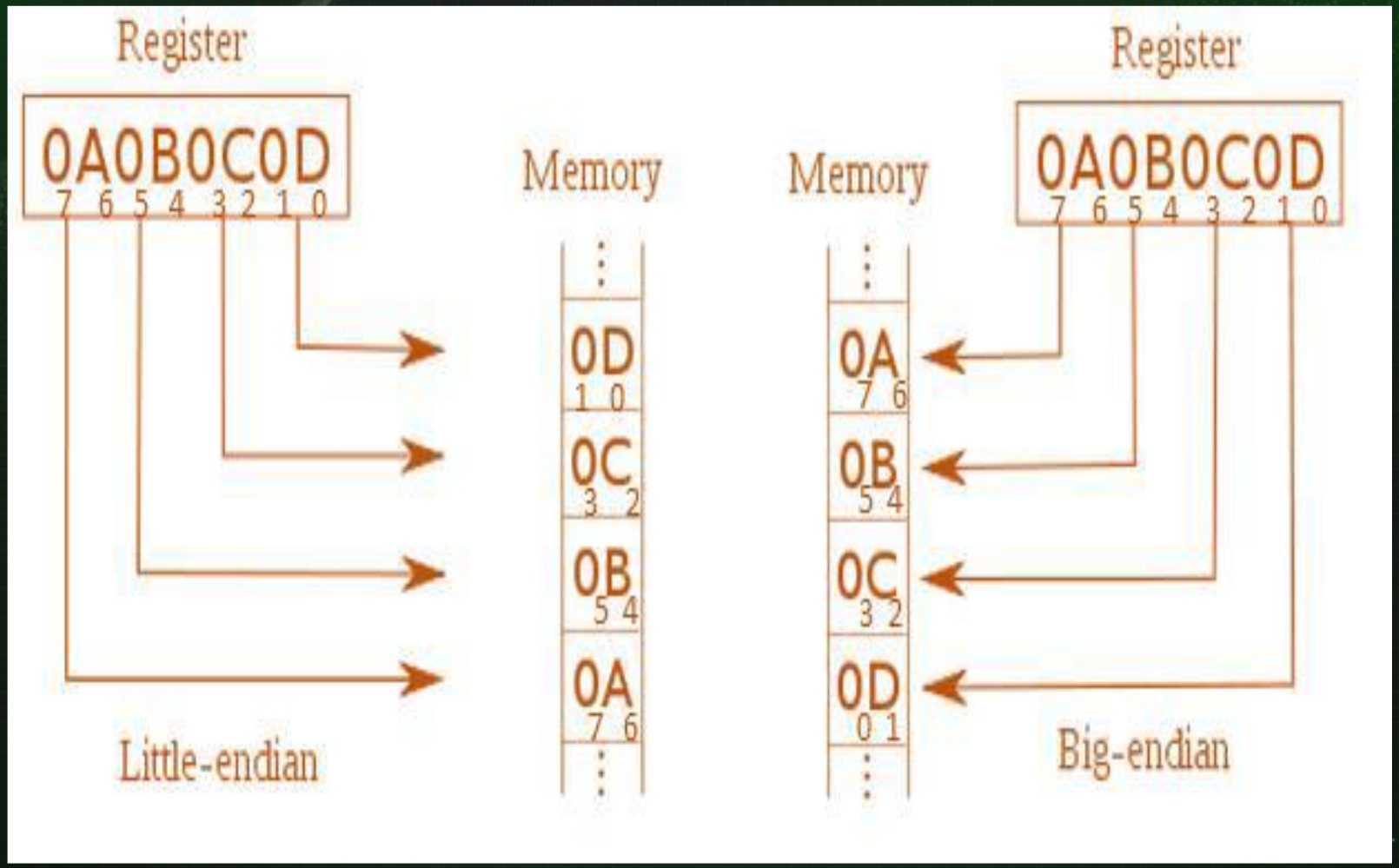
Architecture CISC & RISC

RISC	CISC
RISC signifie (Reduced Instruction Set Computer).	CISC signifie (Complex Instruction Set Computer).
Les processeurs RISC ont des instructions simples prenant environ un cycle d'horloge. Le cycle d'horloge moyen par instruction (CPI) est de 1,5	Le processeur CSIC dispose d'instructions complexes prenant plusieurs horloges pour l'exécution. Le cycle d'horloge moyen par instruction (CPI) est compris entre 2 et 15.
Le jeu d'instructions est réduit, c'est-à-dire qu'il ne contient que quelques instructions dans le jeu d'instructions. Beaucoup de ces instructions sont très primitives.	Le jeu d'instructions comprend diverses instructions pouvant être utilisées pour des opérations complexes.
Le jeu d'instructions comprend diverses instructions pouvant être utilisées pour des opérations complexes.	CISC a de nombreux modes d'adressage différents et peut donc être utilisé pour représenter plus des instructions dans différents langages de programmation de niveau supérieur.
Les modes d'adressage complexes sont synthétisés à l'aide du logiciel.	CISC supporte déjà des modes d'adressage complexes
Le temps d'exécution est très bas	Le temps d'exécution est très élevé

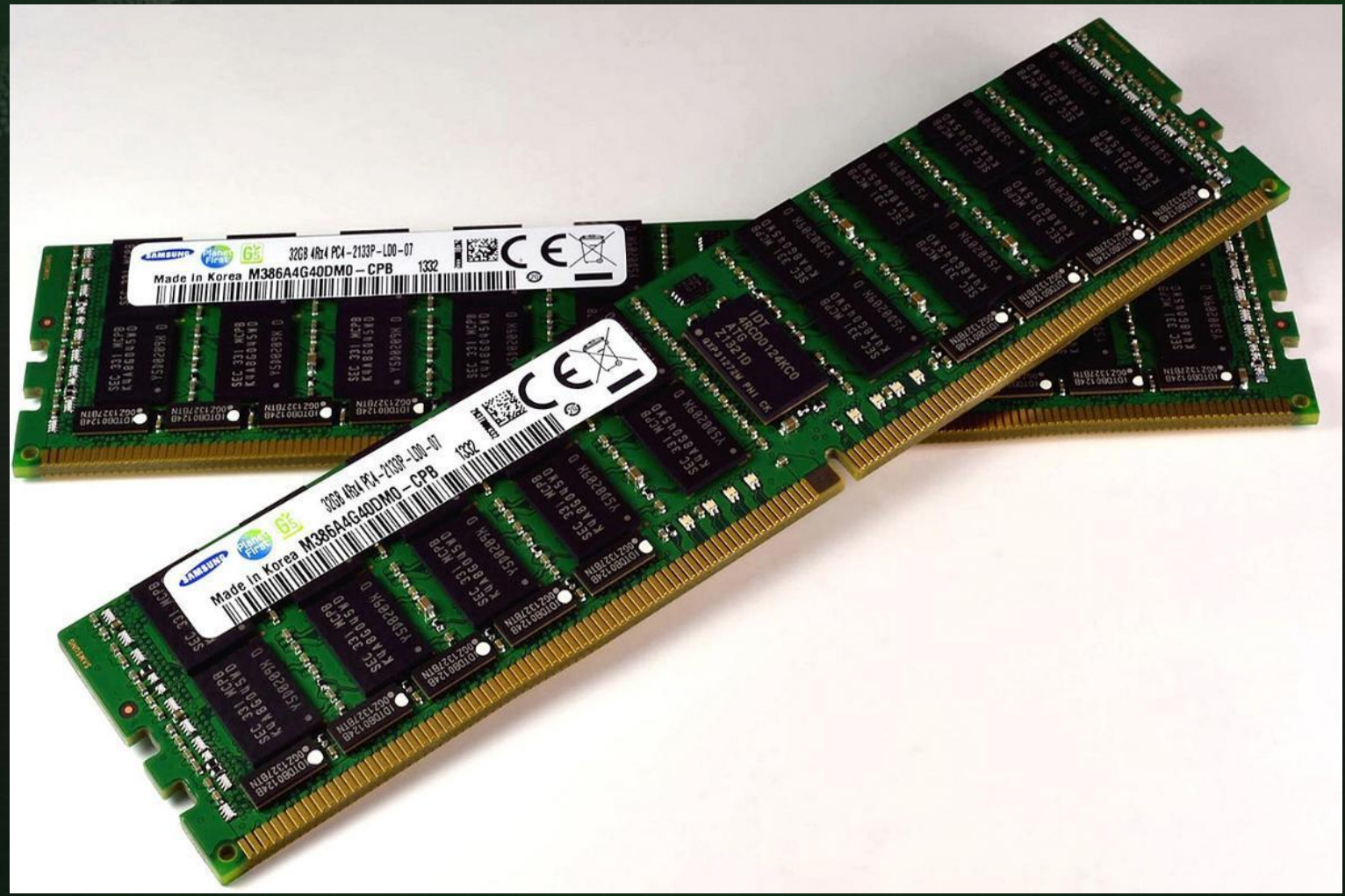
Architecture Big Endian & Little-Endian

C’est la manière de stocker les données dans une mémoire




Register
0A0B0C0D
7 6 5 4 3 2 1 0
Memory
0D
1 0
0C
3 2
0B
5 4
0A
7 6
Little-endian
Memory
0A
7 6
0B
5 4
0C
3 2
0D
0 1
Register
0A0B0C0D
7 6 5 4 3 2 1 0
Big-endian

La Mémoire dans les SE



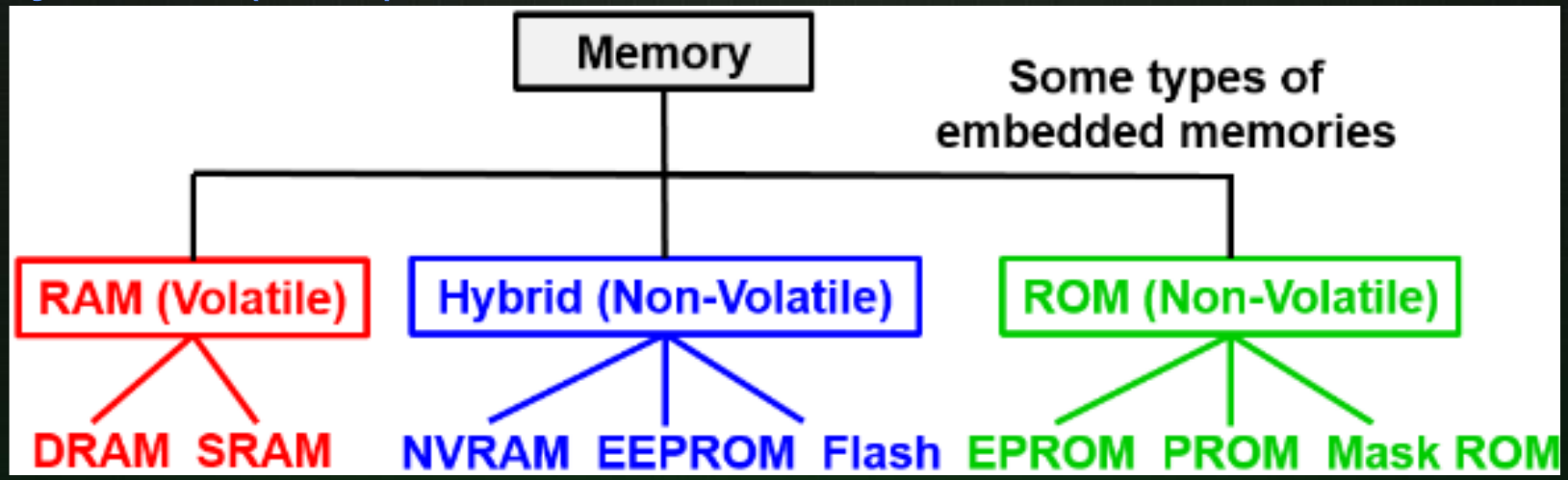
La mémoire dans les SE

Il existe 3 types de mémoires qu'on peut utiliser dans les SE:

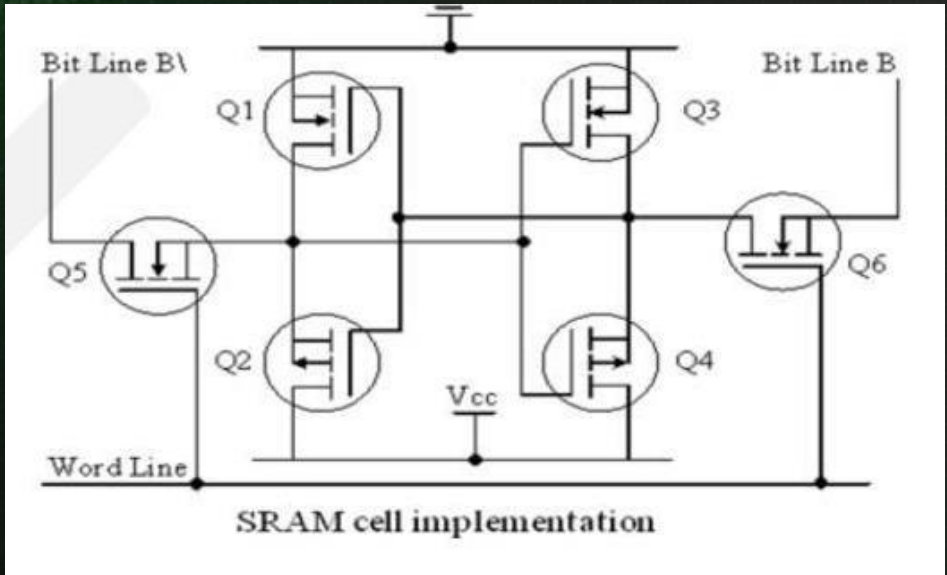
•Les mémoires **ROM** (Read Only Memory) chargée de stocker le programme. C'est une mémoire à lecture seule.

•Les mémoires **RAM** (Random Access Memory) chargée de stocker les données intermédiaires ou les résultats de calculs. On peut lire ou écrire des données dedans, ces données sont perdues à la mise hors tension. (mémoire primaire)

•Les mémoires **hybrides :** (Exemple, NVRAM :Non Volatile RAM avec une batterie de sauvegarde)



La mémoire SRAM et DRAM



SRAM cell implementation

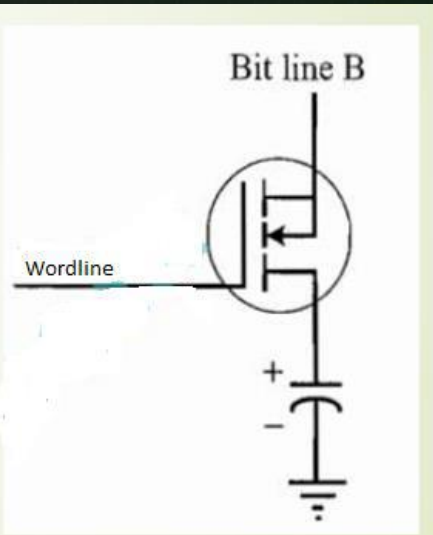


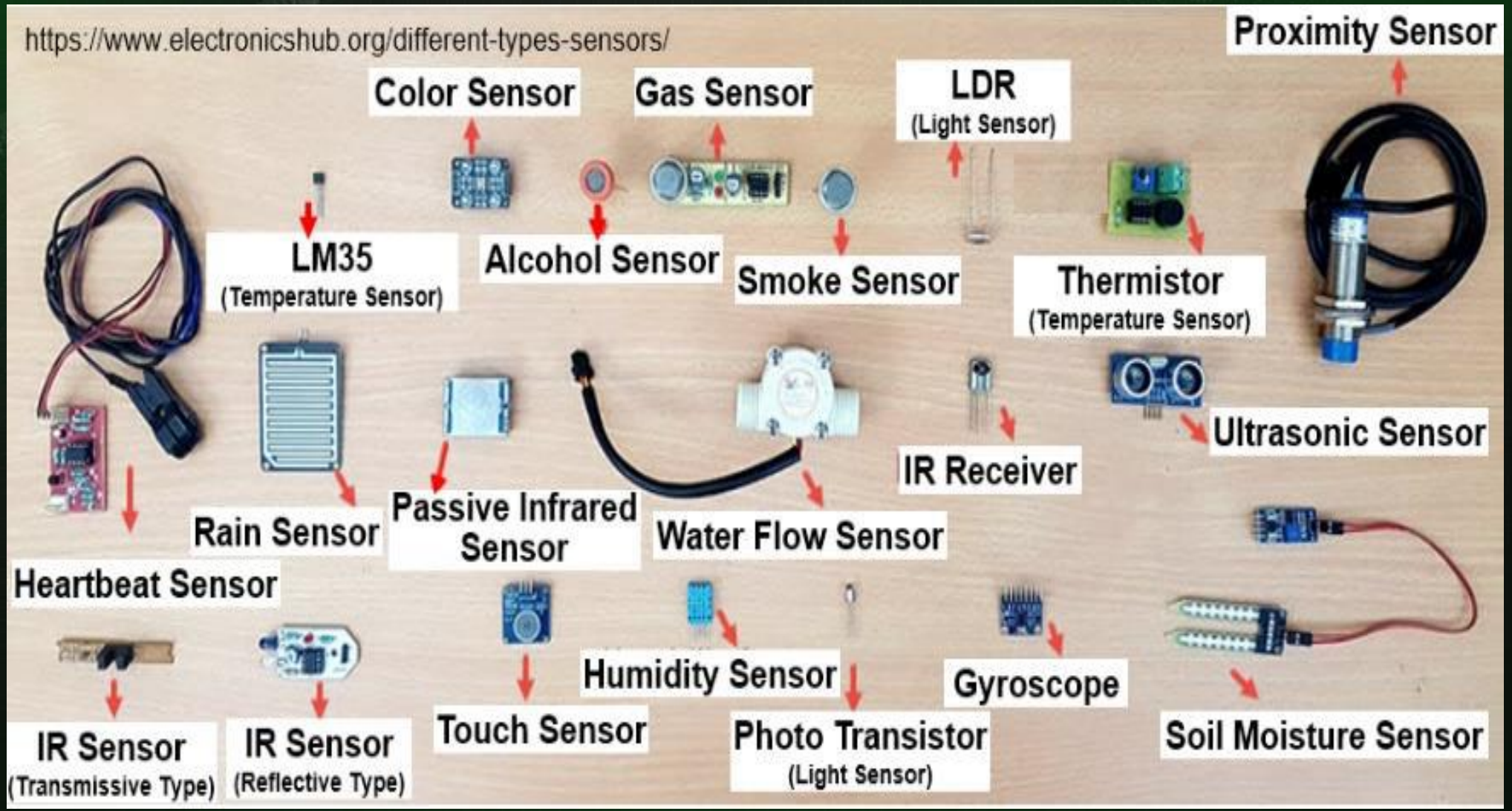
Figure. DRAM Cell Implementation

Les critères pour choisir une mémoire dans les SE

On doit tenir compte des facteurs suivants :

- La vitesse (Suffisamment rapide pour fonctionner en temps réel ??)
- Taille et Capacité de stockage des données (Exemple : Mémoire nécessaire pour contenir le système d’exploitation)
- La largeur des Bus
- La consommation d’énergie
- Le coût

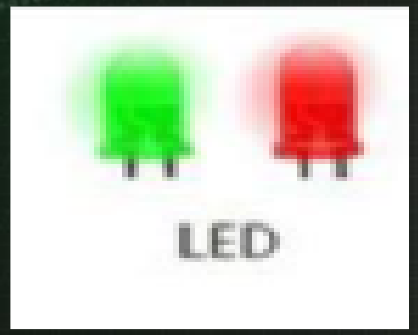
Les capteurs utilisés dans les SE.



Les capteurs dans le domaine de la santé

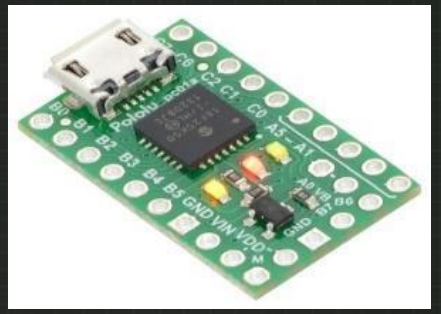
Capteurs de flux : pompes à injection	 
Capteurs électrochimiques : Glucomètres	 
Capteurs photo-optiques : oxymètres	 
Capteurs de Température : Thermomètres	
Capteurs de Pression : tensiomètres	 

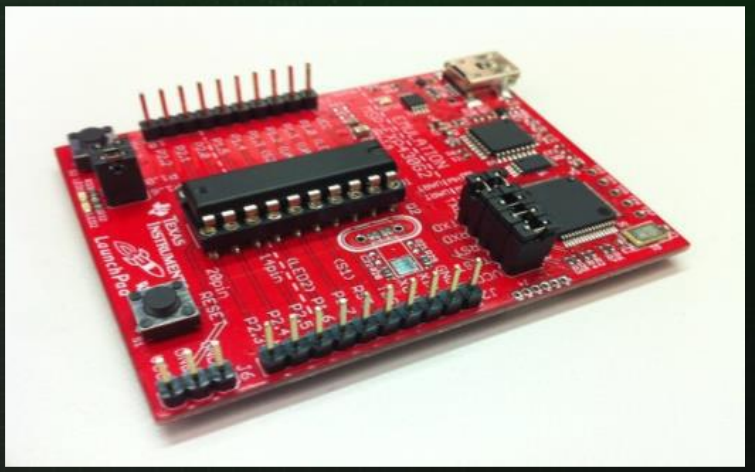
Les actionneurs utilisés dans les SE.

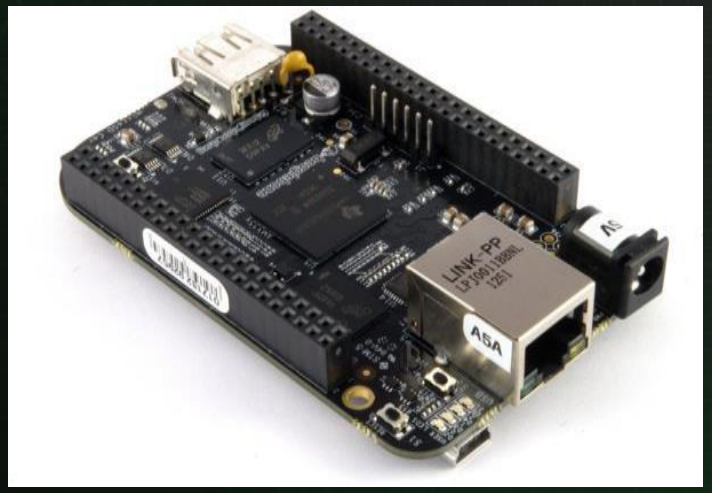


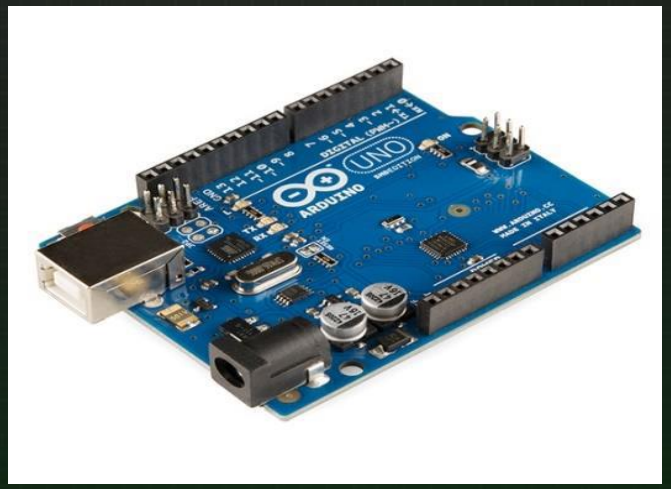
Chapter 2:
Programming an embedded system

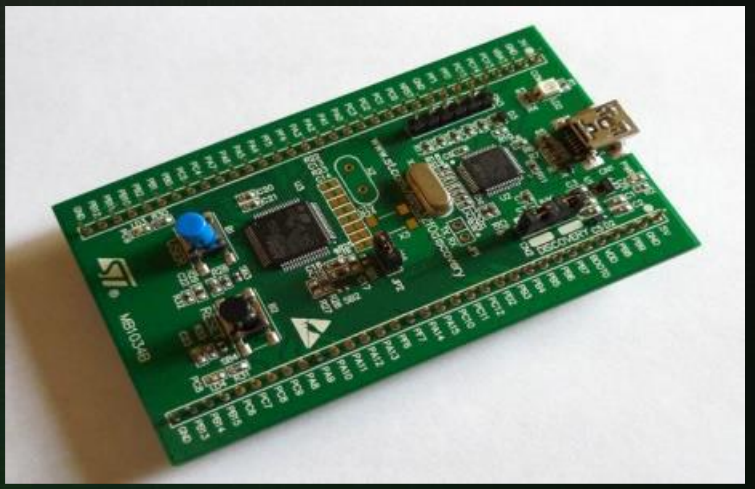
Introduction











Introduction

Le firmware embarqué est chargé de contrôler les différents périphériques du matériel (Hardware) embarqué et générer une réponse conformément aux exigences fonctionnelles mentionnées dans les « exigences » (Requirements) du produit à concevoir.

Le firmware est considéré comme le cerveau principal du système embarqué.

Pour la plupart des systèmes embarqué, le programme (le firmware) est stocké dans une mémoire permanente (ROM) et n’est pas modifiable par l’utilisateur.

La conception d’un firmware nécessite une compréhension du hardware utilisé tel que la carte mémoire, les interfaces, les ports d’E/S,…etc.

Elle nécessite aussi la maîtrise de quelques langages de programmation de haut niveau ou de bas niveau (Assembleur, C, C++, Java, VHDL,…).

Les approches pour concevoir un firmware

Elles dépendent de la complexité du système embarqué à concevoir,

Et de la vitesse de fonctionnement exigée.

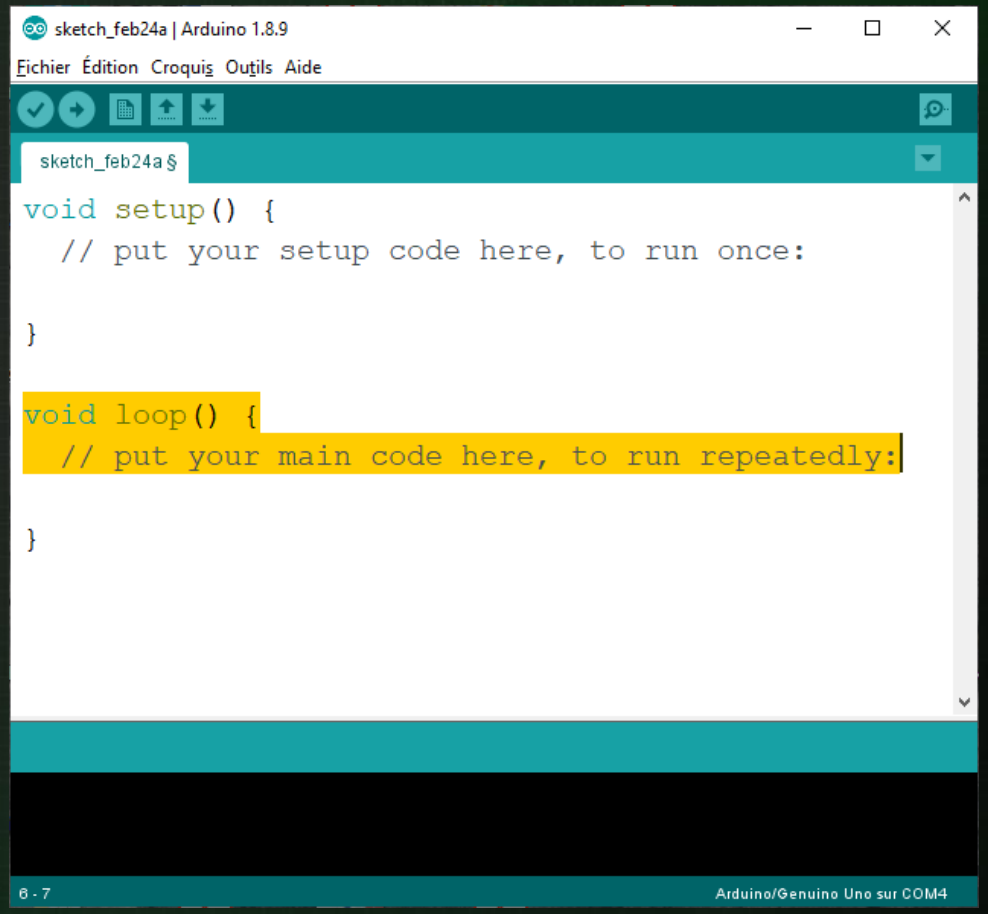
Il existe deux approches de base pour concevoir et implémenter un firmware :

1. Basée sur une procédure conventionnelle «*Super Loop Model* »
2. Basée sur un système d'exploitation embarqué «*OS based Design* »

L’approche Super Loop

- Elle s’applique à des systèmes embarqués pour lesquels le temps de réponse n’est pas très important.
- Le flux d’exécution du code est comme suit :

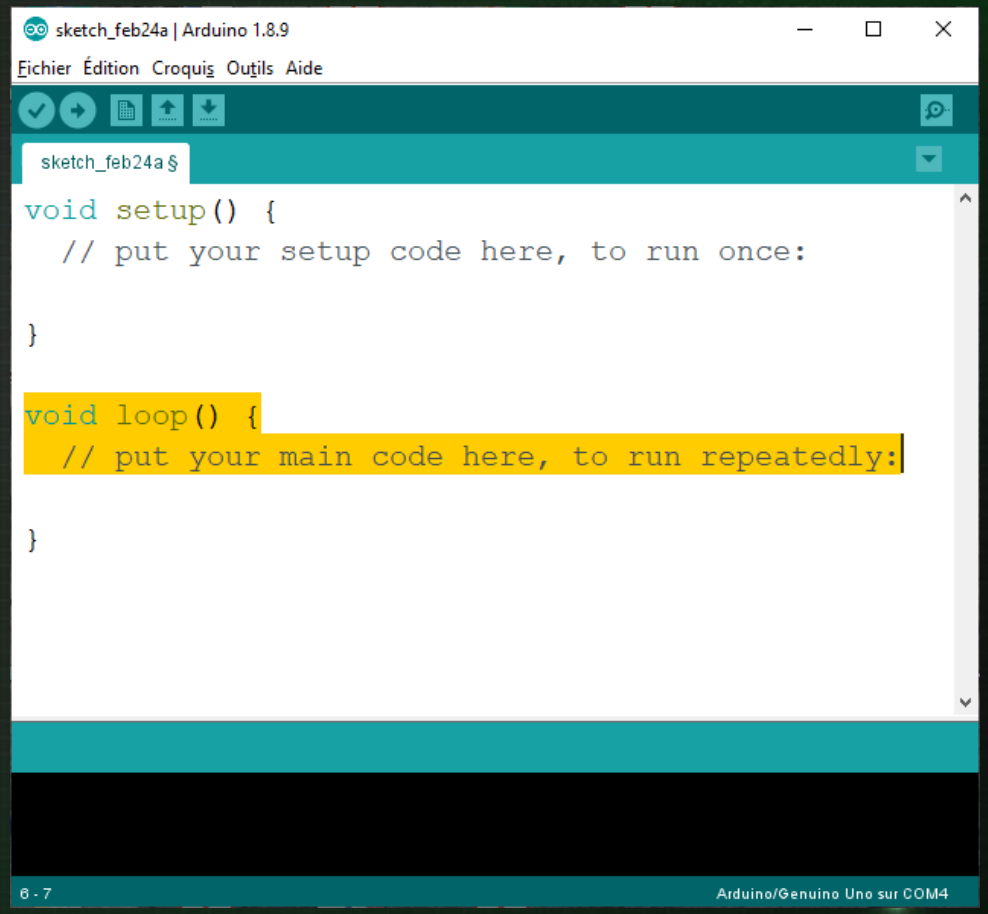
1. Initialiser les différents composants du Hardware (Mémoire, Registres, …etc)
2. Exécuter les tâches, l’une après l’autre.
3. Revenir à la première tâche et refaire la même chose indéfiniment, d’où l’appellation.



Exemple : La Super Loop en C

```
void main ()
{
Configurations ();
Initializations ();
while (1)
{
Task 1 ();
Task 2 ();
:
:
Task n ();
}
}
```

Boucle infinie



Remarques

Pour sortir de cette boucle infinie :

1. A travers un reset (Cela ramène le programme à la boucle principale)
2. Une demande d'interruption (Cela conduit à une suspension temporaire de l'exécution du programme)

L’approche Super Loop

Avantages :

- Ne nécessite pas de système d’exploitation
- Conception simple et directe
- Mémoire réduite

Inconvénients :

- A mesure que le nombre de tâches augmente, la fréquence à laquelle une tâche obtient le temps d'exécution du CPU augmente également (traitement séquentiel)
- Tout problème dans l'exécution d'une tâche peut affecter le fonctionnement du produit. Cela peut être résolu par l’utilisation des timers du WatchDog (Chien de garde).

L’approche basée sur un système d’exploitation

Le produit embarqué contient un système d’exploitation qui peut être soit de type:

1. Un système d’exploitation à temps réel (RTOS)
2. Un système d’exploitation à usage général (GPOS)

L’approche basée sur un système d’exploitation

Le système d'exploitation est responsable de la planification de l'exécution des tâches de l'allocation des ressources entre plusieurs tâches ().

Microsoft Windows XP est un exemple de GPOS pour les systèmes embarqués.

Les stations de jeu et les tablettes sont des exemples de systèmes fonctionnant sur des GPOS.

Windows CE, Windows Mobile, QNX, VxWorks, ThreadX, MicroC/OS-II, Embedded Linux sont des exemples de systèmes d’exploitation en temps réel.

Les téléphones portables et les systèmes de contrôle aérien sont des exemples de systèmes embarqués fonctionnant sur des RTOS.

Les langages de programmation d'un SE

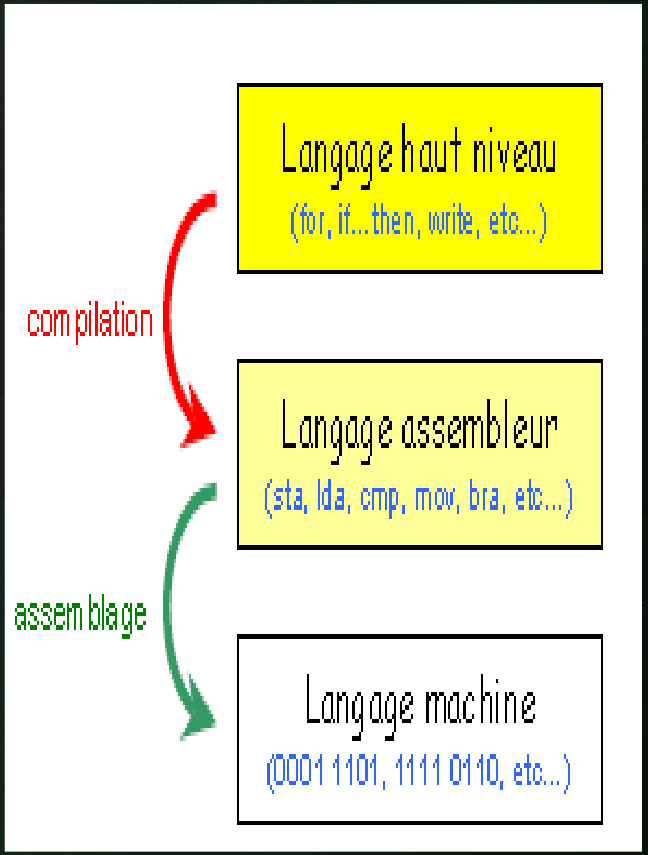
Les langages de programmation de haut niveau sont les langages les plus adaptés à l'homme et aux applications qu'il cherche à développer.

Le programme en langage de haut niveau n'est pas compréhensible par le microprocesseur.

Il faut le compiler pour le traduire en assembleur.

La programmation en assembleur utilise un ensemble d'instructions appelé **jeu d'instructions**.

Chaque microprocesseur à son propre jeu d'instructions.



Les langages de programmation d'un SE

Les micrologiciels (firmwares) utilisés dans la conception des systèmes embarqués peuvent être développés de différentes manièrers :

1. Ecrit dans un Langage Assembleur en utilisant les instructions propres au processeur utilisé
2. Ecrit dans un langage de haut niveau (évolué) tel que :

•Le C embarqué (Un sous-ensemble du C)

•Le C++ embarqué (Un sous-ensemble du C++)

•Tout autre langage évolué (Python, Java,…)

Ceci implique l'utilisation d'un IDE (Integrated Development Environment) contenant un éditeur, un compilateur, un éditeur de lien (linker), un débogueur (debugger), un simulateur, …etc

Les langages de programmation d’un SE

3. Mélange de langage assembleur et de langage de haut niveau :

•Mélanger un langage de haut niveau (exemple C) avec un code en assembleur

•Mélanger un code en assembleur avec du langage de haut niveau,

•Ou un Assembleur en ligne (Inline Assembly) qui est une fonctionnalité de certains compilateurs qui permet d'intégrer du code de bas niveau écrit en langage d'assembleur dans un autre code qui a été compilé à partir d'un langage de niveau haut tel que C.

LANGAGE ASSEMBLEUR

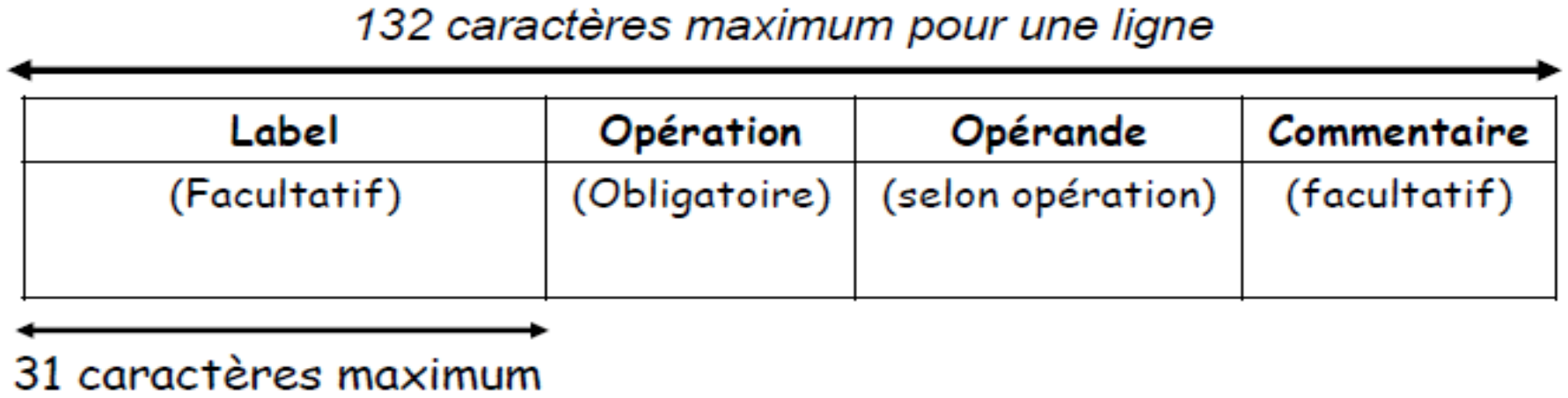
Dans un programme en langage assembleur, les instructions et les opérandes sont représentés par des symboles alphanumériques significatifs et faciles à retenir, appelés mnémoniques.

Il est relativement plus facile d'écrire et d’analyser un programme en langage assembleur que dans le langage machine (une suite de 0 et de 1). Le programme en langage assembleur est traduit en langage machine pour pouvoir être exécuté.

Le module logiciel qui réalise cette traduction s’appelle aussi Assembleur.

Structure d'une instruction assembleur

Une instruction Assembleur est scindée en 4 zones appelés champs :



Exemple :

Etiquette	Mnémonique	Opérande Destination		Opérande Source	Commentaire
Debut :	Mov	AX	,	BX	; (facultatif)

JEU D'INSTRUCTIONS

L'ensemble des instructions d'un langage pris en charge par le microprocesseur comprend généralement cinq groupes de types d'instructions :

1. Instructions de transfert de données

Ces instructions permettent de déplacer des données entre les registres dans le microprocesseur, entre registre et emplacement de mémoire ou entre 2 emplacements mémoire.

2. Instructions arithmétiques et logiques

Ces instructions permettent d’effectuer l’addition, la soustraction, l’incrémentation ou décrémentation des données dans un registre ou dans la mémoire, ainsi que les opérations logiques bit à bit AND, OR, NOT, XOR, comparaison, décalage et rotation des données dans un registre ou en mémoire.

3. Instruction de saut (branchement) :

Les instructions de saut peuvent être conditionnelles ou inconditionnelles, elles incluent aussi,

les appels de sous-programme et leur instruction de retour, ainsi que les routines d’interruption.

Une instruction conditionnelle est exécutée uniquement si un flag est à un état binaire particulier, par exemple si le résultat de la dernière opération arithmétique est nul.

4. Instructions diverses :

Généralement il y a des instructions qui ne sont dans aucun des groupes précédents : manipulation de la pile, modification de la valeur des flags, mise en halte du microprocesseur, instruction "pas d'opération", etc…

Avantages de la programmation en assembleur

1. **Optimisation de la mémoire :** Le développeur connait bien l'architecture du processeur et l'organisation de la mémoire, ce qui permet d'écrire un code optimisé.
2. **Haute performance**
3. **Accès au matériel au plus bas niveau**

Inconvénients de la programmation en assembleur

1. **Temps de développement élevé :** Le développeur a besoin de temps pour apprendre l'architecture du processeur et l'organisation de la mémoire, les différents modes d'adressage et le jeu d'instructions du processeur .
2. **Non portable :** Chaque code en assembleur est propre au processeur utilisé (Si on change de processeur, on change de programme)

Les langages de Haut niveau

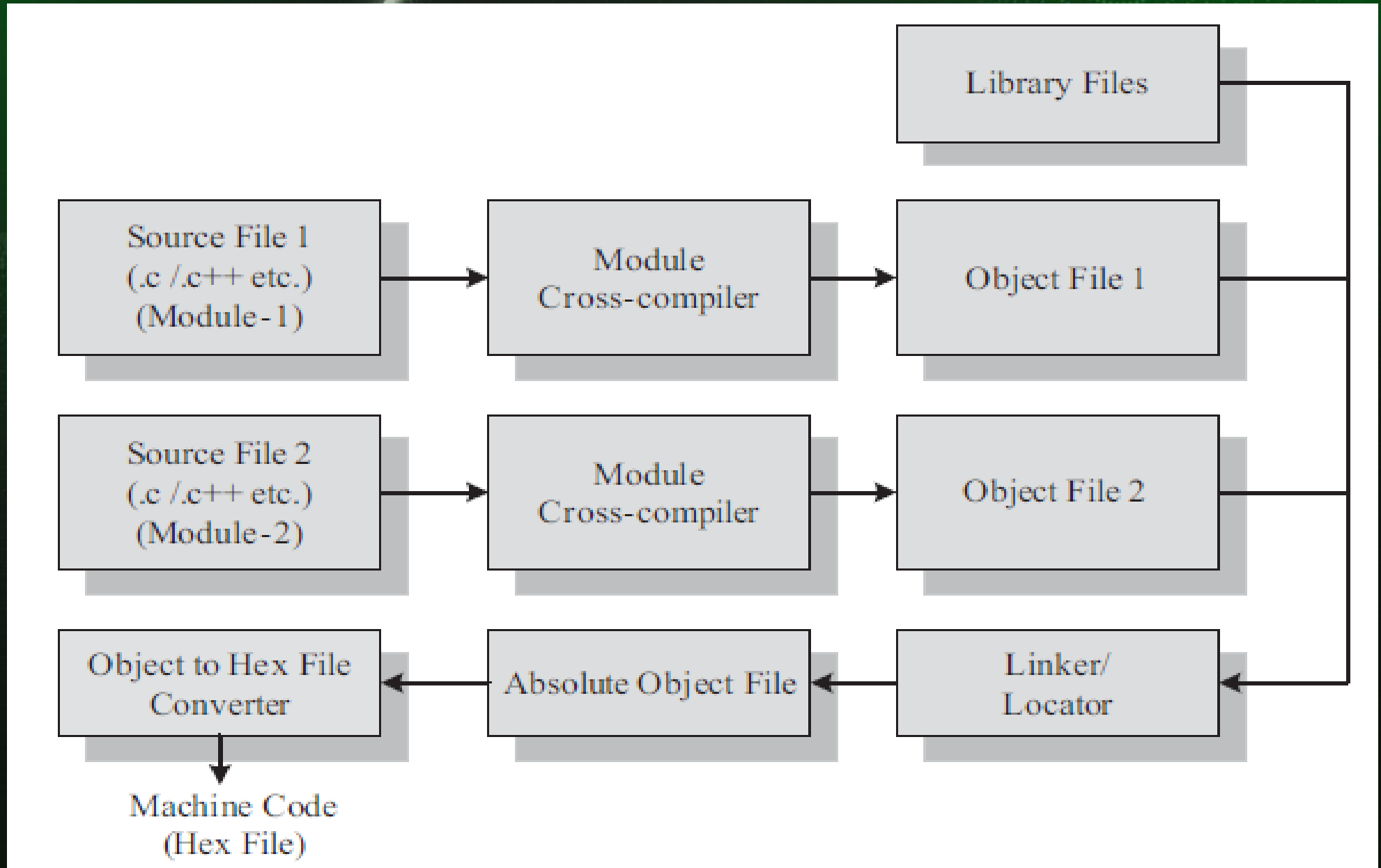
Le firmware embarqué est écrit dans n'importe quel langage évolué tel que C ou C++, …

Un utilitaire logiciel appelé « Cross Compiler» convertit le langage de haut niveau en code machine spécifique au processeur cible.

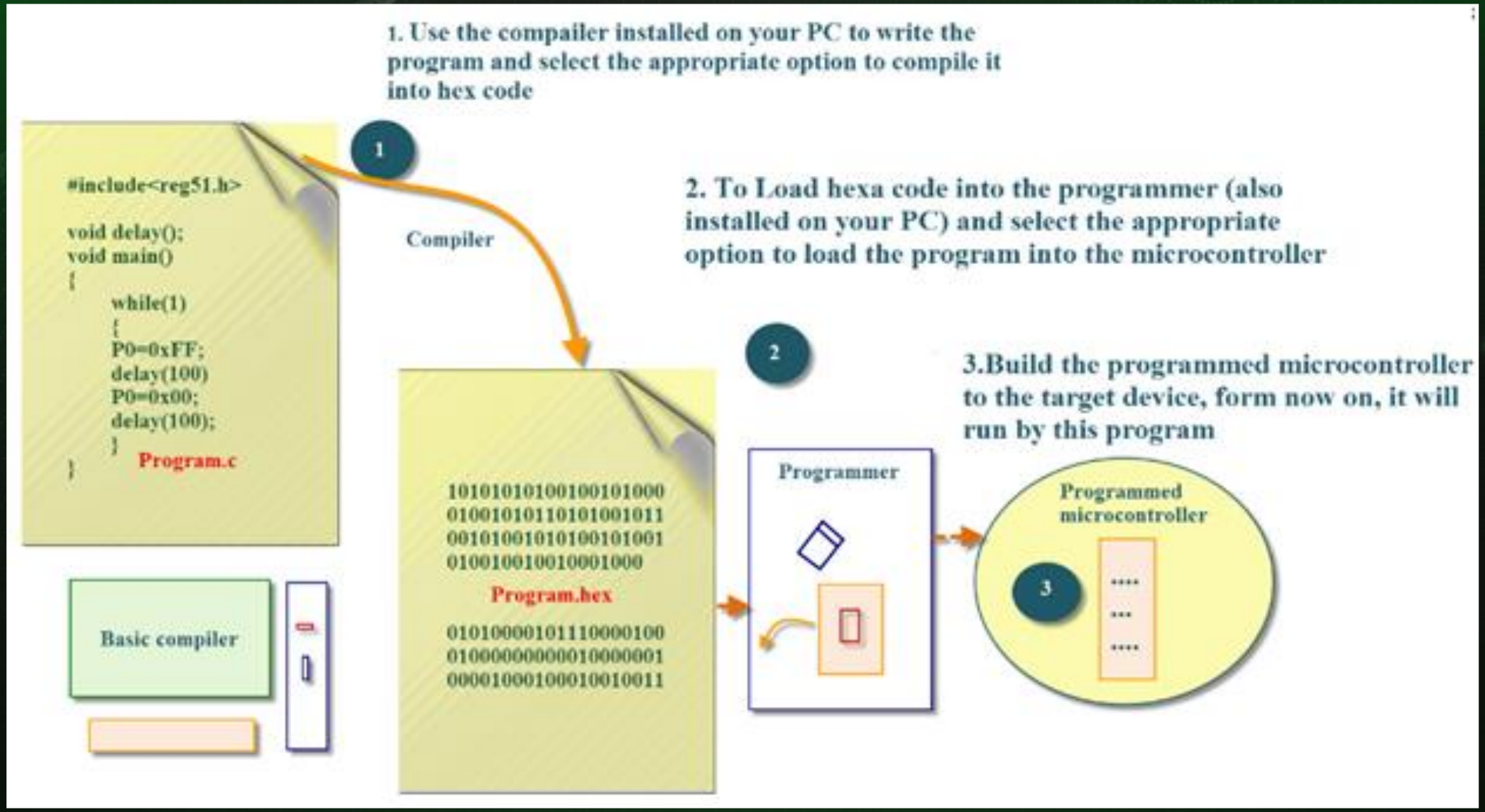
La cross-compilation de chaque module génère un fichier objet correspondant. Le fichier objet ne contient pas l'adresse où le code doit être placé.

Le logiciel appelé Linker est responsable de l'attribution d'une adresse aux fichiers objets pendant le processus de liaison.

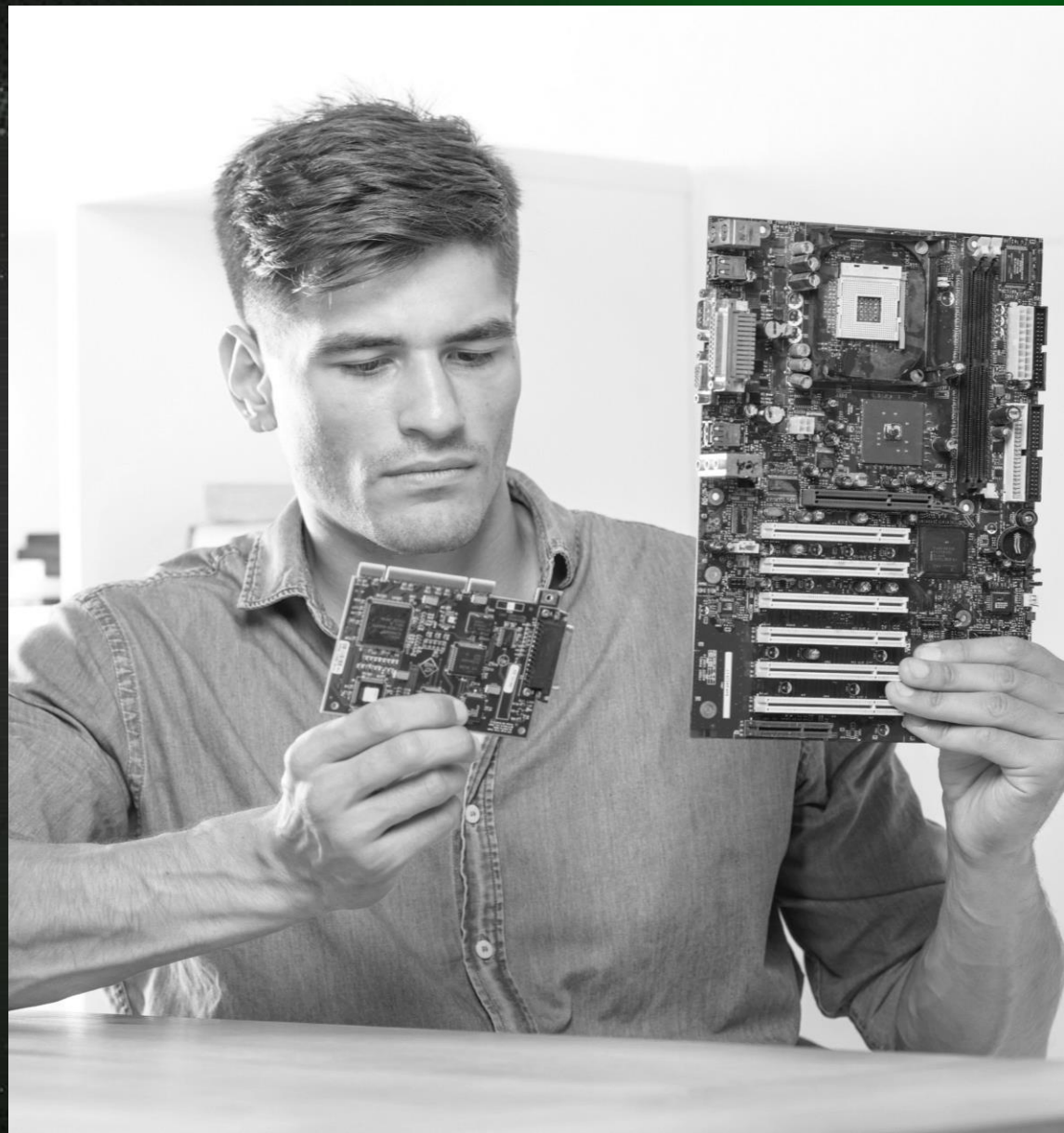
Un logiciel appelé « Convertisseur de fichier objet en fichier hexadécimal » traduit le fichier objet absolu en fichier hexadécimal correspondant (fichier binaire).


Library Files
Source File 1
(.c /.c++ etc.)
(Module-1)
Module
Cross-compiler
Object File 1
Source File 2
(.c /.c++ etc.)
(Module-2)
Module
Cross-compiler
Object File 2
Object to Hex File
Converter
Absolute Object File
Linker/
Locator
Machine Code
(Hex File)

La Programmation des Systèmes Embarqués

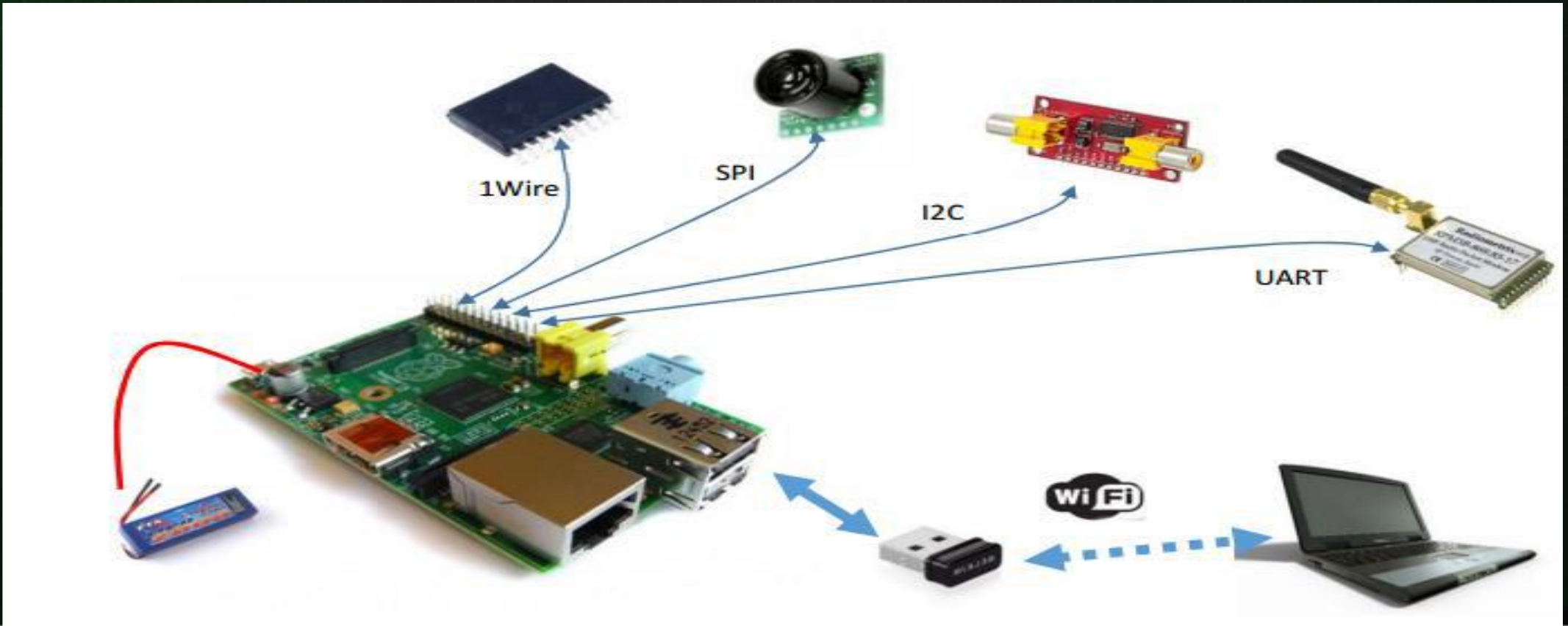


Chapter 3:
Interfaces of an embedded system



LES Interfaces dans les Systèmes Embarqués

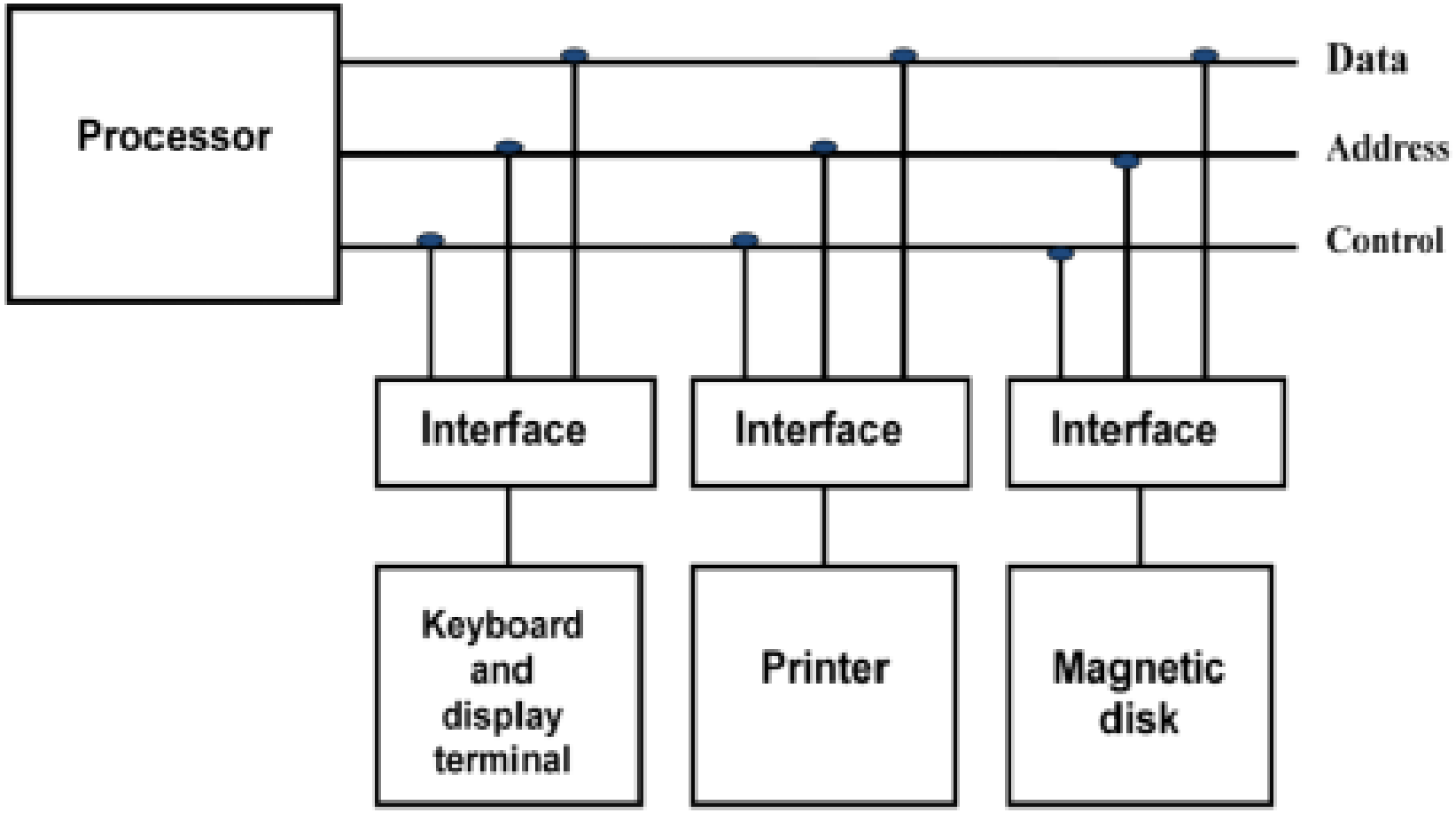
- Les systèmes embarqués sont en interaction constante avec le monde réel.
- Cette interaction a lieu à travers les capteurs et les actionneurs qui sont connectés respectivement aux ports d'Entrée/Sortie du système embarqué.



Les Interfaces dans les Systèmes Embarqués

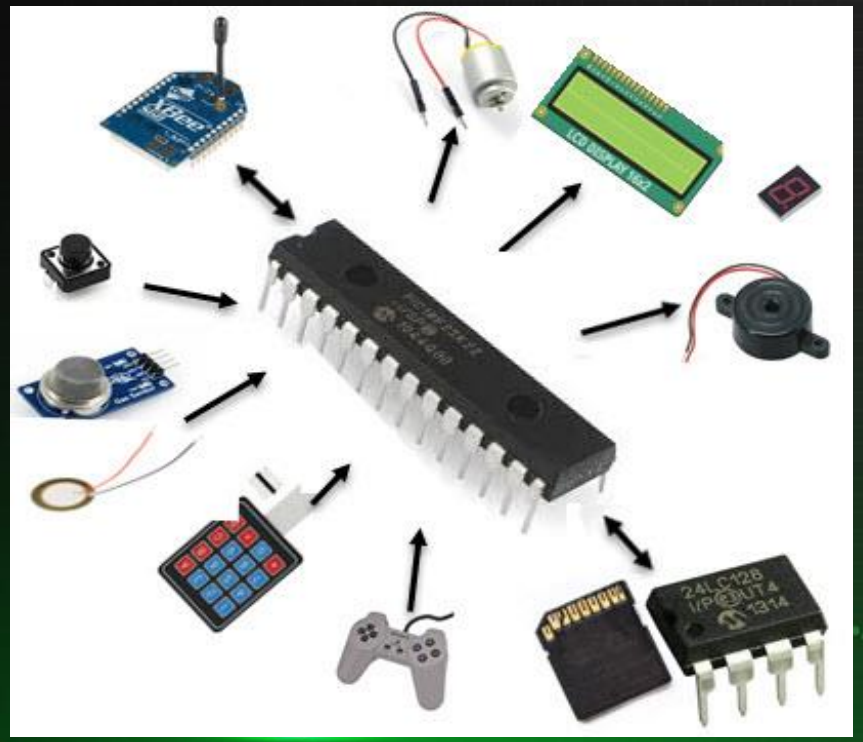
1. Les périphériques sont des appareils électromécaniques et électromagnétiques et le processeur et la mémoire sont des circuits électroniques. Une conversion des valeurs des signaux peut donc être nécessaire.
2. Le taux de transfert de données des périphériques est généralement plus lent que le taux de transfert du CPU et par conséquent, un mécanisme de synchronisation peut être nécessaire.
3. Les codes et les formats de données dans les périphériques diffèrent du format des mots dans la CPU et la mémoire.

Les Interfaces dans les Systèmes Embarqués



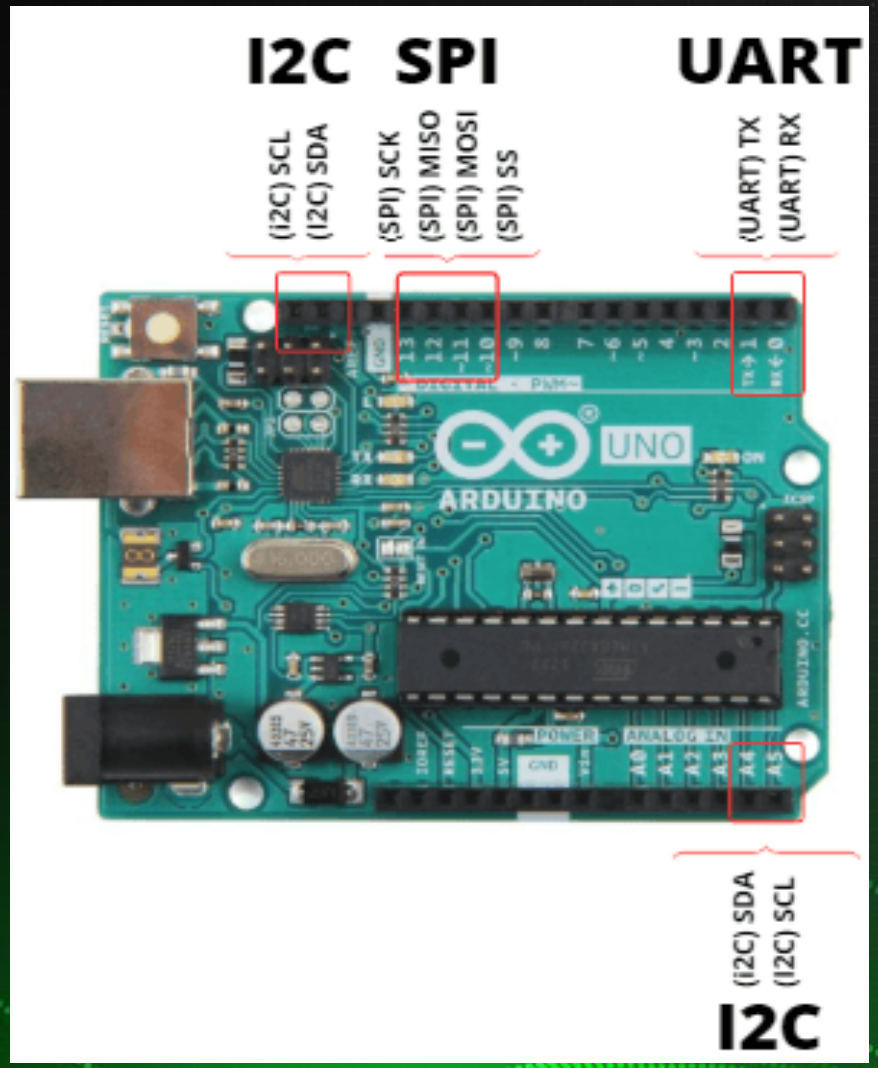
Connection of I/O bus to input-output devices

Les Interfaces dans les Systèmes Embarqués



- Les modes de fonctionnement des périphériques sont différents les uns des autres et doivent être contrôlé pour ne pas perturber le fonctionnement des autres périphériques connectés au processeur.
- Pour résoudre ces problèmes de synchronisation, les systèmes embarqués incluent des composants entre le CPU et les périphériques pour superviser tous les transferts d’entrée/sortie.
- Ces composants sont appelés **Unités d’Interface** car ils font l'interface entre le processeur et les périphériques.
- Ces Interfaces peuvent être Numériques ou Analogiques

Quelques Exemples d’Interfaces d’Entrée/Sortie



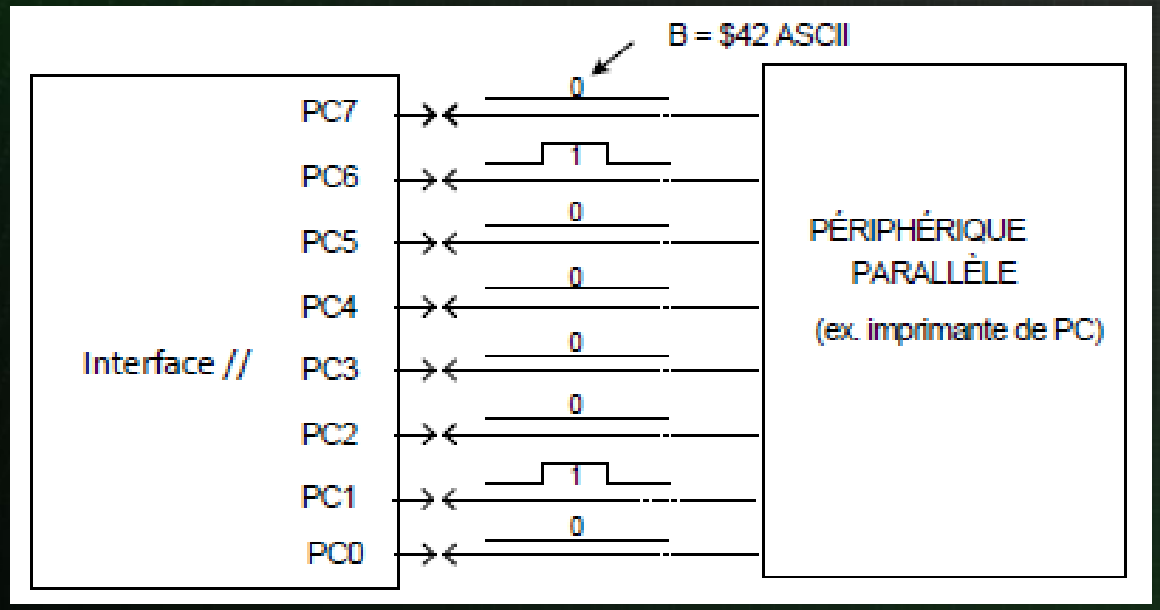
- Entrées/Sorties numériques : Ces interfaces traitent des informations binaires et peuvent être utilisées pour lire des commutateurs (Switches), des boutons ou contrôler des LEDs.
- Entrées/sorties analogiques : les interfaces analogiques gèrent des signaux continus et sont utilisées pour des tâches telles que la lecture des capteurs de température, ou le contrôle de moteurs.
- Ports de communication série : Les systèmes embarqués incluent souvent des ports série comme UART, SPI, I2C pour la connexion d'autres appareils, modules ou microcontrôleurs

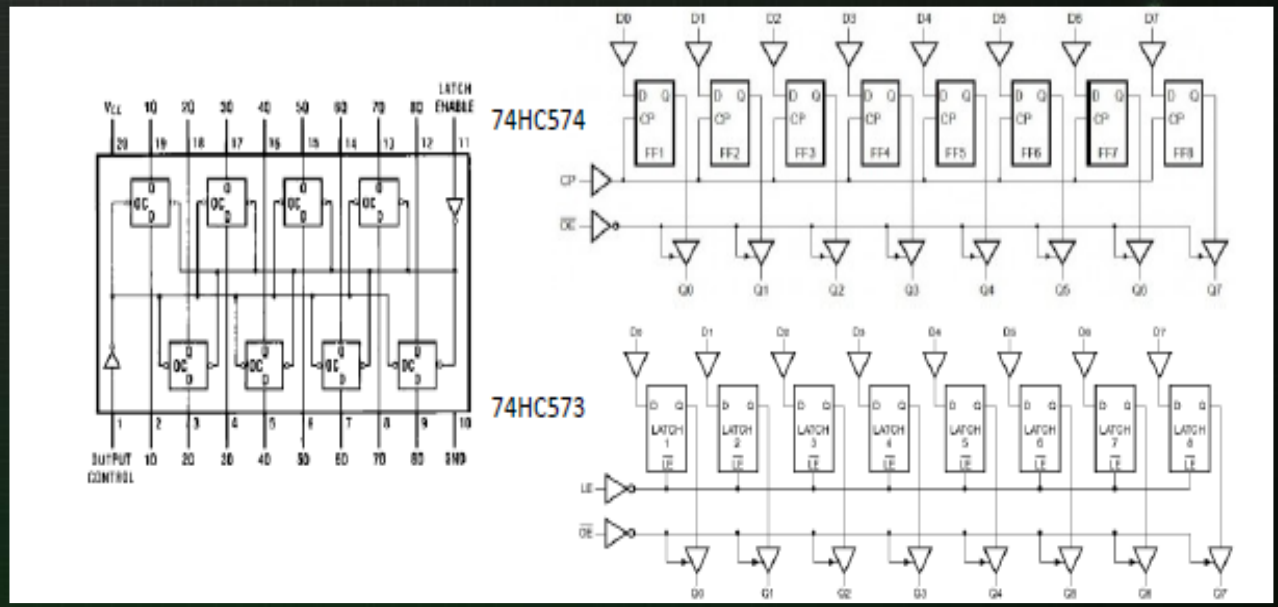
Les Lignes d’Entrée/Sortie (GPIO:General Purpose Input Output)

C’est un ensemble de lignes binaires programmables.

La direction des données est programmable, individuellement ou en groupe

Réalisées à l’aide de tampons, des bascules synchrones, ou de ports de µC dédiés



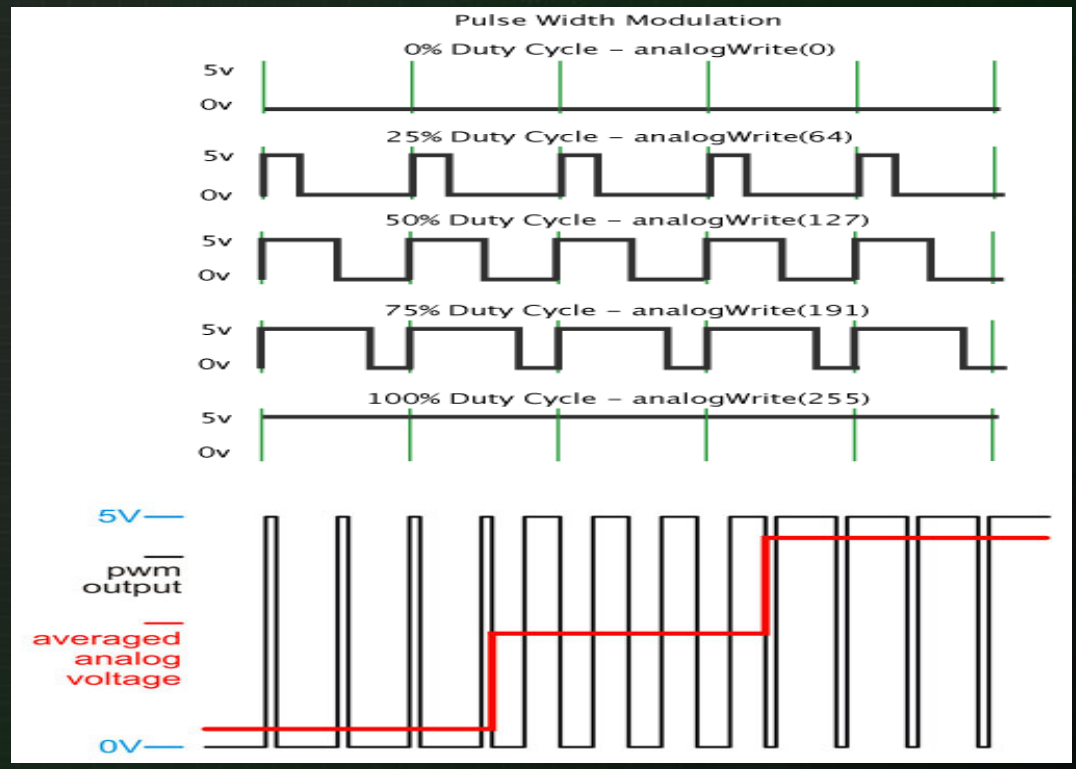


Les Interfaces Analogiques (Pulse Width Modulation : *PWM*)

La modulation à largeur d'impulsion est un moyen d'obtenir des résultats analogiques à l'aide de moyens digitaux.

Génère des impulsions avec des cycles de travail spécifiques.

Peut être utilisée pour piloter un moteur électrique (Plus simple que le convertisseur numérique-analogique DAC).



Les Interfaces de Communication dans les Systèmes Embarqués

- Dans le systèmes embarqués, les interfaces de communication peuvent être vues selon deux perspectives différentes :

1. Une interface de communication Embarquée (Onboard communication Interface)

Exemple : I2C, SPI, UART

2. Une Interface Externe : peut être câblée ou non câblée.

Exemple : Bluetooth, USB, …etc

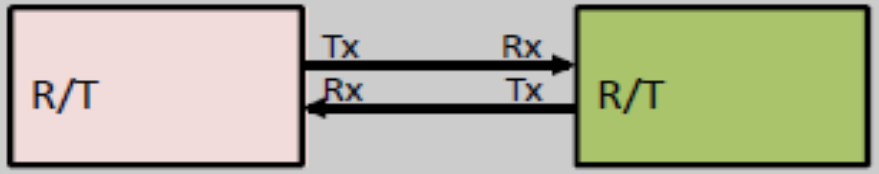
Onboard communication Interface (L’Interface Embarquée)

- Elle peut être classée en :

1. Un Bus **I2C** (Inter Integrated Circuit)
2. Un Bus **SPI** (Serial Periphera Interface)
3. UART (Universal Asynchronous Receiver Transmitter)
4. L’Interface câblée
5. L’Interface parallèle

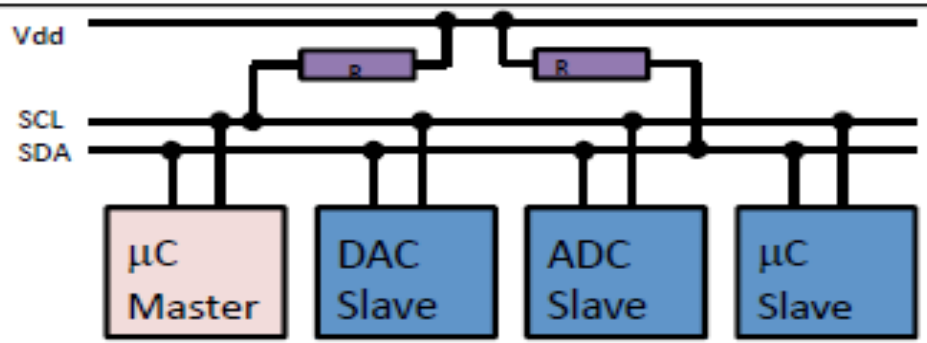
Protocoles sériels communs

- UART
  - Universal Asynchronous Receiver/ Transmitter
  - Full duplex



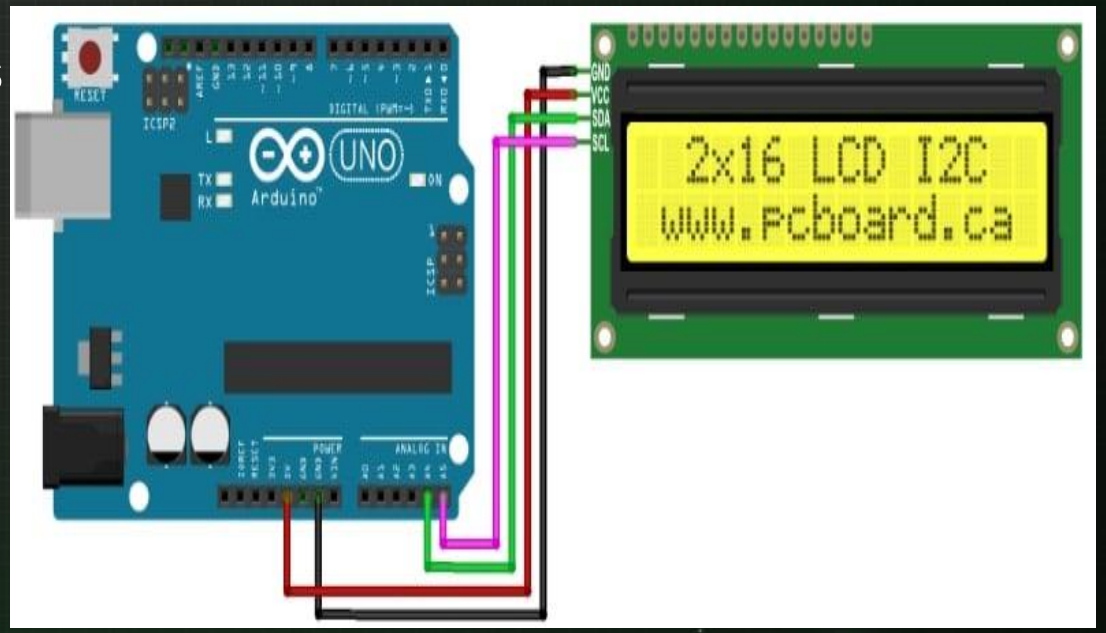
- SPI
  - Serial Peripheral Interface
  - Single Master/Single Slave

- I2C
  - Inter-Integrated Circuit Interface
  - Single Master/Multiple Slaves



L’Interface I2C (Inter Integrated Circuit)

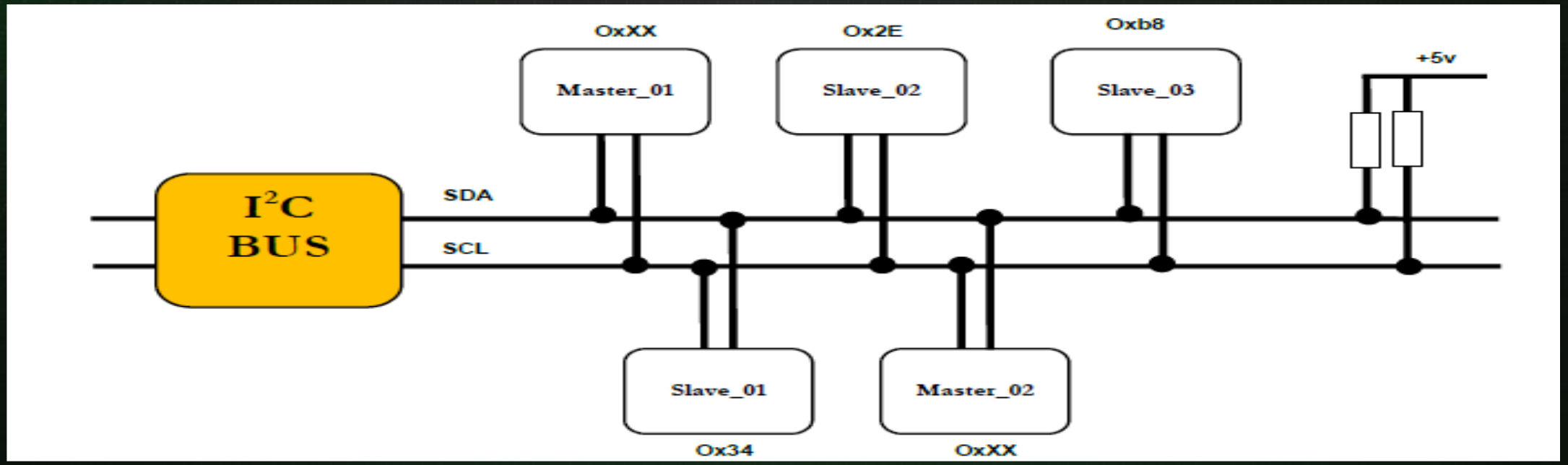
- Une ligne bidirectionnelle pour les données et une ligne d’horloge ;
- Un maître et un ou plusieurs esclaves ;
- Le maître contrôle l’horloge et les signaux de départ et d’arrêt des transferts
- Chaque esclave possède une adresse unique (7 ou 10 bits) et réagit uniquement lorsque concerné.



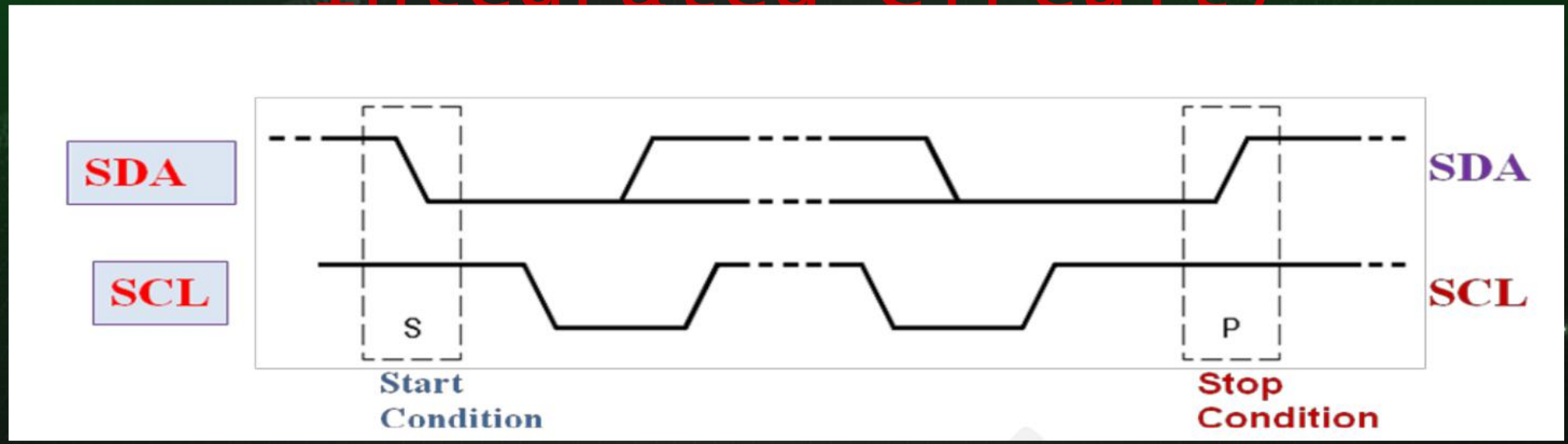
L'Interface I2C (Inter Integrated Circuit)

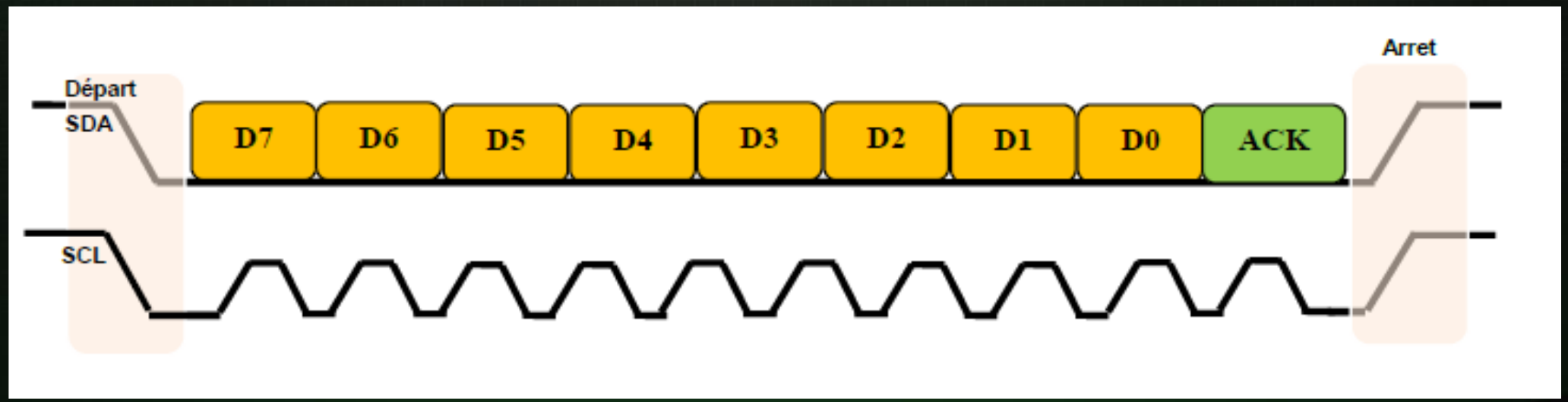
– SDA : Serial DAta ; va dans les deux sens

– SCL : Serial CLock ; habituellement sous le contrôle d'un maître



L'Interface I2C (Inter Integrated Circuit)



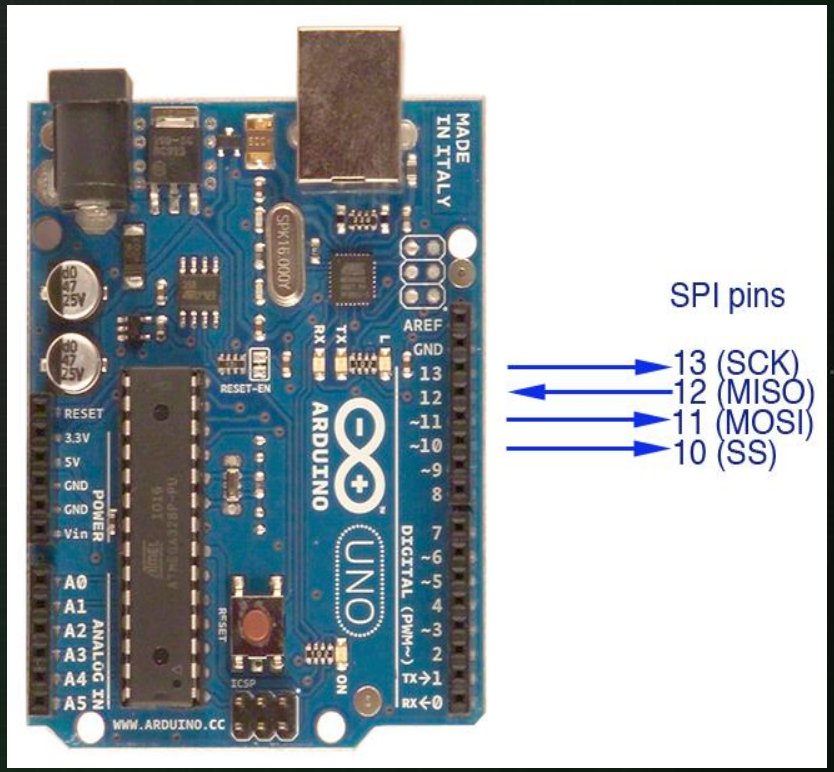


L’Interface SPI (Serial Peripheral Interface)

Est un bus d'interface série à 4 fils bidirectionnel synchrone en duplex intégral.

Le concept de SPI est introduit par Motorola.

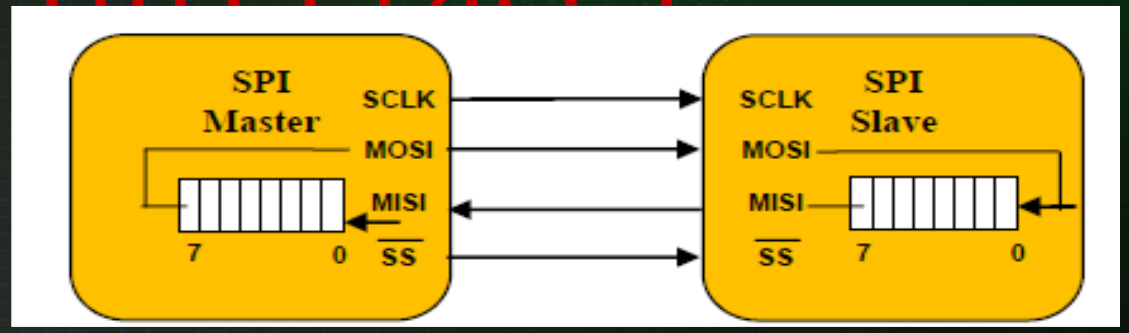
SPI est un système multi-esclave à maître unique.

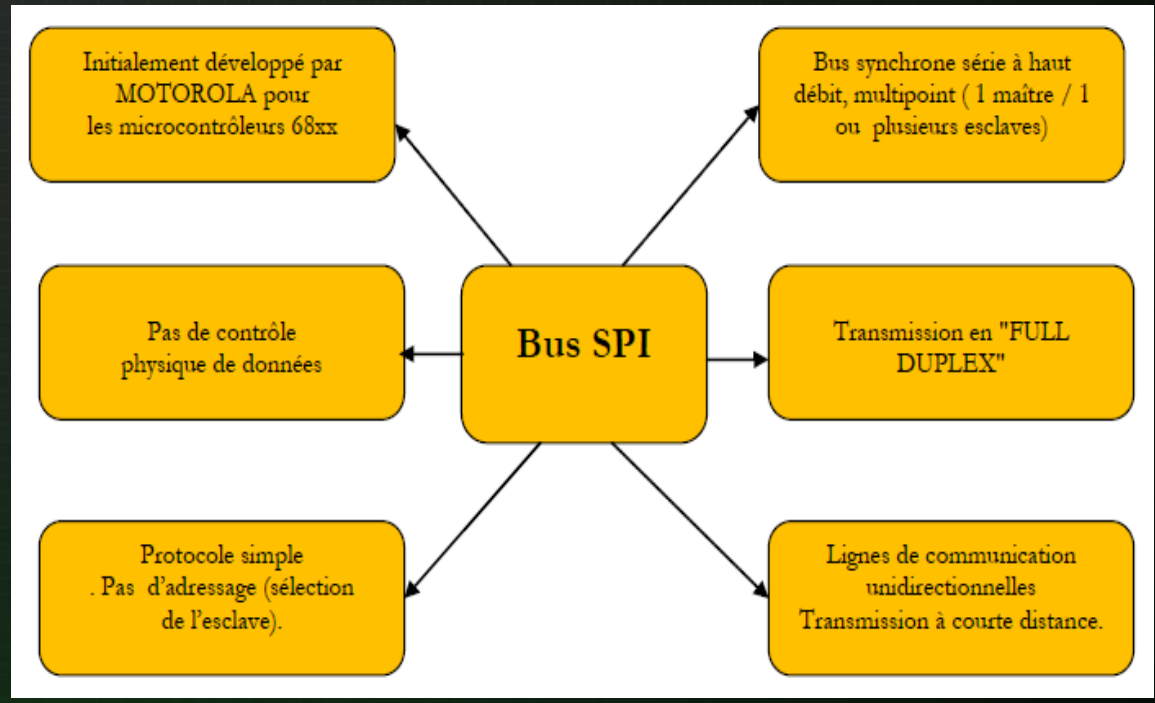


L’Interface SPI (Serial Peripheral Interface)

SPI utilise 4 lignes qui sont :

1. **MOSI** : Ligne transportant les données du maître vers l‘élément esclave.
2. **MISO**: Ligne transportant les données de l'esclave vers l'appareil maître.
3. **SCLK** : Ligne transportant le signal D’horloge
4. **SS** : Ligne pour sélectionner l’élément Esclave (active au niveau Bas)

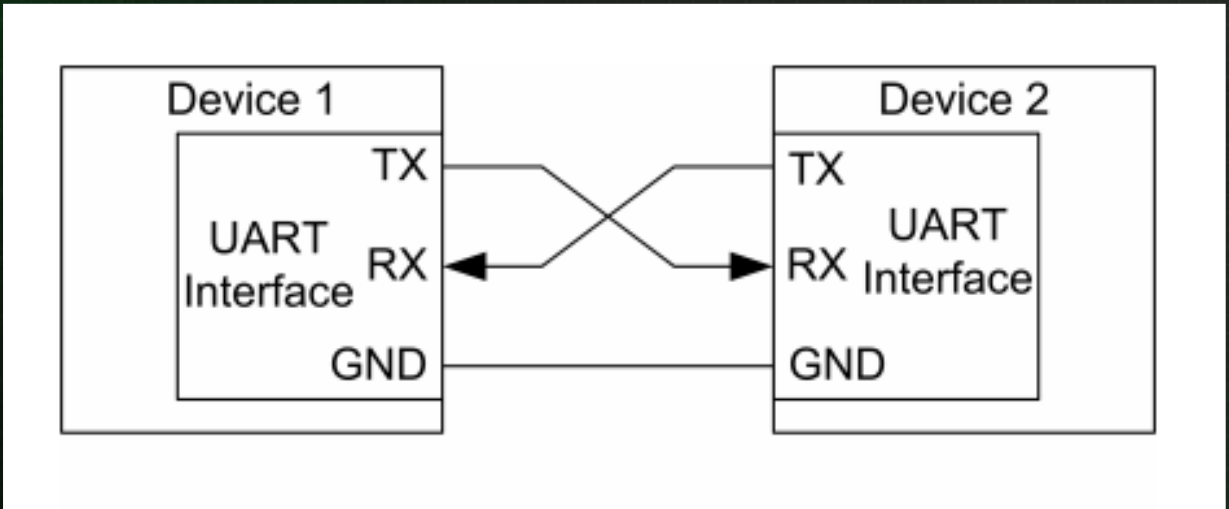


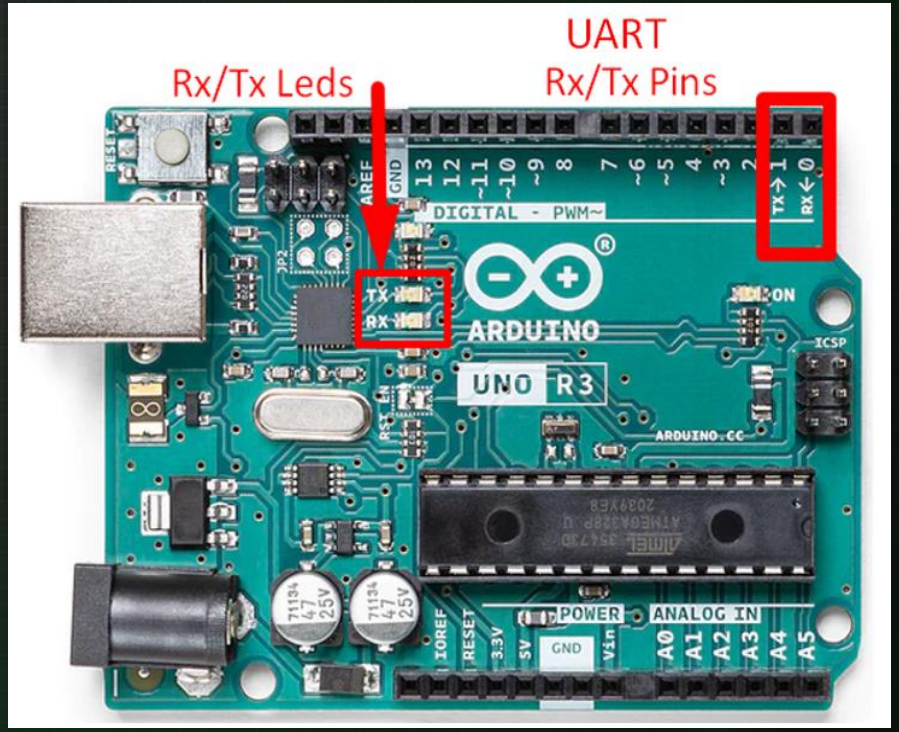


L’Interface UART (Universal Asynchronous Receiver Transmitter)

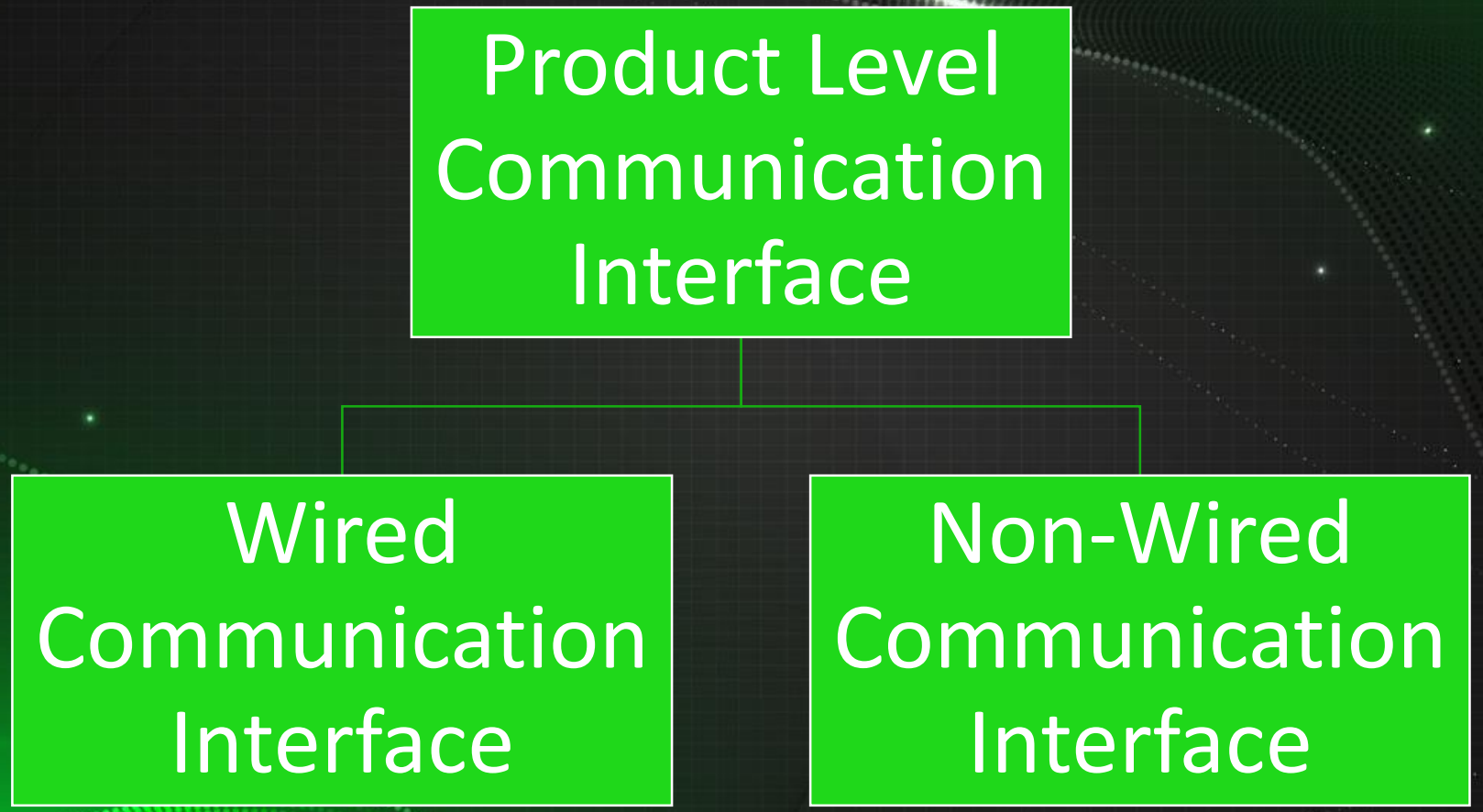
C’est un émetteur-récepteur asynchrone universel.

Il transforme les données envoyées en parallèle par le processeur et les fait passer en série

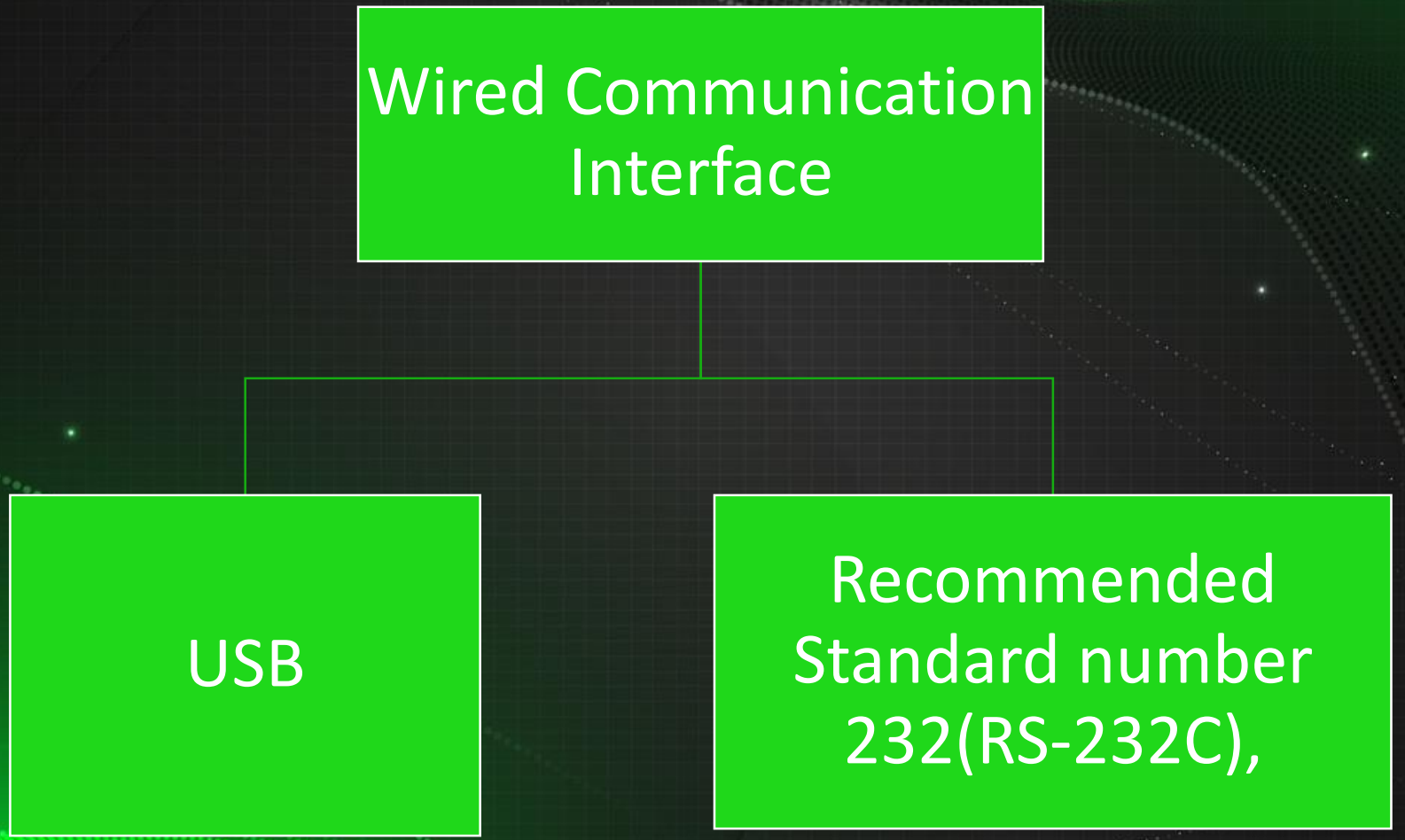




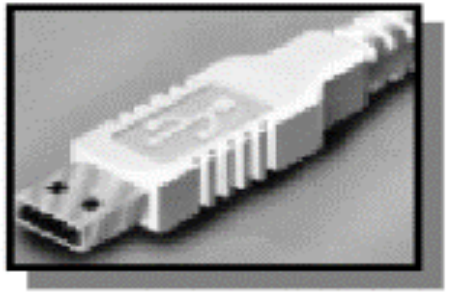
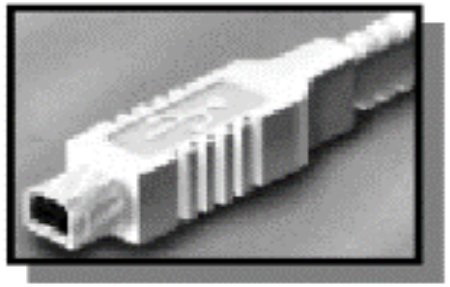
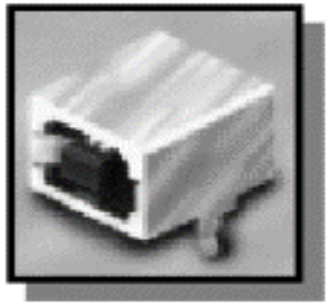
Interface de Communication Externe



Interface de Communication câblée



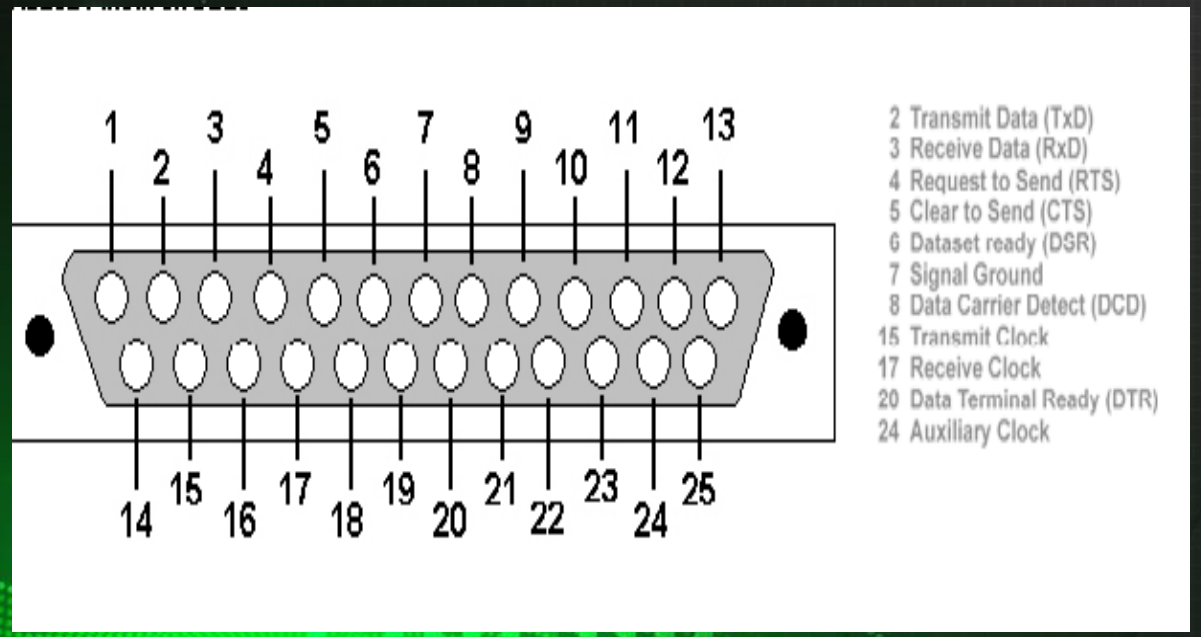
Universal Serial Interface (USB)

Series "A" Connectors	Series "B" Connectors
◆ Series "A" plugs are always oriented **upstream** towards the *Host System*	◆ Series "B" plugs are always oriented **downstream** towards the *USB Device*
 **"A" Plugs** (*From the USB Device*)	 **"B" Plugs** (*From the Host System*)
"A" Receptacles (*Downstream Output from the USB Host or Hub*) 	**"B" Receptacles** (*Upstream Input to the USB Device or Hub*) 

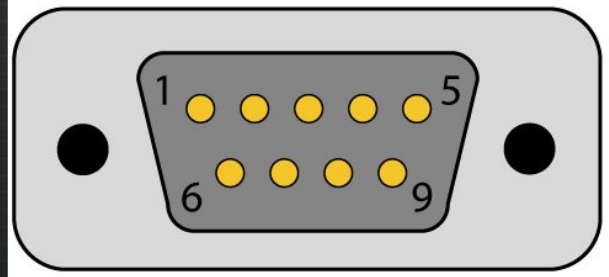
RS-232C

Il admet deux types de connecteurs :

1. DB-25 : Le connecteur à 25 broches
2. DB-9 : Le connecteur à 9 broches



DB9M Connector



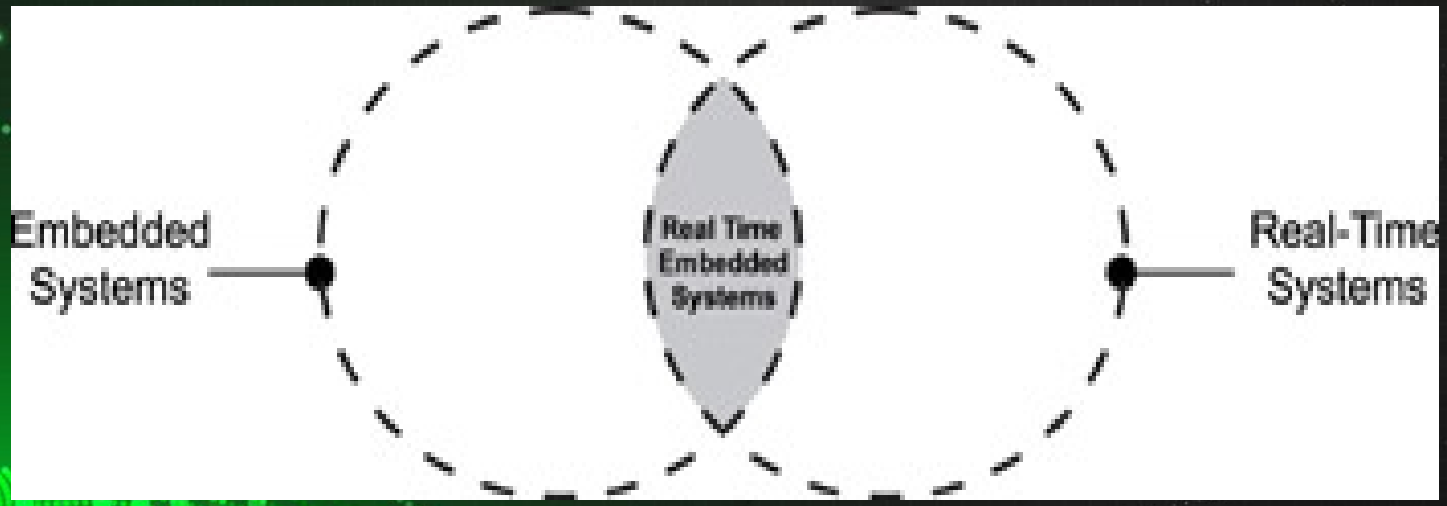
RS232 Pin Out

Pin #	Signal
1	DCD
2	RX
3	TX
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

Chapter 4: Real Time Embedded Systems and RTOS

Qu’est-ce qu’un Système Temps réel (Real time System)

- Un système est dit **temps réel** s’il doit terminer son travail et fournir ses services à temps ⟹ Il dépend des contraintes de temps.
- Dans leur forme la plus simple, les systèmes en temps réel peuvent être définis comme des systèmes qui répondent à des évènements externes à temps.



Qu’est-ce qu’un Système Temps réel (Real time System)

- Nous pouvons extraire deux caractéristiques essentielles des systèmes temps réel :

1. Les systèmes en temps réel doivent produire des résultats de calcul logiques (Exactitute fonctionnelle ou functional correctness),
2. et que ces calculs doivent être effectués dans un délai prédéfini (Exactitude en temps ou timing correctness).

Remarque : **Dans certains systèmes temps réel, l'exactitude fonctionnelle est parfois sacrifiée au profit du timing.**

Types des Systèmes Temps réel

Il existe deux types :

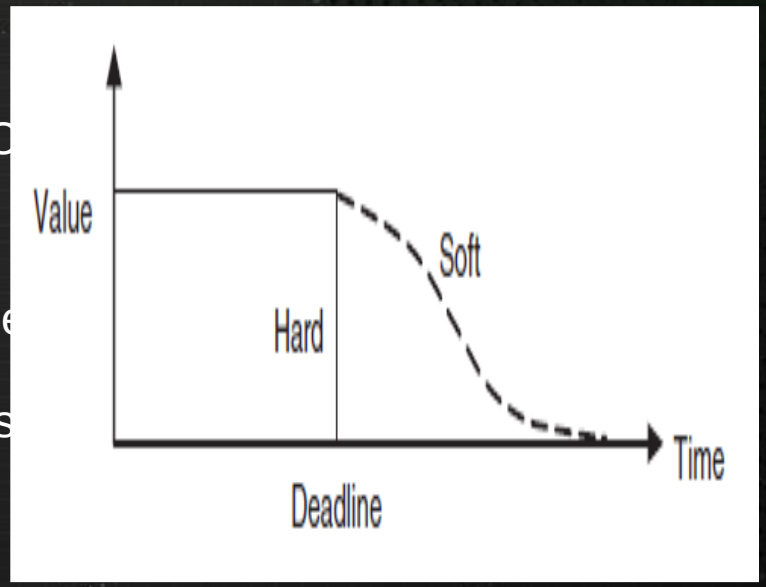
1. Systèmes temps réel souples (Soft real time systems):

- les tâches sont exécutées le plus rapidement possible.
- L'achèvement tardif des travaux n'est pas souhaitable mais pas fatal
- les performances du système se dégradent à mesure que les tâches ne respectent pas les échéances
- La majorité des OS sont des systèmes souples.

2. Systèmes temps réel durs (Hard real time systems): C systèmes

exigent le respect strict de leurs contraintes de temps, car le

non-respect du temps de réponse peut entraîner de graves conséquences.



Real-Time Operating Systems (RTOS)

- Le cœur de nombreux systèmes embarqués est le RTOS.
- Un RTOS est un système d'exploitation qui prend en charge la construction d'applications qui doivent répondre à des contraintes de temps réel en plus de fournir des résultats de calcul logiquement corrects.
- Il fournit des mécanismes et des services pour effectuer la planification des tâches en temps réel, la gestion des ressources et communication inter-tâches.

Le principe du système d’exploitation

- Un système d'exploitation (OS) est le logiciel qui se situe entre le hardware et les applications exécutées.
- Un système d'exploitation est un gestionnaire de ressources (Le processeur, la mémoire, les contrôleurs d'E/S, les disques et d'autres périphériques tels que les terminaux et les réseaux).
- Un système d'exploitation est composé de plusieurs composants logiciels.
- Les composants principaux dans le système d'exploitation forme son noyau (Kernel). Le noyau fournit le niveau le plus élémentaire de contrôle sur tous les périphériques matériels

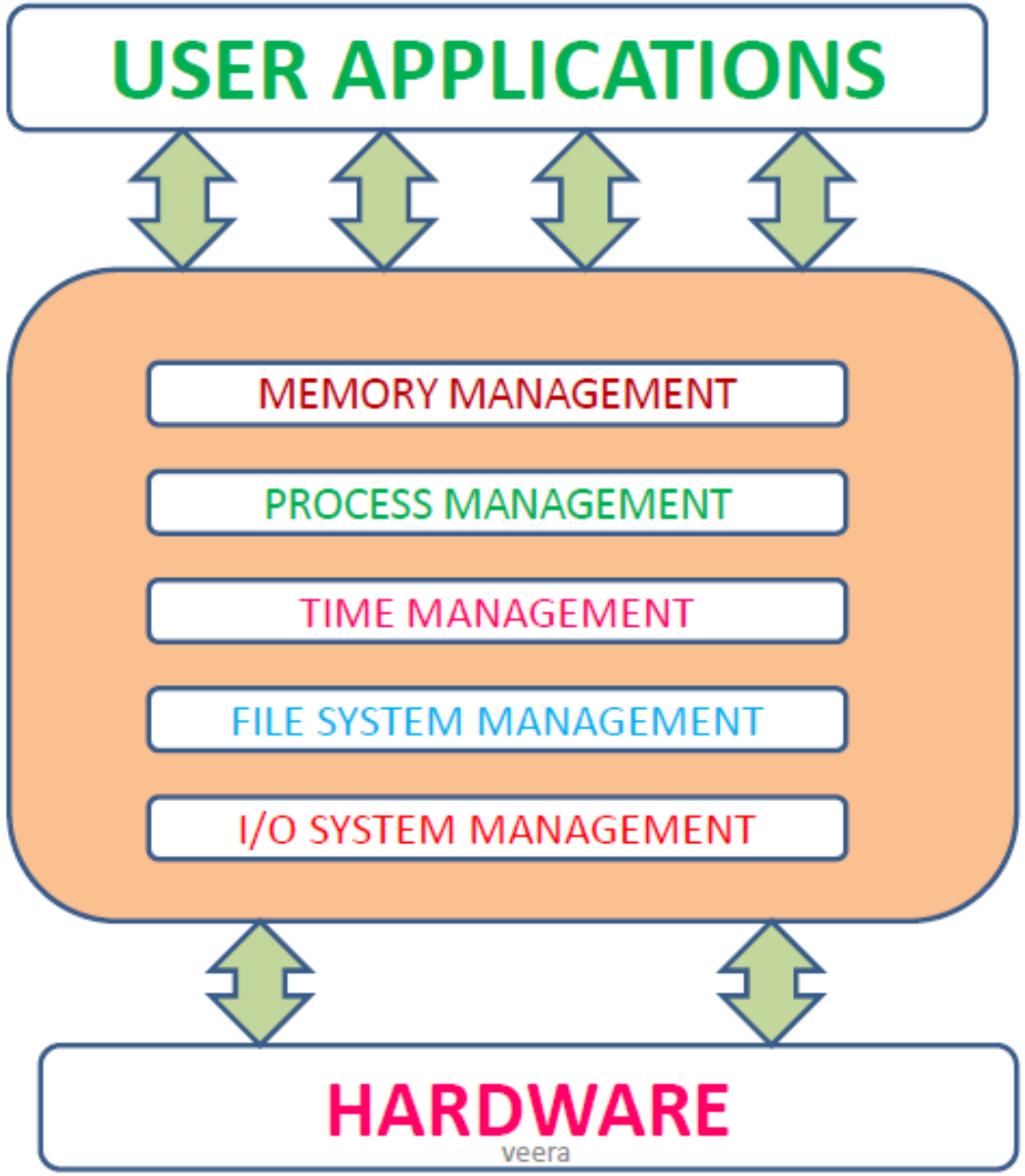
Le principe du système d’exploitation

- Les principales fonctions d'un système d'exploitation sont

a) Rendre le système facile à utiliser,

b) Organiser et gérer le système et les ressources de manière efficace et correcte.

OS components or Architecture



Le noyau (The Kernel)

- Le noyau est le cœur du système d'exploitation
- Il est responsable de la gestion des ressources et de la communication entre le matériel et les autres services du système.
- Le noyau agit comme une couche d'abstraction entre les ressources du système et les applications de l’utilisateur.
- Le noyau contient un ensemble de bibliothèques et de services.

Types de Systèmes d'Exploitation (OS)

- Les OS sont classés en différents types :

1. Les GPOS (General Purpose OS)
2. Et les RTOS (Real Time OS)

Les GPOS

- Le noyau d'un GPOS est plus généralisé et contient toutes sortes de services requis pour exécuter des applications génériques
- Les GPOS sont souvent assez non déterministes dans le comportement
- Leurs services peuvent injecter des retards aléatoires dans le logiciel d'application et peut ralentir la réponse d'une application à des moments inattendus
- Windows XP/MS DOS, etc. sont des exemples de GPOS

- Pour un GPOS, le noyau contient différents services pour effectuer les tâches suivantes :
  - Process management
  - Memory (primary(RAM) and secondary) management
  - File system management
  - I/O system(Device) management
  - Protection systems
  - Interrupt handling

Les RTOS

- Les RTOS impliquent un comportement déterministe, cela signifie que les services du système d'exploitation utilisent uniquement des durées connues, quel que soit le nombre de prestations.
- Le RTOS décide quelles applications doivent s'exécuter dans quel ordre et combien de temps faut-il attribuer pour chaque application.
- WindowsCE QNX, VxWorks MicroC /OS I I, etc. sont des exemples de RTOS

Le Noyau Temps Réel

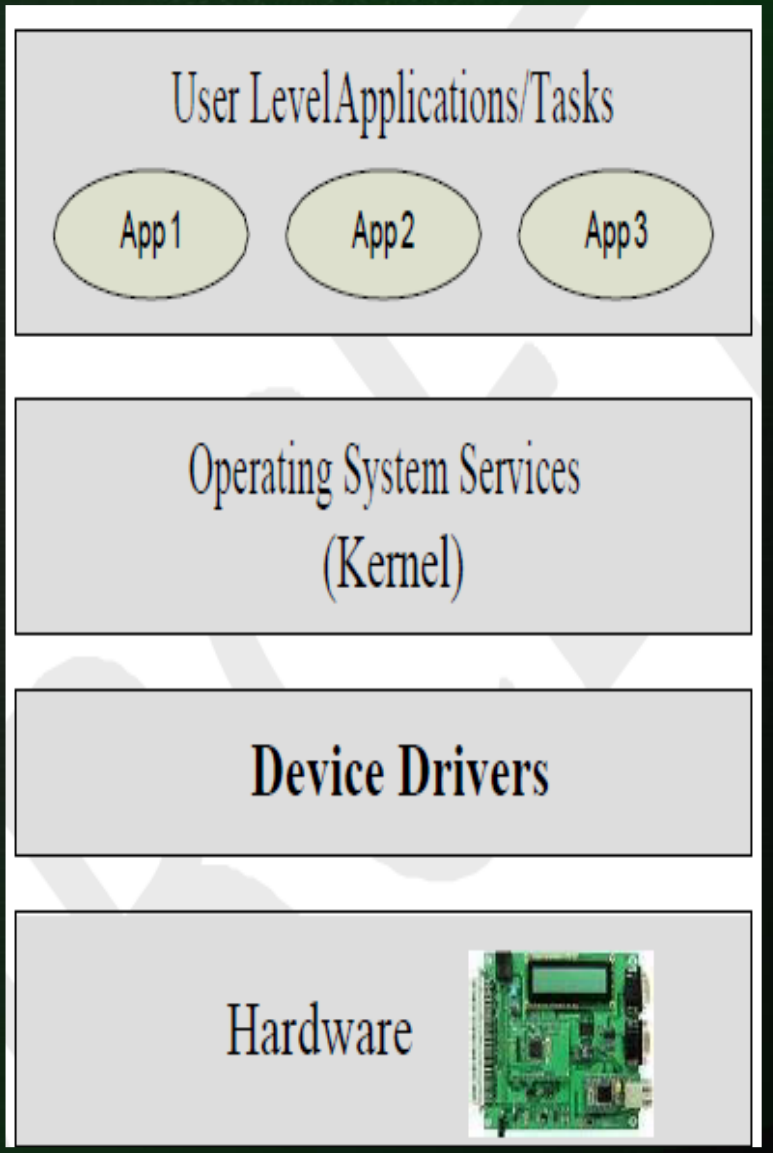
- Le noyau d'un système d'exploitation temps réel est appelé noyau temps réel (RTK).
- le RTK est hautement spécialisé et ne contient que l'ensemble minimal de services requis pour exécuter les.
- Les fonctions de base d'un noyau temps réel sont :

Rôle du RTK

a) Gestion des tâches/process
b) Planification des tâches/process (Scheduling)
c) Synchronisation des tâches/process
d) Gestion des erreurs/exceptions
e) Gestion de la mémoire
f) Gestion des interruptions
g) Gestion du temps

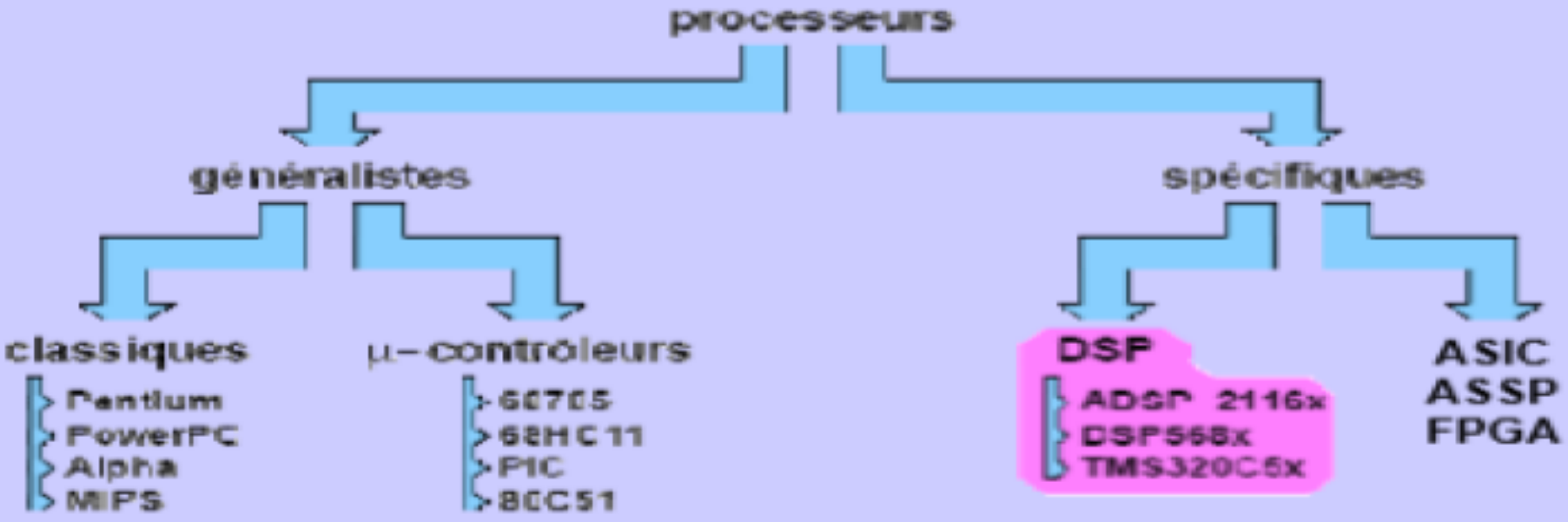
Pilotes de périphériques

- Le pilote de périphérique est un logiciel qui sert de pont entre le système d'exploitation et le matériel (Le Hardware).
- Les applications communiquent avec le noyau du système d'exploitation pour tous les échanges d'informations nécessaires, y compris la communication avec les périphériques.



Chapter 5: CPLD and FPGA in an Embedded System

• Classification des processeurs



spécialisation

ASIC : Application Specific Integrated Circuit
ASSP : Application Specific Standard Product
FPGA : Field Programmable Gate Array

Source : d'après [1]

Années 70 : Apparition des composants programmables *par l'utilisateur.* Ce sont des composants entièrement configurables par programmation (Ils furent introduits sur le marché par la société MMI sous l'appellation PAL (Programmable Array Logic).

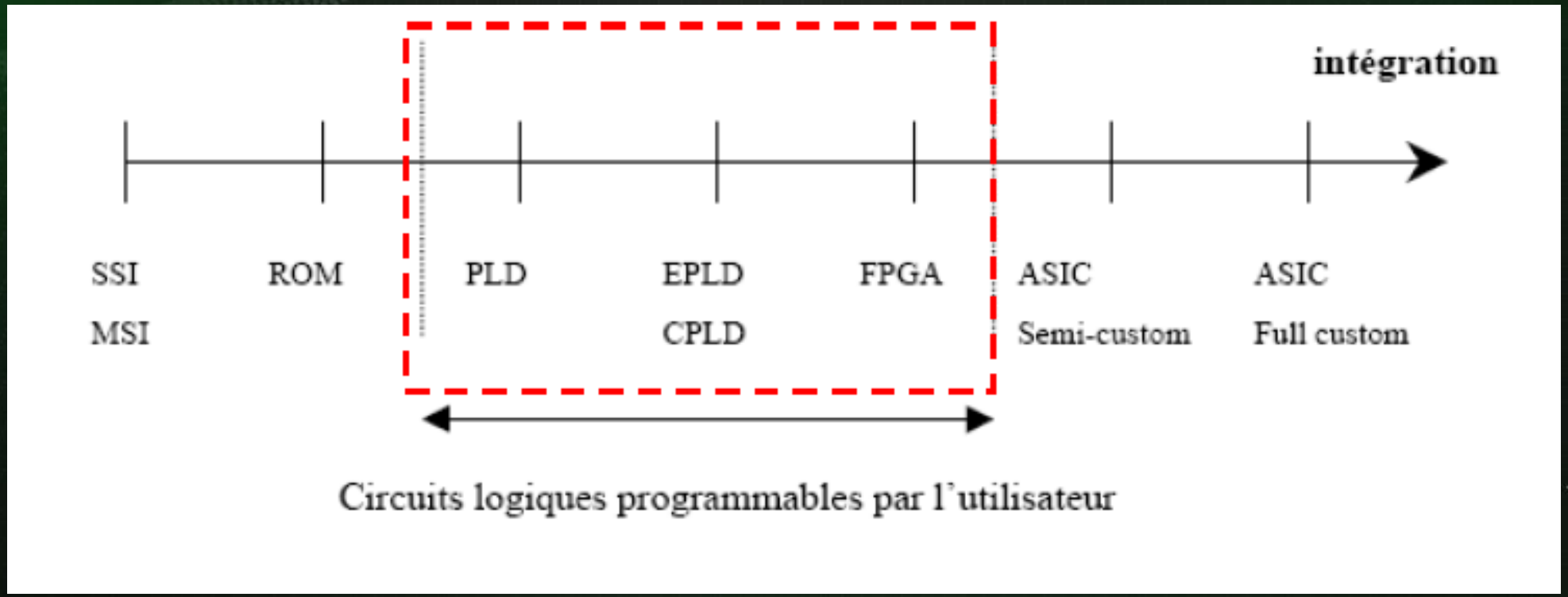
La première génération de ces composants permettait uniquement *la réalisation de fonctions combinatoires.* Très rapidement, l'apparition de composants intégrant des bascules D (en plus de réseau combinatoire) a permis la réalisation de systèmes synchrones. Ce type de composants de faible densité est aujourd'hui désigné sous l'appellation PLD (Programmable Logic Device).

Classification des PLD

Les PLD sont des circuits qui peuvent être configurés par l'utilisateur pour réaliser un grand nombre de fonctions logiques. Ces circuits disposent des entrées et des sorties programmables qui les rendent flexibles et adaptables à différentes applications.

Le nombre d'entrées, de sorties, de connexions programmables et le niveau d'intégration conduisent à la classification suivante des PLD :

Classification des PLD



Les **PLD** simples sont basés sur un réseau logique programmable et un module de sortie d'utilisation souple comportant une bascule D. Ils offrent diverses configurations répondant à un grand nombre de besoins.

Les **CPLD** (Complex PLD) sont en première approximation une intégration de plusieurs PLD simples dans un même boîtier. Cette intégration a permis de résoudre le problème d'affectation optimale des termes produit. Les CPLD sont aussi appelés EPLD (Erasable PLD).

Les **FPGA** (Field Programmable Gate Array) qui signifie réseau de portes programmables sur site. sont aussi des circuits logiques programmables par l'utilisateur. C'est une évolution des CPLD mais avec un concept différent. Ils offrent un niveau d'intégration très élevé par rapport aux CPLD.

Structure des PLD simples

Les PLD simples comportent les blocs suivants :

Un bloc d’entrée,

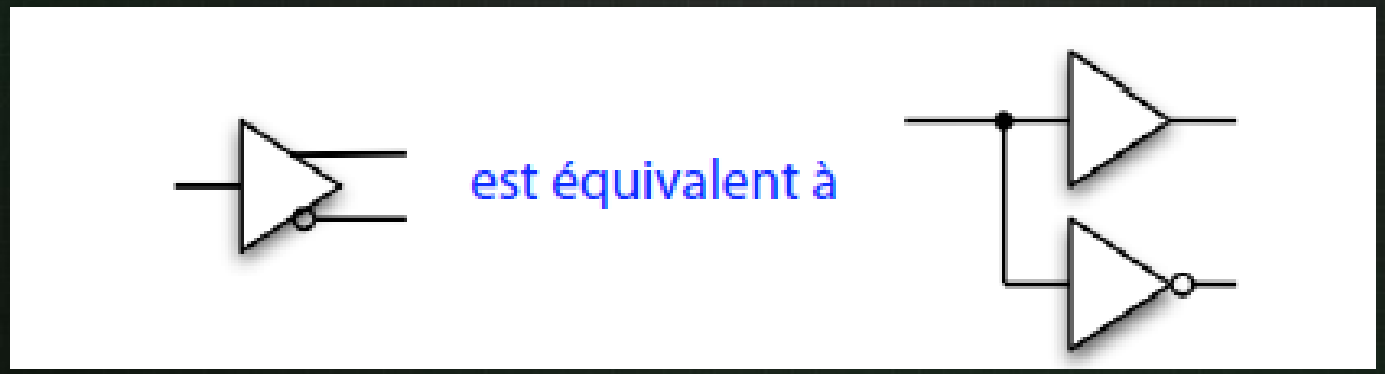
Une partie combinatoire constituée d’une matrice ET et d’une matrice OU,

Un bloc de sortie,

Un bloc d’entrée-sortie.

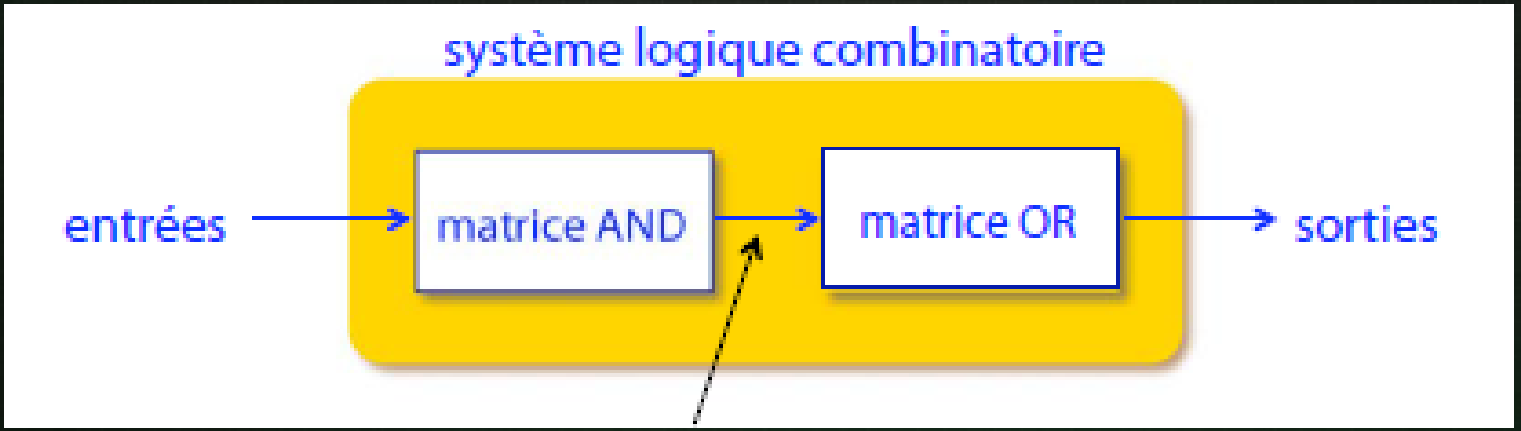
Le bloc d'entrée

Il permet de fournir au bloc combinatoire l'état de chaque entrée et de son complément

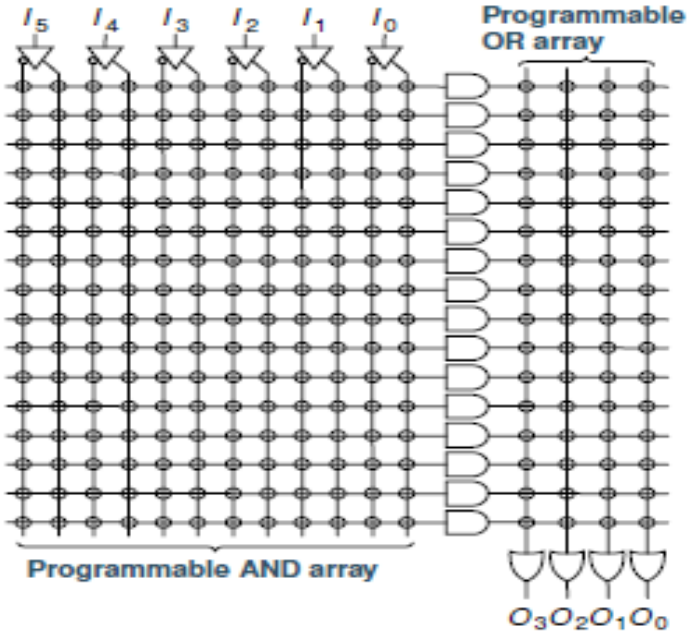


Le Bloc combinatoire

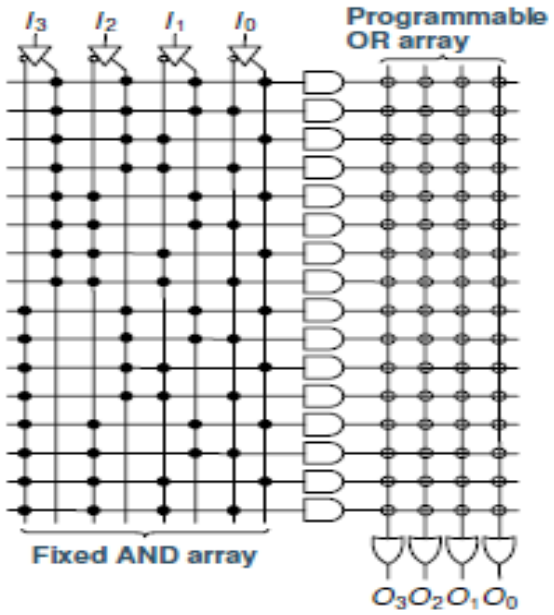
Etant donné que toute fonction logique peut être exprimée comme une somme logique de produits, les circuits logiques programmables les plus communs sont formés par une matrice AND suivie d'une matrice OR. L'une des deux matrices, ou les deux, est *programmable.*



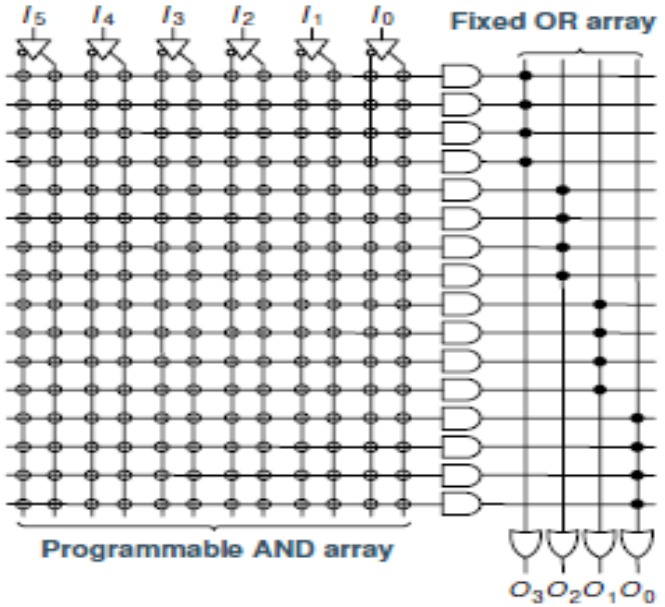
1.Réseaux logiques programmables : PLA, PAL et PROM



PLA



PROM



PAL

Indicates programmable connection

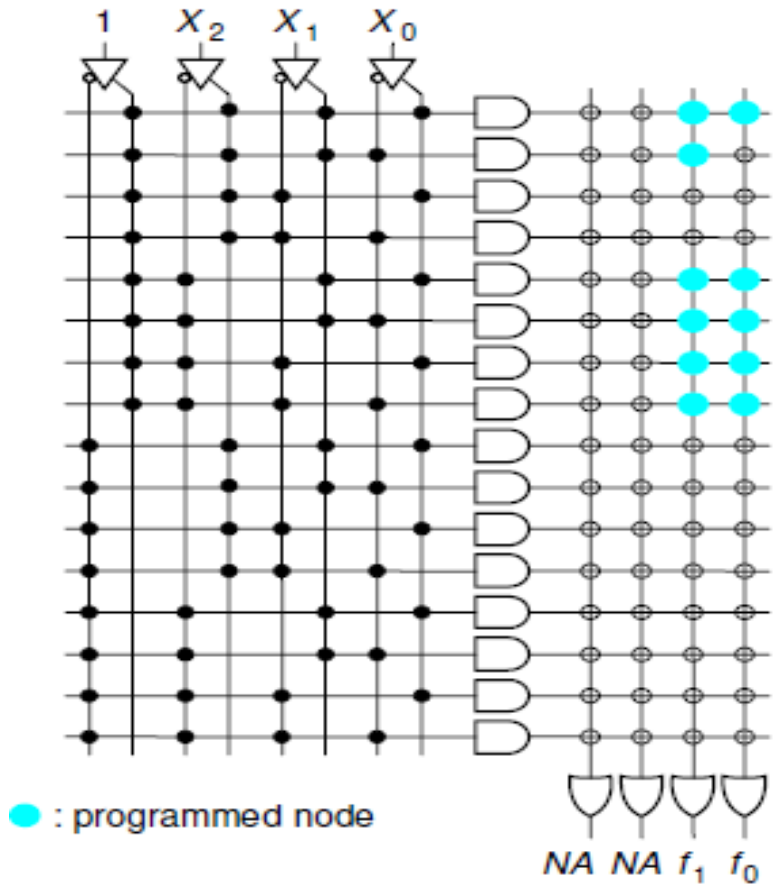
Indicates fixed connection

Selon le caractère programmables des matrices AND et OR, il existe trois types de circuits logiques programmables:

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.

Une fonction F est donc réalisée en programmant sa table de vérité. Donc en mémorisant la valeur de F pour toutes les combinaisons des entrées.

Programmation d'une PROM



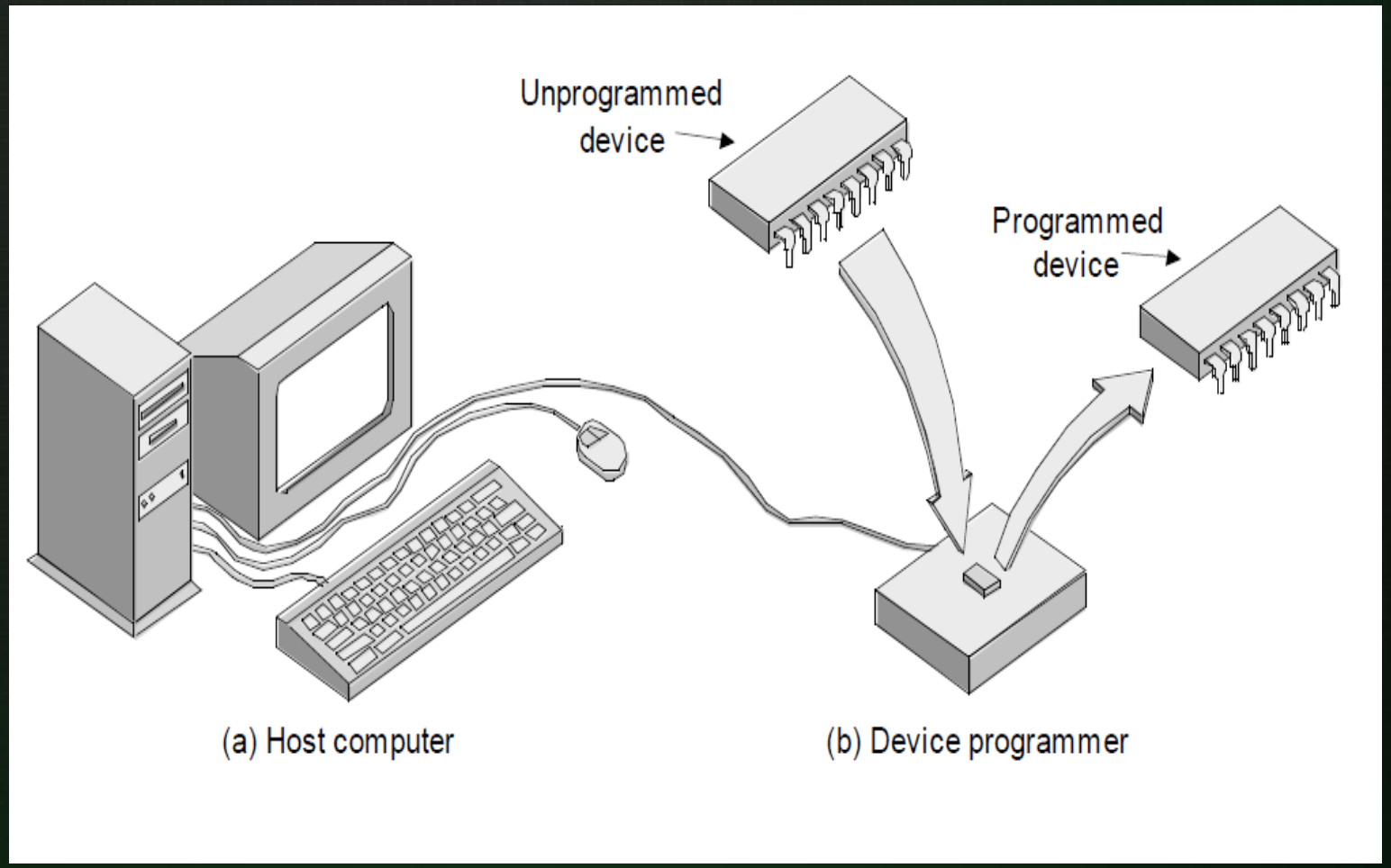
$$f_0 = x_0x_1 + \overline{x_2}$$

$$f_1 = x_0x_1x_2 + \overline{x_2} + \overline{x_0}x_1$$

PAL (Programmable Array Logic) : La structure de base d'une PAL est l'opposée de celle d'une PROM. C'est une matrice AND programmable suivie d'une matrice OR fixe.

PLA (Programmable Logic Array) : Les deux matrices sont programmables. Ce qui offre une très grande souplesse.

La programmation d'un PLD est très complexe, à cause du nombre et de l'emplacement des fusibles, ainsi que des tensions de programmation : Un logiciel et un programmateur sont nécessaires.



Structure des sorties (E/S)

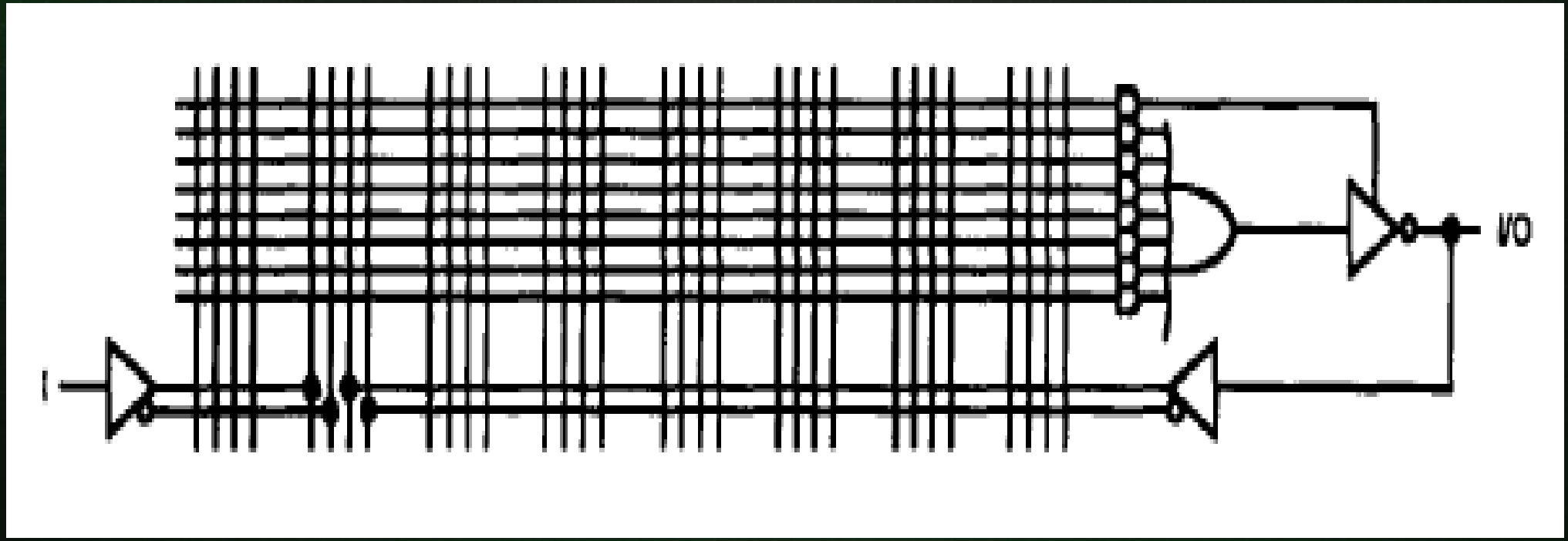
Tous les PLD possèdent un certain nombre d'entrées qui se retrouvent sous forme directe ou inversée sur la matrice AND.

Il existe 3 structures de sortie qui peuvent être aussi des entrées/sorties:

- ✓Combinatoires
- ✓Séquentielles
- ✓Versatiles

Entrées/Sorties combinatoires

La forme la plus simple d'un bloc élémentaire d'un PLD peut être représentée par :



E/S combinatoires

La sortie de la porte OU est reliée à la patte d'E/S du circuit via un buffer tri-state (3 états) dont l'état est commandé par une ligne de produit.

La broche de sortie est reliée, via un inverseur/non inverseur sur le réseau AND d'entrée. Toute sortie est réinjectée dans la matrice programmable.

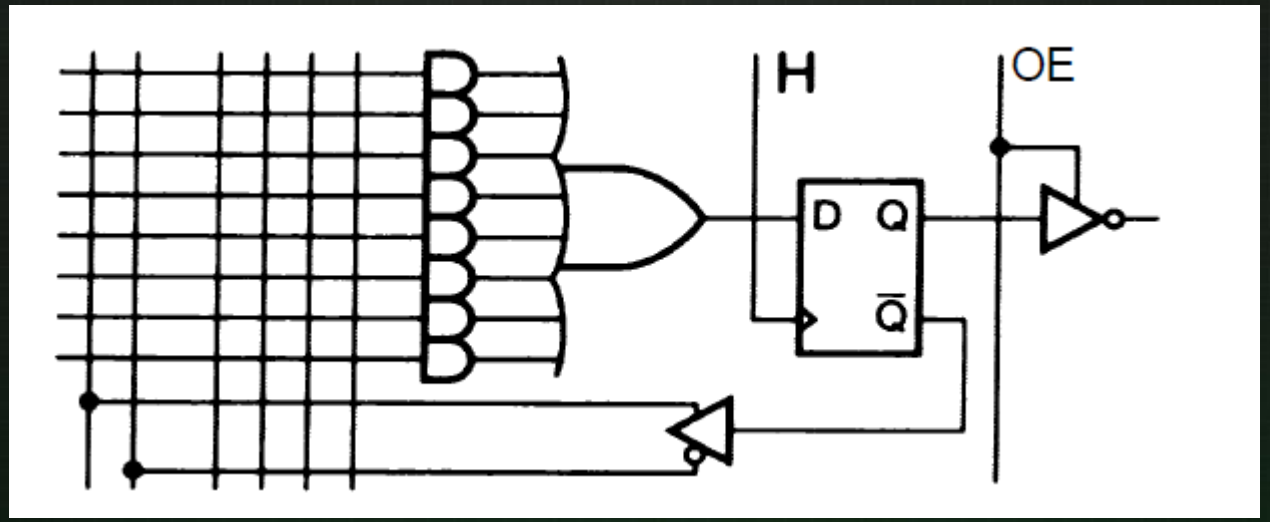
Cette broche peut servir d'entrée (I/O) si le buffer est en haute impédance.

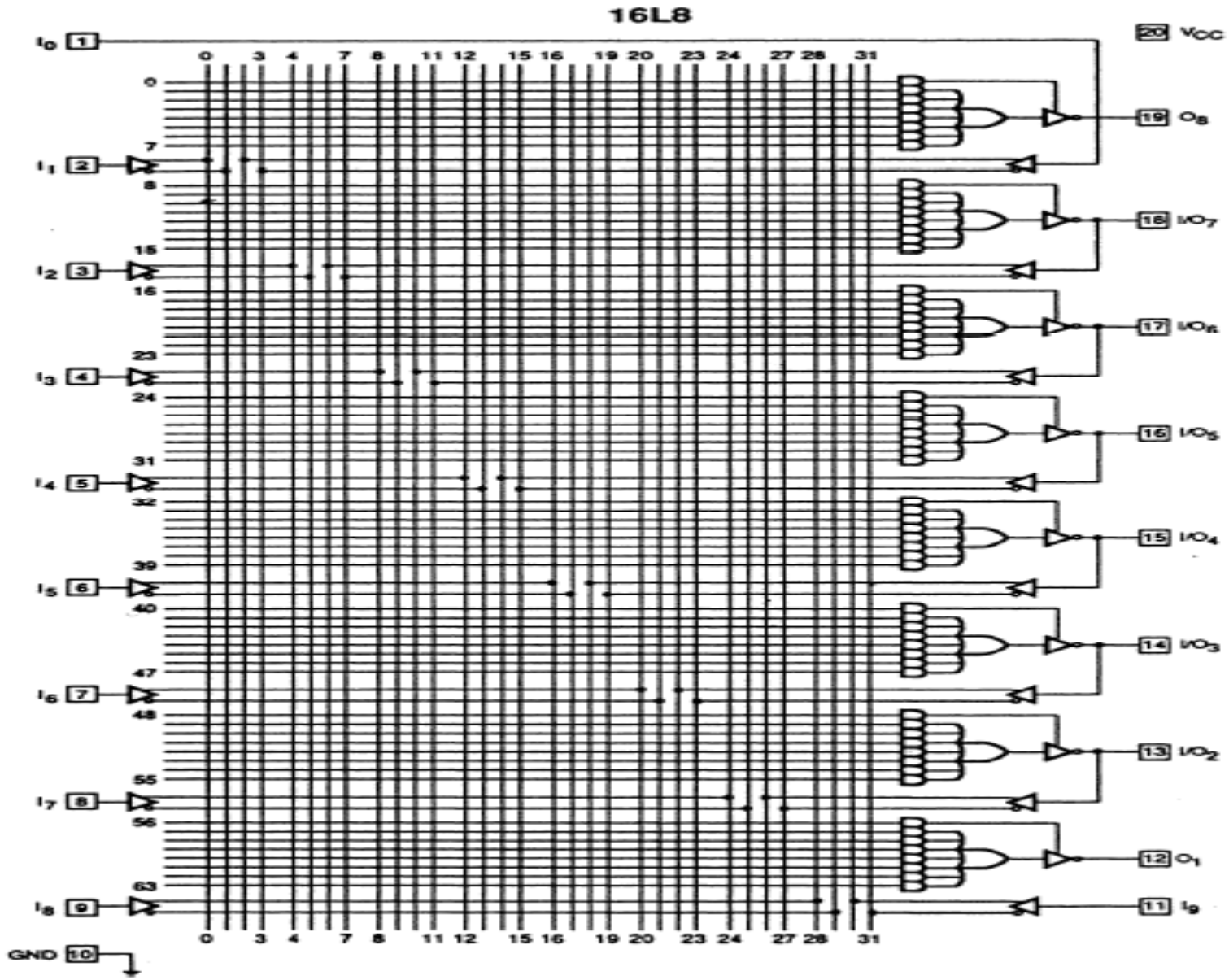
Seuls varient d'un PLD à un autre le nombre d'entrées, d'I/O et le nombre de produit à l'entrée de la porte OR.

Les PLD à registre (synchrones)

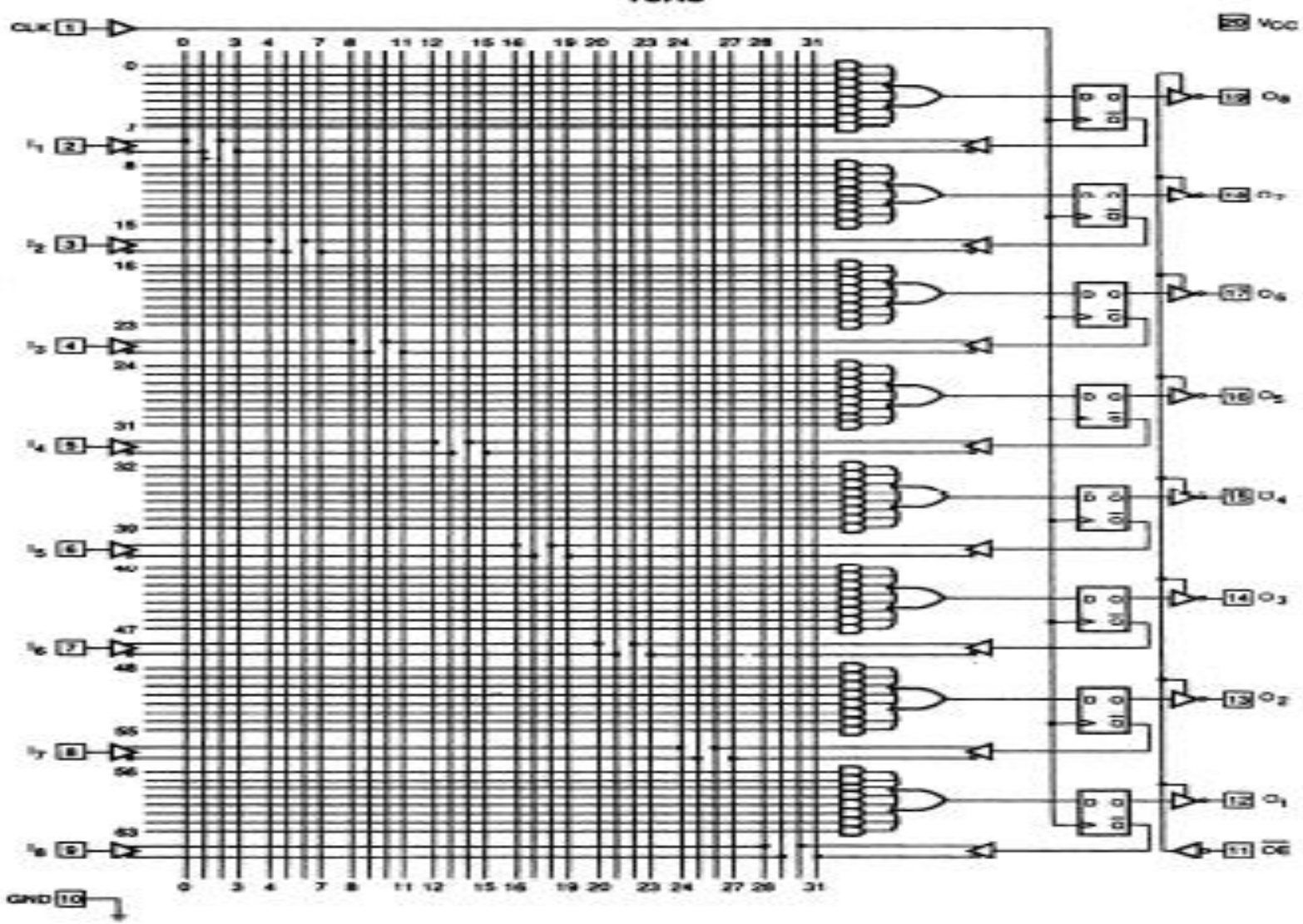
Une bascule D permet la logique séquentielle (sorties séquencées sur une horloge).

Une commande OE (output enable) permet de désactiver la sortie.



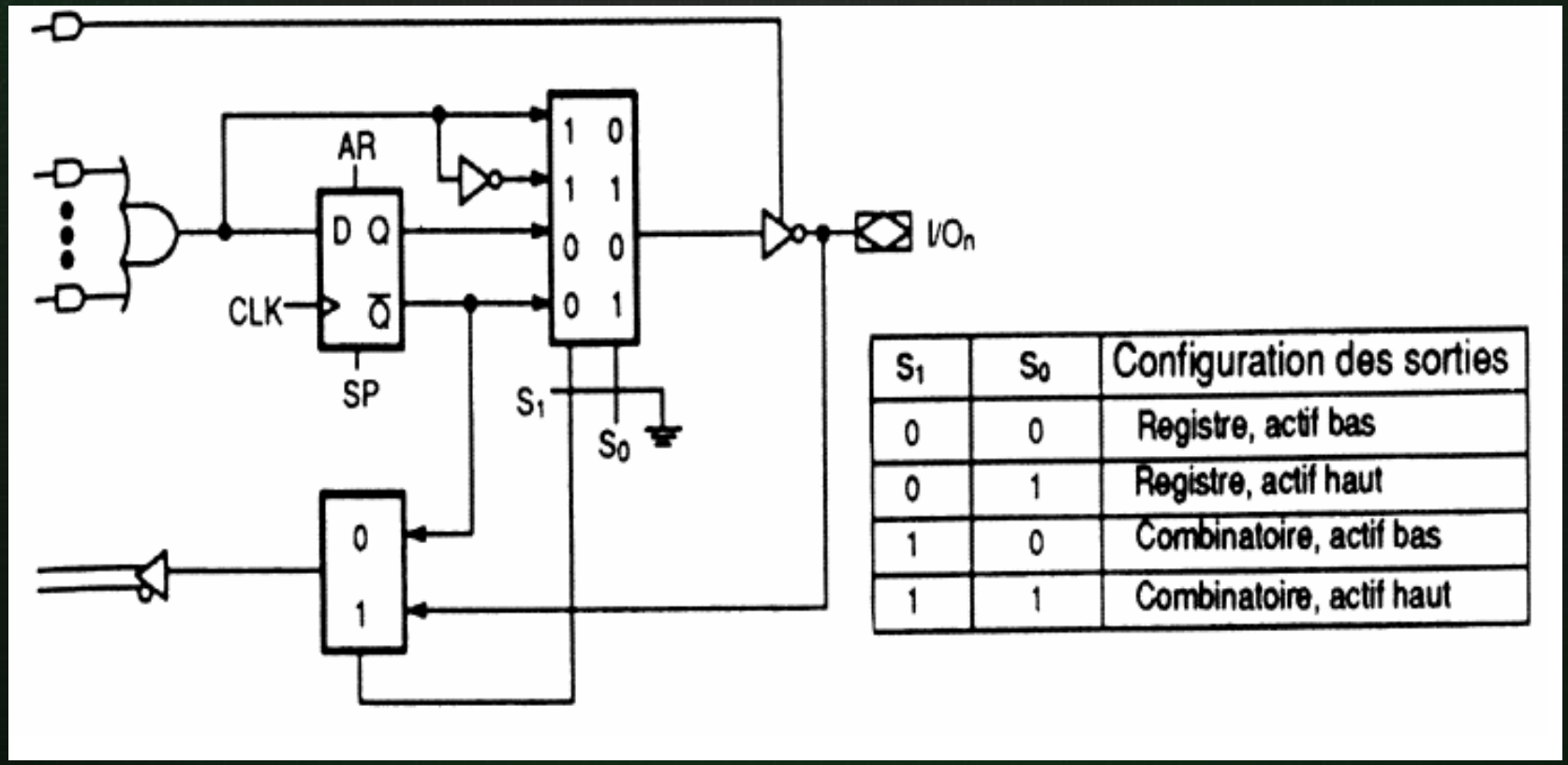

16L8
I0 1
I1 2
I2 3
I3 4
I4 5
I5 6
I6 7
I7 8
I8 9
GND 10
20 VCC
19 O8
18 I/O7
17 I/O6
16 I/O5
15 I/O4
14 I/O3
13 I/O2
12 O1
11 I9
0 3 4 7 8 11 12 15 16 19 20 23 24 27 28 31
0
7
8
15
16
23
24
31
32
39
40
47
48
55
56
63

16R8


CLK 1
GND 10
20 VCC
19 O8
18 O7
17 O6
16 O5
15 O4
14 O3
13 O2
12 O1
11 OE

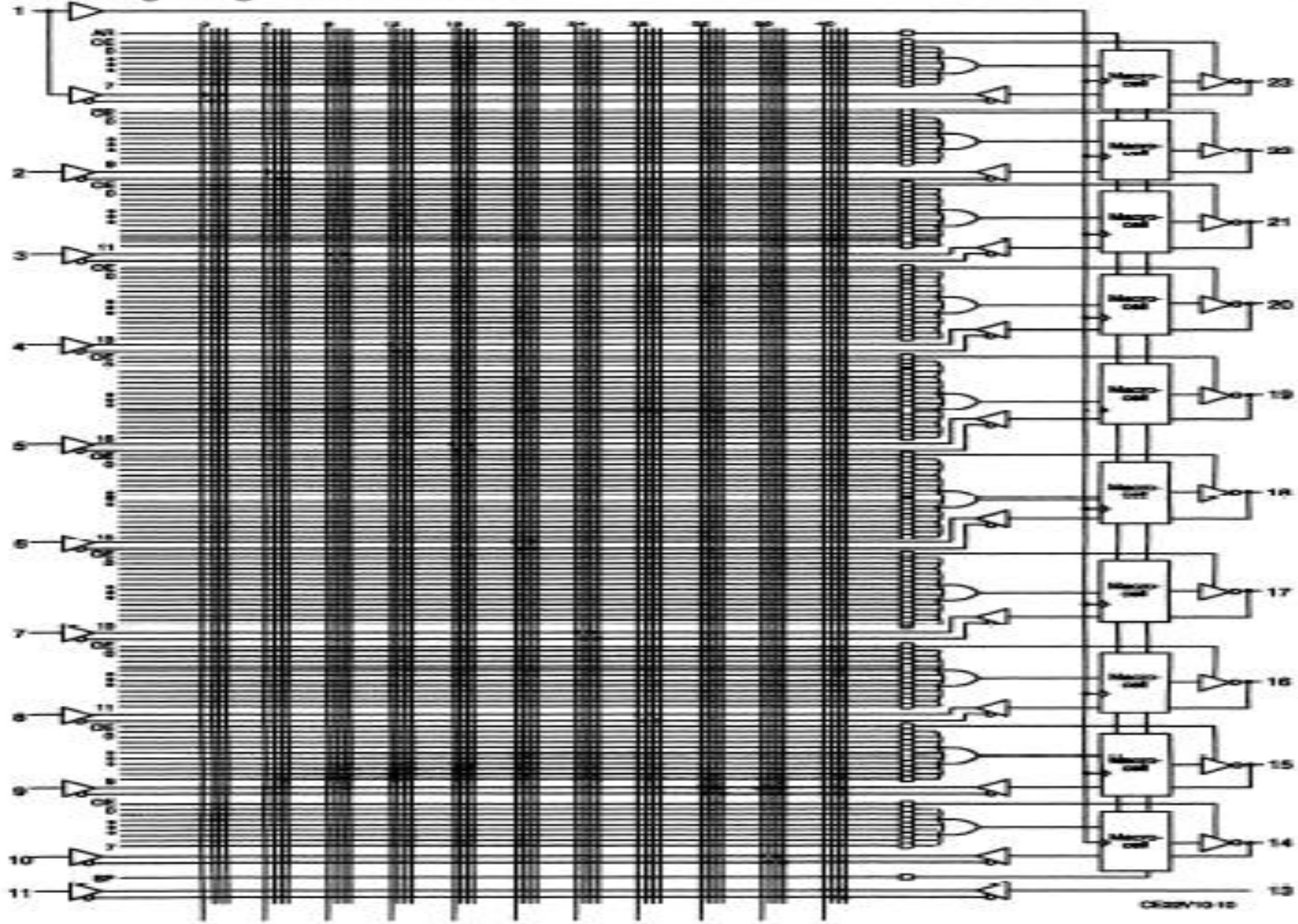
Les PLD versatiles

- C'est des circuits programmables au niveau



S_1	S_0	Configuration des sorties
0	0	Registre, actif bas
0	1	Registre, actif haut
1	0	Combinatoire, actif bas
1	1	Combinatoire, actif haut

Functional Logic Diagram for PALCE22V10



Technologies d’interconnexion (de programmation)

Le terme d’interconnexion désigne des cellules qui permettent, suivant la technologie utilisée, une connexion *temporaire* ou *définitive* entre deux réseaux.

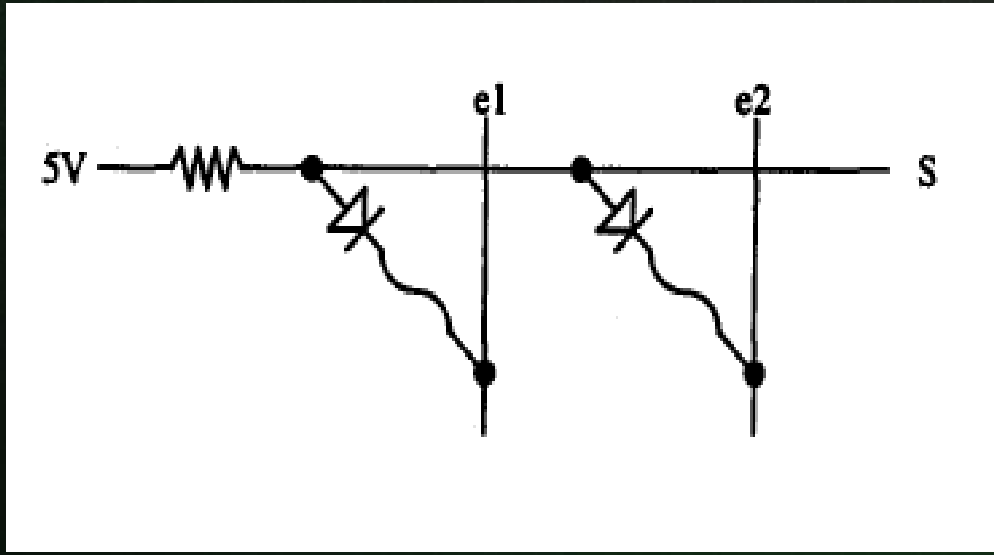
Un circuit logique programmable est un circuit contenant un ensemble de portes logiques élémentaires dont l’interconnexion est laissée aux soins de l’utilisateur en vue de créer la fonction désirée.

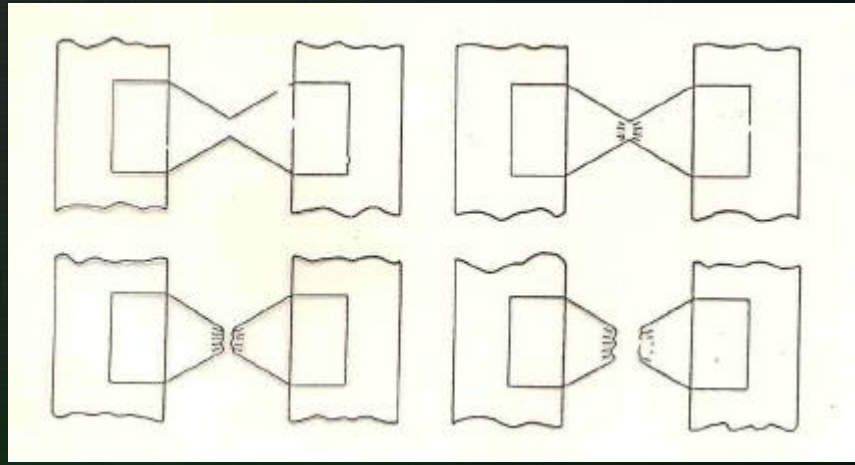
Circuits programmables une seule fois (OTP)

Ce sont des réseaux utilisant des cellules à *fusible* ou à *antifusible* pour leur interconnexion.

Cellules à fusible

C’est la plus ancienne des technologies. Le circuit non programmé contient, par défaut, toutes les connexions possibles. Ces connexions sont réalisées par des fusibles (mélange de titane et de tungstène).



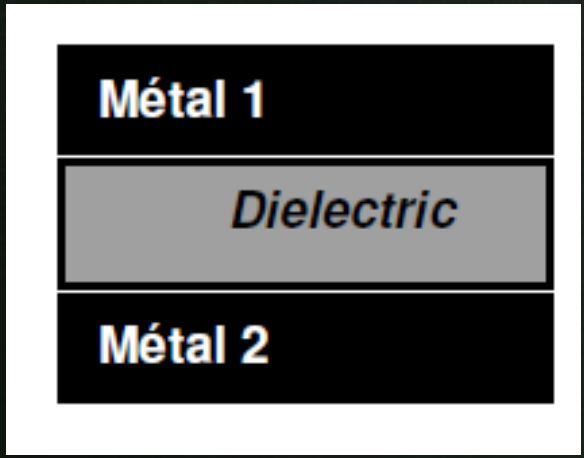


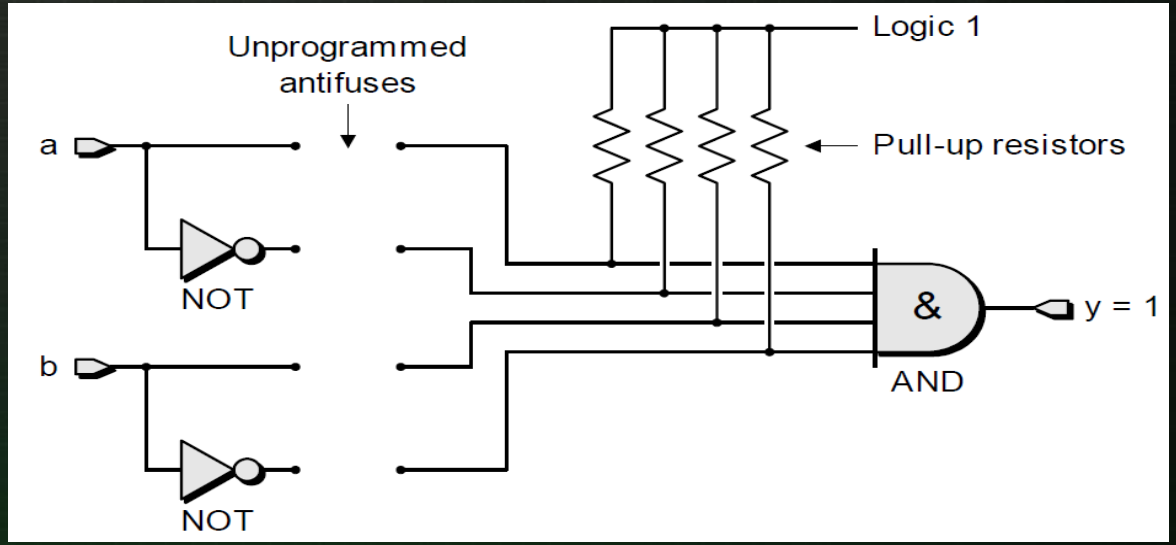
Cellules à antifusible

Antifusible = condensateur (ACTEL 1986)

Par défaut, les connexions n'existent pas.

La programmation est réalisée par claquage d'un diélectrique (isolant).





Circuits programmables et reprogrammables

Ce sont des circuits utilisant des cellules à transistors MOS.

Avantages

Consommation faible

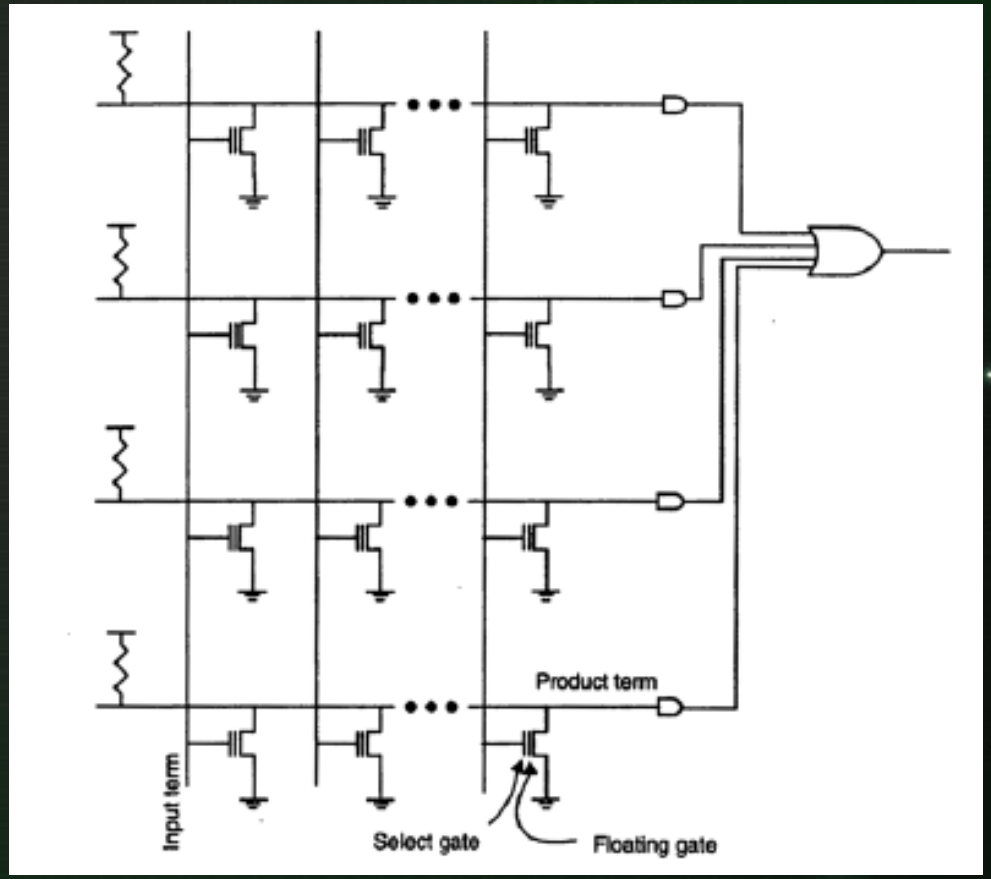
Grande intégration

Rapidité

Interconnexions reprogrammables

Cellules UVPROM :

- Utilisation de transistors du type FAMOS pour la programmation.
- Effacement par UV.



Interconnexions reprogrammables

Cellules EEPROM :

- Utilisation de transistors du type FLOTOX pour la programmation.

-Effaçables électriquement.

Interconnexions reprogrammables

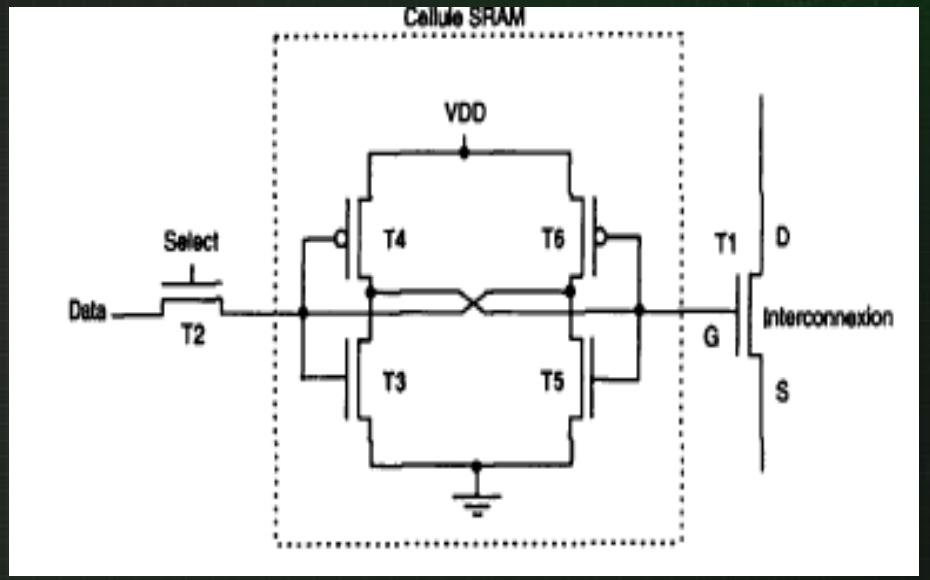
Cellules SRAM :

-Forte densité d'intégration

-Rapidité

Inconvénients :

-Reprogrammation à chaque mise sous tension



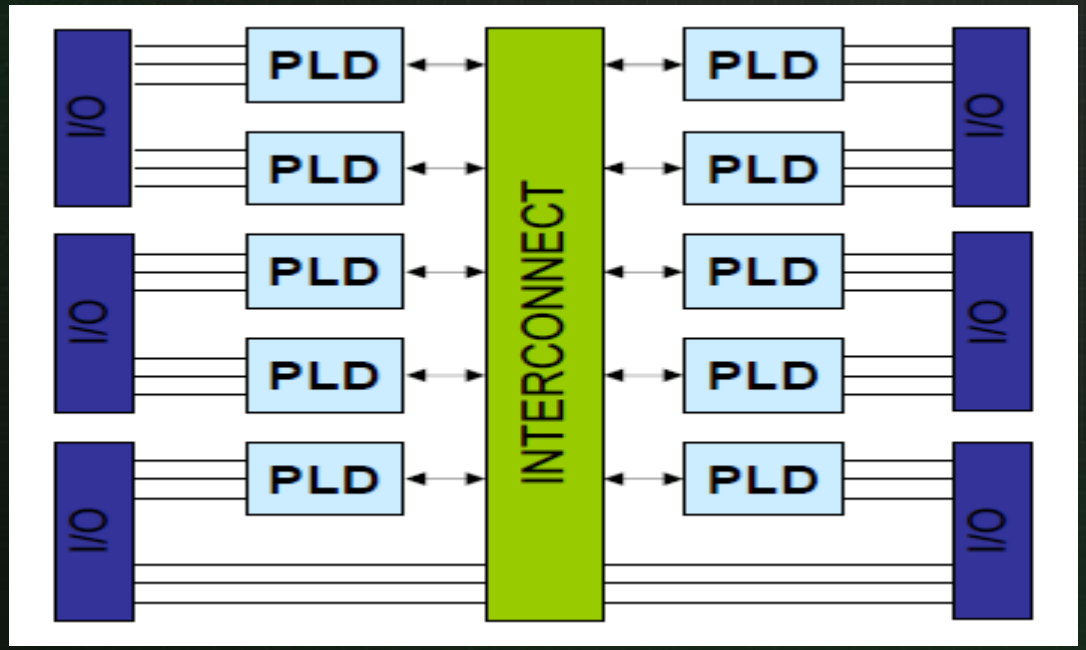
2. Circuits logiques programmables complexes (CPLD)

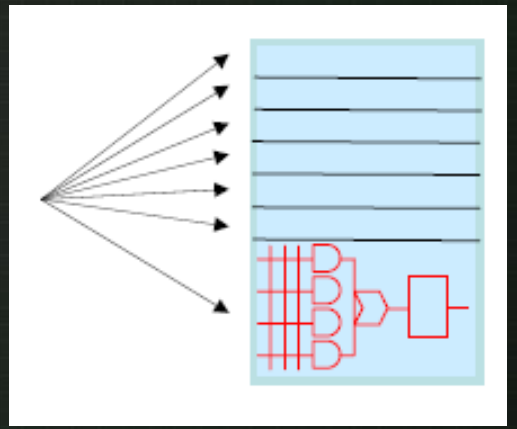
Les CPLD sont une extension des PAL.

Un CPLD incorpore plusieurs PALs ou PLAs sur une seule puce.

Les blocs communiquent par l'intermédiaire d'interconnexions programmables.

2. Circuits logiques programmables complexes (CPLD)





2. Les CPLD

Un CPLD est constitué :

-d’un bloc des entrées dédiées,

-De plusieurs blocs d’entrées-sorties,

-De plusieurs blocs de macrocellules,

-D’un réseau d’interconnexion entre les blocs de macro-cellules.

2. Les CPLD

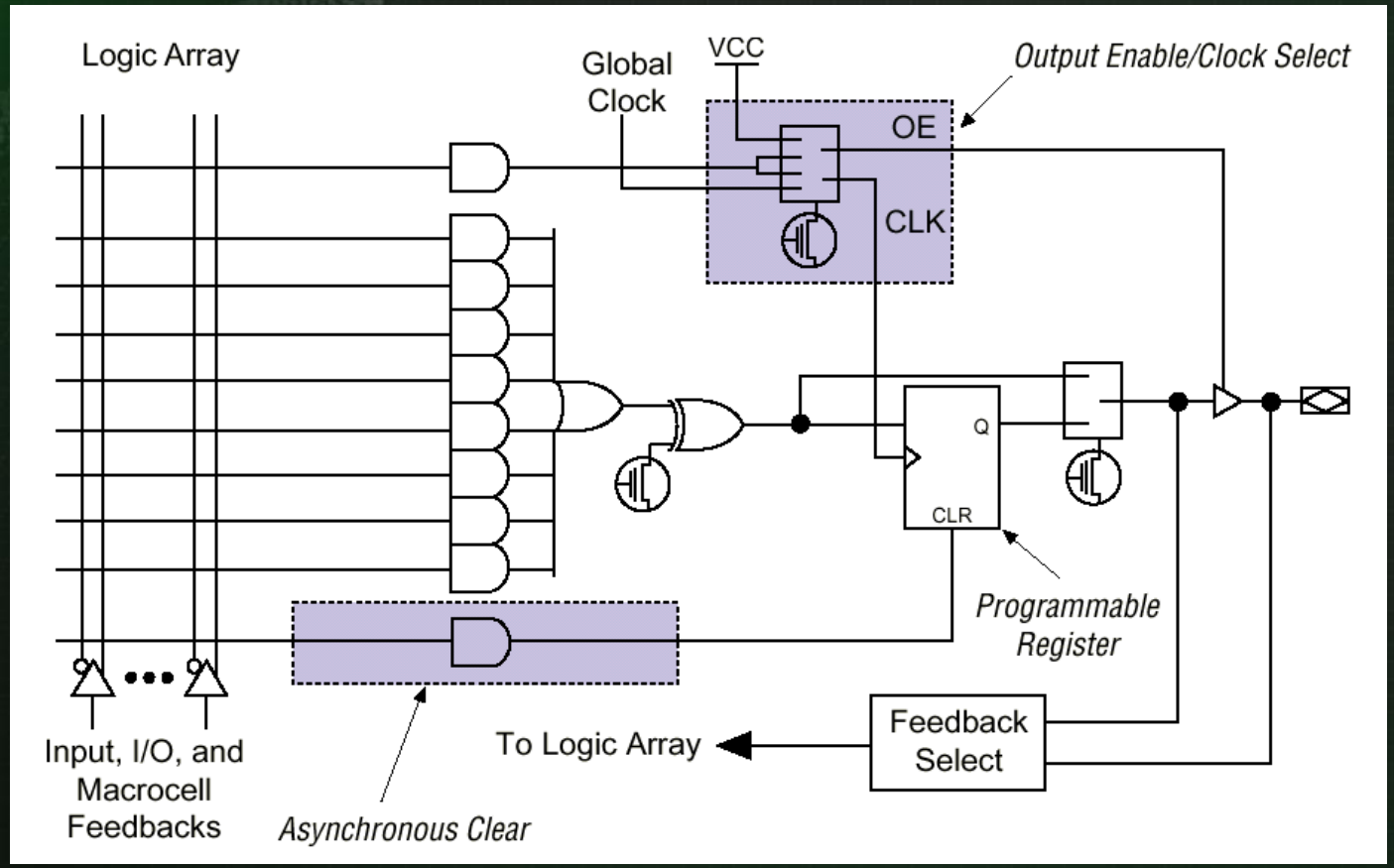
Un bloc de macro-cellule comporte :

-Plusieurs macro-cellules (8 au moins)

-Des blocs d’entrées-sorties,

-Une matrice ET pour toutes les macro-cellules du bloc,

Structure de la macro-cellule



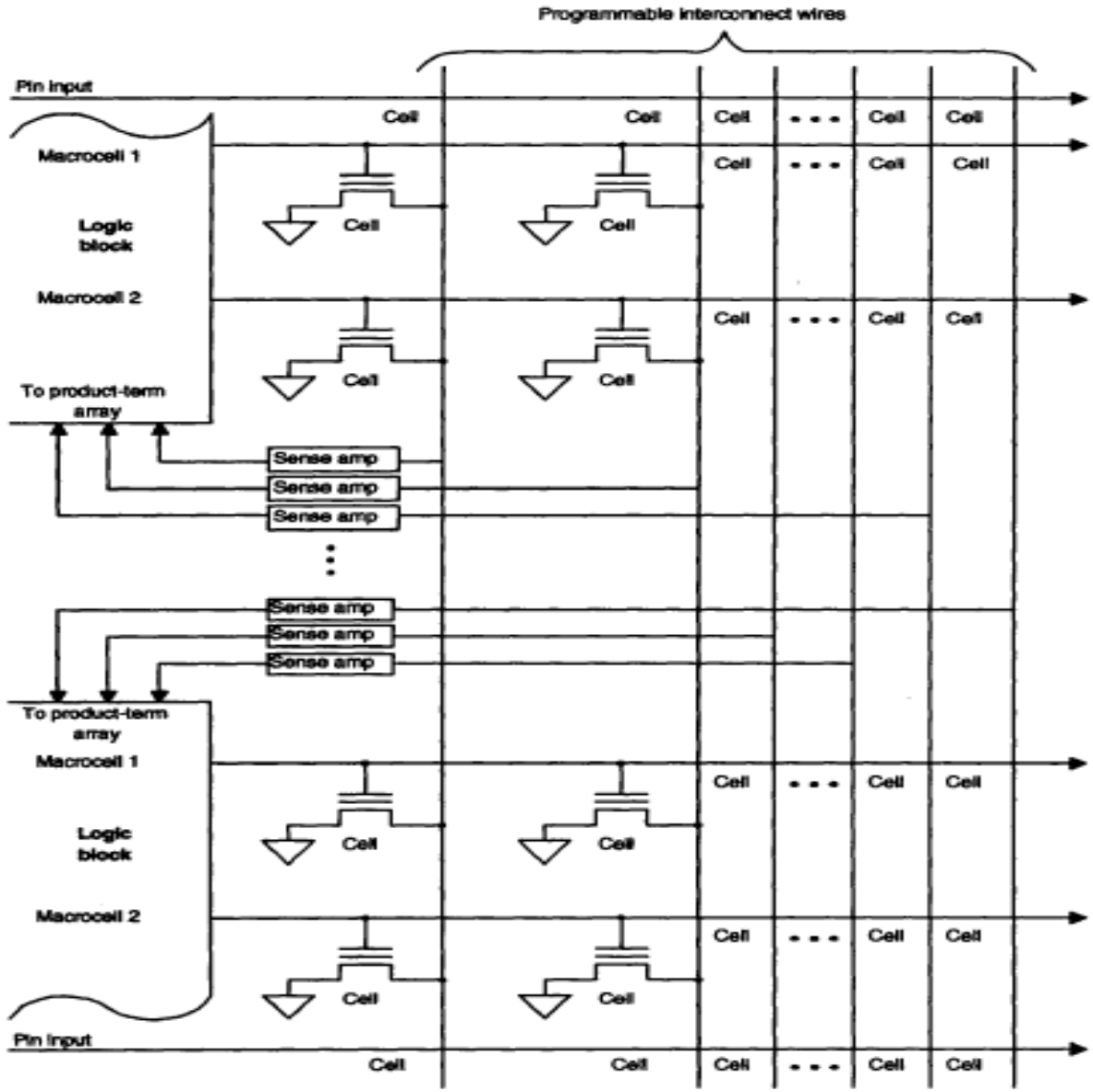
Matrice d’interconnexion

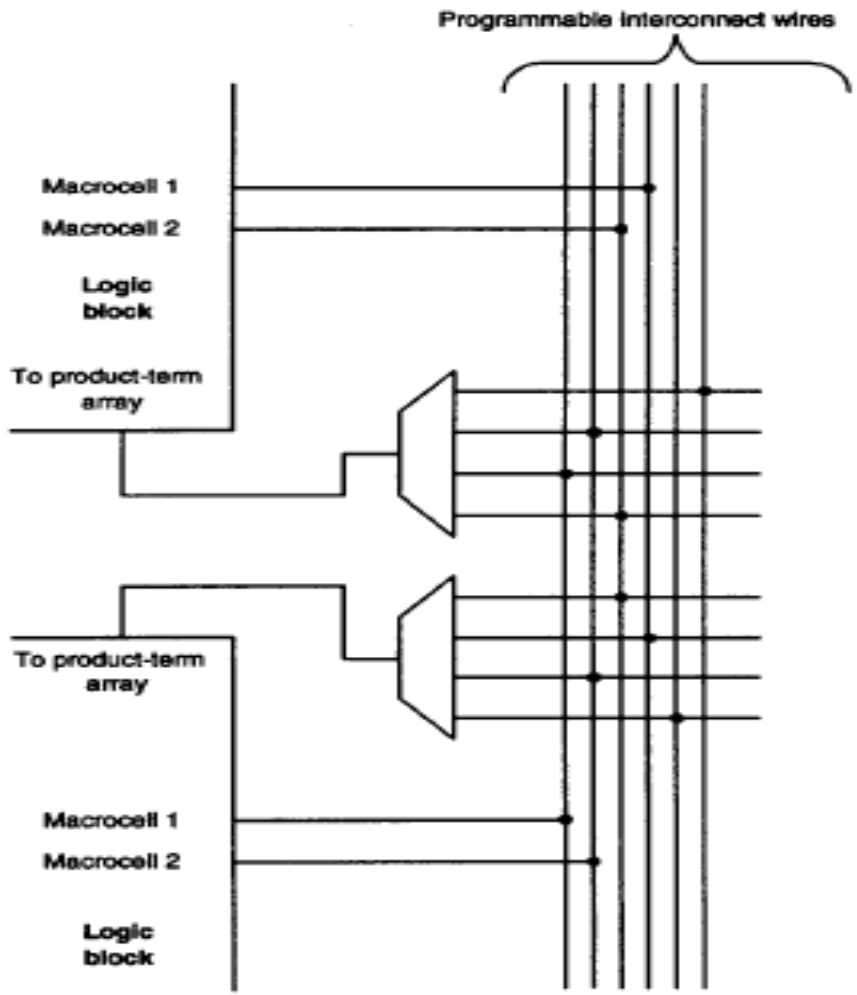
Le réseau d’interconnexions programmable route les signaux des sorties des blocs logiques (LAB : Logic Array Block) aux entrées des mêmes ou des autres LABs.

Deux implémentations sont rencontrées :

-L’interconnexion basée sur un réseau,

- L’interconnexion basée sur des multiplexeurs.


Programmable interconnect wires
Pin input
Macrocell 1
Logic block
Macrocell 2
To product-term array
Sense amp
Sense amp
Sense amp
Sense amp
Sense amp
Sense amp
To product-term array
Macrocell 1
Logic block
Macrocell 2
Pin input
Cell

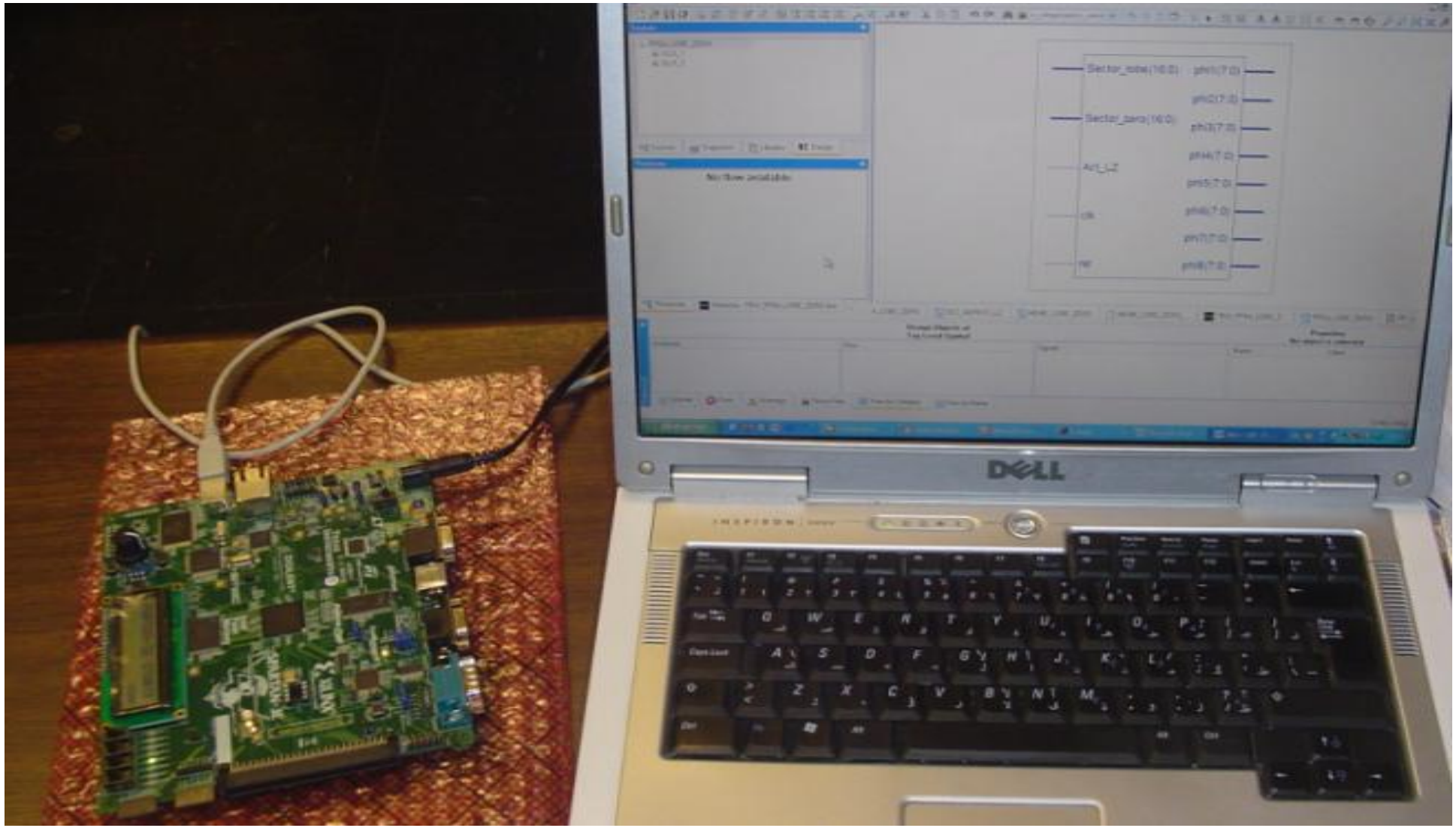

Programmable interconnect wires
Macrocell 1
Macrocell 2
Logic block
To product-term array
To product-term array
Macrocell 1
Macrocell 2
Logic block

3. Réseaux pré-diffusés programmables (FPGA : Field Programmable Gate Array)

Les FPGA se sont imposés sur le marché des circuits logiques depuis la fin des années 1980 (Xilinx 1984).

Ils concurrencent de nos jours certains microprocesseurs et microcontrôleurs en complexité et en performance.


XILINX
XINTEX


DELL

3. Les FPGA (Architecture)

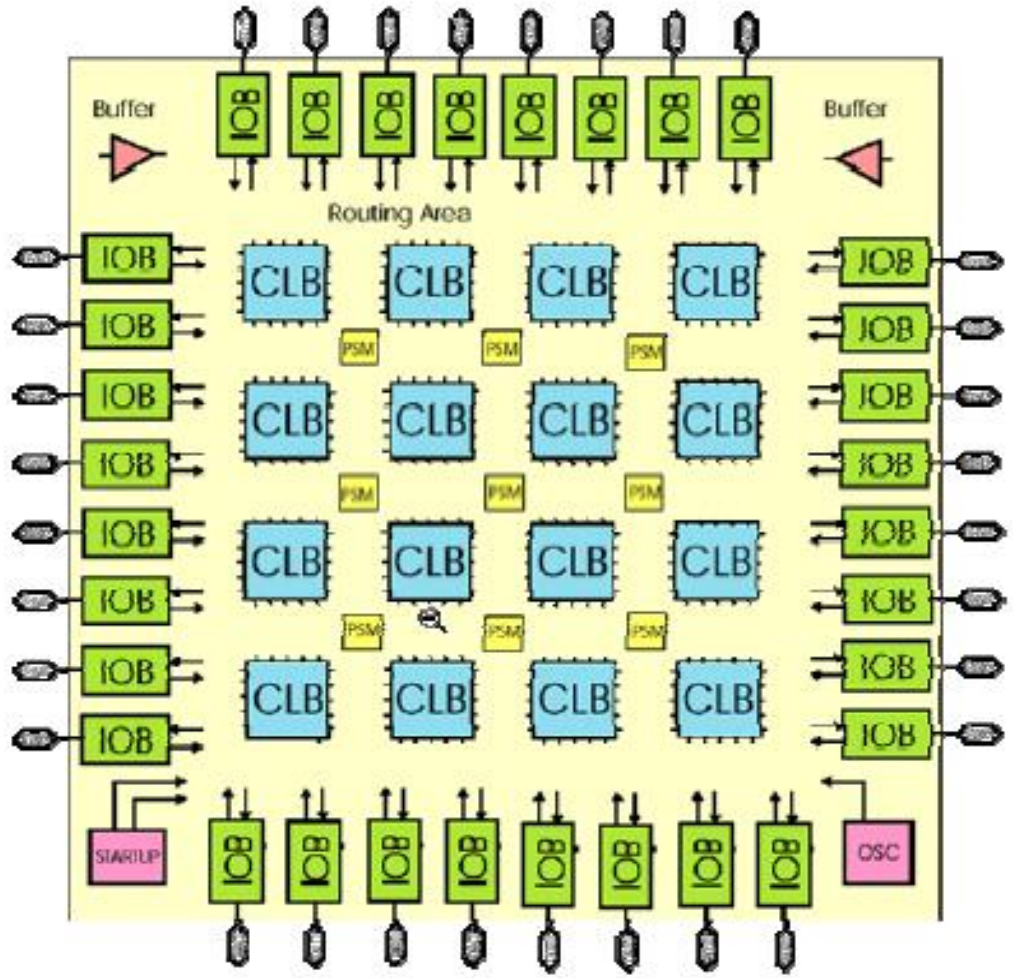
Les FPGA constituent les composants programmables *par l'utilisateur* contenant le *plus grand* nombre de portes logiques (VLSI).

C'est un arrangement matriciel de blocs logiques avec configuration des :

1. Interconnexions entre les blocs logiques,
2. La fonction de chaque bloc.

Par rapport aux CPLD, les FPGA utilisent des modules logiques plus réduits mais beaucoup plus nombreux (100-1000 blocs).

Ces modules sont entourés de blocs d'entrées/sorties programmables et connectés ensemble via des interconnexions programmables.


Buffer
Buffer
IOB
Routing Area
CLB
PSM
STARTUP
OSC

Les FPGA (Architecture)

Un FPGA comporte :

1. Des tables de conversion ou d'observation (LUT : Look Up Table),
2. Des Multiplexeurs,
3. Des portes logiques,
4. Des bascules.

Les FPGA

Un FPGA est composé à la base de :

- Un réseau de blocs de logique programmable (Configurable Logic Bloc :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 ; et

-Des blocs spéciaux d'entrée et de sortie avec le monde extérieur (Input/Output Blocs :IOB)

Blocs Logiques Configurables (CLB)

Les CLB des FPGA sont beaucoup plus nombreux et plus simples que les blocs fonctionnels des CPLD.

Ils favorisent la réalisation et l'intégration d'une multitude de circuits indépendants (combinatoires ou séquentiels).

Le degré de complexité des CLB est appelé granularité (Grain fin et Grain épais)

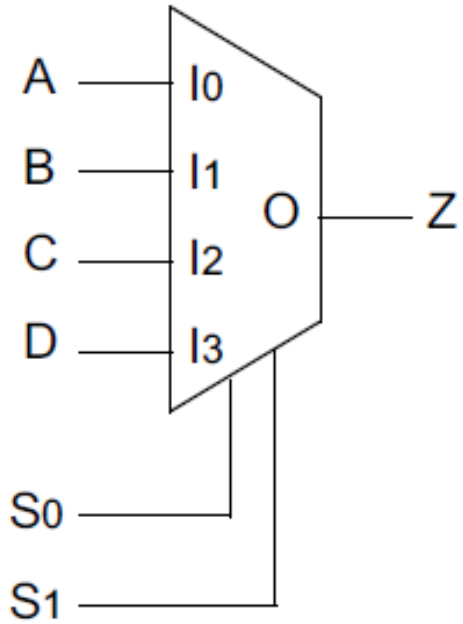
Les CLB

Il y a deux catégories de blocs logiques programmables (CLB) :

1. Ceux basés sur les multiplexeurs (Altera),
2. Et ceux basés sur les tables de conversion (LUT) (Xilinx)

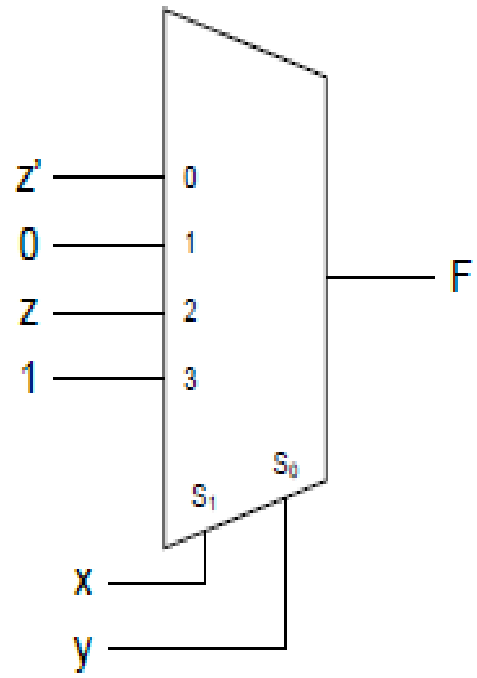
Les Multiplexeurs

Le multiplexeur est un sélecteur de données.



S1	S0	Z
0	0	A
0	1	B
1	0	C
1	1	D

#	x	y	z	F	entrée D_i	valeur
0	0	0	0	1	D_0	z'
1	0	0	1	0		
2	0	1	0	0	D_1	0
3	0	1	1	0		
4	1	0	0	0	D_2	z
5	1	0	1	1		
6	1	1	0	1	D_3	1
7	1	1	1	1		



Look Up Table (LUT)

Les CLB basés sur les LUT utilisent de petites mémoires programmables (SRAM) au lieu de Mux.

Il faut un Mux deux fois plus grand pour réaliser la même fonction.

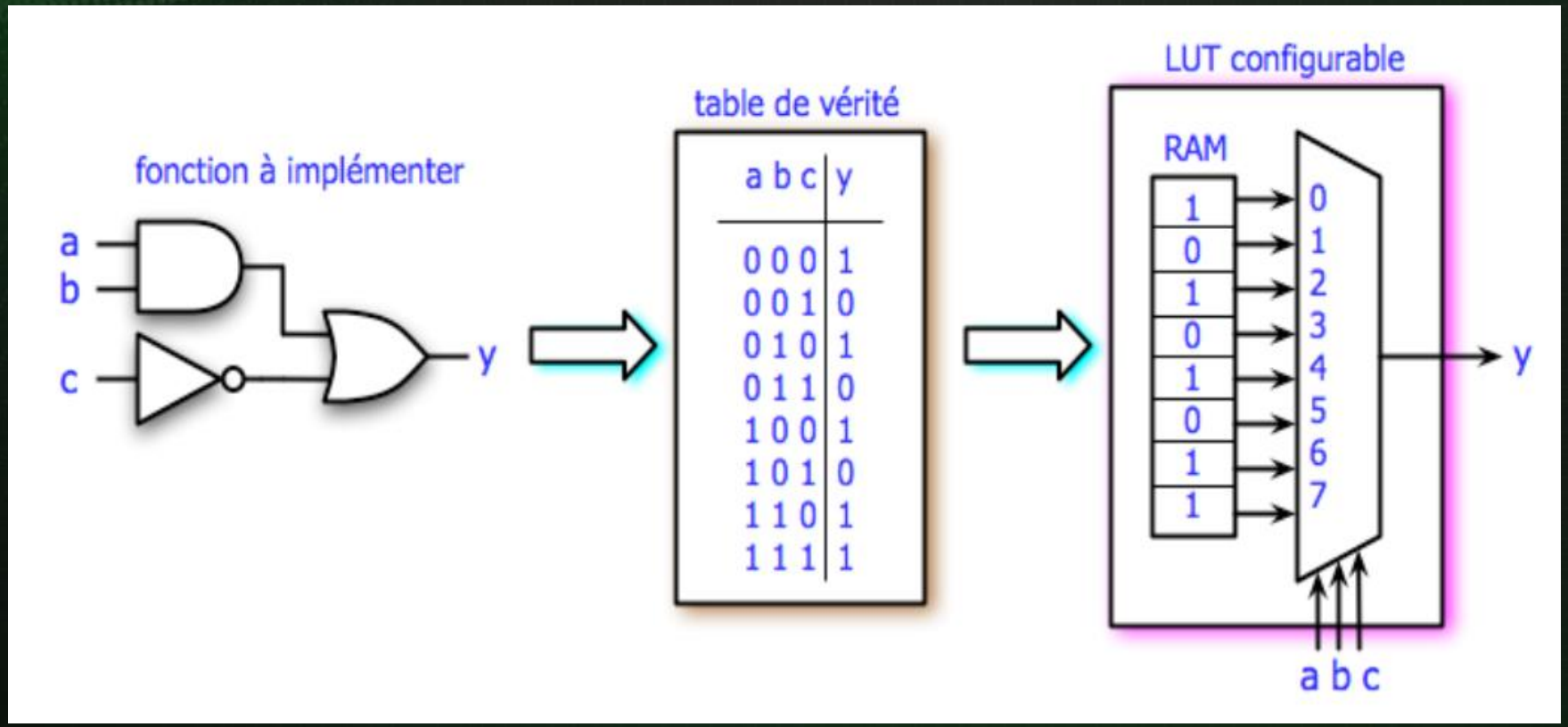
Avantages:

Routage plus simple

Circuit plus rapide (Les entrées du Mux sont des constantes).

LUT

La fonction de la LUT est de stocker la table de vérité de la fonction combinatoire à implémenter dans la cellule.



LUT

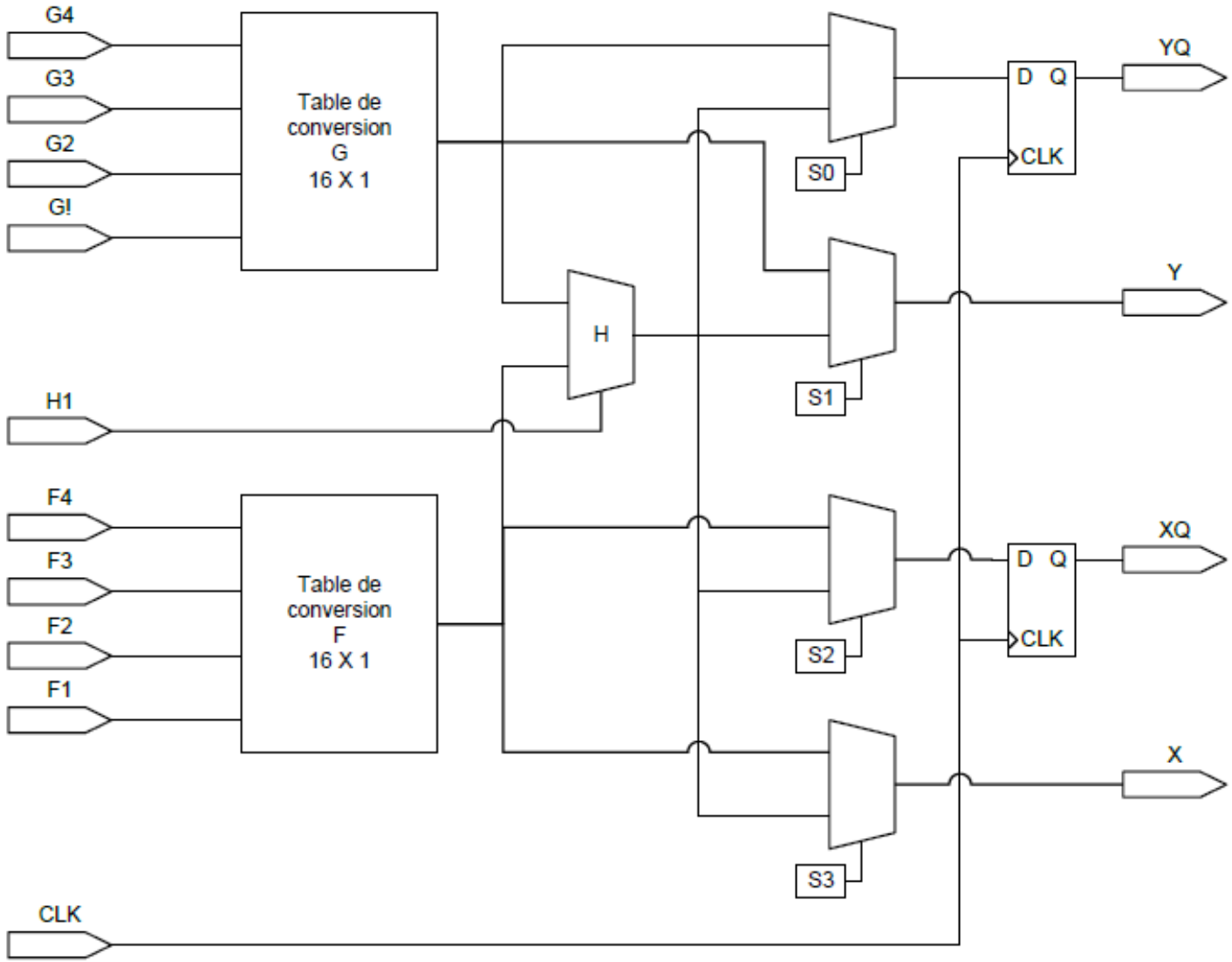
Un LUT correspond à un arbre de Mux connectés à des mémoires (SRAM).

Le contenu des mémoires est stockée dans un composant externe (EPROM).
La configuration est chargée à chaque mise sous tension.

CLB (FPGA xilinx)

Le CLB est composé de :

1. Deux LUT programmables à 4 entrées chacune, F et G qui sont des mémoires de 16 bits chacune;
2. Un Mux H qui permet de choisir la sortie de l'une des 2 LUT.
3. Quatre Mux dont les signaux de contrôle S_0 à S_3 sont programmables.
4. Deux éléments à mémoire configurables en bascules ou verrous.



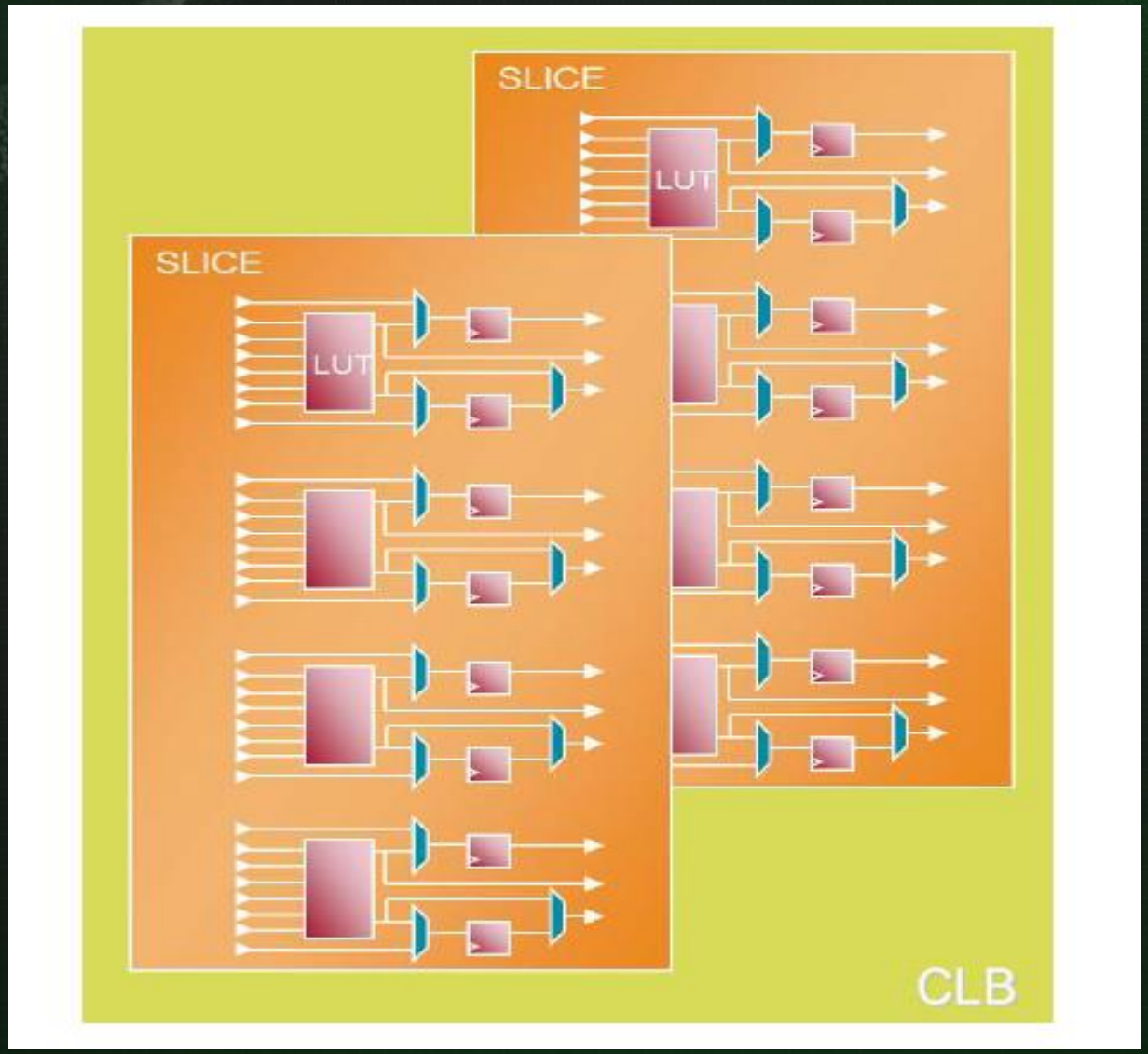
bloc de logique programmable simplifié – Xilinx

Terminologie

Plusieurs termes sont utilisés pour parler de l'architecture interne des FPGA.

Pour la famille Cyclone, Altera utilise le terme : LE :Logic Element pour une cellule incluant une LUT, un additionneur et un registre. Un LAB (Logic Array Block) regroupe 10 LEs.

Pour les familles Spartan et Virtex , Xilinx utilise le terme slice pour un module de base incluant 2 LUT, 2 additionneurs et 2 registres. Un CLB regroupe 2 ou 4 slices, selon la famille de FPGA.


SLICE
LUT
SLICE
LUT
CLB