République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique
Centre **U**niversitaire **E**lcherif **B**ouchoucha **A**flou

Institut des **S**ciences
Département d'**I**nformatique

Systèmes embarqués

Préparé par : **Dr. Azzouzi Oussama**

2025/2026

Table des matières

La liste de figures

1. Introduction

Depuis l'avènement des premiers microprocesseurs dans les années 1970, le monde de l'électronique et de l'informatique embarquée a connu une évolution remarquable. À l'origine, les microprocesseurs étaient conçus pour des applications simples, mais la rapide progression technologique a permis d'augmenter considérablement leur puissance de calcul, leur densité d'intégration et leur efficacité énergétique. L'évolution des familles de processeurs — des premiers Intel 4004 et Motorola 6800 aux architectures modernes multicœurs et hétérogènes — a ouvert la voie à une nouvelle ère : celle des systèmes embarqués intelligents. Aujourd'hui, ces systèmes sont omniprésents dans notre quotidien : téléphones mobiles, automobiles, dispositifs médicaux, objets connectés, satellites, drones et bien d'autres.

Un système embarqué se définit comme une combinaison harmonieuse entre matériel (hardware) et logiciel (software) dédiée à l'exécution d'une ou plusieurs tâches spécifiques, souvent en temps réel. Contrairement aux ordinateurs généraux, il est conçu pour répondre à des contraintes précises de performance, de consommation, de fiabilité et de coût. Le schéma fonctionnel d'un système embarqué met en évidence ses interactions avec l'environnement : des capteurs assurent la collecte de données physiques, le processeur effectue le traitement, et les actionneurs traduisent les décisions en actions concrètes. À cela s'ajoutent des interfaces de communication qui permettent l'échange d'informations avec d'autres systèmes ou réseaux.

Les composants principaux d'un système embarqué incluent donc un processeur (microcontrôleur, DSP, SoC, ou FPGA), une mémoire, des périphériques d'entrée/sortie et souvent un système d'exploitation temps réel (RTOS). La conception de tels systèmes repose sur une approche méthodologique intégrée, combinant matériel et logiciel (co-design matérielle/logicielle). Cette approche permet d'atteindre un équilibre optimal entre les performances, la flexibilité et le coût. Le partitionnement des tâches entre matériel et logiciel est une étape clé, visant à déterminer quelles fonctions doivent être implémentées en logique matérielle pour des raisons de rapidité, et lesquelles doivent être programmées en logiciel pour des raisons de flexibilité.

Sur le plan architectural, les systèmes embarqués modernes adoptent de plus en plus des structures hétérogènes, intégrant plusieurs cœurs de traitement, des accélérateurs matériels dédiés et des blocs de communication avancés. Ces architectures permettent de répondre à des besoins variés, allant du traitement de signaux en temps réel à l'intelligence artificielle embarquée.

Dans ce contexte, les FPGA (Field Programmable Gate Arrays) se sont imposés comme des composants essentiels. Ces circuits logiques reconfigurables offrent une flexibilité unique, permettant aux concepteurs de définir ou de modifier le matériel après la fabrication. Contrairement aux ASICs, figés après production, les FPGAs permettent la reprogrammation du

matériel, favorisant l'adaptabilité et la mise à jour continue des systèmes. Grâce à leur capacité à exécuter plusieurs opérations en parallèle, ils offrent également des performances temporelles élevées, tout en conservant une consommation maîtrisée.

Aujourd'hui, la convergence entre les systèmes embarqués et les technologies FPGA représente une avancée majeure dans la conception de plateformes intelligentes, reconfigurables et performantes. Cette combinaison ouvre la voie à des solutions innovantes dans divers domaines tels que la cryptographie, le traitement du signal, les télécommunications, l'aéronautique ou encore les systèmes médicaux.

Ce cours est spécifiquement destiné aux étudiants de Master 2 Informatique, spécialité Intelligence Artificielle, qui seront amenés à travailler sur des systèmes avancés alliant électronique, informatique embarquée et algorithmes intelligents. L'objectif est de leur fournir une compréhension approfondie des architectures matérielles modernes, en particulier celles basées sur les FPGA, ainsi que des méthodologies de co-conception matériel/logiciel indispensables pour le développement de solutions embarquées intelligentes. Les notions abordées constituent un socle essentiel pour appréhender les défis actuels liés à l'optimisation des performances, à la fiabilité des systèmes et à l'intégration de l'IA dans des environnements contraints.

2. Historique et progression du marché des microprocesseurs

2.1 Historique

L'histoire des microprocesseurs est intimement liée à l'évolution de l'informatique et à la miniaturisation des circuits électroniques, ce qui a permis le développement de machines plus puissantes, plus rapides et plus accessibles.

Le tout premier microprocesseur a été inventé par Intel en 1971, le **Intel 4004**, souvent considéré comme une percée dans le domaine de l'électronique. C'était un processeur **4 bits** capable de traiter des instructions simples. Il a été initialement conçu pour des calculatrices, mais il a rapidement montré le potentiel des microprocesseurs dans d'autres applications. Le 4004 contenait environ **2 300 transistors** et travaillait à une fréquence de **92,5 kHz**.

Un an plus tard, Intel a développé le **8008**, un processeur **8 bits**, qui contenait environ **2 800 transistors** et pouvait exécuter 60 000 opérations par seconde. Bien qu'il soit plus puissant que le 4004, il était encore limité par rapp ort aux standards actuels.

L'**Intel 8080** a représenté une avancée significative. Bien qu'il s'agisse également d'un processeur 8 bits, il était nettement plus rapide que ses prédécesseurs et a constitué le cœur de nombreux ordinateurs personnels des années 1970, notamment le célèbre Altair 8800. Ce

processeur a joué un rôle clé dans l'essor de l'industrie des micro-ordinateurs. Fonctionnant à une fréquence de 2 MHz, il était compatible avec le système d'exploitation CP/M-80 développé par Digital Research.

Peu après l'Intel 8080, des concurrents comme **Zilog** et **MOS Technology** ont lancé leurs propres processeurs. Le Zilog Z80, compatible avec l'8080, est devenu très populaire dans les systèmes intégrés et les ordinateurs personnels comme le TRS-80 de Radio Shack. MOS Technology a également sorti le 6502, qui a été utilisé dans des machines célèbres comme l'Apple II et la Commodore 64.

L'**Intel 8085** était une version améliorée de l'8080, offrant plus d'efficacité énergétique et une intégration plus facile dans les systèmes embarqués. Ce microprocesseur utilisait moins de composants externes pour fonctionner, ce qui le rendait plus pratique pour les systèmes plus simples.

Le **8086** a marqué une étape cruciale dans l'évolution de l'informatique. Souvent vu comme le précurseur de l'architecture x86, qui reste aujourd'hui au cœur des PC modernes. Il fonctionnait à une fréquence de 4,77 MHz, employait une technologie de **3 µm**, intégrait **29 000 transistors**, et disposait d'un bus d'adresse de 20 bits avec un débit de 9,1 Mo/s, ainsi qu'un bus de données de 16 bits.

La course au leadership entre **Intel** et **Motorola** à la fin des années 1970 et dans les années 1980 est marquée par une série de développements technologiques majeurs dans le domaine des microprocesseurs, qui ont façonné l'industrie de l'informatique personnelle.

En juin 1979, Intel introduit le microprocesseur **8088**, une version 8 bits du 8086 (qui, lui, est un processeur **16 bits**). Cette décision stratégique d'Intel permet à l'entreprise de s'adapter à une demande croissante pour des systèmes compatibles avec les anciens standards tout en offrant une avancée vers la performance des 16 bits. Ce microprocesseur devient un acteur clé lorsqu'il est choisi par IBM pour équiper son tout premier PC en 1981, ce qui donne à Intel une position dominante dans le marché des ordinateurs personnels naissant.

La même année, Motorola lance le **68000**, un microprocesseur révolutionnaire de 16 bits, qui combine des performances de traitement élevées avec une architecture RISC (Reduced Instruction Set Computer). Le 68000 est utilisé dans des systèmes tels que l'Apple Macintosh et l'Atari ST, et son architecture flexible permet à Motorola de concurrencer directement Intel. La conception innovante du 68000, avec ses registres larges et son adresse mémoire de 32 bits, ouvre la voie à une gamme de processeurs puissants qui marquent l'industrie.

Au début des années 1980, Intel continue d'améliorer ses processeurs avec le **80186** et le **80286**. Le 80186 est une amélioration du 8086, avec des capacités intégrées pour réduire les

coûts des systèmes, mais c'est surtout le 80286, introduit en 1982, qui marque une avancée majeure. Le 80286 est un processeur 16 bits capable de gérer jusqu'à 16 Mo de mémoire, ce qui constitue une percée dans l'architecture informatique. Il est utilisé dans la série IBM PC/AT, ce qui renforce la domination d'Intel dans l'industrie des PC.

Motorola riposte en 1982 avec la sortie du **68010**, un processeur entièrement **32 bits**, qui offre des fonctionnalités avancées comme la gestion des exceptions et la commutation de contexte, le rendant attractif pour les environnements multitâches. Bien que le 68010 n'ait pas eu le même succès commercial que ses prédécesseurs, il montre l'engagement de Motorola dans le développement de processeurs puissants et compétitifs.

Intel franchit une nouvelle étape en 1985 avec la sortie du **80386**, un processeur 32 bits véritablement révolutionnaire. Le 80386 est le premier processeur d'Intel à offrir une architecture 32 bits complète, avec la possibilité de gérer une mémoire virtuelle et de passer d'un mode réel à un mode protégé. Cela ouvre la voie aux systèmes d'exploitation modernes, et ce processeur est adopté largement par les fabricants d'ordinateurs, renforçant la position d'Intel.

Motorola continue de rivaliser avec Intel en 1987 avec la sortie du **68030**, une mise à jour du 68020 avec une unité de gestion mémoire améliorée (MMU). Le 68030 est utilisé dans des systèmes haut de gamme comme le Macintosh IIx d'Apple, et se distingue par ses capacités de multitâche, son efficacité en traitement, et ses performances graphiques élevées.

En 1989, Intel solidifie sa domination avec le lancement du **80486**, un processeur 32 bits avec des performances accrues grâce à l'intégration d'une unité de calcul flottant (FPU) et d'un cache de niveau 1. Le 80486 est une avancée majeure en termes de puissance de traitement, permettant des systèmes plus rapides et plus efficaces, et devient un standard pour les ordinateurs personnels et professionnels.

Pendant ce temps, Motorola continue d'innover avec la sortie du **68040**, un processeur 32 bits avancé, utilisé dans des stations de travail graphiques et des serveurs, mais qui n'arrive pas à capter autant de part de marché qu'Intel, en raison de l'adoption massive des standards Intel par l'industrie des PC.

La course entre Intel et Motorola dans les années 1980 se traduit par une série de progrès technologiques, où chaque entreprise introduit des innovations qui repoussent les limites des performances informatiques. Si Intel a fini par s'imposer en grande partie grâce à son partenariat avec IBM et à l'adoption généralisée de ses processeurs dans les ordinateurs personnels, Motorola a joué un rôle crucial dans le développement des architectures alternatives qui ont influencé le design des systèmes embarqués et de certaines machines grand public comme le Macintosh.

En 1993, Intel a marqué une étape importante dans l'évolution des microprocesseurs en lançant le **Pentium**, une puce révolutionnaire qui a permis des performances accrues et une meilleure gestion des calculs complexes. Ce lancement a ouvert la voie à une série d'innovations dans la famille des processeurs Intel.

Deux ans plus tard, en 1995, Intel a introduit le **Pentium Pro**, une version plus performante destinée aux serveurs et aux stations de travail, offrant une capacité de traitement encore plus sophistiquée pour les besoins professionnels.

En 1997, l'innovation s'est poursuivie avec la sortie du **Pentium MMX**, optimisé pour les applications multimédia, et du **Pentium II**, qui a permis une amélioration notable des performances générales des ordinateurs personnels.

Puis, en 1999, Intel a annoncé le **Pentium III**, qui apportait des optimisations supplémentaires pour les calculs complexes et des instructions améliorées pour les applications Internet émergentes.

L'année suivante, en 2000, le **Pentium IV** a été lancé, offrant une vitesse d'horloge plus élevée et des capacités accrues pour les tâches intensives comme les jeux vidéo et le traitement graphique.

En 2003, Intel a franchi une nouvelle étape en lançant l'**Itanium**, un processeur capable d'exécuter plus d'un milliard d'instructions par seconde, démontrant les avancées spectaculaires dans la puissance de calcul.

Ces progrès sont en grande partie dus à l'évolution de l'intégration sur silicium. En effet, l'industrie est passée rapidement des premiers processeurs 4 bits aux processeurs 8 bits, puis aux processeurs 16 bits et 32 bits, offrant à chaque étape des gains en puissance et en efficacité. Aujourd'hui, les **processeurs 64 bits** et **multicœurs** sont devenus la norme, permettant des performances incomparables et l'exécution simultanée de multiples tâches à une vitesse impressionnante.

Les processeurs modernes augmentent continuellement le nombre de cœurs pour améliorer la capacité multitâche et les performances globales. Les processeurs grand public comportent généralement entre **4** et **16 cœurs**, tandis que les processeurs pour serveurs et superordinateurs peuvent en contenir des **dizaines**, voire des **centaines**. Cela permet de traiter des tâches de plus en plus complexes en parallèle, ce qui est crucial pour des applications comme les simulations scientifiques, les analyses de données massives et les jeux vidéo de pointe.

Avec l'essor de l'intelligence artificielle et du machine learning, les processeurs spécialisés dans le traitement des tâches IA sont devenus essentiels. Les unités de traitement graphique (**GPU**) et les processeurs spécialisés tels que les **TPU** (Tensor Processing Units) de Google sont conçus pour exécuter les algorithmes d'IA à grande échelle. Ces puces permettent un traitement efficace des réseaux neuronaux, accélérant les calculs nécessaires pour l'apprentissage automatique, la reconnaissance d'image et de parole, ainsi que les systèmes autonomes comme les véhicules.

Avec la demande croissante pour des appareils mobiles plus puissants et des objets connectés, les microprocesseurs doivent être de plus en plus efficaces sur le plan énergétique. **ARM** domine ce secteur avec ses processeurs à faible consommation, largement utilisés dans les smartphones, tablettes et objets connectés. Les efforts d'optimisation de la consommation d'énergie sont également cruciaux pour les centres de données, où la réduction de la consommation électrique devient une priorité pour minimiser l'empreinte carbone.

La recherche sur les **processeurs quantiques** progresse rapidement, bien qu'ils en soient encore à un stade expérimental pour de nombreuses applications commerciales. Les processeurs quantiques exploitent les principes de la mécanique quantique pour effectuer des calculs à une vitesse exponentiellement supérieure à celle des processeurs traditionnels dans certains domaines, comme la cryptographie, l'optimisation complexe et la simulation de systèmes physiques. Des entreprises comme IBM, Google et Intel investissent massivement dans cette technologie prometteuse.

En conclusion, les microprocesseurs continuent d'évoluer à un rythme rapide, répondant aux besoins croissants en puissance de calcul tout en restant économes en énergie. Les avancées en matière de multicœurs, d'IA, de processeurs spécialisés et d'architectures plus efficaces ouvrent de nouvelles possibilités pour le futur. Parallèlement, les enjeux liés à la durabilité et aux défis écologiques sont de plus en plus pressants, façonnant le développement des technologies de demain.

2.2 Evolution des microprocesseurs

La loi de Moore, énoncée par Gordon Moore, cofondateur d'Intel, en 1965 (et révisée en 1975), observe que le nombre de transistors sur un circuit intégré double environ tous les deux ans. Cette loi n'est pas une loi physique, mais plutôt une tendance technologique, qui a influencé les progrès rapides dans le domaine des semi-conducteurs et de l'informatique.

Elle se fonde sur l'idée que l'augmentation de la densité des transistors entraîne des gains en performance et une réduction des coûts de production par transistor, comme le montre la figue 1. Cela a permis l'essor de l'informatique moderne, où les microprocesseurs sont devenus plus puissants tout en étant de plus en plus abordables.

Depuis le processeur Intel 4004 de 1971, qui contenait 2 300 transistors, cette tendance a fait exploser la capacité des puces modernes, qui intègrent désormais des centaines de millions, voire plusieurs milliards de transistors. Toutefois, la loi de Moore semble aujourd'hui atteindre des limites physiques et économiques, ce qui pousse l'industrie à rechercher de nouvelles approches pour maintenir l'innovation, comme l'architecture des puces 3D et le calcul quantique.

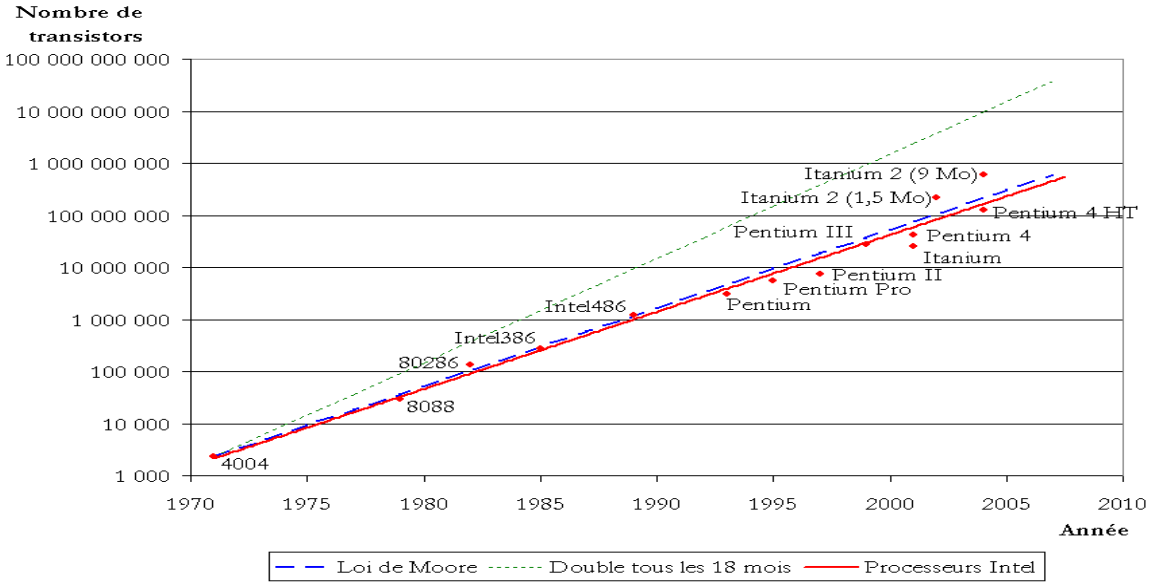


Figure 1. Loi de Moore

La table 1 présente un aperçu des évolutions potentielles de la technologie de gravure, de la complexité des puces et des applications principales à travers différentes périodes, en s'appuyant sur l'historique et les avancées anticipées dans le domaine des semi-conducteurs.

Années	Technologie de Gravure	Complexité	Applications Principales
1970-1980	~10 000 nm	Des milliers de transistors	Calculatrices, premières machines de bureau
1980-1990	~1 000 nm	De dizaines de milliers à centaines de milliers de transistors	Ordinateurs personnels (PC), jeux vidéo, appareils de bureau
1990-2000	~500 - 250 nm	Plusieurs millions de transistors	PC plus puissants, serveurs, premiers téléphones mobiles
2000-2010	~90 - 45 nm	Centaines de millions de transistors	Laptops, smartphones, consoles de jeu, Internet grand public
2010-2020	~22 - 14 nm	Plusieurs milliards de transistors	Smartphones avancés, tablettes, centres de données, IoT

2020 - 2025	~10 - 5 nm	Plus de 10 milliards de transistors	IA, IoT, serveurs hautes performances, véhicules autonomes
2025 et au-delà	~3 nm et moins	Trillions de transistors (ciblé pour les années 2030)	IA avancée, IoT massif, edge computing, réalités virtuelle et augmentée

Table 1. Evolution de la technologie de gravure

La technologie de gravure : La taille en nanomètres (nm) indique la finesse de gravure des transistors, influençant directement la densité et la performance de la puce. Les réductions permettent d'augmenter le nombre de transistors sur une même surface de silicium, mais nécessitent des avancées en lithographie, comme la lithographie extrême ultraviolet (EUV).

La complexité : Le nombre de transistors par puce augmente de manière exponentielle, conformément à la loi de Moore. Cette complexité croissante permet d'exécuter plus d'instructions en parallèle, de gérer des algorithmes plus sophistiqués et d'améliorer la performance et l'efficacité énergétique.

Les applications principales : Avec l'augmentation de la puissance de calcul, les applications se diversifient. Au début, les processeurs étaient utilisés principalement pour des calculs basiques. Aujourd'hui, ils supportent des technologies avancées, comme l'intelligence artificielle, le calcul en périphérie (edge computing), et des applications massives dans l'IoT et l'IA pour les industries et la recherche.

2.3 Domaines d'application du microprocesseur

Les microprocesseurs trouvent des applications variées dans de nombreux domaines. Dans le secteur de l'informatique, ils sont utilisés comme processeurs centraux et intégrés dans divers périphériques intelligents, ainsi que dans les micro-ordinateurs servant de terminaux intelligents. Ils jouent également un rôle essentiel dans les circuits de télécommunication, comme les routeurs et les commutateurs, pour gérer et diriger les flux de données.

Dans le domaine du contrôle, les microprocesseurs sont employés pour piloter des systèmes industriels complexes et interviennent dans les équipements médicaux, permettant une grande précision et fiabilité dans les opérations et les diagnostics.

Le grand public bénéficie également de la présence des microprocesseurs dans des domaines tels que l'automobile, où ils gèrent des systèmes électroniques, la sécurité et l'efficacité énergétique. Ils se retrouvent également dans les appareils électroménagers intelligents, l'aviation, et le secteur aérospatial, offrant des fonctionnalités avancées et une meilleure performance.

Enfin, le domaine militaire utilise les microprocesseurs pour des applications stratégiques telles que les systèmes d'armement, le contrôle satellitaire, les missiles et les sous-marins, où la précision et la fiabilité sont primordiales pour la défense et la sécurité.

3. Généralités sur les systèmes embarqués

3.1 Définitions

Les progrès dans la technologie de fabrication des circuits intégrés permettent aujourd'hui d'intégrer des systèmes entiers sur un seul circuit de faible encombrement, appelé System on Chip (SoC). Cela représente une avancée importante pour la miniaturisation et la performance des dispositifs électroniques.

Un système embarqué, ou "embedded system", est un système mixte composé de matériel et de logiciel qui est entièrement intégré dans l'environnement qu'il contrôle. Ce type de système est généralement autonome et conçu pour exécuter une tâche précise dédiée à une application spécifique.

L'interface homme-machine (IHM) d'un système embarqué peut varier en complexité : elle peut être aussi simple qu'une LED clignotante ou aussi avancée qu'un système de vision de nuit fonctionnant en temps réel.

3.2 Caractéristiques des systèmes embarqués

Les systèmes embarqués présentent plusieurs caractéristiques essentielles qui leur permettent de fonctionner efficacement dans des environnements variés tout en respectant des contraintes strictes :

- **Capacité mémoire limitée** : Les systèmes embarqués disposent souvent d'une mémoire réduite pour des raisons de coût et d'efficacité. Ils doivent donc optimiser leur utilisation de la mémoire pour exécuter leurs tâches sans gaspiller d'espace.
- **Faible consommation d'énergie** : Les systèmes embarqués sont conçus pour minimiser leur consommation d'énergie, car une consommation élevée nécessiterait des batteries de grande capacité, augmentant ainsi le coût de production. Par exemple, les appareils portables, comme les smartphones ou les capteurs sans fil, doivent fonctionner longtemps sur des batteries de petite taille.
- **Résistance aux conditions environnementales** : Les systèmes embarqués doivent pouvoir fonctionner dans des environnements exigeants avec des contraintes comme la chaleur, l'humidité, les vibrations, les chocs, et même les radiations ou les interférences électromagnétiques. Cela leur permet d'être déployés dans des applications comme

l'automobile, l'aéronautique ou l'industrie, où ils sont exposés à des conditions extrêmes.

- **Fiabilité maximale** : La fiabilité est cruciale pour les systèmes embarqués, car ils doivent rester fonctionnels même en cas de défaillances matérielles. Ils sont donc souvent conçus pour tolérer les pannes et pour assurer la sécurité et la stabilité de l'ensemble du système, particulièrement dans les applications critiques (médicales, aérospatiales, etc.).
- **Faible coût de production** : Enfin, le coût de production doit être minimisé. Comme les systèmes embarqués sont souvent fabriqués en grande quantité, il est essentiel de contrôler leur prix de revient pour rester compétitif, tout en maintenant un bon niveau de performance.

Ces caractéristiques permettent aux systèmes embarqués de répondre aux exigences d'efficacité, de durabilité et de sécurité nécessaires pour les applications modernes dans des secteurs variés.

3.3 Importance du marché de l'embarqué

Le marché des microprocesseurs fait référence à la production et à la vente de processeurs, qui sont les circuits intégrés essentiels permettant aux dispositifs électroniques d'exécuter des instructions et des tâches. Historiquement dominé par les ordinateurs personnels (PC), ce marché s'est diversifié pour inclure une multitude de secteurs comme l'automobile, les appareils électroménagers, l'électronique grand public et les dispositifs industriels. Cette diversification est largement due à la croissance des systèmes embarqués, qui utilisent des microprocesseurs pour des applications spécifiques et intégrées.

Dans les PC, des processeurs de type x86 dominent, souvent associés au système d'exploitation Windows. Cependant, ces processeurs ne représentent qu'une petite fraction (environ 5 %) de la demande mondiale. L'essentiel du marché des microprocesseurs (environ 95 %) est désormais destiné aux systèmes embarqués, où des processeurs 8 bits, 16 bits, ou 32 bits plus simples et économes en énergie sont utilisés. Par exemple, dans l'automobile, les systèmes électroniques embarqués jouent un rôle de plus en plus important, représentant jusqu'à 35 % de la valeur totale d'une voiture neuve. Ainsi, le marché des microprocesseurs ne se limite plus aux PC mais s'étend aux systèmes embarqués dans divers produits, permettant une intégration de plus en plus poussée de la technologie dans notre quotidien.

3.4 Systèmes embarqués et temps réel

Les systèmes informatiques peuvent être classés en différentes catégories selon leur manière de traiter les informations et d'interagir avec leur environnement, comme le montre la figure 2.

Chaque type de système répond à des besoins spécifiques en fonction de sa structure et de son rôle.

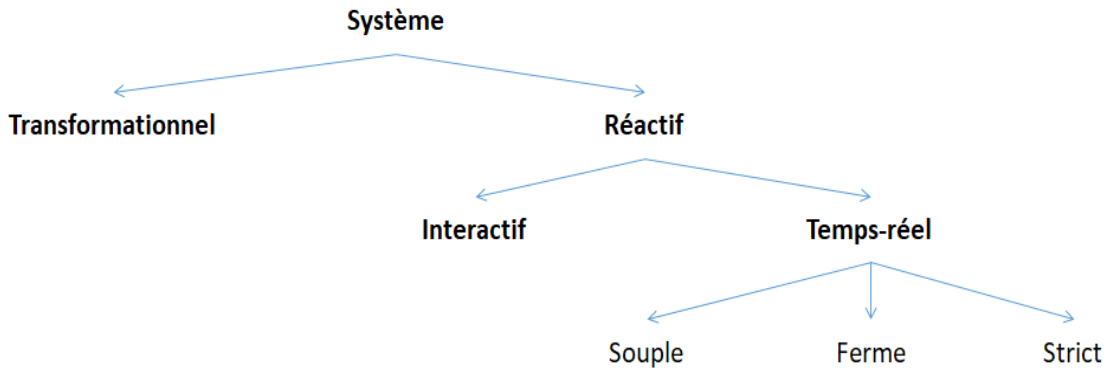


Figure 2. Classification des différents types de systèmes

Un système transformationnel produit des sorties en fonction d'entrées, suivant un processus de calcul qui ne dépend pas de l'environnement extérieur. Il transforme les données d'entrée en résultat final de manière autonome, sans s'adapter aux changements externes. Par exemple, un programme de conversion de devises prend une valeur en entrée (comme un montant en dollars) et la transforme en une autre unité (comme l'euro) en utilisant un taux de conversion fixe. Ce processus de calcul est autonome et ne dépend pas d'événements extérieurs pendant son exécution.

Un système réactif génère des sorties en fonction des entrées reçues et en réponse aux événements qui surviennent dans son environnement. Ce type de système adapte son comportement aux modifications de son contexte, en interagissant en temps réel avec l'environnement pour produire ses réponses. Par exemple, un système d'alarme domestique réagit aux événements tels que l'ouverture d'une porte ou le déclenchement d'un capteur de mouvement. Dès qu'un événement survient, il génère une réponse immédiate, comme un signal sonore ou l'envoi d'une alerte.

Un système interactif réagit aux stimuli reçus en entrée à son propre rythme, sans suivre nécessairement un schéma de réponse immédiat aux événements externes. Il ajuste son traitement en fonction de sa vitesse de fonctionnement interne. Un bon exemple est un assistant virtuel, comme un chatbot ou un système d'aide en ligne. Lorsque l'utilisateur entre une question, le système répond en fonction de sa propre vitesse de traitement, sans exigence de réponse en temps réel, mais en maintenant une interaction fluide.

Un système temps réel est conçu pour fournir des réponses dans un délai précis, souvent dans des situations où un retard dans la réponse pourrait entraîner des conséquences indésirables.

Ces systèmes sont essentiels dans des applications critiques, comme l'automobile ou l'aéronautique, où la précision temporelle est cruciale. Par exemple, un système de freinage antiblocage (ABS) dans une voiture doit détecter le blocage des roues et ajuster la pression de freinage en quelques millisecondes pour éviter un accident. Ici, le délai de réponse est critique pour la sécurité, car un retard pourrait entraîner un échec de la fonction de freinage.

3.4.1 Classification des systèmes temps-réel

Les systèmes temps réel peuvent être classés en fonction de leurs contraintes temporelles, qui déterminent la sévérité des conséquences si les délais ne sont pas respectés. Voici les principales catégories avec leurs définitions et des exemples concrets.

3.4.1.1 Systèmes à contraintes souples (Soft)

Les systèmes à contraintes souples sont moins exigeants en termes de délais stricts. Une petite probabilité de non-respect des délais est tolérée, et les tâches peuvent avoir un léger retard sans compromettre la fonction générale du système. Ces systèmes supportent des retards occasionnels sans conséquences graves, bien que la qualité de service puisse être dégradée.

Voici quelques exemples :

- Programme d'un distributeur automatique de billets : Dans ce cas, le délai pour délivrer l'argent après validation peut être de quelques secondes sans que cela ait des conséquences graves pour l'utilisateur, bien que des retards trop importants puissent altérer l'expérience client.
- Streaming vidéo en ligne : Lors de la diffusion en continu, un léger retard ou quelques interruptions temporaires peuvent survenir sans compromettre l'expérience globale de l'utilisateur.
- Systèmes de reconnaissance vocale : Un assistant vocal peut prendre un peu de temps pour répondre à une commande, sans altérer de manière significative l'expérience de l'utilisateur.
- Jeux en ligne : Un faible niveau de latence est essentiel, mais de petites dégradations sont tolérées sans affecter gravement la jouabilité.

3.4.1.2 Systèmes à contrainte ferme

Ces systèmes requièrent également le respect des délais, mais les conséquences d'un échec ne sont pas catastrophiques. Si une contrainte temporelle n'est pas respectée, la tâche est annulée, et les performances globales du système peuvent être affectées, mais sans compromettre la sécurité ou la fiabilité du système de manière critique.

Voici quelques exemples :

- Systèmes de réservation de billets : Dans des applications de réservation, comme pour le transport aérien ou les concerts, le système doit gérer les demandes en temps réel. Si une demande échoue en raison d'un retard, elle est annulée, mais le système reste globalement opérationnel.
- Répartition de réseau téléphonique : Lors des appels en masse, certaines connexions peuvent être abandonnées si les délais ne sont pas respectés, sans causer de défaillance pour l'ensemble du réseau.
- Contrôle d'un robot : Ce système doit réagir dans un délai de 500 ms pour exécuter une manœuvre de contournement d'objet après la détection d'un obstacle. Si ce délai n'est pas respecté, la tâche est annulée, mais cela n'entraîne pas de conséquences graves, même si la performance du système est dégradée.

3.4.1.3 Systèmes à contraintes strictes ou critiques (Hard, Critical)

Ces systèmes doivent absolument respecter leurs délais, car tout retard entraîne une erreur de fonctionnement pouvant provoquer des conséquences graves, telles que des pertes humaines, des dommages matériels, ou des impacts financiers importants. Ils sont souvent qualifiés de systèmes critiques et soumis à des contraintes de fiabilité très strictes.

Voici quelques exemples :

- Contrôle de processus industriels sensibles : Dans des installations comme les centrales nucléaires, les systèmes de régulation doivent intervenir dans un délai précis pour garantir la sécurité.
- Systèmes embarqués dans les véhicules : Les systèmes de freinage et de suspension des voitures, ou les systèmes de pilotage des avions, nécessitent des réponses immédiates pour assurer la sécurité.
- Médecine assistée par ordinateur : Les dispositifs médicaux, comme les respirateurs ou les moniteurs cardiaques, doivent respecter des délais rigoureux pour éviter des risques pour le patient.
- Systèmes de contrôle des airbags : Dans un accident, l'airbag doit se déployer en quelques millisecondes pour protéger les occupants. Tout retard pourrait être fatal pour les passagers.

Les systèmes temps réel avec des contraintes strictes doivent respecter trois critères essentiels pour assurer leur bon fonctionnement et éviter des erreurs critiques.

1. **Déterminisme logique** : Cela signifie que si on fournit les mêmes données d'entrée au système, il doit toujours produire les mêmes résultats. Par exemple, si un système reçoit une commande, il doit répondre de la même façon chaque fois qu'il la reçoit, ce qui garantit la cohérence et la prévisibilité de ses actions.

2. **Déterminisme temporel** : Ce critère implique que chaque opération doit être effectuée dans un délai précis et fixe. Cela signifie que le système doit respecter des échéances définies pour chaque tâche afin de fonctionner correctement. Si les délais sont imprévisibles ou variables, des erreurs peuvent survenir. Par exemple, un système de freinage de voiture doit réagir dans un certain délai pour éviter un accident.
3. **Fiabilité** : La fiabilité garantit que le système est capable de fonctionner sans erreurs, même dans des conditions difficiles ou si des composants tombent en panne. Un système fiable reste opérationnel et stable, ce qui est crucial dans des applications critiques comme les dispositifs médicaux ou les systèmes de sécurité aérienne, où des pannes peuvent avoir de graves conséquences.

En fonction de leur niveau de criticité, ces différentes classes de systèmes temps réel répondent à des exigences de délai plus ou moins strictes, selon les applications et les environnements dans lesquels ils sont déployés.

3.4.2 Organisation des sous-systèmes temps réel

Un système embarqué est souvent un système réactif, ce qui signifie qu'il doit effectuer des calculs en réponse à des événements extérieurs dans son environnement. Les systèmes embarqués sont composés de plusieurs sous-systèmes, souvent classés en fonction de leurs contraintes de temps :

- Sous-systèmes temps réel critiques : Ces tâches doivent impérativement respecter des délais précis, car leur non-exécution dans le temps imparti pourrait avoir des conséquences graves. Dans une voiture, le contrôle de freinage en est un exemple : un retard pourrait compromettre la sécurité.
- Sous-systèmes temps réel non-critiques : Ces tâches doivent également respecter certains délais, mais un léger retard est tolérable. Ce retard peut dégrader la qualité du service, sans provoquer de problèmes graves. Par exemple, le déclenchement automatique des essuie-glaces en cas de pluie doit répondre rapidement pour le confort du conducteur, mais un retard minime n'aurait pas de conséquences critiques.
- Sous-systèmes sans contraintes de temps réel : Ces tâches n'ont pas d'exigences strictes de délai, mais elles doivent tout de même être effectuées efficacement pour éviter de ralentir le système global. Leur temps d'exécution est minimisé pour optimiser les performances.

En classant les sous-systèmes en fonction de leur criticité et de leurs contraintes de délai, les systèmes embarqués peuvent garantir à la fois la sécurité et la réactivité nécessaires dans des applications comme l'automobile, l'aéronautique, et bien d'autres domaines où la fiabilité est cruciale.

3.5 Applications des systèmes embarques

Les systèmes embarqués sont aujourd'hui indispensables dans de nombreuses applications, couvrant une large gamme de secteurs :

- **Le transport** : Les systèmes embarqués sont présents dans les domaines de l'aéronautique, de l'aérospatiale, de l'automobile et du ferroviaire. Ils assurent des fonctions critiques, telles que le contrôle de freinage, les systèmes de navigation, le pilotage automatique des avions, et les systèmes de sécurité pour les trains.
- **Les appareils électriques et électroniques** : On retrouve des systèmes embarqués dans les appareils du quotidien, comme les appareils photo, les jouets, les téléviseurs, les appareils électroménagers, les systèmes audio, et les téléphones portables. Ces systèmes rendent ces dispositifs plus intelligents et efficaces en intégrant des fonctionnalités avancées.
- **La distribution d'énergie** : Dans ce secteur, les systèmes embarqués contribuent à la gestion et à la distribution de l'énergie. Ils surveillent les réseaux électriques, optimisent la distribution et garantissent un fonctionnement stable des infrastructures.
- **L'automatisation industrielle** : Les systèmes embarqués jouent un rôle clé dans l'automatisation des processus de production en usine. Ils contrôlent et surveillent les machines, améliorent l'efficacité et réduisent les coûts en minimisant les interventions humaines.

Grâce à leur polyvalence et à leurs capacités avancées, les systèmes embarqués sont devenus essentiels dans des domaines variés, offrant des solutions de contrôle, de surveillance et d'optimisation dans des environnements de plus en plus diversifiés.

3.6 Schéma fonctionnel d'un système embarqué

Le schéma fonctionnel d'un système embarqué et ses interactions avec son environnement, comme le montre la figure 3, peut être illustrée par les principales composantes et flux de données qui permettent au système de remplir sa fonction de manière autonome et réactive. Voici les éléments clés d'un tel schéma :

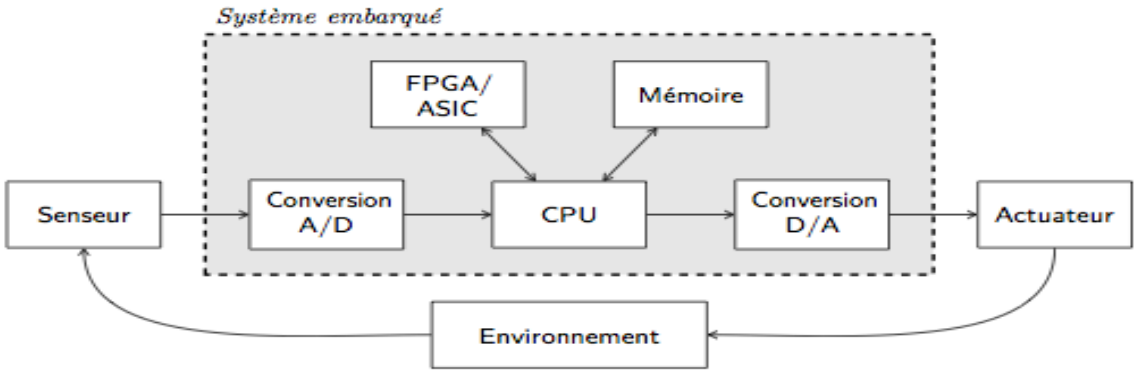


Figure 3. Schéma fonctionnel d'un système embarqué et sa relation avec son environnement

Capteurs : Ils constituent le lien entre l'environnement et le système embarqué en captant des informations extérieures, comme la température, la vitesse, la position, ou la pression. Les capteurs convertissent ces données physiques en signaux électriques, que le système peut traiter. Par exemple, dans une voiture, les capteurs de vitesse ou de distance sont utilisés pour réguler le système de freinage ou d'assistance au conducteur.

Microcontrôleur ou processeur : C'est le cœur du système embarqué, chargé de traiter les informations reçues des capteurs. Il exécute les algorithmes et instructions programmés pour analyser les données d'entrée, prendre des décisions, et envoyer les commandes nécessaires aux actionneurs.

Mémoire : Elle stocke le programme (logiciel embarqué) qui contrôle les fonctions du système et conserve temporairement les données recueillies ou les paramètres d'opération. Dans certains systèmes, une mémoire persistante permet aussi de garder les informations entre les cycles d'alimentation.

Actionneurs : Les actionneurs exécutent les commandes émises par le microcontrôleur pour influencer l'environnement. Par exemple, un système embarqué dans une machine automatisée pourrait commander des moteurs, des vannes, ou des relais. Dans une voiture, cela inclut des éléments comme les freins ou les servos de direction.

Interface de communication : Le système embarqué peut communiquer avec d'autres systèmes ou appareils via des interfaces (comme USB, Ethernet, ou interfaces sans fil). Cela permet l'échange de données, la mise à jour du programme, ou le diagnostic à distance.

Alimentation : Le système embarqué est généralement alimenté par une batterie ou une source d'énergie externe. La gestion de l'énergie est cruciale pour optimiser la durée de vie de la batterie, notamment dans les dispositifs portables ou autonomes.

Le système embarqué interagit constamment avec son environnement en boucle fermée. Les capteurs capturent les données environnementales et envoient ces informations au microcontrôleur, qui les analyse. En fonction de l'analyse, le système peut prendre une action en activant les actionneurs pour ajuster ou contrôler l'environnement. Par exemple, dans un système de régulation de température, le capteur de température envoie les mesures au processeur, qui les compare à la température souhaitée, puis ajuste le chauffage ou le refroidissement via un actionneur pour maintenir des conditions idéales.

Cette boucle permet au système embarqué de répondre rapidement aux changements extérieurs, garantissant ainsi un fonctionnement autonome et précis.

3.7 Exemple typique de système embarqué

Un appareil photo numérique est un exemple typique de système embarqué qui combine plusieurs technologies pour réaliser une tâche spécifique : capturer, traiter et afficher une image. La figure 4 présente une description plus détaillée du fonctionnement de cet appareil et des composants clés qui en font un système embarqué.

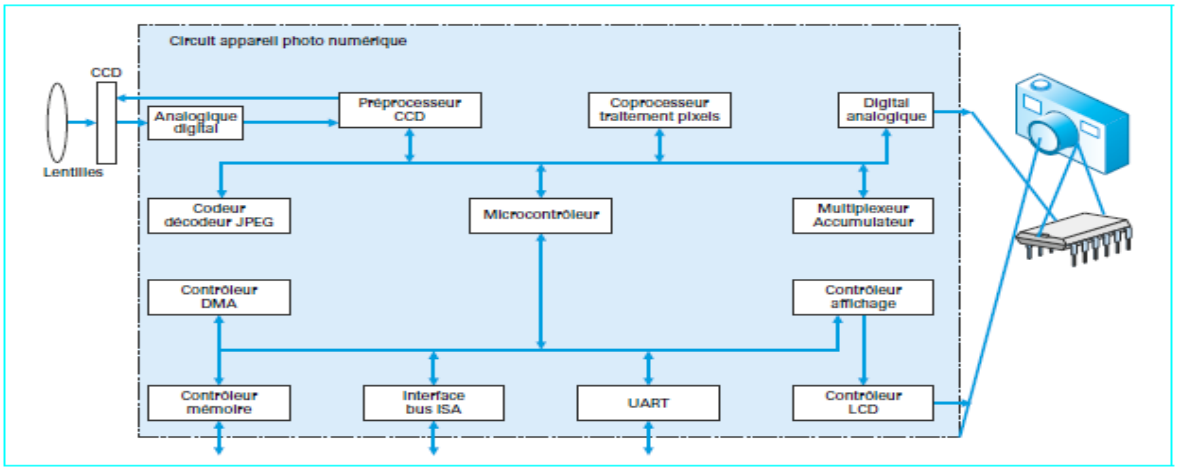


Figure 4. Schéma fonctionnel d'un appareil photo

L'exemple classique d'un appareil photo numérique montre comment il fonctionne avec un processeur simple (microcontrôleur). Ce processeur est connecté à la mémoire, au bus ISA et à l'UART, et il contrôle deux coprocesseurs. Le premier, le préprocesseur CCD, reconstruit l'image à partir des données fournies par le capteur CCD, tandis que le second, le coprocesseur de traitement des pixels, ajuste l'image. L'interface avec l'extérieur comprend une conversion analogique-numérique pour transformer les données du capteur CCD en numérique, et une conversion numérique-analogique pour restituer l'image finale.

4. Composants d'un système embarqué

Un système embarqué est un ensemble cohérent de composants matériels et logiciels conçus pour accomplir une ou plusieurs tâches précises au sein d'un dispositif autonome. Il se compose généralement de capteurs, convertisseurs, actionneurs, d'un calculateur (souvent un microcontrôleur ou un processeur embarqué) et parfois de circuits spécialisés tels que les FPGA ou ASIC, selon les besoins en performance. Ces différents éléments coopèrent pour assurer la perception, le traitement et l'action dans un environnement donné.

4.1 Capteurs : perception de l'environnement

Les capteurs sont les interfaces entre le monde physique et le monde numérique. Leur rôle est de mesurer des grandeurs physiques (température, pression, vitesse, luminosité, etc.) et de les convertir en signaux électriques exploitables par le système.

Les signaux issus des capteurs sont généralement analogiques (variation continue de tension ou de courant). Pour être traités numériquement, ils passent par un convertisseur analogique-numérique (CAN).

Exemples :

- Un capteur de température LM35 produit une tension proportionnelle à la température mesurée (10 mV/°C), ensuite convertie en valeur numérique par un CAN pour être interprétée par le microcontrôleur.
- Un capteur de lumière LDR (Light Dependent Resistor) change sa résistance selon la luminosité. Ce signal est également numérisé pour le traitement.
- Un capteur de position (potentiomètre) ou un accéléromètre MEMS peut fournir des données servant au contrôle d'un robot ou d'un drone.

Les capteurs jouent donc un rôle crucial dans la collecte de données nécessaires à la prise de décision du système embarqué.

4.2 Actionneurs : interaction avec l'environnement

Les actionneurs sont les composants responsables de la réalisation d'une action physique en réponse à une commande émise par le calculateur. Ils transforment un signal électrique en mouvement, lumière, son, ou autre effet physique.

Comme ils reçoivent souvent des signaux numériques provenant du calculateur, ceux-ci doivent être convertis en signaux analogiques via un convertisseur numérique-analogique (CNA).

Exemples :

- Une LED (diode électroluminescente) utilisée pour indiquer un état (ON/OFF, erreur, communication active).

- Un moteur à courant continu (DC Motor) pour déplacer un robot mobile.
- Un servomoteur pour ajuster un angle précis dans un bras robotique.
- Un haut-parleur recevant un signal analogique modulé pour produire du son.
- Un relais permettant de contrôler des dispositifs à forte puissance à partir d'un signal de commande faible.

Les actionneurs constituent ainsi le moyen par lequel le système embarqué agit sur son environnement en fonction des données collectées et traitées.

4.3 Calculateur : cœur du système

Le calculateur est le cerveau du système embarqué. Il assure la gestion des entrées/sorties (E/S), le traitement des données issues des capteurs, et la prise de décision en fonction du programme embarqué.

Il est souvent composé d'un microcontrôleur, d'un microprocesseur, ou d'un système sur puce (SoC), selon la complexité du projet.

Exemples :

- Le microcontrôleur Arduino ATmega328P, utilisé dans de nombreuses applications éducatives et industrielles, intègre processeur, mémoire et ports d'E/S.
- Le Raspberry Pi, un micro-ordinateur à base de processeur ARM, est utilisé pour les systèmes nécessitant plus de puissance de calcul (par exemple, en vision artificielle ou en IoT).
- Un ESP32 combine microcontrôleur et connectivité Wi-Fi/Bluetooth, idéal pour les systèmes embarqués connectés (domotique, capteurs sans fil).

Le calculateur peut également exécuter un système d'exploitation temps réel (RTOS), permettant de gérer plusieurs tâches simultanées avec des délais précis.

4.4 Circuits spécialisés : FPGA et ASIC

Pour certaines applications nécessitant une haute performance, une faible latence ou une parallélisation des traitements, on ajoute des circuits dédiés tels que les FPGA ou ASIC pour assister le calculateur.

Les FPGA (Field Programmable Gate Array) sont des circuits logiques reconfigurables après fabrication. Ils permettent de créer des architectures matérielles personnalisées et sont utilisés comme coprocesseurs matériels.

Exemples :

- Un FPGA Xilinx ou Intel Altera utilisé pour accélérer le chiffrement AES dans les systèmes de sécurité.
- Dans l'automobile, un FPGA peut traiter les images issues de caméras embarquées en temps réel.

Les ASIC (Application Specific Integrated Circuit) sont des circuits spécialisés et non reprogrammables, conçus sur mesure pour une tâche donnée.

Exemples :

- Un ASIC pour traitement vidéo dans un smartphone.
- Un ASIC de calcul cryptographique utilisé dans les cartes bancaires ou les systèmes de blockchain.

Les FPGA sont privilégiés pour le prototypage rapide et les petites séries, tandis que les ASIC sont choisis pour les productions en grande quantité, lorsque la performance et la consommation doivent être optimisées.

4.5 Mémoires et la programmation du matériel

Les systèmes embarqués utilisent différents types de mémoires pour stocker, traiter et protéger les données. Les deux principales catégories sont la **RAM** (Random Access Memory) et la **ROM** (Read Only Memory), chacune ayant des rôles et des caractéristiques spécifiques, comme il est montré dans la figure 7.

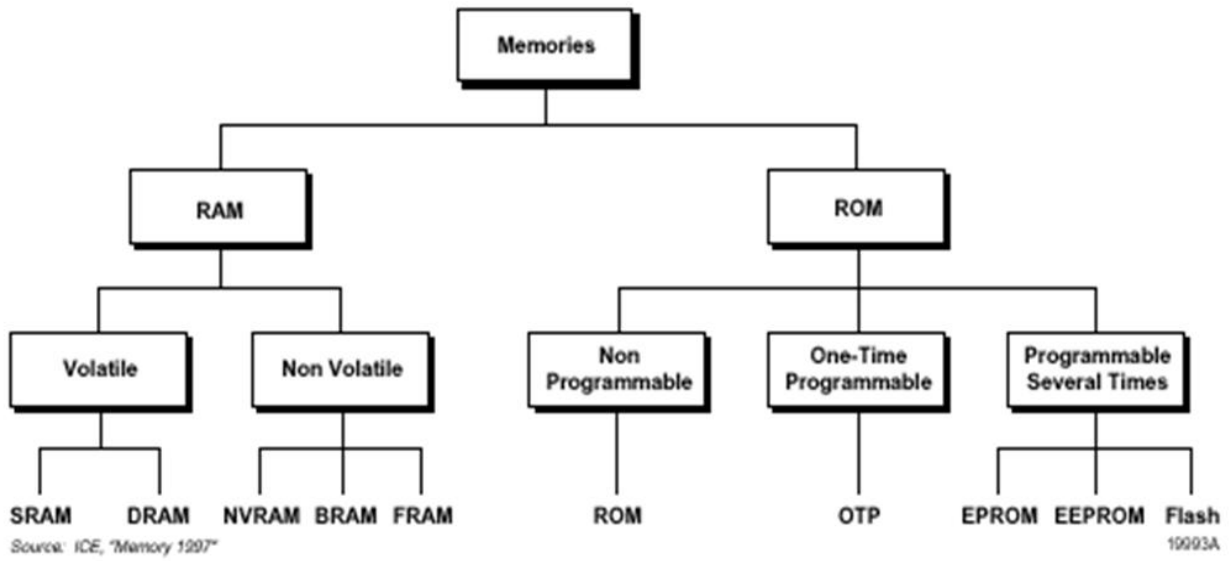


Figure 5. Technologies des mémoires dans les systèmes embarqués.

4.5.1 RAM – Random Access Memory

La RAM est une mémoire à accès aléatoire, ce qui signifie que le processeur peut accéder directement à n'importe quelle adresse mémoire sans devoir suivre un ordre particulier. Elle permet à la fois la lecture et l'écriture de données. Cependant, elle est volatile, c'est-à-dire que son contenu est perdu dès que l'alimentation électrique est coupée.

Dans un système embarqué, la RAM est utilisée pour stocker des informations temporaires, telles que les données d'exécution des programmes ou les informations saisies par l'utilisateur. Elle peut ou non conserver certaines données selon le type de RAM utilisé.

4.5.2 Types de technologies RAM

4.5.2.1 SRAM (Static RAM)

La SRAM conserve les données tant que le système reste alimenté. Elle est composée de transistors et ne nécessite pas de rafraîchissement périodique, ce qui la rend rapide et fiable, mais aussi plus coûteuse. Elle est souvent utilisée dans les mémoires cache ou les petites zones de stockage rapides des processeurs.

4.5.2.2 DRAM (Dynamic RAM)

Contrairement à la SRAM, la DRAM doit être rafraîchie en permanence pour conserver ses données, car elle stocke les informations sous forme de charges électriques dans des condensateurs. Elle est moins chère et offre une plus grande capacité, mais est plus lente. Elle est souvent utilisée comme mémoire principale (RAM centrale) dans les ordinateurs et certains systèmes embarqués puissants.

4.5.2.3 NVRAM (Non-Volatile RAM)

La NVRAM combine les avantages de la RAM et de la ROM. Elle est équipée d'une batterie de sauvegarde qui maintient les données en cas de coupure d'alimentation. Elle se comporte comme une RAM pendant le fonctionnement et comme une ROM lorsqu'elle est hors tension. On la retrouve dans les systèmes critiques (routeurs, contrôleurs industriels, etc.) où la conservation des données est essentielle.

4.5.2.4 FRAM (Ferroelectric RAM)

La FRAM est une mémoire non volatile de nouvelle génération qui conserve les données sans alimentation, tout en offrant une grande rapidité et une faible consommation d'énergie. Elle n'a pas besoin de cycles de rafraîchissement et utilise des cristaux ferroélectriques pour stocker les informations. Elle est de plus en plus utilisée dans les systèmes embarqués basse consommation, comme les dispositifs médicaux ou les cartes à puce.

4.5.2.4.5 MRAM (Magnetic RAM)

La MRAM stocke les données à l'aide de champs magnétiques au lieu de charges électriques. Elle est non volatile, rapide, et résistante aux rayonnements, ce qui la rend idéale pour les applications spatiales et les environnements industriels exigeants.

4.5.3 ROM – Read Only Memory

La ROM est une mémoire en lecture seule, utilisée pour stocker des données permanentes. Contrairement à la RAM, elle est non volatile, c'est-à-dire que les informations qu'elle contient sont conservées même après la coupure d'alimentation. Elle assure une protection contre les modifications accidentelles ou malveillantes, comme les attaques logicielles ou les virus.

La ROM contient souvent le système d'exploitation de base, les paramètres essentiels à l'initialisation du système et les programmes permanents, comme le BIOS dans les ordinateurs.

4.5.4 Types de technologies ROM

4.5.4.1 ROM (Read Only Memory)

Mémoire fabriquée avec un contenu fixe, non programmable, utilisée pour stocker des données permanentes dans les appareils électroniques.

4.5.4.2 PROM (Programmable ROM)

Mémoire programmable une seule fois après fabrication à l'aide d'un programmateur. Une fois écrite, elle ne peut plus être modifiée.

4.5.4.3 EPROM (Erasable Programmable ROM)

Mémoire pouvant être effacée à la lumière ultraviolette et reprogrammée plusieurs fois. Elle est reconnaissable à sa petite fenêtre transparente sur le boîtier.

4.5.4.4 EEPROM (Electrically Erasable Programmable ROM)

Mémoire pouvant être effacée électriquement, sans besoin d'exposition UV. Elle permet une modification bit par bit, ce qui la rend pratique pour stocker des paramètres système ou des configurations utilisateur.

4.5.4.5 Mémoire Flash

Type avancé d'EEPROM qui permet l'effacement par blocs (8 à 64 Ko). Elle est rapide, réinscriptible et compacte, et peut être effacée sans retirer le composant du circuit. Elle est utilisée dans les cartes SD, clés USB, BIOS, microcontrôleurs et systèmes embarqués modernes. Certaines variantes permettent même l'effacement bit par bit, offrant une densité plus élevée et une flexibilité accrue.

4.5 Autres composants

4.5.1 Alimentation

L'alimentation joue un rôle essentiel dans le bon fonctionnement d'un système embarqué. Elle peut provenir de différentes sources selon les besoins du dispositif. Si le système n'a pas besoin d'être portable, on peut utiliser un adaptateur secteur. En revanche, pour les systèmes portables, l'énergie est souvent fournie par une batterie. Dans tous les cas, il faut que l'alimentation offre plusieurs niveaux de tension afin de répondre aux exigences variées des composants du système, tels que : 5.0 V ±0.25 V, 3.3 V ±0.3 V, 2.0 V ±0.2 V et 1.5 V ±0.2 V.

Un bon sous-système d'alimentation doit avant tout fournir une tension stable et régulière. Cette stabilité est cruciale pour certains composants sensibles, comme les convertisseurs analogiques-numériques, qui nécessitent une tension constante pour garantir la précision de la conversion du signal. C'est pourquoi on utilise souvent des régulateurs de tension.

De plus, l'alimentation doit être capable de fournir un courant suffisant pour faire fonctionner l'ensemble du système et de ses périphériques. Cette exigence peut parfois limiter le nombre d'appareils pouvant être alimentés directement par le système embarqué.

L'efficacité énergétique est également un critère important. L'alimentation doit rester performante et stable, même lorsque la température varie ou que la circulation d'air est faible (cas typique dans les systèmes compacts).

Enfin, il est recommandé de séparer l'alimentation de certains composants sensibles pour éviter les interférences. Par exemple, l'horloge et les circuits de reset doivent être isolés des ports d'entrée/sortie et des timers. Les périphériques d'E/S peuvent consommer plus d'énergie que le processeur, tandis que les timers maintiennent une consommation constante, même en mode veille.

4.5.2 Horloge

L'horloge est un élément fondamental dans les systèmes embarqués, juste après l'alimentation. Elle a pour rôle de rythmer le fonctionnement du processeur et de synchroniser l'exécution des instructions. Grâce à ses impulsions régulières, le processeur effectue en continu le cycle fetch–decode–execute : il récupère les instructions depuis la mémoire, les décode puis les exécute.

En plus de l'horloge principale du processeur, les systèmes embarqués intègrent souvent une horloge en temps réel (**RTC** – Real-Time Clock). Ce type d'horloge permet de mesurer le temps avec précision, même lorsque le système est en veille. Les RTC sont utilisées par plusieurs composants, notamment l'ordonnanceur de tâches dans les systèmes à microprocesseur ou

dans les applications temps réel, où la synchronisation et la ponctualité des opérations sont essentielles.

4.5.3 Entrée/Sortie

Les systèmes embarqués doivent être capables d'échanger des informations avec leur environnement. Pour cela, ils utilisent des dispositifs d'entrée et de sortie. Les entrées permettent de recevoir des données provenant de capteurs, de boutons ou d'autres périphériques externes, à travers des ports d'entrée ayant chacun une adresse spécifique. De la même manière, les sorties servent à envoyer des informations vers l'extérieur, par exemple vers des voyants lumineux (LEDs) ou des écrans LCD, qui disposent eux aussi d'adresses particulières.

Un exemple typique de périphérique d'entrée/sortie est le port **GPIO** (General Purpose Input/Output). Il permet de relier directement le processeur aux éléments externes ou à d'autres dispositifs électroniques.

La transmission des données entre les différents composants peut se faire de deux façons : en série ou en parallèle. Dans une communication série, les bits sont envoyés un à un, tandis qu'en parallèle plusieurs bits sont transmis simultanément. De nos jours, la transmission parallèle est surtout utilisée pour des échanges entre composants très proches, alors que la communication série est privilégiée pour des distances plus longues.

Enfin, les bus de communication peuvent fonctionner selon deux modes : **synchrone** ou **asynchrone**. Dans un bus synchrone, une horloge commune rythme la transmission et garantit la validité des données échangées. En revanche, un bus asynchrone n'utilise pas d'horloge partagée ; il emploie plutôt des signaux spéciaux pour indiquer le début et la fin des données valides.

4.5.3 Watchdog

Le Watchdog a pour rôle principal de réinitialiser le microcontrôleur lorsqu'il se retrouve dans un état indéfini, souvent causé par une erreur ou un blocage du programme. Autrement dit, il agit comme un système de sécurité qui redémarre automatiquement le dispositif en cas de dysfonctionnement.

Il existe deux types de watchdogs : internes et externes.

Le watchdog interne est directement intégré dans le microcontrôleur.

Le watchdog externe, quant à lui, est un circuit indépendant qui peut assurer d'autres fonctions supplémentaires, telles que la surveillance de l'alimentation et du signal d'horloge du système.

En général, un circuit de watchdog externe est conçu pour vérifier régulièrement le bon fonctionnement du processeur et intervenir en cas d'anomalie. L'utilisation d'un watchdog, qu'il soit interne ou externe, est fortement recommandée dans tout système embarqué afin de garantir sa fiabilité et d'éviter les blocages.

5. Approche de conception des systèmes embarqués (Codesign)

La conception des systèmes embarqués repose sur une approche méthodique visant à intégrer de manière harmonieuse les composants matériels et logiciels afin de répondre à des exigences strictes de performance, de fiabilité et de consommation énergétique. Cette approche s'appuie sur des étapes bien définies — de la spécification à la validation — et sur des outils de modélisation et de simulation permettant d'optimiser le système avant sa mise en œuvre. Cependant, malgré les progrès des méthodes de conception, plusieurs défis persistent et rendent le processus de développement de plus en plus complexe.

Le premier problème réside dans **l'hétérogénéité des applications mixtes**. En effet, les systèmes embarqués modernes intègrent à la fois des composants matériels et logiciels, souvent issus de domaines différents, ce qui rend leur intégration complexe et exige une approche de conception multidisciplinaire.

Le deuxième problème concerne **la sûreté et la robustesse du système**. Un système embarqué doit fonctionner de manière fiable, même dans des environnements contraignants ou face à des défaillances partielles. Il est donc essentiel de garantir une sécurité et une tolérance aux fautes élevées tout au long de son cycle de vie.

Le troisième problème est lié à **la complexité grandissante des systèmes embarqués**. Ces systèmes deviennent de plus en plus sophistiqués, ce qui rend difficile la recherche immédiate d'une solution optimale, la détection et la correction des erreurs, ainsi que la maintenance du système en fonctionnement normal. Cette complexité croissante impose de nouvelles méthodes de modélisation, de validation et de test pour assurer la cohérence globale du système.

Les délais de conception de plus en plus courts constituent un quatrième défi majeur. Dans un contexte de forte concurrence, les entreprises cherchent à réduire au maximum le temps de développement et de mise sur le marché. Par exemple, dans l'industrie automobile, la durée de développement des composants électroniques est passée de 3–5 ans à seulement 1–3 ans. Cette pression temporelle nécessite des outils et des méthodologies de conception plus efficaces et automatisées afin d'assurer la compétitivité tout en maintenant la qualité.

La conception des systèmes embarqués peut être abordée selon deux grandes méthodologies : la **conception traditionnelle** et la **méthodologie de codesign**.

5.1 Conception traditionnelle

Dans l'approche classique, les concepteurs matériels (Hardware) et logiciels (Software) travaillent généralement de manière séparée, avec des objectifs et des contraintes propres à chaque domaine. Les concepteurs matériels peuvent avoir une mauvaise appréciation des besoins réels en ressources, tandis que les concepteurs logiciels exploitent parfois mal les capacités offertes par le matériel. Cette séparation conduit souvent à des problèmes majeurs lors de la phase d'intégration, nécessitant des modifications importantes au niveau du matériel et/ou du logiciel. Cette méthode, bien qu'adaptée à des systèmes simples, devient rapidement inefficace face à la complexité croissante des systèmes embarqués modernes.

5.2 Méthodologie de conception « Codesign »

Face à ces limites, la méthodologie de codesign matériel/logiciel est apparue comme une évolution naturelle de la conception des systèmes embarqués. Cette approche repose sur une conception conjointe du matériel et du logiciel, en tenant compte dès le départ des interactions entre les deux. Elle vise à intégrer les différents composants d'un système mixte tout en respectant les fonctionnalités et les contraintes associées (performance, consommation, coût, etc.). Contrairement à la méthode traditionnelle où les choix matériels sont effectués dès le début, le codesign permet de retarder ces décisions jusqu'à une phase plus avancée du processus, offrant ainsi une plus grande flexibilité et une optimisation globale du système.

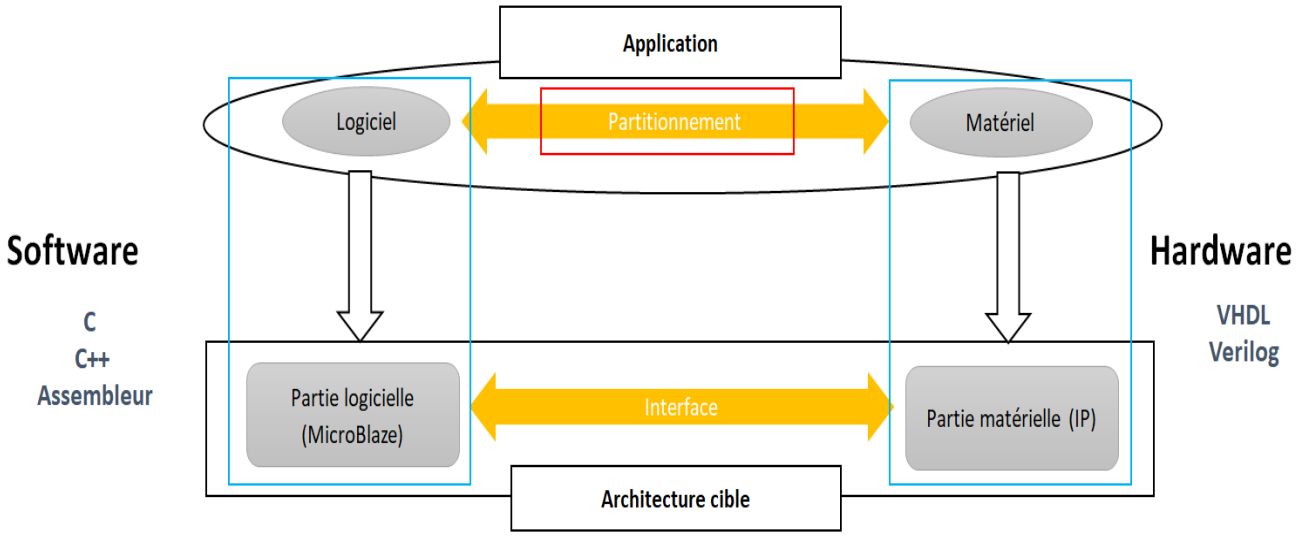


Figure 6. Principe du codesign

5.2.1 Avantages du codesign

Le codesign présente de nombreux avantages : il améliore les performances grâce au parallélisme, aux architectures spécialisées et aux algorithmes distribués ; il garantit une meilleure indépendance vis-à-vis des évolutions technologiques, et favorise une plus grande flexibilité dans le développement grâce à l'utilisation de différents langages de programmation. De plus, il permet une exploration efficace de l'espace de conception, une réduction des coûts par la réutilisation de composants IP (Intellectuelle propriété), ainsi qu'une diminution des phases d'intégration et de test. En conséquence, le prototypage rapide et la réduction des délais de mise sur le marché deviennent des atouts essentiels pour la compétitivité.

5.2.2 Contraintes et exigences du codesign

Cependant, la mise en œuvre du codesign impose certaines contraintes. Concevoir un système embarqué répondant à toutes les exigences de performance, de consommation et de fiabilité nécessite des compétences multidisciplinaires. Le concepteur doit maîtriser à la fois les aspects matériels et logiciels — tels que les microprocesseurs, DSP, FPGA, langages VHDL et C/C++, systèmes temps réel et interfaces d'entrée/sortie. En outre, une bonne connaissance des systèmes numériques et la capacité à travailler au sein d'équipes pluridisciplinaires sont indispensables pour mener à bien un projet de codesign. Ainsi, le codesign s'impose aujourd'hui comme une méthodologie incontournable pour la conception de systèmes embarqués performants, flexibles et adaptés aux contraintes industrielles modernes.

5.3 Types d'architectures dans les systèmes embarqués

La conception d'un système embarqué repose en grande partie sur le choix de l'architecture matérielle adaptée. Cette décision est cruciale, car elle influence directement les performances, le coût, la flexibilité et le temps de développement du système. Le choix de la plateforme d'implémentation, appelée aussi architecture cible, vise à atteindre un fonctionnement correct du système tout en minimisant le coût et en respectant diverses contraintes telles que la consommation énergétique, la taille ou la fiabilité.

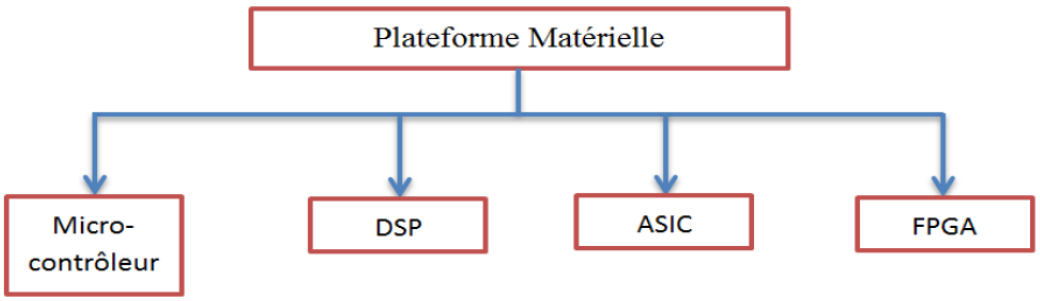


Figure 7. Plateforme d'implémentation

Les processeurs à usage général (ou microprocesseurs) sont largement utilisés pour leur souplesse, rendue possible grâce aux différents langages de programmation. Ils offrent une solution économique, adaptée aux systèmes ne nécessitant pas une puissance de calcul élevée. Cependant, leur principal inconvénient réside dans leur vitesse relativement limitée, ce qui restreint leur utilisation dans des applications exigeant des traitements intensifs.

Les DSP sont des processeurs spécialisés dans le traitement numérique du signal. Ils sont optimisés et plus rapides pour les opérations mathématiques récurrentes, telles que la convolution ou la transformation de Fourier. Programmables comme les processeurs classiques, ils offrent de bonnes performances pour les applications audio, vidéo ou de communication. Toutefois, ils sont plus coûteux qu'un microprocesseur en raison de leur conception dédiée à des tâches spécifiques.

Les FPGA représentent une solution intermédiaire entre les processeurs programmables et les circuits dédiés. Ce sont des composants matériels reconfigurables, offrant une grande souplesse tout en permettant une exécution parallèle des traitements, ce qui les rend plus rapides qu'un processeur. Bien qu'ils soient plus coûteux qu'un microprocesseur, ils demeurent moins onéreux qu'un circuit dédié (ASIC). Leur flexibilité en fait un excellent choix pour le prototypage et les petites séries de production.

Les ASIC sont des circuits intégrés spécifiques à une application donnée. Ils offrent des performances très élevées et une consommation optimisée, mais au prix d'un coût de conception élevé et d'un manque total de flexibilité une fois fabriqués. Leur utilisation est donc justifiée uniquement dans les cas de production en grande série, où le coût initial de conception peut être amorti sur un grand nombre d'unités.

Le choix entre ces différentes architectures dépend des exigences du système embarqué à concevoir, notamment du volume de production, du budget, du délai de développement et des performances attendues. Pour des systèmes simples et peu coûteux, les microcontrôleurs ou microprocesseurs constituent une solution adéquate. Lorsque les besoins en calcul augmentent, les DSP deviennent plus appropriés. En revanche, pour des applications nécessitant des performances supérieures, les FPGA ou ASIC sont privilégiés.

On distingue ainsi deux grandes familles de plateformes : Les systèmes sur circuits ASIC (**SoC** – System on Chip), adaptés à la production de masse grâce à leur performance et leur efficacité énergétique. Les systèmes sur circuits programmables (**PSoC** – Programmable System on Chip), reposant sur des FPGA, idéaux pour le prototypage rapide et la production à faible volume.

6. FPGAs (Field Programmable Gate Array)

Les FPGA sont des circuits logiques reconfigurables et à haute densité, capables d'être reprogrammés même après leur fabrication. Ils ont été développés pour pallier le manque de flexibilité des SoC traditionnels, notamment ceux basés sur des ASICs, dont la configuration est figée une fois conçue. Comme les ASICs, les FPGAs peuvent implémenter n'importe quelle fonction logique, tout en offrant l'avantage d'une reprogrammation rapide et adaptable.

Un FPGA est composé de plusieurs éléments essentiels, comme le montre la Figure 8.

- Une matrice de blocs logiques reconfigurables (**CLB**) : ces blocs constituent le cœur du FPGA. Chaque CLB comprend des éléments logiques programmables tels que des registres (bascules), des tables de correspondance (LuTs), des multiplexeurs et diverses portes logiques, organisés sous forme de matrice.
- Des blocs spécialisés : on y trouve des mémoires (BRAMs), des multiplicateurs, des blocs DSP pour le traitement numérique du signal, ainsi que parfois des processeurs embarqués.
- Des blocs d'entrée/sortie configurables (I/O) : ils permettent au FPGA de communiquer avec l'environnement extérieur et d'interfacer différents périphériques.
- Un réseau d'interconnexion programmable : il relie l'ensemble des blocs logiques et les cellules d'E/S à l'aide de commutateurs et de boîtes de communication configurables.

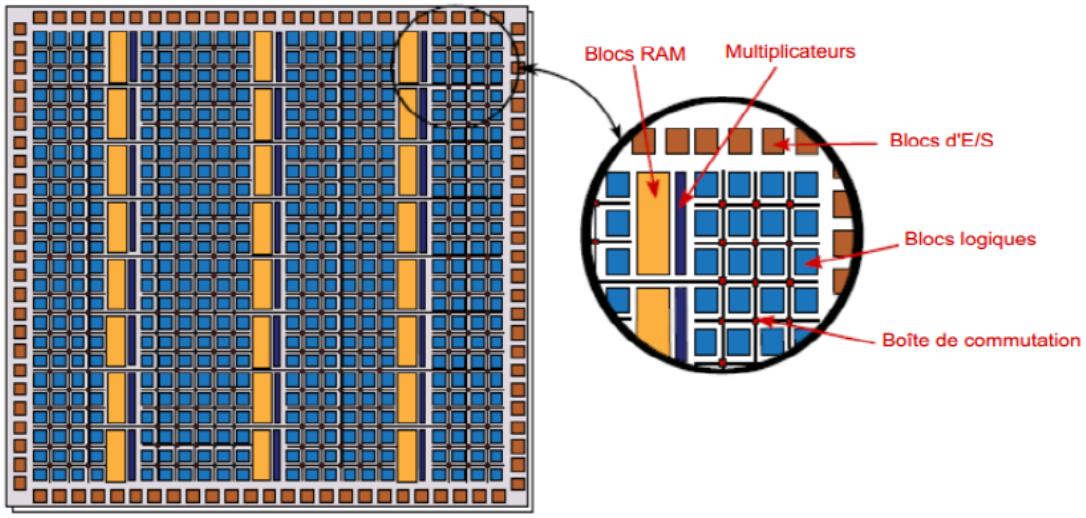


Figure 8. Différents composants d'un FPGA.

Grâce à cette architecture flexible, les FPGAs offrent un excellent compromis entre performance, adaptabilité et rapidité de développement.

Les FPGA se distinguent des anciens circuits logiques programmables comme les SPLD et les CPLD. Au lieu d'utiliser de simples matrices logiques AND/OR, ils reposent sur des blocs logiques complexes (appelés macrocells) reliés entre eux par des interconnexions programmables.

Chaque macrocell contient généralement une LuT (Look-up Table), qui est une petite mémoire intégrée. Cette mémoire stocke à l'avance les résultats d'opérations logiques pour différentes combinaisons d'entrées, un peu comme une table de vérité avec une seule sortie. Ainsi, au lieu de recalculer chaque opération (comme AND, OR, XOR, etc.), le FPGA lit directement le résultat dans la mémoire, ce qui rend l'exécution beaucoup plus rapide.

La taille d'une LuT dépend du nombre d'entrées : par exemple, une LuT3 possède 3 entrées et peut stocker $2^3 = 8$ valeurs possibles. C'est donc une table de vérité 3×1, comme le montre la Figure

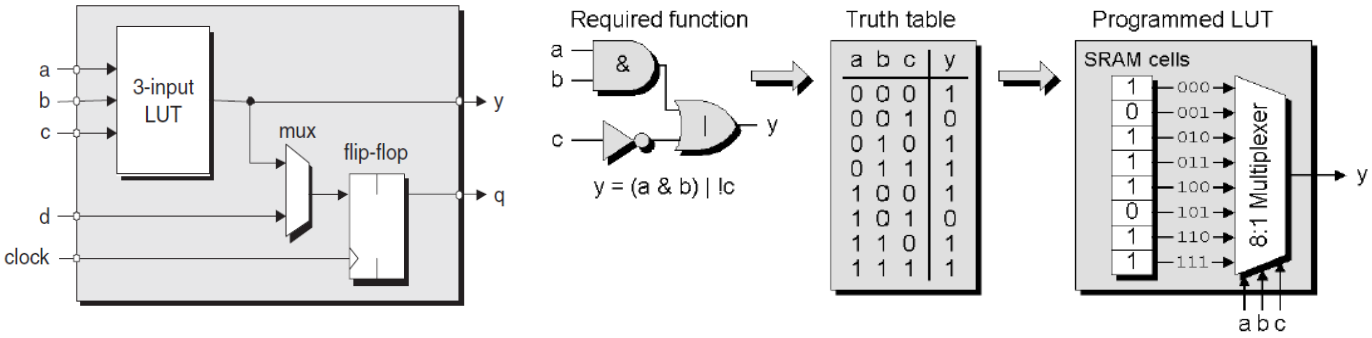


Figure 9. LuT3

Les LuTs jouent un rôle fondamental dans les FPGA, car elles permettent d'implémenter une grande variété de fonctions logiques de manière flexible et efficace sur le matériel.

Aujourd'hui, il est clair que les FPGAs occupent une place essentielle dans de nombreux domaines d'application, allant des systèmes embarqués et des télécommunications jusqu'aux systèmes de contrôle industriel, à l'aéronautique et à l'intelligence artificielle. Leur succès repose principalement sur leur **flexibilité**, leur **vitesse d'exécution**, et leur **facilité d'intégration** dans des architectures complexes.

Intégrer un système embarqué dans un circuit FPGA est devenu une approche moderne et très efficace. Cette méthode permet de combiner sur une même puce des fonctions matérielles et logicielles, offrant ainsi une grande adaptabilité et une forte optimisation des performances.

Le processus de conception sur FPGA se déroule en plusieurs étapes, illustrées dans la figure 10. À l'aide d'un outil de développement comme ISE ou Vivado, le concepteur commence par écrire le code de description matérielle (en VHDL ou Verilog). Le synthétiseur transforme ensuite ce

code en une Netlist, c'est-à-dire une description structurelle de l'architecture du circuit indiquant les connexions entre les différents blocs logiques.

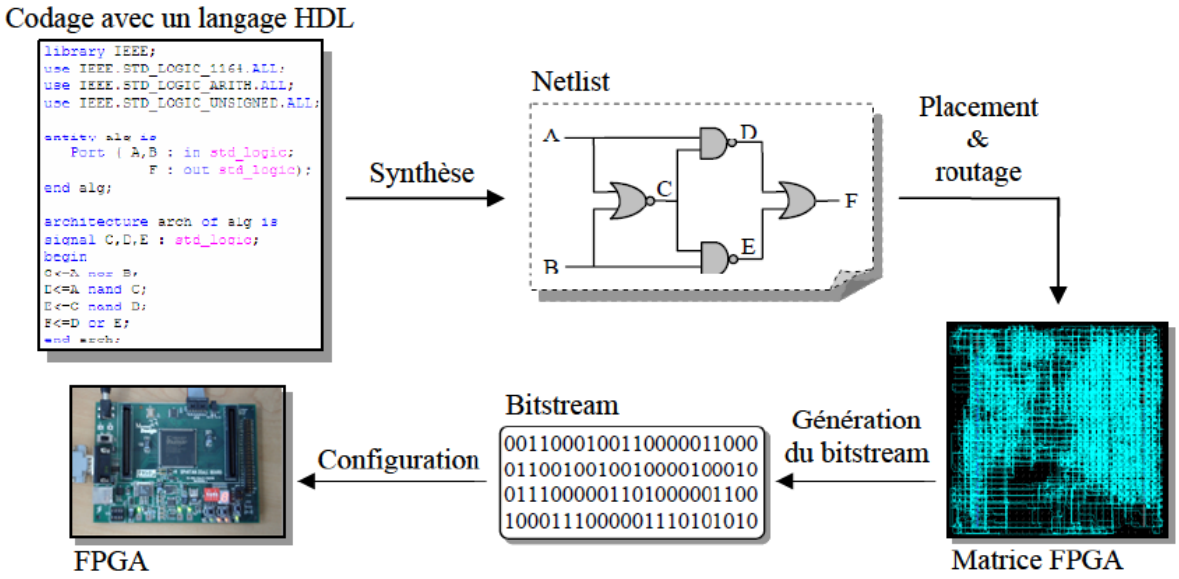


Figure 10. Développement d'un système embarqué sur FPGA

Vient ensuite la phase de placement et routage, où l'outil de conception positionne les composants logiques de manière optimale dans la matrice du FPGA et établit les interconnexions nécessaires entre eux. Ces deux étapes aboutissent à la génération d'un fichier de configuration, appelé Bitstream. Ce dernier contient toutes les informations nécessaires pour configurer le FPGA, notamment la fonction de chaque cellule logique, les interconnexions entre les blocs et les paramètres d'horloge. Il peut ensuite être chargé directement dans le FPGA à partir d'un ordinateur hôte, rendant le circuit prêt à exécuter l'application prévue.

L'un des grands avantages des FPGAs est la disponibilité d'une vaste bibliothèque de composants embarqués, qui peuvent être soit des HardCores (blocs matériels intégrés, comme des processeurs ARM ou des unités DSP), soit des SoftCores (composants programmables implémentés dans la logique du FPGA, comme des processeurs MicroBlaze, des mémoires ou des contrôleurs personnalisés).

Grâce à cette architecture modulaire et reconfigurable, les FPGAs offrent une plateforme idéale pour le codesign matériel/logiciel, alliant performance, évolutivité et réduction du temps de développement.

6.1 Architecture d'un système embarqué sur FPGA

L'architecture d'un système embarqué sur puce (de type SoC ou PSoC) réunit sur un même circuit plusieurs éléments essentiels : un ou plusieurs processeurs pour le contrôle, un bus système pour les

échanges de données, un générateur d'horloge, de la mémoire interne, ainsi qu'un ensemble de périphériques ou de coprocesseurs dédiés à des tâches spécifiques. Le circuit intègre également des interfaces de communication permettant d'échanger des données avec l'extérieur.

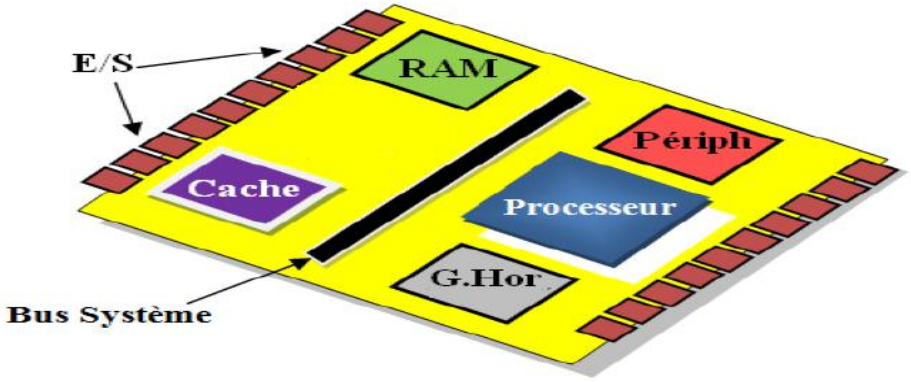


Figure 11. Architecture d'un système embarqué sur FPGA

Grâce à cette intégration de nombreux composants différents, ce type de système est qualifié d'architecture hétérogène. L'approche PSoC combinée au concept de codesign matériel/logiciel (Hw/Sw) présente plusieurs avantages :

- Elle offre de meilleures performances temporelles qu'une solution purement logicielle.
- Elle permet de mettre à jour ou modifier facilement une application existante sans tout refaire.
- Elle assure une grande flexibilité, car il est souvent possible de modifier uniquement la partie logicielle pour adapter le système à une nouvelle application.

Dans la conception d'un tel système, le matériel (Hardware) et le logiciel (Software) se complètent. Lorsque le contrôle ou la logique de l'application est complexe, il est préférable d'ajuster ou d'enrichir la partie logicielle. En revanche, si l'objectif est d'améliorer la vitesse d'exécution ou la performance, il est plus efficace d'ajouter ou d'optimiser la partie matérielle.

La figure 12 illustre la relation entre le coût de conception et les performances du système. On observe que plus le coût augmente, plus la solution tend à être basée sur du matériel seul. Cependant, ce coût a toujours une limite raisonnable, et le codesign matériel/logiciel permet d'obtenir un bon compromis entre performance et coût.

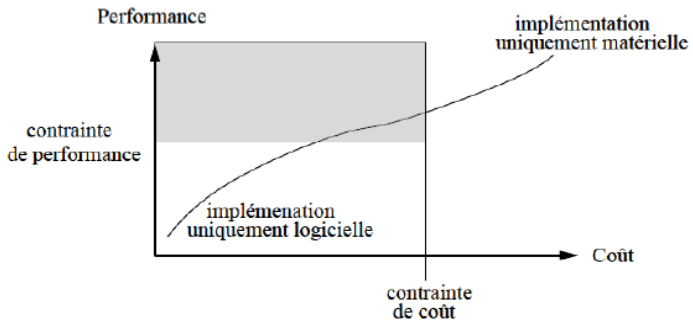


Figure 12. Relation entre le coût de conception et les performances du système dans les systèmes embarqués

6.2 Partitionnement matériel/logiciel

La conception d'un système embarqué sur puce repose sur une série de décisions successives, généralement réparties en trois grandes étapes :

- La sélection de la plateforme d'implémentation : il s'agit de choisir si le système sera réalisé entièrement en matériel (Hardware), en logiciel (Software), ou selon une combinaison des deux.
- Le partitionnement des fonctions : cette étape consiste à répartir les différentes fonctions du système entre le matériel et le logiciel, en s'appuyant sur des bibliothèques de composants disponibles.
- L'ordonnancement des tâches : il permet de déterminer dans quel ordre et dans quels délais les différentes fonctions doivent être exécutées.

Parmi ces étapes, le partitionnement est la phase la plus délicate du processus de conception. Elle consiste à définir quelles parties de l'application seront implémentées en matériel et lesquelles le seront en logiciel. L'objectif est de réduire le coût global du système tout en maintenant les performances requises. En transférant une partie des fonctions vers le logiciel, on limite l'utilisation du matériel spécifique, ce qui permet d'optimiser la surface occupée et la consommation d'énergie.

Dans les applications simples, où les besoins en calcul sont faibles, l'ensemble du système peut être implémenté entièrement en logiciel : le programme est alors stocké en mémoire et exécuté par le processeur. En revanche, pour des applications exigeant de hautes performances, notamment lorsque la quantité de données à traiter est importante, le processeur se charge des opérations simples et du contrôle général, tandis que les opérations complexes sont confiées à des blocs matériels dédiés placés autour du processeur principal.

Pour réaliser un bon partitionnement, il est d'abord nécessaire de décomposer l'application en plusieurs fonctions, puis d'en construire des modèles mathématiques. Ces modèles s'appuient sur différentes techniques, telles que les méthodes de regroupement (clustering) ou les algorithmes génétiques, afin d'obtenir une répartition optimale entre matériel et logiciel.

Enfin, ce partitionnement doit tenir compte de plusieurs contraintes essentielles, notamment :

- L'espace occupé par les éléments matériels et logiciels ;
- Le temps d'exécution des fonctions dans chaque domaine (Hw ou Sw) ;
- Le coût et la latence des communications entre les modules (Hw-Hw, Hw-Sw, Sw-Sw) ;
- Et d'autres paramètres liés à la consommation, la fiabilité ou la complexité du design.

6.3 Méthodologie de conception d'un système embarqué sur FPGA

L'un des principaux avantages de développer des applications embarquées sur une plateforme PSoC est la possibilité d'y ajouter des blocs IP personnalisés, comme il est montré sur la Figure 13. Ces blocs, implémentés directement en matériel, peuvent être modifiés ou améliorés à tout moment pour obtenir des versions plus performantes du système.

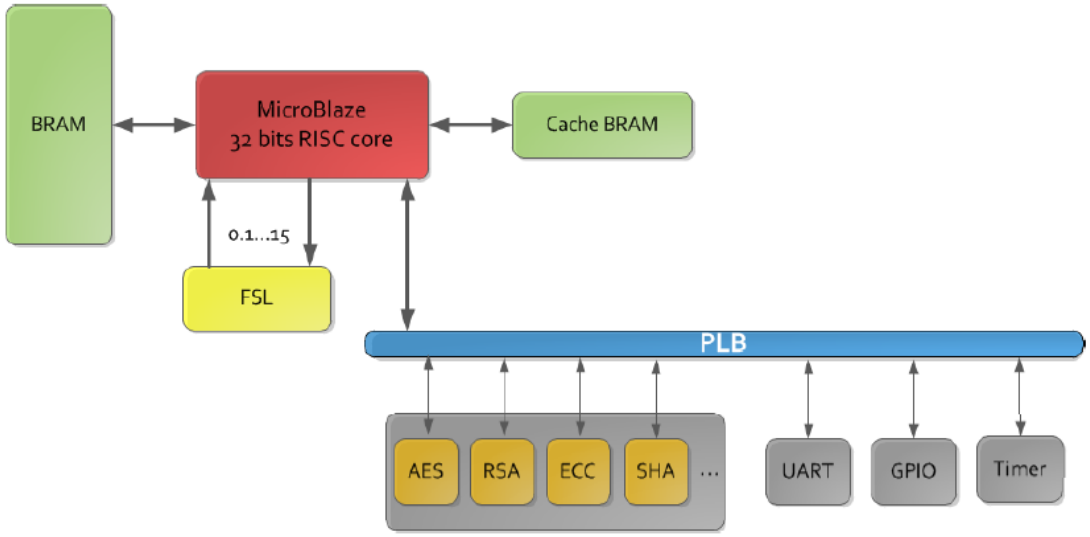


Figure 13. Méthodologie de conception d'un système embarqué sur FPGA.

L'intégration d'un bloc IP autour d'un processeur peut se faire de deux manières principales :

- Implémentation en tant que co-processeur via une liaison **FSL** (Fast Simplex Link) : Dans cette approche, le bloc IP agit comme un co-processeur relié directement au microprocesseur à travers une liaison FSL. Cette méthode est utilisée lorsque le système nécessite un débit de communication élevé entre le processeur et le bloc IP. Les liaisons FSL sont indépendantes du bus système principal et disposent de leurs propres instructions, permettant à la partie logicielle d'accéder directement au bloc IP. Ces instructions, généralement écrites en langage C, sont spécifiques à l'architecture du processeur et offrent une communication rapide et efficace.
- Implémentation via le bus système **OPB** ou **PLB** : Dans ce cas, la communication entre le processeur et le bloc IP passe par le bus système (OPB ou PLB). Une zone mémoire dédiée est alors réservée dans la BRAM pour permettre l'échange de données entre le bloc IP et le bus. Ce mode d'implémentation est le plus couramment utilisé, bien qu'il

soit plus lent et plus complexe à gérer. Cette complexité provient du nombre d'éléments impliqués dans le transfert des données — notamment la mémoire et le bus système. Pour assurer la communication correcte, une interface spécifique, appelée IPIF (IP Interface), est utilisée. Son rôle est de gérer le transfert des données entre le bloc IP et le bus, tout en décodant le protocole de communication.

En résumé, le recours à des blocs IP personnalisés sur une plateforme PSoC offre une grande flexibilité d'évolution et permet d'adapter les performances matérielles du système en fonction des besoins, que ce soit par une liaison rapide (FSL) ou une intégration plus standardisée via le bus système (OPB/PLB).

6.4 Cycle de conception des systèmes embarqués sur FPGA

De façon générale, la conception d'un système embarqué basé sur un processeur implémenté sur un FPGA Xilinx se déroule en trois grandes étapes :

- Conception de la partie matérielle : Cette étape consiste à configurer le processeur, à définir ses périphériques de base (mémoires, interfaces de communication, minuteries, etc.) et à ajouter les blocs IP personnalisés nécessaires à l'application. C'est ici que l'architecture matérielle complète du système embarqué est mise en place.
- Développement de la partie logicielle : Une fois le matériel configuré, il faut développer la partie logicielle du système, notamment l'API (Application Programming Interface). Cette interface logicielle assure la gestion, la coordination et la synchronisation entre les différents composants matériels. Elle permet également au programme principal de contrôler le fonctionnement global du système.
- Chargement sur le FPGA : La dernière étape consiste à générer et transférer le fichier de configuration sur le FPGA. Ce chargement permet de programmer le circuit et de rendre le système opérationnel sur la plateforme matérielle choisie.

Ainsi, ces trois étapes — matériel, logiciel et implémentation — forment le cycle complet de conception d'un système embarqué sur FPGA, comme le montre la Figure 14.

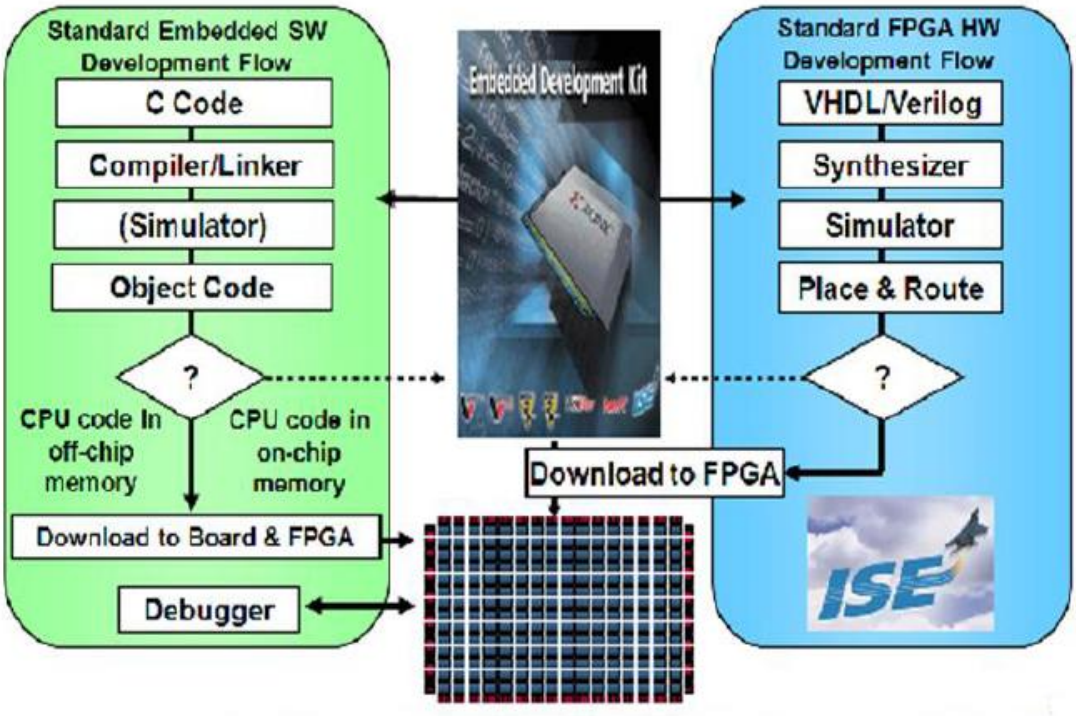


Figure 14. Cycle de conception des systèmes embarqués sur FPGA.

6.5 Modes de configuration FPGA

Les FPGA peuvent être configurés de plusieurs façons pour exécuter des fonctions logiques spécifiques. Ces modes de configuration se divisent en modes externes et modes internes, chacun adapté à des besoins précis.

6.5.1 Modes de configuration externes

6.5.1.1 Port série (Serial Configuration Port)

Le FPGA reçoit les données de configuration bit par bit via un port série. Ce mode est simple et économique, mais plus lent.

6.5.1.2 Port SelectMap

Ce mode permet de configurer le FPGA en parallèle ou de ne modifier que certaines parties du circuit, sans réinitialiser tout le dispositif. Il est utilisé pour la reconfiguration partielle.

6.5.1.3 Port JTAG (Joint Test Action Group)

Le port JTAG est l'un des plus utilisés. Il sert à :

- Charger le bitstream (fichier de configuration) dans le FPGA.
- Déboguer et analyser les signaux internes et les registres.
- Tester et vérifier le bon fonctionnement du circuit.
- Mettre à jour le firmware sur le terrain sans remplacer le matériel.

- Faire de la reconfiguration dynamique, en modifiant une partie du FPGA pendant que le reste continue de fonctionner.

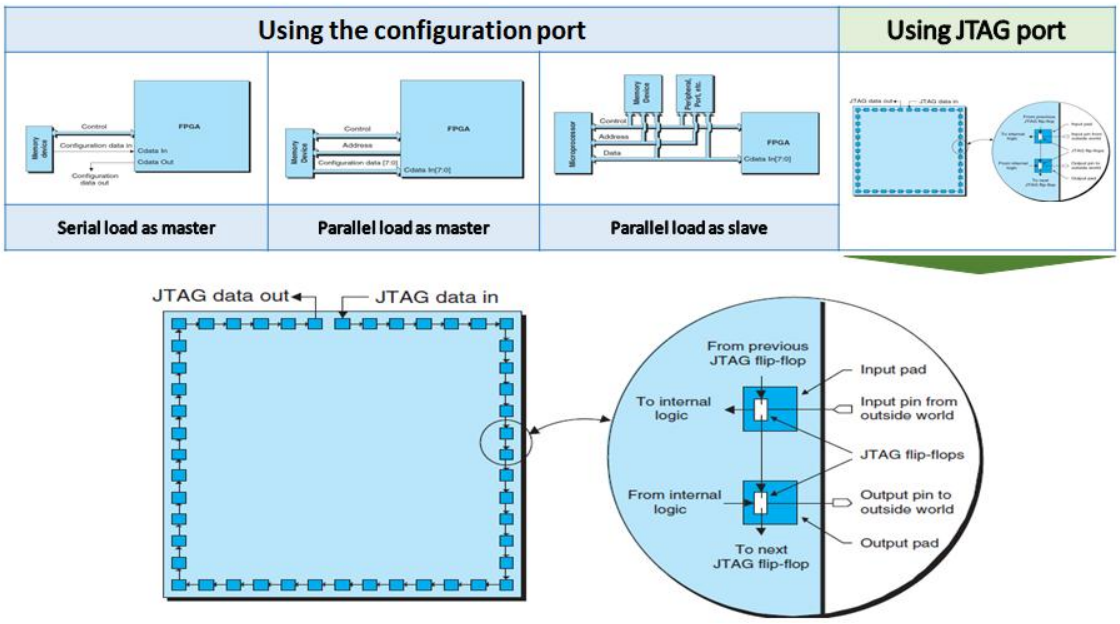


Figure 15. Utilisation du port JTAG (JTAG – *Joint Test Action Group*)

6.5.2 Modes de configuration internes

6.5.2.1 ICAP (Internal Configuration Access Port)

Ce mode permet au FPGA de se configurer lui-même à l'aide d'un processeur embarqué interne. Il est idéal pour les systèmes autonomes nécessitant une reconfiguration dynamique selon les conditions du système.

6.5.3 Modes de chargement

6.5.3.1 Chargement série – FPGA maître

Le FPGA contrôle le transfert de données bit à bit vers un autre composant. Ce mode est lent mais simple, souvent utilisé pour les mises à jour séquentielles.

6.5.3.2 Chargement parallèle – FPGA maître

Le FPGA envoie plusieurs bits simultanément sur plusieurs lignes, ce qui permet un transfert rapide de grandes quantités de données. Utilisé dans les systèmes hauts performance.

6.5.3.3 Chargement parallèle – FPGA esclave

Le FPGA reçoit les données envoyées en parallèle par un autre dispositif maître (processeur, microcontrôleur, ou un autre FPGA). Ce mode est efficace pour les applications nécessitant un traitement parallèle intensif.

6.6 Reconfiguration Partielle Dynamique (DPR)

La DPR est une technique utilisée sur certains FPGA qui permet de modifier une partie du circuit pendant qu'il fonctionne, sans arrêter le reste du système. Contrairement à une reconfiguration complète, seule une zone spécifique du FPGA est mise à jour, ce qui apporte une grande flexibilité.

Principe : Le matériel peut être reconfiguré pendant l'exécution de l'application.

Applications typiques :

- Cryptographie
- Codage vidéo
- Protocoles réseau
- Architecture à jeu d'instructions dynamique

Exemple : Dans un système de communication, une section du FPGA peut être reconfigurée pour gérer un nouveau protocole, tandis que le reste continue à fonctionner normalement. Cela optimise les ressources et améliore les performances en temps réel.

Avantage :

- Réduction de la surface et de la consommation d'énergie.
- Réutilisation du matériel.
- Plus de flexibilité et de parallélisme.
- Démarrage plus rapide (chargement progressif des modules).
- Bitstreams plus petits et portabilité améliorée.

Inconvénient :

- Temps de reconfiguration parfois long.

7. Conclusion

Les systèmes embarqués occupent aujourd'hui une place centrale dans la majorité des technologies modernes, allant des appareils mobiles aux systèmes industriels, médicaux et automobiles. Leur principal objectif est d'assurer un traitement rapide, fiable et adapté à des applications spécifiques, souvent sous de fortes contraintes de performance, de coût et de consommation énergétique.

Dans ce contexte, les FPGA représentent une solution matérielle particulièrement flexible et performante. Leur architecture reconfigurable permet d'adapter dynamiquement le matériel aux besoins de l'application, offrant ainsi un compromis idéal entre la rapidité du matériel dédié et la souplesse du logiciel. Grâce à des fonctionnalités avancées comme la reconfiguration partielle dynamique, les FPGA permettent une évolution continue des systèmes embarqués sans interruption de service.

Exercices et travaux pratiques

Exercice 01 : (QCM)

(1) Quel composant électronique est spécifiquement mentionné dans la Loi de Moore ?

- ☐ Condensateur
- ☐ Résistance
- ☐ Transistor
- ☐ Diode

(2) Quelle affirmation caractérise le mieux la Loi de Moore ?

- ☐ La puissance des processeurs double environ tous les deux ans.
- ☐ La taille des transistors double tous les deux ans.
- ☐ La vitesse des disques durs double tous les deux ans.
- ☐ La capacité des mémoires RAM double tous les deux ans.

(3) Quelle est la principale caractéristique d'un système embarqué par rapport à un ordinateur personnel traditionnel ?

- ☐ Il n'a pas de système d'exploitation.
- ☐ Il est dédié à une fonction spécifique.
- ☐ Il a toujours une connexion Internet.
- ☐ Il utilise uniquement des logiciels open source.

(4) Qu'est-ce qu'un microcontrôleur dans le contexte des systèmes embarqués ?

- ☐ Un petit ordinateur portable.
- ☐ Un composant électronique intégré qui comprend un processeur, une mémoire et des périphériques d'entrée/sortie.
- ☐ Un dispositif de stockage de données.
- ☐ Un périphérique de réseau sans fil.

(5) Quel rôle joue le système d'exploitation dans un système embarqué ?

- ☐ Il n'est pas nécessaire dans les systèmes embarqués.
- ☐ Il gère les ressources matérielles et fournit des services aux applications.
- ☐ Il est utilisé uniquement pour les mises à jour logicielles.
- ☐ Il est responsable uniquement de l'interface utilisateur.

(6) Quelle est la caractéristique essentielle d'un système embarqué temps réel ?

- Il n'a pas besoin de répondre rapidement aux événements.
- Il fonctionne uniquement en mode asynchrone.
- Il n'a pas de contraintes temporelles.
- Il doit réagir aux stimuli ou aux événements dans des délais spécifiques.

(7) Qu'est-ce qui distingue un système souple d'un système strict en termes de contraintes temporelles ?

- Le système souple a des contraintes temporelles plus strictes que le système strict.
- Le système strict peut être adapté à différentes conditions.
- Le système souple peut ignorer certaines échéances sans compromettre la fiabilité.
- Les deux systèmes n'ont aucune contrainte temporelle.

(8) Que représente le codesign dans le contexte de la conception de systèmes embarqué ?

- Un processus qui ignore les interactions entre le matériel et le logiciel.
- Une méthodologie où le matériel et le logiciel sont conçus de manière collaborative dès le début.
- Une approche où le matériel et le logiciel sont conçus de manière indépendante.
- Une méthode de conception exclusivement centrée sur le matériel.

(9) Quel est l'objectif principal du codesign ?

- Isoler le matériel et le logiciel pour simplifier la conception.
- Assurer une indépendance totale entre le matériel et le logiciel.
- Optimiser la conception en prenant en compte les interactions entre le matériel et le logiciel.
- Ignorer les aspects logiciels pour se concentrer uniquement sur le matériel.

(10) Quelle est la principale caractéristique d'un FPGA qui le distingue des ASIC ?

- Les FPGA sont moins chers.
- Les FPGA sont réutilisables pour différentes applications.
- Les FPGA sont toujours plus rapides.
- Les FPGA ne peuvent pas être programmés.

(11) Quel type de mémoire est utilisé pour stocker le BIOS d'un ordinateur, qui doit conserver des données même en cas de coupure d'alimentation ?

- SRAM
- DRAM
- PROM
- EPROM

(12) Quelle mémoire est souvent utilisée comme cache dans les processeurs en raison de sa rapidité d'accès ?

- SRAM
- DRAM
- EPROM
- EEPROM

(13) Quelle affirmation est vraie concernant la mémoire Flash ?

- Elle appartient uniquement à la catégorie de la RAM.
- Elle est uniquement utilisée pour le stockage à long terme.
- Elle est non volatile et peut être réécrite.
- Elle n'est pas utilisée dans les dispositifs électroniques modernes.

(14) Que représente un actionneur dans un contexte de système embarqué ?

- Un dispositif qui mesure une grandeur physique.
- Un composant qui stocke des données permanentes.
- Un composant qui effectue une action en réponse à un signal de commande.
- Un dispositif qui génère des signaux électriques à partir de données numériques.

(15) Pourquoi est-il important d'avoir un Watchdog dans un système embarqué ?

- Pour empêcher toute utilisation non autorisée.
- Pour accélérer le traitement des données.
- Pour surveiller l'état du système et le redémarrer en cas de défaillance.
- Pour effectuer des sauvegardes automatiques.

(16) Quel est l'avantage principal d'utiliser un FPGA dans la conception de systèmes électroniques ?

- Flexibilité et capacité de réconfiguration.
- Performances toujours plus élevées que les ASIC.
- Absence de consommation d'énergie.
- Réduction des coûts.

(17) Qu'est-ce qu'une LUT (Look-Up Table) dans un FPGA ?

- Une unité de traitement spécialisée.
- Un composant mémoire pour le stockage de données.
- Un élément logique programmable pour la mise en œuvre de fonctions logiques.
- Un module de communication.

(18) Quel est le rôle principal d'une BRAM dans un FPGA ?

- Stocker des données temporaires pendant l'exécution.
- Mettre en œuvre des fonctions logiques.
- Faciliter la communication entre les CLBs.
- Gérer les opérations en virgule flottante.

(19) Que fait l'étape de placement & routage dans la conception d'un FPGA ?

- Elle sélectionne les composants électroniques nécessaires.
- Elle détermine l'emplacement physique des éléments logiques sur le FPGA et établit les connexions entre eux.
- Elle convertit le code source en un fichier exécutable.
- Elle génère un fichier Bitstream à partir de la description du circuit.

(20) C’est quoi le Bitstream dans le contexte des FPGA ?

- Un fichier binaire qui représente la configuration du FPGA.
- Une représentation textuelle du code VHDL.
- Un fichier contenant des données temporaires pendant l'exécution.
- Une mesure de la complexité du circuit.

(21) Comment MicroBlaze est-il utilisé dans un FPGA ?

- Il est chargé dans la mémoire RAM.
- Il est converti en code VHDL.
- Il est implémenté en tant que processeur softcore.
- Il est utilisé exclusivement pour les applications graphiques.

(22) Que signifie l'acronyme IP dans le contexte des systèmes embarqués ?

- ☐ Internet Protocol
- ☐ Intellectual Properties
- ☐ Instruction Pointer
- ☐ Input Power

(23) Comment les IP peuvent-elles simplifier le processus de conception dans un FPGA ?

- ☐ En introduisant des complications supplémentaires.
- ☐ En fournissant des blocs de conception préconçus qui peuvent être intégrés.
- ☐ En remplaçant complètement le processeur MicroBlaze.
- ☐ En restreignant les options de conception.

(24) Comment les bus FSL et PLB diffèrent-ils dans leur conception et utilisation typiques ?

- ☐ FSL et PLB ont la même conception et fonction.
- ☐ PLB est utilisé exclusivement pour la communication sans fil, tandis que FSL gère la communication externe.
- ☐ Ils sont interchangeables et ont des utilisations similaires.
- ☐ FSL est destiné à une communication rapide entre processeurs, tandis que PLB facilite la communication avec des périphériques locaux.

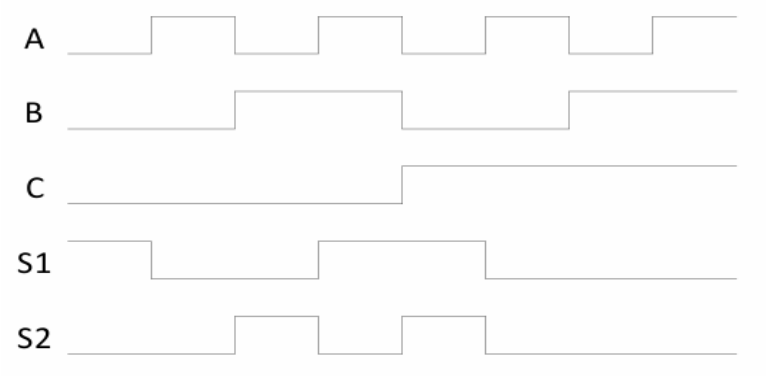
Exercice 02 : Développement VHDL

- Créez des modèles VHDL pour implémenter les fonctions suivantes en utilisant l'affectation concurrente de signaux.
  - $F(A,B) = \bar{A}B + A + A\bar{B}$
  - $F(A,B,C,D) = (\bar{A} + B) \cdot (\bar{B} + C + \bar{D}) \cdot (\bar{A} + D)$

- Développez des modèles VHDL pour implémenter les mêmes fonctions en utilisant l'affectation conditionnelle et l'affectation sélectionnée (`select`).

Exercice 03 : VHDL comportemental

- Donner un modèle VHDL comportemental pour un circuit logique de trois entrées (A, B et C) et deux sorties (S1 et S2) qui répond au chronogramme suivant :



Exercice 04 : Décompteur 4 bits

Elaborer une description VHDL d'un décompteur de 4 bits avec un signal de reset « **rst** » (remise à 0) asynchrone actif à l'état haut, un signal de chargement parallèle « **D_in** », et un signal d'activation de chargement « **load** » synchrone actif à l'état bas.

Exercice 05 : Compteur synchrone

- Développer un compteur synchrone de 0 à 15 en utilisant le langage VHDL. Les spécifications requièrent que le compteur soit déclenché par le front montant d'un signal d'horloge. De plus, le compteur doit suivre une séquence de comptage de 0 à 15, puis revenir à 0 dès qu'il atteint la valeur maximale de 15.

- Identique à la question 1, mais cette fois-ci sans utiliser l'instruction Process.

Bibliographie

[01] Maxfield, C. (2011). FPGAs: Instant Access. Oxford: Newnes.

[02] Trimberger, S. M. (2015). Field-Programmable Gate Array Technology. Springer.

[03] Kuon, I., & Rose, J. (2008). Measuring the Gap Between FPGAs and ASICs. Springer.

[04] Anderson, J. H., & Brown, S. D. (2012). FPGA Architectures: An Overview. Morgan & Claypool Publishers.

[05] Wolf, W. (2012). Computers as Components: Principles of Embedded Computing System Design (3rd ed.). Morgan Kaufmann.

[06] Soares, L. F. G. and Araújo, G. (2020). Embedded System Design on a Shoestring: Achieving High Performance with Limited Resources. Springer.

[07] Azzouzi, O. (2015). Système embarqué flexible pour un chiffrement hybride symétrique/asymétrique, Mémoire de Magister, École Nationale Supérieure d'Informatique (ESI), Alger.