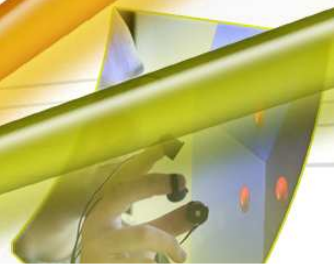
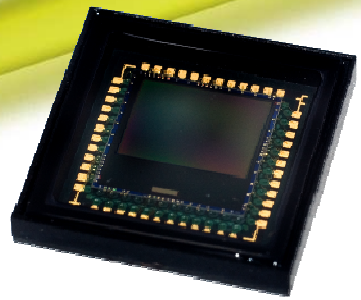
eISP : Conception et validation d'un processeur programmable de traitement du signal à faible consommation et à faible empreinte silicium : application à la vidéo HD sur téléphone mobile

Laboratoire Calculs Embarqués – CEA LIST



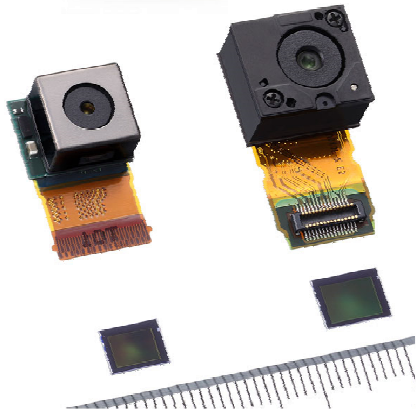
Mathieu THEVENIN
Mathieu.Thevenin@CEA.fr



Contexte : le marché de la vidéo embarqué

Les modules vidéo embarqués sont largement répandus

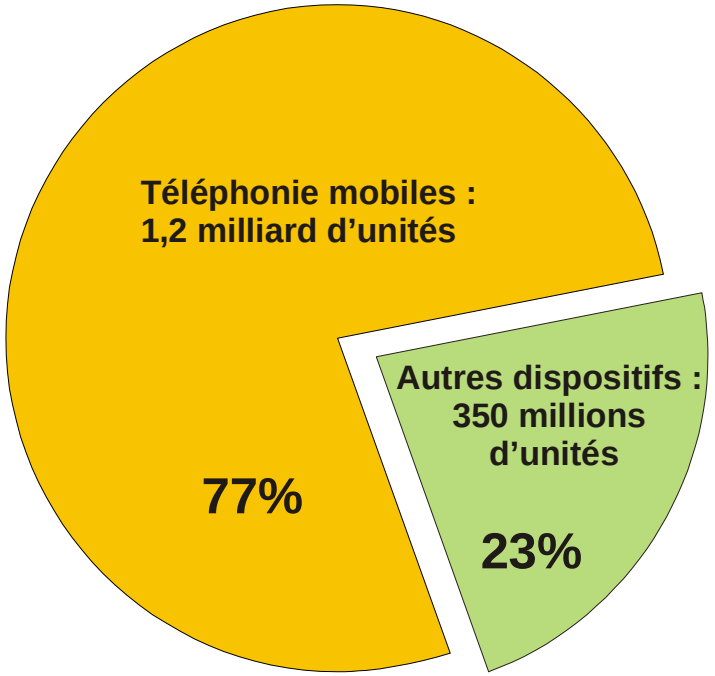
- Téléphonie mobile
- PDA et smart phones
- Caméscopes
- Appareils photos
- Vidéo-surveillance
- Automobile



Quelques chiffres (2009)

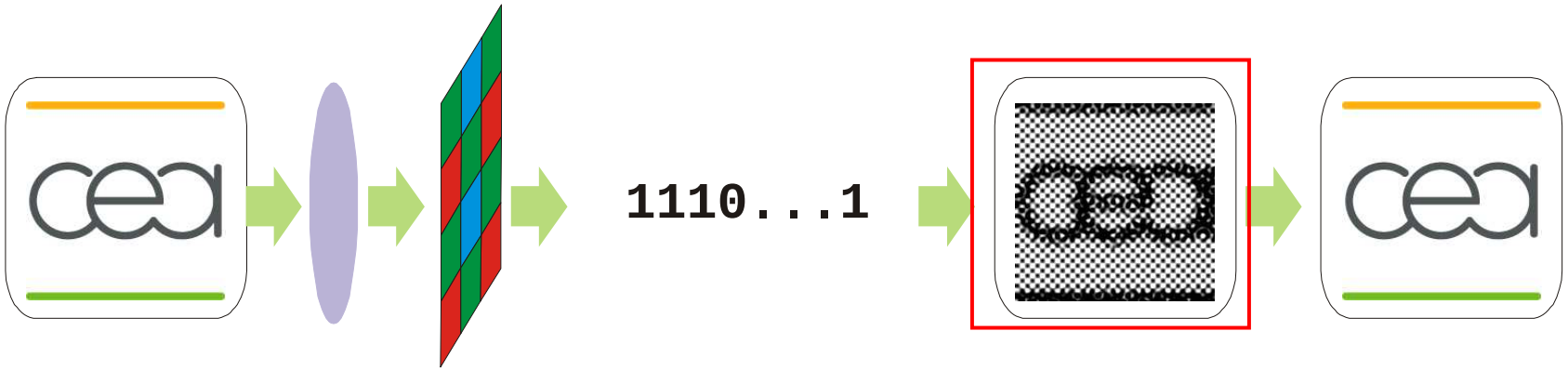
- 6 à 8 milliards de dollars
- Coût de production 5 à 10 dollars
- ~90% du marché <4MPx

Objectif de coût de revient à court terme : $1



Contexte : nécessité de reconstruire les images

- **La demande**
  - ✓ **De la haute résolution**
    - Photo > 10MPixels
    - Vidéo Full HD 1080p
  - ✓ **Une forte autonomie**



Il faut reconstruire les images issues des capteurs

Objectifs et contraintes

Un besoin en flexibilité

- ✓ **Aujourd'hui des solutions dédiées existent**
  - → Problème de réutilisation des IPs
  - → Problème d'évolution des algorithmes

Proposer une architecture de traitement d'image flexible

Des contraintes fortes (en téléphonie mobile)

- la consommation électrique
  - → ½ Watt
- la surface silicium
  - → 2 à 3 mm² en technologie 65 nm
    - » Interdit toute mémoire d'image
- Capacité de calcul
  - →Image brute HD 1080p 25fps : 52MPx/s
  - → Plusieurs centaines d'opérations par pixel

Plan de cette présentation

- Introduction et contexte

✓ Les traitements en sortie d'imageur

✓ Un état de l'art des architectures

✓ L'analyse des algorithmes de traitement d'image

1. Les choix architecturaux

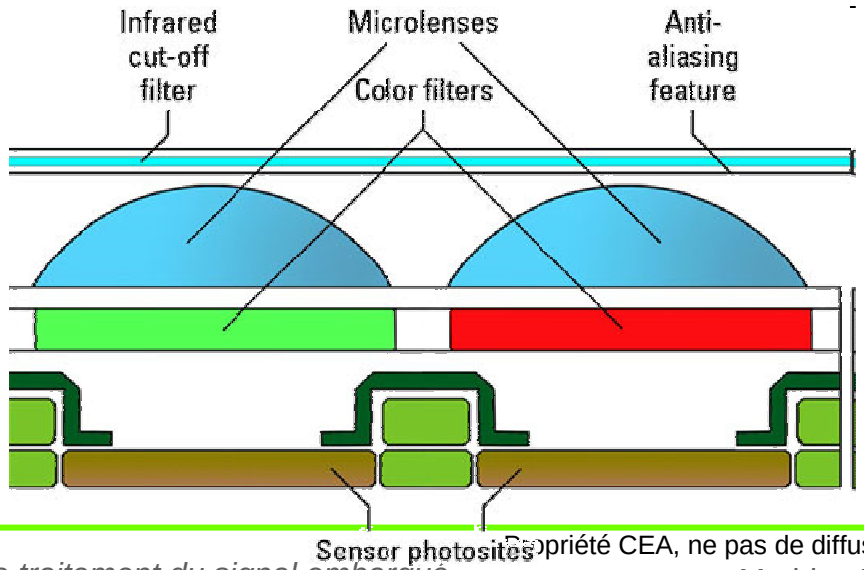
2. La validation et les résultats d'implémentation

- Conclusions et perspectives

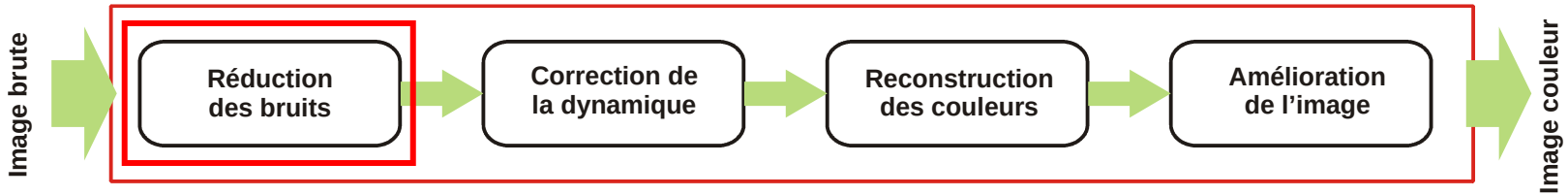
Les traitements de reconstruction d'image

- **Eléments impactant la qualité de l'image**
  - ✓ Qualité des capteurs
  - ✓ Qualité optique
  - ✓ Exposition
  - ✓ Filtres logiciels

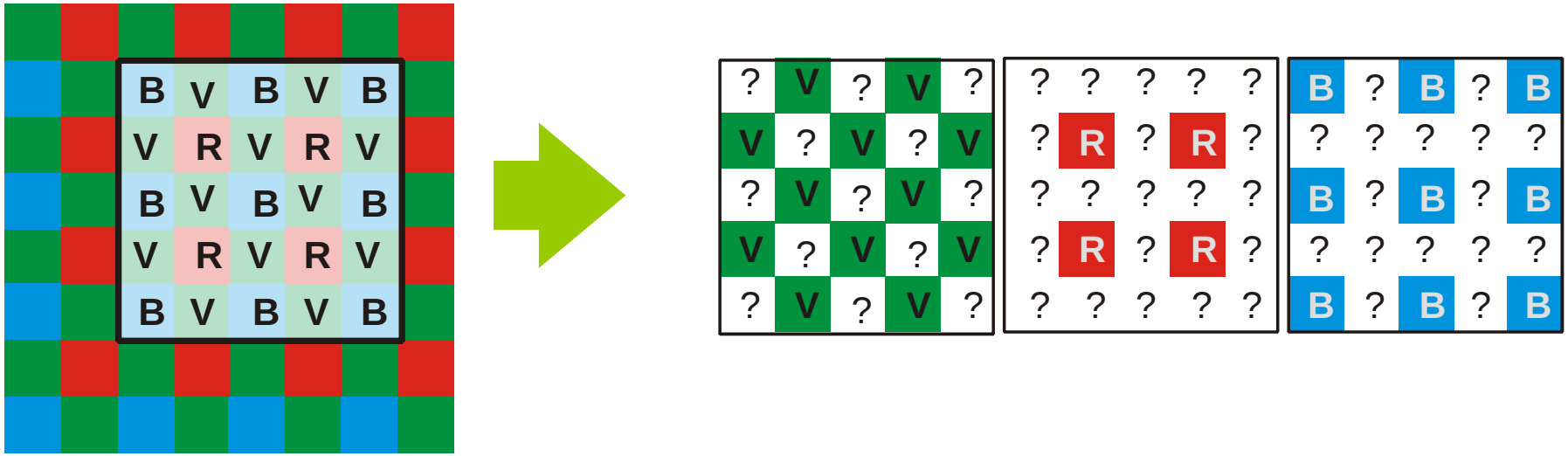
- **Les couleurs des images doivent être reconstruites**
  - ✓ Systèmes monocapteurs
  - ✓ Capteurs mesurent un niveau de luminance
    - → Utilisation d'un filtre de couleur (Bayer ou autre)



Les traitements de reconstruction d'image



➢ Reconstruction des couleurs :



Présentation animée de la chaîne algorithmique

non supportée par le format PDF

Les traitements de reconstruction d'image

- **Une grande variété de traitements**
  - ✓ De complexité variable
  - ✓ De nouveaux algorithmes apparaissent régulièrement
  - ✓ Aucune standardisation des traitements

  Besoin de flexibilité

- **Définition de plusieurs chaînes algorithmiques « de référence »**
  - ✓ De complexité variable
  - ✓ Différentes approches algorithmiques

Plan de cette présentation

- Introduction et contexte

✓ Les traitements en sortie d'imageur

✓ Un état de l'art des architectures

✓ L'analyse des algorithmes de traitement d'image

1. Les choix architecturaux

2. La validation et les résultats d'implémentation

- Conclusions et perspectives

Solutions matérielles existantes

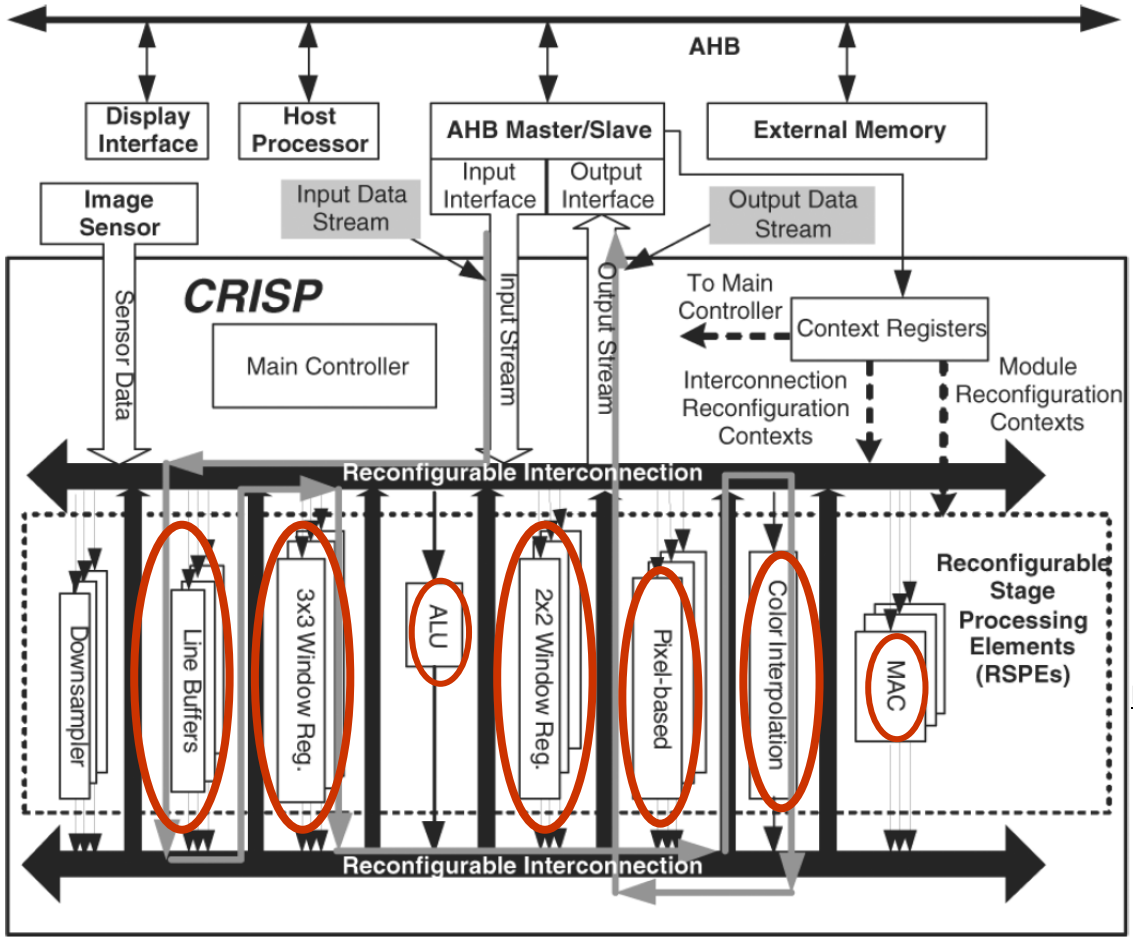
- **Les solutions industrialisées aujourd'hui**
  - ✓ Des composants dédiés
  - ✓ Des IPs associées à des processeurs généralistes
  - ✓ Des plates-formes « multifonctions »

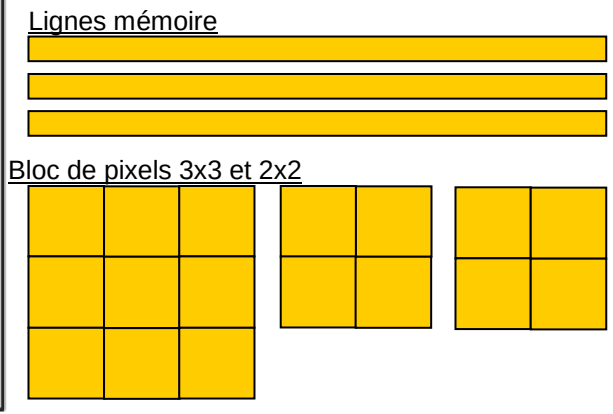
- **Les solutions académiques**
  - ✓ Multi-coeurs programmables
  - ✓ Simple/multi-cœurs avec extensions configurables
  - ✓ Reconfigurable à différentes granularités

- **Comment introduire de la flexibilité ?**
  - ✓ Le reconfigurable
  - ✓ Le programmable

Solutions matérielles existantes : CRISP

- **CRISP (Coarse Grained Image Stream Processor) (Univ. Taïwan)**
  - ✓ Mode flux
  - ✓ Reconfigurable gros grain





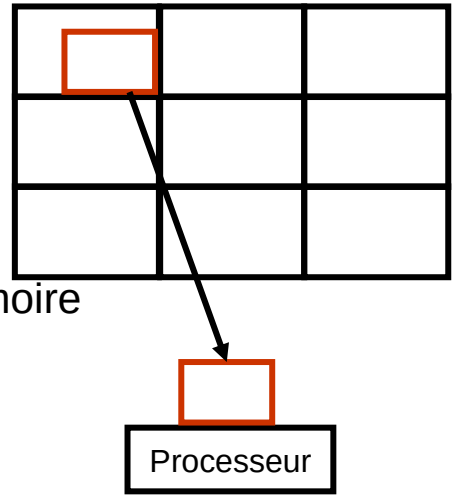
Solutions matérielles existantes : Hive'Flex

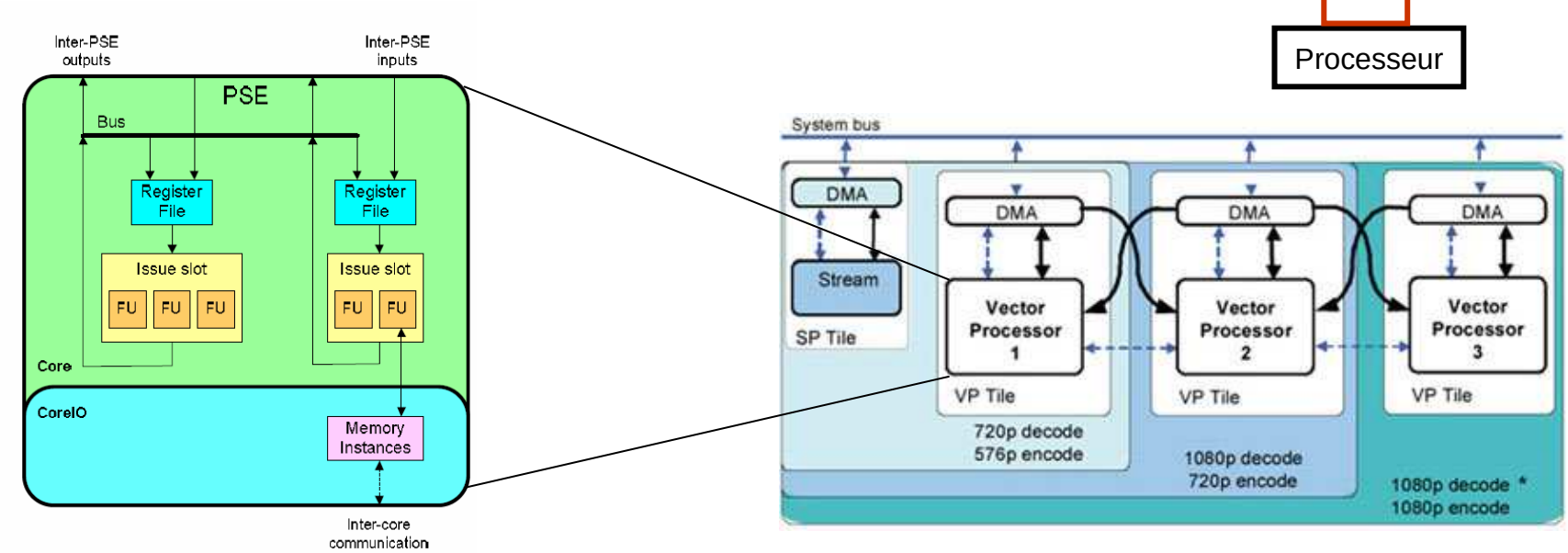
- **Hive'Flex (Silicon Hive : Spin Off Philips Research Labs)**
  - ✓ **Programmable vectoriel 32 éléments (VSP 2308)**
  - ✓ **VLIW 2 à 16 voies**
  - ✓ **Interconnect reconfigurable**
  - ✓ **Hiérarchie mémoire à 2 niveaux**

Capacité de calcul : 75 à 135 GOPs

Surface silicium : 5 mm² en 65nm hors mémoire

Consommation électrique : 600 mW





Solutions matérielles existantes : comparaison

➢ Comparaison entre CRISP et Hive'Flex

✓ Points communs :

- Faible surface silicium < 5mm²
- Faible consommation < 600 mW

- Hiérarchie mémoire

→Pour alimenter les unités fonctionnelles en données

	Surface mm²	Conso. mW	Capacité GOPs	Capacité GOPs/mm²	Capacité MOPs/mW	**Capacité MOPs/mW/mm²**
Hive'Flex VSP 2300 32x7	5,00	600	135	27,00	225,00	**45,00**
CRISP	1,50	218	13	8,67	59,63	**39,76**

Plan de cette présentation

- Introduction et contexte

✓ Les traitements en sortie d'imageur

✓ Un état de l'art des architectures

✓ L'analyse des algorithmes de traitement d'image

1. Les choix architecturaux

2. La validation et les résultats d'implémentation

- Conclusion et perspectives

Analyse des algorithmes : approche utilisée

- **Cadence importante : 52MPixels / seconde**
- **Plusieurs dizaines/centaines d'opérations par pixel**
  - Hypothèses :
    - → Un seul processeur capable d'exécuter 1 opération par cycle
    - → Un traitement de 100 opérations/pixel
  - Conséquence
    - → Fréquence à 5,2 GHz
      - » **IMPOSSIBLE**
- **Exploiter au maximum le parallélisme**
  - Parallélisme spatial
  - Parallélisme temporel
  - Parallélisme de tâche
  - Parallélisme d'instruction
- **Adéquation Algorithme Architecture**

Analyse des algorithmes : approche utilisée

- **Analyse algorithmique à différents niveaux**
  - ✓ Niveau global
  - ✓ Niveau fonctionnel
  - ✓ Niveau opérations
  - ✓ Analyse dynamique

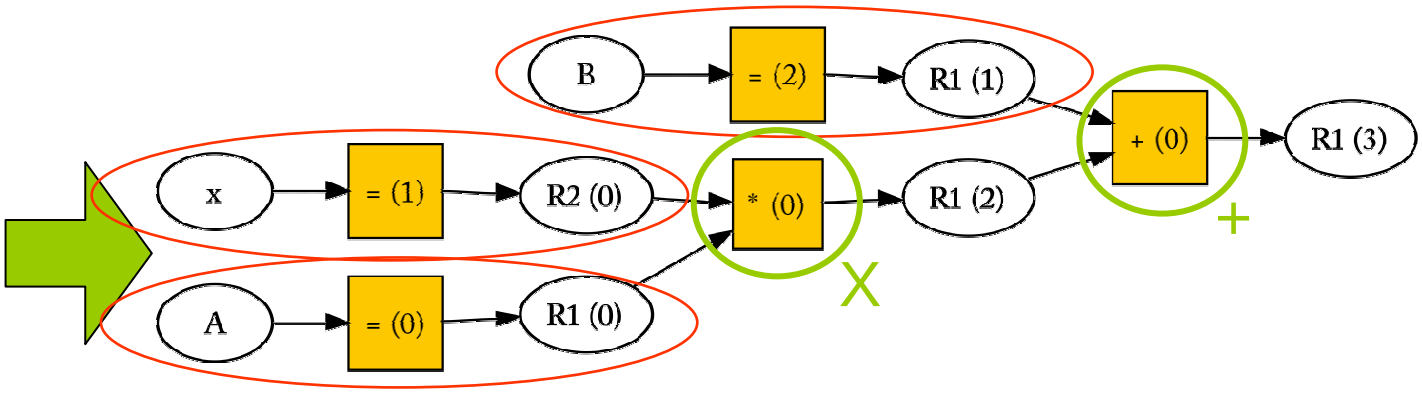
- **Conception de la chaîne d'outils MAsS**
  - ✓ Indépendance de l'étude par rapport à l'architecture
  - ✓ Identification des opérations utilisées
    - Nature
    - Dynamique
    - Fonction
  - ✓ Etude de l'impact d'extensions sur l'exécution des algorithmes
    - Macro-opérations
    - Opérateurs vectoriels
    - Générateur d'adresses etc.
  - ✓ Cibler l'étude algorithmique à des cœurs de boucles
    - Entrée
      - → Algorithme mixte assembleur MAsS et C/C++
      - → Fichier de définition comportementale des instructions
    - Sortie
      - → Résultats de l'exécution (image etc.)
      - → Rapport de profilage
      - → Graphe d'exécution

Analyse des algorithmes : graphes d'exécution

- **Construction du graphe d'exécution**
  - ✓ A chaque appel d'une opération
    - → 1 nœud par opération
    - → 1 nœud par ressource
- **Le graphe est généré pour une exécution du programme**
  - ✓ Dépend des données utilisées
  - ✓ De taille importante
    - → Plusieurs millions/milliards de nœuds
  - ✓ Le parallélisme doit être explicités
    - → Renommage automatique des ressources

$$Y = A \times x + B$$

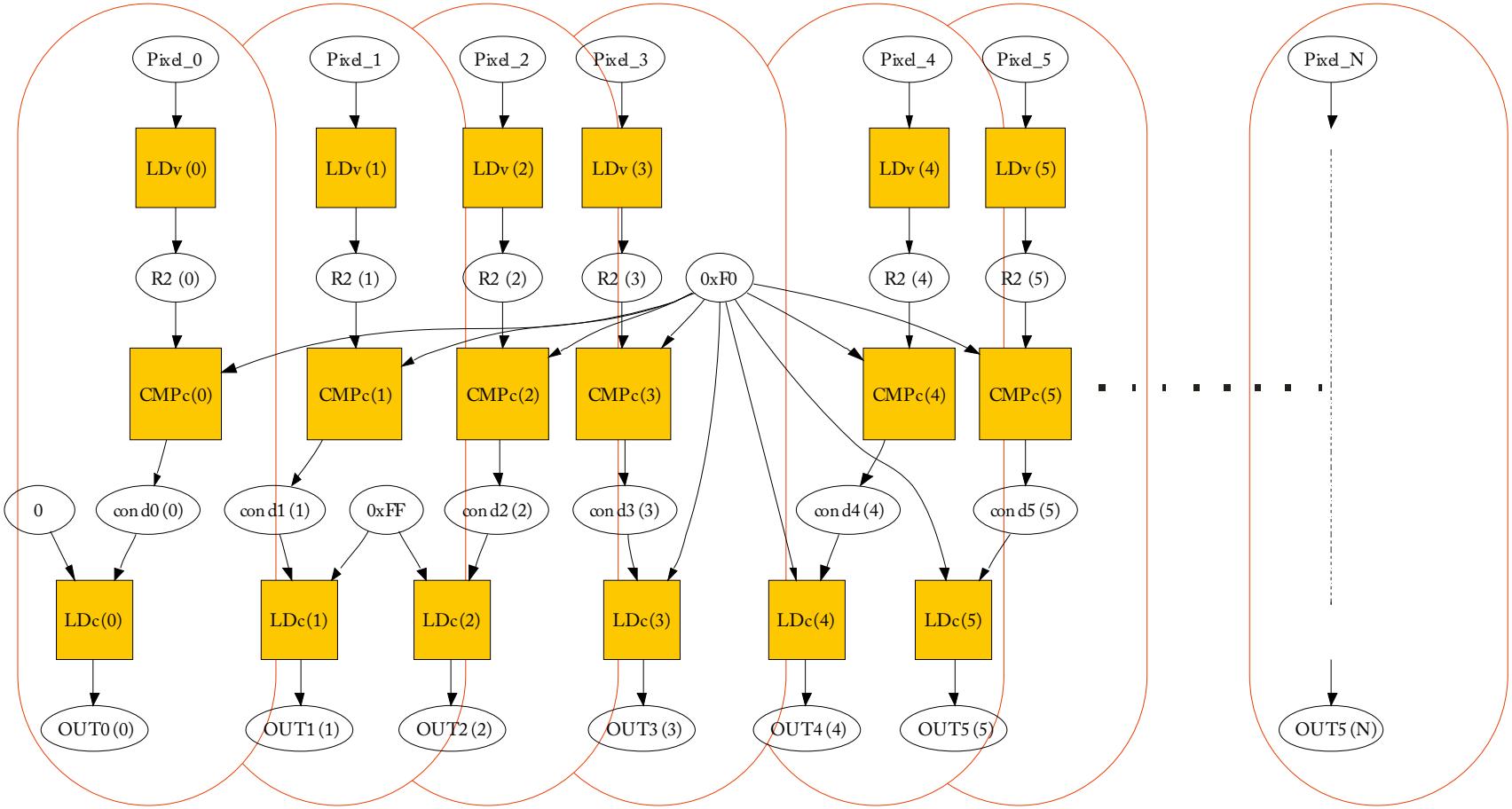
```
R1 ← A
R2 ← x
R1 ← R1 × R2
R2 ← B
R1 ← R1 + R2
```



Analyse des algorithmes : parallélisme spatial

Extraction de sous-graphes

- ✓ Associés aux traitements de chaque pixel
  - Similarités/disparités dans les traitements



Analyse des algorithmes : parallélisme d'instruction

Mise en évidence du parallélisme d'instruction (ILP)

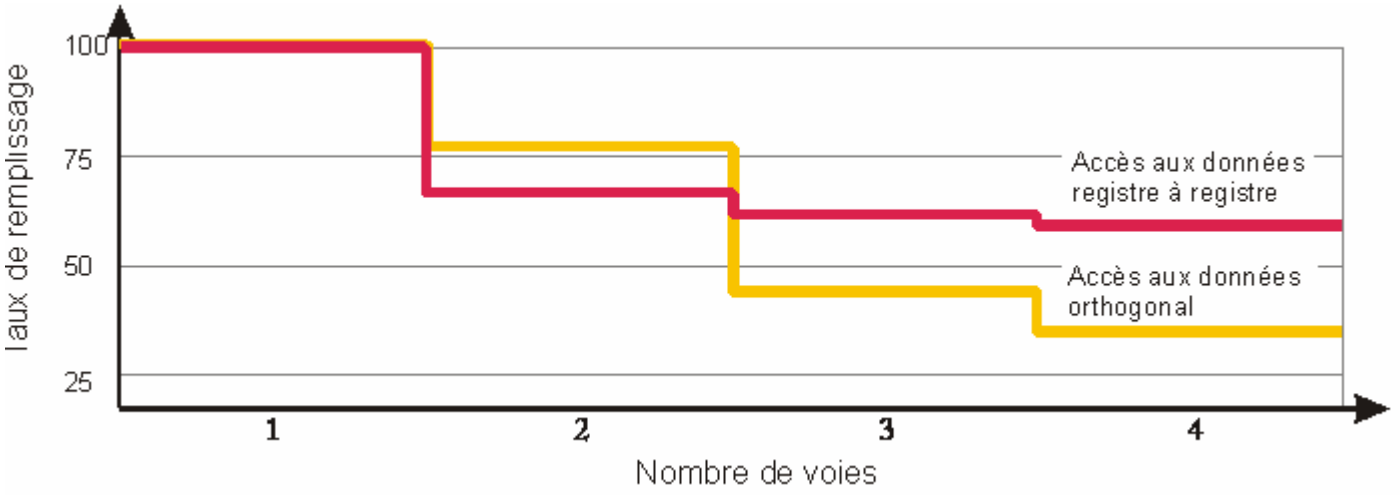
- ✓ **Littérature : ILP = 2 à 4**

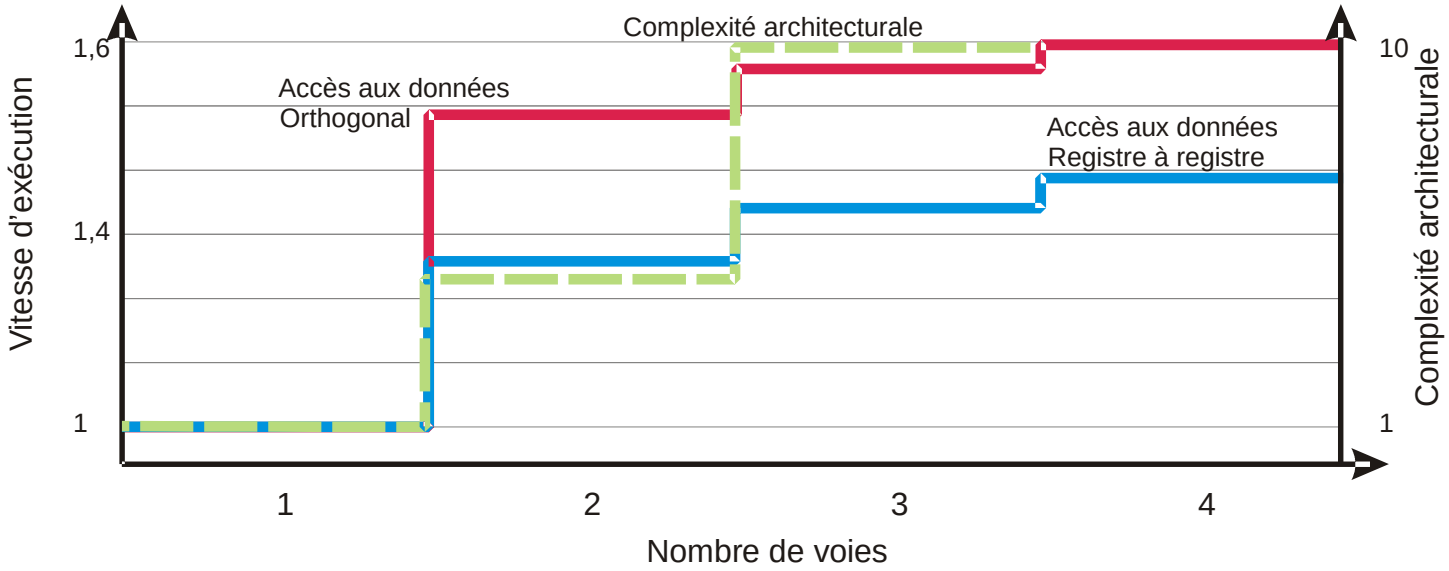
- ✓ **Approche statique : ILP = 2 à 3**
  - Développée au laboratoire, adaptée pour nos besoins
  - A partir d'un code RISC compilé (SPARC)
  - Allocation des ressources sur N voies

- ✓ **Approche dynamique : ILP = 2 à 3**
  - Par analyse des graphes d'exécution
    - → Méthode adaptée du « coloriage du graphe d'interférence »
      - » Allocation des ressources selon la disponibilité
      - » Permet d'obtenir le nombre de voies d'un processeur VLIW
      - » Extraction des opérateurs potentiellement concurrents

- ✓ **Les notions prises en compte**
  - Vitesse d'exécution
  - Complexité architecturale
    - → ex. surface de la RF, nombre et surface des opérateurs …
  - Taux de remplissage moyen des voies
  - Mode d'accès aux données étudiés
    - → Registre à registre : opérandes en file de registres
    - → Orthogonal : opérandes de source multiples

Analyse des algorithmes : parallélisme d'instruction





Analyse des algorithmes : choix des opérateurs

Analyse statistique au niveau opérateurs

✓ Mutualisation des opérateurs

- Identification statistique des opérateurs utilisés « simultanément »
  - → Addition/soustraction
  - → Ecriture d'une donnée en registre
- Surface 1,5 fois moindre (à flexibilité équivalente)

✓ Mode d'accès aux données

- un opérateur sur 14 nécessite 2 opérandes de sources externes simultanément
  - → Un seul opérande peut donc être issu d'une source quelconque
    - » Mode d'accès « orthogonal partiel » : réduit le nombre de registres nécessaires de 20% à 30%

Analyse des algorithmes : parallélisme de tâches

Séparation par groupes fonctionnels

1. Identification des **opérations de contrôle**
   - » Gestion des boucles;
   - » sauts (sous programmes etc.).

2. Identification des **opérations d'accès aux données**
   - » Parcours de l'image
   - » Parcours de fenêtres glissantes

3. Identification des **opérations de traitement d'image**
   - » Celles qui contribuent directement au résultat

Analyse des algorithmes : parallélisme de tâches

Fonctions de traitement d'image

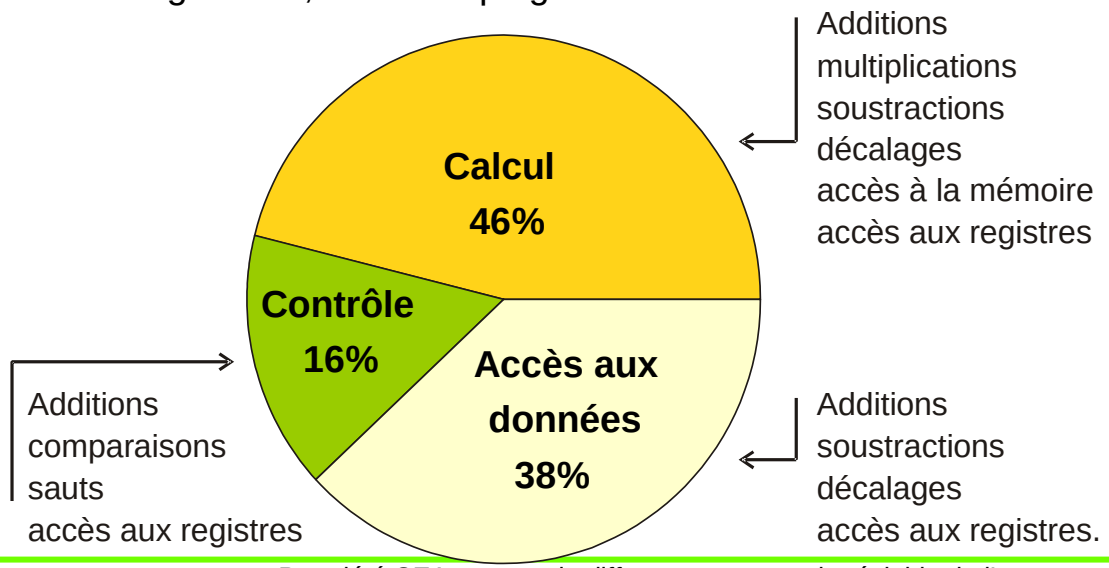
- ✓ Différentes mais basées sur des opérations communes
- ✓ Opérations élémentaires entières signées
  - 10 à 30 GOPs requis pour du traitement HD 1080p

Fonctions d'accès aux données

- ✓ Accès au « voisinage » du pixel à traiter
- ✓ Images brutes, couleurs, YUV/RGB, complexes etc.
- ✓ Des données de dynamique de 8 à 16 bits

Fonctions de contrôle

- ✓ Evénementiel : l'arrivée d'un pixel, début d'image etc.
- ✓ Segment de code en fonction de la nature ou de la valeur de la donnée (IF/CASE)
  - → traitement d'image brute, démosaïquage…



Analyse des algorithmes : parallélisme de tâche

Analyse de différentes chaînes algorithmiques

- Répartition moyenne des opérateurs
  - → Additions/soustractions (30%)
  - → Chargements en registre (30%)
  - → Multiplications (18%)
  - → Autres (2%)
- Dynamique des opérateurs
  - → Obtenue par étude expérimentale sur jeu de données réel
  - → Variable selon le type de capteur, algorithmes etc.
    - » Etude exhaustive impossible

✓ Cas d'une chaîne algorithmique complexe :

- Besoin total en ressources de calcul : 70,89 GOPs
- Partie dédiée au calcul : 23,34 GOPs

Plan de cette présentation

- Introduction et contexte

✓ Les traitements en sortie d'imageur

✓ Un état de l'art des architectures

✓ L'analyse des algorithmes de traitement d'image

1. Les choix architecturaux

2. La validation et les résultats d'implémentation

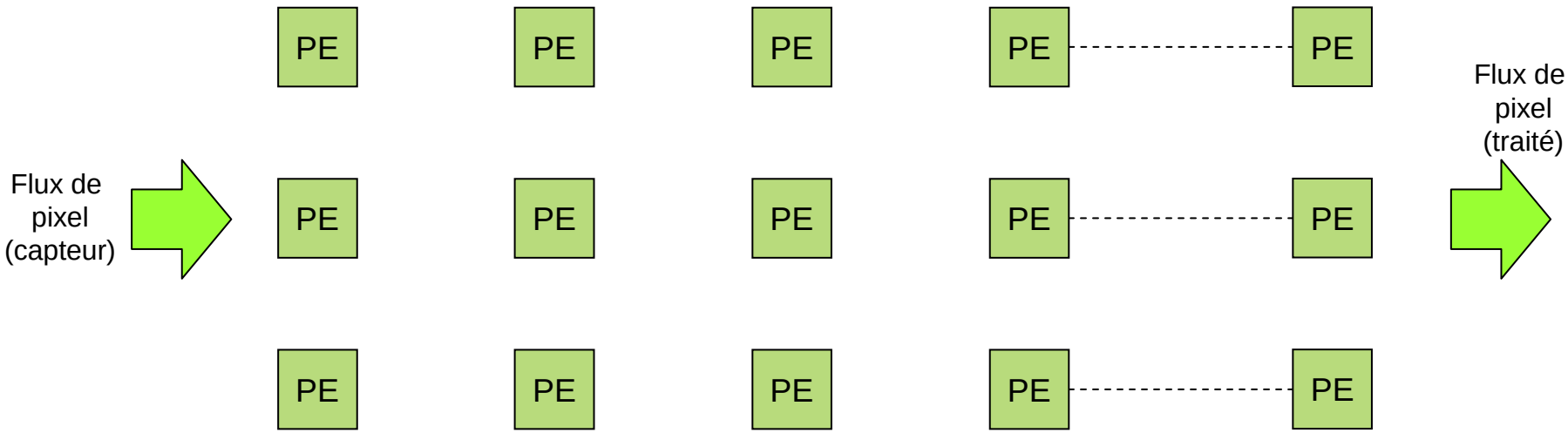
- Conclusion et perspectives

L'architecture eISP

- **Analyse des algorithmes :**
  - ✓ **Parallélisme temporel**
    - Enchaînement des traitements
  - ✓ **Parallélisme spatial**
    - SIMD
    - Multi-SIMD
  - ✓ **Parallélisme d'instruction**
    - VLIW
  - ✓ **Parallélisme des tâches**
    - Accès aux données
    - Contrôle
    - Calcul des résultats

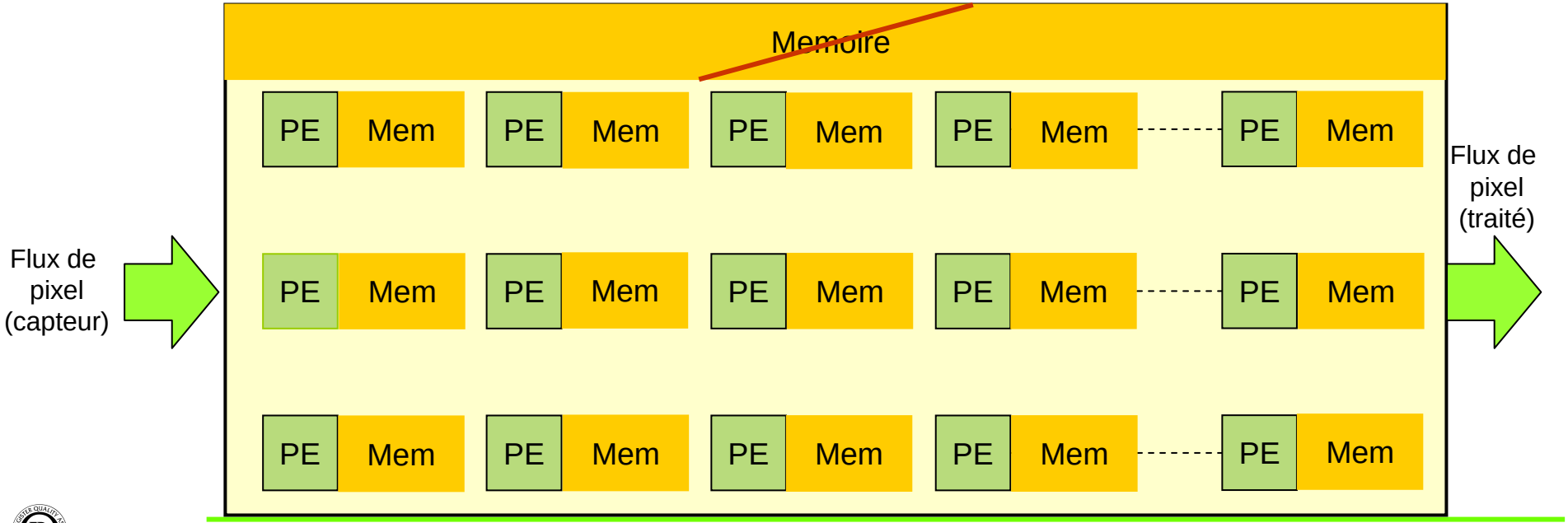
L'architecture eISP : comment organiser les PEs

- **Capacité de calcul requise : 20 à 30 GOPs**
- **Utilisation de plusieurs « petits » processeurs**
  - **Ex : sur 1,5mm² ~50 UAL + registres (PE élémentaires)**
- **Comment organiser les processeurs ?**



L'architecture eISP : la tuile de calcul

- **Un seul regroupement de processeurs en une tuile de calcul**
  - Dépendances de données entre traitements
    - → Aller et retour en mémoire
  - Mémoire centrale
    - → Mémoire d'image nécessaire : dépassement du budget
    - → Accès multiples
  - Mémoire distribuée
    - → Communications entre processeurs

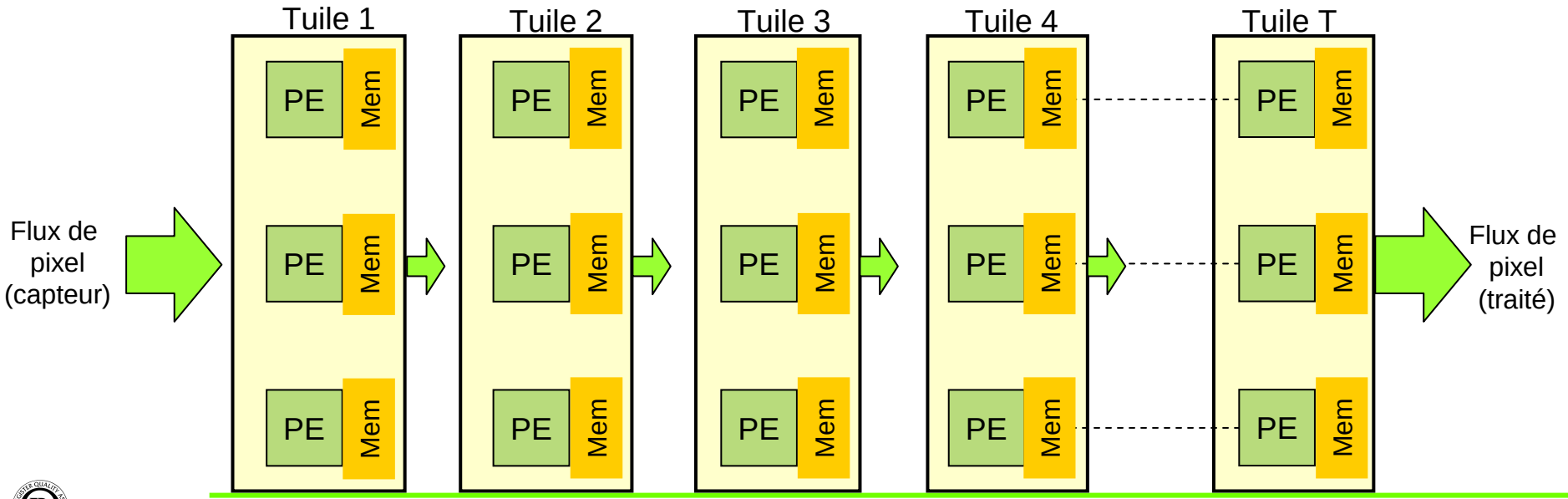


L'architecture eISP : la tuile de calcul

Plusieurs tuiles de calcul homogènes ?

Au moins 4 tuiles de calcul sont nécessaires

- Surface mémoire importante
- Portage des algorithme complexes ?
  - → Rendre configurable le nombre de processeurs par tuile
    - » Coûteux en surface silicium

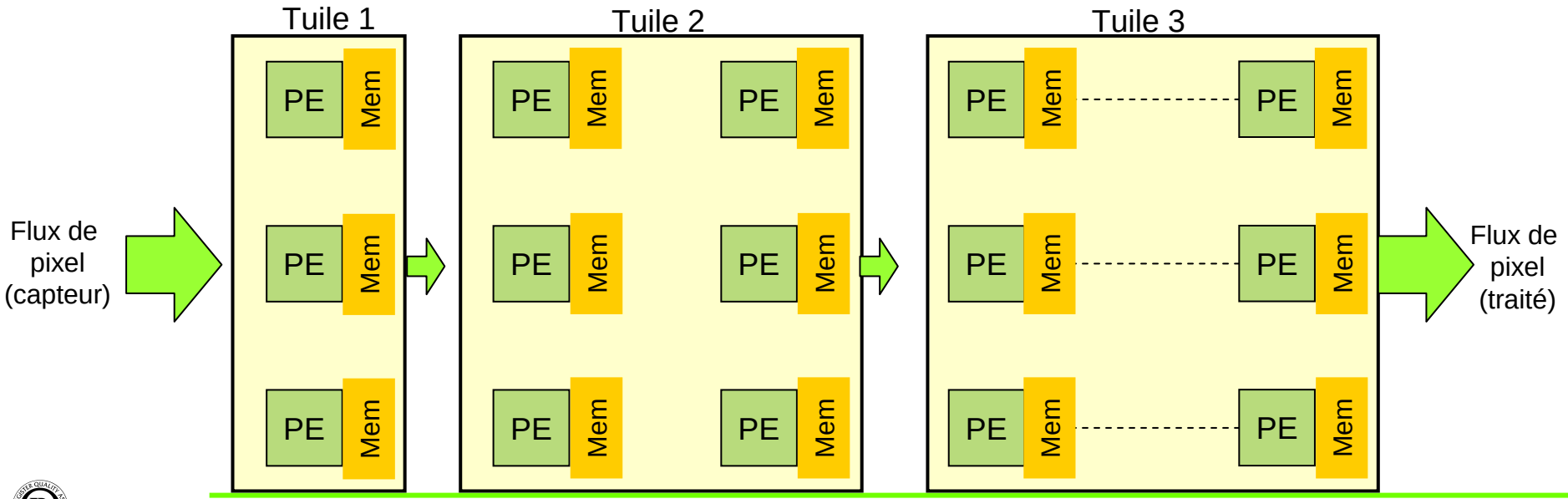


L'architecture eISP : la tuile de calcul

Plusieurs tuiles de calcul hétérogènes ?

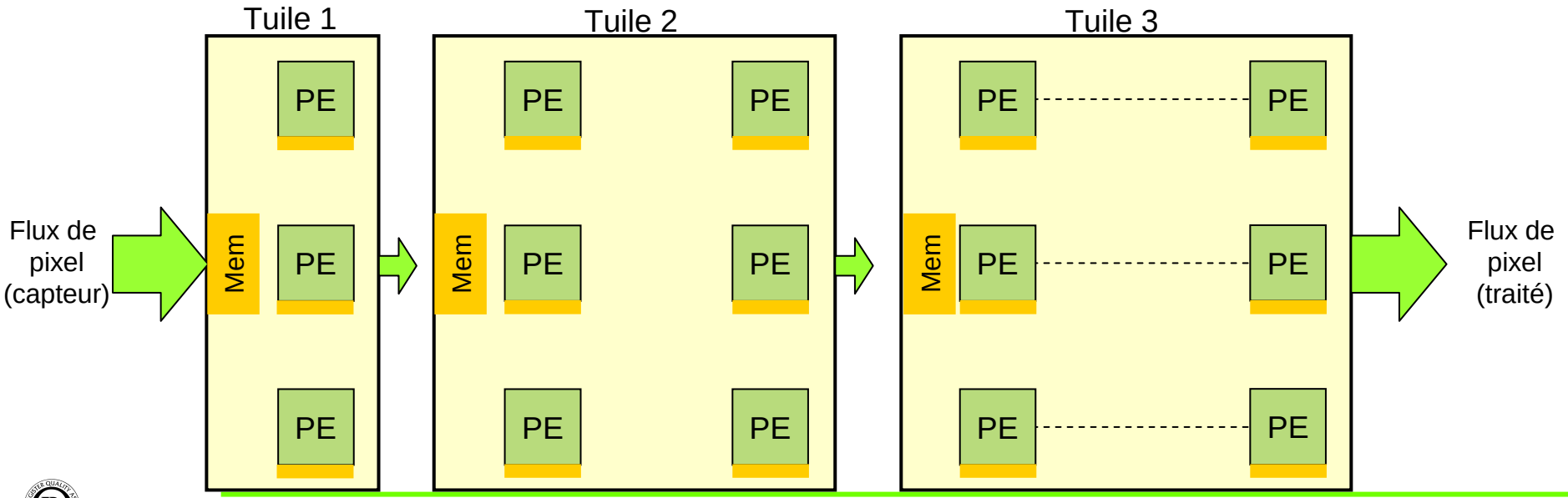
Supporter des algorithmes de différentes complexités

- Surface mémoire importante
- Compromis entre
  - → Duplication des données
  - → Communication entre processeurs



L'architecture eISP : la tuile de calcul

- **Plusieurs tuiles de calcul hétérogènes ?**
  - **Avec hiérarchie mémoire**
    - → Eviter la duplication d'information
      - » Mutualiser les données communes
    - → Limiter les communications
      - » Stocker les données au plus près du processeur
  - ✓ **Temps disponible pour le calcul (pour une tuile)**
    - $Nb_{Cycles} = F_{Proc} / F_{Pixel} \times Nb_{Proc}$

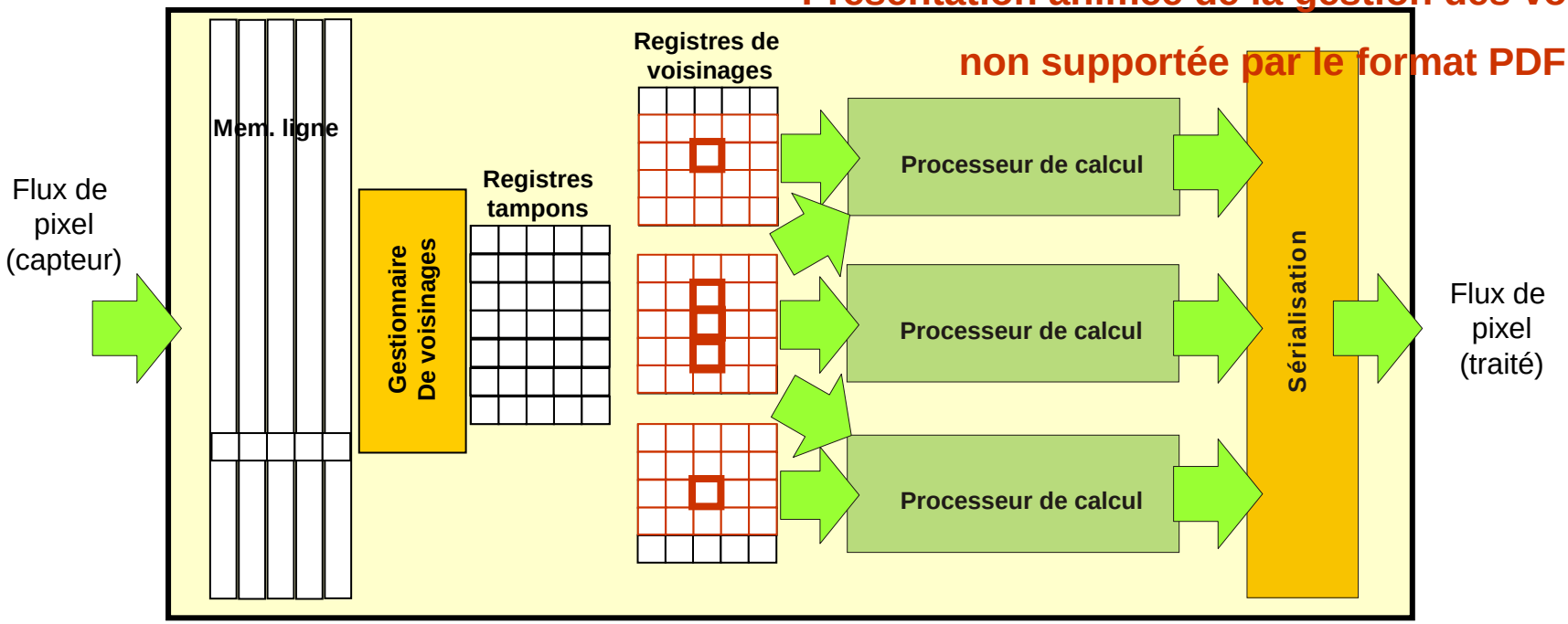


L'architecture eISP : le gestionnaire de voisinages

- **Redondance d'information**
  - ✓ Mutualisation des ressources
- **Transfert des données depuis la mémoire vers les registres de voisinage**
  - ✓ Utilisation de mémoires simple port (surface)
  - ✓ Copie d'une « colonne » à l'arrivée de chaque nouveau pixel
    - Un accès par pixel et par ligne (consommation)
  - ✓ Registres de voisinages constamment disponibles pendant
    - $Nb_{Cycles} = F_{Proc} / F_{Pixel} \times Nb_{Proc}$

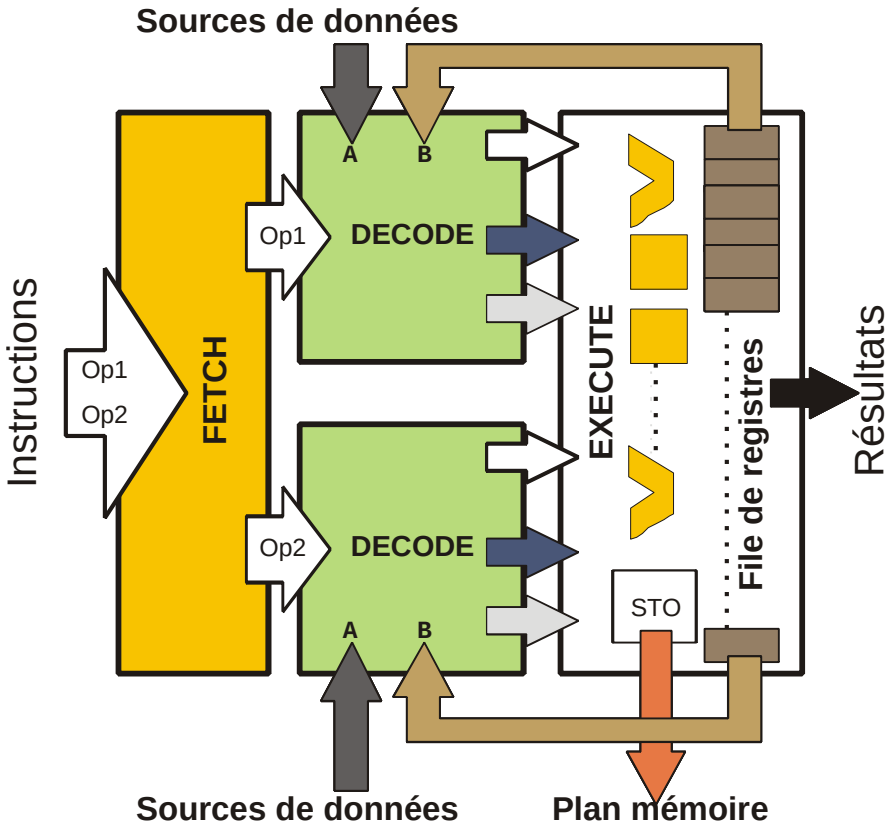
Présentation animée de la gestion des voisinages

non supportée par le format PDF



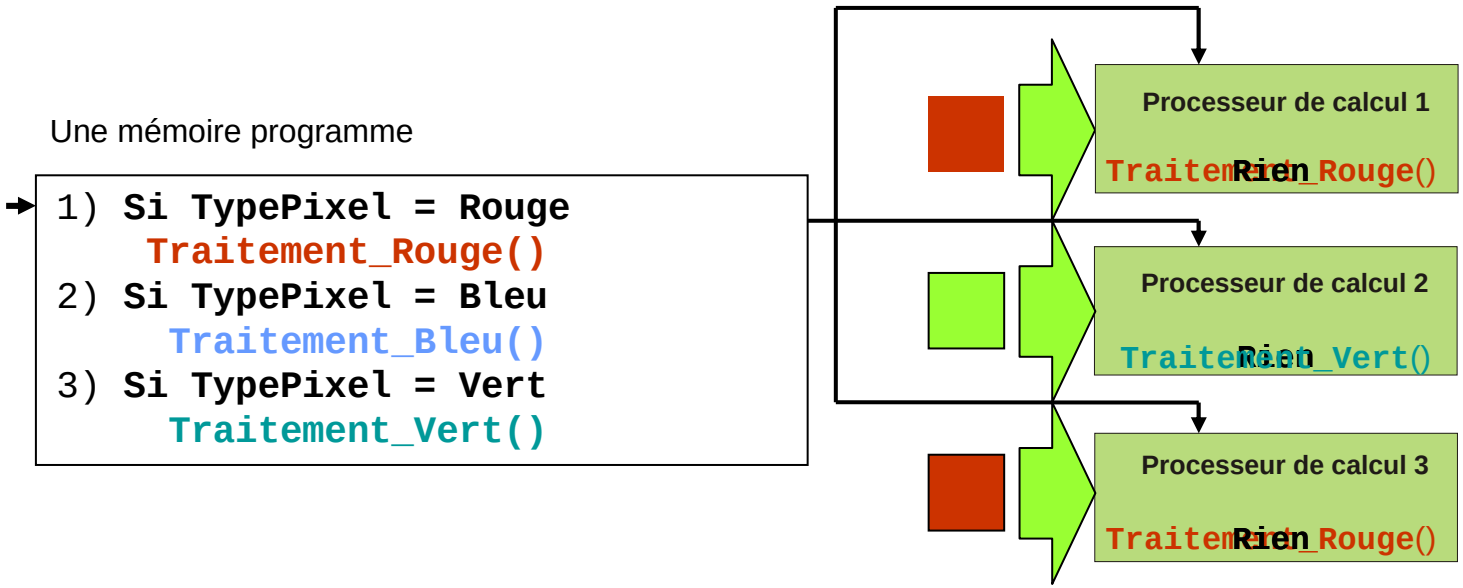
L'architecture eISP : le processeur « SplitWay »

- **Le processeur SplitWay**
  - ✓ **Limiter la surface**
    - Mutualisation d'opérateurs
    - RF réduite
    - Nombre d'étages
    - Mémoire de travail SP
  - ✓ **Limiter la consommation**
    - IS orthogonal
    - Nombre d'étages
    - Modes veille
    - Clock gating
  - ✓ **Flexibilité et performance**
    - Taille mémoire de travail
    - IS orthogonal
    - Mutualisation d'opérateurs
      - → IF/ELSE 1 cycle
    - 2 opérations par cycles



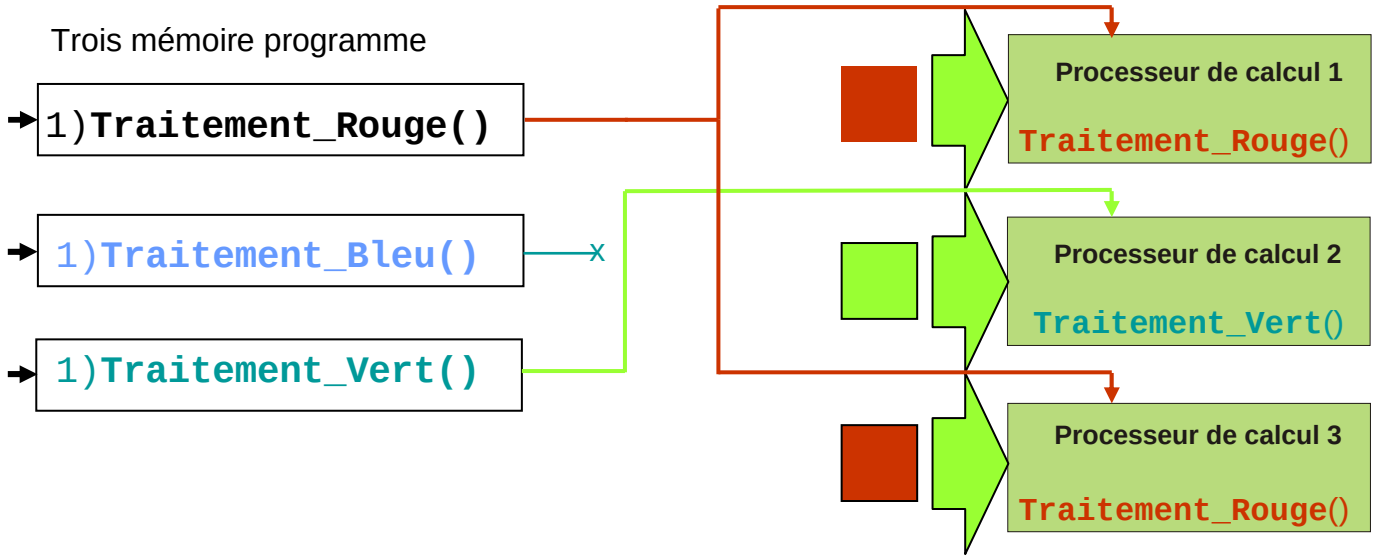
L'architecture eISP : le modèle d'exécution

- **Le cas SIMD classique**
  - ✓ **Tous les processeurs exécutent les mêmes instructions**
    - → Problème
      - » Traitement des prédicats de type « case »
      - » Traitement d'images brutes (alternance de couleurs)



L'architecture eISP : le mode Multi-SIMD

- **Le cas Multi-SIMD**
  - ✓ **Les processeurs exécutent des instructions différentes**
    - → Support optimal des prédicats de type « case »
    - → Support optimal d'images brutes (alternance de couleurs)



Information de commande associée au pixel nécessaire

L'architecture eISP : les métadonnées

Les métadonnées

✓ Mots de données composites

- Rouge vert et bleu
- Nombre complexes
- Pixel + résultat précédent

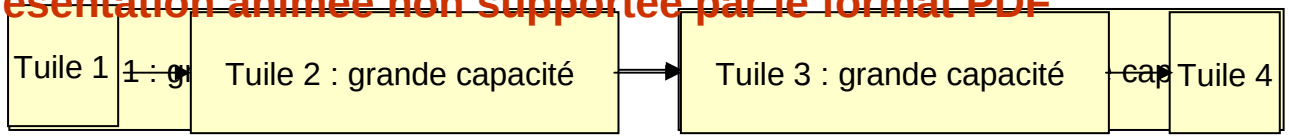
✓ Câbler la lecture des mots de données

- → Masques configurables par le programmeur
- → Accès aux données câblé
- → Programmation de type « registres sécables »

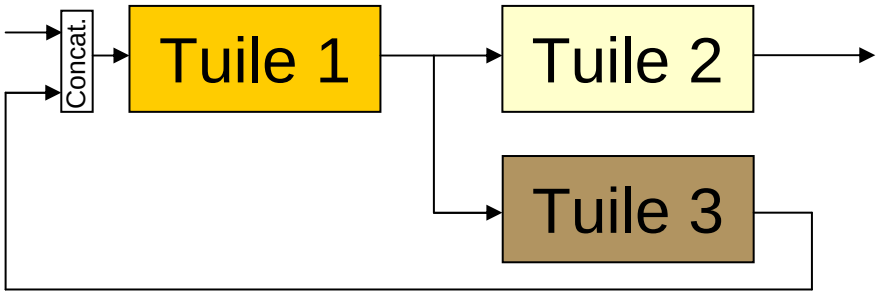
L'architecture eISP : le module de communication

- **Communication entre tuiles de calcul**
  - ✓ **Tuiles hétérogènes**
    - » Ressources de calcul
    - » Fréquences de fonctionnement

Présentation animée non supportée par le format PDF



→ Nécessité de rendre l'enchaînement paramétrable



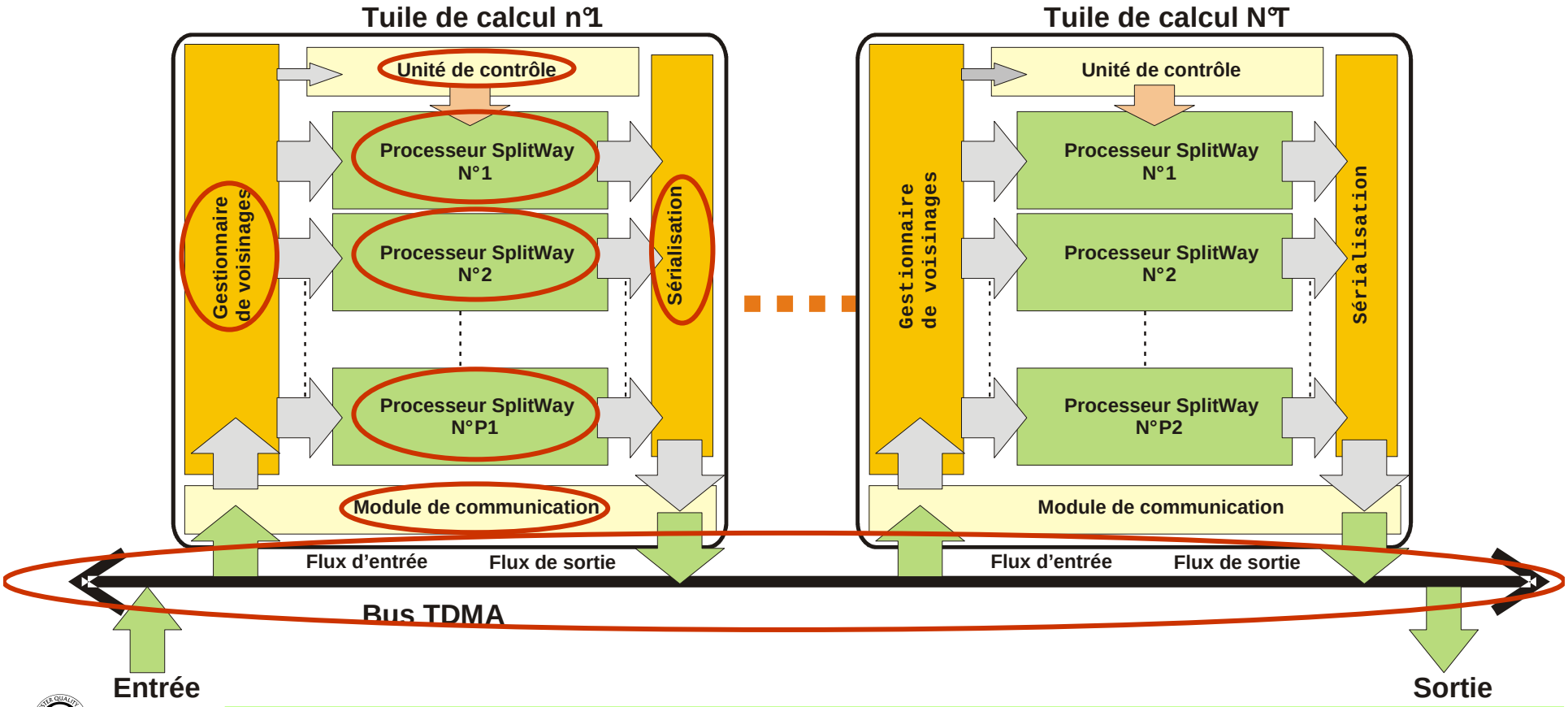
  - ✓ **Utilisation d'un bus TDMA**
    - L'horloge pixel est l'élément commun à toutes les tuiles
      - → Diviser cette horloge en slots temporels
        - » Une écriture par slot et par pixel
        - » Une lecture par slot et par pixel
      - → Maintient l'ensemble des tuiles synchronisées

L'architecture eISP : vue générale

Présentation animée

non supportée par le format PDF

- **L'architecture eISP vue dans son ensemble**
  - ✓ Les processeurs SplitWay
  - ✓ L'unité de contrôle + les mémoires programme
  - ✓ Le gestionnaire de voisinages + la sérialisation des résultats
  - ✓ L'unité de communication
  - ✓ Le bus TDMA



Plan de cette présentation

- Introduction et contexte

✓ Les traitements en sortie d'imageur

✓ Un état de l'art des architectures

✓ L'analyse des algorithmes de traitement d'image

1. Les choix architecturaux

2. La validation et les résultats d'implémentation

- Conclusions et perspectives

Implémentation et résultats

- **Les algorithmes**
  - ✓ **De nombreuses combinaisons algorithmiques possibles**
    - Nécessiter de spécifier des architectures différentes selon les besoins
- **L'architecture**
  - ✓ **De nombreuses combinaisons possibles**
    - Processeurs 8 bits 12 bits 16 bits 24 bits etc ?
    - Choix des opérateurs ?
    - Organisation des ressources mémoires
    - Intégration d'IPs dédiées

  - ✓ **Création d'un générateur automatique de tuiles de calcul**
    - → Choix des opérateurs
    - → Taille du chemin de données
    - → Nombre de processeurs
    - → Taille des mots de données
    - → Mode de fonctionnement (SIMD Multi-SIMD)
    - → Taille des mémoires de travail
    - → Taille maximale des voisinages supportés
    - Génère un VHDL Synthétisable
    - Génère un assembleur

  - ✓ **Compatibilité des assembleurs et des binaires entre les instances**

Résultats : surface & consommation

- **Validation**
  - ✓ **Modèle SystemC**
  - ✓ **Modèle VHDL**

- **Estimation de la surface silicium**
  - ✓ **Technologie TSMC 65nm Low Power High Voltage Threshold**
    - Librairie Worst Case 0.9V 125°C
  - ✓ **Synthèse d'une tuile de calcul**
    - *Synopsys Design Compiler*
    - *Cadence GateBuilder*
  - ✓ **Simulation & validation Post-synthèse**
    - ModelSim + données réelles
    - Algorithmes testés individuellement

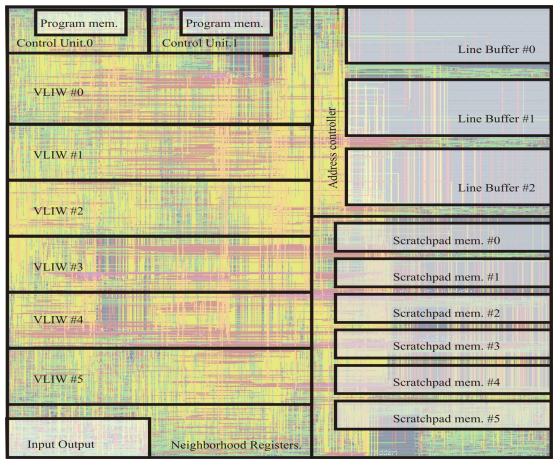
- **Estimation de la consommation**
  - ✓ **Post-Synthèse par analyse dynamique**
    - *Synopsys PrimePower*
      - → Sur jeu de données réelles
      - → Processeur à 133 et 250 MHz chargé à 80% (convolution)
  - ✓ **Post Placement-routage par analyse statique**
    - Cadence *SoCEncouter*, avec un taux d'actvité de 0,25
      - → Taux obtenu par l'analyse dynamique

Résultats : surface & consommation

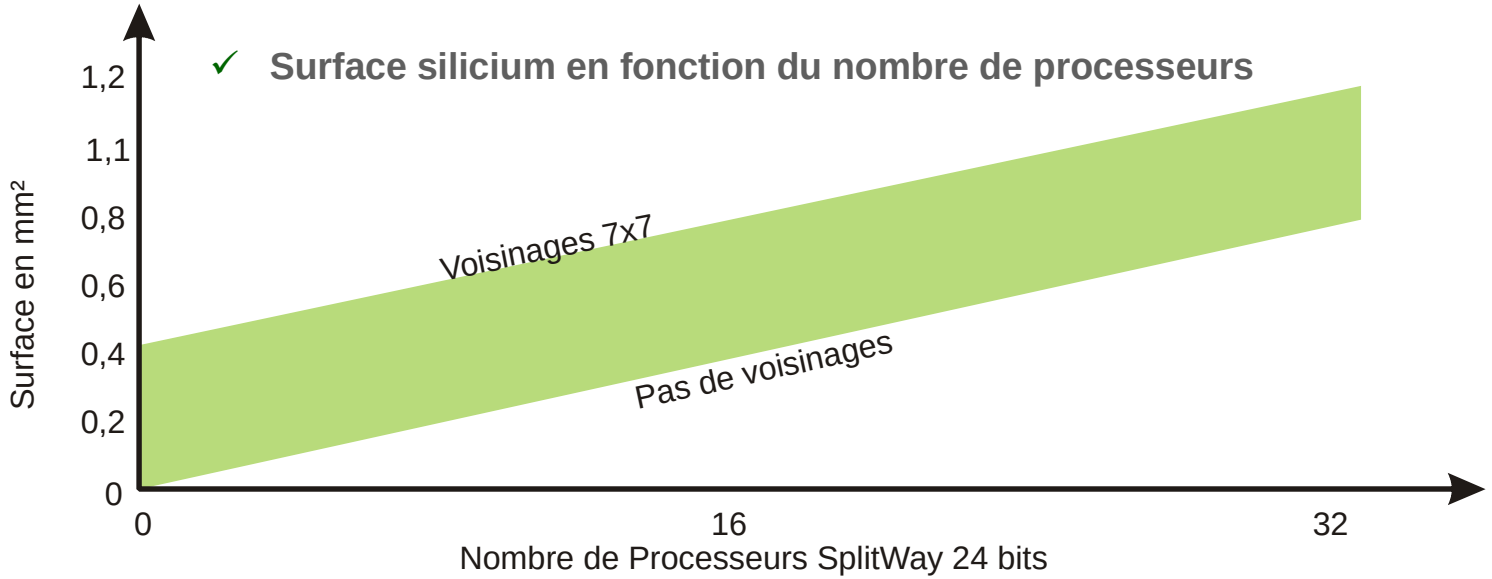
➢ **Caractérisation de l'architecture**

- En surface
- En consommation

Composant	Consommation à 133 MHz dynamique	Consommation à 133 MHz statique	Consommation à 200 MHz dynamique	Consommation à 200 MHz statique
6× processeurs *SplitWay*	6×0.84 mW	6×39 μW	6× 1.81 mW	6× 45 μW
Mémoire de travail 256×24 bits	6×0.24 mW	6×9 μW	6× 0.36 mW	6× 9 μW
3× mémoires lignes 2048× 8 bits	3×0.32 mW	48μW	3×0.32 mW	48 μW
Gestionnaire de voisinages	0.98 mW	77 μW	2.21 mW	80 μW
2× unités de contrôle[1]	2×1.17 mW	2×20 μW	2× 2.16 mW	2× 21 μ
Module de communication	0.38 mW	9 μW	0.46 mW	11 μW
Total	11.04 mW	0.46 mW	20.87 mW	0.53 mW
Total (dynamique + statique)	11.50 mW		21.4 mW	

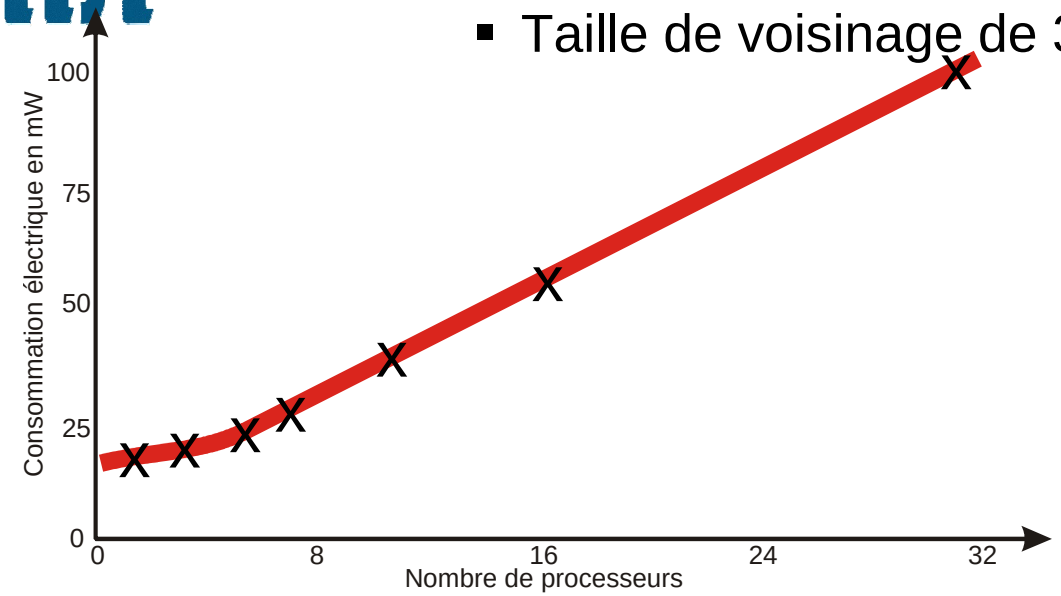


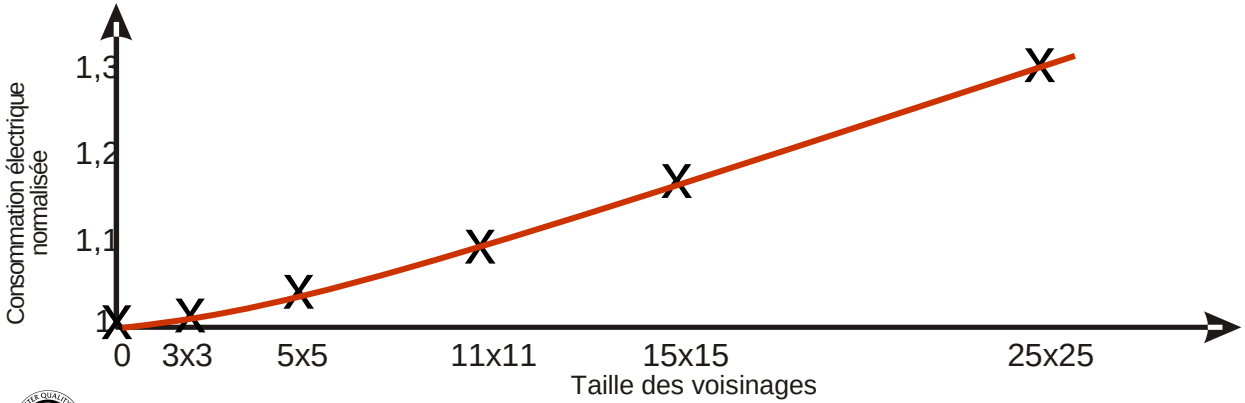
✓ **Surface silicium en fonction du nombre de processeurs**



Résultats : surface & consommation

- **Caractérisation de l'architecture (suite)**
  - ✓ **Consommation électrique en fonction du nombre processeurs**
    - De 2 à 32 PE SplitWay 24 bits
    - Taille de voisinage de 3x3 à 25x25

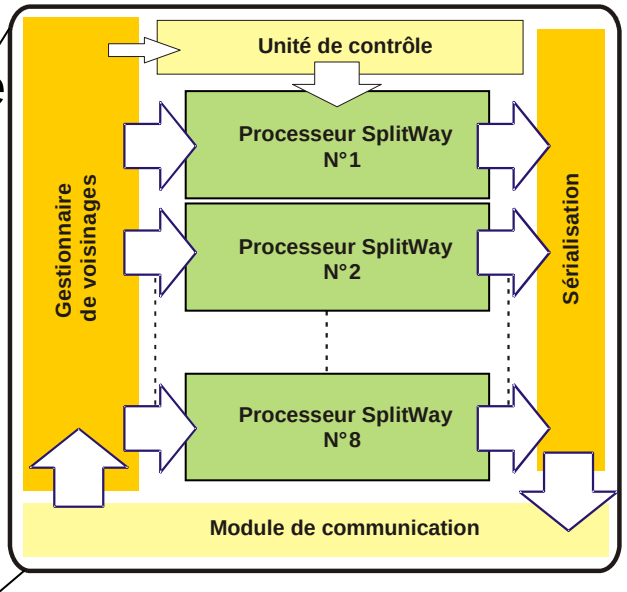


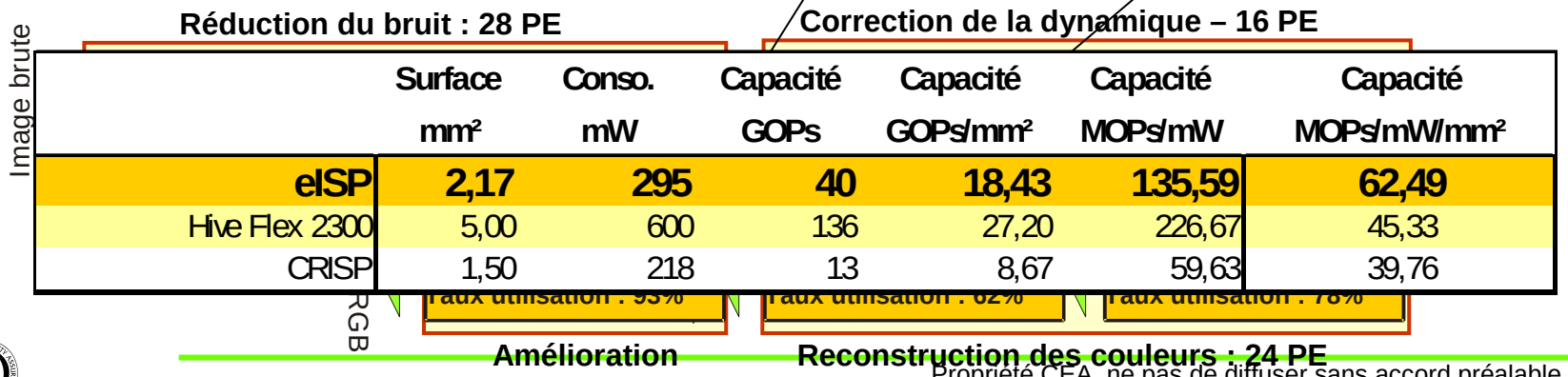


- **Caractérisation du processeur SplitWay**
  - ✓ **Surface silicium**
    - 6 à 10 kGates par processeur
    - 0,02 mm² en 65nm
    - 0,01 mm² 45nm
  - ✓ **Consommation électrique**
    - 1,81 mW @ 250MHz
    - 0,84 mW @ 133MHz

Implémentation et résultats

- **Surface et consommation d'une instance complète**
  - ✓ 7 tuiles de calcul
  - ✓ 80 processeurs SplitWay 24 bits
    - 40 GOPs
    - 130 à 295 mW selon la fréque
  - ✓ Portage d'une chaîne algorithmique
    - 70 GOPs total
    - 23 GOPs pour le calcul
- **Comparaison à l'état de l'art**
  - ✓ CRISP
  - ✓ Hive Flex





	Surface mm²	Conso. mW	Capacité GOPs	Capacité GOPs/mm²	Capacité MOPs/mW	Capacité MOPs/mW/mm²
eISP	**2,17**	**295**	**40**	**18,43**	**135,59**	**62,49**
Hive Flex 2300	5,00	600	136	27,20	226,67	45,33
CRISP	1,50	218	13	8,67	59,63	39,76

Plan de cette présentation

- Introduction et contexte

✓ Les traitements en sortie d'imageur

✓ L'état de l'art

✓ L'analyse des algorithmes

1. Les propositions architecturales

2. La validation et les résultats de l'architecture

- Conclusion et perspectives

Conclusions et perspectives

Dans un contexte embarqué dur, une réelle alternative aux solutions dédiées

✓ L'architecture eISP (et processeur SplitWay)

- Modèle originaux
- Méthodologie de conception réutilisable

- Répond aux contraintes initialement fixées
  - → Programmable
  - → Faible surface silicium (quelques mm² en 65nm)
  - → Faible consommation < ½ mW

- Scalables par
  - → Augmentation du nombre de tuiles
  - → Augmentation du nombre de processeurs
  - → Augmentation de la fréquence de fonctionnement
  - → Emplacements prévus pour extensions (IPs dédiées etc)

- Sont spécialisables
  - → Par intégration d'IP dédiées
  - → Emplacements prévus

Conclusions et perspectives

Perspectives à court terme

✓ Architectures « PEPS » au CEA LIST

- Reprendre et industrialiser les principes d'**eISP** :
  - → plusieurs petit VLIW (2 ou 3 voies)
  - → Mémoire distribué
  - → Modèle d'exécution
    - » mode flux
    - » Mémoire distribuée
  - → Projets basés sur ces architectures (dont un projet bilatéral)
  - → Chaine de compilation en développement

✓ Portage FPGA

- 24 à 50 processeurs SplitWay sur Virtex 5 FX70T (5 à 13 GOPs)
- 4 à 10 tuiles de calcul à 133MHz
- Pas de mémoire externe au composant
  - → Vers une démonstrateur (PEPS plateforme Eve, eISP ML507)

✓ Tester sur d'autres domaines applicatifs

- Utilisation de l'architecture pour réaliser des traitements dans le domaine fréquentiel
  - → Adaptable pour le support de nombres complexes
  - → Traitement en flux derrière la TF et avant TF^{-1}
- Autres applications de traitement du signal (audio, télécoms)

Conclusions et perspectives

Perspectives à long terme

✓ Processeur SplitWay : un µDSP à part entière

- → Faible consommation
- → Faible surface
- → Performances ½GOPs à 250MHz

✓ Evolutions architecturales

- Utilisation eISP pour des applications moins contraintes
- Traitement vidéo haut niveau avec mémoire d'image
  - → Applications moins contraintes en surface et consommation
- Autres domaines applicatifs moins contraints
  - → Caméras
  - → Appareil photos

✓ Un modèle générique d'architecture réutilisable

- → Pour des applications flux de données
  - » Adaptations pour des « grands » voisinages (200x200 pixels)

✓ Outils d'Adéquation Algorithme Architecture

- → Exemple: partitionnement automatique des tuiles de calcul d'eISP

Contributions Scientifiques

- ***Brevets & Journaux:***
  - ✓ **Dispositif de traitement en parallèle d'un flux de données**
    Brevet BD 10767 – 65563 Déposé en octobre 2008
    Mathieu Thevenin et Laurent Letellier
  - ✓ **Processeur vidéo HD pour la téléphonie mobile Architecture de calcul très basse consommation pour le traitement vidéo haute définition.**
    Technique et Science Informatiques , N°29 ,numéro s pécial (Fevrier 2010) Editions LAVOISIER
    Mathieu Thevenin, Michel Paindavoine, Laurent Letellier et Barthelemy Heyrman

- ***Colloques & Conférences :***
  - ✓ ***eISP: a Programmable Processing Architecture for Smart Phone Image Enhancement***
    DASIP 2009, 22-24 Septembre 2009, Nice Sophia Antipolis
    Mathieu Thevenin, Michel Paindavoine, Laurent Letellier, Renaud Schmit, Barthelemy Heyrman
  - ✓ ***eISP : une architecture de calcul programmable pour l'amélioration d'images sur téléphone portable.***
    GRETSI 2009, 8-11 Septembre 2009, Dijon
    Mathieu Thevenin, Laurent Letellier, Barthelemy Heyrman et Michel Paindavoine
  - ✓ ***eISP : processeur vidéo pour la téléphonie mobile***
    GDR SoC-SIP 2009 10-12 Juin 2009, Paris-Orsay
    Mathieu Thevenin, Laurent Letellier et Michel Paindavoine
  - ✓ ***Embedded Processor Extensions for Image***
    Photonics Europe 2008, 7-10 Avril 2008 Strasbourg
    Mathieu Thevenin, Laurent Letellier, Barthelemy Heyrman et Michel Paindavoine
  - ✓ ***Extensions matérielles pour processeurs embarqués de traitement d'images***
    Symposium sur les Architectures Nouvelles de Machines (SympA - 2008), 11-13 Fevrier 2008, Fribourg
    Mathieu Thevenin, Laurent Letellier, Barthelemy Heyrman et Michel Paindavoine

cea

list

MERCI

Références

[1] Power-Efficient Reconfiguration Control in Coarse-Grained Dynamically Reconfigurable Architectures
Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation: 18th International Workshop, PATMOS 2008, Lisbon, Portugal
Dmitrij Kissler, Andreas Strawetz, Frank Hannig Jürgen Teich

[2] HiveFlex VSP2500 Series Video Signal Processor
Databrief
http://www.siliconhive.com

[3] XeTaL-II: a low-power multiprocessing SIMD architecture
MEDEA+ Design Automation Conference 2006
Marc Heijligers

[4] Xetal family description
NXP Eindhoven, NL - Copyright Chess 2006-2009
R. Kleihorst

[5] Reconfigurable Instruction Cell Array
IEEE Transactions on Very Large Scale Integration (VLSI) Systems
Volume 16 , Issue 1 (January 2008)
Khawam et al.

[6] CRISP: Coarse-Grained Reconfigurable Image Stream Processor for Digital Still Cameras and Camcorders
Ieee Transactions On Circuits And Systems for Video Technology, Vol. 18, No. 9, September 2008 1223
Jason C. Chen and Shao-Yi Chien,