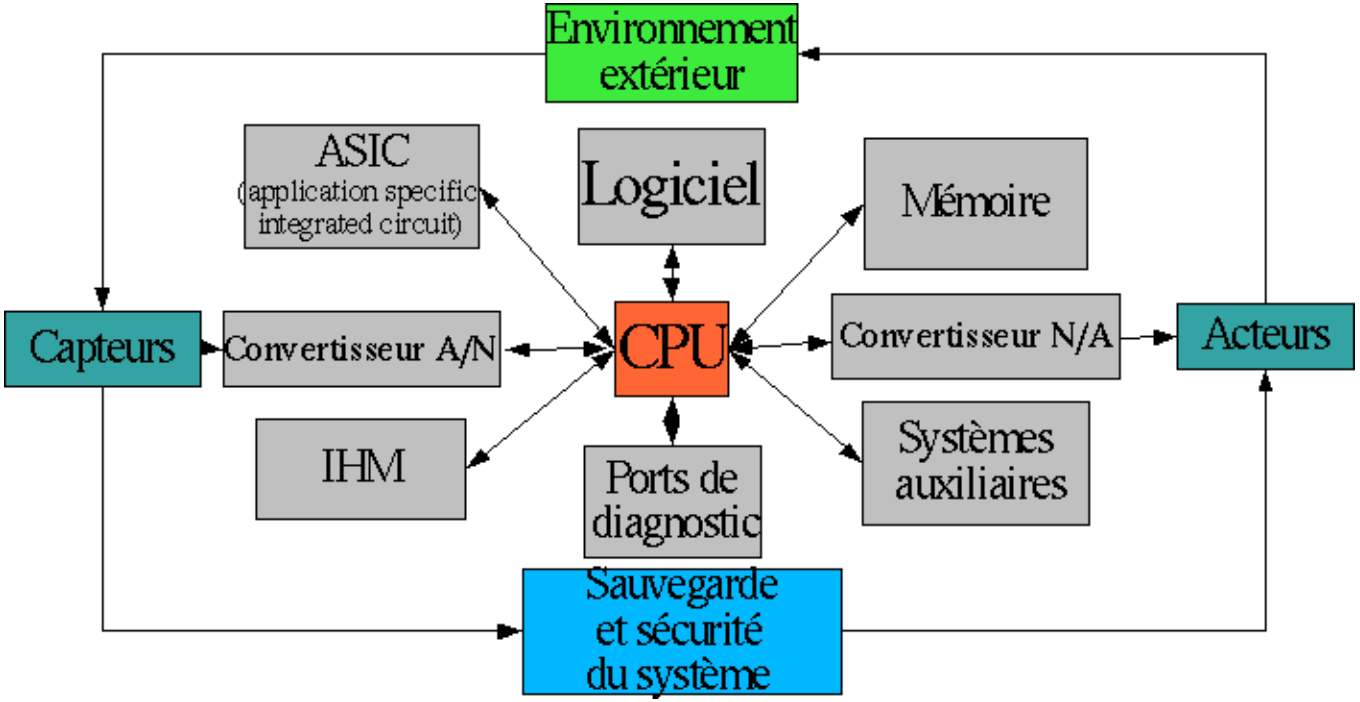
Méthodologie de conception SoC

Architecture d'un système embarqué (1)



Le fonctionnement du système se résume ainsi:

- Il reçoit des informations de l'environnement extérieur qu'il converti en signal numérique
- L'unité de traitement composée du CPU, de la mémoire, du logiciel, de l'ASIC et éventuellement de système externes traite l'information
- Le traitement génère éventuellement une sortie qui est envoyée vers la sortie, les systèmes auxiliaire, les ports de monitoring ou l'IHM

Architecture d'un système embarqué (2)

Selon la société ARM computer:

Un système: *« Something so complex that it doesn't work first time »*
«Quelque chose de si complexe que ça ne marche pas du premier coup »

Une architecture: « the sum of partitioning and implementation decisions taken in the design phase… (including Conventions, Reuse, Economics, History and Experience)… which together… Implement a Serviceable Solution »

« *L'ensemble des décisions de partitionnement et de mise en œuvre prises lors de la phase de conception… (y compris les conventions, la réutilisation, l'économie, l'histoire et l'expérience)… qui, ensemble… mettent en œuvre une solution réparable* »

Architecture d’un système embarqué (3)

Une architecture n’est pas seulement une finalité structurelle et fonctionnelle, de façon à garantir le respect des contraintes fonctionnelles, économiques et physiques,

Elle nécessite préalablement des études:

- d’adaptation à l’algorithme
- de partitionnement
- de méthodologie de conception.

Architecture d'un système embarqué (4)

Logicielles (SW): processeur + logiciel

- ✓ Flexibilité
- ✓ Faible temps de conception

Matérielles (HW): ASIC

- ✓ Performance
- ✓ Consommation
- ✓ Protection industrielle

Mixtes

- ✓ Tire profit des 3 approches ➔ cas des SoC

Nécessité d'un nouveau cycle de développement

✓ L'implantation matérielle permet d'obtenir des performances dynamiques plus élevées qu'une implantation logicielle mais à un coût plus élevé..

✓ L'avènement de la technologie VLSI et des premiers microprocesseurs,

Les équipes de conception recherchent le meilleur partitionnement matériel/logiciel permettant de respecter les contraintes de coûts et de performances imposées.

A l'époque où les possibilités d'implantation matérielle et logicielle étaient limitées, l'expérience seule des concepteurs pouvait suffire à les guider vers une solution proche de l'optimum.

Historique de conception:

Les progrès réalisés dans:

- Le domaine du génie logiciel (environnement de programmation, langages de haut niveau, méthodes et langages orientés objet)

- Aussi bien, dans le domaine du génie matériel (technologie VLSI, outils de synthèse logique, outils de synthèse de haut niveau) permettent de réaliser des systèmes de plus en plus complexes.

Jusqu'à encore peu de temps, la réalisation des systèmes électroniques embarqués de grande complexité reposait sur une approche d'*ingénierie système caractérisée par une séparation nette de la conception et* réalisation des parties logicielles et matérielles.

Les ingénieurs système avaient en effet la responsabilité de concevoir l'architecture du système et d'identifier et spécifier les parties matérielles et logicielles dont les réalisations étaient ensuite à la charge d'équipes de conception distinctes ou étaient sous-traitées.

Cette approche a entraîné un certain cloisonnement et une absence de dialogue entre les concepteurs des deux catégories professionnelles de culture différente: « les concepteurs d'architectures matérielles et les informaticiens ».

⇒ Conduit ensuite à des difficultés importantes découvertes tardivement durant l'intégration du logiciel sur le matériel.

Malgré la malléabilité du logiciel, les problèmes rencontrés lors de l'intégration se traduisent généralement par un surcoût financier et temporel important.

L'approche d'ingénierie système est indispensable pour la conception des systèmes hétérogènes complexes et/ou caractérisés par une sous-traitance importante (avionique, automobile par exemple).

Mais l'utilisation d'un tel cycle de développement pour les systèmes électroniques embarqués s'est révélé très pénalisante et inadaptée pour répondre à des contraintes de qualité, de coûts et délais de plus en plus sévères.

La méthodologie « Adéquation Algorithme Architecture » (AAA)

Objectifs

- La complexité des applications visées, au niveau des algorithmes, de l'architecture matérielle, et des interactions avec l'environnement sous contraintes temps réel, nécessite **des méthodes pour minimiser la durée du cycle de développement,** depuis **la conception jusqu'à la mise au point.**

- Autant des prototypes que des « produits finis » dérivés de ces prototypes.

- La méthodologie AAA est basée sur un modèle unifié de graphes, autant pour spécifier l'Algorithme et l'Architecture multicomposant, que pour déduire les implantations possibles en termes de transformations de graphes.

- L'Adéquation est un problème d'optimisation qui consiste à choisir une implantation dont les performances, déduites des caractéristiques des composants, respectent les contraintes temps réel et d'embarquabilité.

La méthodologie « Adéquation Algorithme Architecture » (AAA)

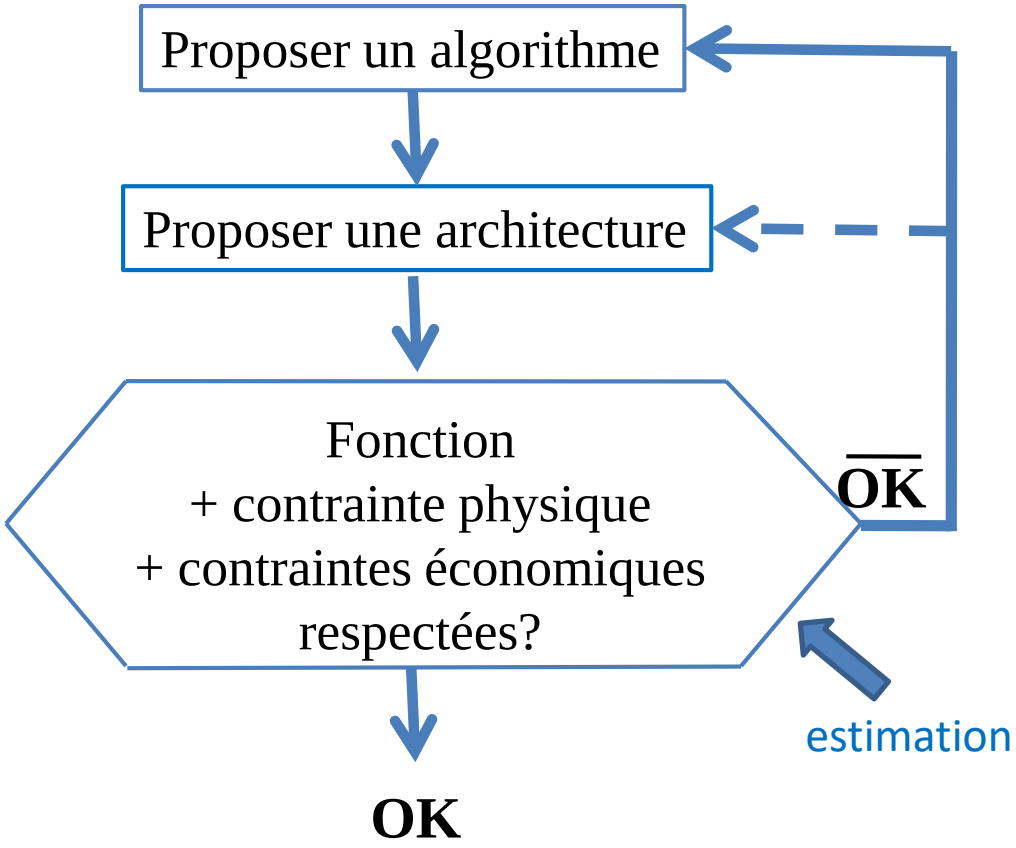
Objectifs

- L'intérêt principal de ce modèle réside:

  - dans sa capacité à exprimer tout le parallélisme, concrètement décrit sous la forme de schémas-blocs, non seulement dans le cas de l'algorithme (graphe flot de données : exemple Simulink)
  - et de l'architecture (interconnexion de composants : exemple VHDL structurel), mais aussi dans le cas de l'implantation de l'algorithme sur l'architecture (distribution et ordonnancement des calculs et des communications).

- Il permet d'effectuer des optimisations précises prenant en compte la conception conjointe logiciel/matériel (co-design) et de simplifier la génération de code (exécutifs les plus statiques possibles et/ou ``netlists'').

Méthode d' « Adéquation Algorithme Architecture » (Faisabilité architecturale)

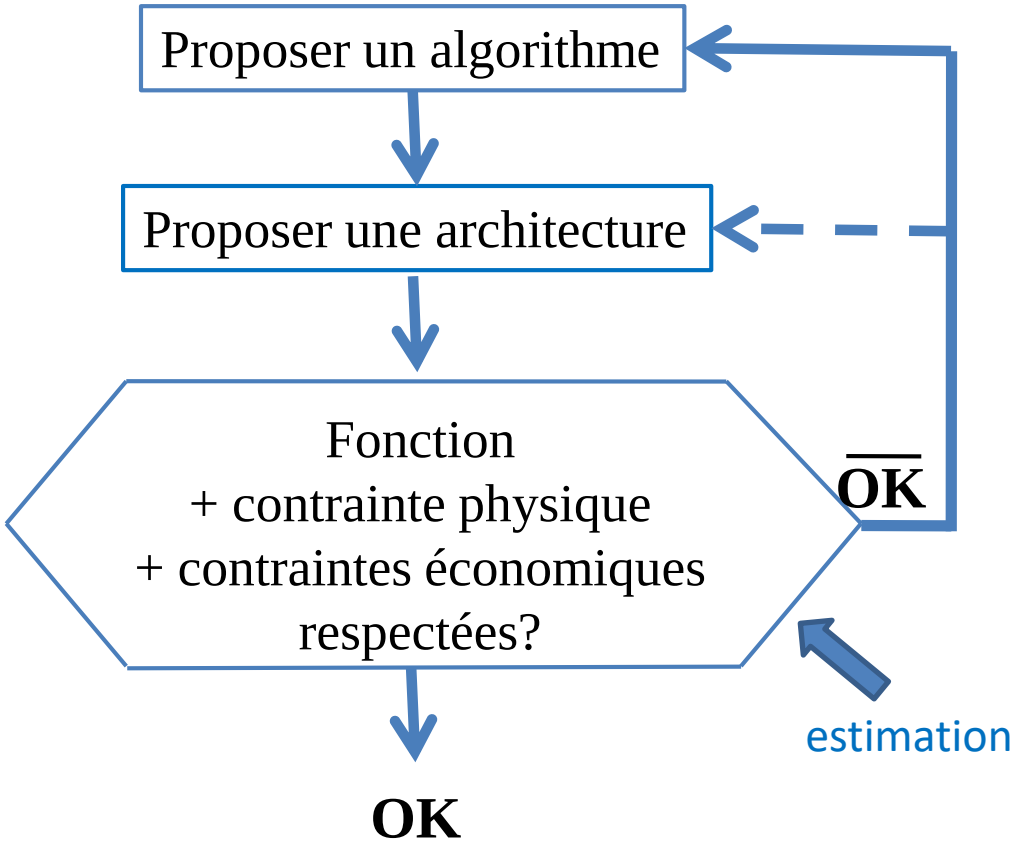


Adéquation Algorithme Architecture

Contraintes physiques

- Puissance de calcul
- Capacité de mémoire
- Débits mémoire
- Débits d'E/S
- Consommation
- Fréquence d'horloge
- Taille totale
- …

Faisabilité architecturale



Adéquation
Algorithme
Architecture

Contraintes physiques

- Puissance de calcul
- Capacité de mémoire
- Débits mémoire
- Débits d'E/S
- Consommation
- Fréquence d'horloge
- Taille totale
- …

Adéquation Algorithme Architecture consiste à étudier en même temps les aspects algorithmiques et architecturaux en prenant en compte leurs interactions, en vue d'effectuer une implantation optimisée de l'algorithme (minimisation des composants logiciels et matériels) tout en réduisant les temps de développement et les coûts finaux de l'application étudiée.

Modèle d'algorithme

Un algorithme tel que défini est une séquence (ordre total) finie d'opérations directement exécutable par une machine à nombre d'états fini.

Modèle d'architecture

Les modèles les plus classiquement utilisés pour spécifier des architectures parallèles ou distribuées sont les PRAM « Parallel Random Access Machines » et les DRAM « Distributed Random Access Machines ».

- Le premier modèle correspond à un ensemble de processeurs communiquant par mémoire partagée
- le second correspond à un ensemble de processeurs à mémoire distribuée communiquant par passage de messages.

Modèle d'implantation

Une implantation d'un algorithme sur une architecture est une distribution et un ordonnancement non seulement des opérations de l'algorithme sur les opérateurs de calcul de l'architecture, mais aussi des opérations de communication, qui découlent de la première distribution, sur les opérateurs de communication.

Tout d'abord, pour faire face aux problèmes de l'intégration tardive des parties matérielles et logicielles

- Les concepteurs ont exploité la *simulation conjointe de la partie logicielle et la* partie matérielle du système (co-simulation).

Ils se sont très vite aperçus que leurs besoins allaient bien au delà de la co-simulation et que pour réduire la durée des développements et accroître la complexité des systèmes et la qualité de conception

- ✓ Il fallait utiliser une *méthodologie de conception complète et les outils supports* associés permettant de développer conjointement la partie matérielle et logicielle tout au long du cycle de développement.

Historique et besoin

Le développement des systèmes électroniques composés d'une partie matérielle et d'une partie logicielle n'est pas un problème nouveau!!!

Concevoir et réaliser de tels systèmes nécessite une compétence technique dans au moins 3 domaines:

- ❖ l'électronique analogique,
- ❖ l'électronique numérique
- ❖ et l'informatique.

La nature spécifique du traitement à effectuer et le couplage du système avec son environnement nécessitent aussi des compétences complémentaires:

- ❖ traitement de l'information (signal, image, parole...),
- ❖ électronique de puissance lorsque l'environnement utilise des courants forts,
- ❖ réseaux et télécommunications, etc.

L’activité de CO-Design

SOC

La complexité croissante des systèmes pour lesquels la réalisation résulte de l’association d’une partie matérielle et d’une partie logicielle

La diversité des choix technologiques et les contraintes de coûts et délais de plus en plus sévères

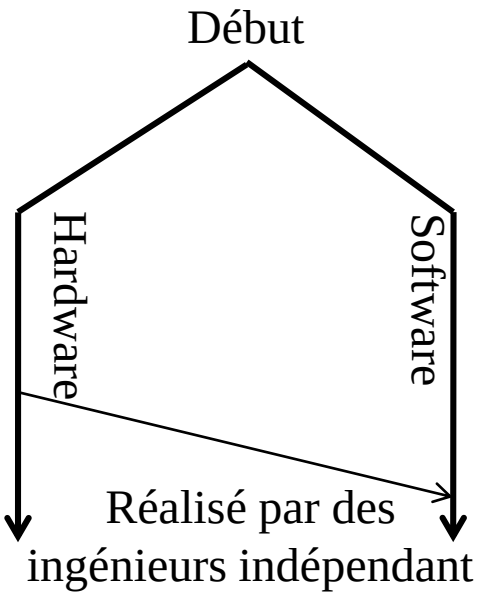
Nécessitent l’utilisation de nouvelles méthodologies et outils logiciels associés pour diminuer leur durée de conception et accroître leur qualité.

Le co-design est l’une de ces méthodologies.

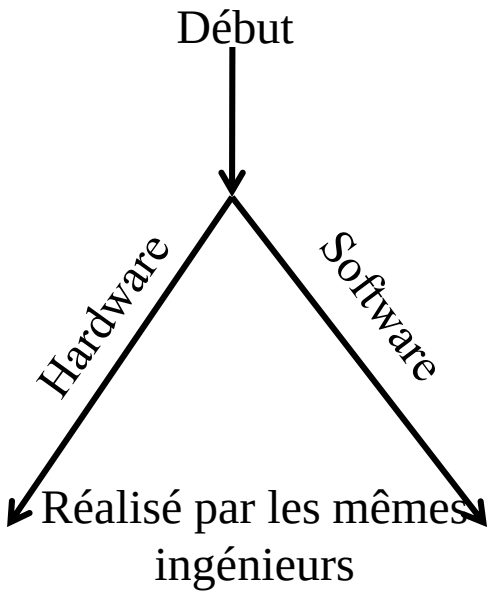
Les outils de co-conception et de co-simulation sont apparus avec le début de l'automatisation de la conception des systèmes mixtes logiciels/materiels.

Conception traditionnelle et Codesign

Conception traditionnelle



Codesign (flot concurent)



Décrire l'activité de conception conjointe matériel/logiciel appelée usuellement **co-design**

Son domaine d'application et les raisons de l'émergence de ce nouveau cycle de développement.

Méthodologie de co-design:

Enrichissement d'une méthodologie de conception système existante.

Une méthodologie de conception système est habituellement organisée selon quatre étapes principales:

- l'élaboration des spécifications,
- la conception fonctionnelle ou préliminaire,
- la conception architecturale ou détaillée
- et la réalisation.

L'enrichissement d'une méthodologie système pour l'activité de co-design concerne principalement l'étape de conception architecturale.

Hardware / Software Codesign (HSCD)

1. Hardware / Software Codesign (HSCD) fait partie intégrante du flot de conception de niveau de système électronique (ESL) modernes.

2. HSCD (Hardware / Software Codesign) désigne des méthodologies de conception pour les systèmes électroniques qui exploitent les compromis entre le matériel (HW) et le logiciel (SW).

- Les aspects importants de la conception matérielle / logicielle?
- L'évolution historique des techniques du flot et résumera ensuite les principales étapes du Codesign?

Les étapes dans le Codesign

- ✓ **Spécifications:** listes des fonctionnalités du système de façon abstraite.
- ✓ **Modélisation:** conceptualisation et affinement des spécifications produisant un modèle du matériel et du logiciel.
- ✓ **Partitionnement:** partage logiciel matériel.
- ✓ **Synthèse et optimisation:** synthèse matérielle et compilation logicielle.
- ✓ **Validation:** co-simulation.
- ✓ **Intégration:** rassemblement des différentes modules.
- ✓ **Test d'intégration:** vérification du fonctionnement.

Les spécifications

- Les spécifications fonctionnelles répondent aux points précis du Cahier des Charges. Ils ne sont pas négociables en général, c'est des besoins primaires du client.
- Les spécifications non fonctionnelles représentent tout ce qui dans le cahier des Charges et n'est pas une exigence fonctionnelle.
  - ✓ Il s'agit habituellement d'un ensemble de contraintes d'intégration avec l'environnement (taille, puissance consommée), de performances (temps de réponse, débit), d'ordre économique (coût, délai de fabrication), de qualité (durée de vie), de sûreté de fonctionnement, etc.

Le modèle

Un modèle est une représentation formelle d'un système à un niveau d'abstraction donné.

N.B. Il est bon de noter qu'une méthode efficace doit reposer sur un ensemble de concepts de modélisation restreint mais suffisant pour décrire n'importe quel système.

Rq. Les outils ESDA se composent d'outils de capture de modèles, de simulation, d'analyse statique ou dynamique, de recherche de compromis, de synthèse et de co-vérification.

Le partitionnement

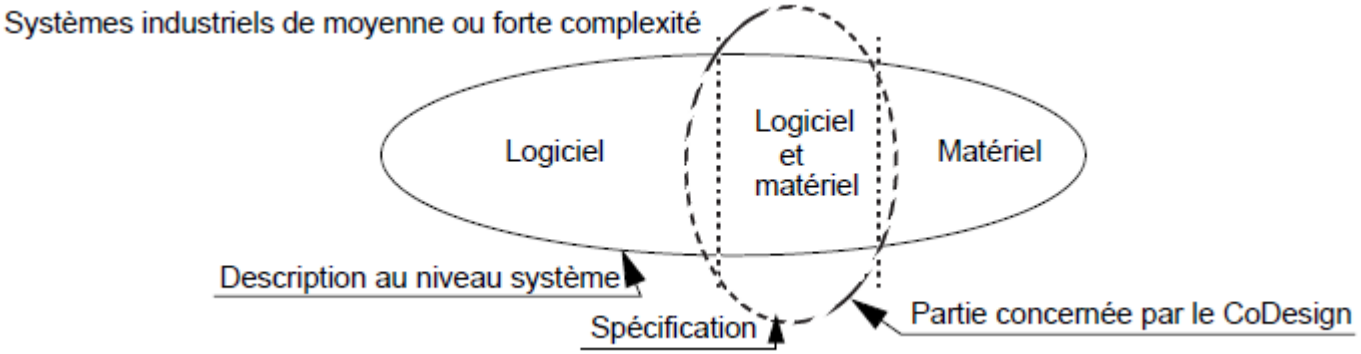
A partir de ces spécifications, le concepteur doit définir :

- l'architecture matérielle constituée de processeurs matériels (ASICs, FPGA, composants standards) et de processeurs logiciels (MPU, ASIP, DSP),
- l'allocation des éléments ,
- de la solution fonctionnelle sur cette architecture: c'est la problématique communément appelée *partitionnement matériel/logiciel.*

Partitionnement et co-design

Le terme « Hardware/Software Concurrent Design » souvent abrégé par «Hw/Sw Codesign » et qui se traduit par **conception conjointe matériel et logiciel .**

- Exemple de sous-système concerné par le co-design.



- ✓ La description fonctionnelle détaillée de la partie spécifique ou sous-système sert de point de départ de l'activité de co-design.
- ✓ Ce processus doit permettre aux concepteurs de transformer correctement du premier coup les spécifications d'un système en un produit industriel comportant une partie logicielle et une partie matérielle et satisfaisant les contraintes fonctionnelles et non fonctionnelles de son cahier des charges.
- ✓ Il doit également permettre d'accroître la qualité de conception et de réduire le temps de développement.

De quoi bénéficie les concepteurs:

Le concepteur bénéficie d'une grande diversité de choix pour la réalisation d'un système:

- ✓ L'offre en matière de microprocesseurs (MPU, ASSP, ASIP, DSP),
- ✓ de langages de programmation et compilateurs associés est vaste.
- ✓ Parallèlement, on peut aussi implanter de plus en plus de fonctions dans le silicium (ASICs, FPGA, system-on-a-chip).

Même pour les petits systèmes, cette diversité rend de plus en plus difficile la conception intuitive basée sur l'expérience du concepteur.

Techniques de partitionnement matériel/logiciel

Différentes techniques de partitionnement *matériel/logiciel «l'étape de conception architecturale »:*

- Dans le cas général, l'architecture matérielle peut être quelconque, c'est-à-dire hétérogène et distribuée. Plus l'architecture est particularisée, plus les concepts et les outils associés sont actuellement avancés.
  - **Souvent, l'architecture générique cible est choisie.**

- Dans un ensemble de travaux, on retrouve très souvent l'emploi d'une architecture du type maître/esclave utilisant un microprocesseur conventionnel ou un microcontrôleur comme maître associé à un ou plusieurs ASICs comme esclaves. Il résulte que la solution est fortement dépendante de l'architecture et des composants standards utilisés pour la réalisation.

La co-synthèse

La co-synthèse (étape de réalisation) qui se charge de transformer les descriptions fonctionnelles en descriptions directement implantables sur les processeurs matériels et logiciels de l'architecture cible.

✓ Considérés séparément:

- Le développement des constituants logiciels (debuggeur symbolique, générateur d'IHM, profiler, approche objet, exécutif temps-réel)..
- Le développement des constituants logiciels matériels (synthèse logique et haut niveau) sont aujourd'hui bien maîtrisés.

N.B.

- Les outils de synthèse logique sont très robustes.
- Les outils de synthèse comportementale même s'ils manquent encore un peu de maturité, sont aussi maîtrisés par les compagnies EDA/ESDA (Electronic System Design Automation).
- La synthèse logicielle souffre cependant de deux problèmes:
  - Le manque d'outil de compilation efficace pour des microprocesseurs spécifiques,
  - Le manque d'outil de génération de code exécutable pour une description multi-tâches (ordonnancement des tâches compris).

Remarques:

- La plupart des techniques de partitionnement automatique se limitent à une architecture matérielle constituée d'un microprocesseur, d'un ensemble de composants matériels programmables et éventuellement d'une mémoire commune.
  - ✓ Cette architecture est intéressante pour son aspect générique mais elle correspond de moins en moins à la réalité industrielle.
- La communauté du co-design a donc pris peu à peu conscience que le partitionnement automatique plus ou moins rejeté par le milieu industriel n'est efficace que sur une classe limitée de systèmes et que les recherches doivent s'orienter vers des solutions plutôt de nature semi-automatique guidées par le concepteur et basées sur des outils d'estimation rapides.

La validation

- La validation du système embarqué fait référence aux outils et techniques utilisés pour valider que le système satisfait ou dépasse les exigences.
- La validation vise à confirmer que les exigences dans des domaines tels que la fonctionnalité, les performances et la puissance sont satisfaites.

- Il répond à la question : « **Avons-nous construit la bonne chose ? »**
  - ✓ La validation confirme que l'architecture est correcte et que le système fonctionne de manière optimale.

La co-simulation

- La co-simulation est une simulation de la description mixte matériel-logiciel résultant d'un partitionnement

"Hardware-software cosimulation is a means of verifying the functionality of mixed hardware-software descriptions" [THOMAS-93].

- La difficulté de la co-simulation est liée à la différence de modèles et/ou de niveaux d'abstraction des représentations du logiciel et du matériel.
- Les solutions de co-simulation reposent sur l'emploi d'un langage de représentation unique et exécutable ou pour une simulation hétérogène sur l'utilisation d'un mécanisme de communications entre différents simulateurs.

Pour réduire la durée des développements et accroître la complexité des systèmes et la qualité de conception, Il est très vite aperçu que leurs besoins allaient bien au delà de la co-simulation:

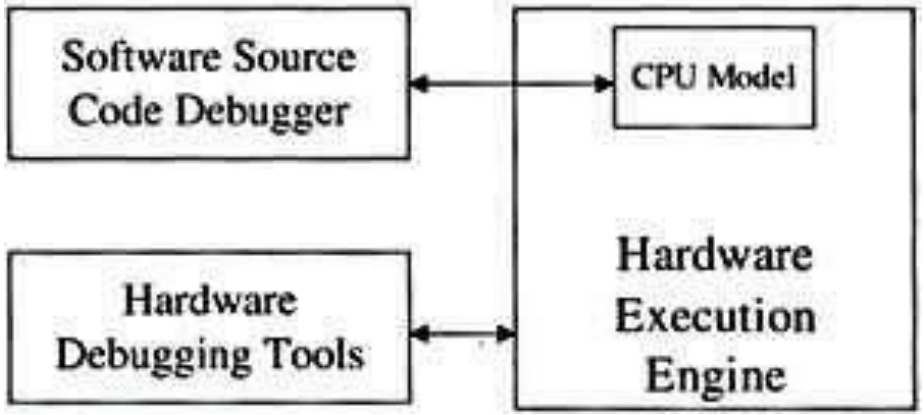
- Il fallait utiliser une ***méthodologie de conception complète et les outils supports*** associés permettant de développer conjointement la partie matérielle et logicielle tout au long du cycle de développement.

Intégration matérielle et logicielle

- L'étape la plus cruciale dans la conception de systèmes embarqués est l'intégration du matériel et des logiciels.
  - ✓ Quelque part au cours du projet, le logiciel nouvellement codé rencontre le matériel nouvellement conçu.
  - ✓ Comment et quand le matériel et le logiciel se rencontreront pour la première fois pour résoudre les bogues doivent être décidés au début du projet.
- Il existe de nombreuses façons de réaliser cette intégration:
  - ✓ Il est préférable de le faire plus tôt que plus tard, même si cela doit être fait intelligemment pour éviter de perdre du temps à déboguer un bon logiciel sur du matériel défectueux ou à déboguer un bon matériel exécutant un logiciel défectueux.
- Deux concepts importants d'intégration de matériel et de logiciel sont:
  - ✓ La vérification et la validation.
  - ✓ Ce sont les étapes finales pour s'assurer qu'un système de travail répond aux exigences de conception.

La Co-vérification

- Au niveau le plus basique, la co-vérification HW/SW signifie vérifier que le logiciel du système embarqué s'exécute correctement sur le matériel du système embarqué. Cela signifie exécuter le logiciel sur le matériel pour s'assurer qu'il n'y a pas de bogues matériels avant que la conception ne soit engagée dans la fabrication.
- Cela signifie que pour qu'une technique soit considérée comme un produit de co-vérification, elle doit fournir au moins un débogage logiciel à l'aide d'un débogueur de code source et un débogage matériel à l'aide de formes d'onde, comme illustré à la figure:



La Co-vérification

- La co-vérification HW/SW est le processus de vérification du bon fonctionnement du logiciel du système embarqué sur la conception matérielle avant que la conception ne soit validée pour la fabrication.
- La co-vérification est souvent appelée prototypage virtuel car la simulation de la conception matérielle se comporte comme le matériel réel, mais est souvent exécutée comme un programme logiciel sur un poste de travail.
- La co-vérification offre deux avantages principaux. Il permet aux logiciels qui dépendent du matériel d'être testés et débogués avant qu'un prototype ne soit disponible. Il fournit également un stimulus de test supplémentaire pour la conception du matériel.

La Co-vérification

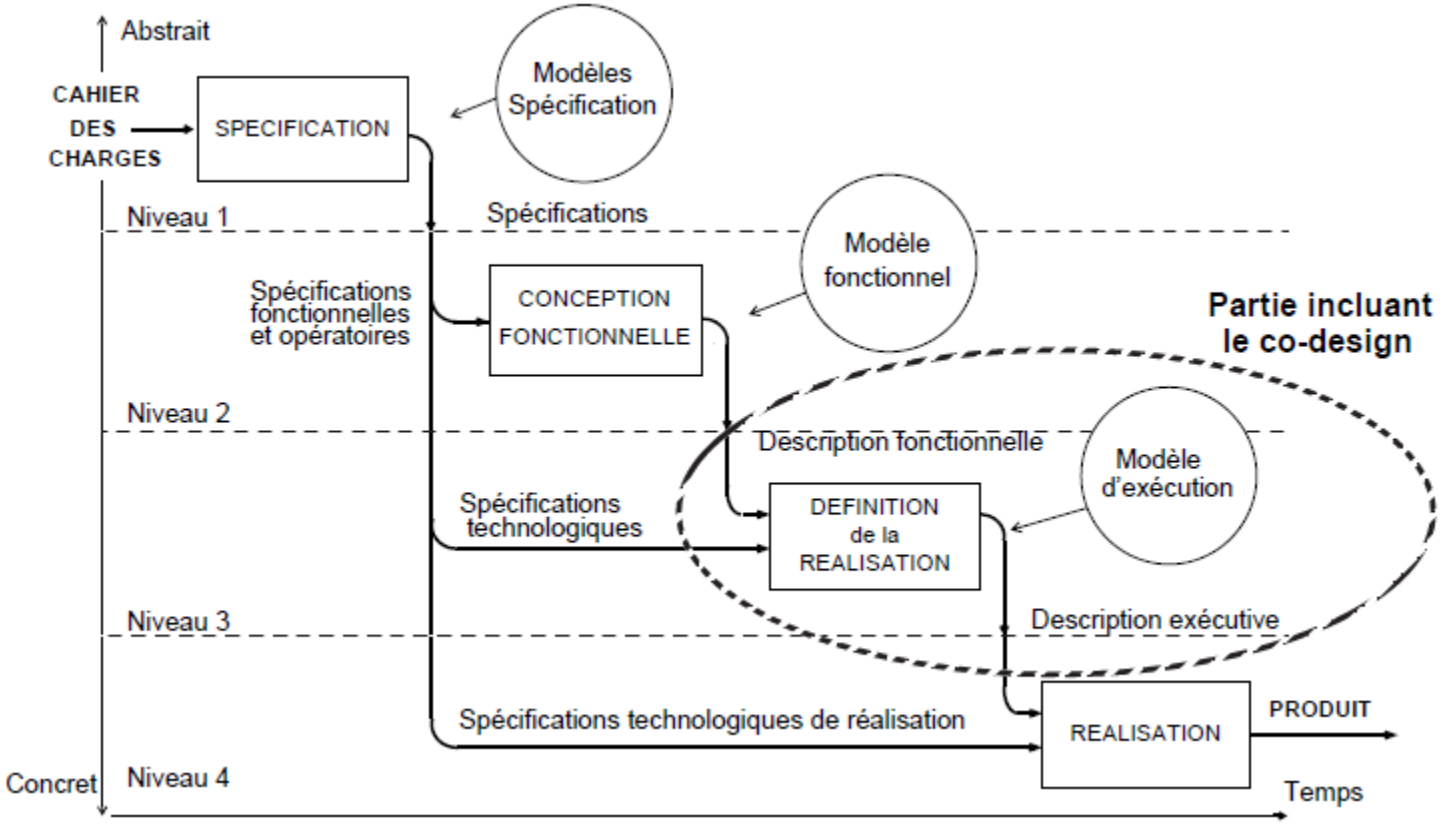
- Les avantages de la co-vérification résolvent le problème d'intégration matérielle et logicielle et se traduisent par un calendrier de projet plus court, un projet à moindre coût et un produit de meilleure qualité.
- Les principaux avantages de la co-vérification sont les suivants :

  1) Accès anticipé à la conception matérielle pour les ingénieurs logiciels

  2) Stimulation supplémentaire pour les ingénieurs en matériel

Les quatre étapes de développement, selon MCSE:

➢ l'étape de Spécification qui a pour objectif d'élaborer une description externe la plus complète possible du système à concevoir, et ceci à partir du cahier des charges.



Démarche de développement avec MCSE.

Le partitionnement matériel/logiciel

❑ Le partitionnement matériel/logiciel assure la transformation des spécifications de la partie du système relevant de l'activité co-design en une architecture composée d'une partie matérielle et d'une partie logicielle.

❑ Les spécifications considérées sont en réalité une description fonctionnelle détaillée résultant d'une approche système.

Cette transformation s'effectue habituellement en deux phases:

- ✓ la sélection d'une architecture matérielle et
- ✓ l'allocation des éléments (fonctions et éléments de relations) du modèle fonctionnel sur les éléments de cette architecture.

Plusieurs critères peuvent intervenir sur le double choix (architecture et allocation) d'un partitionnement tels que par exemple:

- **les performances statiques** (consommation, surface de silicium, coûts, taille du code, taille de la mémoire, etc) et dynamiques (contraintes de temps, débit, temps de latence, taux d'occupation, etc). Elles influent surtout sur l'allocation des fonctions.
- **la sécurité:** La prise en compte de la sûreté de fonctionnement peut induire des contraintes au partitionnement matériel/logiciel (redondance de composants matériels et de tâches logicielles par exemple),
- **la flexibilité:** l'implantation logicielle d'une fonction offre des possibilités d'évolution plus importantes qu'une implantation matérielle,
- **la réutilisation:** La réutilisation de composants est un facteur important de productivité, mais introduit des contraintes au niveau du partitionnement,
- **la testabilité:** l'extraction d'informations en temps-réel nécessite l'ajout de composants matériels supplémentaires (Bist, Boundary Scan) ou l'ajout d'instructions de capture.

Le partitionnement automatique

❖ Le problème du partitionnement est souvent présenté comme un problème NP complexe dépendant d'un grand nombre de paramètres.

❖ Pour résoudre ce problème, la plupart des méthodes automatiques réduit le nombre des paramètres (prise en compte d'un nombre limité de critères) et utilise une heuristique basée sur une fonction de coût pondérée par les critères retenus.

❖ Actuellement de nombreuses heuristiques de partitionnement ont été développées. Une synthèse des méthodes et algorithmes de partitionnement matériel/logiciel automatiques.

On peut citer en autres (liste non exhaustive):

➢ **L'algorithme Greedy** (Greedy algorithm): Toutes les fonctions sont initialement implantées en matériel et sont migrées vers le processeur logiciel en vérifiant les contraintes temporelles.

- ✓ La fonction de coût utilisée est pondérée par la surface du matériel, la taille du programme de code (mémoire) et le taux d'occupation du processeur.

➢ **L'approche de COSYMA**: Les fonctions sont au départ toutes implantées en logiciel puis migrées vers le matériel jusqu'au respect des contraintes de performances.

➢ **Les algorithmes utilisés dans Co-Saw** sont basés sur la construction progressive de groupes de fonctions pouvant partager la même ressource matérielle ou logicielle.

Les algorithmes cités précédemment dépendent fortement (coefficient de pondération de la fonction de coût) des caractéristiques de l'architecture cible choisie.

N.B

- Souvent l'architecture cible est composée d'un seul processeur logiciel couplé à un ou plusieurs FPGA ou ASICs et éventuellement une mémoire commune.
- Peu de techniques de partitionnement ciblent vers une architecture hétérogène composée d'un ensemble de processeurs logiciels (microprocesseur, DSP, ASIP) et matériels (FPGA, ASIC).
- L'architecture générique mono-processeur et constituée d'un ensemble de FPGA correspond peu à la réalité industrielle. Les composants programmables ont des performances plus faibles (surface de silicium occupée, fréquence maximale de fonctionnement) et un coût plus élevé (production en grande série) que les circuit non programmables.

- De plus, même les "System on a chip" ne sont pas mono-processeur car ils disposent de plus en plus souvent d’un coeur de DSP et d’un coeur de microcontroleur (MCU).

Ces techniques de partitionnement automatique ne sont donc intéressantes que pour le prototypage rapide sur une carte constituée d’un microprocesseur, d’un ensemble de FPGA et éventuellement une mémoire commune.