Architecture SoC-FPGA adaptable dédiée à l'analyse d'images

Alain AUBERT, Nathalie BOCHARD, Virginie FRESSE

Projet EmSoC

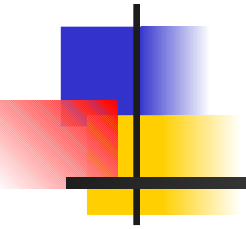
Villard de Lans, 8-9 juin 2006

Université Jean Monnet SAINT-ETIENNE

tsi

Plan de la présentation

- Description de l'architecture adaptable
- Application d'analyse d'images : PIV
- Prédictabilité de l'architecture après modifications
- Perspectives de travail



Description de l'architecture adaptable

- Caractéristiques propres à l'analyse d'images
- Analyse des différents modules de l'architecture
- Protocole de communication

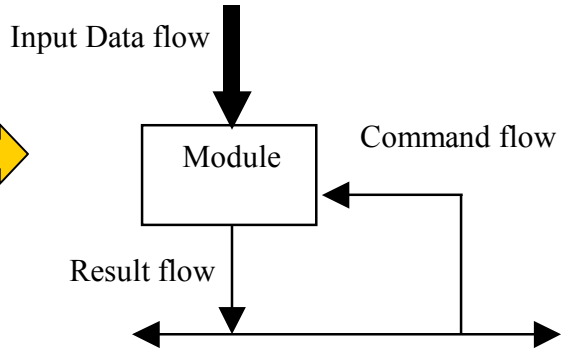
Caractéristiques algorithmiques

- Algorithmes d'analyse d'images
  - Plusieurs opérations à réaliser:
    - Opération d'acquisition
    - Opération de stockage (images)
    - Opération de traitement (corrélation)
    - Opération de contrôle

→ Organisation en modules réalisant ces opérations

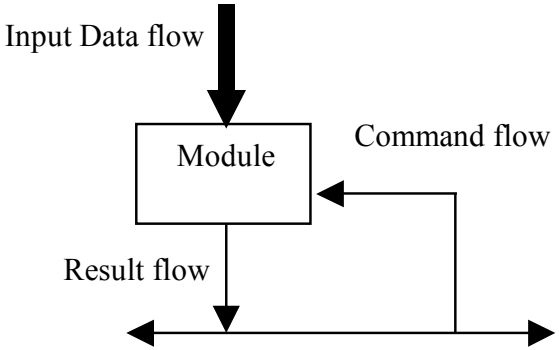
  - Flot d'entrée important : images à traiter
  - Flot de sortie faible : résultats (vecteurs)
  - Parallélisme implicite :
    - Plusieurs zones de l'image peuvent être traitées simultanément

→



Modèle Harvard modifié

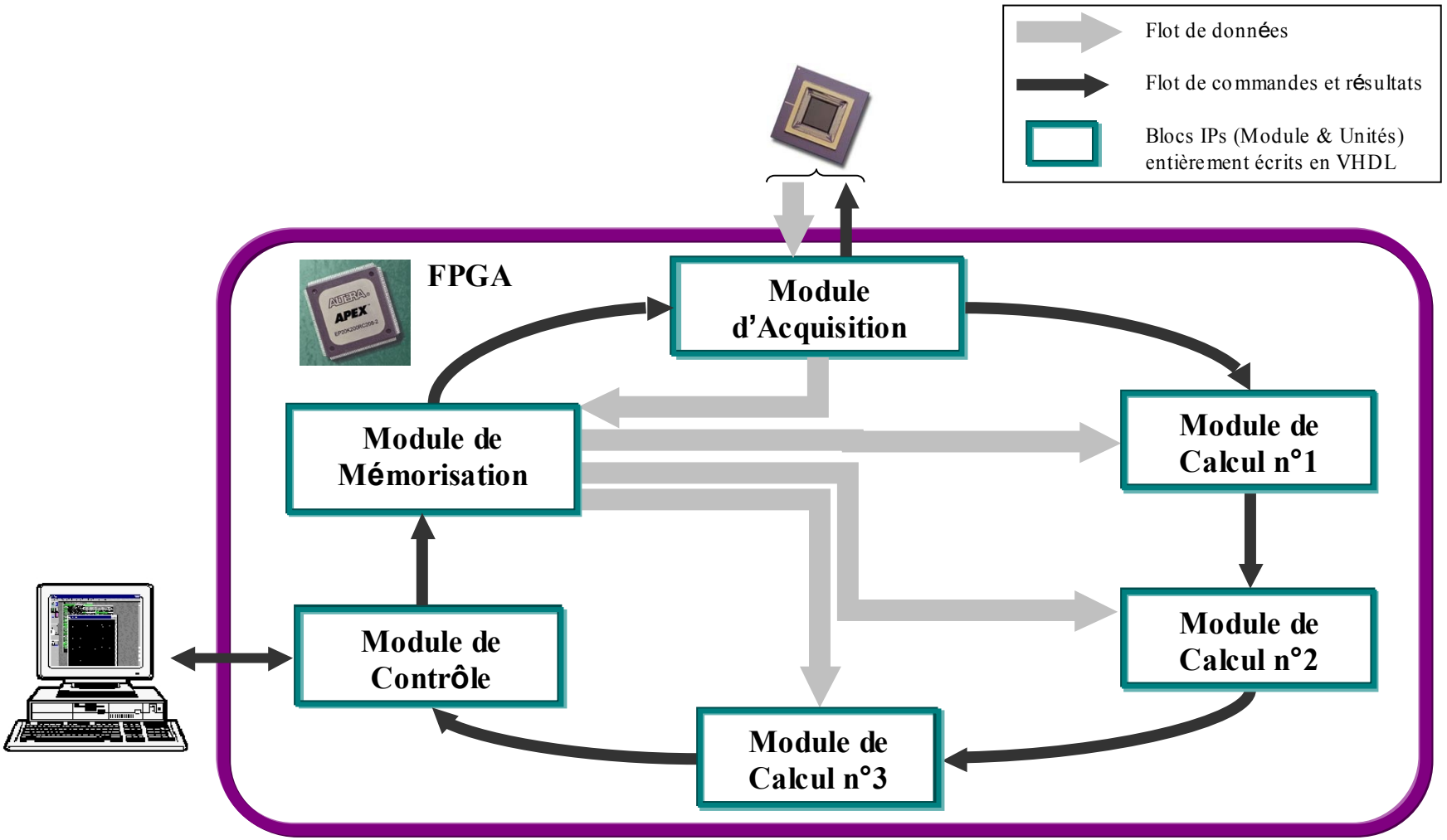
Modèle Harvard modifié proposé



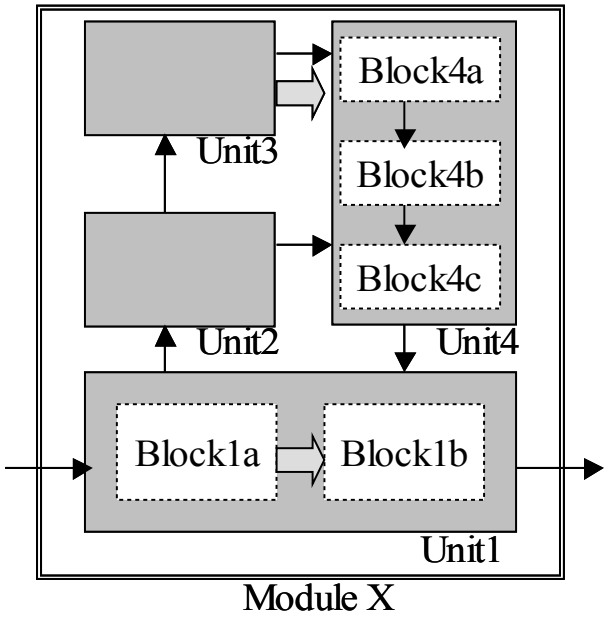
Modèle Harvard modifié

- Approche NoC:
  - Flot de données d'entrée **via un bus** à bande passante élevée
  - Flot de sortie associé aux commandes **via un anneau**
- Approche GALS (Globally Asynchronous Locally Synchronous):
  - Chaque module possède sa propre fréquence

Architecture proposée



Analyse des modules: structure générale



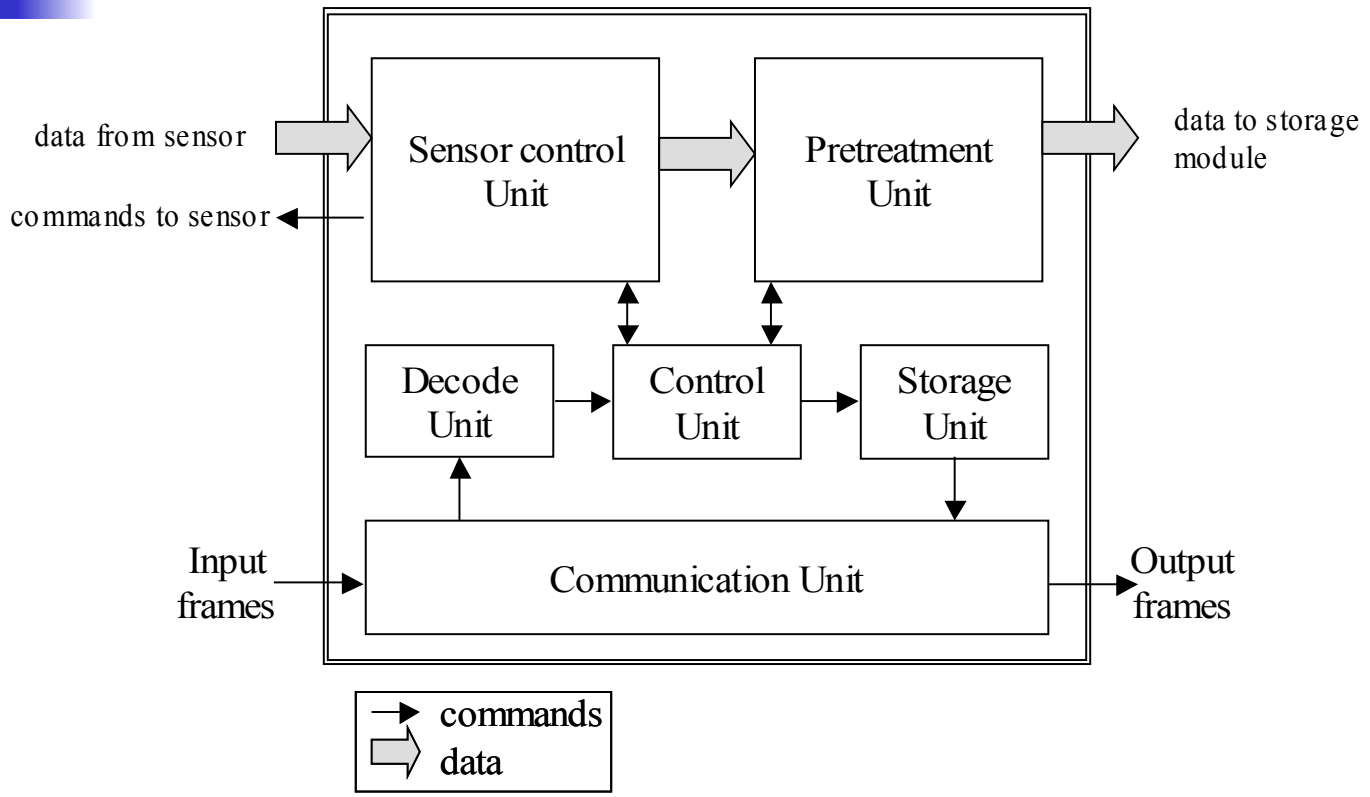
- Principe de modularité
  - 1 module = 1 opération
    - Opération d'acquisition
    - Opération de stockage (images)
    - Opération de traitement (corrélation)
    - Opération de contrôle
  - 1 unité = 1 fonction principale
    - Décodage
    - Contrôle
    - Corrélation
    - Interfaçage
  - 1 bloc = 1 fonction élémentaire
    - Mémoire
    - Comparateur
    - Machine d'état

Analyse des modules: adaptabilité / réutilisation

- Architecture modulable construite autour du principe de réutilisation directe du maximum de blocs lors des modifications nécessaires aux différentes adaptations possibles
  - 2 catégories de modules:
    - Module statique → réutilisable sans modification
    - Module dynamique → dépendant du type de changement
      - Répartition/ordonnancement vers les différents modules
      - Changement de paramètres de l'algorithme
      - Changement d'algorithme
      - Changement d'éléments physiques (source de données, capteur...)

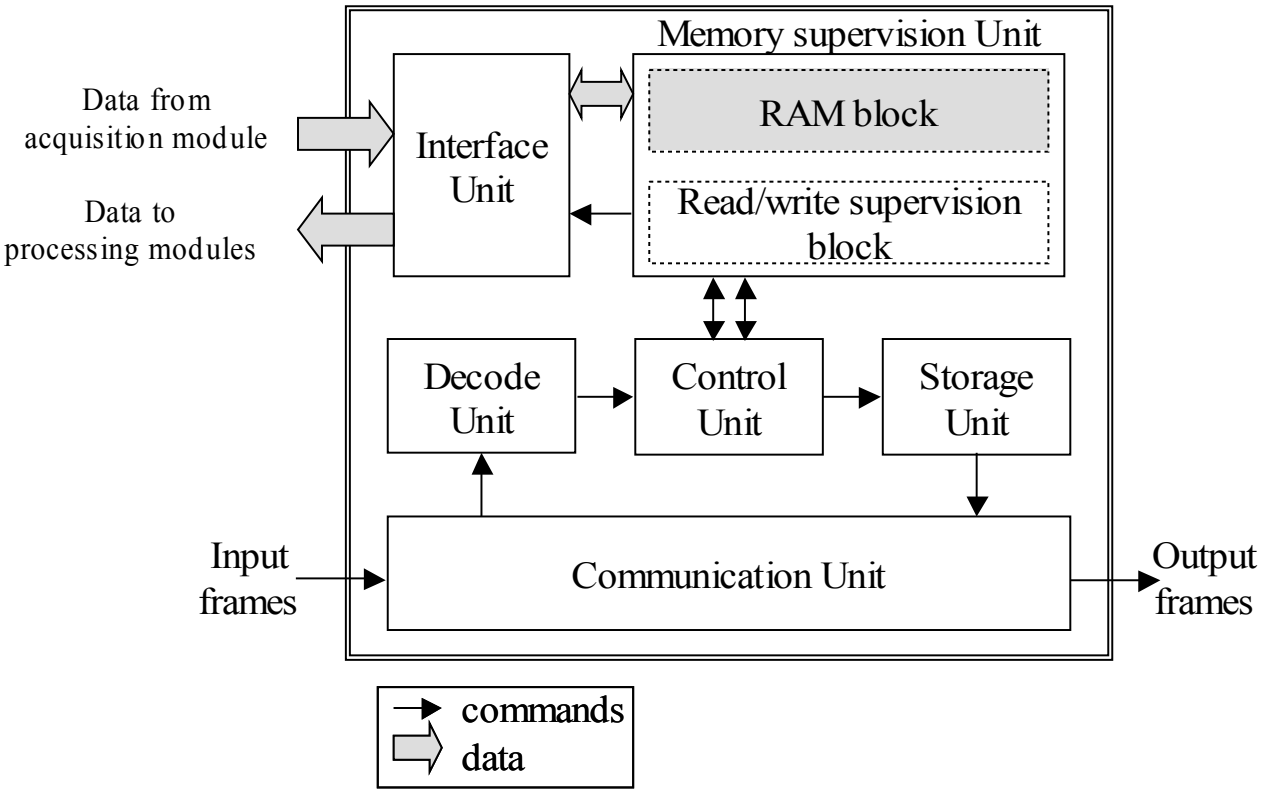
Le degré de réutilisation de l'architecture s'évalue par le pourcentage entre les modules dynamiques et statiques

Module d'acquisition



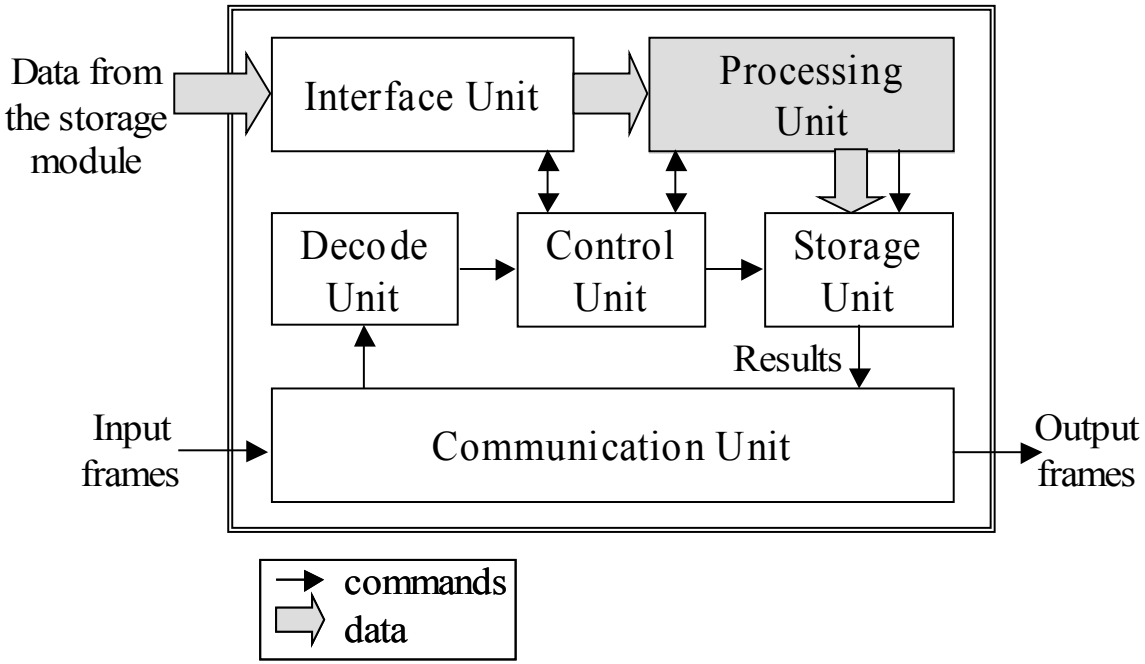
- Permet l'acquisition d'images/données via un imageur CMOS.

Module de mémorisation



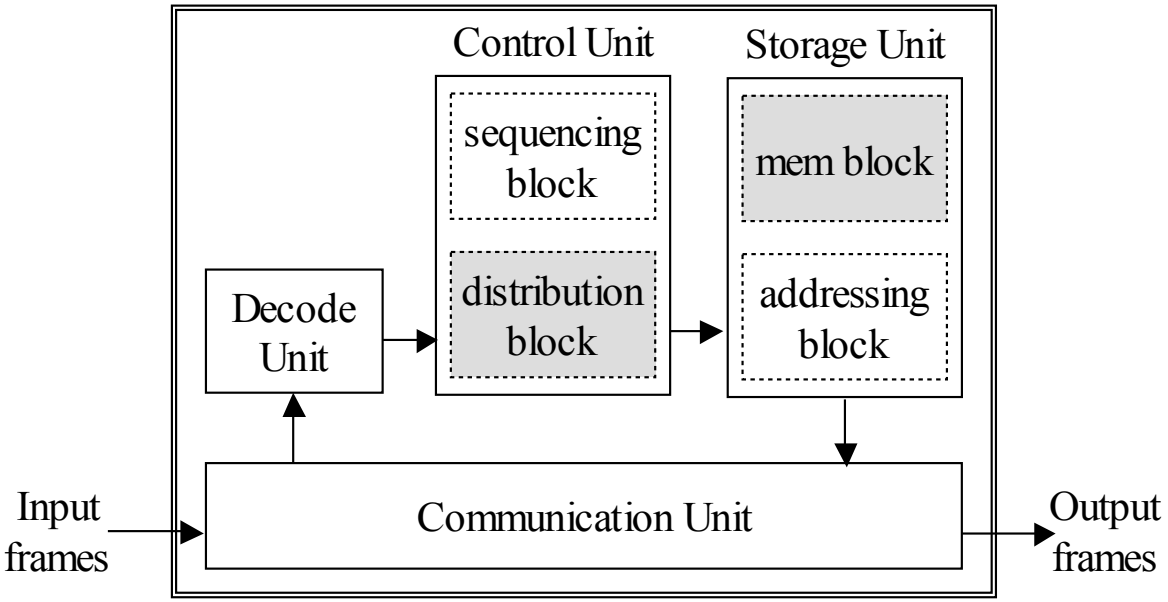
- Mémorise les données provenant de l'imageur dans des mémoires embarquées du FPGA.

Module de traitement



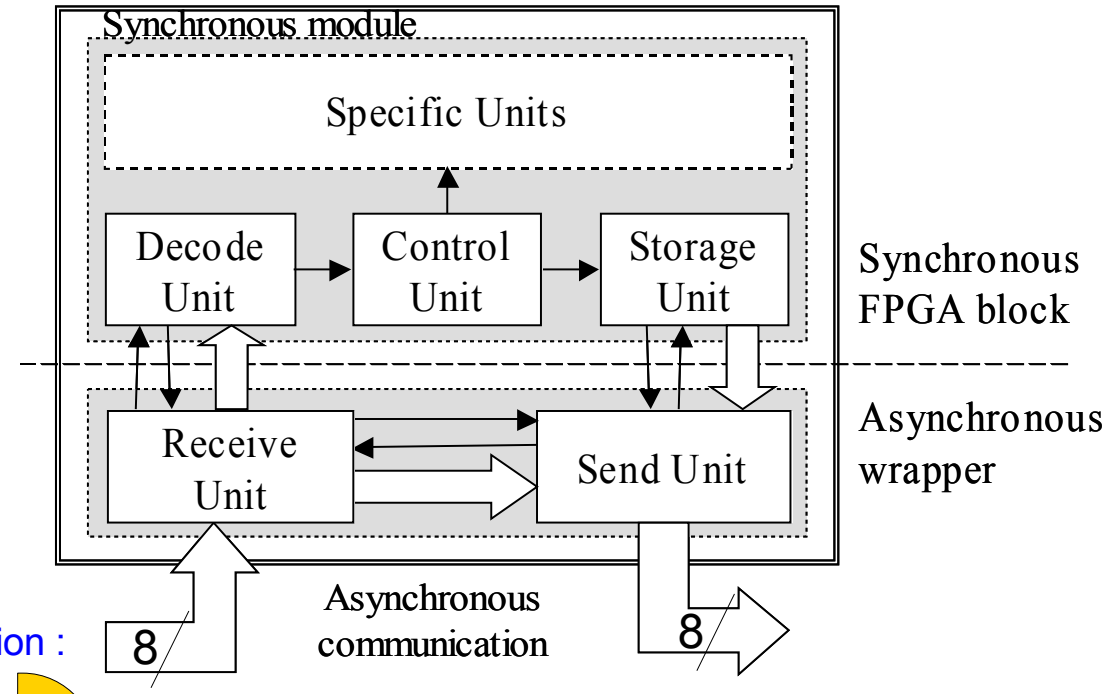
- Contient la logique nécessaire à la corrélation binaire. Le nombre de modules de traitement dépend du paramétrage de l'algorithme.

Module de contrôle

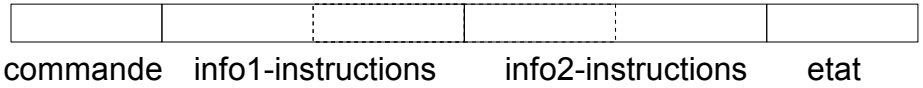


- Contrôle entièrement centralisé sur le module de contrôle. Il envoie les commandes aux divers modules via l'anneau de communication.

Protocole de communication



Anneau de communication :
1 trame = 6 octets



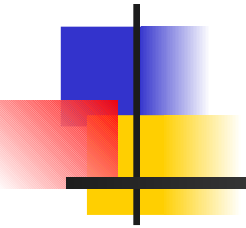
Ex : ordre de lecture d'une imagette pour le module de stockage

MS + lecture

Adresse départ

Taille

Occupé
Traitée
Problème

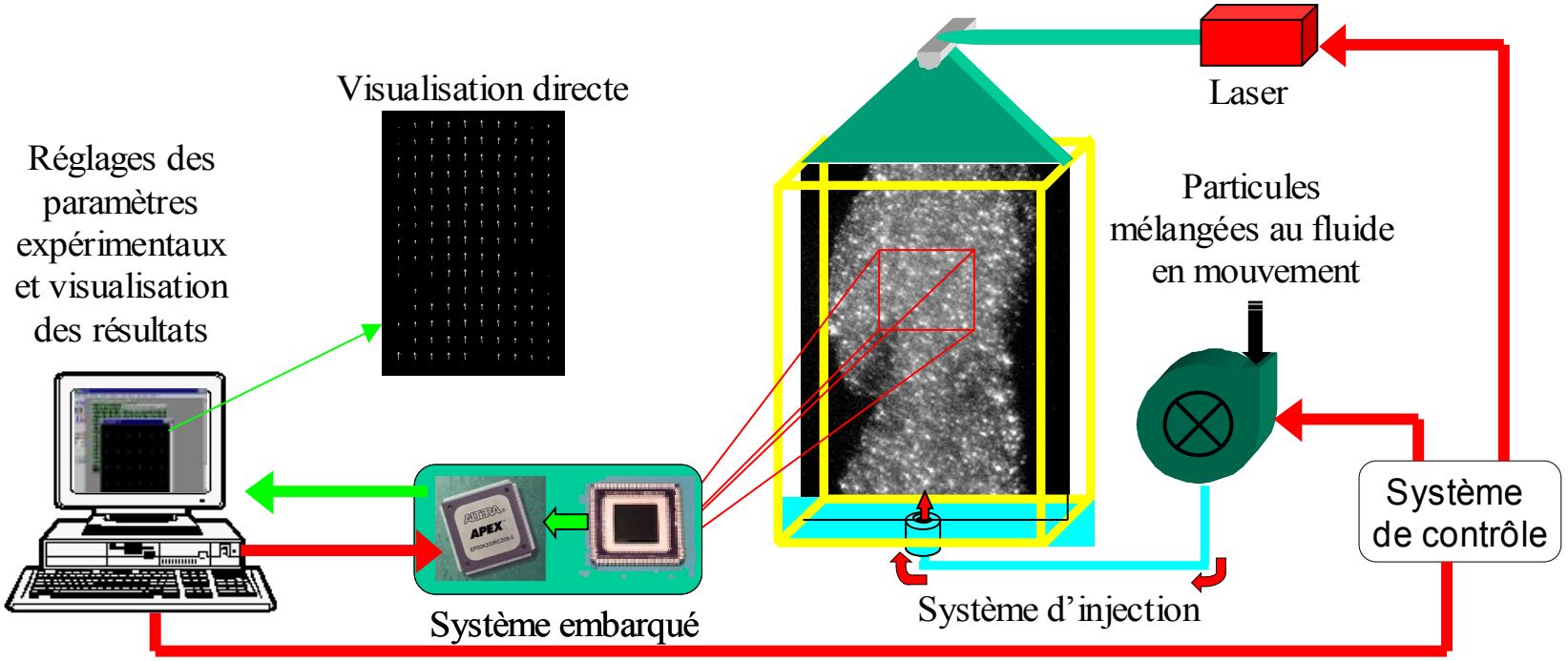


Application d'analyse d'images : PIV

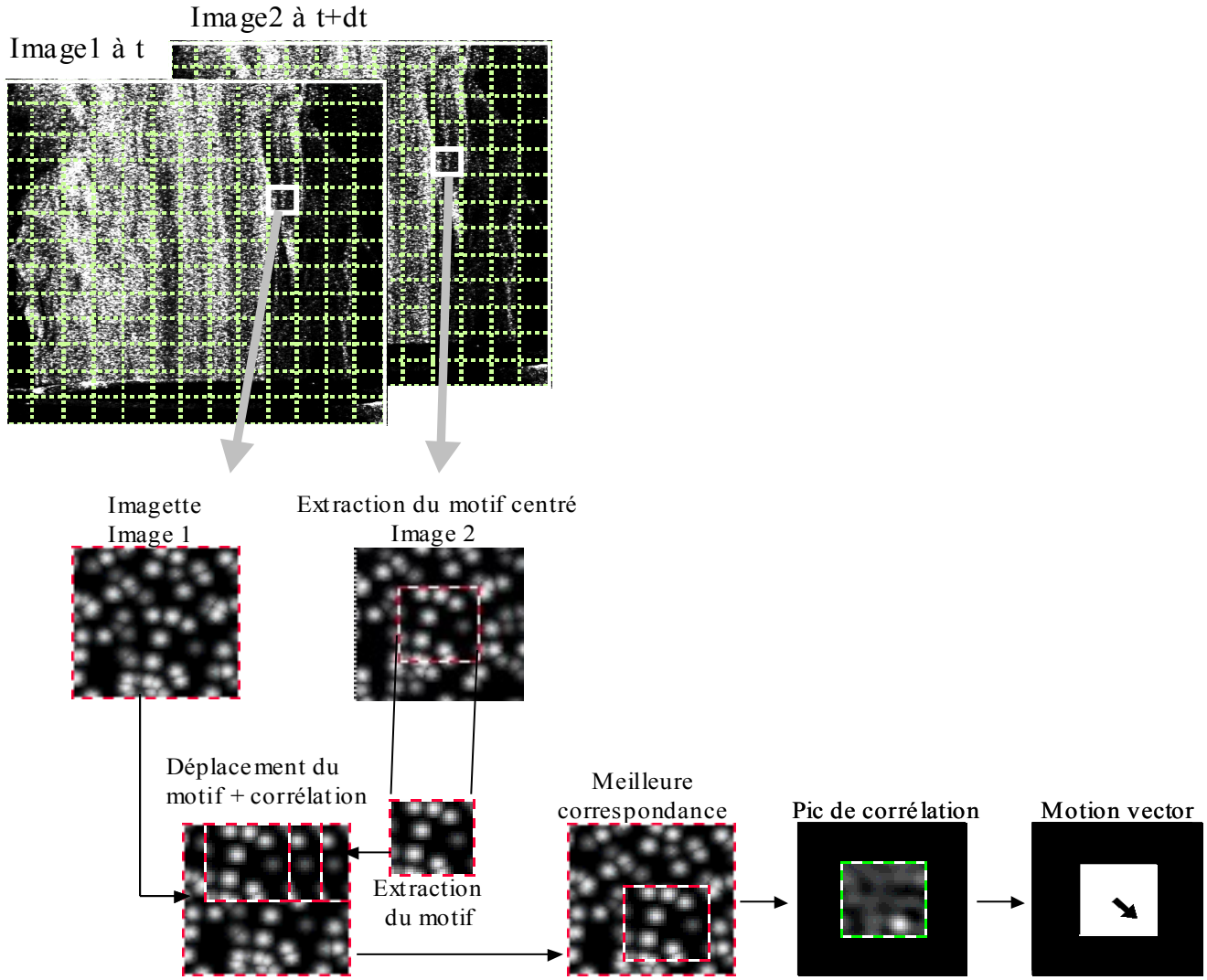
- Présentation et principe
- Algorithme implanté : corrélation binaire

PIV: présentation

- Application PIV : Vélocimétrie par image de particules
  - Calcul de champs de vecteurs vitesse dans l'écoulement d'un fluide

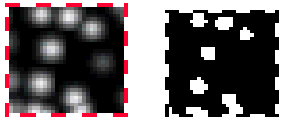


PIV: principe

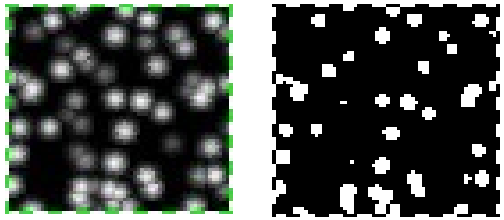


PIV: algorithme implanté

Motif s_1 (M×M)



Imagette s_2 (N×N)



- Corrélation Direct :

$$f(i,j) = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} \left[s_1(x,y) \times s_2(x-i, y-j) \right]$$

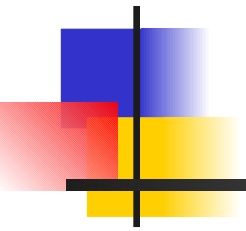
$i,j \in [0,N-M]$

adaptation à une structure matérielle

- Corrélation Binaire :

$$f(i,j) = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} \left[s_1(x,y) \; XNOR \; s_2(x-i, y-j) \right]$$

  - Précision suffisante des calculs de déplacement pour la plupart des applications (précision sub-pixel) si seuillage adaptatif pour chaque imagette.



Prédictabilité de l'architecture après modifications

- Impact des différents types de modifications sur les modules de l'architecture
- Exemples de modifications pour la PIV
  - Modification de la répartition des tâches
  - Modification des paramètres de l'algorithme
- Modèles de prédiction pour ces exemples
- Résultats expérimentaux vs modèles

Impact des différents types de modifications sur les modules

Modifications \ Modules	Contrôle	Traitement	Acquisition	Mémorisation
Répartition des tâches	PIV: parallèlisation des opérations (nombre de modules de calcul)			
Paramètres de l'algorithme		PIV: taille de l'imagette		
Sources de données			Passage d'un capteur CCD à un capteur CMOS	
Algorithme		PIV:corrélation niveau de gris ou par FFT ou autre algorithme d'analyse d'image		

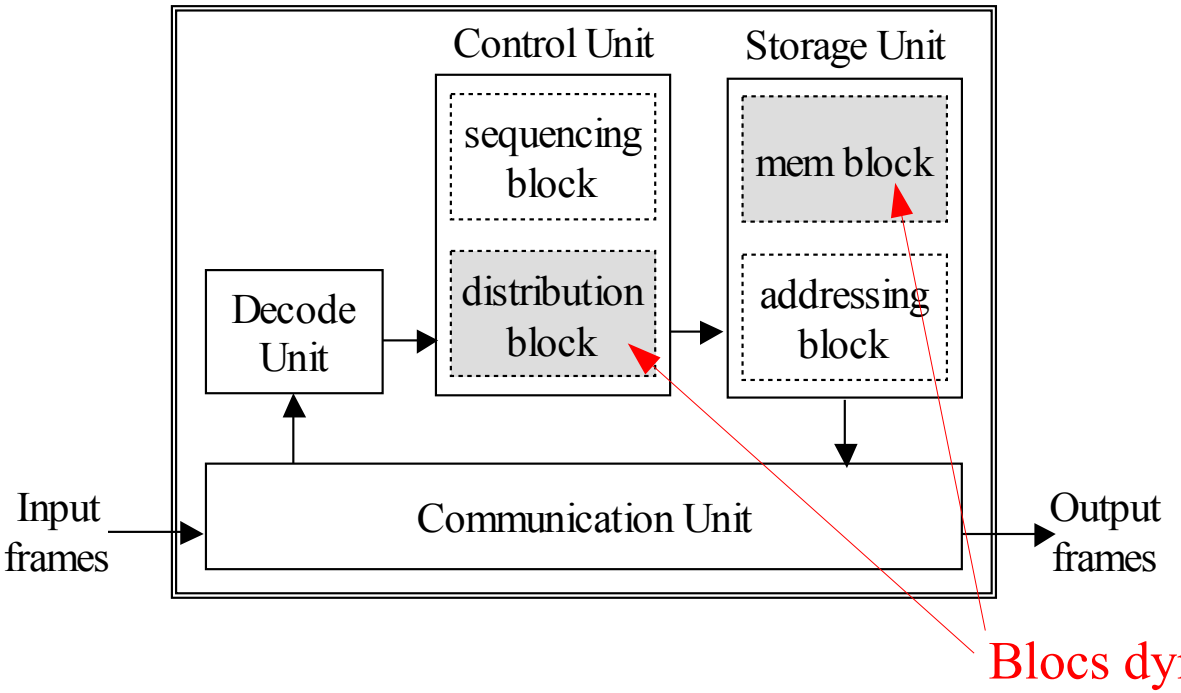
Modules dynamiques Modules statiques

- Peut-on prévoir à l'avance (prédiction par modèles) les ressources et les timing des modules dynamiques ?
  - Cela dépend du type de modifications

Exemples de modification pour la PIV

- PIV : Modification de la répartition des tâches
  - Parallèlisation du traitement : augmentation du nombre de modules de calcul

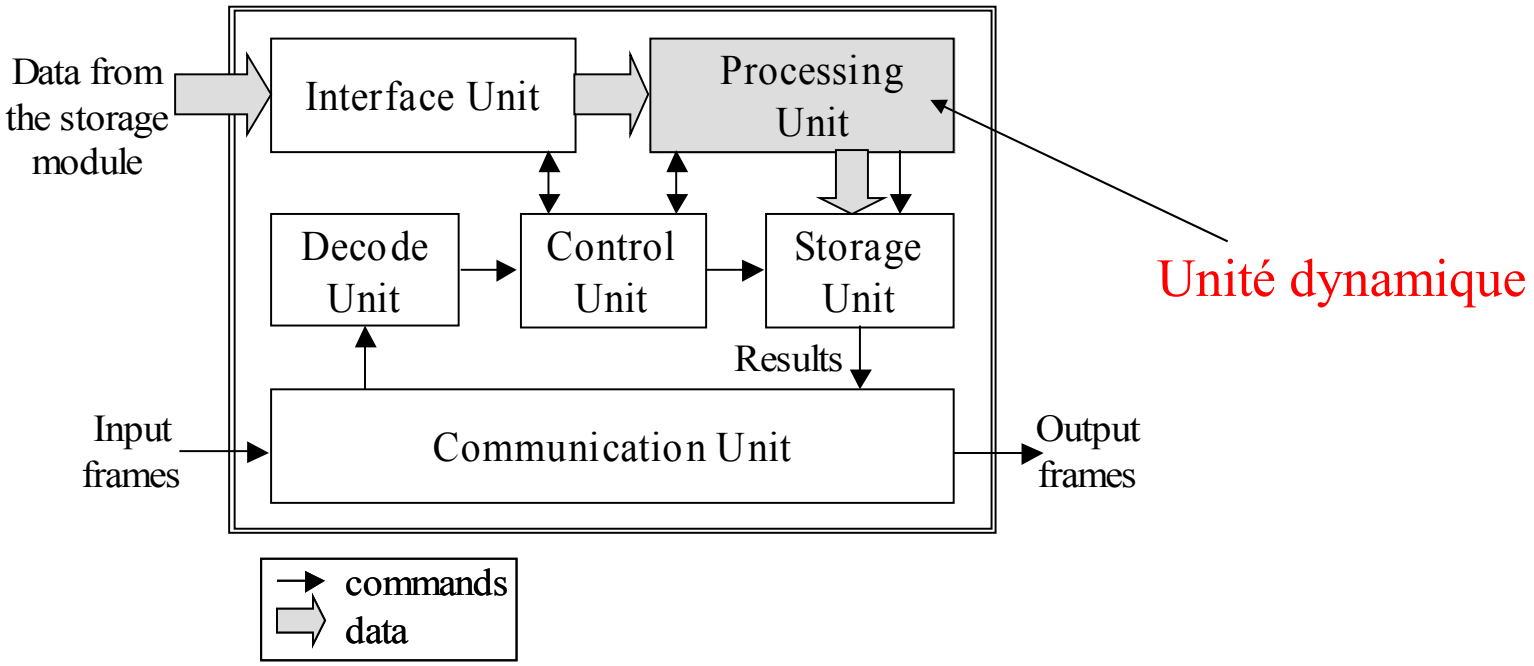
→ Module dynamique : module de contrôle



Exemples de modification pour la PIV

- PIV : Modification des paramètres de l'algorithme
  - Passage d'une imagette 32x32 à une imagette 64x64

→ Module dynamique : module de calcul



Exemples de modification pour la PIV

- Peut-on prévoir à l'avance pour ces 2 exemples les ressources et les timing des modules dynamiques ?
  - Oui à partir d'une première implantation
  - Caractéristiques de la première implantation :
    - Taille de l'image : 320x256 pixels
    - Taille de l'imagette : 32x32 pixels
    - Taille du motif : 16x16 pixels
    - Nombre de modules de calcul : 1
    - Fréquences :
      - $F_{acq} = 50$ Mhz
      - $F_{mem} = F_{cal} = 100$ Mhz
      - $F_{ctrl} = 150$ Mhz

Modèles de prédiction

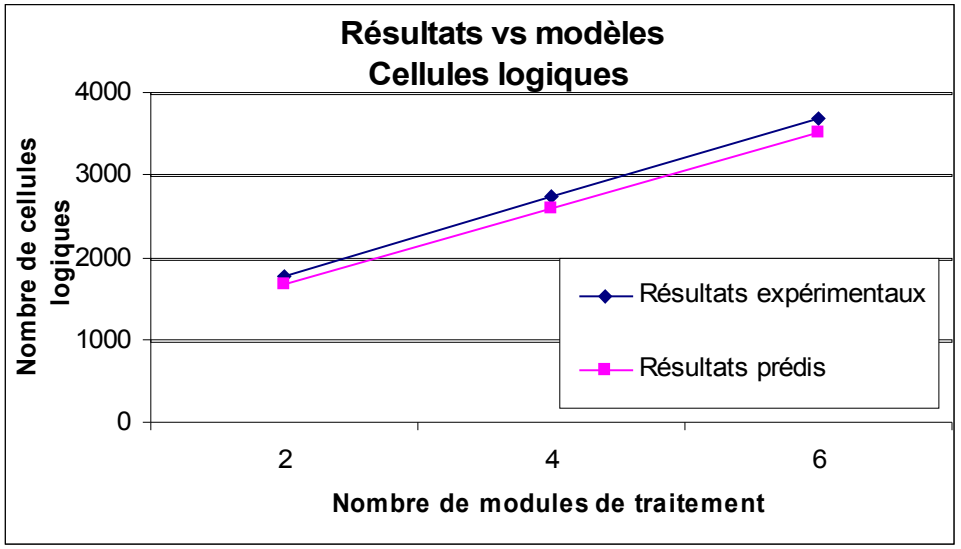
- Modification de la répartition des tâches
  - La première implantation nous permet d'extraire les modèles suivants :
    - Au niveau des ressources :
      - $R_{global} = R(AM) + R(SM) + R(CM) + N \text{ x } R(PM)$
        - $Lc = 767 + 457 \times N$
        - $Rg = 867 + 492 \times N$
        - $Mb = 524\,320 + 1280 \times N$
    - Au niveau du temps d'éxécution (temps moyen de calcul d'un vecteur) :
      - $$T_v = \frac{N \times T_{com} + T_{proc}}{N}$$

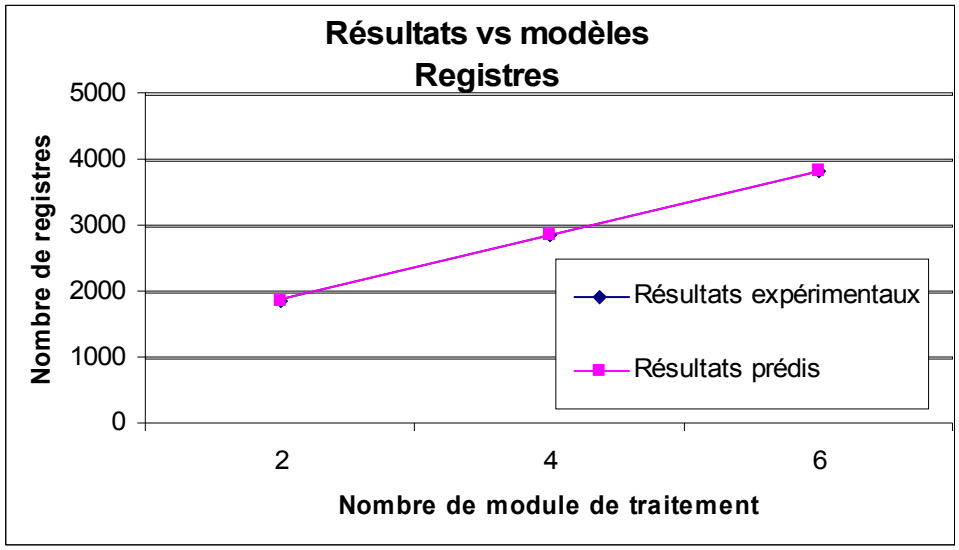
Modèles de prédiction

- Modification des paramètres de l'algorithme
  - La première implantation nous permet d'extraire les modèles suivants :
    - Au niveau des ressources :
      - $Lc = 1\,038 + 186 \times S/32$
      - $Rg = 1\,142 + 217 \times S/32$
      - $Mb = 524\,320 + S^2 \times (S/32)^2$
    - Au niveau du temps d'éxécution (temps moyen de calcul d'un vecteur) :
      - $T_v(\mu s) = 5{,}6 + 4{,}9 \times (5/4) \times (S/32)^2 + T_{clk} \times S/2 \times (S/2+1)^2 \quad si\ S > 32$
      - $T_v(\mu s) = 5{,}6 + 4{,}9 \times (3 \times S/64) + T_{clk} \times S/2 \times (S/2+1)^2 \quad si\ S \leq 32$

Résultats expérimentaux vs modèles

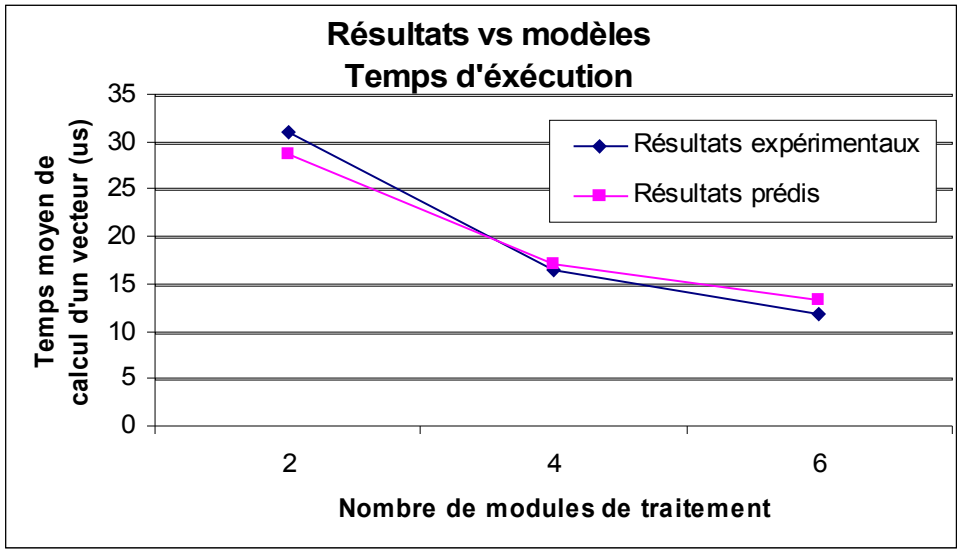
- Modification de la répartition des tâches
  - En terme de ressources





Résultats expérimentaux vs modèles

- Modification de la répartition des tâches
  - En terme de temps d'exécution



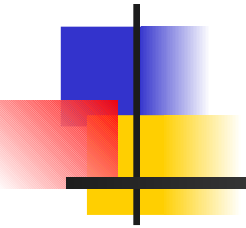
Résultats expérimentaux vs modèles

- Modification des paramètres de l'algorithme
  - En terme de ressources

	16×16 window		32×32 window	64×64 window	
	Result	Prediction	Result	Result	Prediction
Logic cells	1 068	1 131	1 224	1 375	1 410
Registers	1 211	1 251	1 359	1 600	1 576
Mem. bits	524 608	524 640	525 600	529 344	529 440

  - En terme de temps d'éxécution

	16×16 window		32×32 window	64×64 window	
	Result	Prediction	Result	Result	Prediction
Tv (µs)	16,8	15,8	59,6	381,0	378,6

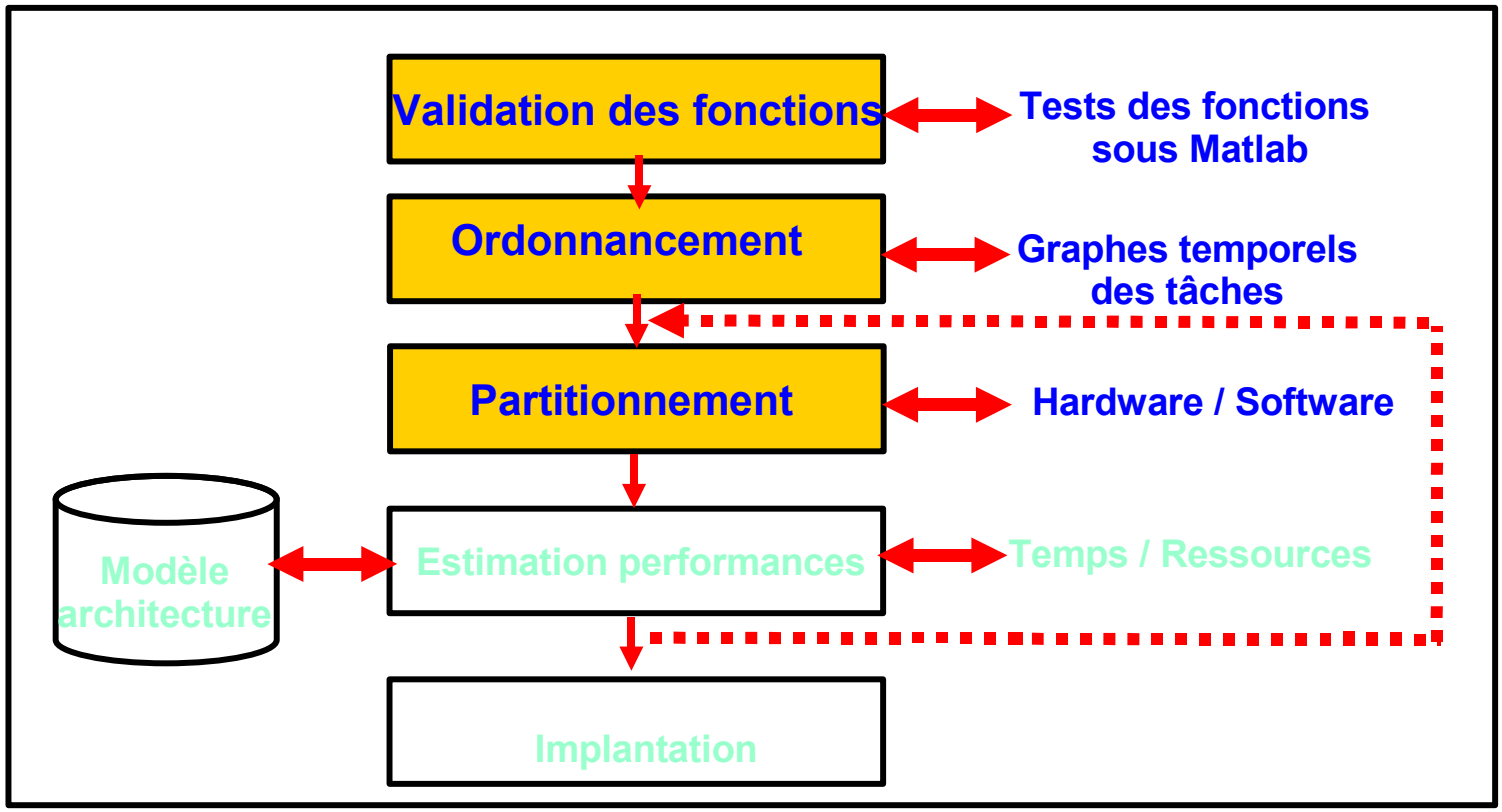


Perspectives de travail

Perspectives de travail

- Objectif : mettre au point un flot de conception automatique ou semi automatique de portage d'algorithmes vers une cible matérielle
  - Contraintes :
    - Classe d'applications : analyse d'images (flot d'entrée dominant)
    - Architecture cible prédéfinie :
      - architecture modulaire développée au sein de l'équipe
    - Cible matérielle : mono FPGA
    - Portage optimisée en :
      - temps de traitement
      - et/ou ressources utilisées
      - et/ou temps de développement (Time to Market)
    - Utilisation de processeurs SoftCore
  - Tâches à effectuer:
    - Caractériser et modéliser l'architecture cible prédéfinie
    - Sélectionner et évaluer les outils existants correspondant à nos contraintes, puis proposer des améliorations et créer d'éventuelles passerelles entre outils

Perspectives de travail



- 1ère application de cette nouvelle méthodologie : reconnaissance de signatures multi-spectrales d'oeuvres d'art