Le CacheND-AP : pré-chargement adaptatif dans les tableaux

S. Mancini

Plan

- Le memory wall
- Le prefetch
- Le Cache nD-AP
- Conclusion & perspective

Motivations

Constats:

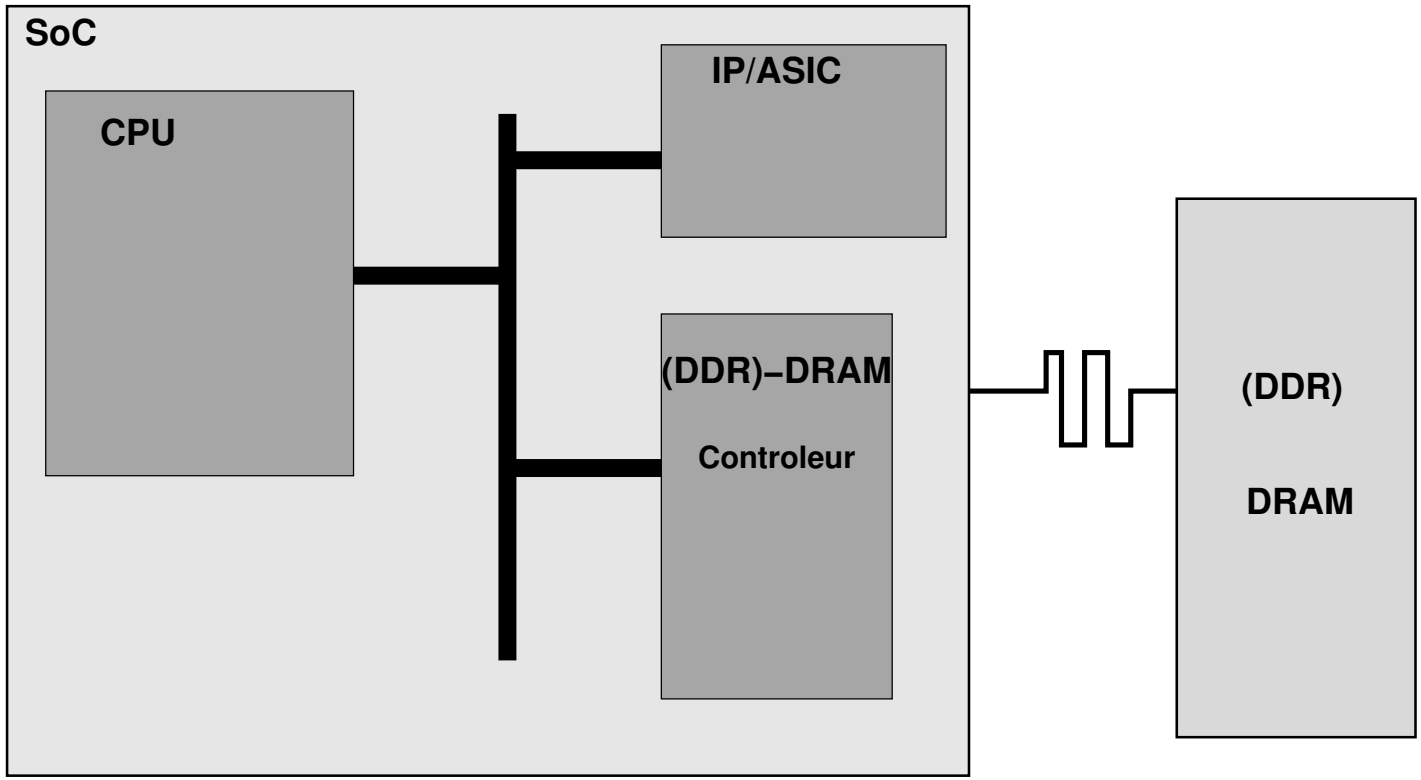
- Concevoir des unités de traitement est devenu relativement aisé (HLS).
- Acheminer les données devient de plus en plus pénalisant (surface, performance, consommation)

Pour concevoir une unité de traitement, il faut gérer:

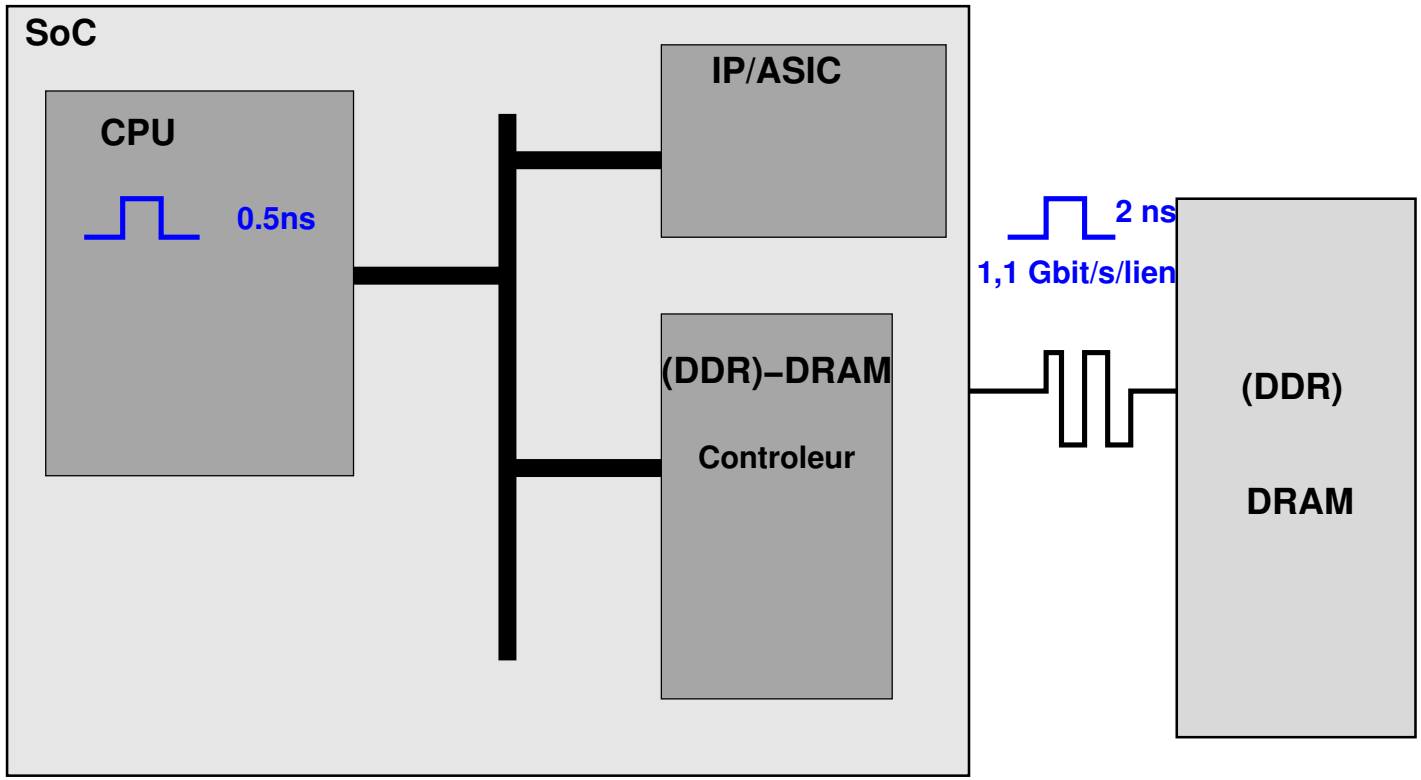
- Les bus
- La synchronisation des transferts & calculs
- Les interfaces mémoire

☞ L'ITRS prévoit que 80% de la surface et de la consommation d'énergie seront dus aux mémoires.

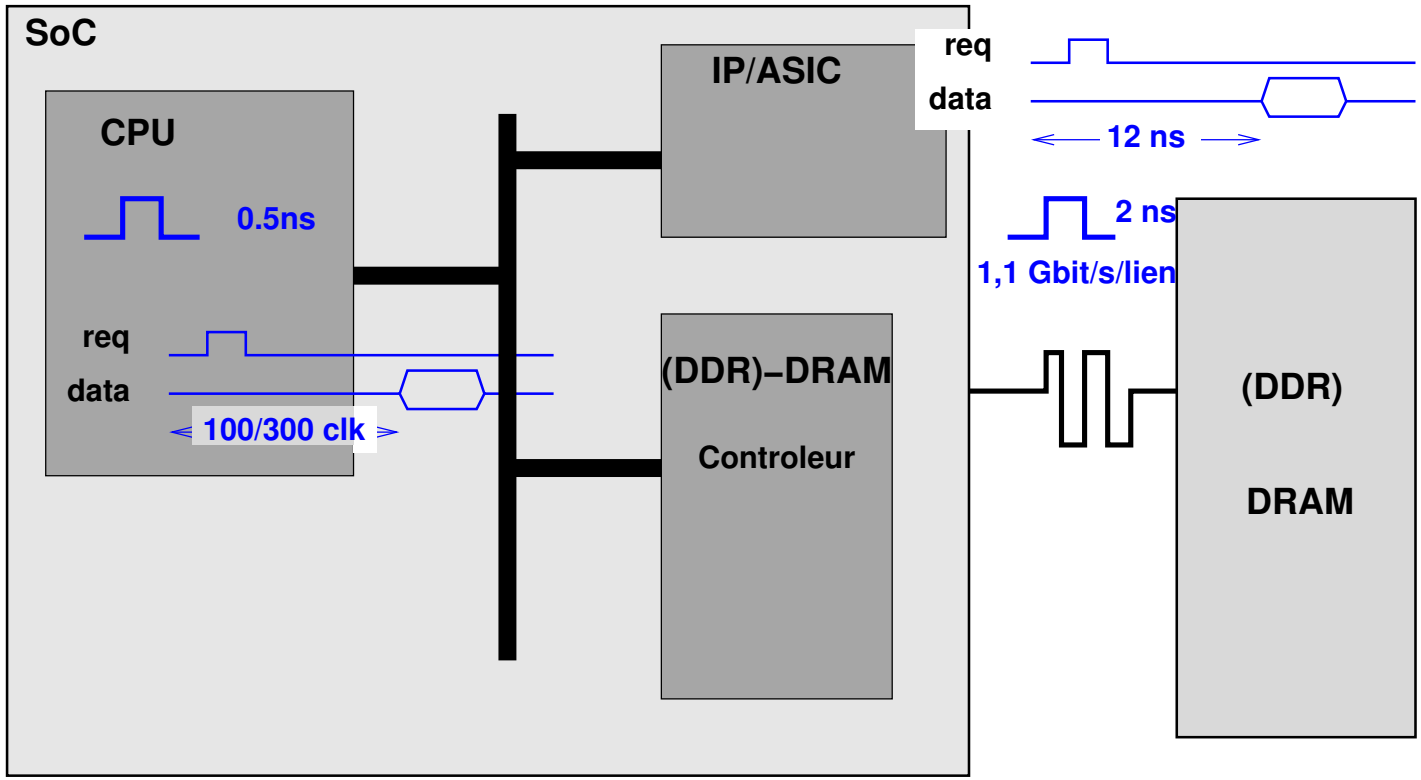
Problématique du “Memory Wall”



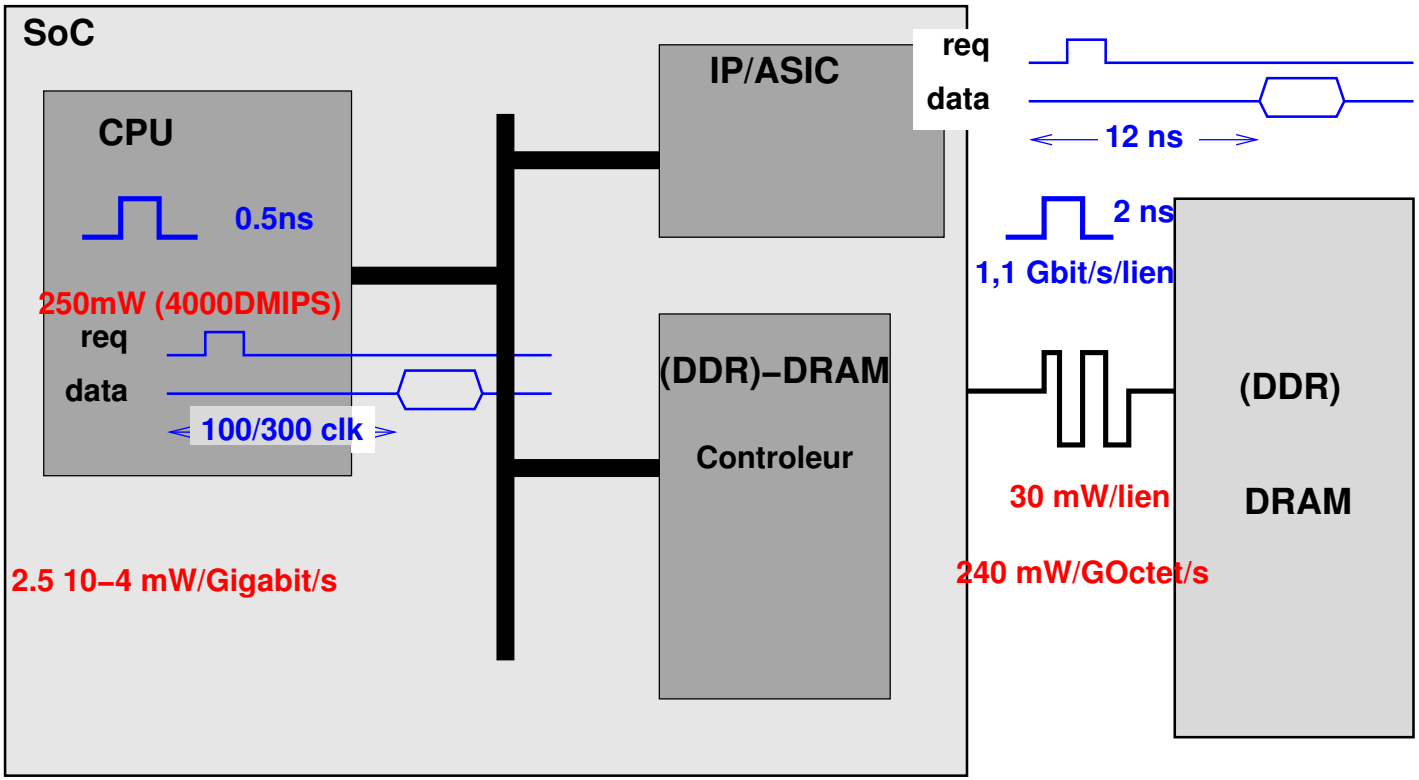
Problématique du “Memory Wall”



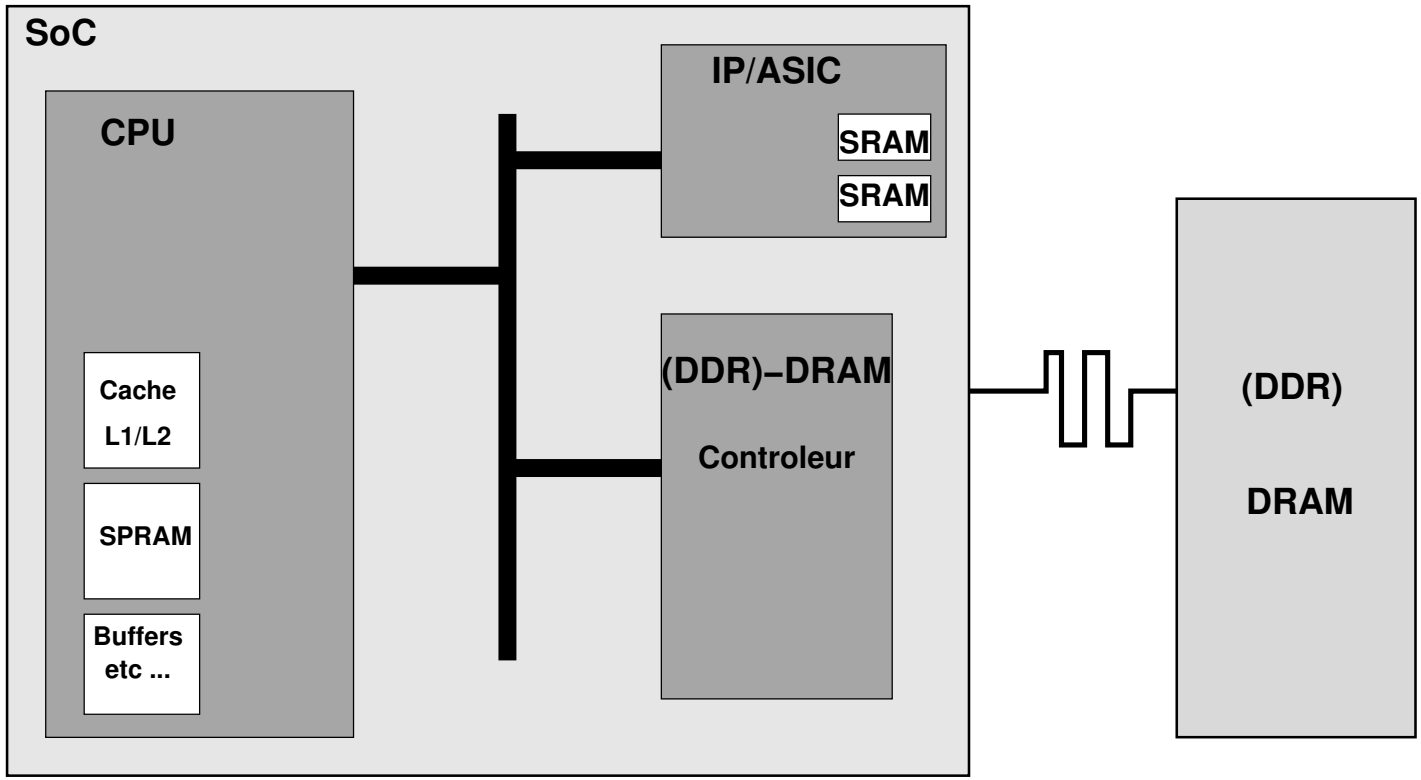
Problématique du “Memory Wall”



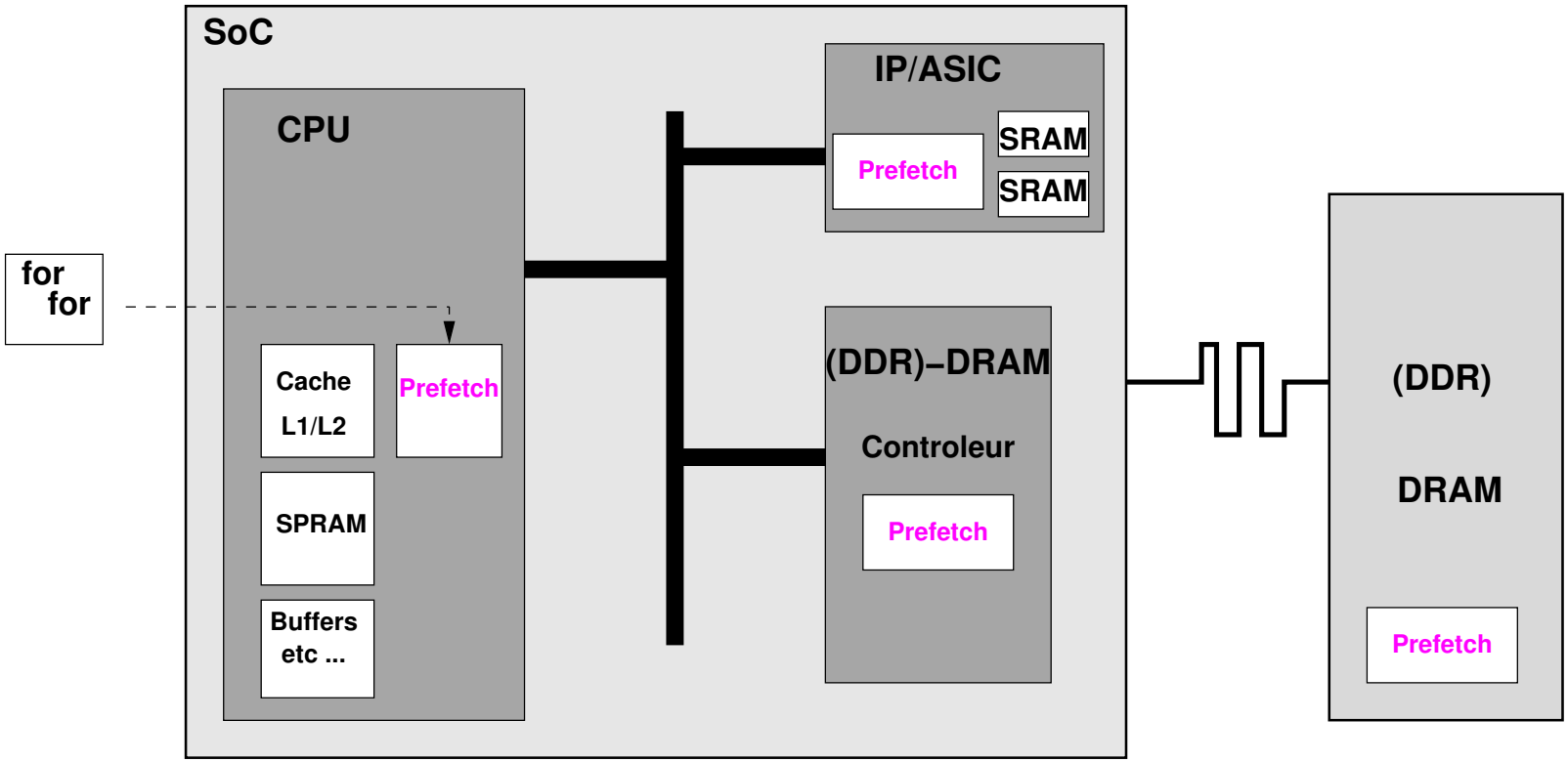
Problématique du “Memory Wall”



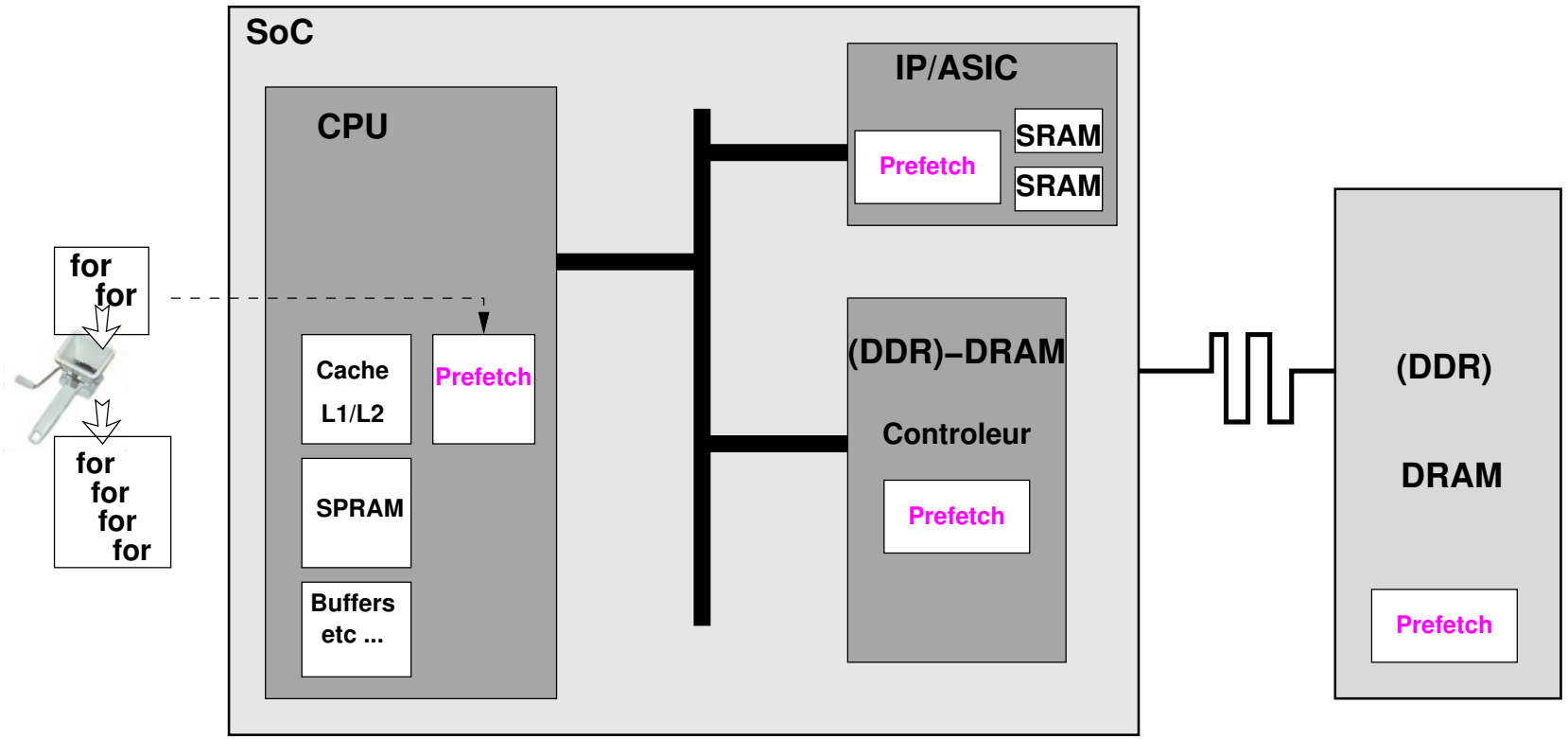
Problématique du “Memory Wall”



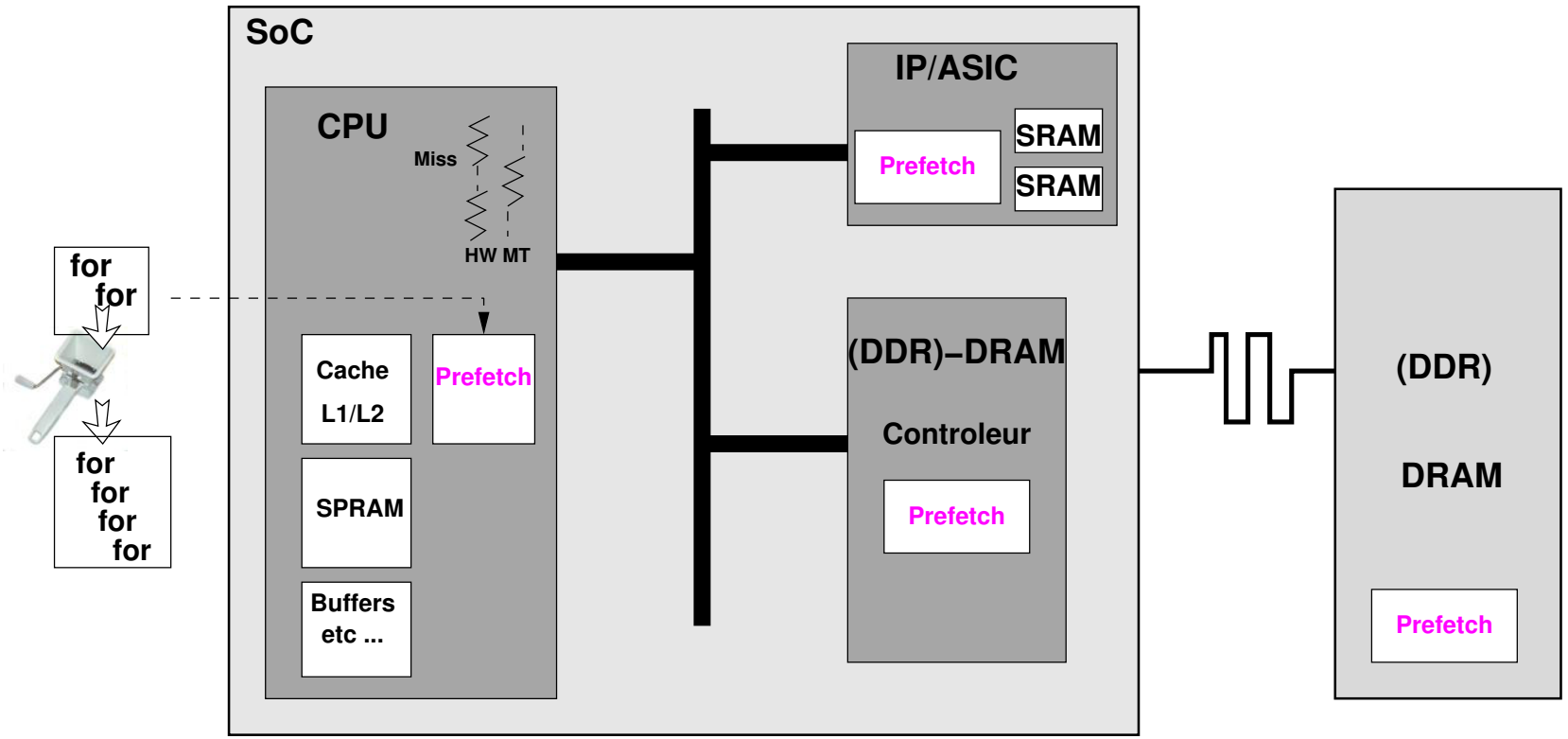
Problématique du “Memory Wall”



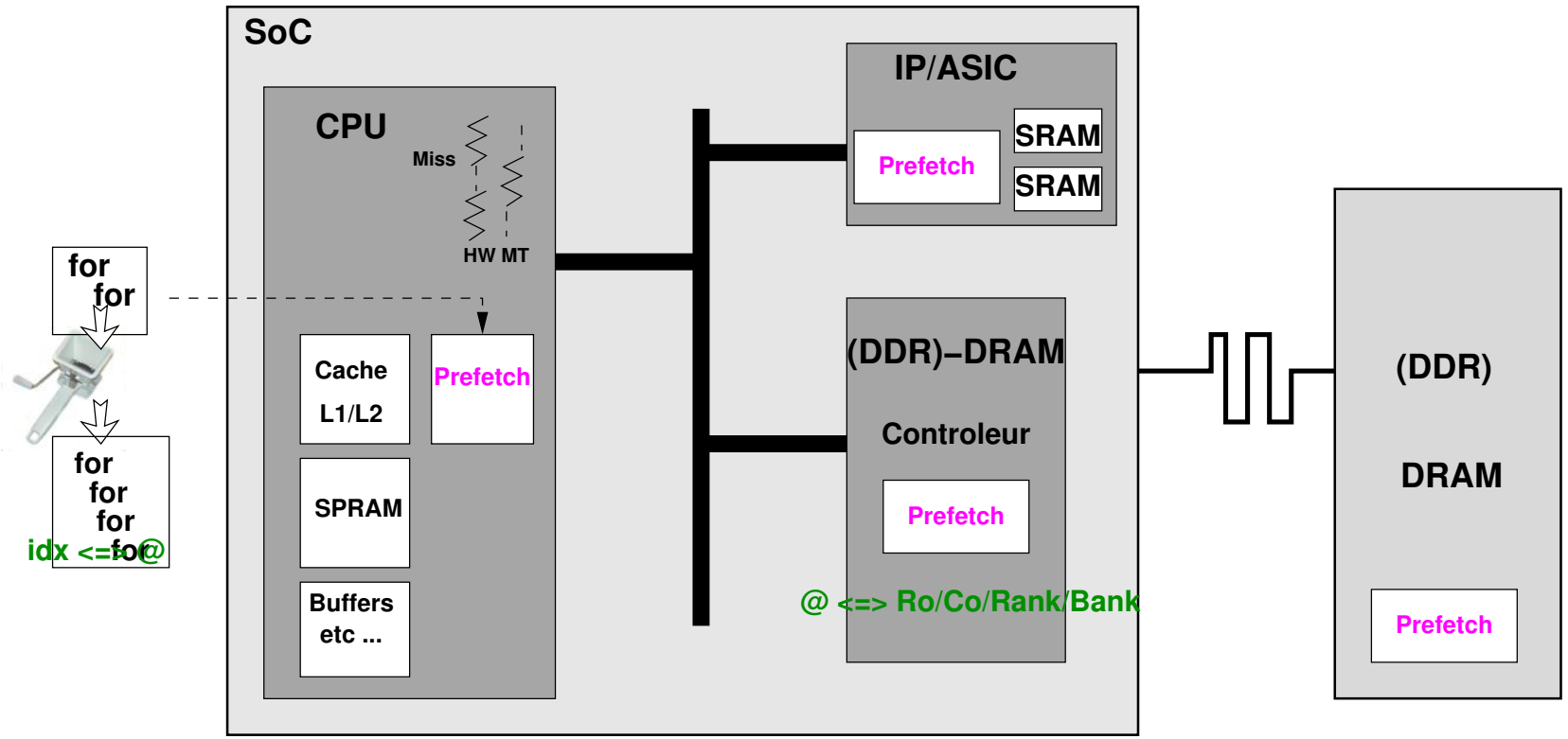
Problématique du “Memory Wall”



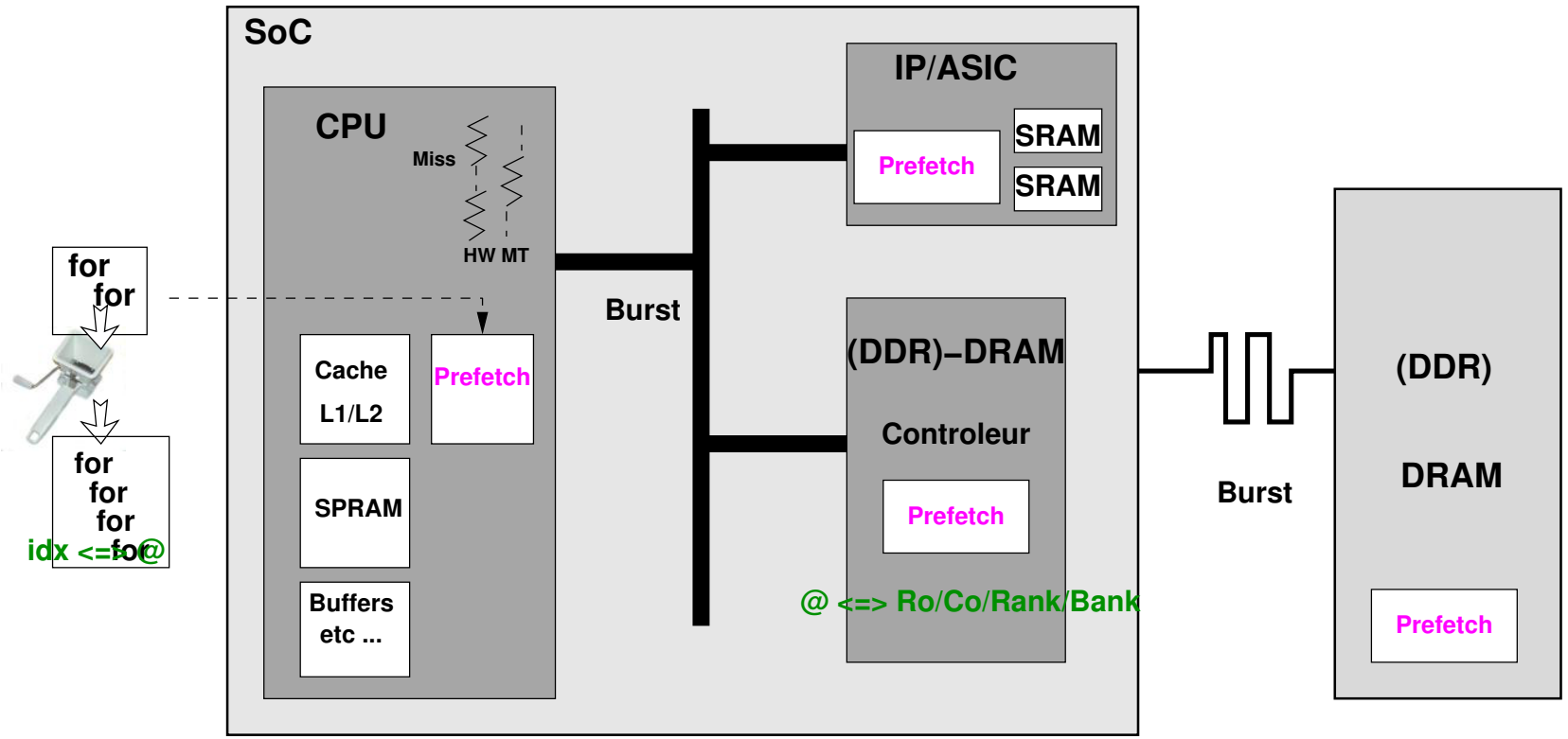
Problématique du “Memory Wall”



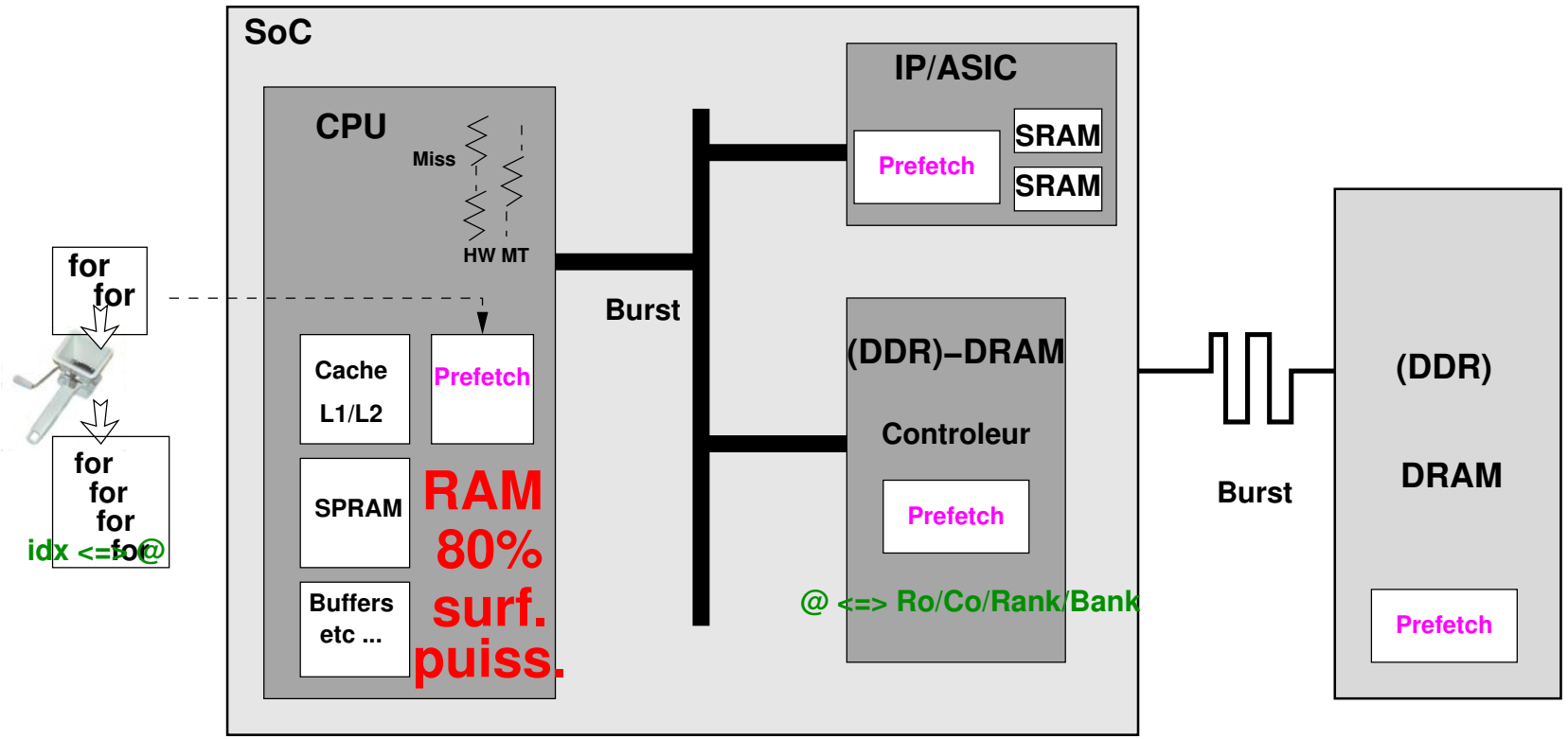
Problématique du “Memory Wall”



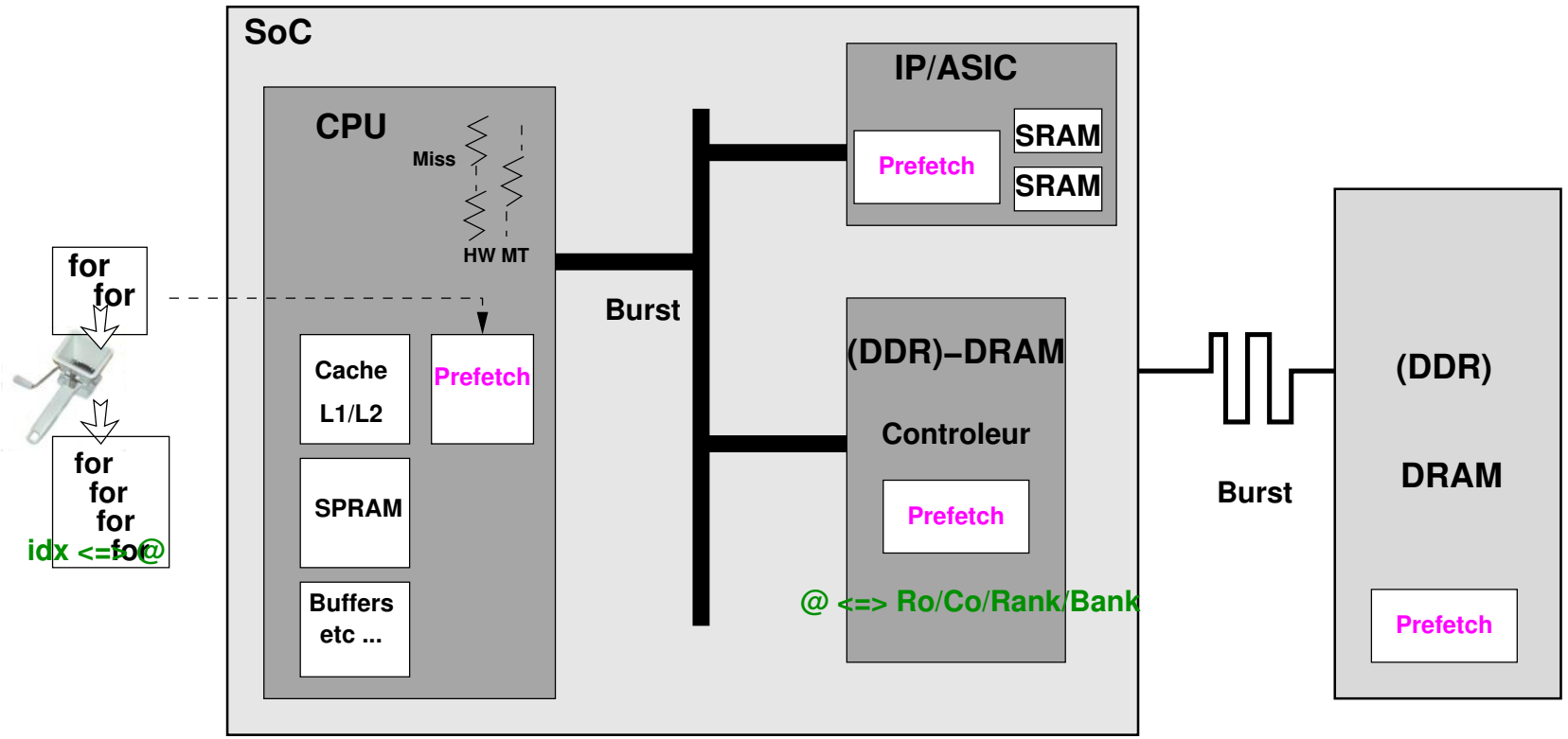
Problématique du “Memory Wall”



Problématique du “Memory Wall”



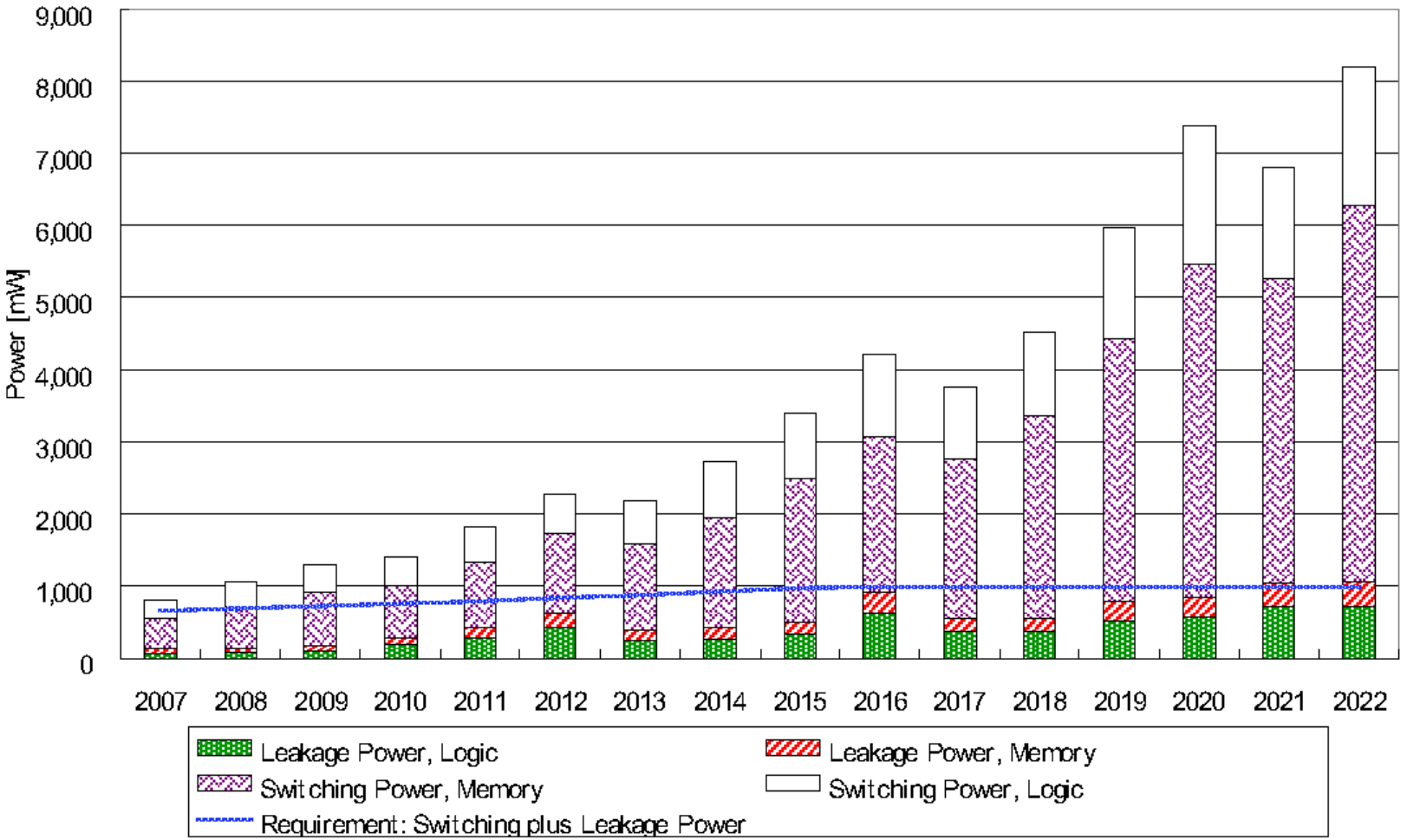
Problématique du “Memory Wall”



Pour de nombreuses classes d’applications,

les strategies de prefetch usuelles ne sont pas efficaces

Consommation d'énergie due aux mémoires



source: ITRS

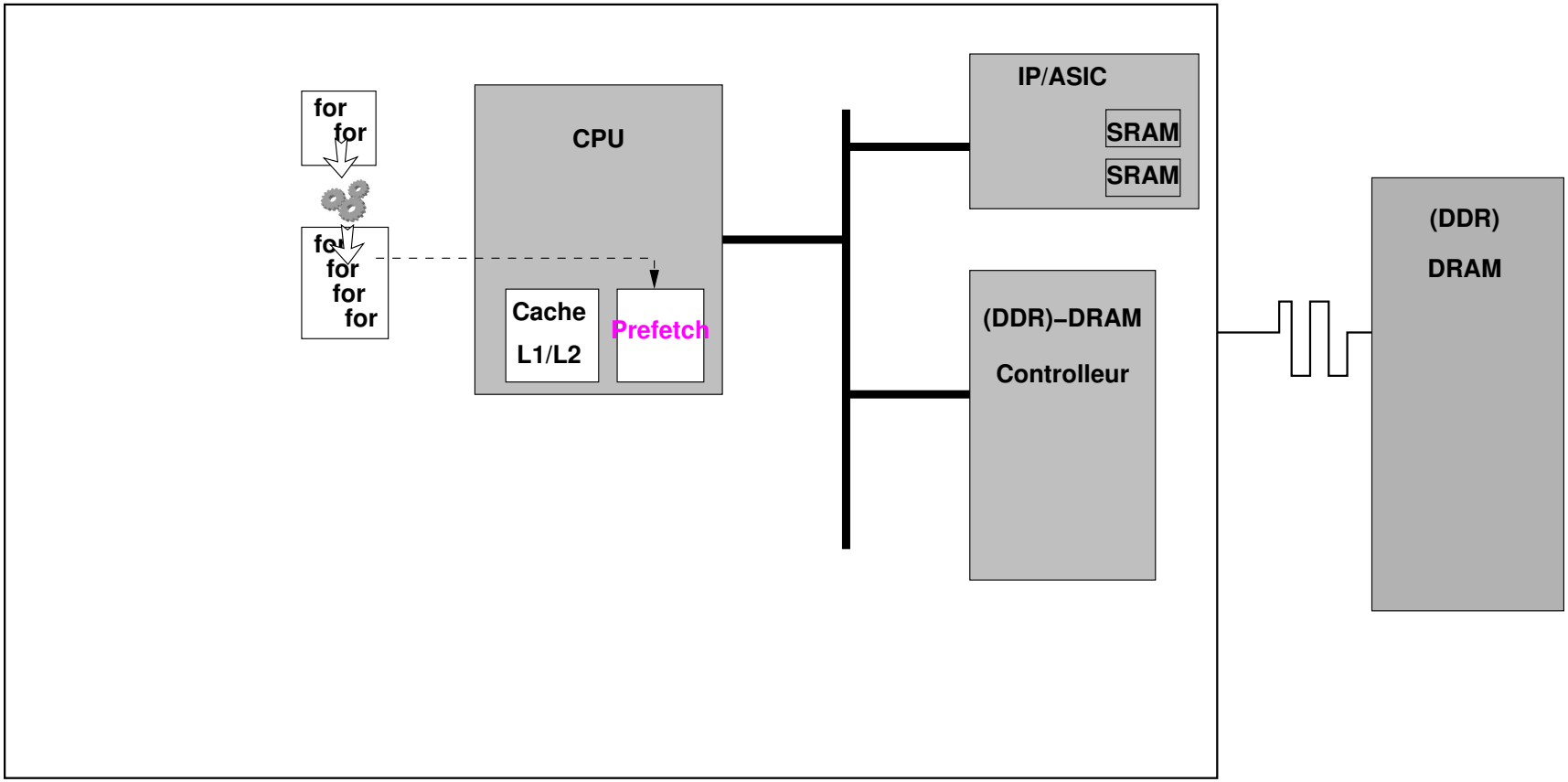
Comment “passer le mur” mémoire?

Optimisation de la gestion des données:

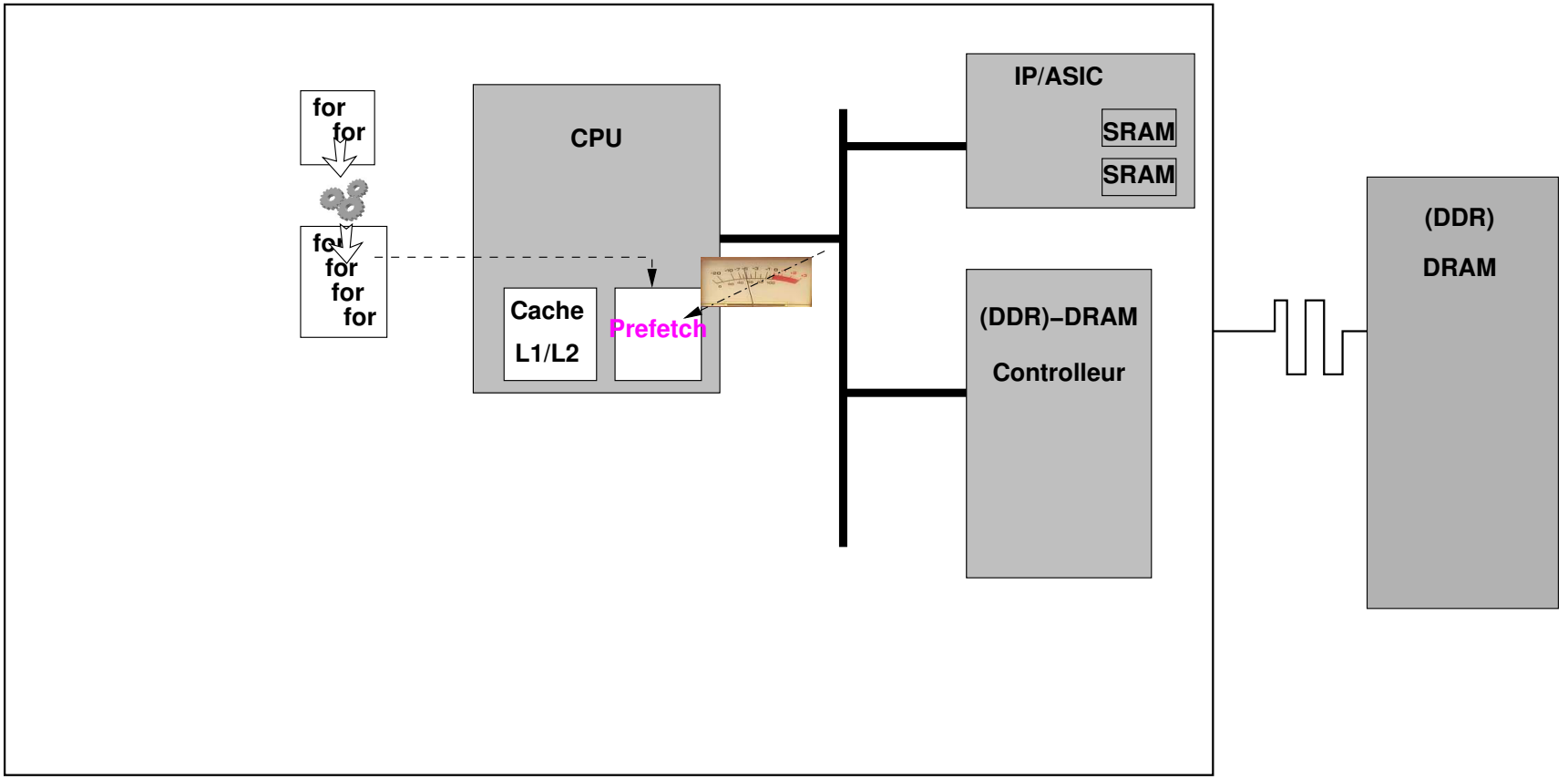
- Stratégies ”hors-ligne”
  - Connaissance des séquences d’accès
  - Ne prend pas en compte les variations de performance

- Stratégies “en-ligne”
  - Adaptation aux conditions de fonctionnement
  - Décision “locale”, seul le passé est connu, modèle de prédiction du futur

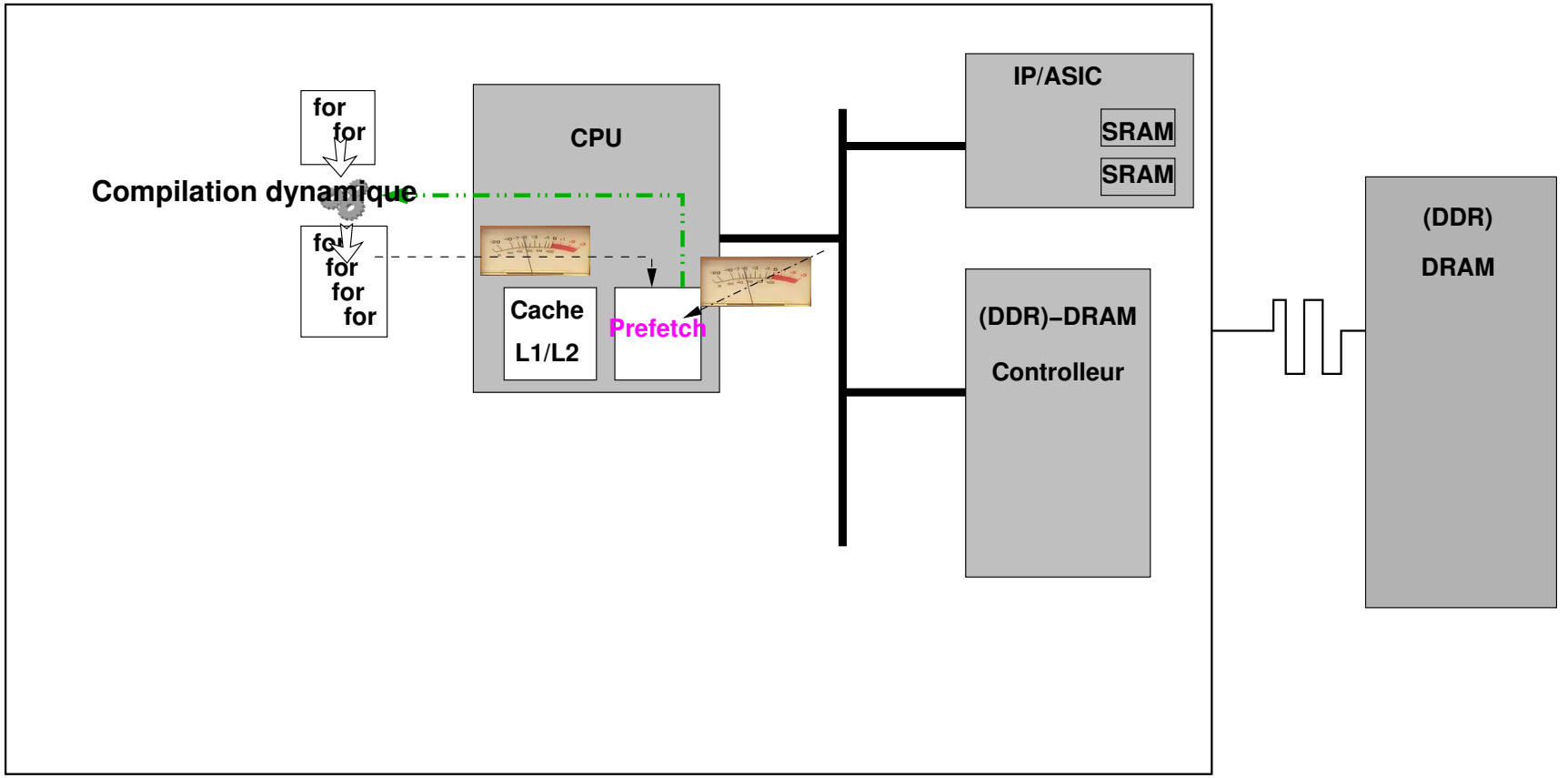
Une vision globale



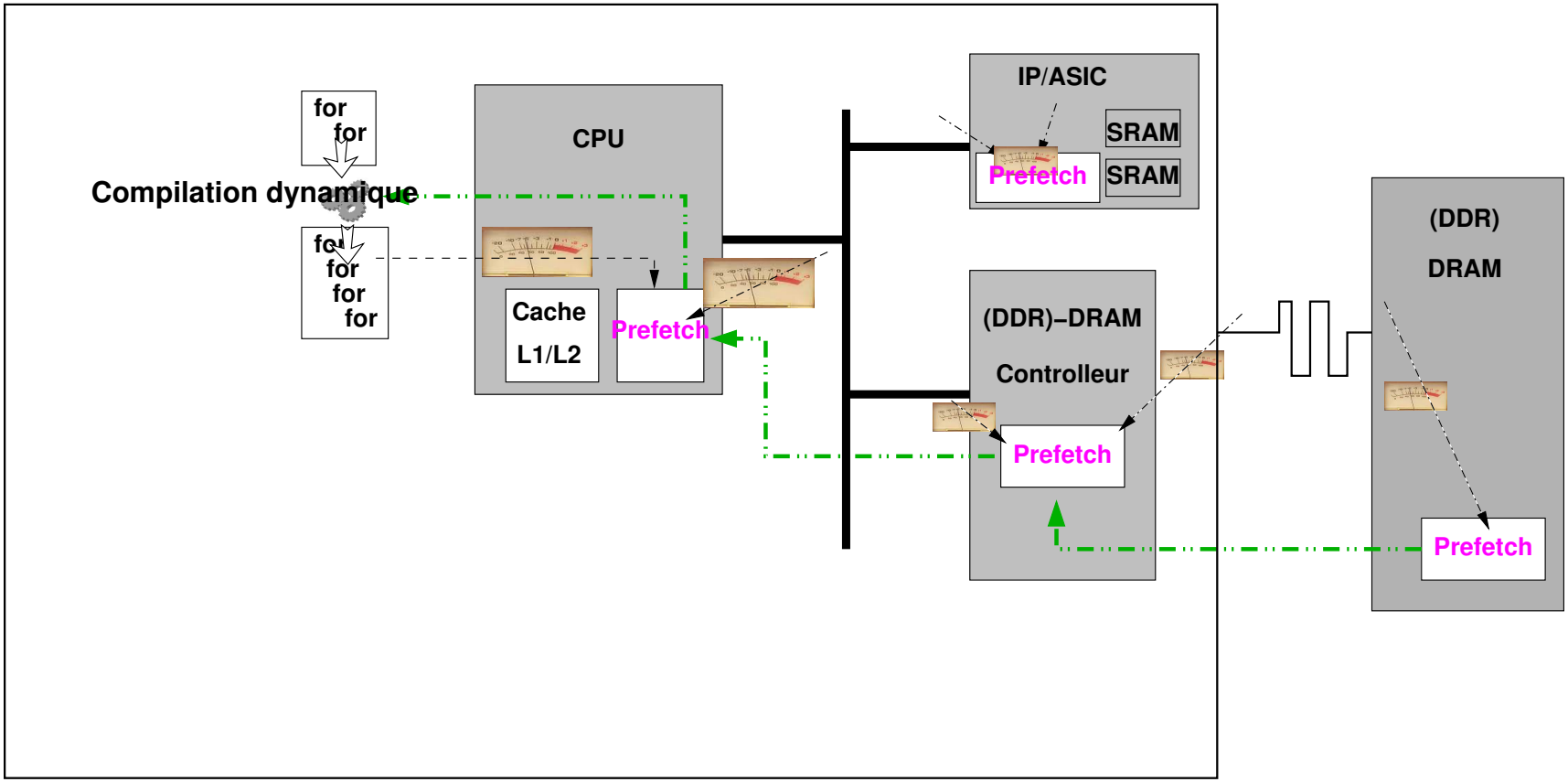
Une vision globale



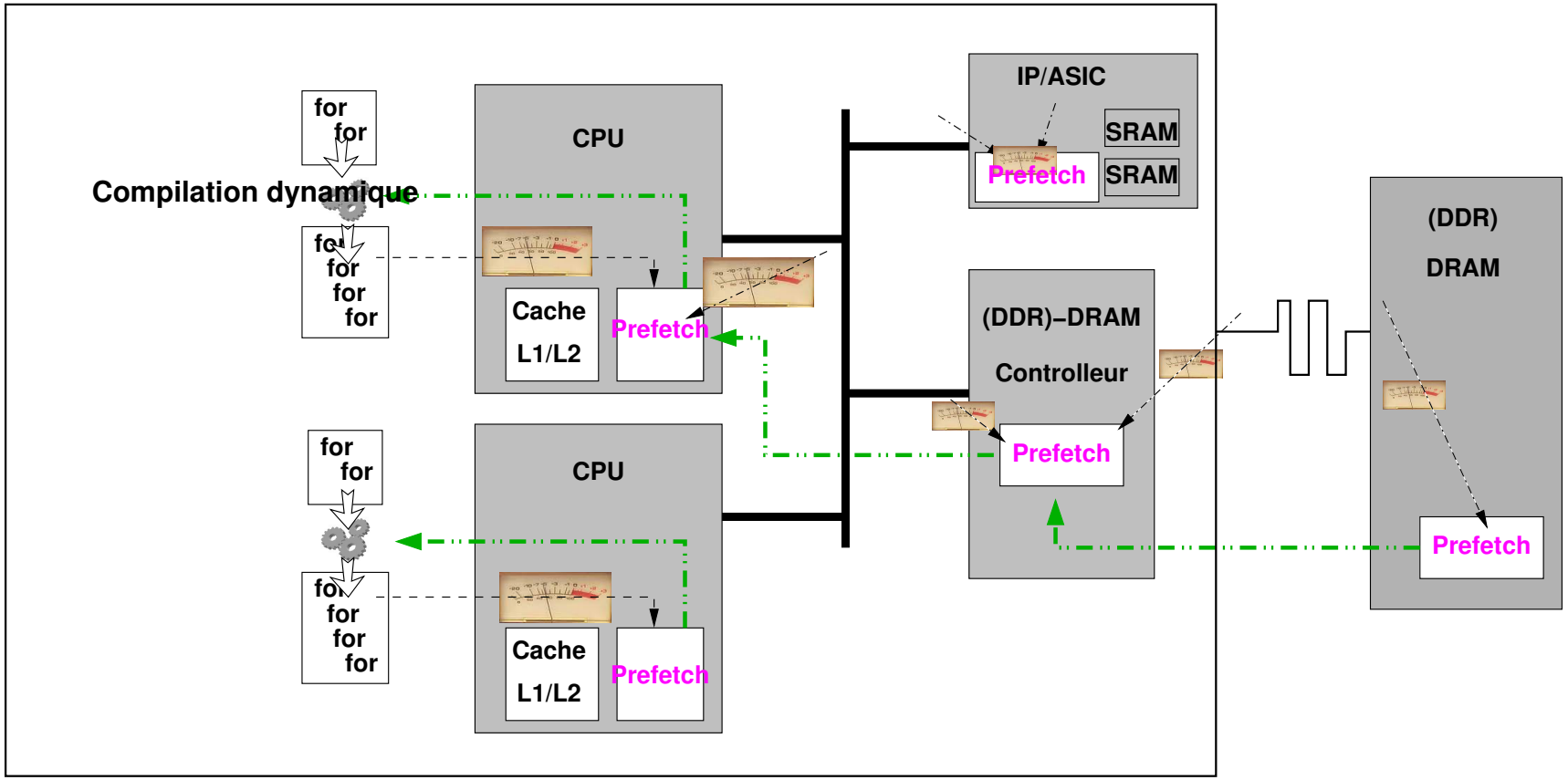
Une vision globale



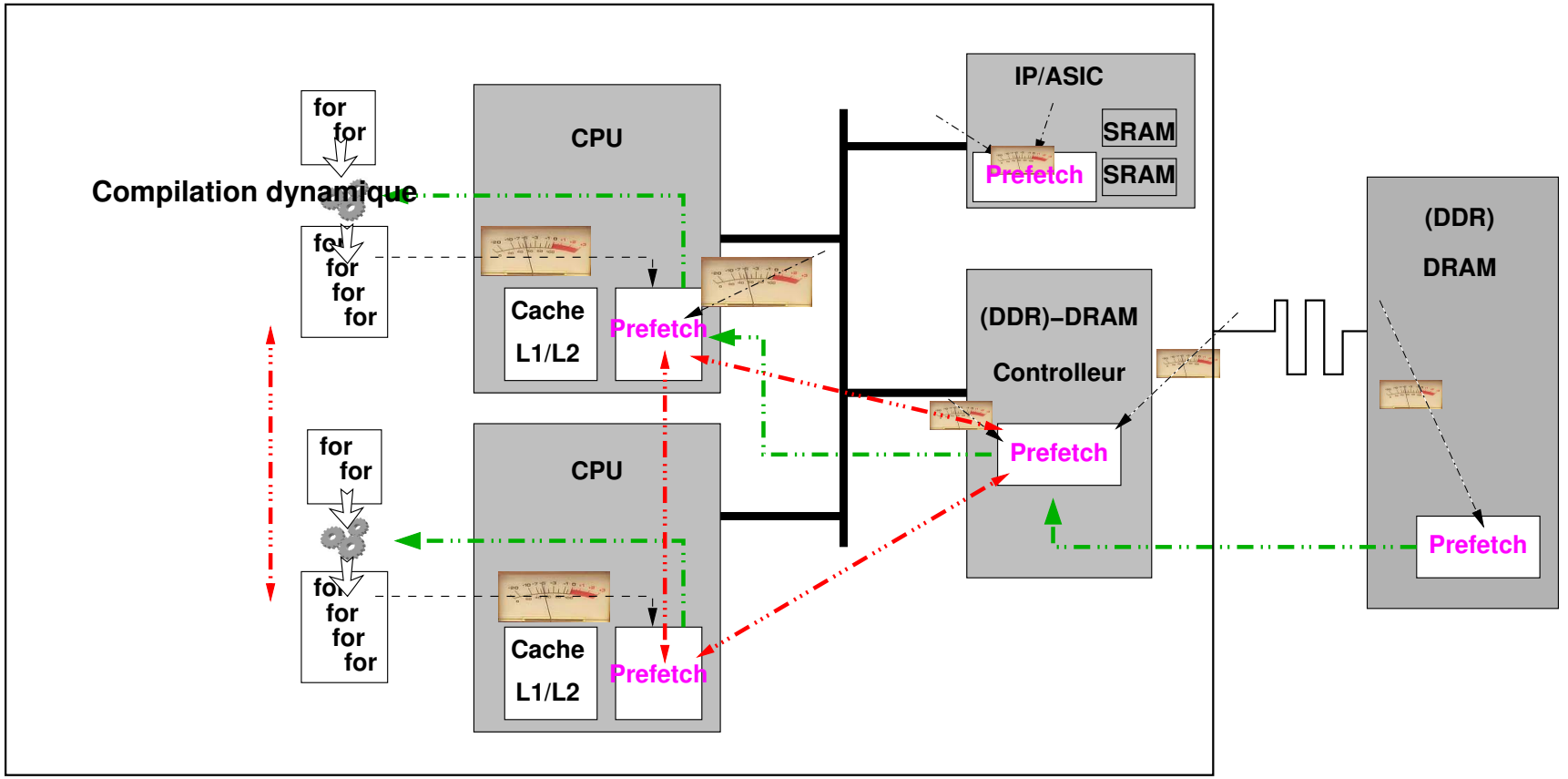
Une vision globale



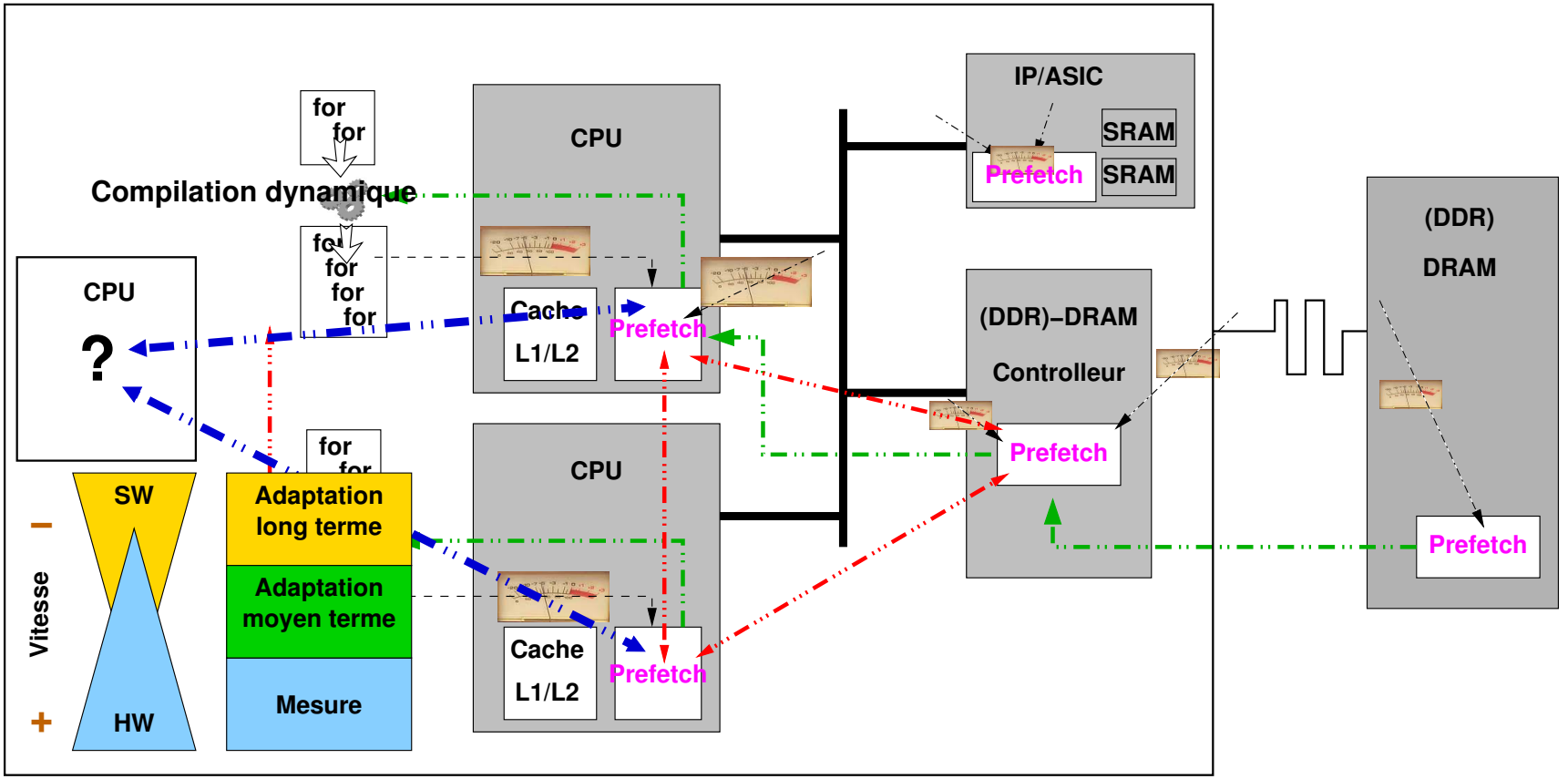
Une vision globale



Une vision globale



Une vision globale



Plan

- Le memory wall
- Le prefetch
- Le Cache nD-AP
- Conclusion & perspective

Le prefetch

Objectifs: pré-charger les données en cache avant d'en avoir besoin.

- SW: insertion d'instructions de pré-charge (nonbloquante)
- HW: prédire au fur et à mesure des accès mémoire

Concevoir une stratégie de prefetch, c'est choisir:

- Quelles informations exploiter
- Quand lancer la pré-charge
- Quelles données pré-charger
- Où placer les données pré-chargées

Critères d'efficacité:

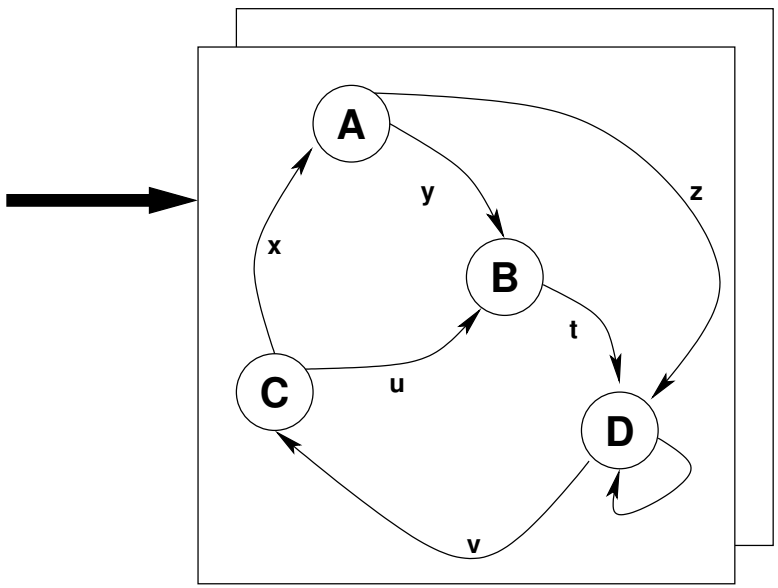
- Temps d'exécution totale
- Pollution des bus
- Complexité

Prefetch usuels

Historique:

- Caches “classique” (ligne de cache)
- OBL (One Block Lookahead)
- SPT
- Stream-buffers

Généralisation: prefetch à base de chaîne de Markov

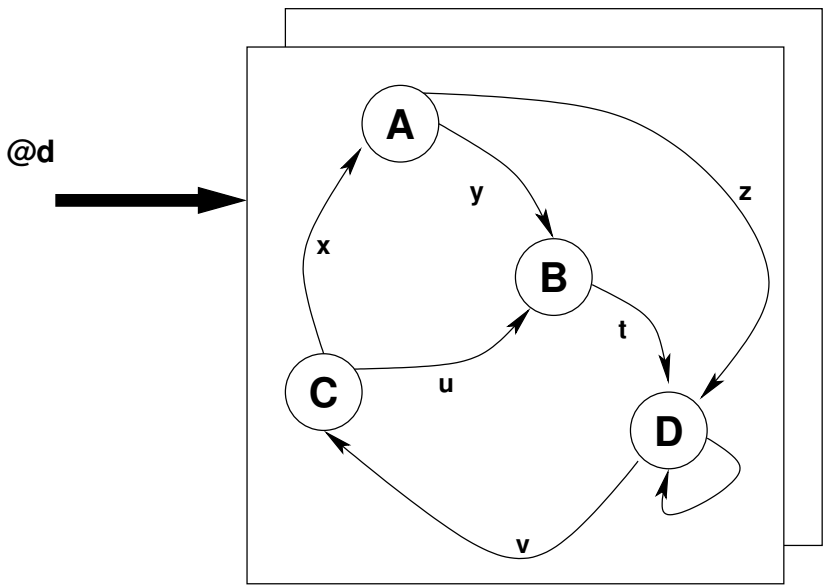


Prefetch usuels

Historique:

- Caches “classique” (ligne de cache)
- OBL (One Block Lookahead)
- SPT
- Stream-buffers

Généralisation: prefetch à base de chaîne de Markov

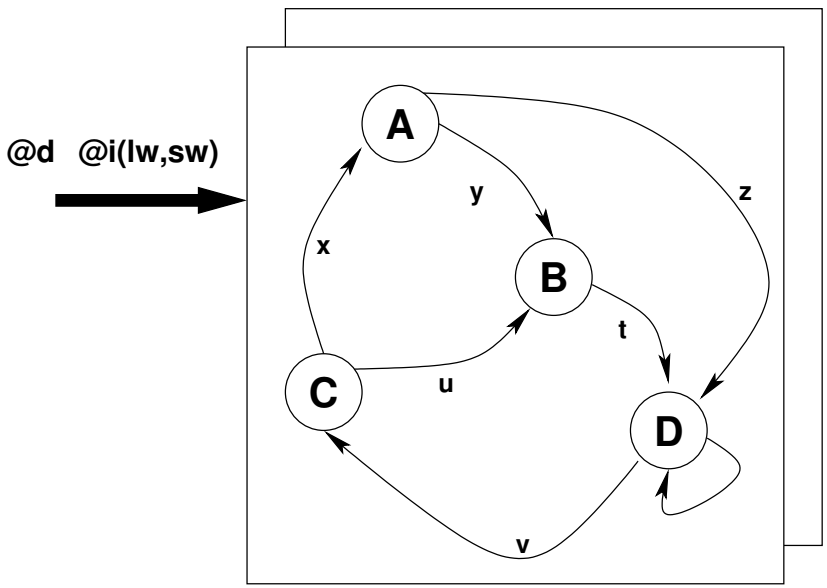


Prefetch usuels

Historique:

- Caches “classique” (ligne de cache)
- OBL (One Block Lookahead)
- SPT
- Stream-buffers

Généralisation: prefetch à base de chaîne de Markov

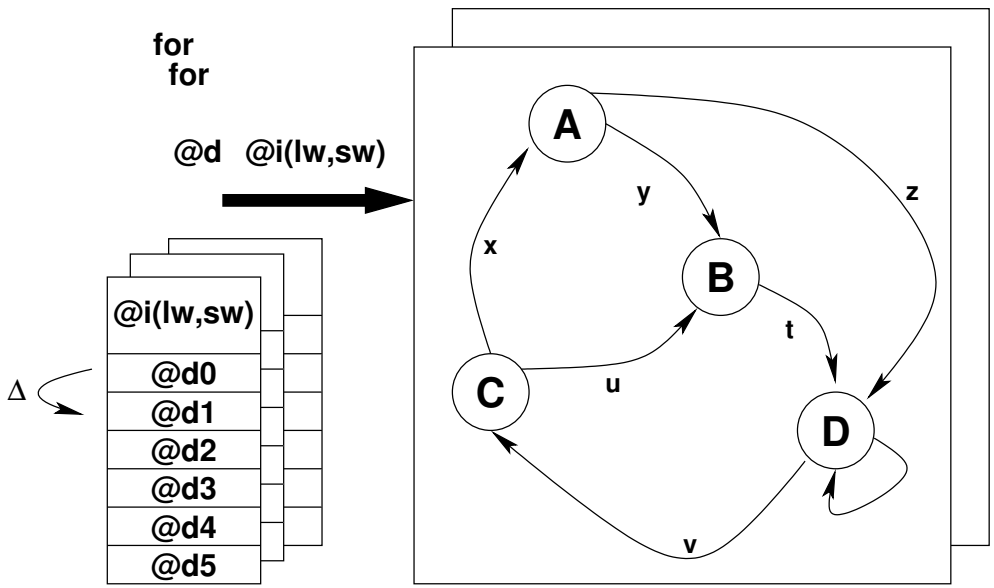


Prefetch usuels

Historique:

- Caches “classique” (ligne de cache)
- OBL (One Block Lookahead)
- SPT
- Stream-buffers

Généralisation: prefetch à base de chaîne de Markov

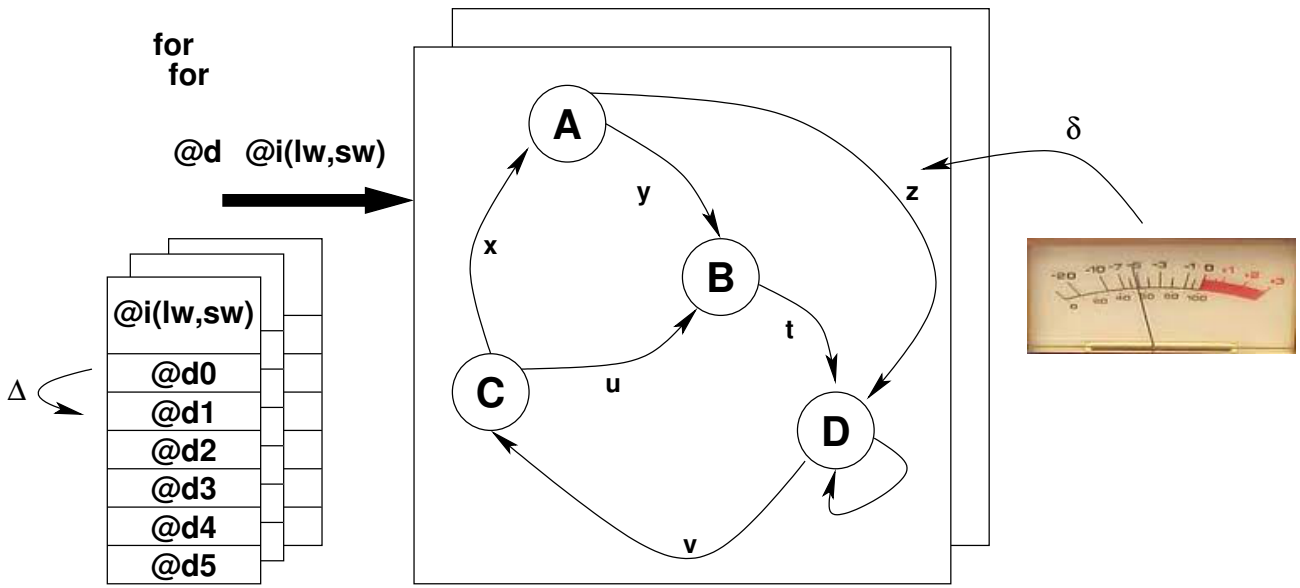


Prefetch usuels

Historique:

- Caches “classique” (ligne de cache)
- OBL (One Block Lookahead)
- SPT
- Stream-buffers

Généralisation: prefetch à base de chaîne de Markov



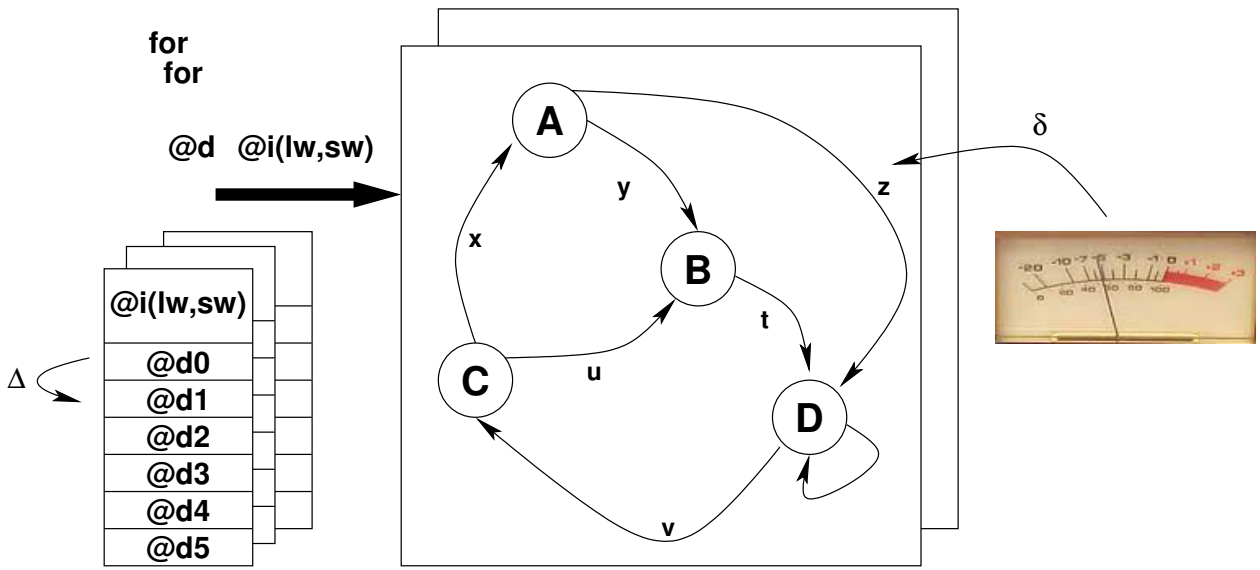
Prefetch usuels

Historique:

- Caches “classique” (ligne de cache)
- OBL (One Block Lookahead)
- SPT
- Stream-buffers

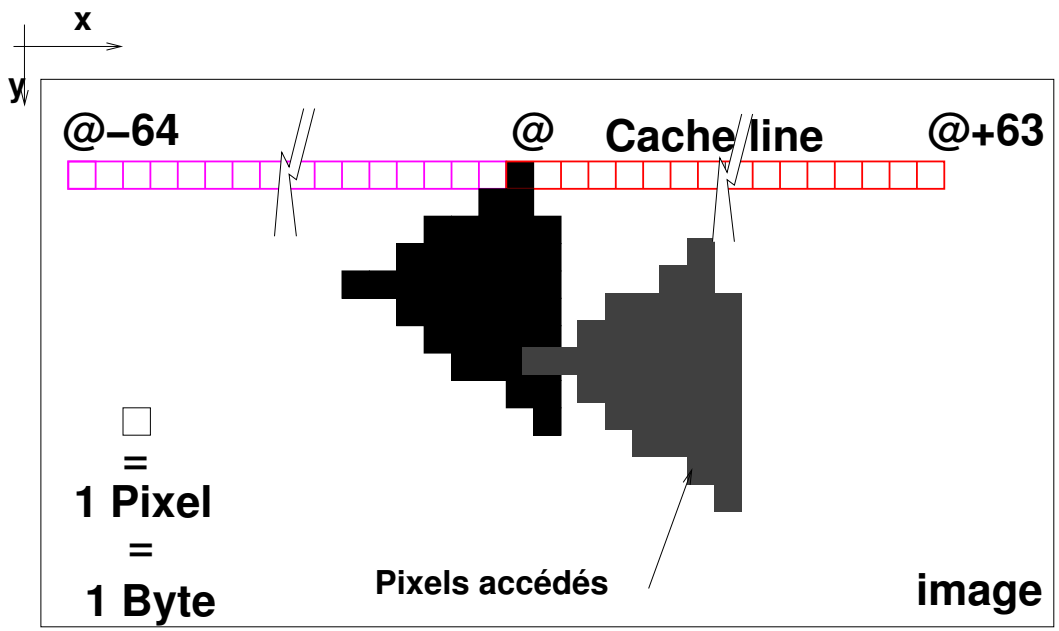
Généralisation: prefetch à base de chaîne de Markov

- Aucune stratégie ne surpasse SPT



Limitations

Les caches classiques ne sont pas adaptés au traitement d'image car ils exploitent la localité **temporelle** et la localité **spatiale en adresse**.



Les défauts de cache sont déclenchés lors d'un déplacement:

- Horizontal: à la fin d'une ligne de cache
- Vertical: à chaque nouvelle ligne de l'image

Plan

- Le memory wall
- Le prefetch
- Le Cache nD-AP
- Conclusion & perspective

Positionnement

Pré-chargement dynamique

- Détection de motifs répétitifs
  - Générique car le prefetch exploite les adresses
    - Implanté dans les processeurs modernes (SPT)
  - Techniques dérivées des méthodes “Markoviennes”
  - Inefficaces pour les séquences non strictement régulières en adresse

- Estimation de paramètres
  - Prise en compte de la structure de donnée, pour une classe d’applications
  - Résistant aux fluctuations “système” et irrégularités des accès
  - Usage restreint aux applications conformes au modèle

Objectifs

Une stratégie de prefetch adaptée au TDSI, et, plus largement, aux **applications embarquées** qui utilisent des **données structurées** (tableaux, arbres, multi-résolution,. . .).

Caractéristiques souhaitées:

- Exploiter la localité temporelle et la **localité spatiale en index** dans la structure
- Proposer un mécanisme **semi-général**, pour une grande classe d'application
- Une architecture de **complexité réduite** pour cible **ASIC ou FPGA**

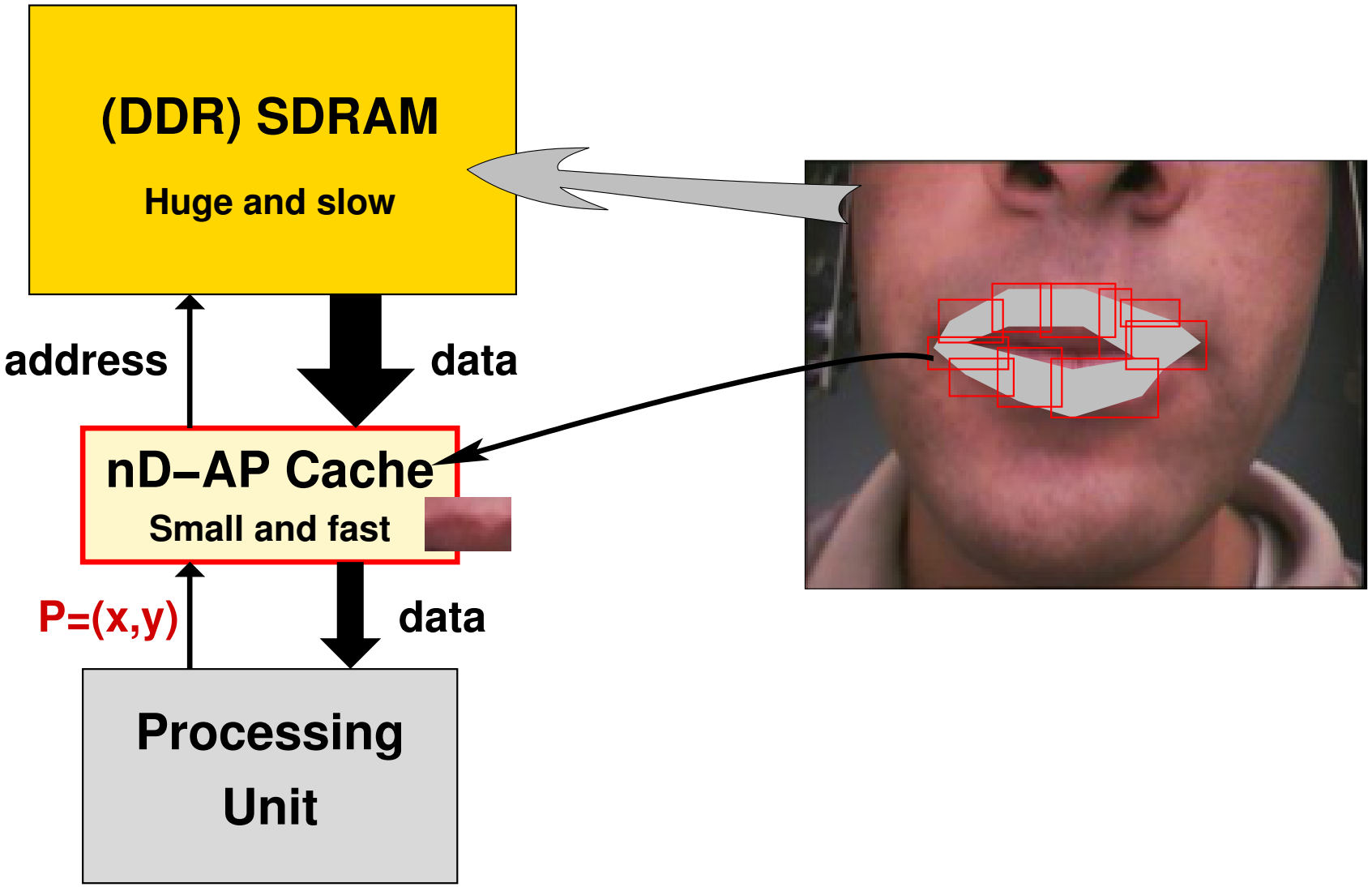
Séquences d'accès ciblées:

- Déterministes fonction de paramètres statiques
- Déterministes dépendant de paramètres dynamiques
- Dépendant des données

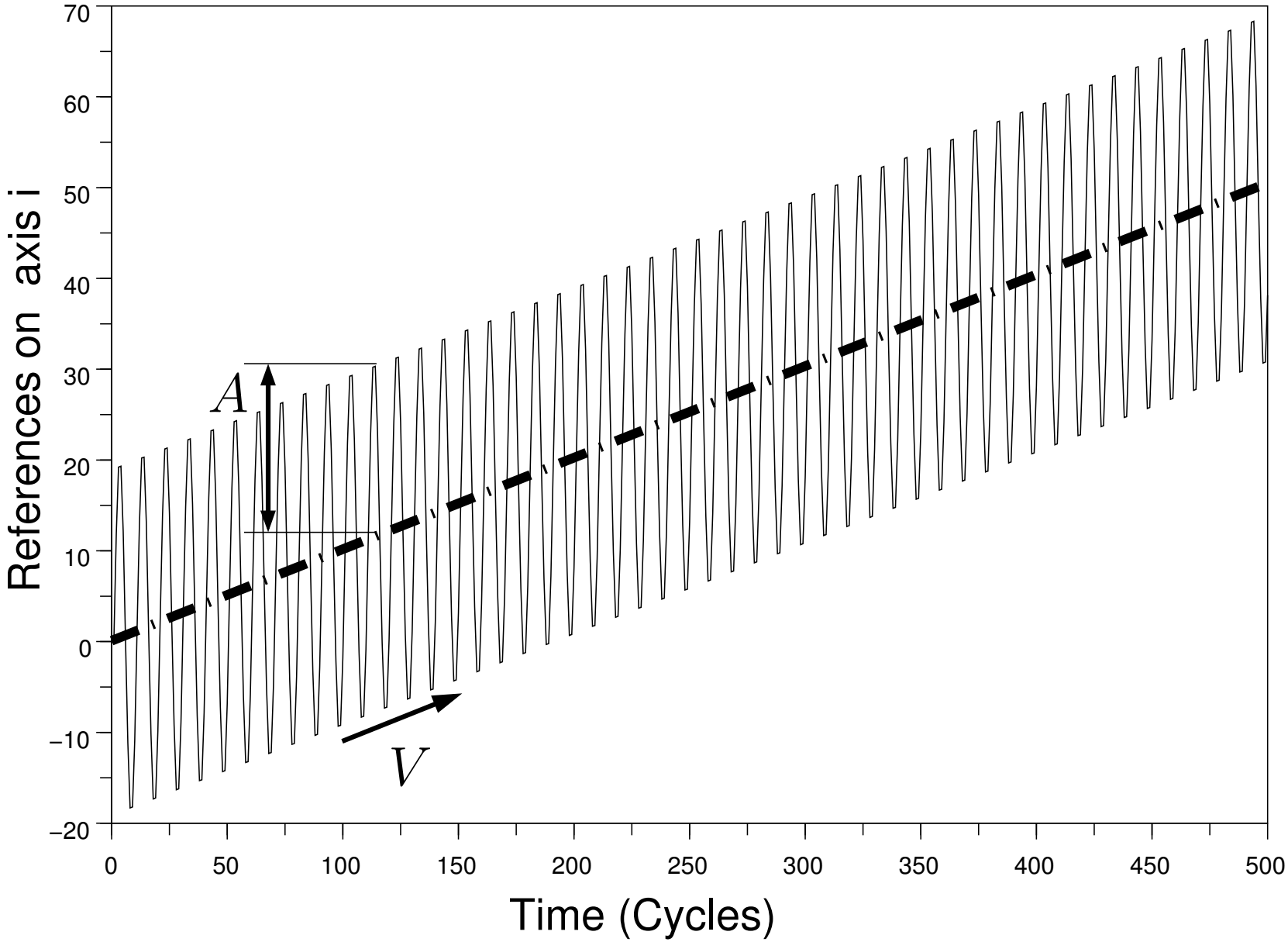
Idée:

- ☞ Intégrer l'aspect **multi-dimensionnel**
- ☞ Faire des prédictions à **moyen terme** sur un **voisinage proche** (en index)

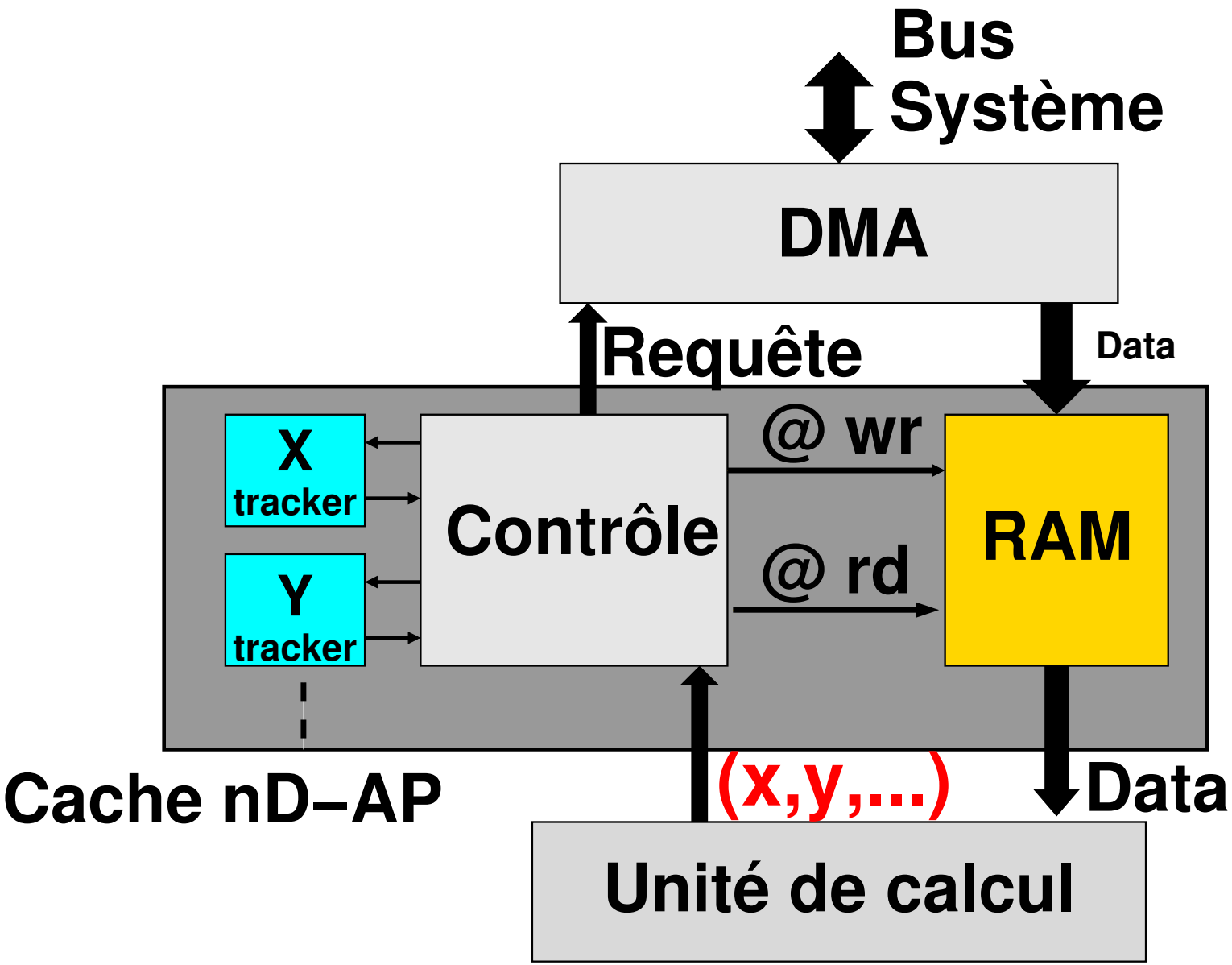
Concept du CachenD



Modèle de séquence



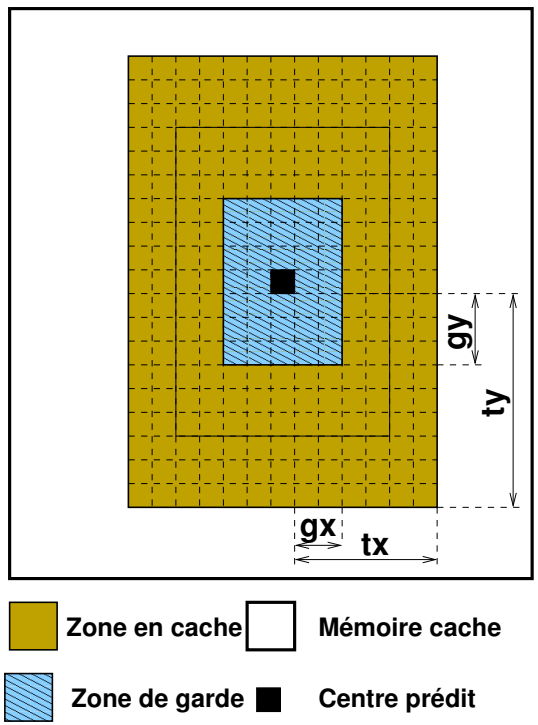
Architecture du Cache nD-AP



Mécanisme de prédiction et gestion mémoire

Sur chaque axe, un estimateur calcule une zone de références prédites:

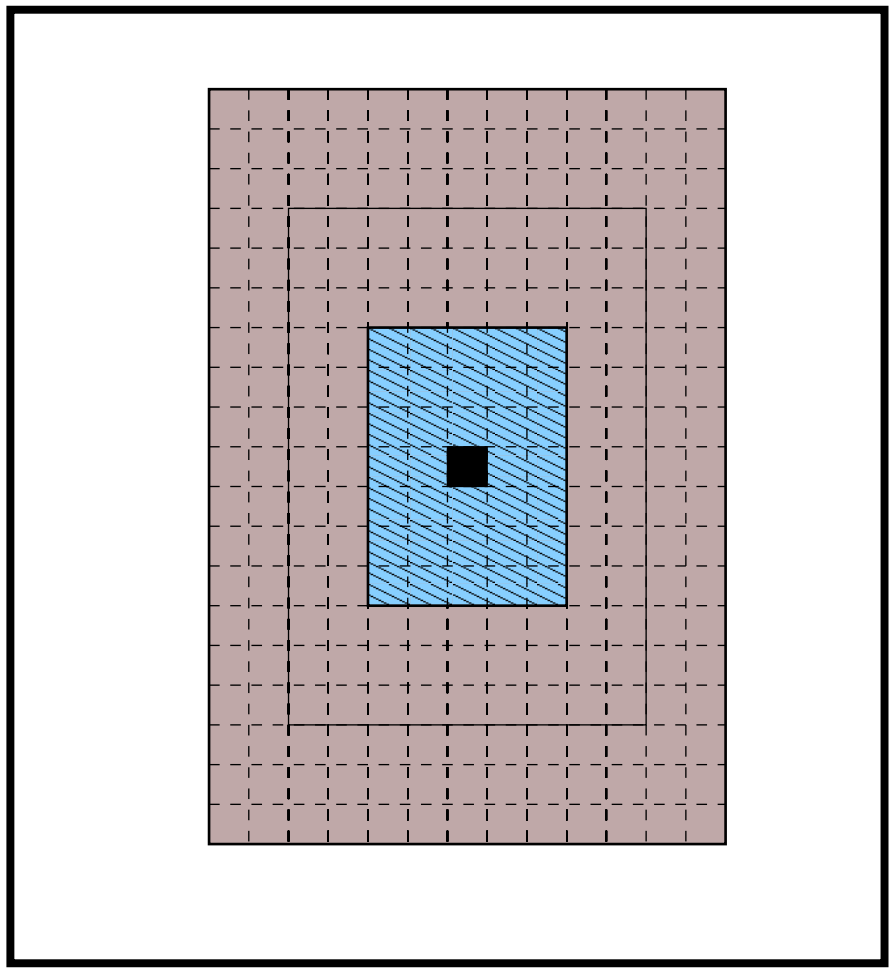
- La prédiction, basée sur un modèle, tient compte du passé et estime les tendances moyennes (filtrage)
- La charge à la mémoire centrale est limitée par un mécanisme à hystérésis.



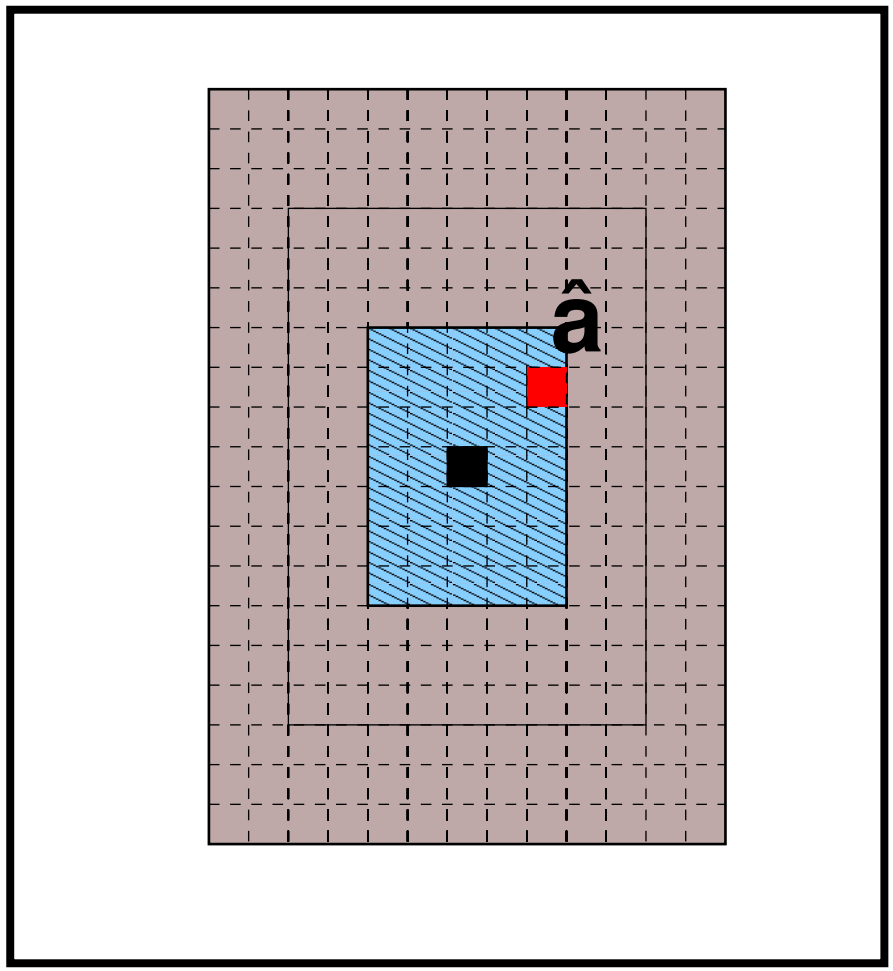
Sont estimés:

- c, le centre courant de la zone en cache
- t, la taille de la zone en cache
- g, la taille de la zone de garde
- d, la vitesse du cache

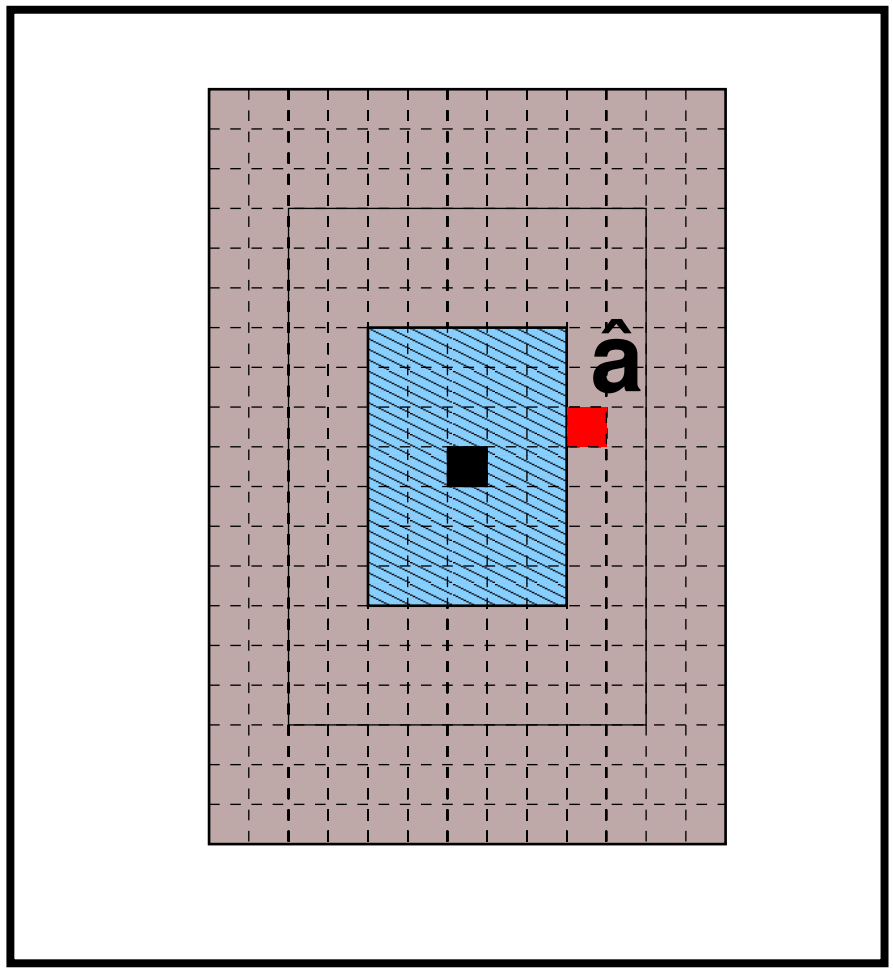
Comportement du Cache nD-AP



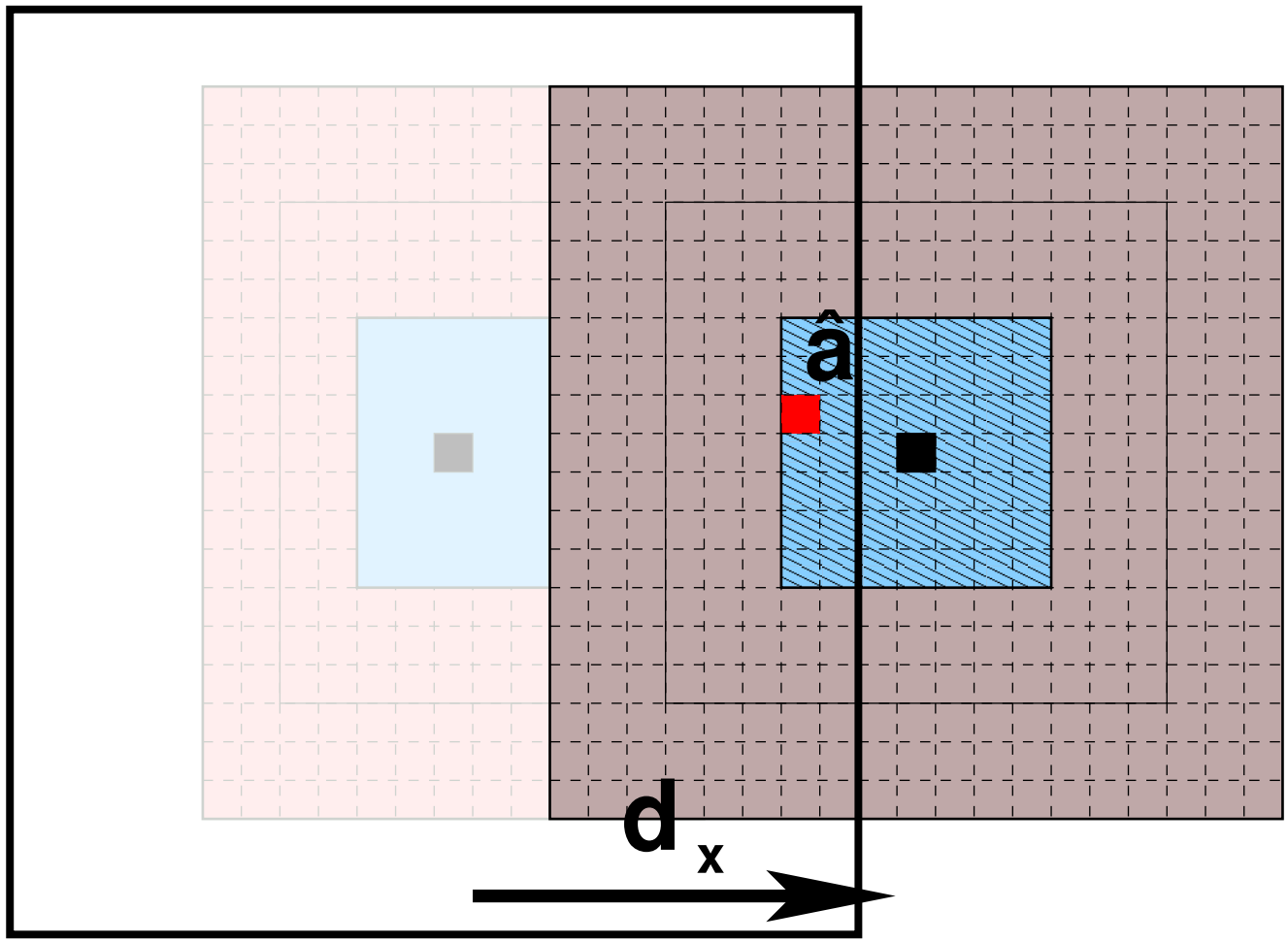
Comportement du Cache nD-AP



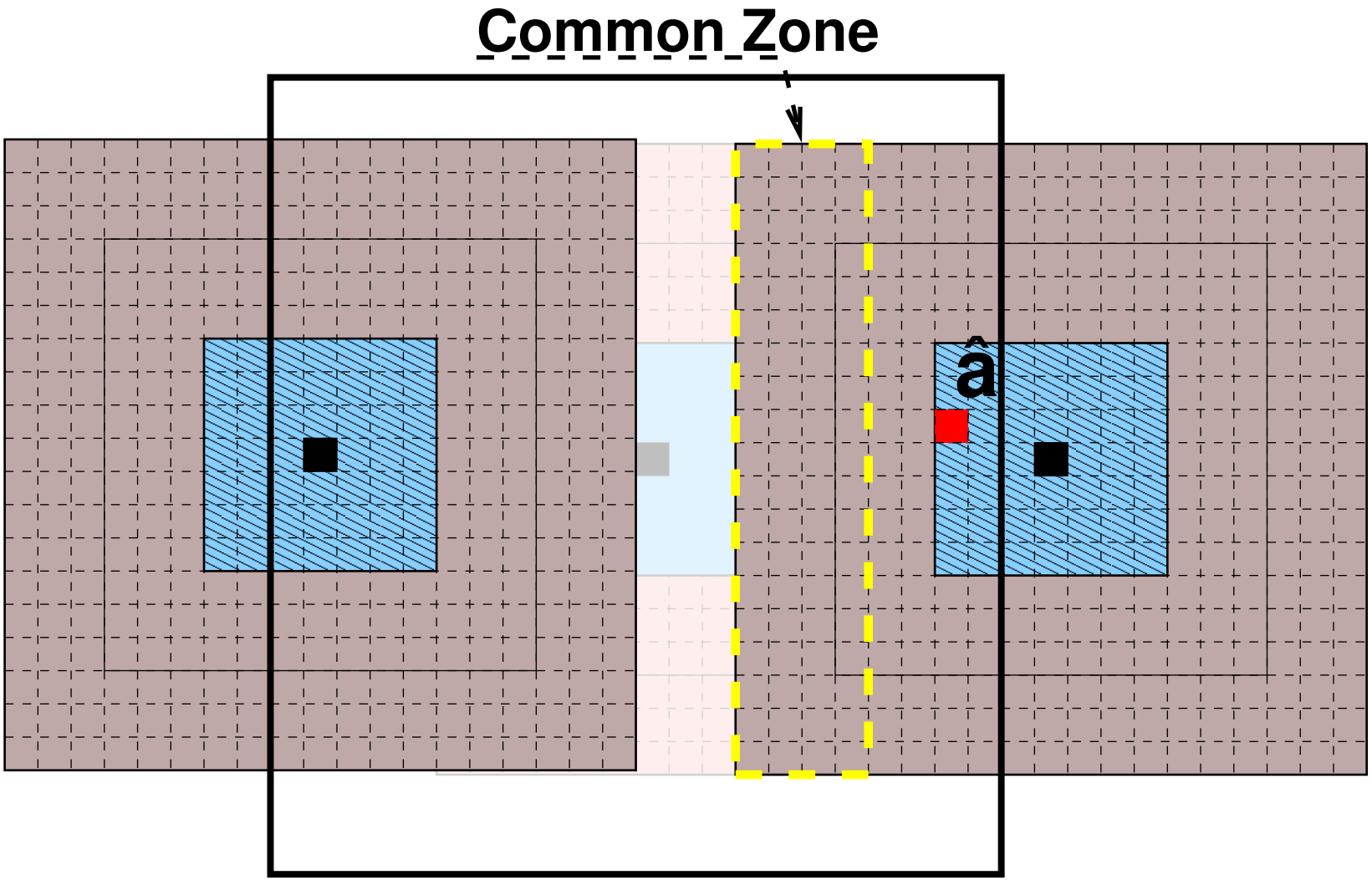
Comportement du Cache nD-AP



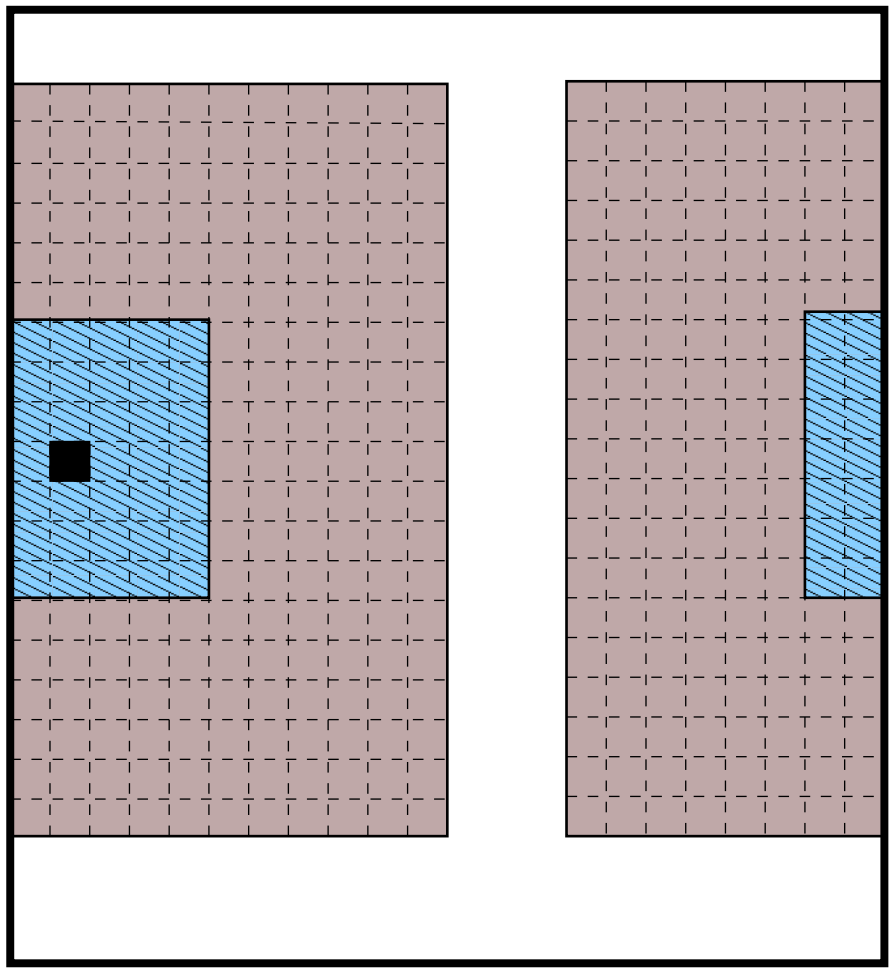
Comportement du Cache nD-AP



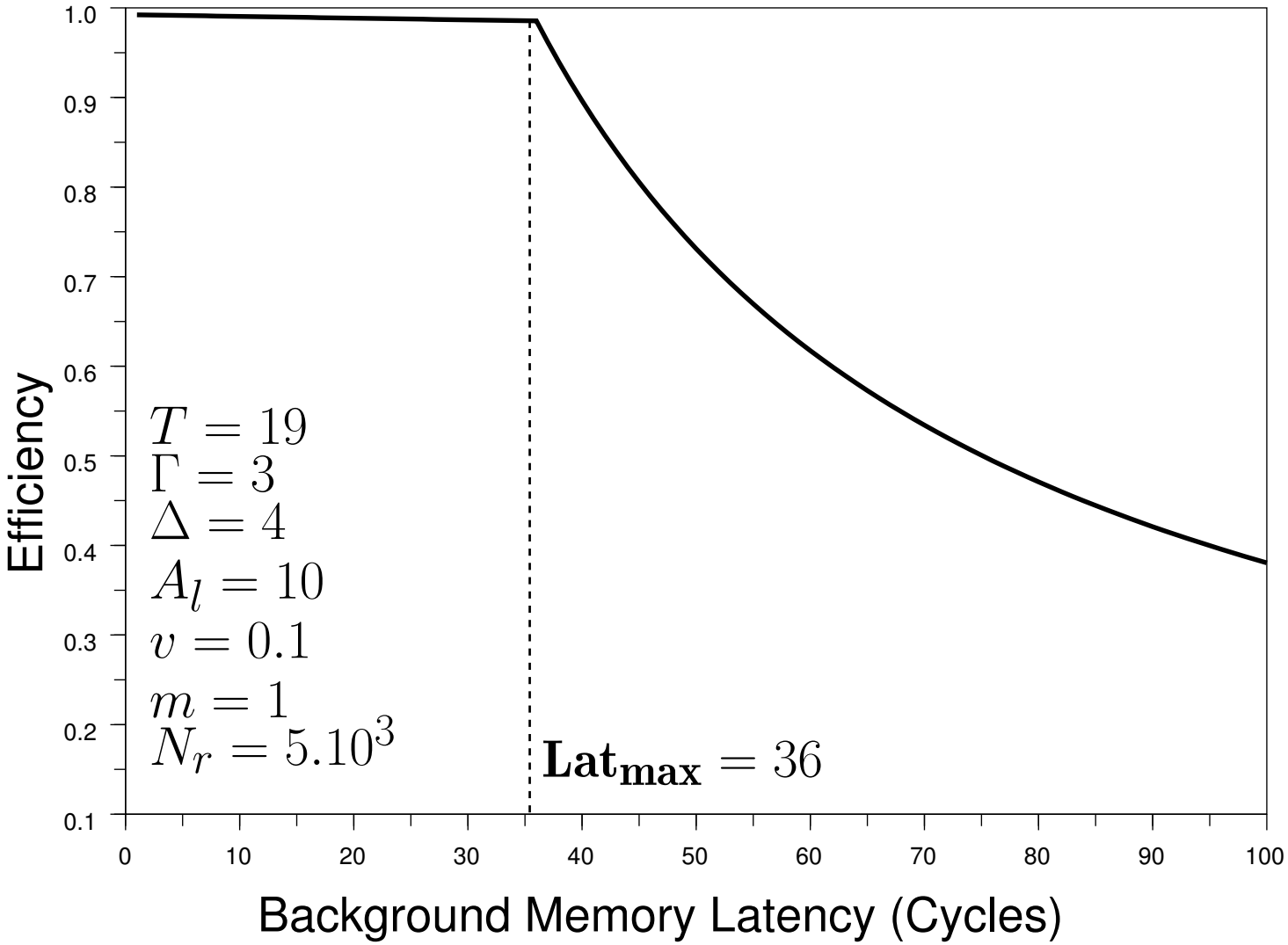
Comportement du Cache nD-AP



Comportement du Cache nD-AP



Modèle de performance

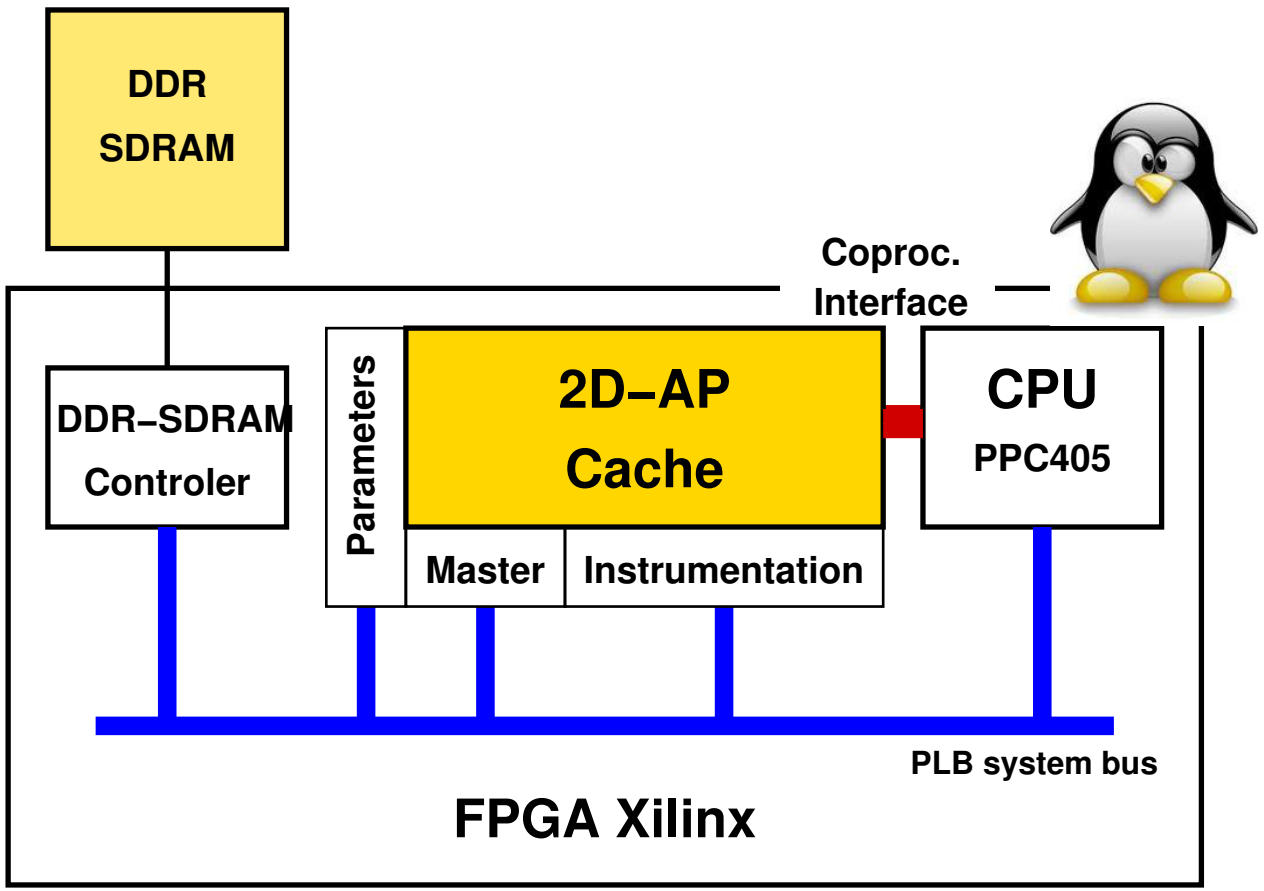


Conditions: paramètres fixes, estimateur au premier ordre

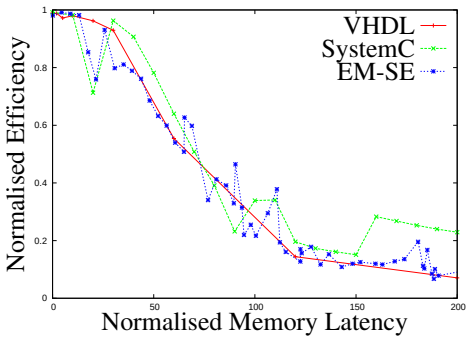
Plateforme d'émulation

La stratégie est **validée** par **simulation** et sur un **prototype** FPGA.

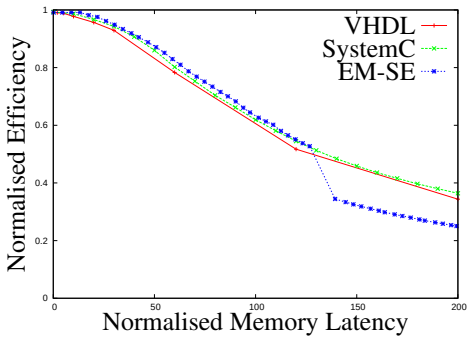
☞ Effets d'un environnement plus réaliste (latence mémoire variable, période des références variables, etc . . .).



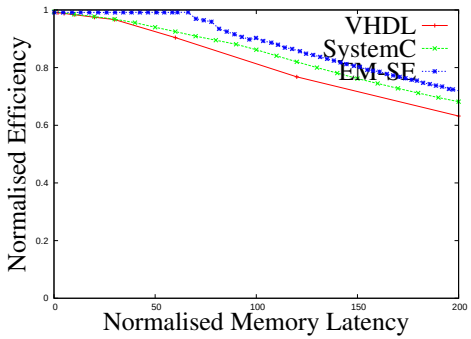
Quelques résultats



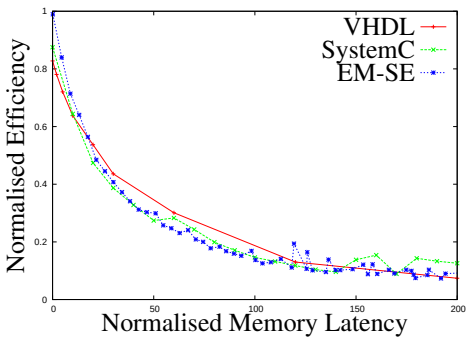
*Rotation - 1st line of 32*32 tiles*



*Rotation - 1st line of 48*48 tiles*



2D Ray casting, interleaved



*Rotation - 2 lines of 32*32 tiles*

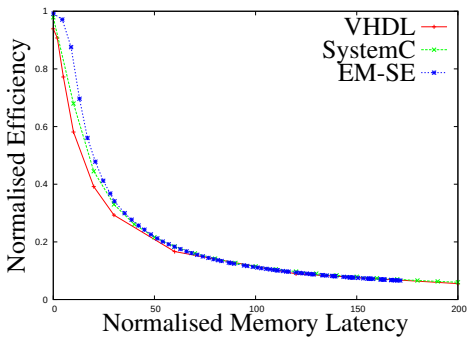
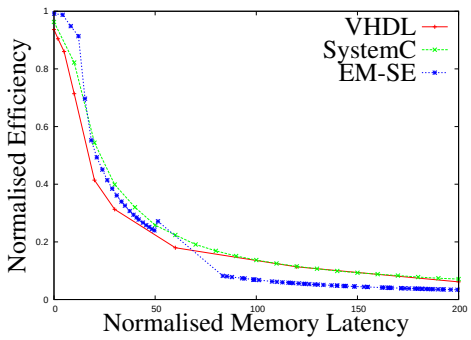


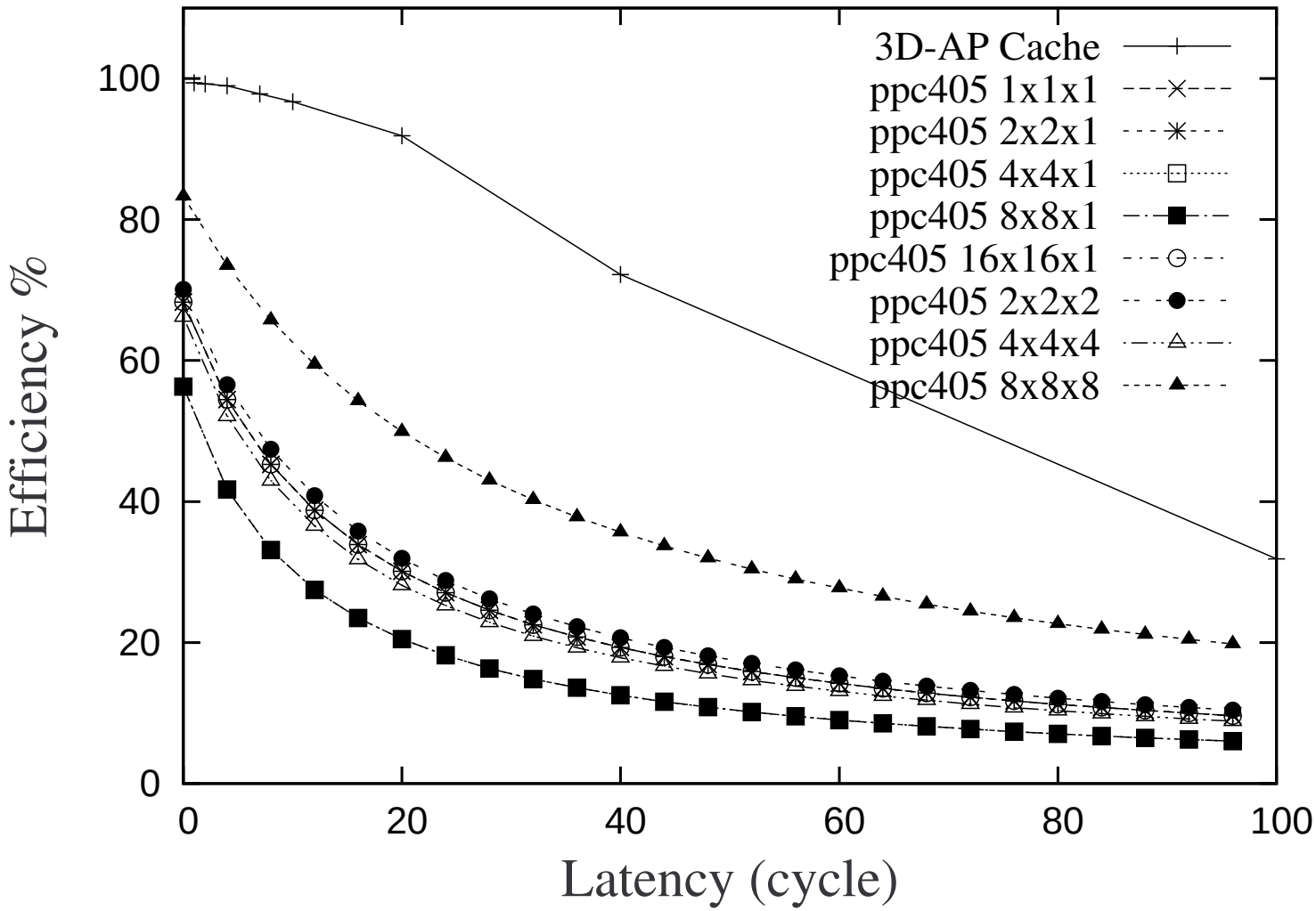
Image rendering



Lip contour finding

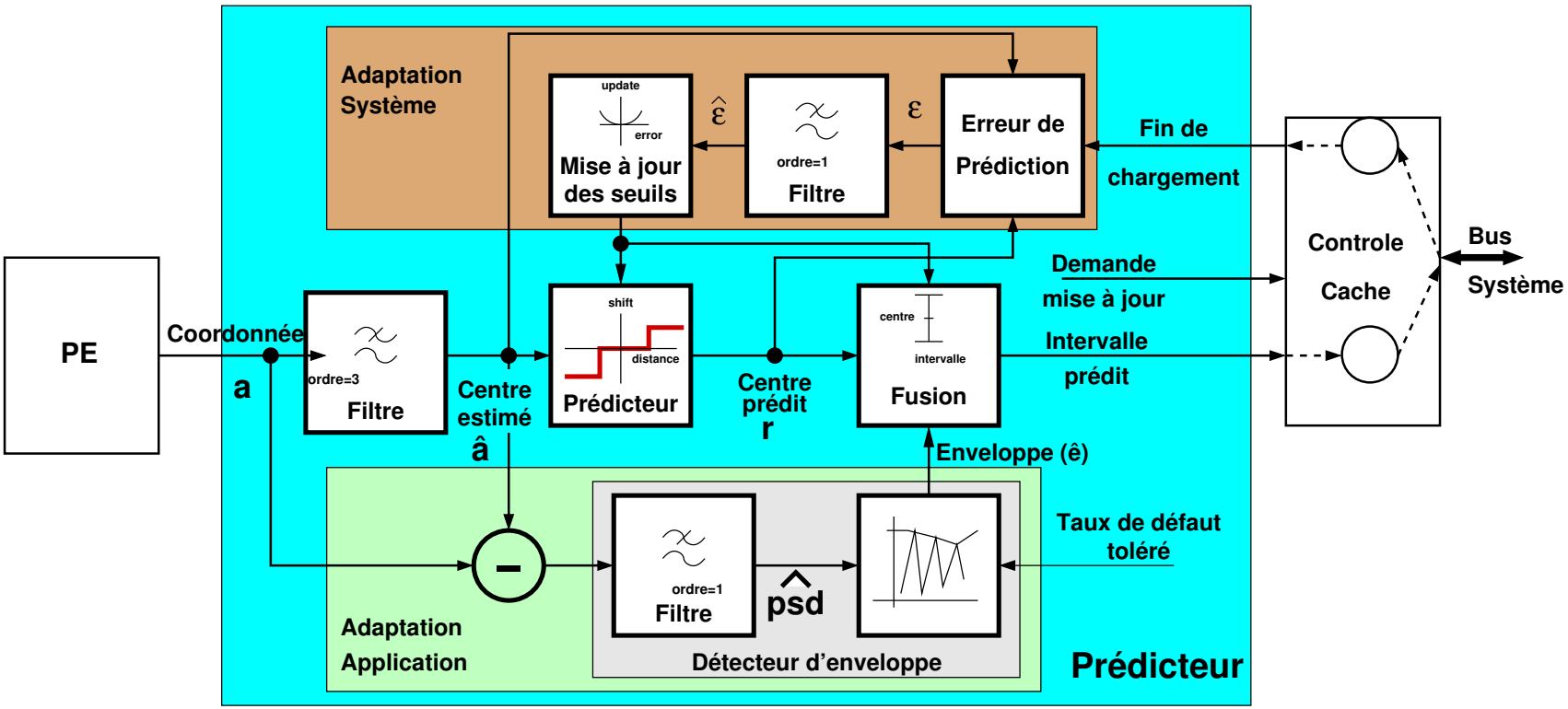
Les mesures sur plateforme (EM-SE) sont calibrées par rapport au mesures en simulation (VHDL & SystemC).

Comparaison avec un cache standard (Ray casting 3D)

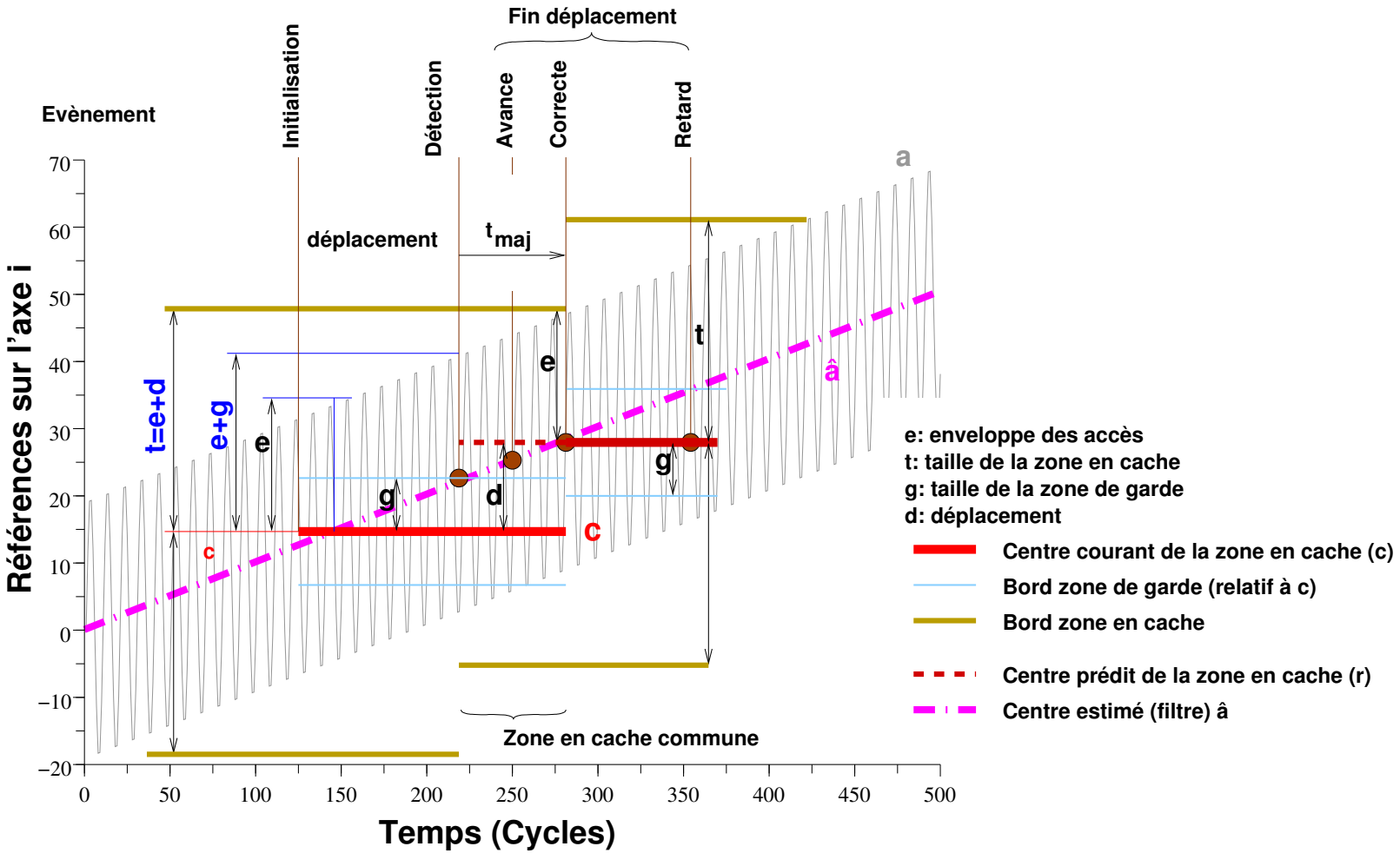


4KB 3D-AP Cache v.s. 16KB ppc405 cache for different memory layout

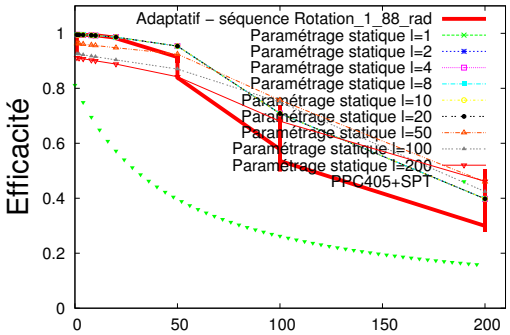
Précharge adaptative



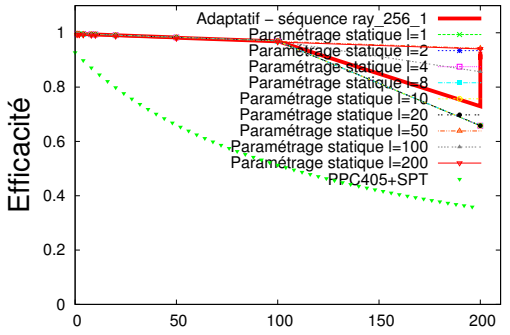
Adaptation au système



Quelques résultats



*Rotation - 1st line of 32*32 tiles*



2D Ray casting

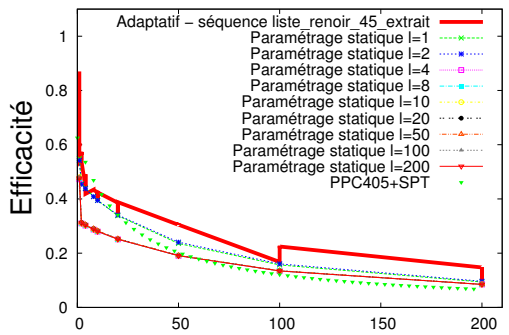
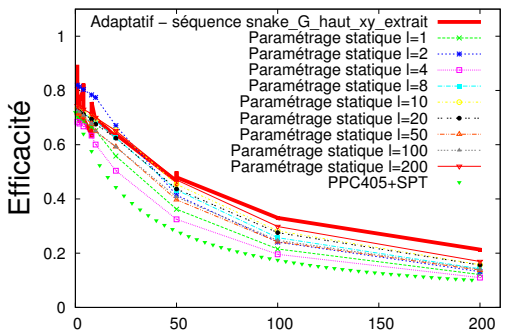


Image rendering (1)



Lip contour finding

Pour chaque valeur de latence, la performance de l'auto-adaptation (-) est comparée au prédicteur non-adaptatif paramétré statiquement par le résultat de l'auto-paramétrage.

Bilan de l'autoadaptation

Analyse:

- Le réglage des prédicteur est simplifié

  La fréquence de coupure devient le seul paramètre

- Performance "semblable" à celle obtenue manuellement

  Et même meilleure pour quelques séquences

- Le coût d'une mise à jour est anisotropique

  Dépend du rangement des données en mémoire!

- Décrochage au-delà d'une latence seuil

  L'objectif devient la minimisation du temps d'attente et non plus de pré-charger à temps

Pistes:

- Détecter le changement d'objectif
- Accorder l'adaptation entre les prédicteurs de chaque dimension

Interaction avec la compilation

- Utiliser le cache depuis le logiciel ⇨ API
  - Accéder au cache avec les indexes, par l'interface coprocesseur
  - De nombreux programmes "codent" la transformation des indexes en adresses
- Transformer les applications pour produire de la **localité** temporelle et spatiale en **indexes**
  - ⚠ Et non plus en *adresse* !
- Prendre en compte le rangement des données
- Calculer la fréquence de coupure à la compilation

  Boucles à bornes statiques v.s variables
- Et la compilation dynamique?

Plan

- Le memory wall
- Le prefetch
- Le Cache nD-AP
- Conclusion & perspective

Conclusion & perspective

Nous avons:

- Défini et validé une stratégie de pré-chargement pour le traitement d'image
- Réalisé des modèles de simulation et prototypes FPGA
- Expérimenté l'auto-adaptation

Perspective:

- Améliorer l'auto-adaptation
- Concevoir des modèles de séquence plus complexes
- Intégrer ce mécanisme dans une infrastructure SW
- Rétro-action avec la compilation

Persyval HPES

Le CacheND-AP : pré-chargement adaptatif dans les tableaux

S. Mancini

Plan Détaillé

- Le memory wall
  - Motivations
  - Problématique du “Memory Wall”
  - Consommation d’énergie due aux mémoires
  - Comment “passer le mur” mémoire?
  - Une vision globale
- Le prefetch
  - Le prefetch
  - Prefetch usuels
  - Limitations
- Le Cache nD-AP
  - Positionnement
  - Objectifs
  - Concept du CachenD
  - Modèle de séquence
  - Architecture du Cache nD-AP
  - Mécanisme de prédiction et gestion mémoire
  - Comportement du Cache nD-AP
  - Modèle de performance
  - Plateforme d’émulation
  - Quelques résultats
  - Comparaison avec un cache standard (Ray casting 3D)
  - Précharge adaptative
  - Adaptation au système
  - Quelques résultats
  - Bilan de l’autoadaptation
  - Interaction avec la compilation
- Conclusion & perspective
  - Conclusion & perspective