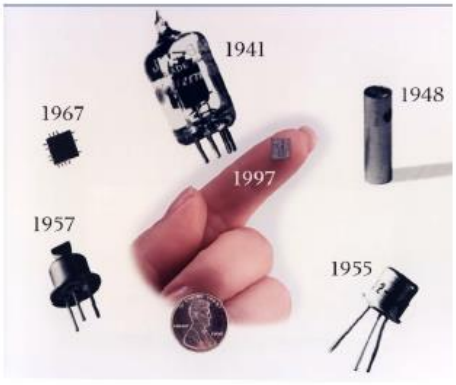
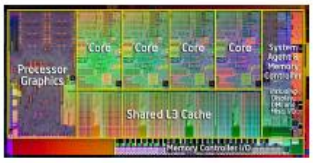
INTRODUCTION

1. Introduction

- Evolution de l'électronique depuis 1948

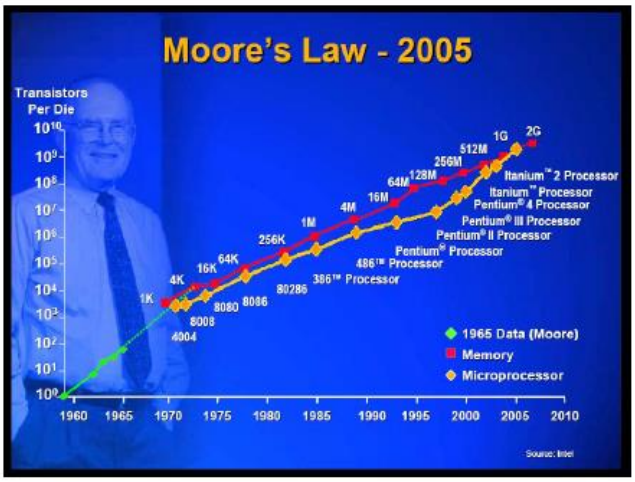




2011 : Intel Core I7 2600K, 32nm :
Die : 216mm^2
1,16 x 10^9 transistors !

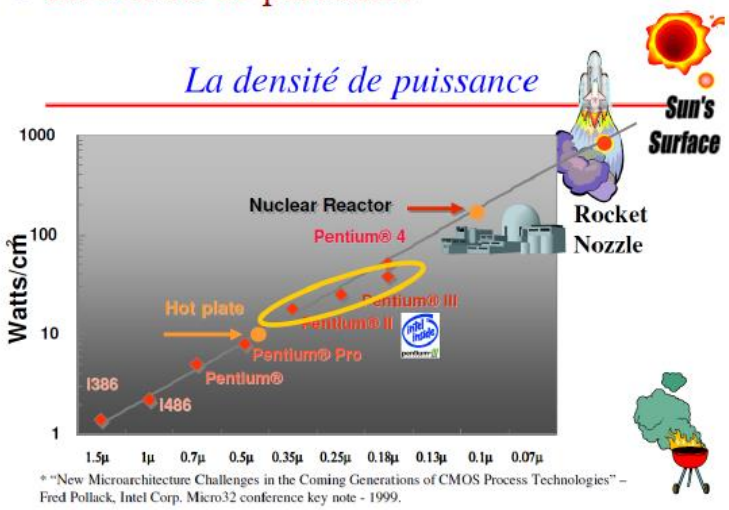
1. Introduction

- La loi de Moore



1. Introduction

- La densité de puissance



* "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies" – Fred Pollack, Intel Corp. Micro32 conference key note - 1999.

1. Introduction

➢ Les systèmes numériques aujourd'hui :




Sony Playstation 2



Philips TiVo Recorder

1. Introduction

➢ La conception des systèmes numériques :

Spécification de l'application

Langage de modélisation
- UML, XML, esterel

Langage de programmation
- Matlab, C, C++, JAVA, assembleur

Langage de description matériel
- Système C, VHDL, Verilog

-Méthodes de conception

- Outils de conception

Réalisation de l'application

Cible logicielle
- µP, µC, DSP

Cible matérielle spécifique
- ASIC

Cible matérielle non-spécifique
-FPGA, CPLD

Cible mixte
- SOC, RSOC

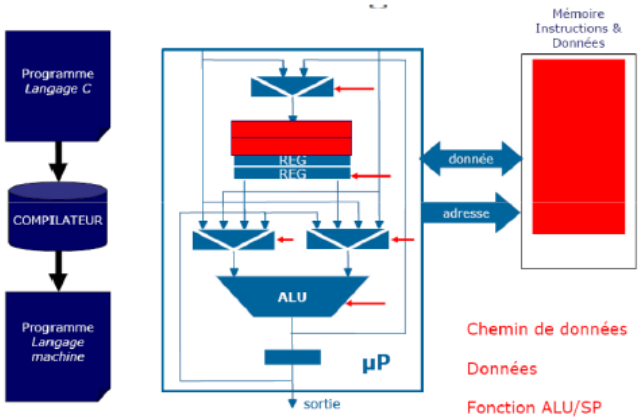
Contraintes de conception

Performances du système
- Vitesse de fonctionnement
- Consommation de puissance
- Performances de calculs
- surface de Silicium
- Prix
- Sécurité
- ...

Matériel
- Outils logiciels
- Performances PC/Station

1. Introduction

- Les cibles logicielles et matérielles :
  - Les cibles logicielles (=µP, µC, DSP)



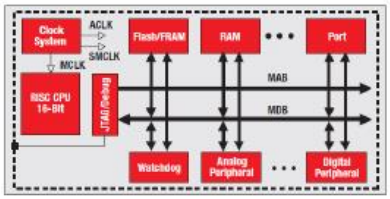
1. Introduction

- Les µP :

- **Performances**
  - Temps par tâche = I x C x T
  - I : nombre d'instructions par tâche
  - C : nombre de cycles machine par instructions
  - T : temps d'un cycle machine (dépend de la technologie et de l'efficacité de l'ALU)
- **3 types de processeurs**
  - CISC (Complex Instruction Set Computer) : I faible, C grand
  - RISC (Reduce Instruction Set Computer) : I élevé, C faible
  - VLIW (Very Large Instruction Word) : I réduit car macro-instruction RISC, C faible
- **Avantages**
  - Polyvalents
  - Bon rapport performances/prix
  - Facile à programmer : langage de programmation de haut niveau (ex :C/C++)
  - Performant pour les algorithmes complexes (contrôle)
- **Inconvénients**
  - Difficilement temps réel sous contraintes dures
  - Performances limitées pour les algorithmes de traitements de données
  - Dépendance technologique

1. Introduction

✦Les µC :



1. Introduction

✦Les µC :

- Caractéristiques
  - Architecture simple, jeux d'instructions réduit
  - Basé sur des architectures de processeurs connus
  - Exemple : 68HC11, 8051, PIC, AVR, ARM
- Avantages
  - Très économique : pas besoin d'acheter des périphériques
  - Spécialisé
  - Simple d'utilisation
- Inconvénients
  - Peu performant
  - Spécialisé: ne convient pas à tout les domaines d'application

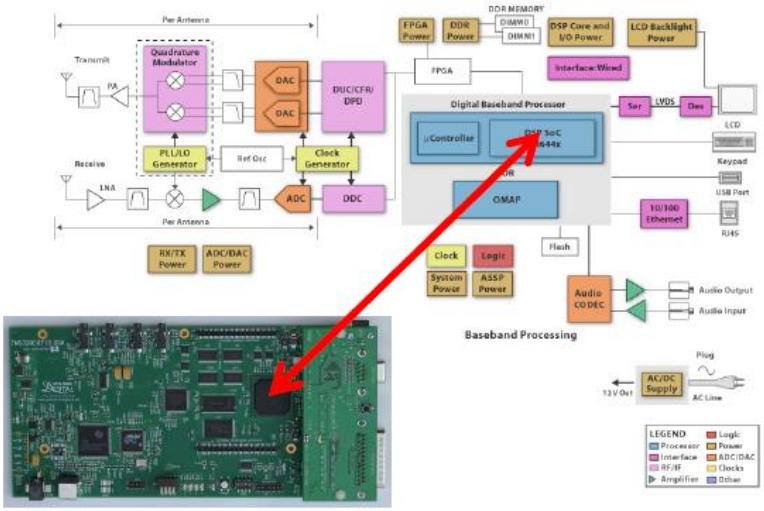
1. Introduction

✦Les Digital Signal Processors (DSP) :

- Caractéristiques
  - Architecture RISC complexe, super scalaire (plusieurs unités de traitements), pipeline
  - Architecture Harvard et Super Harvard (nombreux bancs mémoire)
  - Instructions complexes mais jeux d'instructions réduit
  - Exemple : Texas Instrument C6x
- Avantages
  - Très économique : pas besoin d'acheter des périphériques
  - Spécialisés traitement du signal
  - Peuvent mélanger calcul flottant et virgule fixe
- Inconvénients
  - Circuit spécifique
  - Consommation d'énergie élevée

1. Introduction

✦Les Digital Signal Processors (DSP) :



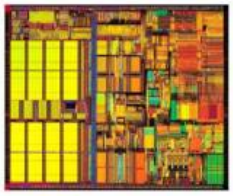
1. Introduction

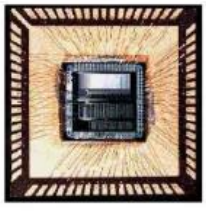
- En résumé sur les cibles logicielles :
  - Avantages :
    - Flexibilité: il suffit de modifier le programme pour modifier l'application
    - Simple à mettre en œuvre grâce à la programmation de haut niveau (langage C) (possibilité de grande abstraction par rapport au matériel)
    - Temps de conception courts et coûts de conception faibles
    - Prix de revient faible
  - Inconvénients :
    - Faibles performances (consommation de puissance, vitesse de fonctionnement, puissance de calcul, etc,) à cause d'une architecture séquentielle (une opération à la fois, ou quelques unes dans le cas superscalaire) et de trop nombreux accès à la mémoire (instructions + données)

1. Introduction

- Les cibles matérielles spécialisées (ASIC) :

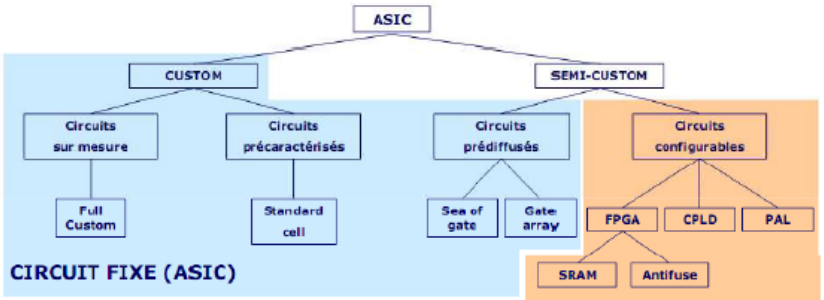
- ASIC : Application Specific Integrated Circuit
  - Numérique, analogique ou mixte (télécommunication)
  - Spécialisé pour une application
  - Réalisation complexe (de la spécification haut niveau à la synthèse physique)
  - Extrêmement performant : dédié + réalisation parallèle + technologie de pointe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





1. Introduction

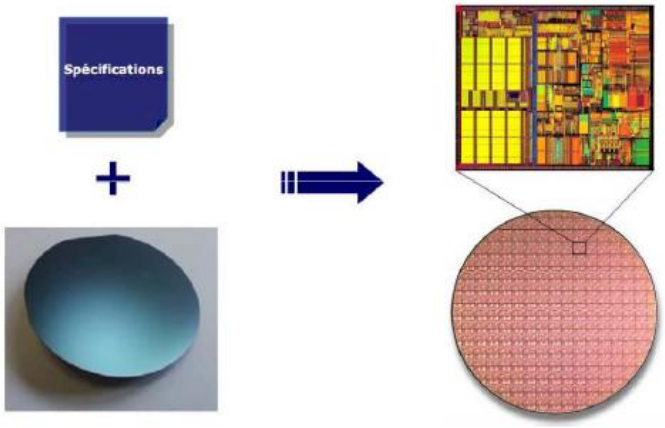
- Les différentes cibles matérielles :



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

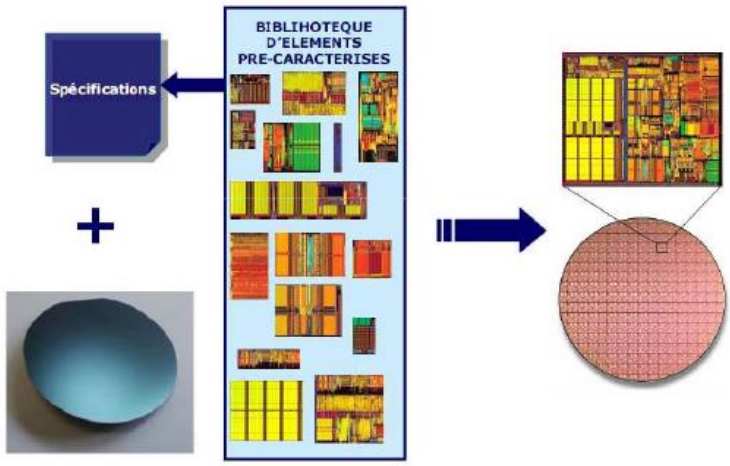
1. Introduction

- ASIC full custom :



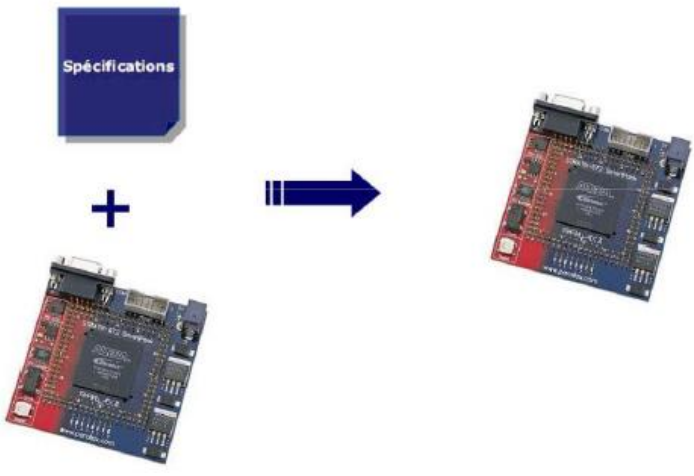
1. Introduction

- ASIC standard cell :



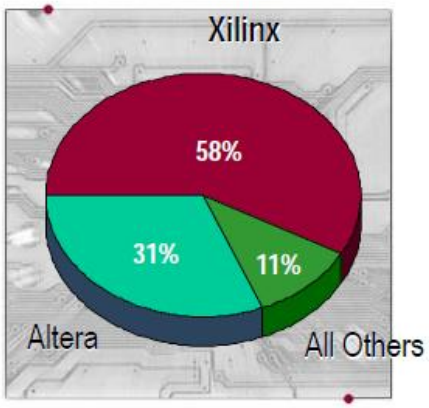
1. Introduction

- Circuit configurable (ici FPGA) :



1. Introduction

- Le marché des FPGA :



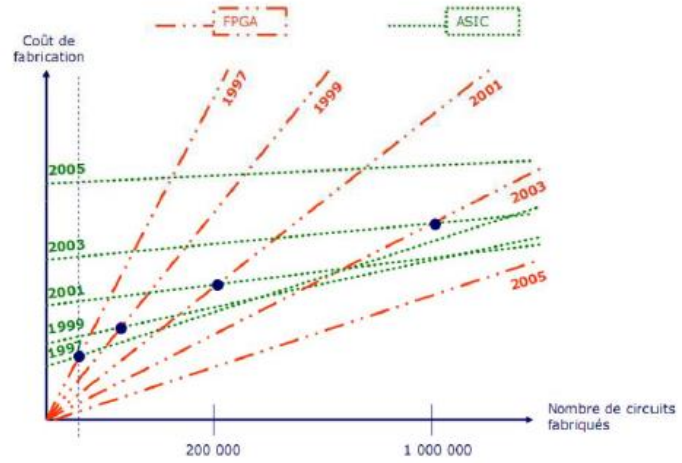
1. Introduction

- ASIC vs FPGA:

Caractéristiques	Circuits Configurables		ASIC	
	FPGA SRAM	CPLD	Standard Cell	Full Custom
Densité (portes/m²)	1500 à 6500 0,13µm 90 nm	Faible 0,18 µm	Grande 90 nm	Grande 90 nm
Performance (Hrz, Bits/s)	500 MHz 10 Gbits/s	200 MHz	qq Ghz 12 Gbits/s	qq Ghz 20 Gbits/s
Consommation (Watt)	Grande	Grande	Faible	Faible
Flexibilité	Grande	Grande	Aucune	Aucune
Temps de conception	Court	Court	Long	Très Long
Utilisation des outils	Facile	Facile	Complexe	Complexe
Cout de conception	Faible	Faible	Elevé	Très élevé
Volume de production	Faible et moyen	Faible et moyen	> 1 000 000 de pièces	> 1 000 000 de pièces

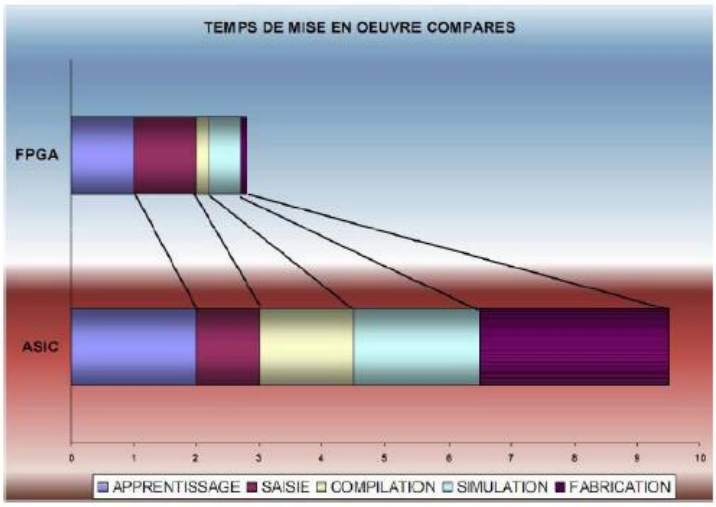
1. Introduction

- De 1997 à 2005 : évolution des coûts



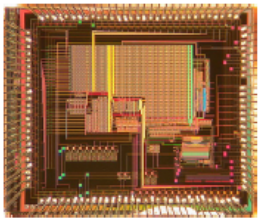
1. Introduction

- Temps de conception comparés :



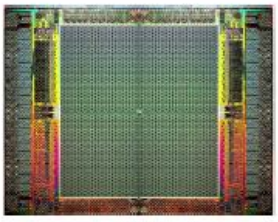
1. Introduction

- Conclusion ASIC :
  - Avantages :
    - Haute intégration,
    - Hautes performances (vitesse, consommation),
    - Coûts unitaires faibles en production de masse
    - Personnalisation
    - Sécurité industrielle
  - Inconvénients :
    - Prix du 1er exemplaire,
    - Pas d'erreur possible
    - Non-flexible
    - High time to market
    - Fabrication réservée aux spécialistes (fondeurs),



1. Introduction

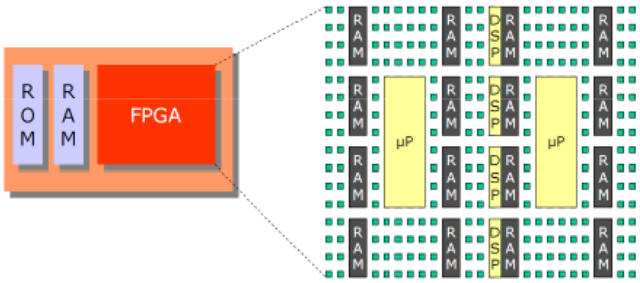
- Conclusion FPGA :



  - Avantages :
    - Possibilité de prototypage,
    - Low time to market
    - Adaptabilité aux évolutions futures (reconfiguration)
    - Flexibilité
  - Inconvénients :
    - Intégration limitée,
    - Moins performant qu'un ASIC
    - Prix unitaire élevé en production de masse

1. Introduction

- Mais les méthodes de conception évoluent car :
  - Toujours plus d'intégration (SoC)
  - Les FPGA sont de plus en plus performants et de moins en moins chers,
  - Les FPGA remplacent peu à peu les ASIC…



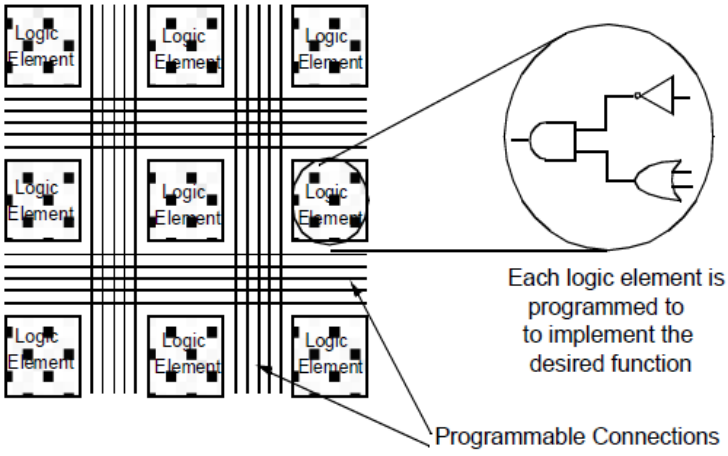
2. Technologie des FPGA

What is an FPGA ?

- **Field Programmable** Gate Array
- Gate Array
  - Two-dimensional array of logic gates
  - Traditionally connected with customized metal
  - Every logic circuit (customer) needs a custom-manufactured chip
- Field Programmable
  - Customized by programming after manufacture
  - One FPGA can serve every customer
- FPGA: re-programmable hardware

2. Technologie des FPGA

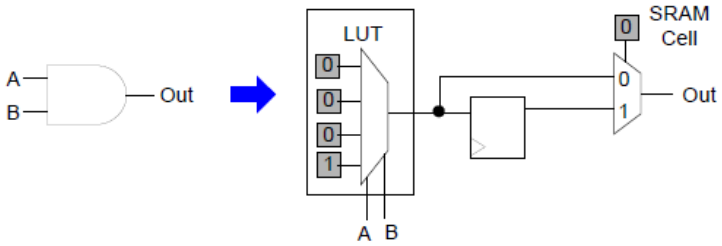
Basic Internals of an FPGA



2. Technologie des FPGA

FPGA Logic Element

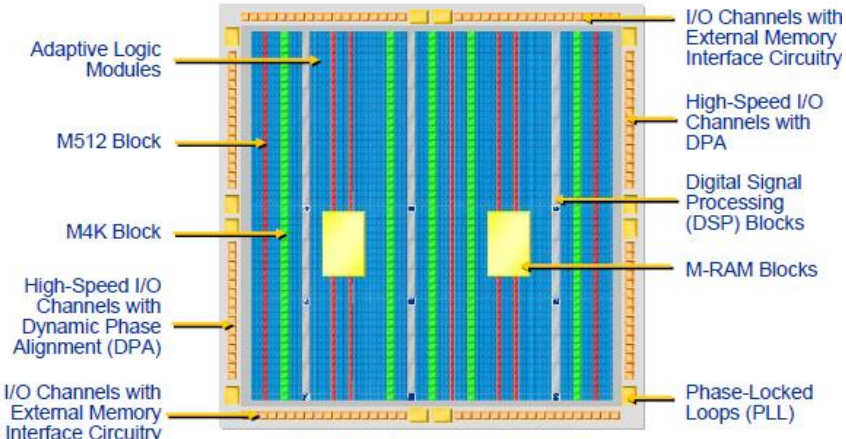
- Look-Up Table (LUT) + register + extra …



- FPGAs typically use 4-input or larger LUTs
  - Cyclone family (low cost): 4-inputs
  - Stratix II: Adaptive Logic Module implements 4 – 6 input LUTs efficiently
  - Virtex 5: 6 inputs

2. Technologie des FPGA

Modern, mid-size FPGA – 2S60



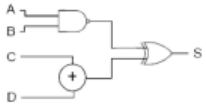
2. Technologie des FPGA

- 400MHZ max mais traitement parallèle !
  - Exemple soit à réaliser : $\overline{(A \bullet B)} \oplus (C + D)$
    - Réalisation logicielle à 400MHz : 7 cycles machine = 17.5 ns

```
MOV   AL,A
AND   AL, B
NEG   AL
MOV   AH, C
ADD   AH, D
XOR   AL, AH
MOV   S, AL
```

Les cibles logicielles fonctionnent plus vite (fréquence plus grande) mais calculent plus lentement !!!
=> Pas de temps réel

    - Réalisation matérielle : temps de traversée des portes = 2 ns



Les cibles matérielles fonctionnent moins vite (fréquence plus petite) mais les données sont traitées plus vite !!!
=> Temps réel

3. Les HDL

Définition :

- Un langage de description de matériel (Hardware Description Language, HDL) est une instance d'une classe de langage informatique ayant pour but la description formelle d'un système électronique. Il peut généralement :
  - décrire le fonctionnement du circuit,
  - décrire sa structure,
  - et servir à vérifier sa fonctionnalité par simulation ou preuve formelle.
- Un HDL est une représentation textuelle d'un comportement temporel ou d'une structure d'un circuit. En comparaison avec un langage de programmation classique, la syntaxe et la sémantique des HDL inclut des notations pour exprimer ***la concurrence et le temps***, qui sont les principaux attributs du matériel. Les classes de langages dont la seule caractéristique est de décrire un circuit par une hiérarchie de blocs interconnectés est appelée une ***netlist***.

3. Les HDL

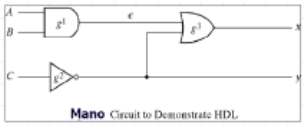
But :

- la simulation :
  - L'un des objectifs des HDL est d'aboutir à une représentation exécutable d'un circuit, soit sous forme autonome, soit à l'aide d'un programme externe appelé ***simulateur***. Cette forme exécutable comporte une description du circuit à simuler, un ***générateur de stimuli*** (vecteurs de test), ainsi que le dispositif implémentant la sémantique du langage et l'écoulement du temps.
    Il existe deux types de simulateurs, à temps discret, généralement pour le numérique, et à temps continu pour l'analogique. Des HDL existent pour ces deux types de simulations.
- La synthèse :
  - En n'utilisant qu'un sous-ensemble d'un HDL, un programme spécial appelé ***synthétiseur*** peut transformer une description de circuit en une ***netlist*** de portes logiques ayant le même comportement que le circuit de départ. Le sous-ensemble du langage utilisé à ce propos est alors dit synthétisable. La sémantique synthétisable ignore typiquement toutes les constructions ayant un rapport avec le temps.

3. Les HDL

- Il existe un grand nombre de HDL :
  - VHDL (Europe)
  - Verilog (USA)
  - SystemC
  - SystemVerilog
  - …

```
// FILENAME: circuit_simple
module Circuit_Simple(A,B,C,x,y);
    input A, B, C;
    output x, y;
    wire e;
    and g1(e, A, B);
    not g2(y, C);
    or g3(x, e, y);
endmodule
```



Mano Circuit to Demonstrate HDL

```
-- FILENAME: circuit_simple
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Circuit_Simple is
    Port ( A, B, C : in std_logic;
           x, y    : out std_logic);
end Circuit_Simple;
architecture Comportement of Circuit_Simple is
begin
    e <= A and B;
    y <= not C;
    x <= e and y;
end Comportement;
```

VHDL est très VERBOSE! …
… Comparé à Verilog d'ou la simplicité de ce dernier.
VERILOG sera ainsi adopté comme le HDL de choix pour le reste des chapitres.

4. Le langage VHDL

- VHDL (VHSIC Hardware Description Langage) est un langage de description de matériel, c'est-à-dire un langage utilisé pour décrire un système numérique matériel, comme, par exemple, un flip-flop (bascule D) ou un microprocesseur
- Il peut modéliser un système par n'importe quelle vue, ***structurelle*** ou ***comportementale***, à tous les niveaux de description.
- De plus il peut servir non seulement à simuler un système mais aussi à le ***synthétiser***, c'est-à-dire être transformé par des logiciels adaptés (synthétiseurs) en une série de portes logiques prêtes à être gravées sur du silicium.
- VHDL est l'un des trois grands langages de description de matériel utilisés majoritairement dans l'industrie, avec VHDL et SystemC.
- Le langage standard IEEE VHDL a été développé par le Groupe d'Analyse et de Standardisation VHDL (VASG, pour "VHDL Analysis and Standardization Group").

4. Le langage VHDL

- Structure d'une description VHDL :
  - Une description **VHDL** est composée de 2 parties **indissociables** à savoir :
    - ***L'entité*** (***ENTITY***), elle définit les entrées et sorties.
    - ***L'architecture*** (***ARCHITECTURE***), elle contient les instructions **VHDL** permettant de réaliser le fonctionnement attendu.

4. Le langage VHDL

- Instructions concurrentes et séquentielles :
  - Comme tout langage de description de matériel, le VHDL décrit des structures par assemblage d'instructions **concurrentes** dont l'ordre d'écriture n'a aucune importance, contrairement aux instructions **séquentielles** qui sont exécutées les unes après les autres, comme c'est la cas du C.
  - VHDL offre cependant la possibilité d'utiliser des instructions séquentielles, plus naturelles pour l'homme, par le biais de processus ***process***. Les processus peuvent avoir leurs propres variables locales ***variable*** .

4. Le langage VHDL

- Le fonctionnement du processus est régi par les règles suivantes :
  - Un processus est une boucle infinie , lorsqu'il arrive à la fin du code, il reprend automatiquement au début,
  - Un processus doit être sensible des points d'arrêt de façon à le **synchroniser**. La synchronisation est donc indiquée par un point d'arrêt qui est **évènement** particulier. Il existe 2 types de points d'arrêts :
    - Le processus est associé à une "**liste de sensibilité**" qui contient une liste de signaux qui réveillent le processus lors d'un **changement** d'un des signaux. Sa syntaxe est `process(liste de signaux)`
    - Le processus a des instructions d'arrêt `wait` dans sa description interne. Le wait est sensible soit à un **signal** soit à un **temps physique,**
  - Les variables sont internes au processus et sont affectées **immédiatement**, contrairement aux signaux qui eux ne sont pas affectés directement mais par le biais de leur échéancier qui est mis à jour en fin de processus avec la nouvelle valeur et le temps d'affectation qui correspond à un delta-cycle après le signal ayant réveillé le processus.

Introduction

Systèmes sur puce vs systèmes informatiques

- Ressources limitées (coût, consommation) :
  - ✓taille mémoire faible
  - ✓petite taille de mots
  - ✓fréquence de fonctionnement (relativement) faible
- Se focaliser sur l'efficacité :
  - ✓programmation à bas niveau (asm/C)
  - ✓systèmes d'exploitation minimalistes
  - ✓architectures spécialisées
  - ✓coprocesseurs *ad-hoc*
  - ✓processeurs spécialisés : DSP, NP, GPU
  - ✓communications spécialisées

Systèmes embarqués

Définition (1)

Un système électronique embarqué est un élément constitutif d'un système plus complexe pour lequel il rend des services bien précis (contrôle, surveillance, communication…).
Il est constitué de parties matérielles et logicielles qui sont conçues spécifiquement pour réaliser une fonction dédiée.

Systèmes embarqués

Définitions (2)

Un système embarqué (SE) est un système informatisé spécialisé qui constitue une partie intégrante d'un système plus large ou une machine. Typiquement, c'est un système sur un seul processeur et dont les programmes sont stockés en ROM. A priori, tous les systèmes qui ont des interfaces digitales (i.e. montre, caméra, voiture…) peuvent être considérés comme des SE.
Certains SE ont un système d'exploitation et d'autres non car toute leur logique peut être implantée en un seul programme.

Systèmes embarqués

Définitions (3)

Un système embarqué est une combinaison de logiciel et matériel, avec des capacités fixes ou programmables, qui est spécialement conçu pour un type d'application particulier. Les distributeurs automatiques de boissons, les automobiles, les équipements médicaux, les caméras, les avions, les jouets, les téléphones portables et les PDA sont des exemples de systèmes qui abritent des SE. Les SE programmables sont dotés d'interfaces de programmation et leur programmation est une activité spécialisée.

Systèmes embarqués

Définitions (4)

Un système embarqué est une composante primordiale d'un système (i.e. un avion, une voiture…) dont l'objectif est de commander, contrôler et superviser ce système.

Systèmes embarqués

Principales caractéristiques

- ✓ Encombrement mémoire (mémoire limitée, pas de disque en général)
- ✓ Consommation d'énergie (batterie : point faible des SE)
- ✓ Poids et volume
- ✓ Autonomie
- ✓ Mobilité
- ✓ Communication (attention : la communication affecte la batterie)
- ✓ Contraintes de temps réel
- ✓ Contraintes de sécurité
- ✓ Coût de produits en relation avec le secteur cible

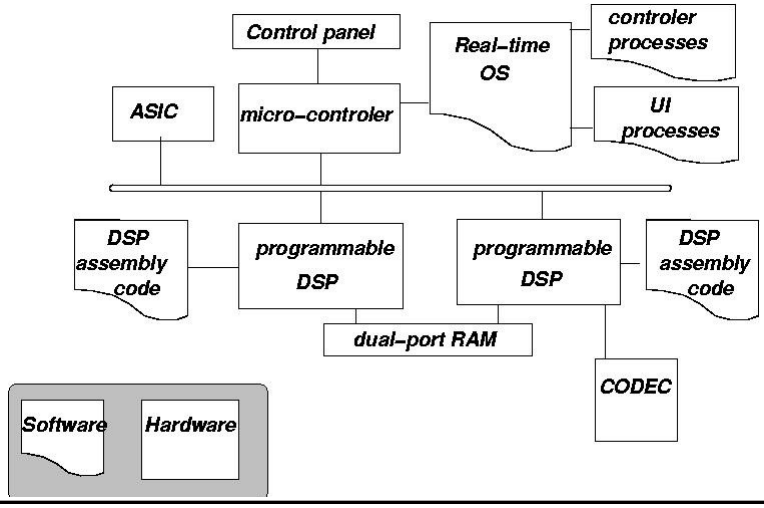
Systèmes embarqués

Domaines d'application

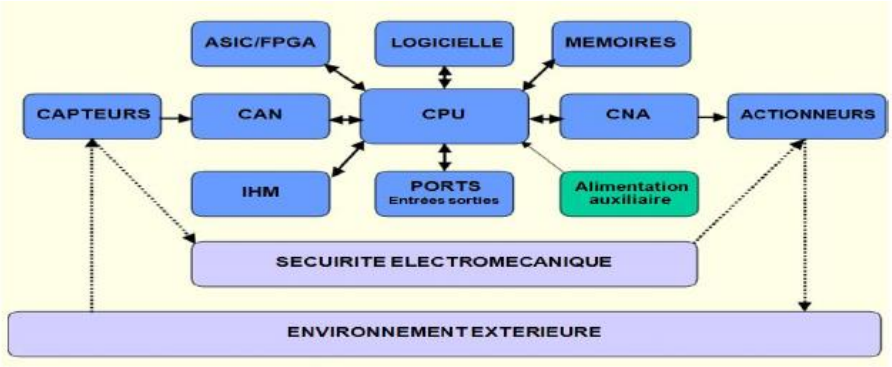
- **Applications utilisateur**
  - ✓ Jeux, bureautique, services, multimédia
  - ✓ Téléphones, PDA, lecteurs MP3…
- **Contrôle**
  - ✓ Contrôle-commande (automobile, usines, robots…)
- **Traitement du signal**
  - ✓ Radar, sonar, avionique
  - ✓ Traitement de srosses quantités de données
- **Network computing**
  - ✓ Téléphonie, routeur…

Systèmes embarqués

Architecture représentative



Système embarqué typique



Systèmes embarqués

L'art de bien concevoir un système embarqué(1)

Du point de vue technique, la conception d'un système embarqué demande à son concepteur d'être pluridisciplinaire : électronique, informatique, réseaux, sécurité ...

Mais le concepteur se doit aussi d'être un bon gestionnaire car concevoir un système embarqué revient finalement à un exercice d'optimisation : minimiser les coûts de production pour des fonctionnalités optimales.

Systèmes embarqués

L'art de bien concevoir un système embarqué (2)

Le système embarqué se doit d'être :

- Robuste.
- Simple.
- Fiable.
- Fonctionnel. Le système doit toujours fonctionner correctement.
- Sûr, surtout si la sécurité des personnes est en jeu.
- Tolérant aux fautes.

Systèmes embarqués

L'art de bien concevoir un système embarqué (3)

D'autres contraintes sont aussi à prendre en compte :

- L'encombrement.
- Le poids.
- Le packaging : difficulté de faire cohabiter dans un faible volume, électronique analogique, électronique numérique et RF sans interférences.
- L'environnement extérieur.
- La consommation électrique. Le système embarqué nomade doit être de faible consommation car il est alimenté par des batteries. Une consommation excessive augmente le prix de revient du système embarqué car il faut alors des batteries de plus forte capacité.

Systèmes embarqués

L'art de bien concevoir un système embarqué (5)

- Le coût. Beaucoup de systèmes embarqués sont fabriqués en grande série et doivent avoir des prix de revient extrêmement faibles.
- Le temps de développement. Dans un marché concurrentiel, il convient d'avoir un système opérationnel le plus rapidement possible pour être le premier sur le marché.

Systèmes embarqués

L'art de bien concevoir un système embarqué (5)

Devant toutes ces contraintes, le concepteur adopte des règles de bon sens :

- ✓ Faire simple.
- ✓ Utiliser ce que l'on a déjà fait ou fait par d'autres.
- ✓ Ne pas se jeter sur les technologies dernier cri. Quelle est leur pérennité dans le temps ?
- ✓ Ne pas se jeter sur le dernier composant sorti surtout s'il est grand public. Quelle est sa pérennité dans le temps surtout s'il l'on travaille pour la défense où l'on demande une maintenance sur 30 ans !

Systèmes embarqués

L'art de bien concevoir un système embarqué (5)

Utiliser des technologies éprouvées qui ont fait leur preuve. Ces technologies peuvent d'ailleurs avoir plusieurs générations de retard par rapport à leurs homologues grand public.

Pour le grand public, le concepteur de systèmes embarqués peut sembler faire de l'inertie face aux nouvelles technologies mais il faut le comprendre : c'est pour le bien du système qu'il conçoit surtout si la sécurité des personnes est en jeu...

Cela explique en partie le décollage difficile des logiciels libres et de Linux pour l'embarqué.

Systèmes embarqués

Les logiciels et les systèmes embarqués

Différence avec l'informatique générale

- Dans les systèmes embarqués, les tâches concurrentes sont définies statiquement
- Un système est conçu pour un nombre limité de fonctionnalités
- Elles ne changent pas avec le temps
- Cela autorise de dimensionner au mieux et d'optimiser le système en fonction de ses besoins

Systèmes embarqués

Les logiciels et les systèmes embarqués

Différence avec l'informatique générale

Informatique :

• Processeur standard
– Multiples unités fonctionnelles (flottant)
– Vitesse élevée (> GHz)
– Consommation électrique élevée
– Chaleur
– Taille
• MMU (mémoire virtuelle)
• OS
• Cache
• Grand nombre de périphériques

Embarqué :

• Processeur dédié (contrôleur)
– Architecture adaptée
– Vitesse faible (~200 MHz)
– 8-32bits : mémoire limitée
– Basse consommation
– Petite taille, grand volume => faible coût
• Processeur DSP (traitements)
– Très puissants
• Qqs Mo de mémoire
• RTOS

Systèmes embarqués

Les logiciels libres et les systèmes embarqués

Les logiciels libres et en particulier GNU/Linux sont de plus en plus employés dans l'embarqué depuis qu'ils ont commencé à faire leur preuve il y a quelques années.

Pourquoi retrouve-t-on Linux dans l'embarqué ? Tout d'abord pour ses qualités qu'on lui reconnaît maintenant dans l'environnement grand public :

Systèmes embarqués

Les logiciels libres et les systèmes embarqués (1)

- C'est un logiciel libre disponible gratuitement au niveau source.
- Il est stable et efficace.
- Il n'y a pas de royalties à reverser sur chaque produit le mettant en oeuvre.
- C'est un système d'exploitation ouvert.
- Différentes distributions sont disponibles pour coller au mieux à un type d'application.
- Il existe une aide rapide en cas de problèmes par la communauté Internet des développeurs Linux.
- Il y a un nombre de plus en plus important de logiciels libres disponibles.

Systèmes embarqués

Les logiciels libres et les systèmes embarqués (2)

• La connectivité IP chère aux systèmes embarqués est en standard.
Linux possède d'autres atouts très importants pour l'embarqué :
• Il existe un portage pour processeurs autres que la famille x86 : PowerPC, ARM, MIPS, 68K, ColdFire…
• La taille du noyau est modeste et compatible avec les tailles de mémoires utilisées dans un système embarqué (800 Ko pour µClinux sur processeur ColdFire).
• Différentes distributions spécialisées existent pour coller à un type d'application : routeur IP, PDA, téléphone…

Systèmes embarqués

Les logiciels libres et les systèmes embarqués (3)

• Le chargement dynamique de modules (drivers) est autorisé, ce qui permet d'optimiser la taille du noyau.
• La migration pour un spécialiste Linux à Linux embarqué est rapide et en douceur, ce qui réduit les temps de formation et les coûts…
On retrouve généralement aussi un certain nombre de suppressions de fonctionnalités dans les adaptations de Linux pour l'embarqué.
Il n'y a pas de gestion de la MMU (Memory Management Unit) pour ne pas pénaliser les performances globales du système. C'est le cas de µClinux, le noyau Linux adapté aux microcontrôleur sans MMU.

Systèmes embarqués

Conclusion (1)

Nous voilà donc au terme de cette introduction sur les systèmes embarqués.

- Les points les plus importants vous ont été présentés afin d'aborder avec sérénité et sans à priori les articles suivants.
- Quel avenir aux systèmes embarqués ? Un avenir radieux.
- Quelles sont les évolutions techniques à venir ? Deux pistes semblent se dessiner :

Systèmes embarqués

Conclusion (2)

✓ Un couplage fort entre matériel et logiciel via le développement conjoint matériel/logiciel ou codesign et l'approche système sur silicium SoC (System on Chip).

✓ L'explosion de l'électronique/informatique ambiante (ubiquitous computing) bon marché couplée à l'Internet ambiant. Les réseaux de capteurs (sans fil) seront omniprésents notamment en domotique.