Un réseau cellulaire VLSI fonctionnellement asynchrone pour le filtrage morphologique d'images

Functionnally Asynchronous VLSI Cellular Array for Morphological Filtering of Images

par Frédéric ROBIN, Marc RENAUDIN*, Gilles PRIVAT

France Telecom / CNET-Grenoble,
* Telecom Bretagne-Antenne de Grenoble
BP 98, 38243 Meylan Cedex, France
e-mail : frederic.robin@cnet.francetelecom.fr

résumé et mots clés

A travers la présentation de la conception d'un réseau cellulaire VLSI asynchrone à grain fin, il est montré comment la notion d'asynchronisme peut être exploitée à la fois au niveau fonctionnel et au niveau architectural. Une étude conjointe algorithme-architecture a abouti à la conception d'un circuit intégrant 16×16 processeurs élémentaires. Le flot de conception des chemins de données et de contrôle est basé sur une approche « cellules standard » qui combine des portes CMOS et DCVSL (Differential Cascode Voltage Switch Logic). Ce circuit d'environ 800.000 transistors permet de mettre en oeuvre en temps réel des algorithmes itératifs de filtrage morphologique par reconstruction.

circuit VLSI asynchrone, algorithme asynchrone, réseau cellulaire, traitement d'images, morphologie mathématique.

abstract and key words

The design of a fine grain asynchronous VLSI cellular array is presented. It is shown how asynchronism can be exploited at both functional and structural levels. A joint algorithmic-architectural study has led to the fabrication of an integrated circuit including 16×16 processing elements.The data and control paths are designed using a standard-cell approach, combining CMOS and DCVSL (Differential Cascode Voltage Switch Logic) gates. The 800,000 transistor circuit enables real time morphological filtering of images.

asynchronous VLSI circuit, asynchronous algorithm, cellular array, image processing, mathematical morphology.

1. introduction

Le modèle de calcul cellulaire itératif a été largement utilisé dans des applications de traitement d'images. Son principe est d'effectuer des traitements globaux par l'intermédiaire de la propagation itérative de dépendances purement locales entre les variables attachées à chaque pixel. Une génération passée de réseaux de processeurs synchrones à grain fin fonctionnant en mode SIMD (Single Instruction Multiple Data), comme MPP [1] et CLIP [2], a précisément été l'application architecturale directe de ce modèle. Les nouvelles machines massivement parallèles MIMD basées sur des processeurs RISC, si elles sont au contraire asynchrones et à gros grain, en permettent toutefois l'émulation par des mécanismes de coordination globale dans un mode de fonctionnement SPMD (Single Program Multiple Data). Dans la suite, un modèle d'architecture parallèle asynchrone à grain fin, qui se démarque de ces deux approches tout en retenant leurs avantages respec-

tifs, est présenté. Il s'agit d'étudier des architectures qui peuvent exploiter le concept d'*asynchronisme fonctionnel* [3][4]. Il est montré que les opérateurs morphologiques itératifs peuvent tirer parti d'un tel modèle de calcul parallèle asynchrone à la fois au niveau algorithmique et au niveau architectural. Enfin la méthode de conception utilisée pour l'implémentation d'un réseau VLSI asynchrone spécifique est présentée.

2. relaxation asynchrone d'opérateurs morphologiques

2.1. asynchronisme fonctionnel

Le modèle cellulaire itératif permet de réaliser des traitements globaux à partir de la propagation de dépendances locales. La forme canonique d'exécution parallèle d'opérateurs itératifs correspond à un mode de mise à jour globalement synchrone : l'ensemble des variables de l'algorithme est mis à jour à chaque étape, à partir des valeurs calculées à l'étape précédente. Il existe aussi des modes de mise à jour séquentiels dits récursifs, où la nouvelle valeur d'une variable est utilisée pour la mise à jour d'autres variables dans la même itération, correspondant par exemple au balayage par lignes d'une image. Ces modes récursifs améliorent en général la vitesse de convergence, mais sont séquentiels et anisotropes par définition.

Des modes de mises à jour asynchrones, qui allient les avantages des deux modes précédents, ont été étudiés depuis longtemps pour des algorithmes de relaxation [5][6][7]. L'idée est d'autoriser à chaque itération la mise à jour de seulement un sous-ensemble aléatoire de variables à partir de valeurs éventuellement « retardées », c'est-à-dire antérieures à l'itération précédente. Ce mode correspond donc à un ordre de mise à jour moins contraint. En fait, l'algorithme peut alors s'écrire sous la forme d'un ensemble de processus non synchronisés à communications locales, qui peuvent s'exécuter à des vitesses indépendantes. Plus précisément, un nouveau calcul peut être initié localement pour la mise à jour d'une variable, quels que soient les nombres d'itérations effectuées par les autres processus. Ce principe définit un asynchronisme fonctionnel, c'est à dire au niveau des dépendances algorithmiques, en contraste avec l'asynchronisme architectural qui fait référence à une structure d'implémentation.

2.2. filtres morphologiques pour la segmentation d'images

Le concept d'asynchronisme fonctionnel est appliqué dans cet article à la segmentation d'images. L'approche basée sur la morphologie mathématique est particulièrement intéressante, car relativement simple, robuste et adaptée à des images quelconques. L'algorithme présenté dans [8] comporte quatre étapes : le pré-traitement, l'extraction d'éléments représentatifs, la décision puis l'estimation de qualité. L'étape de pré-traitement est une phase de simplification dont le but est de générer des régions dont les intensités sont quasi-constantes. Elle est implémentée avec des filtres morphologiques par reconstruction [9], qui préservent l'information de contour de l'image originale. On s'intéresse à ces filtres non-linéaires dans la suite.

Les opérateurs de base du filtre de simplification sont l'érosion et la dilatation :

Erosion :

$$y_i = \varepsilon_n(x_i) = \mathrm{Min}\{x_{i+k}, k \in M_n\},$$

Dilatation :

$$y_i = \delta_n(x_i) = \mathrm{Max}\{x_{i-k}, k \in M_n\},$$

où l'indice i est le vecteur de coordonnées du pixel x, M_n un ensemble de coordonnées relatives, appelé élément structurant plat, qui spécifie le motif binaire à combiner avec l'image, et n la taille de cet élément structurant. Ce dernier paramètre contrôle le niveau de simplification. Par exemple, si on utilise un élément structurant de grande taille, alors le résultat de la segmentation ne fera apparaître que les plus grandes composantes.

Le processus de reconstruction qui régénère les contours de l'image originale, tout en maintenant le niveau de simplification désiré, est basé sur la dilatation géodésique de taille unitaire [9] :

Dilatation géodésique :

$$y_i = \delta^{(1)}(x_i, r_i) = \mathrm{Min}\{\delta_1(x_i), r_i\}.$$

L'application itérative de dilatations géodésiques unitaires à la suite d'une érosion de taille n définit une ouverture par reconstruction (le signal de référence r_i étant égal à l'image originale) :

Ouverture par reconstruction :

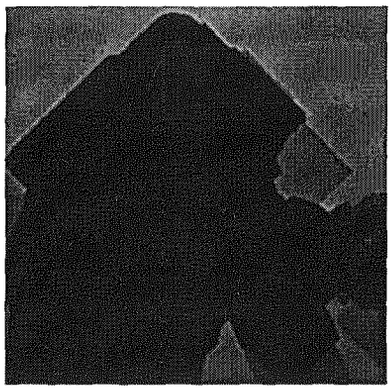
$$y_i = \delta^{(1)}(\ldots\delta^{(1)}(\delta^{(1)}(\varepsilon_n(x_i), x_i))\ldots, x_i) \qquad (1)$$

La fermeture par reconstruction est définie de manière duale, et le filtre « d'ouverture-fermeture par reconstruction », qui est le filtre de simplification effectivement utilisé pour le pré-traitement, consiste à effectuer une ouverture puis une fermeture. La figure 1 illustre l'utilisation de tels opérateurs morphologiques.

2.3. reconstruction parallèle asynchrone

L'application de l'asynchronisme fonctionnel à des algorithmes itératifs de traitement d'images [3] permet d'élargir l'espace de conception d'algorithmes parallèles. On s'intéresse ici à son application aux filtres morphologiques [10]. Ce modèle peut être défini à partir d'une transformation de l'algorithme à itération

a) Image originale



b) Erosion

c) Ouverture par reconstruction

Figure 1. – Exemple de filtre morphologique.

globale en un ensemble de processus à itération locale chacun lié à un pixel, où certaines contraintes sur les indices sont relâchées. La notation globale de l'équation (Equ. 1) est équivalente à (Equ. 2), où r_i est l'image originale.

$$x_i(n) = \mathrm{Min}(\mathrm{Max}\{x_{i-k}(n-1), k \in M_1\}, r_i), \qquad (2)$$

avec $x_i(0) = \varepsilon_n(r_i)$.

Au lieu d'utiliser le même indice d'itération n pour tous les processus-pixels, un indice d'itération locale n_i peut être introduit, conduisant à (Equ. 3), où chaque processus a sa propre vitesse d'évolution.

$$x_i(n_i) = \mathrm{Min}(\mathrm{Max}\{x_{i-k}(n_{i-k}), k \in M_1\}, r_i) \qquad (3)$$

Alors que n_i s'accroît à la même vitesse pour tous les processus dans (Equ. 2), la nouvelle contrainte est seulement que chaque n_i est une fonction croissante du temps.

Tout en étant une approche complètement parallèle, ce mode de mise à jour tire parti des propriétés du mode récursif, car lorsqu'une variable est mise à jour, sa nouvelle valeur peut être utilisée par un processus voisin dès qu'il commencera un nouveau calcul, ce qui augmente donc la probabilité de propager de l'information utile entre les étapes successives.

Des simulations ont confirmé qu'un tel mode asynchrone peut améliorer de manière significative la vitesse de convergence de la reconstruction morphologique, par rapport au mode globalement synchrone [10]. L'approche asynchrone peut non seulement améliorer certaines propriétés algorithmiques, mais aussi procurer de nouvelles opportunités d'exploration du spectre de conception d'architectures parallèles.

3. une architecture cellulaire fonctionnellement asynchrone

La combinaison des asynchronismes fonctionnel et structurel ont été mis en œuvre par l'étude et la conception d'un processeur asynchrone massivement parallèle. Un processeur élémentaire (PE) asynchrone est associé à chaque pixel. Les paragraphes suivants montrent comment l'asynchronisme architectural est mis au service du concept algorithmique, et décrivent l'architecture du réseau de processeurs et ses principales caractéristiques. Ces principes s'appliquent en fait à la classe générale des algorithmes cellulaires itératifs convergeant vers un point fixe.

3.1. l'asynchronisme architectural au service de l'asynchronisme algorithmique

Le modèle de calcul fonctionnellement asynchrone décrit au paragraphe § 2.1 n'est effectif en pratique que s'il existe une dispersion entre les temps de mise à jour des différentes variables. Les calculs effectués par les différents PEs se désynchronisent alors dans le temps. Ceci est la première condition nécessaire qui permet à des variables calculées à des itérations différentes d'être utilisées pour un nouveau calcul local.

La seconde condition nécessaire est que le système de communication entre PEs ne doit pas resynchroniser localement les itérations. Un PE donné doit donc être capable de lire les variables des processeurs voisins sans attendre que ceux-ci aient fini leurs calculs en cours.

L'asynchronisme architectural est la clé de notre implémentation. La première condition est remplie par l'utilisation d'opérateurs asynchrones à détection de fin de calcul, dont le temps de calcul peut varier en fonction des données (*cf.* § 4.4.). La deuxième condition est remplie grâce à l'utilisation d'interfaces de communication inter-PE spécifiques, décrites au paragraphe § 4.3.

3.2. architecture du réseau de processeurs

Le réseau de processeurs élémentaires est organisé suivant une grille bi-dimensionnelle à maille carrée, dans laquelle chaque processeur communique avec ses quatre plus proches voisins (Fig. 2). Chaque PE met à jour sa propre variable d'état de manière non synchronisée par rapport aux voisins. Un réseau de 16×16 processeurs-pixel a été implémenté dans un circuit VLSI, composé d'un cœur de 14×14 PEs et de cellules de bord qui contiennent principalement des registres d'E/S. Le réseau de processeurs échange des données avec l'extérieur par le biais de trois liens sériels par ligne de PEs. Les pixels sont introduits dans le réseau par l'intermédiaire de deux liens d'entrée (un pour les données intermédiaires, un autre pour l'image de référence, *cf.* Equ. 1), selon un mode bit-série de manière à limiter le nombre de plots. Seize lignes de seize pixels sont stockées dans des registres à décalage inclus dans les PEs pour chacune des deux images d'entrée, sous le contrôle de signaux d'horloge dédiés aux E/S. En parallèle, les données du calcul précédent sortent du réseau par l'intermédiaire du troisième lien série. Ce schéma de

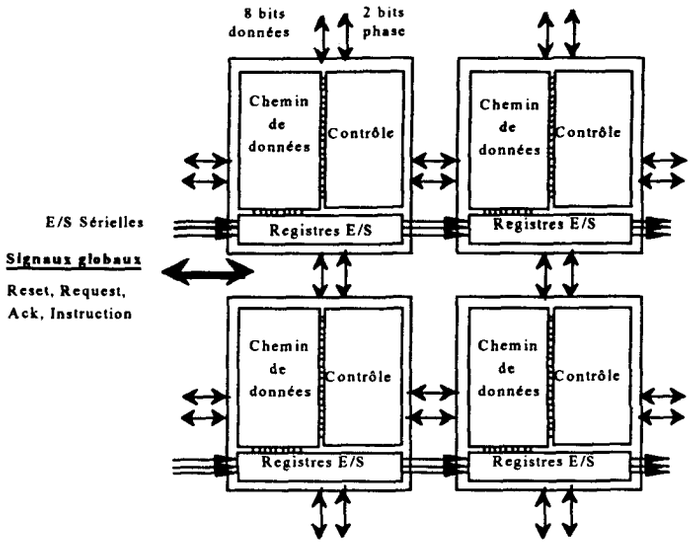


Figure 2. – Architecture du cœur du réseau de processeurs.

pipeline permet aux PEs d'effectuer des calculs pendant que les prochaines données sont introduites et que les résultats précédents sont extraits.

L'instruction de commande peut être initialement stockée dans le réseau à l'aide d'une entrée bit-parallèle qui est distribuée à tous les PEs. Enfin, trois signaux de contrôle sont fournis à l'ensemble des processeurs pour contrôler l'exécution. Le signal « Reset » permet de positionner les PEs dans un état initial déterminé. Le signal « Request » permet au système hôte de lancer un calcul dans le réseau dès que ce dernier a été initialisé avec l'instruction et les pixels d'entrée. Losque le calcul est terminé, le signal « Ack » devient actif et un autre calcul peut alors être initié. De manière à optimiser le temps global de calcul, il faut que le temps d'entrée/sortie des données soit égal au temps de calcul moyen correspondant à l'instruction.

Comme il manque des voisins aux PEs des bords du réseau, ceux-ci ne font pas de calcul mais se chargent simplement de répondre correctement aux requêtes de leurs voisins appartenant au cœur. Ces PEs dégradés sont initialisés comme les autres PEs, puis délivrent toujours la même valeur durant les itérations.

3.3. propriétés architecturales

L'implémentation d'un tel réseau de processeurs asynchrones présente un certain nombre de propriétés intéressantes. La propriété majeure est la capacité d'exécuter un ensemble de calculs en temps moyen, plutôt que sur une base de type « pire cas », grâce à l'utilisation d'un additionneur à propagation de retenue et à détection de fin de calcul (*cf.* § 4.3) et d'un schéma spécifique de communication inter-PE (*cf.* § 4.4). Puisque les PEs n'ont pas à attendre le plus lent d'entre eux à chaque itération, le temps total de calcul est de l'ordre du temps moyen de calcul d'une itération multiplié par le nombre d'itérations. On peut donc clairement tirer parti de la structure itérative de l'algorithme, puisque plus le réseau exécute d'itérations sans aucune synchronisation, plus le temps total de calcul s'approche du cas correspondant au temps moyen. En fait, le temps de calcul est moyenné sur l'ensemble des itérations de l'ouverture ou fermeture par reconstruction. Avec un réseau de processeurs synchrones, une synchronisation globale serait nécessaire à la fin de chaque itération. Au contraire, dans ce réseau asynchrone, une seule synchronisation globale est requise à la fin du calcul de l'ensemble des itérations. Le taux d'accélération théorique de cette implémentation, par rapport à une implémentation synchrone, est donc le produit de deux termes : le temps de calcul du pire cas divisé par le temps de calcul moyen d'une itération, et le taux d'accélération du taux de convergence dû à l'asynchronisme fonctionnel (qui s'exprime comme le nombre total d'itérations nécessaires avec une mise à jour synchrone divisé par le nombre correspondant au mode asynchrone).

Deuxièmement, la puissance consommée peut être améliorée de trois façons. Chaque PE ne consomme que lorsque nécessaire et s'arrête automatiquement à la fin du calcul, sans coût additionnel.

Il n'y a pas de pics de consommation correspondant à des fronts d'horloge, l'activité électrique s'étalant dans le temps, et aucune énergie n'est consommée pour piloter les signaux d'horloge dans tout le circuit. De plus, grâce à la robustesse de l'implémentation asynchrone, la tension d'alimentation peut être diminuée sans modifier le comportement fonctionnel du système. En fonction des applications, la puissance consommée peut donc être ajustée en adaptant la tension d'alimentation, ce qui permet de s'approcher de la consommation optimale.

Troisièmement, on obtient une implémentation parfaitement locale qui n'utilise pas de signal global d'horloge et qui est basée sur un ensemble de blocs auto-séquencés, ce qui accroît la modularité, l'extensibilité et la fiabilité du système. Même pour les registres à décalage synchrones, le schéma de pipeline adopté reste modulaire et extensible, car les signaux de données et d'horloge sont amplifiés dans chaque PE et propagés le long des lignes de processeurs.

La synchronisation globale est effectuée en tenant compte des signaux de fin de calcul de l'ensemble des PEs. De manière à préserver la modularité de la structure, les informations de fin de calcul sont assimilées par lignes, de PE en PE, et finalement combinées pour générer le signal d'acquittement global. La diffusion de l'instruction de commande dans tous les PEs n'est faite qu'une fois pendant la phase d'initialisation, et ne constitue pas un point bloquant.

4. conception d'un processeur élémentaire

4.1. approche « standard-cells »

A partir de l'expérience acquise à travers la conception et la fabrication de plusieurs circuits asynchrones [11][12][13][14], il a été choisi d'utiliser des blocs logiques à précharge implémentés avec la logique DCVS (Differential Cascode Voltage Switch Logic) [15] pour concevoir les chemins de données, et d'utiliser des portes logiques CMOS pour concevoir les parties de contrôle.

De manière à être capables de concevoir rapidement des circuits indépendants de la vitesse basés sur des blocs logiques à précharge, et pour ne pas avoir à concevoir à chaque fois les mêmes cellules, il a été décidé de spécifier et de concevoir une librairie de cellules standard asynchrones. L'idée est de concevoir des fonctions logiques à précharge comme des cellules standard qui peuvent être facilement associées à des portes CMOS standard dans un circuit. Les blocs logiques à précharge ont été conçus comme des « standard cells » en respectant le format des bibliothèques de cellules CMOS disponibles par ailleurs. La technologie CMOS 0.5 μm à trois niveaux de métal du Centre Commun CNET SGS-THOMSON, pour laquelle une librairie de cellules standard est disponible et maintenue à jour, a été adoptée. Cette approche permet de concentrer les moyens de conception sur les seules cellules asynchrones spécifiques, en laissant la tâche de maintenir la librairie de cellules standard CMOS à la fonderie (la référence [16] décrit le flot de conception en détails).

4.2. programmabilité des processeurs élémentaires

Le processeur élémentaire est configuré par un mot d'instruction (Fig. 3), pour permettre le calcul de différents opérateurs morphologiques par le réseau. Un bit indique si on effectue une opération de minimum ou de maximum entre les variables voisines, ce qui correspond respectivement à une érosion et une dilatation. Un autre bit spécifie si on utilise l'opérateur géodésique, qui effectuera en fait l'opération duale avec le pixel de référence à la suite de l'opération normale (*cf.* Equ. 3). Un champ de 5 bits spécifie les valeurs voisines à prendre en compte (le voisinage comprend au plus les 5 valeurs labellées Nord, Est, Sud, Ouest et Local). Un autre champ de 5 bits donne le nombre d'itérations à effectuer localement. Le mot d'instruction est donc équivalent à un petit programme, qui contient des boucles et des exécutions conditionnelles, ce qui correspond à un modèle SPMD puisqu'il n'y a aucune synchronisation entre les processeurs pendant toute l'exécution de l'instruction. L'ouverture par reconstruction se programme alors avec deux instructions, correspondant à l'érosion et à la dilatation géodésique itérée.

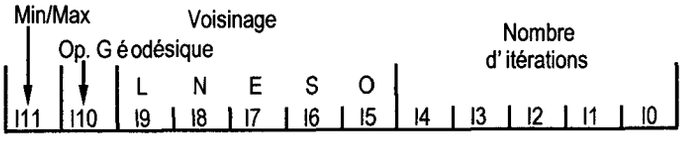


Figure 3. – Format de l'instruction.

4.3. interface de communication inter-PE

Comme expliqué plus haut, l'asynchronisme fonctionnel ne requiert aucune synchronisation entre les calculs effectués par les différents PEs. Il faut donc le mettre en œuvre au niveau du réseau de processeurs en utilisant un mécanisme spécifique d'échange entre PEs. Dans cette structure, chaque processeur élémentaire doit être capable de lire les valeurs les plus récemment mises à jour par les PEs voisins à n'importe quel moment, et ceci indépendamment de leur déphasage mutuel et sans signal de synchronisation préalable. Au niveau matériel, ceci correspond à autoriser un PE à échantillonner les variables de ses voisins à n'importe quel instant.

Le schéma de communication adopté n'utilise pas les règles classiques de séquencement de type requête-acquittement, qui ne sont en fait pas strictement nécessaires. Un échantillonnage direct des valeurs données par les registres de sortie des processeurs voisins est donc mis en oeuvre de manière inconditionnelle.

Le problème majeur dans la conception matérielle d'une telle interface est l'absence d'une quelconque relation de phase entre l'activation du signal de lecture et la mise à jour de la variable qui doit être lue. Si un conflit de lecture/écriture survient, il se peut qu'il y ait une tentative d'échantillonnage d'un signal électrique transitoire. Alors il se peut que des états électriques non-logiques se propagent dans le circuit, ou que l'opérateur d'échantillonnage entre dans un état métastable [18], ce qui la plupart du temps conduit à un comportement imprévisible. Pour éliminer ce problème, une solution à bases de « Q-Flops » [19] a été conçue. Un circuit Q-Flop est un point mémoire qui est capable d'échantillonner un signal d'entrée de manière asynchrone par rapport au signal de requête. Après l'activation du signal de requête, ce circuit produit de manière sûre une sortie à un niveau logique valide, zéro ou un, cependant en un temps théoriquement non borné. Il génère ensuite un signal d'acquittement qui indique que l'échantillonnage est terminé et que la sortie est à un niveau valide. Le protocole de communication ainsi que l'architecture de l'interface sont décrits dans [17].

4.4. chemin de données

Le processeur élémentaire doit pouvoir stocker trois pixels codés sur 8 bits, un pour la valeur d'itération courante qui est visible à partir des voisins, un autre pour une valeur temporaire utilisée à l'intérieur d'une itération, et un troisième pour le pixel de référence. Le chemin de données comprend les registres correspondants (respectivement L, B, et I), de même que des registres série/parallèle pour les données d'entrée/sortie qui se propagent indépendamment des calculs du réseau (E et R pour les pixels intermédiaires d'entrée et les pixels de référence respectivement, et S pour la sortie). Un multiplexeur sélectionne une valeur parmi le pixel de référence et les quatre valeurs des voisins. La partie opérative doit pouvoir calculer un minimum ou un maximum entre la valeur temporaire stockée dans B et la valeur voisine sélectionnée. Puisque la valeur temporaire B doit être initialisée avec la valeur du pixel d'entrée stocké en E, un multiplexeur est utilisé pour sélectionner la sortie du registre E ou la sortie de la partie opérative. En fait, la figure 4 montre que B n'est pas stocké dans un registre mais dans le multiplexeur lui-même, qui est conçu en logique DCVS avec latch [16]. Comme l'ALU est aussi implémentée en LDCVSL, l'ensemble constitue un anneau auto-séquencé avec seulement deux étages [14].

L'ALU est contrôlée par deux signaux qui spécifient si la sortie doit être égale au minimum ou au maximum des entrées A et B, ou encore égale à l'entrée A. Des signaux de requête et d'acquittement (qui ne figurent pas sur la figure 4) sont connectés

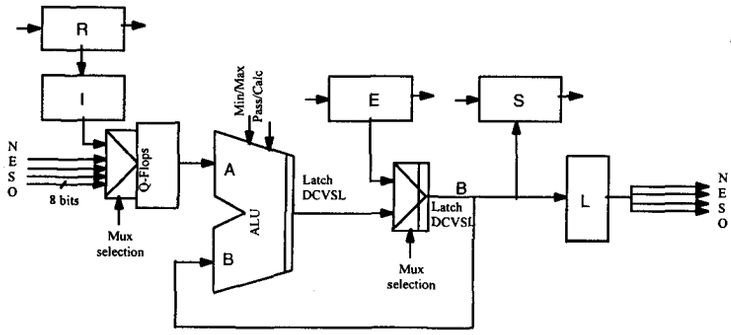


Figure 4. – Chemin de données.

à l'ALU, le multiplexeur B et les Q-Flops. Les registres reçoivent aussi des signaux locaux d'activation.

L'ALU est implémentée avec un additionneur et des multiplexeurs. Un additionneur asynchrone à propagation de retenue et détection de fin de calcul est utilisé, de telle sorte que l'ALU puisse tirer parti de l'existence de chemins critiques dynamiques et puisse donner lieu à des dispersions de temps de calcul significatives, conduisant à un temps de calcul moyen en $O(Log(n))$ pour une addition sur n bits [12].

4.5. contrôle asynchrone

Le schéma de contrôle consiste principalement à attendre une requête externe, copier le dernier résultat B dans le registre parallèle/série de sortie S, copier les valeurs d'entrée des registres série/parallèle E et R dans les registres de second niveau B et I, et démarrer le calcul itératif. Ce calcul contient deux boucles imbriquées : la boucle externe compte le nombre d'itérations, la boucle interne balaye les différents PEs appartenant au voisinage spécifié. Une fois que le résultat de l'itération précédente a été copié dans le registre d'itération courante L, la boucle interne sélectionne séquentiellement les valeurs voisines, et les combine à la valeur temporaire B.

Une approche modulaire a été utilisée pour implémenter le contrôle asynchrone. Il est relativement simple de spécifier un ensemble de modules de contrôle de base, qui peuvent être assemblés pour concevoir le contrôle global du chemin de données. Une fois que cet ensemble de modules correspond effectivement à la séquence de contrôle principale, des automates asynchrones plus complexes peuvent être ajoutés comme des modules « feuilles » de l'arbre de contrôle, par exemple les interfaces de lecture et d'écriture qui échangent des signaux de contrôle entre les PEs.

La figure 5 montre l'architecture du contrôle. Toutes les flèches correspondent à des signaux de requête/acquittement qui connectent entre eux les sous-modules de contrôle et le chemin de données. Le module de contrôle d'itération génère autant de cycles requête-acquittement à quatre phases que le spécifie le champ $< I4 - I0 >$. Chaque module « seq2a » est un automate asynchrone simple qui génère de manière conditionnelle un cycle

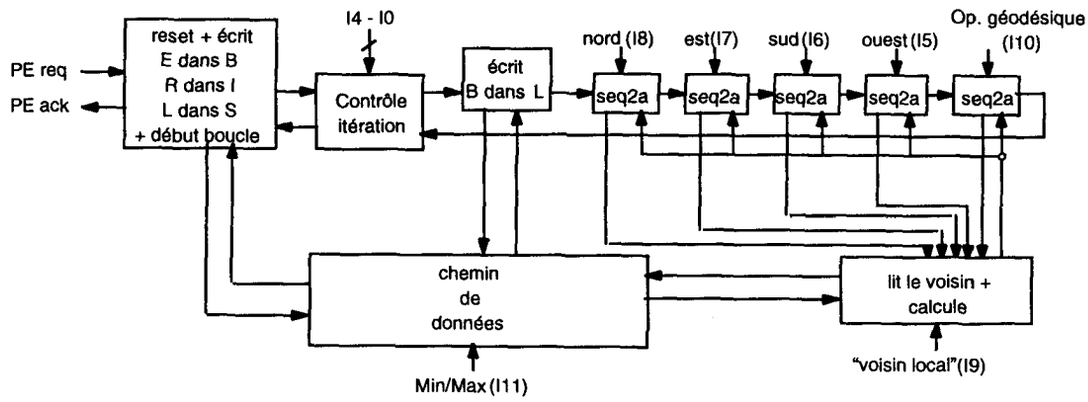


Figure 5. – Architecture simplifiée du contrôle.

req/ack vers le contrôle du chemin de données, en fonction de la valeur du bit de l'instruction qui lui est attaché. Si ce bit ($I8$ par exemple) est inactif, alors le voisin correspondant (Nord dans ce cas) n'est pas pris en compte (car l'instruction spécifie qu'il n'appartient pas à l'élément structurant) et le signal de requête est immédiatement propagé au module « seq2a » suivant.

Lorsque toutes les valeurs voisines ont été prises en compte et que l'opération géodésique duale a été éventuellement effectuée (en fonction du bit $I10$ de l'instruction), le module de contrôle d'itération peut commencer la prochaine itération. Finalement un signal d'acquittement est activé par l'entité correspondant au PE.

Chaque module de contrôle a été spécifié sous la forme d'un automate asynchrone à l'aide d'un graphe de transitions de signaux sans choix en entrée (STG/NC) [20]. La procédure de synthèse utilisée est inspirée des techniques présentées dans [21] et [22].

5. circuit test basé sur un réseau de 16 x 16 processeurs-pixel

5.1. layout du circuit

Le processeur élémentaire a été routé de manière à obtenir un bloc de niveau layout, dans lequel les positions relatives des ports d'entrée/sortie ont été fixées. Ceci permet ensuite de générer le layout global du réseau de processeurs, par simple aboutement de 14×14 blocs de layout élémentaires. Les cellules de bord sont routées séparément puis ajoutées au cœur. Les plots du circuit sont finalement connectés aux plots de bord du layout.

Le processeur élémentaire contient environ 3500 transistors pour une surface de $500 \times 500 \mu m^2$. La complexité du réseau de processeurs atteint 800000 transistors et une surface globale de $8 \times 9 mm^2$, dans une technologie CMOS 0.5 μm à trois niveaux de métal (Fig. 6).

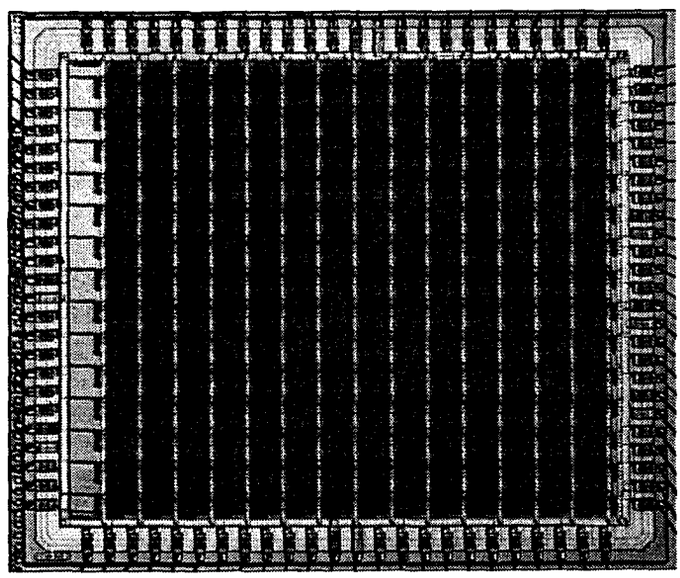


Figure 6. – Photomicrographie du circuit.

5.2. intégration système

Ce réseau de processeurs VLSI sera intégré dans un prototype de traitement temps réel. Puisque le circuit ne peut seulement traiter qu'un morceau de l'image complète, un schéma de traitement partitionné localement parallèle globalement séquentiel doit être utilisé. Une image doit ainsi être parcourue bloc par bloc. De plus, pour propager les résultats entre blocs, l'image doit être parcourue avec un certain recouvrement, d'au moins un pixel entre blocs

adjacents. Enfin, l'image doit être parcourue plusieurs fois jusqu'à ce que la convergence du traitement soit atteinte. Pour démontrer les capacités du circuit, des séquences d'images seront acquises en temps réel, traitées et visualisées. Deux interfaces spécifiques seront conçues pour interfacer le circuit avec un système standard d'acquisition d'images, comme le montre la figure 7.

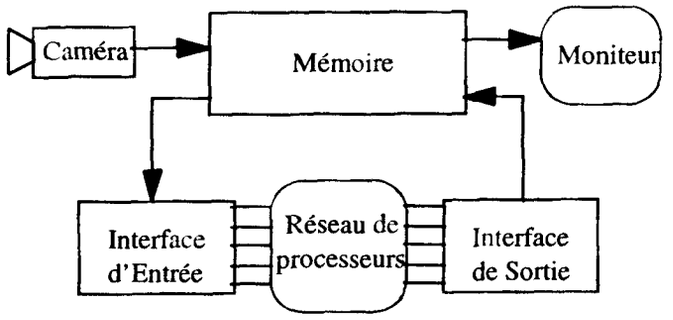


Figure 7. – Architecture du système.

5.3. résultats de test

Le circuit a été fabriqué au Centre Commun CNET SGS-THOMSON, et le test a montré qu'il était totalement fonctionnel. Un certain nombres de mesures concernant la vitesse et la consommation du circuit sont présentées ci-après.

A 3.3 V, l'exécution par un seul PE d'une instruction correspondant à une seule étape d'itération, avec le voisinage complet et l'opération géodésique duale, prend entre 200 ns et 260 ns, en fonction des données. Ce calcul correspond à cinq additions sur huit bits, cinq accès aux registres voisins et une opération d'écriture pour stocker le résultat. Le temps de calcul total correspondant à 31 itérations atteint 9 μs, ce qui inclut tous les temps dus au contrôle (liés aux itérations multiples, aux communications inter-PE, et au protocole de communication externe).

Selon ces chiffres, des images de 256×256 pixels peuvent donc être traitées à une fréquence maximale de 43 Hz avec un seul circuit. En effet, avec un recouvrement de un pixel, 17×17 blocs doivent être traités. D'après des simulations algorithmiques, la convergence globale est atteinte après au moins huit parcours de l'image. En surestimant le temps de calcul d'un bloc à 10 μs, le temps total de traitement d'une image est égal à 23 ms.

Le circuit est fonctionnel dans une plage de tension d'alimentation variant de 1.75 V à 4.75 V. Les figures 8 et 9 donnent respectivement le temps de calcul d'un bloc et la consommation en fonction de la tension d'alimentation. A 3.3 V, la consommation du circuit est d'environ 900 mW. A 1.8 V, elle est seulement de 100 mW, c'est-à-dire que la consommation est réduite d'un facteur 9, alors que le temps de calcul est seulement multiplié par 2 (20 μs).

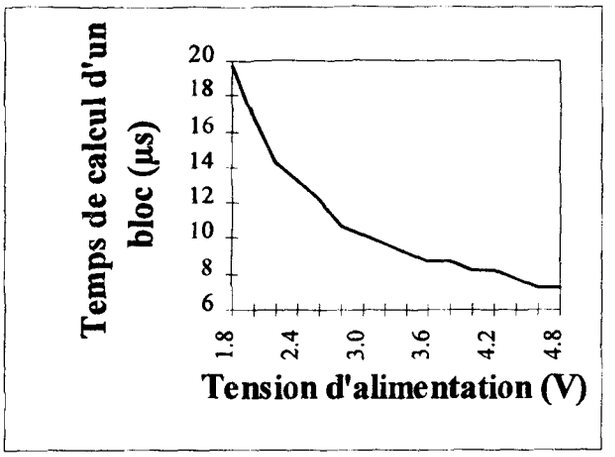


Figure 8. – Temps de calcul d'un bloc en fonction de la tension d'alimentation.

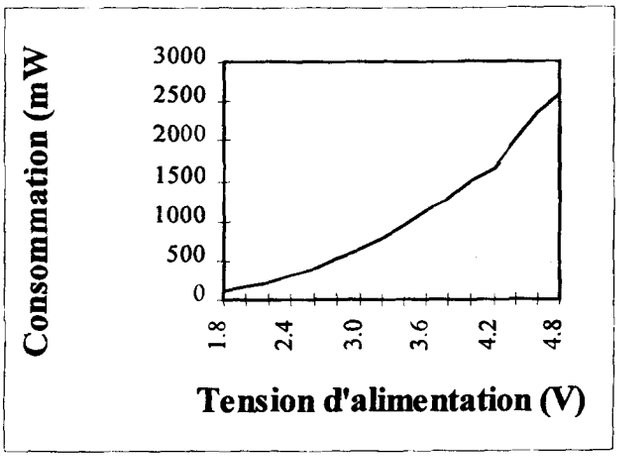


Figure 9. – Consommation en fonction de la tension d'alimentation.

6. conclusion et perspectives

Il a été démontré sur une application de traitement d'images que les circuits asynchrones élargissent le spectre de solutions possibles pour la conception conjointe d'algorithmes et d'architectures. La possibilité d'appliquer un asynchronisme fonctionnel au problème du filtrage morphologique, associée à une solution matérielle originale basée sur un asynchronisme structurel, a été présentée. Ceci illustre la façon dont le concept d'asynchronisme peut être exploité à différents niveaux, au niveau algorithmique pour améliorer la vitesse de convergence, et au niveau structurel pour accroître la modularité, l'extensibilité et la robustesse. Les architectures asynchrones apparaissent clairement comme une alternative prometteuse qui permet d'entrevoir de nouvelles possibilités de conception de systèmes intégrés.

Cette étude introduit l'idée que l'approche asynchrone autorise l'implémentation d'une classe plus étendue d'algorithmes parallèles, dont les contraintes de synchronisation sont relâchées. Ce travail ouvre un nouveau champ d'application pour les architectures asynchrones, qui devrait encourager le développement d'algorithmes basés sur des ensembles de processus concurrents communicants faiblement synchronisés [23].

Remerciements

Les auteurs remercient Nadia Van Den Bossche ainsi que leurs collègues du CNET Grenoble qui ont participés à la conception, vérification et test du circuit.

BIBLIOGRAPHIE

[1] K. Batcher, «Design of a Massively Parallel Processor», *IEEE Transactions on Computers*, vol. C-29, 1980, pp. 836-840.

[2] K. Preston, M. Duff, *Modern Cellular Automata*, Plenum Press 1984.

[3] G. Privat, P. Planet, M. Renaudin, «Asynchronous relaxation of locally-coupled automata networks, with application to parallel VLSI implementation of iterative image processing algorithms», *Proceedings of the International Conference on Application Specific Array Processors*, October 1993.

[4] G. Privat, F. Robin, M. Renaudin, B. El Hassan, «A fine-grain asynchronous VLSI cellular array processor architecture», *Proceedings of the International Symposium on Circuits And Systems*, Seattle, 1995.

[5] D. Chazan, W. Miranker, «Chaotic relaxation», *Linear Algebra and its Applications*, vol. 2, 1969, pp. 199-222.

[6] G. Baudet, «Asynchronous iterative methods for multiprocessors», *Journal of the Association for Computing Machinery*, vol. 25, n°2, April 1978, pp. 226-244.

[7] A. Üresin, M. Dubois, «Sufficient conditions for the convergence of asynchronous iterations», Parallel Computing, vol. 10, 1989, pp. 83-92.

[8] P. Salembier, J. Serra, «Morphological multiscale image segmentation», *Proceedings SPIE on Visual Communications and Image Processing*, vol. 1818, 1992, pp. 620-631.

[9] L. Vincent, «Morphological grayscale reconstruction in image analysis : applications and efficient algorithms», *IEEE Transactions on Image Processing*, vol. 2, n°2, April 1993, pp. 176-201.

[10] F. Robin, G. Privat, M. Renaudin, «Asynchronous relaxation of morphological operators : a joint architecture-algorithm perspective», *Proceedings of the International Workshop on Parallel Image Analysis*, Lyon, France, December 1995.

[11] B. El Hassan, «Architecture VLSI asynchrone utilisant la logique différentielle à précharge : application aux opérateurs arithmétiques», Ph.D. Thesis, INPG, Grenoble, France, September 1995.

[12] M. Renaudin, B. El Hassan, «The design of fast asynchronous adder structures and their implementation using DCVS logic», *Proceedings of the International Symposium on Circuits And Systems*, London, 1994.

[13] M. Renaudin, B. El Hassan, «A minimum power, 100 MHz, 12×18 +30-b Multiplier-Accumulator operating in asynchronous and synchronous mode», *Proceedings of the European Solid-State CIRcuits Conference*, Ulm, Germany, September 1994.

[14] B. El Hassan, A. Guyot, M. Renaudin, V. Levering , «New self timed rings and their application to division and square root extraction», *Proceedings of the European Solid-State CIRcuits Conference*, Lille, France, September 1995, pp. 226-229.

[15] C.H. Erdelyi, W.R. Griffin, R.D. Kilmoyer, «Cascode Voltage Switch Logic Design», *VLSI Design*, October 1984, pp. 78-86.

[16] M. Renaudin, B. El Hassan, A. Guyot, «A new asynchronous pipeline scheme : application to the design of a self-timed ring divider», *IEEE Journal of Solid-State Circuits*, vol. 31, n°7, July 1996, pp. 1001-1013.

[17] F. Robin, M. Renaudin, G. Privat, N. Van Den Bossche, «Functionally asynchronous array-processor for morphological filtering of greyscale images», IEE Proceedings on Computers and Digital Techniques, special section on Asynchronous Architecture, vol. 143, n°5, September 1996, pp. 273-281.

[18] L. Kleeman, A. Cantoni, «Metastable behavior in digital systems», *IEEE Design & Test of Computers*, December 1987, pp. 4-19.

[19] F.U. Rosenberger, C.E. Molnar, T.J. Chaney, T.P. Fang, «Q-modules : internally clocked delay-insensitive modules», *IEEE Transactions on Computers*, vol. 37, n°9, September 1988.

[20] S. Hauck, «Asynchronous Design Methodologies : An Overview», *Proceedings of the IEEE*, vol. 83, n°1, January 1995, pp. 69-93.

[21] T.A. Chu, «Synthesis of hazard-free control circuits from asynchronous finite state machines specifications», *Journal of VLSI Signal Processing*, vol. 7, 1994, pp. 61-84.

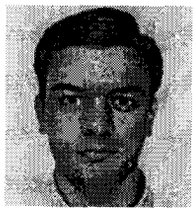
[22] T.H. Meng, R.W. Brodersen, D.G. Messerschmitt, «Automatic synthesis of asynchronous circuits from high-level specifications», *IEEE Transactions on Computer-Aided Design*, vol. 8, n°11, November 1989, pp. 1185-1205.

[23] F. Robin, «Etude d'architectures VLSI numériques parallèles et asynchrones pour la mise en œuvre de nouveaux algorithmes d'analyse et rendu d'images», Thèse de doctorat de l'ENST Paris, spécialité Electronique et Communications, 27 octobre 1997.

Manuscrit reçu le 23 Juillet 1997.

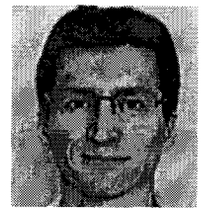
LES AUTEURS

Frédéric ROBIN



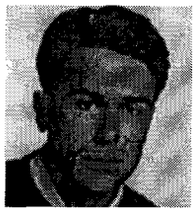
Frédéric Robin, né à Nantes, en 1972, est diplômé de l'Ecole Nationale Supérieure des Télécommunications de Bretagne (1994). Il a obtenu le titre de docteur de l'Ecole Nationale Supérieure des Télécommunications de Paris, spécialité «Electronique et Communications», en octobre 1997. Sa thèse a porté sur «L'étude d'architectures VLSI numériques parallèles et asynchrones pour la mise en œuvre de nouveaux algorithmes d'analyse et rendu d'images».

Marc RENAUDIN



Marc Renaudin est docteur ingénieur de l'Institut National Polytechnique de Grenoble (1990). Il a rejoint l'Ecole Nationale Supérieure des Télécommunications de Bretagne, Antenne de Grenoble, en 1990, où il occupe un poste de Maître de Conférences. Il enseigne l'Architecture des Ordinateurs, les Circuits et Systèmes Asynchrones et le Traitement Numérique du Signal. En 1995, il a effectué un séjour de six mois aux Etats-Unis, dans le groupe du Professeur Alain Martin à Caltech. Ses activités de recherche portent conjointement sur le concept d'Asynchronisme et les Architectures VLSI pour l'analyse/rendu d'images. Il étudie en particulier comment l'asynchronisme peut être valorisé, de la spécification à l'implémentation, pour intégrer des systèmes complexes, performants, faible consommation.

Gilles PRIVAT



Gilles Privat a obtenu son diplôme d'ingénieur et son doctorat en théorie du signal et des systèmes de l'Ecole Nationale Supérieure des Télécommunications de Paris, respectivement en 1981 et 1986. En 1982-1983 il était chercheur doctorant au laboratoire INPG-IMAG à Grenoble. Depuis 1983 il travaille au Centre National d'Etudes des Télécommunications de Grenoble. Ses principales activités de recherche ont été sur les VLSI pour le traitement du signal, les architectures et algorithmes parallèles, les réseaux d'automates et le codage d'images. Il travaille actuellement sur les architectures logicielles pour les systèmes multimédia répartis . Il est membre du comité éditorial de la revue IEEE MICRO. Il est auteur ou co-auteur de plus de 40 communications et articles et de 4 brevets.