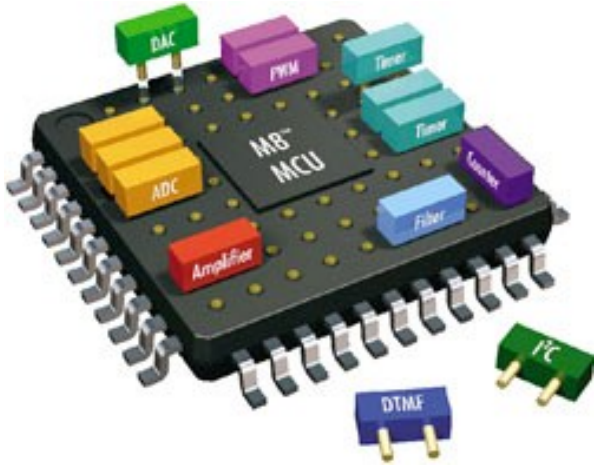
# Outils de développement et méthodes de conception des systèmes embarqués

**Informatique embarquée**

**1ère année master RSD**

**Dr: K.BARKA**



# 1. Introduction

- **La complexité des systèmes embarqués accentuent le besoin de disposer de méthodes et d'outils d'aide à la conception.**

- **Ces derniers permettant de concevoir ces systèmes de façon automatisée et surtout en réduisant leur temps de conception.**

- **Différents outils et langages sont disponibles pour la conception et le développement des systèmes embarqués.**

- **Au niveau matériel des langages de description matériel (HDL) peuvent être utilisés tels que VHDL ou SystemC.**

- **Des langages de programmation peuvent être adaptés au SE pour le niveau logiciel, ainsi que des langages de modélisation tel que UML.**

- **La conception des SE implique ainsi une modélisation de systèmes hétérogènes constitués de logiciel et de matériel.**

# 2. Outils de développement du matériel (1)

▪ **L'objectif de ces outils est l'aide à la conception et la validation des circuits électroniques..**

**a. VHSIC HDL (Very high speed integrated circuits Hardware Description Language)**

✓ **VHSIC ou VHDL est un langage de description matériel destiné à représenter le comportement ainsi que l'architecture d'un système électronique numérique.**

✓ **Une spécification décrite en VHDL peut être vérifiée par simulation, avant que la conception détaillée ne soit terminée.**

✓ **En outre, les outils de conception assistée par ordinateur permettant de passer directement d'une description fonctionnelle en VHDL à un schéma en porte logique ont révolutionné les méthodes de conception des circuits numériques.**

# 2. Outils de développement du matériel (2)

✓ **Le langage VHDL a été commandé par le Département de la Défense des États-Unis.**

✓ **Le VHDL reprend la même syntaxe que celle utilisée par le langage Ada.**

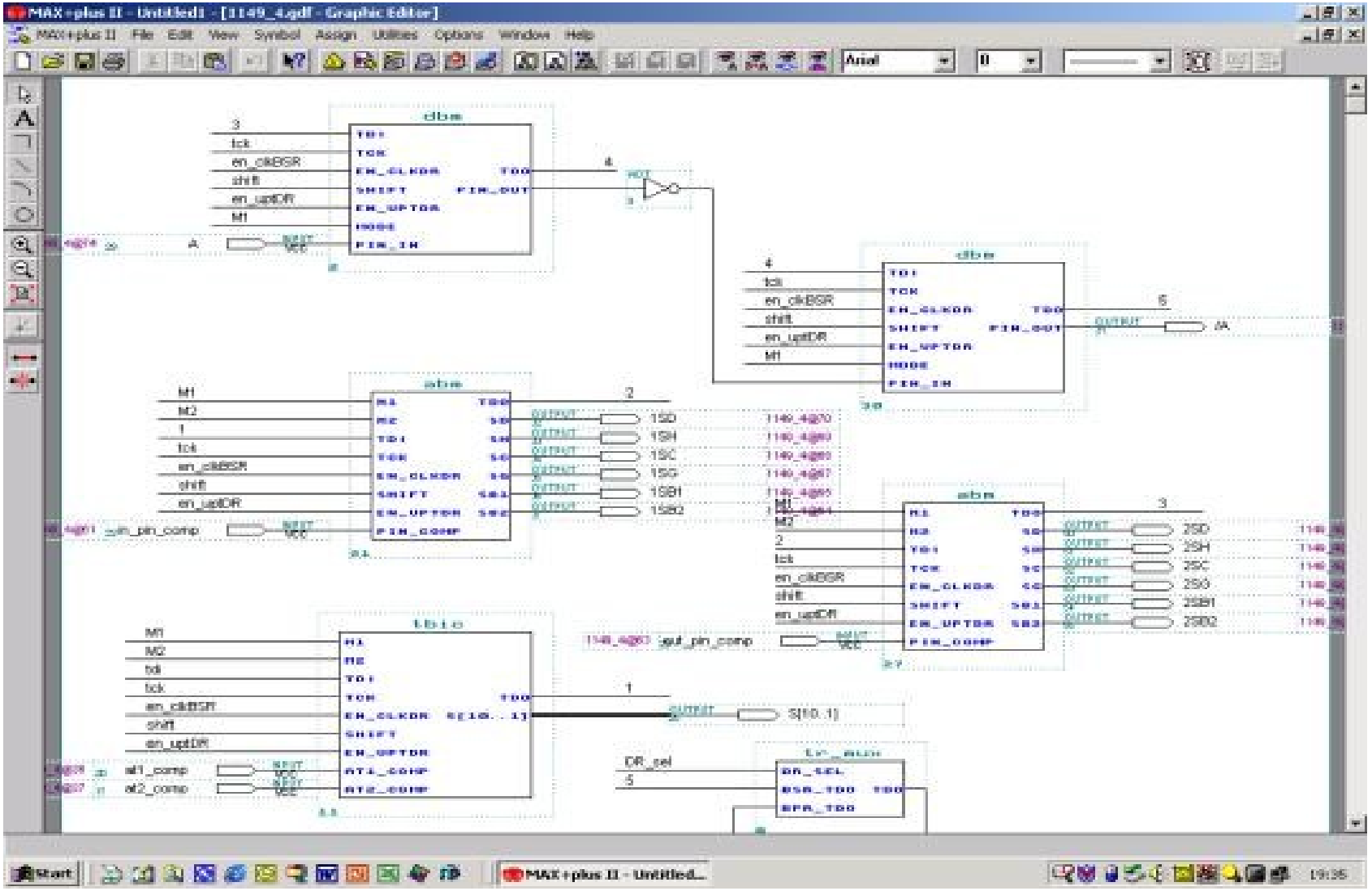
✓ **Le langage VHDL est maintenant le langage de description matérielle majoritairement utilisé par les entreprises européennes alors que Verilog est souvent préféré de l'autre côté de l'Atlantique.**

✓ **Le but d'un langage de description matériel tel que le VHDL est de faciliter le développement d'un circuit numérique en fournissant une méthode rigoureuse de description du fonctionnement et de l'architecture du circuit désirée.**

✓ **L'idée est de ne pas avoir à réaliser (fondre) un composant réel, en utilisant à la place des outils de développement permettant de vérifier le fonctionnement attendu.**

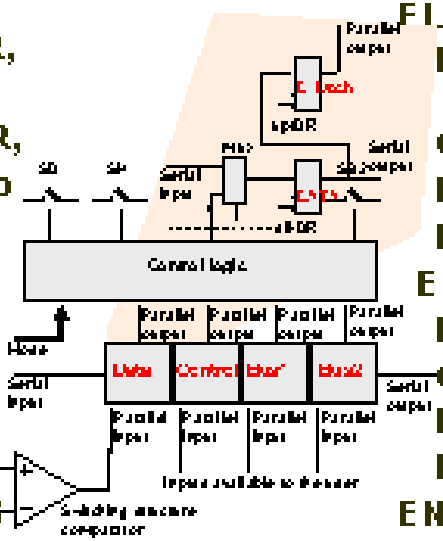
✓ **Ce langage permet en effet d'utiliser des simulateurs, dont le rôle est de tester le fonctionnement décrit par le concepteur.**

# 2. Outils de développement du matériel (2)



# 2. Outils de développement du matériel (2)

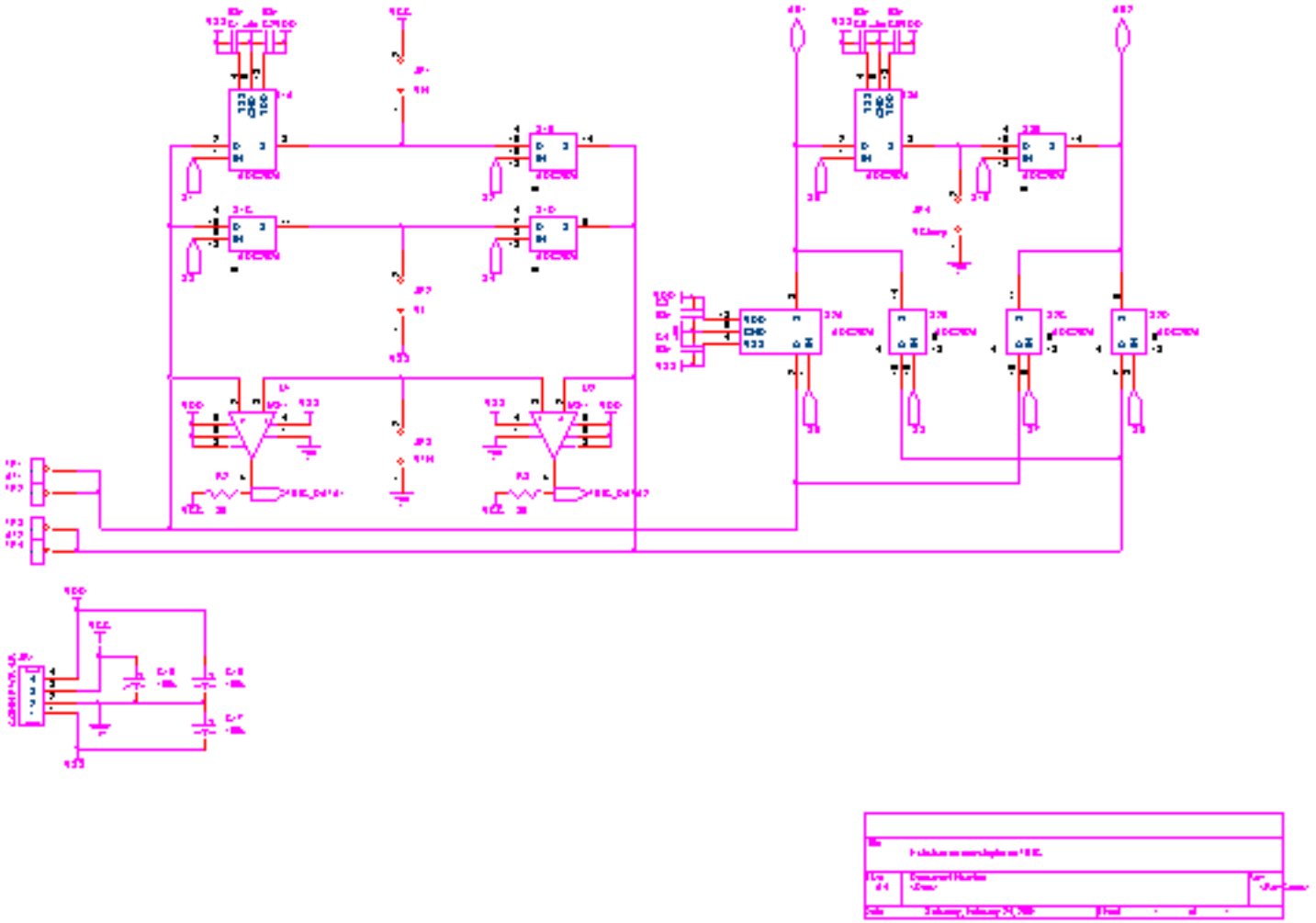
```
TITLE " ABM control register ";

SUBDESIGN ABM_CR
(
   TDI,
   TCK,
   en_clkDR,
   shift,
   en_uptDR,
   pin_comp
 : INPUT;
   TDO,
   D,
   C,
   B1,
   B2
 : OUTPUT;
)
(...)
```



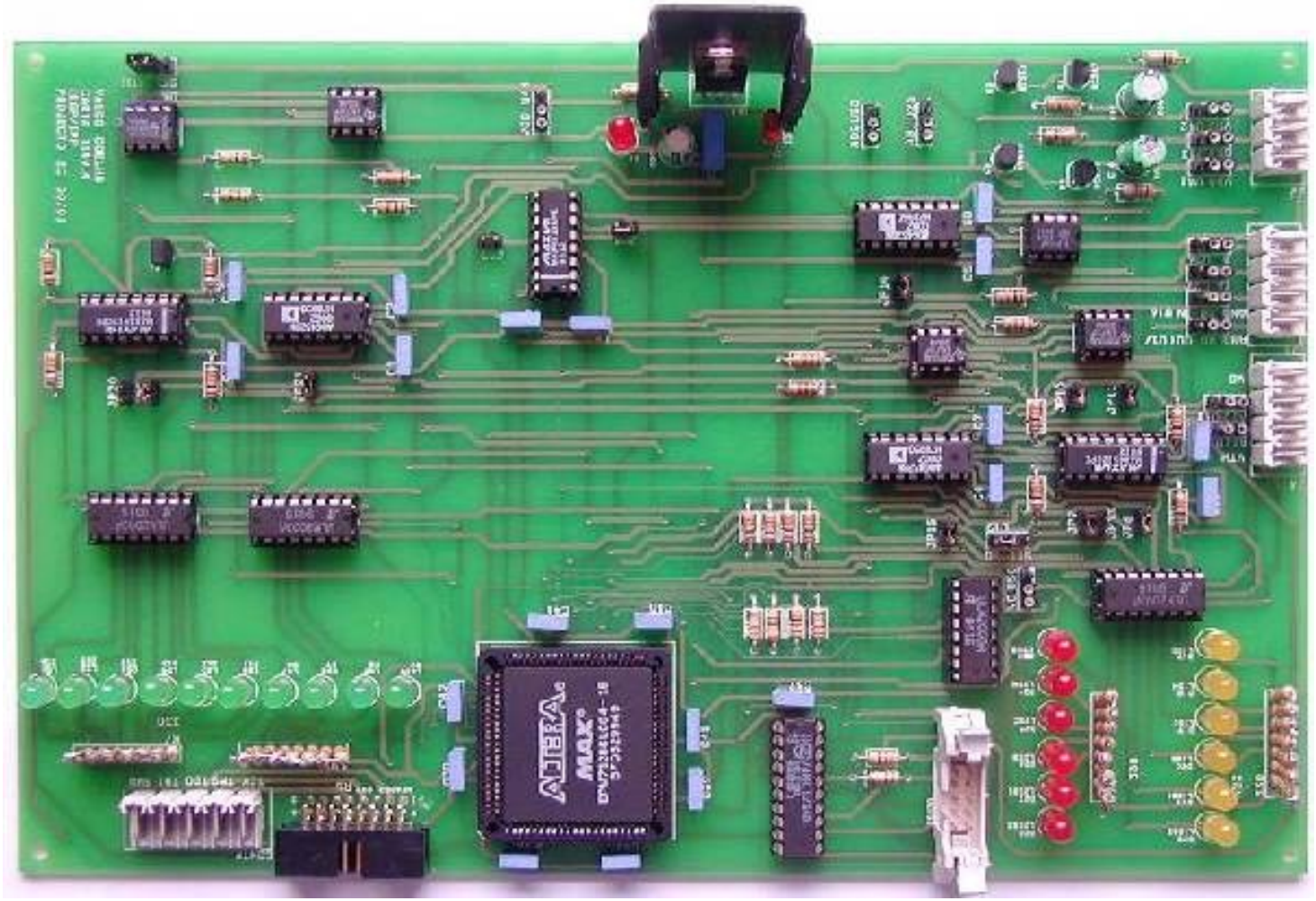
```
IF ( !en_clkDR ) THEN
   DATA = DATA ;
   CONTROL = CONTROL ;
   BUS1 = BUS1 ;
   BUS2 = BUS2 ;
ELSIF ( !shift ) THEN
   DATA = pin_comp;
   % Capture %
   CONTROL = GND;
   BUS1 = GND;
   BUS2 = GND;
ELSE
   DATA = TDI;      % Shift %
   CONTROL = DATA;
   BUS1 = CONTROL;
   BUS2 = BUS1;
END IF;

TDO = BUS2;
```

```
IF ( !en_uptDR ) THEN
   D_LATCH  = D_LATCH ;
   C_LATCH  = C_LATCH ;
   B1_LATCH = B1_LATCH ;
   B2_LATCH = B2_LATCH ;
 ELSE
   D_LATCH  = DATA;
     %  SHIFT -> LATCH –
     update %
   C_LATCH  = CONTROL ;
   B1_LATCH = BUS1 ;
   B2_LATCH = BUS2 ;
 END IF;

 D  = D_LATCH.q ;
 C  = C_LATCH.q ;
 B1 = B1_LATCH.q ;
 B2 = B2_LATCH.q ;

END;
```

# 2. Outils de développement du matériel (2)



# 2. Outils de développement du matériel (2)



# 2. Outils de développement du matériel (3)

## b. Le Verilog HDL (Hardware Description Language)

- **C'est un langage de description matériel de circuits logiques en électronique.**

- **Utilisé pour la conception d'ASICs (application-specific integrated circuits, circuits spécialisés) et de FPGAs (field-programmable gate array, réseau de portes programmables).**

- **À l'origine, il s'agissait d'un langage propriétaire, mais le succès grandissant de VHDL a incité ses concepteurs à faire de Verilog un standard ouvert;**

- **La syntaxe de Verilog est réputée largement inspirée du langage de programmation C. Ceci explique en partie son succès et sa diffusion rapide dans la communauté des ingénieurs qui ont déjà appris le langage C.**

# 2. Outils de développement du matériel (4)

- **La structure du langage Verilog permet de décrire les entrées et les sorties de modules électroniques, pour définir des portes logiques virtuelles.**

- **La combinaison de modules permet de réaliser des schémas électroniques virtuels complexes qu'il est alors possible de tester dans un programme de simulation. De tels tests ont pour objectif de :**

➢ **Valider le comportement des circuits décrits**

**(le résultat qu'ils délivrent est bien celui attendu) ;**

➢ **Valider les performances de ces circuits**

**(ils répondent dans un temps donné et les signaux qui parcourent les différents modules sont correctement synchronisés).**

# 2. Outils de développement du matériel (5)

## c. SystemC

- **Est souvent présenté comme un langage de description de matériel, au même titre que VHDL ou verilog.**

- **En fait, SystemC est un langage de description de plus haut niveau, puisqu'il permet une modélisation de systèmes au niveau comportemental.**

- **SystemC n'est pas un langage à part entière mais un ensemble de classes C++ qui introduisent les concepts nécessaires à la modélisation du matériel.**

- **SystemC permet donc de modéliser des systèmes matériels, logiciels, mixtes ou même non-partitionnés. Il est donc particulièrement approprié à la conception de systèmes de type SoC.**

- **SystemC intègre également la possibilité de simuler le modèle conçu, puis, par raffinements successifs, d'aboutir à une représentation implémentable.**

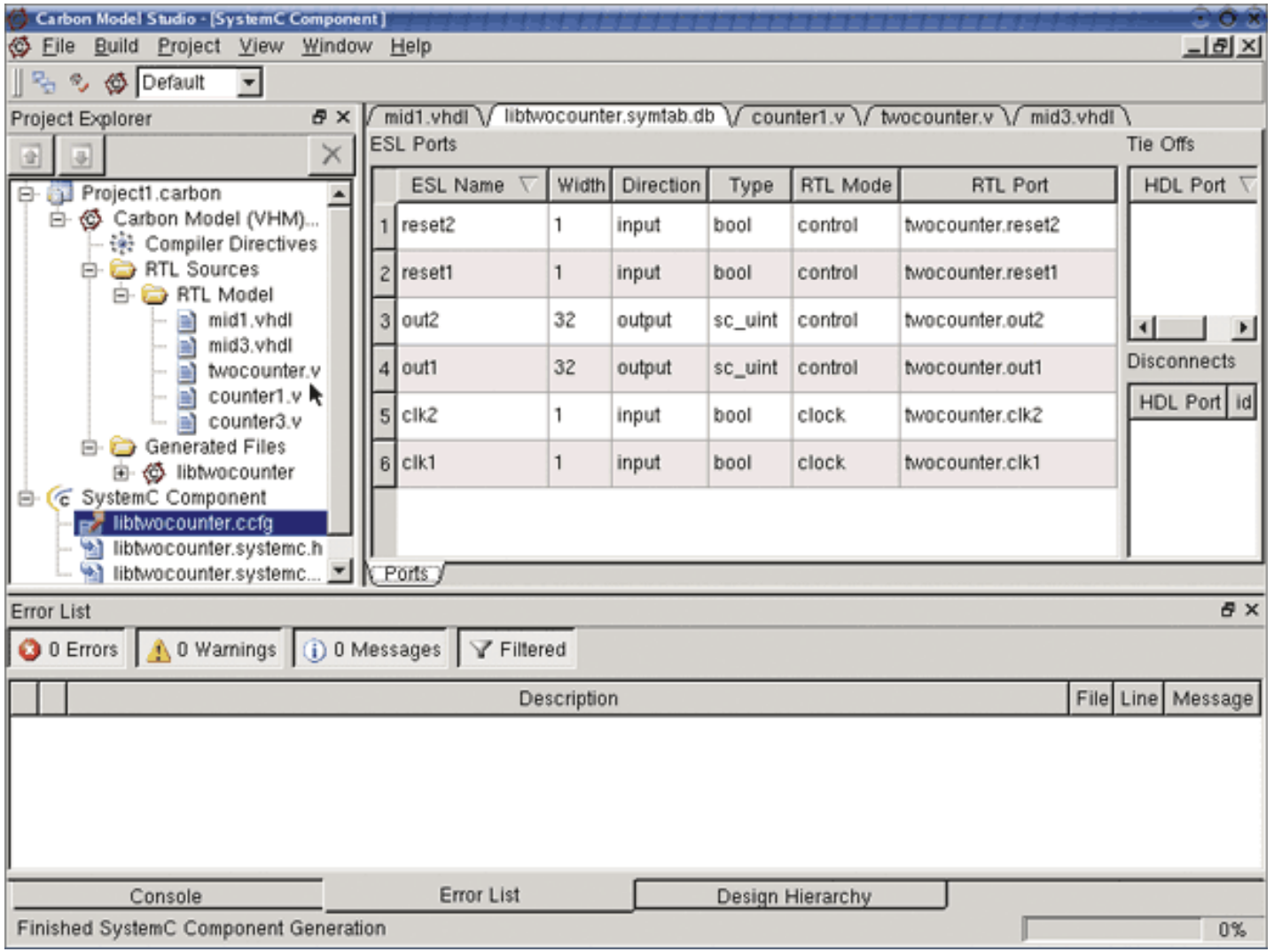
# 2. Outils de développement du matériel (6)

- **SystemC a été développé en commun par plusieurs entreprises. À cette fin, l'OSCI (Open SystemC Initiative) a été créé, chargé de diffuser, promouvoir et rédiger les spécifications de SystemC.**

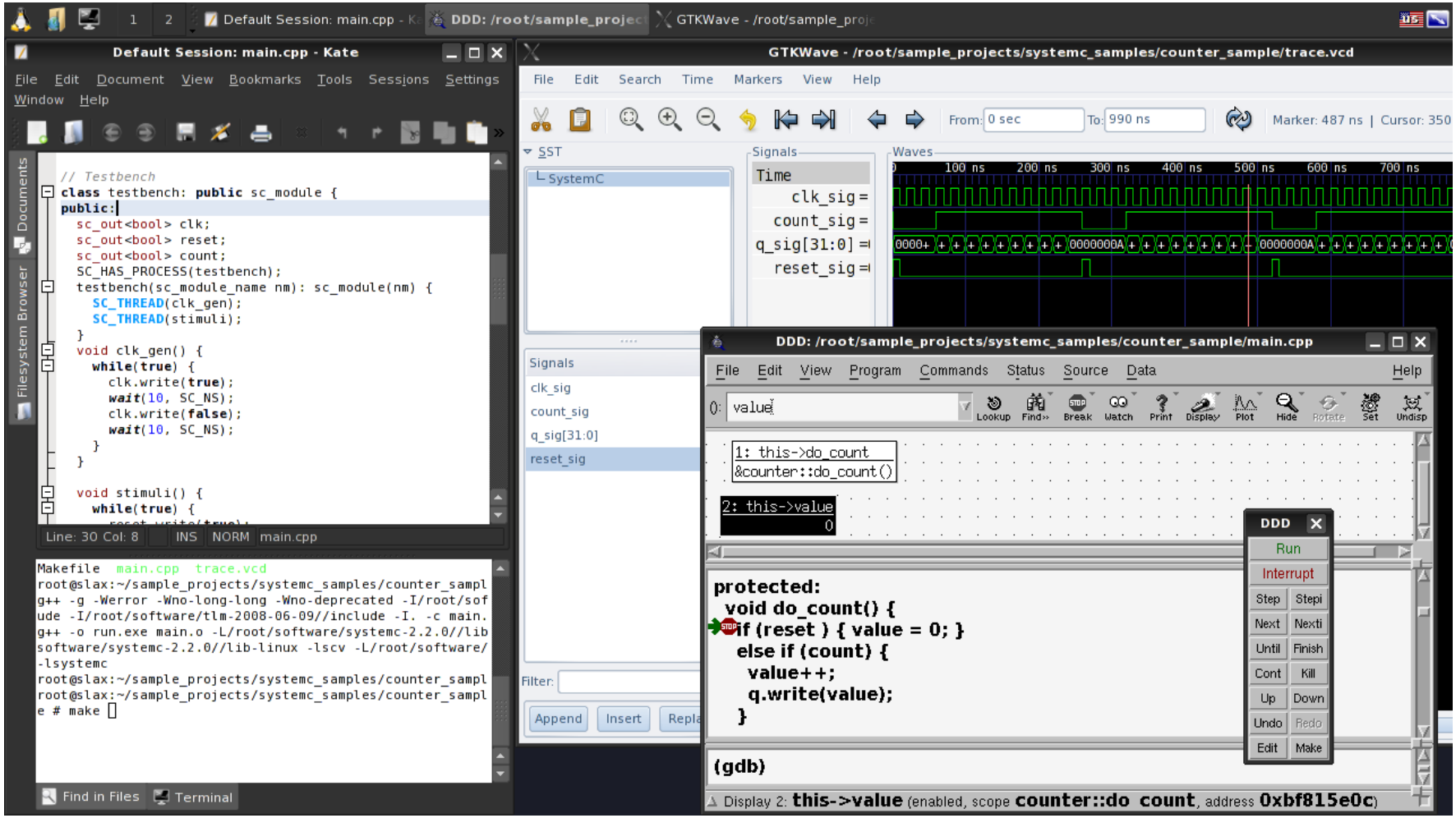
- **Ci dessous, les différentes versions de SystemC :**

  - **Version 0.9 par Synopsys en 1989**
  - **Version 1.0 par Frontier Design**
  - **Version 1.1 par CoWare en 2001**
  - **Création de L'OSCI (Open SystemC Initiative) en 2001**
  - **Version 2.0 par L'OSCI**
  - **Version 2.2 approuvée par IEEE appelée IEEE1666-2005 en 2005.**
  - **En Juin 2011 une alliance entre Accellera et l'OSCI est annoncée.**

# 2. Outils de développement du matériel (6)



# 2. Outils de développement du matériel (6)



# 3. Outils de développement du logiciel (1)

- **Le choix d'un langage de programmation repose sur les critères suivant :**
  - **En fonction de l'application, on choisit**
    - ✓ **Un langage temps-réel (dur ou mou) ou pas**
    - ✓ **Un langage interprété ou compilé**
    - ✓ **Un langage interactif ou non**
  - **En fonction de la cible, il faut**
    - ✓ **Que le langage soit supporté**
    - ✓ **Que les capacités soient suffisantes**

# 3. Outils de développement du logiciel (2)

## Quelques langages utilisés

### a. Le langage Assembleur

- Cas le plus fréquent
- Code généré très efficace
- Totalement non portable
- Difficile à maintenir

### b. Le langage Ada

- Utilisé notamment pour les systèmes critiques
- Langage offrant certaines garanties
- Manipulation de bits très facile
- Facile à maintenir

# 3. Outils de développement du logiciel (3)

**c. Le langage Esterel**

- **Conçu dans les années 1980.**
- **Ce langage est dit synchrone et réactif. Il est impératif et permet l'expression simple du parallélisme et de la préemption.**
- **En tant que langage appartenant à la classe des systèmes informatiques réactifs :**
  - **le programme ne se termine pas, il est cyclique ;**
  - **il possède une grande vitesse de réaction;**
  - **le parallélisme lui permet de gérer plusieurs capteurs et/ou incidents simultanément ;**
  - **ses processus peuvent être interrompus ou stoppés.**
  - **Langage orienté temps-réel dur.**

# 3. Outils de développement du logiciel (4)

**d. Le langage Forth**

- **Forth est un langage de programmation interactif atypique, crée par Charles H. Moore dans les années 1960.**
- **Une des importantes caractéristiques du langage est l'utilisation des mots, qui sont les constituants d'un programme Forth.**
- **Un mot Forth est l'équivalent des sous-programmes, fonctions ou procédures dans les autres langages.**
- **Forth a été utilisé principalement dans des systèmes embarqués et des contrôleurs, en raison de leur caractère compact et de la facilité d'utiliser des mots définis en assembleur dans des programmes de plus haut niveau.**
- **Parmi les applications les plus prestigieuses, on relèvera sa présence sur quelques missions de la NASA.**

# 3. Outils de développement du logiciel (5)

**e. Le langage Erlang**

- **Erlang est un langage de programmation (fonctionnels), supportant plusieurs paradigmes : concurrent, temps réel, distribué.**

- **Il possède des fonctionnalités de tolérance aux pannes permettant le développement d'applications à très haute disponibilité.**

- **Il a été créé par Ericsson, qui l'utilise dans plusieurs de ses produits, tel que le commutateur ATM AXD 301.**

- **Prévus pour les systèmes de communication répartis avec des contraintes de temps-réel mou**

# 3. Outils de développement du logiciel (6)

**f. Le langage C**

- **L'augmentation de performance des micro-contrôleurs permettant désormais la programmation des applications de ce type en langage C,**

- **C offre des opérateurs qui sont très proches de ceux du langage machine (manipulations de bits, pointeurs…). C'est un atout essentiel pour la programmation des systèmes embarqués.**

- **C est un langage près de la machine (microprocesseur) mais il peut être utilisé sur n'importe quel système ayant un compilateur C.**

# 4. Méthodes de conception d'un système embarqué (1)

## a. Approche traditionnelle

**1. Choix du matériel pour le système embarqué.**

**2. Donner le système ainsi conçu aux programmeurs.**

**3. Les programmeurs doivent réaliser un logiciel qui « colle » au matériel en n'exploitant que les ressources offertes.**

# 4. Méthodes de conception d'un système embarqué (2)

## a.1 Approche traditionnelle et la complexité des systèmes embarqués

- **Les systèmes embarqués sont de plus en plus complexes.**
- **Il est de plus en plus difficile de penser à une solution globale optimisée du premier jet.**
- **Il est de plus en plus difficile de corriger les « bugs ».**
- **Il est de plus en plus difficile de maintenir le système au cours du temps.**
- **Dans l'électronique embarquée, le temps de conception devient de plus en plus réduit (pour diminuer les coûts, gagner un appel d'offre).**
- **Et le système doit toujours être aussi sûr et robuste.**
- **En conséquence, l'approche traditionnelle de développement d'un système embarqué doit évoluer...**

# 4. Méthodes de conception d'un système embarqué (3)

## a.2 Solution a la complexité (modèle à composant)

- **Dans le processus de conception du système, on doit garder un niveau d'abstraction important le plus longtemps possible.**

- **Le système doit pouvoir être décomposé en sous-systèmes suivant une hiérarchie logique (approche par composant).**

- **Si le design change, on doit pouvoir en réutiliser une bonne partie (design reuse).**

- **Il ne faut pas jeter à la poubelle un précédent design et repartir à zéro.**

- **On doit pouvoir utiliser des outils de vérification automatique.**

# 4. Méthodes de conception d'un système embarqué (4)

## a.3 Méthode de conception à base de composant

### a.3.1 La méthode BIP (Behavior-Interaction-Priorities)

- **Propose un cadre pour la modélisation des systèmes embarqués avec approche à composants.**

- **Dans cette approche, un composant est considéré comme une superposition de trois modèles :**

  - **Un modèle comportemental pour décrire le comportement dynamique.**
  - **Un modèle d'interaction qui décrit l'architecture du système induit par la connexion inter composant**
  - **Un modèle de priorités qui permet de coder plusieurs modes d'exécution parallèle.**

# 4. Méthodes de conception d'un système embarqué (5)

## a.3.2 La méthode Metropolis

- **Fournit une infrastructure basée sur un méta-modèle (modèle de modèle) inspiré de POLIS (approche de conception des systèmes temps réel)**

- **Cette approche distingue entre deux modèles : modèle fonctionnel et un autre architectural.**

- **Metropolis inclut des outils de simulation, vérification et de synthèse afin de répondre aux besoins des concepteurs de systèmes embarqués.**

- **C'est une approche plutôt dédié aux concepteurs de matériel.**

# 4. Méthodes de conception d'un système embarqué (6)

## a.3.3 La méthode Ptolemy

- **Probablement l'environnement de modélisation de systèmes hétérogènes le plus connu. Composants (continus, discrets, synchrones, asynchrones, etc…)**

## a.3.4 Autre méthodes

- **Cossap : est un environnement de conception de systèmes de traitement du signal**
- **Music : conçu pour la conception de systèmes hétérogènes multi langages.**
- **Simulink : environnement interactif pour la modélisation, l'analyse et la simulation des systèmes dynamiques.**
- **Easy5 : développé par la compagnie Boeing. Est un outil de modélisation, de conception et de simulation des systèmes dynamiques.**

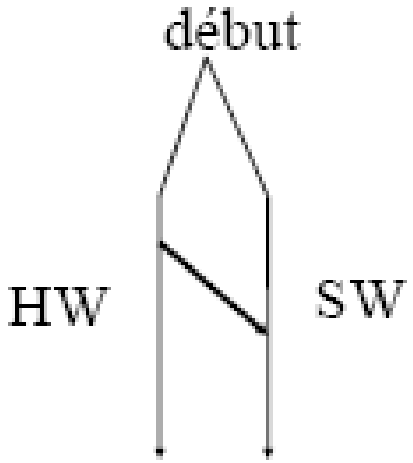
# 4. Méthodes de conception d'un système embarqué (7)

## b. L'approche codesign Hardware/Software

- **Le codesign dans la méthodologie de conception d'un système embarqué est de plus en plus utilisé.**
- **Le codesign permet de concevoir en même temps à la fois le matériel et le logiciel pour une fonctionnalité à implémenter.**
- **Cela est maintenant possible avec les niveaux d'intégration offerts dans les circuits logiques programmables.**
- **Le codesign permet de repousser le plus loin possible dans la conception du système les choix matériels à faire contrairement à l'approche classique où les choix matériels sont faits en premier lieu !**

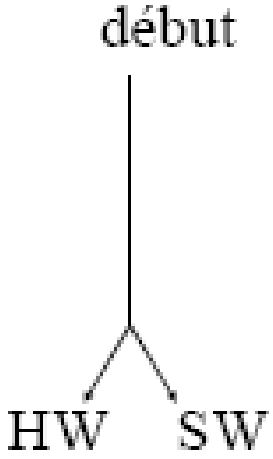
# 4. Méthodes de conception d'un système embarqué (8)

**Conception traditionnelle**




Réalisée par des groupes d'ingénieurs indépendants

**Codesign (flot concurrent)**




Réalisée par le même groupe d'ingénieurs en coopération

# 4. Méthodes de conception d'un système embarqué (9)

## b.1 Les étapes du codesign

• **Durant le processus de codesign, on distingue les étapes suivantes :**

– **Spécifications : liste des fonctionnalités du système de façon abstraite.**

– **Modélisation : conceptualisation et affinement des spécifications produisant un modèle du matériel et du logiciel.**

– **Partitionnement.**

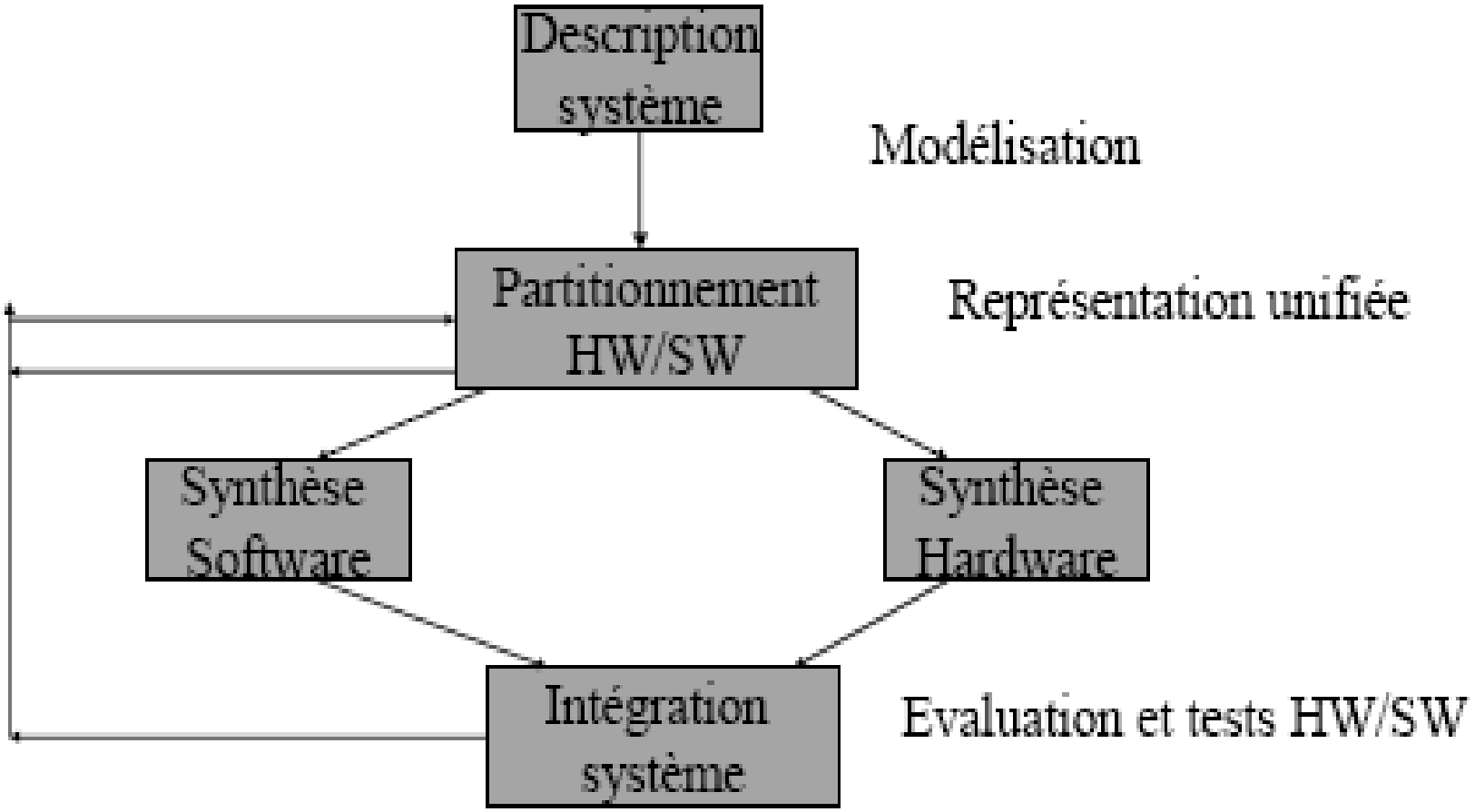
– **Synthèse et optimisation : synthèse matérielle et compilation logicielle.**

– **Validation : cosimulation.**

– **Intégration.**

– **Tests d'intégration.**

# 4. Méthodes de conception d'un système embarqué (10)



# 4. Méthodes de conception d'un système embarqué (11)

## b.2 Avantages du codesign

- **Le codesign est intéressant pour la conception de systèmes embarqués (ou de SoC) :**

  **– Amélioration des performances**
  **– Reconfiguration statique ou dynamique en cours de fonctionnement.**
  **– Indépendance vis à vis des évolutions technologiques des circuits logiques programmables.**

# 4. Méthodes de conception d'un système embarqué (12)

## b.3 Codesign et systèmes embarqués : un bilan

- **Les systèmes embarqués étant de plus en plus complexes, une méthodologie rigoureuse de conception doit être maintenant mise en oeuvre.**

- **Le codesign fait partie intégrante de cette méthodologie.**

- **Cela est maintenant possible grâce à des niveaux d'intégration sur silicium très importants des circuits logiques programmables.**

- **L'apparition de modules IP, véritables circuits électroniques sous forme logicielle n'a fait qu'accentuer l 'importance du codesign et du design-reuse.**

# 4. Méthodes de conception d'un système embarqué (13)

## b.4 Les modules IP (Intellectual Property)

- **Les IPs constituent un ensemble de composant sur étagère prêts à l'emploi qui peuvent être réutilisés pour la conception de systèmes matériel plus complexes tels que les systèmes sur puce (SoC).**

- **Un IP consiste en un bloc fonctionnel complexe fortement optimisé, la normalisation des interfaces OCP (Open Core Protocol) permet la réutilisation de ce type de composants.**

- **La spécification OCP 2.2 a été définie pour assurer l'interopérabilité entre blocs d'IP via des interfaces configurables.**

- **Plusieurs types d'IP existent, ils sont répartis en plusieurs classe telles que les interfaces de bus, l'acquisition des données, les processeurs, ect.**

# 5. Programmation d'un système embarqué (1)

- **La compilation est rarement réalisée sur le système embarqué lui-même car il n'inclut pas les outils de développement très consommateurs de place.**

- **La notion de compilation croisée est fortement utilisée, afin de palier les limite des SEs.**

## 3.1 La compilation croisée

- **Compilation effectuée sur une machine puissante avec code assembleur génèré pour le CPU du système embarqué.**

- **Le code est transmis au système embarqué par une procédure du "flashage".**

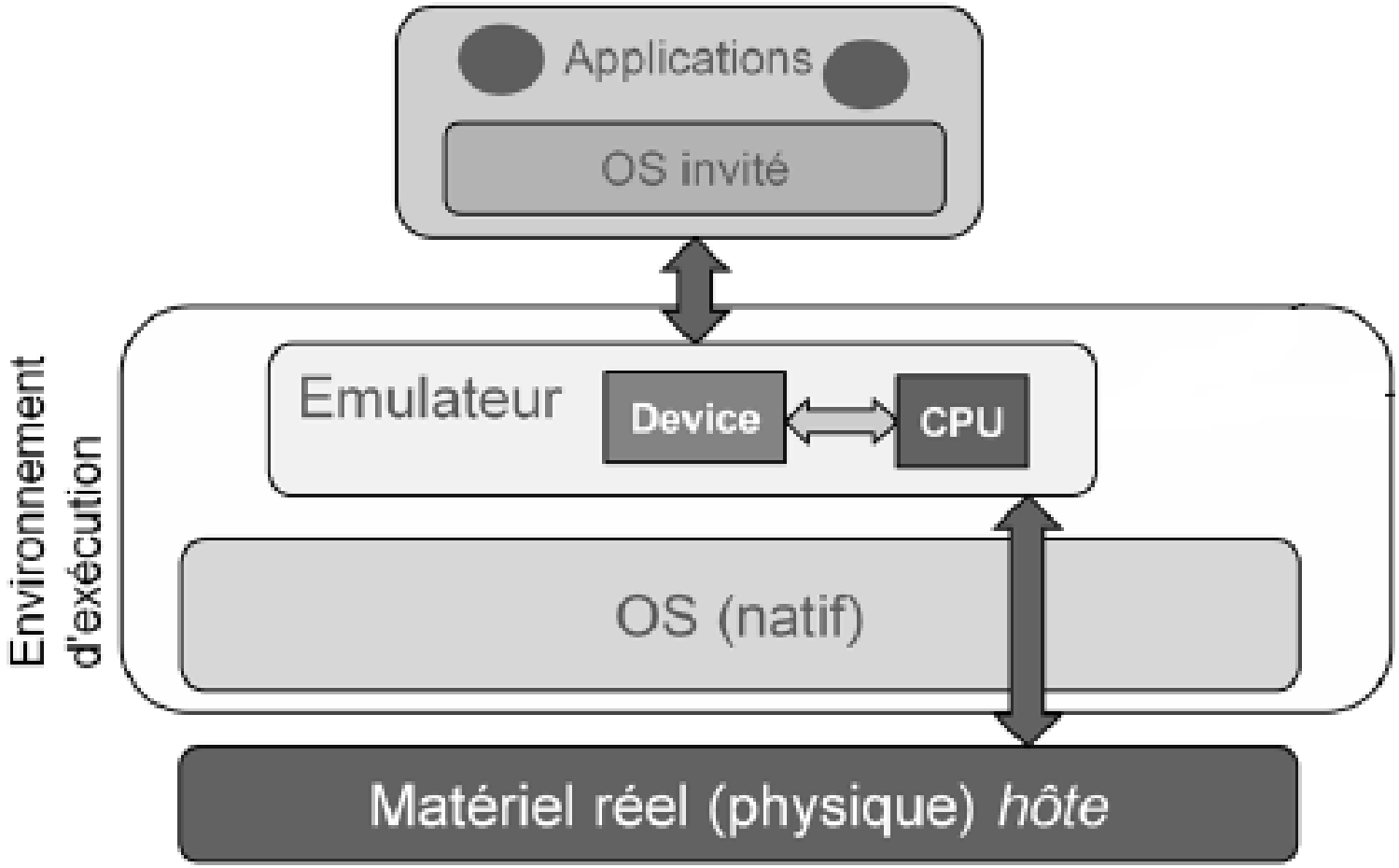
- **L'environnement de compilation croisée s'appelle Tool chain.**

# 5. Programmation d'un système embarqué (2)

## 3.2 Emulation de SE

- **Pour accélérer le développement on peut développer le système dans un émulateur**
- **Il faut voir dans l'émulation une imitation du comportement physique d'un matériel par un logiciel, et ne pas la confondre avec la simulation.**
- **L'émulateur reproduit le comportement d'un modèle dont toutes les variables sont connues, alors que le simulateur tente de reproduire un modèle mais en devant envisager une partie des variables qui lui sont inconnues**
- **Le recours à un émulateur, selon le contexte, permet de faciliter le développement ou le débogage d'un système.**
- **Les émulateurs de processeur, et de plate-forme quasi-complète, sont nombreux et relativement stables.**
- **Ils constituent aujourd'hui des outils de choix pour la mise au point d'applications embarquées. Des OS complets peuvent facilement tourner sur un émulateur.**

# 5. Programmation d'un système embarqué (3)



# 5. Programmation d'un système embarqué (4)

- **Le système d'exploitation (OS) natif et l'émulateur fournit un environnement d'exécution complet pour du code compilé pour une architecture cible différente que celle de la machine hôte.**

- **Par exemple, une application cross-compilée pour un processeur ARM peut tourner sur la machine hôte, équipée d'un processeur x86 par exemple.**

- **De même, un OS entier (avec ses applications) peut tourner sur ce même environnement.**

- **L'émulateur peut émuler des périphériques présents sur la carte cible en respectant les conditions d'accès.**

# 5. Programmation d'un système embarqué (5)

**Exemple d'émulateurs**

- **QEMU: Un excellent émulateur de matériel, comprenant une vaste gamme de processeurs et périphériques suportés.**
  - **Rapide, optimisé**
  - **Tourne sur x86**
  - **Extension KQEMU**

# 5. Programmation d'un système embarqué (6)

**Plate-formes supportée par QEMU:**

- **PC (x86 or x86_64 processor)**
- **ISA PC (old style PC without PCI bus)**
- **PREP (PowerPC processor)**
- **G3 Beige PowerMac (PowerPC processor)**
- **Mac99 PowerMac (PowerPC processor, in progress)**
- **Sun4m/Sun4c/Sun4d (32-bit Sparc processor)**
- **Sun4u/Sun4v (64-bit Sparc processor, in progress)**
- **Malta board (32-bit and 64-bit MIPS processors)**
- **MIPS Magnum (64-bit MIPS processor)**
- **ARM Integrator/CP (ARM)**
- **ARM Versatile baseboard (ARM) (iMX21)**
- **ARM RealView Emulation baseboard (ARM)**
- **Spitz, Akita, Borzoi, Terrier and Tosa PDAs (PXA270 processor)**
- **Luminary Micro LM3S811EVB (ARM Cortex-M3)**
- **Luminary Micro LM3S6965EVB (ARM Cortex-M3)**
- **Freescale MCF5208EVB (ColdFire V2).**
- **Arnewsh MCF5206 evaluation board (ColdFire V2).**
- **Palm Tungsten|E PDA (OMAP310 processor)**
- **N800 and N810 tablets (OMAP2420 processor)**
- **MusicPal (MV88W8618 ARM processor)**
- **Gumstix "Connex" and "Verdex" motherboards (PXA255/270).**
- **Siemens SX1 smartphone (OMAP310 processor)**

# 5. Programmation d'un système embarqué (6)

**Autre émulateur:**

- **Bochs, VMWare (x86),**
- **PearPC (PCC),**
- **Softgun (ARM)…**