Etude d'un cache 2D adaptatif et prédictif pour le traitement d'image

Stéphane Mancini, Nicolas Gac, Michel Desvignes

Etude d'un cache 2D adaptatif et prédictif pour le traitement d'image

S. Mancini, N. Gac, M. Desvignes
Laboratoire des Images et des Signaux, France, Grenoble
{stephane.mancini, nicolas.gac, michel.desvignes}@lis.inpg.fr
Tel : 04 76 57 43 58, Fax : 04 76 57 47 90

***Abstract*— La nécessité de définir des infrastructures matérielles capables de supporter la grande diversité et complexité des algorithmes de traitement d'image, pour des tâches telles que la vision, se fait toujours plus forte. Aussi nous proposons une hiérachie mémoire bidimensionnelle susceptible de compenser, pour une large classe d'applications, la latence de mémoires de type SDRAM. En effet, ces mémoires deviennent un goulot d'étranglement lorsqu'y sont stockées des images sur lesquelles sont fait des accès aléatoires. Le cache 2D adaptatif et prédictif (cache 2D-AP) proposé repose sur une analyse statistique des coordonnées des pixels utilisés au cours du traitement. Cette analyse nous permet de déterminer une taille optimale de la zone en cache et de prédire les prochains accès à l'image. Comparativement à des caches standards comme ceux des TM32 ou PowerPC405, cette stratégie améliore la latence des accès mémoire jusqu'à un facteur 6 et la taille du cache jusqu'à un facteur 40 pour de nombreuses applications. Ce travail illustre la possibilité d'utiliser les techniques du traitement du signal pour le contrôle des architectures complexes.**

I. Introduction

La conception de systèmes de traitement de l'image se heurte souvent à la mise au point de structures de données complexes et d'une hiérachie mémoire adaptée pour compenser les contraintes technologiques des mémoires. Elle serait facilitée par l'utilisation de mécanismes automatiques ou semi-automatiques plus performants que les caches classiques, prévus pour une organisation linéaire des données. Plus les algorithmes deviennent complexesplus leurs séquences de traitement deviennent imprédictibles du fait des nombreux choix opérés pendant leur déroulement. En conséquence il devient de plus en plus difficile de mettre au point des hiérarchies mémoires qui exploitent les particularités de l'algorithme car les séquences d'accès aux pixels des images deviennent de plus en plus aléatoires. Ainsi il devient intéressant de proposer des modules matériels susceptibles de satisfaire une large classe d'application et de les insérer dans les flots de conception de systèmes complexes de traitement d'image.

Le réduction du taux de défaut de cache des caches de micro-processeurs classiques est un enjeu majeur [6] et de nombreuses stratégies ont été mises au point pour optimiser différents paramètres tels que le débit ou la consommation [1]. Cependant les caches conventionnels restent assez inefficaces pour les tâches de traitement d' images 2D du fait de leur structure fondamentalement linéaire [4]. En effet, la localité en adresse, exploitée par les caches standards, est incompatible avec la localité 2D qui produit des séquences d'adresses plus ou moins aléatoires.

Pour résoudre ce problème dans des applications de type multimédia ou codage vidéo, quelques auteurs proposent des tentatives de prédiction[2] et des caches bidimensionnels, comme celui du "video instruction processor" [7]. Cependant, ces caches sont essentiellement des extensions des caches conventionnels au cas bidimensionnel et les mécanismes de prédiction reposent principalement sur une analyse des adresses émises par l'unité de calcul.

La mise au point de caches capables de s'adapter aux applications devient un enjeu

important. En effet, cela permet de libérer le concepteur du réglage de leurs paramètres ou, au contraire, de l'optimisation de l'application à une structure mémoire donnée. De telles architectures proposent des ressources fixes dont les mécanismes d'allocation sont réglés dynamiquement par une analyse à la volée de l'application [8]. Le réglage fin des hiérarchies mémoire peut aussi se faire par un analyse à priori de l'application [5], mais, à nouveau, ces stratégies reposent sur un modèle 1D des accès mémoire.

Le cache adaptatif et prédictif (cache 2D-AP) proposé permet d'améliorer les temps d'accès mémoire pour des applications dont les algorithmes ont une forte localité spatiale 2D. Il repose sur un algorithme prédictif de suivi de trajectoire à hypothèse de vitesse constante. L'analyse statistique des accès pixel se fait par simple filtrage récursif d'ordre 1, à coefficient en puissance de 2 pour réduire le coût du contrôle. Des démarches classiques d'adéquation algorithme architecture permettent d'améliorer son efficacité en favorisant cette localité par le traitement de paquets de pixels.

Dans la première partie de cet article, nous trouverons les critères de performance auxquels le cache 2D-AP tente de répondre. Suivent la description de l'algorithme de poursuite dans sa version simple et améliorée et une analyse de performance du cache 2D-AP, comparé à deux caches "n-way set associative". Pour terminer, avant la conclusion, nous aborderons l'implémentation, l'utilisation de ce cache dans un SoC (System on Chip) et la construction d'un cache 2D-AP hiérarchique.

II. Principe du cache Adaptatif et Prédictif

A. Principe général et contraintes

Selon le principe classique du cache, l'objectif du cache 2D-AP est de maintenir une copie locale à l'unité de traitement d'une zone d'une image stockée dans une mémoire plus lente et, souvent, de très grande capacité, comme une SDRAM (interne ou externe), comme illustré figure 1. L'idée originale du cache 2D-AP est de calculer la position de cette zone dans l'image originale, ainsi que sa taille, par un algorithme de poursuite des coordonnées des pixels émises par l'unité de calcul. Cette stratégie exploite la localité spatiale 2D de l'algorithme de traitement et se montre très efficace pour des applications de vision telles que la recherche de courbe dans une image, le calcul d'intégrales curvilignes ou encore pour des applications de type reconstruction 2D en imagerie médicale.

L'exploitation directe de la nature 2D des traitements permet de résoudre les difficultés posées par l'inadaptation des caches classiques au traitement d'image [4]. En effet, les lignes des caches standard sont constituées de mots consécutifs en mémoire. La progression verticale d'un traitement produit donc systématiquement un défaut de cache, tout comme un déplacement horizontal à la fin d'un segment de longueur égale à la ligne de cache. Pour résoudre ce problème, les traitements sont souvent réalisés sur des blocs de l'image dont les tailles et positions sont connues et systématiques mais ces stratégies influencent le type de traitement et peuvent ne pas convenir à de nombreuses applications.

L'objectif du cache 2D-AP est de proposer un mécanisme système semi-automatique utilisable pour une large classe d'application et susceptible de dépasser les limitations des caches classiques. Ses caractéristiques sont les suivantes:

a) Il est plus rapide que les caches conventionnels
b) Il utilise moins de mémoire
c) La taille de la zone en cache et sa vitesse de déplacement est adatée à l'application
d) La complexité du contrôle est indépen-

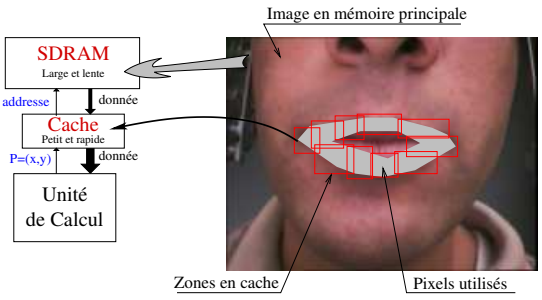

Fig. 1. Principe du cache 2D-AP

dante de la taille mémoire du cache
e) La technologie cible peut être un ASIC ou un FPGA

Les deux derniers points différencient fortement le cache 2D-AP d'un cache classique qui nécessite souvent la mise en place d'une mémoire associative dont les performances sont directement dépendantes du nombre de lignes par groupe associatif. Dans le cache 2D-AP, le test de défaut de cache est un test d'appartenance à la zone en cache des coordonées du pixels requis et a un coût quasi-indépendant de la taille de cette zone.

B. L'algorithme du cache 2D-AP

La prédiction de la zone en cache permet de réduire le taux de défaut de cache lorsque les accès pixels suivant se déplacent sur l'image. Le centre et la taille de la zone sont déterminés par le calcul de la moyenne et du pseudo écart type (PSD=pseudo standard deviation) des accès pixels. La moyenne et le PSD sont calculés à l'aide de filtres passe-bas récursifs du premier ordre. En supposant une distribution uniforme des accès pixels autour de la moyenne, nous pouvons estimer que, pour une courte durée, ils sont dans un rectangle centré sur la moyenne et dont les demi-largeurs valent deux fois le PSD sur chaque axe. La zone ainsi calculée est mise à jour lorsque la moyenne varie.

Un mécanisme de prédiction permet d'anticiper la position du centre du cache lorsque les accès pixels suivent un chemin complexe dans l'image et réduit ainsi le coût de la latence des accès mémoire. La mise à jour ne se produit que lorsque la moyenne a suffisament variée et la nouvelle zone est centrée sur une anticipation de la moyenne,

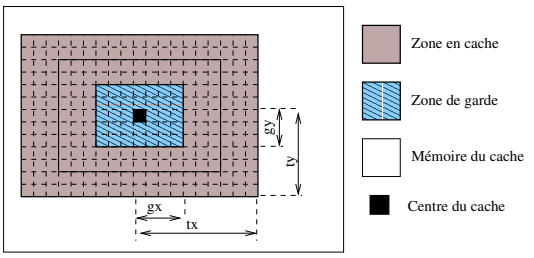

Fig. 2. Zones du cache 2D-AP

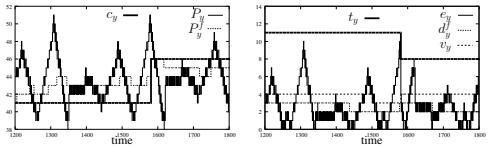

Fig. 3. Etat du cache 2D-AP en fonction du temps (composante y)

supposée se déplacer à vitesse constante. Ainsi, une zone de garde est définie autour du centre de la zone en cache et la mise à jour se fait lorsque la moyenne calculée sort de cette zone de garde, dans la direction de sortie, comme illustrée figure 2.

Pour réduire le coût matériel du contrôle et obtenir de bonnes performances les filtres passes bas pour le calcul des moyennes sont réalisés à l'aide de simples additions et décalages. Ce sont des filtres RII du premier ordre dont l'équation est $s_n = ce_n + (1-c)s_{n-1}$, e signal d'entrée et s signal filtré, et nous notons $s = \mathrm{f}^c(e)$. Le paramètre $1-c$ correspond à la constante de temps du filtre RC équivalent. Lorsque les suites s et e sont représentées en virgule fixe et c une puissance de $\frac{1}{2}$, nous ne réalisons plus que des additions et décalages.

C. Les paramètres du cache

Les principaux paramètres du cache sont les fréquences de coupure nécessaires aux calculs de la moyenne et du PSD, $a = (a_x, a_y)$ et $b = (b_x, b_y)$, ainsi que les vitesses de déplacement du mécanisme de prédiction, $\alpha = (\alpha_x, \alpha_y)$. Nous avons aussi les paramètres de réglage des dimensions des différentes zones du cache : les facteurs de proportionalité de la zone de garde, $\gamma = (\gamma_x, \gamma_y)$, et de la taille du cache, $\tau = (\tau_x, \tau_y)$.

L'état du cache, à l'instant n comme illustré figure 3, est déterminé par les variables suivantes :

- $P = (x_n, y_n)$, coordonnées du point utilisé par l'UT
- $P_f = \mathrm{f}^a(P) = (x_{f_n}, y_{f_n})$, moyenne des points utilisés
- $d = |P - P^f|$, PSD courant (distance du point à la moyenne calculée)
- $d_f = \mathrm{f}^b(d)$, PSD moyen
- c, centre courant du cache

- v, PSD courant du cache. Il définit:
  - $g = \gamma v$, taille de la zone de garde
  - $t = \tau v$, taille de la zone en cache

  Les paramètres v, g et t sont remis à jour à chaque déplacement du cache.
- $e = P - c$, distance du point courant au centre du cache

Le bloc mémoire du cache 2D-AP a une taille fixe et définit la valeur maximum de t. A chaque déplacement, seule la nouvelle partie de l'image en cache est chargée, celle étant à l'extérieur est écrasée par les nouvelles données. En effet, la position des pixels de la zone dans la mémoire est calculée modulo la taille maximum. Un défaut de cache se produit lorsqu'un pixel est hors de la zone en cache ($e > t$) et seul le pixel en défaut est chargé depuis la mémoire externe.

D. Effet des paramètres et mécanismes de stabilisation

Pour compenser la latence de la mémoire, α doit être assez grand et les fréquences de coupures assez basses. Cela permet d'obtenir des déplacements du cache importants et ainsi de réduire le coût du chargement des données, particulièrement dans le cas où l'on se déplace horizontalement.

Les fréquences de coupures a et b correspondent grosso-modo au nombre de points sur lesquels sont fait le calcul de la moyenne. Les contraintes imposées par le calcul en virgule fixe limitent la fréquence de coupure la plus basse, donc le nombre de points pris en compte, mais cela est compensé par un facteur de sous-échantillonnage qui permet d'abaisser encore la fréquence de coupure lorsque les signaux sont très basse fréquence.

Le paramètre α est une estimation de la vitesse de déplacement et permet de compenser le retard de phase dû aux filtres passe-bas.

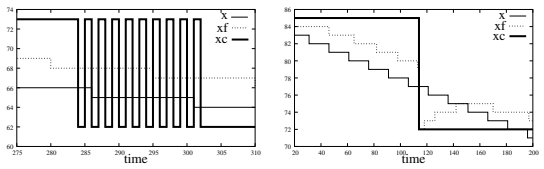

Fig. 4. Poursuite a vitesse $\alpha > 2\gamma$ sans et avec stabilisation

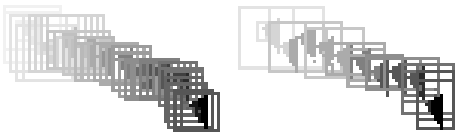

Fig. 5. Poursuite pour $\alpha = 1$ et $\alpha = 3$ (les niveaux de gris représentent le temps). Les triangles sont les points utilisés par l'algorithme, les rectangles les bords des zones en cache.

Il a une grande influence sur la stabilité de l'algorithme et nécessite l'adjonction d'un mécanisme de stabilisation. En effet, si α est supérieur à 2γ, le processus de suivi devient instable car lors d'un déplacement, le point P_f est dépassé et n'est pas dans la nouvelle zone de garde, ce qui produit un nouveau déplacement en sens inverse. Pour obtenir de grandes vitesses de déplacement et un suivi stable, le point P_f est forcé à la valeur du nouveau centre du cache, ainsi qu'illustré figure 4, et une contrainte de continuité limite le déplacement. Stabilisé, l'algorihme de prédiction est efficace et permet de réduire le nombre de déplacements (c.f. figure 5).

III. Comparaison avec un cache standard

Un simulateur au cycle près permet d'évaluer l'efficacité de l'algorithme et de régler le jeu de paramètres pour comparer le cache 2D-AP à un cache standard. Les métriques extraites (temps de chargement pour les déplacements en x ou y, défaut de cache, largeur maximale atteinte par t, temps d'accès, etc ...) nous aident à déterminer les paramètres de cache pour une application déterminée. Ici, nous nous intéressons à deux étapes d'un algorithme de suivi de lèvres et à un calcul de reconstruction 2D en imagerie TEP.

Le premier calcul du suivi de lèvres [3] consiste à chercher des courbes de maximun (ou minimum) de niveau (ou gradient) par la méthode du "jumping snake" : à partir d'un point du "snake", les intégrales curvilignes (ou les flux) d'un faisceau de courbes issues de ce point servent à déterminer le prochain point du "snake" pour itérer l'algorihme. Le second calcul consiste à déterminer le flux de gradient

sur des courbes de Bézier pour déterminer celle qui est la plus proche de la forme cherchée. Pour améliorer la localité temporelle et spatiale dans ce dernier cas, les courbes sont traitées par groupes.

La reconstruction 2D (Retro2D) consiste à calculer une intégrale curviligne selon une sinusoïde sur l'image issue du capteur pour chaque point de l'image reconstruite. Les points reconstruits sont regroupés en paquets pour améliorer la localité temporelle et spatiale. Les mesures sont effectuées sur le pire cas qui est la sinusoïde d'amplitude maximale.

L'efficacité du cache 2D-AP est mesurée par le ratio $t = \frac{t_{\text{std}} - t_{\text{ideal}}}{t_{\text{2D-AP}} - t_{\text{ideal}}}$, avec t_{std} le temps de calcul avec un cache standard et t_{ideal} le temps pour une mémoire idéale, sans latence. Si $r > 1$, le cache 2D est plus efficace que le cache standard. Nous mesurons également la quantité de mémoire utilisée et son ratio avec celle du cache standard, $m = \frac{\text{taille 2D-AP}}{\text{taille std}}$. Le cache utilise moins de mémoire lorsque $m > 1$.

Les caches standards ont les paramètres suivant :

- TM32 cache
  16K, 8 way set-associative, LRU, 256 lignes de 16 mots (32bit)
- PowerPC 405 cache
  16K, 2 way set-associative, LRU, 512 lignes de 8 mots (32bit)

Les performances du tableau I sont mesurées avec l'hypothèse d'un bus système de latence 10 cycles et un débit de 2 mots par cycles. Les paramètres du cache sont adaptés à chaque application et permettent un compromis entre le temps et la surface. Certains paramètres

		Snake	Bezier	Retro2D
Sans cache	t	35	15	163
2D-AP	t	1	1	1
	m	1(980)	1 (840)	1 (1280)
TM32	t	4.3	1.7	7.8
	m	16.7	19.5	12.8
PPC	t	2.9	1.2	6.1
	m	16	19.5	12.8

TABLE I

Comparaison de performance entre le cache 2D-AP et des caches standards : le cache 2D-AP est toujours plus efficace ($t > 1$ et $m > 1$). Entre parenthèses la taille du cache.

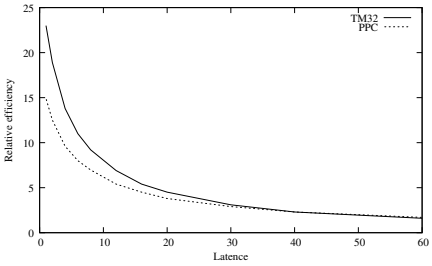

Fig. 6. t en fonction de la latence de la mémoire

permettent un gain de mémoire m de l'ordre de 40, pour un ratio t proche de 1. Le graphe 6 nous montre l'évolution de t en fonction de la latence de la mémoire (et du bus système), dans le cas de l'application Retro2D.

IV. Implémentation et utilisation

A. Implémentation

Le cache 2D-AP a été modélisé en VHDL et tous ses paramètres sont réglables statiquement ou dynamiquement. Le tableau II donne les performances en surface et taille des blocs critiques du cache (excepté la gestion du bus) pour une application comme la Retro2D. Un point intéressant est que la taille du contrôle est quasiment indépendante de la taille de la mémoire du cache. Ces résultats peuvent être améliorés si on considère les paramètres α, γ, τ et les fréquences de coupures constants.

B. Intégration dans un SoC

Le cache 2D-AP est conçu pour être intégré dans un SoC et il est maître sur un bus système, l'unité de calcul pouvant être une IP ou un processeur. Dans ce cas, le processeur doit produire des coordonnées (x, y) par construction d'adresses adéquates (par exemple une concaténation de x et y sur les poids faibles). Le cache 2D-AP peut être directement

Technologie	Surface	Fréquence
AMS 0.35	3000 cellules	120 MHz
Altera Stratix	950 LC	100 MHz
Xilinx V2-Pro	1000 LUT	80 MHz

TABLE II

Synthèse du contrôle du cache pour une application de type Retro2D

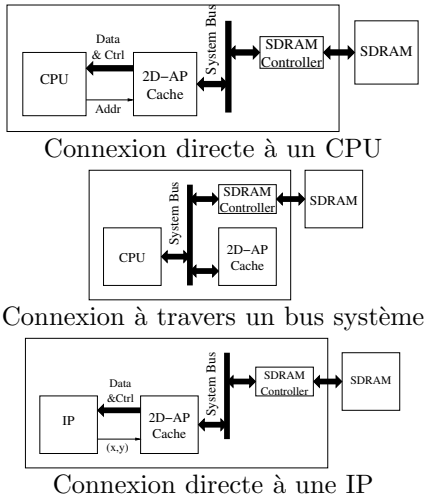

Fig. 7. Insertion d'un cache 2D-AP dans un SoC

lié au processeur ou accessible à travers un bus système, ainsi qu'illustré figure 7. Il est alors possible de construire un environnement de programmation adapté à l'utilisation du cache 2D-AP, à travers des API dédiées.

A l'opposé d'une mémoire embarquée, comme une "scratch-pad memory", le cache 2D-AP libère le concepteur d'un réglage fin des accès mémoire. En effet, une stratégie efficace consiste à construire l'application pour favoriser la localité spatiale 2D et temporelle pour ensuite régler les fréquences de coupure du cache et les autres paramètres. Le concepteur n'a alors plus besoin de régler la mise à jour de la mémoire embarquée.

V. Conclusion

Le cache 2D-AP est toujours plus rapide et consomme moins de mémoire qu'un cache classique pour les applications testées. Le logiciel de simulation nous permet d'interpréter l'effet des différents paramètres du cache et de le régler pour une application donnée. Des pistes intéressantes seraient leur calcul automatique par une pré-analyse automatique des accès pixel de l'application et il semble possible d'améliorer la prédiction par une estimation dynamique de la vitesse de déplacement du cache.

Pour améliorer les débits mémoire et favoriser les traitements parallèles sur les images, il semble envisageable de construire une hiéarchie de cache 2D-AP. Chaque cache de niveau n mettant à jour sa zone à partir d'un cache de niveau $n+1$ dont la zone est calculée à partir des accès de tous les caches qui lui sont connectés.

Le cache 2D-AP est réalisé et sera inséré dans des plateformes SOPC (FPGA) pour la validation d'applications de traitement d'image temps réel. Nous aurons ainsi des informations permettant de valider les simulations effectuées dans un environnement plus complexe que le simple modèle de bus utilisé.

Plus les traitements d'image seront complexes et les séquences dépendantes des traitements, plus nous aurons besoin de mécanismes systèmes efficaces et adaptés à ces traitements. Le cache 2D-AP est une proposition pour faciliter la conception de plates-formes de traitement d'image par la construction d'une hiérarchie mémoire adaptée aux traitements 2D. La statégie adoptée se révèle très efficace et montre le potentiel d'utilisation des techniques issues du traitement de signal pour le contrôle des systèmes embarqués.

References

[1] Y.-J. Chang, S.-J. Ruan, and F. Lai. Design and analysis of low-power cache using two-level filter scheme. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(4):568 – 580, 2003.

[2] R. Cucchiara, M. Piccardi, and A. Prati. Improving data prefetching efficacy in multimedia applications. *Multimedia Tools and Applications*, 20(3):159–178, 2003. Kluwer Academic Press.

[3] N. Eveno, A. Caplier, and P-Y Coulon. Automatic and accurate lip tracking. *Circuits and Systems for Video technology, IEEE Transactions on*, 2004. To appear in may 2004.

[4] D. Kim, R. Managuli, and Y. Kim. Data cache and direct memory access in programming mediaprocessors. *IEEE Micro*, 21(4):33–42, 2001.

[5] P. R. Panda, N. D. Dutt, A. Nicolau, F. Catthoor, A. Vandecappelle, E. Brockmeyer, C. Kulkarni, , and E. de Greef. Data memory organization and optimizations in application-specific systems. *IEEE Design and Test of Computers (D & T)*, 18(3), May/June 2001.

[6] D.A. Patterson and J.L. Hennessy. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, San Francisco, 2nd ed. edition, 1996.

[7] W. Raab and al. A 100-gops programmable processor for vehicle vision systems. *Design & Test of Computers, IEEE*, 20(1):8–15, Jan-Feb 2003.

[8] C. Zhang, F. Vahid, and R. Lysecky. A self-tuning cache architecture for embedded systems. In *Design Automation and Test in Europe Conference (DATE)*, February 2004.