EN325: Conception Numérique Avancée

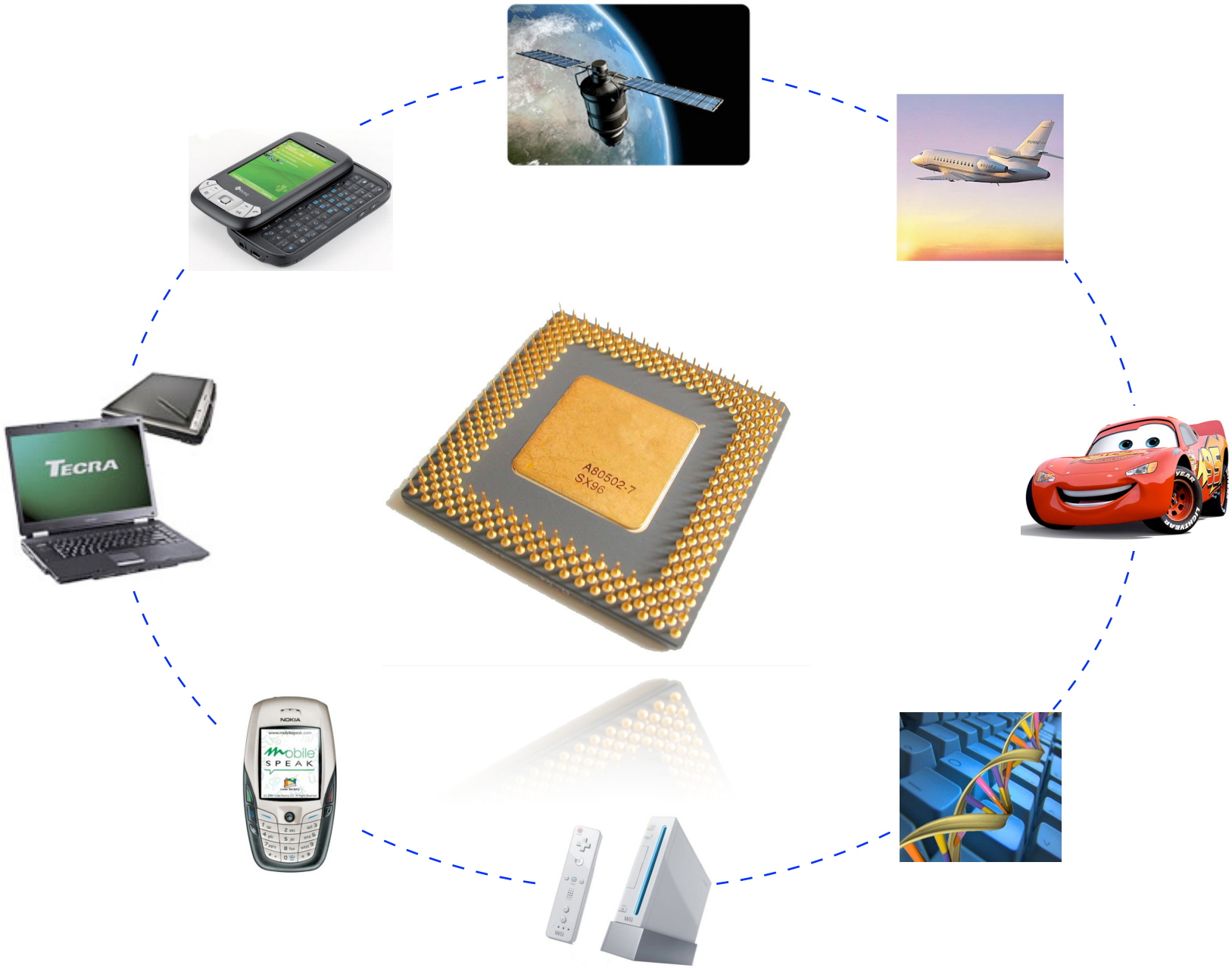
Bertrand LE GAL

bertrand.legal@ims-bordeaux.fr

Filière Electronique - 3ème année
ENSEIRB-MATMECA - Bordeaux INP
Talence, France

Evolution des Circuits Numériques

L'évolution du marché des systèmes numériques

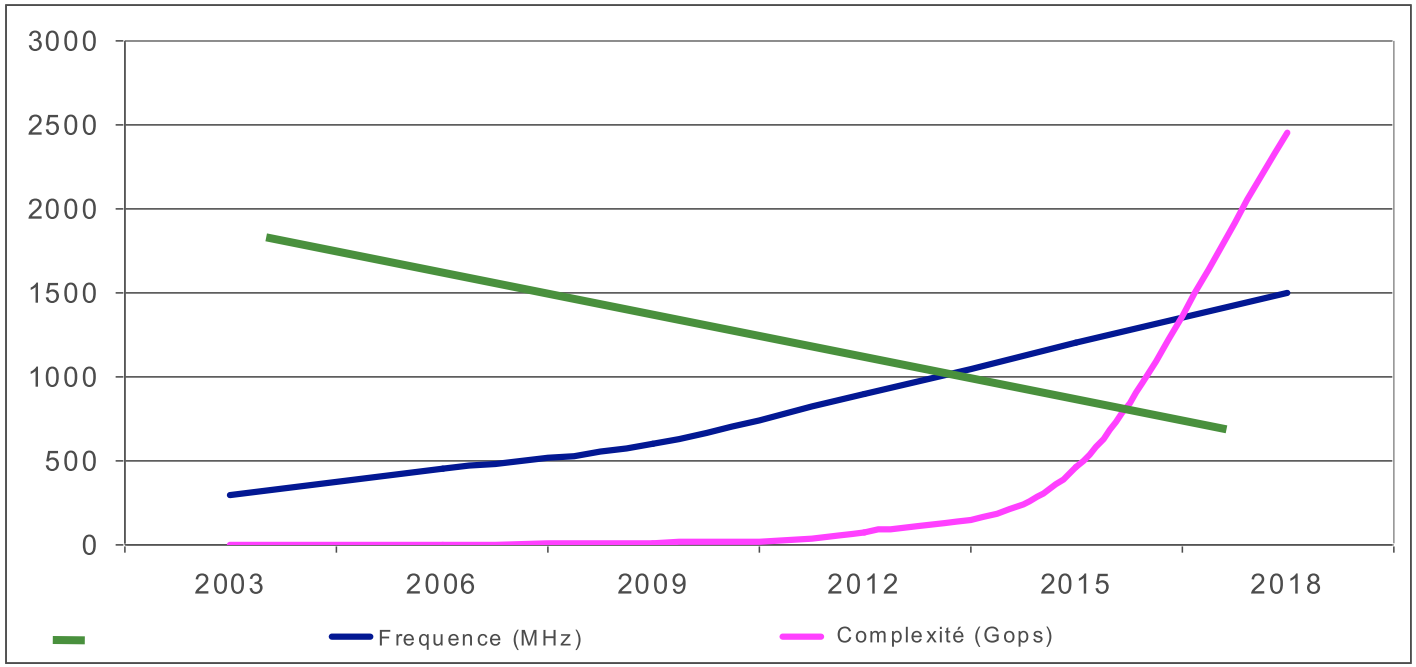


Evolution de la complexité des systèmes embarquées

=> Evolution des applications (TSI)

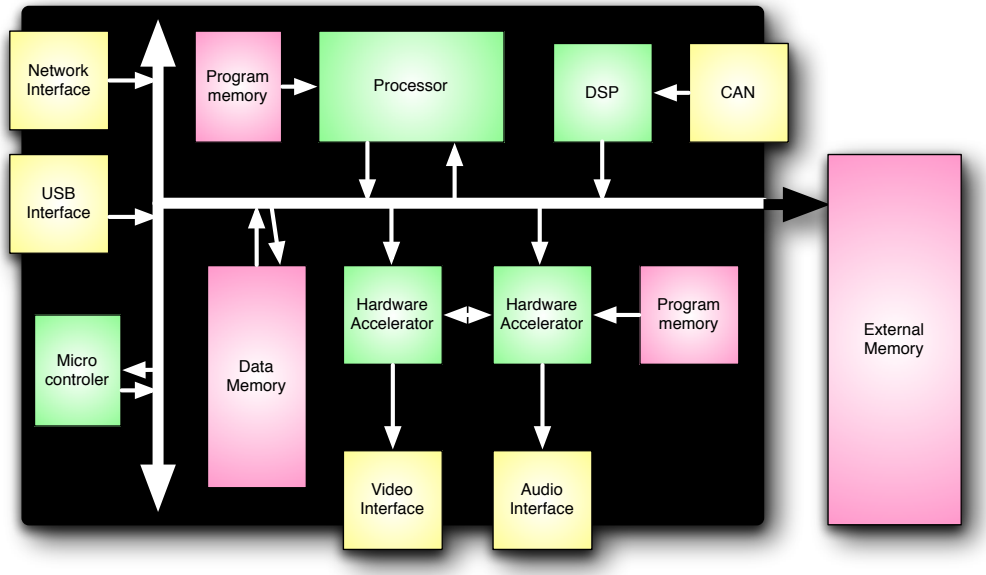
=> Complexification des systèmes électroniques

=> Réduction des délais accordés à la conception

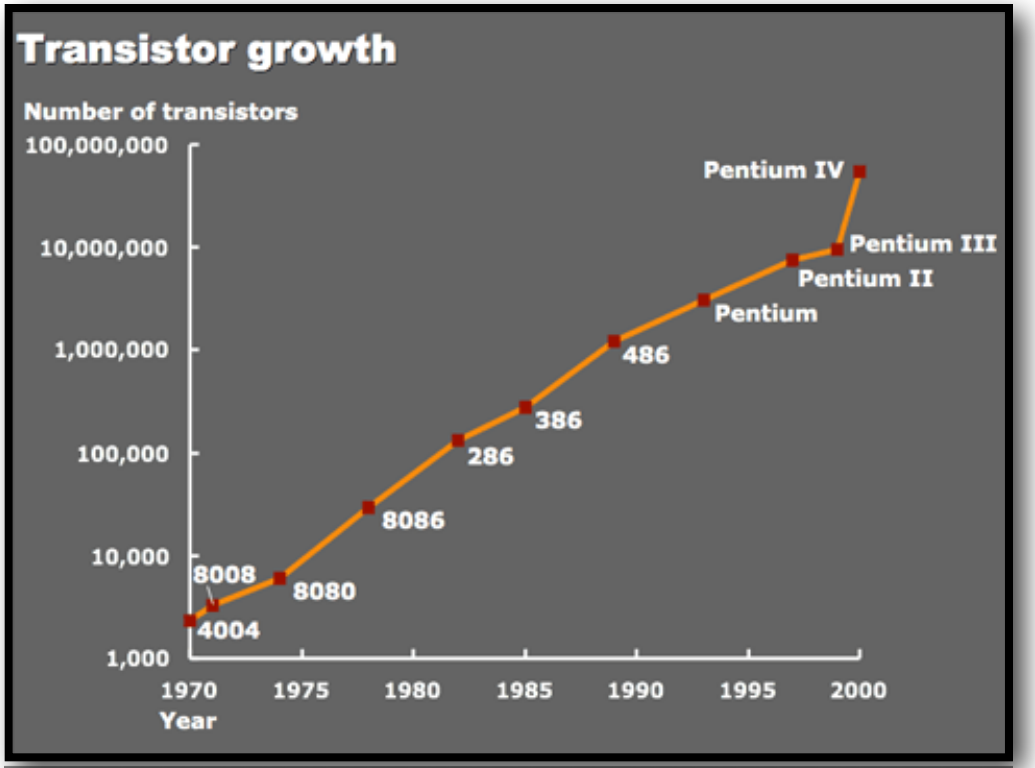


Evolution de la complexité des systèmes embarquées

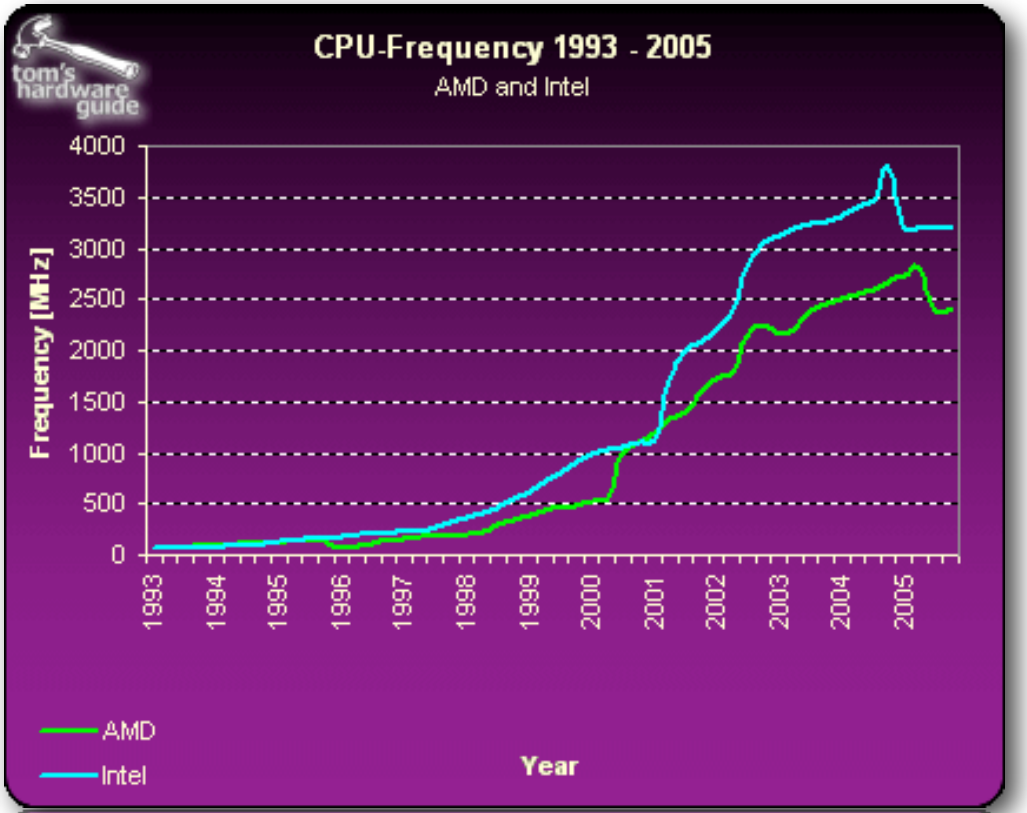
- Applications TSI,
  - Complexité grandissante
- Hétérogénéité :
  - HW/SW
  - Contrôle/Traitement
  - Numérique / Analogique
  - Technologie
- Contraintes :
  - Temps de conception
  - Surface (Coût $)
  - Consommation d'énergie,
  - Fiabilité, vieillissement.



Evolution des performances des processeurs généralistes



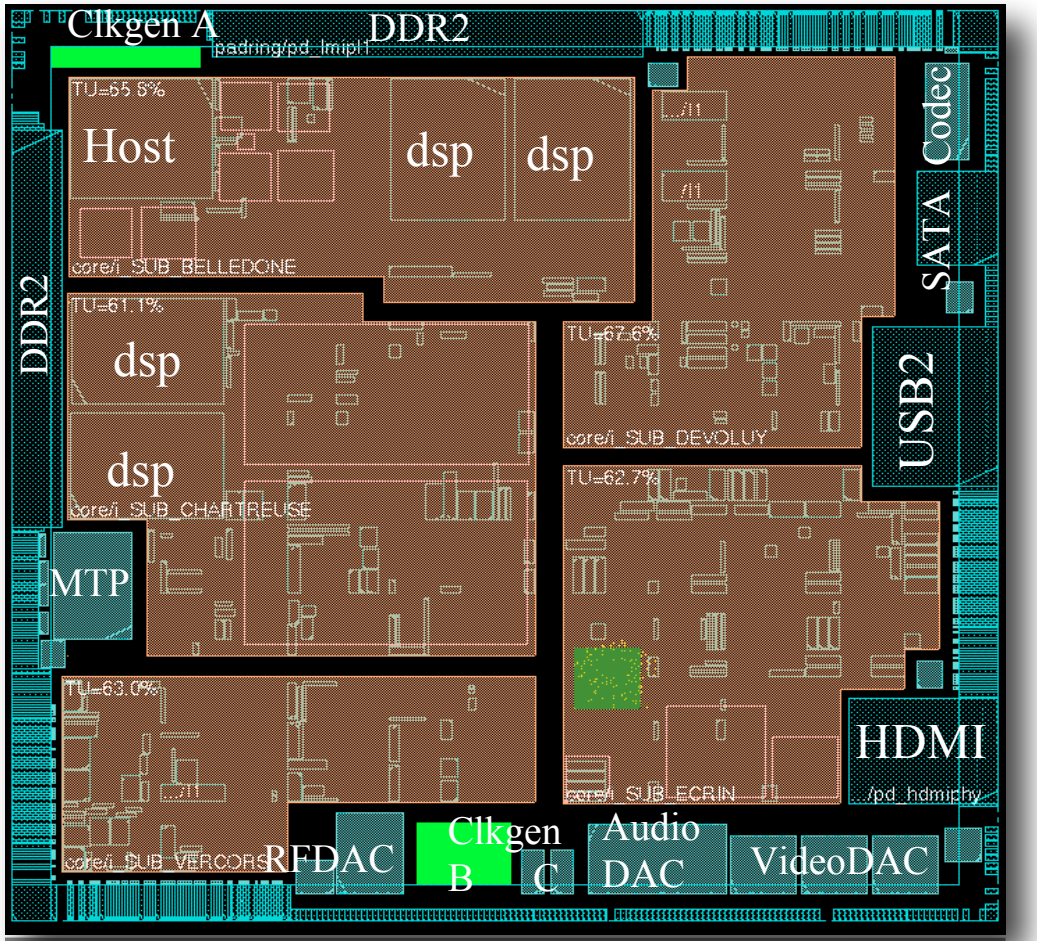
L'augmentation de l'intégration des transistors a suivi la loi de Moore



Du point de vue de la fréquence, une stabilisation est apparue ces dernières années aux alentours de 3 GHz (consommation et dissipation...)

Exemple, le décodeur “HD 264” de STMicro

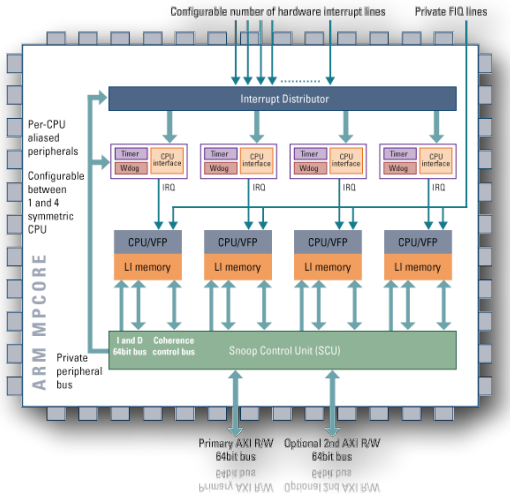
- Circuit dédié au décodage de la TV HD (norme H264)
  - Circuit contenant 150M transistors et 886 pads IOs (~5 GMIPS)
  - 128 sources d’intérruption
  - 73 initiateurs et 96 cibles sur les bus,
  - 115 réseaux d’horologe (19 pour les interconnexions),
- Construction du système
  - 4 processeurs (2 DSP pour la vidéo, 1 DSP pour l’audio et 1 généraliste pour la configuration),
  - 36 Soft IPs + 2 Hard IPs
  - 140 memory cuts

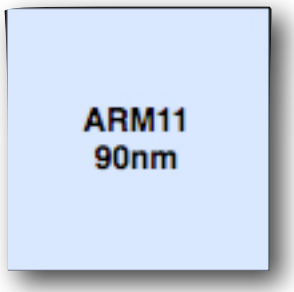


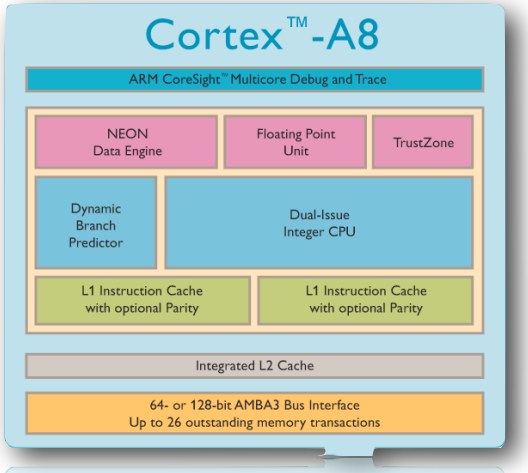
Circuit développé en 200X ?!

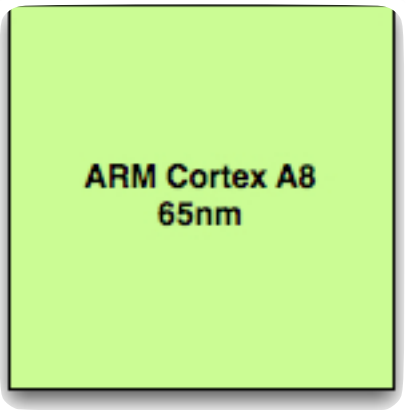
Que fait on alors aujourd'hui ...

Augmentation de la complexité des architectures matérielles

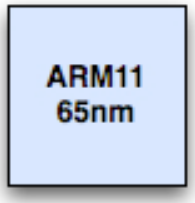


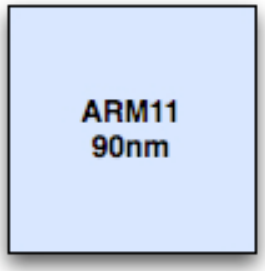


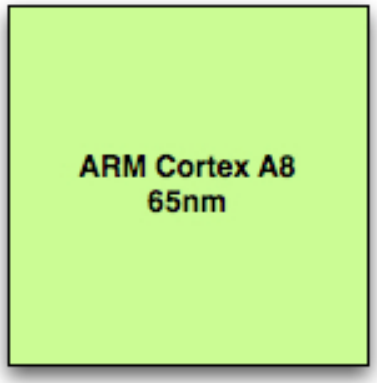




La technologie évolue (diminue) mais la complexité des systèmes augmente !

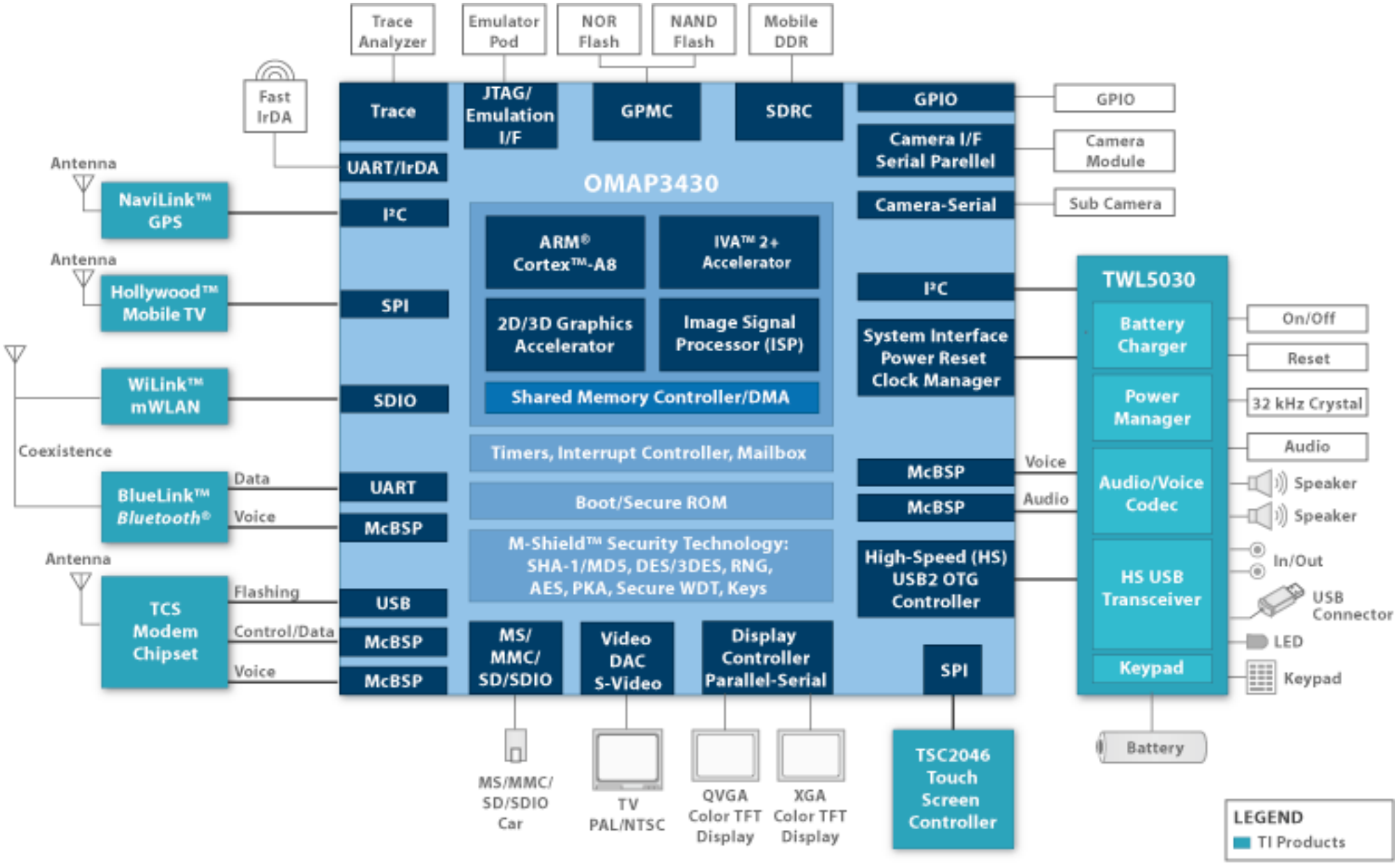






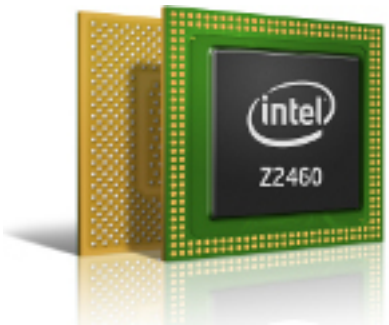
http://www.engadget.com/2009/10/14/core-values-the-silicon-behind-android/

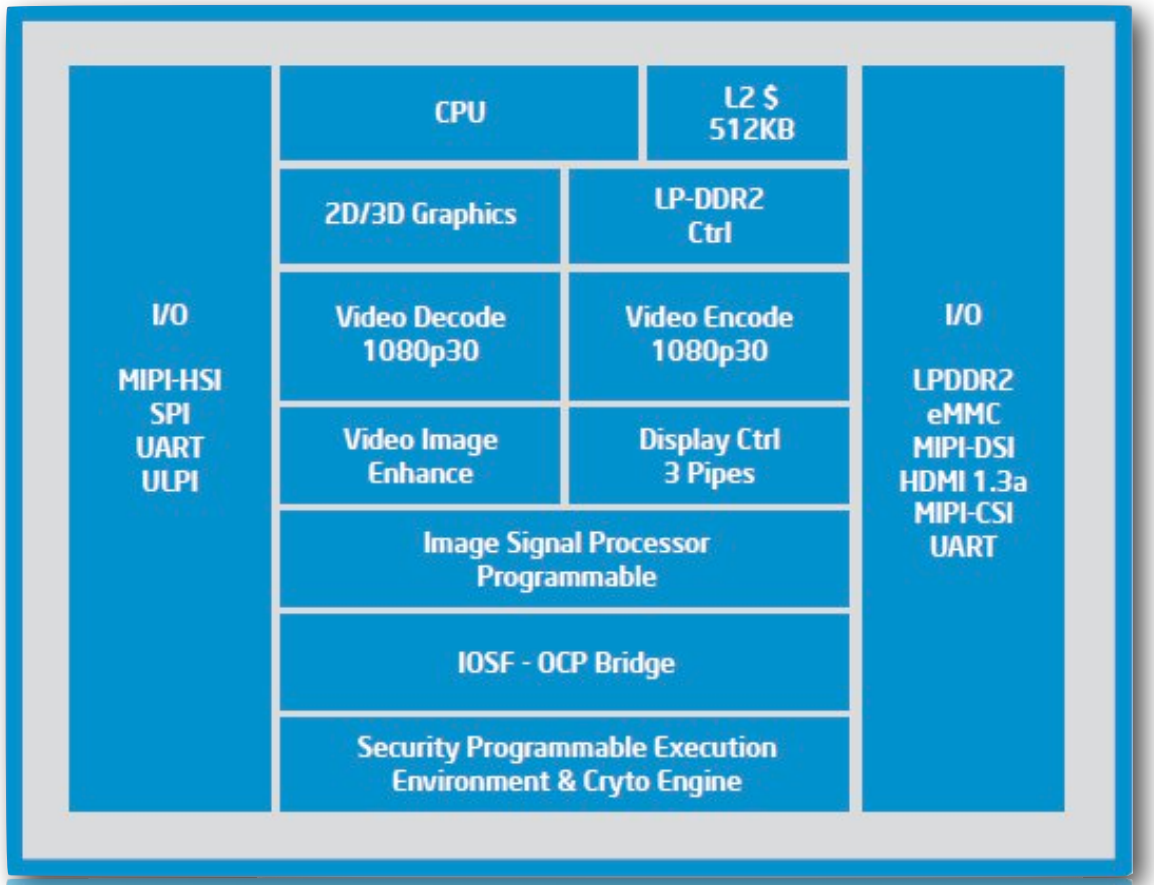
D'autres exemples de circuits vus de plus loin...



Les SoCs, systèmes plébiscités pour le domaine de l'embarqué

Les SoCs sont indissociables du domaine des systèmes embarqués à cause des contraintes fortes en consommation d'énergie !



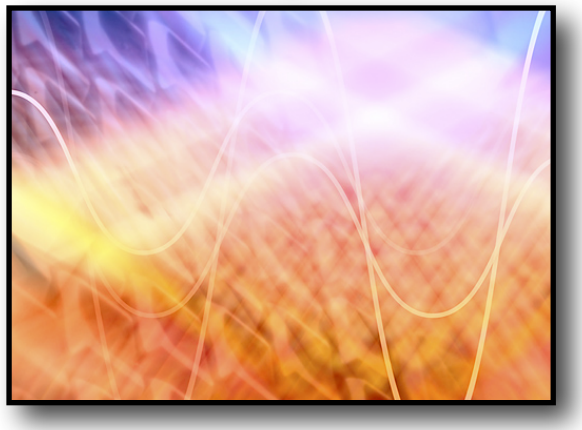


Introduction au contexte

Cahier des charges

Introduction au contexte

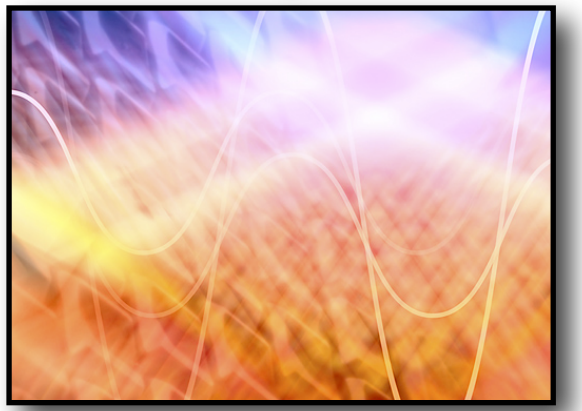
Cahier des charges



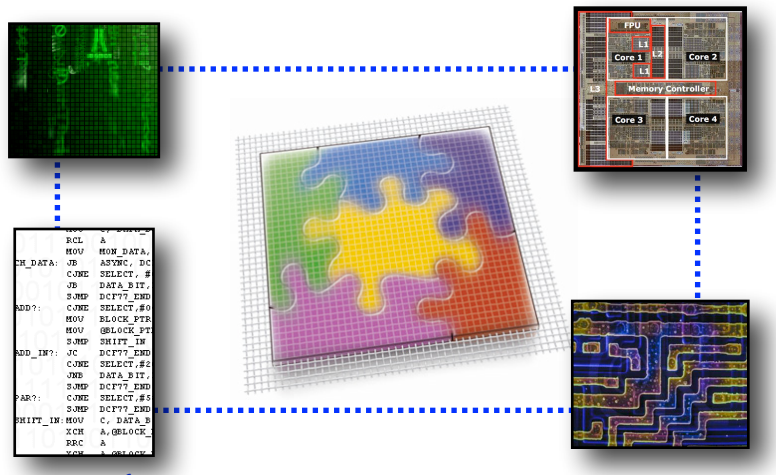
Algorithmes sous contraintes

Introduction au contexte

Cahier des charges



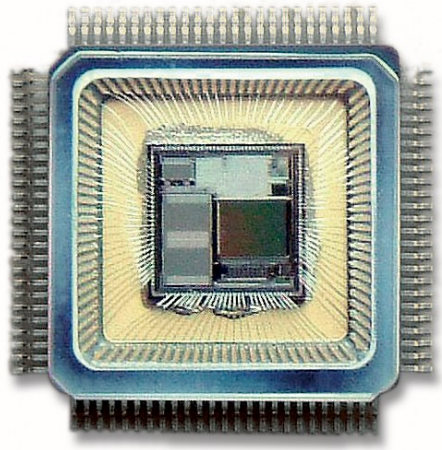
Algorithmes sous contraintes



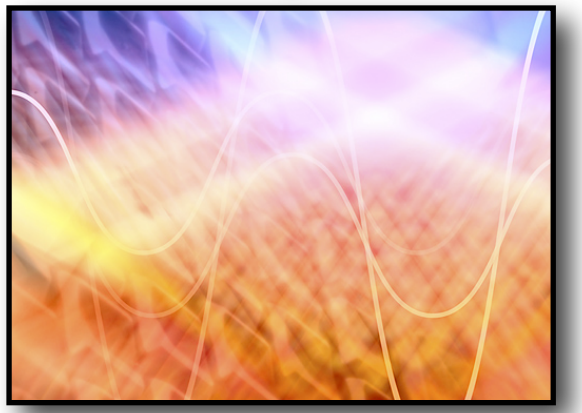
Architecture du circuit

Introduction au contexte

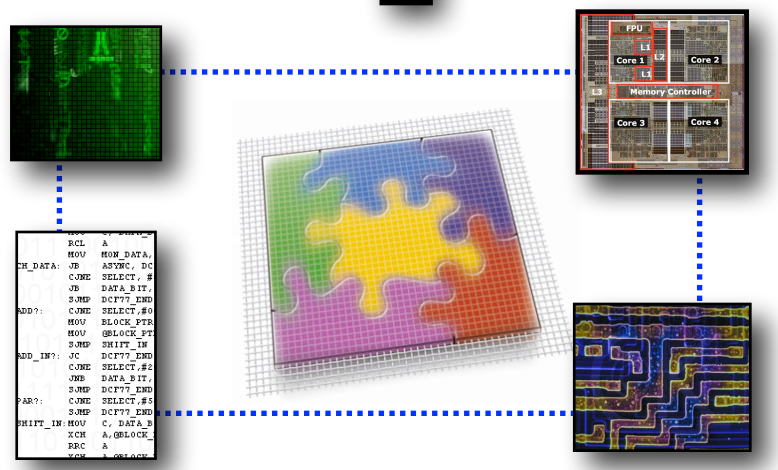
Cahier des charges



Circuit hétérogène



Algorithmes sous contraintes

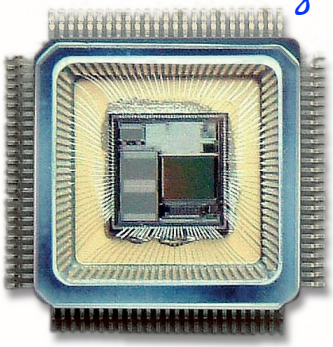


Architecture du circuit

Flot de développement de niveau système

Spécifications

Circuit numérique



Cahier des charges de l'application

Modélisation du système

Partitionnement logiciel / matériel

Modélisation du matériel

Modélisation des interfaces

Modélisation du logiciel + OS

Raffinement du matériel

Raffinement des interfaces

Raffinement du logiciel + OS

Synthèse du matériel

Co-vérification

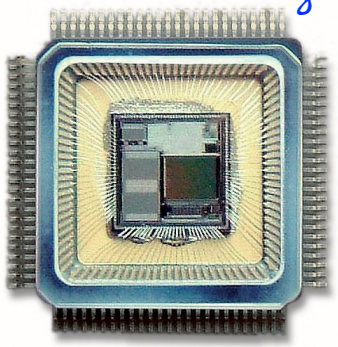
Compilation du logiciel

Intégration système

Flot de développement de niveau système

Spécifications

Circuit numérique



Cahier des charges de l'application

Modélisation du système

Partitionnement logiciel / matériel

Modélisation du matériel

Modélisation des interfaces

Modélisation du logiciel + OS

Raffinement du matériel

Raffinement des interfaces

Raffinement du logiciel + OS

Synthèse du matériel

Co-vérification

Compilation du logiciel

Intégration système

Le cahier des charges, point de départ de toute conception !

Flux binaire normalisé au format JPEG

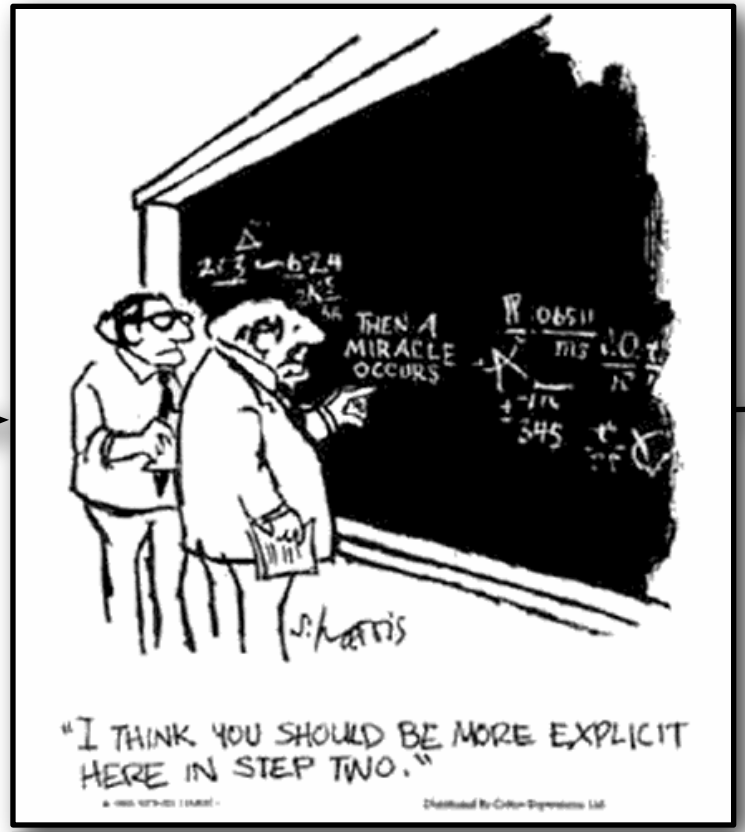


Image de type RGB codée avec 8 bits par pixel

Contraintes d'intégration:

- 12 Images par seconde minimum (),
- Encodage haute qualité,
- Faible cout de production.

Les contraintes liées aux systèmes numériques

- Des contraintes hétérogènes sur les étapes de conception,
  - Sécurité de fonctionnement,
  - "Time to market"
  - Prix de développement / production,
  - Performance (calculs),
  - Consommation d'énergie & la dissipation thermique,
  - Sûreté de fonctionnement,
  - Sécurité (virus & piratage),
  - Conditions de fonctionnement (humidité, vibrations, etc.),
  - Type des applications (contrôle, traitement de données, etc.),



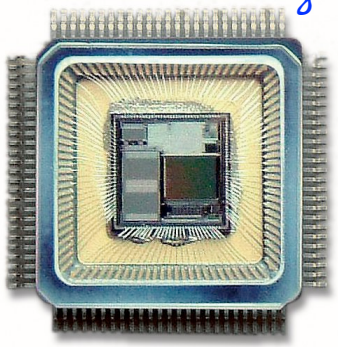




Flot de développement de niveau système

Spécifications

Circuit numérique



Cahier des charges de l'application

Modélisation du système

Partitionnement logiciel / matériel

Modélisation du matériel

Modélisation des interfaces

Modélisation du logiciel + OS

Raffinement du matériel

Raffinement des interfaces

Raffinement du logiciel + OS

Synthèse du matériel

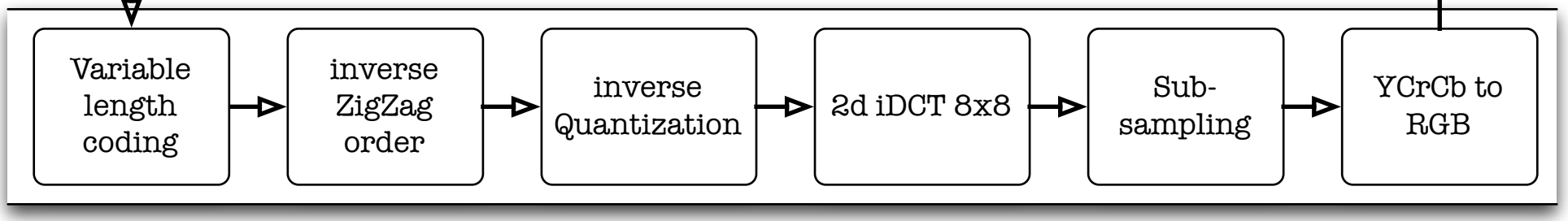
Co-vérification

Compilation du logiciel

Intégration système

Modélisation fonctionnelle de l'application à concevoir

Dans un premier temps il est nécessaire de valider le fonctionnement de modèles purement théoriques !



Lors de cette étude on va:

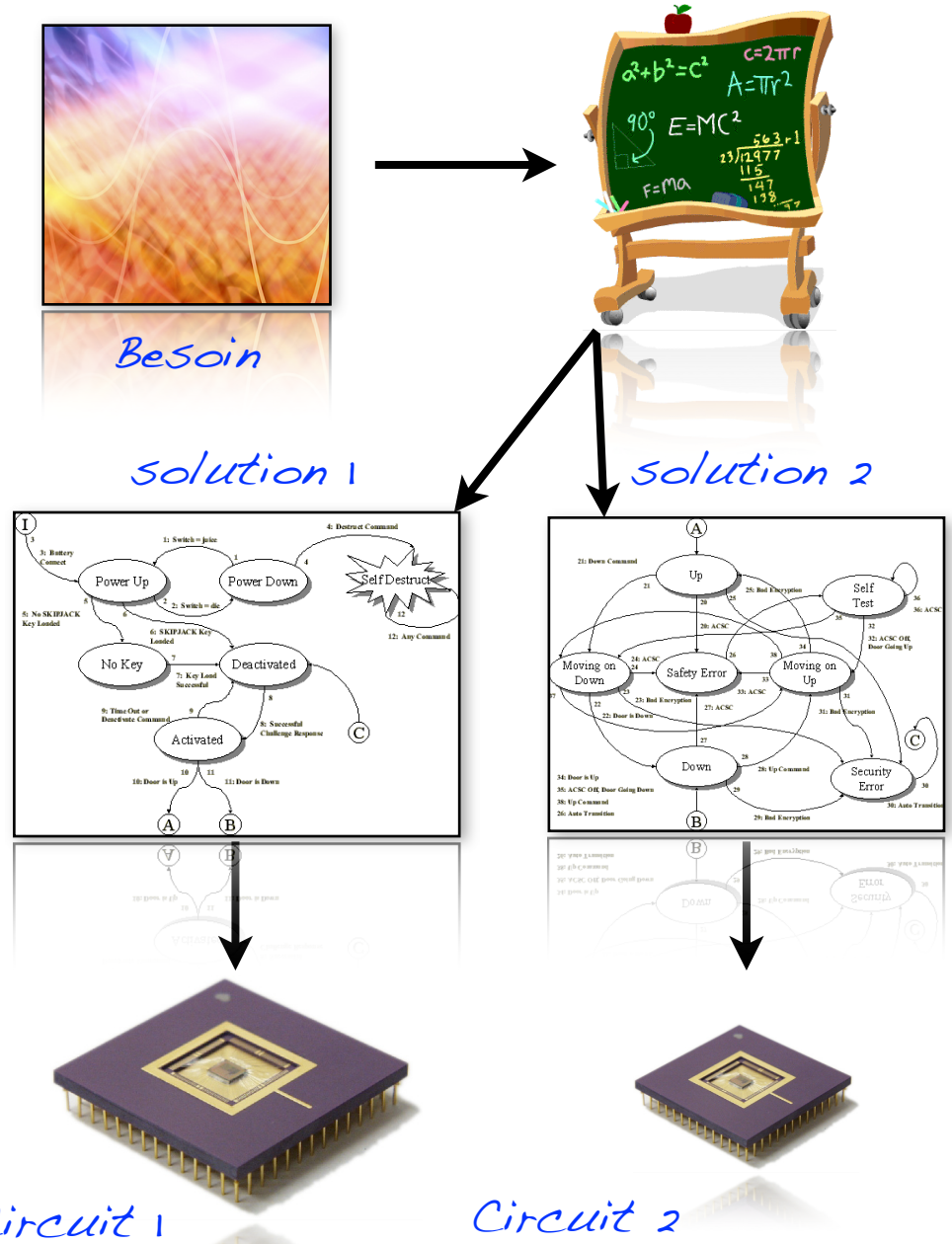
=> analyser l'application,

=> Choisir les algorithmes à implanter,

=> Transformer les algorithmes si nécessaire...

Etude des algorithmes à implanter

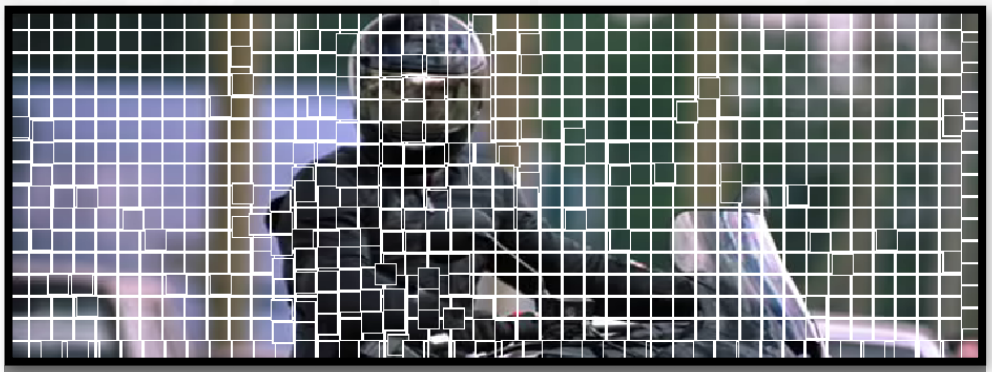
- Le choix des applications du TSI à intégrer a un impact décisif sur le circuit électronique final !
  - ➡ Nombre de calculs à réaliser (nombre total),
  - ➡ Complexité des opérations à réaliser (flottant, entier, sqrt, etc.),
- Le choix est parfois limité en fonction à cause des spécifications techniques,
- Impact fort sur
  - ➡ Consommation, surface, débit...
  - ➡ Temps de «remplacement» minime,



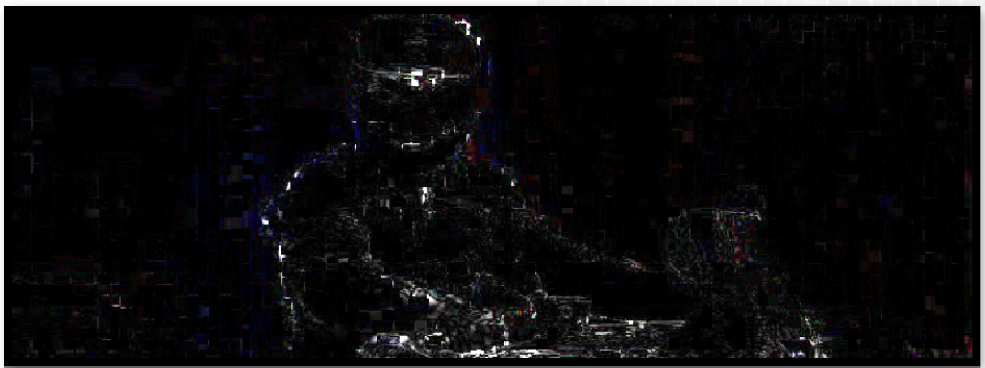
Exemple de la détection de mouvements

Première image de la séquence

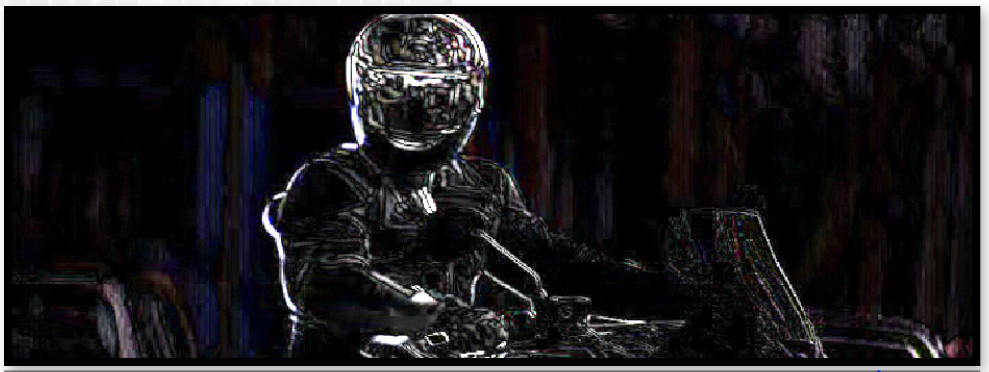
Image suivante



Mouvements détectés dans l'image



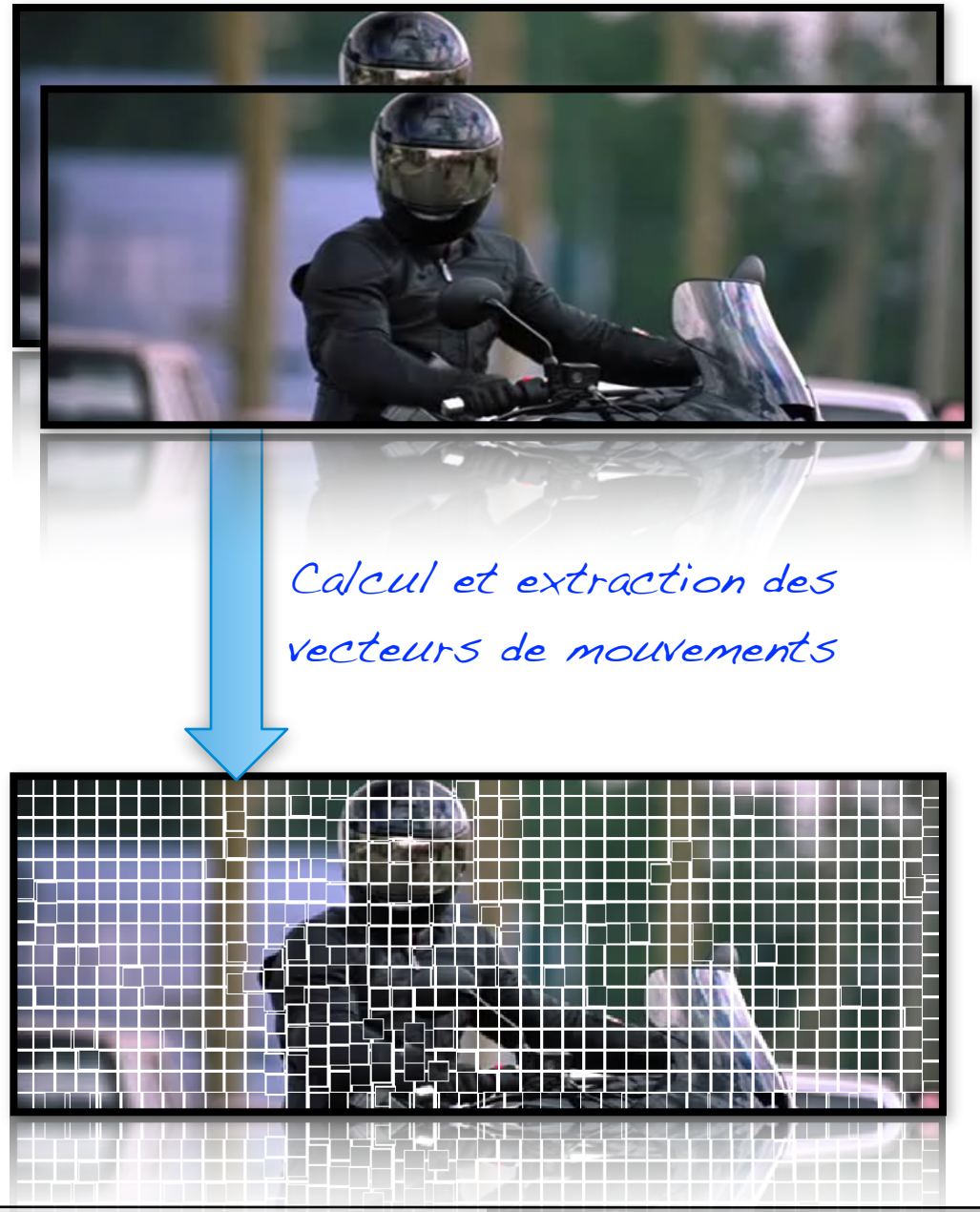
Erreurs résiduelles (algorithme 1)



Erreurs résiduelles (algorithme 2)

Les différentes méthodes disponibles dans la littérature

- Optimisation de la complexité algorithmique,
  - Full Search Algorithm (256 opr)
  - Three Step Search Algorithm (25 opr)
  - Othogonal Search Algorithm (13 opr)
  - Cross Search Algorithm (17 opr)
- Optimisation de la complexité calculatoire,
  - SAD (Sum of Absolute Differences)
  - SSE (Sum of Squared Errors)
  - SATD (Sum of Absolute Hadamard Transformed Differences)
  - RD (Rate Distortion Optimal)



Exemple des simplifications calculatoires

$$Y(n) = A \times X + A \times Y(n-1)$$

Réduction du nombre de calculs à réaliser en factorisant une expression mathématique

$$Y(n) = A \times (X + Y(n-1))$$

$$Y(n) = x(0) \times H(0) + x(1) \times H(1) + x(2) \times H(2) + x(3) \times H(3)$$

Sélection d'un ensemble de coefficients symétriques dans le filtre afin de simplifier les calculs en factorisant les échantillons

$$Y(n) = (x(0) + x(3)) \times H(0/3) + (x(1) + x(2)) \times H(1/2)$$

Changement des algorithmes utilisés

Pour calculer la corrélation entre plusieurs blocs de données, il existe plusieurs techniques présentées dans la littérature.

$$\delta = \sum_{y=0}^{7}\sum_{x=0}^{7}(I_1(x,y) - I_2(x,y))^2$$

Sum of Square Differences

$$\delta = \sum_{y=0}^{7}\sum_{x=0}^{7}|I_1(x,y) - I_2(x,y)|$$

Sum of Absolute Differences

- Quelles sont les complexités calculatoires de ces 2 méthodes ?
- Quel impact le choix de la méthode aura t'il ?

Réduction de la complexité d'implantation

$$\delta = \sum_{y=0}^{3}\sum_{x=0}^{3} |I_1(x,y) - I_2(x,y)|$$

On va essayer de modifier l'algorithme afin de réduire le nombre moyen de calculs à effectuer...

```
int ComputeSAD(int* I1, int* I2){
    int diff = 0;
    for(int y=0; y<8; y++){
        for(int x=0; x<8; x++){
            diff = diff + abs(I1[x,y] - I2[x,y]);
        }
    }
    return diff;
}
```

```
int ComputeSAD(int* I1, int* I2, int max){
    int diff = 0;
    for(int y=0; y<8; y++){
        for(int x=0; x<8; x++){
            diff = diff + abs(I1[x,y] - I2[x,y]);
            if( diff > max ) return diff;
        }
    }
    return diff;
}
```

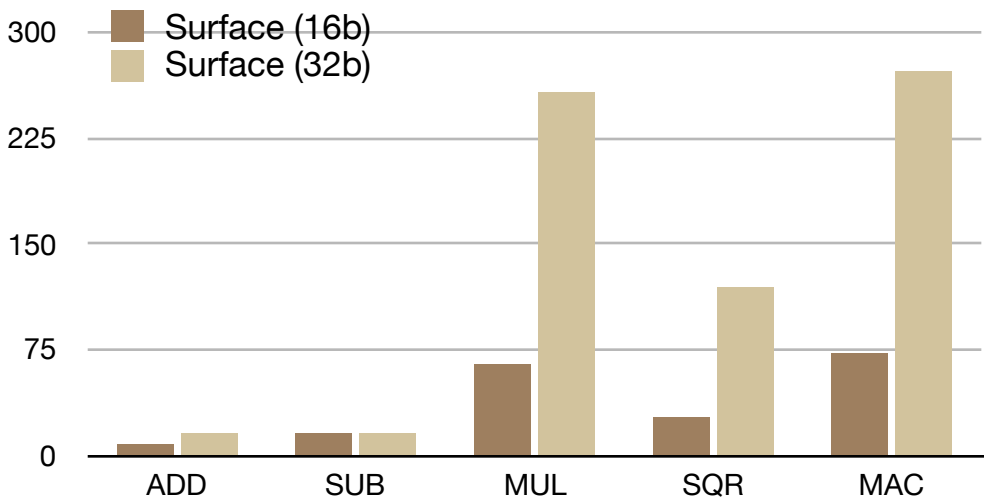
```
int ComputeSAD(int* I1, int* I2, int max){
    int diff = 0;
    for(int y=0; y<8; y++){
        for(int x=0; x<8; x++){
            diff = diff + abs(I1[x,y] - I2[x,y]);
        }
        if( diff > max ) return diff;
    }
    return diff;
}
```

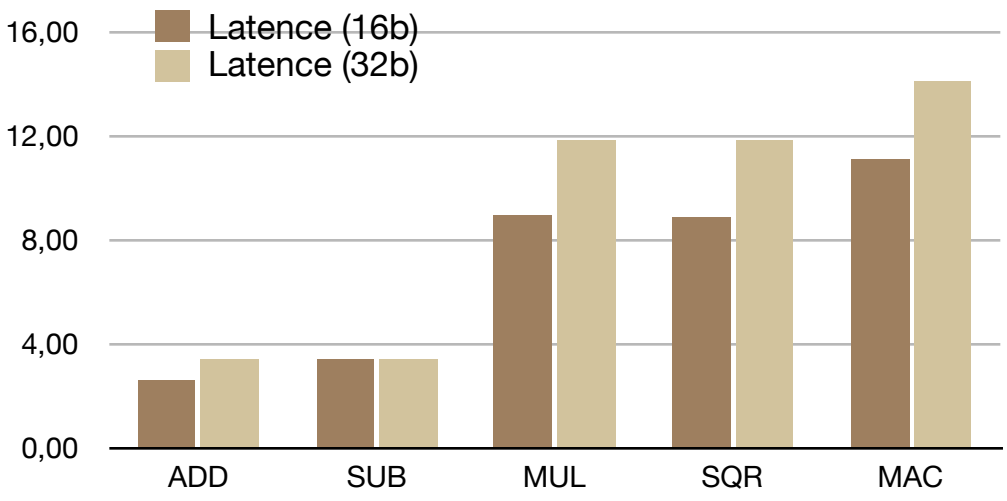
Exemples de simplifications opératoires

$$Y = 2 \times A + B \xrightarrow{\text{Simplification de la multiplication}} Y = (A << 1) + B$$

$$Y = 5 \times A + B \xrightarrow{\text{Idem}} Y = ((A << 1) + A) + B$$

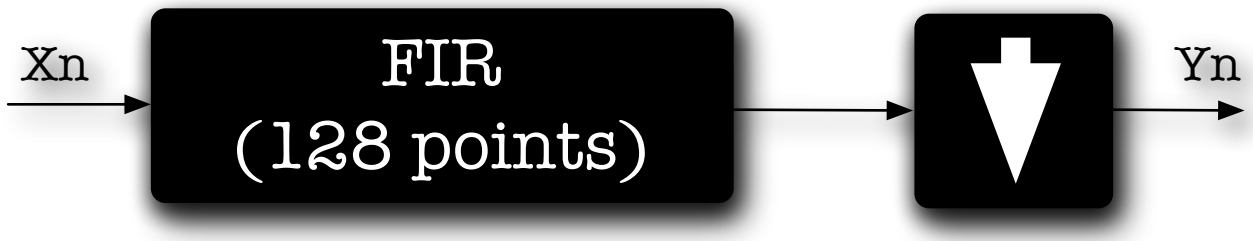
$$Y = \frac{A}{4} + B \xrightarrow{\text{Simplification de la division}} Y = (A >> 2) + B$$





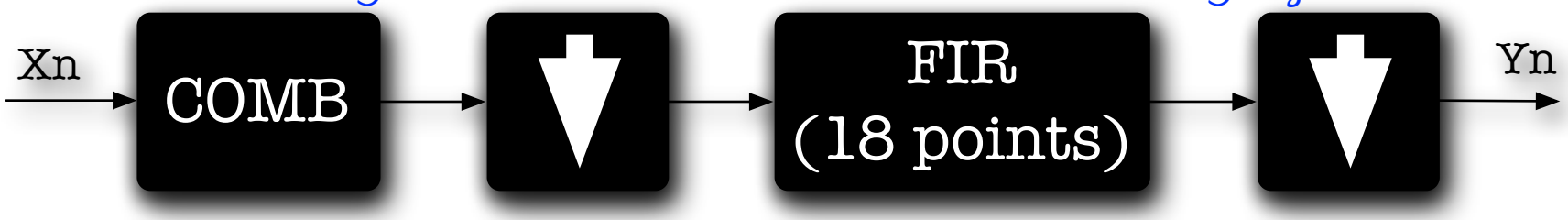
Modification des techniques de filtrage

Filtrage passe bas limitant la bande passante avant décimation afin d'éviter le repliement spectral.

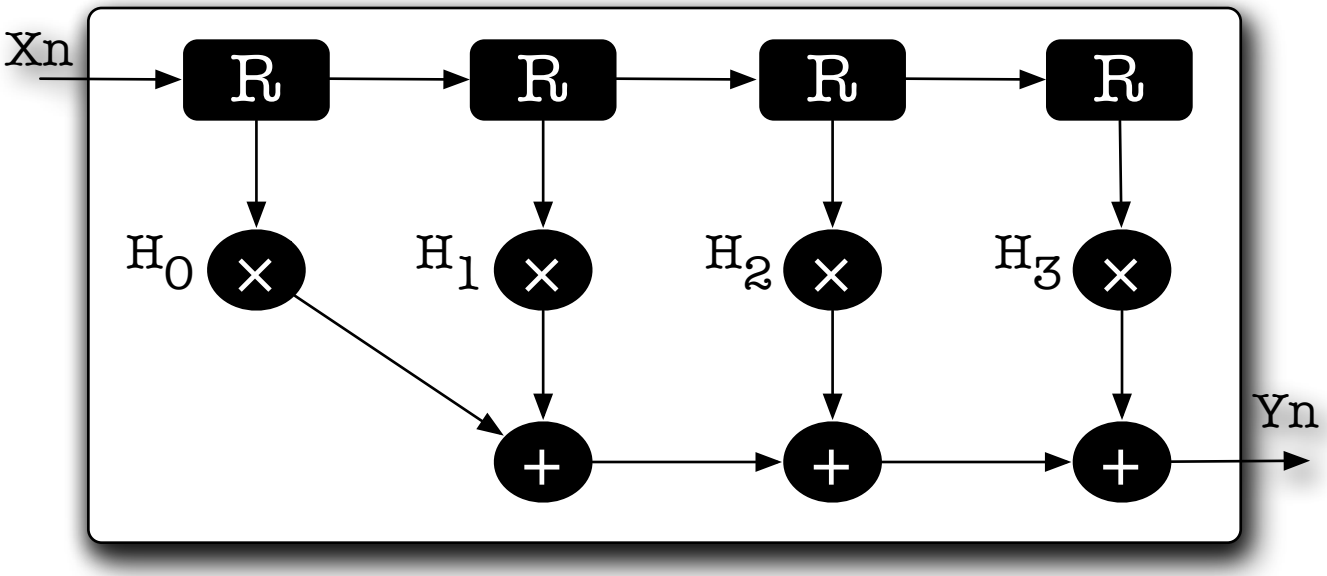


Transformation algorithmique

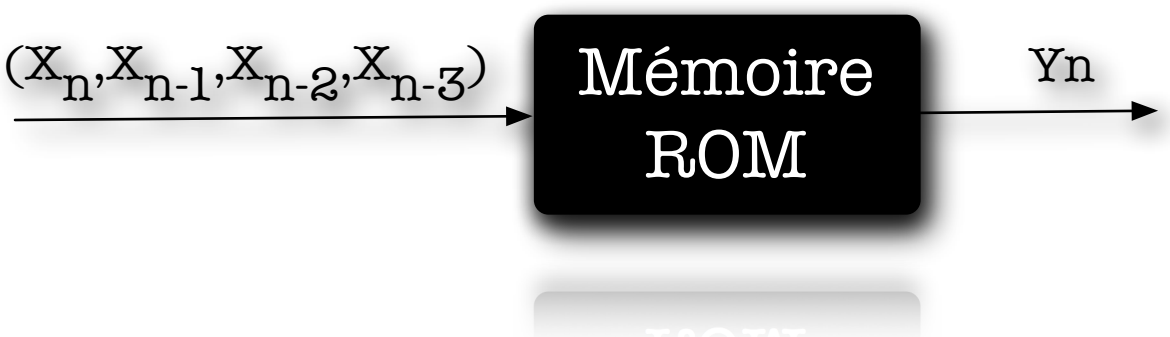
Modification des algorithmes assurant le fonctionnement de la chaîne de filtrage afin de réduire la consommation énergétique.



Exemple de simplification architecturale



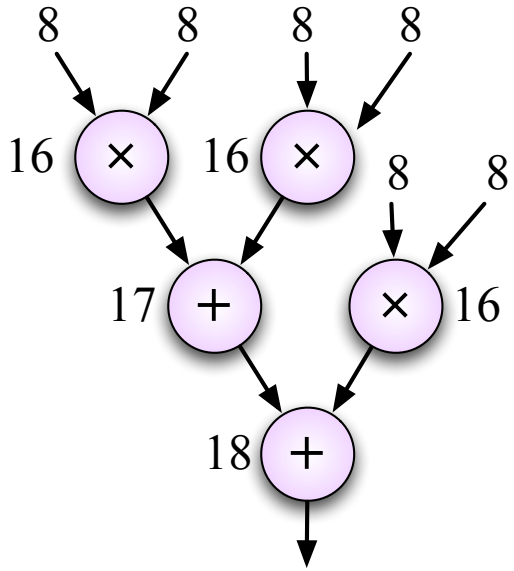
Si l'entrée du filtre (Xn) est codée sur 2 bits alors il existe 4^4 valeurs possibles pour la sortie en considérant (H/x) comme des constantes.

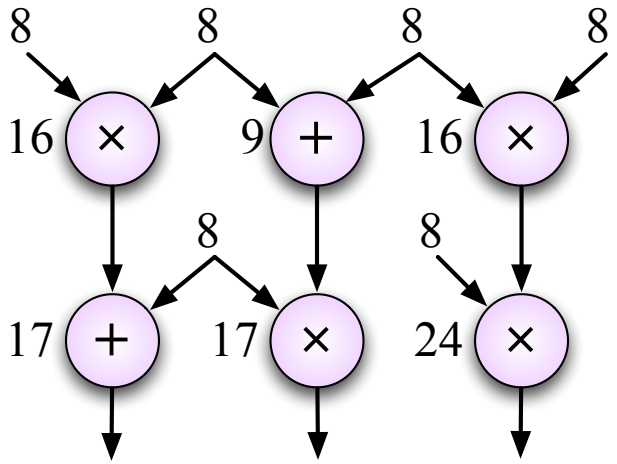


Une solution avantageuse est de remplacer le circuit qui réalise les calculs par une mémoire contenant l'ensemble des valeurs pré-calculées !

Etude de la dynamique des données

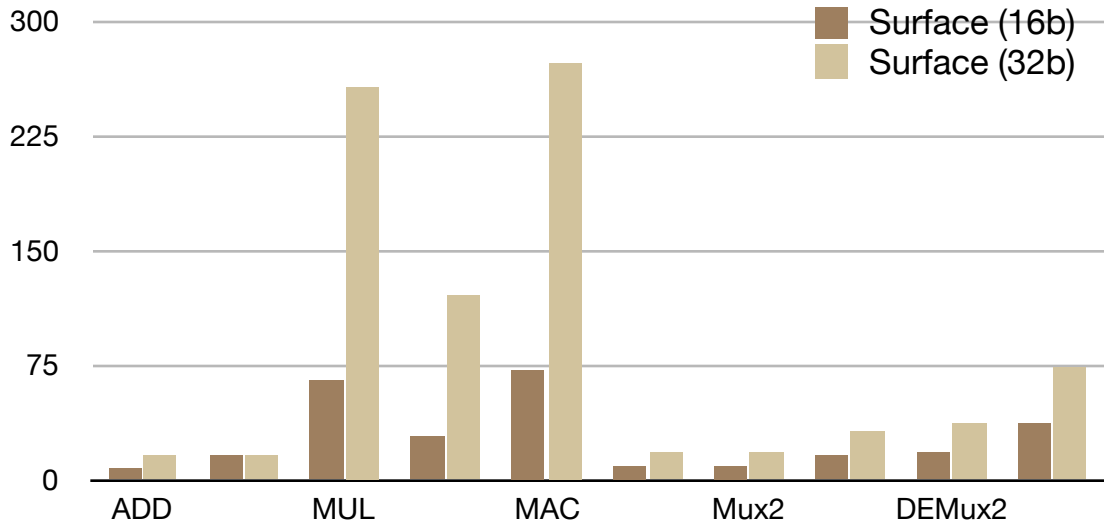
- La majorité des modèles algorithmiques sont développés sans “trop” se préoccuper de de la dynamique des calculs,
  - ➡ Utilisation de nombres flottants (32b ou 64b),
- La problématique est similaire en informatique : les ressources sont codées sur 32 bits...
- La connaissance et l’étude de la dynamique des données est complexe sur des applications réelles !

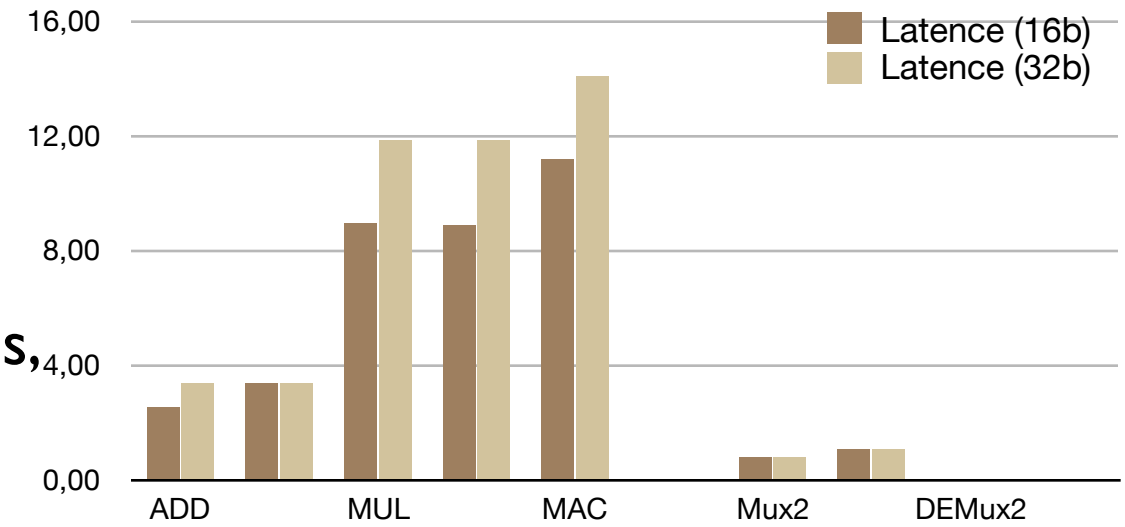




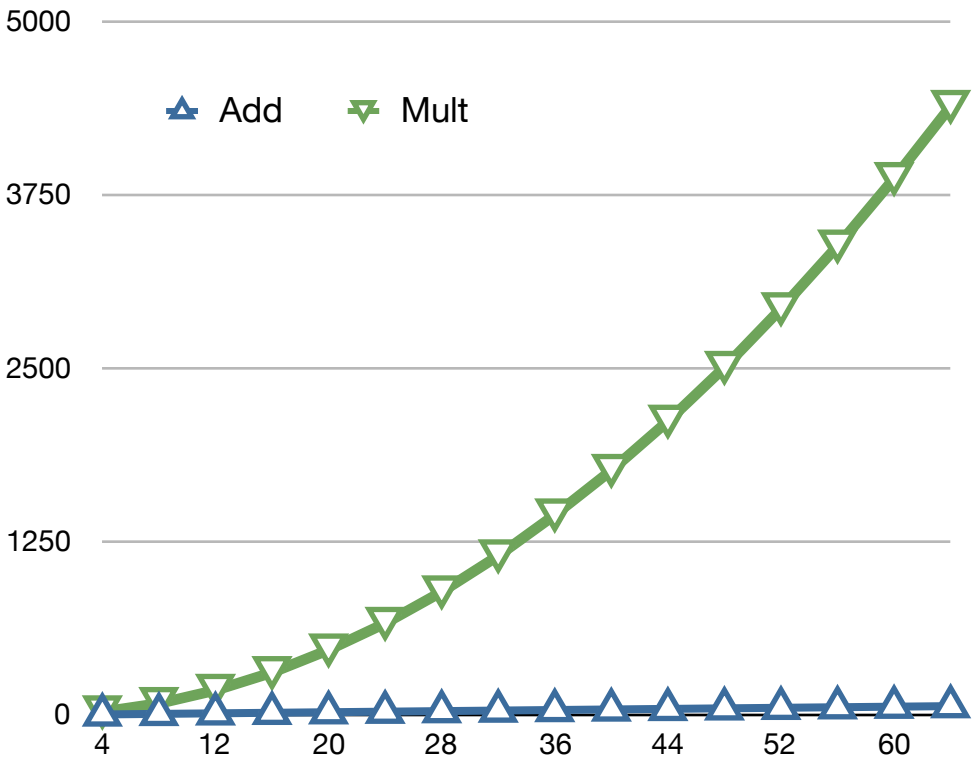
Impact de la dynamique des données sur les performances

- La dynamique des données et des opérations et un point crucial car il impact sur :
  - La surface (coût du circuit),
  - La latence et le débit (performances),
  - La fonctionnalité (Ariane 5),
  - La qualité (données tronquées),
- Pour étudier la dynamique il faut de préférence connaître l'application à intégrer !
- Utilisation de : modèles formels, simulations expérimentales...

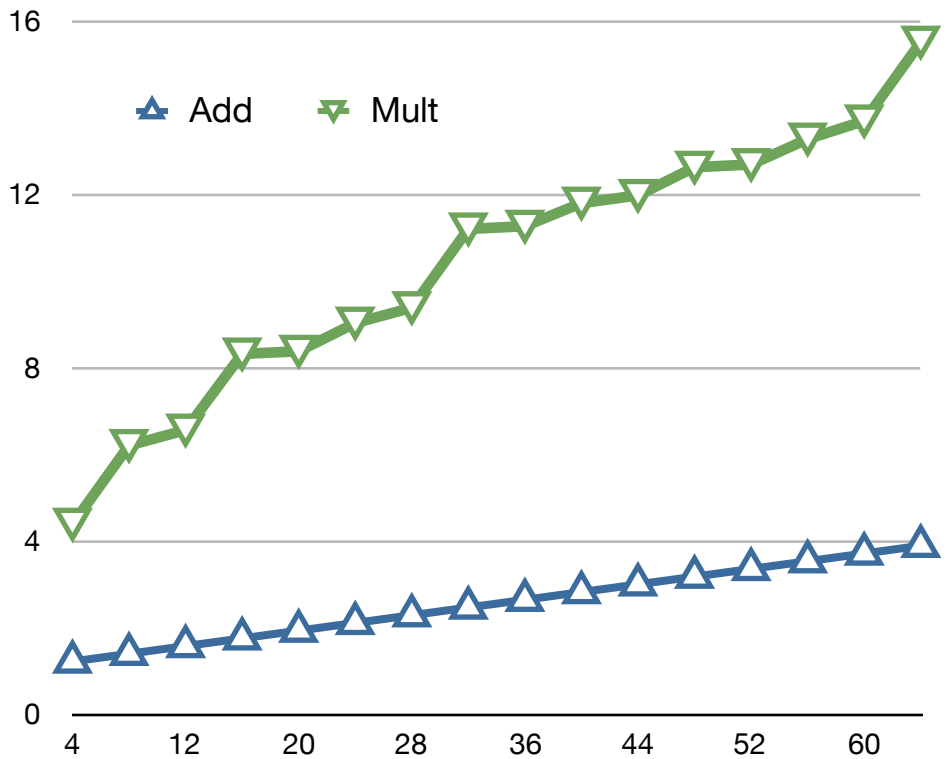




Analyse des performances des opérateurs (ADD & MULT)



Surface des ressources (LUTs) - XC4VSX35-FF668-10



Chemin critique des ressources (ns) - XC4VSX35-FF668-10

Sur les technologies ASIC et FPGA:

=> Les paramètres changent en fonction de la dynamique de l'opération,

=> Certaines ressources matérielles sont plus efficaces que d'autres,

=> Les performances sont très dépendantes de la cible technologique.

Exemple d'étude de la dynamique

- Prenons l'exemple d'un algorithme de calcul de corrélation employé dans l'estimation de mouvement (MPEG),
  - Les données en entrée du circuit sont des pixels codés sur 8 bits,
  - Calculez la dynamique des données et des calculs présent dans l'algorithme,
  - Pour simplifier votre étude, nous considérons que nous travaillons sur des macroblocs de 4x4 pixels,

$$\delta = \sum_{y=0}^{3} \sum_{x=0}^{3} abs(I_1(x,y) - I_2(x,y))$$

Comment pourrait on affiner nos prévisions ?

=> Simulation sur un panel d'images important (profiling)

=> Méthode statistique permettant d'estimer les valeurs extrêmes

=> Changement du comportement de l'algorithme (valeur de dégagement)

Utilisation du codage en virgule fixe

$$y = \pi \times x^2 - 1.45 \times x + 1.2453$$

Nous souhaitons implanter cette opération sur un uP possédant des unités de calcul entières uniquement !

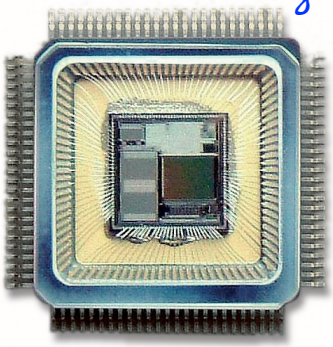
=> Exprimez ce calcul dans le cas ou les données flottantes sont codées sur 16 bits puis {8, 12, 24} bits.

=> Evaluez la précision du résultat du calcul si l'on considère que «y» est une donnée de type INT (troncature à l'inférieur)

Flot de développement de niveau système

Spécifications

Circuit numérique



Cahier des charges de l'application

Modélisation du système

Partitionnement logiciel / matériel

Modélisation du matériel

Modélisation des interfaces

Modélisation du logiciel + OS

Raffinement du matériel

Raffinement des interfaces

Raffinement du logiciel + OS

Synthèse du matériel

Co-vérification

Compilation du logiciel

Intégration système

L'hétérogénéité des circuits actuels

- Une des problématiques vient de l'hétérogénéité possible des solutions !
- Lien fort entre la solution et l'application à intégrer :
  - Traitement vidéo : processeur (interactions) et accélérateur matériel (traitements),
  - Téléphonie : accélérateurs (décodage canal et corrections) et processeur (gestion des configurations),
- Les choix technologiques dépendent aussi des contraintes d'intégration.

Exemple d'étude pour une application

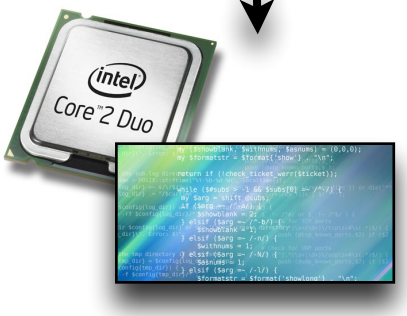
Compression vidéo h264

Développement, Temps réel

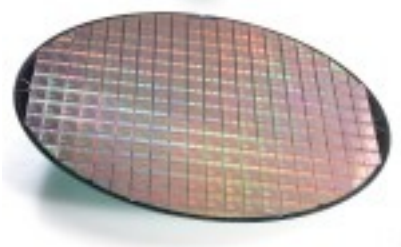
Qualité, temps réel

Coût, Qualité

Coût, Energie

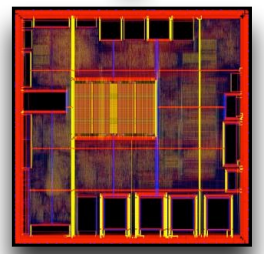


Processeur généraliste + programme



ASIC dédié hautes performances

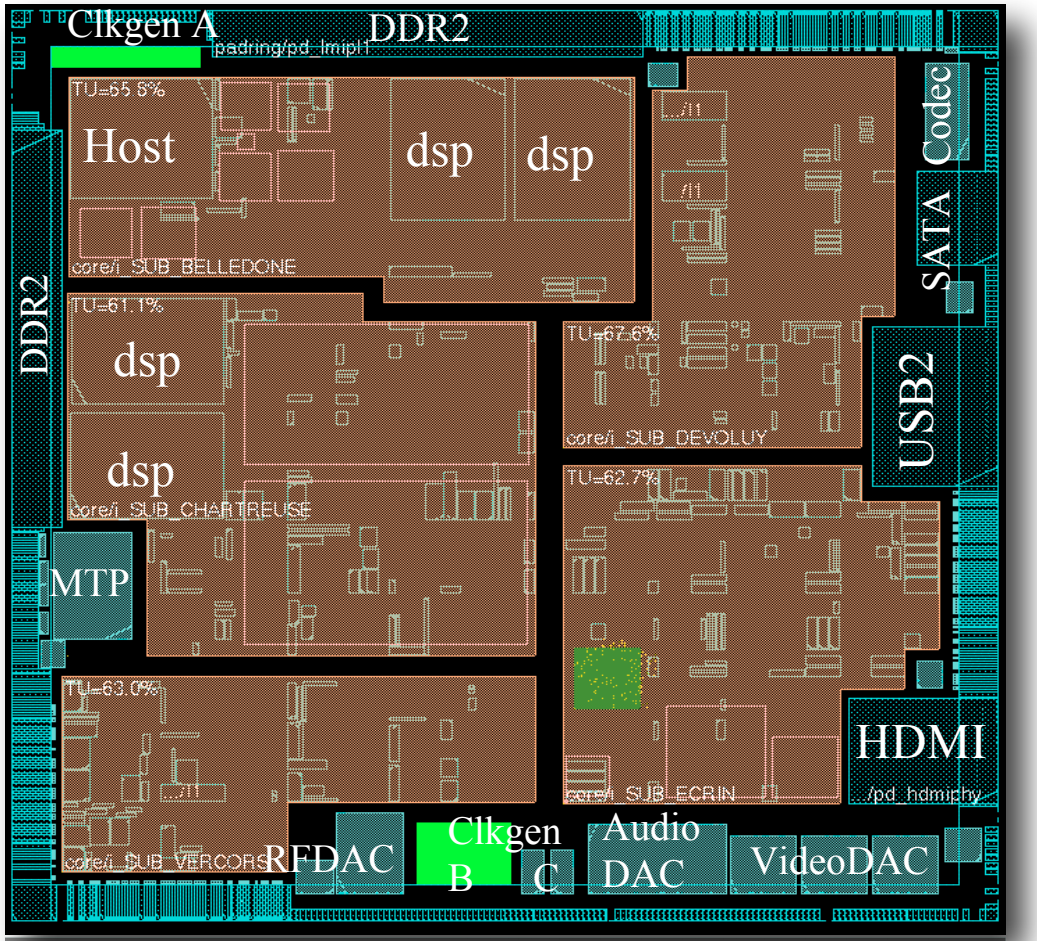
ASIC dédié et partagé performances moyennes



Sous système (HW/SW) faibles performances

Exemple, le décodeur “HD 264” de STMicro

- Circuit dédié au décodage de la TV HD (norme H264)
  - Circuit contenant 150M transistors et 886 pads IOs (~5 GMIPS)
  - 128 sources d’interruption
  - 73 initiateurs et 96 cibles sur les bus,
  - 115 réseaux d’horloge (19 pour les interconnexions),
- Construction du système
  - 4 processeurs (2 DSP pour la vidéo, 1 DSP pour l’audio et 1 généraliste pour la configuration),
  - 36 Soft IPs + 2 Hard IPs
  - 140 buffers (mémoires).

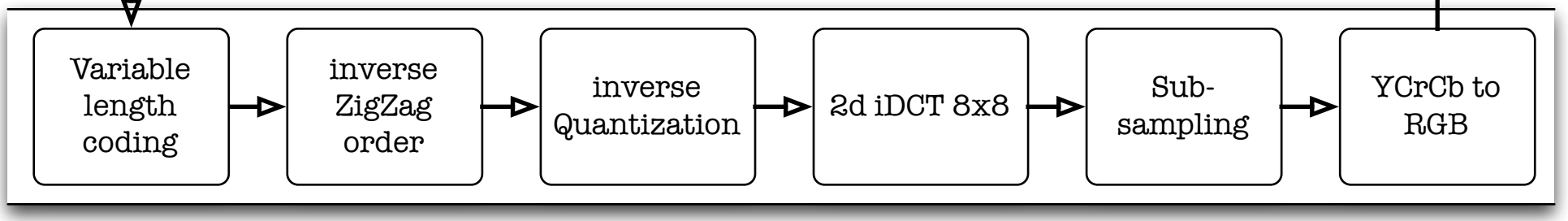


Circuit développé en 200X ?!

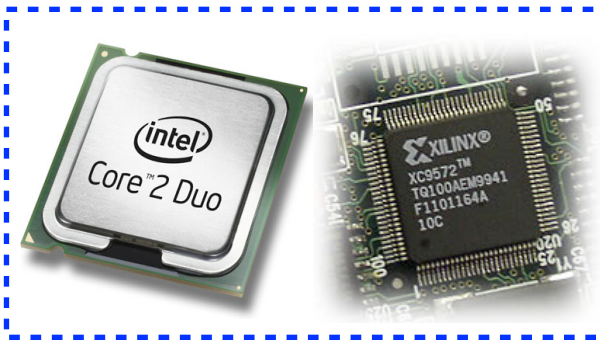
Que fait on alors aujourd'hui ...

Le partitionnement d'une application

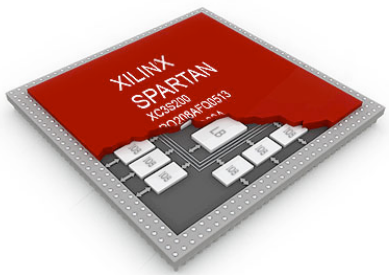
Comment faire afin de déterminer l'architecture optimum pour implanter le système ?



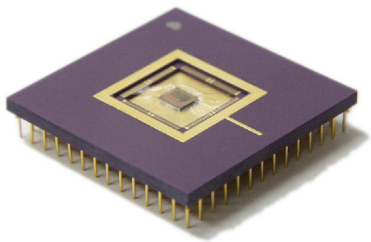
Processeur



Processeur + accélérateur

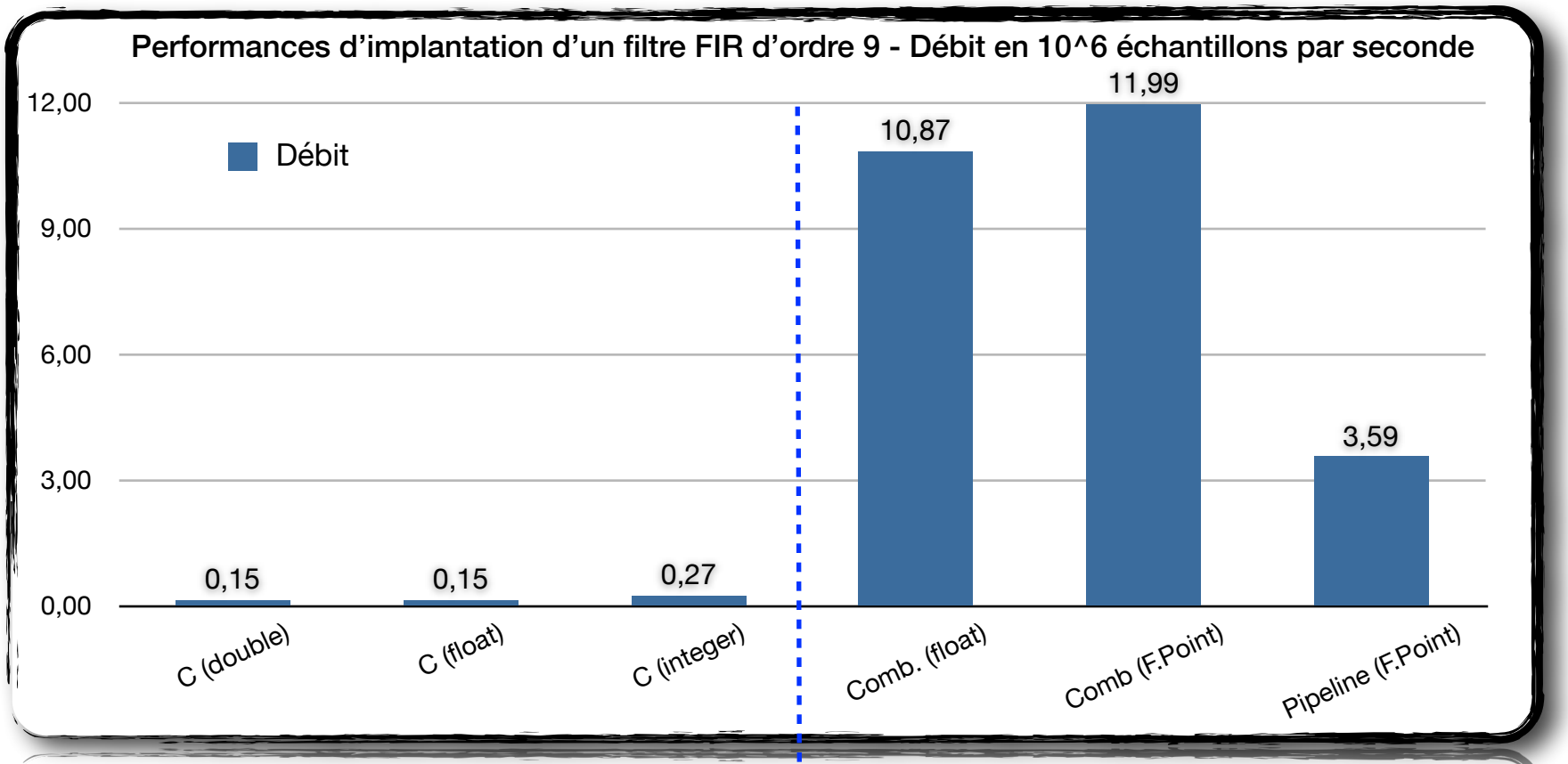


FPGA



ASIC

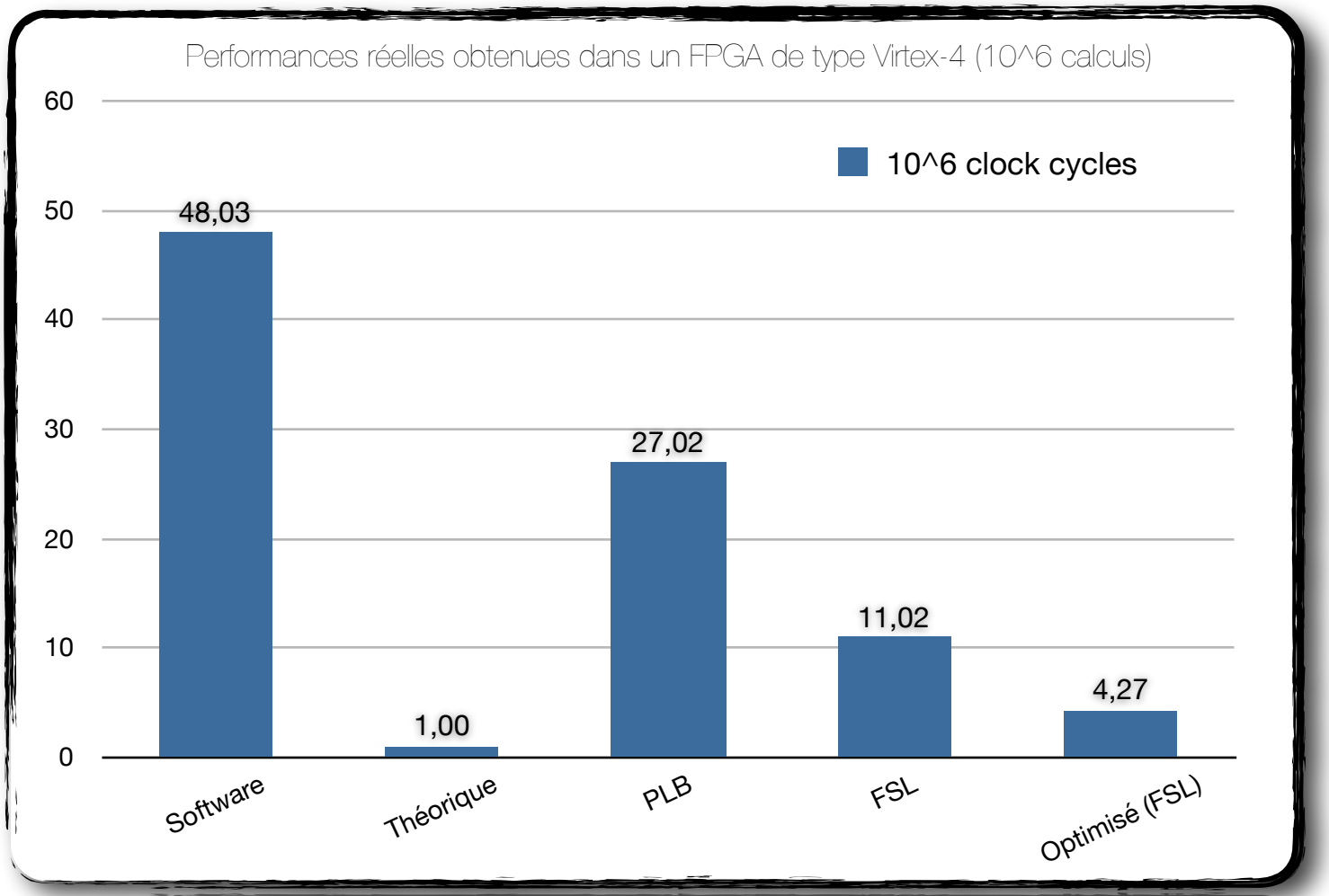
Les processeurs généralistes ne sont pas toujours efficaces...



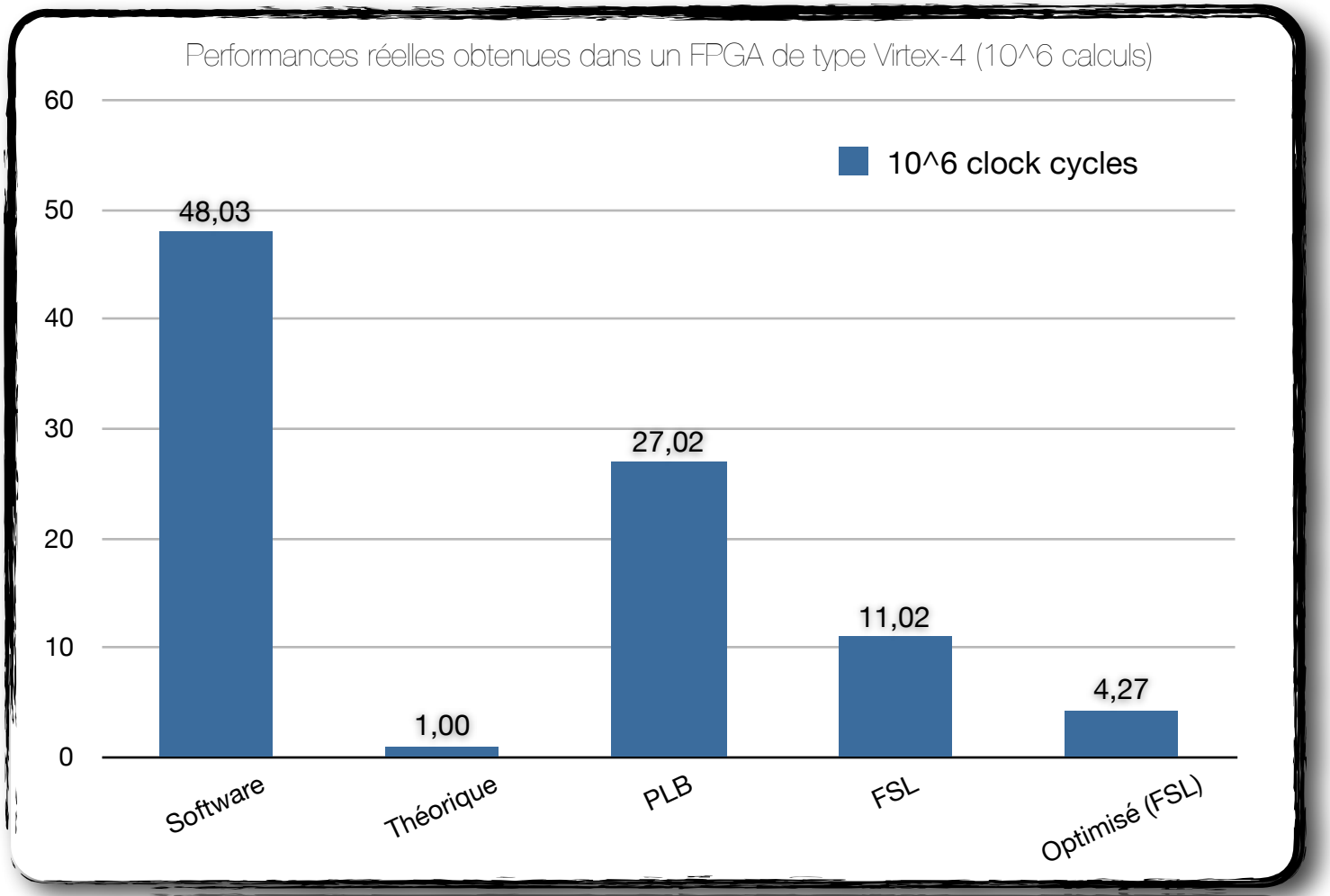
Implantations sur un processeur de type Core2Duo 2,4 GHz

Implantations dans un FPGA de type Virtex-4 (10- 200MHz)

Evaluation de l'accélérateur en conditions réelles

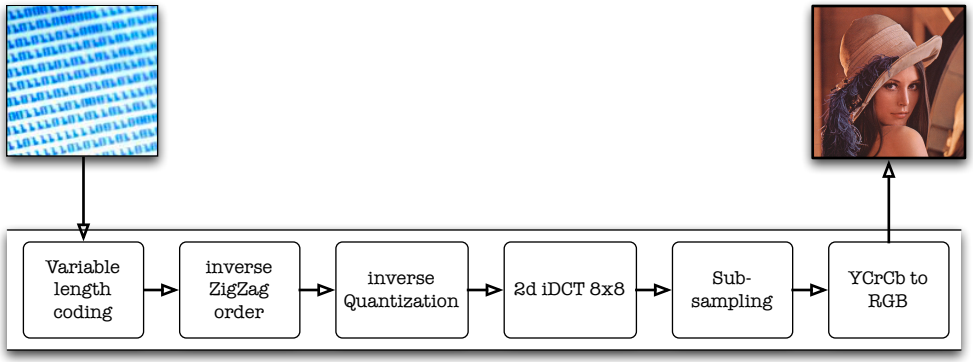


Evaluation de l'accélérateur en conditions réelles



Implantation en virgule fixe sur un coeur de processeur softcore versus l'utilisation d'un accélérateur matériel (ML-402)

Le partitionnement d'une application



Nombre de calculs réalisables en parallèle,

Nombre d'accès à la mémoire ?

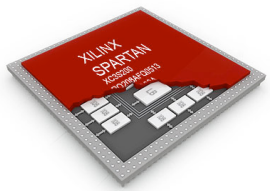
Transfert d'information / liens avec les autres blocs,

Contraintes temporelles à assurer,

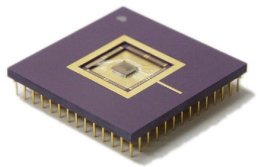
Temps de conception à disposition...

Processeur

Processeur + accélérateur



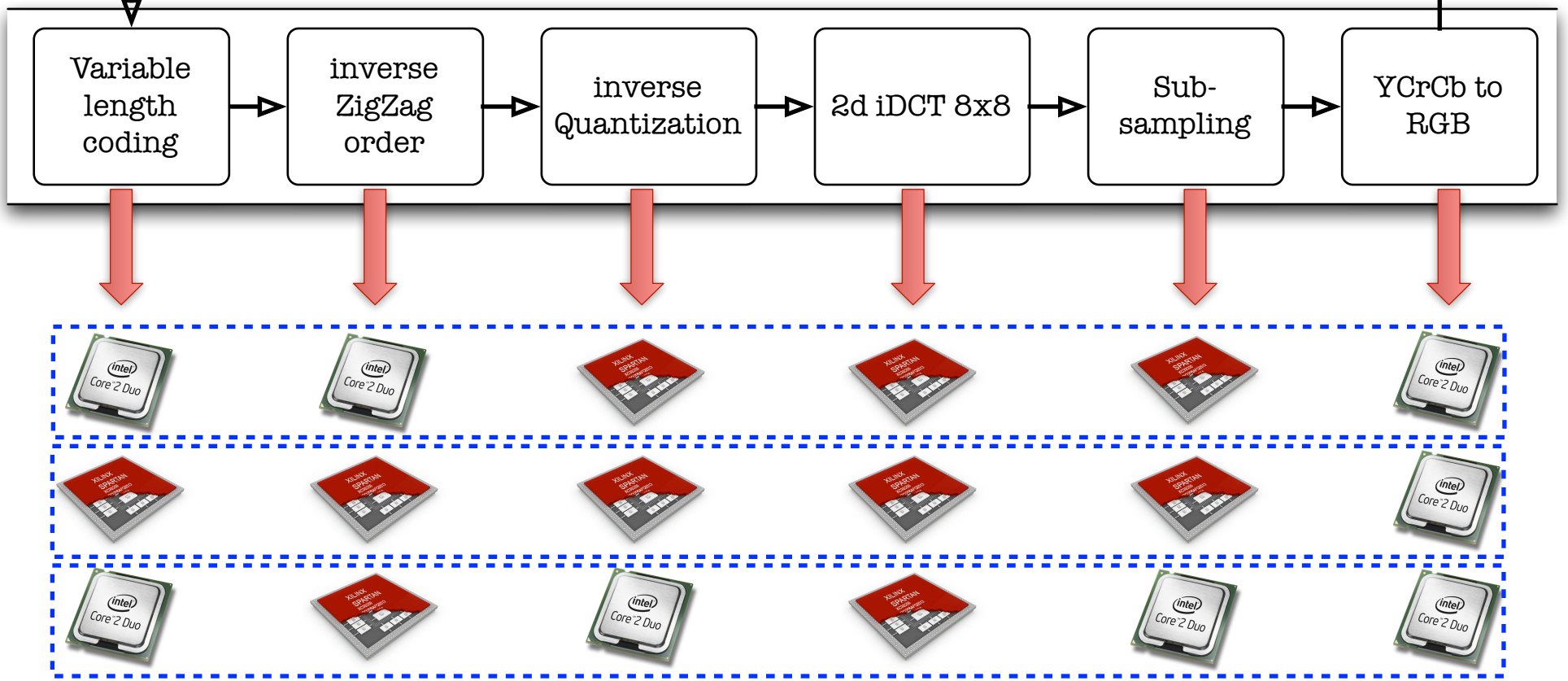
FPGA



ASIC

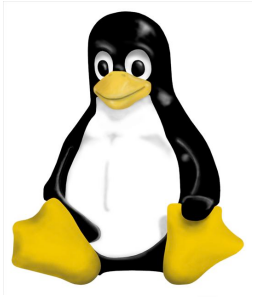
Différentes solutions sont imaginables...

Lorsque les algorithmes sont connus et couramment utilisés, il peut exister des composants déjà conçus !

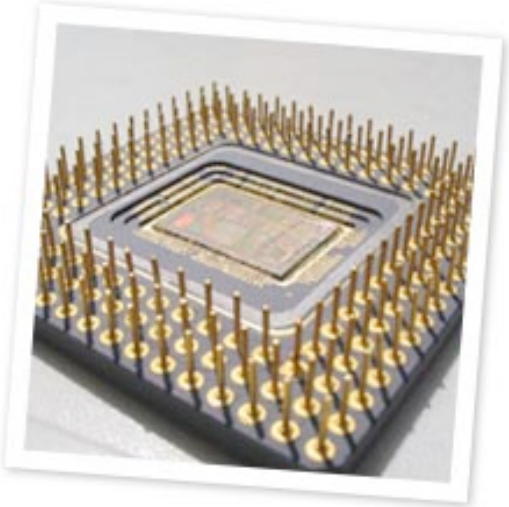


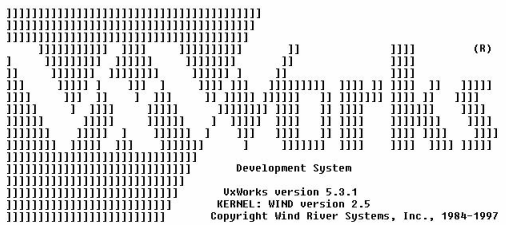
Gestion de la partie logicielle du co-design

- La partie logicielle sera développée avec pour cible un processeur embarqué ou non,
  - PicoBlaze, MicroBlaze, PPC, ARM, etc.
  - DSP C6x de Texas Inst.
- Il est possible de rajouter :
  - Les connexions aux réseaux, bluetooth, IR, USB, etc.
  - D'autres fonctionnalités : agenda, e-mail, SMS, MMS, etc.
- Il existe un grand nombre d'OS développés pour les systèmes embarqués temps réel,



Microsoft Windows Automotive



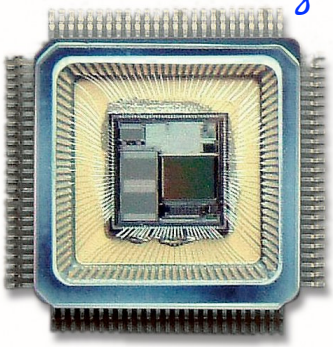


QNX SOFTWARE SYSTEMS

Flot de développement de niveau système

Spécifications

Circuit numérique



Cahier des charges de l'application

Modélisation du système

Partitionnement logiciel / matériel

Modélisation du matériel

Modélisation des interfaces

Modélisation du logiciel + OS

Raffinement du matériel

Raffinement des interfaces

Raffinement du logiciel + OS

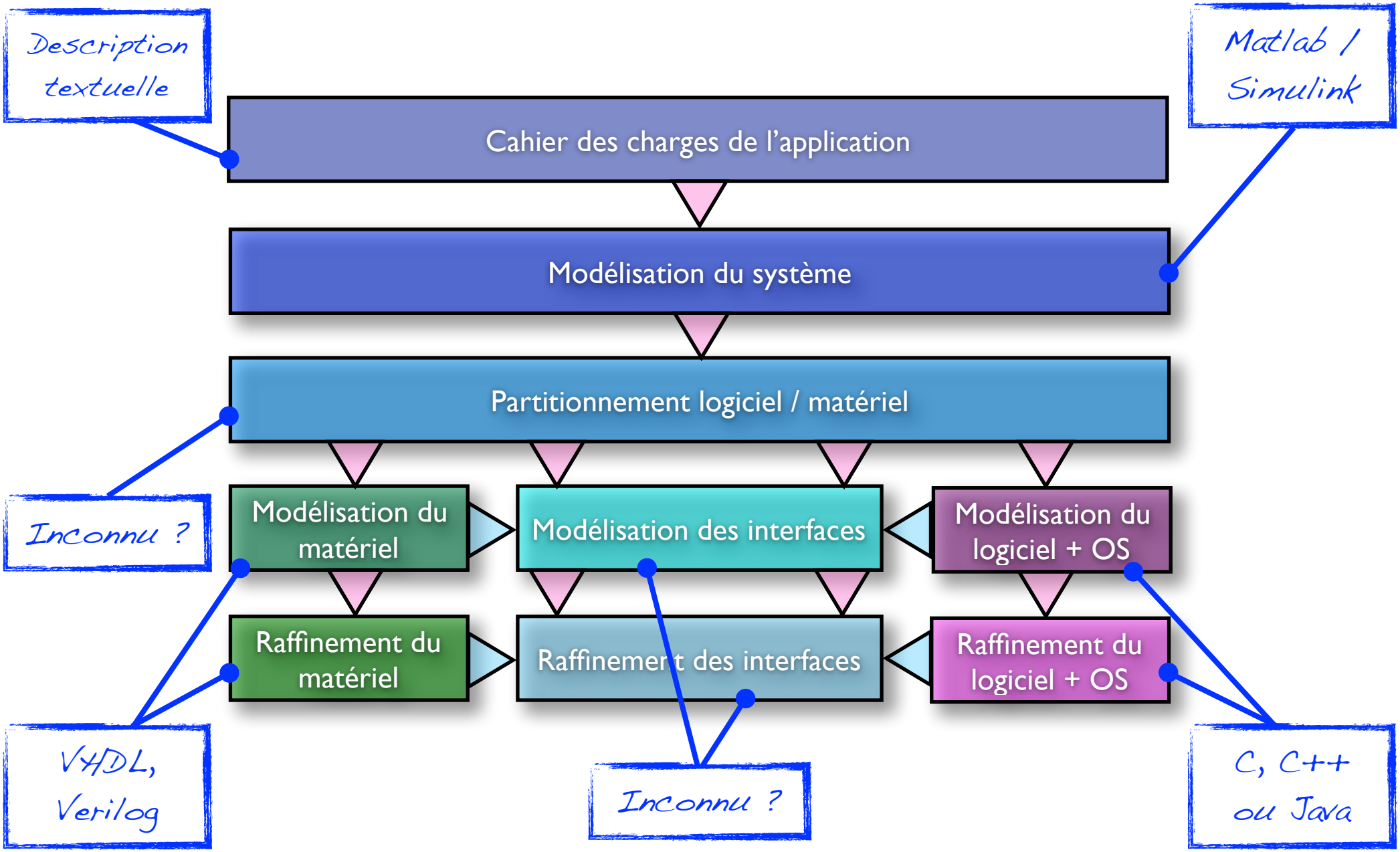
Synthèse du matériel

Co-vérification

Compilation du logiciel

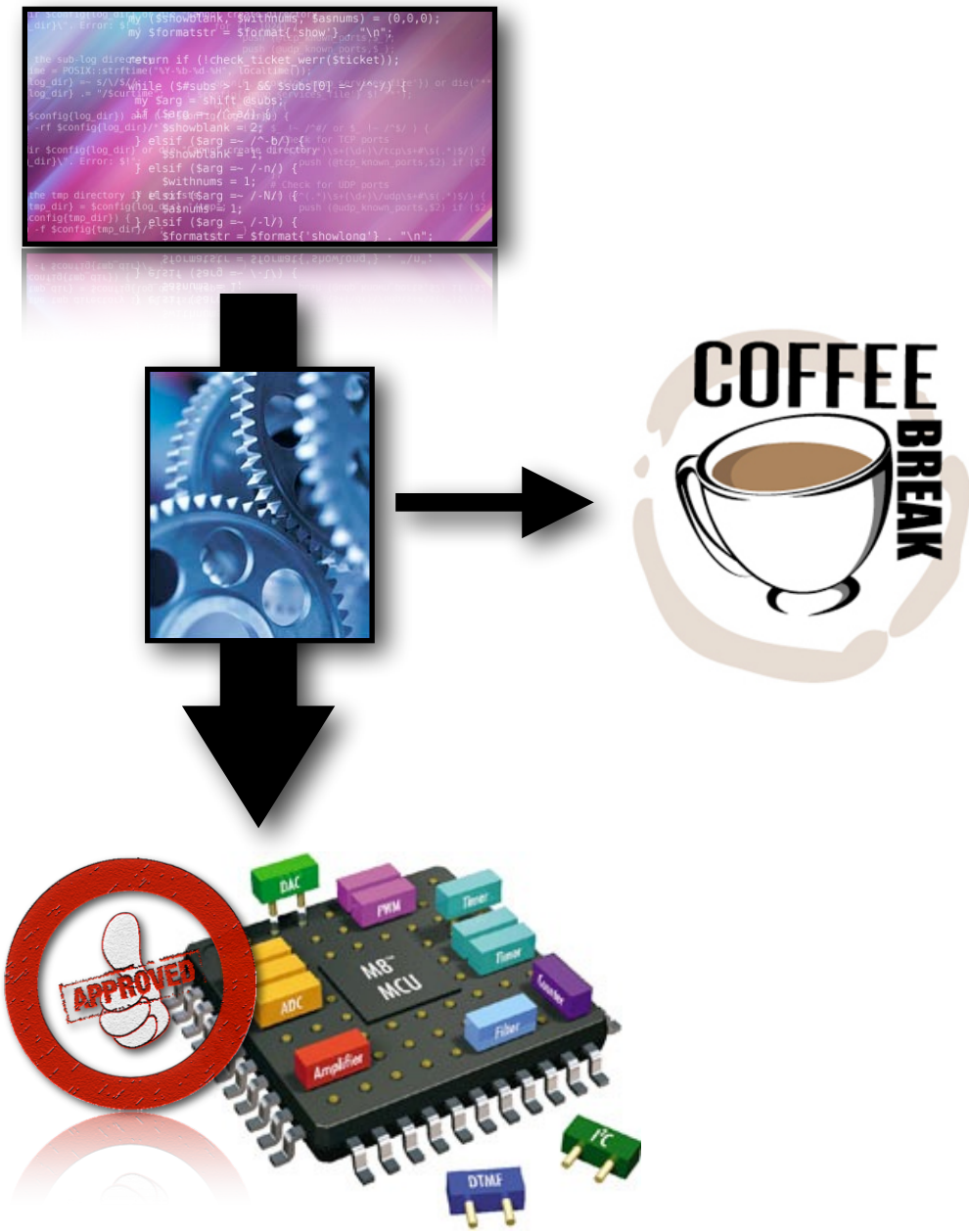
Intégration système

Changement des langages de modélisation durant le flot

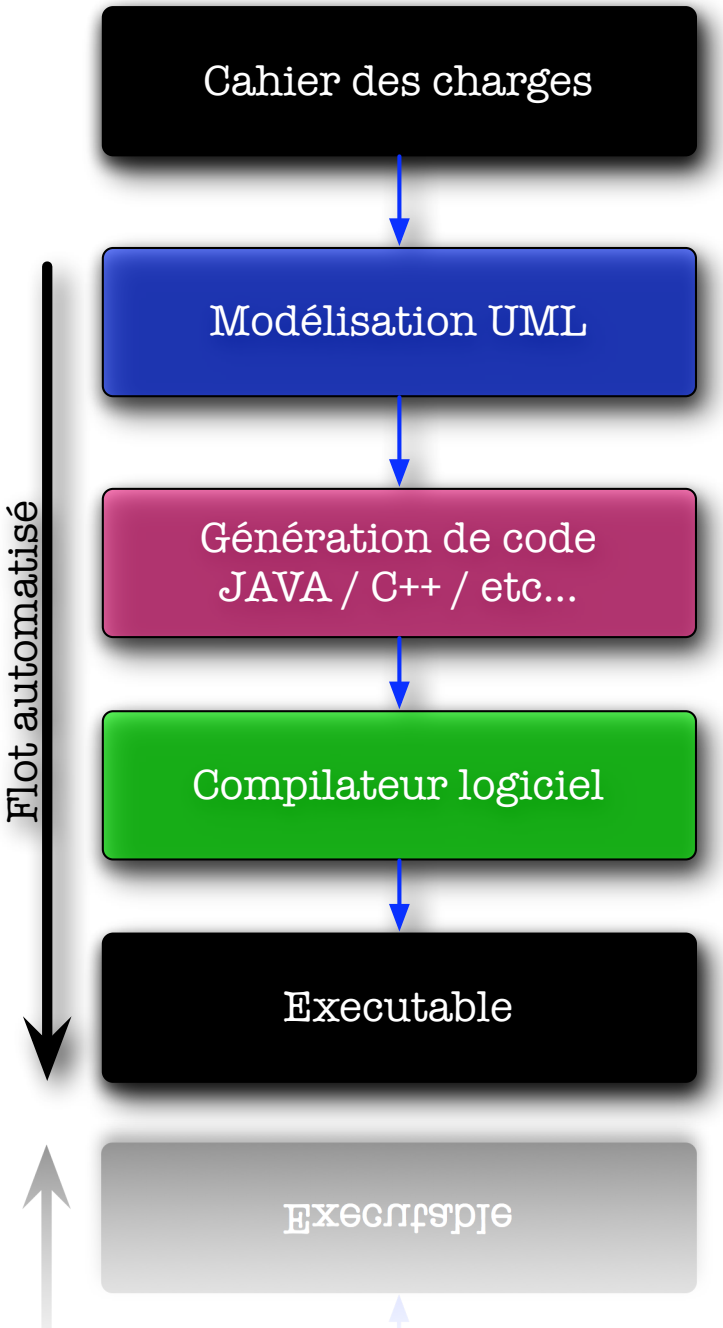


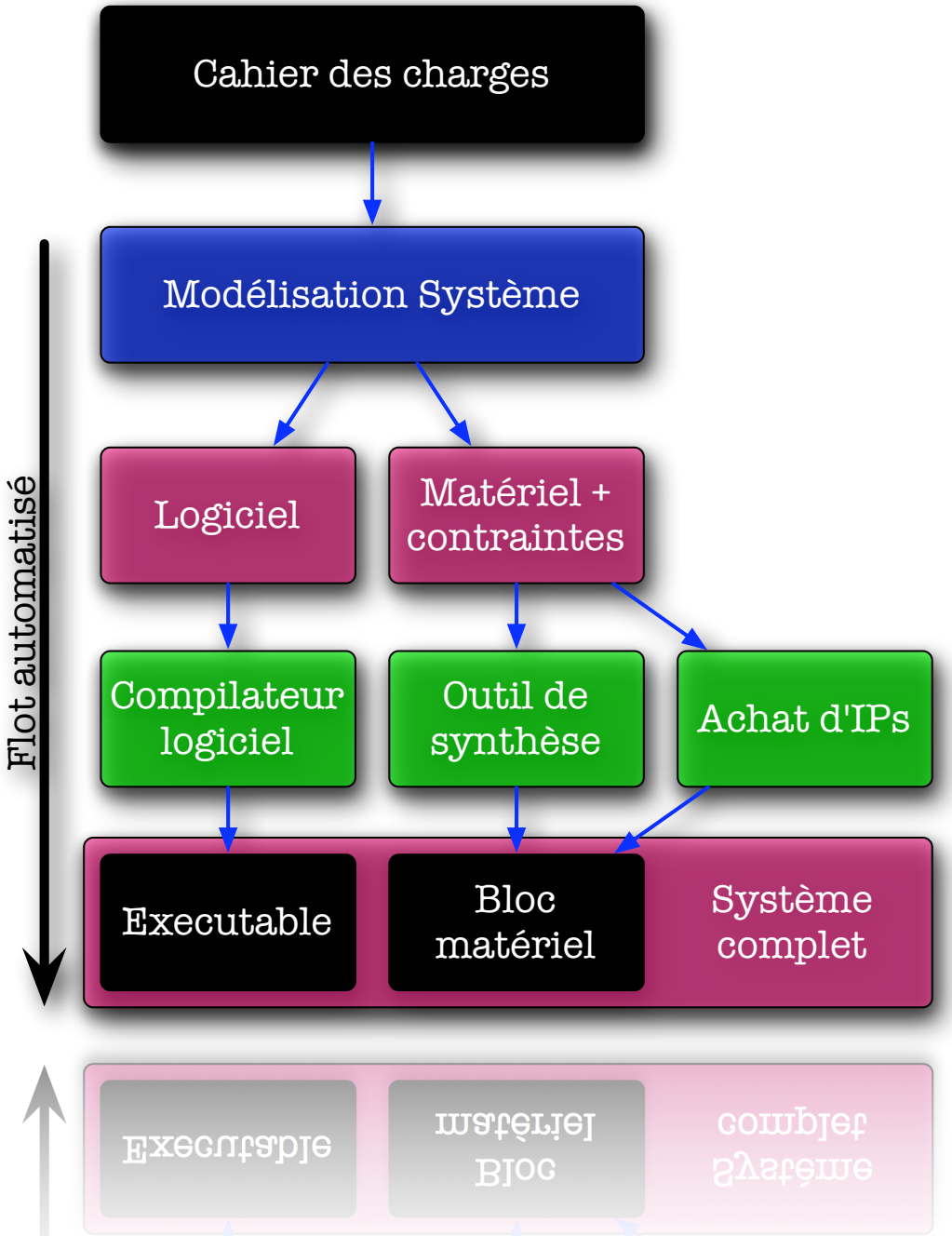
L'automatisation des processus

- Avantages de l'automatisation,
  - Gain de temps (développement),
  - Gain de temps (vérification),
  - Exploration d'espaces de solutions difficiles réalisable manuellement,
  - Remonter le niveau d'abstraction => confier de nouvelles taches aux hommes,
- Inconvénients de l'automatisation,
  - Perte de la maîtrise technique des processus (confiance aveugle à l'outil),
  - Diminution des performances des composants ainsi générés (pas une généralité),



Automatisation des flots de conception

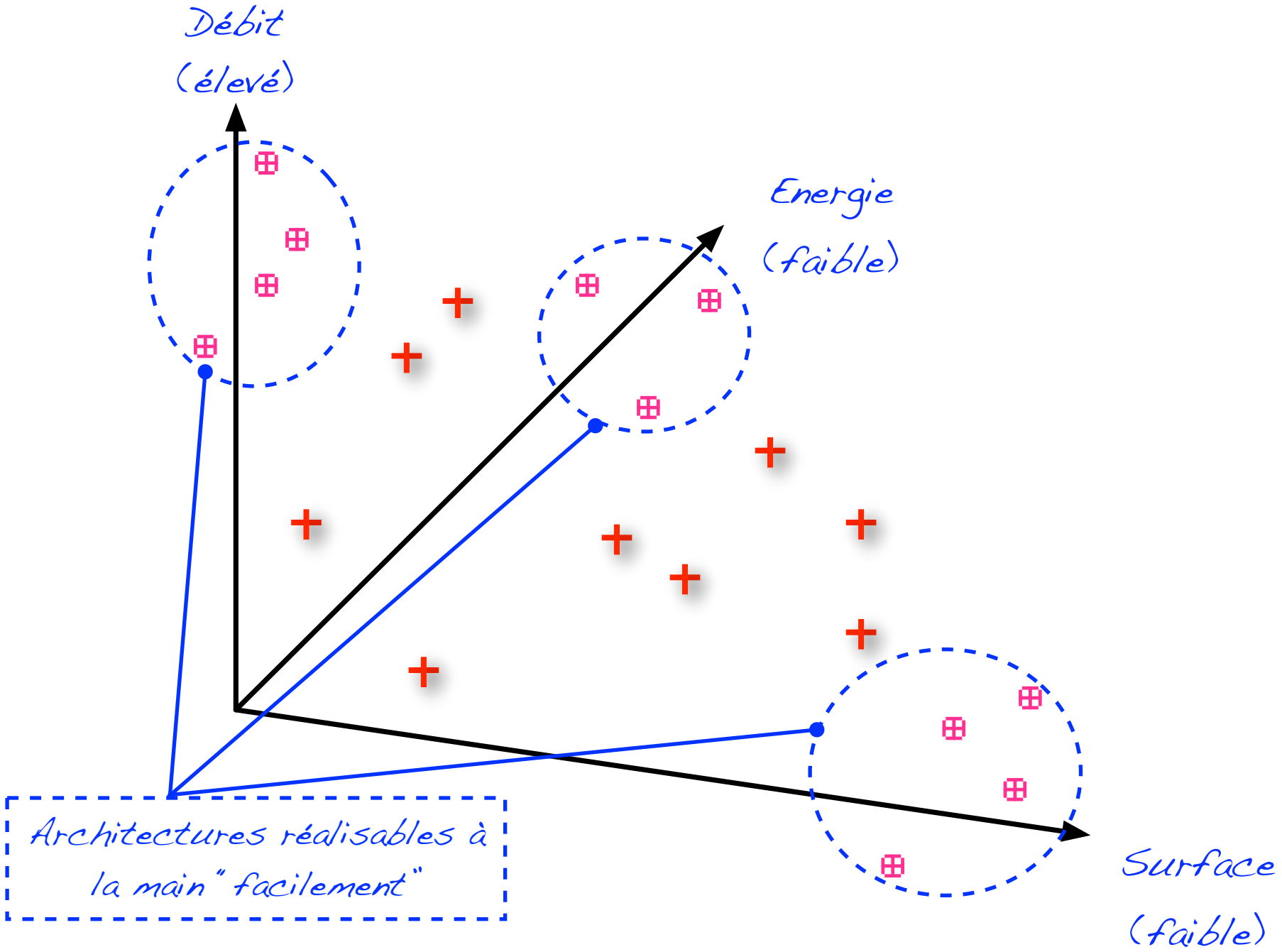




Pourquoi générer automatiquement des archis ?

- Réduction du temps de conception du circuit,
  - FFT développée à la main = plusieurs jours / semaines,
  - A l'aide d'un outil, cela prend quelques secondes...
- Exploration de l'espace des solutions (meilleur compromis)
- Réduction de la taille des spécifications
  - Taille du code / 10 vis-a-vis du RTL => moins d'erreurs,
  - Codage du circuit plus naturel (lien plus fort avec l'algorithme),
  - Simulation plus rapide qu'à bas niveau d'abstraction,
- Technique indépendante de la technologie
- Prise en considération des contraintes à plus haut-niveau
  - Consommation d'énergie,
  - Performances temporelles, etc.

Exploration de l'espace des solutions

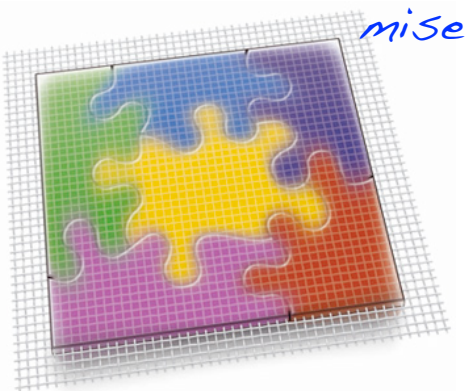


Introduction aux composants virtuels (IP)

- Le “time to market” implique une conception plus rapide de systèmes plus complexes,
- Réutiliser l'existant afin de gagner du temps à la conception et à la vérification,
- Développer des composants évolutifs qui serviront dans le circuit courant et dans d’autres,
- Les composants virtuels peuvent être développés en interne ou achetés à l’extérieur.

Ensemble de composants pré-existants

Sélection, assemblage et mise au point



Circuit = Agglomérat de composants développés sur mesure et/ou pré-existants

Les différents niveaux d'abstraction des IP

- Les IPs Softs
  - Ces composants sont disponibles sous forme de codes sources,
  - Ils sont très génériques et réutilisables mais peu prévisible (pas de cible),
- Les Firm IPs
  - Ces composants sont disponibles sous forme de NetList (code ASM),
  - Ils sont moyennement génériques mais plus prévisible (lien avec la cible),
- Les IPs Hard
  - Ces composants sont disponibles sous forme de plan de masse (code binaire)
  - Non modifiable, mais connaissance précise de leurs performances.

	Soft IP Not Predictable Very Flexible	Firm IP Predictable Flexible	Hard IP Very Predictable Not Flexible
Design Flow	System Design RTL Design	Floor Planning Synthesis Placement	Routing Verification
Representation	Behavioral RTL	RTL & Block Netlist	Polygon Data
Libraies	N/A	Reference Library - Footprint - Timing Model - Writing Model	Process Specific Library & Design Rules
Technology	Technology Independent	Technology Generic	Technology Fixed
Portability	Unlimited	Library Mapping	Process Mapping

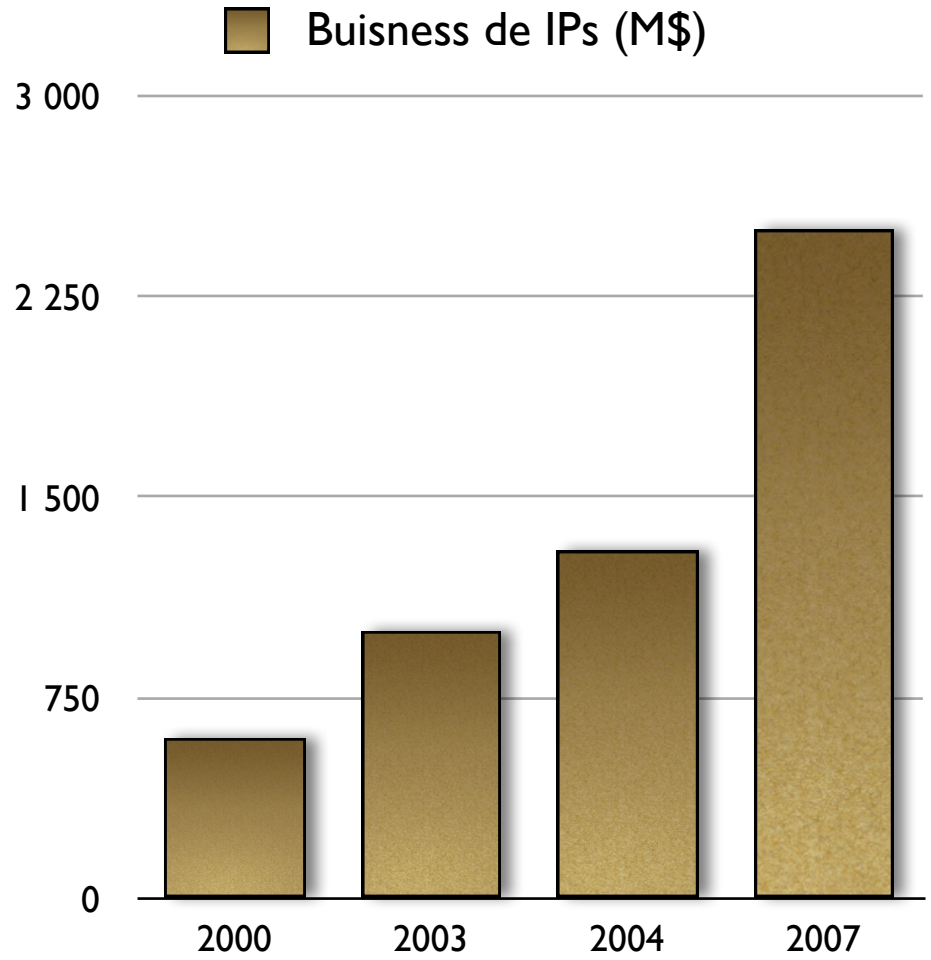
Exemple d'un IP (Compression JPEG 2k)

- Intopix IPX-JHPD
  - Ondelettes 3/5 & 9/5
  - Décomposition 0 à 6
  - Profondeur 8 à 18 bits
  - Format d'entrée 1920x1080
  - Flux d'entrée 500Mbps
  - Mémoire externe 64Mo
  - Cibles conseillées Virtex 5 LXT/SXT
- Business
  - Modèles VHDL pour ModelSim
  - Achat de l'IP 100/200k€
  - Cartes comp/décomp, env. 10k€

- Barco Silex
  - Ondelettes 3/5 & 9/5
  - Décomposition 0 à 6
  - Profondeur 12 bits
  - Taille des codes blocs 32x32
  - Format d'entrée 4096x2160
  - Flux d'entrée 250/500Mbps
  - Mémoire externe 384Mo
  - Cibles conseillées Virtex 5 LXT85T
- Business
  - Location de l'IP (carte) 5k€ par mois
  - Achat de l'IP 100/200k€

Exemples de composants virtuels (IP)

- Il est possible de trouver en quelques clics des architectures fonctionnelles pour :
  - La compression, décompression vidéo (MPEG-X, h264, etc.),
  - Des contrôleurs de périphériques (USB, TCP/IP, SATA, PCI, etc.),
  - Des fonctions mathématiques (FFT, Viterbi, RS, FIR, etc.),
  - Des processeurs généralistes ou DSP (MicroBlaze, ARM, etc.),
  - Systèmes processeurs + système d'exploitation temps réel,
- Les prix sont généralement à la demande avec des Royalties...



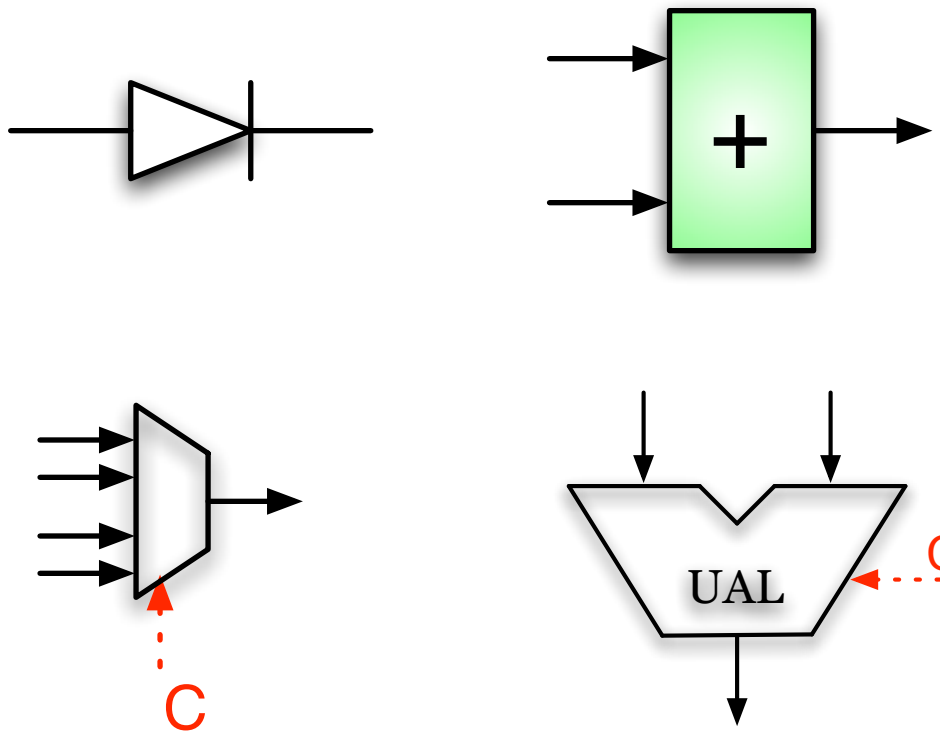
Augmentation de l'utilisation d'IPs dans les FPGA (les IPs pour FPGA deviendraient la source de revenu principal du marché des IPs (Gartner))

Les contraintes liées à l'utilisation des IPs

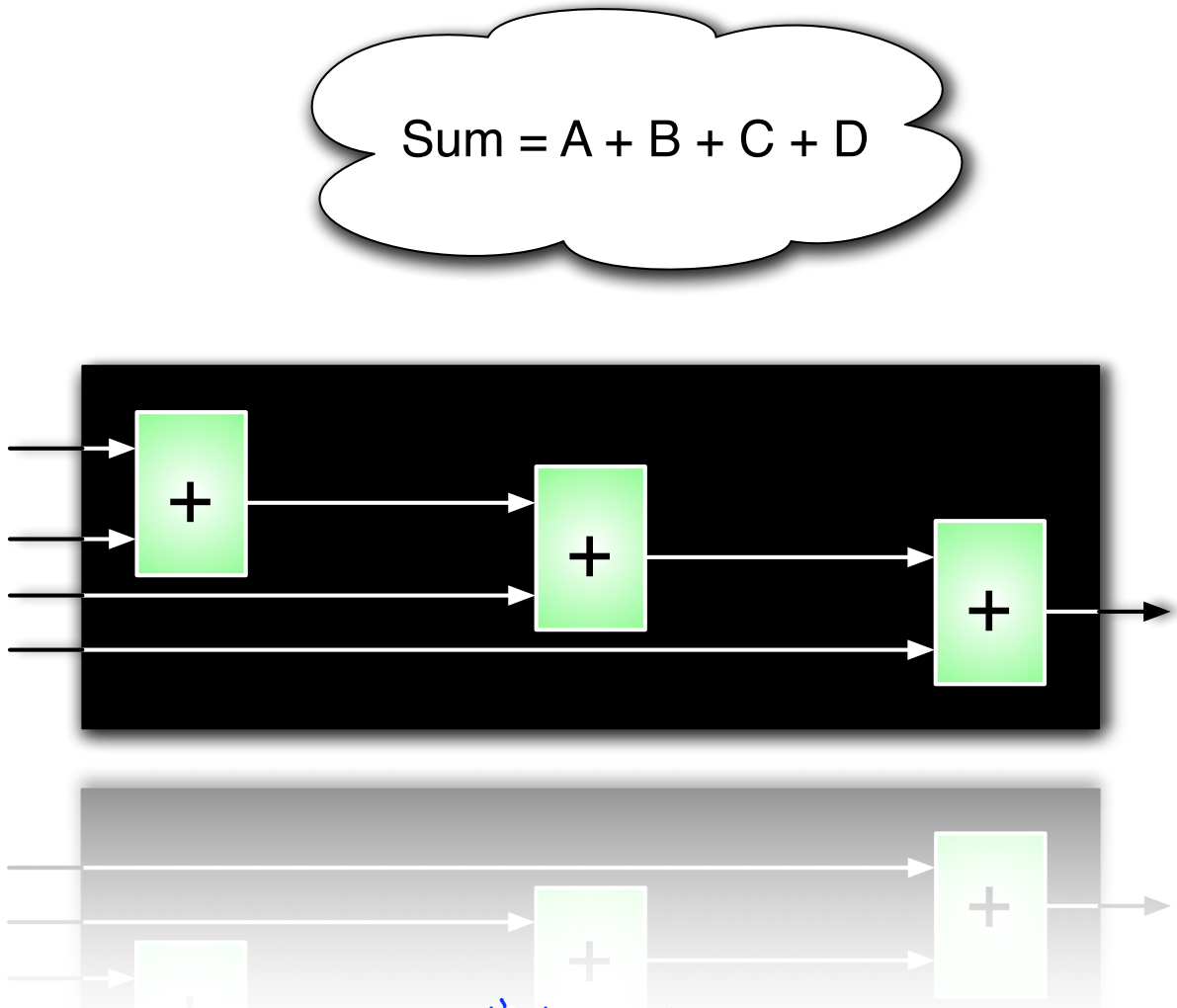
- La recherche et la comparaison est fastidieuse !
  - ➡ Problèmes de référencement des IPs,
  - ➡ Informations et niveaux d'abstraction différents,
- Evaluation de la qualité et de la validité d'un IP complexe,
  - ➡ Comment vérifier sa pertinence avant de l'acheter ?
  - ➡ Comment être sur qu'il est intégralement fonctionnel ?
- Pour un vendeur, comment livrer assez d'informations sans pour autant risquer la copie ?

Comment construit on un circuit numérique ?

- Tous les circuits numériques ne sont que des assemblages :
  - Opérateurs mathématiques, logiques, registres, fils de registres, multiplexeur, tri-states, bus, etc.
- Un Pentium (X) n'est qu'un simple assemblage de composants électroniques élémentaires (mûrement réfléchi tout de même),
- L'efficacité du circuit dépend du choix des ressources et de leur l'interconnexion,



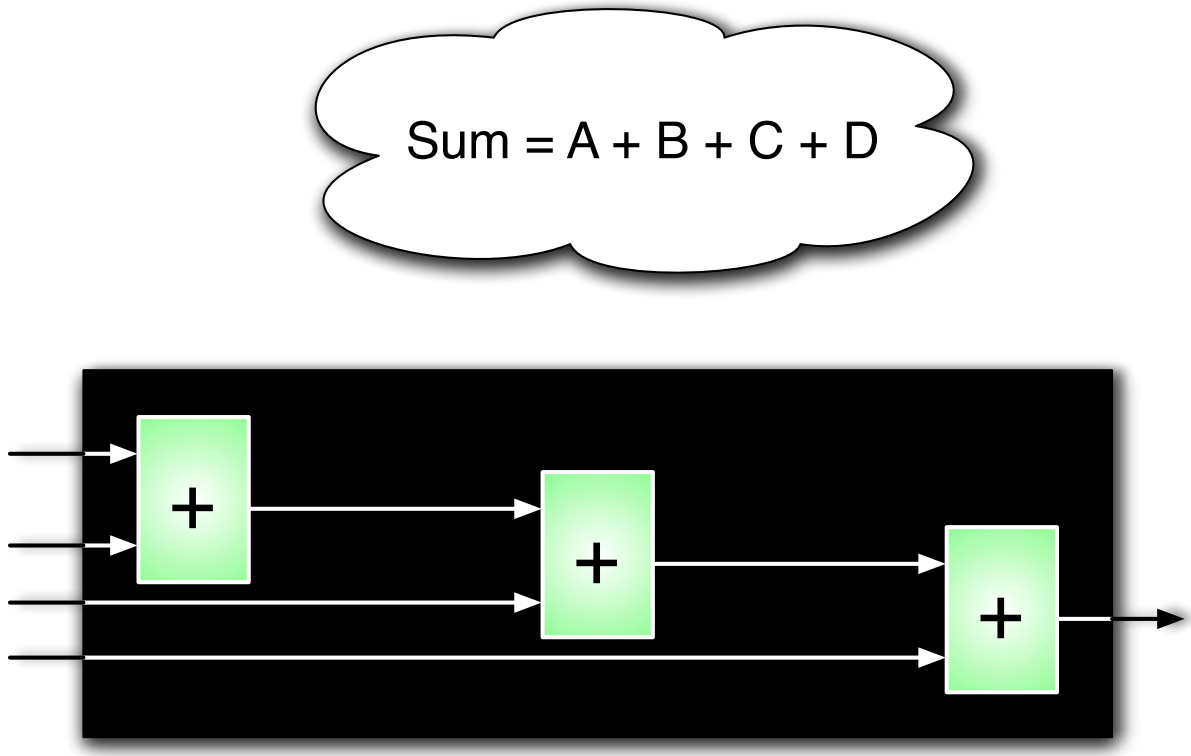
Les architectures combinatoires



Pour chaque opération contenue dans l'algorithme on va instancier une ressource matérielle dans l'architecture matérielle d'implémentation.

Les architectures combinatoires possèdent une latence faible, une faible consommation, mais leur cadence est elle aussi faible !

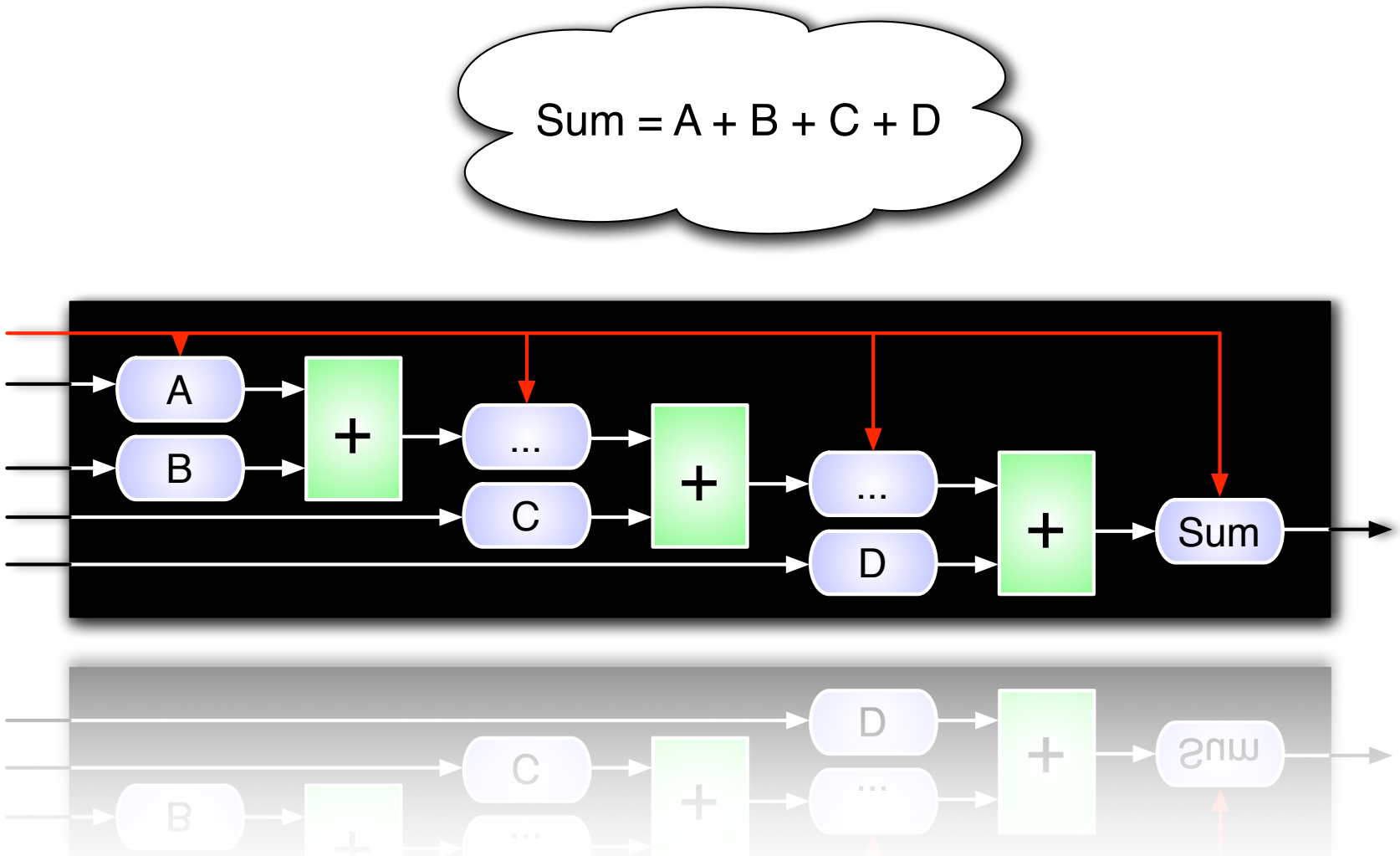
Les architectures combinatoires



Si le chemin critique trouvé dans ce circuit combinatoire est de 30ns,

- Quelle est sa fréquence (horloge) de fonctionnement maximale ?
- Quelle est la latence du circuit combinatoire ?
- Quel est le débit pratique du circuit ?
- Quel est son coût matériel ?

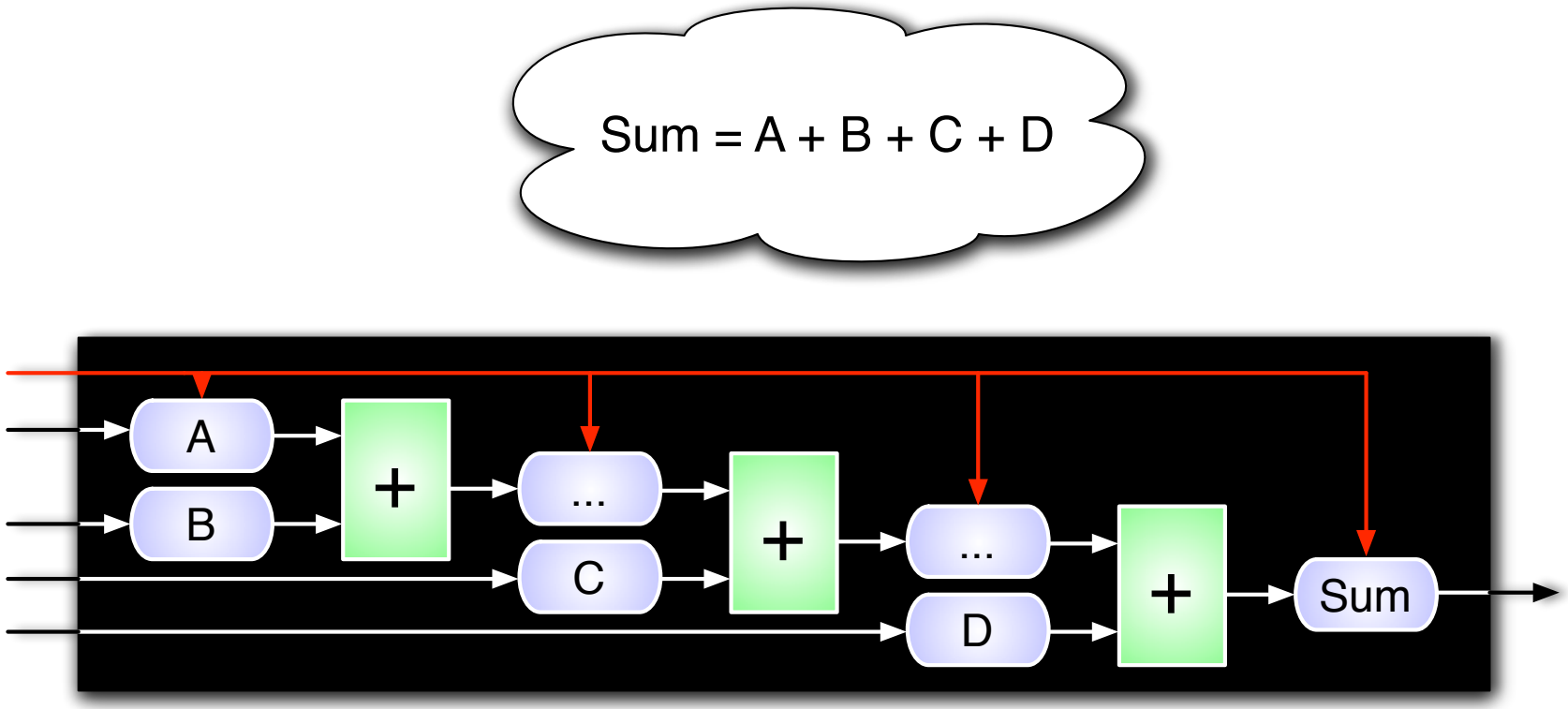
Les architectures dites "pipeline"



L'introduction de registres aux points critiques du circuit va permettre de "casser" les chemins critiques et donc d'augmenter la fréquence d'horloge.

Les architectures pipelines sont plus onéreuses en surface et consommation mais offrent des débits très élevés !

Les architectures dites “pipeline”



Si le chemin critique trouvé dans ce circuit est de (10+1)ns,

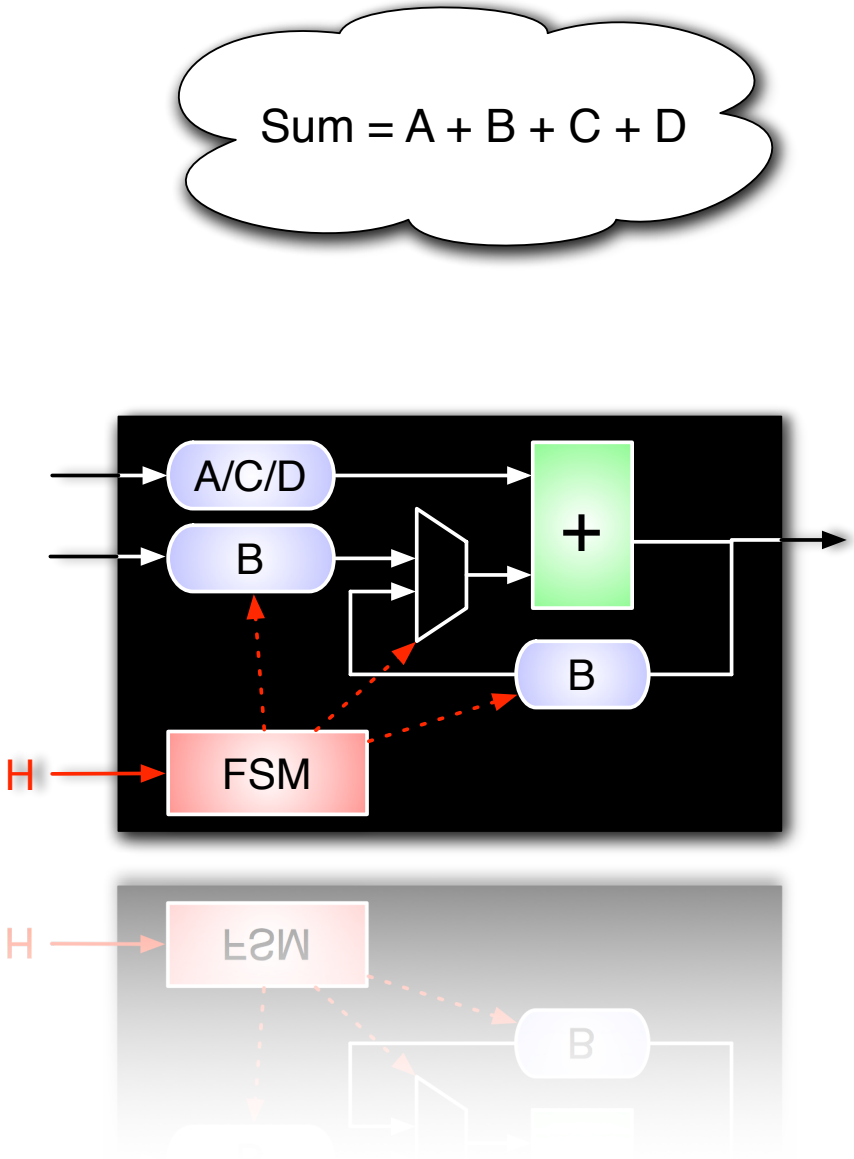
- Quelle est sa fréquence (horloge) de fonctionnement maximale ?

- Quelle est la latence du circuit ?

- Quel est le débit pratique du circuit en fonctionnement ? à l'origine ?

- Quel est son coût matériel ?

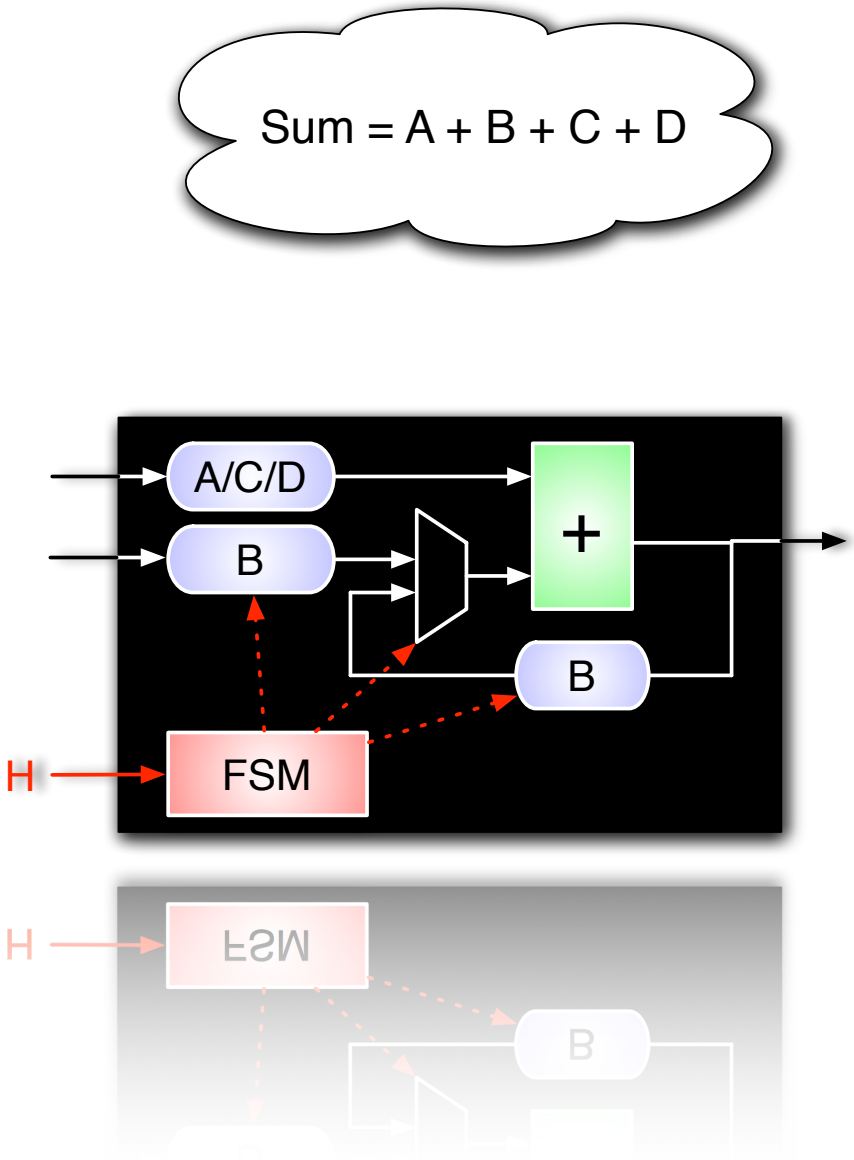
Les architectures séquentielles



En fonction des contraintes imposées lors de l'implémentation, on va pouvoir "jouer" sur l'architecture du circuit : nombre d'éléments, interconnexions entre les éléments, types des ressources...

Les calculs sont réalisés de manière séquentielle avec réutilisation des ressources. Ce réduit le cout d'implantation mais aussi la latence et la cadence du système.

Les architectures séquentielles

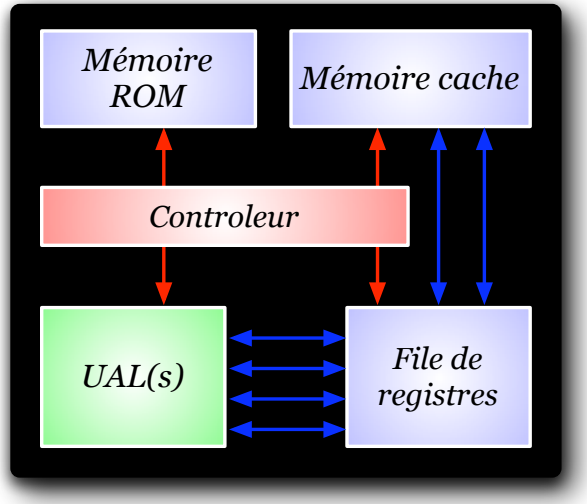


Si le chemin critique trouvé dans ce circuit est de (10+2)ns et que le contrôleur est composé de 3 états séquentiels,

- Quelle est sa fréquence (horloge) de fonctionnement maximale ?
- Quelle est la latence du circuit ?
- Quel est le débit pratique du circuit en fonctionnement ?
- Quel est son coût matériel ?

Les architectures de processeurs généralistes

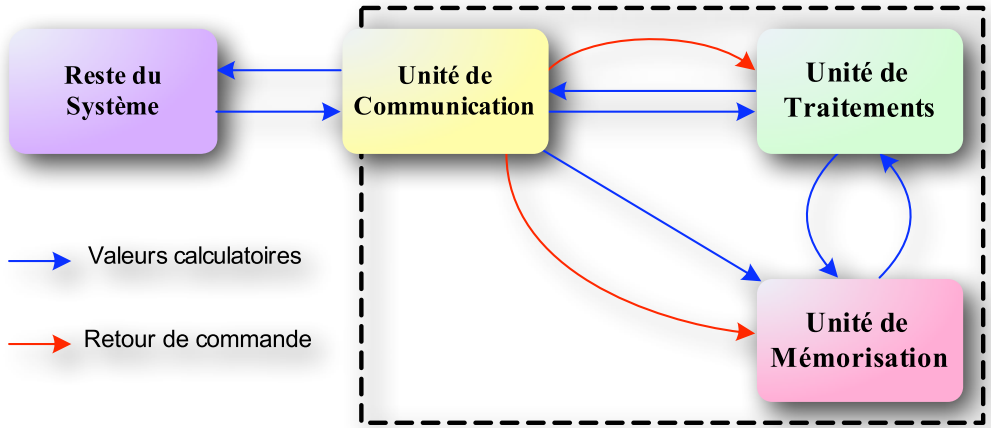
- Ces architectures de µP sont peu flexibles mais très simplement programmables,
- Il est possible de facilement rajouter des ressources,
  - Opérateurs de calculs,
  - Ressources de mémorisation,
- Augmentation rapide de la complexité du contrôleur,
  - Baisse des performances globales du circuit,
  - Un compilateur pour la cible matérielle,

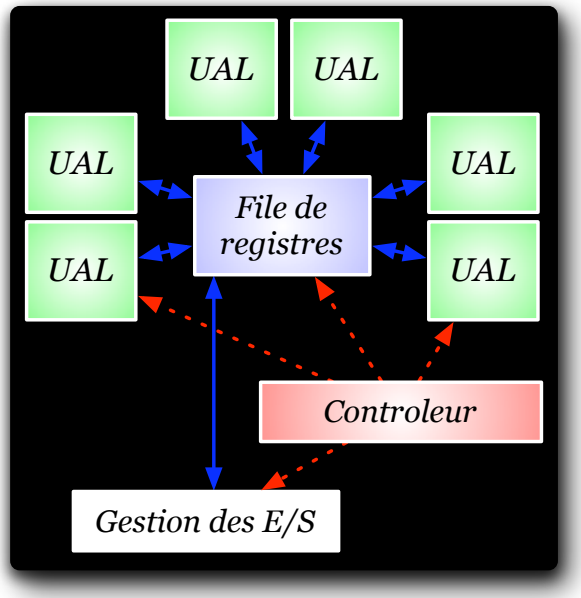


Die d'un Pentium 3

Architecture classique d'un coeur de DSP

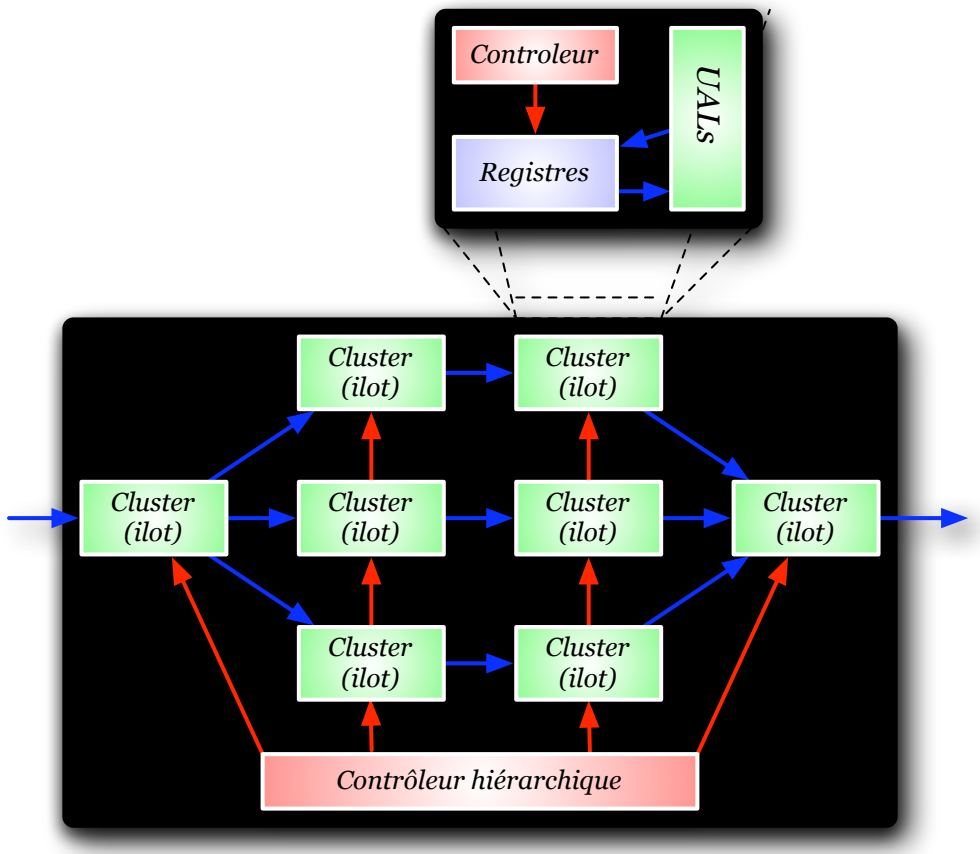
- Ces architectures très flexibles car elles sont construites en fonction des besoins réels,
  - Ressources matérielles,
  - Canaux de communication,
- Elles sont généralement composées de 3 pôles :
  - La gestion des calculs,
  - La gestion des communications,
  - La mémorisation des données,
- Cette flexibilité va de paire avec la complexité de mise en oeuvre de telles architectures.





Le cas particulier des îlots de calculs

- Dans le cadre d'applications bien particulières !
- Généralement employé pour des nids de boucles,
- Les clusters ne sont pas nécessairement tous identiques,
  - ➡ Mais généralement ils implémentent tous les mêmes fonctions,
- Le coût de ces architectures haute perf. est prohibitif,
- Inadaptées à la majorité des applications réelles.




```
for(i=0; i<256; i++){
    C[i] = A[i] + B[i];
}
for(i=0; i<256; i++){
    D[i] = C[i] + 4;
}
```

Exemple complexe... équation polynomiale (degré 2)

- Vous allez maintenant définir les architectures des circuits matériels réalisant une équation polynomiale d'ordre 2. L'équation générale vous est rappelée ci-dessous :
  - y = a.x2 + b.x + c
- Vous ferez l'hypothèse que les facteurs "a", "b" et "c" sont des constantes vues de l'extérieur du circuit.
- Nous considérons dans les questions suivantes que les composants matériels possèdent les caractéristiques suivantes:
  - ADD = 4ns / 32 slices
  - MUL = 8ns / 94 slices
  - REG = instantané / 16 slices
  - MUX = instantané / 8 slices

Architecture combinatoire - design

Architecture combinatoire - caractéristiques

- Coût en surface de l'architecture
- Latence de traitement des échantillons
- Cadence maximum de fonctionnement

Architecture pipeline - design

Architecture pipeline - caractéristiques

- Coût en surface de l'architecture
- Latence de traitement des échantillons
- Cadence maximum de fonctionnement

Architecture contrainte en surface - design

Architecture contrainte en surface - caractéristiques

- Coût en surface de l'architecture
- Latence de traitement des échantillons
- Cadence maximum de fonctionnement

Architecture contrainte à “n” cycles d’horloge

Architecture contrainte à n cycles d'horloge - caractéristiques

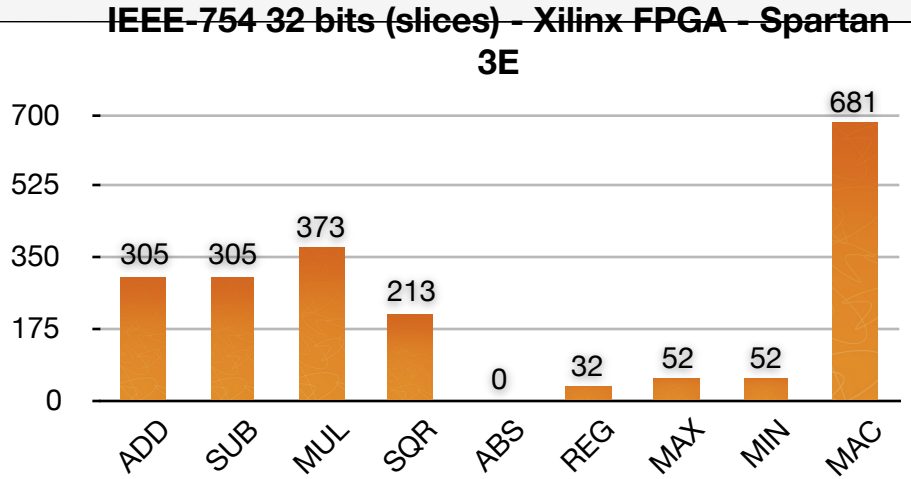
- Coût en surface de l'architecture
- Latence de traitement des échantillons
- Cadence maximum de fonctionnement

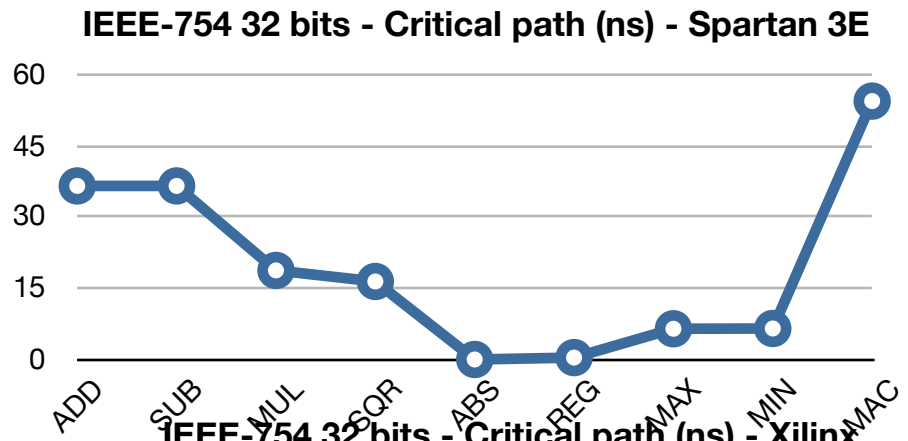
Et maintenant que se passe t'il si l'on change de technologie ?

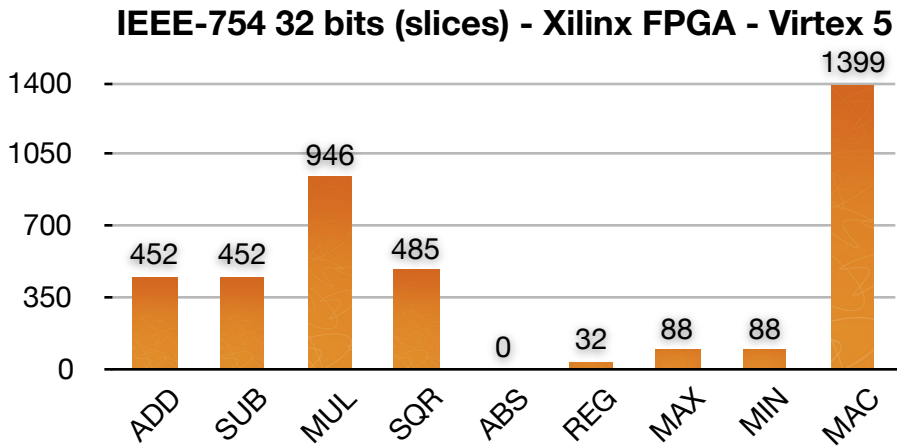
- Maintenant nous considérons que les caractéristiques réelles des composants sont les suivantes :
  - ADD = 4ns / 32 slices
  - MUL = 8ns / 94 slices
  - REG = 1ns / 16 slices
  - MUX = 2ns / 8 slices

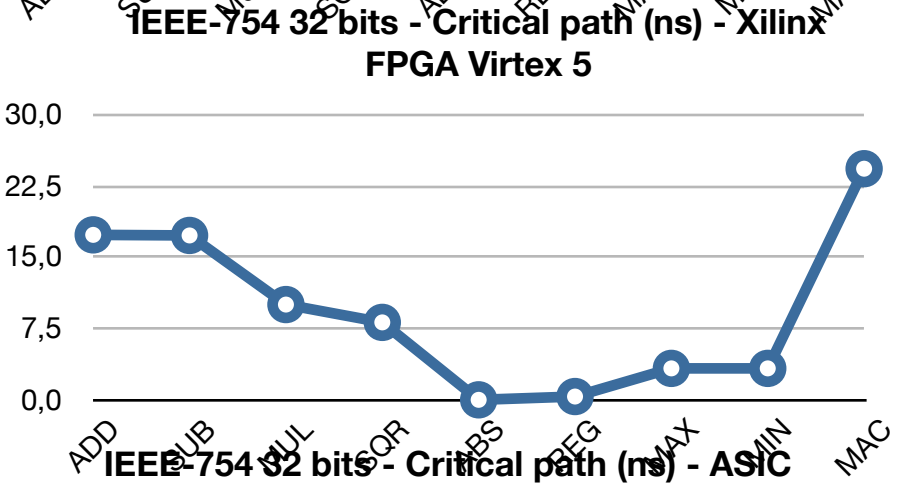
- Réévaluez les caractéristiques des architectures que vous avez conçues dans les parties précédentes

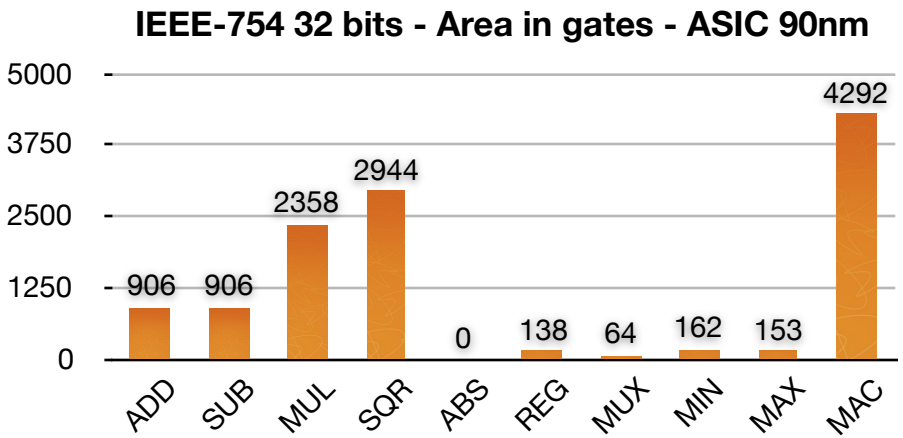
Repassons dans le monde réel...

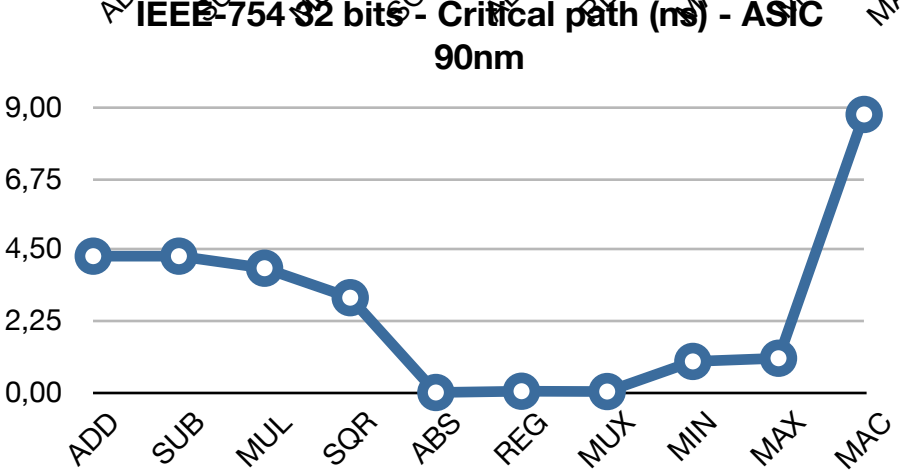












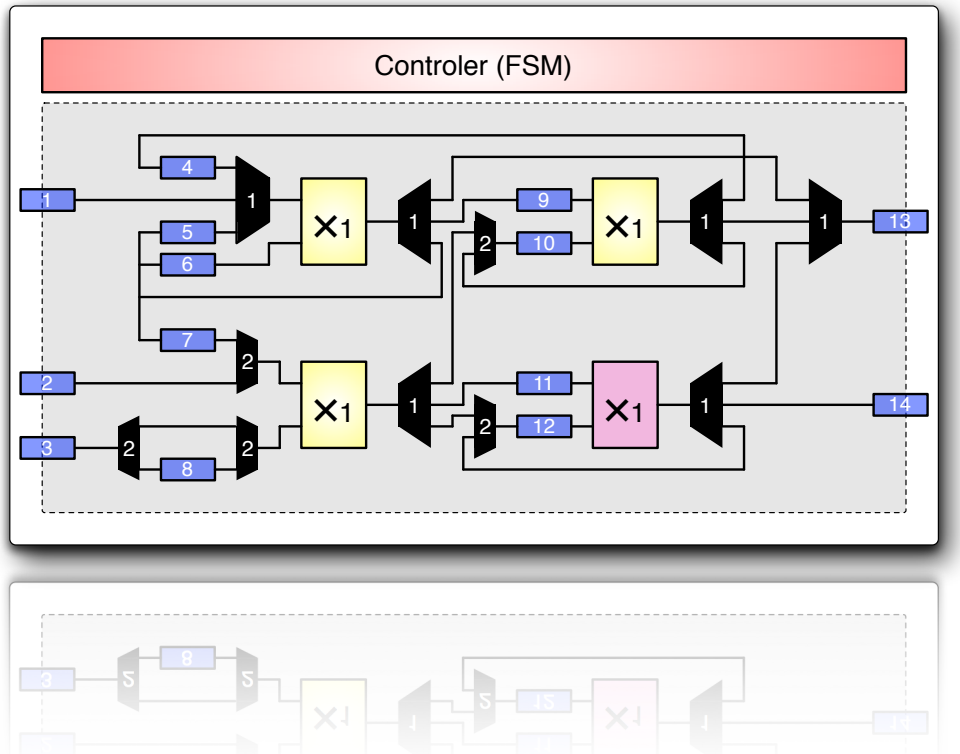
Conception manuelle - Bilan & Conclusion

$$\delta = \sum_{y=0}^{3} \sum_{x=0}^{3} abs(I_1(x,y) - I_2(x,y))$$

Comment peut on passer d'un algorithme à la réalisation d'un circuit "complexe" contraint par la surface, la consommation de puissance, etc.

Chaque réalisation répond à un jeu de contrainte particulier... rien n'assure que le circuit répondra aux prochains besoins...

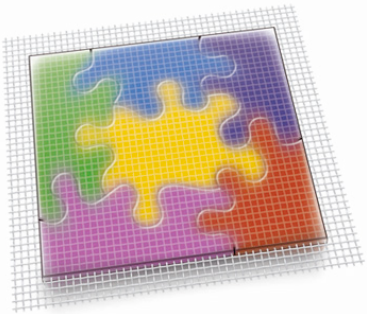
Le choix de la technologie d'implantation est cruciale car elle influe beaucoup sur les résultats !



La communication entre les blocs, un réel problème ?

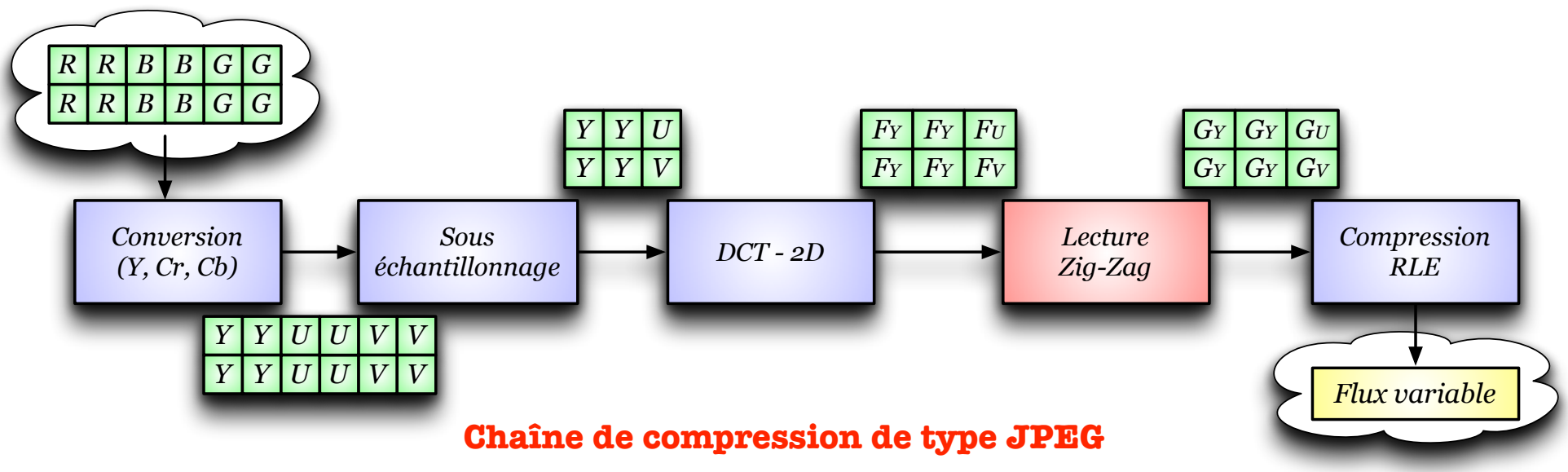
- Un circuit est divisé en sous-élements (développés en interne ou achetés),
- Généralement impossible des les interconnecter directement,
  - Timing des E/S différents,
  - Taille et norme des bus différentes,
  - Fréquences d'horloge différentes,
  - Adaptation générique aux bus,
- Il est nécessaire de prendre soin aux interconnexions des composants,

Sélection des composants et développement de la "glue"



La communication avec le système

- Il ne faut pas oublier que l'architecture générée fait partie intégrante d'un système de complexité supérieure. Il est donc important de considérer les contraintes système lors de la phase de conception,
  - Les autres circuits peuvent posséder des contraintes très fortes sur la production et la consommation des données,
  - Les contraintes de consommation et de production des données ne sont pas toujours compatibles,
  - Ces problèmes de communication réduisent les performances globales du système,

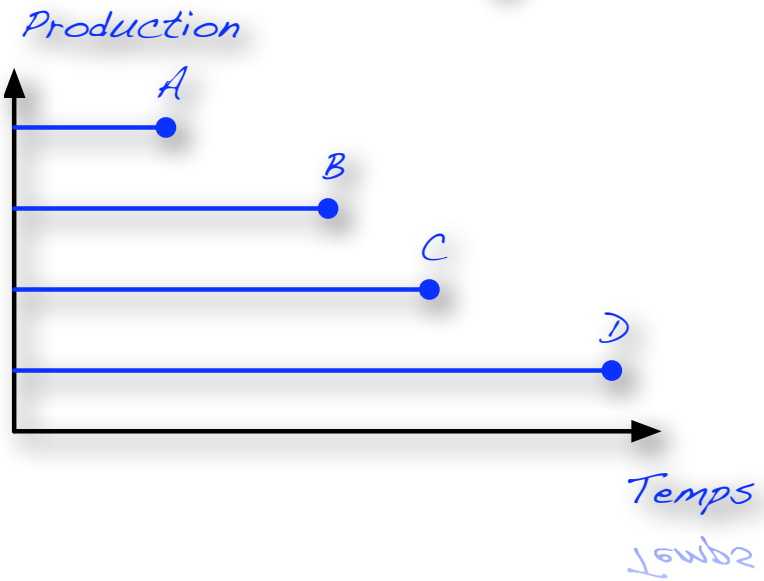


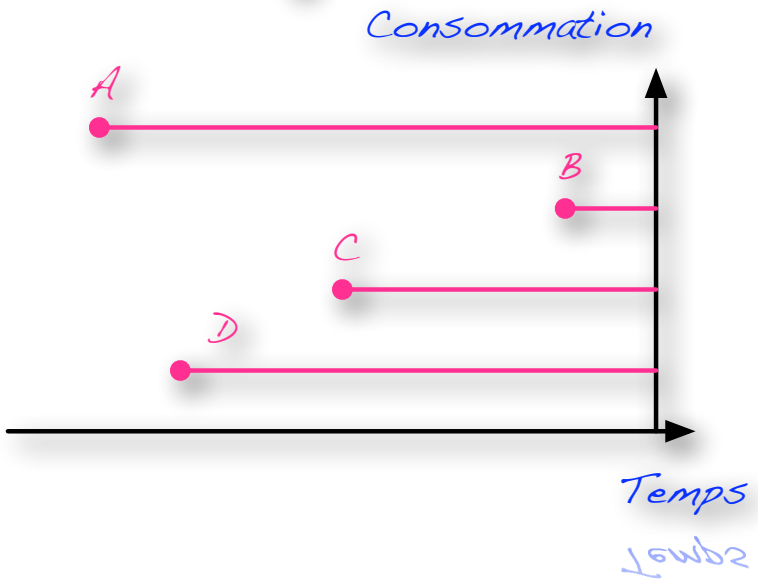
Chaîne de compression de type JPEG

La problématique de la communication (1)

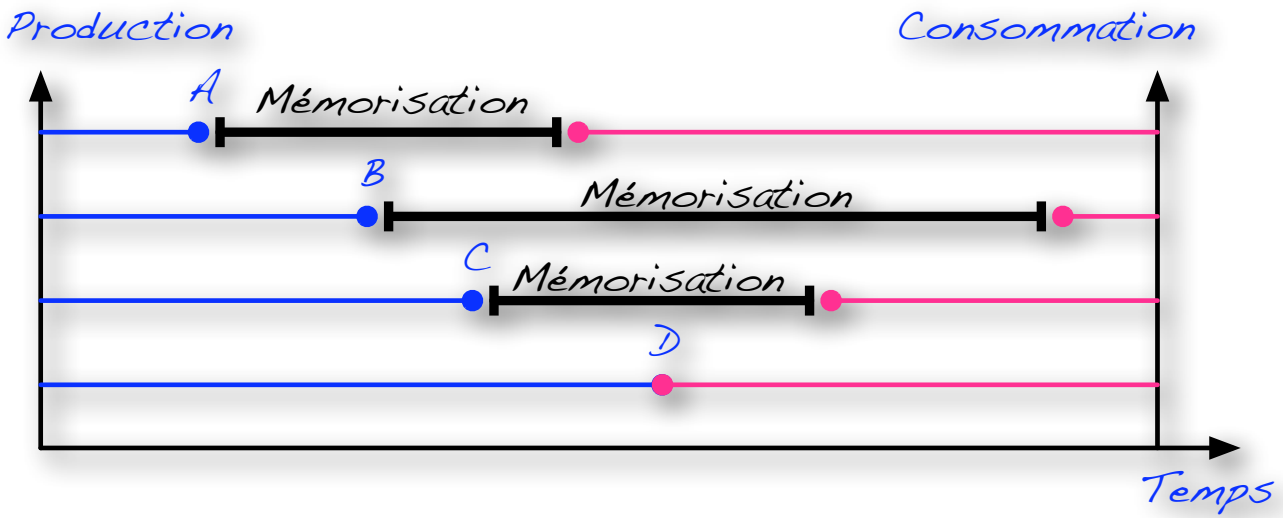
Extraction du timing des sorties du composant (A).

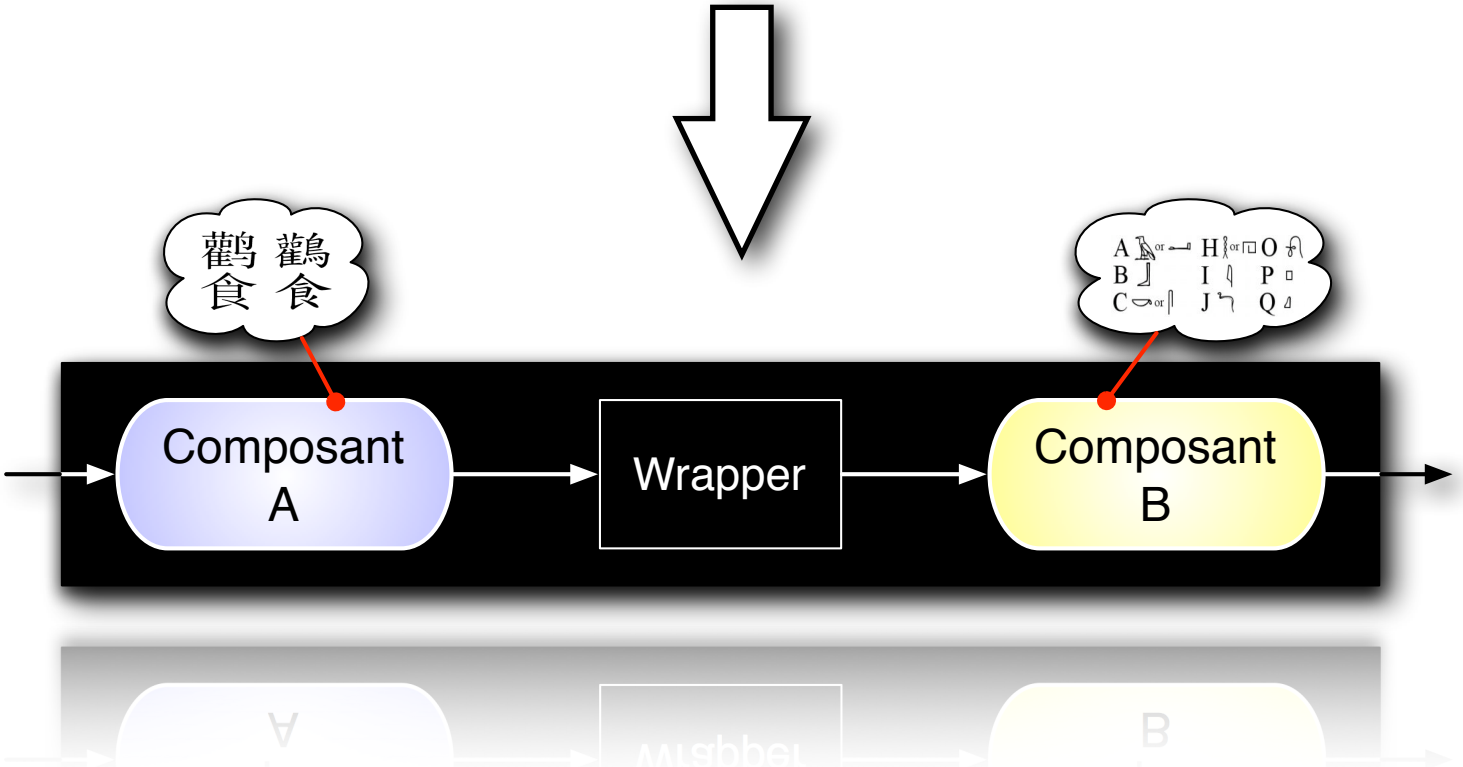
Extraction du timing des entrées du composant (B).





Les solutions existantes - les wrappers

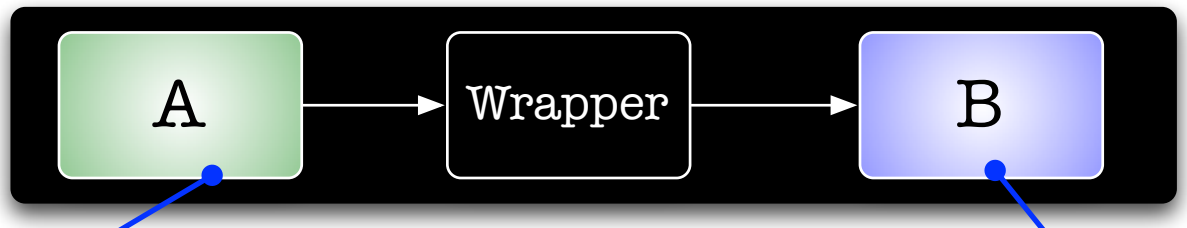




Les wrappers font chuter les performances globales du système (surface, consommation, latence)

Exercice n°1 : Packaging des données

- Nous souhaitons interconnecter 2 composants matériels dont les interfaces sont spécifiées dans la figure ci-dessous,
  - Décrivez une solution matérielle permettant de gérer les problèmes entre les protocoles des 2 circuits,
  - Quel impact votre solution à t'il sur l'implantation globale de l'application ?
  - Quelle aurait pu être une autre solution ?

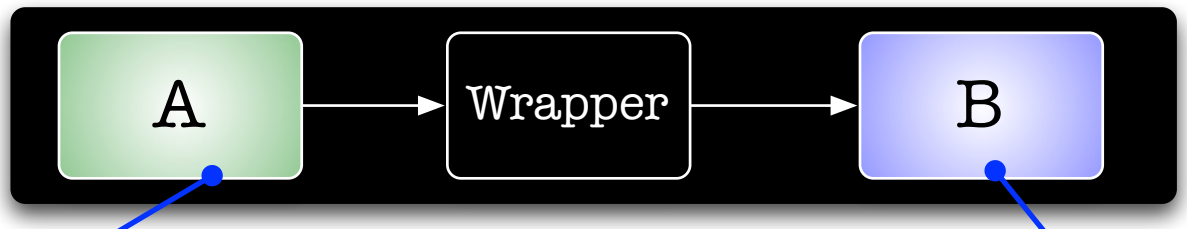


Le composant A produit ses données (X1, X2, X3, X4) codées sur 8 bits à hauteur de 4 données par cycle

Le composant B consomme les données (X1, X2, X3, X4) sur 16 bits à hauteur de 1 donnée par cycle

Exercice n°2 : Permutation de l'ordre

- Nous souhaitons interconnecter 2 composants matériels dont les interfaces sont spécifiées dans la figure ci-dessous,
  - ➡ Décrivez une solution matérielle permettant de gérer les problèmes entre les protocoles des 2 circuits,
  - ➡ Quel impact votre solution à t'il sur l'implantation globale de l'application ?

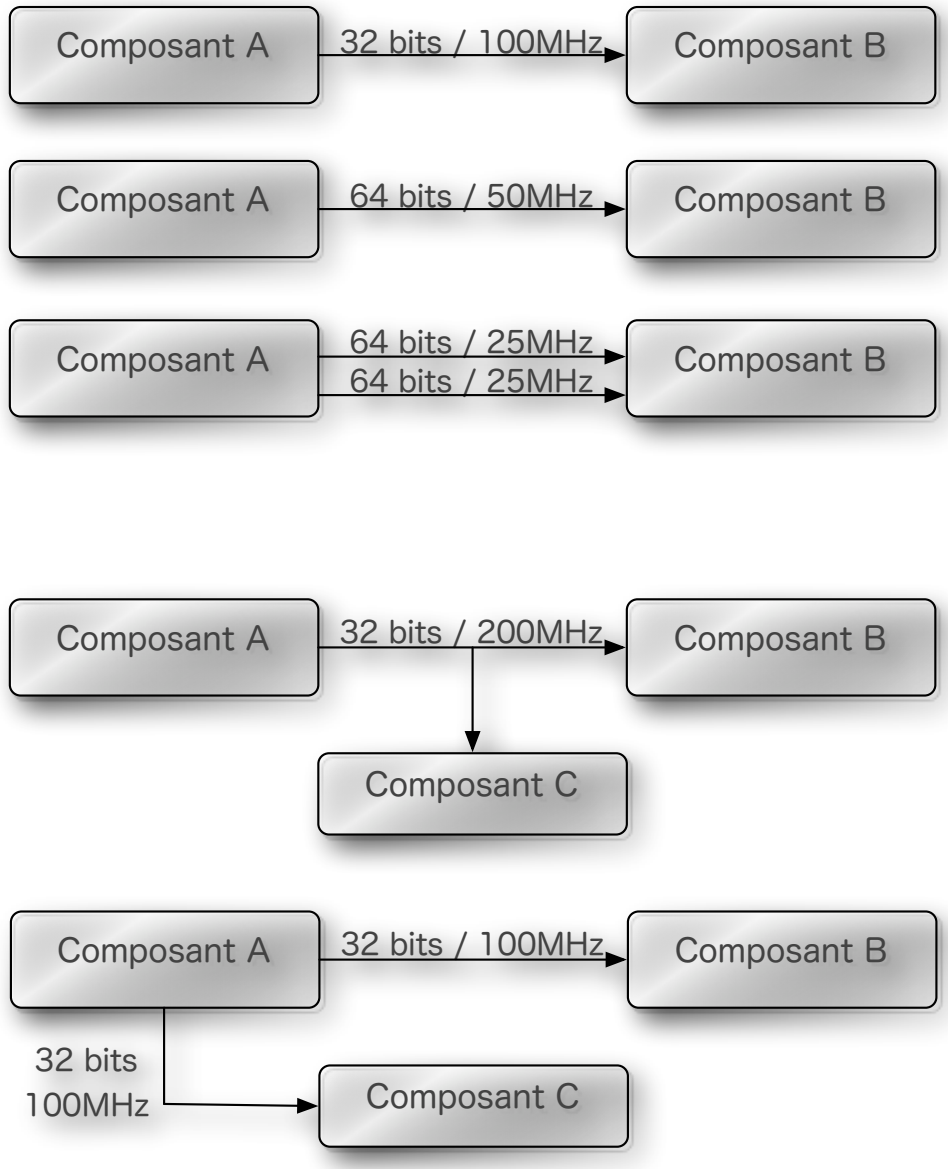


Le composant A produit les données (X1, X2, X3) en émettant un signal (ready)

Le composant B consomme les données (X3, X1, X2) tout en recevant un signal (start)

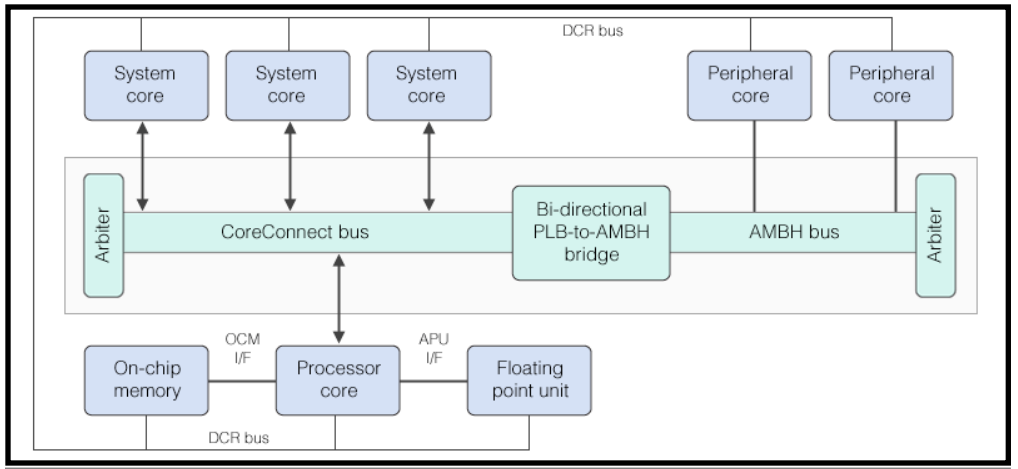
Les contraintes de débit dans les flux

- Dans les applications multimédia la contrainte forte des systèmes provient des débits à respecter :
  - Vidéo : 24 fps
  - Audio : 44kHz
- Il est donc important de considérer ces points critiques,
  - Interconnexions points à points
  - Interconnexions partagées,
  - Format des données (largeur des bus)
  - Vitesse des bus de données et d'adresse,

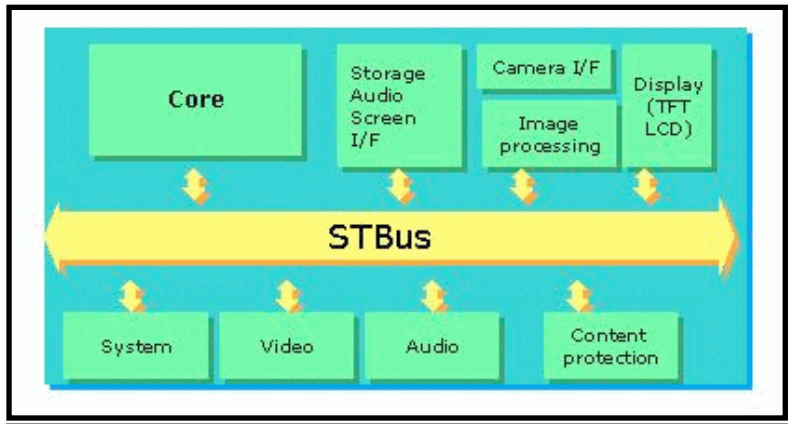


Interconnexion des différents composants

- L'interconnexion point à point a rapidement un coût prohibitif !
- Problèmes lors de phases de routage,
- Utilisation de bus partagés :
  - Partage des ressources grâce à un contrôleur d'accès au bus,
  - Généricité des interfaces (wrapper) => plus grande réutilisation des IPs,
  - Augmentation de la latence des transferts de données,
  - Tous les bus ne sont pas temps réel,



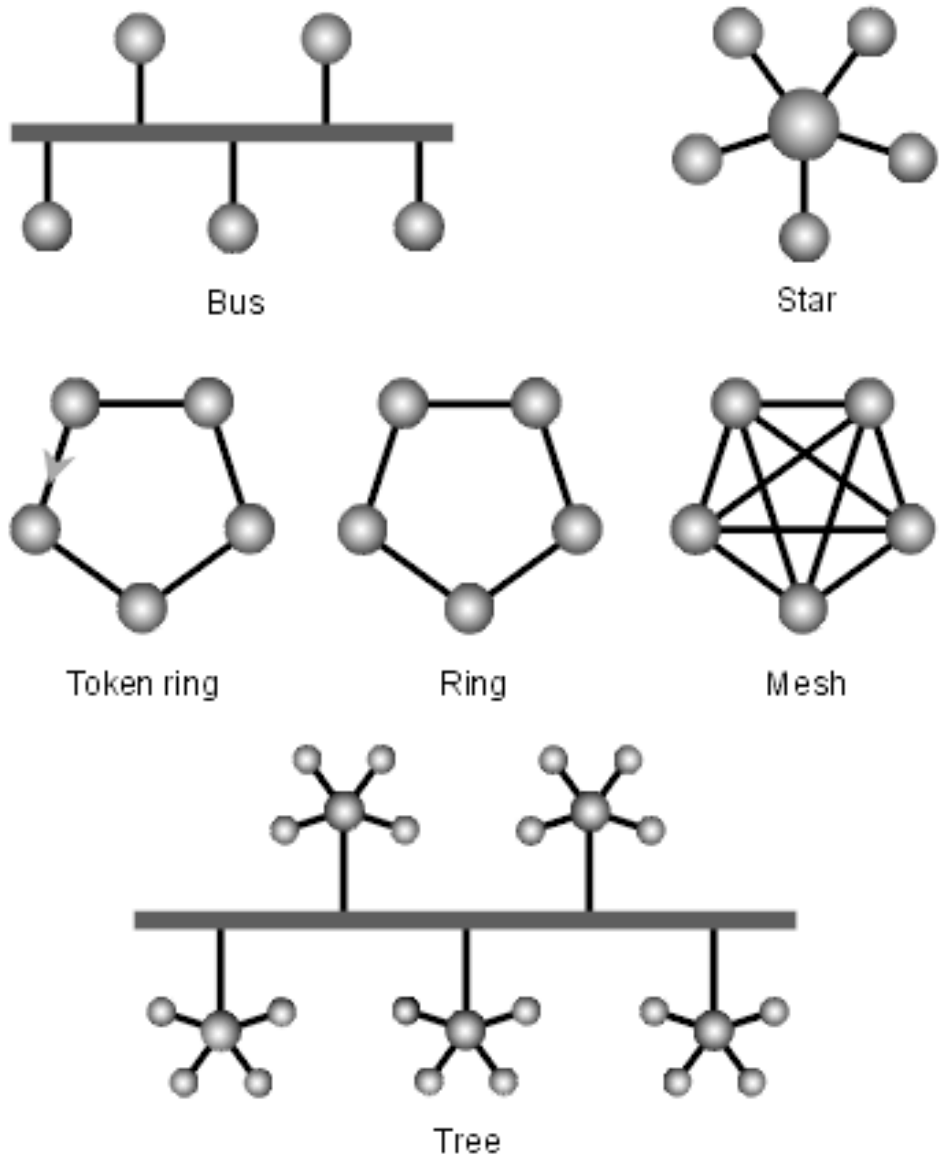
IBM



ST Micro

L'avènement des réseaux reconfigurables sur puce

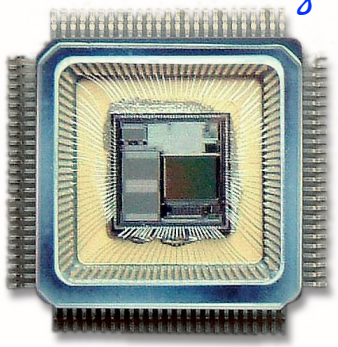
- Les bus sont plus insuffisants !
- Des réseaux sont nommés NoC (Network On Chip),
- Leur coût de mise en oeuvre sont prohibitif,
  - ➡ Ils permettent une meilleure décorrélation des blocs matériels,
  - ➡ Problèmes liés au déterminisme des communications (temps réel durement assuré !),
  - ➡ Intéressant pour la reconfiguration dynamique partielle des circuits,



Flot de développement de niveau système

Spécifications

Circuit numérique



Cahier des charges de l'application

Modélisation du système

Partitionnement logiciel / matériel

Modélisation du matériel

Modélisation des interfaces

Modélisation du logiciel + OS

Raffinement du matériel

Raffinement des interfaces

Raffinement du logiciel + OS

Synthèse du matériel

Co-vérification

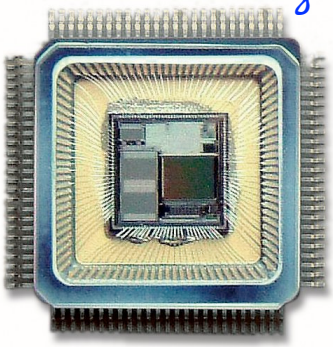
Compilation du logiciel

Intégration système

Flot de développement de niveau système

Spécifications

Circuit numérique



Cahier des charges de l'application

Modélisation du système

Partitionnement logiciel / matériel

Modélisation du matériel

Modélisation des interfaces

Modélisation du logiciel + OS

Raffinement du matériel

Raffinement des interfaces

Raffinement du logiciel + OS

Synthèse du matériel

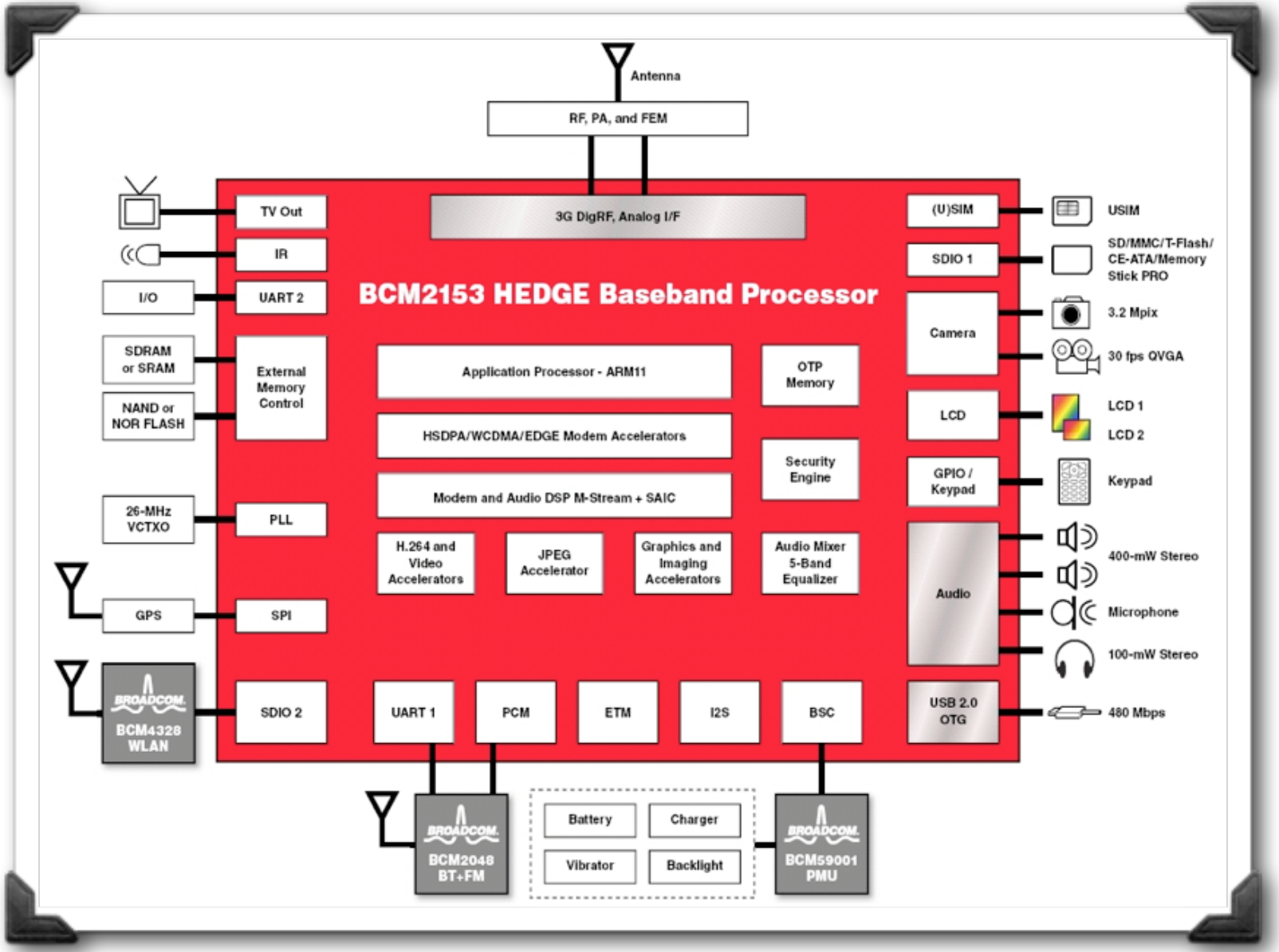
Co-vérification

Compilation du logiciel

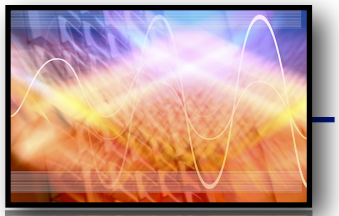
Intégration système

Les problématiques connexes

Exemple, ChipSet pour la téléphonie mobile

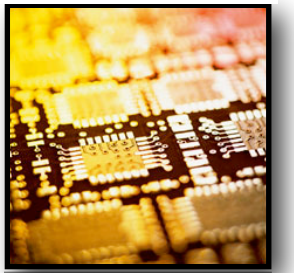


Limites engendrées par l'hétérogénéité



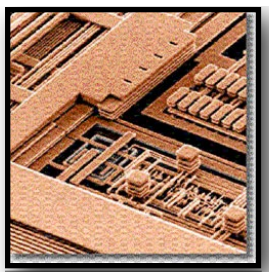
Connaissances théoriques, pratiques
et outils nécessaires dans l'équipe

ASIC

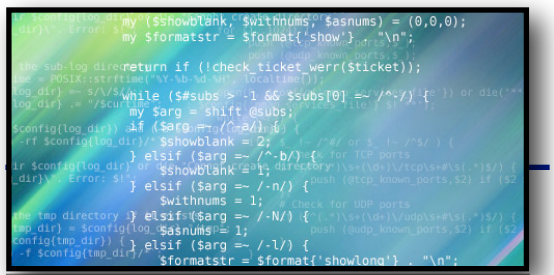


Interconnexion

Analogique



Développement logiciel

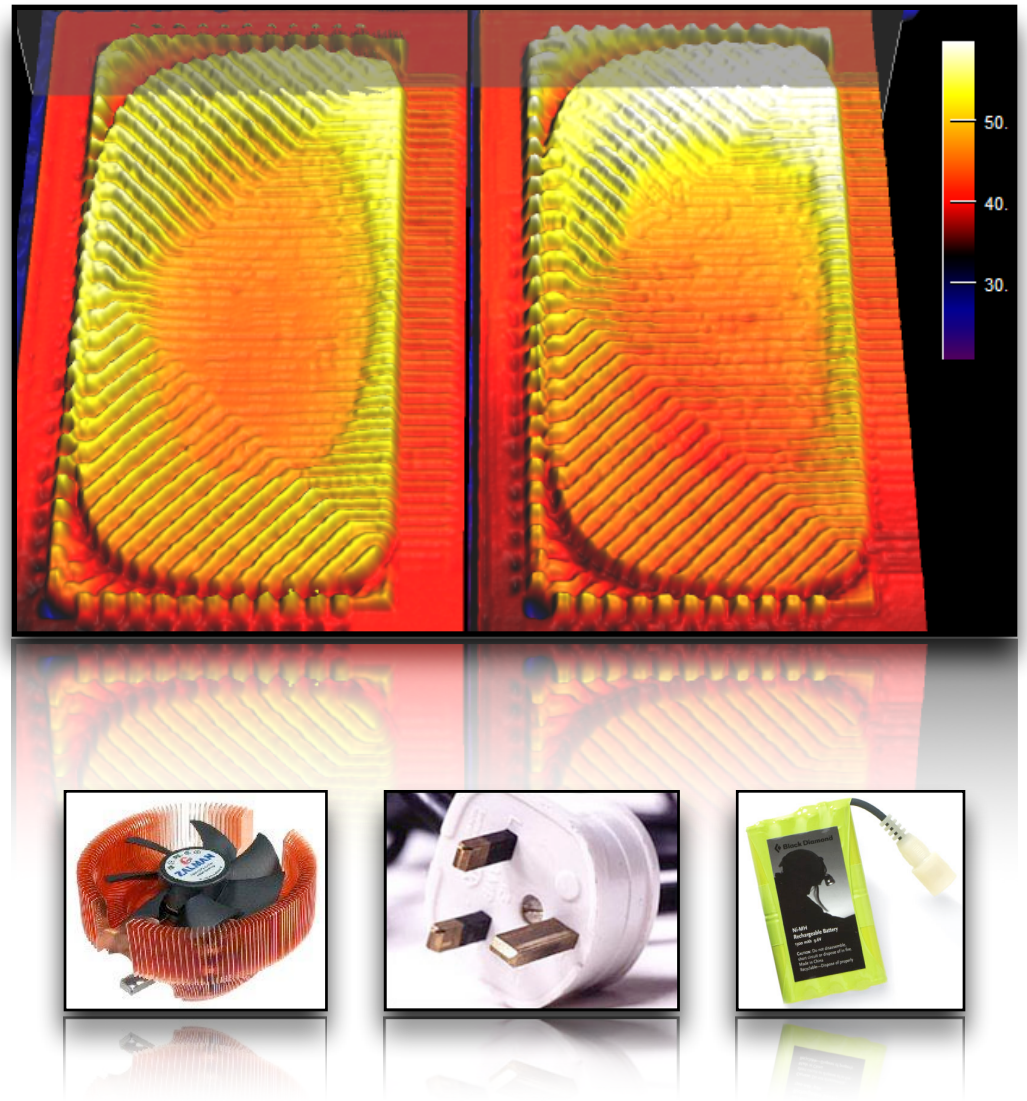


Processeurs / DSP

La consommation d'énergie

La consommation d'énergie

- La consommation d'énergie dans les systèmes embarqués devient une contrainte majeure,
- Plusieurs raisons
  - ➡ Augmentation de la consommation d'énergie (versus la capacité des batteries qui évolue moins vite),
  - ➡ Dissipation thermiques importantes (moyens de refroidissement coûteux, destruction des matériaux),
- Lla consommation provient :
  - ➡ Unités de calculs et registres associés,
  - ➡ Des arbres d'horloges,
  - ➡ Des mémoires de données,



Les cause de la consommation de puissance

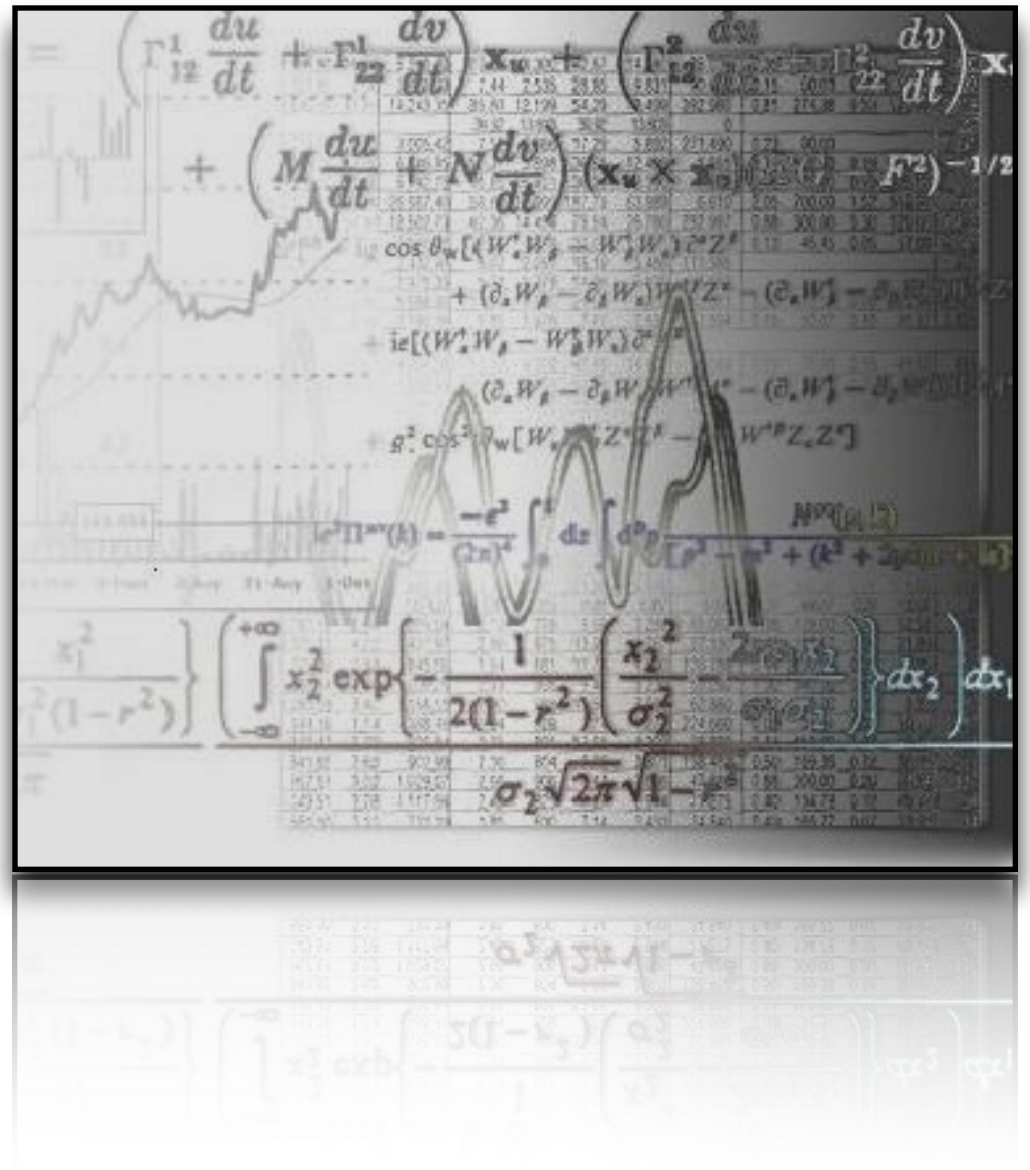
- Static power consumption
  - ➡ Due to leakage currents circulating when circuit is in idle state or has a little activity.
  - ➡ Depends on the design area

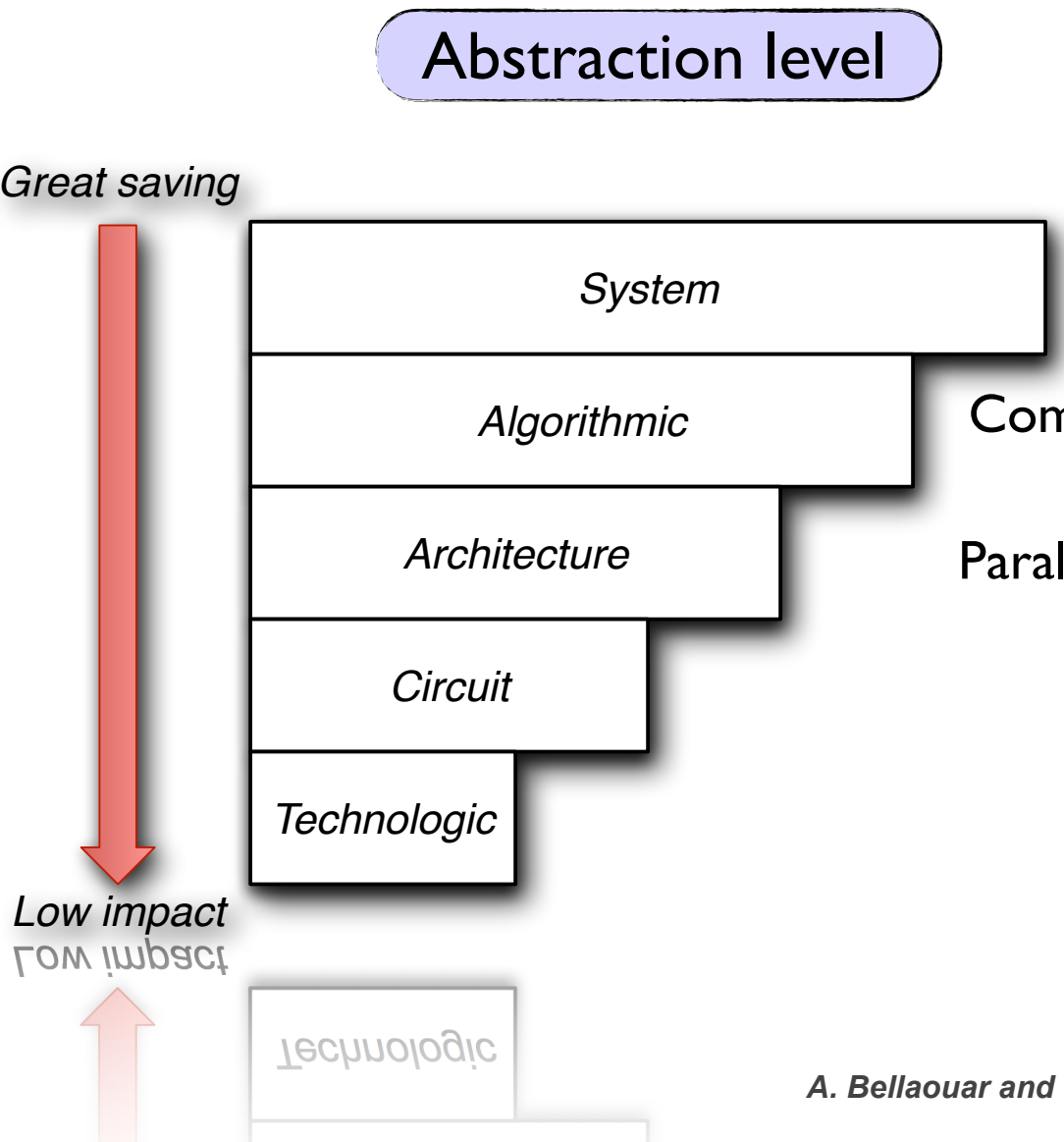
- Dynamic power consumption
  - ➡ Dissipated when charging and discharging the gate capacitances of the circuit,
  - ➡ Issue from the circuit activity (no usage ⇒ no power consumption)

A. Chandrakasan, S. Sheng, R. Brodersen, "Low-power CMOS Digital Design" IEEE Journal of Solid-state circuit, pp. 473-484, April 1992.

Pistes de progrès pour réduire ce phénomène

- Quelques unes des pistes de progrès employées par les concepteurs
  - Changement des algorithmes de calcul à intégrer (modification des caractéristiques du système)
  - La réduction des fréquences de fonctionnement (modifications l'architecturale => Intel P-m)
  - Réduction des taux de commutation des ressources,
  - L'inhibition de l'horloge lorsque des modules sont inutilisés,
  - Changements technologiques afin de réduire la tension d'alimentation,





A. Bellaouar and M. I. Elmasry. "Low-power digital VLSI design circuits and systems". Kluwer Academic Publishers, Norwel, MA, USA, 1995.

«size sometimes does matter...»

- Why worry about power ?
  - Batterie lifetime represents a quality metric for embedded devices
    - Adding more and more functionalities increase the global power consumption,
    - Increasing the battery size (⇒ weight) is not possible with embedded devices,
    - Packaging cost increases with power dissipation requirements,
- Our objective
  - Adapting the low area multimode design to minimize its power consumption,
  - Find an efficient tradeoff (area/power),

- Multiplier (*n* bits) is composed of *n* adders
  - Sum of partial products (commutations, area, latency)
  - Can be optimized while manipulating constant numbers,
- Canonical Signed-Digit number format:
  - Reducing the number of partial products,
  - Multiplication = additions and substractions of partial products,

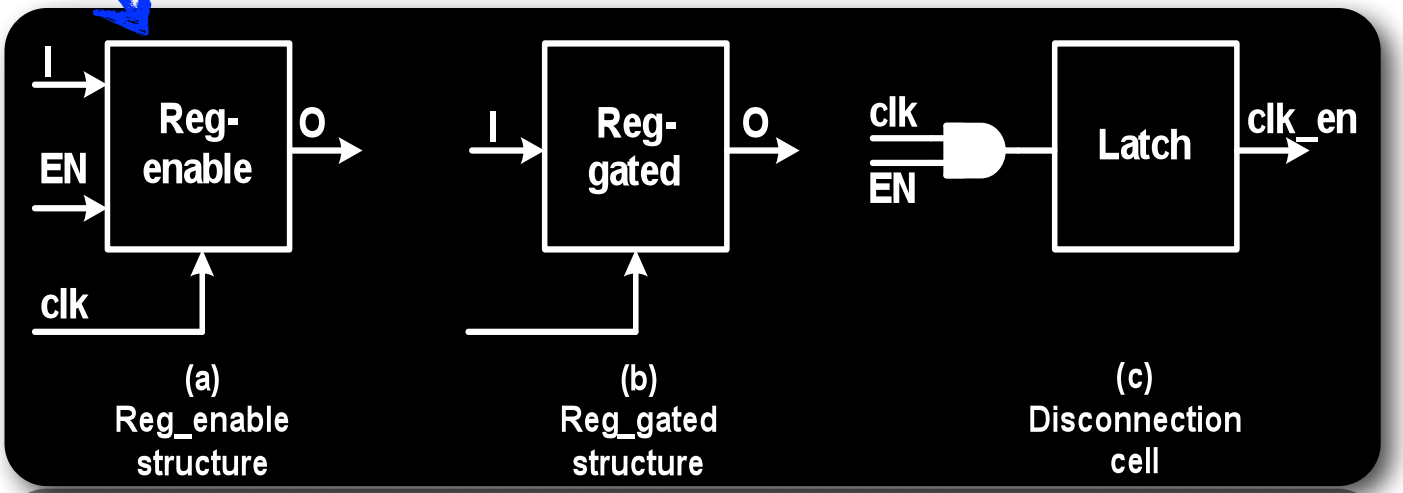
Decimal value over 13 bits	Two's complement representation	CSD coded representation
33	0000000100001	0000000100001
-78	1111110110010	000000-10-10010
172	0000010101100	000010-10-10-100
-376	1111010001000	000-1010001000
1283	0010100000011	001010000010-1
2048	0100000000000	0100000000000

For (-78) coefficient :

- Two's complement : 9 additions
- CSD coded : 1 addition, 2 subtractors

N.Khouja, K.Grati, and A.Ghazel "Low Power Implementation of Decimation Filters in Multistandard Radio Receiver Using Optimized Multiplication-Accumulation Unit" IEEE International Conference On Electronics, Circuits and Systems, pp 1183-1186, December 2007.

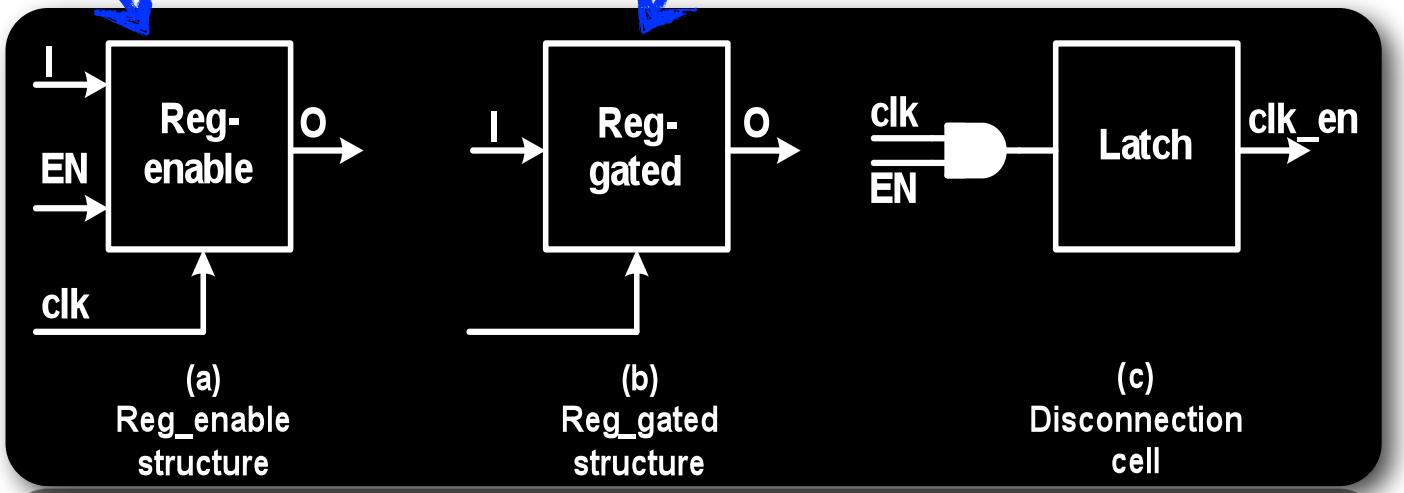
With or without the load signal, the register activity is linked to the clock frequency



Hai Li, Bhunia, S., Yiran Chen, Roy, K., Vijaykumar, T.N. DCG: deterministic clock-gating for low-power microprocessor design. IEEE Transactions on Very Large Scale Integration (VLSI) Systems Volume 12, Issue 3, pp(s): 245 – 254, March (2004).

With or without the load signal, the register activity is linked to the clock frequency

The controller send rising edge to the register only on load requirements

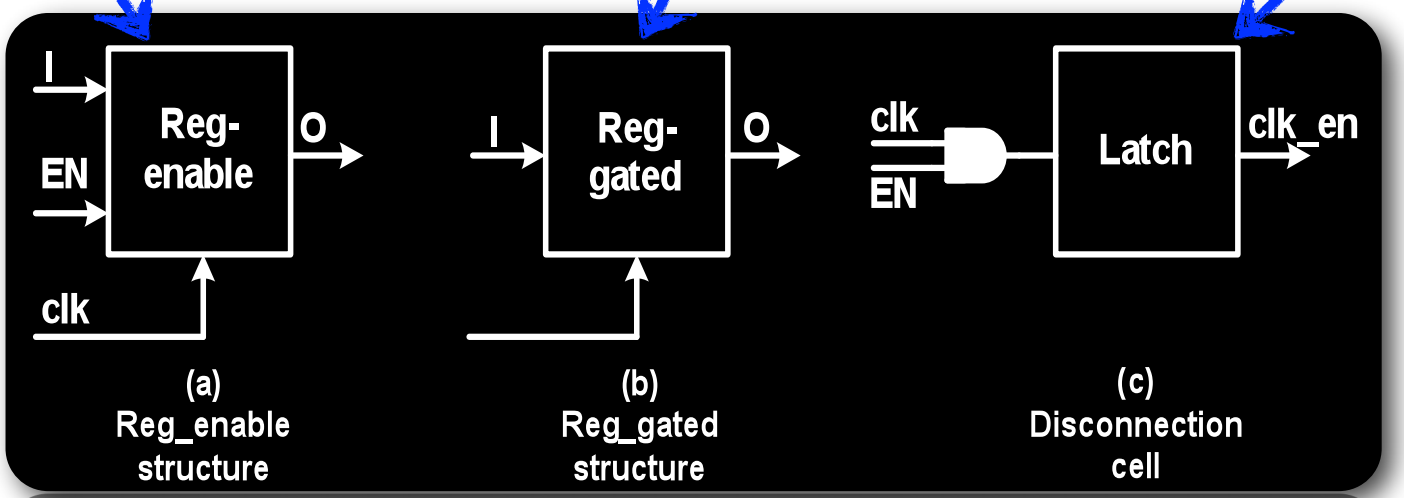


Hai Li, Bhunia, S., Yiran Chen, Roy, K., Vijaykumar, T.N. DCG: deterministic clock-gating for low-power microprocessor design. IEEE Transactions on Very Large Scale Integration (VLSI) Systems Volume 12, Issue 3, pp(s): 245 – 254, March (2004).

With or without the load signal, the register activity is linked to the clock frequency

The controller send rising edge to the register only on load requirements

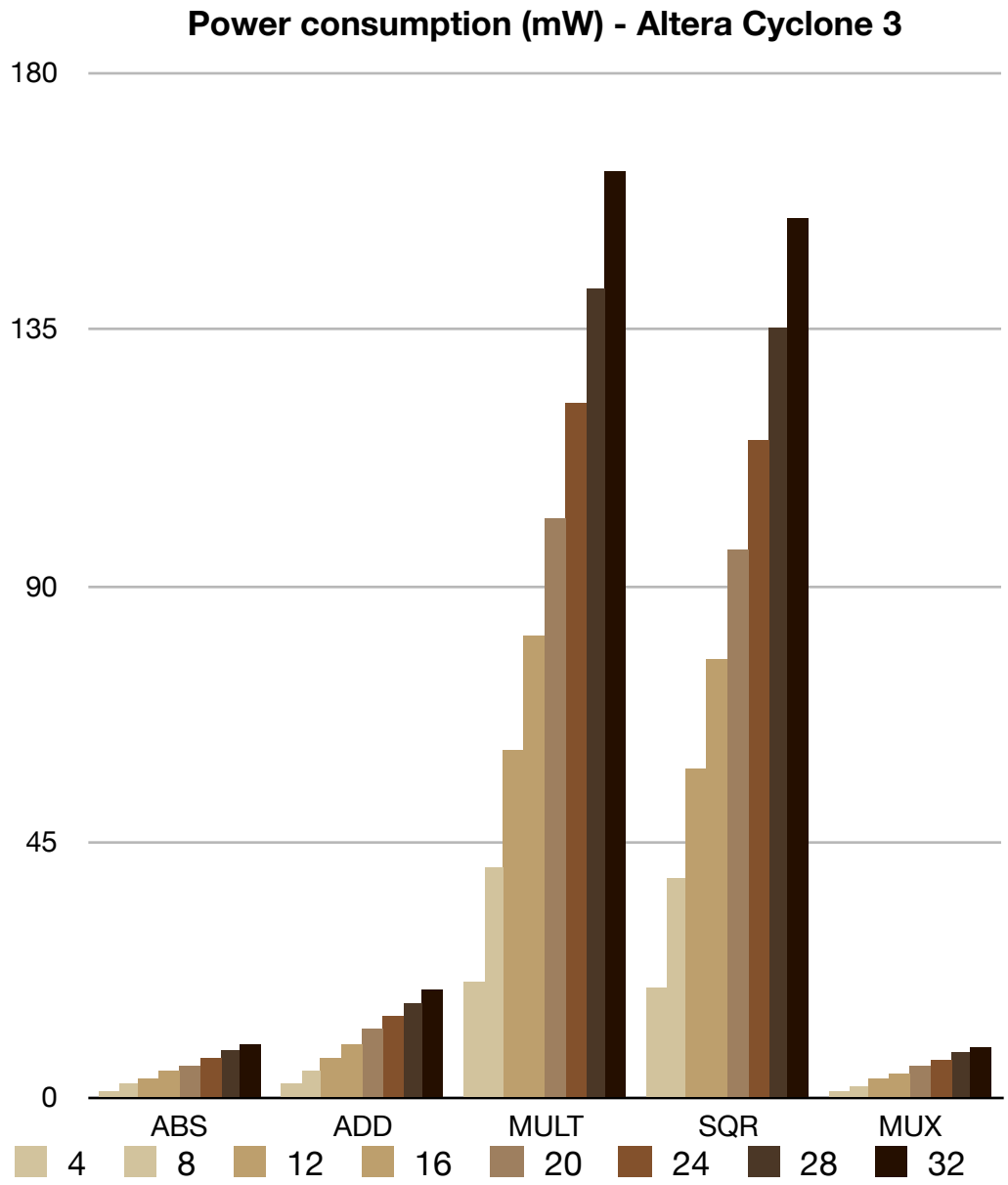
Similar but simpler for the controller design (frequency)



Hai Li, Bhunia, S., Yiran Chen, Roy, K., Vijaykumar, T.N. DCG: deterministic clock-gating for low-power microprocessor design. IEEE Transactions on Very Large Scale Integration (VLSI) Systems Volume 12, Issue 3, pp(s): 245 – 254, March (2004).

Evolution de la consommation dynamique

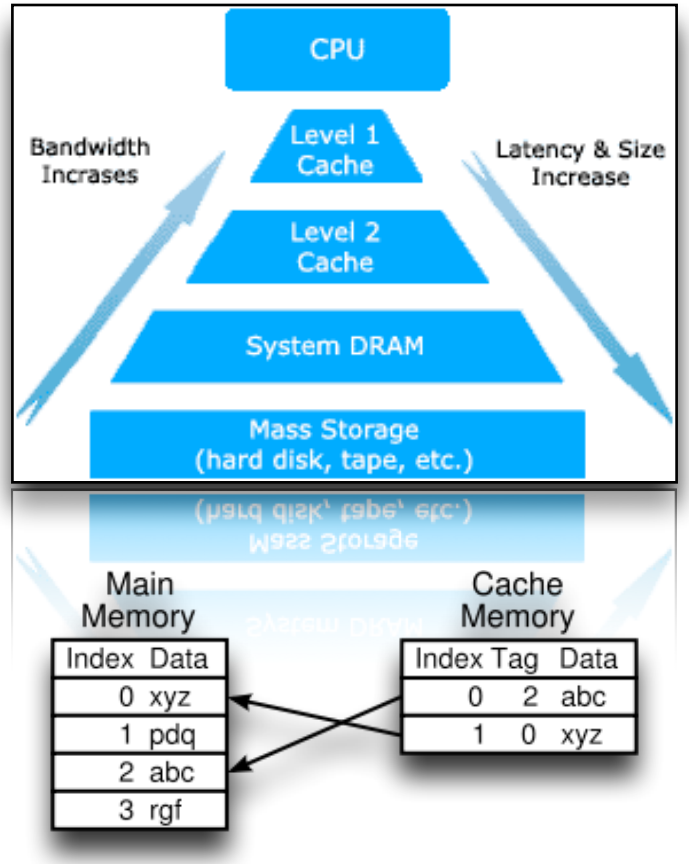
- La consommation d'energie est fortement liées aux calculs réaslisés,
- Importance de dimensionner rigoureusement,
  - ➡ Le nombre de calculs à réaliser (complexité calculatoire),
  - ➡ Le type des données et leur dynamique,
  - ➡ La fréquence de fonctionnement du système,
- La taille du circuit impact sur la consommation statique...



Les structures de mémorisation

Les structures de mémorisation

- L'accès aux données est généralement le goulot d'étranglement des applications,
- Augmentation de la mémoire,
  - ➡ Disques durs (2 TeraOctets),
  - ➡ Mémoires (2 GigaOctets),
  - ➡ Caches (8 MegaOctets),
- L'augmentation des fréquences implique une gestion efficaces de la mémoire,
  - ➡ Pénalité en cas de rupture du pipeline,
  - ➡ Temps d'accès aux données,



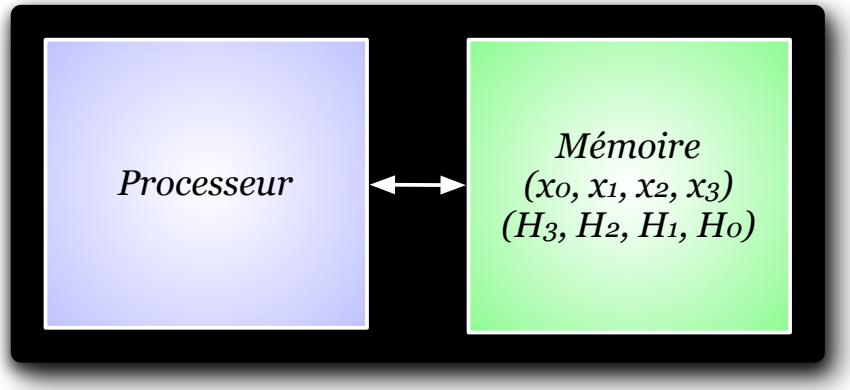
Timings pour un Athlon 64 3800+

- Cache de niveau 1 : 27,7 Mbits/s
- Cache de niveau 2 : 8,8 Mbits/s
- Cache de niveau 2 : : 3/13 cycles (latence)
- Système RAM : 4/103 cycles (latence)
- Disque dur : 4,16 ms (latence)

Impact du nombre de bancs mémoire

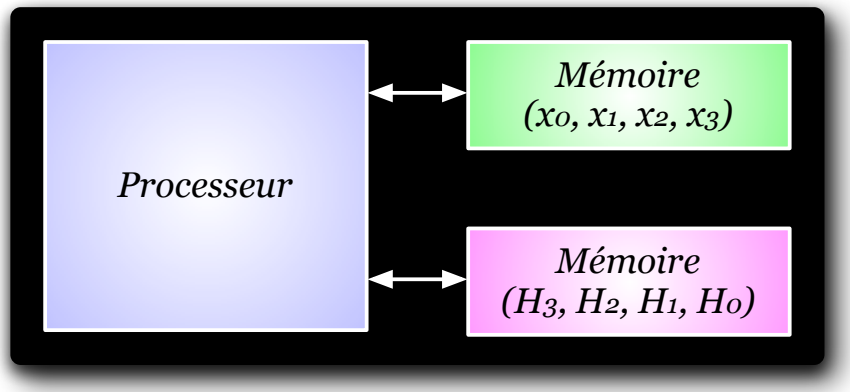
Considérons une application classique en traitement du signal.

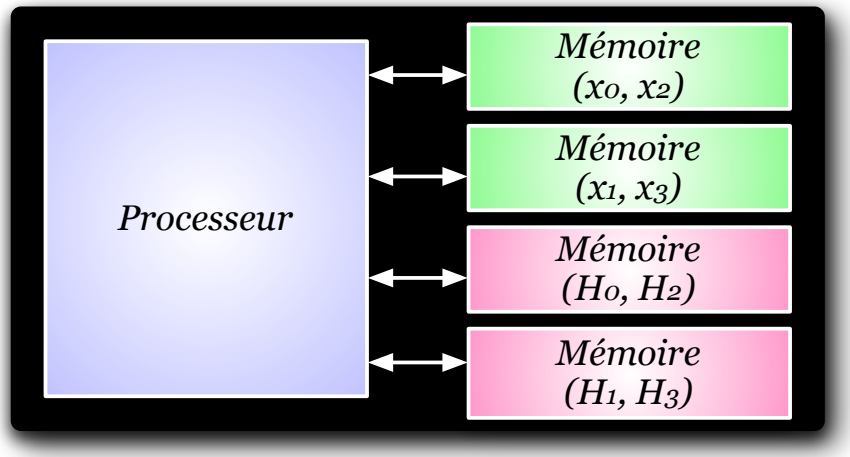
$$y = \sum_{i=0}^{3} x(i) \times H(i)$$



Combien de cycles sont nécessaire pour réaliser le calcul en fonction des architectures mémoires ?

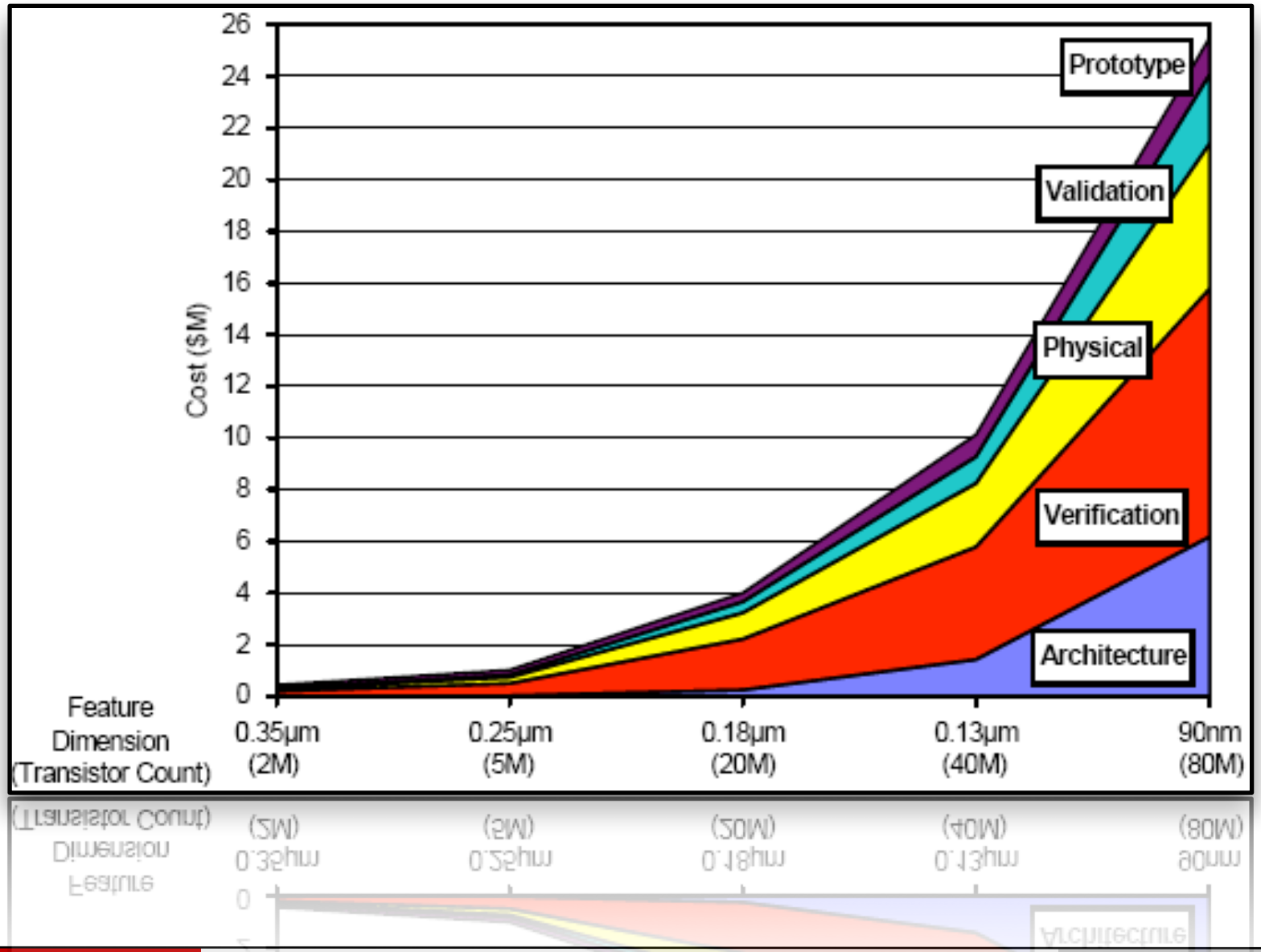
On considérera qu'un accès mémoire prend un cycle tout comme les opérations + et *



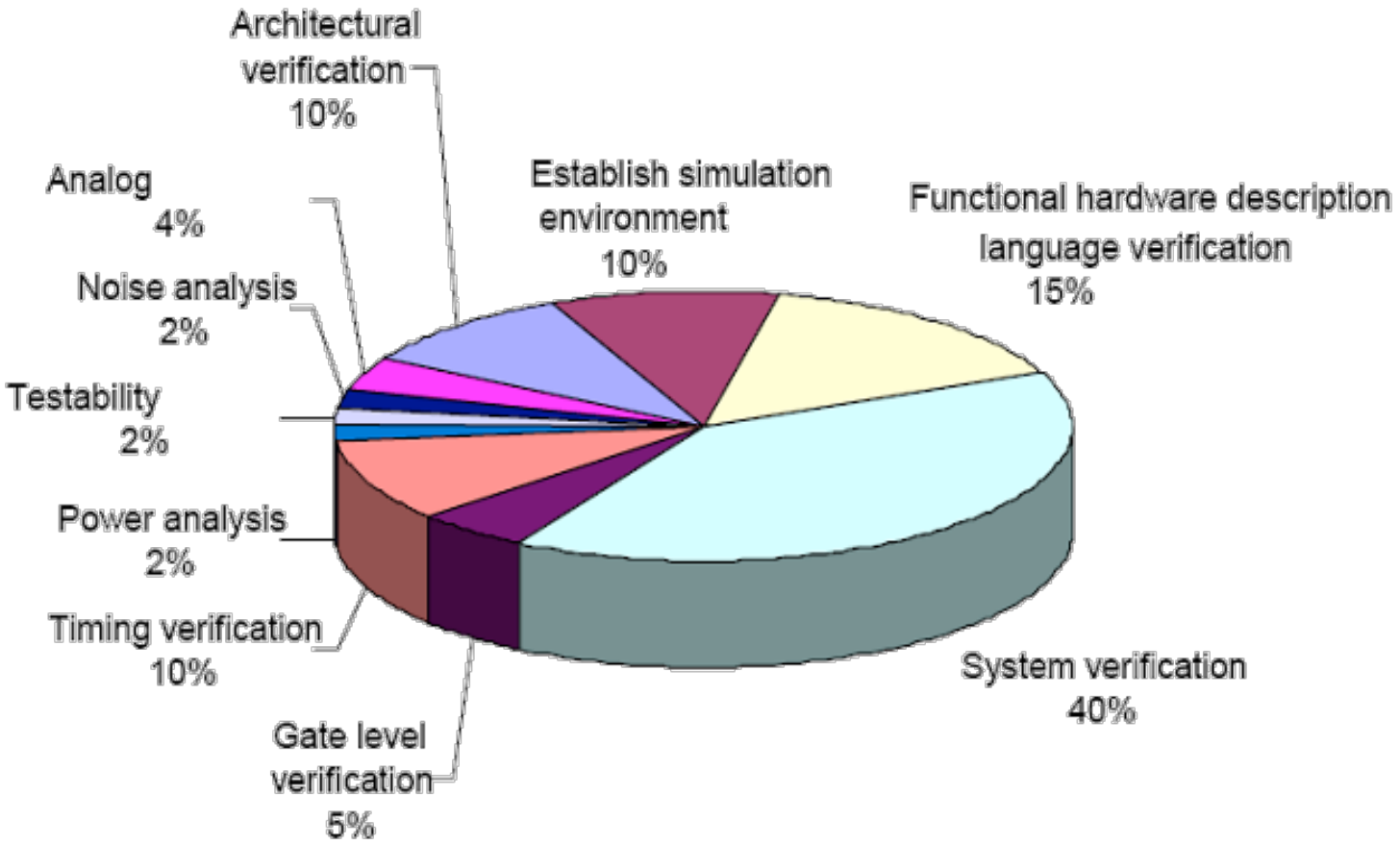


Le test et la vérification des systèmes

Répartition des coûts de développement



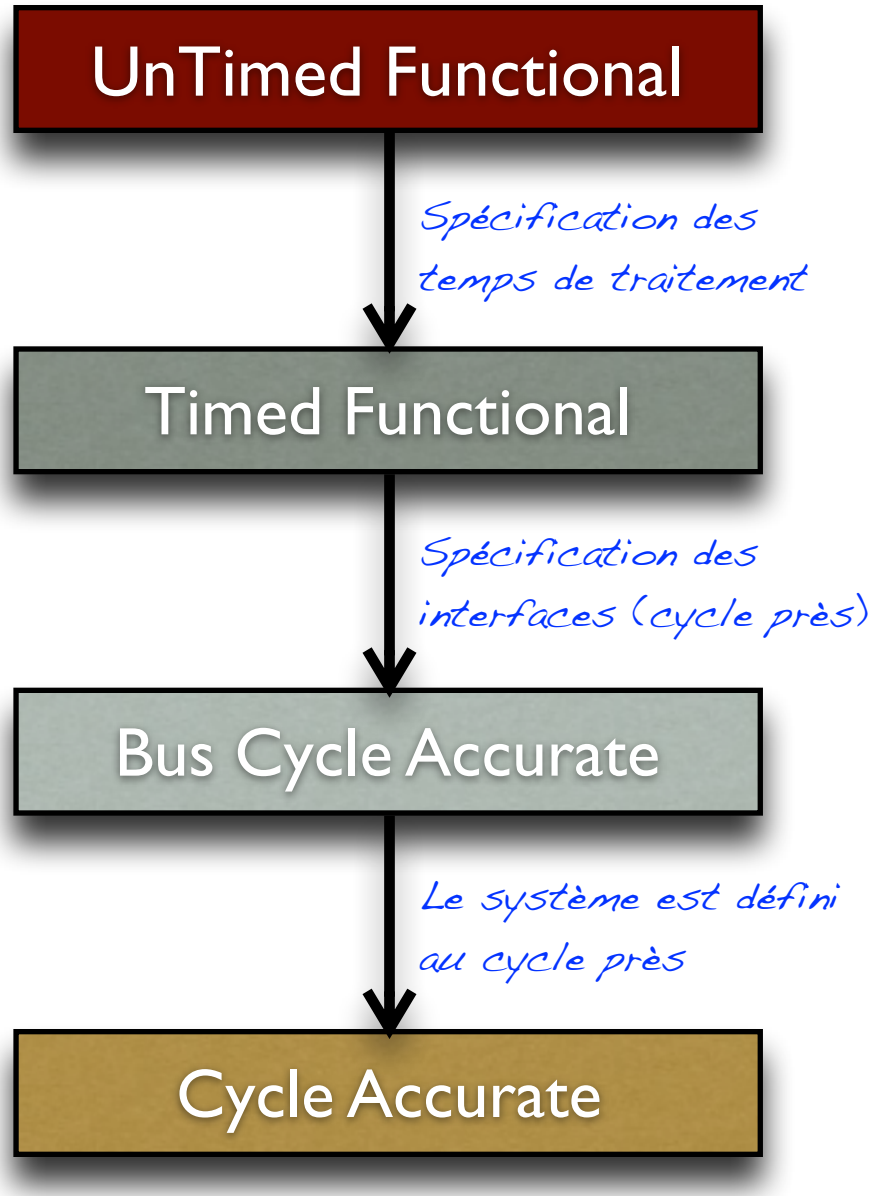
Test et vérification des systèmes numériques



[Source : ITRS 2003]

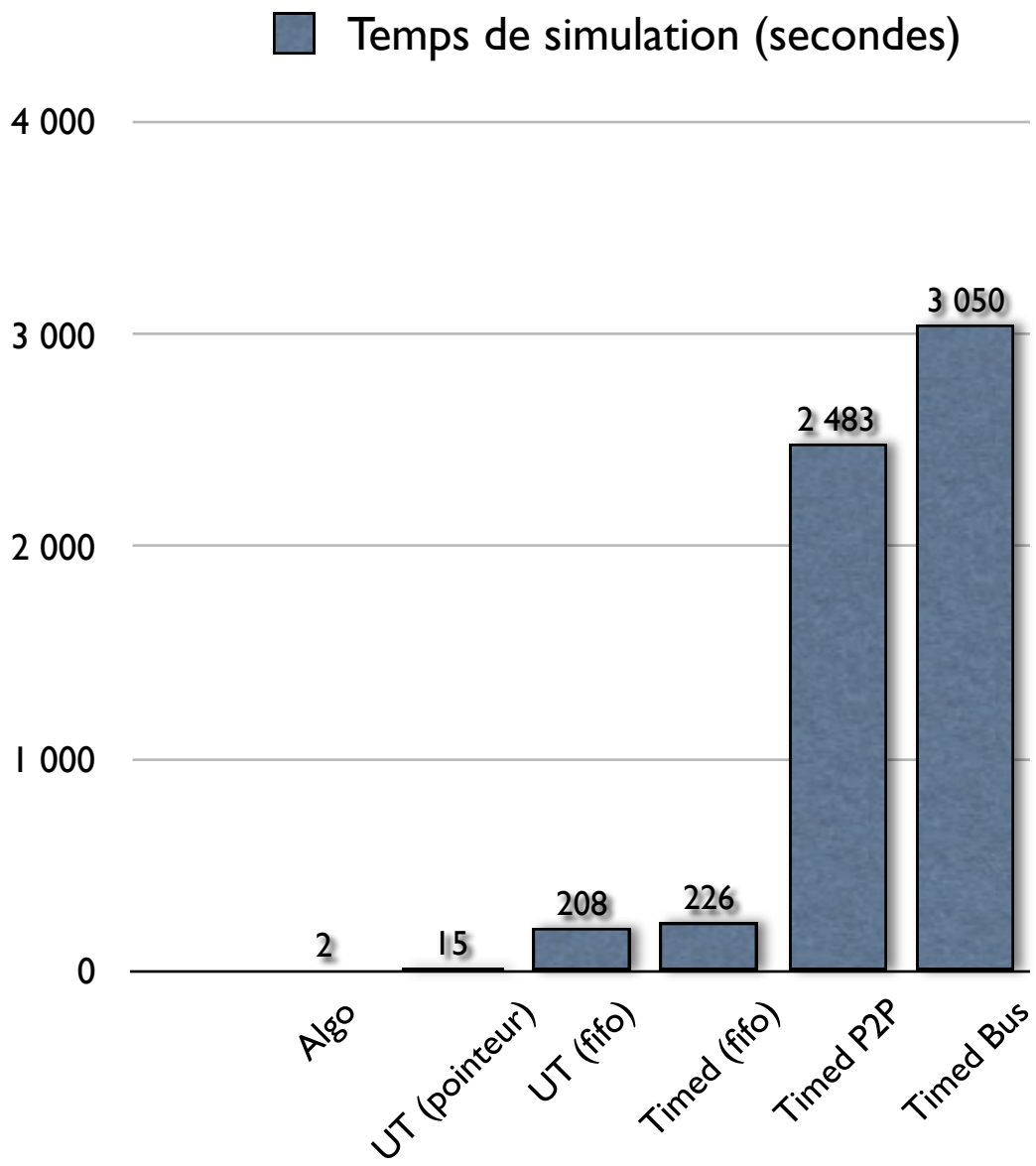
Les niveaux d'abstraction usuels

- Lors des phases de conception, il faut s'assurer de la validité des solutions développées,
- Cette tache est réalisée durant l'ensemble du processus de conception,
- A chaque niveau de raffinement, on s'attache à des points différents,
- Vérification fonctionnelle à bas niveau d'un décodeur MPEG-2,
  - ➡ 2 images => 30 jours de simulation



Niveaux d'abstraction et temps de simulation

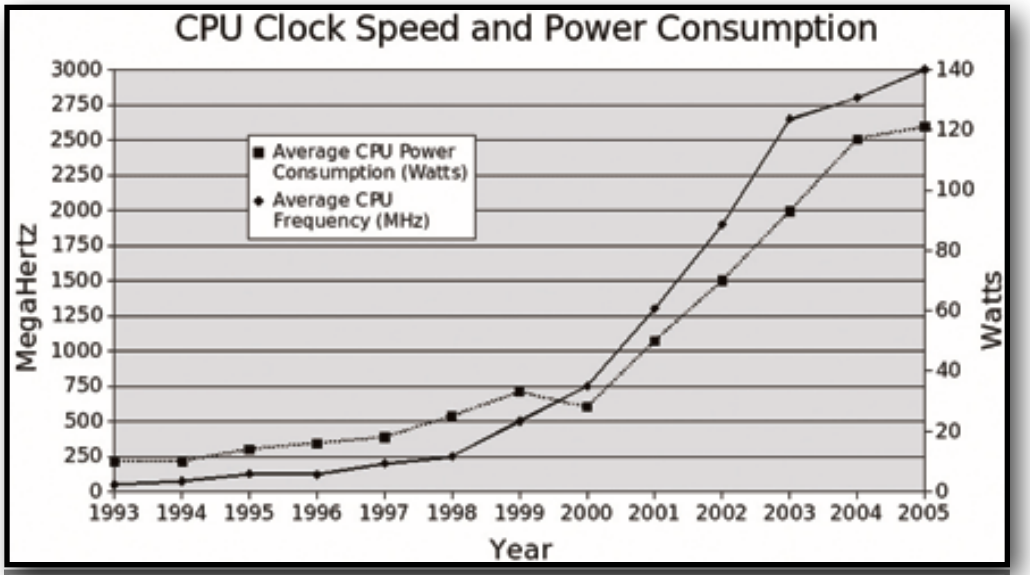
- Le sujet de l'expérimentation était un encodeur vidéo de la norme h263+,
- Modélisation d'un accélérateur matériel couplé à un uP pour réaliser les calculs,
  - ➡ Calcul de la distorsion entre 2 imagettes (identification des mvts),
- Le langage de modélisation utilisé était SystemC 2.0
- Le test portait sur la compression d'une série de 3 images uniquement !



Les architectures de demain ?

Evolution de la ratio (Fréquence / Puissance)

- L'augmentation de la fréquence des circuits s'est interrompue,
- La fréquence d'horloge n'est pas représentative des performances,
  - ➡ consommation et de la dissipation,
- Il est préférable de travailler sur les architectures internes que sur l'horloge,
  - ➡ Prédiction de branchement,
  - ➡ Gestion et taille des caches,
  - ➡ SIMD, MIMD, EPIC, etc.



Pentium 4 EE 3.73GHz
- Conso au repos 161W
- Conso en charge 238W

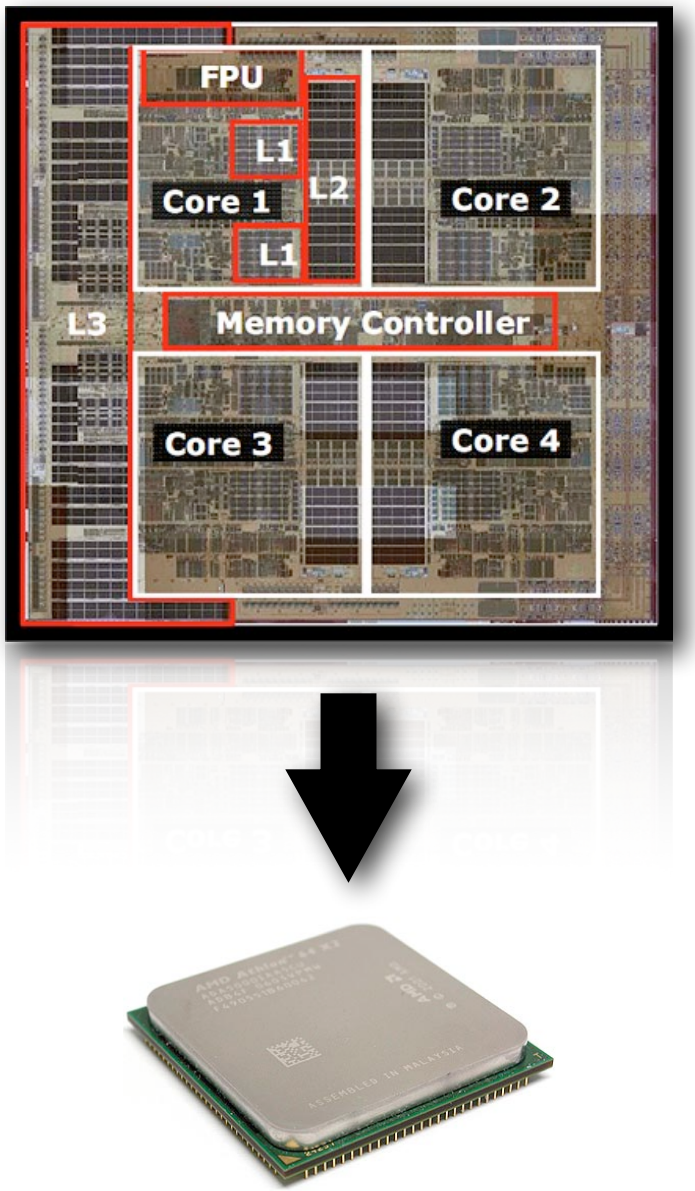
Pentium M 755
- Conso au repos 92W
- Conso en charge 101W

Ecart de performances : environ 6% sur du traitement vidéo...

http://techreport.com/reviews/2005q1/pentium4-600/index.x?pg=1

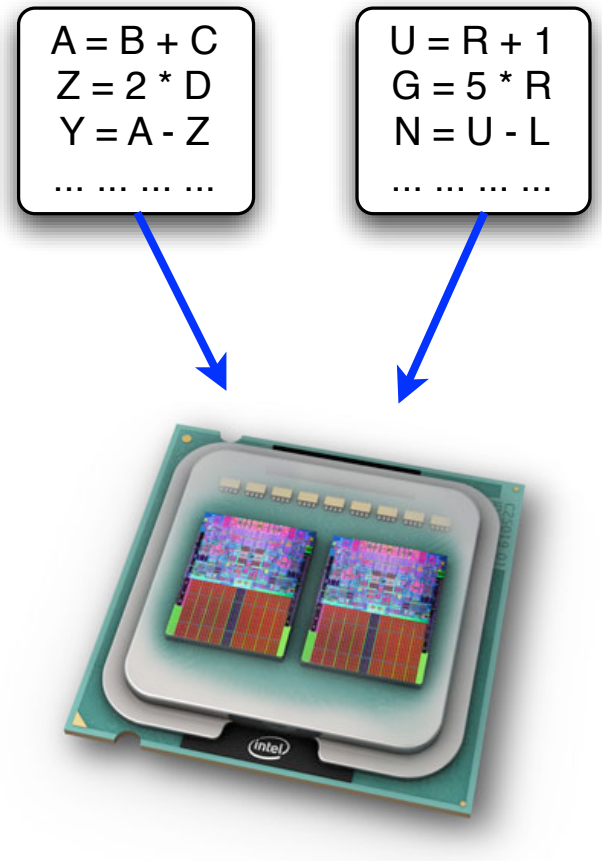
L'approche retenue : Le calcul parallèle

- Les nouvelles tendances consistent à intégrer des coeurs de processeurs plus "lents" mais en nombre,
- Cette technique vise à permettre l'exécution réellement parallèle des applications,
- Attention, n processeurs ne signifie pas que tout va aller n fois plus vite,
  - ➡ L'approche est analogue à la multiplication de la fréquence d'horloge...



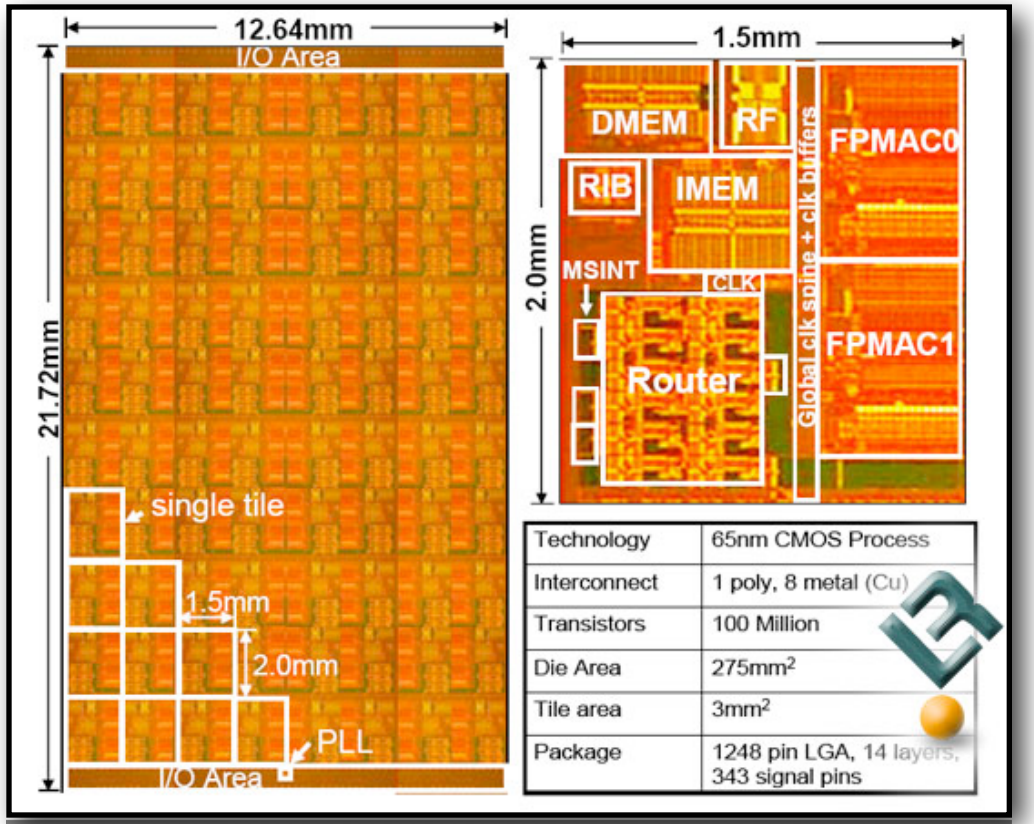
Exemple de mise en oeuvre

- Lorsque l'on exécute 2 programmes distinct, l'ordonnanceur du système peut les répartir efficacement sur les ressources disponibles,
- Il faut que les ressources utilisées soient différentes (ES, zones mémoires, accès disque),
- Un programme contenant plusieurs thread, peut aussi bénéficier de l'accélération,
  - ➡ Traitement d'image sur des zone distincts par exemple (fragmentation du travail),



Les perspectives d'avenir (Intel)

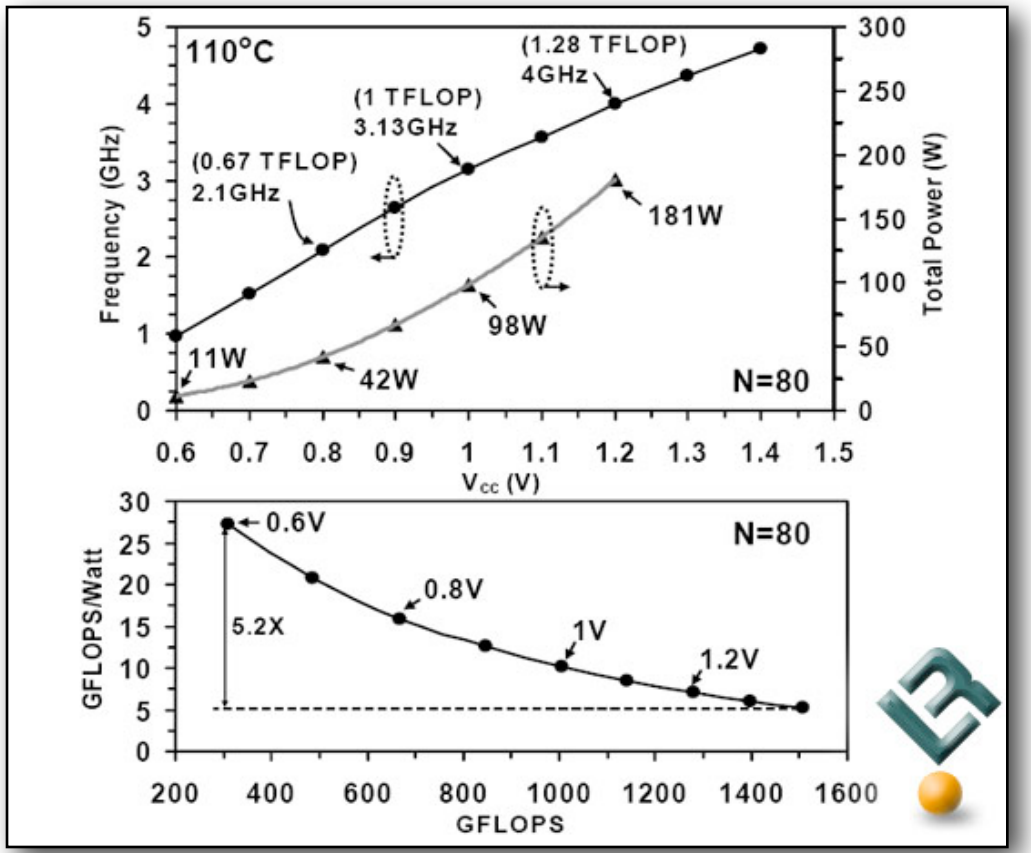
- Intel vient de concevoir une structure systolique à base de 80 processeurs,
- Le circuit contient 80 FPU autonomes (2mm*1.5mm)
- L'objectif est une puissance de calcul de 1 TeraFlops,
- En théorie, jusqu'à 80 opérations flottantes par cycle d'horloge,
- Commercialisation aux alentours de 2010 / 2012...



Technology	65nm CMOS Process
Interconnect	1 poly, 8 metal (Cu)
Transistors	100 Million
Die Area	275mm²
Tile area	3mm²
Package	1248 pin LGA, 14 layers, 343 signal pins

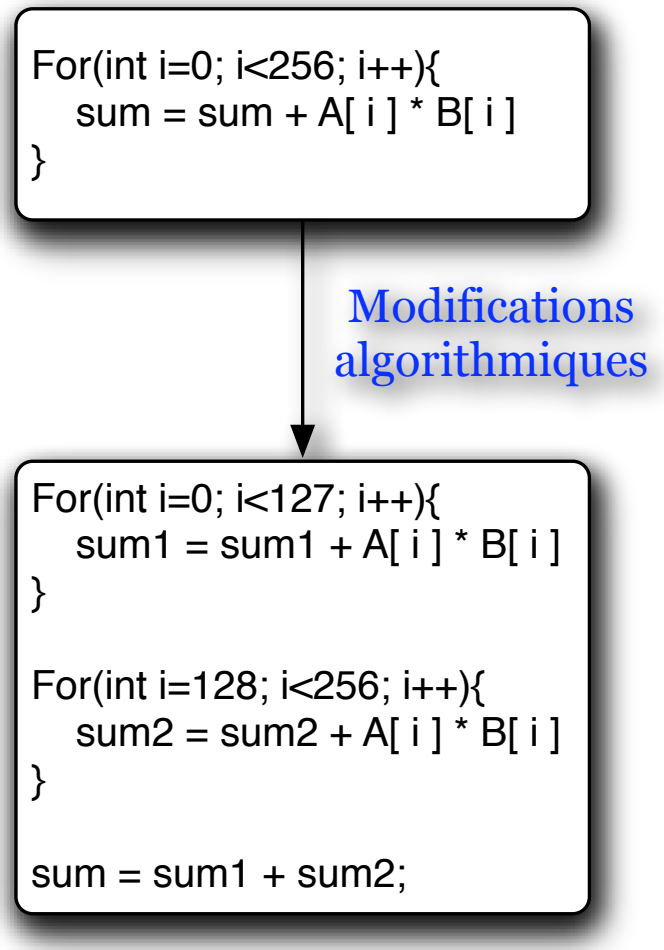
Les perspectives d'avenir (Intel)

- Cette approche permet de réduire la fréquence d'horloge et la consommation pour une puissance de calcul donnée,
- L'efficacité du circuit pour des applications parallèles est importante,
- Il est nécessaire de posséder :
  - Une bande passante gigantesque,
  - Des mémoire et des caches de tailles importantes,
  - Des applications (algorithmes) adaptées à la structure du circuit,



La problématique sous-jacente

- Besoin d'avoir des applications fractionnables en taches,
- L'insertion de points de synchronisation à un coût,
- Le partage des ressources est complexe à gérer (duplication des zones mémoire, gestion des conflits, etc.),
- Les compilateurs logiciels gérent pas cette complexité,
  - ➡ Besoin d'ingénieurs bien formés en électronique, informatique et traitement du signal !



Exemple : Modification de codes sources

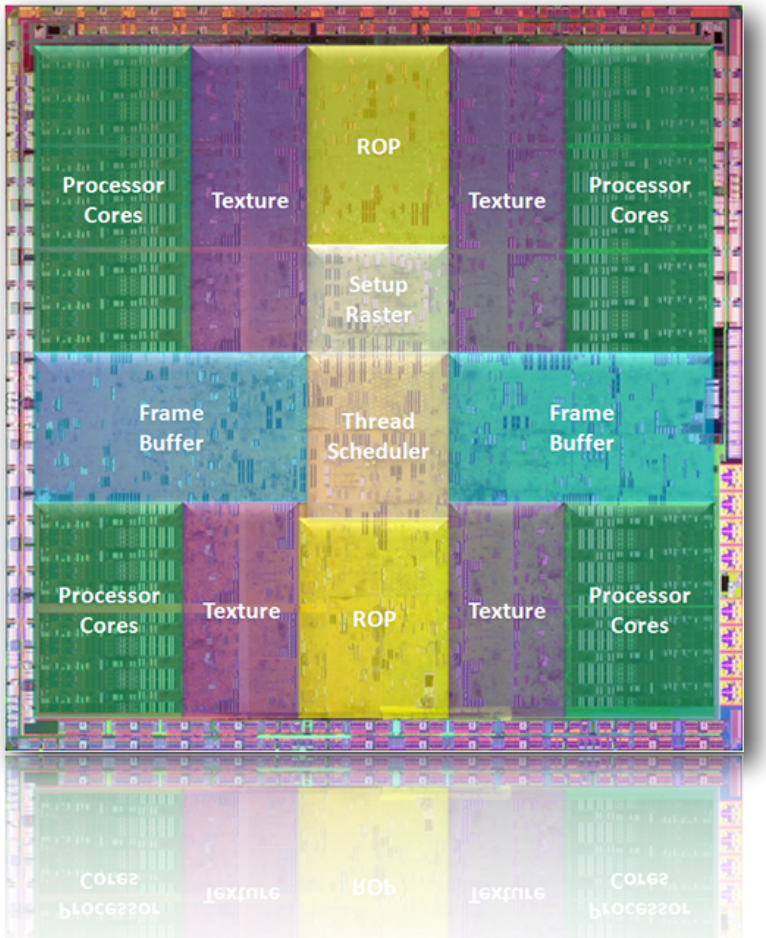
```
int SommeMatrice(int matrice[64,64]){
    int sum = 0;
    for(int y=0; y<64; y++)
        for(int x=0; x<64; x++)
            sum += matrice[y, x];
    return sum;
}
```

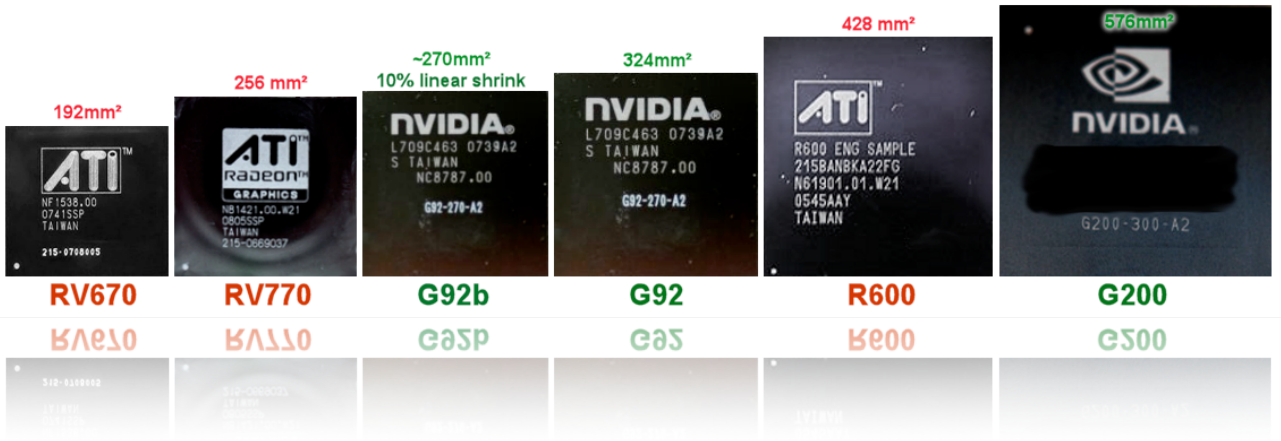
```
int PGDC(int a, int b){
    while( a != b ){
        if( a > b )
            a = a - b;
        else
            b = b - a;
    }
    return b;
}
```

Exemples : Codes sources modifiés

```
int SommeMatrice2(int matrice[64,64]){
    int sum = 0;
    int tmp[64];

    /* Calculs parallele sur les lignes */
    for(int y=0; y<64; y++)
        for(int x=0; x<64; x++)
            tmp[y] += matrice[y, x];

    /* Synchronisation la colonne temporaire */
    for(int y=0; y<64; y++)
        sum += tmp[y];

    return sum;
}
```

Pour les calculs sur GPU, le constat est similaire...





=> Architectures de calculs massivement parallèles (systoliques)

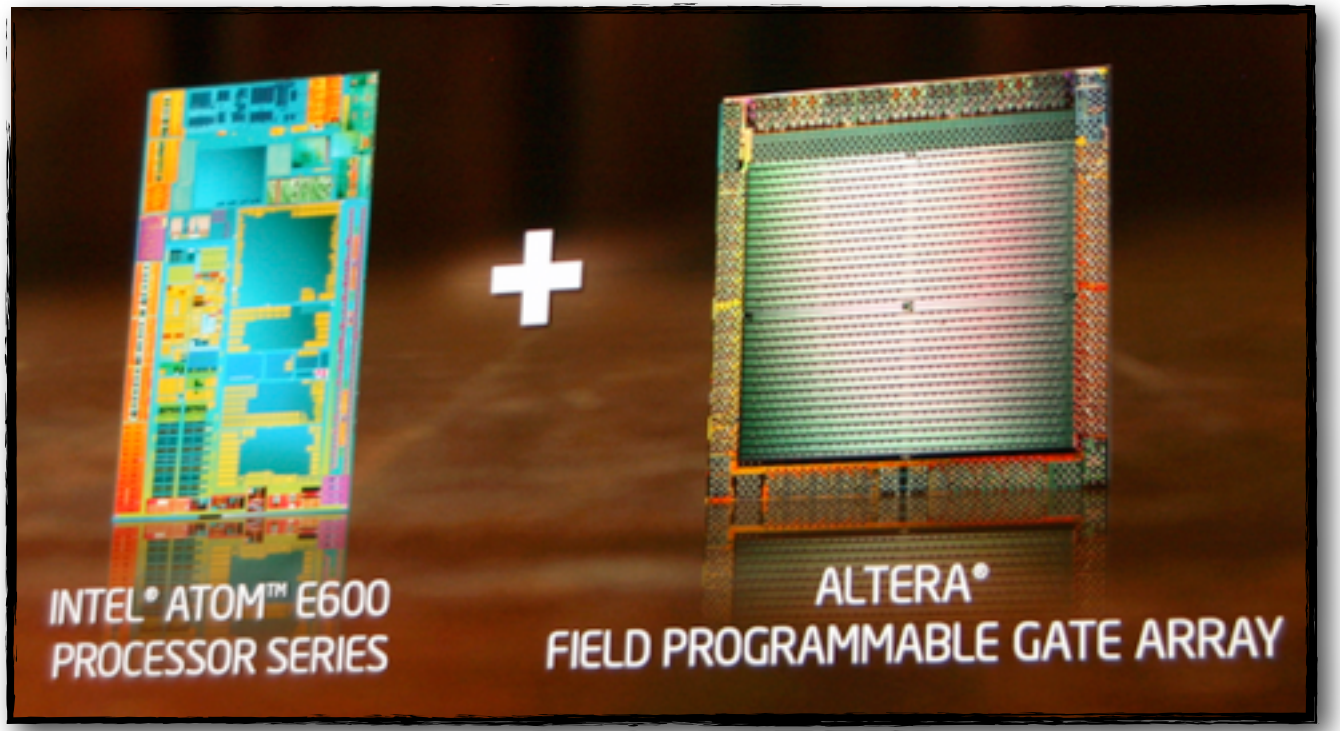
=> Les GPU transforment les données entières sous forme flottantes afin de réaliser les calculs.

=> Ces systèmes sont gourmands en énergie

=> Afin de profiter de la puissance de calcul, il faut posséder de larges bandes passantes,

=> Ces architectures sont utilisables pour les structures de calcul régulières (sans opérations conditionnelles).

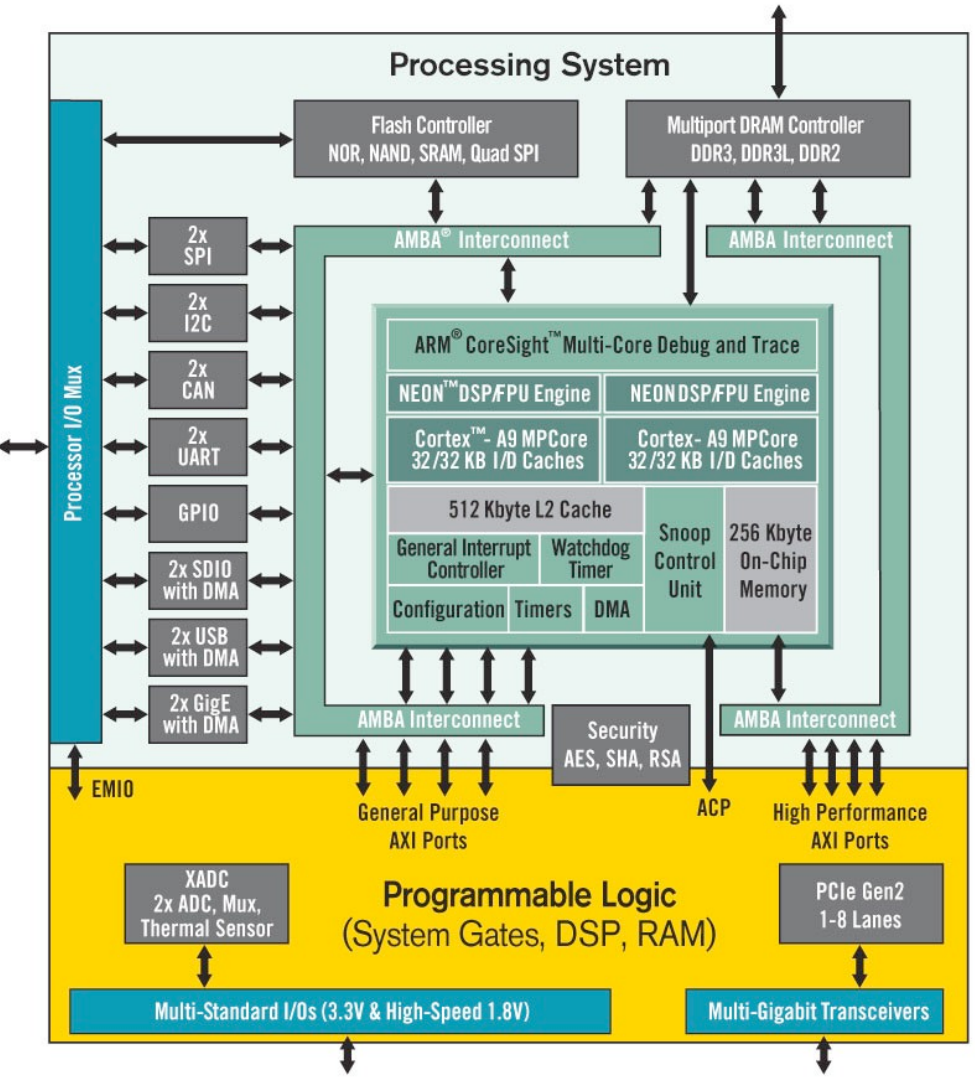
Architectures du futur pour les systèmes embarqués (1/2) ?

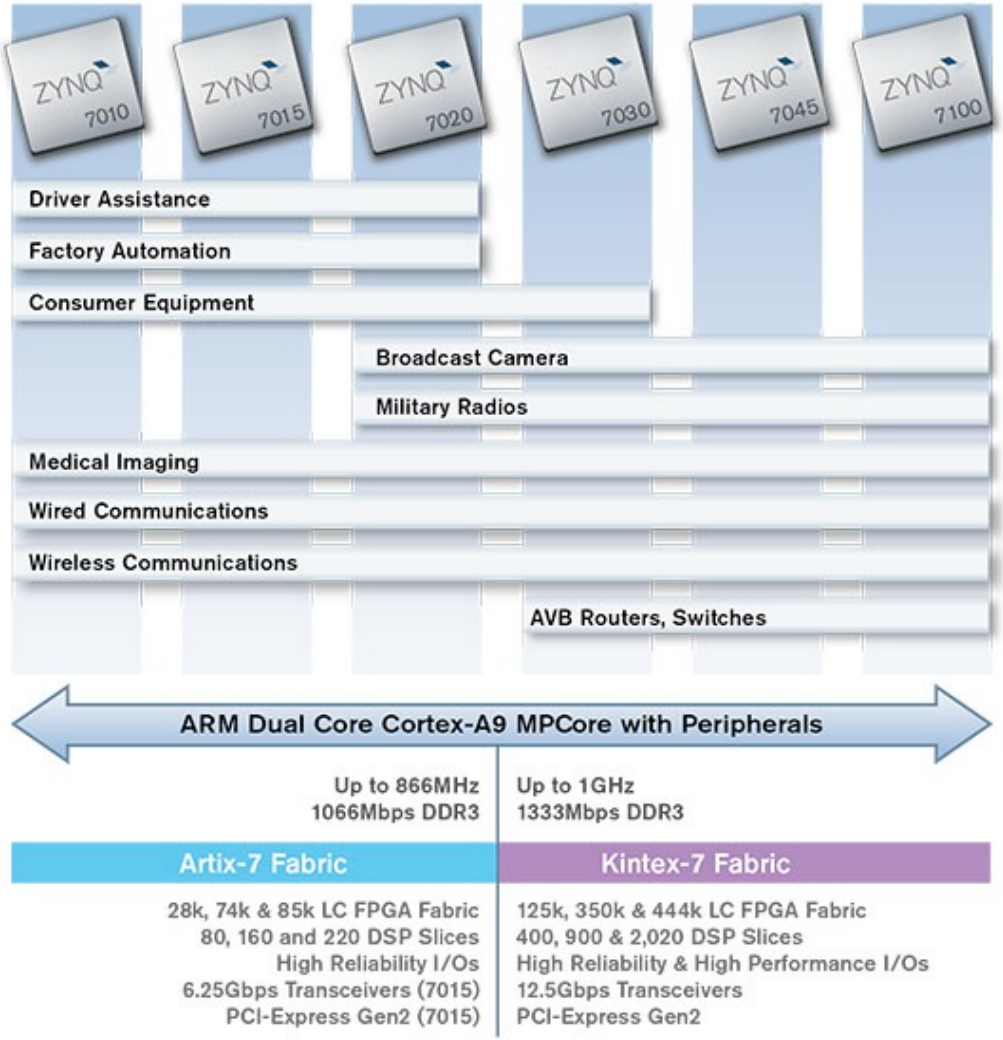


Intel Stellarton Atom E600+FPGA promises flexible embedded devices

http://www.slashgear.com/intel-stellarton-atom-e600fpga-promises-flexible-embedded-devices-14102251/

Architectures du futur pour les systèmes embarqués (2/2) ?





Conclusion

Conclusion

- Comme nous venons de la voir, le domaine de la conception des circuits numériques réels est complexe de par la multitude des contraintes à gérer simultanément.
- Beaucoup de points ont été occultés pour ne pas vous effrayer...
  - ➡ Reconfiguration dynamique des circuits (mise à jour ou adaptation à l'environnement),
  - ➡ Gestion de la sécurité des systèmes (données et propriété intellectuelle),
  - ➡ Automatisation des étapes de conception à plus haut niveau,
- Les connaissances indispensables appartiennent à différents domaines : électronique, informatique et traitement du signal,
- La création d'un circuit passe par une multitude de raffinements successifs.
- Dans la suite de ce cours, nous allons nous attacher à apprendre un langage de modélisation de haut niveau nommé SystemC.