Introduction à la Conception de SoC

Michel Auguin

Laboratoire I3S - Université de Nice Sophia Antipolis, CNRS

Plan de l'Exposé

Introduction

Le nombre de SoC produits croît (~30% par an) en fonction des avancées de la technologie et des besoins des applications

Les traitements dans les applications sont de plus en plus complexes. La mobilité renforce les contraintes (coûts, énergie)

GPS
GSM/GPRS/EDGE/UMTS
WLAN
PDA
MP3
MPEG4
Internet (Java)
IHM
Reconnaisance vocale
Jeux .....

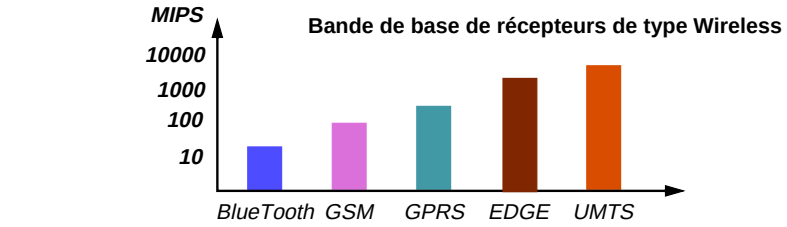
Puissance de Calcul

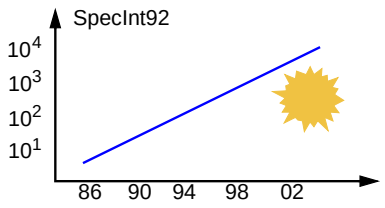
MIPS/Watt

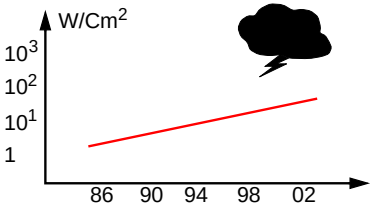
MIPS/mm^2

Mémoire intégrée

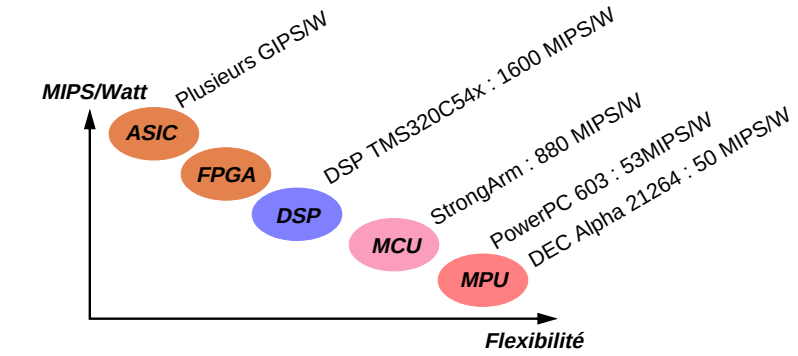
Evolution des besoins en puissances de traitement intégrées





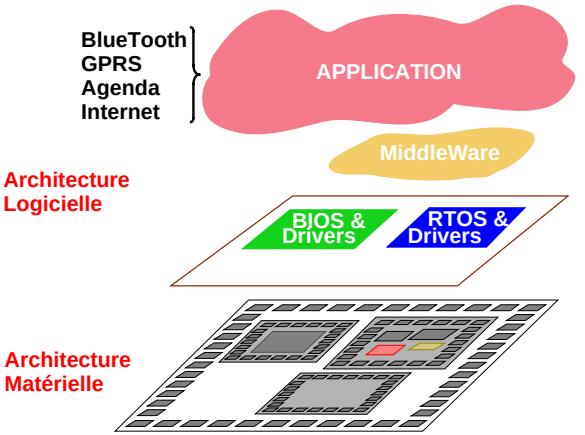


Compromis Performances/Consommation



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

Architecture d'un SOC



Architecture Logicielle

Architecture Matérielle

BIOS :
Basic I/O System

RTOS :
Real Time OS

Nombreux Choix HW/SW

Performances ?
Temps Réel ?
Best Effort ?
Consommation ?
Coûts ?

Tâches de type Traitement du Signal et Images

- Bande de base GSM
- MPEG4
- Opérations graphiques

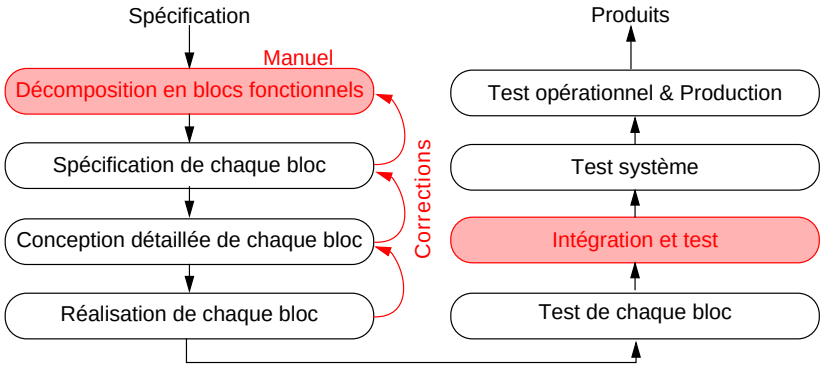
Tâches orientées contrôle

- Protocole réseau
- Power Management
- IHM

Les tâches se partagent les ressources du SoC

Contraintes Temps, Consommation, Coûts vérifiées ?

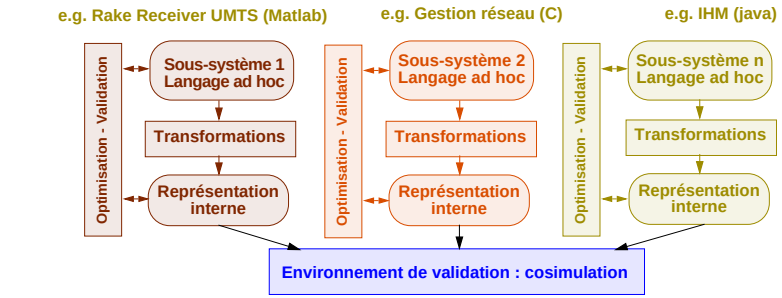
Flot classique de conception



Blocs
- Matériel
- Logiciel
- Communication

Chaque bloc est conçu de façon indépendante
Techniques de vérification : (co-)simulation
Test des communications : en phase d'intégration

Approche "Langage" de conception de chaque bloc fonctionnel



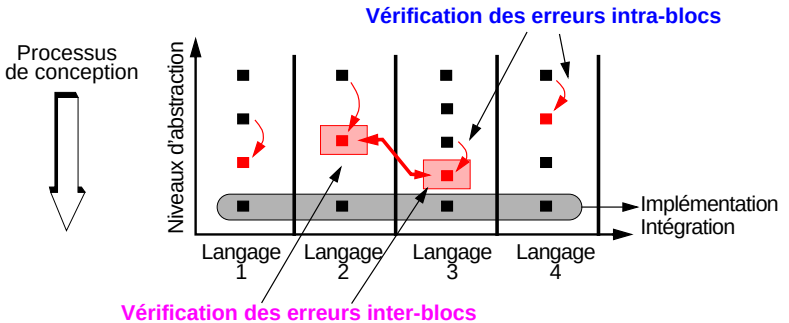
Cosimulation C - VHDL
Cosimulation Assembleur (ISS) - VHDL

- Aucune optimisation globale : optimisations par sous-système (dépendant langage)
- Nécessite de décrire précisément les communications et les interfaces
- Les migrations éventuelles entre sous-systèmes ne sont pas aisées

Conception verticale suivant chaque langage

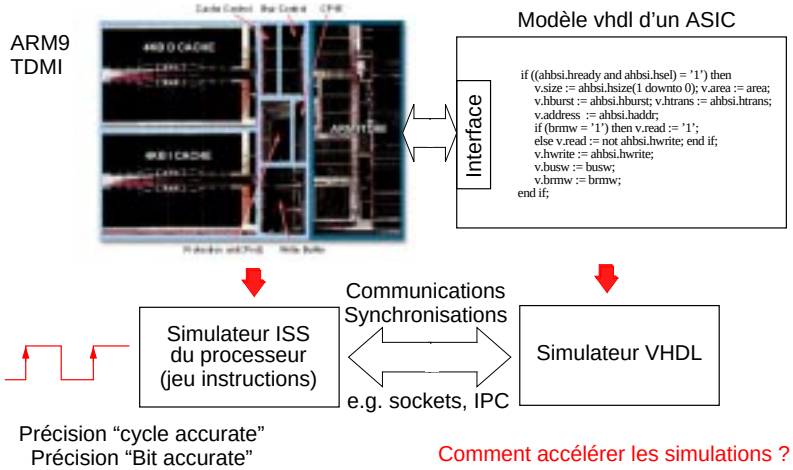
Approche traditionnelle :

Décomposition en blocs fonctionnels et conception verticale



- Intégration explicite : communications détaillées
- Cosimulation

Vérification par co-simulation



Précision "cycle accurate"
Précision "Bit accurate"

Comment accélérer les simulations ?

Limitations de l'approche "Langage"

Optimisations locales

Vérification en simulation de chaque bloc

Absence de vérification globale hors cosimulation

Difficultés pour déterminer a priori les performances globales

- *Temps réel vérifié ?*
- *Temps d'exécution satisfaisant ?*
- *Consommation d'énergie maîtrisée ?*

Le debugging est délicat

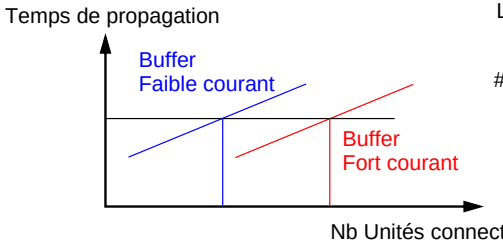
- *Localisation des erreurs*
- *Couverture de test*

Où les choses se compliquent encore : les communications

Temps de communication sur les fils dépendent de la conception :

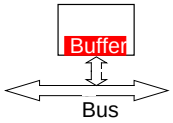
Nombre d'unités connectées sur le bus

Incertitude sur les temps de propagation dans les fils



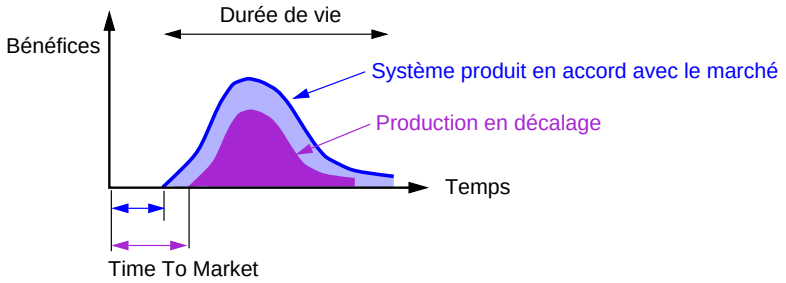
Le bruit dans le chip augmente avec le courant

Watt augmente avec le courant



☞ **Les performances réelles sont connues tardivement**

Et n'oublions pas la contrainte du Time to Market



Les produits ont une durée de plus en plus faible

Réduire le «time to market»

Réutilisation pour concevoir d'autres produits (rentabiliser)

Comment aborder ce challenge ?

Améliorer le flot de conception

- *Méthodes qui opèrent sur des descriptions hétérogènes*
- *Vérification formelle*
- *Compilation*
- *Analyse plus précises : WCET, consommation, ordonnançabilité*
- *Architectures adaptées : NoC*

Méthodes d'exploration multi-critères : temps/consommation/coûts

Renforcer la réutilisaton

- *Notion d'IP et de Plate-forme*

Plan de l'Exposé

Introduction : Contexte et Besoins

☞ *Evolution des Méthodes de conception*

- *Modélisation des applications*
- *Vérification*
- *Compilation*
- *Analyses et Estimations*
- *Architectures adaptées : exemple des NoC*
- *Exploration d'architectures*
- *Plateforme et IP*

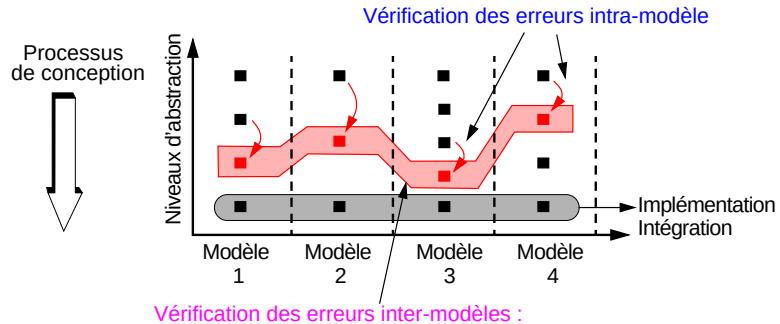
Exemple : la plateforme TI- OMAP

Conclusion

Evolution souhaitée des méthodes de conception

Approche de modélisation intégrée :

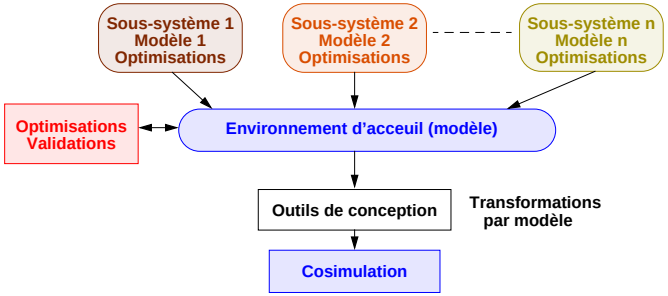
L'environnement permet des raffinements "sans coutures"



- Communications inter-modèles : types prédéfinis
- Vérifications sémantiques

[www.vptinc.com]

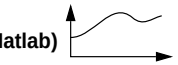
Approche "Modèle"

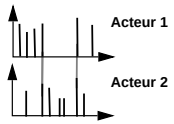


- Optimisations locales et globales
- Exploration de l'espace de conception plus aisée
- Les communications et interfaces sont décrites de manière plus abstraite

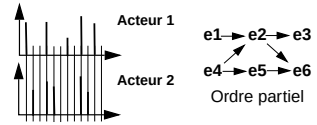
Les modèles diffèrent par leur sémantique

- **Calcul analogique, équations différentielles (temps continu - Matlab)**
  L'évolution du modèle s'effectue dans l'ensemble des réels (ordre total)
- **Temps échantillonné (équation aux différences)**
  L'évolution du modèle s'effectue dans un sous-ensemble des réels (ordre total)
- **Systèmes à événements discrets**
  Idem
- **Réseaux de Processus (Kahn), Graphes Flots de Données**
  Ordre partiel sur les événements
- **Processus séquentiels avec rendez-vous (CSP)**
  Ordre partiel sur les événements
- **Systèmes réactifs synchrones (Esterel)**
  (Ordre total)
- **Codesign Finite State Machine (CFSM)**
  (Globalement partiel, localement total)





Evénements discrets



Réactif synchrone

Ordre partiel

Choix du modèle de spécification en fonction du type de l'application

Classiquement les concepteurs considèrent qu'un système est dominé :

Par le contrôle

Réactions à des événements discrets :
Système cadencé par les événements

- **Systèmes à événements discrets**
- **Systèmes réactifs synchrones**
- **Machine d'Etats Finis**
- **Réseaux de Petri**
- **Processus concurrents**

⇨ Contrôle/Commande de processus

Par les données

Sorties fonctionnellement dépendantes des entrées :
Entrées périodiques ou pseudo-périodiques

- **Kahn Process Networks**
- **Dataflow Process Networks**

⇨ Traitement du signal et des images

Langages associés aux modèles

Systèmes réactifs synchrones	⇨	Esterel, Lustre, Signal, ECL	Codesign Finite State Machine
Systèmes à événements discrets	⇨	VHDL, Verilog	
Machine d'Etats Finis	⇨	*Charts (StateCharts, ...)	SDL (Standard de l'ITU)
Processus concurrents	⇨	CSP, Occam, Lotos	
Dataflow Process Networks *Kahn Process Networks*	⇨	Dynamic Data Flow Synchronous Data Flow	
Modèle Objet	⇨	Java, C++, OO-VHDL	

SystemC : Classes C++ pour décrire réactivité, Kahn Process Networks, CSP

Environnements pour applications hétérogènes

☞ **Un seul modèle de calcul est insuffisant pour décrire toute l'application**

Exemples d'approches globales proposées :

Langage et sémantique pour accueuillir différents modèles de calcul

ROSETTA
Projet de l'initiative SLDL
System Level Description Language
http://www.inmet.com/SLDL/

VHDL-AMS

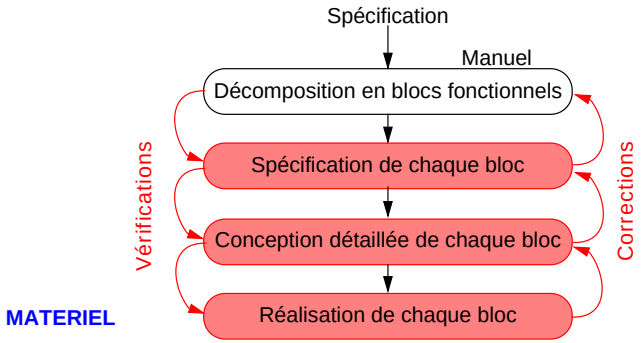
Plusieurs modèles de calcul intégrés dans un même environnement

PTOLEMY
- ***Systèmes réactifs synchrones***
- ***Machines d'Etats Finis hiérarchisées***
- ***Processus communicants***
- ***Dataflow Process Networks***
- ***Systèmes à événements discrets***

SPW, COSSAP, Matlab

☞ Hormis les techniques à base de cosimulation, il existe peu de méthodes de conception unifées qui opèrent sur une description hétérogène.

Vérification Formelle

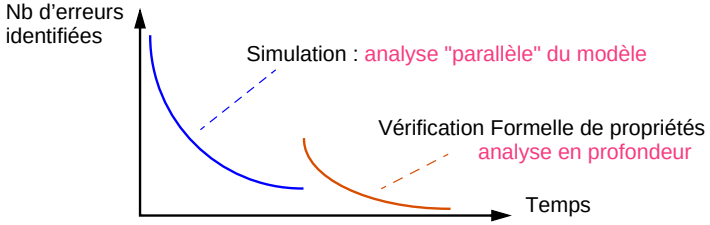


Comment s'assurer que chaque niveau de description est correct ?

Approche classique : Simulation i.e. animation du modèle

Exécution sur des séquences de test

☞ **Vérification par preuve formelle de propriétés**



Difficulté pour le concepteur : déterminer les propriétés qui font du sens

☞ **Vérification par équivalence de modèles**

Modèle abstrait ⇨ Comportements "Equivalents" ? ⇦ Modèle après raffinement

Compilation pour Cœurs de Processeurs

Techniques classiques de compilation : processeurs hautes performances

- *Mémoire virtuelle, caches données et instructions*
- *Fichiers de registres homogènes*
- *Renommage, lancement dans le désordre ...*

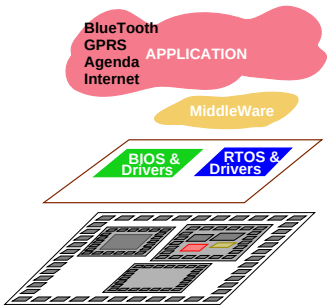
Cœurs de processeurs :

- *Mémoire locale vs Cache et mémoire off-chip*
- *Mémoire parallèle on-chip et Registres spécifiques répartis (DSP)*
- *Temps réel et Consommation*
- *RTOS*

Les contraintes et objectifs sont différents

Analyses et estimations plus précises

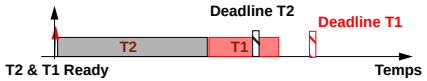
Analyses temporelles



Les échéances des tâches critiques sont elles vérifiées ?

❑ *Déterminer les WCET des tâches*
Worst Case Execution Time

❑ *Déterminer si tâches ordonnançables*



Analyse d'ordonnançabilité

Problèmes : Architectures peu déterministes (Caches, branchements, ...)
Politique d'ordonnancement (Mono vs Multiprocesseur)
Gestion d'événements sporadiques

Comportement des autres tâches ? (Agenda, IHM ...)

Analyses Consommation

$P_{dynamique} = P_{court\text{-}circuit} + P_{commutation}$

Due à l'activité du circuit
Prépondérante en fonctionnement

$P_{dynamique} = P_{court\text{-}circuit} + P_{commutation}$

90% de la consommation globale

$$P_{commutation} = V_{dd}^2 F \sum \alpha_i C_i$$

α_i : activité moyenne de chaque élément logique
C_i : capacité chargée et déchargée à chaque variation d'état de l'élément logique
F : fréquence de fonctionnement

Optimisation de la Consommation

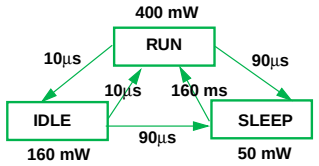
Pour Réduire $V_{dd}^2 F \sum \alpha_i C_i$

Technologie, Choix des composants

Limiter les changements d'états inutiles

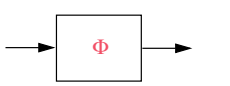
(Clock Gating, encodage d'adresses, mise en veille)

Changements de modes de l'Intel StrongARM

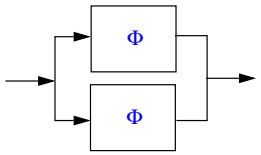


Parallélisme (ou pipeline) réduit l'énergie

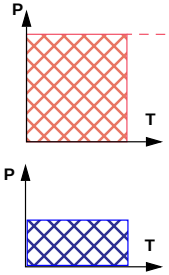
Puissance = $V_{dd}^2 F \Sigma \alpha_i C_i$



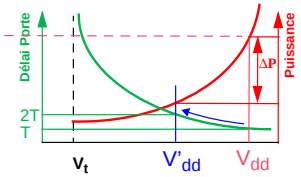
Puissance P, Fréquence F
Temps exécution T
Tension alimentation V_{dd}



On se ramène au temps T :
Fréquence F/2
Tension alimentation V'_{dd}



Puissance par Φ :
$0{,}5V'^2_{dd} F \Sigma \alpha_i C_i$
Gain en puissance :
V_{dd}^2 / V'^2_{dd}



Délai = $\frac{kC_L \cdot V_{dd}}{(V_{dd} - V_t)^2}$

Permet de fixer V'_{dd}

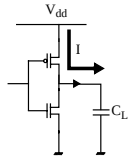
Ex : V_{dd} = 3,7V; V_t = 0,7V
V'_{dd} = 2,4V
Gain en puissance 2,36

Pour Réduire $V_{dd}^2 F \Sigma \alpha_i C_i$

Réduire V_{dd} implique de réduire F donc la performance : Voltage Scaling

En effet, temps de réponse d'une porte :

$T_d = \frac{C_L V_{dd}}{I}$ $I = \frac{(V_{dd} - V_t)^2}{k}$ V_t : tension de seuil

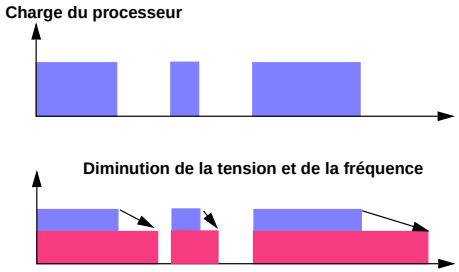


Exemple avec V_t = 0,7V et V_{dd} passe de 5V à 1,5V : Temps de réponse x 8

Gestion Dynamique de la fréquence et de la tension (DVS)

Gestion par OS de la vitesse et de la tension d'alimentation du processeur

Vitesse et tension d'alimentation processeur : Intel Xscale 4 V_{dd} et 11 clocks

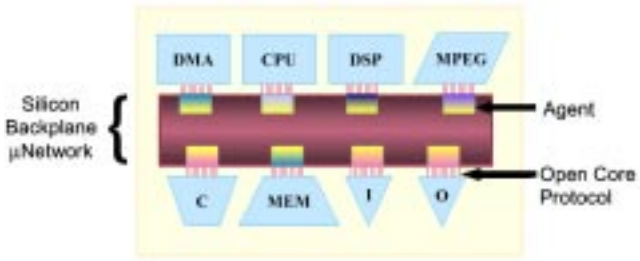


Temps d'exécution plus longs : échéances toujours respectées ?

Architectures Adaptées : Exemple des NoC (Network On Chip)

Architecture à base de bus : Temps de communication dépendent de la conception

Solution Microréseau Sonics



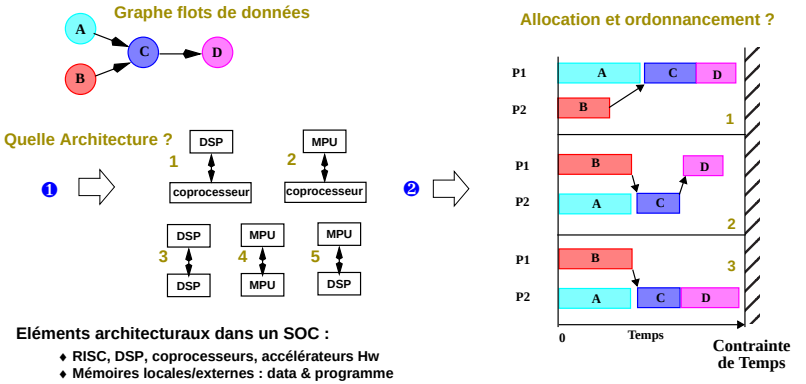
Communications via Agents avec protocole OCP (Open Core Protocol)

Réseau : Bus pipeliné avec gestion TDMA, latence fixe
Bande passante configurable à allocation statique
Reconfiguration dynamique possible

Outils logiciels de configuration et de synthèse du réseau (et agents)

Méthodes d'Exploration Multi-Critères

Illustration des difficultés



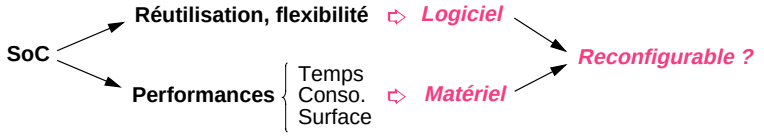
Eléments architecturaux dans un SOC :

- RISC, DSP, coprocesseurs, accélérateurs Hw
- Mémoires locales/externes : data & programme
- Caches
- Ports I/O, caractéristiques des bus ...

Ordonnancement statique non préemptif

Renforcer la réutilisation : IP et Plateformes

Déduire l'effort de conception : privilégier réutilisation et flexibilité

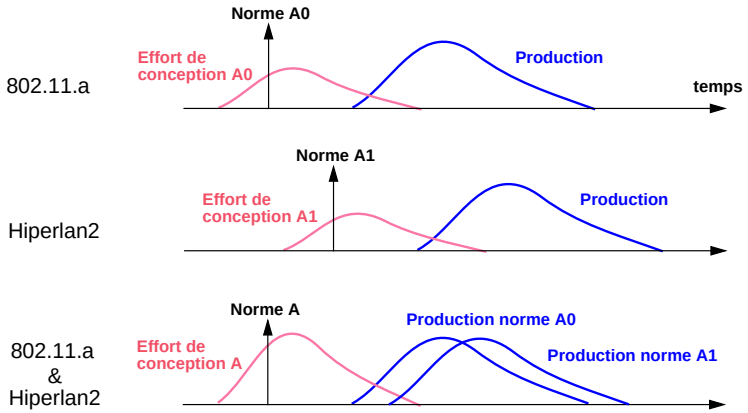


Conception de SoC par assemblage d'IP (Intellectual Property)

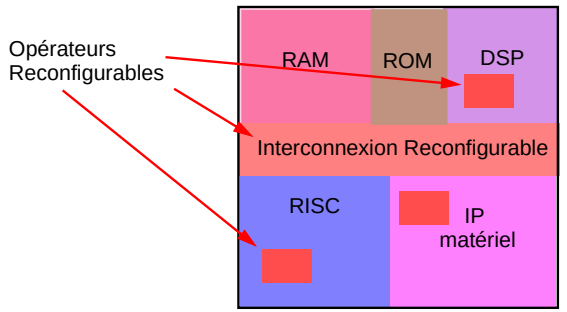
SoC pour une classe d'applications : les Plateformes

"Solution industrielle"

Plateforme : Factoriser les efforts



Utilisation de la Technologie Reconfigurable



Compromis Performances/Consommation/Surface/Flexibilité

Outils pour le Reconfigurable ?

Plan de l'Exposé

Introduction : Contexte et Besoins

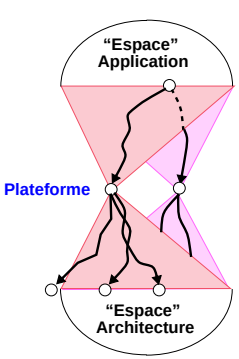
Evolution des Méthodes de conception

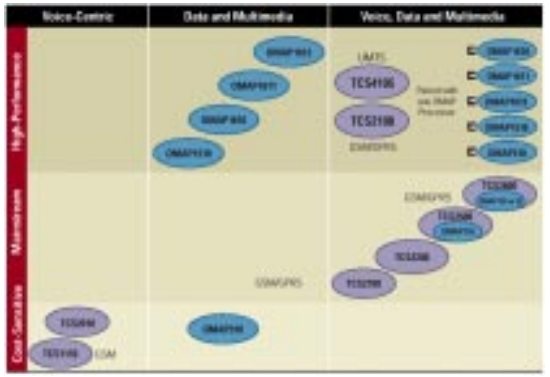
- *Modélisation des applications*
- *Vérification*
- *Compilation*
- *Analyses et Estimations*
- *Architectures adaptées : exemple des NoC*
- *Exploration d'architectures*
- *Plateforme et IP*

☞ *Exemple : la plateforme TI- OMAP*

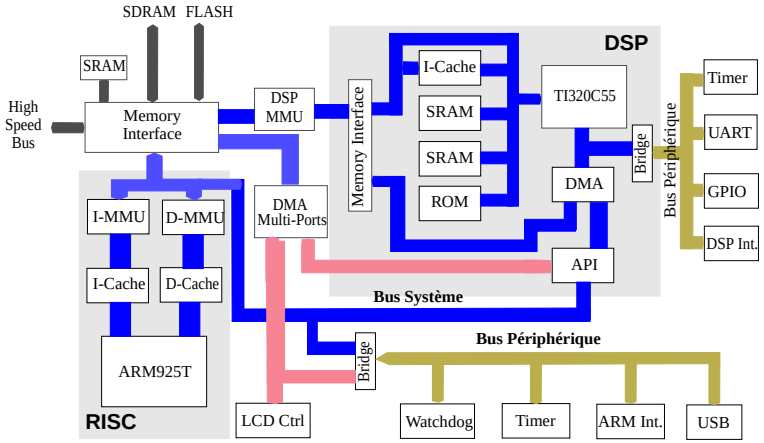
Conclusion

Exemple de Plateforme : OMAP de Texas Instruments





TI-OMAP1510



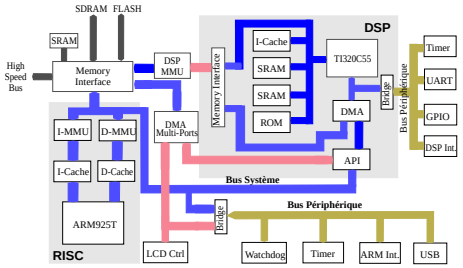
DSP vs RISC

Exemples de comparaison RISC/DSP (Texas Instruments)

Applications	ARM9E	StrongARM 1100	TMS320C5510
Annulation Echos (16bits)	**24**	**39**	**4**
Annulation Echos (32 bits)	**37**	**41**	**15**
décodage MPEG4/H263 QCIF 15 fps	**33**	**34**	**17**
codage MPEG4/H263 QCIF 15 fps	**179**	**153**	**41**
décodage JPEG QCIF	**2.1**	**2.06**	**1.2**
décodage MP3	**19**	**20**	**17**

Millions cycles/S

Caractéristiques de l'OMAP1510



ARM 925 :

- *Cache Instr. 16Ko*
- *Cache Data 8 Ko*
- *Write Buffer*
- *MMU Instr*
- *MMU Data*
- *TLB (mémoire virtuelle)*

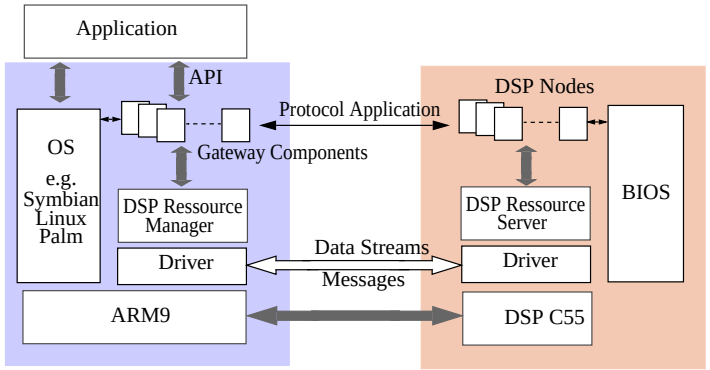
C55 :

- *Mémoire interne*
  - ***SARAM 48Kwords***
  - ***DARAM 32 Kwords***
  - ***PDROM 16Kwords***
- *Cache Instr. 24Ko*
- *Accélérateur graphique*
- *Interface avec ARM*

Communications

- *DMA (Direct Memory Access)*
- *SRAM interne (mailbox)*
- *Bus système*
- *Bus périphérique*

Architecture Logicielle de l'OMAP : compatibilité entre les circuits

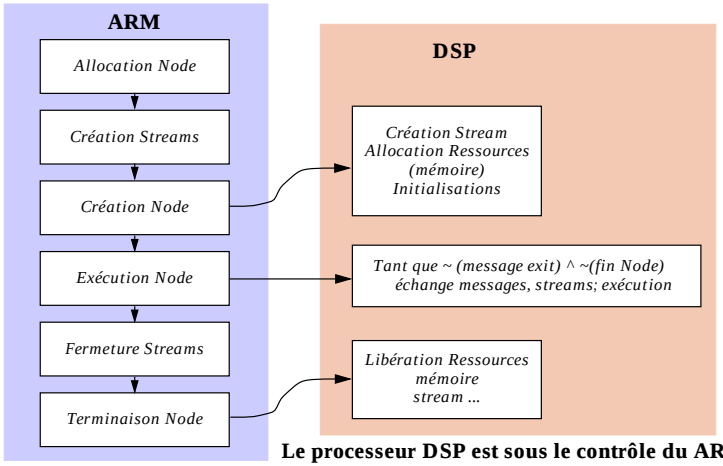


DSP Node : descripteur de tâche DSP

Messages : contrôles l'exécution du DSP

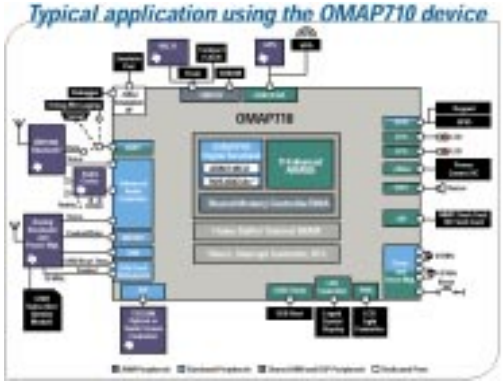
Data Stream : communications de données

Schéma d'exécuton parallèle



Le processeur DSP est sous le contrôle du ARM

Architecture de l'OMAP710



Un ARM7TDMI est couplé au DSP : Applications 2G et 2.5G avec PDA

Conclusion

Plateforme : approche pragmatique des industriels

- *But : contourner les faiblesses des méthodes de conception*
- *"Standardisation" des composants architecturaux (Hw & protocoles)*
  Limiter la conception manuelle de fonctions spécifiques

Les contraintes militent pour conserver des éléments spécifiques

- *Nouveaux standards, Compromis performances/coûts*
  Temps - Consommation - Surface - Vérification - Test - Fabrication ...

La conception de SoC est un vaste problème d'optimisation multi-critères

- *Des solutions partielles peuvent exister (un MoC) - Reste insuffisant*
- *Modélisation, WCET, Compilation, Exploration, Architecture, Debugging, ...*

Que de travaux de recherche en perspective !!!