Asynchronisme dans les rétines artificielles

Valentin Gies

N° D'ORDRE : 8145

Asynchronisme dans les rétines artificielles

THÈSE DE DOCTORAT

SPÉCIALITÉ : ÉLECTRONIQUE

présentée et soutenue publiquement le 12 Décembre 2005, pour l'obtention du

Doctorat de l'Université Paris-Sud XI
Faculté des Sciences d'Orsay

par

Valentin Gies

Composition du jury

Président :	Marc Renaudin
Rapporteurs :	Michel Couperie Marc Renaudin
Examinateurs :	Thierry Collette Patrick Garda
Directeurs :	Thierry Bernard Alain Mérigot

Laboratoire d'Electronique et d'Informatique. ENSTA - 32 Boulevard Victor F75015 Paris

Institut d'Electronique Fondamentale. Université Paris-Sud - CNRS UMR 8622
Bât. 220 - Centre scientifique d'Orsay - F91405 Orsay Cedex

Mis en page avec la classe thloria.

Remerciements

Je tiens à remercier vivement tous ceux que j'ai pu côtoyer à l'ENSTA durant ces trois années de thèse, mais avant d'en faire une liste qui ne sera bien sûr jamais exhaustive, je tiens à remercier en particulier les personnes qui ont contribué de manière essentielle à ce travail.

En premier lieu, je tiens à remercier mes parents pour l'éducation qu'ils m'ont donnée, pour leur soutien constant dans un contexte parfois difficile et pour la liberté de choix qu'ils m'ont donnée tout au long de mes études.

Je remercie Marie, mon épouse, pour son soutien de tous les jours et pour les sacrifices qu'elle a consentis durant la rédaction de cette thèse.

La réalisation d'un projet de 3 ans tel qu'une thèse ne peut se faire sans l'aide d'une personne en qui on peut avoir une vraie confiance scientifique et humaine. Je remercie de tout coeur Thierry Bernard pour m'avoir proposé cette thèse et pour avoir encadré mon travail scientifique de ses conseils avisés au cours de ces 3 années. Sa philosophie de la recherche de la *pureté* dans les rétines, sur un plan électronique autant que sur un plan algorithmique a été pour moi une source d'inspiration très forte. Je le remercie également pour avoir passé de nombreuses heures, parfois le week-end, à m'aider à rédiger des articles scientifiques. Enfin, je le remercie pour m'avoir donné une grande autonomie dans mon travail, et pour avoir su trouver un équilibre subtil dans son encadrement afin de me permettre d'avancer dans mon travail, sans pour autant m'imposer une piste de recherche particulière.

Mes remerciements vont également à Alain Mérigot, mon co-directeur de thèse avec Thierry Bernard, chacune de nos rencontres a été pour moi une source de nouvelles idées qui se sont toujours avérées fructueuses.

Je tiens également à remercier les rapporteurs de cette thèse, dans l'ordre alphabétique Michel Couperie et Marc Renaudin.

Merci à la DGA et au CNRS, qui ont financé cette thèse dans le cadre d'une bourse DGA/CNRS, et à M. Desruelle qui a participé à l'encadrement de cette thèse en tant que responsable du groupe POLOQ (Prospective Orientée sur les Lasers et l'Optronique) auquel cette thèse était rattachée.

J'adresse également mes remerciements à deux personnes qui ont été proches de moi durant cette thèse, Antoine Manzanera pour ses conseils avisés et ses critiques très pertinentes qui m'ont permis d'aller plus en profondeur sur certains points notamment algorithmiques, et Julien Richefeu avec qui j'ai pu échanger de nombreuses idées et avec lequel nous avons passé de bons moments.

Je tiens à adresser mes remerciements à Alain Sibille, directeur du LEI pour son

accueil, et pour l'attention dont il a fait preuve à l'égard des doctorants et stagiaires au quotidien. Je remercie également Mme Darrozes pour son aide précieuse pour toutes les formalités administratives et sa bonne humeur. Un grand merci également à Martine, Gilles et Vladimir qui assurent sans faillir le soutien technique nécessaire à la réalisation des projets de recherche à l'ENSTA.

Enfin, j'adresse toute ma sympathie à toutes les personnes que j'ai côtoyées de près ou de loin à l'ENSTA, les permanents du laboratoire, les doctorants et stagiaires que j'ai pu croiser et qui se reconnaîtront, les élèves de l'ENSTA avec lesquels j'ai eu l'occasion de travailler dans le cadre de la Coupe E=M6 de robotique, et enfin et surtout tout le personnel de l'ENSTA qui reste dans l'ombre et dont je ne connais peut-être pas le nom, mais qui participe à l'entretien, à la bonne marche de l'école et de notre laboratoire et rend agréable notre vie quotidienne.

A mes parents,
A Marie,

Résumé

Le traitement d'image se divise usuellement en opérations de bas, moyen et haut niveau. Les opérateurs régionaux de moyen niveau sont un maillon essentiel de la chaîne de traitement de l'image. Dans une étude préliminaire, nous montrons leur importance dans le cadre d'une approche énergétique du traitement d'image et nous déterminons une forme générale sous laquelle ces opérateurs peuvent être implantés afin de pouvoir être utilisés de manière efficace.

Après avoir montré la nécessité de l'asynchronisme pour implanter ces opérateurs régionaux, nous étudions les structures existant dans la littérature. Le coût matériel élevé de l'asynchronisme et de ces structures étant trop important pour une implantation dense dans des rétines artificielles, nous tenons à limiter au minimum possible les ressources utiles aux calculs régionaux asynchrones.
Cela nous conduit à une nouvelle structure de calcul régional, les *micropipelines associatifs*, implantant partiellement le modèle des réseaux associatifs, en particulier la somme régionale. Cette structure est basée sur un élément de circuit asynchrone, le *micropipeline convergent* ayant un coût d'implantation matériel faible. Il utilise un mode alternatif de transmission asynchrone que nous appelons la *transmission par jetons*.
Afin de réduire encore le coût des opérateurs régionaux définis précédemment, nous optimisons ensuite le réseau de connexion asynchrone.

Dans la partie algorithmique, nous montrons que la possibilité d'utiliser des opérateurs régionaux de manière efficace conduit à élargir le cadre du traitement d'image. Nous illustrons cette extension dans le cadre de la segmentation d'images. Après une étude des méthodes de segmentation existantes, une méthode basée sur des mesures statistiques régionales permettant d'améliorer la segmentation à base de ligne de partage des eaux sera proposée.
Enfin, nous introduisons une nouvelle méthode de segmentation, la *segmentation sociétale* basée sur une analogie avec la segmentation d'un territoire en villes et villages, utilisant de manière intensive les mesures régionales.

Mots-clés: rétine artificielle, asynchronisme, opération régionale, reconfigurabilité, traitement d'image, segmentation

Abstract

Image processing is usually divided into low- mid- and high-level operations. Mid-level regional operators are a key feature in image processing. In a preliminary study, we present their importance considering image processing with an energetic point of view. We also determine a general way of implementing these operators in order to use them in an efficient way.

After having the necessity of asynchronism for implementing regional operators, we review existing structures for massively parallel regional computation. Because the hardware cost of asynchronism is too important for a dense implementation in artificial retinas, we propose to reduce to the minimum possible the hardware ressources needed for regional asynchronous computations.
This leads to a new regional computation structure, the *associative micropipelines*, implementing part of the associative mesh model, especially the regional sum. This structure is based on an asynchronous element, the *convergent micropipeline* having a reduced hardware cost. The *associative micropipelines* use an alternative mode of asynchronous communication we call the *token transmission*.
In order to reduce again the regional operators hardware cost defined before, we propose a way to optimize the asynchronous interconnexion network.

In the algorithmic part, we show that the possibility of using efficient regional operators allow to extend the image processing framework.
We propose to illustrate this extension applying it to image segmentation. After a review of the existing segmentation methods, we propose a statistical solution to watershed over-segmentation issue.
Finally, we present a novel segmentation method, *societal segmentation*, based on an analogy between segmentation of a land into towns and villages and image segmentation, using regional measures in an intensive way.

Keywords: Artificial retina, asynchronism, region-based operation, reconfigurability, image processing, segmentation

Table des matières

Table des figures

Introduction générale

Lorsque l'on termine une thèse en vision et traitement d'image, plus particulièrement axée sur les architectures destinées à faciliter la compréhension et l'utilisation de l'image, dans un capteur de petite taille et ne consommant que peu d'énergie, il est légitime, sinon important de se poser la question de l'utilisation qui peut en être faite.

Comment ne pas avoir en tête les télécrans de 1984 de Georges Orwell, scrutant les

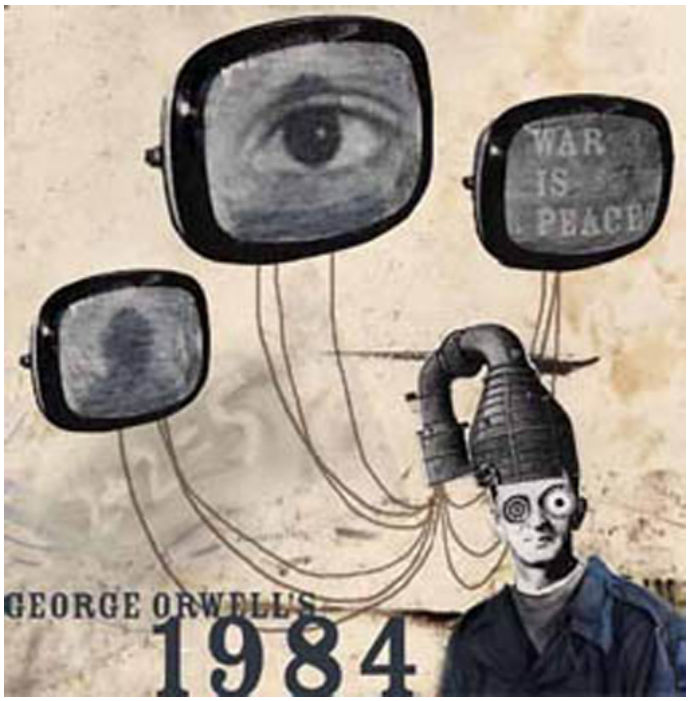


Fig. 1 – 1984, Georges Orwell (1949)

moindres faits et gestes, les réflexes et même les pensées des individus. Ces télécrans n'étaient à l'époque qu'un symbole, celui de la volonté des régimes communistes d'imposer une vision unique et de contrôler tout ce qui peut se passer. Dans cette utopie, publiée en 1949, à l'heure où le transistor naissait et bien avant que le traitement d'image ne soit devenu un objet de recherche, les télécrans étaient bien loin d'une quelconque réalité.

Cinquante ans ont passé et le rideau de fer est tombé. Mais dans le même temps les télécrans se sont rapprochés de notre quotidien avec l'essor du traitement d'image et le développement fulgurant des réseaux et outils de télécommunications fixes et mobiles. Aujourd'hui, les applications de traitement d'image ont quitté leur statut de symbole pour devenir des réalités : comptage et identification de personnes ou de véhicules, télé-surveillance, détection de présence...
Et demain, que se passera-t-il ? Avec le développement rapide des applications nomades, ces systèmes aujourd'hui limités à un usage assez spécifique seront probablement à la disposition du grand public sous des formes toujours plus réduites, toujours plus autonomes et ayant des capacités de communication toujours plus importantes.

Les risques potentiels inhérents au développement des systèmes de traitement d'image doivent cependant être nuancés par l'état actuel de l'avancée technologique. Nous sommes encore bien loin du mythe des télécrans, capables n'analyser les faits et gestes des individus et même leurs pensées. Aujourd'hui, nous n'en sommes qu'à exécuter des tâches simples et bien définies telles que la détection de la présence d'une personne ou l'identification d'un visage dans un environnement bien défini. De plus, nous pouvons espérer que ces applications seront utilisées de manière positive dans le futur, une orientation qui a déjà été engagée dans certaines applications :

- Le développement de caméras destinées à améliorer de manière active la sécurité des véhicules routiers : capteurs anti-collision...

- L'utilisation des caméras pour assister les chirurgiens dans leur travail.

- L'identification des personnes à l'entrée de bâtiments, la détection d'intrusions illicites de personnes ou de véhicules sur un territoire en vue de protéger les populations d'actes terroristes ou malveillants.

- Guidage de drônes à l'aide de caméras permettant d'éviter de faire courir des risques à des pilotes humains, mais également permettant d'améliorer les capacités de ciblage des frappes aériennes en vue d'épargner les populations locales.

Les rétines artificielles, une analogie biologique

Comme l'attestent les exemples précédents, le champ d'utilisation du traitement d'image est très diversifié mais les applications sont pour l'instant limitées à des opérations simples et très dédiées à un système donné, qu'un oeil pourrait effectuer aisément et de manière beaucoup plus générique. Le bien fondé des applications rappelées ainsi que la distance restant à parcourir pour se rapprocher de la vision humaine, ont contribué à motiver les efforts de la communauté scientifique

pour mieux comprendre les mécanismes de la vision humaine. Ces études n'ont pas apporté de réponses claires sur le fonctionnement de l'oeil, mais ont en revanche apporté des connaissances sur la structure fonctionnelle de l'oeil.

Le concept de *rétine artificielle*, cadre de travail de cette thèse, tente de se rapprocher des réalités biologiques de l'oeil humain par différents aspects :

- L'information transcrite par les photorécepteurs est traitée de manière locale massivement parallèle au sein du pixel.
- Les interactions entre pixels voisins s'effectuent sur un voisinage de taille limitée spatialement.
- La quantité d'information transmise au système assurant la prise de décision est réduite par les traitements locaux effectués dans le pixel.

Le concept de rétine artificielle tel qu'il est défini est toutefois un peu éloigné de la réalité structurelle biologique. La limitation des interactions entre processeurs à un voisinage de taille limitée réduit le champ d'utilisation des rétines artificielles aux *opérations locales de bas niveau*. Or, dans les rétines biologiques, les cellules permettant les interactions entre photorécepteurs sont disposées en couches, chaque couche ayant un voisinage de plus en plus large à mesure que l'on se rapproche du nerf optique. Cette organisation permet aux rétines biologiques d'effectuer des opérations locales à résolutions multiples. Si l'on va un peu plus en avant dans la chaîne de traitement de l'image, on trouve le cerveau où s'effectuent les opérations d'interprétation de l'image. Une des premières étapes de ces opérations consiste à considérer des ensembles de points comme des objets (ou régions) afin de pouvoir effectuer des opérations dessus. Ces opérations sur des régions sont communément appelées *traitements de moyen niveau*. Ces traitements ont un intérêt dans la mesure où ils permettent de réduire les quantités d'informations transmises tout en élevant le niveau d'abstraction de ces informations.

Vers une extension du concept de rétine artificielle aux traitements régionaux

L'extension du concept de rétine artificielle aux opérations de moyen niveau constitue donc une étape importante pour envisager d'effectuer des opérations de traitement d'image plus complexes et plus proches des opérations effectuées par l'oeil et le cerveau. L'intérêt potentiel de ces opérations de moyen niveau a conduit à proposer cette thèse sous la direction conjointe de Thierry Bernard (ENSTA) et d'Alain Mérigot (IEF, Université de Paris Sud Orsay).

Cette thèse est divisée en trois parties :

– La première partie présente le concept des rétines artificielles classiques et montre l'intérêt des traitements de moyen niveau dans la chaîne de traitement d'image.

– La deuxième partie traite de la régionalisation dans les rétines artificielles. Après avoir montré l'utilité des opérations asynchrones sur des graphes pour implanter les opérateurs régionaux, une analyse des architectures existantes permettant d'implanter ces opérateurs est effectuée. Elle conduit à proposer une structure alternative, appelée *micropipeline convergent*, associée à un mode de transmission asynchrone nouveau, la *transmission par jeton*. Elle permet d'effectuer des calculs régionaux pour un coût en transistor au sein de chaque pixel faible. Les primitives régionales élémentaires associées à cette structure de calcul sont également proposées.

– La troisième partie présente un exemple d'utilisation des opérateurs régionaux proposés précédemment. La segmentation d'image en objets, opération de traitement d'image typiquement régionale est abordée. Les méthodes de segmentation les plus courantes sont rappelées. Une méthode de segmentation améliorant par utilisation de mesures régionales la ligne de partage des eaux est proposée. Enfin une nouvelle méthode de segmentation utilisant intensivement les opérateurs régionaux est présentée, la *segmentation sociétale*.

Nous présentons le cheminement des idées de cette thèse sous la forme du schéma synoptique présenté à la figure 2.

Contexte de rédaction

Dans cette thèse, nous avons travaillé sur l'asynchronisme dans les rétines artificielles dans un but de prospection et sans contrainte théorique telle qu'un cadre de travail rigide et strict basé par exemple sur des aspects algorithmiques, électroniques ou encore sur une forme précise d'asynchronisme. A cette liberté, a été ajoutée une contrainte forte : la possibilité d'implanter réellement dans une rétine artificielle les fonctionnalités asynchrones proposées.

Cette liberté de pensée assortie d'un objectif concret m'a conduit à travailler sur l'asynchronisme et la régionalisation dans les rétines artificielles selon plusieurs axes : la prospection sur les opérateurs implantables dans une rétine, l'utilisation de ces opérateurs (régionaux) en traitement d'image et l'implantation électronique des opérateurs. Un des fils conducteurs de cette thèse a été la pureté électronique et algorithmique visant à épurer au maximum les différentes versions des opérateurs régionaux mis en jeu de manière à se rapprocher d'une structure où chaque transistor a un rôle qui le rend indispensable.

La grande liberté d'action et de pensée qui a été le cadre de cette thèse m'a permis de découvrir des domaines de recherche diversifiés, depuis l'électronique jusqu'au traitement d'image, en passant par la géométrie discrète, l'asynchronisme, ou encore l'étude des circuits à faible consommation (point sur lequel j'ai travaillé durant ma thèse [GB05d] mais qui ne figure pas dans ce manuscrit). Cette découverte de divers domaines scientifiques m'a pris du temps, et chacun des domaines abordés, par sa richesse respective, aurait pu donner lieu à lui seul à une thèse si je m'y étais attaché.
L'objectif de cette thèse était de mettre en parallèle ces différents domaines de manière à aboutir à une solution globalement satisfaisante, mais qui pourra parfois paraître incomplète dans sa justification au spécialiste d'un domaine particulier. Cette tentative de mise en adéquation de différents domaines scientifiques dans le but de réaliser une rétine artificielle ayant des capacités régionales m'a donc conduit à laisser de côté (au moins dans la rédaction de ce manuscrit) certains aspects des domaines que j'ai pu aborder, ce qui justifie le fait que le lecteur puisse être parfois déçu par le manque de détails ou de comparaisons de certaines parties.

Concernant le choix d'aborder de nombreux domaines dans cette thèse, il résulte de l'orientation volontaire qui m'a permis de rencontrer des communautés scientifiques et des personnes passionnantes, et il m'a permis également de me faire une idée diversifiée et assez complète de ce qu'est la recherche scientifique, et c'est pour moi également un des objectifs d'une thèse.

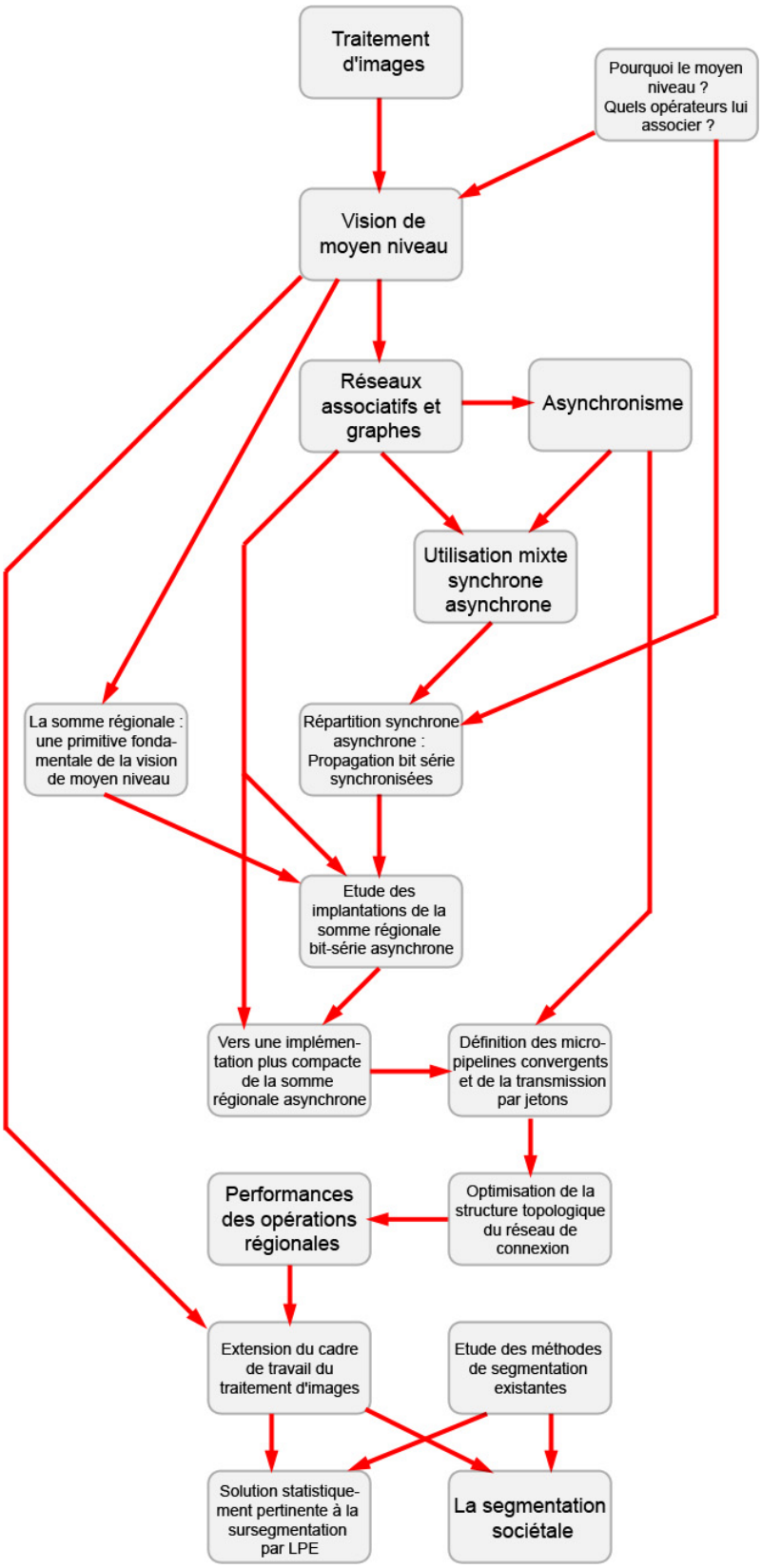


FIG. 2 – Schéma synoptique de la thèse

Première partie

Les rétines artificielles

1

Concepts et enjeux du traitement d'image dans les rétines artificielles

1.1 Analogie biologique

Afin de débuter l'étude des systèmes de vision artificielle, observons un dispositif de vision que nous utilisons en permanence : l'oeil (fig. 1.1). L'oeil est le premier maillon de la chaîne de traitement d'image, il a un rôle de dispositif optique, d'acquisition et de pré-traitement de l'image.

Les maillons suivants de la chaîne de traitement de l'image sont constitués par

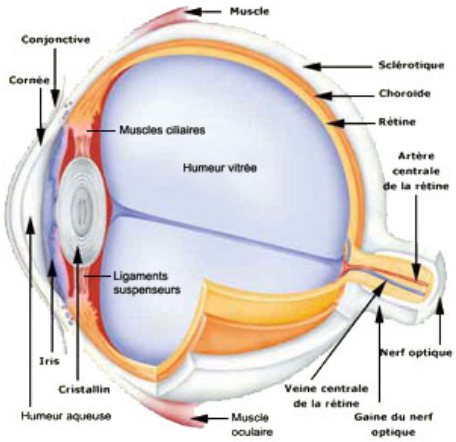


FIG. 1.1 – Schéma en coupe d'un oeil humain

le nerf optique et les zones du cerveau utilisées principalement pour l'analyse des informations transmises par l'oeil. Nous n'étudierons ici que la structure d'acquisition et de traitement de l'information dans l'oeil. Les informations transmises par l'oeil au cortex ne sont pas les images brutes acquises par les récepteurs sensoriels

de l'oeil. En effet, l'étude de la structure des récepteurs et des connexions nerveuses qui lui sont attachées montre que les informations transmises par l'oeil sont d'abord le fruit de la combinaison des informations acquises par les cônes et les bâtonnets (récepteurs sensoriels de l'oeil) dans des voisinages de taille spatiale limitée (fig. 1.2 et 1.3). Concrètement, la rétine est une fine pellicule de tissu nerveux ayant la

FIG. 1.2 – Image microscopique de l'organisation d'une rétine humaine

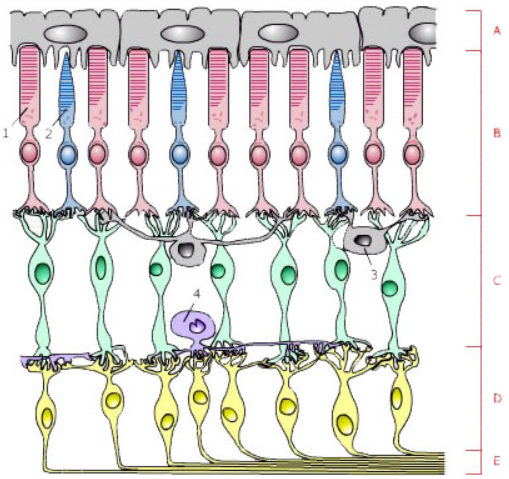


FIG. 1.3 – Schéma organisationnel d'une rétine humaine

consistance et l'épaisseur de quelques dixièmes de millimètres. Les neurones de la rétine sont organisés en trois couches principales séparées par deux couches intermédiaires où se font surtout des connexions entre les différents neurones (fig. 1.4).

La première couche principale est constituée des cônes et des bâtonnets permettant l'acquisition du signal. Nous ne détaillerons pas cette partie.

Le seconde couche principale est constituée de cellules bipolaires. Entre la couche d'acquisition et la couche bipolaire, se situe une couche intermédiaire constituée par les cellules horizontales. Les cellules horizontales sont connectées latéralement à plusieurs cônes, bâtonnets et neurones bipolaires. Leur rôle est d'inhiber l'activité des cellules avoisinantes. Cette suppression sélective de certains signaux nerveux s'appelle l'inhibition latérale et son rôle général est d'augmenter l'acuité d'un signal sensoriel. Dans le cas de la vision, quand une source lumineuse atteint la rétine, elle peut illuminer fortement certains photorécepteurs et d'autres beaucoup moins. En supprimant le signal de ces photorécepteurs moins illuminés, les cellules horizontales assurent que seul le signal des photorécepteurs bien illuminés est transmis aux cellules ganglionnaires, améliorant ainsi le contraste et la définition du stimulus visuel.

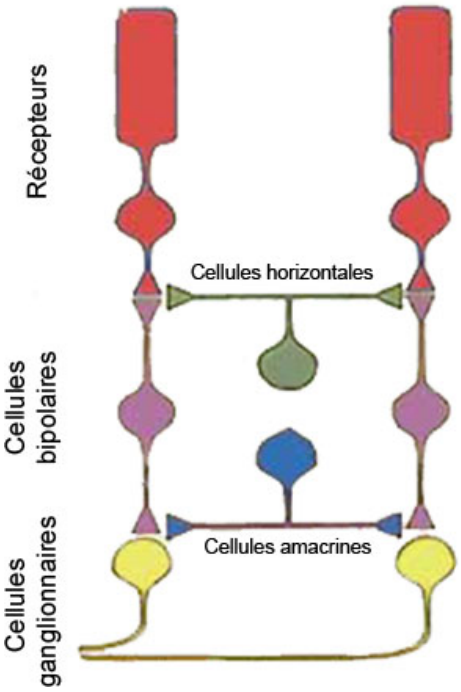


FIG. 1.4 – Schéma fonctionnel d'une rétine humaine

La troisième couche est constituée de cellules ganglionnaires. Elle possède deux types principaux de cellules nerveuses, les cellules P et les cellules M. Chacun de ces types de cellules possède une localisation et une connectivité particulières.
Les petites cellules ganglionnaires de type P (pour *parvus* (fig. 1.5), petit en latin) représentent environ 90% de la population totale de cellules ganglionnaires. Les grandes cellules de type M (pour *magnus* (fig. 1.6), grand en latin) constituent environ 5% de la population ; il existe également des cellules ganglionnaires non M-non P qui ne sont pas encore bien caractérisées et qui forment le 5% restant.

Les cellules de type P (fig. 1.5) sont placées près des récepteurs de type bâtonnets et cônes, elles ont un voisinage limité et une réponse maintenue aussi longtemps que leur stimulus d'entrée agit. Les cellules de type M (fig. 1.6) présentent de plus grands champs récepteurs, et propagent les potentiels d'action plus rapidement dans le nerf optique, et sont plus sensibles aux stimuli à faible contraste. La réponse d'une cellule M à une stimulation est limitée dans le temps.
L'idée la plus couramment admise est que les cellules P, avec leur petit champ récepteur, sont sensibles à la forme et aux détails de l'image, et que les cellules M sont particulièrement impliquées dans la détection du mouvement du stimulus.

Entre la couche bipolaire et la couche ganglionnaire, se situent les cellules amacrines. Leur morphologie est très diversifiée, elles utilisent un nombre impression-

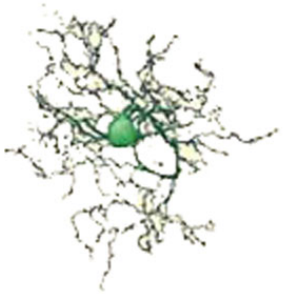


FIG. 1.5 – Cellules rétiniennes de type P

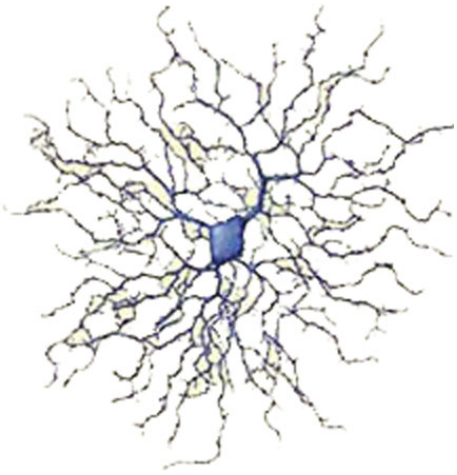


FIG. 1.6 – Cellules rétiniennes de type M

nant de neurotransmetteurs. En reliant les neurones bipolaires et ganglionnaires, elles forment une route alternative indirecte entre ceux-ci. Elles semblent avoir plusieurs fonctions mais leur rôle précis est pour l'instant encore mal compris.

Le fonctionnement de la rétine comme dispositif de traitement d'image est donc très fortement structuré. Les interactions entre cellules s'effectuent dans un voisinage de plus en plus large à mesure que l'on s'approche du nerf optique. Ces interactions permettent d'effectuer des calculs locaux multi-résolution. Au voisinage des récepteurs sensoriels, les cellules horizontales permettent de rehausser le contraste, opération de bas niveau basique de traitement d'image. Au voisinage du nerf optique, les cellules ganglionnaires de type M permettent d'effectuer des opérations de détection de mouvement sur des voisinages larges. Cette dernière opération de bas niveau relativement complexe permet ensuite au cortex visuel primaire (qui est une prolongation du nerf optique et qui à ce titre peut être considéré comme étant rattaché à l'oeil) d'interpréter les images comme un ensemble d'objets et non pas comme un ensemble de pixels. Cette dernière fonction est à la base des traitements d'image de moyen niveau reposant sur la régionalisation. Ainsi, l'ensemble cortex primaire et rétine permet d'effectuer des opérations de bas et moyen niveau massivement parallèles proches de l'endroit où s'effectue l'acquisition de l'image.
L'image prétraitée et régionalisée par la rétine et le cortex est ensuite transmise au cerveau. L'interprétation de l'image n'est pas effectuée à l'intérieur de la rétine (dont la complexité est relativement faible), mais à l'intérieur du cerveau (dont la complexité est très forte). Toutefois les informations transmises au cerveau ont fait l'objet de traitement ayant une échelle spatiale croissante à mesure que l'on se rapproche de celui-ci.

Le système de vision humaine et d'interprétation est intriguant car il n'existe aucun système de traitement d'image aussi performant à l'heure actuelle. Il permet en particulier de percevoir et d'interpréter une multitudes d'objets contenus dans une scène complexe en un temps quasi instantané. Ce système de vision étant l'aboutissement d'une longue évolution, on peut donc légitimement penser qu'il permet de réaliser le traitement d'images de manière efficace, en réduisant par exemple le coût de traitement à une valeur acceptable par un être humain.

Cette manière de traiter l'information localement à l'intérieur de la rétine a inspiré une classe d'imageurs utilisés en traitement d'images, les rétines artificielles. Cette analogie ne permet pas cependant de justifier la pertinence de l'utilisation des rétines artificielles. Leur pertinence sera montrée lors d'une étude sur le coût énergétique de traitement de l'information. Comme nous le verrons dans l'état de l'art, ces rétines intègrent des fonctionnalités de traitement d'image à l'intérieur même des pixels où s'effectue l'acquisition. Ces fonctionnalités permettent d'effectuer des calculs locaux massivement parallèles utilisant les valeurs des pixels voisins. Cependant, à l'inverse des rétines biologiques, les rétines artificielles n'intègrent pas d'opérateurs permettant d'effectuer des calculs régionaux (telles que les cellules ganglionnaires de type M ou le cortex primaire par exemple). Un des objectifs de cette thèse est d'étudier les possibilités permettant d'étendre le champ des opérateurs présents dans les rétines artificielles aux opérateurs régionaux.

1.2 Enjeux et contraintes des rétines artificielles

Avant de proposer un inventaire des rétines artificielles existant dans la littérature du traitement d'image, examinons tout d'abord les enjeux et contraintes du concept de rétine artificielle.

1.2.1 Enjeux des rétines artificielles

Vitesse de traitement

Les rétines artificielles ont été inspirées par les rétines biologiques. Outre le côté esthétique de l'analogie biologique, c'est avant tout en espérant obtenir les avantages des rétines biologiques que les chercheurs se sont dirigés vers les rétines artificielles. Quels sont ces avantages ? Comme nous l'avons rappelé précédemment, le système visuel humain est capable de percevoir et d'interpréter une multitude d'objets contenus dans une scène en un temps très bref. Rappelons toutefois que l'interprétation de l'image est le chef du cortex, et qu'elle ne sera pas étudiée ici. Le rôle de la rétine est donc de fournir une image prétraitée au cerveau, ces prétraitements de bas et moyen niveaux étant réalisés à proximité du pixel et en temps réel. Le premier enjeu des rétines artificielles est donc d'*effectuer des calculs de moyen-bas niveau à grande vitesse*, au minimum temps réel, au sein même du pixel.
Ceci étant, il n'est pas nécessaire d'avoir à disposition une rétine pour réaliser des

traitements en temps réel. Une simple caméra associée à un processeur de type DSP (Digital Signal Processing), ou même à un ordinateur puissant permet d'effectuer des opérations de traitement d'images temps réel. Cet enjeu n'est donc pas le seul enjeu majeur des rétines artificielles.

Faible consommation d'énergie

Revenons au fonctionnement de la rétine biologique. Elle est le fruit d'un processus évolutionniste qui l'a façonnée au fil du temps. Ce processus a conduit à une rétine capable d'effectuer des calculs temps réels, mais également comme pour beaucoup de fonctions biologiques, à effectuer les calculs pour un coût énergétique faible. Par analogie, la *faible consommation d'énergie* est un deuxième enjeu fondamental des rétines artificielles.
Si l'on considère le couple vitesse de traitement élevée et faible consommation d'énergie, alors les système évoqués précédemment tels que le traitement d'image avec une caméra couplée à un DSP ou à un ordinateur sont beaucoup moins compétitifs. Des arguments relatifs à la consommation d'énergie dans les dispositifs de traitement d'image seront développés à ce sujet. Toutefois, ce manque de compétitivité sur le plan du couple vitesse-énergie est compensé par une plus grande diversité des opérations pouvant être effectuées. Dans cette confrontation, les rétines artificielles souffraient de leur incapacité à implanter les opérateurs régionaux, ce qui restreint leur champ d'utilisation aux opérations locales dites de bas niveau, alors que des opérateurs génériques tels qu'un microprocesseur peuvent bien évidemment gérer des structures complexes telles que les régions. Pour cette raison, cette thèse propose une implantation d'opérateurs régionaux génériques au sein des rétines artificielles. En outre, afin de conserver l'avantage existant des rétines artificielles au niveau de l'énergie consommée, il est impératif que ces opérateurs régionaux aient une consommation d'énergie très faible.

Implantation intra-pixellique

La faible consommation d'énergie dans les rétines artificielles a une explication. Les fonctions de traitement de bas niveau synchrone SIMD sont placées dans le pixel, très près de l'endroit où s'effectue l'acquisition des données. Ceci permet d'éviter des transferts massifs de données sur de longues distances, transferts qui peuvent se révéler très onéreux en consommation d'énergie en raison des capacités des bus de transmission. De plus, ces traitements de données au sein du pixel permettent de ne pas avoir un goulot d'étranglement au niveau des entrées sorties du capteur comme sur une caméra. Toutefois, avec les technologies actuelles, ce goulot d'étranglement n'est un problème que dans le cas d'applications fonctionnant à vitesse très élevée.

1.2.2 Des contraintes : rendre possible une implantation de forte densité

L'intérêt d'une implantation pixellique d'opérateurs de traitement d'image est de réduire la consommation d'énergie du circuit. Son inconvénient principal est en revanche d'augmenter la taille du pixel, par ajout des fonctions à l'intérieur de celui-ci. Cette augmentation de la taille du pixel se traduit, si celle-ci est trop importante par une diminution de la densité des pixels implantés sur le silicium. De plus, pour des raisons de coût de fabrication, il est nécessaire de limiter la taille globale du capteur, ce qui conduit à diminuer le nombre de pixels de l'imageur si l'on ajoute des fonctionnalités à chaque pixel. Afin d'avoir un circuit le plus dense possible, il est donc nécessaire de réduire au maximum les ressources matérielles utilisées à l'intérieur de la rétine. Au vu de la complexité des autres ressources pixelliques (mémorisation, calcul, communication), nous avons considéré un budget maximal de 50 transistors pour implanter des fonctions supplémentaires telles que les opérateurs régionaux.

Cette limitation très restrictive nous a conduit dans cette thèse à discuter de l'importance de chacun des opérateurs régionaux qu'il est possible d'utiliser, et à proposer un nouvel opérateur régional répondant aux contraintes posées.

2

Tour d'horizon des rétines artificielles

Après avoir présenté les enjeux et contraintes du traitement d'image sur rétines artificielles, il est à présent opportun de faire un tour d'horizon des implantations existant dans la littérature des rétines artificielles et autres circuits programmables de type tableau d'imageurs ayant un processeur élémentaire associé. Cet état de l'art s'inspire de l'étude très complète et détaillée effectuée dans [Ber98a] et dans la thèse de Fabrice Paillet [Pai01]. Cette étude sera complétée par quelques circuits ayant vu le jour entre 2001 et 2005. Les différents types de circuits présentés sont les circuits analogiques dédiés, les circuits analogiques programmables, et enfin les circuits numériques polyvalents.

2.1 Circuits analogiques dédiés

Le modèle biologique ayant inspiré la philosophie des rétines artificielles a également inspiré des solutions quant à leur implantation physique. Ces solutions, à l'instar des neurones sont principalement de type analogique. Les projets de rétines utilisant les circuits analogiques dédiés ont été les plus étudiés par la communauté. Ces projets se sont établis en tant que domaine de recherche international [Mea89] [Del93] à la fin des années 80 (détection des discontinuités spatiales et temporelles [DeW92], détection des contours [AB96], détection et analyse du mouvement [HKLM88] [Moi97] [ASB+96] [TH99], stéréovision [Mah94],...) dans des laboratoires aussi prestigieux que le CalTech, le MIT et l'Université John Hopkins aux USA, l'Université Adelaide en Australie, le Centre Suisse d'Électronique et de Microtechnique de Neufchatel en Suisse, l'Université de Seville en Espagne, l'Institut d'Électronique Fondamentale de l'université Paris XI Orsay et l'INPG en France. La neurophysiologie de la vision animale a constitué une source d'inspiration importante pour la majeure partie des conceptions de circuits qui ont été publiées. Ces types de circuits exploitent les propriétés physiques du transistor CMOS et du silicium qui le compose, offrant une grande richesse en utilisant ses différents modes (faible/forte inversion). On obtient alors un excellent rapport vitesse sur

consommation par surface.

Il y a cependant des limitations importantes au traitement analogique de l'information dans le pixel. Deux transistors géométriquement identiques sur un même circuit intégré n'ont pas exactement les mêmes caractéristiques de fonctionnement (même s'ils sont voisins) en raison des inhomogénéités du processus technologique de fabrication. Cela génère un bruit dit spatial fixe sur l'ensemble de la matrice. D'autre part, avec des courants permanents dans chaque pixel, les contraintes relatives à la consommation de puissance incitent à utiliser les transistors en faible inversion. Pour cela, il faut trouver les solutions techniques spécifiques au niveau du circuit ou du système, avec plus ou moins de succès.

La plupart des opérateurs analogiques pour la vision artificielle semblent avoir une structure assez simple par rapport aux opérations qu'ils sont capables de réaliser. Cela provient surtout de la forte régularité des interactions implantées. Par exemple, des interactions qui décroissent avec la distance ou avec le temps se prêtent assez naturellement à l'exploitation de transistors sous la forme de composants passifs, permettant des implantations très compactes. Moins de régularité se traduit généralement par une hausse sensible de la consommation en surface. Il semble que l'irrégularité devient prédominante avec l'émergence des considérations de forme. Or ceci se produit dès que l'on passe de la notion de discontinuité spatiale à la notion de contour. Une solution à ce problème est venue avec l'utilisation de boucle de rétroaction et la mise en place de système adaptatif permettant l'auto-calibration et l'auto-compensation [HKS+90].

Une autre source de consommation de surface réside dans les spécificités fonctionnelles et de mise en oeuvre des opérateurs élémentaires. Lorsqu'on les combine, il est assez rare de pouvoir mettre en commun des transistors, et il faut même en rajouter chaque fois que le couplage de deux opérateurs n'est pas possible directement en raison de l'utilisation de grandeurs physiques distinctes ou de plages de variations différentes. Il en résulte une surface occupée qui est plus qu'additive par rapport à celle de ses constituants fonctionnels. La spécificité est également un obstacle à la réalisation de systèmes polyvalents : le jeu de commande des opérateurs se limite souvent à quelque(s) constante(s) d'espace ou de temps. Choisir les bons opérateurs analogiques, les interfacer, en imaginer de nouveaux pour de meilleures performances en surface et en puissance consommée, tous ces aspects font que la conception analogique reste très délicate. Finalement, l'investissement n'est rentable que pour des structures d'une certaine universalité, seule la production de masse pouvant justifier de conceptions plus spécifiques.

2.2 Circuits analogiques programmables

Devant le coût de réalisation des circuits, et devant les limitations en terme de possibilités algorithmiques dues à l'utilisation de circuits analogiques dédiés, sont

apparues les rétines et circuits de type analogique programmable. Les réseaux de neurones cellulaires en sont le plus fameux et prolifique représentant [RR00].

2.2.1 Réseaux de neurones cellulaires

Un réseau de neurones cellulaires (CNN pour Cellular Neural Network) est un modèle de calcul massivement parallèle. Les CNN sont une évolution du modèle des automates cellulaires. Dans [CR93], la définition des réseaux de neurones cellulaires proposée par Leon O. Chua et Tamás Roska est la suivante. Un réseau de neurone cellulaire (CNN) est :

- un tableau à 2, 3, ou n dimensions.
- il est constitué de systèmes dynamiques quasi identiques appelés *cellules*, qui satisfont à deux propriétés principales :
- les interactions sont locales, dans un voisinage de rayon r.
- toutes les variables d'état et paramètres sont des signaux à valeurs continues.

Les interactions entre les cellules et leurs voisines sont définies par un modèle de type matriciel définissant l'évolution de l'état des variables d'entrée u et de sortie y et de la variable interne x. Les variables d'état sont continues dans le temps, cependant une cellule est identifiée par ses coordonnées dans le tableau, ce qui revient à discrétiser les variables spatiales. De même le temps peut être continu ou discret. Les interconnexions dans le modèle des réseaux de neurones cellulaires peuvent s'effectuer à partir des variables u, y et x des voisins situés dans un voisinage de rayon r. Les coefficients des matrices représentant ces interconnexions peuvent varier dans le temps pour faire apparaître un terme de délai. L'évolution dynamique du système est régie par une loi d'évolution qui peut être de type équation aux différences partielles ou encore équation fonctionnelle. Cette loi permet de déterminer l'évolution de la variable interne $x(t)$ connaissant l'entrée de la cellule $u(t)$ et les conditions initiales.

La caractéristique principale d'un réseau de neurones cellulaires est la localité des interactions entre les cellules. Ce mode de fonctionnement permet toutefois d'obtenir des calculs régionaux par propagation. Les réseaux de neurones cellulaires fonctionnent donc typiquement par itérations, ce sont des *réseaux récurrents*.

Architecture et équation d'évolution

L'architecture d'une cellule de réseau de neurones cellulaires est présentée à la figure 2.1. Les équations correspondantes peuvent être du type suivant :

$$C\,\frac{\partial x_{ij}(t)}{\partial t} = -\frac{1}{R}x_{ij}(t) + \sum_{v\ in\ Voisinage} A_v y_v(t) + \sum_{v\ \in\ Voisinage} B_v u_v(t) + I(t)$$

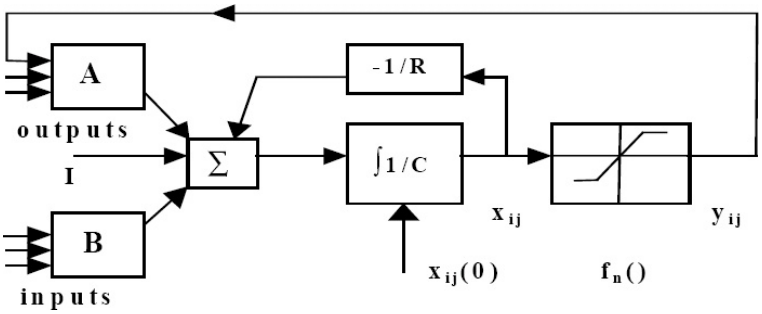


FIG. 2.1 – Architecture d'une cellule de réseau de neurones cellulaires (*source [MAJo99]*)

Dans cette équation, u correspond aux entrées des cellules, x à l'état interne, et y à la sortie des cellules. Comme indiqué sur le schéma de la figure 2.1, la sortie y dépend de manière non linéaire de l'état interne x. L'équation d'évolution fait apparaître les paramètres A_v et B_v. Ces paramètres peuvent être regroupés en matrices A et B. Ces matrices sont des noyaux de convolution. Pour des raisons d'encombrement, ces noyaux de convolution ont le plus souvent un support réduit au plus proche voisinage (noyau centré de taille 3x3). La non-linéarité de la sortie peut être exprimée sous diverses formes [CR93], une des plus simples et des plus utilisées étant la saturation, qui peut s'exprimer sous la forme suivante :

$$y_v(t) = \frac{1}{2}\left(|x_v(t)+1| - |x_v(t)-1|\right)$$

Implantation

Les réseaux de neurones cellulaires forment un modèle adapté au traitement d'image sur les machines massivement parallèles SIMD. Le traitement de l'image d'entrée se fait d'une manière parallèle sur l'ensemble de la matrice par des interactions uniquement locales, mais se répercute globalement sur l'image à travers des phénomènes de propagation de proche en proche.

Plusieurs circuits de vision ont été développés. Des circuits CNNUM (Cellular Neural Network Universal Machine) ont été réalisés pour le traitement de l'image [RR00]. Ce sont des sortes de rétines artificielles analogiques produisant et utilisant des images en niveau de gris. Des implantations plus anciennes telles que les rétines de filtrage spatio-temporel analogiques [MM88] [Mea89] [BG92] constituent une sous-classe des réseaux de neurones cellulaires dans la mesure où leur programmabilité est très limitée. Un exemple de cellule d'un réseau de neurones cellulaires non reconfigurable utilisé pour la détection de composantes connexes est proposé à la figure 2.2.

A l'instar des rétines artificielles analogiques, le modèle des réseaux de neurones cellulaires a un coût d'implantation important. Le circuit correspondant devient rapidement complexe, en particulier si l'on cherche à ajouter des fonctions permettant à l'utilisateur de régler les coefficients A_v et B_v définis avant. En raison

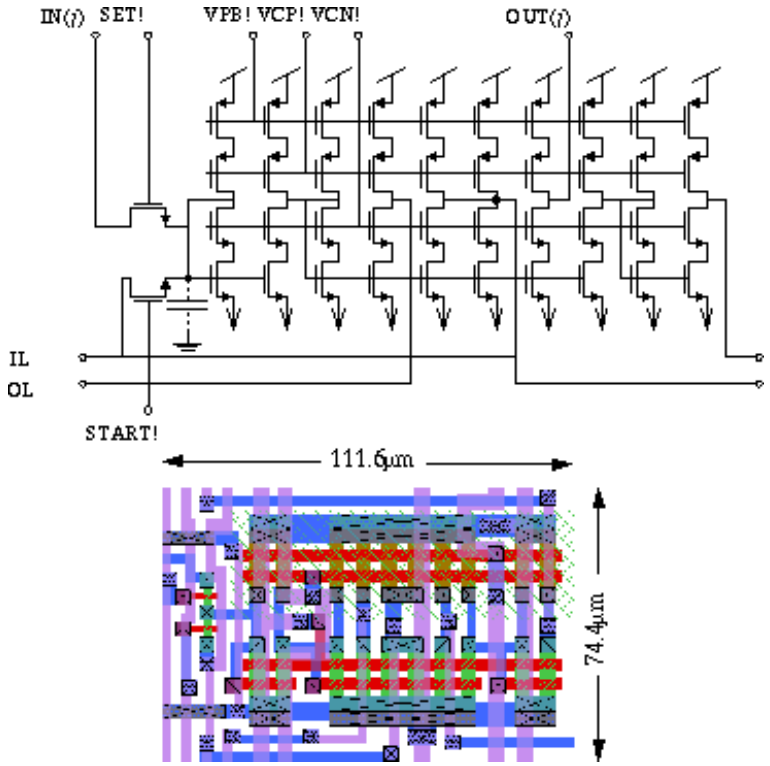


FIG. 2.2 – Schéma électronique et layout d'une cellule d'un réseau de neurones cellulaires permettant la détection de composantes connexes (Université de Séville)

d'un coût d'implantation élevé, certaines applications se sont résignées à utiliser seulement un transistor par coefficient, ce qui limite beaucoup les possibilités de réglage. Finalement, avouant les limitations des solutions analogiques et afin de pouvoir réaliser des rétines de grande résolution, de forte densité et offrant les aspects de programmabilité et de polyvalence nécessaires à l'utilisation dans des applications réelles, les concepteurs commencent à faire intervenir des traitements locaux numériques.

Limitations

Une des limitations principales des réseaux de neurones cellulaires analogiques est le coût engendré par les fonctions destinées à programmer les coefficients des matrices de convolution. Ces fonctions se trouvent le plus souvent hors du pixel, ce qui impose d'extraire l'image de la matrice si l'on souhaite adapter les traitements à effectuer aux données. D'autre part pour effectuer des traitements complexes, il est nécessaire de séquentialiser les traitements effectués au sein des CNN. Ces deux difficultés ont conduit à introduire dans la structure asynchrone des réseaux de neurones cellulaires des éléments logiques synchrones de commande comme dans l'implantation réalisée à l'université de Séville [LFE+99]. Cette migration nécessaire mais progressive vers le numérique s'est même étendue aux fonctions de traitement

des CNN dans certains projets [DC97]. Toutefois, le mode de calcul par relaxation des CNN impose d'effectuer les calculs en parallèle sur tous les bits, ce qui conduit à utiliser des opérateurs numériques multi-bits dont le coût devient rapidement très important. Une réalisation de l'Université de Séville [DC97] utilise des opérateurs à 2 bits. L'utilisation d'équations différentielles et de relaxations lorsque les valeurs peuvent prendre 4 états semble toutefois discutable, le modèle des réseaux de neurones cellulaires n'est donc pas adapté à une implantation numérique intégrale.

Du point de vue de cette thèse, la limitation plus fondamentale des réseaux de neurones cellulaires réside dans leur propriété fondatrice. Afin de permettre une implantation du modèle, les interactions ne se font que de manière locale. Ainsi, pour effectuer des opérations régionales, il est nécessaire d'effectuer une succession d'opérations locales itérées, ce qui est très inefficace. Des opérateurs permettant d'effectuer ces calculs régionaux sans itérations seraient préférables. Ils sont d'ailleurs présents de manière théorique dans d'autres modèles de réseaux de neurones. Toutefois, l'implantation de ces opérateurs régionaux requiert un investissement matériel trop important, ce qui a conduit à leur éviction dans le modèle des réseaux de neurones cellulaires. Cette limitation due au mode de communication local se retrouve dans les rétines numériques synchrones à voisinage limité où les communications s'effectuent de proche en proche et dans les méthodes de traitement d'image par équations aux dérivées partielles, qui sont d'ailleurs très bien implantées à l'aide des réseaux de neurones cellulaires.

2.3 Circuits mixtes analogiques numériques

Comme nous l'avons constaté précédemment avec les réseaux de neurones cellulaires, les circuits analogiques ont un point faible, ils ne sont pas simplement reconfigurables. Afin d'éviter ce problème, certaines équipes ont eu l'idée de tirer parti des avantages du numérique en terme de facilité de pilotage, de séquentialisation et de programmabilité et des avantages de l'analogique en terme de fonctionnalité. Parmi ces circuits, nous en citons deux principaux.

2.3.1 Les microprocesseurs génériques

Deux projets récents de circuits basés sur la réalisation de processeurs analogiques génériques ont vu le jour avec des approches assez différentes : l'un s'intéresse à l'intégration d'un processeur comparable à un processeur numérique (Université de Manchester), l'autre préconise l'implantation de processeurs dotés à la fois d'une unité de calculs analogique et numérique (projet PARIS de l'Institut d'Électronique Fondamentale de l'Université Paris XI Orsay), en bordure du capteur.

Le microprocesseur générique analogique $A\mu P$

Le concept mis en oeuvre à Manchester est celui d'un microprocesseur générique analogique (appelé $A\mu P$) travaillant sur des données analogiques échantillon-

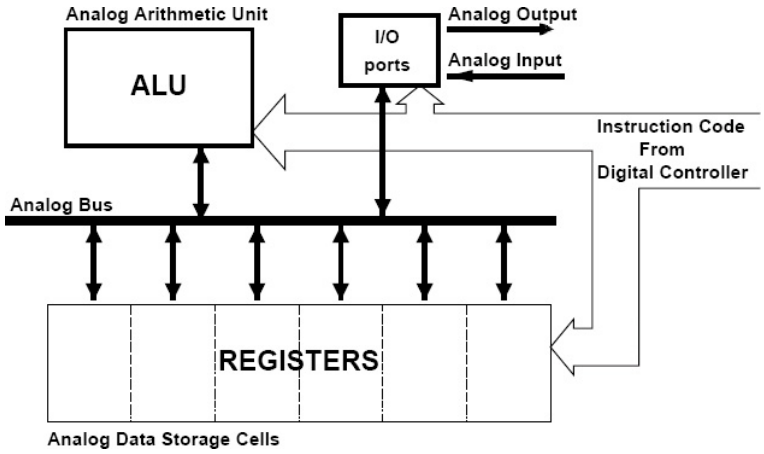


FIG. 2.3 – Schéma bloc du processeur analogique des microprocesseurs génériques

nées et disposant d'un contrôle numérique. Le $A\mu P$ exécute des programmes de la même manière que le ferait un microprocesseur numérique, mais en manipulant des données analogiques échantillonnées dans ses unités de calcul et sur ses chemins de données au lieu de données binaires codées sur plusieurs bits. Il est capable, en exécutant de manière séquentielle les instructions qui lui sont envoyées par le contrôleur externe d'échanger des données analogiques entre ses registres (copies) et à travers ses ports entrées/sorties, à destination de voisins et d'effectuer des calculs sur celles-ci, y compris des comparaisons et des branchements conditionnels.

La figure 2.3 présente l'architecture globale du microprocesseur sous la forme d'un diagramme fonctionnel. Elle ressemble fortement à celle d'un processeur numérique. Le processeur s'articule autour d'un bus analogique qui sert à interconnecter un banc de registres analogiques, une unité arithmétique analogique et une unité de communication (entrées et sorties analogiques), chacun étant contrôlé par un ensemble de signaux numériques jouant le rôle d'instructions, envoyés par un contrôleur externe. Les registres mémoire du processeur sont capables de stocker une donnée analogique avec une précision fixée sur une durée de l'ordre du dixième de seconde.
Le $A\mu P$ est capable d'exécuter séquentiellement (avec une synchronisation externe de type numérique) des échanges de données analogiques avec les voisins et des calculs sur ces données analogiques.

L'ensemble forme ainsi un système mixte analogique/numérique capable d'opérer en mode SIMD. Le principe de fonctionnement du processeur $A\mu P$ repose sur les techniques dites de commutation de courants. La solution technique sous-jacente est compacte et performante, dans la mesure où sommation ou soustraction de courants sont réalisées facilement grâce à la loi des noeuds. Cependant, le stockage des

données dans les registres présente une perte de mémoire. Lorsque des algorithmes récursifs sont exécutés, l'erreur accumulée à chaque transfert de données risque de mettre en péril la viabilité de l'algorithme. En terme de compacité, les microprocesseurs génériques sont de taille comparable aux CNN. En terme de performance, les fréquences sont faibles. Enfin en matière de consommation d'énergie, la nécessité d'utiliser des transistors dans leur zone de saturation rend les dépenses énergétiques importantes, ce qui exclut l'usage de ce type de circuit dans des applications nomades.

Le projet PARIS

Le projet PARIS de l'Institut d'Électronique Fondamentale à l'université de Paris XI fait appel à des solutions différentes de celles exposées ci-dessus dans le cadre du processeur analogique générique sur données échantillonnées. L'objectif du projet est la conception d'une nouvelle génération de rétines programmables utilisant un processeur doté d'une unité de calcul analogique et d'une unité de calcul booléen. La complexité du processeur envisagé ne permet pas son intégration dans chaque cellule de la matrice de capteurs. Le processeur booléen est donc intégré en bordure de matrice, sous la forme d'une ligne de processeurs SIMD. La matrice de capteurs joue un rôle de mémoire analogique : chaque cellule contient quatre capacités de stockage de données analogiques, l'une d'entre elles servant à intégrer le courant en provenance du photorécepteur. Le processeur, en bordure, est connecté aux bus mixtes analogiques/numériques de 3 colonnes successives, à travers un multiplexeur 3 vers 1, afin de pouvoir effectuer facilement des calculs sur un voisinage local. Il se compose d'une unité de calcul analogique qui exécute des instructions sous le contrôle d'une unité logique et d'un banc de registres mémoire mixtes analogiques/numériques.

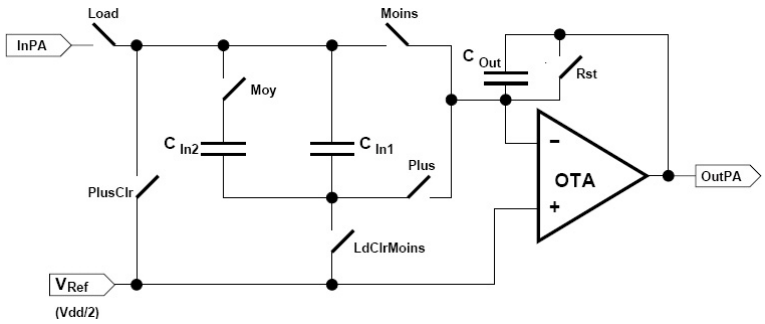


FIG. 2.4 – Schéma bloc du processeur analogique de la rétine PARIS

L'architecture de l'unité de calcul analogique (fig. 2.4) repose sur l'usage de capacités commutées de bonne précision permettant grâce à un jeu d'interrupteurs MOS bien choisi d'effectuer un grand nombre d'opérations arithmétiques. Toutefois, l'usage de ces capacités requiert une implantation sur silicium très spécifique.

La précision du système est en effet conditionnée par la précision des capacités. Celles-ci doivent donc être protégées des perturbations (par ajout d'un caisson spécifique par exemple), et de grande taille afin d'être suffisamment précises.

Ces contraintes conduisent à une taille de processeur importante qui limite la densité d'implantation et à des temps d'accès mémoire importants (du fait des grandes capacités). D'autre part, l'usage d'un processeur synchrone par colonne conduit à diminuer l'efficacité des traitements synchrones si le nombre de processeurs de chaque colonne augmente trop. Ce type d'implantation semble donc présenter des difficultés dès lors que l'on souhaite augmenter le nombre de pixels de l'imageur.

2.3.2 Limitations

Dans les deux microprocesseurs génériques considérés, l'analyse des points faibles met en évidence les problème engendrés par les fonctionnalités analogiques. L'utilisation de capacités précises conduit à une vitesse faible et un encombrement élevé. Finalement, on peut se poser légitimement la question de la pertinence de l'utilisation de l'analogique dans les rétines artificielles. Celui-ci ne serait-il pas un héritage idéologique né avec l'analogie rétinienne biologique ? Malgré cet héritage, l'intérêt du numérique s'est néanmoins imposé dans le monde des rétines analogiques.

2.4 Circuits numériques programmables

Par rapport à la conception de circuits analogiques, les méthodes de conception numériques disposent d'outils performants de vérification et de synthèse, conduisant à des circuits compacts et optimisés. Cet avantage ajouté aux capacités de reconfigurabilité, et de polyvalence des circuits numériques ont conduit plusieurs équipes de recherche à abandonner l'analogie biologique sur laquelle repose les circuits analogiques au profit du pragmatisme offert par les circuits numériques.

Toutefois, cette migration vers le numérique ne se fait pas sans difficultés, la première étant que le signal transmis par le photorécepteur est de type analogique. Il est donc nécessaire de le convertir en signal numérique au sein même du pixel ou au minimum dans le plan focal. Cette conversion peut être effectuée sous diverses formes, et elle a été depuis longtemps et est toujours aujourd'hui au coeur des préoccupations de certaines équipes de recherche [PF94] [RJL96] [YFG99].

Après avoir payé le coût d'entrée imposé par la conversion analogique numérique au sein du pixel, la structure des circuits numériques programmables massivement parallèles est assez cohérente d'une implantation à l'autre. Cette partie propose de les comparer.

2.4.1 Le système de vision dit "milliseconde" de l'Université de Tokyo

Le système de vision milliseconde de l'Université de Tokyo au Japon implante l'unité de traitement numérique au niveau du pixel. Le parallélisme obtenu est comparable au parallélisme des rétines artificielles de type *PVLSAR* élaborées à l'ENSTA. L'objectif de ce projet est la réalisation d'un système de vision rapide pour des applications en robotique. La complexité de l'élément de calcul introduit dans chaque pixel est limitée. Toutefois, elle reste inspirée par une structure conventionnelle de processeur, bien que fortement dépouillée. Le processeur intégré ici reçoit des instructions sur 5 bits qui sont décodées dans le pixel lui-même. Ainsi le pixel se compose d'une photodiode, comme élément photorécepteur et du PE constitué d'un banc de mémoire SRAM de 24 bits et d'une unité arithmétique et logique réalisée à partir d'un Full Adder auquel sont connectés deux verrous en entrée et un verrou en sortie. Les communications sont gérées par *mapping* mémoire (des adresses spéciales sont affectées pour récupérer la donnée d'un processeur dans le plus proche voisinage Nord-Est-Ouest-Sud ou la lecture seuillée du capteur). La lecture du capteur et la conversion analogique numérique est obtenue par simple seuillage.

2.4.2 Le circuit NSIP de l'Université de Linkoping.

Le concept NSIP (Near Sensor Image Processing) [JEE96] [ARF96] est une généralisation aux images en niveaux de gris d'une rétine permettant d'effectuer des traitements sur des données binaires correspondant au seuillage d'une image. Le traitement en niveau de gris s'effectue en ne manipulant que des données binaires en sortie du capteur, et cela, sans stockage systématique du niveau de gris sous forme explicite. Ce concept d'encodage temporel des niveaux de gris repose sur la capacité d'interroger de manière successive et périodique l'état du photorécepteur sans le perturber. Ce sont les coupes de l'ombre de l'image qui sont alors obtenues et peuvent être utilisées pour effectuer des traitements en niveaux de gris (suivant le même formalisme que celui qui a permis d'étendre le domaine des traitements de la morphologie mathématique, initialement uniquement binaire, aux niveaux de gris).

La rétine NSIP [JEE96] est constituée d'une matrice de processeurs élémentaires commandés en mode SIMD. La matrice est organisée en tableau de processeurs connectés entre eux. Le signal provenant du photorécepteur (photodiode fonctionnant en intégration) peut être seuillé et stocké dans un des huit registres binaires que comprend le processeur. L'unité de traitement du processeur élémentaire se compose de trois unités distinctes par lesquelles cheminent les données et au travers desquelles elle peut subir un traitement. Les trois unités sont de nature différente, permettant de réaliser divers traitements (fig. 2.5) :

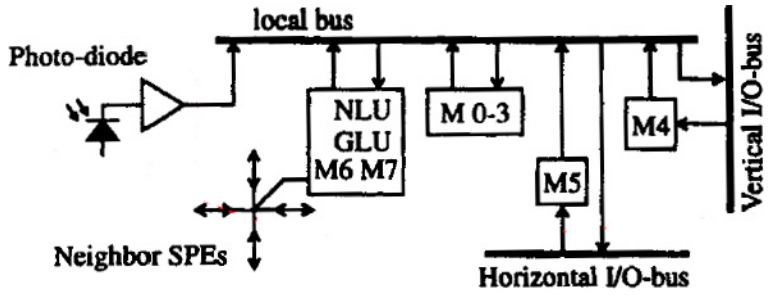


FIG. 2.5 – Schéma bloc du processeur élémentaire de la rétine NSIP (Near Sensor Image Processing)

- La GLU (Global Logical Unit) met en oeuvre des propagations sur l'image, dans les 4 directions Nord Est Ouest Sud (NEWS).

- la NLU (Neighbor Logical Unit) réalise des traitements locaux impliquant les voisins dans la matrice.

- le bus local peut être utilisé en tant qu'opérateur *Non-Et* logique câblé. Par décomposition booléenne, on peut donc effectuer toutes les fonctions logiques à l'aide de cet opérateur. L'ensemble du circuit est contrôlé par une machine extérieure, chargée du séquencement des instructions.

Le processeur élémentaire effectue les opérations en niveaux de gris de manière bit série.

L'approche et la démarche suivies par l'équipe de l'université de Linkoping, dans le développement de leur rétine NSIP est celle qui se rapproche la plus de la rétine PVLSAR. Elles tiennent compte dès l'élaboration de l'architecture des contraintes sévères en terme de surface et de consommation, avec pour objectif le meilleur compromis entre coût en silicium, efficacité et polyvalence.

2.4.3 Les Rétines TCL

Issu des travaux de recherche sur l'algorithme TCL (Traitements Combinatoires Locaux) [Zav81], menés depuis le début des années 80 au Centre Technique d'Arcueil et à l'Institut d'Électronique Fondamentale, un circuit appelé la rétine TCL a été réalisé en 1984 par P.Garda et F.Devos. La classe des traitements TCL découle logiquement d'une observation des caractéristiques d'invariance par translation et de localité des traitements d'image de bas niveau. Le processus d'acquisition est homogène et se déroule simultanément de manière équivalente et en parallèle sur chaque pixel. Les traitements réalisés sur les pixels acquis sont indépendants de leur position dans l'image (invariance par translation). Un traitement combinatoire local (TCL) correspond au calcul d'une fonction booléenne, dont l'application à chaque

pixel entraîne le remplacement de la valeur du pixel par une combinaison logique de sa valeur courante et de celle de certains de ses voisins. Les transformées en tout ou rien de la morphologie mathématique sont des TCL. Disposer d'une architecture massivement parallèle SIMD avec un élément de calcul par pixel, permet l'exécution d'un TCL sur tous les pixels d'une image simultanément.
Par mise sous forme disjonctive de la fonction booléenne équivalente, un TCL nécessite peu d'opérateurs pour s'implanter : 3 bits de mémorisation, un inverseur, un ET logique et un OU logique. Par ailleurs, pour implanter un TCL dans une architecture SIMD où une unité de calcul est présente dans chaque pixel, il est nécessaire, afin d'aller chercher les différentes variables, de faire subir un décalage à l'image binaire traitée. Ceci est facilement réalisable en utilisant une structure de registre à décalage semi-statique. L'image peut ainsi être stockée et déplacée.

Dans la poursuite des objectifs du projet de rétine TCL, une deuxième génération de ces rétines a été mise au point au centre Technique d'Arcueil de la Direction Générale pour l'Armement. Partant du constat que le nombre de signaux d'horloge contrôlant le processeur TCL de la première génération limite sa densité d'intégration, les développements s'orientent vers une uniformisation du rôle des bits de mémorisation et la mise en commun d'un seul et même opérateur de calcul.
La conception de ce circuit (fig. 2.6) est le résultat des travaux de thèse de T.Bernard

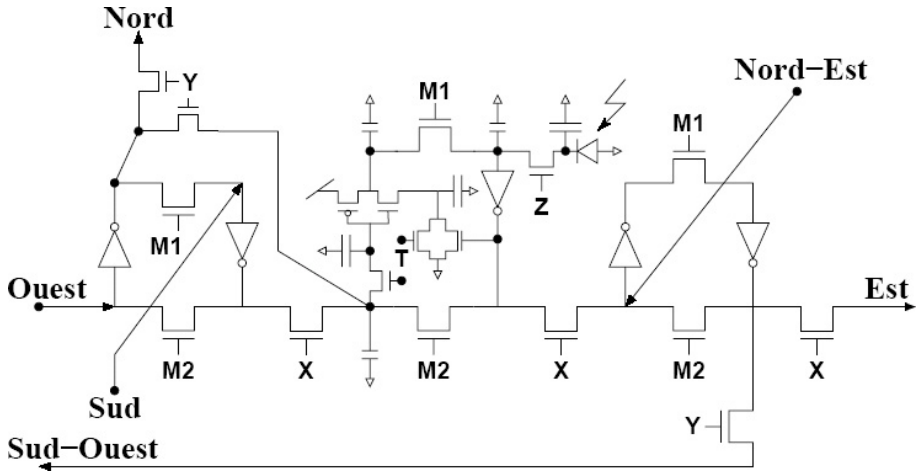


FIG. 2.6 – Schéma électronique du processeur élémentaire de la rétine TCL de deuxième génération

[Ber92]. Les efforts de minimisation du nombre de signaux nécessaires au contrôle du processeur TCL booléen portent leurs fruits. Les 3 bits de mémoire dont dispose celui-ci font maintenant partie d'un seul et unique registre à décalage semi-statique bidimensionnel et tridirectionnel. Cependant, il en résulte une lourdeur de programmation et une inefficacité tant énergétique qu'algorithmique, en raison du grand nombre de déplacements élémentaires à réaliser pour obtenir un décalage quelconque souhaité, et du fait que tous les bits de mémorisation de la matrice

soient systématiquement déplacés. Sa mise en oeuvre, son test et son exploitation furent l'objet des travaux de nature système dans les années qui ont suivi [Man00] [MBPL99] [Ber98b] [NMJB97]. La période 1993-1997 a permis de s'intéresser aux aspects système dans la conception de machines de vision à base de *Rétines Artificielles Numériques Programmables* (RANP).

F. Paillet [Pai01] [PMB99] [PMB98] a contribué à la conception et la mise en

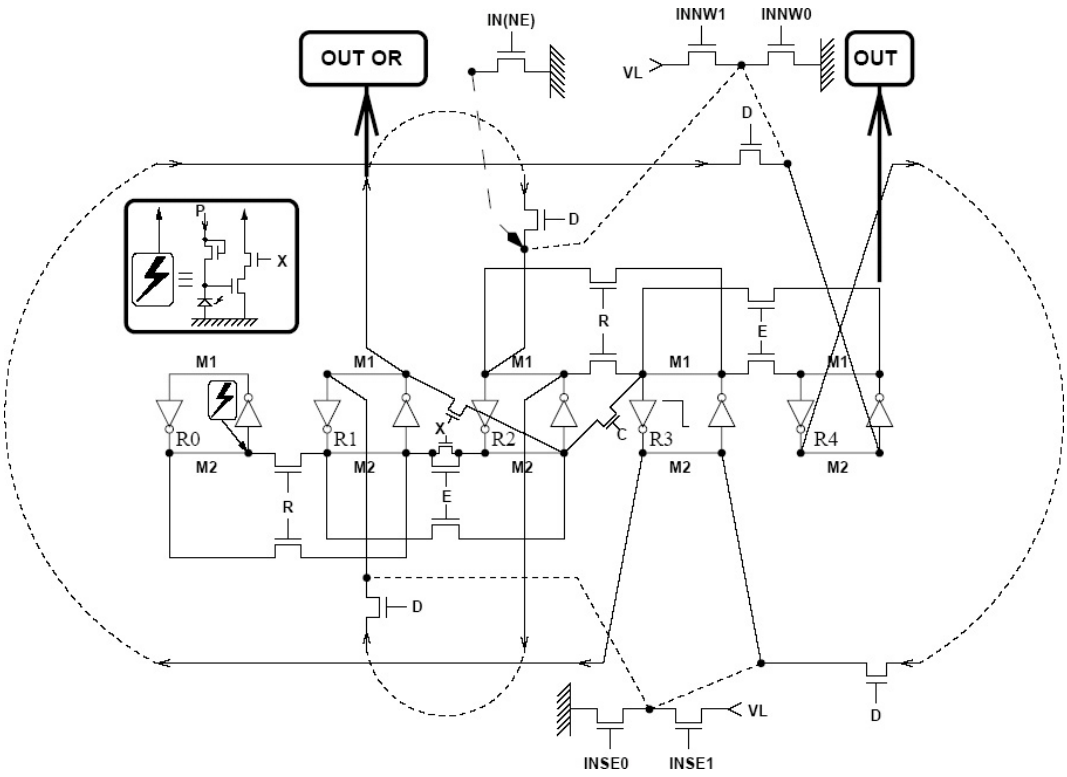


FIG. 2.7 – Schéma électronique du processeur élémentaire de la rétine Pvlsar 2.2

oeuvre d'une rétine de dimension plus importante appelée Pvlsar 2.2 (*Programmable and Versatile Large Size Artificial Retina*). Cette rétine est constituée d'une matrice de 128∗128 pixels, chacun ayant 5 bits mémoire. La structure du processeur élémentaire (fig. 2.7) reprend celle de la rétine TCL de deuxième génération (fig. 2.6) avec certaines améliorations visant notamment à permettre la multigranularité des traitements et permettant de manipuler les données plus efficacement et pour un coût plus réduit.

Les rétines TCL de deuxième génération puis Pvlsar 2.2 ont permis d'avoir un banc de test très significatif pour l'élaboration d'algorithmes de traitement d'image sur les rétines artificielles numériques programmables. Le développement de ces traitements a fait l'objet de deux thèses à l'ENSTA, la thèse d'A. Manzanera [Man00] de 1997 à 2000 puis la thèse de J. Richefeu [Ric06] de 2002 à 2005.

2.4.4 Les rétines numériques récentes

Depuis 2001 et l'arrivée de la rétine Pvlsar 2.2, divers travaux visant à augmenter la taille, la résolution et la capacité des rétines artificielles ont été proposés. Nous pouvons citer en particulier les travaux de Komuro [KKI04] à l'Université de Tokyo qui ont abouti à l'élaboration en 2004 d'une rétine de taille modeste (64*64) ayant une capacité mémoire de 24 bits par pixel. Outre sa capacité mémoire importante, cette rétine présente également une fonctionnalité novatrice. Il s'agit d'un réseau de calcul combinatoire permettant d'effectuer des sommations globales sur l'ensemble de la rétine. Il est de plus possible moyennant certaines limitations d'effectuer ces sommations sur des blocs de pixel (dont la forme n'est cependant pas quelconque). Cette fonctionnalité est un premier pas vers la régionalisation dans les rétines et permet d'envisager l'élargissement du champ des opérateurs de traitement d'image rétiniens aux fonctions régionales telles qu'une somme distribuée sur une région de taille et de forme variable.
Les possibilités d'utilisation de ce réseau sont pour l'instant relativement limitées en raison de la forme linéaire du réseau utilisé pour couvrir la région, mais la volonté d'aller vers un niveau de traitement d'image supérieur est bel et bien présente.

Les potentialités de ce type d'opérateurs régionaux étant encore mal connues, l'objectif principal de cette thèse est d'explorer le champ des opérateurs régionaux implantables dans les rétines et leurs applications. Dans ce contexte, nous reviendrons en détail sur l'architecture asynchrone proposée par Komuro [KKI04].

Enfin, nous terminons cet état de l'art en citant la rétine Pvlsar 34 (fig. 2.8), dernière née de la famille des rétines PVLSAR, développée à l'ENSTA par Thierry Bernard mi-2004. Il s'agit d'une rétine $200 * 200$, en technologie 0.35 μm, chaque pixel ayant une taille de 37.5 μm de coté et une capacité mémoire de 44 bits. Cette rétine opérationnelle constitue aujourd'hui la plus grande jamais réalisée. D'autre part, sa capacité mémoire pixellique lui confère une polyvalence algorithmique sans précédent.

2.4.5 Comparaison des circuits numériques programmables

En conclusion de ce chapitre, une comparaison des rétines numériques programmables existantes est proposée. Cette comparaison en terme de nombre de pixels, de capacité mémoire, de taille de pixel, en fonction du caractère récent ou non du projet est proposée à la figure 2.9.
Sur cette figure, la taille des rectangles correspondant à chacun des circuits considérés est proportionnelle à celle du pixel correspondant dans le circuit. La densité de rouge de chaque rectangle correspond au caractère récent du projet. Un projet récent étant indiqué en rouge vif, un projet ancien étant indiqué en rouge sombre. On

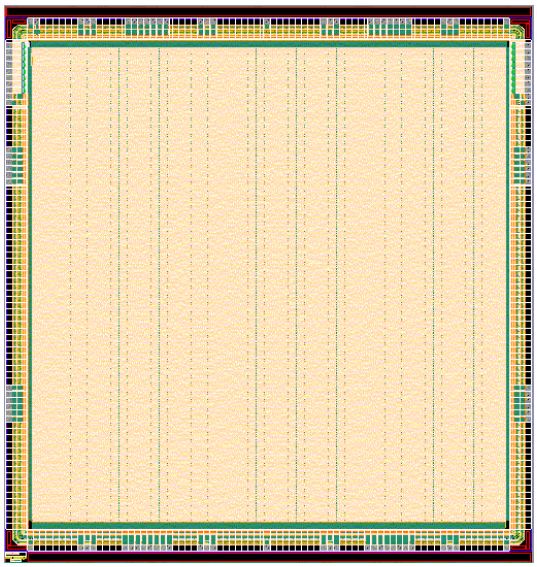


FIG. 2.8 – Rétine Pvlsar 34

voit sur ce diagramme que la rétine Pvlsar 34 représente un bond important en matière de capacité mémoire et de taille de la matrice. Toutefois, il ne faut pas oublier que la dernière rétine de l'Université de Tokyo [KKI04] intègre des fonctionnalités avancées permettant d'effectuer des opérations régionales. Ces fonctionnalités régionales ne sont pas représentées sur ce diagramme.

2.5 Limitations des rétines artificielles existantes

Nous avons vu dans l'état de l'art que la tendance générale de l'évolution des rétines artificielles est d'aller vers une plus grande souplesse d'utilisation, une plus grande résolution du capteur, une plus faible consommation et une plus grande capacité de calcul. Ces évolutions sont d'ordre quantitatif dans la mesure où elles améliorent les capacités existantes des rétines, à l'exception de l'augmentation de la mémoire qui permet d'effectuer des algorithmes plus complexes.

Les évolutions quantitatives des capacités des rétines artificielles sont intéressantes d'un point de vue fonctionnel et applicatif [Pai01] [Man00] [Ric06], mais ne permettent cependant pas de s'affranchir de limitations plus fondamentales liées à l'architecture utilisée. Une de ces limitations fondamentales est la restriction des fonctionnalités des rétines au traitement de bas niveau.

2.5.1 Limitation à la vision de niveau peu élevé

Comme présenté dans la thèse d'Antoine Manzanera [Man00], la complexité des opérations qu'il est possible d'effectuer dans une rétine artificielle est limitée par

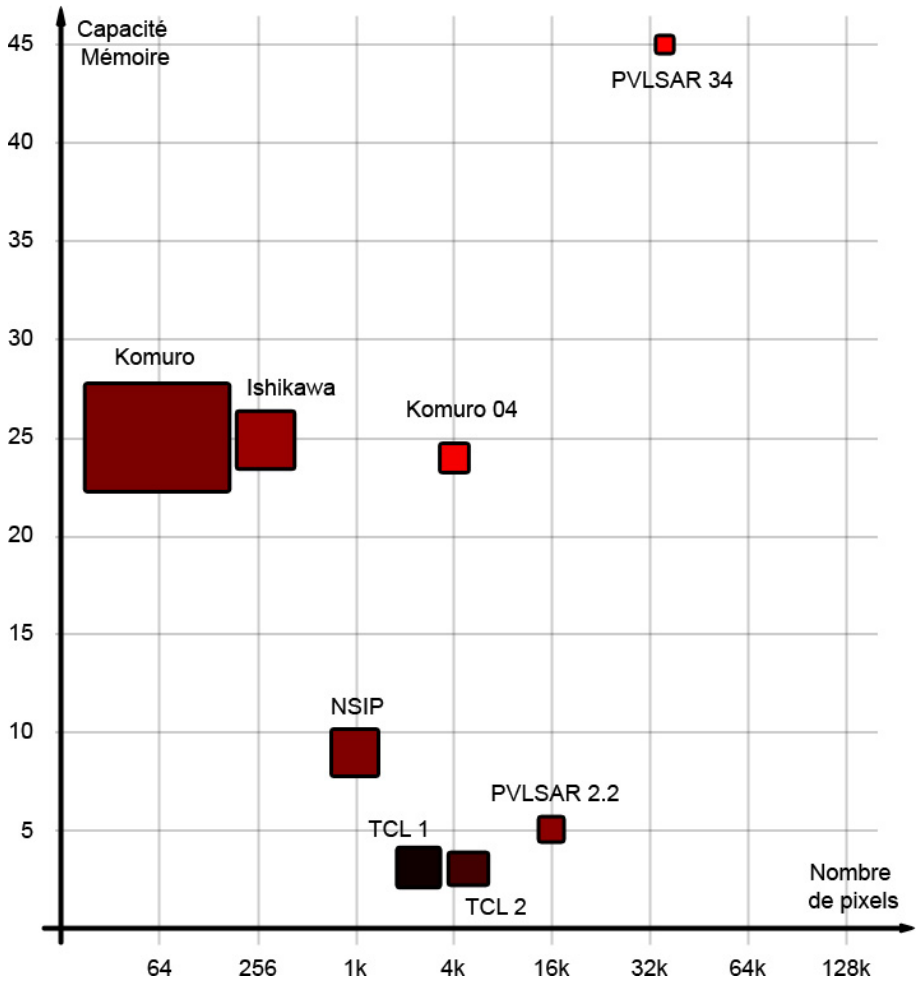


FIG. 2.9 – Comparaison des rétines numériques en 2005. Les plus récentes apparaissent en rouge vif.

le nombre de bits mémoire disponibles dans chacun des processeurs élémentaires de cette rétine. Cette complexité est en effet liée au nombre de données binaires qu'il est nécessaire de placer dans un même pixel pour effectuer des opérations locales dessus. Une grande capacité mémoire permet donc d'effectuer des opérations complexes. Ceci est exact d'un point de vue fonctionnel, mais ne donne en revanche aucune indication sur le temps et l'énergie nécessaire à ce type de calcul. Ainsi par dilations itérées, l'exemple de la reconstruction géodésique (propagation d'un marqueur à une zone connexe), peut être implanté de manière synchrone avec un nombre très limité de registres mémoire (3 registres suffisent). Toutefois, comme indiqué dans [Man00], une telle implantation n'est pas efficace d'un point de vue temporel car elle nécessite d'effectuer un grand nombre d'itérations. Elle est également inefficace énergétiquement puisque peu de processus sont productifs à chaque itération. Ce résultat a été également mis en évidence par B. Galilée [Gal02]

Ces limitations énergétiques et temporelles empêchent d'utiliser efficacement les rétines artificielles pour les opérations de moyen voire haut niveau, même dans le cas où celles-ci sont très simples, comme la reconstruction géodésique.

2.5.2 Evolutions possibles

Dans le cas de la reconstruction géodésique, Antoine Manzanera a suggéré l'intérêt de l'utilisation d'une structure matérielle dédiée à la reconstruction géodésique [Man00]. Pour cette opération, un simple réseau de connexions programmables permettant de lier tous les pixels d'une même région entre eux de manière à former une équipotentielle permet de résoudre le problème. T. Bernard en a proposé une implantation très compacte dans [Ber00]. Toutefois, cette structure manque de généralité dans la mesure où elle ne permet que d'effectuer l'opération de reconstruction géodésique.
Une autre structure dédiée a une fonction régionale bien plus complexe (la ligne de partage des eaux) a été proposée par B. Galilée [Gal02]. Cette architecture asynchrone présente des résultats intéressants en termes de vitesse de calcul, toutefois elle est difficilement utilisable en raison d'un coût matériel trop important pour une opération dédiée.
Une solution permettant d'effectuer une plus grande diversité d'opérations de moyen niveau à l'aide de rétines artificielles a été proposée par Komuro à l'université de Tokyo [KKI04]. Elle sera étudiée au chapitre 6.1. Cette solution présente toutefois des limitations fonctionnelles et des limitations en terme de performance assez importantes.
Une autre structure cellulaire très intéressante permettant d'effectuer des opérations de traitement d'image de moyen niveau est la Maille Associative d'Orsay [Dul96] [Moh96] [DMM95]. Cette structure sera également étudiée au chapitre 6.1. Elle permet d'effectuer une grande diversité d'opérations régionales associatives de manière très efficace. Sa limitation principale est son coût d'implantation important qui ne permet pas d'en faire à l'heure actuelle une rétine artificielle du fait de sa trop faible densité de processeurs par millimètre carré de silicium.

Dans cette thèse, une nouvelle solution matérielle et algorithmique est proposée, permettant d'effectuer des opérations de moyen niveau dans le contexte des rétines artificielles en essayant de repousser le plus possible les limitations algorithmiques des opérations de moyen niveau implantables et tout en essayant de réduire au minimum utile le nombre de transistors utilisés pour l'implantation des fonctionnalités régionales.

3

De l'utilité de la régionalisation

Dans cette partie, nous examinons l'utilité de la régionalisation en traitement d'images. Afin de mieux comprendre les enjeux de celle-ci en terme d'efficacité organisationnelle, de dépense d'énergie et de vitesse de traitement, les similitudes entre le contrôle d'un robot, le fonctionnement d'un État et celui d'une entreprise sont étudiées. Cette étude nous conduit ensuite à rechercher les facteurs communs permettant d'optimiser les échanges informationnels. Nous établissons ensuite un parallèle avec le traitement d'image, qui nous conduit à retrouver le découpage communément accepté du traitement d'image en opérations de bas, moyen et haut niveaux. Nous déterminons ensuite quels sont les types d'opérateurs efficaces d'un point de vue énergétique pour effectuer les différents types de traitements d'image au sein d'une rétine, le rôle fondamental joué par la régionalisation dans l'optimisation des échanges informationnels nous conduisant à attacher une attention toute particulière aux traitements d'image régionaux.

3.1 Analogies

Quel peut être le rapport entre le fonctionnement d'un État, le pilotage d'un robot, le fonctionnement d'une entreprise et l'utilisation des informations contenues dans une image ?

Ces concepts apparemment très différents ont un point commun fondamental : ils reposent sur des échanges d'ordres et/ou d'informations entre les entités assurant un bon fonctionnement de l'ensemble du système à un niveau local (les individus dans le cas de l'État, les actuateurs dans le cas du pilotage d'un robot, les pixels dans le cas d'un imageur) et les entités assurant le pilotage (le gouvernement dans le cas de l'État, la direction dans le cas des entreprises, un ordinateur dans le cas des imageurs et des robots). Le bon déroulement de ces échanges d'informations repose sur un système de communication efficace.

La question initiale devient donc : qu'est-ce qu'un système de communication efficace ? Pour répondre à cette question, regardons comment s'effectuent les échanges

informationnels dans le fonctionnement d'un État, d'un robot et quels sont les coûts engendrés par ces échanges.

3.1.1 Échanges informationnels dans un État

Dans cette section nous examinons la chaîne de transmission de l'information émise par les individus et transmise au gouvernement. L'analyse débute par une étude des sources d'information à la disposition du gouvernement. Ces sources d'information sont notamment publiques. Parmi les informations de type public figurent principalement les statistiques établies par des organismes publics de type ministères ou instituts nationaux. Les informations qu'ils fournissent sont synthétisées de manière à réunir en un document compact les informations provenant de diverses sources secondaires.

Prenons par exemple les rapports établis par le ministère de l'Économie et des Finances en France. Ces rapports contiennent une grande densité d'informations de haut niveau, telles que le produit intérieur brut (PIB), les dépenses et les recettes de l'État ou encore des informations sur le financement de la dette publique. Les informations très synthétiques contenues dans ces rapports sont complétées par des analyses faites par des experts du domaine pour être fournies au gouvernement. Pour arriver à produire ces rapports à contenus synthétique et analytique, il est nécessaire d'utiliser des informations provenant de diverses sources secondaires, qui sont elles-mêmes déjà synthétiques.
Les sources secondaires d'information sont typiquement les collectivités locales, des organismes gouvernementaux décentralisés (typiquement les trésoreries générales)... Les informations concernant les recettes de l'État par exemple sont centralisées par des organismes comme les trésoreries générales qui font remonter au ministère des informations sur les recettes de l'État limitées à un département. A ce stade de la collecte, les informations sont déjà synthétiques mais elles n'ont aucune valeur ajoutée en terme d'analyse.
Ces informations départementales synthétiques sont quant à elles générées de manière locale en interrogeant (pour simplifier) directement les personnes ou les entreprises par le biais des déclarations d'impôts, de TVA... Ces informations locales sont brutes et n'ont aucune valeur ajoutée de type synthétique ou analytique.

Ainsi, on peut constater que la collecte d'informations destinées au gouvernement est le résultat d'un processus hiérarchique. L'information brute détenue par chaque individu est collectée à un niveau régional pour être synthétisée, puis collectée à un niveau national pour être analysée, puis utilisée par les dirigeants pour les prises de décisions. La quantité d'informations manipulées décroît à mesure que celle-ci s'approche de l'appareil dirigeant. A l'inverse, la valeur analytique de l'information augmente à mesure que l'on se rapproche du gouvernement. On voit donc que la quantité d'information est transformée en qualité de l'information lors du processus de transmission. Ce processus est rendu nécessaire par l'incapacité d'un être humain à traiter des quantités gigantesques d'informations à la fois.

3.1.2 Échanges informationnels lors du pilotage d'un robot

Considérons à présent le cas d'un robot simple muni de deux roues motrices lui permettant de se déplacer d'un point à un autre et d'effectuer des actions. Un tel robot est usuellement muni d'un cerveau de commande (que l'on supposera ici constitué par un ordinateur), d'actuateurs, de capteurs et de cartes d'interfaçage permettant de piloter les actuateurs et de lire les capteurs à partir du cerveau.
Le cerveau, pour piloter efficacement le robot doit pouvoir avoir des informations de haut niveau sur son environnement et sur l'état de ses actionneurs. Le cerveau du robot doit par exemple pouvoir connaître sa position à tout instant pour pouvoir prendre une décision. Une telle fonctionnalité de niveau analytique avancé peut par exemple consister à combiner et intégrer les données synthétiques de type vitesse de déplacement provenant des cartes de gestion des capteurs odométriques placés sur chacune des roues. L'information transmise ensuite au cerveau (la position du robot) est donc de types synthétique et analytique.
Les informations synthétiques utilisées pour le calcul de la position sont fournies par les cartes d'interface avec les capteurs. Elles sont générées par comptage des impulsions envoyées par les odomètres. L'information de vitesse est donc une information synthétique et elle utilise des informations brutes de type signaux électriques en créneaux envoyés par les codeurs. Cette information de vitesse est synthétique mais n'a aucune valeur analytique. L'information impulsionnelle des codeurs n'est quant à elle ni synthétique, ni analytique.

On retrouve finalement la structure de la chaîne de transmission de l'information vue précédemment dans le cas des échanges informationnels dans un État. La quantité d'informations manipulées décroît à mesure que celles-ci s'approchent du cerveau. Simultanément, la valeur analytique de l'information augmente. La quantité d'informations est transformée en qualité de l'information lors du processus de transmission. Ce processus est rendu nécessaire par l'inadéquation des processeurs à traiter un flux important de données basiques telles que des créneaux de tensions arrivant de manière asynchrone.

3.1.3 Échanges informationnels dans le fonctionnement d'une entreprise

Le fonctionnement d'une entreprise est basé comme les deux exemples précédents sur un fonctionnement hiérarchique. Nous n'avons pas la prétention de vouloir rentrer dans le détail du fonctionnement d'une entreprise, il s'agit seulement de décrire un processus simplifié de transmission de l'information de manière à comprendre les enjeux de ces échanges informationnels.

Essayons de voir quelles transformations subit l'information provenant des ouvriers et ayant pour destinataire la direction de l'entreprise. Celle-ci a pour objectif d'assurer le fonctionnement de l'entreprise. Pour cela, elle doit être en mesure d'avoir des informations sur l'état de fonctionnement global de l'entreprise afin de prendre

des décisions permettant d'ajuster sa stratégie de pilotage. Les informations proviennent des différents services de l'entreprise par exemple sous forme de rapports d'activité internes synthétiques et ayant un contenu analytique.

Ces rapports d'activité fournis par les différents services de l'entreprise sont générés à partir de données permettant d'évaluer la quantité et la qualité du travail effectué. Ils sont synthétiques dans la mesure où ils regroupent des informations issues de diverses sources en un même document, et analytiques dans la mesure où les personnes établissant les rapports internes vont trier les informations de manière à mettre en lumière celles qui sont les plus intéressantes.

Les données utilisées pour établir les rapports internes sont en revanche seulement synthétiques : elle sont par exemple le fruit d'une évaluation de la quantité ou de la qualité du travail effectué à partir de données brutes fournies par des appareils de mesure ou des grilles d'évaluation du travail. De telles données n'ont pas de contenu analytique, elles sont seulement le fruit d'un regroupement d'informations.

On retrouve donc à nouveau la structure de la chaîne de transmission de l'information vue précédemment : la quantité d'informations transmises décroît à mesure que l'on s'approche du cerveau (la direction de l'entreprise), en parallèle le contenu analytique de l'information augmente.

En outre, dans le cas des entreprises, il est important de noter que la structure de l'entreprise et avec elle la chaîne de transmission de l'information tend à s'adapter dynamiquement aux tâches à effectuer. Il est en effet possible de regrouper des personnes, machines ou services de manière à améliorer l'efficacité de la communication et du travail au sein d'un même projet. Si l'on considère l'entreprise comme un opérateur capable de traiter des données (commandes ou projets à réaliser), alors force est de constater que l'opérateur s'adapte aux données à traiter. Cette capacité à s'adapter permet aux entreprises d'améliorer leur rendement et leur compétitivité et de diminuer leur coût de communication interne.

3.1.4 Hiérarchisation de la transmission de l'information et réduction du coût de transmission

Dans les trois exemples proposés précédemment, nous avons pu constater que la stratégie de transmission de l'information utilisée consiste à réduire la quantité de données transmises tout en augmentant la valeur analytique de ces données à mesure que l'on se rapproche de l'organe de prise de décision. Nous avons également pu constater que cette transformation des données s'effectue en deux étapes principales. La première consiste à produire une information synthétique regroupant des informations locales à un échelon régional. La deuxième consiste à analyser les informations régionales afin de les enrichir et de permettre la prise de décision.

Partant de ces constations, on peut se demander pourquoi la transmission de l'information comporte plusieurs étapes. Cette répartition, dans la vie publique, a été façonnée par le temps et les différents modes d'organisation qui se sont succédés dans l'histoire. Les différentes administrations qui se sont succédées ont globalement contribué à améliorer la transmission des informations et leur analyse. Les deux aspects principaux de ces améliorations ont été une meilleure efficacité de transmission et une minimisation du coût que ces transmissions imposent à la société. Dans le cas du pilotage d'un robot, la hiérarchisation de la transmission de l'information s'impose également naturellement. Une structure de pilotage bien choisie permet d'améliorer l'efficacité tout en minimisant le coût global du système. Ce coût global peut avoir une formulation différente selon l'orientation du projet, par exemple énergétique ou financière. Dans le cas d'une application robotique basse consommation, le choix de la structure s'orientera vers une minimisation du coût énergétique induit par les transmissions et par le fonctionnement des opérateurs de traitement. Dans le cas d'une application faible coût, on cherche à utiliser au mieux les composants choisis, en adaptant par exemple le type d'opérations effectuées par un opérateur aux capacités de cet opérateur.

Les optimisations de la chaîne de transmission de l'information se rejoignent en partie. En effet, dans le cas de l'optimisation du coût financier comme dans le cas de l'optimisation du coût énergétique, il est préférable d'utiliser au mieux les composants présents dans la mesure où un composant inutilisé totalement ou partiellement consomme de l'énergie et coûte de l'argent pour rien. L'exemple des échanges informationnels dans les entreprises apporte une dimension supplémentaire : la structure permettant la transmission hiérarchique de l'information doit être adaptée aux données à traiter de manière dynamique si celles-ci viennent à changer au cours du temps.

On constate finalement sur des cas concrets que la transmission de l'information a été découpée en étapes intermédiaires afin d'accroître son efficacité et de minimiser son coût. Les facteurs permettant d'améliorer l'efficacité de la transmission de l'information rejoignent ceux qui permettent de minimiser le coût des échanges informationnels. Ce sont les suivants :

- **Adéquation entre l'information à traiter et l'opérateur effectuant ce traitement**. Dans le cas de la robotique, l'utilisation d'un ordinateur pour lire les codeurs de position n'est pas adéquate, elle conduit à utiliser une machine aux fonctionnalités très avancées et synchrones pour une tâche asynchrone effectuant des calculs simples. Il est préférable dans ce cas d'utiliser un circuit dédié de type $LM629$ par exemple.
  En traitement d'image, une parallélisation massive convient par exemple très bien pour traiter des grandes quantité de données similaires et simples, alors qu'un traitement sériel à l'aide d'opérateur flexibles et plus complexes sera plus adapté au traitement d'informations de haut niveau.

- **Adéquation entre l'information à transmettre et le mode de transmission**. Ce point rejoint le précédent dans la mesure où le mode de transmission de l'information dépend fortement de la manière dont on traite ces informations.
  Dans le cas de la vie publique, la collecte des informations concernant chaque individu est répartie sur les collectivités territoriales, ce qui permet de limiter à la taille de petites entités (cantons par exemple) les traitements à effectuer sur les formulaires reçus. Ces traitements s'effectuent en parallèle dans chaque centre, ce qui assure un fonctionnement du traitement bas niveau de l'information de type massivement parallèle très efficace. A l'inverse, une centralisation des informations collectées auprès de chaque individu et leur traitement de manière sérielle conduirait à des temps de traitement très longs et inadaptés et à un coût de transmission potentiellement élevé. Il est donc important d'assurer l'adéquation de l'information à transmettre et le mode de transmission.

- **Adaptation dynamique de la structure de traitement de l'information aux données à traiter**. Outre l'adéquation entre le mode de transmission, le mode de traitement et les informations manipulées, la structure de l'information peut évoluer au cours du temps. Il est donc nécessaire d'adapter les structures de traitement et de transmission à ces évolutions.// Par exemple, dans les cas des entreprises, l'adaptation de la structure de l'entreprise aux projets en cours est un facteur permettant d'améliorer l'efficacité des échanges informationnels. Cette dimension n'est pas présente dans le cas des échanges informationnels dans les États ou en robotique dans la mesure où nous avons considéré des systèmes figés dans le temps. Ces aspects dynamiques se retrouveront cependant en traitement d'image dans la mesure où les structures de traitement des données doivent s'adapter à la forme des objets des images qui se succèdent.

La transmission et le traitement de l'information étant constitués de plusieurs étapes, il est important d'optimiser chacune des ces étapes en tenant compte des critères présentés ci-dessus. Partant d'une information locale et basique, non synthétisée et non analysée, le rôle des premiers opérateurs de traitement et transmission est donc d'effectuer des traitements basiques et locaux sur l'information et de la synthétiser à une échelle régionale. L'information générée n'a pas ou peu de contenu analytique car elle est principalement le fruit d'une agrégation synthétique de données. Cette première étape peut évidemment être découpée en plusieurs parties, elle requiert des outils simples et répartis, adaptés dynamiquement aux données à manipuler. Le rôle de la deuxième couche de traitement et de transmission est d'analyser les informations déjà synthétisées afin de fournir et transmettre une information de haut niveau permettant des prises de décisions. L'information générée a un contenu analytique : celui-ci provient par exemple de la comparaison des caractéristiques extraites des ensembles d'informations agrégées aux caractéristiques de données de référence ou à des seuils préalablement fixés ou calculés.

Cette analyse requiert des outils complexes permettant d'effectuer des traitements avancés et analytiques.

3.2 La régionalisation : une étape utile dans le traitement de l'image

Au travers des exemples présentés précédemment, nous avons pu constater que la transmission d'informations réparties dans le temps (impulsions codeurs) ou dans l'espace (formulaires) vers un organe de décision nécessite d'utiliser une structure de transmission de l'information par étapes successives, chaque étape utilisant des outils adaptés au type de données à traiter. Nous envisageons dans cette partie l'adaptation de ce principe au cas du traitement d'images.

3.2.1 Échanges informationnels en traitement d'images

Dans les applications d'acquisition et traitement d'images, nous retrouvons la problématique soulevée dans les exemples du traitement d'informations dans la vie publique, en robotique ou dans les entreprises. Il s'agit en effet de traiter des quantités de données très importantes et avec un fort débit dans le but d'extraire des informations synthétiques et analytiques permettant à un organe de contrôle de prendre des décisions. Cette section étudie un exemple concret : la chaîne de transmission de l'information d'un imageur vers un système de prise de décision tel qu'un système de détection de défaut dans des objets connus sur une chaîne de fabrication.

Les informations synthétiques et analytiques transmises au système de prise de décision sont de haut niveau. Il s'agit par exemple dans le contexte de notre application d'informations du type : un objet de taille différente de la taille standard est observé par la caméra. Une telle information permet au cerveau de prendre une décision très rapidement sans le surcharger en calculs. La génération de cette information nécessite toutefois un certain nombre d'étapes préliminaires.
L'étape immédiatement précédente est la comparaison de l'objet vu aux objets d'une base de données. Cette comparaison permet d'ajouter à la donnée objet vu un contenu analytique expliquant si l'objet vu est de type correct ou non. Pour effectuer cette tâche, il est nécessaire de disposer d'une structure permettant d'effectuer des opérations complexes et irrégulières (typiquement un ordinateur).

Les données utilisées pour la comparaison sont les caractéristiques d'un objet. Ces données sont synthétiques mais n'ont pas nécessairement de contenu analytique complexe. Pour qu'un objet puisse être utilisé par une structure permettant des opérations complexes, il est nécessaire qu'il soit décrit de manière compacte. En effet, pour effectuer des opérations complexes irrégulières, les données en entrée sont le plus souvent insérées de manière sérielle, il est donc nécessaire de limiter le nombre de données à des données compactes. La description d'un objet doit donc

pour être efficace être effectuée de manière compacte. Une telle description peut reposer par exemple sur des caractéristiques globales de l'objet telles que sa surface, sa compacité, son périmètre, la topologie de son squelette, etc...

L'extraction de ces caractéristiques utilise des données très simples et locales pour générer une mesure régionale. Comme indiqué ci-dessus, l'utilisation de dispositifs de traitement des données complexes fonctionnant de manière sérielle repose sur une structure permettant de faire des calculs abstraits de grande complexité. Effectuer des opérations simples et répétitives telles que des additions avec ce type de structure conduit à une inadéquation entre l'opérateur et les données à traiter. Il serait préférable d'utiliser pour ces tâches régionales des opérateurs plus simples et adaptés à ces types de traitement. Ces opérateurs doivent de plus être dynamiquement reconfigurables car les régions sur lesquelles s'effectuent les opérations régionales sont susceptibles de changer à chaque image.

Les données utilisées par les opérateurs d'extraction d'informations régionales sont locales. Elles ne sont ni synthétiques, ni analytiques. Toutefois, elles peuvent faire l'objet de traitement locaux préalables. Ces opérations locales nécessitent de manipuler de grandes quantités de données en parallèle. Une solution intéressante est donc d'utiliser des opérateurs massivement parallèles pour leur traitement. Les opérateurs simples massivement parallèles permettent d'effectuer de tels traitements localement. Il est également possible d'effectuer ces opérations de manière sérielle ou à parallélisme limité sur un ordinateur, mais cette solution n'est pas optimale en terme d'adéquation entre l'opérateur et les données à traiter. De plus, il est nécessaire dans ce cas de déplacer de grandes quantités de données simples sur de longues distances, ce qui n'est pas optimal d'un point de vue adéquation entre l'information à transmettre et le mode de transmission.

3.2.2 Classification des opérateurs de traitement d'images

En considérant un exemple simple, nous avons retrouvé la classification en niveaux des opérations de traitement d'images. De plus, il apparaît que le type d'opérateurs adapté à chaque niveau de traitement d'image n'est pas le même. Les différents niveaux sont les suivants :

- **Opérations de bas niveau** : transformations locales de l'image. L'opérateur adapté à ce type d'opérations est de type massivement parallèle et localisé près des données de manière à éviter les transmissions sur de longues distances.
  Il existe dans la littérature de nombreuses implantations permettant d'effectuer ces opérations locales de manière massivement parallèle. Nous pouvons citer en particulier les rétines artificielles numériques programmables qui ont été à la source de cette thèse et qui effectuent ce type de calcul très efficacement.

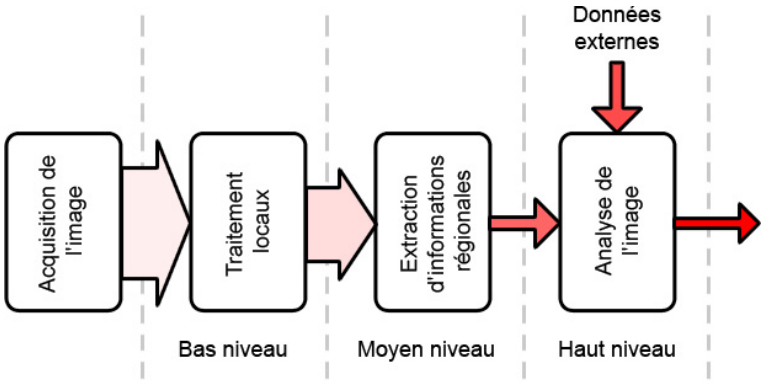


FIG. 3.1 – Classification des opérations de traitement d'images. La taille des flèches représente le volume d'informations transmises, la densité du rouge des flèches représente le niveau d'abstraction de l'information.

- **Opérations de moyen niveau** : extractions d'informations régionales synthétiques sur des ensembles de pixels. L'idéal pour ce type d'opérations est un opérateur adapté au sens de l'adéquation entre l'information locale à traiter, les traitements relativement simples mais régionaux à effectuer et le mode de transmission de l'information. Cet opérateur, pour être adéquat, nécessite d'être régional tout en étant réparti sur la région à traiter de manière à minimiser les transferts de données locales. De plus il doit pouvoir s'adapter dynamiquement aux données à traiter. L'étude des structures permettant de réaliser ces opérations régionales étant au coeur de cette thèse, nous étudierons les structure existantes adaptées à ces opérations avant d'en proposer une nouvelle.

- **Opérations de haut niveau** : analyses utilisant des informations régionales synthétiques afin de leur donner un contenu analytique. Les opérateurs adaptés à ces traitements doivent être capables de gérer un faible volume de données de type irrégulier, utilisant des outils de haut niveau d'abstraction. Un ordinateur constitue un opérateur adéquat pour ce type de traitements de haut niveau.

Cette classification est représentée au schéma de la figure 3.1. Elle est communément adoptée par la communauté scientifique. En revanche, la définition des opérateurs adéquats associés à chacune de ces classes de traitement du signal dépend du contexte d'utilisation du système. Il est en effet aujourd'hui possible d'utiliser un ordinateur pour effectuer toutes les tâches de traitement d'un signal provenant d'une caméra grâce à la vitesse de calcul importante du ou des processeurs. L'utilisation d'un ordinateur pour des opérations de bas niveau ne correspond cependant pas à une adéquation entre les informations à traiter (qui sont simples et régulières) et la structure utilisée (permettant de gérer des données complexes et irrégulières). Cette inadéquation ne se traduit pourtant pas forcément au niveau de la vitesse

de calcul. Elle se situe avant tout au niveau de la consommation d'énergie. Le traitement d'images de bas niveau est typiquement 100 fois moins coûteux en énergie sur une rétine artificielle que sur un ordinateur standard.
A la partie suivante, les opérateurs permettant d'être en adéquation avec les données à traiter pour chaque niveau de vision seront étudiés. Nous verrons également de quelle manière il est possible de les faire cohabiter efficacement.

3.3 Quels types d'opérateurs pour les différents niveaux de traitement d'image ?

A la faveur de la montée en fréquence des microprocesseurs et de l'augmentation en taille de leurs caches mémoire, il est devenu possible aujourd'hui d'implanter des applications industrielles de traitement d'image et de vision en temps réel sur un système associant imageur CMOS, microprocesseur et RAM. Toutefois, en termes de complexité, les tâches de vision ainsi réalisables demeurent loin de celles assurées par les systèmes de vision biologiques.

Dans un contexte où la vision artificielle demeure un marché de faible volume, il est très commode de pouvoir ainsi associer un petit nombre de composants sur étagère pour réaliser un système de vision : capteur à interface numérique, microprocesseur, mémoire.

Mais le jour où la vision artificielle constituera un marché de volume, ce qui arrivera tôt ou tard avec la généralisation des systèmes de surveillance et l'avènement des robots dans le quotidien, le découpage ci-dessus demeurera-t-il le plus approprié ? Cela pose la question de la structure idéale d'un système de vision. La vision artificielle étant encore un domaine scientifique très ouvert, il serait bien présomptueux de vouloir répondre à une telle question. Mais l'on peut néanmoins tenter d'apporter des éléments de réponse.

3.3.1 Opérateurs de bas-niveau et parallélisme massif

L'évolution actuelle des microprocesseurs généralistes ne semble guère laisser d'avenir à l'utilisation d'une unique ressource de calcul centralisée. Le décalage croissant entre la vitesse des processeurs et celle des mémoires tend vers une fracture qui n'est aujourd'hui supportable qu'au prix de mécanismes de hiérarchisation et de prédiction de plus en plus sophistiqués. Les microprocesseurs atteignent leurs limites en termes de complexité, de vitesse, d'énergie, et il devient plus opportun de les dédoubler, ou plus, aboutissant finalement à la mise en oeuvre parallèle de plusieurs coeurs de processeur sur une même puce.

Côté vision artificielle, la tentation du parallélisme est apparue bien avant que les

contraintes technologiques n'y conduisent "de force" les microprocesseurs généralistes. En effet, l'image est un objet très particulier où chaque donnée est amenée à interagir préférentiellement avec ses voisines. Cela ne concerne d'ailleurs pas seulement l'image captée mais aussi les différentes interprétations qui en sont faites. C'est ainsi que le cortex humain maintient des représentations "rétinotopiques" de l'information jusqu'aux plus hauts niveaux d'analyse de l'image.

Cette interaction privilégiée entre données voisines amène naturellement à penser qu'une grille de processeurs communiquant avec leurs plus proches voisins peut constituer un support efficace au traitement de l'image et à la vision artificielle. Ce type d'architecture est qualifiée de "cellulaire". Profitant du parallélisme spatial massif ainsi disponible, les processeurs de la grille peuvent être simples et cadencés à faible fréquence, relâchant ainsi les contraintes sur le temps et l'énergie, si critiques aujourd'hui. Par ailleurs, là où un microprocesseur devra décoder n fois de suite une même instruction prise dans une boucle de balayage, le contrôle des multiples processeurs de la grille est mis en commun, économisant ressources et énergie. On obtient typiquement une réduction de la consommation d'énergie d'un facteur 100 pour le calcul d'une opération telle qu'une somme entre deux images sur 8 bits [Ber05]. Tout cela suppose néanmoins qu'une proportion suffisante de processeurs soit exploitée efficacement et de manière soutenue dans le temps[1].

Une grille de processeurs cellulaires utilisée pour traiter les images constitue également une réponse à la fracture processeur/mémoire évoquée plus haut. En effet, dans une grille, les données manipulées par un processeur sont soit logées dans sa mémoire propre soit dans celle d'un de ses voisins proches et il est donc rapide de les obtenir. Bien sûr, cet avantage s'effondre s'il faut régulièrement charger/extraire une image depuis/vers l'extérieur de la grille. Les rétines artificielles règlent cette question d'une part en attachant un photocapteur à chaque processeur et d'autre part en s'interdisant purement et simplement tout échange d'image avec l'extérieur. Les données participent à des calculs près de l'endroit où elles ont été captées, et elles ne sont manipulées que sur de toutes petites capacités électriques : cela procure un avantage énergétique majeur, de plusieurs ordres de grandeur.

La mise en oeuvre SIMD d'une grille de processeurs cellulaires convient pleinement aux opérations de traitement d'image de bas niveau, dont un exemple typique est l'application d'un filtre sur l'image captée. Les filtres appliqués peuvent être linéaires ou non, tels ceux issus de la Morphologie Mathématique. Les filtrages sont souvent suivis de seuillages et, très tôt, images binaires ou images faiblement multivaluées se mêlent aux images en niveaux de gris dans la succession des opérations

[1]Parmi les raisons permettant de justifier ce gain énergétique figure la valeur très réduite des capacités agitées dans la rétine lors de l'envoi des instructions aux processeurs et lors des échanges de données. La capacité moyenne agitée pour transmettre des instructions à un processeur élémentaire actif est celle la grille complète divisée par le nombre de processeurs actifs de celle-ci. Sa valeur est donc de l'ordre de grandeur d'une capacité locale si le nombre de processeurs actifs est important

réalisées. Étant donné la diversité de ces opérations et la dynamique variable des images manipulées, l'opérateur de calcul bit-série est le plus approprié par son universalité, et bien sûr aussi par sa compacité dans le contexte matériellement très contraint des rétines artificielles. Ainsi le calcul de la somme entre deux images en niveaux de gris est-il l'objet d'une séquence d'instructions, travaillant bit après bit, dans le code déroulé sur la rétine. Le choix d'opérateurs bit-série pour un processeur rétinien est en fait analogue à la démarche RISC consistant à simplifier le jeu d'instructions d'un microprocesseur pour gagner en efficacité, et à reporter sur la programmation la réalisation des instructions complexes éliminées.

3.3.2 Opérateurs de moyen niveau : les enjeux des traitements régionaux

Les traitements de bas niveau permettent d'extraire de l'image des informations locales. Cette localité rend cependant difficile l'utilisation de ces données par un opérateur de haut niveau pour directement prendre des décisions efficacement. Avoir recours à des traitements moins localisés permettant d'agréger au sein d'une même grandeur des informations de bas niveau provenant d'un horizon spatial plus large est donc une nécessité.

Pour cela, deux pistes simples peuvent être envisagées : l'utilisation d'un opérateur global à l'échelle de la rétine et l'utilisation d'opérateurs de bas niveau multi-résolution.
Les opérateurs globaux à l'échelle de la rétine permettraient d'extraire une grandeur caractéristique de l'ensemble ou d'un sous-ensemble de pixels (ce qui est la même chose si l'on masque au préalable les pixels n'appartenant pas au sous-ensemble) de la rétine. Par exemple, il est intéressant de savoir quelle est la proportion de pixels ayant atteint tel seuil après application de tel filtre. Cela rend possible une analyse intermédiaire permettant par exemple de conditionner le choix du filtre ou du seuil suivant. Pour cela, un opérateur de sommation, capable de compter le nombre de points à 1 dans une image binaire est un opérateur très utile. Or cela est peu coûteux à implanter sous forme analogique dans une rétine artificielle et l'on ne s'en prive donc pas. On peut déjà assurer ainsi des applications simples de vision, comme la détection et l'écartométrie d'un objet dont les caractéristiques photométriques le distinguent facilement du fond. Une telle fonctionnalité est cependant limitée à l'extraction d'informations sur un unique ensemble de pixels de la rétine. Pour extraire des informations sur plusieurs ensembles de points de l'image, une solution est de traiter chaque ensemble l'un après l'autre, ce qui conduit à une inefficacité temporelle et surtout énergétique.
Les opérations de bas-niveau multi-résolution reposent sur des tessellations hiérarchiques de formes prédéfinies. Cette solution permet de travailler sur plusieurs ensembles de pixels en parallèle, ce qui permet de résoudre en partie le problème évoqué précédemment. Toutefois, les découpages utilisés se moquent de la forme des objets présents dans l'image, ils conduisent à agréger des informations prove-

nant de pixels issus d'objets différents. Pour remédier à cela il est possible comme dans les algorithmes de segmentation d'image de type *split and merge* de combiner le découpage à différentes échelles de manière à se rapprocher de la forme réelle des objets. Toutefois, ces itérations sont coûteuses en énergie et en temps.

Le recours à des traitements à horizon spatial large mais définis à partir des données de l'image serait une réponse plus fondamentale à cette question. Les traitements s'effectuant sur des formes proches des objets ou régions de l'image, on peut naturellement les qualifier de régionaux. Parmi ces opérations figurent par exemple le calcul d'une surface, d'un périmètre ou encore d'un moment sur une région. Ce sont des traitements de moyen niveau. Sur le plan de la quantité et du type d'information manipulée, ces opérations ont une spécificité : elles utilisent en entrée une grande quantité de données locales distribuées sur les régions et générées à l'aide de traitements de bas niveau, et elles génèrent une quantité réduite de données agrégées destinées à être utilisées par les opérateurs de haut niveau. Cette spécificité n'est pas sans conséquence sur l'implantation des opérateurs associés. Pour ne pas perdre le gain énergétique obtenu en traitant les données là où elles sont captées, les opérateurs régionaux ont intérêt à se situer dans chacun des processeurs de la rétine afin de réduire les distances sur lesquelles seront déplacées les très nombreuses informations de bas niveau. Le coût de déplacement des informations agrégées sera quant à lui beaucoup plus réduit dans la mesure où leur nombre est dépendant du nombre de régions et non pas du nombre de pixels de l'image. Il est à noter que l'implantation cellulaire des opérateurs régionaux ne les contraint pas à fonctionner en mode SIMD.

Les opérateurs régionaux étant implantés de manière répartie dans les pixels, n'est-il pas possible de les réaliser à partir de combinaisons d'opérateurs locaux de bas niveau ? L'utilisation de traitements locaux itérés un grand nombre de fois peut permettre d'obtenir des informations régionales à partir d'informations locales réparties spatialement sur une grille. Cette manière de représenter l'information et d'effectuer des opérations locales conduisant à l'extraction de grandeurs régionales n'est pas sans rappeler certaines modélisations physiques[2]. On peut constater que chacune de ces modélisations repose plus ou moins directement sur des équations aux dérivées partielles. En algorithmique du traitement d'image, on utilise également de telles méthodes, par exemple dans les algorithmes de segmentation à l'aide d'équations aux dérivées partielles (qui seront rappelés dans la 3e partie de la thèse). Cette démarche, en raison de ses analogies avec des modèles physiques, a motivé les chercheurs et les a conduits à proposer des implantations matérielles telles que les réseaux de neurones cellulaires. Elle a également été envisagée en version rétine artificielle SIMD synchrone (réseau de neurones cellulaire à temps

[2]On citera en particulier les modélisations par éléments finis en mécanique, thermique ou électromagnétisme. Les opérations locales qui leurs sont associées vont des modèles numériques complexes traduisant les observations effectuées dans le cas de la mécanique des solides aux modèles théoriques microscopiques tels que les équations de Maxwell en électromagnétisme, l'équation de la chaleur en thermique ou encore l'équation d'Euler en mécanique des fluides.

discret).

Dans le cas d'une implantation sur rétine SIMD, le fonctionnement basé sur un phénomène de propagation de proche en proche permettant des échanges d'informations à l'échelle régionale a un sérieux inconvénient : seuls les processeurs situés sur le front de propagation sont actifs à un instant donné. Ce front de propagation peut de plus avoir une taille très réduite (par exemple, dans le cas d'une propagation sur une région filaire, la taille du front est 1 pixel), ce qui remet sérieusement en cause le gain en énergie obtenu grâce à l'utilisation de rétines artificielles. Dans les réseaux de neurones cellulaires le problème est différent dans la mesure ou aucun ordre n'est transmis au réseau de l'extérieur. Le problème se pose dans ce cas en termes temporels. L'information est analogique et doit être précise si l'on souhaite effectuer des opérations dépendant de chacune des données de la région (par exemple le calcul du bit de poids faible du nombre de pixels d'une région). Cette précision nécessite d'attendre que les propagations de proche en proche (charge de capacités) soient terminées, chacune devant attendre la fin de la précédente pour aboutir à un état final stable, ce qui conduit à un gaspillage de temps[3].

L'utilisation itérative d'opérateurs locaux est inefficace en raison d'un taux d'utilisation trop faible ou à un gaspillage de temps. Un taux d'utilisation faible des processeurs est cependant une caractéristique des mouvements de données à base de propagations dont on a besoin pour effectuer des opérations régionales de manière cellulaire. Durant ces phases, il faut donc réduire au minimum le nombre d'ordres envoyés à toute la rétine et qui ne seront utilisés que par un nombre limité de processeurs. Pour cela, l'ajout d'un réseau dédié à la propagation et au calcul régional est une nécessité. Il permettrait de réduire le nombre d'instructions envoyées pour guider les données et effectuer les calculs durant les phases de propagation[4]. Ce réseau représente un investissement matériel fait une fois pour toute pour une région donnée, qui permet ensuite de réduire le coût de déplacement de chacune des données sur l'image. De plus, ce réseau permet d'accélérer de manière importante les phases de propagation de données. Un tel réseau peut être vu comme une "autoroute" permettant aux données de se déplacer rapidement et sans coût de contrôle. L'établissement de ce réseau sur une région de forme et de taille quelconques est toutefois une difficulté à prendre en compte. Nous verrons que cette opération a

[3]On peut voir un réseau de neurones cellulaire comme un ensemble de capacités (C) communiquant entre elles. Ces capacités sont placées en parallèle, ce qui fait que la capacité totale du réseau est de l'ordre de $n * C$ où n est le nombre de cellules du réseau. De plus, chacune des capacités C est assez importante de manière à ce que l'on puisse différentier 2^k niveaux de tension si l'on souhaite une précision sur k bits. La capacité totale du réseau devient donc très importante, ce qui limite grandement la vitesse du circuit. Un réseau de neurones cellulaire simplifié fonctionnant sur un bit et n'assurant dès lors que des fonctions de propagation permettrait de réduire la capacité globale du réseau et donc d'augmenter la vitesse d'un facteur environ égale à 2^k. Pour effectuer des calculs sur k bits, il suffirait alors d'effectuer k opérations bit-série, ce qui diminuerait globalement le temps de calcul d'un facteur $\frac{2^k}{k}$. Le "luxe" des réseaux de neurones cellulaires conduit finalement à diminuer leur potentiel.

[4]Nous verrons qu'il est même possible de réduire ce nombre à 0

un coût énergétique et temporel très limité.

D'un point de vue plus fondamental, l'utilisation d'un réseau de propagation dédié permet d'effectuer des opérations régionales sans que chaque cellule ait connaissance des informations qui la parcourent. Ces informations sont en réalité dirigées vers un pixel unique, que l'on peut considérer comme un pointeur vers la région et que nous appellerons dans la suite la *racine*. Il s'agit donc d'une certaine manière "d'intégrer" les informations réparties dans l'image en un seul et unique endroit sans se préoccuper de l'ordre dans lequel les données circulent dans la région. Cette manière de procéder rappelle l'utilisation des théorèmes basés sur une formulation intégrale de principes microscopiques en physique[5]. Cette dualité entre approche locale et approche intégrale se retrouve en segmentation d'image, l'approche locale correspondant à l'utilisation d'équations aux dérivées partielles et l'approche intégrale correspondant à la minimisation d'une fonctionnelle énergétique type équation de Mumford et Shah. Cette dualité sera rappelée dans la partie 3 de la thèse.

L'utilisation des formulations intégrales conduit en général à des calculs plus simples que l'utilisation des formulations microscopiques couplées à un modèle par éléments finis. Cette simplification repose toutefois sur la possibilité de formuler un problème local sous forme intégrale, ce qui n'est possible que pour des problèmes assez simples[6], en utilisant un outil adéquat permettant de réaliser l'intégration[7]. Nous retrouvons dans notre cas des résultats assez similaires, l'ajout d'un réseau de propagation dédié permettant "d'intégrer" les données améliore les performances des opérations régionales en termes énergétiques et temporels. Toutefois, avoir recours à des modèles à base d'éléments finis peut être nécessaire dans le cas où l'on souhaite utiliser des opérateurs microscopiques n'ayant pas de forme intégrale simple.

[5]On peut citer en particulier les théorèmes de Gauss ou d'Ampère qui sont les formulations intégrales de deux équations de Maxwell, le théorème de l'énergie cinétique qui est une formulation intégrale du principe fondamental de la dynamique, le théorème de Bernoulli en mécanique des fluides qui est une intégrale de l'équation d'Euler...
Par exemple, l'équation de Maxwell-Ampère s'écrit dans l'approximation des régimes quasi-permanents :

$$\overrightarrow{rot}(\overrightarrow{B}) = \mu_0 \overrightarrow{j}$$

Ce qui s'intègre en :

$$\iint_{\Sigma} \overrightarrow{rot}(\overrightarrow{B}).d\overrightarrow{S} = \iint_{\Sigma} \mu_0 \overrightarrow{j}.d\overrightarrow{S}$$

Le théorème de Stokes nous permet ensuite d'obtenir le théorème d'Ampère :

$$\oint_{C} \overrightarrow{B}.d\overrightarrow{l} = \mu_0 I_{enlace}$$

[6]Par exemple, dans le cas de l'approximation des régimes quasi-permanents en électromagnétisme ou dans le cas d'un écoulement irrotationnel en mécanique des fluides.

[7]Par exemple, le théorème de Stokes pour passer de l'équation de Maxwell-Ampère au théorème d'Ampère, le théorème de Green-Ostrogradski pour passer de l'équation de Maxwell-Gauss au théorème de Gauss, ou une intégrale sur une ligne de champs pour passer de l'équation d'Euler au théorème de Bernoulli.

Dans ce cas, seule une simulation numérique peut fournir le résultat. En traitement d'image, les calculs régionaux sont le plus souvent assez simples et conduisent à un résultat "intégral" unique pour chaque région. L'implantation intégrale des opérateurs régionaux est donc dans ce cas mieux adaptée et préférable.

3.3.3 Quels types d'opérateurs pour les traitements régionaux ?

Nous avons montré précédemment la nécessité d'utiliser des opérateurs de moyen niveau. Des pistes concernant leur implantation ont été proposées. Afin de minimiser les coûts de déplacement des données, il est intéressant d'effectuer les opérations de moyen niveau à l'intérieur de la rétine. Ce point étant acquis, reste à choisir le mode d'implantation des opérateurs régionaux cellulaires. La dualité entre formulation locale couplée à des méthodes de calcul par élément finis et formulation intégrale se retrouve dans les implantations électroniques des opérateurs régionaux. Par exemple, les réseaux de neurones cellulaires permettent de réaliser des opérations régionales analogues aux calculs par éléments finis basés sur des équations aux dérivées partielles non-linéaires. Une telle implantation n'offre pas la possibilité d'effectuer de calcul intégral direct, ce qui peut conduire à une relative inefficacité pour des opérations régionales simples (et qui sont souvent les plus utiles en traitement d'image). Autre exemple, la *Maille Associative d'Orsay* (qui sera détaillée dans la suite de cette thèse) offre à l'inverse une fonctionnalité de calcul intégral qui retourne un résultat unique par région. Toutefois, de par son implantation, cette fonctionnalité ne se limite pas uniquement au calcul intégral dans la mesure où elle permet également d'effectuer des calculs régionaux conduisant à des résultats différents dans chacun des pixels d'une même région. Cette extension des capacités des opérateurs de calcul intégral a une contrepartie : un coût d'implantation réparti dans chaque pixel assez élevé et trop coûteux pour une implantation dense dans une rétine artificielle.

Nous proposons dans cette thèse de prendre une orientation forte, en nous focalisant sur les opérateurs régionaux permettant uniquement le calcul intégral. Cette orientation est motivée par la relative simplicité des opérateurs régionaux utiles en traitement d'image, et la possibilité de les exprimer sous forme intégrale. Une étude des opérateurs régionaux utilisant des structures dédiées de propagation des données est proposée, le but étant de dégager une structure permettant lors des opérations régionales un fonctionnement de la rétine sans envoi d'instruction lorsque le taux d'occupation des processeurs est faible, les instructions étant à l'opposé transmises à chaque processeur pour un coût modique lorsque que le taux d'occupation des processeurs est élevé. Une implantation du modèle des réseaux associatifs (utilisé dans la *Maille Associative d'Orsay* et qui sera rappelé ultérieurement) permettant de réaliser ces opérations régionales intégrales en répondant aux objectifs ci-dessus est ensuite proposée. La structure électronique qui lui est associée permet de réduire le coût en transistor des opérations régionales à

une valeur acceptable pour une implantation dense dans les rétines artificielles, en contre-partie d'un coût algorithmique un peu plus élevé. Les aspects électroniques et algorithmiques de cette implantation seront étudiés en détails.

Cette implantation permet d'élargir le champ d'application des rétines en leur conférant des fonctionnalités ayant un niveau d'abstraction supérieur à celles d'une rétine humaine (limitée aux traitements de bas niveau multi-résolution). D'un point de vue algorithmique, l'introduction de traitements régionaux dans les rétines permet pour un coût raisonnable d'effectuer des calculs régionaux (des applications sont présentées dans la 3e partie de la thèse). Cette capacité est très importante car comme montré dans la thèse de B. Ducourthial [Duc00], elle ouvre la porte à la manipulation sur rétine d'objets abstraits de moyen niveau tels que des graphes, ce qui en élargit de manière fantastique le champ d'utilisation.

3.4 Primitives régionales fondamentales

Dans la fin de ce chapitre et dans la suite de la thèse, l'utilisation du terme régional impliquera que l'opérateur ou la primitive associée est de type intégral (par opposition aux primitives régionales réalisées par itération d'opérateurs locaux), à moins que cela ne soit précisé.

Nous avons montré précédemment que le traitement d'images de moyen niveau a un intérêt en terme de minimisation de l'énergie nécessaire au traitement de l'information répartie sur les pixels d'un imageur. Nous avons également montré que les opérateurs régionaux doivent être de préférence placés à l'intérieur même des pixels de manière à limiter les déplacements de données brutes non synthétiques. Dans cette section nous étudions les primitives utiles au traitement d'images afin de dégager un certain nombre de primitives élémentaires et fondamentales. Ces primitives, utilisées de manière séquentielle, permettront d'effectuer de nombreuses opérations régionales utiles au traitement d'images.

La première question que l'on se pose est : comment caractériser une primitive régionale ? La réponse à cette question s'appuie sur la partie précédente. Un opérateur régional permet d'extraire des grandeurs régionales caractéristiques de la région à l'aide d'opérations simples prenant en compte les valeurs locales des pixels. Cette prise en compte des valeurs locales des pixels peut s'effectuer sous différentes formes selon que la variation d'une seule et unique valeur locale de la région a de manière certaine ou n'a pas forcément un impact sur la grandeur régionale extraite. Partant de ce constat, deux grandes classes de primitives régionales se dégagent. En reprenant l'analogie entre le traitement d'images et le fonctionnement d'un État, on peut constater que ces deux types d'opérateurs régionaux se retrouvent dans la vie quotidienne. Dans une élection par exemple, le fait de changer la valeur d'un unique vote individuel ne change la plupart du temps rien au résultat. A l'inverse,

dans le calcul des recettes d'un État, un changement de déclaration de revenus d'une seule personne change la valeur globale des recettes.

Considérons tout d'abord les opérations régionales dont le résultat change lorsque un et un seul des opérandes change. Nous nous attachons à rechercher l'opération canonique permettant d'avoir ce comportement. Considérons la forme binaire de cet opérateur régional, le résultat de l'opération à n entrées (les valeurs des n pixels de la région) change dès lors qu'une entrée est modifiée. Pour remonter à la forme canonique, considérons maintenant le cas où les opérandes et la sortie sont représentés sur un bit. L'opérateur le plus simple dont la sortie s'inverse lorsque que l'on inverse la valeur d'une entrée est le *OU* exclusif à n entrées. Cet opérateur peut être vu comme donnant la parité du nombre d'entrées à l'état logique 1. Il peut également être considéré comme retournant le bit de poids faible d'une addition à n entrées.
L'opérateur canonique sur un bit dont la sortie change d'état lorsqu'une de ses entrées change d'état est le *OU* exclusif. Sa généralisation à n bits, en ajoutant la contrainte que cet opérateur soit croissant par rapport à chacune de ses variables, est la généralisation d'un opérateur retournant le bit de poids faible d'une addition : l'additionneur sur n bits. Finalement, l'opérateur somme régionale est l'opérateur symétrique strictement croissant par rapport à chacune de ses variables le plus élémentaire que l'on puisse trouver.

Considérons à présent les opérateurs régionaux dont le résultat ne change pas nécessairement lorsqu'une seule entrée change. Ces opérateurs changent d'état lorsqu'un ensemble de valeurs changent, on peut donc les concevoir comme des processus de type vote. L'implantation de ces votes peut se faire de manière simple à partir de l'opérateur somme régionale déterminé précédemment. Il suffit de sommer les contributions de chacun des pixels puis de comparer cette valeur à un seuil. L'opération globale la plus simple possible mise en jeu est donc là encore une addition. La comparaison étant faite ensuite de manière locale, elle ne rentre pas dans le champs des opérateurs régionaux.

Finalement, la primitive essentielle qui se dégage pour les traitements régionaux est la somme régionale. Nous utilisons dans la suite de cette thèse, cet opérateur régional comme *primitive exemplaire*. Comme nous l'avons montré précédemment, il est préférable d'utiliser des opérateurs régionaux de type bit-série afin de les rendre utilisables dans le cas d'opérations de moyen niveau multi-résolution. Cette orientation nous conduit dans la suite à décomposer la primitive somme régionale en itération d'opérateurs *OU* exclusif, opérateur qui se révèle donc être la *primitive élémentaire régionale intégrale fondamentale*.
Il est à noter que la détermination de cette primitive repose uniquement sur des considérations fonctionnelles. La primitive somme régionale permet de réaliser un grand nombre d'opérations régionales, toutefois elle n'est pas forcément optimale en terme de temps de calcul sur une architecture donnée. Pour cette raison, certaines implantations du modèle des réseaux associatifs telles que la *Maille Associative*

d'Orsay reposent sur la somme régionale mais également sur d'autres opérateurs implantés matériellement tels que le *OU* logique régional.

Deuxième partie

Opérateurs régionaux dans les rétines artificielles

4

De la nécessité des graphes : le modèle des réseaux associatifs.

4.1 Limites du fonctionnement SIMD

Nous avons mis en évidence précédemment l'utilité de la régionalisation dans le traitement de l'image et l'intérêt d'implanter les opérateurs régionaux au sein même du pixel et de manière distribuée sur une région. Nous considérons donc dans cette partie que les opérateurs régionaux sont placés au sein même des pixels.
Dans une rétine synchrone de type SIMD, chaque processeur effectue la même opération au même moment. Le déplacement de données dans ce mode de fonctionnement conduit à déplacer des plans mémoire entiers simultanément sur de courtes distances. Pour effectuer une opération régionale de type intégral, il est nécessaire de déplacer l'image par plans mémoire entiers de manière à ce que chacun des pixels de la région parcoure tous les autres pixels de la région (ce parcours est nécessaire pour réaliser par exemple la primitive régionale fondamentale de somme régionale introduite à la partie précédente). Cette succession de déplacements dépend de la forme de la région à traiter. Or, l'intérêt du mode SIMD est de traiter tous les pixels de l'image et donc de chaque région en parallèle. De nombreux problèmes se posent :

- Les déplacements effectués pour que chaque pixel de la région parcoure tous les autres pixels de la région dépendent de la forme de la région et ne sont donc pas identiques d'une région à l'autre. Pour les effectuer, il est nécessaire de connaître la forme des régions, ce qui impose de lire l'image au préalable, ou bien de faire parcourir à chacun des pixels de l'image la totalité de l'image. Dans les deux cas, l'intérêt de la régionalisation est totalement perdu. Dans le premier cas, elle nécessite la connaissance détaillée de chaque point de l'image au préalable (ce qui requiert une extraction de l'image vers une unité de traitement de haut niveau). Dans le second cas, elle nécessite un nombre de déplacements synchrones supérieur au nombre de pixels de l'image.

- Dans le cas de déplacements des plans mémoire de manière à ce que chaque

point de l'image parcoure tous les autres points de l'image, il est nécessaire de marquer l'appartenance d'un pixel à une région afin de n'utiliser localement (dans les racines des régions) que l'information relative à la région traitée.

– Des calculs régionaux redondants sont effectués dans chaque pixel de la région du fait du fonctionnement SIMD, ce qui conduit à une perte énergétique.

De tels défauts ont empêché l'implantation efficace de primitives régionales intégrales sur les machines massivement parallèles[8]. Le cas simple de la reconstruction géodésique a été abordé par Antoine Manzanera [Man00] [Man02] qui conclut par la nécessité d'opérateurs asynchrones de propagation et de connexions programmables. L'ensemble de ce problèmes montre que le mode SIMD utilisant des déplacements massifs de données n'est pas efficace pour les traitements régionaux. Cette inefficacité est principalement due au fait que les données soient déplacées toutes ensemble et de la même manière.

Nous nous intéressons dans ce chapitre uniquement aux déplacements de données, les calculs seront abordés dans un chapitre ultérieur, ils seront pour l'instant supposés synchrones SIMD.

Afin d'avoir un fonctionnement n'ayant pas les limitations présentées précédemment, il est nécessaire de guider les déplacements de données de manière à ce que ceux-ci ne s'effectuent pas de manière identique pour tous les pixels. Ceci revient à doter chaque pixel d'une structure l'informant de la manière dont il va déplacer ses données. A une échelle régionale, le but de cette structure est de réduire les déplacements de proche en proche permettant d'effectuer une opération précise. La donnée de chaque pixel ne sera donc déplacée qu'un nombre limité de fois et selon un chemin donné et pré-déterminé.
Ce type de fonctionnement correspond au cheminement d'informations dans des régions munies d'une structure de graphe permettant de guider les flux de données. Ces graphes constituent une sorte d'*autoroute* pour les données provenant de chaque pixel, ils constitueront donc un support intéressant pour les opérations régionales intégrales.

4.2 Organisation des régions en graphes

Le placement sur chacune des régions d'une structure de type graphe permet de guider les déplacements de données dans des régions.
La forme et la taille de ces régions variant selon les images, il est nécessaire de pouvoir adapter le graphe à la région considérée. Ceci correspond à un besoin de reconfigurabilité. Le voisin destinataire des informations contenues dans un pixel

[8]Le fonctionnement SIMD ne se prêterait bien qu'aux calculs régionaux réalisés par itérations successives à partir d'opérateurs locaux, opérateurs n'entrant pas dans notre champ d'investigation.

devant être connu au préalable, l'état des connexions locales du graphe doit donc être stocké dans chaque pixel de la région. La structure reconfigurable permettant d'établir une connexion d'un voisin vers un autre et permettant le stockage de son état est appelée *connexion programmable*. Les connexions programmables peuvent avoir différentes formes qui seront détaillées à la section 4.4.

L'utilisation des graphes peut se faire de différentes manières en traitement d'image, selon le type de graphe considéré. Les trois types de graphes les plus utilisés sont les suivants :

- Graphe fortement connexe : pour toute paire de noeuds du graphe, il existe une chemin permettant d'aller d'un noeud à l'autre dans le graphe, et ce dans les deux sens (fig. 4.2).
- Graphe symétrique : les connexions existant sont toutes bidirectionnelles (fig. 4.1). Un graphe symétrique est fortement connexe (la réciproque n'est pas vraie).
- Arbre couvrant : graphe orienté en direction d'un point terminal appelé racine, dans lequel tout processeur élémentaire a un et un seul antécédent connecté par une connexion unidirectionnelle, à l'exception du point racine qui n'en a pas (fig. 4.3).

L'utilisation de ces graphes est définie dans le modèle des réseaux associatifs présenté à la section suivante.

4.3 Le modèle des réseaux associatifs

Le modèle des réseaux associatifs, introduit par Alain Mérigot à l'université d'Orsay [Mer97] est défini et formalisé dans les thèses de Siamak Mohammadi [Moh96] et Bertrand Ducourthial [Duc00]. Dans cette section, nous rappelons ses éléments essentiels et sa philosophie sans présenter à nouveau son formalisme.
Le modèle des réseaux associatifs permet de définir des régions, ensembles de processeurs élémentaires de taille et de forme arbitraire afin de leur appliquer des primitives régionales appelées *associations*. Ces ensembles de processeurs forment des composantes connexes qui sont définies par un graphe dont les noeuds sont connectés directement ou indirectement par des arcs. Ces graphes sont des sous-graphes du graphe d'interconnexion physique des processeurs, ils sont définis localement par l'état des connexions programmables entre un processeur local et les processeurs voisins. Les particularités du modèle des réseaux associatifs sont les suivantes :

- Les primitives régionales ne sont appliquées qu'à des processeurs faisant partie d'une même composante connexe (région).

- Les données issues de chaque processeur sont combinées grâce à des opérateurs logiques ou arithmétiques associatifs, ce qui permet de garantir l'indépendance du résultat quelque soit l'ordre dans lequel sont traitées les variables. Grâce au réseau d'interconnexion, il est possible d'effectuer des opé-

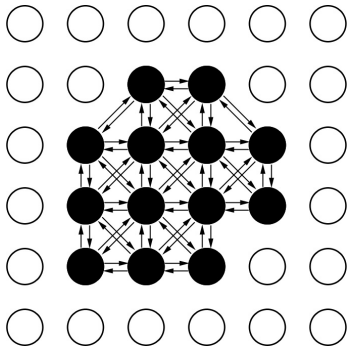


FIG. 4.1 – Graphe symétrique

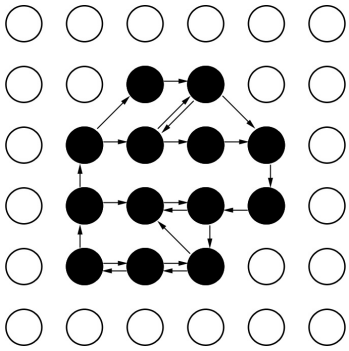


FIG. 4.2 – Graphe fortement connexe

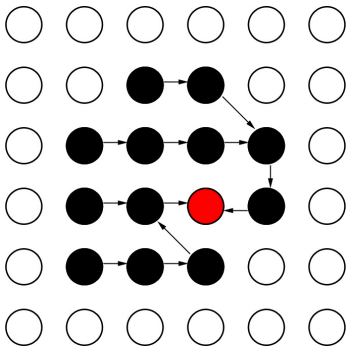


FIG. 4.3 – Arbre couvrant

rations régionales distribuées sur les données.

Dans le modèle des réseaux associatifs, deux types principaux d'*associations* sont définis :

- *association* lorsqu'elle est exécutée sur un graphe symétrique (les connexions entre paires de processeurs sont bidirectionnelles, cf fig.4.4)

- *prefix-association* lorsqu'elle est exécutée sur un arbre couvrant (cf fig.4.5).

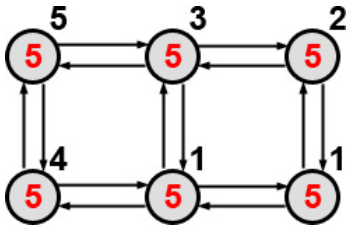


FIG. 4.4 – Example d'association utilisant l'opérateur max (les données des pixels sont en noir, le résultat de l'association en rouge)

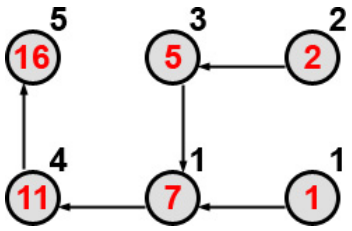


FIG. 4.5 – Example de *prefix-association* utilisant l'opérateur addition (les données des pixels sont en noir, le résultat de la prefix-association en rouge)

Ces deux types de primitives régionales sont fondamentalement différentes, par la structure des graphes sur lesquels elles reposent et par les opérateurs locaux qu'elles utilisent.
Une *association*, qu'elle soit *prefix* ou non, est associée à un opérateur. **Elle retourne dans chaque processeur de la région une valeur correspondant à l'itération de l'opérateur sur chacun des *ancêtres* du processeur**. Les *ancêtres* sont définis comme l'ensemble des processeurs reliés au processeur considéré par un chemin de connexion orienté de l'ancêtre vers le processeur considéré.

– Dans le cas d'une *association* non *prefix*, le graphe des connexions est bidirectionnel entre chaque paire de processeurs adjacents. Par récurrence, toute paire de processeur de la région est connectée par un réseau bidirectionnel. Ainsi l'ensemble des ancêtres d'un processeur de la région est l'ensemble des processeurs de cette région. Le résultat de l'*association*, qui ne dépend que des ancêtres et de l'opérateur considéré, est nécessairement le même dans chacun des pixels puisque tous les pixels ont le même ensemble d'ancêtres. Une *association* permet donc d'avoir un résultat identique sur chacun des processeurs de la région considérée.
En pratique, l'implantation d'une primitive *association* n'est cependant pas définie comme dans le modèle des réseaux associatifs. Celui-ci spécifie que le résultat est obtenu par itération de l'opérateur sur les données de chacun des processeurs élémentaires (une fois et une seule sur chaque donnée). Cependant une telle implantation est contraignante car elle nécessiterait non seulement de déplacer les données à travers chaque processeur une fois et une seule en finissant par un processeur donné (ce que l'on peut obtenir avec une structure d'arbre couvrant), mais surtout de faire cela pour chaque processeur. Compte tenu du résultat théorique que l'on doit obtenir, à savoir le même résultat dans chacun des processeurs, la solution utilisée en pratique est de limiter le champ des opérateurs utilisables dans les *associations* aux opérateurs idempotents, et de laisser le réseau se stabiliser seul. L'*association* réalisée dans ce cadre d'application est alors appelée *direct-association* [Duc00]. Le résultat obtenu est le même que celui que l'on pourrait obtenir en utilisant le modèle théorique, mais la contrainte très forte de ne passer dans chaque processeur qu'une seule fois n'est plus présente.
Les primitives régionales correspondant à ce compromis sont typiquement les

fonctions logiques (OR, AND), le maximum et le minimum.

– Dans le cas d'une *prefix-association*, les processeurs étant connectés selon une structure de type arbre couvrant, les ancêtres d'un processeur sont différents d'un processeur à l'autre, de fait le résultat de la *prefix-association* peut être différent d'un processeur à l'autre dans l'arbre couvrant.
Outre le fait que le résultat dépende du processeur considéré, un des fondements de la *prefix-association* est l'utilisation de la structure d'arbre couvrant. Celle-ci permet en effet d'obtenir dans la racine le résultat de l'*association* non *prefix* sur la région couverte par l'arbre. En effet, l'opérateur associatif sera appliqué une fois et une seule en chacun des pixels de la région en partant des feuilles (processeurs situés à l'extrémité du graphe) vers la racine de l'arbre.
La *prefix-association* sera donc utilisée pour implanter deux types de primitives :

1. les *associations* avec des *opérateurs non idempotents*, le résultat devant être redistribué à chaque pixel de la région ensuite (grace à un opérateur max par exemple). La primitive fondamentale de ce type la plus importante pour le traitement d'image est la somme régionale.
2. les *prefix-associations* pour lesquelles le résultat n'est pas le même dans chacun des processeurs, et qui sont bien moins utiles que la primitive décrite précédemment.

Nous appelons *modèle limité des réseaux associatifs* la limitation du modèle des réseaux associatifs aux primitives de *direct-association* et de *prefix-association*. Dans la suite de cette thèse, nous n'envisagerons que des implantations asynchrones de la direct-association et de la prefix-association. Il est possible d'envisager des implantations synchrones de ces primitives (car celles-ci n'ont besoin que de la structure de graphe). Toutefois, ces implantations synchrones sont inefficaces d'un point de vue énergétique (les opérateurs fonctionnent en permanence alors qu'ils ne sont utiles uniquement lorsque des données sont présentes). Le passage à des opérateurs combinatoires (sans mémorisation asynchrone) ou asynchrones permet de résoudre ce problème.

4.4 Implantation des connexions programmables

Les connexions programmables peuvent être implantées de différentes manières. Elles sont définies comme étant la structure reconfigurable permettant d'établir une connexion d'un voisin vers un autre et permettant le stockage de son état. Afin d'étudier le coût matériel des connexions programmables nous en étudions les principales implantations possibles. Nous recensons également les principaux types de réseau de connexion que l'on peut trouver dans la littérature.
Un premier type d'implantation est l'utilisation d'un transistor NMOS piloté par

une cellule mémoire, lorsque la cellule est à l'état logique 1, le transistor est passant et transmet les informations, lorsque la cellule est à l'état logique 0, le transistor est bloqué il se comporte comme une connexion ouverte (fig. 4.6). Un telle implantation a le désavantage de transmettre mieux les 0 que les 1 logiques à cause de l'utilisation d'un transistor NMOS. Il est donc nécessaire dans ce cas que la connexion programmable soit branchée à un circuit de forte impédance. Ce type de connexion programmable est celui qui est utilisé dans les micropipelines associatifs qui seront introduit plus tard dans la thèse. La connexion programmable étant branchée en entrée d'un C-élément (connexion programmable sur le signal *ack*) ou d'un arbitre (connexion programmable sur le signal *req*), la condition de forte impédance est respectée (connexion programmable connectée à une grille de transistor).
Le deuxième type d'implantation (utilisé dans la Maille Associative d'Orsay) uti-

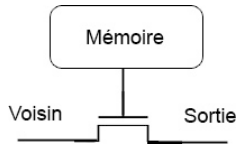


FIG. 4.6 – Connexion programmable à base de transistor NMOS

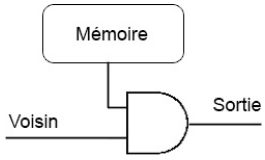


FIG. 4.7 – Connexion programmable à base de porte *ET*

lise une porte logique de type *ET* logique à deux entrées (fig. 4.7). Une entrée est connectée au signal provenant du voisin, la seconde est connectée à la cellule mémoire stockant l'état de la connexion. Cette structure a l'avantage de jouer le rôle d'un buffer CMOS, et d'assurer l'adaptation d'impédance. Son inconvénient est son coût en transistors assez élevé, 6 transistors contre 1 pour l'implantation précédente. De plus, le temps de propagation de l'information dans la deuxième structure est plus important que dans la première.

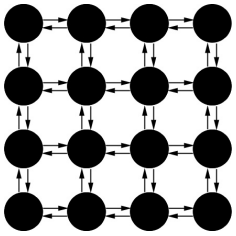


FIG. 4.8 – Maille 4-connexe

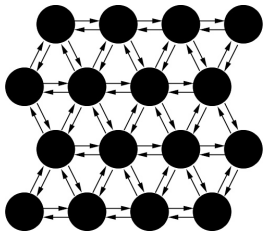


FIG. 4.9 – Maille 6-connexe

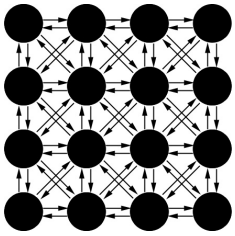


FIG. 4.10 – Maille 8-connexe

Outre le coût d'implantation d'une connexion programmable, le nombre de connexions programmables utilisées par pixel ou processeur élémentaire est important. Ce nombre de connexions programmables dépend de la connexité de la maille utilisée. Les connexités les plus communes sont la 4, la 6 et la 8-connexité (fig. 4.8, 4.9 et 4.10). Il est à noter que la connexité de la maille n'est pas forcément égale au nombre de connexions programmables nécessaires. Une augmentation du nombre

connexions programmables pourrait être utile pour diminuer le coût des ressources matérielles implantées au sein de chaque processeur.

5

De la nécessité de l'asynchronisme

5.1 Limitations du modèle des réseaux associatifs en fonctionnement synchrone SIMD

Le chapitre précédent a montré que l'utilisation des graphes comme support du modèle des réseaux associatifs permet d'ordonner les déplacements de données liés aux calculs régionaux en plaquant une structure quasi-linéique et orientée (un arbre couvrant) sur une surface. Cependant l'amélioration n'est obtenue que pour les déplacements de données (nous ne nous sommes pas intéressés aux calculs dans le chapitre précédent). Les calculs restant de type synchrones SIMD, ils doivent être effectués sur chaque pixel de l'image simultanément. Cependant, les pixels où doivent se faire les opérations locales destinées à servir le calcul régional sont peu nombreux. Ils correspondent aux pixels où une donnée vient d'arriver en provenance d'un voisin. Or, comme nous l'avons vu dans la première partie de la thèse, l'envoi d'instructions lorsque le taux d'occupation des processeurs est faible remet en cause l'intérêt énergétique des rétines. Il est donc préférable d'effectuer les déplacements de données sans avoir recours à des instructions extérieures à la rétine. Une manière de résoudre ce problème est d'utiliser des opérateurs de calculs se déclenchant uniquement lorsqu'une donnée arrive sur une de leurs entrées. Ce mode de fonctionnement est caractéristique des opérateurs asynchrones que nous étudions dans ce chapitre.

Outre l'aspect économie énergétique, nous verrons que l'asynchronisme permet d'améliorer les performances en terme de vitesse de propagation sur un graphe. En effet, le temps total de propagation sur un arbre est proportionnel au temps de traversée d'un processeur élémentaire (incluant le traitement des données). Ce temps est dans le cas asynchrone le temps de traversée moyen d'un processeur élémentaire. Dans le cas synchrone, il est supérieur au temps de traitement et propagation maximum d'un processeur élémentaire. Le fait de passer du synchronisme à l'asynchronisme permet donc d'avoir un coût temporel caractéristique d'un cas moyen au lieu d'un pire cas.

Bien que l'asynchronisme ait beaucoup d'avantages, nous verrons qu'il a également des inconvénients, en particulier un coût d'implantation élevé très pénalisant pour une implantation au sein d'une rétine artificielle. Dans la dernière section de ce chapitre nous proposerons donc une répartition entre synchronisme et asynchronisme spécifiquement adaptée au cadre des rétines artificielles.

5.2 Principe de fonctionnement de l'asynchronisme

La logique asynchrone est apparue au milieu des années 1950 avec l'avènement des circuits intégrés numériques. En raison de sa grande difficulté de conception, elle fût rapidement abandonnée au profit de la logique synchrone. Après 50 ans de quasi absence sur un marché dominé par la logique synchrone, les évolutions technologiques récentes et les besoins toujours accrus en terme de performance ont fait renaître l'intérêt de la logique asynchrone [Ren00a]. Nous commençons cette partie dédiée aux généralités sur l'asynchronisme par les concept fondamentaux de la logique asynchrone.

Dans le cadre des systèmes intégrés tels que les rétines artificielles ou les processeurs, le terme « asynchrone » qualifie les échanges d'information entre éléments constitutifs du système. Ces échanges d'information peuvent être de plusieurs types, échange de données ou échange de signaux de contrôle. Il est possible de faire cohabiter dans un même circuit des parties synchrones et asynchrones. Dans les parties synchrones, les échanges d'information sont déclenchés par une horloge globale assurant la synchronisation. Dans les parties asynchrones, les échanges d'information peuvent se faire à tout instant.
L'association d'éléments de circuits synchrones mis en parallèle à l'aide de moyens de communication asynchrones a été utilisée dans les circuits de type GALS (Globally Asynchronous, Locally Synchronous). Un des atouts de ces circuits est de pouvoir ajouter aisément des blocs fonctionnels sans avoir à optimiser le routage d'une horloge globale. Notre approche, comme nous le montrerons plus loin est fondamentalement différente dans la mesure où nous effectuons en parallèle des opérations asynchrones à l'intérieur de chaque région, synchronisées entre elles par une horloge externe. Il s'agit donc dans ce cas d'un fonctionnement que l'on pourra qualifier de GSLA (Globally Synchronous, Locally Asynchronous).

Asynchronisme ne signifie pas anarchie. Si les échanges d'informations peuvent se faire à tout instant, l'ordre de ces échanges est quant à lui parfaitement déterminé afin d'assurer le bon développement d'un algorithme. Ainsi, une des différences fondamentales entre un système synchrone et un système asynchrone est la suivante : les échanges d'information (dont l'ordre est parfaitement défini) sont déclenchés par une horloge globale dans le cas synchrone et par un mécanisme local de synchronisation dans le cas d'un système asynchrone. Le mécanisme global de déclenchement des échanges synchrones pose donc une contrainte temporelle importante, le temps entre deux déclenchements doit être supérieur au temps maximal mis par

un échange d'information, sans quoi cet échange est interrompu avant son terme, et le résultat incertain. A l'opposé, les échanges asynchrones sont insensibles au temps mis par un échange d'information puisqu'ils attendent la fin de l'exécution de l'échange précédent pour démarrer. On parle de circuits et système insensibles aux délais [Hau95].

5.2.1 Le C-élément : structure de base de l'asynchronisme

Dans les circuits synchrones, l'horloge déclenche les échanges d'information. A chaque top d'horloge, les signaux nécessitent d'être valides et stables. Entre deux tops, les signaux peuvent en revanche avoir un état plus ou moins bien défini et pouvant osciller avant de se stabiliser à une valeur. Ces états intermédiaires sont appelés *aléas*, ils ne posent pas problème à condition qu'ils soient stabilisés lors des fronts d'horloges déclenchant les échanges d'information.
Dans les circuits asynchrones, la situation est différente car il n'y a plus d'horloge. Les échanges d'informations ne sont donc pas déclenchés par un signal global. Les échanges d'information doivent être déclenchés localement à chaque fois qu'un nouvel échange devient possible.

Le point fondamental du mode de fonctionnement asynchrone est que les échanges d'information sont déclenchés localement. Les différentes parties du circuit se synchronisent localement afin d'échanger des informations indépendamment des autres parties du circuit auxquelles elles ne sont pas connectés. Une fois l'échange d'information effectué, une nouvelle connexion avec une autre partie du circuit est établie en fonction de la fonction globale à effectuer. Ce mécanisme garantit la synchronisation et la causalité des événements au niveau local et un fonctionnement valide au niveau global. Le contrôle local doit en conséquence remplir les fonctions suivantes :

- Acquérir les informations entrantes.
- Informer les prédécesseurs que leurs données ont été utilisées.
- Déclencher les traitements localement quand toutes les informations sont disponibles (rendez-vous).
- Fournir aux successeurs les valeurs retournées.
- Invalider ces valeurs une fois qu'elles ont été utilisées par le successeur.

Cette gestion locale de l'ordonnancement des échanges repose sur une cellule permettant de réaliser la fonction *rendez-vous*. Cette cellule est le C-élément ou porte de Muller (fig. 5.1). Propre aux circuits asynchrones, elle permet de créer la synchronisation entre plusieurs signaux. Lorsque toutes les entrées sont à 0 la sortie passe à 0 ; lorsque toutes les entrées sont à 1 la sortie passe à 1 ; pour les autres combinaisons des entrées, la sortie reste inchangée (fig. 5.2). Couplée à d'autres cellules du même type, elle permet d'établir un protocole de poignées de main (*handshake*), utilisé entre autres dans la structure de contrôle des micro-pipelines.

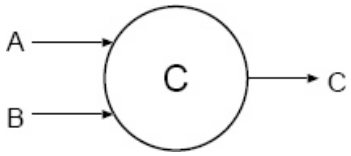


FIG. 5.1 – C-élément

A	B	C
0	0	0
0	1	C
1	0	C
1	1	1

FIG. 5.2 – Table de vérité d'un C-élément

5.2.2 Modes de transmission asynchrone

Un *mode* de transmission asynchrone utilise un support physique de communication appelé *canal* associé à un ensemble de règles de communication appelé *protocole de communication*.
Un canal de transmission asynchrone est constitué d'un ensemble de structures de contrôle et de transmission de données assurant le bon déroulement des échanges d'information. Les formes usuelles que peuvent prendre ce canal sont le canal à donnée groupée, et le canal double rail.
Un protocole de transmission asynchrone décrit la succession fonctionnelle des opérations (phases) intervenant dans l'échange d'informations sur un canal. Les protocoles de transmission les plus courants sont le protocole à 2 phases et le protocole à 4 phases.

Cette partie rappelle les modes de transmission asynchrone les plus couramment utilisés.

Canal à donnée groupée avec protocole de transmission à 4 phases

Ce mode de transmission utilise un canal dans lequel les signaux de contrôle permettant d'effectuer les opérations de type poignées de main sont codés sur 2 voies séparées du bus de données. Ces deux voies sont appelées *Requête (req)* pour la voie sur laquelle circulent les demandes de transmission et *Acquittement (ack)* pour la voie ou circulent les accusé de réception. Les données sont groupées (*bundled data*) dans un bus et chaque bit est porté par un unique fil du bus de données. Pour cette raison, un tel canal de transmission est appelé également *simple rail* (fig. 5.3).
Le protocole de transmission à 4 phases associé au canal de transmission à donnée

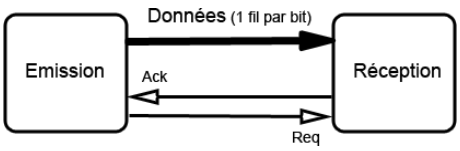


FIG. 5.3 – Canal à donnée groupée

groupée est appelé protocole *Return To Zero* (RTZ), car au début de la transmission

de toute information les signaux *ack* et *req* sont à l'état 0. Le protocole est le suivant (fig. 5.4) :

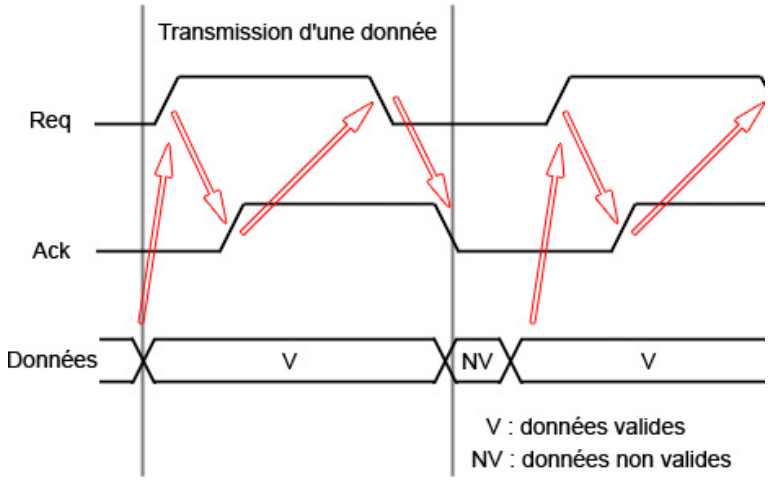


FIG. 5.4 – Protocole de transmission à 4 phases sur canal à donnée groupée

1. Une fois la donnée à transmettre (placée par l'émetteur sur le bus de données) stabilisée dans un état dit valide, l'émetteur émet une requête pour demander le transfert en mettant le signal de requête *req* à 1.
2. Le récepteur reçoit la requête, récupère les données, et renvoie un accusé de réception à l'émetteur en plaçant le signal d'acquittement *ack* à 1.
3. L'émetteur reçoit l'accusé de réception, et met le signal *req* à 0 pour indiquer qu'il invalide la donnée.
4. Le récepteur ayant terminé la réception place le signal d'acquittement *ack* dans l'état 0 pour indiquer qu'il est prêt pour une nouvelle transmission.

Le protocole de transmission décrit indique uniquement les transitions qui s'effectuent sur les signaux de contrôle et qui décrivent un protocole de poignée de main. Le transfert des données est quant à lui déclenché par le récepteur. Celui-ci reçoit une demande de transfert lorsque les traitements effectués à la cellule précédente de la chaîne de transmission sont terminés. Il effectue alors le transfert des données (verrouillage des données) avant d'envoyer un accusé de réception. Compte tenu du fait qu'il soit indispensable que les traitements effectués à la cellule précédente soient terminés lors de la transmission des informations, on introduit un délai dans la chaîne de contrôle asynchrone. Ce délai doit être supérieur au temps de traitement des données.

Sur le plan de l'implantation électronique, comme nous le verrons plus tard, un protocole de transmission à 4 phases est assez aisé à mettre en oeuvre et son coût matériel est assez réduit. Ceci provient en partie du fait que les transmissions sont initiées sur les fronts montants du signal de demande de transfert.

Canal à donnée groupée avec protocole de transmission à 2 phases

La structure du canal est la même que celle décrite précédemment. Le protocole de transmission est quant à lui différent. Nous avons vu précédemment que dans le protocole de transmission à 4 phases, la transmission effective des données a lieu entre la réception de la demande de transfert et l'envoi de l'accusé de réception. Les phases 3 et 4 du protocole ne servent donc à rien sur le plan du transfert des données. Le principe de la transmission à 2 phases consiste donc à utiliser ces deux dernières phases pour transmettre une nouvelle donnée. Le protocole est le suivant (fig. 5.5) :

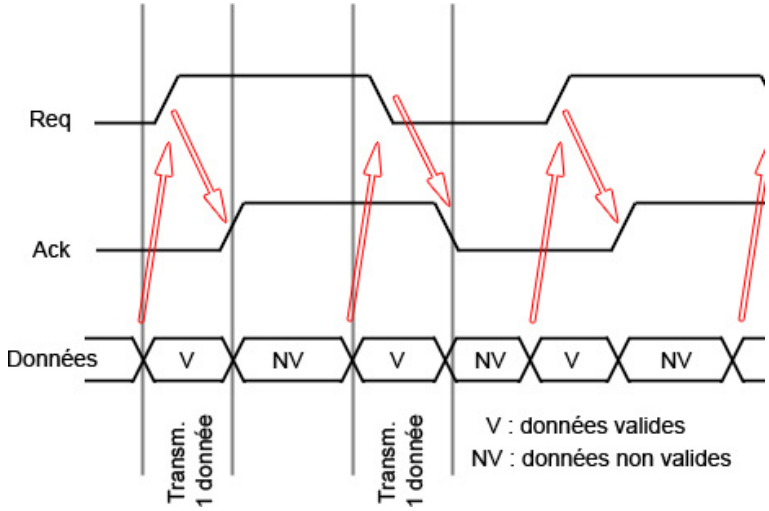


FIG. 5.5 – Protocole de transmission à 2 phases sur canal à donnée groupée

1. Une fois la donnée à transmettre (placée par l'émetteur sur le bus de données) stabilisée dans un état dit valide, l'émetteur émet une requête pour demander le transfert en inversant le signal *req* ($0 \rightarrow 1$ ou $1 \rightarrow 0$).
2. Le récepteur reçoit la requête, récupère les données, et renvoie un accusé de réception à l'émetteur qui inverse l'état du signal d'acquittement *ack* ($0 \rightarrow 1$ ou $1 \rightarrow 0$). Les données présentes sur le bus de données sont alors invalidées.

Le protocole de transmission asynchrone à 2 phases permet donc de transmettre en théorie deux fois plus d'informations, et le coût énergétique de transfert associé au bus de contrôle est réduit de moitié (car le nombre de transitions est réduit de moitié). Ce résultat est en réalité bien nuancé car la logique de traitement des requêtes et accusés de réception doit être sensible aux fronts montants et descendants, ce qui la complique beaucoup. Les conséquences de cette complexité sont une réduction de la vitesse de transmission, une hausse de l'énergie consommée, et une augmentation de la surface de silicium occupée. En conséquence, aujourd'hui, le protocole de transmission à 4 phases est privilégié.

Canal à double rail avec protocole de transmission à 4 phases

Nous avons vu qu'un des inconvénients lié à la séparation du bus de données et des bus de contrôle est la nécessité d'insérer des temps d'attentes entre la réception d'une demande de transfert et l'envoi de l'accusé de réception. Afin de s'affranchir de cette contrainte, une idée est de faire passer les données directement par le fil de requête *req*. Mais cette idée se heurte au fait qu'une requête est caractérisée par une transition sur *req*. Cette transition permet de coder la présence d'une donnée mais pas sa valeur. Pour remédier à ce problème, on utilise deux fils de requête par donnée, le premier portant l'information demande de transmission d'un 0 logique, le second portant l'information demande de transmission d'un 1 logique. La nécessité de coder chaque bit de donnée sur 2 fils justifie l'appellation *double rail*.
Un canal numérique est donc constitué de $2n$ fils permettant de transmettre les n bits de donnée, et d'un fil permettant l'envoi des accusés de réception (*ack*) (fig. 5.6). Le protocole de transmission à 4 phases associé à ce canal est le même que

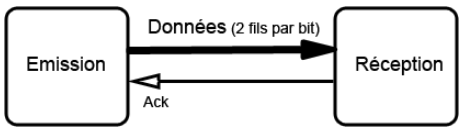


FIG. 5.6 – Canal double rail

celui qui est associé au bus à donnée groupée (à la différence près que les requêtes portent de l'information). Il se compose des phases suivantes (fig. 5.7) :

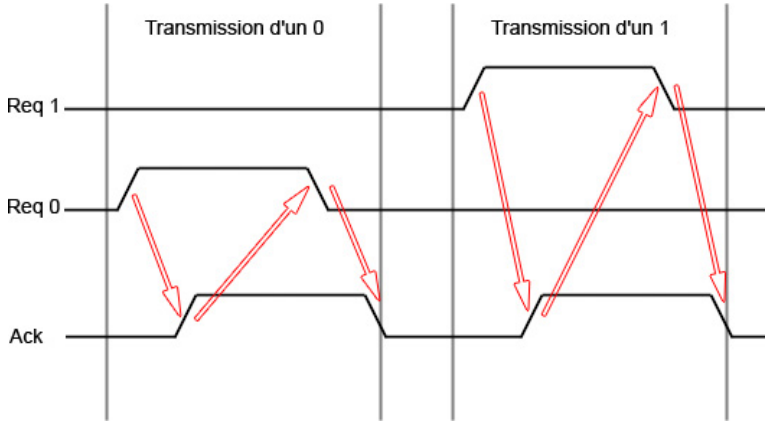


FIG. 5.7 – Protocole de transmission à 4 phases sur canal double rail

1. Une donnée à transmettre est placée par l'émetteur sur le fil de requête correspondant à la valeur à transmettre (*Req 0* pour transmettre un 0, *Req 1* pour transmettre un 1).
2. Le récepteur reçoit la requête, et renvoie un accusé de réception à l'émetteur en plaçant le signal d'acquittement *ack* à 1.

3. L'émetteur reçoit l'accusé de réception, et place le signal de requête dans l'état 0.
4. Le récepteur ayant terminé la réception place le signal d'acquittement *ack* dans l'état 0 pour indiquer qu'il est prêt pour une nouvelle transmission.

5.2.3 Classification des différents types de circuits asynchrones

Les circuits asynchrones peuvent être classés selon les hypothèses temporelles sur lesquelles ils reposent.

Circuits insensibles aux délais (DI)

La caractéristique d'un circuit insensible aux délais (DI pour Delay Insensitive) est que son fonctionnement ne dépend pas des délais introduits par les fils et les portes logiques. Ce délais peuvent être arbitrairement longs. Dans un tel mode de fonctionnement le récepteur accuse réception des informations une fois qu'il a terminé de recevoir celles-ci, et non pas au bout d'un temps borné. L'émetteur quant à lui attend d'avoir reçu l'accusé de réception pour initier une nouvelle transmission. Dans un tel circuit, la communication se fait de manière optimale, et les performances en terme de vitesse sont théoriquement les meilleures que l'on puisse obtenir. Toutefois, il est extrêmement difficile de générer ce type de circuit, leur complexité augmentant plus rapidement que leur fonctionnalité. De tels circuits doivent donc être limités à des fonctions simples. Selon [Sli04], il n'existe pas de circuit DI de taille importante. Nous verrons dans la suite de cette thèse qu'il est possible d'utiliser cette forme pure d'asynchronisme pour effectuer des calculs régionaux avec un coût matériel faible, toutefois pour des raisons d'économies matérielles, une solution de type délai borné sera finalement adoptée.

Circuit indépendant de la vitesse et quasi-insensibles aux délais (QDI)

La caractéristique d'un circuit quasi-insensible aux délais (QDI pour quasi Delay Insensitive) est que son fonctionnement, comme celui des circuits DI, ne dépend pas des délais introduits par les portes logiques et par les fils. L'hypothèse supplémentaire qui est faite est que les délais introduits par les fils obéissent au principe de fourche isochrone : les différences de temps de réception d'un signal par plusieurs récepteurs sont négligeables.

Circuits de Huffman

Le modèle de délai utilisé dans les circuits de Huffman est celui des circuits synchrones : les délais sont supposés bornés. Les transmission et opérations asynchrones déclenchent simultanément des opérations sur les données et des attentes sur les signaux de contrôle. Ces attentes doivent permettre à la sortie des opérateurs de se stabiliser, elles doivent donc être supérieures ou égales au délai maximum introduit par l'opération exécutée. Ce type de fonctionnement correspond au cas des micropipelines introduits par Sutherland et qui sont détaillés à la section suivante.

5.2.4 Une structure permettant les opérations asynchrones : les micropipelines

L'asynchronisme repose comme nous l'avons vu précédemment sur des échanges d'informations moyennant certaines hypothèses temporelles. Parmi les circuits permettant d'assurer ces échanges d'informations figurent les micropipelines. Introduits dans leur forme *à donnée groupée* (Bundled Data) par Ivan E. Sutherland [Sut89], ils sont une alternative aux pipelines élastiques synchrones. Il existe diverses formes de micropipelines, chaque forme étant associée à un des types d'asynchronisme rappelés précédemment.

Micropipelines d'Ivan E. Sutherland

Dans leur forme originale [Sut89], les micropipelines sont composés d'une partie contrôle de type insensible aux délais (DI) dans laquelle ont été introduits des retards, et d'un chemin de données groupées de type délai borné. L'ensemble fonctionne donc avec l'hypothèse temporelle de type délais bornés (circuit de Huffman). La partie contrôle des micropipelines est constituée de porte de Muller (C-élément) placés tête-bêche et assurant une transmission de signaux de contrôle de type DI. Elle est représentée à la figure 5.8. Le protocole de transmission des signaux de

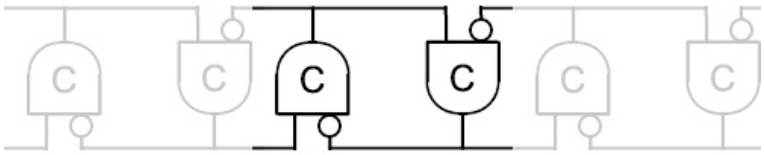


Fig. 5.8 – Structure de contrôle des micropipelines

contrôle est le protocole à 4 phases sur bus à données groupées (fig. 5.4). Comme indiqué précédemment, la transmission de données sur le bus de données impose de rajouter plusieurs éléments dans la structure. Le premier est un registre permettant au récepteur de mémoriser les informations placées sur le bus de données par les sorties des opérateurs de traitement de l'émetteur lorsque celui-ci envoie un signal de demande de transmission. Le deuxième est un retard permettant d'assurer la terminaison des traitements dans l'émetteur avant la réception par le récepteur de la demande de transmission. La figure 5.9 montre l'agencement de ces différents éléments dans la structure de micro-pipeline.

Autre formes de micropipelines

L'hypothèse des délais bornés sur laquelle repose les micropipelines de Sutherland est une hypothèse très limitative. En effet, une grande partie de l'intérêt de l'asynchronisme est perdue dans la mesure ou l'on est obligé d'évaluer le temps maximal (pire cas) nécessaire au cheminement des données pour ajuster les délais

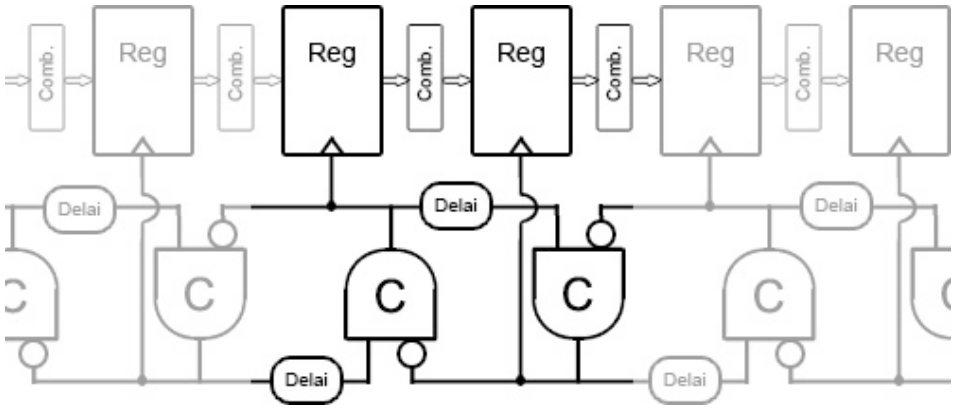


FIG. 5.9 – Structure du micropipeline d'Ivan E. Sutherland

placés dans la structure de contrôle.

Cette structure initiale à délais bornés a ouvert la voie à de nombreuses réalisations basées sur des hypothèses temporelles plus intéressantes telles que l'hypothèse QDI. Nous pouvons par exemple citer les travaux de A. Martin et A. Lines [MLM+97] [Lin96], de Choy [CBC01] dont le micropipeline LDA (fig. 5.10) est une des nombreuses structures proposées dans la littérature permettant de travailler dans l'hypothèse QDI.
Nous pouvons également citer les structures proposées à l'USC (University of Sou-

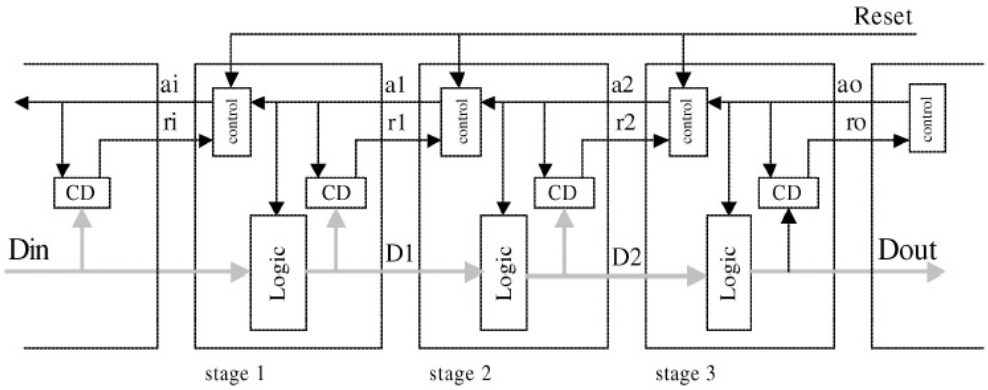


FIG. 5.10 – Locally distributed asynchronous (LDA) micropipeline

thern California) [OB02] [Ozd03] (fig. 5.11, 5.12 et 5.13). Ces différents circuits (WCHB, PCHB et RSPCHB) reposent sur l'hypothèse QDI. On peut constater sur les figures 5.11, 5.12 et 5.13 que la complexité électronique est beaucoup plus importante que dans le cas du micropipelines d'Ivan E. Sutherland. Cette complexité permet d'obtenir des hypothèses temporelles plus intéressantes (grâce au déclenchement du signal request par l'arrivée de données par exemple), et un fonctionnement beaucoup plus rapide. Le différences entre ces différents circuits sont

essentiellement liées à la vitesse de propagation dans ces structures. Par exemple, le RSPCHB (fig. 5.13) est par exemple plus rapide que le PCHB (fig. 5.12) mais a un coût en transistors plus élevé.

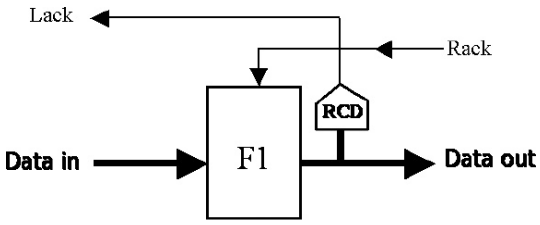


FIG. 5.11 – WCHB [Ozd03]

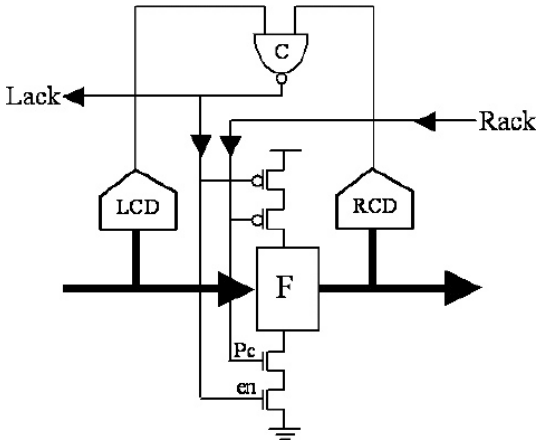


FIG. 5.12 – PCHB (Pre-Charged Half-Buffer) [Ozd03]

Les travaux sur les micropipelines ne se limitent cependant pas à la recherche de nouvelles structures, des améliorations des circuits de contrôle sont également proposées [TB98] [YBA96] ainsi que des exemples d'utilisation de ces structures [Hau95] [NMM98] pour des applications particulières telles que par exemple les DSP [CPPC98].

Il existe dans la littérature de nombreuses implantations QDI ayant des performances variables en terme de vitesse. On constate que la vitesse dépend le plus souvent de la complexité de la structure utilisée, une vitesse élevée nécessitant une structure de contrôle complexe. Dans le cadre des rétines artificielles, cela pose un problème fondamental à l'utilisation des micropipelines, car les ressources matérielles sont très limitées. Cependant, si l'on revient au micropipeline de Sutherland, on constate que sa structure de contrôle est de type DI, et que cette dernière a un coût matériel très réduit. Dans la suite de cette thèse, nous verrons que l'utilisa-

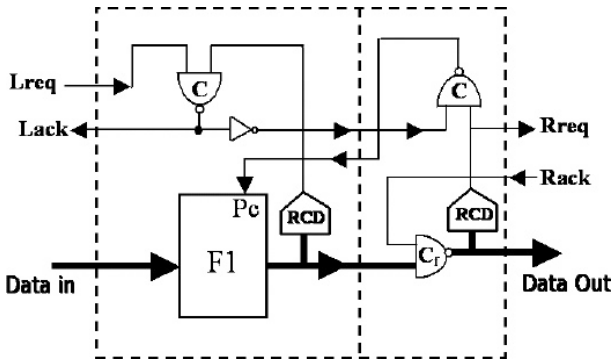


FIG. 5.13 – RSPCHB (Pre-Charged Half-Buffer) [Ozd03]

tion de cette structure de contrôle peut être un moyen efficace et peu coûteux de transmettre de l'information de manière asynchrone.

5.2.5 Avantages et inconvénients de l'asynchronisme

La différence fondamentale entre les circuits asynchrones et les circuits synchrones est le mode de déclenchement des échanges d'informations, qu'ils soient de type calcul ou déplacement de données. La synchronisation globale réalisée par une horloge dans le cas des systèmes synchrones est ici réalisée par un système de communication local de type poignée de main (ou "handshaking" en anglais) entre les différents composants du circuit. Cette communication particulière permet d'assurer le transfert et la synchronisation de données entre les composants dès que ce transfert est possible. Cependant cette communication locale a un coût matériel et énergétique (faible) dans la mesure où elle nécessite l'ajout de signaux de contrôle : le signal de requête et le signal d'acquittement.

Ce mode de gestion délocalisée du contrôle apporte un certain nombre d'avantages et d'inconvénients que nous détaillons ici. Ces avantages et inconvénients généralement reconnus sont décrits de manière claire dans [Rez04] [Sli04] [Ren00a]. Les avantages de la conception asynchrone se présentent à différents niveaux :

– **Optimisation du temps de calcul** : Le temps mis par un opérateur pour déplacer ou traiter une donnée n'est pas le même pour chacune des opérations à effectuer dans tout le circuit. Dans un système synchrone, afin d'assurer la validité des données en sortie de chacun des opérateurs, il est nécessaire d'attendre que le signal d'horloge global atteigne l'opérateur, qu'il le déclenche et que le calcul se termine.
  Le premier facteur correspondant à la différence de temps d'arrivée de l'horloge partant d'une source unique, appelé *clock skew*, peut être en partie résolu lors de la phase de conception synchrone. Il faut pour cela utiliser des techniques de placement routage optimisée en forme d'arbre de manière à diminuer ces écarts de temps à l'arrivée de l'horloge dans chacun des compo-

sants du circuit. Il est à noter que ce problème devient aujourd'hui essentiel dans la conception de circuits car les temps de propagation des horloges sont devenus supérieurs aux délais des portes.
Le deuxième facteur est la différence de temps de calcul dans les différents composants du circuit. Variant énormément d'un composant à l'autre selon les fonctions réalisées, il est très difficile d'égaliser ces temps de calcul lors de la conception du circuit. Cela peut se faire en partie par l'utilisation d'opérateurs répartis ou en divisant une fonction ayant un temps de calcul important en plusieurs sous fonctions connectées entre elles par un pipeline synchrone. Il n'est cependant pas possible de diviser de la sorte toutes les fonctions, et cette division contraint à réorganiser tout ou partie du circuit lors de la phase de conception.

La conséquence de ces décalages temporels entre les sorties des différents composants est qu'il faut attendre que le dernier des composants ait fini son calcul pour relancer une nouvelle opération de manière globale. L'horloge est donc obligée de fonctionner en regard du chemin critique, à savoir le chemin le plus lent. La vitesse de fonctionnement d'un circuit synchrone dépend donc de $\max_{circuit}(\tau_{\text{clock delay+gate delay}})$ où $\tau_{\text{clock delay+gate delay}}$ est la somme du temps mis par le signal d'horloge pour arriver au composant et du temps mis par le composant pour faire son calcul. Il est également à noter qu'une optimisation des chemins non critiques ne sert pas à grand chose en matière de coût temporel dans un circuit synchrone.

Dans les circuits asynchrones, la gestion du transfert des données et l'exécution des instructions étant gérés localement, il n'y a plus de contrainte forte liée à un contrôle global comme il est question dans les circuits synchrones. En conséquence, le temps d'exécution de la fonction correspondant à un opérateur est exactement celle de l'opérateur. On peut en effet négliger les temps relatifs aux signaux de contrôle, ceux-ci étant locaux, ils n'ont pas de propagation à effectuer. En conséquence, la vitesse du circuit est en permanence ajustée aux capacités temporelles des fonctions effectuées, et la vitesse obtenue est donc la meilleure possible. La vitesse de fonctionnement d'un circuit asynchrone dépend donc de la valeur moyenne de $\tau_{\text{gate delay}}$ sur l'ensemble des cellules considérées. Dans un circuit asynchrone, l'optimisation des chemins non critiques permet d'améliorer la système car ils sont pris en compte dans la moyenne des temps de traitement.

En terme de vitesse, un circuit asynchrone a donc potentiellement des performances améliorées par rapport à un circuit synchrone. Ce résultat vérifié doit toutefois être nuancé par la dégradation des performances de certains composants, dégradation due à une complexification du fonctionnement introduite par l'asynchronisme.

– **Consommation d'énergie** : Une des raisons du retour sur le devant de la

scène des circuits asynchrones est leur capacité à travailler à basse énergie. Il y a deux raisons fondamentales à cela. La première est qu'ils cessent leur activité lorsque aucune tâche n'est accomplie. Les parties qui ne participent pas à l'exécution d'une tâche ne consomment aucune énergie dynamique. De plus, ces circuits peuvent se réveiller instantanément pour passer d'un mode de fonctionnement endormi (pas de consommation d'énergie), à la pleine activité (débit maximal). Cette fonctionnalité n'est pas rajoutée au système comme dans certains circuits synchrones mais inhérente au fonctionnement asynchrone.
Le deuxième facteur permettant de réduire l'énergie consommée par le circuit est l'absence d'horloge globale. Comme nous l'avons dit précédemment, les temps de propagation des signaux d'horloges sont aujourd'hui plus grands que les temps de traitement des différentes fonctions. Ceci est dû aux capacités présentées par les rails d'horloge. Ces capacités ont une autre conséquence, leur agitation à coût énergétique important $P = C\ V^2\ f$, avec C la capacité de l'ensemble des rails d'horloge, V la tension d'alimentation et f la fréquence du signal d'horloge. Cette consommation ne fait qu'augmenter à mesure que la fréquence augmente, et que les capacités de rails augmentent (ceci étant du entre autres à la réduction des distances inter-rails, et donc à l'augmentation de leur capacité). Le coût énergétique lié au fonctionnement des horloges est aujourd'hui la source la plus importante de consommation énergétique dans certains circuits. Un circuit asynchrone n'ayant pas recours à une horloge globale, ce coût disparaît, et permet de réduire significativement l'énergie consommée.

– **Modularité** : La modularité des circuits asynchrones est un autre des avantages ayant contribué au retour des circuits synchrones dans le monde de l'architecture de circuits. Grâce au contrôle local, les parties d'un circuit asynchrone forment des briques autonomes et élémentaires qui peuvent être assemblées en se préoccupant principalement d'aspects fonctionnels. Ces briques peuvent être facilement réutilisées d'un circuit vers un autre et archivées dans des bibliothèques de conception automatique ou assistée par ordinateur. Cette propriété permet de rendre les circuits asynchrones très portables, en particulier dans le cas d'une migration technologique. Par rapport aux circuits synchrones, il n'est pas nécessaire d'effectuer une synthèse et une vérification globale du circuit pour vérifier que les décalages entre les instants d'arrivée des horloges (clock skew) ne sont pas rédhibitoires.
Pour la même raison, il est possible d'ajuster le circuit de manière beaucoup plus fine. L'ajout d'une brique asynchrone est en effet sans répercution sur le fonctionnement global du circuit, cela permet donc d'affiner de manière locale chacune des parties d'un circuit asynchrone jusqu'à obtenir le meilleur fonctionnement possible. Une telle démarche est très difficile en synchrone dans la mesure où l'ajout d'un composant peut avoir des conséquences sur tout le reste de la conception. Nous pouvons citer l'exemple de l'ajout d'un étage dans un pipeline synchrone. Pour faire cela, il faut s'assurer que chacun des

chemins convergents continue à avoir le même nombre d'étages de pipeline, et que le circuit fonctionne toujours correctement. En conception asynchrone, l'ajout d'un étage de pipeline se fait sans aucune difficulté.

- **Emission électromagnétique** : Un des avantages des circuits asynchrones est la réduction des émissions électromagnétiques du circuit [PSR04]. Cette réduction s'explique par deux points principaux. Contrairement aux circuits synchrones où toutes les parties fonctionnent à fréquence fixe et à peu près en phase, les transitions ont lieu à des moments beaucoup plus répartis dans un circuit asynchrone. Ceci a pour conséquence d'élargir le spectre d'émission du composant tout en affaiblissant la densité de puissance d'émission électromagnétique pour chacun des bandes de fréquence. De plus, dans un système synchrone, les signaux de fréquence fixes sont transportés sur des rails de taille assez importante qui peuvent donc jouer un rôle d'antenne et favoriser les émissions électromagnétiques. La suppression de ces rails dans les circuits asynchrones contribue donc également à réduire ces émissions.
  Une des conséquences de cette diminution des perturbations électromagnétiques est une réduction des contraintes de blindage du circuit lors de sa conception.
  Cette propriété de faible émission électromagnétique est importante dans les applications embarquées de type télécommunications où les problèmes de compatibilité électromagnétique sont fondamentaux, et dans les applications militaires où le souci de camouflage est constant.

- **Sécurité** : Avec le développement des communications et traitements sécurisés et cryptés, une nouvelle propriété des circuits asynchrones a été mise en lumière. Il s'agit de la sécurité contre les attaques de type DPA (Differential Power Analysis) [BRR+04]. Lors d'une attaque DPA, les variations de courant dans un circuit sont observées et permettent de dévoiler les données qui y sont manipulées. Dans un circuit synchrone, ces variations sont parfaitement localisées, et correspondent aux fronts d'horloge, leur étude permet de dévoiler les données échangées dans le circuit. La dispersion temporelle des traitements de données dans un circuit asynchrone rend plus difficile une telle analyse, ce qui permet d'améliorer la sécurité de ce type de circuit.

Les avantages des circuits asynchrones présentés ici ont favorisé leur retour dans les laboratoires de recherche. Cependant, aujourd'hui, l'utilisation industrielle de composants asynchrones reste encore assez limitée. Il y a deux inconvénients majeurs des circuits asynchrones permettant d'expliquer cela :

- **Augmentation de la surface d'implantation** : Le fait de ne pas avoir d'horloge globale dans le système allège les contraintes de placement routage à l'échelle globale du circuit. Toutefois, cette synchronisation est réalisée localement et implique donc l'ajout de composant matériel à cet effet. Des exemples d'implantations ont montré que l'augmentation de la surface de si-

licium nécessaire à une implantation asynchrone est d'environ un facteur 2 [vGBvB+98].

– **Outils de conception** : En raison de la quasi-totale domination du marché par la logique synchrone, les outils de conception assistée et de vérification ont été développés pour cette technologie. Il n'existe aujourd'hui que peu d'outils permettant de réaliser la même chose en logique asynchrone. De fait la conception d'un circuit asynchrone se révèle être plus longue, plus difficile et plus hasardeuse que la conception d'un circuit synchrone. Ce facteur est une cause importante de l'utilisation très limitée de l'asynchronisme dans les circuits actuels. La solution à ce problème est de développer des outils permettant la conception et le test des circuits asynchrones, ce que font plusieurs équipes universitaires. Nous pouvons citer l'université de Manchester et son outil de conception Balsa [EB02] basé sur Tangram, un autre outil ayant été développé à l'université de Eindhoven en collaboration avec Philips [vBKR+91]. En France, l'équipe CIS du TIMA à Grenoble sous la direction de Marc Renaudin développe actuellement un outil complet appelé TAST [SRSR04].

Les listes d'avantages et inconvénients des circuits asynchrones présentées dans cette partie l'ont été dans un cadre général. Dans le cadre plus restreint des rétines artificielles régionales asynchrones, ces listes sont en réalité plus réduites. Parmi les avantages, figurent le gain en vitesse et en énergie consommée, et la réduction des émissions électromagnétiques. En revanche, la modularité ne nous concerne pas dans la mesure où l'architecture que nous présentons dans la suite n'est pas de type modulaire comme pourrait l'être un circuit de type GALS (Globally Asynchronous, Locally Synchronous). De la même manière, l'architecture étant une maille constituée d'un ensemble de processeurs asynchrones très simples et dont l'architecture a été optimisée manuellement, les problèmes liés aux outils de conception ne se posent pas, les aspects vérification ne posent pas problème non plus compte tenu de la simplicité du circuit. Enfin, l'augmentation de la surface d'implantation due à l'utilisation de circuits asynchrones ne peut être considérée comme un inconvénient dans le cas d'une rétine asynchrone, car l'asynchronisme n'est pas utilisé en remplacement d'une fonction synchrone mais dans le but d'implanter de nouvelles fonctionnalités régionales (et à faible coût matériel).

5.3 Circuits asynchrones et synchrones dans les rétines artificielles : quelle répartition ?

Dans les sections précédentes, nous avons montré la nécessité des connexions programmables, et les limitations du fonctionnement synchrone de type SIMD pour le calcul des primitives régionales de traitement d'image. Les limitations principales de ce type de fonctionnement sont la vitesse limitée à la fréquence d'horloge pour

les propagations sur les graphes, ce qui peut conduire à des temps de propagations longs, et l'énergie dépensée inefficacement pour l'envoi d'instructions lorsque le taux d'occupation des processeurs est faible. Ces limitations nous ont conduit à envisager l'usage de l'asynchronisme dans les rétines artificielles afin de remédier à ces inconvénients. L'asynchronisme est une solution intéressante car seules les cellules actives consomment de l'énergie, il n'y a pas besoin d'envoyer d'instruction permettant de guider les données durant les phases de propagation, celles-ci peuvent s'effectuer à la vitesse maximum possible dans la structure de propagation choisie, et le temps de propagation est proportionnel au temps moyen de traversée d'une cellule et non pas au temps de traversée le plus long comme dans le cas synchrone. Toutefois l'asynchronisme présente un défaut majeur pour une implantation dans les rétines artificielles : son coût matériel est élevé. Or dans les rétines artificielles, une des contraintes majeures est la taille des processeurs élémentaires. Afin d'avoir un circuit ayant une bonne densité de capteurs, ces processeurs doivent être les plus petits possible.

La conclusion de cette analyse est qu'il est nécessaire d'utiliser l'asynchronisme dans les rétines artificielles pour répondre aux contraintes de consommation d'énergie et de vitesse de calcul des opérateurs régionaux intégraux, mais qu'il est préférable de réduire l'asynchronisme au minimum nécessaire afin de ne pas trop augmenter le coût matériel d'implantation des processeurs élémentaires.
La question qui se pose donc est celle de la répartition entre synchronisme et asynchronisme dans les rétines artificielles pour effectuer des opérations régionales intégrales.

Des éléments de réponse à cette question ont déjà été apportés lors de l'étude des opérateurs adaptés aux différents types de traitement d'image. Premièrement, étant donné la diversité de la taille des opérandes à manipuler et la nécessité de minimiser le coût d'implantation des opérateurs dans chaque pixel, il est nécessaire d'utiliser des opérateurs régionaux de type bit-série. Ce choix apporte en même temps une réponse à la question de la répartition synchrone-asynchrone dans les rétines artificielles puisque les opérations bit-série se composent de phases de propagation d'un bit de donnée le long d'une chaîne au travers d'opérateurs locaux, et que ces phases de propagations sont séquentialisées entre elles. Ainsi la répartition entre synchronisme et asynchronisme dans les rétines artificielles s'effectue assez naturellement. **Les traitements régionaux seront implantés sous forme de séquences synchrones de propagations asynchrones au travers d'opérateurs locaux (ou calculs asynchrones) sur** 1 **bit**. Les calculs asynchrones sur 1 bit étant des opérations assez simples, il est raisonnable d'envisager de pouvoir les implanter de manière matérielle dans les rétines artificielles. Les opérations plus complexes intervenant entre les phases de propagations pourront être effectuées de manière synchrone. Il y a généralement besoin d'effectuer ces opérations en parallèle sur de nombreux pixels, ce qui conduit à un taux d'occupation des processeurs élevé pour lequel l'utilisation d'opérations synchrone pilotées de l'extérieur est pertinente. De plus, ces opérations intervenant assez rarement (si on compare

la fréquence à laquelle elles sont effectuées à la fréquence à laquelle s'effectuent les calculs lors d'une propagation synchrone sur un graphe), leur coût énergétique sera d'autant plus réduit.

L'étude de ce type de traitements régionaux est menée dans la deuxième partie de cette thèse. Les implantations existantes sont rappelées puis une nouvelle implantation du modèle des réseaux associatifs est proposée, que nous appelons les *micropipelines associatifs*.

6

Opérateurs régionaux asynchrones : l'exemple de la somme régionale

6.1 Tour d'horizon des implantations asynchrones de la somme régionale dans les mailles

Dans ce chapitre, nous étudions deux exemples d'implantations de la somme régionale de manière asynchrone. Ces deux méthodes proposées par Komuro et al. à l'Université de Tokyo [KKI04] et par A. Mérigot à l'Université d'Orsay [DMM95] sont basées sur le principe de l'addition bit-série. Dans un premier temps, nous rappelons le principe de l'addition bit-série avant de détailler les architectures existantes.

6.1.1 Principe de l'addition bit-série

Une addition bit-série permet d'effectuer une addition de plusieurs nombres binaires en calculant les bits de la somme un par un en commençant par le bit de poids faible. Pour cela, on place chacune des données à additionner dans une unité de calcul élémentaire capable d'effectuer une somme sur des données ayant une profondeur de 1 bit. Ces unités sont connectées entre elles en série. Un exemple de calcul d'une addition bit-série est présenté à la figure 6.1. Les 5 grandeurs à additionner dans cette exemple sont les nombres binaires (valeurs stockées dans les bits de données représentés en rouge et encadrés en gris sur la figure) : 001, 110, 011, 110, 001. La figure se lit de haut en bas pour le calcul des différents bits de la somme, de gauche à droite pour le calcul d'un bit de la somme.

La première opération bit-série est le calcul du bit de poids faible de la somme. Ce calcul utilise les bits de poids faible de chacun des opérandes, soit dans l'ordre les valeurs 1, 0, 1, 0 et 1. Les retenues sont initialisées à zéro. Le calcul s'effectue comme suit :

– On part de la cellule située la plus à gauche. On somme la valeur du bit de

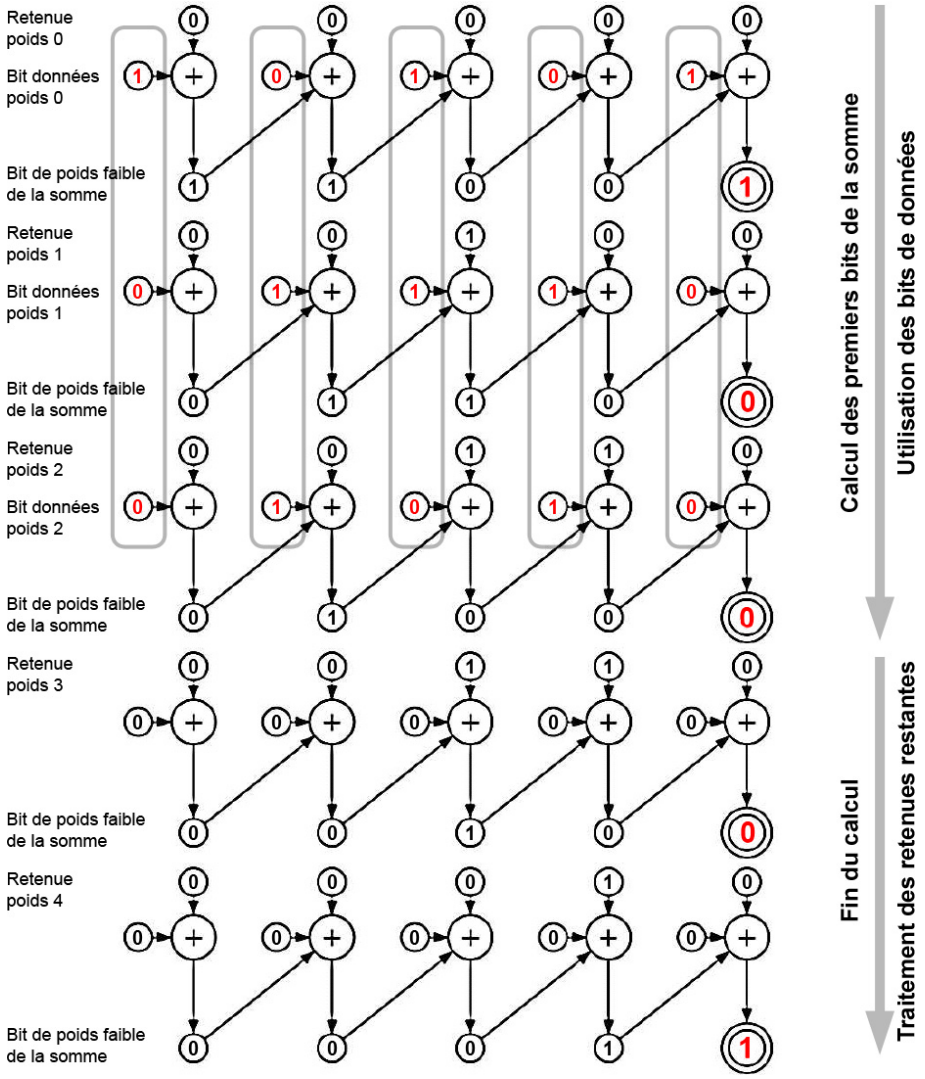


FIG. 6.1 – Principe de l'addition bit-série

donnée (1), celle de la retenue (0) à l'aide d'un additionneur à 3 entrées (la troisième entrée étant connectée à 0, mais non représentée). Le résultat de la somme est 1, et la retenue 0.

– On se place dans la cellule suivante dans la chaîne. On somme la valeur du bit de donnée (0), celle de la retenue (0) et la valeur du bit de poids faible de la somme immédiatement précédente (1) à l'aide d'un additionneur à 3 entrées. Le résultat de la somme est 1, et la retenue 0.

– On effectue les même opérations jusqu'au bout de la chaîne. La valeur du bit de poids faible obtenue à la dernière cellule (entourée de deux cercles) est la valeur du bit de poids faible de la somme (ici 1).

Les opérations bit-série suivantes permettent de calculer les autres bits de la somme. Pour le calcul du bit de poids i de la somme, les retenues générées lors du calcul du bit de poids $i-1$ de la somme sont utilisées.

On continue le calcul jusqu'à ce que l'on ait utilisé tous les bits des données à additionner et que toutes les retenues soient égales à 0, ce qui assure qu'il n'y a plus d'addition à effectuer. Le résultat de la somme est obtenu dans la dernière cellule de la chaîne. A l'issu de la i^{me} opération bit série, la valeur obtenue est le bit de poids i de la somme globale (valeur entourée par deux cercles). Finalement, ici la valeur obtenue pour la somme est 10001 (le bit de poids faible étant en haut sur la figure).
Il est à noter que le nombre total d'opérations bit-série effectuées pour calculer une somme est égal au nombre de bits de la somme.

L'exemple de calcul bit-série présenté est linéaire, ce qui signifie que chaque unité de calcul élémentaire a un et un seul antécédent. Dans ce cas, la somme unaire réalisée dans chaque unité de calcul utilise 3 opérandes. On peut également envisager le cas d'un calcul bit-série à entrées multiples. Dans ce cas, il est nécessaire à chaque étape de sommer plus de 3 opérandes. Par exemple, dans le cas d'un additionneur bit-série où chaque unité de calcul peut avoir 2 entrées, il est nécessaire de sommer 4 opérandes : le bit de retenue, le bit de donnée, et les deux bits provenant des entrées.

6.1.2 Additionneur bit-série linéaire

La première architecture que nous étudions permet d'implanter le calcul de l'addition bit-série linéaire telle qu'elle a été présentée précédemment. Cette structure a été proposée par Komuro, Kagami et Ishikawa à l'Université de Tokyo [KKI04] dans une rétine artificielle.
Nous détaillerons la partie asynchrone de l'architecture utilisée, puis nous décrirons les algorithmes utiles à son fonctionnement. Enfin, nous discuterons des avantages et inconvénients de cette architecture.

Architecture

La structure de la rétine étudiée est une matrice de processeurs à deux dimensions. Chaque processeur élémentaire est constitué d'un photorécepteur couplé à un processeur élémentaire (PE). L'architecture du PE utilisé est proposée à la figure 6.2. Chaque PE est connecté aux voisins haut, bas, gauche et droit et peut communiquer avec eux de manière synchrone.

Chaque processeur élémentaire dispose en outre d'un chemin asynchrone destiné au calcul des opérations régionales. Le long de ce chemin, en partant des pixels immédiatement précédents (à gauche sur la figure 6.2), nous rencontrons un multiplexeur placé en entrée et permettant d'établir une connexion entrante depuis un

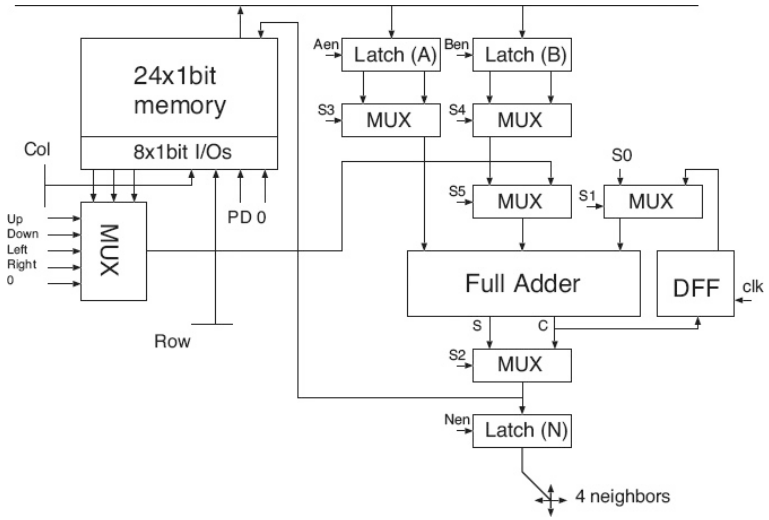


FIG. 6.2 – Structure du processeur élémentaire *(source [KKI04])*

pixel voisin ou un bit de valeur 0. Ce multiplexeur permet de choisir la connexion active, il joue donc un rôle de connexion programmable. Le second élément essentiel du chemin asynchrone est le *Full Adder*, qui permet d'additionner de manière combinatoire 3 données unaires. Cet opérateur est l'opérateur élémentaire permettant d'effectuer l'addition bit-série linéaire. Ses trois entrées sont connectées :

- au processeur antécédent (par le biais du multiplexeur piloté par $S5$)
- à un bit de donnée (par le biais du multiplexeur piloté par $S3$)
- à un bit de retenue (par le biais du multiplexeur piloté par $S1$, la retenue étant stockée dans le registre DFF)

Le verrou *latch(N)* en sortie permet d'obtenir un fonctionnement avec propagation vers le voisin connecté en sortie pour les opérations régionales (verrou ouvert) ou sans propagation (verrou fermé) pour les opérations locales.
Pour les opérations locales, certains calculs sont effectués par le full adder. Dans ce cas, la connexion à un pixel antécédent (utilisée dans les opérations régionales) est redirigée vers un second bit de donnée interne à l'aide des multiplexeurs pilotés par $S4$ et $S5$.
Le processeur élémentaire décrit permet donc d'effectuer des opérations locales de type mixte synchrone-asynchrone ou régionales asynchrones.

Fonctionnement

Nous limitons l'étude du fonctionnement de la rétine proposée au cas des opérations régionales asynchrones.
Grâce à la connexion programmable présente dans chaque pixel, il est possible de connecter de nombreux processeurs selon une topologie de type chaîne. Le premier

pixel de la chaîne est connecté en entrée à 0, puis les autres pixels sont reliés les uns aux autres jusqu'à la fin de la chaîne. Cette chaîne de processeurs élémentaires sert de support à l'addition régionale telle qu'elle a été décrite précédemment. Il est intéressant ici de rentrer un peu plus dans le détail du chargement des données à additionner :

- la donnée provenant du processeur antécédent dans la chaîne est routée jusqu'au full adder à l'aide de la connexion programmable et du multiplexeur piloté par $S5$.

- la donnée interne en mémoire est mémorisée par le $latch(B)$ et routée vers le full adder grâce au multiplexeur piloté par $S4$.

- la retenue mémorisée dans le registre DFF est routée vers le full adder par le multiplexeur piloté par $S1$.

Une fois les données placées en mémoire et routées vers le full adder, la phase proprement dite d'addition asynchrone peut commencer. La propagation est déclenchée par l'ouverture du $latch(N)$, le multiplexeur piloté par $S2$ ayant au préalable été configuré de telle sorte que le bit de poids faible de l'addition réalisée par le full adder soit transmis. Le calcul d'un bit de la somme régionale est effectuée dès lors que le réseau est stabilisé.

La mise en série de n processeurs permet de réaliser des additions bit-série sur n bits. Cette structure permet d'extraire des grandeurs scalaires sur des régions pouvant être représentées sous forme d'une chaîne continue de processeurs. Toutefois, cette fonction n'est pas la seule fonction régionale qu'il est possible d'effectuer avec cette structure. En effet, si l'on relie le bit de poids fort de la sortie du full adder (la retenue C) au verrou de sortie (*latch N)* à l'aide du multiplexeur piloté par $S2$, et si l'on place un 1 logique en $S0$ et que l'on route ce 1 vers le full adder au lieu de router la retenue, alors, on peut réaliser un OU-global sur la chaîne de processeurs. Le bit de poids fort de la sortie d'un Full Adder se comporte en effet comme un opérateur majorité à 3 entrées, qui devient un opérateur OU à 2 entrées si l'on connecte l'une des 3 entrées à un 1 logique.

Les opérations régionales fondamentales réalisables à l'aide de l'architecture présentée sont donc la somme régionale et le OU logique régional.

Evaluation des performances

L'architecture de l'additionneur bit-série proposée par Komuro, Kagami et Ishikawa à l'Université de Tokyo [KKI04] offrent certaines fonctionnalités n'existant pas dans les rétines SIMD classiques, en particulier :
- elle permet de calculer des sommes à haute vitesse sur une région.
- elle permet d'effectuer un OU logique régional.

Ses performances sont ici évaluées sur une région de taille $200 * 200$ pixels, en supposant qu'elle peut être couverte par un réseau linéaire.
Dans une chaîne linéaire de processeurs asynchrones, le nombre de propagations nécessaires au calcul de chacun des bits de la somme est égal à n également. En supposant le délai asynchrone entre deux cellules égal à t_{async} en moyenne, le temps de calcul total est égal à nt_{async}.

En technologie $0.35\mu m$, $t_{async} \simeq 3ns$ (simulations effectuées à l'aide de Spice). Dans une région constituée de 40000 pixels, cela conduit donc à un temps de calcul pour un bit de la somme d'environ $120\mu s$. Si l'on envisage le calcul d'une somme dont le résultat fait 28 bits (ce qui correspond au cas d'une région comportant 40000 pixels, les données à sommer ayant une profondeur de 12 bits) on obtient un temps de calcul d'environ $3.4ms$ pour les opérations bit-série asynchrones uniquement. Ce temps est multiplié par 3 environ si l'on tient compte des opérations synchrones à effectuer pour la mémorisation des retenues, le chargement des données...
On arrive finalement à un temps de calcul de l'ordre de $10ms$ pour effectuer une somme régionale. Le nombre d'opérations régionales qu'il est possible d'effectuer est donc de l'ordre de 100 opérations par seconde, soit environ 3 opérations par image dans le cas d'un imageur à 30 images secondes.

Dans cette évaluation, nous n'avons pas pris en compte le temps nécessaire à l'établissement de la structure du réseau couvrant, celui-ci devant être fait depuis l'extérieur de la rétine.

Inconvénients

La structure proposée par Komuro, Kagami et Ishikawa, qui a l'avantage d'offrir aux rétines artificielles un nouveau champ d'application présente toutefois un certain nombres d'inconvénients :

– Le circuit proposé ne peut être utilisé pour calculer des sommes distribuées sur une région ayant une forme quelconque 6.4. Pour couvrir tous les réseaux,

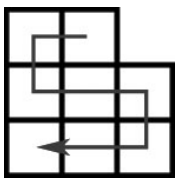


FIG. 6.3 – Région couverte avec un réseau linéaire

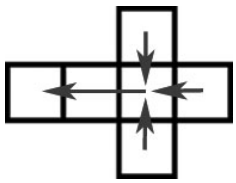


FIG. 6.4 – Région impossible à couvrir avec un réseau linéaire

une structure de type arbre couvrant est préférable. Il est toutefois possible de tenter de résoudre ce problème en utilisant uniquement un additionneur bit-série linéaire, mais cela implique de ne pas calculer en même temps des sommes sur toutes les régions de l'image dans la mesure où il est nécessaire

de traverser des pixels n'appartenant pas à la région sur laquelle s'effectue le calcul. La gestion de l'ordonnancement des calculs régionaux sur ces différentes régions se révèle dans ce cas être un problème complexe, et impossible à résoudre de manière autonome sur un imageur massivement parallèle, ce qui réduit fortement l'intérêt de cette méthode.

– La méthode proposée conduit à obtenir des chaînes de processeurs élémentaires très longues. L'utilisation d'un réseau de connexion de type chaîne conduit à un temps de propagation proportionnel au nombre n de pixels formant la région. L'utilisation d'une structure de type arbre couvrant (voir 6.1.3) permet de réduire la longueur de la chaîne significativement d'un facteur environ $\sqrt{n}$, ce qui conduit pour une région de taille 10000 pixels à une réduction du temps de propagation de deux ordres de grandeur, ce qui est extrêmement important. Cette réduction permettrait d'utiliser de manière intensive les opérateurs régionaux, une limitation à environ 3 opérations régionales par seconde étant trop limitative.

– La robustesse est limitée. Dans le cas où le chemin asynchrone de l'un des processeurs viendrait à tomber en panne, tous les pixels situés dans la chaîne avant ce pixel ne seront pas pris en compte, ce qui réduit fortement la robustesse du système. Une solution dans laquelle le pixel incriminé serait le seul à être exclu de la somme serait bien meilleure.

Ces différents inconvénients nous conduisent donc à nous tourner vers une autre implantation de la somme régionale, l'additionneur bit-série à entrées multiples.

6.1.3 Additionneur bit-série à entrées multiples

Cette structure et le mode de calcul qui lui est associé a été proposée par Alain Mérigot, Didier Dulac et Siamak Mohammadi [Dul96] [Moh96] [DMM95] à l'université d'Orsay dans les années 90 sous le nom de *Maille Associative d'Orsay*. Bien qu'antérieur de plus de dix ans par rapport à la précédente, elle présente des fonctionnalités plus avancées que cette dernière, en particulier l'addition bit-série à entrées multiples que nous avons évoqué précédemment. Ces fonctionnalités ont toutefois une contrepartie en terme de coût d'implantation électronique de chaque processeur élémentaire. Dans cette partie, nous détaillons cette architecture ainsi que l'algorithmique qui lui est associée.

Aspects algorithmiques

L'algorithmique associée à la *maille associative d'Orsay* est présentée et formalisée en détails dans la thèse de Bertrand Ducourthial [Duc00].
La Maille Associative d'Orsay constitue une implantation du *modèle limité des réseaux associatifs*, permettant de définir des régions, ensembles de processeurs de taille et de forme arbitraire afin de leur appliquer des primitives régionales.

Les *associations* implantées dans la *Maille Associative d'Orsay* sont celles définies dans la version limitée du modèle théorique des réseaux associatifs :

- *direct-association* lorsque l'association repose sur des opérateurs *idempotents*. Elle est exécutée sur un graphe symétrique (les connexions entre paires de processeurs sont bidirectionnelles). Les primitives régionales correspondant à ce compromis sont typiquement les fonctions logiques (OR, AND), le maximum et le minimum. L'algèbre booléenne nous permet d'exprimer ces différentes fonctions à l'aide d'une primitive électronique unique implantée dans chaque processeur élémentaire, **l'opérateur logique** *OU*.

- *prefix-association* lorsque l'association est exécutée sur un arbre couvrant (graphe dans lequel tout processeur élémentaire à un et un seul antécédent connecté par une connexion unidirectionnelle, à l'exception du point racine qui n'en a pas). La *prefix-association* permet les *associations* avec des *opérateurs non idempotents*, le résultat devant être redistribué à chaque pixel de la région ensuite (grace à un opérateur max par exemple). La primitive de ce type la plus importante pour le traitement d'image est la somme régionale. Elle nécessite d'avoir recours à un **additionneur asynchrone** dans chaque processeur élémentaire.

Architecture

Après avoir présenté les types d'associations utilisables dans la *Maille Associative d'Orsay* et les opérateurs qui lui sont associés, nous allons maintenant détailler son architecture et en particulier l'architecture du processeur élémentaire. La présentation est effectuée en référence à l'additionneur bit-série linéaire détaillé avant, de manière à faciliter la compréhension.
L'architecture du processeur élémentaire de la *Maille Associative d'Orsay* est celle d'un processeur élémentaire d'une machine massivement parallèle asynchrone ayant une largeur de donnée de 1 bit. Elle peut être connectée à ses 8 voisins pour former un réseau de connexion 8-connexe. A l'instar de l'additionneur bit-série linéaire présenté précédemment, le processeur élémentaire de la *maille associative d'Orsay* n'est pas totalement asynchrone. Son architecture se divise en deux parties :

- Couche asynchrone mettant en oeuvre les communications et les calculs de primitives asynchrones régionales distribuées. Un additionneur unaire bit-série à 8 entrées fait notamment partie de cette couche asynchrone.

- Couche synchrone permettant les mouvements de données locaux et les opérations arithmétiques locales standards.

La couche asynchrone

Elle permet d'exécuter les calculs régionaux sous forme de propagation bit-série asynchrone.

Considérons tout d'abord la fonction additionneur bit-série asynchrone correspondant à une *prefix-association*. Son mode de fonctionnement est similaire à celui de l'additionneur bit-série linéaire, à une différence majeure près, le nombre d'entrées permettant de connecter la cellule aux voisins. Celui-ci est égal à 8 entrées (les 8 voisins) dans le cas de la *maille associative d'Orsay*. De plus, la retenue interne et le bit de donnée utilisés dans le calcul des opérations bit-série linéaires présentées précédemment sont ici fusionnés (ce qui nécessite d'effectuer des opérations synchrones) et considérés comme une entrée supplémentaire de l'additionneur asynchrone (combinatoire). De fait, l'additionneur asynchrone placé à l'intérieur du chemin asynchrone doit comporter 9 entrées, 8 destinées aux données provenant des voisins, et la dernière destinée à une donnée locale. Le résultat de l'addition étant à valeurs dans $[0, 9]$, il s'exprime sur 4 bits. Le bit de poids faible est transmis de manière bit-série vers les additionneurs situés en aval dans la chaîne, les 3 autres bits de retenue sont stockés dans la mémoire. Le multiplexeur permettant de choisir l'unique entrée active dans le cas de l'additionneur bit-série linéaire est ici remplacé par un masque permettant de sélectionner les entrées actives parmi les 9 possibles. Ce masque permet d'établir des connexions fonctionnelles entre les pixels, et il permet donc de gérer localement la topologie reconfigurable dynamiquement de la *maille associative d'Orsay*.

Considérons à présent la fonction *OU* logique sur une région utilisé dans les *associations* non *prefix*. Dans le cas de l'additionneur bit-série linéaire, l'élément local permettant d'effectuer un OU à deux entrées est le bit de poids fort de la sortie du Full Adder. Il n'en est pas de même dans le cas de la *maille associative d'Orsay*, pour laquelle il est nécessaire d'utiliser un élément de circuit dédié au calcul d'un OU logique à 9 entrées placé en parallèle de l'additionneur à 9 entrées.
Les opérations nécessaires à la mémorisation des résultats, des retenues ou les opérations de calcul locales sont supportées par la couche synchrone qui sera décrite au paragraphe suivant.

La couche synchrone

L'étude approfondie de la couche synchrone est présentée dans la thèse de Didier Dulac [Dul96], nous ne donnons ici qu'un aperçu des fonctionnalités clés de cette couche. Cette couche synchrone se compose globalement d'une unité arithmétique et logique (ALU), de registres, et d'une mémoire. Outre les fonctions de calcul local synchrone que l'on peut trouver dans les différentes architecture SIMD, elle est également utilisée pour effectuer des calculs intermédiaires entre les phases de propagation d'une addition de type bit-série. Une de ces fonctions est à présent détaillée dans la mesure où elle a des répercutions sur l'implantation des registres de la couche synchrone.

Comme présenté précédemment, l'additionneur asynchrone présent dans chaque pixel a un nombre d'entrées égal à 9, soit une entrée de plus que le nombre de voisins. Si l'on regarde l'additionneur présent dans le processeur élémentaire de

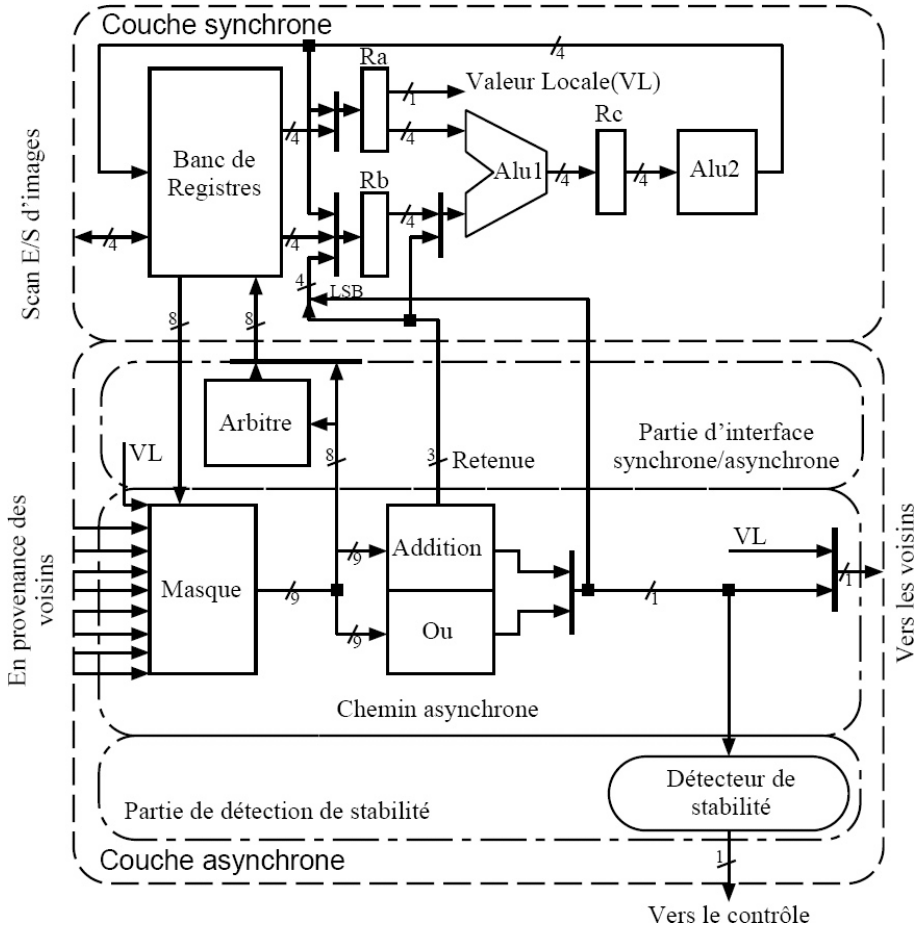


FIG. 6.5 – Structure du processeur élémentaire *(source [Moh96])*

l'additionneur bit-série linéaire, force est de constater que son nombre d'entrée est 3, soit deux entrées de plus que son unique prédécesseur dans la chaîne. Cette différence provient du fait que dans le cas de l'additionneur bit-série linéaire, le calcul des différents bits de la somme peut s'effectuer séquentiellement avec juste un stockage de la retenue et le chargement du bit local à additionner entre deux itérations. Pour procéder de même dans le cas de la *maille associative d'Orsay*, il serait nécessaire à chaque calcul d'un bit de la somme, de stocker les 3 bits de retenue générés. Pour calculer le bit suivant de la somme, il faudrait prendre en compte le bit de poids 1 de la retenue (le bit de poids faible étant le bit de poids 0) généré lors du calcul d'addition bit-série précédent, le bit de poids 2 de la retenue généré lors du calcul d'addition bit-série effectué deux cycles plus tôt, et le bit de poids 3 de la retenue généré trois itérations plus tôt. A ces bits s'ajoute également le bit de donnée locale. Finalement, un total de $8 + 3 + 1 = 12$ entrées serait nécessaire dans l'additionneur, ce qui augmente de manière importante le coût d'implantation physique de l'additionneur.
Au lieu de cela, la solution choisie est de calculer avant chaque itération du calcul d'addition bit-série, le bit de poids faible de la somme des retenues préalablement stockées. Ce calcul est effectué en parallèle de manière synchrone dans chacun des

processeurs, et son temps d'exécution ne dépend donc pas du nombre de processeurs de la région considérée. En conséquence, il est peu pénalisant d'effectuer cette opération de manière synchrone, alors que le gain matériel est important dans la partie asynchrone, et que cela n'apporte aucun surcoût matériel au niveau de la partie synchrone. Nous ne détaillerons pas ici le rôle de chacun des registres et des différents composants de la couche synchrone du processeur, cette étude étant disponible dans la thèse de Didier Dulac [Dul96]. Il est toutefois important de préciser que les registres principaux de la couche synchrone ont une largeur de 4 bits afin de permettre un calcul simple et efficace de l'addition des 3 retenues et du bit de donnée locale.

La structure générale du processeur élémentaire de la *maille associative d'Orsay* est décrite à la figure 6.5. Outre les composants situés dans les couches synchrone et asynchrone que nous avons décrit, il est important de noter la présence d'un arbitre à 8 entrées à l'interface entre la couche synchrone et la couche asynchrone. Le rôle de cet arbitre dans l'établissement des arbres couvrants sera détaillé dans les chapitres suivants.

Implantation de l'asynchronisme dans la maille associative

Dans ce paragraphe, nous nous focalisons sur l'implantation de l'opérateur somme distribuée asynchrone dans la *Maille Associative d'Orsay*. Nous ne quantifions pas la partie synchrone qui n'est pas au centre de notre étude.
La conception de cet opérateur de somme distribuée, l'additionneur à 9 entrées, est orientée par un paramètre principal, la transmission optimale en terme de vitesse du bit de poids faible. Une structure de type arbre de Wallace [Wal64] est envisageable, elle comporte 5 Full Adder (FA) et 2 Half Adder (HA). Le résultat global est calculé en un temps égal à la traversée de 4 additionneurs. Toutefois, afin de

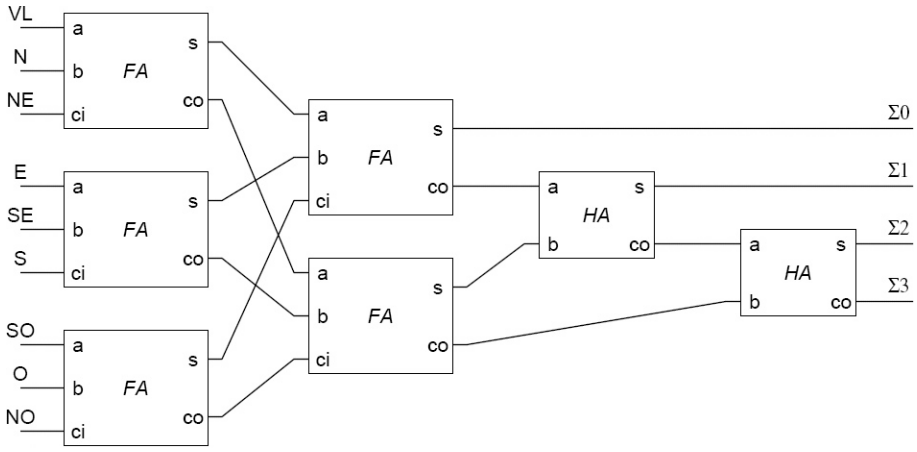


FIG. 6.6 – Structure de l'additionneur 9 entrées *(source [Moh96])*

diminuer la charge capacitive du chemin asynchrone de l'opérateur de somme et ainsi diminuer le temps de propagation du bit de poids faible de la somme, il est préférable de se tourner vers une structure où le calcul de ce bit et le calcul des 3

autres bits est découplé. Cela conduit à utiliser un XOR à 9 entrées pour le calcul du premier bit de la somme. L'optimisation de cette structure a été réalisée par Mohammadi [Moh96], et le circuit optimisé permettant le calcul du bit de poids faible de la somme est présenté à la figure 6.7. Le calcul des autres bits de la somme est implanté en se référant aux arbres de Wallace. Le temps moyen de traversée de

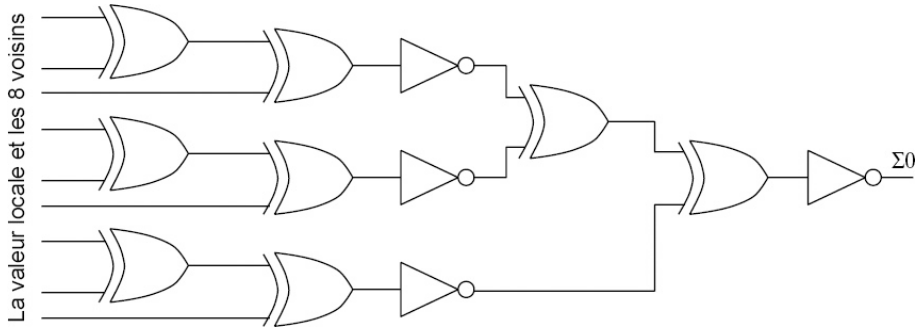


FIG. 6.7 – Structure de calcul du bit de poids faible de l'addition à 9 entrées : XOR à 9 entrées *(source [Moh96])*

la structure de calcul du premier bit de la somme est environ 2,2 ns sur l'implantation effectuée à l'Université d'Orsay.
Parmi les autres composants présents dans la couche asynchrone, le OU logique est implanté de manière particulière dans la mesure où il correspond au OU utilisé dans une *association* et que c'est donc un opérateur idempotent. La structure choisie est donc un OU à précharge, qui fonctionne en deux phases, la première étant le préchargement de la sortie à 0, la deuxième étant l'évaluation de l'opérateur, sachant que la sortie peut passer à 1 mais ne peut pas revenir à 0.
Le troisième composant asynchrone utilisé dans la maille, l'arbitre, sera étudié en détails dans les prochains chapitres.

Les limitations de la maille associative

Les limitations principales de la maille associative sont :

- La taille du processeur élémentaire : la densité d'une implantation destinée à des applications de vision de moyen-haut niveau est limitée par cette taille importante, d'autant plus que la structure de la *Maille Associative d'Orsay* n'intègre pas de dispositif destiné à la capture d'image, qui augmenterait d'autant la taille du processeur élémentaire.

- Les fonctionnalités dédiées et complexes des opérateurs asynchrones : les opérateurs présents dans le chemin asynchrone sont au nombre de deux et sont dédiés à deux tâches très spécifiques (le OU logique et l'addition). Ces tâches sont fondamentales pour le traitement régional des données, mais leur coût d'implantation est très important (en particulier pour l'additionneur à 9 entrées). Un opérateur unique et moins dédié serait préférable.

La limitation principale de la maille associative est la taille du processeur élémentaire. La taille importante du processeur de la maille associative, en particulier la taille de la partie dédiée à l'asynchronisme nous incite à analyser l'utilité de chacun des opérateurs asynchrones présents et les possibilités de les réduire. Cette analyse est effectuée dans la partie suivante.

6.2 Vers une nouvelle architecture dédiée aux traitements régionaux

Dans cette partie, nous effectuons une analyse des limitations de la maille associative afin de permettre au lecteur de comprendre les motivations qui nous ont poussé à proposer une nouvelle architecture dédiée aux traitements régionaux, en particulier l'addition sur une région, dans les imageurs.
Nous commençons cette analyse par l'étude de l'adéquation entre le modèle des réseaux associatifs et la *Maille Associative d'Orsay*. Considérons tout d'abord les ressources utilisée par les primitive de *direct-association* et de *prefix-association*. La première, basée sur des opérateurs idempotents utilise le *OU* logique. La seconde basée sur des arbres couvrants utilise le plus souvent l'additionneur. L'usage de ces deux structures est exclusif, une solution potentielle pour réduire le coût matériel de la maille associative serait donc de renoncer à l'une de ces primitives.

La définition de la *direct-association* a conduit à en montrer les limites et à définir la *prefix-association* pour le calcul des sommes sur une région. Il est donc impossible de se passer de la *prefix-association*. La question est donc de savoir si la *prefix-association* n'utilisant que l'additionneur comme ressource matérielle permettrait de remplacer la *direct-association* moyennant une contrepartie algorithmique, par exemple des itérations.

Pour cela, comparons les résultats obtenus avec une *direct-association* et une *prefix-association*. Le résultat d'une *direct-association* dans chaque pixel est celui de l'*association* correspondante. Le résultat d'une *prefix-association* est celui de l'*association* correspondante si l'on considère uniquement le processeur racine de l'arbre couvrant. Pour obtenir une *direct-association* à partir d'une *prefix-association*, il faut donc effectuer une prefix-association, puis redistribuer le résultat de cette association dans tous les processeurs.
La distribution d'une valeur à partir de la racine d'un arbre est possible avec une prefix-association utilisant des propagations sur l'arbre couvrant orientées de la racine vers les feuilles de l'arbre couvrant. L'opérateur associé à cette distribution est l'addition, avec une initialisation à 0 de la valeur à additionner dans chacun des pixels autres que la racine. Cette *prefix-association* est appelée *leaffix-association* [Mer97] par opposition à la *prefix-association* utilisée pour propager les données des feuilles vers la racine et qui est appelée *rootfix-association*.

Ainsi une *direct-association* (utilisant un opérateur idempotent $\otimes$) peut être calculée en deux étapes en effectuant successivement une *rootfix-association* utilisant l'opérateur $\otimes$ associé à la *direct-association*, puis une *leaffix-association* utilisant l'opérateur addition, en ayant pris soin de remettre à 0 les valeurs locales à additionner de tous les pixels de la région sauf la racine de l'arbre. Il est possible de généraliser cette méthode de calcul de la *direct-association* à l'*association*. L'idempotence de l'opérateur $\otimes$ n'est pas nécessaire dans la première étape, ce qui signifie que l'on peut également utiliser des opérateurs non-idempotents dans la *rootfix-association*. Il est donc possible de réaliser des *associations* sur la maille associative en utilisant uniquement des *prefix-associations*. Finalement, le fait de ne plus utiliser de *prefix-association* conduit à remettre en cause l'utilité du *OU* logique placé dans le chemin asynchrone.
Il reste toutefois à vérifier que les *prefix-associations* basées sur l'opérateur *OU* logique sont implantables à l'aide de l'opérateur d'addition. L'opérateur d'addition, comme nous l'avons vu, utilise pour la propagation un XOR à 9 entrées. Il est donc difficilement envisageable de réaliser avec cet opérateur un *OU* logique de manière simple. Toutefois, il est possible de réaliser cet opérateur en réalisant une addition de chacun des bits participant au *OU* global sur un arbre couvrant. Si le résultat de la somme est nul, alors la valeur de chacun des pixels de l'arbre est nulle et le résultat du *OU* global est 0. Dans le cas contraire, le résultat du *OU* global est 1. Il est donc possible d'implanter un substitut de *prefix-association* basée sur l'opérateur *OU* à l'aide d'une *prefix-association* basée sur l'opérateur addition. Il est donc possible de supprimer l'opérateur *OU* du chemin asynchrone de la maille associative.

Cette argumentation valide d'un point de vue théorique fonctionnel doit toutefois être nuancée par des contraintes d'efficacité temporelle. La suppression de l'opérateur *OU* asynchrone conduit comme nous l'avons vu à implanter une *prefix-association* utilisant l'opérateur *OU* à l'aide d'une prefix-association utilisant l'opérateur addition. Du point de vue de la complexité temporelle, l'opérateur *OU* permet d'obtenir le résultat en une propagation, alors qu'en utilisant l'opérateur d'addition, il est nécessaire d'effectuer $log(p)$ propagations, p étant le nombre de processeurs de la région, pour calculer chaque bit de la somme avant de pouvoir conclure si celle-ci est nulle ou non. La suppression de l'opérateur *OU* et l'économie correspondante en terme de coût d'implantation ont donc des répercussions significatives sur la vitesse de calcul de l'architecture.

En conclusion, nous avons donc montré que l'additionneur est un bloc fonctionnel indispensable au fonctionnement de la maille associative telle qu'elle est conçue. De plus, le *OU* à 9 entrées, s'il n'est pas indispensable, permet néanmoins de réduire significativement le temps de calcul des *associations* utilisant l'opérateur *OU*. Les perspectives de réduction du nombre de transistors utilisés dans le chemin asynchrone sont donc assez limitées. Cette analyse est toutefois incomplète. Comme nous l'avons dit dans l'introduction sur les limitations de la *Maille Associative*, les opérateurs sont dédiés à une tâche précise et relativement complexe dans le

cas de l'additionneur. Ils ne sont donc pas facilement utilisables pour une autre tâche, comme nous avons pu le constater dans le cas de l'additionneur utilisé pour effectuer un calcul de *OU* global. Afin de remédier à ce problème, il est nécessaire d'envisager l'utilisation d'opérateurs asynchrones simples en remplacement de ceux existants actuellement.

Dans la suite de ce travail de thèse, nous définissons une nouvelle architecture pour la couche asynchrone permettant d'effectuer des *prefix-associations*, qu'elles soient *rootfix* ou *leaffix*, en ayant recours à un opérateur asynchrone local simple, ayant un coût d'implantation en transistor réduit. Afin d'obtenir une solution viable et efficace, cet opérateur doit permettre d'effectuer les *prefix-associations* fondamentales avec la même complexité temporelle que la *Maille Associative d'Orsay*, à savoir un *OU* global en $\mathcal{O}(1)$ propagations, et une addition régionale en $\mathcal{O}(n)$ propagations, avec n le nombre de bits de la somme.

7

Les micropipelines associatifs : une architecture dédiée aux traitements régionaux

7.1 L'arbitre asynchrone, un élément indispensable aux *prefix associations*

7.1.1 Établissement des arbres couvrants

L'architecture de la couche asynchrone que nous allons définir a pour but de permettre le calcul de *prefix-associations* uniquement. Dans une telle structure, les arbres couvrants sont des éléments clés puisque les propagations s'effectuent uniquement sur des arbres couvrants, ceci permettant d'assurer que le calcul de l'*association* utilise les données de chaque processeur élémentaire une fois et une seule. Il est donc important de pouvoir les construire et les utiliser efficacement. L'utilisation des arbres couvrants comme support de la *prefix-association* a été présentée dans la section précédente. En revanche l'algorithme permettant l'établissement de ces arbres couvrants n'a pas été proposé. Il repose sur l'utilisation d'une structure asynchrone, l'*arbitre asynchrone* et sur un opérateur permettant de déterminer un point unique dans une région qui servira de racine à l'arbre couvrant. L'arbitre asynchrone et l'opérateur de détermination du point unique seront étudiés dans les prochaines sections.
Compte tenu de l'importance des arbres couvrants dans le calcul des *associations*, la construction de ces arbres doit pouvoir se faire de manière rapide et robuste. Un arbre couvrant ne contenant pas de boucles par définition, sa construction ne peut être que le résultat d'un processus itératif.

Nécessité de l'asynchronisme

Montrons tout d'abord la nécessité de l'asynchronisme pour établir les arbres couvrants. Supposons que l'on veuille construire un arbre couvrant de manière

massivement parallèle. Chaque pixel doit savoir si la connexion qu'il va établir ne risque pas de créer une boucle dans le réseau. De fait il doit connaître la topologie du réseau auquel il va ultérieurement appartenir. Cette connaissance ne peut toutefois pas être acquise par le biais d'opérations sur un réseau de type arbre couvrant, puisqu'elle est justement nécessaire à la mise en place de ce réseau. Il ne reste donc plus qu'une manière directe de procéder, à savoir la connaissance de chacun des pixels situés dans la région. Une telle connaissance est très coûteuse à acquérir de manière synchrone, et très difficile à utiliser de par sa très faible compacité, ce qui rend la méthode inefficace.
De fait un processus itératif de construction est nécessaire. Une approche synchrone de ce processus peut alors être envisagée. Toutefois on peut s'interroger sur le bien-fondé d'une telle approche. Les arbres couvrants sont utiles pour permettre l'utilisation de propagations asynchrones permettant de suppléer aux limitations en terme de vitesse des propagations synchrones. Partant de ce constat, il parait assez peu opportun d'utiliser une méthode de construction d'arbre par itération synchrone. La construction de l'arbre couvrant doit donc, pour être efficace, utiliser une propagation de type asynchrone.

Algorithme de construction

Convaincus de la nécessité de l'asynchronisme pour établir les arbres couvrants, nous examinons à présent un algorithme permettant de construire ces arbres. Un point particulier d'un arbre couvrant est sa racine, qui est unique et qui reçoit le résultat des *prefix-associations* sur cet arbre. Ce point est donc connecté à chaque autre processeur faisant partie de l'arbre, et dans le cas général, c'est le seul point ayant cette propriété (les connexions étant directionnelles). C'est donc par ce point qu'il est opportun de commencer la construction de l'arbre couvrant.
Il est à noter ici que le processeur racine n'étant pas défini a priori, il faut donc le choisir dans la région et de manière unique. Le choix de ce processeur unique étant une des difficultés de l'établissement asynchrone d'arbres couvrants, nous développerons ce point dans les parties suivantes.
Supposant la racine de l'arbre choisie, il faut alors construire un arbre couvrant où chaque processeur n'a qu'un seul antécédent. Pour cela il suffit d'établir toutes les connexions possibles à l'intérieur d'une région et de laisser se propager un marqueur depuis la racine dans chacun des processeurs de la région, en mémorisant la connexion entrante dans un processeur dès qu'il a été atteint par le marqueur sur une de ses entrées. Une telle propagation permet de construire très efficacement un arbre couvrant sur une région, celui-ci ayant une structure irrégulière liée à la dispersion des temps de propagation dans les structures électroniques utilisées. L'arbre obtenu est orienté de la racine vers les feuilles et permet les *leaffix-associations*. L'inversion des connexions programmables sur cet arbre permet d'obtenir un arbre utilisé dans les *rootfix-associations*.
Un problème peut toutefois se poser lors de la construction de l'arbre. Il arrive qu'un processeur voit le marqueur arriver sur deux ou plus de ses entrées à la fois. Dans ce cas, il doit donc choisir arbitrairement ou non une des entrées qui sera

considérée comme la connexion vers son antécédent dans l'arbre. Ce choix doit être fait de manière asynchrone et il requiert un composant dédié présent dans la structure de la *Maille Associative d'Orsay*, l'arbitre asynchrone. Nous détaillons cette structure dans la section suivante.

Robustesse

Un des aspects essentiels des primitives de calculs régionaux est leur robustesse. Ces primitives ayant comme support un circuit électronique, il est important de prendre en compte une éventuelle défaillance de quelques processeurs isolés de ce circuit. Comme nous le verrons plus tard, la partie asynchrone de notre structure se réduit à un opérateur de propagation. Une défaillance du circuit consiste pour notre circuit en une absence de propagation dans le processeur défectueux.
De fait lors de l'établissement de l'arbre couvrant, le processeur défectueux, ne propagera pas le marqueur qui lui arrive. Il n'aura donc pas de connexion sortante, ce sera donc une feuille de l'arbre couvrant. Après inversion du réseau pour effectuer des *rootfix-associations* telles que des calculs de somme, la seule donnée transitant dans ce processeur est sa valeur locale. Une panne dans le chemin asynchrone implique donc une perte de la donnée locale uniquement. Cette propriété confère à la primitive d'établissement d'arbre couvrant une bonne robustesse aux défaillances du circuit.
Toutefois, il est à noter que ces propriétés ne sont plus valables dans le cas où le pixel défaillant se situe dans une zone ne faisant qu'un seul pixel d'épaisseur. Dans ce cas, le seul réseau de connexion implantable localement est de type linéaire. Couper la propagation dans un tel réseau conduit à isoler une partie du circuit qui ne sera pas couverte par l'arbre couvrant. Il est à noter que ce cas de figure rejoint celui de l'additionneur bit-série linéaire, proposé par l'Université de Tokyo [KKI04], qui présente une robustesse limitée aux défaillances.

7.1.2 Recherche d'un point singulier dans une région

Comme indiqué à la partie 7.1.1, la construction des arbres couvrants nécessite de choisir un point unique dans une région qui sert de racine.

Axiome du choix

D'un point de vue formel, le choix de ce point unique relève de la problématique de l'*axiome du choix*. Cet axiome est défini comme suit :

> Étant donnée une famille d'ensembles non vides, il existe une fonction qui à chacun d'entre eux associe un de ses éléments. Cette fonction est appelée fonction de choix.

Cette formulation littérale correspond à la formulation mathématique suivante où $P(E)$ est l'ensemble des sous-parties de E :

$$\forall C \subset P(E), \quad \exists f : C \to E \quad \text{tq} \quad \forall X \in C, \quad f(X) \in X$$

Il est toutefois important de noter que cet axiome fait partie des axiomes optionnels et controversés de la théorie des ensembles. Étant relative à un axiome, la fonction de choix n'est donc pas définie par construction. En outre, l'utilisation de l'axiome du choix conduit parfois à certains résultats contraires aux conceptions usuelles et implique l'existence d'objets étranges ou contre-intuitifs. Un exemple de ces étrangetés, cité ici sans explication et à titre d'illustration anecdotique, est la décomposition paradoxale de Banach-Tarski qui, en utilisant l'axiome du choix permet de démontrer qu'il est possible de découper une sphère en un nombre fini de morceaux et de les déplacer par une suite de mouvement rigides (translation et rotation), en permettant à certaines pièces de traverser d'autres pour les rassembler en formant deux copies de la sphère d'origine.
Finalement, l'axiome du choix peut paraître d'un intérêt limité et c'est pourquoi certains mathématiciens se montrent plus satisfaits d'une démonstration s'ils peuvent éviter d'avoir recours à cet axiome. Cependant, il est utilisé sans réticence particulière surtout dans des variantes plus faibles telles que l'axiome du choix dénombrable qui est la restriction de l'axiome du choix à une famille dénombrable de sous-ensembles de E.

Recherche d'un point singulier dans une région

Un parallèle peut être établi entre le problème de la détermination d'un point singulier dans une région et la nécessité ou non d'avoir recours à l'axiome du choix. Les situations où l'axiome du choix est nécessaire pour conclure sur l'existence d'une fonction de choix sont les suivantes : il s'agit de cas où il est impossible de choisir de manière algorithmique entre deux ou plusieurs processeurs lequel sera utilisé comme point singulier (racine). Dans le cadre du traitement d'image, ce cas de figure peut se produire dans une région autonome, qui ne connaît ni son orientation, ni sa position dans l'image. Dans ce cas, une région de type carré de 4 processeurs est invariante par rotation d'angle multiple de $\frac{\pi}{2}$, et chacun des processeurs a donc un rôle parfaitement invariant par rotation. Le recours à la fonction de choix existant de manière théorique est donc nécessaire pour choisir un processeur racine unique.

Les cas de figure, pour lesquels le recours à une fonction de choix que l'on ne peut déterminer serait nécessaire, sont donc les situations où la région ne connaît rien sur son orientation et sa position. Dans le cas contraire, il est toujours possible de choisir par exemple l'ensemble des pixels les plus à droite, puis le pixel le plus en haut parmi cet ensemble. De la sorte nous obtenons une fonction de choix parfaitement déterministe sur toute région de l'image. Ainsi, dans le cas où la fonction de choix ne peut être définie (son existence est assurée d'un point de vue théorique, mais en pratique cela ne sert à rien), il est nécessaire d'ajouter une information dans les régions de manière à rendre cette fonction définissable. Une telle information peut être la notion d'orientation ou de position dans l'image.

Recherche d'un point singulier dans une région à l'aide d'opérateurs abstraits

Nous étudions à présent l'effet de l'ajout de l'information d'orientation ou de position de la région en termes algorithmiques, et en termes de coût d'implantation. Considérons tout d'abord le cas de l'ajout de l'information de position. Cette information revient à étiqueter chacun des pixels de la région avec ses coordonnées en abscisse et en ordonnée. La détermination d'un point unique dans la région est dans ce cas très simple : il suffit de déterminer par exemple quel pixel est le plus en bas à droite, ce qui est fait en appliquant consécutivement un minimum sur les ordonnées, ce qui permet de sélectionner tous les pixels de la région ayant une ordonnée minimale, puis de sélectionner parmi ces pixels celui qui a une abscisse minimale. La complexité algorithmique de cette opération est donc en $\mathcal{O}(log(L)+log(l))$ avec L et l la longueur et la largeur de la rétine (on ne peut en effet pas se limiter à la taille de la région considérée, car on ne connaît pas sa taille).
Considérons à présent le cas de l'ajout de la notion d'orientation de la région uniquement, sans que celle-ci ne puisse connaître sa position. Cette information est nettement moins riche que la précédente dans la mesure où les relations entre pixels ne sont connues que de manière locale, et qu'il n'est pas possible de trouver un minimum sur la région, les pixels n'étant pas étiquetés. Dans cette situation, il est nécessaire de procéder à l'étiquetage des pixels afin de se ramener au cas précédent. Cet étiquetage est le produit d'une *association*, cependant le résultat de cette association n'est pas le même dans chaque pixel (sans quoi l'étiquetage n'aurait pas d'intérêt). De plus cette *association* ne peut être une *prefix-association*, car celles-ci ont recours à la structure d'arbre couvrant, ce qui est impossible sans avoir au préalable un point unique comme racine. L'opérateur utilisé par l'association doit donc être idempotent (pour permettre la *direct-association*) et ne pas donner le même résultat dans chaque pixel. De tels opérateurs dissymétriques ont été étudiés dans la thèse de Bertrand Ducourthial [Duc00][DM98]. Ils sont appelés *R-opérateurs*, par extension des *S-opérateurs* qui sont symétriques.

Un *R-opérateur* permettant d'établir un étiquetage des abscisses sur une région en ne connaissant que son orientation est le suivant :

$$\oplus = max(E-1, N, S, O+1)$$

La numérotation est effectuée de l'Ouest vers l'Est en attribuant l'abscisse 1 aux pixels le plus à l'Ouest. De même un R-opérateur permettant d'établir un étiquetage des ordonnées du Sud vers le Nord en attribuant 1 à l'ordonnée du pixel le plus au Sud et en ne connaissant que l'orientation de la région est :

$$\oplus = max(E, N-1, S+1, O)$$

Dans ces deux opérateurs, N, S, E et O représentent les valeurs locales placées en chacun des noeuds du réseau associatif et qui sont affectées par la relaxation du réseau.

Cette solution, intéressante d'un point de vue théorique a toutefois des applications limitées en pratique dans la mesure où l'implantation bit série de *R-opérateurs* du type de ceux qui ont été proposés n'existe pas. Il s'agit en effet de faire cohabiter des fonctions ayant une priorité aux bits de poids faibles dans l'ordre des calculs (fonctions telles que l'addition) avec des fonctions ayant une priorité aux bits de poids fort dans les calculs (fonctions telles que l'opérateur max).

Finalement, la contrainte minimale permettant d'extraire un point unique d'une région est l'ajout de l'information de position de la région dans l'image. Le coût algorithmique associé à cette contrainte est très faible, et le coût matériel est quant à lui nul (si l'on accepte de charger cette information depuis les bords de la rétine lorsque cela est nécessaire).

Recherche d'un point singulier dans une région à l'aide d'opérateurs implantés électroniquement

Dans le cas d'implantations électroniques telles que les rétines artificielles, il est envisageable d'utiliser les propriétés des circuits électroniques pour réaliser la fonction de choix. Une solution envisageable d'un point de vue fonctionnel est l'utilisation d'un arbitre permettant de sélectionner un pixel parmi n candidats. Cette solution est en réalité impossible à implanter car elle nécessiterait de pouvoir utiliser un arbitre avec un grand nombre d'entrées, l'arbitre pouvant être déplacé selon la configuration de la région dont il faut déterminer la racine. Il serait toutefois intéressant d'étudier cette idée d'un point de vue théorique pour voir où elle se situe par rapport aux solutions proposées précédemment.
L'utilisation d'un arbitre électronique (dont la structure sera décrite à la section suivante) permet de sélectionner une entrée active parmi n. Supposons que le circuit électronique soit parfait (sans aucun bruit) et que les tensions d'entrée soient rigoureusement les mêmes. Dans ce cas, la situation de chaque entrée est la même et l'arbitre a des difficultés à choisir une des entrées. Une telle situation, appelée *métastabilité* ne peut converger que par ajout d'une perturbation modifiant l'état d'une des entrées. Une telle perturbation peut être un bruit électronique, une non homogénéité des tensions des caractéristiques des transistors, une particule venant frapper un fil du circuit... La convergence de l'opération d'arbitrage est donc assurée en pratique par une absence d'invariance spatiale, mais elle ne constitue en aucun cas une fonction de choix sur une région au sens mathématique du terme.

7.1.3 L'arbitre asynchrone

Comme indiqué à la partie 7.1.1, il est nécessaire d'avoir dans le chemin asynchrone un arbitre asynchrone. Cet arbitre sert à établir les arbres couvrants sur les régions en sélectionnant un unique antécédent (une unique connexion entrante) pour chacun des pixels de l'arbre. Dans cette partie nous étudions la structure de cet arbitre asynchrone d'un point de vue électronique.

Tour d'horizon des arbitres existant dans la littérature

Avant d'étudier plus en détails le fonctionnement des arbitres asynchrones, un tour d'horizon des différents arbitres que l'on peut rencontrer dans la littérature s'impose. Ces différentes structures ont été répertoriées dans [Lit03]. Un arbitre a pour fonction, dans le cas où plusieurs signaux d'entrée deviennent actifs en même temps, d'en choisir un et un seul. Cette fonction peut être réalisée de plusieurs manières. Il existe trois grandes catégories comportementales d'arbitres : les arbitres parallèles, les arbitres série et les arbitres par élimination successive.
Les arbitres parallèles peuvent être de deux types, synchrone ou asynchrone. Un arbitre parallèle synchrone fonctionne de manière déterministe, c'est à dire que l'arbitrage sera toujours le même si l'on renouvelle une expérience identique plusieurs fois de suite. Un tel arbitre repose sur une structure de type encodeur de priorité. L'arbitre mémorise les connexions actives au moment de l'arrivée des signaux à l'aide d'un verrou déclenché dès qu'une ou plusieurs entrées deviennent actives, et choisit ensuite de manière déterministe et synchrone le signal en faveur duquel il effectue son arbitrage grâce à l'encodeur de priorité (fig. 7.1).
Un arbitre parallèle asynchrone fonctionne de manière aléatoire. Il est constitué de

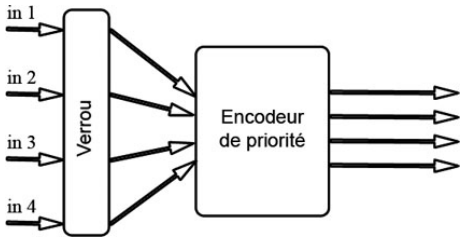


FIG. 7.1 – Arbitre parallèle à encodeur de priorité

composants ayant des fonctionnalités identiques sur chacune des entrées. Le composant usuel utilisé pour la réalisation d'un arbitre asynchrone est la bascule RS (fig. 7.2). Cette bascule n'ayant que deux entrées, il est nécessaire de cascader $n-1$

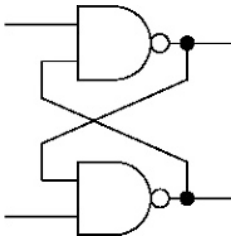


FIG. 7.2 – Structure d'une cellule RS

cellules d'arbitrage de ce type pour réaliser un arbitre à n entrées. Pour ce faire, les sorties complémentées de chaque arbitre doivent être réunies entre elles par un opérateur de type *non OU exclusif* afin que les signaux placés sur chacune des deux sorties de l'arbitre soit transmis au successeur de ce dernier. Il serait théoriquement possible d'utiliser un *non OU* logique en lieu et place du *non OU exclusif*, mais

cette structure ne garantit pas parfaitement l'exclusion mutuelle des deux sorties. Si l'une des deux portes NAND de l'arbitre est plus lente que l'autre, la porte lente peut ne pas avoir terminé de revenir à 0 lorsque la porte rapide sera déjà passée à 1. Il y a donc chevauchement temporel des états logiques 1 sur les deux sorties, et le *OU* logique reste donc dans ce cas dans le même état logique 1 en sortie. Cette exclusion mutuelle étant très importante, le *non OU exclusif* s'impose. L'arbitre

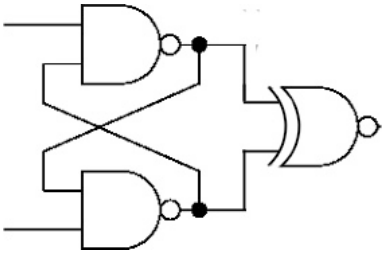


FIG. 7.3 – Arbitre à 2 entrées, avec propagation de la sortie par *non OU exclusif*

construit à base de XOR est toutefois insuffisant. Si l'une des entrées passe à 1 tandis que l'autre passe à 0 simultanément, la sortie du XOR ne va pas changer, ce qui peut provoquer la perte d'informations. Il est donc nécessaire d'utiliser une autre structure permettant d'assurer le retour à 0 de la sortie lors des transitions : il s'agit de l'*exclusion mutuelle*. L'exclusion mutuelle réalise de plus une inversion logique, il est ainsi possible d'utiliser une porte logique OU pour réunir les deux signaux (fig. 7.4). La maille associative d'Orsay utilise un arbitre de ce type, son

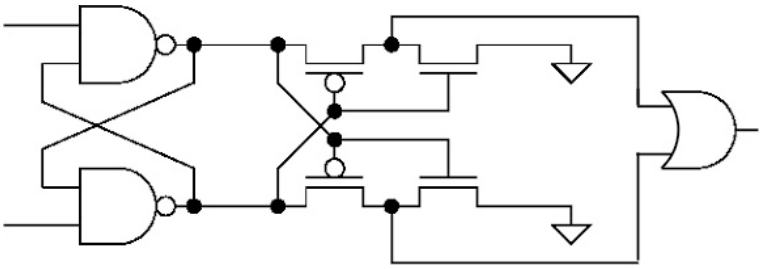


FIG. 7.4 – Arbitre à 2 entrées, exclusion mutuelle et propagation de la sortie par *OU* logique

schéma est proposé à la figure 7.5. Le coût d'implantation en transistor d'un arbitre parallèle asynchrone peut être séparé en deux parties, le coût dû aux bascules et le coût dû aux portes *NAND* destinées à interpréter les sorties des bascules. Le coût lié aux bascules croît linéairement avec le nombre d'entrées n, le nombre de bascules étant $n-1$. Le coût lié aux portes *NAND* croît en $\mathcal{O}(n\log(n))$ car le nombre de porte *NAND* nécessaire est n, le nombre d'entrée de chaque porte est $\log(n)$ et le coût en transistor de chaque porte *NAND* est proportionnel à son nombre d'entrées. L'utilisation d'un tel arbitre doit donc se faire pour un nombre d'entrées assez faible.

Un arbitre série ou *daisy-chain* peut également être de type synchrone ou asynchrone, mais son fonctionnement est basé dans les deux cas sur la propagation

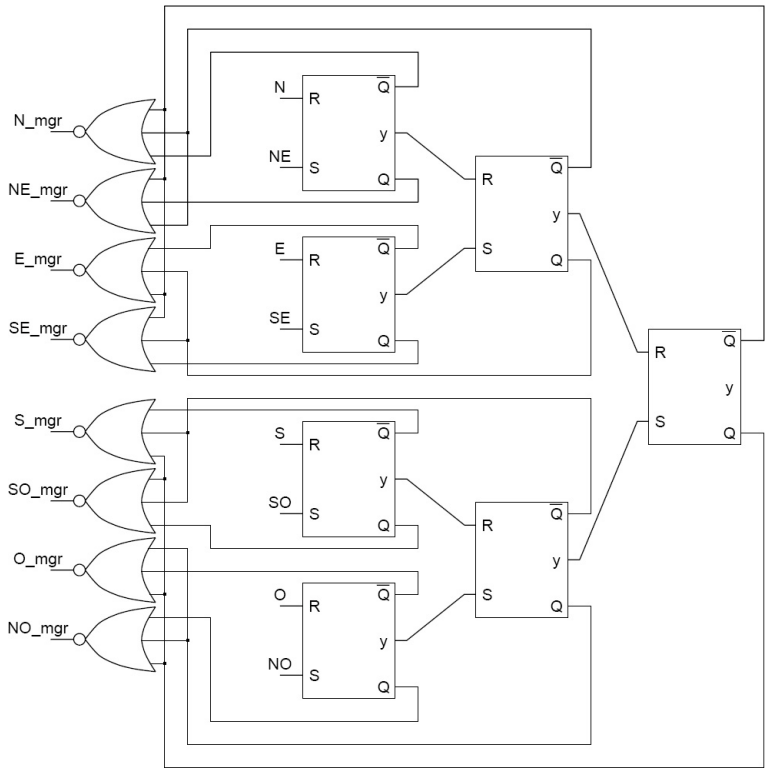


FIG. 7.5 – Arbitre parallèle asynchrone de la maille associative d'Orsay

d'un jeton tournant. Cette propagation peut être synchrone ou asynchrone. Le jeton active successivement chacun des détecteurs de signaux placés sur les entrées du processeur. Si un détecteur activé par le jeton tournant détecte qu'une entrée est active, alors le jeton est arrêté et l'entrée correspondante est sélectionnée. Compte tenu du mode de propagation, le jeton ne peut activer qu'un seul détecteur à la fois, ce qui assure que l'arbitre ne sélectionne qu'une seule entrée à la fois. Une fois l'entrée désactivée, le jeton recommence à tourner. Un tel arbitre a un avantage, qui est son coût matériel linéaire en fonction du nombre d'entrées (fig. 7.6). Toutefois, son

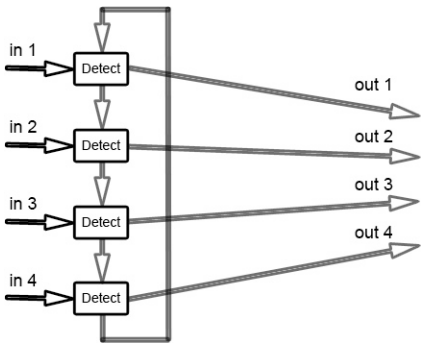


FIG. 7.6 – Arbitre série à jeton

fonctionnement a également un inconvénient important : la propagation d'un jeton

en continu consomme de l'énergie, ce qui proscrit son utilisation dans un système basse consommation. Une manière de remédier à ce problème est de déclencher la propagation du jeton tournant uniquement lorsqu'un signal est disponible sur une entrée. Dans ce cas, le jeton fait au maximum un tour avant d'arriver à l'entrée active. Cette solution a cependant un coût en transistors élevé, de par l'utilisation d'un détecteur sur chaque entrée, du réseau de propagation du jeton et d'un *OU* permettant de savoir si l'une des entrées est active pour libérer le jeton.
Enfin, l'arbitre à éliminations successives est de type asynchrone. Les entrées sont hiérarchisées entre elles. L'arbitrage est initié par une commande. L'entrée active ayant le plus haut niveau hiérarchique prend la main et bloque chacune des autres entrées. Une particularité de ce type d'arbitre est qu'un signal de haute priorité peut venir interrompre un signal de priorité plus faible si un arbitrage est déclenché durant l'utilisation du signal de priorité plus faible. Un tel fonctionnement hiérarchique peut être intéressant dans une structure de type micro-contrôleur standard, mais elle n'a que peu d'intérêt dans le cas des mailles où il n'y a pas de hiérarchie entre les entrées.

Choix de l'arbitre

Parmi les arbitres présentés ci-dessus, l'arbitre asynchrone ayant le meilleur compromis entre coût d'implantation et consommation d'énergie est l'arbitre parallèle asynchrone utilisé dans la *maille associative d'Orsay*. Cet arbitre aurait un coût relativement important de l'ordre de 140 transistors dans une version 4-connexe de la *Maille Associative*. Malgré son coût, il est utilisé uniquement pour la construction des arbres couvrants. La section suivante propose d'en tirer parti de manière plus poussée en construisant une architecture basée sur cet arbitre.

7.2 Un mode alternatif de transmission asynchrone bit-série : la *transmission par jetons*

Les avantages et inconvénients liés à l'utilisation de l'asynchronisme ont été présentés à la partie 5.2.5. Un de ces inconvénients, l'augmentation de la surface de silicium utilisée dans les circuits, pose un problème majeur pour l'utilisation de l'asynchronisme dans les rétines artificielles. Le nombre de transistors étant très limité, il est important de réduire au maximum la taille de chacune des fonctionnalités du processeur élémentaire. Pour ce faire, nous étudions les modes de communication asynchrone afin de chercher quel est le meilleur compromis *coût matériel - capacités de transmission* que l'on peut obtenir dans le contexte de notre application, à savoir la réalisation d'opérations bit-série régionales avant de proposer un mode de communication alternatif aux solutions existantes.

7.2.1 Comparaison des différentes méthodes de communication asynchrone dans le cas d'opérations bit-série

Différentes techniques de communication asynchrones existant dans la littérature ont été présentées à la partie 5.2.2. A ces techniques s'ajoute la communication asynchrone sans protocole de communication utilisée dans l'additionneur bit série linéaire [KKI04], et dans la maille associative d'Orsay ([Moh96][Dul96]. Dans cette partie nous analysons quels sont les avantages et inconvénients de ces différents types de transmission, avant de proposer une nouvelle méthode de transmission basée sur le concept de jeton.

- **Communication asynchrone sans protocole** : utilisé dans la maille associative, ce mode de communication peut être assimilé à un fonctionnement de type combinatoire. Cette forme d'asynchronisme sans contrôle est efficace pour réaliser des *associations*. Toutefois elle requiert des opérateurs dédiés tels qu'un additionneur ou des portes logiques dans le chemin asynchrone. Ces opérateurs dédiés ont un coût matériel important, qui contrebalance l'économie de coût faite sur les transmissions asynchrones utilisant un simple fil entre processeurs élémentaires adjacents.

- **Communication utilisant un bus à données groupées** (voir fig. 5.3 à la section 5.2.2) : ce mode de communication utilise une structure de contrôle de type micropipeline associée à une transmission de la donnée sur un fil. En outre il faut ajouter des retards dans la structure de contrôle, le fonctionnement global étant celui d'un circuit de type synchrone où les synchronisations sont effectuées localement. Une telle structure présente un surcoût important dans le cas d'une implantation de type calcul réparti bit-série. Le bit à transmettre de cellule en cellule est codé sur un fil de donnée auquel il faut ajouter un réseau de synchronisation à base de micropipelines, des cellules de retard, et des registres de stockage placés sur le fil de donnée. Cette structure, outre ses performances limitées du fait de l'utilisation d'un modèle de type Huffman avec insertion de retard à un coût élevé en transistor. Ce coût peut se justifier dans le cas d'une transmission de données sur un bus à plusieurs fils, dans le cas d'une transmission sur un fil, le surcoût à payer est trop important.

- **Communications utilisant un bus double rail** (voir fig. 5.6 à la section 5.2.2) : ce mode de communication repose sur un codage des données utilisant deux fils par bit de donnée. Il permet d'effectuer des transmissions en fusionnant les signaux de contrôle et de données. Dans le cas de la transmission d'un seul bit de donnée, le surcoût à payer par rapport à une transmission directe sur un fil est la double structure de contrôle de type micropipeline associée aux communications de type double rail. Une telle structure présente un coût relativement important, qui est toutefois moins élevé que celui de la structure précédente. Un point vient ternir cette solution : les opérateurs asynchrones associés à une telle structure ont un coût prohibitif.

Finalement, les deux modes de communication présentant les meilleures propriétés en terme de coût d'implantation sont le bus double rail et les communications combinatoires. La représentation d'un bit de donnée associée au bus double rail est un créneau de tension sur l'un des deux fils de requête, créneau dû au protocole de communication à 4 phases. Nous appelons ce créneau un *jeton*. La représentation d'un bit de donnée associée aux communications combinatoire est un niveau de tension.

7.2.2 Définition d'un nouveau mode de communication asynchrone insensible aux délais

Les deux modes de communication présentant les meilleurs potentialités sont très différents, sans pour autant nous satisfaire totalement. Un mode de communication alternatif à ces méthodes sera proposé, ayant un coût d'implantation inférieur dans le cadre d'une implantation adaptée à nos contraintes. Ces contraintes sont les suivantes :

- Les opérations à effectuer sont de type bit série afin de pouvoir les effectuer itérativement sur un bit de donnée.

- Parmi les opérations qu'il est nécessaire de pouvoir effectuer au minimum figurent l'addition et le *OU* global. La complexité temporelle maximale admissible pour effectuer ces opérations a été définie à la section 6.2, à savoir un *OU* global en $\mathcal{O}(1)$ propagations, et une addition régionale en $\mathcal{O}(n)$ propagations, avec n le nombre de bits de la somme.

- L'implantation matérielle doit être peu coûteuse.

A ces contraintes s'ajoute une possibilité : celle de séquentialiser les phases asynchrones et d'effectuer des opérations synchrones de type SIMD entre les phases asynchrones.

Nous débutons l'analyse de ces contraintes en se focalisant sur la réalisation de l'addition. Effectuer une somme nécessite de pouvoir compter des bits. Pour cela la représentation d'une donnée sous forme binaire n'est pas forcément nécessaire. Supposons que les informations soient codées sous forme de jetons, un jeton signifiant qu'il faut incrémenter la somme d'une unité, l'absence de jeton signifiant qu'il ne faut pas toucher à la valeur de la somme. Le codage physique d'un jeton est défini à la section 7.2.1 Dotons à présent une région d'une structure d'arbre couvrant propageant les jetons vers la racine de l'arbre, les jetons ne pouvant être fusionnés. Il suffit de placer dans la racine de l'arbre un compteur destiné à compter les jetons qui passent pour connaître le nombre de jetons initialement présents dans l'arbre, puisque tous les jetons passeront dans la racine.

L'analogue de ce dispositif pourrait être un compteur de voitures sur un réseau autoroutier convergent. Pour compter le nombre de voitures présentes sur le réseau, il suffit d'orienter toutes les voitures présentes sur un réseau autoroutier vers un péage unique dans lequel elles sont forcées de passer pour sortir de réseau.

Le codage sous forme de jeton des bits de la somme à additionner suffit donc au calcul de cette somme (nous verrons que ce codage peut être réalisé de manière simple et peu coûteuse). Nous verrons dans la suite qu'il est possible de se passer du compteur de jetons localisé dans la racine de l'arbre, et qui aurait un coût matériel important. Examinons à présent l'implantation du *OU* global à base de jetons. Son fonctionnement est très simple : il suffit de placer un jeton dans chacun des processeurs de l'arbre couvrant ayant un état logique 1 et contribuant au calcul du *OU* global. Le réseau de propagation sur arbre couvrant fait donc propager les jetons vers la racine. Si le compteur de jeton placé dans la racine voit passer un jeton ou plus, alors le résultat du *OU* global est 1, 0 sinon.

A présent, examinons comment un tel codage de l'information satisfait à la contrainte de réaliser un *OU* global en $\mathcal{O}(1)$ propagations, et une addition régionale en $\mathcal{O}(n)$ propagations, avec n le nombre de bits de la somme. Pour le *OU* global, le résultat est clairement obtenu en une propagation. Pout l'addition régionale, il est possible d'obtenir dans la racine une somme de bits réparti sur la région en une propagation, à condition que chaque bit ait le même poids. Pour réaliser une addition sur des données de taille maximale k bits dans chaque pixel, il est nécessaire de réaliser k propagations, qui vont permettre d'obtenir k sommes de poids différents dans la racine. Il suffit de terminer le processus en ajoutant (avec les pondérations nécessaires) ces k sommes de manière synchrone et locale pour obtenir les n bits de la somme. Le nombre n de bits de la somme étant nécessairement supérieur ou égal au nombre k de bits de chacun des termes de la somme, le calcul de l'addition régionale s'effectue donc en moins de n propagations, ce qui répond à la contrainte fixée.

Le mode de transmission défini permet donc de répondre aux deux premières contraintes que nous nous sommes fixées. Avant de conclure sur la pertinence de ce mode de transmission il est nécessaire d'examiner le coût de la structure matérielle permettant d'effectuer les opérations de propagation de jetons. Cette structure est bien connue, il s'agit de la structure de contrôle des micropipelines introduite pas Ivan E. Sutherland [Sut89] (fig. 7.7), qui convient pour effectuer des transmissions linéaires. Le modèle de propagation associé est de type insensible aux délais (DI) localement, et la structure a un coût très réduit : 2 C-éléments par processeur élémentaire seulement.

Finalement, le mode de transmission défini permet de répondre à chacune des contraintes que nous nous sommes fixées. Son mode de transmission est un protocole à phases standards illustré à la figure 7.8. Ce mode de transmission, basé sur le concept de jeton, se situe entre les communications combinatoires et les transmissions double rail :

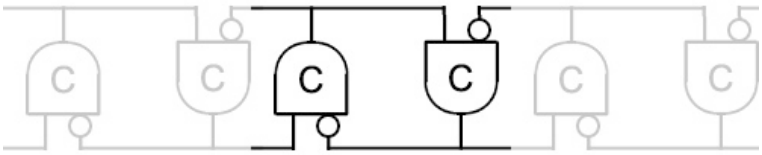


FIG. 7.7 – Structure permettant la transmission par jetons

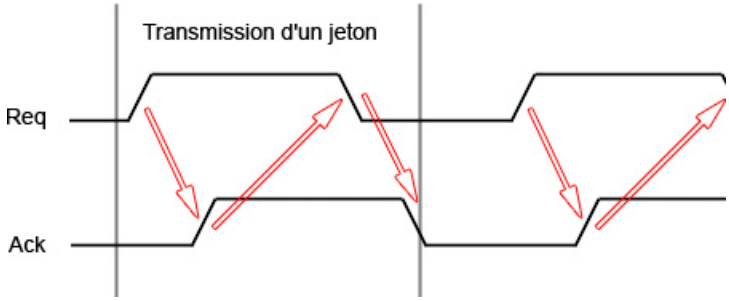


FIG. 7.8 – Protocole de transmission par jetons

– Le niveau d'abstraction de l'information manipulée, le jeton, est supérieur au niveau d'abstraction d'une simple valeur binaire placée sur un fil : la propriété importante des jetons est que placés sur un réseau linéaire, ils ne se fusionnent pas. Cette propriété n'est pas vérifiée pour les valeurs binaires placées sur un fil. Cette élévation du niveau d'abstraction de l'information manipulée permet de diminuer la spécificité des opérateurs d'addition mis en jeu. Un seul opérateur de comptage situé dans la racine est nécessaire, alors que dans le cas de valeurs binaires transmises sur des fils, il est nécessaire de placer un additionneur bit-série dans chaque processeur (nous verrons dans la suite que cet opérateur de comptage est en réalité remplacé par un opérateur distribué).

– L'abstraction d'un jeton est de niveau inférieur à l'abstraction des données binaires non fusionnables manipulées dans les transmissions double rail. Ces données binaires sont codées comme deux jetons, mais comme nous l'avons montré, un tel niveau d'abstraction n'est pas nécessaire pour répondre aux contraintes que l'on s'est fixées.

De même, la structure permettant la transmission des jetons se situe au niveau complexité matérielle entre le simple fil et la structure permettant un fonctionnement double rail. Le mode de transmission de l'information proposé constitue donc un compromis entre une transmission combinatoire et la transmission double rail, de manière à diminuer le coût matériel conjugué des structures de calcul et de propagation mises en jeu. Ce mode de transmission a une propriété intéressante et inhérente au fonctionnement de la structure de contrôle des micropipelines : il est insensible aux délais (DI), ce qui constitue la forme la plus pure d'asynchronisme.

Ce type de fonctionnement, souvent associé à des structures complexes dès lors qu'il est utilisé dans des circuits complexes, conduit ici à une structure très simple et épurée de fonctionnalités parasites telles que l'ajout de délais. Nous appelons le mode de transmission asynchrone défini dans cette partie *transmission par jetons*.

Dans la partie suivante nous voyons comment étendre le *transmission par jetons* d'une structure linéaire à une structure de type arbre couvrant utilisée par les *prefix-associations*. Pour cela, nous nous attachons à tirer parti au mieux du composant clé nécessaire à l'établissement d'arbres couvrants, l'arbitre asynchrone.

7.3 Les micropipelines associatifs et leur implantation

Dans les parties précédentes, conduits par la nécessité d'effectuer des opérations régionales et en gardant à l'esprit la contrainte portant sur le nombre de transistors pouvant être dédiés à l'asynchronisme dans chaque processeur élémentaire d'une rétine artificielle, nous avons examiné les structures permettant de faire des calculs régionaux, afin d'analyser leurs avantages et leurs inconvénients. Finalement, nous avons montré que la seule primitive absolument nécessaire est la *prefix-association*, avec comme contrainte la possibilité de calculer un OU global en $\mathcal{O}(1)$ propagations et une somme globale en $\mathcal{O}(n)$ propagations, n étant le nombre de bits de la somme. Nous avons également montré qu'un opérateur combinatoire dédié tel qu'un additionneur global ne peut répondre à ces contraintes dans le cas de l'utilisation unique de *prefix-associations*, un OU global nécessitant dans ce cas $\mathcal{O}(n)$ propagations.
D'autre part, nous avons étudié les modes possibles de communication asynchrone afin de déterminer un mode de communication permettant d'effectuer des opérations d'addition et de OU global sur une région avec les contraintes de complexité temporelle rappelées ci-dessus. Cela nous a conduit à définir un mode de communication insensible aux délais adapté au calcul bit série sur une région : la transmission par jetons.
Cette partie reprend les résultats de ces deux analyses afin de proposer un modèle et une structure électronique associée permettant d'effectuer des *prefix-associations* sur des régions. Comme nous l'avons montré, cela permet d'envisager une implantation limitée du modèle des réseaux associatifs dans les rétines artificielles. Pour cela, nous utilisons le mode de transmission par jetons défini précédemment, la structure de contrôle des micropipelines qui lui est associée, et la structure d'arbitre indispensable à l'établissement des arbres couvrants. Nous sommes amenés dans cette partie à définir une architecture pour le chemin asynchrone permettant de mettre en commun ces différentes ressources tout en limitant le nombre de transistors utilisés au strict minimum.

7.3.1 De l'arbitre asynchrone au multiplexeur automatique

Nous débutons cette partie par une étude de l'arbitre asynchrone en vue de l'intégrer à la structure de contrôle des micropipelines. L'arbitre asynchrone est indispensable à la construction des arbres couvrants et doit donc être présent dans le chemin asynchrone. Le coût de l'arbitre asynchrone étant élevé (comme indiqué à la section précédente), il est important d'en tirer le meilleur parti possible. Nous montrons dans cette partie qu'un arbitre asynchrone peut être utilisé comme un multiplexeur automatique. Nous montrons également que cette utilisation conduit naturellement à utiliser un codage de l'information sous forme de jetons.

La fonction de l'arbitre asynchrone est de choisir une et une seule de ses entrées actives. La sortie correspondant à l'entrée sélectionnée est alors activée à l'exclusion de toute autre. Une telle fonctionnalité est intéressante dans la mesure où elle peut permettre de transférer une par une les données du processeur considéré vers son successeur dans l'arbre couvrant. Pour cela, les sorties de l'arbitre doivent toute être collectées dans une sortie commune, liée au processeur suivant dans l'arbre couvrant. Pour que le processeur suivant puisse faire la distinction entre deux données successives, il est indispensable que les données ne se chevauchent pas. On utilise pour cela l'opérateur d'*exclusion mutuelle* avant de récupérer les données provenant des deux sorties grâce à une porte *OU*. Une telle fonctionnalité a déjà été décrite à la partie 7.1.3. Nous rappelons cette structure à la figure 7.9.

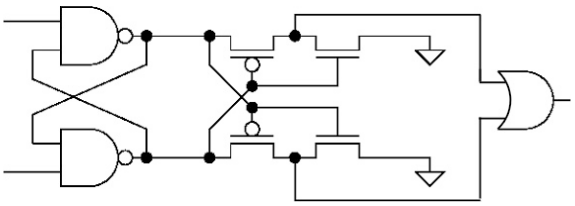


FIG. 7.9 – Arbitre à 2 entrées, exclusion mutuelle et propagation de la sortie par *OU* logique

Cet ensemble {arbitre + porte *OU*} permet de transmettre le signal arrivant en premier sur une des entrées de l'arbitre vers la sortie commune de la porte *OU*. Un tel fonctionnement peut être vu comme un multiplexeur automatique sélectionnant automatiquement le premier signal présent sur une entrée pour le transmettre à la sortie. En itérant ce processus, il est possible de transmettre de manière sérielle les informations présentes sur chacune des entrées de l'arbitre. Toutefois, le type d'information qu'il est possible de transmettre grâce à cette structure est limité. L'ensemble {arbitre + porte *OU*} se comporte comme un multiplexeur automatique, qui aiguille les informations des entrées vers la sortie à condition que ces informations aient la valeur logique 1. Cette structure ne permet pas de transférer la valeur logique 0. En conséquence toute l'information à transmettre doit être représentée sous forme de *jetons*, un *jeton* étant un signal à l'état logique 1 transmis sur un réseau de propagation ayant un état logique de repos 0. L'exclusion mutuelle

assure que les jetons successifs ne peuvent pas se chevaucher.

L'usage de l'arbitre en tant que micropipeline automatique nous a conduit à un codage de l'information sous forme de jetons. Ces jetons sont définis de manière analogue aux jetons introduits dans l'étude des modes de communications asynchrones ayant abouti à la définition de la *transmission par jetons*. La structure de contrôle du micropipeline et l'utilisation de l'arbitre asynchrone en multiplexeur automatique apparaissent donc comme étroitement réunis par le mode de transmission par jetons. Le composant asynchrone les associant est étudié dans la partie suivante.

7.3.2 La structure de base des micropipelines associatifs : le micropipeline convergent

L'arbitre asynchrone permet de multiplexer, c'est-à-dire de faire converger des jetons issus de plusieurs sources vers une seule destination. La structure de contrôle des micropipelines permet de propager des jetons. La structure électronique permettant d'implanter des communications de type transmission de jetons sur un arbre couvrant (ayant des convergences) apparaît donc naturellement comme la mise en commun d'un arbitre et de la structure de contrôle des micropipelines. Cette structure, que nous appelons *micropipeline convergent* est étudiée dans cette partie.

La mise en commun de l'arbitre asynchrone associé à une porte OU et d'un micropipeline est présentée à la figure 7.10. Pour permettre la convergence et la trans-

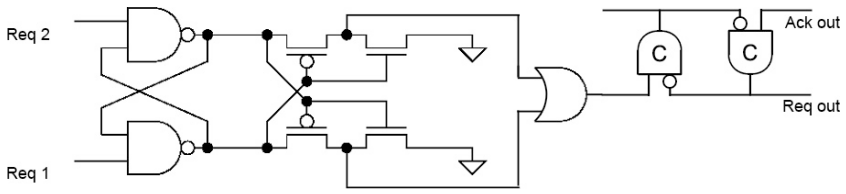


FIG. 7.10 – Association d'un arbitre à 2 entrées avec propagation par OU logique et de la structure de contrôle d'un micropipeline

mission d'informations de type jetons grâce à cette structure et un protocole de communications à 4 phases, il reste à diriger les accusés de réception des transmissions vers l'émetteur du signal. Pour cette fonction, il est nécessaire de rajouter un démultiplexeur dont les 2 voies sont pilotées par les signaux d'entrée de la porte OU. Ce démultiplexeur a pour rôle de diriger le signal *ack* vers l'entrée 1 lorsque la requête a eu lieu sur $Req1$ et vers l'entrée 2 lorsque la requête a eu lieu sur la voie 2. Le schéma final de la structure, appelée *micropipeline convergent* est proposé à la figure 7.11 [GBM05a] [GB05c].

Comme nous pouvons le constater sur la figure 7.11, l'adéquation entre la structure

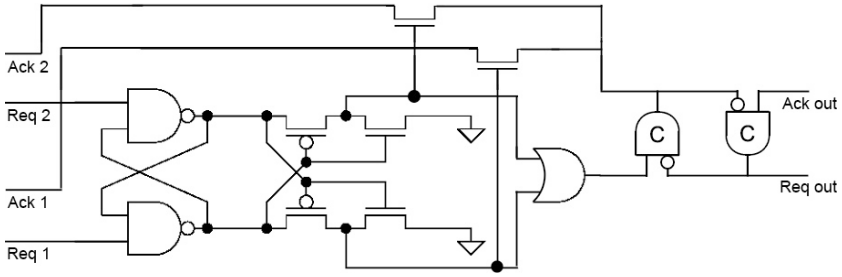


FIG. 7.11 – Micropipeline convergent à 2 entrées

de micropipeline et l'arbitre asynchrone en vue de créer une structure permettant de rassembler et de transmettre les jetons est presque parfaite, et a nécessité l'ajout de seulement deux transistors [9]pour parvenir à la structure finale du *micropipeline convergent*. Cette adéquation est principalement due au fait que les deux parties assemblées reposent sur un codage de l'information sous forme de jetons.

Le modèle des micropipelines associatifs repose sur cette structure permettant d'assurer la primitive de propagation. Cette structure, compte tenu de la contrainte d'invariance par translation du réseau doit être présente dans tous les pixels sous une forme figée. La topologie du réseau de type arbre couvrant est définie par des connexions programmables situées en entrée du chemin asynchrone du processeur élémentaire.
Il est à noter qu'une structure proche des micropipelines convergents a déjà été proposée dans la littérature [Ren00b] : il s'agit de l'arbitre asynchrone (fig. 7.12), destiné à arbitrer l'arrivée simultanée de deux données sur une même entrée d'un opérateur asynchrone. Nous avons simulé ce composant sous PSpice, et son comportement est fonctionnellement très proche de celui de notre structure. En conséquence nous ne revendiquons pas dans cette thèse le caractère novateur de cette structure (que nous avons toutefois reconstruite), mais son mode d'utilisation en tant qu'outil de calcul distribué comme nous le verrons plus tard.

La différence entre notre structure et celle proposée par M. Renaudin réside dans le fait que notre structure est plus légère (14 transistors en moins) mais n'est pas de type QDI alors que la structure proposée au TIMA est QDI. Cet affaiblissement de l'hypothèse temporelle est consécutif à l'hypothèse suivante nécessaire au bon fonctionnement de notre circuit : le temps de transmission identique entre chacune des sorties de la mutuelle exclusion et les entrées de la porte OU est identique. Cette perte de l'hypothèse temporelle QDI n'est cependant pas critique dans le cadre d'une utilisation dans les rétines artificielles. Le design du layout des processeurs élémentaire est en effet réalisé entièrement à la main, ce qui permet d'ajuster

[9]Nous n'utilisons qu'un transistor NMOS pour multiplexer chaque voie du signal au lieu des 4 transistors usuellement utilisés. Cette simplification vient du fait que l'impédance d'entrée des C-elements est forte (grille de transistor). Un 1 faible (1 logique transmis par un transistor NMOS) est donc suffisant pour définir correctement l'entrée du C-element à 1.

la longueur des connexions entre les sorties de la mutuelle exclusion et les entrées de la porte OU. De plus, en ajustant la taille des transistors de la mutuelle exclusion, il est possible d'obtenir un fonctionnement où le retour à 0 des sorties se fait plus rapidement que le passage à 1 de celles-ci[10]. Ceci nous permet d'avoir une marge suffisante pour assurer que deux jetons successifs ne seront pas fusionnés. Ainsi nous préférerons notre structure dans le cadre d'une implantation sur rétine artificielles car elle permet de réduire le coût en transistors. Dans le cas d'une implantation utilisant un routage automatique, il serait alors préférable d'utiliser la structure proposée par M. Renaudin [Ren00b].

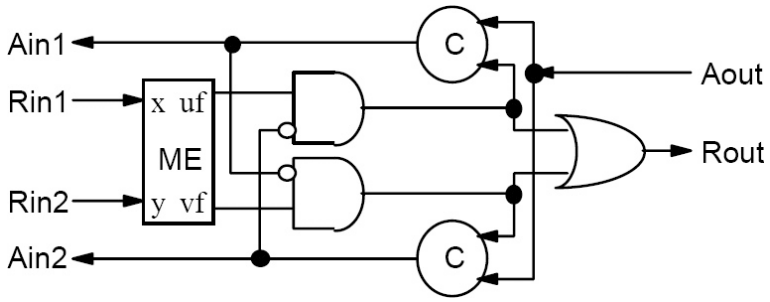


FIG. 7.12 – Arbitre asynchrone

7.3.3 Une primitive fondamentale des micropipelines associatifs : la propagation de jetons

Comme nous l'avons montré à la section 7.2.2, les propagations de jetons sur un arbre couvrant orienté vers la racine permettent, couplées à une détection des jetons passant dans la racine, d'effectuer des *prefix-associations* telles que la somme ou le OU global régional. Ces propagations peuvent se substituer à l'utilisation d'opérateurs dédiés de calculs ayant un coût matériel important, tels qu'un additionneur ou un OU logique. Elles sont donc assimilables à des opérateurs de calcul et nous leur donnons le nom de *primitive* de calcul asynchrone. De plus, les propagations n'étant pas dédiées à une tâche bien précise, elles permettent d'effectuer une plus grande diversité d'opérations.

Cette partie étudie la primitive de propagation de jetons en commençant par la forme prise par un jeton dans le réseau de propagation. Au delà du mécanisme de propagation en lui-même, il est nécessaire de pouvoir placer des jetons dans un réseau préalablement vidé, et de pouvoir lire ces jetons une fois la propagation effectuée. L'initialisation du réseau de propagation, l'insertion des jetons sur ce réseau et la lecture de ces jetons sont des points qui font l'objet d'un développement détaillé dans le cas d'un réseau simple de type linéaire. L'exemple d'un réseau de

[10]C'est déjà le cas si l'on choisi des transistors PMOS et NMOS de même taille. Dans ce cas, compte tenu de la vitesse de déplacement des porteurs de charges, le retour à 0 de la sortie s'effectue 3 fois plus rapidement que le passage à 1 de celle-ci.

type convergent (fig. 7.13) est présenté ensuite, avec en particulier l'illustration de la propriété de conservation du nombre de jetons.

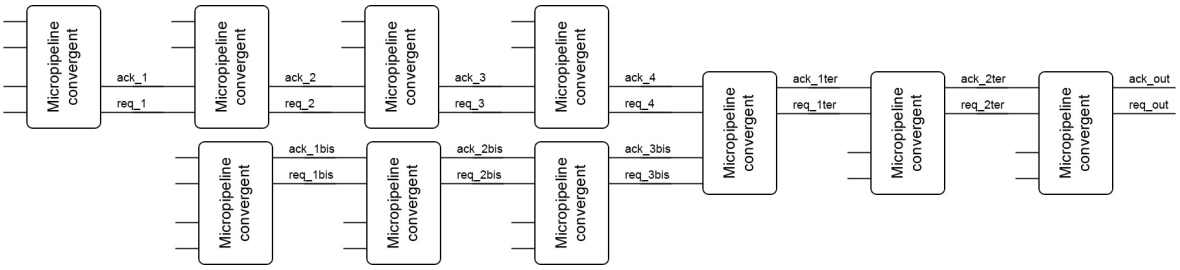


FIG. 7.13 – Exemple de réseau de propagation

Forme des jetons dans le réseau de propagation

Dans cette partie nous étudions la forme que peut prendre un jeton dans le réseau de propagation. Contrairement aux transmissions asynchrones de type donnée groupée où les données sont déplacées de registres en registres, dans une transmission de type jeton, l'information n'est pas localisée en un endroit précis, mais peut être répartie sur plusieurs pixels durant sa transmission. L'étude de la forme des jetons est réalisée en se basant sur une structure de propagation de type linéaire représenté à la figure 7.14.
Un jeton peut se concevoir comme un créneau de tension temporel mais également

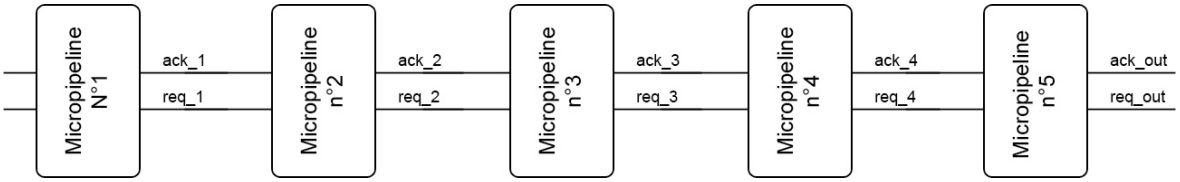


FIG. 7.14 – Exemple de réseau de propagation linéaire

spatial. Un créneau de tension temporel est caractérisé par un passage de la tension du rail à la valeur logique 1, puis à son retour à l'état initial 0. Un créneau de tension spatial sur un micropipeline est caractérisé par un passage à l'état logique 1 puis par son retour à 0 de la tension des signaux *ack* et *req* pris en alternance en suivant la structure de propagation. Sur la figure 7.15 à droite, on peut voir cette caractérisation spatiale d'un jeton :

- La tension $V(ack_1)$ à la valeur logique 0.
- Si l'on se déplace le long du réseau de propagation dans la direction de propagation, en alternant de plus entre *ack* et *req*, on constate que la tension sur le réseau a la valeur logique 1 entre les signaux $V(req_2)$ et $V(req_3)$.
- Enfin, si l'on avance encore sur le réseau, la tension vaut à nouveau 0 en ack_3.

Cette forme en créneau spatial de tension est caractéristique d'un jeton unique. Ce créneau peut avoir une taille arbitraire.

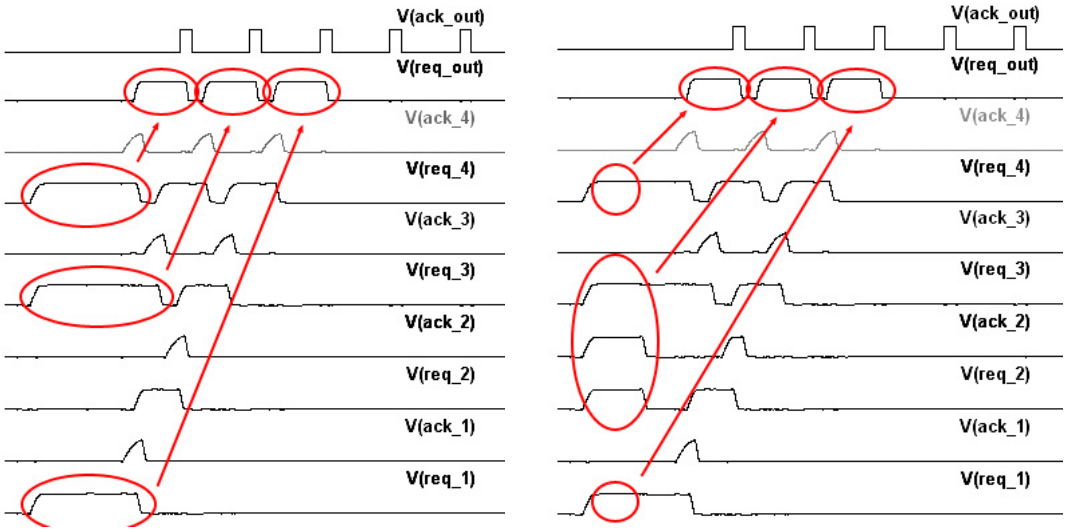


FIG. 7.15 – Propagation de 3 jetons sur une structure de type linéaire

Les chronogrammes issus de simulations SPICE présentés et commentés à la figure 7.15 montrent le résultat de la propagations de 3 jetons. Le chronogramme de gauche montre la propagation de 3 jetons placés dans 3 des micropipelines du réseau, chaque jeton étant à l'origine localisé dans une cellule de base du micropipeline (cellules 1, 3 et 4). On constate que ces trois jetons spatialement répartis se propagent et ressortent du réseau sous forme de trois jetons se succédant temporellement dans la cellule n°5. Le chronogramme de droit présente des résultats similaires, à une différence près : un jeton (le jeton central) est distribué au départ sur 2 cellules de micropipeline (cellules 2 et 3) comme nous l'avons présenté précédemment. On constate que malgré sa répartition spatiale, ce jeton est bien unique, et que le nombre de jetons propagés est bien conservé.

D'un point de vue utilisation du micropipeline en tant que vecteur de propagation de jetons, il est nécessaire de pouvoir insérer et lire la présence de jetons sur le réseau : l'insertion est décrite à la section suivante.

Initialisation de l'état du réseau de propagation

La propagation de jetons sur le réseau nécessite que celui-ci soit dans un état vide avant l'insertion des jetons. Un état vide signifie qu'il n'y a ni jetons ni informations parasites présentes sur le réseau. Pour garantir cela en toute situation, il est nécessaire de pouvoir réinitialiser ce réseau, ce qui peut se faire en ajoutant une fonction *reset* à chacun des C-elements constituant le micropipeline. Cette fonction *reset* permet de placer la sortie du C-element à la valeur logique 0. Elle peut être réalisée en remplaçant l'inverseur de sortie dans le C-element dynamique par une porte *Non-Ou* ayant une entrée connectée au condensateur interne au C-element et l'autre connectée au *reset* (fig. 7.16).

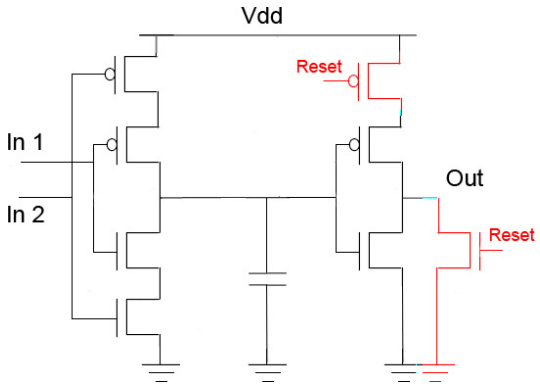


FIG. 7.16 – C-element dynamique avec reset

Initialisation des jetons

Une fois le réseau de propagation initialisé à la valeur valeur 0 logique pour chacune des sorties des C-element, il est nécessaire de pouvoir placer les jetons sur le réseau. Cette fonction conduit à définir ce qui caractérise un jeton dans le réseau. Comme nous l'avons vu, un jeton a une forme spatio-temporelle. Une initialisation sous forme temporelle étant difficile à réaliser (car devant être exécutée rapidement pour éviter des chevauchements de jetons), nous initialisons les jetons sous forme spatiale, opération pouvant être réalisée en un temps quelconque. La forme spatiale la plus compacte d'un jeton est celle où le jeton tient dans une cellule élémentaire de micropipeline, à savoir le codage suivant pour placer le jeton dans la n^{eme} cellule :

- $V(ack_{n+1}) = 0$
- $V(req_n) = 1$
- $V(ack_n) = 0$

Pour cela, il est nécessaire de pouvoir initialiser la valeur logique 1 en sortie de la moitié des C-elements. Pour cela nous modifions à nouveau le schéma précédent pour obtenir un C-element avec *reset* et *set*, le *reset* étant prioritaire. La structure obtenue est proposée à la figure 7.17. La phase d'initialisation correspondant au chargement des jetons est la suivante :

- Le reset de tous les C-elements du micropipeline est mis à 1 de manière à réinitialiser la structure.

- Le reset sur les C-elements ayant comme sortie req est remis à 0.

- Le set sur les C-elements ayant comme sortie req est mis à 1 dans les cellules micropipeline que l'on doit charger. Les signaux req correspondant passent à 1.

- Le reset sur tous les C-element ayant comme sortie ack est remis à 0. La

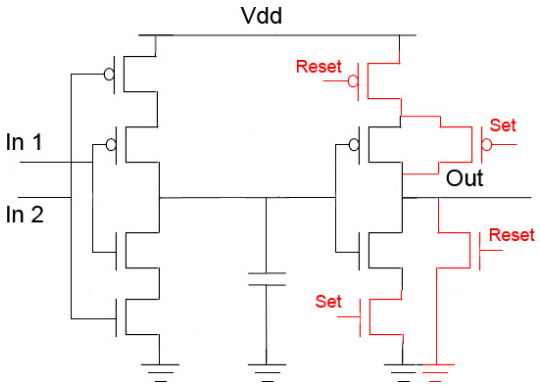


FIG. 7.17 – C-element dynamique avec set et reset prioritaire

propagation des jetons commence alors.

Lecture des jetons

Les opérations de remise à 0 et de remplissage du micropipeline par des jetons ayant été définies et leur structure correspondante proposée, il reste à présent à définir le mode de lecture des jetons. L'opérateur compteur de jetons dans la racine a été introduit à la partie 7.2.2 en précisant que ce n'est qu'un artefact destiné à introduire le modèle de la propagation de type transmission de jetons, mais qu'il n'est absolument pas destiné à être implanté. Un tel opérateur de comptage nécessite de prendre en compte la dimension temporelle des jetons. Comme nous l'avons vu pour l'initialisation des jetons, il est plus simple d'utiliser les propriétés spatiales de ceux-ci. Pour cela, il est nécessaire de conserver et d'arrêter la propagation des jetons dans l'arbre, afin d'identifier statiquement et de manière synchrone où ils sont localisés. L'arrêt de la propagation des jetons et leur conservation dans le réseau sont assurés par la mise à 0 de la sortie *req* placée dans la racine. Cette mise à 0 est effectuée en activant le *reset* du second C-element de la cellule. Ce faisant, la sortie ack_{out} ne peut plus changer d'état, et les jetons sont arrêtés dans leur propagation par le jeton situé immédiatement précédemment. Tous les jetons initialement présents dans le réseau de propagation sont donc agglutinés près de la racine dans leur forme la plus compacte possible, à savoir un jeton par cellule micropipeline. Le chronogramme correspondant à une telle propagation est présenté à la figure 7.18. On constate sur le chronogramme que les jetons sont accolés les uns aux autres au plus près de la racine dans l'état stable final. Cette adjacence des jetons après convergence est exploitable pour éliminer des paires de jetons, donc sans changer la parité du nombre total de jetons, opération utile au calcul de la somme.

Pour savoir si un jeton est présent ou non dans une cellule de micropipeline, il faut lire la valeur logique présente sur *ack* et sur *req*. La présence d'un jeton dans la cellule n correspond à la combinaison logique :

$$Jeton_n = \overline{ack_n}.req_n$$

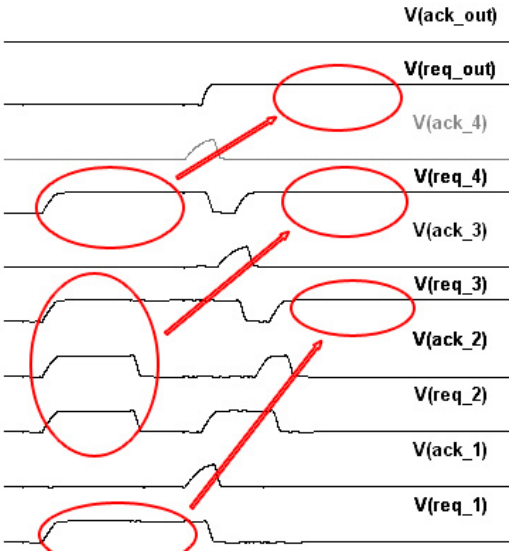


FIG. 7.18 – Propagation de 3 jetons arrêtés par la racine

Les valeurs de *ack* et de *req* peuvent être lues simplement de manière synchrone puisque, après propagation, les jetons sont dans une position stable.

7.3.4 Exemple de propagation sur un réseau de type arbre couvrant

Nous illustrons à présent la propagation de jetons sur un exemple simple d'arbre orienté présenté à la figure 7.19. Le chronogramme associé à cette propagation en

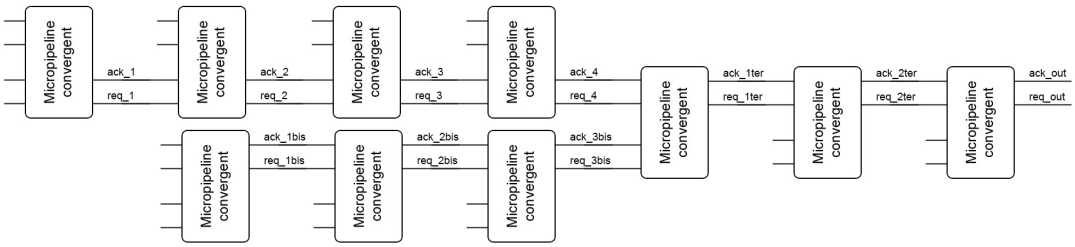


FIG. 7.19 – Exemple de réseau de propagation de type arbre couvrant

direction de la racine (non bloquant dans le cas présenté, les jetons passent) est présenté à la figure 7.20. Dans cette figure, les 3 jetons initialement placés dans le réseau sont signalés par 3 ellipses rouge alignées verticalement. La phase d'initialisation est représentée, avec les deux signaux de *reset* actifs, puis la désactivation d'un des *reset*, l'activation du *set* dans les 3 cellules à charger du micropipeline, puis la libération de la propagation des jetons par la remise à 0 du dernier *reset*. Les deux flèches rouges mettent en évidence l'action du micropipeline convergent qui permet de réunir les jetons provenant des deux réseaux de micropipelines amonts :

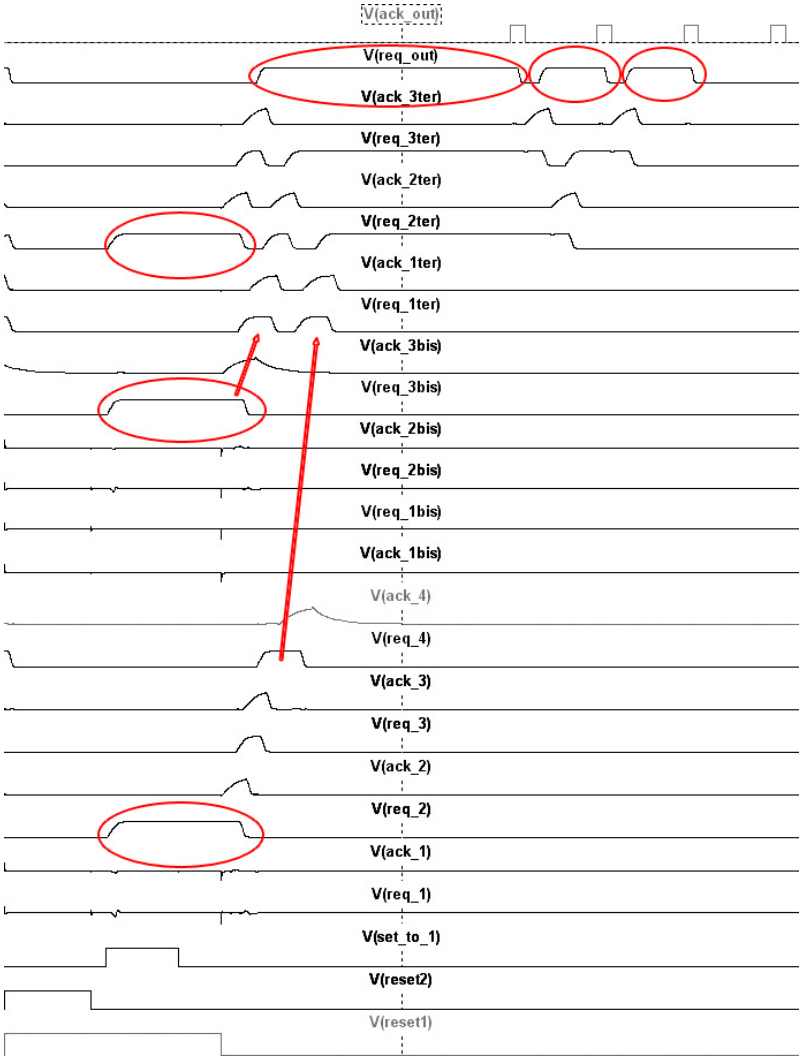


FIG. 7.20 – Propagation de 3 jetons dans un arbre orienté

le jeton présent sur req_4 et celui présent sur req_{3bis} sont mis en série sur req_{1ter}. Les jetons sont transmis vers l'extérieur de l'arbre grâce à l'envoi d'un signal ack_{out} destiné à simuler les accusés de réception envoyé par une cellule micropipeline placée en aval de l'arbre.

7.3.5 Coût d'implantation

Le coût d'implantation d'un micropipeline convergent à 2 entrées permettant d'implanter la primitive de propagation de jetons, et aussi d'initialiser le réseau et de le charger, est réduit. Il nécessite un C-element doté d'une fonction *reset* ayant un coût de 8 transistors, et d'un C-element doté d'une fonction *set* et d'une fonction *reset* ayant un coût de 10 transistors. Ces éléments ont été décrits précédemment. A ces 18 transistors s'ajoutent les 8 de l'arbitre (4 par porte *Non-Et*), les 4 de la mutuelle exclusion, les 6 de la porte *OU* ainsi que les 2 transistors du multi-

plexeur. On arrive donc à un coût total de 38 transistors dans le chemin asynchrone de chaque processeur élémentaire. Ce coût, comparativement aux autres méthodes présentées [DMM95] [KKI04], est très faible.

L'étude des opérateurs et structures électroniques permettant d'implanter le modèle des micropipelines associatifs a été menée en considérant 2 entrées par opérateurs. Nous montrerons en section 8.1 que des opérateurs à 2 entrées suffisent à établir une structure d'arbre couvrant sur une région de forme et de taille quelconque en 4-connexité.

7.3.6 Micropipeline convergent à 4 entrées

Afin de justifier par avance nos efforts ultérieurs pour réduire à 2 le nombre d'entrées des opérateurs situés dans le chemin asynchrone, nous calculons ici le coût de la structure du micropipeline convergent à 4 entrées, permettant d'implanter de manière simple le modèle des micropipelines associatifs en 4-connexité. La structure de l'élément de propagation ne change pas et coûte toujours 18 transistors. Le reste du circuit assurant la convergence des jetons doit quant à lui permettre la gestion des 4 entrées. On aboutit à la structure présentée à la figure 7.21. L'arbitre

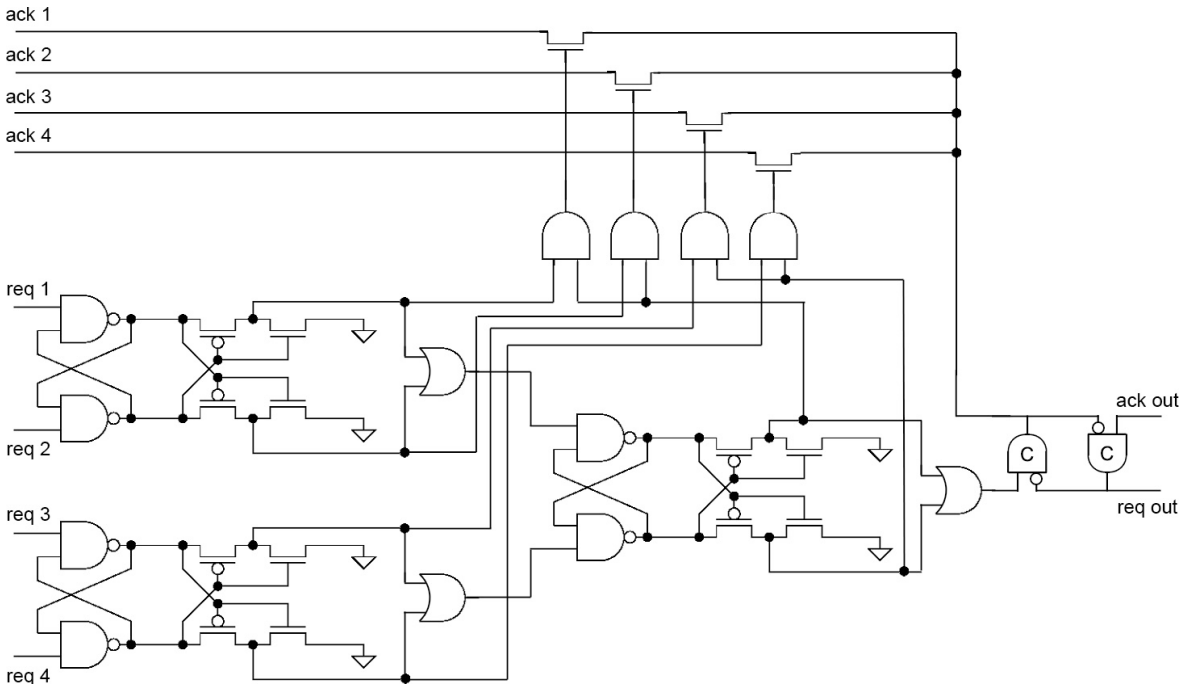


FIG. 7.21 – Micropipeline convergent à 4 entrées

à 4 entrées, la mutuelle exclusion et la propagation de la sortie sont assurés par une cascade de 3 cellules à 2 entrées. Le pilotage du multiplexeur est quant à lui assuré par 4 transistors pilotés par 4 porte *Et*. Le coût total de cette structure est de 98 transistors. Cette structure permet de réduire de plus de moitié le coût de l'asynchronisme présent dans la *Maille Associative d'Orsay*, tout en en conservant

les fonctionnalités essentielles en terme d'implantation du modèle des réseaux associatifs. Pour une implantation dans une rétine artificielle, le coût est toutefois toujours élevé comparé au nombre de transistors disponibles dans un processeur élémentaire (de l'ordre de 200 transistors). Il serait vraiment préférable d'utiliser des micropipelines convergents à 2 entrées seulement. Cela permettrait d'économiser de l'ordre de 60% de ressources matérielles. De plus, le temps de propagation du signal dans le chemin asynchrone serait 35% plus rapide.

7.4 Etude de la primitive somme régionale associée aux micropipelines associatifs

L'analyse des solutions existantes permettant d'implanter partiellement le modèle des réseaux associatifs dans le but d'effectuer des opérations régionales de traitement d'images nous a conduit à proposer l'architecture des micropipelines associatifs. Cette architecture permet d'utiliser une primitive de propagation de jetons sans perte sur un arbre couvrant. Elle envoie ainsi dans la racine l'information nécessaire au calcul de la somme régionale. Pour cela, nous avons précédemment évoqué un compteur de jetons installé dans la racine. Mais il a été introduit dans un but purement didactique. Un tel circuit, forcément coûteux devrait être disponible dans chaque processeur. Cela est exclu. De plus, dans l'absolu, il n'est certainement pas simple de compter des jetons passant à toute allure.

Il est préférable d'utiliser pour le calcul de la somme et le comptage des jetons le codage de ceux-ci sous forme spatiale, ce qui peut être fait, en interdisant aux jetons de sortir de l'arbre couvrant par la racine. De cette manière, tous les jetons présents dans l'arbre s'agglutinent les uns aux autres en partant de la racine. Cette juxtaposition de jetons permet ensuite d'effectuer des opérations synchrones sur ces jetons. Une de ces opérations est l'élimination de paires de jetons.

L'élimination de paires de jetons ne change pas la parité du nombre total de jetons. Cette information de parité est importante car elle représente le bit de poids faible du nombre total de jetons présents dans l'arbre couvrant. Cette idée ouvre la voie vers le calcul de la somme régionale et nous allons l'exploiter à fond.

7.4.1 Calcul du bit de poids faible du nombre de jetons présents dans un arbre couvrant

Une somme calculée de manière bit série est équivalente à une succession d'extractions de parité. Le bit de poids faible d'une somme est la parité de cette somme. Le calcul du bit de poids faible du nombre de jetons placés dans un arbre peut donc être réalisé en supprimant les jetons par paires jusqu'à ce qu'il n'y en ait plus aucun ou plus qu'un seul. On peut réaliser ces suppressions de paires de jetons de manière synchrone avec un coût algorithmique acceptable à condition que ces jetons soient

placés à proximité les uns des autres. Ce rassemblement des jetons est rendu possible par la primitive de propagation vers la racine des jetons. La figure 7.22 montre des jetons en cours de propagation vers la racine d'un arbre couvrant (l'arbre est représenté par les flèches rouges orientées vers la racine). La figure 7.23 montre la position des jetons sur l'arbre une fois la convergence terminée.

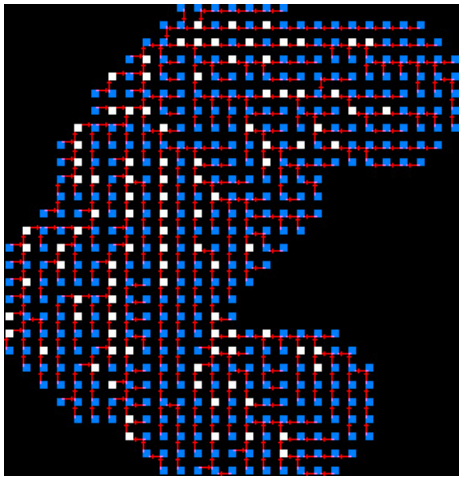


FIG. 7.22 – Jetons en cours de propagation

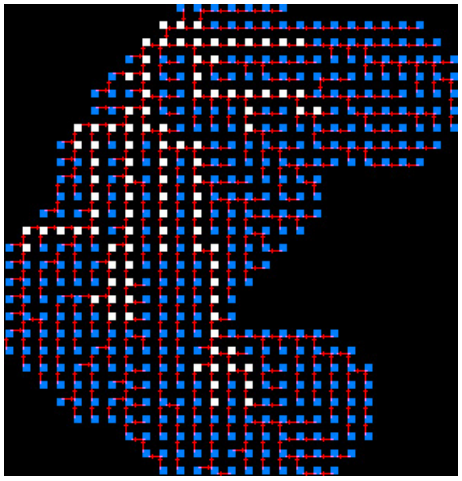


FIG. 7.23 – Jetons rassemblés près de la racine

Une fois le rassemblement des jetons effectué (fig. 7.25), il est simple d'éliminer des paires de jetons adjacents de manière synchrone. Il suffit pour cela de rechercher dans chacun des ensembles de jetons présents dans les régions de l'image des ensemble de deux pixels adjacents. La figure 7.24 présente un moyen d'effectuer cette élimination en 4 opérations synchrones. On recherche des configurations de paires de pixels formant un pavage de la rétine (hormis sur les bords). Les 4 configurations utilisées sont analogues par rotation d'angle $\frac{\pi}{2}$. A chaque rotation un pixel de la paire reste fixe (il est identifié par une croix rouge sur la figure 7.24) : nous l'appelons pivot. Les pivots sont donc disposés en damier sur la rétine, de manière à ce qu'en faisant tourner chaque paire de $\frac{\pi}{2}$, on obtienne toujours un pavage du plan.
La figure 7.24 présente une séquence aboutissant à l'élimination des paires de jetons dans des régions. Les paires éliminées à chaque étape et correspondant à chacune des configurations étudiées sont entourées en rouge, le pixel pivot est matérialisé par une croix. Compte tenu de l'irrégularité de la forme des régions sur lesquelles l'élimination des jetons s'effectue, il peut rester à l'issu de l'élimination des paires de jetons adjacents quelques jetons isolés sur l'arbre (fig. 7.24 et fig. 7.26).

On cherche à avoir au maximum un jeton dans l'arbre, celui-ci étant localisé dans la racine, il est nécessaire d'éliminer par paires les jetons isolés restants. Pour ce faire, il suffit de propager à nouveau les jetons restants dans l'arbre vers la racine. Une fois regroupés, il est possible de les éliminer à nouveau par paires. Ce processus

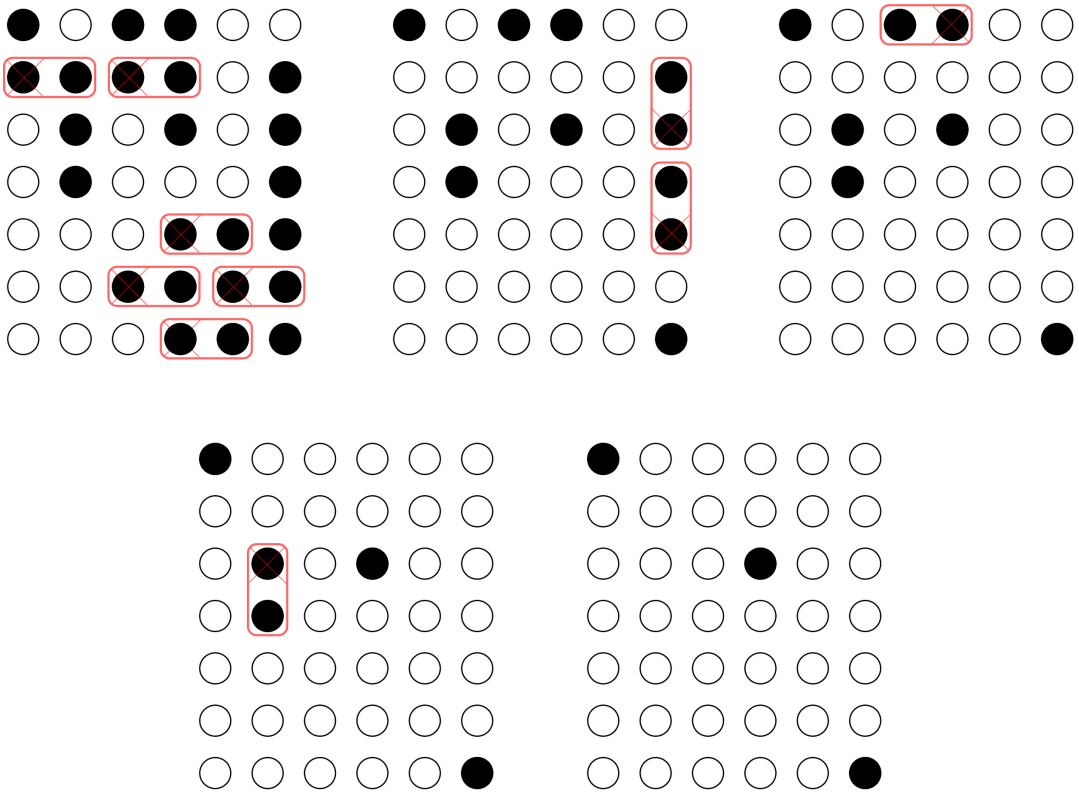


FIG. 7.24 – Elimination des paires de jetons en 4 opérations synchrones

est itéré jusqu'à ce qu'il reste dans l'arbre et dans la racine au maximum un jeton. Pour calculer le bit de poids faible du nombre de jetons présents dans un arbre, il suffit donc d'itérer en alternance des phases de propagation de jetons vers la racine puis des éliminations de paires de jetons adjacents.

Evaluation de l'efficacité des éliminations de paires de jetons

Des tests menés sur des régions de taille et de forme variable tirées aléatoirement ont montré que le nombre total d'itérations nécessaires pour n'avoir plus qu'un ou zéro jeton dans l'arbre est très faible (de l'ordre de 4 ou 5 itérations au maximum pour de grandes régions). Cette efficacité est due au fait que tous les jetons (sauf 1 dans les cas défavorables) sont éliminés dans les alignements constitués de plus de 2 jetons. Un alignement de 2 jetons est éliminé dans tous les cas à moins qu'il ne soit à l'extrémité d'une chaîne. Finalement la configuration la plus défavorable est une chaîne constituée de portions de fragments de longueur égale à 3 pixels. L'élimination des paires de jetons dans cette configuration est présentée à la figure 7.27. On constate que l'élimination permet de supprimer 75% des jetons présents dans un tel arbre. Pour éliminer $2^{10} = 1024$ jetons, il faut donc effectuer 5 cycles de propagation-élimination, ce qui est assez faible et rejoint les résultats obtenus de manière empirique. Il est à noter qu'une telle configuration défavorable n'a que très peu de chances de se produire en pratique. Les pixels étant rassemblés vers

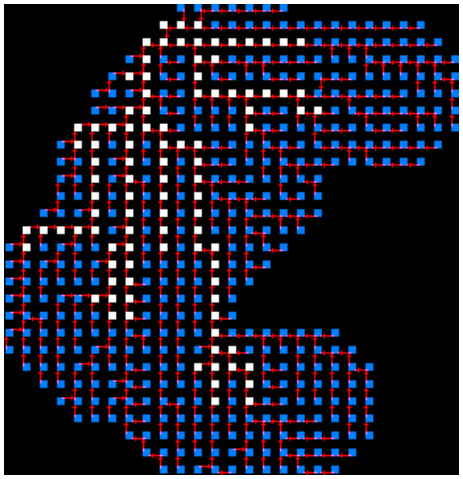


FIG. 7.25 – Jetons rassemblés près de la racine avant élimination des paires

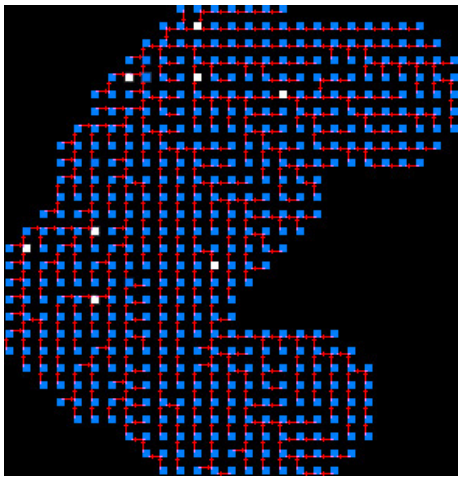


FIG. 7.26 – Jetons après élimination des paires

la racine de l'arbre couvrant, ils forment le plus souvent des blocs assez compacts dont l'élimination se fait de manière très performante (fig. 7.25 et 7.26).

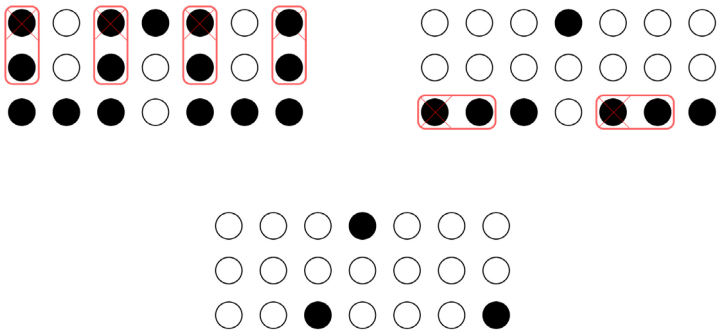


FIG. 7.27 – Elimination des paires de jetons dans une configuration défavorable

7.4.2 Calcul du nombre de jetons présents dans un arbre couvrant

La partie précédente décrit le calcul du bit de poids faible d'une somme. Écrivons le nombre de jetons N présents dans un arbre couvrant sous la forme :

$$N = a_0 + a_1 2 + a_2 2^2 + ... + a_n 2^n$$

a_0 est calculé grâce à l'algorithme présenté précédemment. Le nombre P de paires de jetons ayant été éliminées au cours du calcul du bit de poids faible a_0 est le

suivant :

$$P = \frac{N - a_0}{2}$$
$$= a_1 + a_2 2 + ... + a_n 2^{n-1}$$

Pour calculer a_1, il suffit donc de placer sur le réseau un nombre de jetons égal au nombre de paires de jetons supprimées lors du calcul de a_0. L'opération d'élimination de chacune des paires de jetons décrite précédemment s'accompagne donc du stockage local d'un jeton de poids supérieur utilisé pour le calcul du bit suivant de la somme.
Comme nous l'avons vu, le calcul d'un bit de la somme peut demander plusieurs propagations. Dans certains processeurs élémentaires, plusieurs paires de jetons seront donc éliminées au cours du calcul d'un même bit de la somme, générant plusieurs jetons de poids supérieur. Étant donné qu'il n'est possible d'insérer qu'un seul jeton par processeur élémentaire sur le réseau, ces jetons seront additionnés localement de manière synchrone, générant à cette occasion des jetons de niveau encore supérieur. Une telle opération s'apparente à une gestion des retenues. Elle est effectuée de la même manière dans la partie synchrone de la Maille Associative d'Orsay, à la différence près que les données additionnées sont les 3 retenues générées par l'additionneur à 9 entrées.

L'obtention de chacun des bits de la somme est donc réalisée par itération du calcul du bit de poids faible d'un ensemble de jetons, cet ensemble étant constitué des jetons générés lors de l'élimination des paires de jetons effectuée lors des itérations précédentes.

7.4.3 Calcul d'une somme sur une région

Le calcul d'une somme sur une région rejoint l'algorithme de calcul du nombre de jetons présents dans un arbre couvrant. Dans le cas du calcul du nombre de jetons présents dans un arbre couvrant, la valeur à additionner est 1. De fait, il suffit de matérialiser par un jeton cette valeur au départ et d'appliquer le processus présenté précédemment. Dans le cas où la donnée locale à additionner n'est pas un bit mais un nombre, il est nécessaire de la décomposer sous forme binaire. Les valeurs binaires locales des bits de cette décomposition seront injectées dans le réseau lors de l'itération correspondant à leur poids. Avant son injection dans le réseau, la valeur binaire locale doit toutefois être additionnée aux éventuelles retenues générées par les éliminations de paires ayant eu lieu lors du calcul des bits précédents de la somme (comme nous l'avons vu à la sous-section précédente).

7.4.4 Performances

Le coût temporel de la somme régionale calculée à l'aide des micropipelines associatifs est donc proportionnel au nombre de bits présents de la somme, le calcul de chaque bit de la somme étant un calcul de bit de poids faible d'un nombre de jetons

placé sur un arbre. Le calcul de chacun de ces bits nécessite un nombre limité de propagations, rarement supérieur à 5, en alternance avec des opérations synchrones du type de celles que l'ont peut trouver dans la Maille Associative d'Orsay pour la gestion des retenues. Il est important de savoir si le calcul d'un bit de la somme régionale a été terminé sur toutes les régions de l'image (le calcul s'effectuant en parallèle sur toutes les régions). Pour cela il est possible d'utiliser un *OU* régional sur l'image complète. Si le calcul d'un bit de la somme n'est pas terminé sur l'un des arbres couvrants sur une des régions, alors la racine de celui-ci place un 1 dans le réseau de calcul du *OU* régional qui a donc pour résultat 1. Si le calcul du bit de la somme régionale est terminé dans chacune des régions, le résultat du *OU* global est 0. Il est à noter qu'une telle opération n'est pas coûteuse en temps, en effet nous verrons plus tard que le coût des opérations de propagation (même sur une grande région) est très faible.

Finalement, le nombre de propagations nécessaires au calcul d'une somme sur une région est donc $\mathcal{O}(n)$, avec n le nombre de bits de la somme régionale ayant la plus grande valeur dans toute l'image. Cette complexité temporelle répond aux exigences que l'on s'est fixées lors de l'analyse des solutions bit-série permettant le calcul des sommes régionales. De manière triviale, on peut vérifier que l'opérateur *OU* peut être réalisé en une propagation de jetons vers la racine.

Finalement, les micropipelines associatifs, associés au protocole de transmission par jeton et à la structure de micropipeline convergent, permettent pour un coût matériel réduit d'implanter certaines primitives utiles en traitement d'image du modèle des réseaux associatifs. Le coût temporel de ces primitives est par ailleurs du même ordre de grandeur que celui mis en jeu dans la Maille Associative d'Orsay.

8

Coût d'implantation matérielle et considérations topologiques

8.1 Connexité du réseau et coût matériel

L'examen des réalisations permettant d'effectuer des opérations régionales distribuées permet de constater que la complexité des opérateurs mis en jeu dans chaque processeur élémentaire croît avec le nombre d'entrées de ces opérateurs.

Prenons par exemple l'exemple de l'additionneur combinatoire présent dans l'additionneur bit-série linéaire [KKI04]. Il possède 3 entrées, destinées à additionner une donnée locale, une donnée provenant du processeur précédent et une retenue. Il est construit à l'aide d'un *full adder*. L'additionneur à 9 entrées utilisé dans la maille associative est quant à lui constitué de 5 Full Adder (FA) et 2 Half Adder (HA). Le passage de 3 à 9 entrées se traduit donc par une multiplication par environ 6 du nombre de transistors utilisés. Il en va de même pour l'arbitre.
Concernant l'implantation du modèle des micropipelines associatifs, nous avons pu constater que pour réaliser un micropipeline convergent à 4 entrées, il faut dépenser 98 transistors alors que pour réaliser un micropipeline à 2 entrées, 38 transistors suffisent. Ce résultat est encore plus flagrant si l'on soustrait à ces deux valeurs le nombre de transistors utilisés (18) pour implanter la cellule de propagation à base de C-éléments qui est la même dans les deux cas.
Le coût d'implementation augmente donc plus vite que le nombre d'entrées des opérateurs. Il est donc préférable d'utiliser des opérateurs avec un faible nombre d'entrées, quitte à avoir un réseau de communication plus complexe. Toutefois, la fonctionnalité du réseau doit être respectée, et sa robustesse aux pannes doit être correcte.

Afin de déterminer le nombre d'entrées minimal nécessaire dans chaque opérateur, commençons par déterminer sa borne inférieure. Pour faire converger n jetons provenant de n processeurs élémentaires, il faut au pire des cas $n-1$ micropipelines convergents à 2 entrées. La borne inférieure du nombre minimal d'entrées nécessaire

dans chaque opérateur est donc 2 entrées par opérateur.

8.2 Réduction du coût d'implantation grâce à l'augmentation de la connexité matérielle du réseau

Dans cette partie, une solution permettant d'implanter un réseau de connexion en 4-connexité uniquement à l'aide d'opérateurs à deux entrées est présentée. Pour cela nous distinguons la connexité fonctionnelle du réseau considéré et la connexité matérielle du réseau de connexion. La connexité fonctionnelle est définie au sens du voisinage mathématique. La connexité matérielle correspond aux connexions physiques qu'il est matériellement possible d'établir sur le réseau. La connexité matérielle est donc supérieure ou égale à la connexité fonctionnelle.

Dans beaucoup de circuits de type maille, la connexité matérielle est égale à la connexité fonctionnelle du réseau considéré. La maille Associative d'Orsay est un exemple de ce type d'implantation en 8-connexité fonctionnelle et matérielle. Cependant, à condition de respecter la connexité fonctionnelle lors de l'établissement des connexions, une augmentation de la connexité matérielle peut conduire à créer des réseaux de connexion ayant des propriétés intéressantes.
Considérons le cas d'une croix (fig. 8.1 et 8.2) recouverte par un arbre couvrant en 4-connexité fonctionnelle. La figure 8.1 utilise un réseau de connexion en 4-connexité matérielle, des opérateurs à 4 entrées sont nécessaires pour établir les arbres couvrants. La figure 8.2 utilise un réseau de connexion en 6-connexité matérielle, il est possible sur cet exemple de n'utiliser que des opérateurs à deux entrées pour établir un arbre couvrant. L'augmentation de la connexité matérielle permet donc de

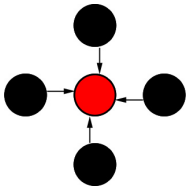


FIG. 8.1 – Arbre couvrant en 4-connexité fonctionnelle et 4-connexité matérielle

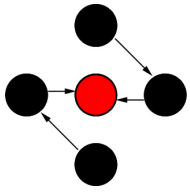


FIG. 8.2 – Arbre couvrant en 4-connexité fonctionnelle et 6-connexité matérielle

diminuer le nombre d'entrées des opérateurs assurant la convergence dans le réseau.

Cependant, le fait qu'un arbre couvrant puisse être placé sur le graphe n'est pas suffisant. Il faut également pouvoir construire cet arbre couvrant. Dans le cas d'un réseau ayant même connexité matérielle et fonctionnelle, avec des opérateurs ayant un nombre d'entrées égal à la valeur de cette connexité, il suffit d'activer toutes les connexions du réseau puis de laisser se propager un marqueur depuis un point racine unique comme nous l'avons déjà vu pour construire l'arbre couvrant de manière asynchrone. Si l'on souhaite utiliser des opérateurs ayant un nombre d'entrées

limité, et inférieur au degré de connexité fonctionnelle et matérielle, alors il est impossible d'établir toutes les connexions du réseau à l'initialisation. Il faut donc choisir parmi les connexions possibles au départ de manière à établir un réseau sur lequel il sera possible de construire un arbre couvrant.
Un tel réseau doit avoir une propriété importante, celle de former une composante fortement connexe, ce qui signifie que tout point de ce réseau est connecté à tout autre point du réseau par une connexion orientée.

La section suivante présente une méthode permettant de construire une telle composante fortement connexe en 4-connexité fonctionnelle, sur un réseau en 6-connexité matérielle, en utilisant uniquement des opérateurs convergents à 2 entrées [GBM05b] [GB05a].

8.2.1 Arbres couvrants en 4-connexité fonctionnelle, 6-connexité matérielle, construits à l'aide d'opérateurs convergents à 2 entrées.

Considérons le réseau de la figure 8.3, il représente des composantes fortement connexes en 4-connexité fonctionnelle et matérielle, avec des opérateurs convergents à 4 entrées. On constate qu'après établissement de l'arbre couvrant (fig. 8.4), certains points du graphe sont connectés en entrée à 4 voisins. Il est donc impossible de réduire le nombre d'entrées des opérateurs convergents.

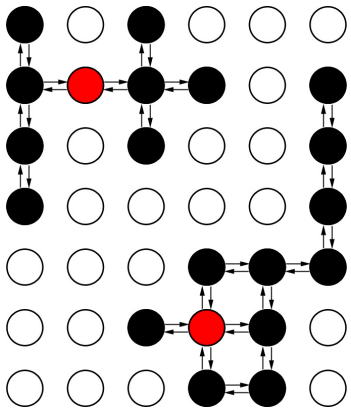


FIG. 8.3 – Composantes fortement connexes en 4-connexité fonctionnelle et matérielle

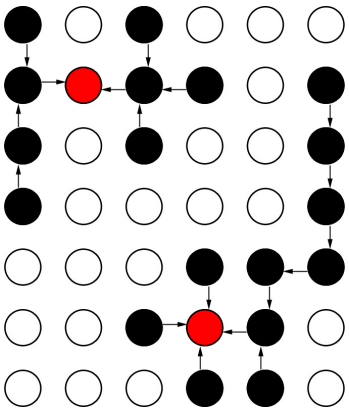


FIG. 8.4 – Arbre couvrant en 4-connexité fonctionnelle et matérielle

Considérons à présent le réseau de la figure 8.5. Il représente des composantes fortement connexes en 4-connexité fonctionnelle, 6-connexité matérielle, et n'utilisant que des opérateurs convergents à 2 entrées. Un arbre couvrant associé est proposé à la figure 8.6. Les composantes fortement connexes sont obtenues en remplaçant certaines chaînes de 2 connexions verticales et horizontales par une connexion diagonale, utilisant la connexion physique rajoutée au réseau. Ce faisant, la connexité

fonctionnelle du réseau ne change pas.
L'ajout de 2 possibilités de connexions supplémentaires dans le réseau permet donc

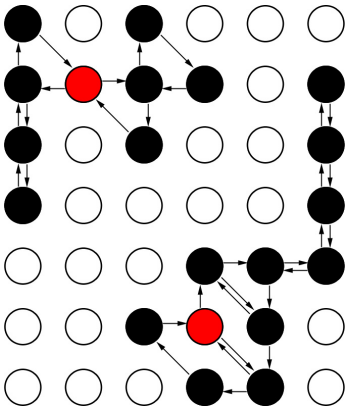


FIG. 8.5 – Composantes fortement connexes en 4-connexité fonctionnelle et 6 connexité matérielle

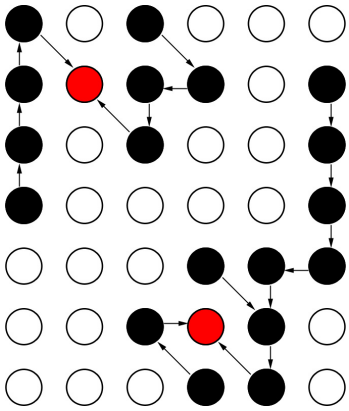


FIG. 8.6 – Arbre couvrant en 4-connexité fonctionnelle et 6-connexité matérielle

de réduire le nombre d'entrées des opérateurs convergents au minimum possible, à savoir 2 entrées. Dans la partie suivante, un algorithme permettant d'établir la composante fortement connexe initiale est présenté.

Algorithme d'établissement de la composante fortement connexe

L'algorithme utilisé pour réaliser les connexions est de type synchrone. Les connexions sont établies en étudiant les 6 configurations locales de pixels suivantes (fig.8.7). Le pixel entouré d'un double cercle est celui dont on souhaite raccorder l'entrée.
L'idée sous-jacente à l'utilisation de ces 6 motifs est de connecter le périmètre de la région à l'aide d'un anneau de connexion orienté dans le sens des aiguilles d'une montre (fig. 8.7, configurations (a), (b), (d), et (e)), et d'établir à l'intérieur de cet

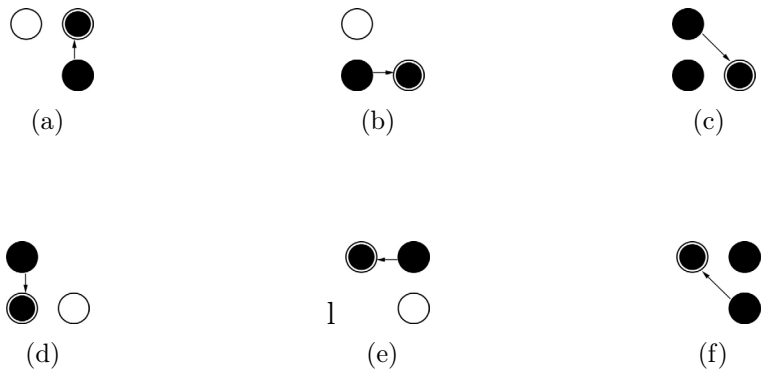


FIG. 8.7 – Configurations locales permettant d'initialiser les connexions des composantes fortement connexes en 4-connexité fonctionnelle et 6-connexité matérielle

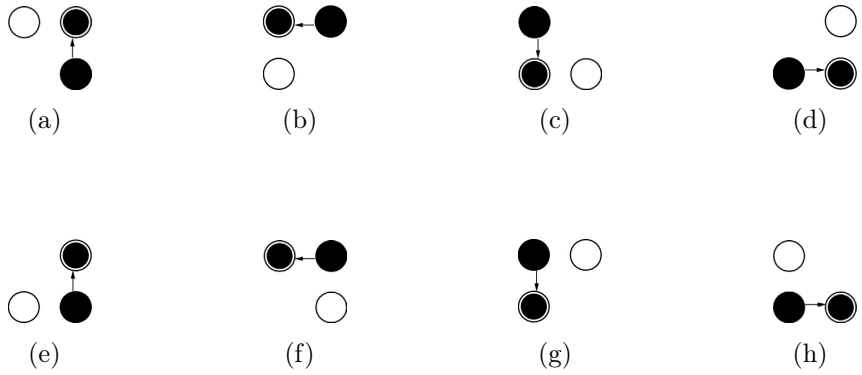


FIG. 8.8 – Démonstration de la validité de l'algorithme d'initialisation de la composante fortement connexe : configurations locales permettant d'établir un anneau de connexions périphériques orientées dans le sens des aiguilles d'une montre.

anneau des connexions entre les pixels et le bord de l'anneau selon une direction privilégiée (fig. 8.7, configurations (c) et (f)).
L'examen de chacune des 6 combinaisons permet d'établir la composante fortement connexe initiale de manière très performante. Le fonctionnement SIMD synchrone permet en effet d'effectuer ce test sur tous les pixels de l'image en même temps. L'algorithme proposé permet donc de réduire le nombre d'entrées des opérateurs pour un coût algorithmique négligeable.

Preuve de la validité de l'algorithme

La validité de l'algorithme d'initialisation de la composante fortement connexe proposé peut être démontrée. La démonstration proposée se compose de deux étapes. La première est la détermination des configurations à tester pour installer une composante fortement connexe. La deuxième est la vérification que cette construction conduit à une composante fortement connexe implantable uniquement à l'aide d'opérateurs à 2 entrées. Le principe de cet algorithme est d'établir un réseau de connexion en anneau orienté dans une seule direction sur la périphérie de l'ensemble de points considérés. Il est également nécessaire de connecter les pixels situés hors de la périphérie par des connexions bidirectionnelles. Les configurations locales à tester pour construire un anneau de connexions périphériques orienté dans le sens des aiguilles d'une montre sont les 8 configurations de la figure 8.8.
Compte tenu de l'ajout de connexions physiques programmables sur l'une des deux diagonales, il est possible de remplacer certains couples de combinaisons proposées à la figure 8.8 par des combinaisons permettant d'établir des connexions en diagonale. Ainsi les configurations (e) et (b) de la figure 8.8 peuvent être remplacées par la configuration (a) de la figure 8.9. Les configurations (g) et (d) de la figure 8.8 peuvent être remplacées par la configuration (b) de la figure 8.9.
Les pixels situés à l'intérieur de l'anneau établi doivent également être connectés. Nous choisissons (arbitrairement) pour cela d'établir des connexions bidirectionnelles selon la diagonale correspondant à la connexion physique rajoutée au réseau.

FIG. 8.9 – Démonstration de la validité de l'algorithme d'initialisation de la composante fortement connexe : configurations locales permettant d'établir des connexions diagonales périphériques.

FIG. 8.10 – Démonstration de la validité de l'algorithme d'initialisation de la composante fortement connexe : configurations locales permettant d'établir des connexions diagonales intérieures.

Pour cela il suffit d'examiner les configurations présentées à la figure 8.10.
Finalement il reste 8 combinaisons à étudier, 6 permettant d'établir un anneau en périphérie de la région considérée, et 2 permettant de relier chacun des points intérieurs à cette région à l'anneau périphérique par une connexion bidirectionnelle. Le réseau de connexion ainsi formé est une composante fortement connexe car un anneau est une composante fortement connexe, et des pixels liés par une connexion bidirectionnelle à cet anneau sont ajoutés à la composante fortement connexe. Il est à noter que les combinaisons (a) de la figure 8.9 et (a) de la figure 8.10 peuvent être remplacées par la figure (a) de la figure 8.11. Il en va de même pour les configurations (b) de la figure 8.9 et (b) de la figure 8.10 qui peuvent être remplacées par le configuration (b) de la figure 8.11.
Ainsi il reste finalement uniquement 6 combinaisons simples à étudier, les combinaisons présentées à la figure 8.7.

Par construction nous avons donc montré que l'étude des 6 configurations proposées permet de placer une composante fortement connexe sur la région considérée. Pour valider l'algorithme, il reste toutefois à vérifier qu'il est impossible que plus de 2 connexions entrent dans un même processeur élémentaire.

FIG. 8.11 – Démonstration de la validité de l'algorithme d'initialisation de la composante fortement connexe : configurations locales permettant de réduire le nombre de configurations à étudier.

Si l'on considère les configurations locales présentées à la figure 8.7, on peut constater que la configuration (a) est exclusive des configurations (b) et (c). De même, (b) est exclusive de (a) et (c), et (c) est exclusive de (a) et (b). Par symétrie, nous avons le même résultat sur (d), (e) et (f). En conséquence, à un voisinage donné de points d'un pixel correspond au mieux 2 des 6 configurations présentées, et donc au maximum 2 connexions entrantes sont établies. Le réseau formé est donc une composante connexe et cette composante connexe peut être construite uniquement à l'aide d'opérateurs convergents à 2 entrées.

8.2.2 Réduction des ressources de calcul nécessaires

Comme nous l'avons déjà indiqué à la partie 7.3.6, le coût de la structure du micropipeline convergent à 2 entrées est de 38 transistors, comparé au coût de la structure du micropipeline convergent à 4 entrées dont le coût est de 98 transistors. La réduction de ressources propres à la structure de micropipeline est donc importante. Toutefois cette comparaison exclut le coût des connexions programmables. Celles-ci nécessitent en effet 1 transistor supplémentaire par connexion entrante (pour le signal *req*) dans le cas du micropipeline à 4 entrées. Dans le cas du micropipeline à 2 entrées la sélection est plus difficile à effectuer, car il faut pouvoir orienter deux des quatre connexions entrantes vers les deux entrées du micropipeline convergent. Pour cela, chacune des deux entrées du micropipeline doit pouvoir être connectée à 3 entrées (fig. 8.12).

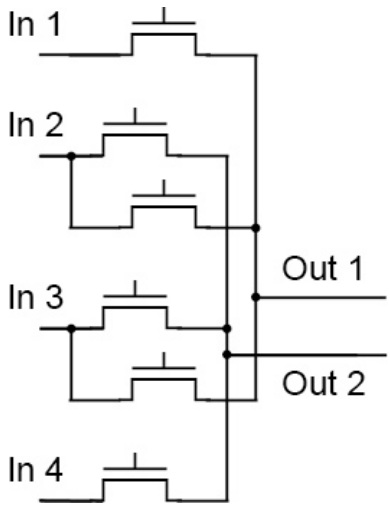


FIG. 8.12 – Connexions programmables en entrée du micropipeline à 2 entrées.

Il faut donc 6 transistors (on n'utilise que des transistors NMOS) pour orienter les bons signaux aux entrées. De plus, à l'inverse du cas du micropipeline à 4 entrées où les connexions du signal *ack* sont faites grâce au démultiplexeur intégré à la structure, il faut également router ici les sorties, ce qui coûte de la même manière 6 autres transistors. Le coût additionnel à ajouter pour le routage est donc de

$6 - 4 + 6 = 8$ transistors. Finalement le coût de la structure du micropipeline à 2 entrées est de 46 transistors, comparé au coût de la structure du micropipeline à 4 entrées qui est de 100 transistors. L'économie réelle réalisée est donc de plus de 50%.

8.2.3 Influence sur la robustesse du système

La robustesse d'un réseau associatif aux pannes se traduit par une faible variabilité du résultat obtenu lors d'une association lorsqu'un ou plusieurs processeurs élémentaires sont en panne. Nous avons déjà vu que dans les réseaux de type linéaire, la robustesse est faible, car en cas de panne tous les pixels situés en amont de la panne sont ignorés. Nous avons également vu à la partie 7.1.1 que les réseaux utilisant une procédure d'établissement asynchrone d'arbre couvrant ont une bonne robustesse aux pannes. Dans le cas d'un réseau utilisant uniquement des opérateurs de propagation à deux entrées, la robustesse est plus faible que dans le cas de l'utilisation d'opérateurs à 4 entrées, mais meilleure que dans le cas de l'utilisation d'un réseau linéaire. La robustesse du réseau établi de manière asynchrone repose sur la manière de construire un arbre couvrant. La propagation d'un jeton se fait uniquement dans les processeurs valides, en évitant les processeurs en panne. De fait l'arbre couvrant construit est constitué de pixels valides uniquement, sauf pour les feuilles qui peuvent éventuellement être constituées par des pixels en panne. Dans le cas de la construction d'un arbre couvrant à partir d'une composante fortement connexe constituée de processeurs à 2 entrées, l'arbre couvrant tend à se construire uniquement à partir de processeurs valides. Toutefois, le nombre de choix pour la propagation du jeton permettant la construction de l'arbre est limité à 2. De fait, si le réseau contient un nombre important de processeurs en panne, des parties du réseau risquent d'être isolées de la racine. La robustesse de ce type de réseau est donc bonne si le nombre de processeurs en panne est faible (dans ce cas, elle est même aussi bonne que pour un réseau utilisant des processeurs à 4 entrées ou plus), mais elle se dégrade fortement si le nombre de processeurs en panne devient important. Dans une rétine artificielle, le nombre de processeurs en panne est censé être extrêmement faible, ce qui ne pose pas problème.

8.2.4 Influence sur la vitesse des primitives

La vitesse des opérateurs asynchrones présents dans le pixel dépend du type et du nombre de composants électroniques traversés. Ainsi une diminution des ressources utilisées dans le chemin asynchrone d'un processeur élémentaire se traduit nécessairement par une diminution du temps de propagation des signaux dans ce chemin asynchrone. Lors des simulations effectuées (voir chapitre suivant), nous avons obtenu une réduction du temps de traversée d'un processeur de l'ordre de 40%. La réduction du nombre d'entrées des opérateurs consécutive à l'utilisation d'arbres couvrants en 4-connexité fonctionnelle et 6-connexité matérielle permet donc de diminuer le temps de calcul d'opérations régionales.

8.2.5 Arbres couvrants en 6-connexité fonctionnelle, 8-connexité matérielle, construits à l'aide d'opérateurs convergents à 2 entrées.

Les micropipelines convergents mis en jeu, dans le chemin asynchrone d'un réseau en 6-connexité fonctionnelle et matérielle, ont 6 entrées. Le coût des opérateurs asynchrones présents dans ce chemin est donc plus de trois fois supérieur au coût des opérateurs présents dans les micropipelines à deux entrées. Une composante fortement connexe et un arbre couvrant associés sont proposés aux figures 8.13 et 8.14. Pour envisager une implantation 6-connexe du modèle des micropipelines associatifs dans les rétines artificielles, il est donc nécessaire de réduire ce coût d'implantation. Pour cela, augmentons comme précédemment la connexité physique du réseau afin de diminuer le nombre nécessaire d'entrées des opérateurs.
Nous avons montré dans les sections précédentes que l'utilisation de la 6-connexité

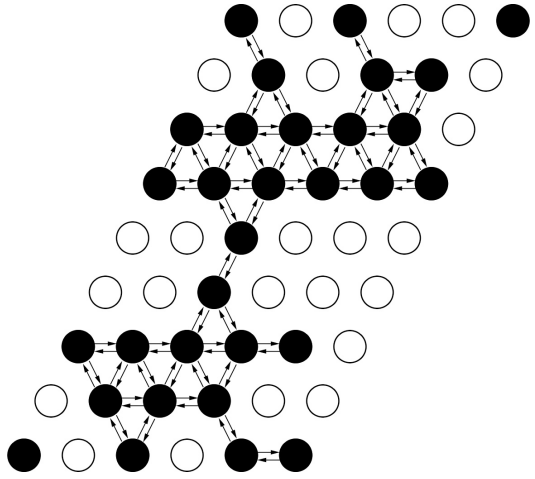


FIG. 8.13 – Composantes fortement connexes en 6-connexité fonctionnelle et matérielle

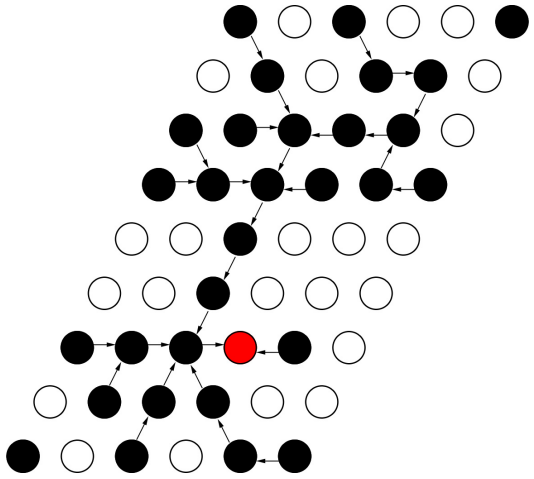


FIG. 8.14 – Arbre couvrant en 6-connexité fonctionnelle et matérielle

matérielle pour implanter la 4-connexité fonctionnelle se traduit par une réduction des ressources matérielles nécessaires à l'implantation du chemin asynchrone dans chaque processeur élémentaire ainsi que d'une réduction du temps de calcul des primitives régionales. Dans cette partie, une extension de ces résultats à l'utilisation de la 8-connexité matérielle pour implanter la 6-connexité fonctionnelle est présentée.

La méthode utilisée est exactement la même que pour la 4-connexité fonctionnelle. Il est nécessaire de construire de manière synchrone à l'initialisation une composante fortement connexe sur la région. La construction de cette composante fortement connexe comporte deux étapes : la construction d'un anneau de connexions périphériques orienté (qui forme une composante fortement connexe) et le rattachement des pixels non-périphériques à l'anneau établi grâce à des connexions

bidirectionnelles.

Prenons l'exemple du réseau de la figure 8.15. Il représente une composante fortement connexe en 6-connexité fonctionnelle, 8-connexité matérielle, et n'utilisant que des opérateurs convergents à 2 entrées. Un arbre couvrant associé est proposé à la figure 8.16.
L'ajout de 2 possibilités de connexions supplémentaires dans le réseau permet donc

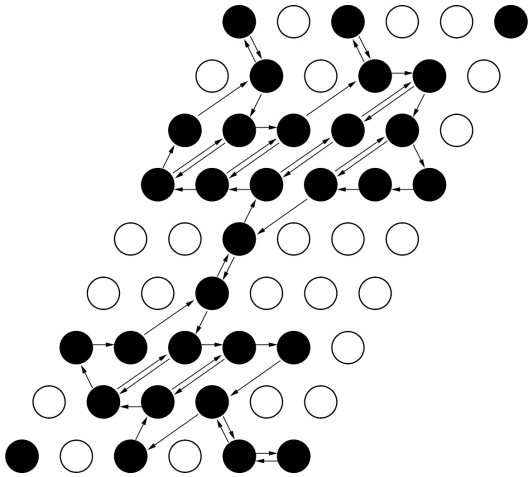


FIG. 8.15 – Composantes fortement connexes en 4-connexité fonctionnelle et 6 connexité matérielle

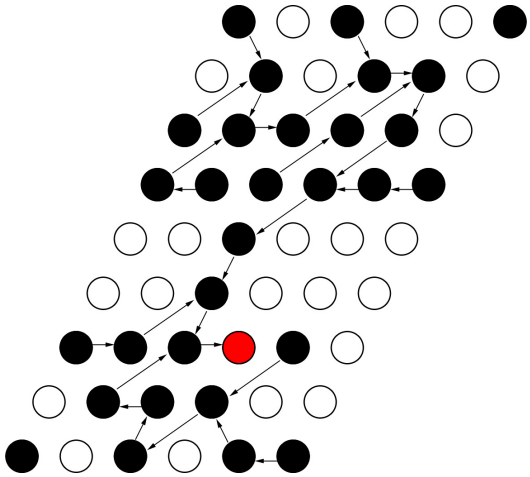


FIG. 8.16 – Arbre couvrant en 4-connexité fonctionnelle et 6-connexité matérielle

de réduire le nombre d'entrées des opérateurs convergents au minimum possible, à savoir 2 entrées. L'algorithme utilisé pour établir la composante connexe fonctionne selon le même principe que celui qui a été utilisé pour le réseau en 4-connexité fonctionnelle et 6-connexité matérielle. Il est présenté à la partie suivante.

Algorithme d'établissement de la composante fortement connexe

La démarche de construction de l'algorithme permettant l'établissement de la composante fortement connexe est proposée ici. Cette construction commence par l'étude des configurations permettant d'installer un anneau de connexions périphériques orientées autour de la région considérée. Ces configurations sont présentées à la figure 8.17.
L'utilisation de ces configurations pour établir l'anneau de connexions périphériques conduit à l'utilisation d'opérateurs de propagation à 3 entrées. Pour remédier à ce problème, il est possible d'augmenter la connexité matérielle du réseau. En utilisant un réseau 8-connexe, il est possible de fusionner des paires de configurations pour en obtenir d'autres qui permettent d'utiliser uniquement des opérateurs à 2 entrées. Ainsi les configurations (b) et (c) de la figure 8.17 peuvent être remplacées par les configurations (a), (b) et (c) de la figure 8.18. De même les configurations (e) et (f) de la figure 8.17 peuvent être remplacées par les configurations (d), (e)

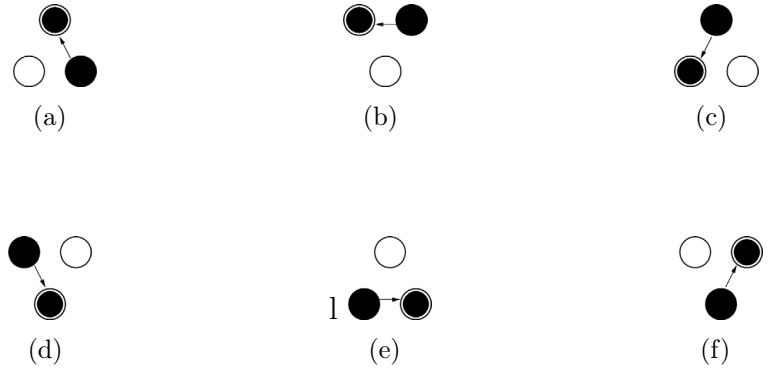


FIG. 8.17 – Algorithme d'initialisation de la composante fortement connexe utilisant un réseau matériel 6-connexe : configurations locales permettant d'établir un anneau de connexions périphériques orienté dans le sens des aiguilles d'une montre.

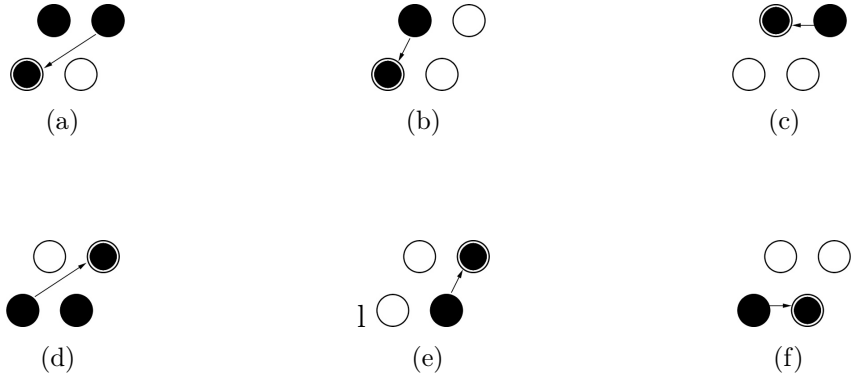


FIG. 8.18 – Algorithme d'initialisation de la composante fortement connexe utilisant un réseau matériel 8-connexe : configurations locales permettant d'établir un anneau de connexions périphériques orienté dans le sens des aiguilles d'une montre.

et (f) de la figure 8.18.
Une fois les connexions périphériques en anneau établies à l'aide d'opérateurs à 2 entrées, il reste à connecter les points intérieurs aux régions par des connexions bidirectionnelles. Pour cela on utilise les deux configurations locales de la figure 8.19.
Finalement il reste 10 configurations à étudier, les configurations (a) et (d) de la figure 8.17, les 6 configurations de la figure 8.18 et les 2 configurations de la figure 8.19. Les configurations (a) et (b) de la figure 8.18 peuvent respectivement se simplifier avec les configurations (a) et (b) de la figure 8.19. On obtient finalement 8 combinaisons à étudier, qui sont récapitulées à la figure 8.20.
L'algorithme utilisé pour réaliser les connexions est de type synchrone. Les connexions sont établies en étudiant les 8 configurations locales de pixels suivantes (fig.8.20). Le pixel entouré d'un double cercle est celui que dont on souhaite raccorder l'entrée.
On vérifie que les configurations (a), (b), (c) et (d) de la figure 8.20 s'excluent mutuellement. Il en est de même pour les configurations (e), (f), (g) et (h) de la figure

FIG. 8.19 – Démonstration de la validité de l'algorithme d'initialisation de la composante fortement connexe : configurations locales permettant d'établir des connexions bidirectionnelles intérieures aux régions.

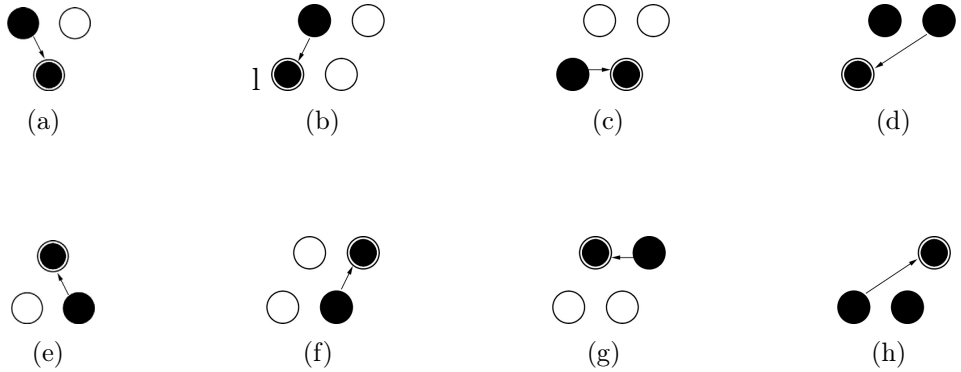


FIG. 8.20 – Algorithme d'initialisation de la composante fortement connexe : configurations locales permettant d'établir un anneau de connexions périphériques orientées dans le sens des aiguilles d'une montre en utilisant un réseau 8-connexe au niveau matériel.

8.20. Ainsi deux configurations au maximum peuvent être vraies en même temps, et donc deux connexions entrantes sont nécessaires au maximum.

L'algorithme est donc obtenu par construction des configurations à étudier de manière à créer un anneau de connexions orientées à la périphérie de la région, puis de manière à connecter les pixels intérieurs de la région par des connexions bidirectionnelles. Il faut toutefois noter que ces connexions bidirectionnelles intérieures peuvent former des sous-réseaux placés en quinconce et pouvant être déconnectés de l'anneau central. Pour corriger ce problème, il est nécessaire d'examiner deux configurations supplémentaires (fig. 8.21) après avoir établi le réseau à partir des 8 configurations préalablement décrites.
L'établissement d'une composante fortement connexe en 6-connexité fonctionnelle à l'aide d'un réseau 8-connexe s'effectue donc en deux étapes synchrones. La première consiste à examiner 8 configurations locales, la seconde consiste à examiner 2 configurations supplémentaires prenant en compte les connexions déjà établies.

A l'instar du cas des composantes fortement connexes en 4-connexité, l'établissement de la composante fortement connexe en 6-connexité fonctionnelle dans une rétine peut donc se faire de manière synchrone massivement parallèle et donc très efficace. Les résultats sur la robustesse du réseau sont toujours valables et le gain matériel est bien évidemment encore plus important qu'en 4-connexité. Ce gain est

FIG. 8.21 – Algorithme d'initialisation de la composante fortement connexe : configurations permettant d'assurer le rattachement des sous-réseaux bidirectionnels internes à l'anneau périphérique

de l'ordre de 70 % en tenant compte de la nécessité de router les signaux *req* et *ack* des 8 connexions possibles vers les 2 entrées du micropipeline convergent. La vitesse de propagation de l'information dans un processeur élémentaire est réduite, ce qui tend à réduire le temps de calcul des opérations régionales (*associations*) distribuées. Cette réduction de vitesse est toutefois à nuancer car l'arbre asynchrone n'utilisant que des fourches à 2 entrées a des branches plus allongées que dans le cas où l'on utilise des fourches à 4 entrées. Cet allongement des branches dépendant de la forme de la région considérée, il est difficile de quantifier la variation de vitesse due à cet effet. Toutefois, lors des simulations, nous avons pu constater que la vitesse globale de calcul est augmentée par la réduction du nombre d'entrées des opérateurs.

9

Les micropipelines associatifs : évaluation des performances

L'architecture des micropipelines associatifs et ses améliorations ont été décrites dans les chapitres précédents. Dans ce chapitre, une évaluation de leurs performances est effectuée.

Tout d'abord, nous évaluons la structure de micropipeline convergent dans sa version à 2 et 4 entrées en la comparant à la structure de contrôle des micropipelines et aux micropipelines existant dans la littérature. Puis nous estimons le temps nécessaire au calcul d'une somme régionale sur une rétine de taille 200 pixels par 200 pixels dans un cas défavorable afin d'avoir une borne maximale pour le temps de calcul de cette opération.

9.1 Evaluation des micropipelines convergents

La validation du circuit a été effectuée à l'aide de modèles SPICE en utilisant les outils LT Spice, simulateur gratuit et disponible en téléchargement sur internet sur le site de Linear Technology, et de HSPICE, produit par SYNOPSYS et qui est l'un des simulateurs les plus performants du marché. Les modèles de transistors utilisés sont les modèles BSIM3V3 mesurés sur des wafers réels et téléchargeables sur le site de Mosis aux États-Unis. Nous avons évalué le circuit dans les technologies suivantes :

- 0.35 μm AMS (Austria Mikro Systems) afin de permettre de se positionner par rapport à la technologie utilisée dans la dernière rétine PVLSAR.

- 0.25 μm TMSC (Taiwan Semiconductor) afin de pouvoir se positionner par rapport aux résultats présentés dans [Bee02].

- 0.18 μm TMSC (Taiwan Semiconductor) afin de quantifier les résultats que l'on pourra obtenir en réalisant le circuit dans les années futures.

Les modèles BSIM3V3 fournis par Mosis sont en level 49, ce qui empêche de les utiliser dans LT Spice. Toutefois, il est possible de travailler en modèle BSIM3V3 sous LT Spice en utilisant le level 7 ou 8. Des comparaisons entre ces deux simulateurs ont montré une différence de comportement minime de l'ordre de 2 à 3 % sur les performances en terme de vitesse. Compte tenu de l'objet de notre étude, un tel écart est négligeable. Dans la suite, les résultats présentés ont été obtenus à l'aide de LT Spice.

9.1.1 Simulation

Le circuit simulé est constitué de micropipelines convergents à 2 et 4 entrées, ainsi que de structures de contrôle de micropipeline appelés micropipelines simples. Sa structure est proposée à la figure 9.1 Les chronogrammes obtenus sous LT Spice

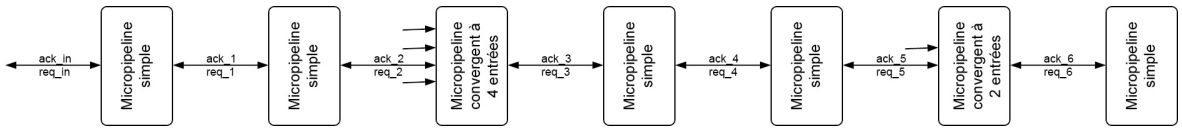


FIG. 9.1 – Circuit de test

en technologie $0.35\mu m$ et associé au circuit de test sont proposés à la figure 9.2. Sur cette figure, on peut constater que le temps de propagation dépend du type de micropipeline considéré. Dans le cas d'un micropipeline simple (c'est-à-dire la structure de contrôle d'un micropipeline standard), ce temps est de l'ordre de la nanoseconde ; dans le cas d'un micropipeline convergent à 2 entrées, ce temps passe à 2.2 ns en raison des composants ayant été rajoutés pour assurer l'arbitrage et la convergence des jetons. Enfin, dans le cas d'un micropipeline convergent à 4 entrées, le temps de propagation passe à 3.2 ns, ce qui est normal car cela correspond là encore à un ajout de logique que les signaux doivent traverser.

La simulation a également été effectuée dans les technologies $0.25\mu m$ et $0.18\mu m$. Les résultats sont récapitulés dans le tableau 9.1.
On constate que les performances en terme de vitesse des micropipelines convergents à deux entrées sont de l'ordre de 50% inférieures à celles de la structure de contrôle des micropipelines. Toutefois, la fréquence reste très acceptable si on la compare à celle de l'horloge synchrone équipant actuellement la rétine PVLSAR et qui est de l'ordre de 10 MHz. On gagne un facteur 50 (par rapport au cas synchrone à 10 MHz) dans la vitesse de propagation en technologie $0.35\mu m$, ce qui permet d'envisager les opérations régionales difficilement réalisables sur la rétine synchrone.

Comme indiqué au chapitre précédent, on constate que le passage d'un micropipeline convergent à 4 entrées à un micropipeline convergent à 2 entrées permet

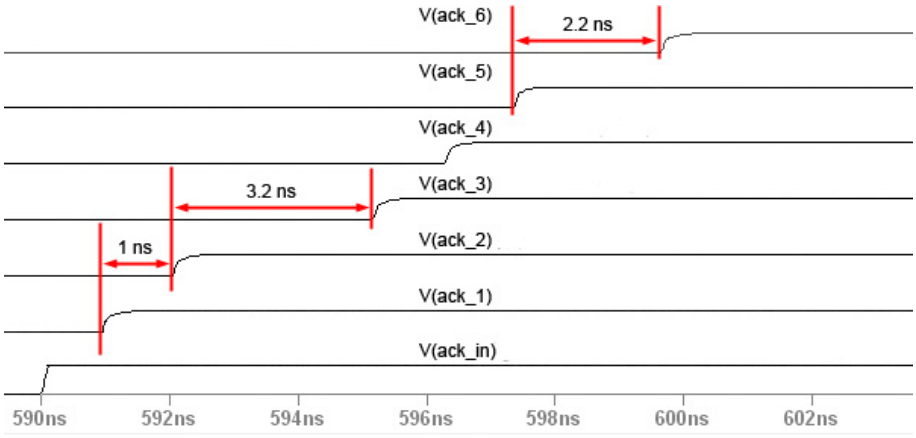


FIG. 9.2 – Résultat de la simulation en technologie 0.35 μm

Technologie	0.35μm AMS		0.25μm TMSC		0.18μm TMSC	
	Période	Fréquence	Période	Fréquence	Période	Fréquence
Micropipeline simple	1 ns	1 GHz	0.7 ns	1.4 GHz	0.43 ns	2.3 GHz
MP convergent 2 entrées	2.2 ns	450 MHz	1.4 ns	720 MHz	0.9 ns	1.1 GHz
MP convergent 4 entrées	3.2 ns	310 MHz	2.1 ns	470 MHz	1.3 ns	770 MHz

TAB. 9.1 – Vitesse de propagation des jetons dans les différents types de micropipelines étudiés.

d'augmenter la vitesse d'environ 40 à 50 %, ce qui est d'autant plus intéressant que dans le même temps, le coût matériel est réduit.

Enfin, on retrouve le même ordre de grandeur de fréquence pour les propagations sur micropipeline que dans [Bee02]. En technologie 0.25μm, la structure de contrôle des micropipelines permet de réaliser des propagations à 720 MHz en mode DI, à comparer avec la vitesse maximale atteinte par les micropipelines les plus performants permettant la transmission de données et opérant à une vitesse maximale de 770 MHz en mode DI, et 1.6 GHz si l'on accepte de ne pas se placer dans un cas DI ou QDI [Bee02]. On retrouve donc bien les mêmes ordres de grandeur, l'écart de quelques pourcents pouvant se justifier par le jeu de paramètres BSIM3V3 choisi, dans la mesure où nous avons pris un jeu de paramètres choisi arbitrairement parmi les jeux de paramètres mesurés expérimentalement par Mosis sur des wafers réels.

9.1.2 Simulation VHDL

La mise au point de la structure des micropipelines convergents a été effectuée en VHDL. Ces tests sont restés simples et étaient uniquement destinés à valider

l'architecture proposée. Ils ont été réalisés sur une carte équipée d'un FPGA Altera de type 10K100. Un exemple de mesure effectuée à l'oscilloscope est présenté à la figure 9.3. Il s'agit simplement d'un test destiné à vérifier que la structure du micropipeline convergent permet de faire converger les jetons provenant de 2 sources ($RI1$ et $RI2$) vers une seule destination (RO).
Il est toutefois à noter que les tests VHDL ne permettent pas l'implantation de

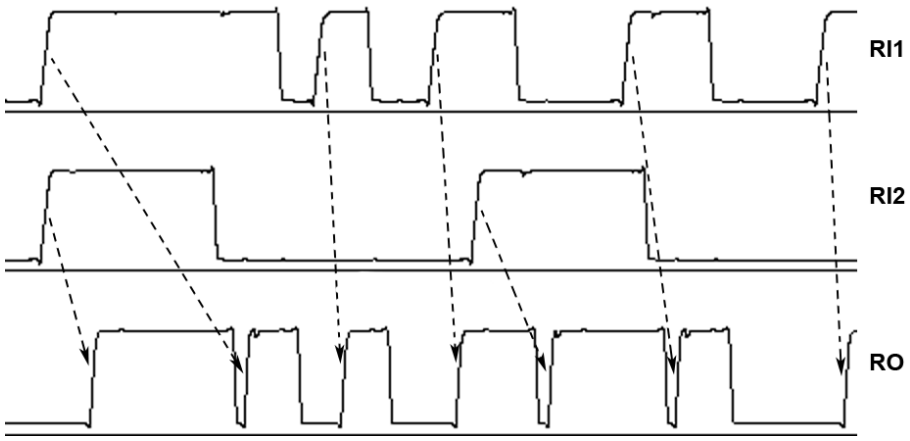


FIG. 9.3 – Résultat des tests VHDL

la mutuelle exclusion, qui doit alors être remplacée par un OU Exclusif. Cette porte n'assure pas à lui seul un fonctionnement sans fusion de jetons. Or le circuit testé fonctionne en pratique, ce qui laisse à penser que les sorties des portes des FPGA Altera de type 10K100 mettent moins de temps à établir un 0 logique qu'un 1 logique. Les retards (parfois importants) introduits par la logique combinatoire traversée dans les cellules du FPGA et par un routage non optimal n'empêchent pas le fonctionnement de notre structure, le chemin potentiellement problématique étant celui qui relie la sortie de l'arbitre et les entrées du XOR.

9.2 Evaluation du calcul d'une somme régionale

Après avoir étudié le comportement en vitesse du micropipeline convergent, quantifions à présent le temps de calcul nécessaire à une opération régionale de type somme sur chacune des régions d'une rétine artificielle.
Nous nous plaçons dans le contexte d'une rétine de taille $200 * 200$ en 4-connexité et en technologie 0.35 μm, taille correspondant à celle de la dernière rétine PVLSAR. Nous considérons le calcul d'une somme régionale de valeurs codées sur 12 bits à l'intérieur de chaque processeur élémentaire.

Un des cas les plus défavorables qui puissent être est celui d'une région unique couvrant toute la surface de la rétine, avec un point racine situé dans un angle de la rétine. Après établissement d'un arbre couvrant, la taille la plus longue de la

chaîne de pixels est nécessairement supérieure à 400 pixels (pour aller de l'angle de la rétine à l'autre situé dans la diagonale en 4-connexité). Par simulation du processus de construction d'arbre couvrant dans le cas de l'utilisation de micropipelines convergents à 2 entrées, nous avons pu constater que la longueur de la chaîne la plus longue de pixel entre la racine et une feuille de l'arbre couvrant est de l'ordre de 600 pixels [11]. Le temps τ_1 nécessaire à faire converger les jetons vers la racine est donc de l'ordre de :

$$\begin{aligned}\tau_1 &\simeq 600\ \tau_0 \\ &\simeq 600 * 2.2\ ns \\ &\simeq 1.3\ \mu s\end{aligned}$$

avec τ_0, le temps de propagation d'un jeton dans un micropipeline convergent à 2 entrées.
L'élimination des paires de jetons s'effectue en plusieurs itérations, toutefois ce processus converge assez vite. Nous avons montré expérimentalement que le nombre d'itérations est de l'ordre de 5, et qu'il ne dépasse quasiment jamais 8 itérations. Finalement, le temps de calcul asynchrone τ_2 d'un bit de la somme est donc dans le pire cas de l'ordre de :

$$\begin{aligned}\tau_2 &\simeq 8\ \tau_1 \\ &\simeq 10\ \mu s\end{aligned}$$

Dans le cas où les données sont codées sur 12 bits, et qu'il y a $200 * 200$ pixels dans la région, le nombre de bits de la somme est donc $12 + \log_2(200 * 200) \simeq 28$ bits. Le temps de calcul asynchrone τ_3 de la somme est donc au maximum :

$$\begin{aligned}\tau_3 &\simeq 28\ \tau_2 \\ &\simeq 280\ \mu s\end{aligned}$$

Finalement, le coût des opérations asynchrones nécessaires au calcul de la somme de valeurs codées sur 12 bits dans une rétine $200 * 200$ est au maximum de l'ordre de 0.3 ms, ce qui est relativement peu compte tenu de la complexité du calcul.

A ce coût temporel asynchrone, s'ajoute le coût des opérations synchrones. Nous supposerons que celles-ci s'effectuent à la vitesse de 10 MHz. Les opérations à effectuer sont de trois types : éliminations des paires de jetons, gestion de l'addition locale des jetons de niveau supérieur générés à la valeur locale stockée et chargement du micropipeline.
Chacune de ces opérations est effectuée après chaque propagation. L'élimination

[11] Nous avons effectué pour cela une centaine de simulations de construction d'arbres couvrants, les temps de propagation dans chaque cellule étant distribués selon une répartition gaussienne. Sur ces 100 simulations, nous avons obtenu à chaque fois un arbre couvrant dont la longueur maximale était inférieure à 600 pixels. Il ne s'agit donc pas ici d'un pire cas, mais un cas possible très défavorable. Le pire cas correspondant à une graphe filaire n'ayant aucune chance de se produire en réalité, nous ne l'avons pas envisagé.

des paires de jetons est une opération booléenne permettant de déterminer si des jetons sont situés dans des pixels adjacents. Pour cela il suffit de considérer un pixel sur deux (comme sur un damier), et de regarder si son voisin Est et lui-même contiennent tous deux un jeton, si c'est le cas, les deux jetons sont éliminés et un jeton de niveau supérieur est placé dans le pixel considéré. On itère ensuite sur le voisin Sud, sur le voisin Ouest, et sur le voisin Nord. Chacun des quatre tests nécessite un déplacement de plan mémoire d'un pixel et une opération de type ET logique permettant de créer un jeton de niveau supérieur. En ajoutant l'élimination du jeton $NONET$ logique, on arrive à 3 périodes d'horloge. Le coût temporel associé est donc de l'ordre de :

$$\begin{aligned}\tau_4 &\simeq 4 * 3 * 0.1\mu s \\ &\simeq 1.2\ \mu s\end{aligned}$$

La gestion des jetons de niveau supérieur se fait quant à elle à l'intérieur du pixel, et sans déplacement de plan mémoire. Compte tenu de l'algorithme d'élimination des jetons, chaque pixel ne peut contenir qu'un seul jeton de niveau supérieur. Ce jeton est donc additionné à la valeur locale stockée dans le pixel. Pour réaliser cette addition, il suffit de réaliser une addition bit-série locale synchrone. Compte tenu du fait que la valeur à additionner est 1 ou 0, cette addition peut être réalisée en 2 n opérations, où n est le nombre de bits de la valeur stockée dans le pixel. A chaque étape de l'addition bit-série synchrone locale, on peut calculer le bit suivant de la somme en réalisant un ET logique pour calculer la retenue, et un $OUexclusif$ pour calculer le bit de poids faible, ce qui justifie le 2 en supposant que l'on dispose d'un $OUexclusif$ câblé. De plus, le nombre de bits de la donnée locale diminue à mesure que l'on avance dans le calcul de la somme globale, finalement cette valeur est en moyenne 6 bits pour une donnée initiale de 12 bits. Durant le calcul du premier bit de la somme régionale, la taille de la donnée locale est en effet 12 bits, mais à la fin du calcul de la somme, la taille de la donnée locale est nulle, soit une moyenne de 6 bits.
Finalement, le coût temporel de la gestion des jetons créés est en moyenne :

$$\begin{aligned}\tau_5 &\simeq 6 * 2 * 0.1\mu s \\ &\simeq 1.2\ \mu s\end{aligned}$$

Le chargement du micropipeline consiste à effectuer la procédure de chargement décrite dans les chapitres précédents. Cette procédure comporte 3 étapes, nécessitant chacune une période d'horloge. Le temps de chargement avant chaque propagation est donc :

$$\begin{aligned}\tau_6 &\simeq 3 * 0.1\mu s \\ &\simeq 0.3\ \mu s\end{aligned}$$

Finalement, le coût temporel dû aux calculs synchrones lors du calcul de la somme régionale est la somme de τ_4, τ_5 et de τ_6 multipliée par le nombre de propagations

(au maximum $8 * 28 = 224$). On aboutit donc au temps suivant :

$$\begin{aligned}\tau_7 &\simeq 224\ (\tau_4 + \tau_5 + \tau_6) \\ &\simeq 600\ \mu s\end{aligned}$$

On trouve donc un temps total $\tau_3 + \tau_7$ environ égal à 0.9 ms dans un cas très défavorable pour le calcul d'une somme régionale, ce qui permet d'effectuer plus de 1000 calculs de sommes régionales par seconde, soit plus de 30 calculs de sommes régionales par image dans le cas d'une rétine ayant une cadence de 30 images par seconde.

Cette évaluation est un pire cas. Dans le cas du calcul d'une surface par exemple, la profondeur des données est de 1 bit et le nombre d'itérations est dans ce cas réduit fortement, même en considérant toujours une région couvrant toute l'image.

On peut également remarquer que le coût temporel le plus important est dû aux opérations synchrones. Toutefois, l'ordre de grandeur du temps nécessaire aux traitements synchrones et aux traitements asynchrones est le même. Ceci, ajouté au fait que le coût matériel de l'asynchronisme est réduit, permet de conclure que l'architecture asynchrone choisie est adaptée à la fonction régionale à réaliser. En effet, une augmentation du coût matériel de l'asynchronisme en vue d'améliorer les performances de cette partie ne conduirait qu'à une faible réduction du temps de calcul.

9.2.1 Comparaison des temps de calcul

Le temps de calcul, comme nous le verrons sur l'exemple de la segmentation sociétale dans la partie suivante, est adapté à l'utilisation massive des mesures régionales. Dans le cas de la segmentation sociétale, une centaine de mesures régionales sont effectuées au cours du déroulement de l'algorithme pour segmenter une image, ce qui conduit à un temps de calcul de l'ordre de 0.15 s. Ce temps de calcul est acceptable dans la mesure où il s'agit d'un pire cas dans une technologie assez ancienne, un cas moyen dans un technologie plus récente permettra sans aucun doute d'effectuer ce type de traitement en temps réel. Toutefois, on peut constater que la marge dont on dispose en matière de vitesse d'exécution est assez réduite, et qu'il serait donc difficile de se passer de tout ce qui contribue à améliorer la vitesse de calcul, à commencer par exemple par la réduction des distances de propagation. A cet égard, la solution proposée par l'Université de Tokyo [KKI04] et utilisant un réseau couvrant de type linéaire ne permet pas une utilisation temps réel sur une rétine de grande taille. En effet, dans un pire cas, sur une rétine $200 * 200$, le nombre de pixel de la chaîne la plus longue est 40000 pixels (si la racine est sur un bord). De plus, l'opérateur de propagation asynchrone présent dans chaque pixel (full adder) a une vitesse comparable au micropipeline convergent. Il en résulte que

le temps de calcul de la somme régionale dans un cas défavorable est au moins 10 fois supérieur (voir la partie sur l'additionneur bit-série linéaire) au temps de calcul de la somme à l'aide des micropipelines associatifs (en prenant en compte la diminution du nombre d'opérations synchrones à effectuer et le fait que l'on ait besoin d'une unique propagation au lieu de 8 au maximum pour chacun des bits de la somme à calculer). Ce facteur 10 rend difficile l'utilisation de cette structure pour des applications complexes telles que la segmentation sociétale en temps réel.

Troisième partie

Régionalisation et traitement d'image

10

Tour d'horizon des méthodes de segmentation d'images

La partie précédente de cette thèse nous aura permis d'élaborer une architecture cellulaire asynchrone, suffisamment compacte pour espérer être intégrée prochainement au sein d'une rétine artificielle, et capable de réaliser des calculs régionaux en parallèle sur toutes les régions considérées dans la grille de processeurs, avec une grande efficacité temporelle et énergétique.

Ces primitives de calcul régionales sont d'un intérêt évident pour le traitement d'image de moyen niveau. Cela constitue de fait la motivation de notre recherche. Il semble qu'elles puissent particulièrement profiter à la segmentation de l'image et à l'exploitation de l'image segmentée. Nous allons le vérifier dans cette partie.

La segmentation d'images a pour objectif de transformer un ensemble de pixels en un ensemble réduit d'objets puis à l'analyser. Cette opération offre de nombreux avantages, parmi lesquels une réduction de la quantité d'information nécessaire à la représentation des objets des images.

Ce chapitre propose un tour d'horizon des méthodes de segmentation existantes. Il ne s'agit pas tant de dénicher les méthodes pouvant bénéficier des avantages liés aux opérateurs régionaux que de comprendre leurs principes, leurs forces, leurs faiblesses, pour pouvoir au chapitre suivant laisser libre cours à la créativité algorithmique pour une pleine valorisation des primitives de calcul régionales.

10.1 Segmentation par découpage ou regroupement spatial

Cette section est consacrée à la description des méthodes adaptatives de segmentation par découpage ou regroupement spatial.

10.1.1 Principe et extensions

La plus ancienne de ces techniques est la décomposition par arbre quaternaire ou *quad-tree*. Le procédé de segmentation basé sur cette représentation revient à découper itérativement l'image. Un critère d'homogénéité d'une région, ayant pour objectif de déterminer si une région a une forte probabilité ou non de faire partie d'un unique objet, est défini. Au départ, l'image forme une seule région qui sera découpée si nécessaire. A une itération donnée, pour chacune des régions de l'image, on regarde si le critère d'homogénéité est satisfait ou non pour la région, puis le cas échéant, on la découpe en quatre parties égales. Au terme de la décomposition, l'image se présente comme une collection de régions aux caractéristiques homogènes. Ces régions peuvent ensuite être fusionnées en s'assurant que les régions participant à la fusion ont des caractéristiques homogènes.
Cette méthode, si elle n'est plus utilisée telle quelle dans les procédés de segmentation, a inspiré de nombreuses techniques de segmentation : segmentation par croissance de régions, partitionnement *top-down* (découpage d'une image initiale en sous-régions), *bottom-up* (regroupement en régions de tailles de plus en plus grandes) ou *split and merge* (méthode utilisant les deux approches précédentes), un exemple de cette dernière méthode étant l'algorithme CSC [RP98] [PKL$^+$94]. Ce dernier algorithme, qui peut être implanté de manière massivement parallèle est de type *split and merge* et utilise des tessellations de forme hexagonale.
Toutes ces techniques ont en commun l'utilisation d'un critère permettant de déterminer si une région est homogène et/ou l'utilisation d'un critère permettant de déterminer si deux régions ont des caractéristiques proches, ce qui permet d'envisager leur fusion. La forme et la taille des tessellations de l'image, utilisées pour tester l'homogénéité des régions, est prédéterminée et ne dépend pas de l'image.

10.1.2 Inconvénients

Les deux inconvénients principaux de ces méthodes sont les suivants :

- Le découpage des régions en sous-régions de forme prédéterminée, qui laisse des artefacts dans l'image, tels que des pans d'hexagones lors d'une segmentation par algorithme CSC d'un gradient en niveau de gris horizontal.

- Un second inconvénient est intrinsèquement mis en évidence par la méthode utilisée dans le cas du *split and merge*. Il s'agit de la nécessité d'itérer en alternant les opérations de *split* et de *merge*. Cette nécessité, qui résulte du découpage (*split*), se fait selon des formes prédéfinies (des hexagones dans le cas du CSC). Il en va de même pour la fusion. Cette approche conduit donc à construire un objet de forme arbitraire comme la réunion d'un ensemble de sous-objets de forme régulière. La convergence itérative d'une telle approche peut être remplacée par une convergence plus directe n'utilisant que des opérations de type *merge*, à condition d'effectuer les mesures d'homogénéités sur des tessellations de forme adaptée aux objets de l'image et donc aux régions

en cours de formation, et non plus arbitrairement définies. Une telle approche est envisagée plus loin dans ce chapitre.

10.2 Ligne de partage des eaux

Cette partie est consacrée à la description du principe de la ligne de partage des eaux introduit par S. Beuscher en 1982 [Beu82] et à ses applications algorithmiques.

10.2.1 Principe

La ligne de partage des eaux (LPE) est une technique classique de segmentation d'images. Cette approche considère la norme du gradient d'une image en niveau de gris comme un relief topographique. Ainsi les zones à fort gradient correspondent à des altitudes élevées, et les zones à faible gradient correspondent à des altitudes faibles. Les premières étant potentiellement des frontières d'objets de l'image, et les secondes l'intérieur des objets de l'image, la segmentation des objets correspond à un découpage du relief topologique en vallées. L'idée de la LPE est la suivante : si une goutte d'eau tombe sur le relief, alors elle va se diriger vers un minimum local. L'ensemble des points pour lesquels les gouttes d'eau se dirigent vers un même minimum local forme un *bassin versant* ou vallée. Les différents bassins versants forment une partition de l'image, qui constitue une segmentation. La frontière de ces bassins est appelée *ligne de partage des eaux* ou *watershed* en anglais.

10.2.2 Technique de l'immersion

Dans les premières approches proposées pour la LPE, les algorithmes essayaient de simuler l'écoulement de l'eau le long du relief. Or cette notion d'écoulement est mal définie sur un relief à valeurs discrètes et la détermination du bassin versant correspondant à chaque pixel nécessite de parcourir un grand nombre de pixels avant d'arriver au minimum local, ce qui engendre un grand nombre de calcul pour une fiabilité discutable. Au début des années 90, une autre approche a été proposée : l'immersion du relief. Cette technique est basée sur le principe suivant :

- Chaque minimum local du relief est considéré comme une source.
- Le relief est inondé par l'eau provenant des sources, le niveau de l'eau étant le même en tout point de l'image à tout instant.
- Les eaux provenant de deux bassins versants ne peuvent pas se mélanger, des digues sont donc construites de manière à éviter ce mélange.
- Lorsque le relief est totalement immergé, les digues formées constituent les lignes de partage des eaux partitionnant l'image en régions.

La figure 10.1 représente un profil de gradient à 1 dimension et la figure 10.2, représente le résultat d'une segmentation par LPE sur ce profil. A noter que la montée de l'eau a été arrêtée dans chacun des bassins versant dès que celui-ci était pris entre deux digues de manière à améliorer la lisibilité du dessin. La technique de l'immersion est efficacement implantée sur différents types d'architecture et en

FIG. 10.1 – Profil d'un gradient à 1 dimension

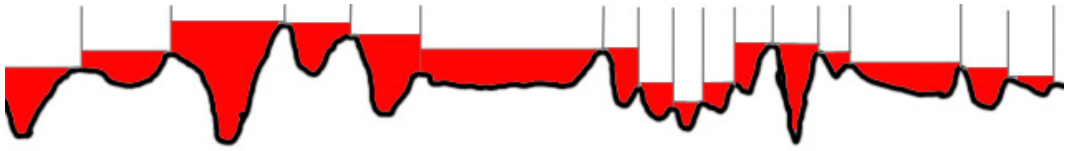


FIG. 10.2 – Ligne de partage des eaux sur un profil à 1 dimension

particulier sur les architectures massivement parallèles. Toutefois le nombre de bassins versants générés par l'algorithme est égal au nombre de minima locaux du relief. Dans le cas d'un relief bruité (correspondant à une image bruitée), le nombre de ces minima locaux peut être extrêmement important, ce qui conduit à une sur-segmentation importante, illustrée par la figure 10.2.

10.2.3 Formalisation

Soit E_h l'ensemble des pixels p de E, tels que $alt(p) \leq h$ où $alt(p)$ est l'altitude du pixel p. On pose h_{min} et h_{max} la valeur minimale et maximale de l'altitude alt sur E. On désigne par $\mathcal{C}(M)$ le bassin versant associé au minimum local M.
L'ensemble $\mathcal{C}_h(M) = \mathcal{C}(M) \bigcap E_h$ est donc l'ensemble des points du bassin versant ayant une altitude inférieure ou égale à h. On pose $\bigcup \mathcal{C}_h = \bigcup_i \mathcal{C}_h(M_i)$, la réunion des ensembles connexes de pixels appartenant aux bassins de capture et ayant une altitude inférieure ou égale à h.

Une zone d'influence géodésique est définie comme suit : soit $X \subset Y$ avec $X = \bigcup_i X_i$, X_i étant des composantes connexes disjointes de X. La zone d'influence géodésique $\mathcal{Z}_Y(X_i)$ de l'ensemble X_i dans Y est égale à :

$$\mathcal{Z}_Y(X_i) = \{p \in Y | d_Y(p, Xi) < d_Y(p, Xj) \forall j \neq i\}$$

avec d_Y la distance géodésique dans l'ensemble Y, à savoir la plus petite distance d'un point à son ensemble en suivant un chemin appartenant à Y.

L'inondation se fait par accroissement des bassins de capture lorsque le niveau de l'eau monte. La première inondation a lieu lorsque l'eau atteint le niveau h_{min},

les pixels alors sont inondés sont :

$$\bigcup \mathcal{C}_{h_{min}} \;=\; E_{h_{min}}$$

Ensuite, à chaque itération où le niveau de l'eau monte de h à $h+1$, trois cas de figure peuvent se présenter pour chacune des composantes connexes E^i_{h+1} de E_{h+1} :

- $E^i_{h+1} \bigcap (\bigcup \mathcal{C}_h) = \emptyset$; E^i_{h+1} est un nouveau minimum régional d'altitude $h+1$ car son intersection avec l'ensemble des points d'altitude h est vide et il forme à lui tout seul un ensemble connexe.

- $E^i_{h+1} \bigcap (\bigcup \mathcal{C}_h) \neq \emptyset$ et est connexe ; E^i_{h+1} est l'expansion d'un bassin versant unique $\mathcal{C}^i_h$. A l'itération suivante, on a : $\mathcal{C}^i_{h+1} = E^i_{h+1}$

- $E^i_{h+1} \bigcap (\bigcup \mathcal{C}_h) \neq \emptyset$ et n'est pas connexe ; E^i_{h+1} est l'expansion de plusieurs bassins versants.Cela signifie que plusieurs bassins versants connexes $\mathcal{C}^i_h$ au niveau h se rejoignent au niveau $h+1$. Dans ce cas une digue est construite à mi-distance entre les deux composantes connexes les plus proches, ce qui signifie en termes d'influence géodésique que pour chaque bassin versant $\mathcal{C}^i_h$ ayant une intersection non vide avec E^i_{h+1}, on a à l'itération suivante : $\mathcal{C}^i_{h+1} = \mathcal{Z}_{E^i_{h+1}} \mathcal{C}^i_h$

La construction des lignes de partage des eaux se fait donc itérativement pour des valeurs de h allant de h_{min} à h_{max}.
Une implantation à base de files d'attente proposée par Vincent et Soille [LS91] est rappelée à la section 10.2.5.

10.2.4 Utilisation de marqueurs

La sur-segmentation présente dans les algorithmes classiques de LPE provient du fait qu'il y a un nombre de minima locaux trop important dans l'image. Ce problème peut être contourné en se restreignant à l'utilisation de marqueurs comme sources d'inondation. Le principe de l'algorithme de LPE présenté ci-dessus est alors légèrement modifié. Les bassins sans marqueurs seront inondés à partir des bassins déjà inondés, mais le principe fondamental interdisant le mélange des eaux provenant de sources différentes est conservé et les digues sont construites pour le garantir. Toutefois, l'utilisation de marqueurs nécessitant d'avoir des informations sur l'image à traiter, elle ne peut donc pas être utilisée dans n'importe quelle situation.

10.2.5 Implantation de l'algorithme

La première implantation de LPE par immersion a été introduite par Vincent et Soille [LS91]. Les pixels sont classés dans un tableau suivant l'ordre croissant de leur altitude. A chaque seuil h, la reconstruction géodésique est réalisée à l'aide d'une file d'attente. Lorsque l'inondation a atteint l'altitude h, tous les bassins versants

déjà découverts ont un label grâce au classement dans le tableau (les bassins de capture déjà découverts contiennent des minima dont l'altitude est inférieure ou égale à h). Les pixels du niveau $h+1$ sont obtenus directement à partir de ce même tableau : ils ne contiennent pas encore de label. Les pixels déjà labellisés sont placés dans une file d'attente et leurs labels sont propagés parmi les pixels d'altitude $h+1$. Lorsque la file d'attente contenant initialement les pixels labellisés est vide, les zones d'influence géodésique sont construites. Cependant il est nécessaire d'effectuer un deuxième passage sur les pixels d'altitude $h+1$, afin d'affecter un nouveau label aux pixels n'ayant pas été rattachés à une zone d'influence et correspondant à des minima locaux. Un label particulier (LPE) est affecté aux pixels où les bassins versants essayent de fusionner. Il désigne les pixels où se trouve la LPE.
L'algorithme de segmentation par LPE implanté par Vincent et Soille souffre d'un défaut important : dans la mesure où les labels LPE sont considérés comme des labels normaux, ils peuvent se propager à de larges plateaux et former des régions. Ainsi les lignes de partage des eaux formées ne sont pas nécessairement d'épaisseur unitaire ou réduite à 2 pixels, et peuvent même former des régions.

10.2.6 Améliorations possibles

Les deux exemples de segmentation par LPE présentés ci-dessus ont été améliorés de différentes manières, tant dans leur concept que dans leur implantation. Le concept a été étendu par Meyer à l'utilisation du relief dans la construction de la LPE en utilisant une distance de type topographique [Mey94]. L'implantation a été améliorée grâce à l'introduction de files d'attente hiérarchiques [BF93].

Par ailleurs, une version cellulaire asynchrone dédiée a été proposée par Bruno Galilée [Gal02]. Il s'agit d'un algorithme de type *Hill-climbing* réordonnancé de manière à s'exécuter en une seule phase asynchrone. Le principe de l'algorithme est le suivant. Initialement, les pixels portent chacun une étiquette différente. Au cours de la phase asynchrone, tout pixel ayant des voisins à une plus faible altitude que lui adopte la plus faible des étiquettes de ses voisins les plus bas. Ainsi les pixels d'un même bassin versant finissent par avoir une même étiquette, qui est l'étiquette la plus faible que l'on trouvait initialement en bas du dit bassin.
A côté des moyens de calcul nécessaires pour comparer altitudes et étiquettes, l'algorithme se contente d'un automate à trois états à l'intérieur de chaque pixel. Cette machine n'ayant aucune contrainte temporelle, elle peut-être implantée sous forme asynchrone à l'intérieur des pixels, ce qui a été proposé par B. Galilée. Cette implantation conduit à des résultats spectaculaires en terme de vitesse de calcul (un maximum de 120000 images par secondes) et de coût énergétique (quelques dizaines de micro-watts en fonctionnement normal). Toutefois, elle nécessite une surface de silicium bien supérieure à celle qui est disponible dans un processeur élémentaire de rétine. Il serait en effet nécessaire d'implanter l'équivalent de 4000 transistors pour le seul calcul de la segmentation par LPE, soit environ 15 fois le nombre de transistors dont on peut disposer à l'intérieur d'un processeur élémentaire de rétine, et 150 fois le nombre de transistors que l'on propose de dédier aux

calculs régionaux en utilisant les micropipelines associatifs.

10.2.7 Inconvénients

Les inconvénients de la segmentation par LPE dépendent de la version utilisée. Dans le cas de l'utilisation de la LPE sans marqueur, le résultat de la segmentation peut être très sur-segmenté, en particulier sur des images bruitées. Des solutions à ce problèmes ont été proposées par l'ajout de pré-traitement de type filtrage passe-bas, efficace mais conduisant également à une perte de résolution spatiale, ce qui est dans certaines situations très ennuyeux. D'autres méthodes ont proposé de fusionner les régions obtenues selon des critères d'homogénéité. Nous proposerons plus loin une méthode basée sur des mesures régionales statistiques offrant une solution statistiquement pertinente au problème de sur-segmentation (cette méthode sera présentée ultérieurement). Toutefois, si le problème de la sur-segmentation peut être partiellement résolu, il n'en est pas de même du problème de la non régularité des contours. Comme indiqué précédemment, la LPE est très sensible aux minima locaux dus au bruit de l'image. Ces minima locaux sont peu présents dans les zones de fort gradient, mais sont en revanche très présents dans les zones de faible gradient. En conséquence, dans ces zones planes correspondant souvent à l'intérieur des objets, la segmentation par LPE initiale ne se fait que sur le bruit de l'image, ce qui conduit à des frontières de forme très bruitée et irrégulière. Même après fusion des régions, ces formes très bruitées sont toujours présentes puisque la fusion ne change pas la forme des contours.
Dans le cas de l'utilisation de la LPE avec marqueurs initiaux, le problème de la sur-segmentation est réglé. Cependant se pose le problème du placement des marqueurs initiaux, à l'instar des snakes (voir plus loin). Ce placement nécessite d'avoir une connaissance *a priori* de l'image à segmenter.

10.3 Equations aux dérivées partielles

Les méthodes de segmentation à l'aide d'équations aux dérivées partielles (EDP) bénéficient des progrès récents en analyse numérique et en géométrie différentielle. Grâce aux avancées informatiques, les EDP se rencontrent maintenant dans de nombreux domaines tels que la physique, les sciences des milieux continus, ou encore la médecine.

Cette section recense les principales contributions en matière d'EDP. En particulier, les différents modèles correspondant à des formulations linéaires ou non linéaires, ainsi que l'approche à base de fonctionnelle énergétique sont présentés en soulignant les différents liens existant entre eux et les difficultés pratiques et théoriques de chacun. Nous aborderons pour terminer la pertinence de l'utilisation des techniques EDP pour la segmentation d'images.
Dans un contexte général, les atouts principaux des EDP sont les suivants :

- Les EDP permettent de modéliser les images dans le domaine continu.
- La formulation par EDP est indépendante de la définition de la maille. Elle est isotrope.
- Les EDP permettent de formuler de nouveaux filtres qui satisfont la causalité et la localité.
- Les algorithmes sont performants et allient précision et stabilité de solution.
- Les EDP permettent d'obtenir des démonstrations de l'existence et l'unicité des solutions.

L'idée d'utiliser des EDP dans le domaine du traitement d'images date des années 60 par Gabor [Gab65] puis plus tard, Jain [Jai77] a continué dans cet axe. Toutefois, les contributions faites par Koenderink [Koe84] et Witkin [Wit83], introduisant l'approche multi-échelle basée sur la convolution gaussienne ont marqué le véritable début de l'utilisation des EDP dans le domaine du traitement d'image. Cette approche produit les mêmes résultats qu'une déformation de l'image par diffusion isotropique telle qu'elle peut être décrite par l'équation de la chaleur. En 1986, Hummel a complété cette approche alternative en démontrant que toute équation satisfaisant au principe du maximum définit un espace multi-échelle.
En 1990, Perona et Malik [PM90], ont étendu l'usage des EDP au cas de la diffusion anisotrope (en remplacement du filtrage gaussien), ce qui a permis d'obtenir de nouvelles propriétés telles que la conservation de contours. Ces travaux, ainsi que de nombreux autres ont été unifiés par le modèle général de segmentation par EDP proposé par Mumford et Shah [MS89]. Parmi les autres travaux intéressants à citer, figurent ceux d'Alvarez, Guichard, Lions et Morel [AGLM93] qui ont établi d'un point de vue théorique les axiomes de base et les EDP fondamentales.

10.3.1 Filtres diffusifs linéaires

Dans ce paragraphe, afin de présenter en quoi les EDP permettent de réaliser des opérations de traitement d'image, nous rappelons tout d'abord l'équivalence entre le premier et plus simple des modèles à avoir été utilisé, la diffusion linéaire, et le filtrage gaussien.
Soit $u_0(x)$ le niveau de gris en un point x de l'image à traiter, et G_σ la gaussienne d'écart-type σ donnée par la formule suivante :

$$G_\sigma(x) = \frac{1}{4\pi\sigma} exp(-\frac{|x|^2}{4\sigma}) \tag{10.1}$$

Le filtrage gaussien de l'image résulte de la convolution de cette fonction u_0 avec des gaussiennes en chaque point de l'image :

$$u(x,\sigma) = (G_\sigma * u_0)(x) = \int_{\mathbb{R}^2} G_\sigma(x-y)u_0(y)dy \tag{10.2}$$

G_σ étant dans $C^\infty(\mathbb{R}^2)$, il en va de même pour $G_\sigma * u$. La convolution est une opération régularisante, en pratique, elle permet de lisser en appliquant un filtre passe-bas spatial coupant les variations spatiales de dimension inférieure à σ.

Afin de d'obtenir une équivalence entre ce filtrage passe-bas et une équation aux dérivées partielles, la transformée de Fourier est calculée sur les deux membres de l'équation 10.2 :

$$\mathcal{F}(u)(x,\sigma) = \mathcal{F}(G_\sigma)(x).\mathcal{F}(u_0)(x) \tag{10.3}$$

Or la transformée de Fourier d'une gaussienne $\mathcal{F}(G_\sigma)$ est une gaussienne :

$$\mathcal{F}(G_\sigma)(x) = exp(-4\pi|x|^2\sigma) \tag{10.4}$$

En calculant la transformée de Fourier inverse après remplacement de 10.4 dans 10.3 dans le cas unidimensionnel où le paramètre σ est assimilé au temps t, on aboutit à :

$$\frac{du}{dt} = \frac{\partial^2 u}{\partial x^2} \tag{10.5}$$

Le paramètre σ correspond au niveau de filtrage spatial du filtre. Dans l'équivalence avec les EDP, ce paramètre correspond au temps de diffusion, il peut donc être assimilé au temps t.
Cette équivalence reste valable dans le cas multidimensionnel. L'image filtrée par convolution avec des gaussiennes u est donc solution de l'équation de diffusion linéaire, appelée également équation de la chaleur, et ayant pour solution initiale $u_0(x)$:

$$\begin{cases} \dfrac{\partial u(x,t)}{\partial t} = \Delta u(x,t) \\ u(x,0) = u_0(x) \end{cases} \tag{10.6}$$

L'équivalence entre le filtrage de gauss et l'équation de la chaleur est classique ; elle a été introduite par Marr et Hildreth [MH80]. Le filtrage (ou lissage) doit être fait en un temps fini, égal au paramètre σ. L'équation parabolique linéaire permet une diffusion identique dans toutes les directions (diffusion isotrope). Ainsi, dans les zones homogènes d'une image, ce filtre permet effectivement d'atténuer le bruit considéré comme étant un signal de haute fréquence, mais dans les zones présentant des discontinuités de niveau de gris, celles-ci seront aussi lissées. Ce filtre ne préserve donc pas les objets significatifs de l'image. Il les floue, ce qui rend la détermination des contours quasi impossible, diminuant en conséquence l'intérêt de ce processus de diffusion. Cela peut être évité par la modification de l'opérateur de diffusion, en renonçant à sa linéarité. Les travaux réalisés dans ce cadre sont abordés ci-après.

10.3.2 Filtres non linéaires

La motivation essentielle des modèles basés sur la diffusion non linéaire est la construction d'un opérateur de diffusion dépendant des propriétés locales de l'image. On peut distinguer deux types de filtres non linéaires :

– Le filtrage isotrope non linéaire, utilisant une équation de diffusion non linéaire avec une diffusivité scalaire adaptée aux propriétés locales du signal.

– Le filtrage anisotrope non linéaire, utilisant une équation de diffusion non linéaire à diffusivité tensorielle et dont l'opérateur de diffusion s'adapte aux structures de l'image ce qui permet de contrôler les directions de diffusion.

Le premier modèle non linéaire a été introduit par Perona et Malik en 1990 [PM90]. Le filtre adaptatif (ou conditionnel) proposé dans cet article permet d'atténuer la diffusion dans les régions à fort gradient, où des discontinuités potentiellement significatives peuvent se trouver, et de la maintenir dans les zones à faible gradient. Cela permet d'éviter le seuillage nécessaire pour les modèles linéaires à la fin du traitement. Ce modèle a permis une avancée très importante dans le domaine de l'application des EDP en traitement d'images. Cependant, le problème abordé sous l'angle choisi par Perona et Malik peut être mal posé ; en conséquence, plusieurs améliorations basées sur la régularisation (par moyenne locale ou convolution avec une gaussienne) ont été introduites depuis.

Le modèle de Perona et Malik

Formulation initiale L'idée de Perona et Malik [PM88] a été de remplacer l'équation de la chaleur par une équation de diffusion de type milieu poreux :

$$\begin{cases} \dfrac{\partial u}{\partial t} = div\left(g(\nabla u)\nabla u\right) \\ u(x,0) = u_0(x) \end{cases} \tag{10.7}$$

Dans cette équation, g est une fonction régulière non croissante avec $g(0) = 1$, $g(s) \geq 0$ et $\lim_{s\to\infty} g(s) = 0$. L'idée est que le traitement obtenu par l'équation 10.7 est conditionnel en chaque point x de l'image ($\Omega \subset \mathbb{R}^2$) :

– Si $\nabla u(x)$ est grand alors la diffusion s'atténue, par suite on a une localisation exacte des bords.
– Si $\nabla u(x)$ est petit alors la diffusion tend à lisser le voisinage de x.

En pratique, le choix de la diffusivité $g(\nabla u)$ correspond à un seuillage comparable à celui du gradient $\nabla u(x)$, utilisé entre autre dans l'étape finale des filtres linéaires.
Pour les tests numériques de ce modèle, on prend généralement comme fonction g :

$$g(\nabla u) = \frac{1}{1 + \left(\dfrac{\|\nabla u\|}{\lambda}\right)^2} \qquad \lambda > 0,$$

ou encore :

$$g(\nabla u) = e^{-\left(\frac{\|\nabla u\|}{\lambda}\right)^2} \qquad \lambda > 0,$$

Ces deux expressions de la fonction g présentent en fait la même approximation au premier ordre. Perona et Malik ont montré que ce modèle améliore la performance du filtre de Canny à détecter les bords dans une image bruitée.

Formulation variationnelle Après avoir vu la formulation initiale du modèle de Perona et Malik, voyons à présent son équivalence avec une formulation variationnelle, en terme de fonctionnelle d'énergie. Une telle formulation permet de se rapprocher de la théorie des contours actifs (cf. section suivante), pour finir par une équivalence entre les EDP et le méthode à base de contours actifs.

La formulation intégrale du modèle de Perona-Malik est une fonctionnelle d'énergie donnée par :

$$E(u) = \frac{1}{2} \int_{\Omega} \int_{0}^{t} g\left(|\nabla u(x,s)|\right)\, dx\, ds \tag{10.8}$$

En supposant que u minimise E et que v est une fonction de test quelconque, la variation de E fournit :

$$\begin{aligned} \lim_{t\to\infty} \frac{E(u+tv)-E(u)}{t} &= \int_{\Omega} g\left(|\nabla u|\right) \nabla u \nabla v\, dx \\ &= -\int_{\Omega} v\ div\left(|\nabla u|\right) \nabla u\, dx \end{aligned}$$

L'équation de Perona Malik peut donc être interprétée comme une descente de gradient de la fonctionnelle E :

$$\frac{\partial u}{\partial t} = -\nabla E \tag{10.9}$$

Améliorations possibles

Malgré une approche théorique intéressante, le problème de Perona-Malik présente de nombreuses difficultés pratiques et théoriques. En présence d'un fort bruit ou d'une forte texture dans l'image, l'information portée par le gradient est non significative, mais est cependant considérée comme relative à des contours potentiels. En conséquence, ceux-ci (et donc le bruit) sont conservés. Le lissage proposé par la méthode n'est donc pas efficace. D'un point de vue plus théorique, l'unicité et même l'existence de la solution de l'équation de diffusion en milieu poreux ne peut être prouvée, ce qui rend le problème mal posé. Sous certaines conditions [NS92], le modèle agit comme une équation de la chaleur rétrograde, qui peut conduire à une divergence de la solution même dans le cas d'une image de départ peu bruitée. Des régularisations spatiales autant que temporelles de la fonction de conductivité ont été proposées afin de rendre le problème bien posé et de stabiliser les résultats [NS92] [CMLC92].

10.3.3 Discrétisation

Sur des architectures analogiques continues telles que les réseaux de neurones cellulaires, des implantations ont été proposées, avec des propriétés de convergence temporelle asymptotique et de stabilité [Cot95].

Pour être utilisées dans un dispositif de traitement d'image discret, les méthodes de traitement d'image à base d'EDP doivent être discrétisées. Pour cela, compte tenu du fait que l'image est définie sur une grille régulière fixe, la méthode des différences finies est une des mieux adaptées, et elle est compatible avec l'implantation sur des machines massivement parallèles telles que les rétines artificielles.
La discrétisation conduit à limiter la portée en distance des opérateurs de diffusion. En conséquence, la relaxation est une opération locale, qui se propage à l'échelle d'une région de manière itérative. L'implantation de tels opérateurs sur un dispositif massivement parallèle ne peut conduire à une convergence rapide de la solution. Le recours à des mesures régionales permettrait sans doute d'accélérer le processus de convergence (ou d'améliorer les résultats). Pour cela, la formulation locale est inadéquate. Une formulation régionale sous forme d'énergie est préférable, ce qui est proposé ci-après.

10.3.4 Vers une formulation énergétique

Comme nous l'avons vu précédemment, la formulation d'un problème sous forme d'une équation aux dérivées partielles peut également s'exprimer sous la forme d'une fonctionnelle énergétique. La nécessité d'itérer les diffusions locales pour aboutir à des minimisations globales conduit à envisager l'utilisation de mesures régionales. Pour cela, l'image doit être considérée à un niveau régional, qui peut être celui des fonctionnelles énergétiques sur des régions. Dans ce paragraphe, nous recensons quelques fonctionnelles énergétiques classiques, et lorsque cela est possible, leur modèle de diffusion associé.

Modèle de Mumford-Shah

Mumford et Shah [MS85] [MS89] ont proposé une méthode de segmentation d'une région d'image Ω basée sur la minimisation de la fonctionnelle suivante :

$$E_{u_0}(u, w) = \beta \int_{\Omega} (u - u_0)^2 \, dx + \int_{\Omega \setminus B} |\nabla u|^2 \, dx + \alpha |B| \tag{10.10}$$

où B est l'ensemble des *bords* et $|B|$ la distance de Hausdorff de l'ensemble des *bords*, α et β étant deux constantes positives. La distance de Hausdorff est définie par :

$$|B| = inf\,(r > 0 \quad tq \quad \forall X, Y \in B, \quad X \subset V(r, Y) \quad et \quad Y \subset V(r, X))$$

Ce modèle consiste à approcher la région initiale u_0 définie sur un domaine $\Omega \subset \mathbb{R}^2$ et à valeurs dans $[0, 1]$ par une fonction u présentant des discontinuités dans un sous ensemble K. Ce type de filtre combine donc en même temps traitement, lissage de l'image, et détection des bords.

Le premier terme de la fonctionnelle correspond à une minimisation de l'écart local entre la valeur des points de la région obtenue et de la région initiale.

Le second terme conduit à éliminer les points n'appartenant pas à la bordure et ayant un fort gradient (ces points sont en effet le plus souvent des bords de régions).

Enfin le troisième terme prend en compte la taille de la région obtenue. Il correspond justement à l'utilisation simple d'une mesure régionale. Cela rend en revanche la fonctionnelle impossible à exprimer sous forme d'un modèle local de diffusion. Comme présenté dans l'exemple suivant, le fait de supprimer les mesures régionales peut permettre de fournir un modèle locale de diffusion à base d'EDP.

Modèle de Nordström

La fonctionnelle utilisée par Nordström est la suivante :

$$E_{u_0} = \int_\Omega \left(\beta(u-u_0)^2 + w|\nabla u|^2 + \lambda^2(w - \ln w)\right)\, dx$$

avec $w : \Omega \to [0,1]$, et $w \approx 1$ à l'intérieur d'une zone homogène et $w \approx 0$ sur les bords, α et β étant deux constantes positives.
L'équation d'Euler correspondante est la suivante :

$$\begin{aligned} \beta(u-u_0)^2 - div(w\nabla u) &= 0 \\ \lambda^2(1-\frac{1}{w}) + |\nabla u|^2 &= 0 \end{aligned}$$

Ce qui permet d'obtenir l'équation de diffusion correspondante :

$$\begin{cases} \dfrac{\partial u}{\partial t} = div(g(|\nabla u|^2)\nabla u) + \beta(u-u_0) \\ u(x,0) = u_0(x) \end{cases}$$

Dans cette équation apparaît le terme de diffusion de Perona-Malik, et un terme de retour élastique vers l'image initiale. Cet ajout permet de réduire la sensibilité de l'algorithme à l'instant où s'effectue l'arrêt des itérations, et d'accélérer la convergence. Toutefois, la fonctionnelle n'étant pas convexe, la solution obtenue n'est pas nécessairement un minimum global.
Dans ce dernier cas, il a été possible d'obtenir l'équation aux dérivées partielles correspondant à la minimisation d'une fonctionnelle énergétique, mais force est de constater que le terme correspondant à la mesure régionale (distance de Hausdorff) a disparu. Ce terme régional posait problème pour passer à une formulation EDP qui est, comme nous l'avons déjà vu, de nature locale.

10.3.5 Inconvénients

L'utilisation des EDP pour le traitement d'image donne des résultats intéressants, mais son utilisation à des fins de segmentation d'image sous forme d'EDP pose un problème de fond. Les EDP décrivent par nature des évolutions locales

alors qu'un des objectifs de la segmentation est de permettre l'extraction d'objets dans l'image, objets ayant des propriétés d'homogénéité à une échelle régionale. Ainsi, une méthode basée purement sur l'utilisation de données locales pour obtenir des informations de type régional a peu de chance d'être efficace. Les opérateurs régionaux peuvent en effet être implantés, mais sous forme d'itérations d'opérateurs locaux, ce qui conduit à un nombre d'itérations dépendant de la taille des régions. Une approche plus régionale des EDP peut être faite en considérant la fonctionnelle d'énergie sur une région. Une telle approche conduit à l'utilisation de l'énergie présente à l'intérieur et sur les contours de régions, ce qui nous amène à étudier le cas des contours actifs ou *snakes*.

10.4 Contours et surfaces actifs

10.4.1 Principe

Le principe des méthodes de segmentation à base de contours et surfaces actives est de « faire » évoluer une courbe fermée, depuis une position initiale, choisie arbitrairement ou non, vers les contours de régions d'intérêt. L'évolution de cette courbe dynamique, communément appelé contour actif, est guidée par la minimisation de l'énergie définie comme fonction du contour actif. Cette énergie est choisie de manière à ce qu'elle soit minimale lorsque le contour délimite les objets que l'on souhaite segmenter.
La définition de cette énergie, ou fonctionnelle, est donc un élément essentiel de la segmentation par contours actifs. Elle consiste à exprimer sous forme objective les caractéristiques discriminantes des régions d'intérêt. On détermine ainsi une énergie constituée de termes décrivant l'objet, son contour, et la région lui étant extérieure. Ces termes s'expriment sous forme d'intégrales définies sur les régions, intérieures et extérieures au contour, et sur le contour.
Bien que très proches conceptuellement, les méthodes de segmentation basées sur les EDP et celles basées sur les contours et surfaces actives ont une différence majeure : dans le cas des contours actifs, le résultat de la relaxation du contour fournit directement une segmentation, alors qu'il est nécessaire d'appliquer une détection de contours (filtre de Canny par exemple) comme étape finale d'une segmentation par EDP.

10.4.2 Formalisme

Le formalisme "actif" est présenté sur un exemple de contour actif [KWT87]. Soit $\mathcal{C}$ une courbe d'abscisse curviligne s. $v(s,t)$ définit le point courant d'abscisse s à l'instant t. Les extrémités de la courbe sont nommées a et b. Une énergie caractérisant l'adéquation de la courbe et de l'image au voisinage de la courbe ainsi que la régularité de cette courbe peut être définie ainsi :

$$E(\mathcal{C}) = E_{interne}(\mathcal{C}) + E_{externe}(\mathcal{C})$$

L'énergie interne $E_{interne}(\mathcal{C})$ dépend de la longueur de la courbe et de sa régularité comme indiqué dans l'équation suivante :

$$E_{interne}(\mathcal{C}) = \int_a^b \alpha(s) \| \frac{\partial v(s)}{\partial s} \|^2 \, ds + \int_a^b \beta(s) \| \frac{\partial^2 v(s)}{\partial s^2} \|^2 \, ds$$

L'influence de la longueur et de la régularité de la courbe est ajustée à l'aide des coefficients $\alpha(s)$ et $\beta(s)$ appelé respectivement coefficients d'élasticité et de rigidité et la plupart du temps pris comme des constantes de long du contour. Cette énergie est d'autant plus grande que le contour est long et qu'il est irrégulier.

L'énergie externe $E_{externe}(\mathcal{C})$ caractérise l'adéquation entre la courbe et sa position dans l'image, elle est définie par la fonction suivante :

$$E_{externe}(\mathcal{C}) = - \int_a^b \alpha(s) |\nabla I \left(v(s) \right)|^2 \, ds$$

Cette énergie est négative et sa valeur absolue est maximale lorsque le contour se trouve localisé dans les zones de fort gradient correspondant aux bords de l'objet à segmenter.
La minimisation de l'énergie globale conduit à régulariser le contour et à le placer dans des régions ayant un fort gradient pouvant s'apparenter aux bords de l'objet à segmenter.

10.4.3 Résolution

Un modèle de snake est un problème d'optimisation fonctionnelle. Comme pour les méthodes à base d'EDP, deux type d'approches peuvent être envisagées :

- L'optimisation directe de la fonctionnelle énergétique.
- La résolution de l'équation d'Euler discrétisée correspondant à la fonctionnelle énergétique.

Optimisation directe de la fonctionnelle énergétique

Cette approche appelée algorithme du snake glouton [WS92] consiste à tester quels déplacements locaux du contour du snake permettent de réduire l'énergie qui lui est associée et à déplacer le snake dans ce nouvel état ayant une énergie plus faible. Cette méthode présente un inconvénient majeur qui est le besoin d'effectuer des itérations sur chaque voisin de chaque point du snake et cela un grand nombre de fois, ce qui conduit à des temps de calculs très importants. En revanche, cette méthode permet d'obtenir une solution même dans les cas où les équations d'Euler associées à la fonctionnelle énergétique sont très complexes ou impossible à déterminer.

Résolution de l'équation d'Euler associée

La minimisation de cette fonctionnelle énergétique peut se faire par la résolution discrétisée de l'équation d'Euler associée, à savoir :

$$\alpha v^{(2)}(s) - \beta v^{(4)}(s) - \nabla E_{image}(v(s)) = 0$$

On utilise les formules d'Euler pour la discrétisation, avec h la largeur d'un pixel :

$$v'(s) = \frac{v(s) - v(s-h)}{h} \quad v''(s) = \frac{v(s+h) - 2v(s) + v(s-h)}{h^2}$$

On obtient l'équation discrétisée suivante, avec i le pixel courant :

$$\alpha(v_{i+1} - 2v_i + v_{i-1}) - \beta(v_{i+2} - 4v_{i+1} + 6v_i - 4v_{i-1} + v_{i-2}) - \nabla E_{image}(v(i)) = 0$$

En regroupant les termes correspondant au filtrage spatial, on obtient la formulation matricielle suivante où γ est une constante :

$$V(t) = (A + \gamma I)^{-1}(V(t-1) - \nabla E_{image}(V(t-1)))$$

avec A la matrice suivante :

$$A = \begin{pmatrix} a & b & c & & & & c & b \\ b & a & b & c & & & & c \\ c & b & a & b & . & & & \\ & c & b & . & . & . & & \\ & & . & . & . & b & c & \\ & & & . & b & a & b & c \\ c & & & & c & b & a & b \\ b & c & & & & c & b & a \end{pmatrix} \quad \text{avec} \quad \begin{cases} a = 2\alpha + 6\beta \\ b = -\alpha - 4\beta \\ c = \beta \end{cases}$$

La solution correspond à la situation de stabilité du vecteur V. Cette stabilité est détectée par un critère d'arrêt correspondant à la variation maximale admissible du vecteur V entre 2 itérations pour que l'on puisse considérer la solution comme ayant convergé. Cette approche correspond à une résolution de type programmation dynamique [AWJ90] bien plus efficace d'un point de vue coût de calcul que l'algorithme du snake glouton. Toutefois, une telle résolution n'est envisageable que dans le cas de fonctionnelles énergétiques assez simples ou bien dans des cas plus complexes moyennant des simplifications.

10.4.4 Avantages et inconvénients

Un des avantages principaux des snakes, inhérent à la méthode, est la régularité des contours. Cette contrainte conduit à un autre avantage : les snakes sont capables de segmenter de manière robuste les contours même lorsqu'ils sont flous [CKS95] [ZY96] [KLM94]. Les inconvénients des méthodes de segmentation à base de contours et surfaces actives sont principalement les suivants :

– L'initialisation doit être faite avec un *a priori* sur l'image. Les contours actifs sont faits pour s'*accrocher* à un objet. Si ils sont placés à l'extérieur d'un objet et loin de celui-ci, ils auront beaucoup de mal à le rejoindre et à l'englober. Il est donc nécessaire d'avoir une connaissance préalable sur l'image afin d'utiliser les méthodes de segmentation par contours actifs. Des exemples d'applications typiques sont la détection du contour des lèvres d'un orateur [Eve03] ou la segmentation et le suivi d'objets bien définis dans des images videos [Pre04].
– Les contours actifs ne permettent que d'isoler un petit nombre d'objets de l'image. Une sélection dépendant de l'algorithme utilisé par l'utilisateur est donc implicitement effectuée préalablement à la segmentation. Comme indiqué précédemment, un tel algorithme ne peut être utilisé pour segmenter une image quelconque ayant des caractéristiques inconnues ou mal définies. Une segmentation de type systématique et statistiquement fiable ne peut être envisagée dans la mesure où, intrinsèquement, l'algorithme élimine la plupart des régions dès son initialisation.

11

Segmentation d'images sur rétines artificielles

Le tour d'horizon des méthodes de segmentation effectué précédemment montre leur diversité ainsi que leurs atouts et leurs inconvénients. Ce chapitre permet de brièvement examiner quelles sont les techniques les mieux adaptées à la segmentation d'image sur rétines artificielles ayant des capacités régionales. En outre, des adaptations ou améliorations des techniques existantes dans le cadre d'une utilisation sur rétines artificielles ayant des capacités régionales sont proposées.

11.1 Amélioration de la ligne de partage des eaux par fusion statistiquement pertinente de régions

L'introduction des mécanismes cellulaires asynchrones dans les rétines artificielles asynchrones vise à implanter efficacement les algorithmes régionaux de moyen niveau. La taille des régions considérées peut, de plus, varier beaucoup dans la mesure où les primitives asynchrones de calcul régional proposées permettent d'effectuer des opérations de manière récurrente au sein d'algorithmes itératifs faisant évoluer des populations de régions dont les tailles peuvent changer.

La ligne de partage des eaux, qui se prête bien aux traitement parallèles sur des régions sera utilisée comme point de départ de notre méthode de segmentation, bien qu'elle conduise à une sur-segmentation causée, entre autres, par le bruit de l'image. La segmentation par découpage ou regroupement spatial, dans un cas synchrone repose sur des tessellations de formes régulières. Ces formes régulières qui ne correspondent pas aux formes réelles des régions conduisent à l'utilisation d'opérations de *split and merge*, ces dernières permettent de s'approcher itérativement de la forme réelle des régions, ce qui pourrait être évité en utilisant directement des régions de forme et de taille adaptées aux données à traiter. Dans ce cas, très bien adapté aux rétines artificielles ayant des capacités régionales, il est possible

de n'utiliser que des opérations de type fusion de régions.
Compte tenu de ces considérations, les méthodes de segmentation par équations aux dérivées partielles, sous leur forme traditionnelle utilisant des opérateurs locaux (qui deviennent régionaux par itération), ne peuvent guère tirer parti des opportunités offertes par les opérateurs régionaux. Nous verrons cependant à la section suivante que les équations aux dérivées partielles peuvent être combinées aux mesures régionales pour améliorer l'efficacité de la segmentation.

Ci-dessous, nous utilisons une méthode de segmentation basée sur la ligne de partage des eaux pour initialiser les régions de l'image, puis les régions de taille et de forme quelconques sont fusionnées itérativement afin de corriger la sursegmentation de l'image. Le critère de fusion des régions est choisi selon une approche statistique [GB04]. Il repose sur l'extraction d'informations régionales à l'intérieur de chacune des régions. Cette méthode de segmentation, tirant parti des facilités offertes pour effectuer des calculs régionaux parallèles est lourde à mettre en oeuvre sur un ordinateur standard. Elle peut cependant y être simulée à des fins de validation.

11.1.1 Modèle statistique des régions d'une image

Afin d'identifier un critère raisonnable de fusion de région, considérons une région d'une image comme la réalisation d'un processus stochastique stationnaire, à énergie finie et ergodique sur la région considérée.
Ce modèle repose sur l'idée qu'un objet d'une image est perçu au travers de l'imageur comme la réalisation d'un processus stochastique. De fait, une région représentant un objet sera un ensemble homogène (spatialement) caractérisé par plusieurs grandeurs statistiques, en particulier la luminance moyenne.
Cette représentation est valable dans le cas où une région représente effectivement tout ou partie d'un objet. Dans le cas où une région couvre plusieurs objets, cette représentation est inexacte. En effet, différents objets ayant des caractéristiques statistiques variées peuvent composer une même région, ce qui viole l'hypothèse d'homogénéité spatiale.

Le modèle proposé est donc limité et la méthode de segmentation l'utilisant doit tenir compte de cette limitation. Chaque région de l'image ne devra donc appartenir qu'à un objet au maximum. Cette contrainte forte oriente donc notre méthode vers un algorithme de type fusion de région où la taille des régions augmente au fur et à mesure que les régions sont fusionnées. Les fusions ont lieu uniquement entre des régions ayant des caractéristiques statistiques proches et pouvant donc être des réalisations d'un même processus stochastique caractéristique d'un unique objet.
Pour savoir si les fusions de régions sont pertinentes ou non, nous considérons que deux régions sont similaires si chacune d'elle peut être la réalisation d'un même processus stochastique. Cette similarité est mesurée par une probabilité dépendant de l'écart entre les mesures statistiques propres à chacune des régions. Dans un premier temps, la moyenne des luminances sur les régions sera utilisée comme me-

sure statistique.

Considérons une région de l'image (ne correspondant qu'à un seul objet). L'échantillon ordonné par valeurs croissantes des réalisations du processus caractéristique de l'objet perçu par l'imageur, sur la région, est noté $(X_1, ..., X_n)$. La moyenne du processus stochastique caractéristique de l'objet perçu est notée m et son écart-type σ.
Nous définissons sur l'échantillon ordonné $(X_1, ..., X_n)$, la moyenne arithmétique échantillon $\bar{X}$. Ces mesures sont intéressantes car la moyenne est un opérateur régional simple et très utilisé. Il est connu que :

$$E(\bar{X}) = m$$

$$\sigma^2(\bar{X}) = \frac{\sigma^2}{n}$$

Le théorème central limite permet d'écrire :

$$\frac{\bar{X} - m}{\frac{\sigma}{\sqrt{n}}} \rightarrow \mathcal{N}(0, 1)$$

où $\mathcal{N}$ est une distribution de probabilité gaussienne.

11.2 Critère de fusion de régions statistiquement pertinent

On constate que les distributions des moyennes sur des échantillons prélevés sur une région ont une dispersion inversement proportionnelle à $\sqrt{n}$. Ainsi, la fiabilité de l'estimation de la moyenne d'un ensemble de points dépend du nombre de points de l'ensemble considéré. Plus une région est grande, plus on connaît précisément sa moyenne, et plus une région est petite, plus cette moyenne est imprécise. La valeur réelle de la moyenne considérée se trouve donc dans un intervalle centré autour de la valeur de la moyenne mesurée, et dont la largeur dépend de la taille de la région. Cet intervalle I, appelé intervalle de confiance dans la littérature est de type : $I = [\ \bar{X} - \frac{K}{\sqrt{n}}\ ;\ \bar{X} + \frac{K}{\sqrt{n}}\ ]$, avec K le niveau de confiance que l'on souhaite. Ce niveau de confiance permet d'évaluer la probabilité pour que la valeur réelle de la moyenne soit dans l'intervalle défini. Par exemple, un K élevé conduit à une probabilité forte que la moyenne réelle soit dans l'intervalle I. Il est à noter que K dépend en réalité de l'écart-type de la distribution, ce qui n'est pas pris en compte ici. Une extension de cette méthode utilisant la valeur de l'écart-type sera proposée dans le chapitre sur la segmentation sociétale.

Ces intervalles de confiance sont utilisés pour déterminer si deux régions peuvent être des réalisations d'un même processus stochastique afin de pouvoir les fusionner. Pour cela, nous regardons l'intersection des intervalles de confiance pour K fixé. Si

cette intersection est vide, alors la probabilité que les moyennes des deux ensembles considérés soient les mêmes est faible. Si l'intersection est non vide, la probabilité pour que les deux régions soit les réalisations d'un même processus est plus élevée, et la fusion est plus pertinente. La valeur de K permet d'ajuster le critère de fusion, plus K est grand, plus il y aura de fusions réduisant la sur-segmentation, mais également moins ces fusions seront pertinentes. En effet, la largeur de l'intersection des intervalles de confiance est une fonction croissante de K. Pour un K très faible, les intervalles de confiance sont de petites tailles, il n'y a donc qu'un nombre limité de fusions, ce qui limite la correction de la sur-segmentation.

Dans le cas où il serait possible de fusionner une région avec plusieurs de ses voisines, on choisi de fusionner les régions ayant la plus grande probabilité d'être identiques, c'est à dire, celles dont l'intersection des intervalles de confiance est la plus grande. Par exemple, si l'on a le choix entre fusionner une région avec deux autres, une grande et une petite, ayant la même moyenne arithmétique, la fusion sera plus pertinente avec la petite car son intervalle de confiance inclus celui de la grande région. Cette méthode permet de s'assurer que la fusion s'effectue en priorité sur les régions les plus similaires.

Soient R_1 et R_2, deux régions à fusionner, soient M_1 et M_2, les moyennes respectives de ces deux régions échantillonnées, et soient n_1 et n_2, le nombre de points respectifs de chaque région. L'estimation de la moyenne de chacun des ensembles nous donne les intervalles de confiance suivants :

$$I_1 \;=\; [\; M_1 - \frac{K}{\sqrt{n_1}} \;;\; M_1 + \frac{K}{\sqrt{n_1}} \;]$$

$$I_2 \;=\; [\; M_2 - \frac{K}{\sqrt{n_2}} \;;\; M_2 + \frac{K}{\sqrt{n_2}} \;]$$

Supposons que $M_1 < M_2$. On a donc une intersection $I_1 \cap I_2$ non vide si :

$$M_1 + \frac{K}{\sqrt{n_1}} \;>\; M_2 - \frac{K}{\sqrt{n_2}}$$

Soit :

$$M_2 - M_1 \;>\; \frac{K}{\sqrt{n_1}} + \frac{K}{\sqrt{n_2}}$$

Si la région R_1 est de grande taille par rapport à la région R_2, on a alors :

$$(M_2 - M_1)\sqrt{n_2} \;>\; K$$

Le critère de fusion pour savoir s'il est pertinent de fusionner une région de petite taille avec une grande région est donc fonction de l'écart entre les moyennes de 2 régions, et de la racine carrée de la taille de la petite région considérée.

Le critère de fusion présenté fait appel à des comparaisons entre régions adjacentes.

L'implantation rétinienne massivement parallèle de cette méthode est cependant peu performante car elle est entravée par le besoin de séquentialiser les comparaisons entre les indicateurs statistiques propres à chaque région.
Il est préférable d'utiliser une méthode n'utilisant pas ce type de comparaisons inter-régionales.

11.2.1 Critère de pertinence d'une région

La nécessité de séquentialiser les données posant des problèmes pratiques d'implantation et limitant les performances de notre algorithme, essayons à présent de nous en passer en utilisant un algorithme de type différent mais toujours basé sur l'utilisation de critères statistiques. Dans la suite, les régions seront modélisées comme précédemment. De manière à être compatible avec le modèle proposé, notre algorithme sera donc toujours de type fusion de régions.
Nous avons vu précédemment que deux régions peuvent être fusionnées si leurs caractéristiques statistiques sont proches. Cette approche nous a conduis à réunir des régions faisant partie d'un même objet, la segmentation initiale étant effectuée à l'aide d'un algorithme de ligne de partage des eaux. Cette segmentation initiale aurait pu être faite à l'aide de n'importe quel autre algorithme de segmentation à la condition que celui-ci sépare les objets distincts. Dans cette seconde méthode de segmentation, nous essayons de tirer parti des caractéristiques de la segmentation effectuée par un algorithme de type ligne de partage des eaux.

L'utilisation de la LPE aboutit à une sur-segmentation dans laquelle les régions segmentées correspondent réellement à tout ou partie d'un objet de l'image, ou correspondent à un bassin versant créé par un bruit dans l'image. Le premier type de région nous intéresse car il correspond à la partie utile de l'information portée par la segmentation, le second type doit être éliminé et rattaché à des régions correspondant à des objets. C'est ce que nous faisons dans la méthode de segmentation présentée.

Afin de ne garder que les régions correspondant à des objets, il est nécessaire de déterminer si une région de l'image est pertinente ou non, c'est à dire, savoir si elle correspond à tout ou partie d'un objet, ou à du bruit. La grandeur statistique utilisée est interne à la région, il s'agit de la profondeur des bassins versants générés par la segmentation à base de LPE, nous l'appellerons profondeur de la région. A chacune des régions segmentées correspond une profondeur h, différence entre le minimum de la valeur du gradient sur le bord de la région, et le minimum de la valeur du gradient dans la région.
Un bassin correspondant à du bruit aura une profondeur faible tandis qu'un bassin correspondant à une région de l'image aura une profondeur significative (les bords de la région correspondent dans ce cas à des zones de l'image ayant un fort gradient tandis que le centre correspond à des zones homogènes de faible gradient). La profondeur peut donc être utilisée comme indicateur de la pertinence d'une région.
On retrouve *a priori* un critère similaire à celui utilisant la différence de moyenne

entre deux bassins, à la différence près que ce critère ne dépend pas des régions voisines, ce qui simplifie son implantation algorithmique.

La valeur du gradient mesuré en un pixel donné peut avoir deux origines : la présence réelle d'un gradient de luminance ou la présence de bruit dans l'image au voisinage de ce pixel. Sur les bords des régions, le gradient observé a de grandes chances de provenir du gradient de luminance. Il n'en est pas nécessairement de même au centre des régions dans les zones homogènes de l'image où le gradient est alors plutôt dû au bruit de l'image.
Ce qui nous intéresse pour savoir si un bassin versant est significatif ou non, est la profondeur du bassin versant qui serait généré si la discrétisation de l'image ne réduisait pas le nombre d'informations utilisables. Pour cela, estimons la profondeur réelle du bassin versant à partir des observations que l'on peut effectuer sur une image unique. Évaluons les valeurs du minimum du gradient sur la région et du minimum du gradient sur la frontière de la région. Chacune de ces valeurs est comprise dans un intervalle, et la profondeur réelle de la région sera au moins égale à la différence entre la borne inférieure du minimum du gradient sur la frontière et la borne supérieure du minimum du gradient sur la région.

Dans la suite, nous nous intéressons à la formalisation de cette idée. L'écart entre le modèle mathématique et statistique utilisé et la réalité étant important, les développements proposés doivent être considérés comme un fil conducteur destiné à orienter la réflexion et non pas comme un ensemble de certitudes inébranlables.

Nous utilisons le modèle des régions introduit précédemment. La luminance d'une région est considérée comme la réalisation d'un processus stochastique stationnaire, il en sera donc de même du gradient de la luminance dans les zones homogènes de la région. Dans les zones frontalières, la luminance n'est pas homogène et le gradient correspond à une différence entre les valeurs correspondant aux réalisations de processus stochastiques propres à chacune des régions. Le gradient obtenu peut donc être également considéré sur toute la longueur de la frontière commune aux deux régions comme la réalisation d'un processus stochastique stationnaire, ergodique et à énergie finie.
Finalement, la distribution du gradient sur la région peut être modélisée comme étant la réunion des réalisations de deux processus stochastiques, le premier correspondant aux pixels situés à l'intérieur de la région, et le second correspondant aux pixels situés à la frontière de la région. En réalité, pour former la frontière, il existe autant de processus stochastiques sous-jacents que de régions voisines de la région considérée, mais nous nous intéressons uniquement à la borne minimale du gradient sur la frontière, ce qui permet de ne s'intéresser qu'au seul processus ayant engendré la portion de frontière contenant cette borne minimale.

Supposons que les distributions des gradients à l'intérieur de la région et sur la frontière ont une loi de probabilité uniforme. A l'intérieur des régions, cette hypothèse est raisonnable compte tenu du fait que les régions sont construites de

manière à ce qu'elles soient homogènes. Sur les frontières, cette hypothèse repose sur le fait que les contours des régions soient placés correctement. En pratique, il y a parfois de petits décalages, mais l'utilisation de l'algorithme de ligne de partage des eaux comme segmentation initiale nous permet d'être quasiment sûrs que les gradients maximaux soient placés sur la frontière des régions. Nous noterons les réalisations des processus stochastiques associés aux distributions, et rangées du plus petit élément au plus grand, $[\ X_{1_{int}}\ ;\ X_{n_{int}}\ ]$ et $[\ X_{1_{bord}}\ ;\ X_{p_{bord}}\ ]$, où n_{int} et p_{bord} sont respectivement le nombre de pixels de l'intérieur de la région et de la frontière de la région.

La valeur du minimum réel (et non celui qui est observé) de chacune de ces deux distributions peut être encadrée par un intervalle. Après renormalisation, l'étude de la convergence du minimum nous donne des intervalles de la forme :

$$I_{int} \ = \ [\ X_{1_{int}} - \frac{K\ (X_{n_{int}} - X_{1_{int}})}{n_{int} - K}\ ;\ X_{1_{int}}\ ]$$

$$I_{bord} \ = \ [\ X_{1_{bord}} - \frac{K\ (X_{p_{bord}} - X_{1_{bord}})}{p_{bord} - K}\ ;\ X_{1_{bord}}\ ]$$

I_{int} est l'intervalle dans lequel se situe le minimum du gradient à l'intérieur de la région. I_{bord} est l'intervalle dans lequel se situe le minimum du gradient sur la frontière de la région. K, très inférieur à p_{bord} permet de définir l'incertitude sur le fait que le minimum réel soit dans l'intervalle définit (une valeur de $K = 3$ donne une incertitude de 5%). On constate que la borne supérieure de chacun de ces intervalles est le minimum observé et que la borne inférieure dépend du nombre de pixels de la région considérée. Plus ce nombre de pixels est grand, plus l'intervalle est réduit, ce qui est tout à fait normal puisqu'on a plus d'informations sur la distribution et en particulier sur son minimum.

Ces intervalles permettent de minorer la hauteur de chaque bassin versant, cette minoration est comme nous l'avons déjà dit la différence entre la borne inférieure du gradient sur les frontières moins la borne supérieure du gradient au centre de la région :

$$H_{min} = X_{1_{bord}} - \frac{K\ (X_{p_{bord}} - X_{1_{bord}})}{p_{bord} - K} - X_{1_{int}}$$

Les régions sont considérées comme pertinentes si la minoration H_{min} de la profondeur de leurs bassins versants est supérieure à une constante Δ que l'on pourra choisir en fonction du niveau de bruit moyen de l'image. On a donc finalement le critère suivant :

$$X_{1_{bord}} - X_{1_{int}} - \frac{K\ (X_{p_{bord}} - X_{1_{bord}})}{p_{bord} - K} > \Delta$$

Le critère de pertinence d'une région est donc fonction de sa profondeur mesurée

($X_{1_{bord}} - X_{1_{int}}$), de la taille de la frontière (p_{bord}), et de l'étendue des valeurs de la frontière ($X_{p_{bord}} - X_{1_{bord}}$). On constate que plus la région est grande, alors plus la frontière est longue et plus la hauteur du bassin versant mesurée est fiable. Dans le cas de petites régions, le nombre de pixels de la frontière est réduit, il faut alors que la profondeur du bassin versant mesurée soit grande pour que celui-ci soit conservé (car sa profondeur mesurée est peu fiable). A l'opposé, un bassin peu profond (mais dont la profondeur observée est supérieure à Δ) sera conservé si le nombre de pixels constituant sa frontière est élevé car dans ce cas, la profondeur observée est le terme dominant dans le critère.
Ce critère permet de déterminer les régions pertinentes de l'image à l'aide de mesures régionales (par exemple la valeur minimum du gradient sur une région, ou le nombre de pixels composant cette région). Cette détermination peut être effectuée de manière massivement parallèle sur l'ensemble des régions. Un tel critère peut être efficacement utilisé sur une rétine ayant des capacités régionales. Nous utilisons dans la suite un algorithme permettant de réduire la sur-segmentation due à la ligne de partage des eaux.

11.2.2 Algorithme parallèle d'élimination de régions

Un algorithme de segmentation de type *merge* sur des régions de taille et de forme variables conduisant à une segmentation de l'image est présenté ici. Cette segmentation se compose de 2 phases, un *split* initial de l'image grâce à un algorithme de ligne de partage des eaux, et une phase itérative de type *merge* permettant de regrouper les régions selon un critère de pertinence.

Initialisation

L'initialisation de l'algorithme est effectuée par un algorithme de ligne de partage des eaux sur l'image gradient obtenue par application d'un filtre de Sobel horizontal et d'un filtre de Sobel vertical. Cette initialisation permet de segmenter l'image de manière à obtenir des bassins versants séparés par les lignes de gradient maximal. Cette manière de segmenter conduit à une sur-segmentation, en particulier due au bruit dans l'image, mais elle permet d'être certain d'avoir segmenté les régions qui doivent l'être. Le fonctionnement de l'initialisation sur un profil de gradient à une dimension est proposé aux figures 11.1 et 11.2.

Fusion des régions

La fusion des régions n'est pas une opération de fusion au sens réunion de plusieurs régions, elle s'effectue automatiquement par augmentation de la taille de régions existantes ou par création de nouvelles régions.
A l'issue de l'initialisation par ligne de partage des eaux sur l'image gradient, l'image est segmentée en régions de tailles et de profondeurs différentes. La première phase de l'algorithme de fusion consiste à trier les régions pertinentes et à traiter celles qui ne le sont pas. Une région est considérée comme pertinente si le critère

FIG. 11.1 – Profil d'un gradient à 1 dimension

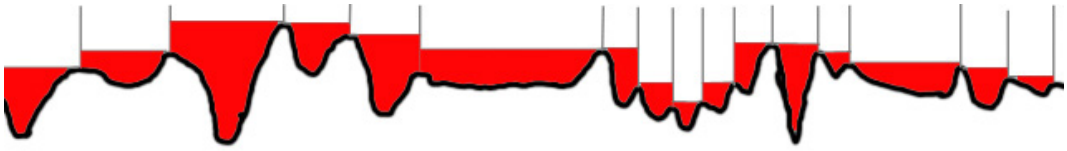


FIG. 11.2 – Ligne de partage des eaux sur un profil à 1 dimension

présenté précédemment est vérifié. Les régions pertinentes de la figure 11.2 sont

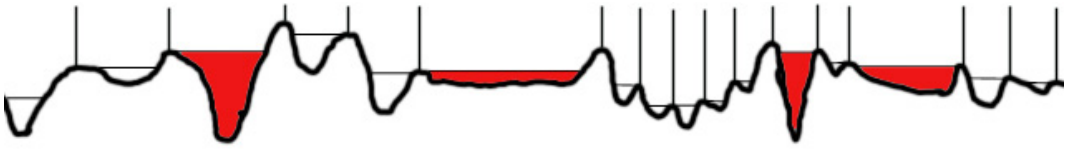


FIG. 11.3 – Régions pertinentes (en rouge)

représentées en rouge à la figure 11.3. Dans ce cas elle est conservée, dans le cas contraire, elle est supprimée et nivelée. La valeur du gradient en chacun de ses pixels est donc remplacée par la valeur minimale du gradient sur les pixels frontières de cette région (fig. 11.4) :

$$\nabla I_{Region} \;=\; Min_{Bords}(\nabla I)$$

Le bassin versant devient un plateau. Les pixels appartenant à la région supprimée redeviennent libres et disponibles pour être ajoutés à une région existante ou à une nouvelle région. A la fin de cette phase de tri, restent les régions qui satisfont au critère de tri, et un ensemble de pixels libres n'appartenant à aucune région dont les gradients ont été nivelés (fig. 11.4).

La deuxième phase de l'algorithme consiste à étendre les régions existantes en leur ajoutant les pixels libres qui répondent à la condition $\nabla I_{Pixellibre} > \nabla I_{Pixel\ region}$, condition utilisée pour ajouter des points à une région dans la ligne de partage des eaux. Une fois les régions agrandies, restent des pixels formant des plateaux qui

FIG. 11.4 – Profil après nivellement des régions non pertinentes

n'appartiennent à aucune région. Ces pixels sont à leur tour réunis en régions par un algorithme de ligne de partage des eaux appliqué uniquement aux pixels libres. A la fin de cette seconde phase, on obtient un ensemble de régions constitué par les régions qui avaient été triées et qui ont augmenté en taille, et les régions qui ont été créées à partir des pixels libérés lors de la première phase (fig. 11.5). Le nombre de

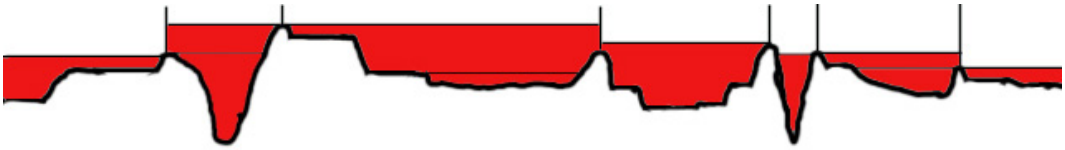


FIG. 11.5 – Profil après la seconde ligne de partage des eaux

régions présentes à la fin de la deuxième phase est nécessairement moins grand que le nombre de régions présentes à la fin de l'initialisation car certaines régions ont augmenté en taille et les régions recréées au cours de la deuxième phase ont une taille supérieure à celles qui ont été supprimées (à cause du nivellement)(fig.11.6).

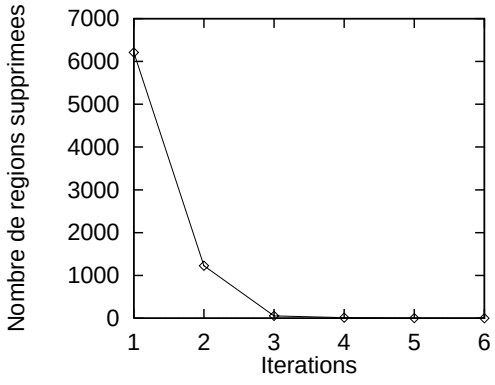
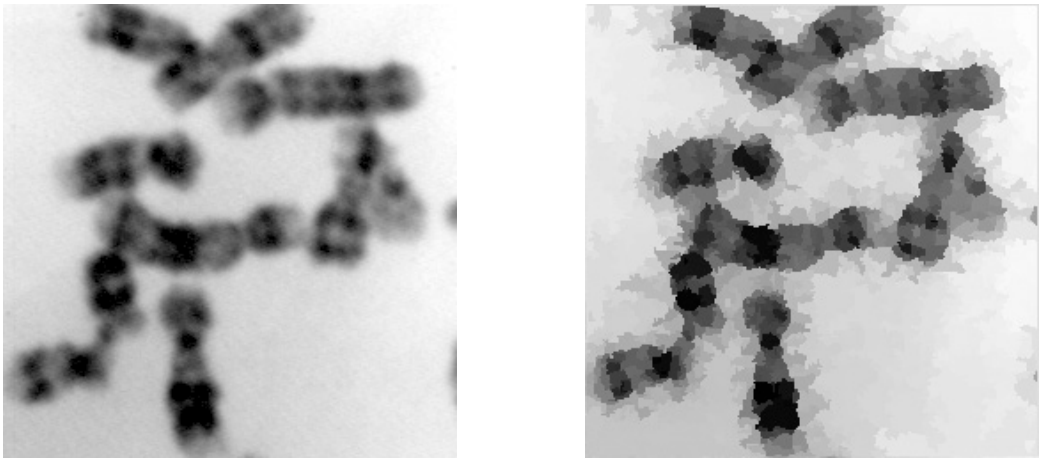


FIG. 11.6 – Nombre de régions supprimées à chaque itération.

En effectuant de manière séquentielle la phase 1 et la phase 2, on réduit itérativement le nombre de régions présentes dans l'image. Lorsque toutes les régions

de l'image satisfont le critère de tri, alors la segmentation n'évolue plus et l'algorithme est terminé. On constate expérimentalement que la convergence vers l'image finale s'effectue très rapidement, le nombre de régions supprimées lors de la phase 1 décroît exponentiellement avec le nombre d'itérations.



(a) Image originale (b) Segmentation

FIG. 11.7 – Exemple de segmentation d'une image médicale.

Terminaison de l'algorithme

Lorsque le nombre de régions supprimées lors de la phase 1 devient nul, chacune des régions de l'image satisfait le critère de pertinence des régions. La dernière phase consiste à retourner l'image segmentée, pour cela, sur chacune des régions de l'image est effectuée la moyenne des intensités des pixels. Cette moyenne vient remplacer la valeur de l'intensité dans chacun des pixels de la région.
Bien que n'utilisant pas la notion d'énergie, la méthode de segmentation proposée permet de retrouver les avantages des méthodes basées sur l'énergie [CKS95] [ZY96] [KLM94] dans les régions, en particulier l'aptitude à séparer des régions très floues (cf. fig 11.7). De plus, il permet d'assurer la pertinence de chacune des régions présentes dans l'image segmentée, ce qui n'est pas le cas dans les autres méthodes de segmentation. Notre algorithme présente toutefois un inconvénient, le découpage des régions est net sur les frontières des régions ayant des écarts de luminosité importants. En revanche, sur les régions assez homogènes, les formes obtenues suivent le découpage effectué lors de l'initialisation et qui est dû principalement au bruit dans ces zones, ce qui leur donne un aspect fractal qui n'est cependant pas forcément gênant.

11.3 Vers une nouvelle méthode de segmentation : la segmentation sociétale

L'aspect tortueux des contours des régions obtenues grâce à l'utilisation d'une segmentation initiale à l'aide d'un algorithme de ligne de partage des eaux nous a motivés pour tenter de trouver un nouvel algorithme de segmentation ayant une initialisation plus régulière. De plus, ayant à disposition des outils d'extraction de grandeurs régionales performants, ce nouvel algorithme en fera un usage intensif.
Comme nous l'avons vu dans l'état de l'art sur la segmentation, les méthodes de segmentation basées sur les équations aux dérivées partielles permettent d'obtenir des contours réguliers mais n'utilisent pas les potentialités offertes par les opérateurs régionaux. Afin de tirer parti de ces potentialités tout en conservant les avantages des méthodes existantes, une méthode de segmentation utilisant les équations aux dérivées partielles couplées à des mesures locales et régionales est proposée.

La segmentation par ligne de partage des eaux a été inspirée par analogie avec la séparation de l'eau de pluie dans différents bassins versants. Nous utilisons dans la méthode de *segmentation sociétale* [GB05b], une autre analogie. Il s'agit de la segmentation d'un territoire en villages. Supposons en effet que l'on place sur un territoire géographique à un instant initial une personne par unité de terrain puis qu'on laisse évoluer cette population. En fonction du relief de la région, la population va avoir tendance à se regrouper en zones d'influences appelées villages. La *segmentation sociétale* procède de la même manière en considérant que l'image en niveaux de gris est un relief sur lequel on fait évoluer une population.

11.3.1 Analogies

Qu'est-ce qu'un territoire géographique ? C'est un ensemble de parcelles de terrain ayant chacune leur propre altitude.
Qu'est-ce qu'une image en niveau de gris ? C'est un ensemble de pixels ayant chacun leur propre luminance.
Qu'est-ce qu'un village ? C'est un sous-ensemble de parcelles adjacentes ayant des caractéristiques géographiques proches.
Qu'est-ce qu'une région ? C'est un ensemble de pixels adjacents ayant des caractéristiques de luminance proches.
Partant de ces considérations, l'analogie entre la segmentation d'une image en régions et la segmentation d'un territoire en villages devient claire en considérant que :

1. Une image est équivalente à un territoire.
   Une région d'une image est équivalente à un village.
2. Un pixel est équivalent à une parcelle de terrain.
   L'altitude est équivalente à la luminance.

Comme indiqué dans l'introduction, un village est en fait une zone d'influence. L'influence n'est pas la même à tout endroit d'un village. Nous considérons que

l'influence est proportionnelle à la densité de population, cette influence sera plus grande au centre d'un village qu'à sa périphérie. Avec cette définition, la densité de population peut être interprétée comme une grandeur proportionnelle à la probabilité d'appartenance au village d'une parcelle de terrain. Par analogie, la notion de population est également utilisée dans notre méthode de segmentation.

11.3.2 Principe et formalisation

Nous présentons à présent le principe de la segmentation sociétale d'un territoire en villages.

Afin d'enlever tout *a priori* sur la distribution géographique des habitants à l'initialisation de l'algorithme, nous initialisons la population du territoire en plaçant une personne dans chaque parcelle de terrain. De plus, la forme des parcelles de terrain est considérée carrée, de manière à couvrir le territoire à l'aide d'une maille 4-connexe.

La question principale se posant concernant l'évolution de la population est : comment est-ce que les villages se forment ? Nous considérons dans ce modèle que la formation des villages résulte de la croissance de la population, des regroupements de villages et des conflits entre villages. Chacun de ces aspects de l'évolution d'un village peut être décomposé en plusieurs règles, que nous répertorions avec la double formulation suivante :

- Signification de la règle d'un point de vue segmentation de territoire.
- Formalisation de la règle à l'aide d'une équation aux dérivées partielles ou d'une règle d'évolution.

Nous utiliserons les notations suivantes :

- x est l'altitude d'une parcelle de terrain. Elle reste bien évidemment constante durant toute la durée de la segmentation
- $\overline{x}$ est la valeur moyenne de l'altitude x sur une région.
- x_b est l'altitude d'une parcelle située à la frontière entre deux villages.
- p est la population résidant en une parcelle de terrain donnée.
- p_b est la population d'une parcelle de terrain située à la frontière entre deux villages.
- S est la surface d'un village.

– C est le périmètre d'un village.

– Les opérateurs max_{reg} et min_{reg} sont respectivement les opérateurs maximum et minimum définis sur une région reg.

– Les a_n et les K_n sont des constantes.

Croissance de la population

Après avoir défini les notations utilisées, nous définissons les règles de croissance de la population. Ces règles de croissance de la population incluent les mouvements de population internes à un village.

L'évolution de la population dans chacune des parcelles composant un village est le résultat de la mise en oeuvre de plusieurs règles ayant pour effet de modifier la population de la parcelle considérée en fonction de paramètres propres à la parcelle mais aussi au village. L'évolution de la population d'une parcelle de terrain peut se mettre sous la forme de l'équation aux dérivées partielles suivantes :

$$\frac{\partial p}{\partial t} = \delta_1 + \delta_2 + \delta_3 + \delta_4$$

où δ_1, δ_2, δ_3 et δ_4 sont les variations de population dues aux différentes règles d'évolution de la population. Ces règles de croissance et de déplacement de population sont à présent détaillées :

– **Possibilité de communiquer localement** : Cette règle locale permet d'exprimer le fait que la population a plus de chances de grandir dans un endroit plat que dans un endroit ou le gradient d'altitude est localement élevé.

$$\delta_1 \;=\; a_1 \; p \; (1 - \frac{|\nabla x|}{a'_1}) \tag{11.1}$$

– **Homogénéité et taille d'un village** : Cette règle globale permet d'exprimer le fait qu'un village de grande taille voit sa population se développer plus rapidement car les possibilités de rencontres sont plus importantes. Ces possibilités de rencontres sont également influencées par l'homogénéité du village. Par exemple, dans le cas d'un village ayant une altitude maximale très éloignée de son altitude minimale, l'effort à faire pour se rencontrer est important, ce qui tend à réduire la vitesse de croissance de la population.

$$\delta_2 \;=\; p \; (a_3 \log S + a_4 \; (1 - \frac{(max_{reg}(x) - min_{reg}(x))}{a'_4})) \tag{11.2}$$

– **Possibilité de communication avec le centre du village** :Cette règle mixte locale-globale prend en compte l'effort à faire pour se rendre d'une parcelle de terrain jusqu'au centre du village, où l'on considère qu'ont lieu les

échanges. Ainsi les habitants d'une parcelle située en altitude par rapport au centre d'un village auront un effort à faire plus important que ceux qui sont situés à l'altitude du centre du village. Il eut paru plus logique de prendre en compte le vrai coût de déplacement (nombre de mètres montés et descendus pour aller de la parcelle de terrain au centre du village) plutôt que la différence des altitudes. Ces coûts sont égaux dans le cas d'un profil de terrain monotone. Pour des raisons de complexité de calcul, nous nous limiterons à la différence d'altitude.

$$\delta_3 \ = \ a_5 \ p \ (1 - \frac{|x - \overline{x}|}{a'_5}) \tag{11.3}$$

– **Diffusion de population** : Cette règle locale prend en compte le fait que si la population n'est pas localement homogène, alors les habitants ont tendance à se déplacer dans un voisinage proche selon une équation de diffusion de type équation de la chaleur.

$$\delta_4 \ = \ a_6 \ \triangle p \tag{11.4}$$

Nous pouvons constater au travers de ces règles que la population est dans de nombreuses situations une fonction croissante du temps. Par itération, la population va diverger. Ceci n'est toutefois pas un problème pour notre modèle dans la mesure où la segmentation est terminée lorsque les regroupements de régions sont terminés (voir sous-section suivante). Toutefois, il serait possible de pallier à ce problème en ajoutant une règle faisant diminuer la population lorsqu'elle dépasse une certaine densité dans les parcelles de terrain.

Regroupements de villages

Outre la croissance de la population, la seconde cause de formation des villages est le regroupement de villages. Ces regroupements se produisent lorsqu'une conjonction de conditions est satisfaite. Les conditions à satisfaire sont les suivantes :

– **Condition de pertinence statistique des regroupements en fonction de l'écart d'altitude moyenne des villages et de leurs tailles** : Cette condition régionale permettant une fusion statistiquement pertinente est justifiée et utilisée au chapitre précédent et dans [GB04]. Elle permet de regrouper des villages en prenant en compte leurs tailles respectives et l'écart entre leurs altitudes moyennes. Deux grands villages doivent avoir une altitude moyenne très proche pour pouvoir être regroupés, deux villages ayant une grande différence d'altitude doivent être très petits pour être regroupés. Cette condition est une condition *sine qua non* car elle assure d'avoir des regroupements statistiquement pertinents.

$$(\overline{x}_2 - \overline{x}_1)\sqrt{max(S_1, S_2)} < K_1 \tag{11.5}$$

Cette première condition est affinée par deux conditions annexes :

– **Condition sur la taille relative de la frontière commune** : Cette condition utilisant des données globales et locales indique s'il est pertinent de regrouper des villages en considérant le quotient longueur de la frontière commune aux deux villages sur le périmètre village. Cette condition permet de refuser la fusion si ce rapport est trop faible, ce qui signifie que les deux villages sont en contact mais seulement sur une partie très réduite de leur territoire. Il est également possible de prendre le quotient de la longueur de la frontière commune par la racine carrée de la surface de la région comme condition de fusion. Toutefois, cette condition initialement choisie [GB05b] conduit à des erreurs de fusion dans le cas d'une région de type fractal où le périmètre et la longueur potentielle de la frontière commune sont largement supérieurs à la racine carrée de la surface.

$$\frac{length(x_b)}{C_1} > K_2 \text{ ou } \frac{length(x_b)}{C_2} > K_2 \tag{11.6}$$

– **Condition sur la hauteur de la frontière commune** : Cette condition utilisant des données globales et locales indique s'il est pertinent de regrouper des villages en fonction de la différence entre l'altitude moyenne de la frontière et l'altitude moyenne des villages considérés. Si cette différence est trop importante, cela signifie que les deux villages sont séparés par des montagnes, et qu'ils ne doivent pas être réunis même s'ils sont homogènes.

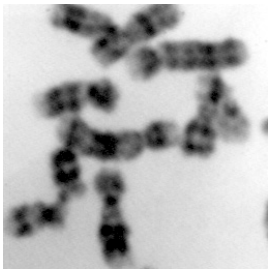
$$max\left((\overline{x}_b - \overline{x}_1), (\overline{x}_b - \overline{x}_2)\right) < K_3 \tag{11.7}$$

L'utilisation des conditions présentées se fait de la manière suivante. Les villages sont regroupés si la condition (11.5) est vérifiée et si au moins l'une des deux conditions (11.6) et (11.7) est vérifiée.

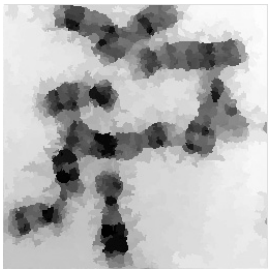
Conflits entre villages

Les règles de croissance de la population et de regroupements de villages présentées ci-dessus conduisent à une distribution de la population en régions. Grâce à leurs caractères locaux, régionaux et mixtes, les interactions existent entre chaque individu, la population d'un village et les villages voisins. Dans chaque village, la population se retrouve concentrée autour du centre, alors qu'elle est moins dense à mesure que l'on se rapproche des frontières. Comme nous l'avons déjà dit, la probabilité pour qu'une parcelle de terrain appartienne à un village peut être estimée par une quantité proportionnelle à sa densité de population. Cette propriété intéressante permet également de quantifier la pertinence d'une frontière.

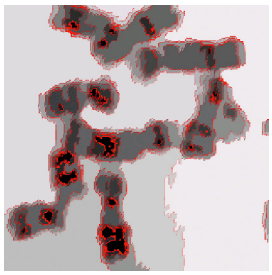
Cependant, concernant la pertinence des frontières, les règles déjà proposées aboutissent à un paradoxe. Les règles de croissance de la population conduisent à une population plus élevée sur les frontières localement plates que sur les frontières ayant un fort gradient. Pourtant, les frontières sont bien plus simples à définir



(a) Image originale



(b) Segmentation de type ligne de partage des eaux améliorée



(c) Segmentation sociétale

FIG. 11.8 – Segmentation sociétale de l'image chromosomes.

lorsque le relief est accidenté (montagneux) que lorsque le relief est plat (ce qui conduit typiquement à des frontières bruitées dans la segmentation par ligne de partage des eaux).
Ce paradoxe peut être levé si l'on ajoute une troisième loi d'évolution : les **conflits entre villages**. Dans les régions plates, le caractère incertain des frontières conduit à des conflits locaux pour le contrôle de cette frontière. De tels conflits vont décimer la population de part et d'autre de la frontière de la région. La diminution de densité de population qui en résulte réduit la pertinence de la frontière, ce qui lève le paradoxe.
L'équation (11.8) modélise les effets des conflits entre villages le long des frontières. Il faut noter que ces frontières, en cas de fusion, sont rapidement repeuplées par diffusion de la population :

$$\frac{\partial p_b}{\partial t} = - a_7 \, p_b \frac{|\nabla x|}{a'_7} \tag{11.8}$$

11.3.3 Algorithme de segmentation

L'algorithme, pour des raisons de cohérence avec la présentation qui a été faite des règles d'évolution, est présenté dans le cadre de la segmentation d'un territoire en villages. La segmentation d'une image en régions s'effectue de la même manière par analogie, il suffit donc au lecteur de garder en mémoire l'analogie présentée précédemment entre la segmentation d'image et la segmentation d'un territoire pour comprendre comment s'effectue la segmentation sociétale d'une image en régions.

L'initialisation de l'algorithme est réalisée de la manière la plus neutre possible en

considérant que la population de chaque parcelle de terrain est égale à 1 individu. Cette initialisation comparée avec une initialisation faite à l'aide d'un algorithme de ligne de partage des eaux, conduira à réduire les irrégularités dans les contours de la segmentation finale.

Les constantes K_n sont réglées en fonction de la taille et du contraste de l'image initiale, tandis que les constantes a_n peuvent faire l'objet d'un réglage automatique. Ce point sera abordé plus tard.
Le déroulement de l'algorithme est une itération des phases de croissance de la population, regroupements de villages et conflits entre villages. Cette itération converge de manière certaine, grâce à la k-idempotence du processus de regroupements de villages.
Durant la phase de croissance de la population, celle-ci évolue simultanément dans chacune des parcelles de terrain du territoire conformément aux 4 règles d'évolution présentée (équations 11.1, 11.2, 11.3 et 11.4). Dans une implantation rétinienne parallèle de l'algorithme de segmentation sociétale, les contributions respectives des différents facteurs d'évolutions sont calculées l'une après l'autre, mais de manière massivement parallèles sur tous les pixels et toutes les régions en même temps, ce qui conduit à une grande efficacité.

Durant la phase de regroupements de villages, les extractions de caractéristiques régionales utilisées par les conditions de regroupements sont également calculées en parallèle sur toutes les régions à l'aide d'opérateurs régionaux tels qu'ils ont été présentés dans les chapitres précédents. Concernant les tests des conditions de regroupements, ils peuvent être faits à l'intérieur de la rétine en déterminant avant d'effectuer les tests quelle est la région voisine la plus susceptible d'être fusionnée (le processus correspondant est décrit après). Cela permet de recopier les paramètres de cette région voisine candidate dans la région considérée, et dans un deuxième temps d'effectuer le test des conditions de regroupement en parallèle sur toutes les régions de l'image.
Pour déterminer quelle est la région la plus susceptible d'être fusionnée, il suffit de recopier les valeurs caractéristiques de régions voisines sur les pixels frontière de la région considérée puis d'effectuer une opération de type test booléen tel qu'un minimum ou un maximum. Par comparaison du résultat du test aux valeurs placées sur les frontières, on peut ensuite identifier la région étant le meilleur candidat à la fusion.
Ce mode opératoire nous permet de s'affranchir de l'usage d'un opérateur de haut niveau extérieur à la rétine, et de l'utilisation d'un graphe d'adjacence des régions. L'utilisation d'opérateurs de haut niveau permettrait toutefois d'effectuer des opérations plus complexes sur le graphe des régions de l'image en vue de la fusion. Cependant le prix à payer est celui d'un échange d'informations (le graphe d'adjacence des régions et les caractéristiques de celles-ci) entre la rétine et le système de gestion des opérateurs de haut niveau, ce qui a un coût temporel, énergétique et matériel.

11.4 Résultats

Les résultats de segmentation obtenus à l'aide de notre algorithme (présentés aux figures 11.8, 11.9, 11.12 et 11.21) sont de très bonne qualité comparés aux autres méthodes de segmentation que nous avons sélectionnées. La segmentation générée conserve les régions de petite taille mais significatives au sens de la pertinence statistique. Les régions de grande taille sont aussi très bien segmentées sans être découpées en morceaux. De plus, la segmentation obtenue donne une indication sur la pertinence des contours des régions (les contours sont représentés en rouge, la densité de rouge étant proportionnelle à leur pertinence). Comme espéré,

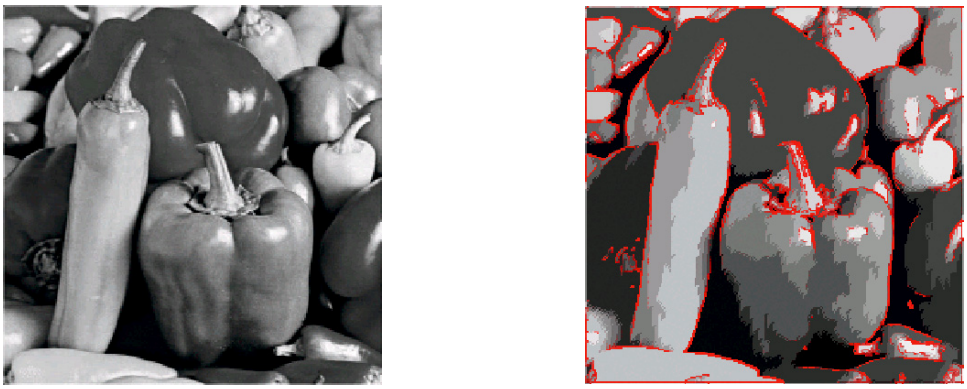
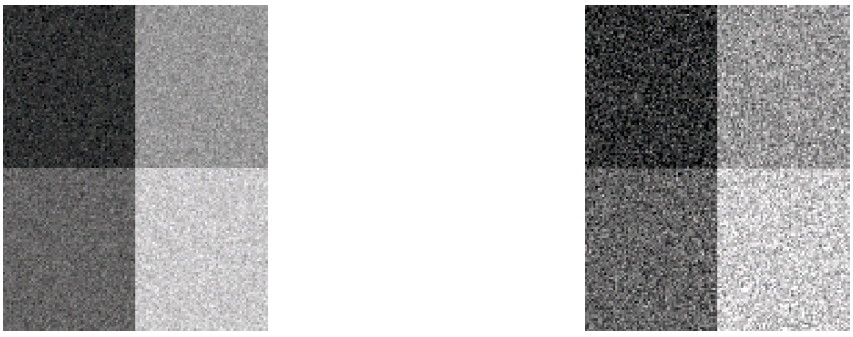


FIG. 11.9 – Segmentation sociétale de l'image "Poivrons".

nous constatons également que les contours sont plus lisses que ceux que l'on peut obtenir en utilisant des algorithmes de type ligne de partage des eaux.
Afin de quantifier la résistance au bruit et aux variations de paramètres de notre algorithme, nous utilisons une mire de référence composée de 4 rectangles de niveaux de gris différents auxquels on ajoute un bruit Gaussien d'écart-type σ (fig. 11.10). Ces images sont similaires à celles obtenues à l'aide de radar à synthèse d'ouverture (SAR) utilisées à des fins de tests dans [PWK00] et [ZY96].



(a) Niveau de bruit : $\sigma = 15$ (b) Niveau de bruit : $\sigma = 30$

FIG. 11.10 – Images de référence.

11.4.1 Robustesse au bruit

Le bruit dans les images conduit à une fausse et sur-segmentation de l'image si le bruit devient trop important (fig. 11.11). Ce problème peut être causé par le processus de regroupement de régions de l'algorithme. Une région ne peut être fusionnée qu'à une seule autre région à chaque itération. Dans le cas d'une image bruitée, après quelques itérations, de larges régions se sont formées, et au sein de ces larges régions, quelques toutes petites régions dues au bruit sont présentes. Le problème est que, à chaque itération, une et une seule de ces petites régions peut-être fusionnée avec la grande qui la contient, ce qui conduit à une perte de temps. Il serait plus efficace d'essayer de fusionner à chaque itération des régions ayant approximativement la même taille. Le même problème se pose dans d'autres méthodes de segmentation [PWK00].
Une solution pour y remédier est d'adapter le critère de fusion au bruit présent dans les régions considérées. Dans [GB04], nous regroupons deux régions si l'intersection des intervalles de confiance de leur luminance moyenne est non nulle. Ceci conduit à la condition de regroupement décrite à l'équation 11.5. Si l'on considère une région comme étant la réalisation d'un processus stochastique stationnaire, ergodique d'énergie finie, l'équation 11.5 est une conséquence du théorème central limite :

$$\frac{\bar{X} - m}{\frac{\sigma}{\sqrt{n}}} \rightarrow \mathcal{N}(0,1)$$

où n est le nombre de pixels de la région considérée, σ l'écart-type et $\mathcal{N}$ la distribution de probabilité Gaussienne. En conséquence, la constante K_1 utilisée dans (11.5) est proportionnelle à $\sigma_1 + \sigma_2$, où σ_1 et σ_2 sont les écarts-type mesurés sur chaque région. Ceci permet de proposer une nouvelle formulation de la première condition de fusion :

$$(\overline{x}_2 - \overline{x}_1)\sqrt{max(S_1, S_2)} < (\sigma_1 + \sigma_2)\ K_1' \tag{11.9}$$

Finalement, la première condition de fusion reformulée a un double sens. Elle permet de prendre en compte la pertinence statistique des regroupements en fonction de l'écart de luminance moyenne des régions et de leurs tailles respectives. Elle permet également de prendre en compte l'homogénéité des régions dans l'évaluation de cette pertinence statistique. Ainsi, pour une différence d'altitude moyenne fixée, il sera moins pertinent de fusionner deux régions très plates que de fusionner deux régions montagneuses. Le saut d'altitude moyenne est en effet plus visible dans le premier cas.
Dans cette solution au problème du bruit, la luminance des pixels est supposée avoir une loi de probabilité Gaussienne. Cette hypothèse est souvent une approximation assez grossière, mais nous l'acceptons dans la mesure où les régions de notre image sont assez homogènes par construction, ce qui permet de définir l'écart-type comme une mesure de la distribution des valeurs de la luminance autour d'une valeur moyenne caractéristique de la région. Dans le cas où le nombre de pixels

est inférieur à 10, cette hypothèse ne peut être vérifiée et la version initiale de la première condition de fusion n'utilisant pas l'écart-type du bruit est utilisée.

Les résultats de la prise en compte du bruit des régions dans la pertinence des regroupements de régions est présentée à la figure 11.11. Les losanges noirs présentent le taux d'erreur de classification en prenant en compte le bruit, les triangles rouges présentent le taux d'erreur de classification si l'on ne prend pas le bruit en compte. On constate que la robustesse est grandement améliorée si l'on prend en

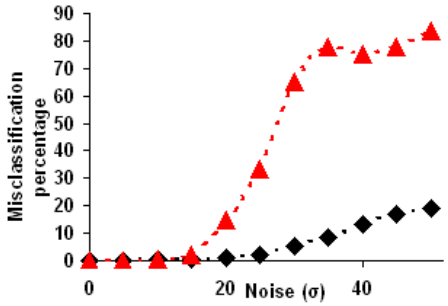


FIG. 11.11 – Influence du bruit de l'image sur le pourcentage d'erreur de classification.

compte le bruit dans les régions. La segmentation reste robuste pour de hauts niveaux de bruit : $\sigma = 50$ est très important dans une image codée sur 256 niveaux de gris. Cette propriété de robustesse permet de segmenter des images très bruitées telles que les images SAR. Un exemple de segmentation d'image SAR est proposé à la figure 11.12. L'image SAR utilisée est la même que dans [PWK00]. La segmen-

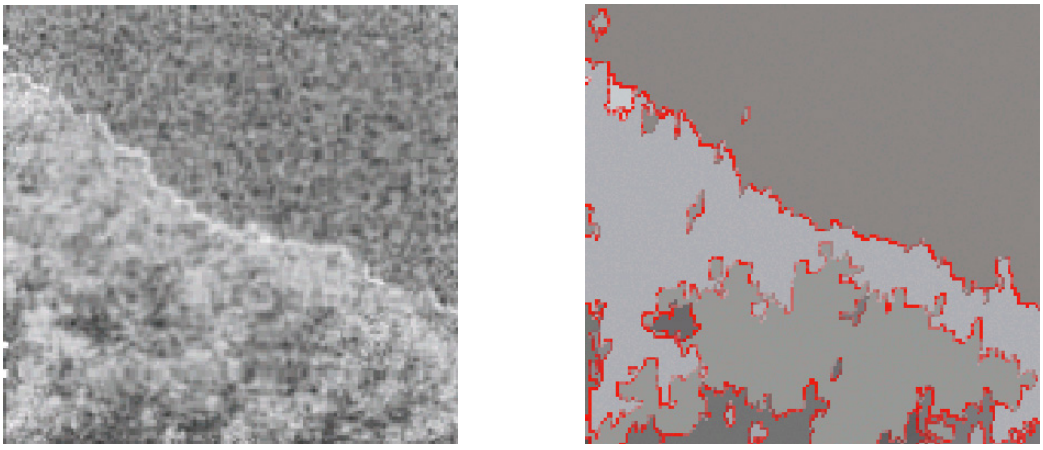


FIG. 11.12 – Segmentation sociétale d'une image SAR

tation effectuée sépare clairement et sans traitement supplémentaire les zones de mer, les zones de végétation et les zones de sable, ce qui est un résultat remarquable.

11.4.2 Robustesse aux variations de paramètres

Les règles de croissance de la population et les conditions de regroupement de régions utilisent un nombre important de constantes. La robustesse de l'algorithme aux variations de ces constantes est importante pour obtenir une bonne robustesse. Des tests menés sur un grand nombre d'images de référence (les tests présentés aux figures 11.16, 11.20, 11.18, 11.14, 11.15, 11.19, 11.17 et 11.13 ont été réalisées avec les mêmes paramètres afin de montrer la robustesse de la méthode de segmentation) et avec différents jeux de constantes ont montré que la majorité de ces constantes pouvaient être ajustées de manière très approximative. En effet, l'évolution de la population n'est pas utilisée en tant que telle dans le regroupement des régions, ainsi tous les coefficients qui lui sont liés n'ont pas d'influence sur le résultat de la segmentation, ils n'ont d'influence que sur la pertinence des frontière trouvées, ce qui est secondaire. Il reste donc trois constantes importantes, les constantes utilisées dans les conditions de regroupement. Les deux constantes correspondant aux conditions de fusion portant sur la hauteur et la longueur des frontières communes peuvent être réglées approximativement dans la mesure où ces conditions servent principalement à rejeter les fusions clairement non pertinentes du point de vue de l'une ou l'autre des conditions.

Image originale Segmentation sociétale Ajout des contours

FIG. 11.13 – Segmentation sociétale de l'image "Perroquet".

Image originale Segmentation sociétale Ajout des contours

FIG. 11.14 – Segmentation sociétale de l'image "Chasseur".

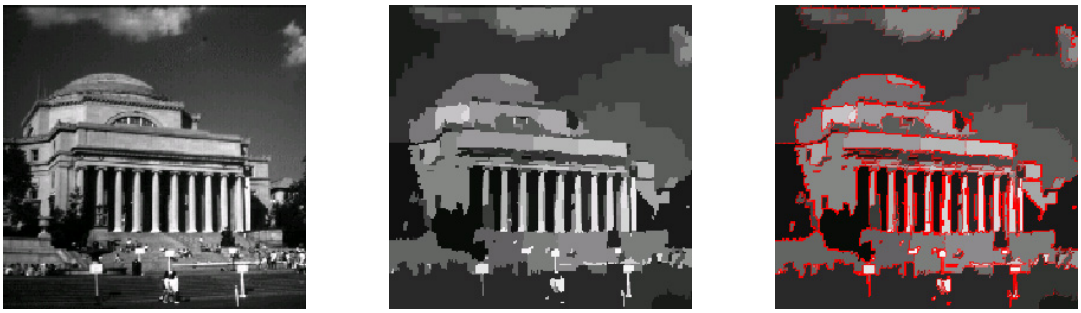


Image originale Segmentation sociétale Ajout des contours

FIG. 11.15 – Segmentation sociétale du l'image "Columbia".

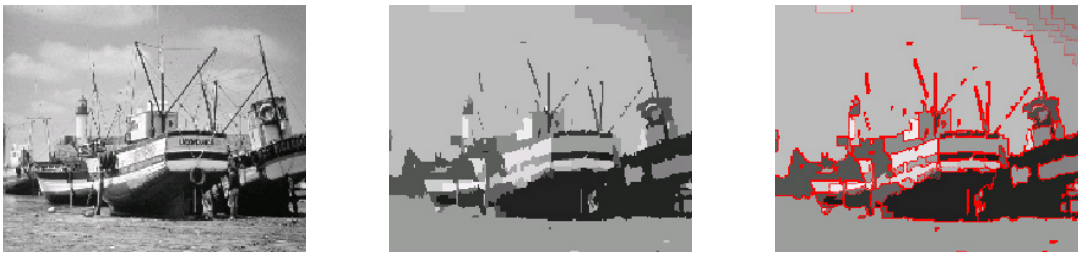


Image originale Segmentation sociétale Ajout des contours

FIG. 11.16 – Segmentation sociétale de l'image "Bateaux".

Reste la dernière constante correspondant à la condition de regroupement de région (11.5 ou 11.9). Cette constante K_1 doit être ajustée précisément dans la mesure où la largeur des intervalles de confiance de la moyenne de la luminance mesurée est proportionnelle à cette constante. En conséquence, K_1 permet d'ajuster le grain de la segmentation effectuée. Des valeurs faibles de K_1 conduisent à une segmentation fine mais retournant un nombre important de régions, une valeur élevée pour K_1 conduit à une segmentation plus grossière, mais avec un nombre très limité de régions. La figure 11.21 présente des segmentations sociétales obtenues pour différentes valeurs de K_1.

Image originale Segmentation sociétale Ajout des contours

FIG. 11.17 – Segmentation sociétale de l'image "Lena".


Image originale


Segmentation sociétale

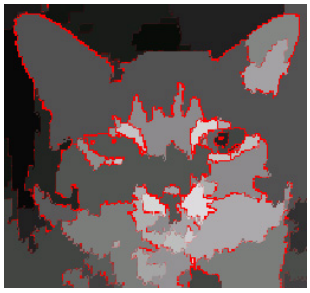

Ajout des contours

FIG. 11.18 – Segmentation sociétale de l'image "Chat".

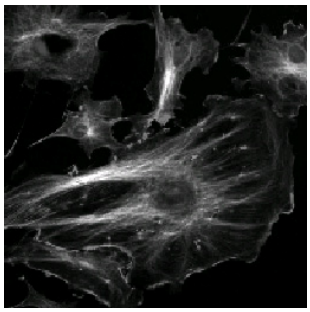
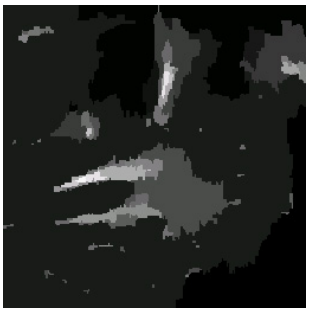


Image originale


Segmentation sociétale

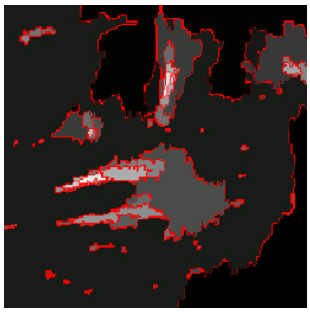

Ajout des contours

FIG. 11.19 – Segmentation sociétale de l'image "Fibres".

11.4.3 Performances en terme de vitesse d'exécution

L'utilisation de l'algorithme de segmentation sociétale dans le cadre des rétines artificielles disposant de traitements régionaux performants est très efficace dans la mesure où les opérations peuvent être quasiment toutes effectuées en parallèle sur les différentes régions.

Détermination du nombre de mesures régionales nécessaires

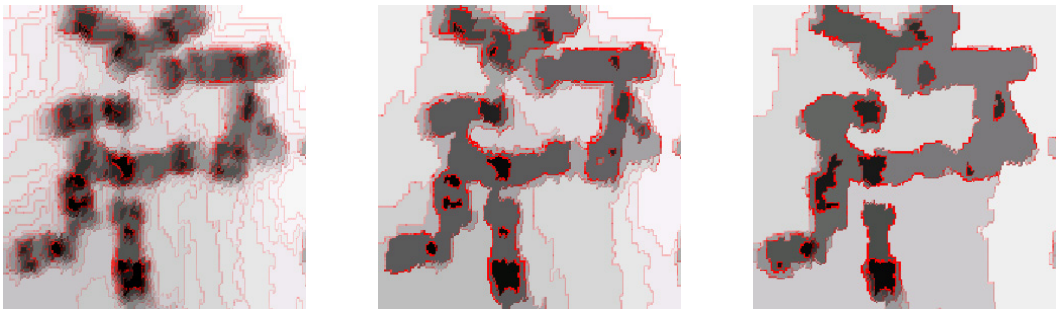
Afin de quantifier précisément le coût temporel de l'algorithme de segmentation sociétale il est nécessaire d'évaluer le nombre de mesures régionales qu'il utilise. Dé-


Image originale


Segmentation sociétale


Ajout des contours

FIG. 11.20 – Segmentation sociétale de l'image "Camera".



(a) $K_1 = 0.8$ (b) $K_1 = 1.2$ (c) $K_1 = 1.6$

FIG. 11.21 – Segmentation sociétale de l'image "Chromosomes" pour différentes valeurs de K_1

terminons tout d'abord le nombre de mesures régionales nécessaires pour effectuer une itération (croissance de la population, regroupement de villages, et conflits inter-villages) de l'algorithme. La phase de croissance des villages utilise les données régionales suivantes :

- altitude maximale du village : elle est obtenue à l'aide d'un opérateur de type max.

- altitude minimale du village : elle est obtenue à l'aide d'un opérateur de type min.

- surface du village : elle est obtenue à l'aide d'un opérateur de type somme régionale, la valeur locale à additionner de chaque pixel étant 1.

- altitude moyenne du village : elle est obtenue à l'aide de la division de la somme de toutes les altitudes de chacun des pixels du village par la surface du village. Elle nécessite donc une opération régionale de sommation supplémentaire destinée à calculer la somme des altitudes de tous les pixels. La division est ensuite effectuée de manière synchrone dans la racine des arbres couvrants des villages. Plutôt que d'effectuer une division puis une comparaison, il est également possible d'effectuer une comparaison entre la somme des altitudes de chacun des pixels et la somme des K_1 placés dans chacun des pixels de la région, ce qui revient au même en passant le dénominateur de la fraction de l'autre coté de l'inégalité. Cette méthode permet d'économiser une opération locale de division en la remplaçant par une somme régionale réalisant la multiplication.

La phase de regroupement des villages utilise les mesures régionales suivantes :

- altitude moyenne de chaque village : déjà calculée précédemment.

– surface de chaque village : déjà calculée précédemment.

– écart-type de l'altitude de chaque village : une fois l'altitude moyenne calculée, elle est redistribuée vers chaque pixel du village. La soustraction de cette valeur moyenne à l'altitude locale est effectuée dans le pixel, de même que l'élévation au carré de cette différence. Ensuite, la somme de toutes ce différences au carré est effectuée à l'aide d'une opération régionale supplémentaire. La division par la surface des villages (déjà calculée) est ensuite effectuée de manière synchrone dans la racine des arbres couvrants des villages. Il est également possible de comparer la somme des altitudes à la somme des K_1 placés dans chaque pixel de la région (voir ci-dessus).

– longueur de la frontière entre deux régions : elle est obtenue à l'aide d'une sommation.

– altitude moyenne de la frontière située entre deux régions : elle est obtenue en divisant la somme de toutes les altitudes de la région par la longueur de la région. Elle nécessite donc une opération régionale de type somme en plus.

Il est important de mentionner la nécessité d'envoyer les informations régionales stockées dans la racine des villages vers les frontières. Toutefois cette opération de propagation peut-être réalisée simplement et n'est pas coûteuse.

La phase de conflits inter-villages n'utilise aucune mesure régionale.

Finalement, on peut dénombrer un nombre total de 7 mesures régionales à effectuer à chaque itération de l'algorithme. Parmi ces 7 mesures 5 font appel à une sommation régionale, et 2 font appel à un opérateur de type max/min. Ces deux dernières opérations régionales ont un coût temporel très réduit.

Comparaison des performances de l'algorithmes sur différents types d'architectures

Afin de valider le fonctionnement de notre algorithme, nous l'avons simulé sur un PC standard. Les opérations régionales asynchrones sont dans ce cas remplacées par des séquences d'opérations élémentaires effectuées en balayant les pixels des régions. Les régions de l'image sont représentées par des structures dynamiques de type files d'attentes. En raison des balayages effectués sur l'image et sur les régions, le fonctionnement de l'algorithme est bien moins performant sur un PC que sur une rétine ayant des capacités régionales. Il faut typiquement 5 à 10 secondes pour segmenter une image $200 * 200$ en utilisant un bi-processeur de 2.2 GHz. La simulation a été effectuée en Python 2.3 compilé et en utilisant des files d'attentes non hiérarchiques. Si l'on considère la qualité de la segmentation obtenue, et l'indication supplémentaire de pertinence des frontières (indiquée par la densité de rouge (fig. 11.8, 11.9, 11.12 et 11.21), le temps de calcul de la segmentation sociétale n'est

pas si élevé comparé à d'autres méthodes de segmentation [ZY96] [PWK00].

Le temps de calcul deviendrait excellent dans le cas d'une segmentation utilisant des opérateurs régionaux implantés sur rétine artificielle. En effet, le nombre de mesures régionales à effectuer en parallèle sur toutes les régions est égal à 7 mesures par itération de l'algorithme de segmentation sociétale. Or le nombre d'itérations varie entre 15 et 20 itérations pour converger vers la segmentation finale. Le nombre total de mesures régionales à effectuer au cours de la segmentation d'une image est donc approximativement égal (en fonction de l'image à traiter) à 100 mesures, ce qui conduit à un temps de calcul de l'ordre de 0.15 seconde dans le pire des cas comme indiqué en introduction de la partie algorithmique. Dans une technologie plus récente telles que le $0.18\mu m$, ce temps de calcul sera réduit à environ 50 ms, ce qui est acceptable pour une application temps réel.

L'utilisation d'une rétine artificielle ayant des fonctionnalités asynchrones se révèle donc être très pertinente dans le cadre d'algorithmes utilisant des mesures régionales (ce qui est le cas de la plupart des algorithmes de traitement d'image de moyen niveau). Dans le cas de la *segmentation sociétale*, on aboutit à une division par 100 du temps de calcul, et à la possibilité d'utiliser l'algorithme en temps réel ou presque.

11.5 Conclusion

L'élargissement du champ des opérateurs de traitements d'image temps réel induite par l'introduction des opérateurs régionaux massivement parallèles dans les rétines artificielles nous a offert une liberté algorithmique nouvelle. Cette liberté nous a conduis à proposer une nouvelle méthode de segmentation, la segmentation sociétale. Au delà de cette adéquation avec une structure matérielle bien définie et pour l'instant non disponible, les résultat obtenus en matière de segmentation sont comparables avec les meilleurs résultats de segmentation que l'on puisse aujourd'hui obtenir avec d'autres méthodes. Ceci est d'autant plus intéressant que la segmentation sociétale n'en est qu'à ses débuts et que les seuls travaux ayant été effectués dessus l'ont été durant la fin de cette thèse et sur une durée de quelques mois. Il est donc raisonnable de penser que cette technique de segmentation a encore un large potentiel non exploité.
Parmi les quelques pistes à envisager pour l'amélioration de cette technique de segmentation figurent les suivantes :

- La population des pixels ne sert dans le modèle proposé qu'à apporter une indication sur la pertinence de l'appartenance d'un pixel à une région. Cette propriété pourrait être utilisée fort à propos dans le processus de regroupement de régions.

- Les coefficients utilisés dans le modèle ont été déterminés de manière empi-

rique. Une étude du type de celle qui a été menée sur la pertinence statistique de la fusion des données serait intéressante et permettrait vraisemblablement de déterminer les constantes à utiliser en fonction de l'image à traiter.

– La modification durant l'exécution de l'algorithme de la valeur de K_1 permettrait d'effectuer une segmentation multi-résolution.

– L'utilisation des informations régionales de type texture permettrait d'améliorer les processus de regroupement de régions. Les informations sur ces textures peuvent être extraites à l'aide d'opérateurs régionaux.

Cette liste de pistes à envisager n'est bien sûr qu'indicative, et il existe vraisemblablement de nombreuses autres pistes de recherche.

Quatrième partie

Conclusion

Principaux résultats

Cette thèse constitue une étape dans un travail prospectif visant à étudier l'intérêt des techniques asynchrones dans le domaine des rétines artificielles, imageurs ayant un processeur élémentaire intégré à chaque pixel.

Après avoir rappelé l'importance des traitements de moyen niveau dans les opérations de traitement d'image, nous avons montré que pour minimiser le coût énergétique en traitement du signal, il est important d'adapter les opérateurs utilisés à chaque étape du traitement d'image aux données à traiter, et qu'il est préférable, dans la mesure du possible, d'effectuer les traitements de moyen niveau à l'intérieur même du dispositif d'acquisition de l'image.

Ces conclusions nous ont conduit à envisager le cas d'une rétine artificielle ayant des capacités de traitement régional à l'intérieur même de l'imageur. L'étude des rétines synchrones SIMD massivement parallèles et de leurs limitations nous a amené à la nécessité d'utiliser des connexions programmables et des modes de communications plus rapides que le déplacement synchrone de proche en proche. Pour ce dernier point, l'asynchronisme s'est révélé être une solution pleine de promesses. Nous avons alors étudié les circuits asynchrones existant dans la littérature et permettant d'effectuer des opérations régionales. Les conclusions de cette étude ont été la pertinence du modèle des réseaux associatifs, mais également l'impossibilité d'utiliser des opérateurs dédiés à une tâche asynchrone précise (telle qu'un additionneur à n entrées) dans les rétines artificielles, et ceci en raison de leur coût matériel trop important et de leur reconfigurabilité très limitée.

Nous avons donc proposé une nouvelle architecture (*les micropipelines associatifs*) permettant d'implanter le modèle des réseaux associatifs de manière économique dans les rétines artificielles. Cette architecture s'appuie sur un élément de circuit asynchrone, le micropipeline convergent, associé à un mode de communication et de calcul distribué asynchrone par jetons. Afin de réduire encore le coût matériel des micropipelines convergents, nous avons effectué une optimisation topologique du réseau de communication.

Conjointement à la structure des *micropipelines associatifs*, un algorithme permettant le calcul de la somme régionale (primitive asynchrone fondamentale) a été proposé. Cet algorithme permet de calculer une somme à partir de valeurs localisées dans chacun des pixels d'une région de manière bit-série, mais sans avoir recours à un seul opérateur d'addition. Le calcul du *OU régional* a également été présenté.

L'utilisation d'un rétine asynchrone ayant des fonctionnalités régionales permet d'envisager sous un angle élargi le traitement d'image dans la mesure où les fonctions régionales implantables permettent de réaliser des calculs de manière très efficace et rapide. Ceci nous a conduit dans une dernière partie algorithmique à envisager l'utilisation de ces opérateurs régionaux dans le cadre de la segmentation

d'images. Deux techniques ont été proposées : l'une améliorant la segmentation par ligne de partage des eaux, la seconde proposant une nouvelle technique de segmentation basée sur un analogie avec la segmentation d'un territoire géographique en villages.

Élargissements possibles

Cette thèse étant de nature prospective, le travail effectué conduit donc naturellement à des élargissement possibles. Parmi ces élargissements, nous présentons ici les principaux, qui feront peut-être l'objet d'un travail ultérieur. Tout d'abord, il est important de noter que si le concept a été validé théoriquement, il reste encore à l'implanter dans une rétine. Le deuxième élargissement est une conséquence du premier. Le fait d'avoir une rétine capable d'effectuer des opérations régionales efficacement et rapidement offre des possibilités algorithmiques nouvelles. Ces possibilités devront être étudiées en pratique, ce qui permettra certainement d'élargir le cadre applicatif des rétines artificielles. Enfin, il serait intéressant d'élargir le cadre de travail des rétines artificielles à celui des processeurs reconfigurables dynamiquement.

Réalisation d'une rétine asynchrone

Les idées présentées dans cette thèse ont toutes été simulées afin d'être validées que ce soit sur un plan algorithmique ou électronique. Toutefois, un travail important reste à effectuer pour passer du concept validé à l'utilisation pratique des fonctionnalités régionales asynchrones décrites dans cette thèse. Ce travail important passe par la réalisation d'une rétine asynchrone mettant en oeuvre les concepts présentés ici. Durant ces trois années, je me suis attaché à ces aspects de réalisation de circuit en travaillant notamment sur des logiciels de fabrication de circuits tels que les outils Tanner. L'investissement en temps étant trop important en particulier en raison du délai nécessaire à la fabrication du circuit, j'ai décidé finalement de ne pas m'orienter vers une réalisation de circuit. Afin de pouvoir utiliser ce travail dans un cadre applicatif, il est donc nécessaire de réaliser une rétine asynchrone mettant en oeuvre les concepts proposés.

Élargissement du cadre applicatif des rétines artificielles

La fabrication d'une rétine artificielle ayant des fonctionnalités régionales permettra de tester en pratique les algorithmes proposés dans la partie segmentation d'image de cette thèse. Toutefois, la segmentation d'image n'est qu'un domaine très spécifique du traitement d'image, et on peut légitimement penser que la mise à disposition de fonctionnalités régionales permettra d'offrir des outils nouveaux dans les différentes branches du traitement d'image. Par exemple, l'usage des informations régionales pour la détection et l'estimation de mouvement peut permettre

de suivre le déplacement du barycentre d'une région, et d'obtenir ainsi une mesure de vitesse peu dépendante des variations de forme de l'objet.

Vers un microprocesseur reconfigurable dynamiquement

Les microprocesseurs les plus courants que l'on peut rencontrer par exemple dans les PC sont des structures assez figées et génériques destinées à traiter des informations ayant un niveau d'abstraction élevé. Les rétines artificielles numériques programmables actuelles sont également des structures figées destinées à traiter des informations de bas niveau.

Comme on peut le constater, ces structures ne sont pas adaptées à un traitement générique efficace des différents types de données que l'on peut rencontrer, ce qui conduit dans le cas des rétines artificielles synchrones à une limitation fonctionnelle aux traitements de bas niveau, et, dans le cas des processeurs CISC, à une utilisation sous-optimale des ressources.

Pour remédier à ces inconvénients, une solution serait de proposer un circuit générique constitué d'une matrice de transistors ou de processeurs élémentaires qu'il serait possible de mettre en commun pour adapter leur architecture à la fonction à effectuer et ce de manière dynamique. Plusieurs solutions allant dans ce sens ont été proposées. Nous citerons entre autres les FPGA permettant d'implanter des structures assez diverses à partir d'une matrice de portes logiques et de bascules et la maille associative d'Orsay permettant d'utiliser des opérateurs pour des fonctions de bas et moyen niveau. Toutefois, ces solutions ne sont pas totalement satisfaisantes : en effet, les FPGA ont un temps de reprogrammation trop important pour reconfigurer dynamiquement le processeur au cours du déroulement d'une application, et le processeur élémentaire de la maille associative d'Orsay a une taille trop importante interdisant une implantation à grande échelle. Les micropipelines associatifs que nous avons proposés offrent également un élément de solution à ce problème dans la mesure où leur reprogrammation est rapide et que la taille de leur processeur élémentaire est très réduite. Toutefois, cette solution n'est pas non plus totalement satisfaisante et générique puisqu'il n'est pas possible d'implanter tous les opérateurs de haut niveau avec cette architecture. Toutefois, nous avons apporté des éléments de solution pour la réalisation d'un processeur générique capable d'effectuer des traitements de moyen et bas niveau.

Une piste de recherche intéressante serait donc de se tourner vers une structure autorisant la mise en commun des ressources de plusieurs processeurs élémentaires de manière à fournir les ressources nécessaires à l'implantation de fonctions de haut niveau, et permettant d'effectuer des opérations de bas niveau et de niveau intermédiaire également. Un tel processeur permettrait d'affecter exactement la quantité de ressources nécessaire à l'exécution d'une tâche donnée et donc d'optimiser en temps réel l'adéquation entre l'algorithme effectué et l'architecture utilisée.

Bibliographie

[AB96] A.G. Andreou and K.A. Boahen. Translinear circuits in subthreshold cmos. *Journal of Analog Integrated Circuits and Signal Processing*, 9(2) :141–166, March 1996.

[AGLM93] L. Alvarez, F. Guichard, P. L. Lions, and J. M. Morel. Axioms and fundamental equations of image processing. *Arch. Ration. Mechan.*, 123 :199–257, 1993.

[ARF96] A. Astrom and J. E. Eklund R. Forchheimer. Global feature extraction operations for near-sensor imageprocessing. *IEEE Transactions on Image Processing*, 4(3) :322–335, Sep 1996.

[ASB+96] X. Arreguit, F. A. Van Schail, F. Bauduin, M. Bidiville, and E. Raeber. A cmos motion detector for pointing devices. *IEEE Journal of Solid-State Circuits*, 31(12) :1916–1921, December 1996.

[AWJ90] A. A. Amini, T. E. Weymouth, and R. C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9) :855–867, 1990.

[Bee02] P. A. Beerel. Asynchronous circuits : An increasingly practical design solution. In *Proceedings of the International Symposium on Quality Electronic Design*, 2002.

[Ber92] T.M. Bernard. *Des Rétines artificielles intelligentes.* PhD thesis, Université Paris-Sud, UFR Scientifique d'Orsay, October 1992.

[Ber98a] T. Bernard. Rétines artificielles et robotique mobile. Technical Report 98 R 033 (204 pp.), CTA, August 1998.

[Ber98b] T.M. Bernard. Processing images in the focal plane for perception applications. In R. De Keyzer, editor, *Proc. International Congress on Imaging Science*, pages 246–251, Antwerp, Belgium, September 1998.

[Ber00] T.M. Bernard. Multipurpose semistatic shift-registers for digital programmable retinas. In *SPIE, Sensors and Camera Systems for Scientific, Industrial and Digital Photography Applications*, volume 3965, pages 277–288, jan 2000.

[Ber05] T. M. Bernard. Consommation d'énergie dans les rétines artificielles. In *Communication Personnelle*, 2005.

[Beu82] S. Beucher. Watersheds of functions and picture segmentation. In *ICASSP*, pages 1928–1931, 1982.

[BF93] S. Beucher and F.Meyer. *The morphological approach to segmentation : the watershed transformation.* Mathematical morphology in Image Processing. E.R. Dougherty, 1993.

[BG92] K. A. Boahen and A. G.Androu. A contrast sensitive silicon retina with reciprocal synapsis. *Neural Information Processing Systems*, 1992.

[BRR+04] F. Bouesse, M. Renaudin, B. Robisson, E. Beigne, P.Y. Liardet, S. Prevosto, and J. Sonzogni. Dpa on quasi delay insensitive asynchronous circuits : Concrete results. In *XIX Conference on Design of Circuits and Integrated Systems*, november 2004.

[CBC01] C. S. Choy, J. Butas, and C. F. Chan. A new control circuit for asynchronous micropipelines. *IEEE Transactions on Computers*, 50(9), 2001.

[CKS95] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In *ICCV*, pages 694–699, 1995.

[CMLC92] F. Catté, J.-M. Morel, P.-L. Lions, and T. Coll. Image selectives smoothing and edge detection by nonlinear diffusion. *SIAM Journal of Numerical Analysis*, 29 :182–193, 1992.

[Cot95] G. H. Cottet. Neural networks : continuous approach and applications to image processing. *Journal of Biological Systems*, 3 :1131–1139, 1995.

[CPPC98] C. S. Choy, T. C. Pang, J. Povazanec, and C. F. Chan. A useful micropipeline architecture to implement dsp algorithms. In *24th EUROMICRO '98 Conference*, 1998.

[CR93] L. O. Chua and T. Roska. The cnn paradigm. *IEEE Transaction on Ciruits and Systems I*, 40(3) :147–156, 1993.

[DC97] R. Dominguez-Castro. A 0.8 um cmos 2-d programmable mixed-signal focal-plane array processor with on-chip binary imaging and instructions storage. *IEEE Journal of Solid State Circuits*, 32(7) :1013–1026, 1997.

[Del93] T. Delbruck. Silicon retina with correlation-based, velocity-tuned pixels. *IEEE Transactions on Neural Networks*, 4(3) :529–541, May 1993.

[DeW92] S. P. DeWeerth. Analog vlsi circuits for stimulus localization and centroid computation. *International Journal of Computer Vision*, 8(3) :191–202, September 1992.

[DM98] B. Ducourthial and A. Merigot. Image analysis with r-operators. Technical report, IEF, Université Paris Sud, 1998.

[DMM95] D. Dulac, S. Mohammadi, and A. Merigot. Implementation and evaluation of a parallel architecture using asynchronous communications. In *CAMP*, pages 106–111, 1995.

[Duc00] B. Ducourthial. *Vision artificielle rétinienne*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, 2000.

[Dul96] D. Dulac. *Contribution au parallélisme massif en analyse d'image : une architecture SIMD fondée sur la reconfigurabilité et l'asynchronisme*. PhD thesis, IEF, Université de Paris-Sud, 1996.

[EB02] D. Edwards and A. Bardsley. Balsa : An asynchronous hardware synthesis language. *The Computer Journal*, 45(1) :12–18, 2002.

[Eve03] N. Evenot. *Segmentation des lèvres par un modèle déformable analytique*. PhD thesis, Institut National Polytechnique de Grenoble, 2003.

[Gab65] D. Gabor. Information theory in electron microscopy. *Lab. Investig.*, 14 :801–807, 1965.

[Gal02] B. Galilée. *Etude d'adéquation algorithme-architecture pour terminaux mutimédia portables : segmentation d'images par un réseau de processeurs asynchrones.* PhD thesis, INPG, Grenoble, France, Oct 2002. Dirigée sous couvert du Prof. M. Renaudin.

[GB04] V. Gies and T. M. Bernard. Statistical solution to watershed oversegmentation. In *IEEE International Conference on Image Processing*, pages 1863–1866, October 2004.

[GB05a] V. Gies and T. M. Bernard. Increasing interconnection network connexity for reducing operator complexity in asynchronous vision systems : The example of the regional adder. In *Discrete Geometry for Computer Imaging (DGCI) International Conference*, 2005.

[GB05b] V. Gies and T. M. Bernard. Societal segmentation : A novel segmentation method. In *EOS Conference on Industrial Imaging and Machine Vision*, June 2005.

[GB05c] V. Gies and T. M. Bernard. Tree extension of micro-pipelines for mixed synchronous-asynchronous implementation of regional image computations. In *SPIE Conference : Sensors and Camera Systems for Scientific and Industrial Applications VI*, pages 35–46, January 2005.

[GB05d] V. Gies and T. M. Bernard. Using an energy time index for reducing consumption in buses according to their application. In *5e Journées Faible Tension Faible Consommation*, May 2005.

[GBM05a] V. Gies, T. M. Bernard, and A. Mérigot. Convergent micro pipelines : A versatile operator for mixed synchronous-asynchronous computations. In *IEEE International Symposium on Circuits ans Systems*, May 2005.

[GBM05b] V. Gies, T. M. Bernard, and A. Mérigot. Hardware reduction using a 6-connexity interconnection network over a 4-connexity vlsi asynchronous array processor. In *IEEE International Symposium on Circuits ans Systems*, May 2005.

[Hau95] S. Hauck. Asynchronous design methodologies : An overview. *Proceedings of the IEEE*, 83(1) :69–93, 1995.

[HKLM88] J. Hutchinson, C. Koch, J. Luo, and C. Mead. Computing motion using analog and binary resistive networks. *IEEE Transactions on Computers*, 21(3) :52–63, March 1988.

[HKS+90] J. G. Harris, C. Koch, E. Staats, , and J. Luo. Analog hardware for detecting discontinuities in early visions. *International Journal of Computer Vision*, 4 :211–223, February 1990.

[Jai77] A. K. Jain. Partial differential equations and finite-difference methods in image processing, part 1 : Image representation. *Optim. Theory Appl.*, 23 :65–91, 1977.

[JEE96] A. Astrom J. E. Eklund, C. Svensson. Vlsi implementation of a focal plane image processor-a realization of the near-sensor image processing concept. *IEEE Transactions on Image Processing*, 5(1) :102–110, Jan 1996.

[KKI04] T. Komuro, S. Kagami, and M. Ishikawa. A dynamically reconfigurable simd processor for a vision chip. *IEEE Journal of Solid-State Circuits*, 39(1) :265–268, January 2004.

[KLM94] G. Koepfler, C. Lopez, and J. M. Morel. A multiscale algorithm for image segmentation by variational method. *SIAM Journal on Numerical Analysis*, 31(1) :282–299, 1994.

[Koe84] J. J. Koenderink. The structure of images. *Biol. Cybern.*, 50 :363–370, 1984.

[KWT87] M. Kass, A. Witkin, and D. Terzopolous. Snakes : Active contour models. *International Journal of Computer Vision*, 1(4) :321–331, 1987.

[LFE+99] G. Linan, P. Foldesy, S. Espejo, R. Dominguez-Castro, and A. Rodriguez-Vasquez. A 0.5 um cmos 106 transistors analog programmable array processor for real-time image processing. In *European Solid State Circuits Conference*, pages 358–361, September 1999.

[Lin96] A. M. Lines. Pipelined asynchronous circuits. Technical Report Master's thesis, California Institute of Technology, 1996.

[Lit03] D. Litaize. Réseaux d'interconnexions pour systèmes multiprocesseurs fortement couplés : évolution, challenges. In *Architectures des systèmes matériels enfouis et méthodes de conception associées*, April 2003.

[LS91] L.Vincent and P. Soille. Watershed in digital spaces : an efficient algorithm based on immersion simulations. *IEEE Transactions on Patterm Analysis and Machine Intelligence*, 13 :583–598, 1991.

[Mah94] M.A. Mahowald. Analog vlsi chip for stereocorrespondence. In *International Symposium on Circuits and Systems*, volume 6, pages 347–350, June 1994.

[MAJo99] M. Milanova, P. E. M. Almeida, J. Okamoto Jr., and M. G. Sim oes. Applications of cellular neural networks for shape from shading problem. *Lecture Notes in Artificial Intelligence*, pages 52–63, 1999.

[Man00] A. Manzanera. *Vision artificielle rétinienne*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, 2000.

[Man02] A. Manzanera. Morphological segmentation on the programmable retina : towards mixed synchronous/asynchronous algorithms. In *ACM ISMM Conf.*, pages 389–399, April 2002.

[MBPL99] A. Manzanera, T. M. Bernard, F. Prêteux, and B. Longuet. Ultra fast skeleton based on an isotropic fully parallel algorithm. In *Lecture notes in Computer Science, Vol. 1568, Discrete Geometry for Computer Imagery*, pages 313–324, Marne-La-Vallée, France, March 1999. Springer Verlag.

[Mea89] C. A. Mead. Adaptive retina. *Analog VLSI Implementation of Neural Systems*, 80 :239–246, 1989.

[Mer97] A. Merigot. Associative nets : A graph-based parallel computing net. *IEEE Transactions on Computers*, 46(5) :558–571, 1997.

[Mey94] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38(1) :113–125, 1994.

[MH80] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society London*, 207 :187–217, 1980.

[MLM+97] A. J. Martin, A. Lines, R. Manohar, M. Nystroem, P. Penzes, R. Southworth, and U. Cummings. The design of an asynchronous MIPS R3000 microprocessor. In *Advanced Research in VLSI*, pages 164–181, 1997.

[MM88] C. A. Mead and M. A. Mahowald. A silicon model of early visual processing. *Neural Network*, 1 :91–97, 1988.

[Moh96] S. Mohammadi. *Techniques asynchrones pour la réalisation d'une machine massivement parallèle reconfigurable*. PhD thesis, IEF, Université de Paris-Sud, 1996.

[Moi97] A. Moini. An insect vision-based motion detection chip. *IEEE Journal of Solid-State Circuits*, 32(2) :279–284, February 1997.

[MS85] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *IEEE Computer Vision and Pattern Recognition (CVPR) Conf.*, pages 22–26, 1985.

[MS89] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42 :577–685, 1989.

[NMJB97] R. Nguyen, D. Mercier, A. Jullian, and T.M. Bernard. Hardware-software aspects of shift-register based NEWS networks for the focal plane. In C. Weems, editor, *Proc. Workshop on Computer Architecture for Machine Perception*, pages 84–93, Boston, MA, USA, October 1997. IEEE Computer Society Press.

[NMM98] Y. Nagata, D. M. Miller, and M. Mukaidono. B-ternary logic based asynchronous micropipeline. In *Proceedings of the Twenty Ninth IEEE International Symposium on Multi-Valued Logic*, pages 402–407, 1998.

[NS92] M. Nitzberg and T. Shiota. Nonlinear image filtering with edge and corner enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7) :826–833, 1992.

[OB02] R. O. Ozdag and P. A. Beerel. High-speed qdi asynchronous pipelines. In *International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 13–22, April 2002.

[Ozd03] R. O. Ozdag. *Template Based Asynchronous Design*. PhD thesis, University of Southern California, November 2003.

[Pai01] F. Paillet. *Intégration et évaluation de Rétines Artificielles Numériques Programmables de hautes performances*. PhD thesis, Université Paris VI, September 2001.

[PF94] B. Pain and E. R. Fossum. Approches and analysis for on-focal-plane analog-to-digital conversion. In *SPIE Aerospace Sensing - Infrared Readout Electronics*, volume 2226, pages 208–218, April 1994.

[PKL+94] L. Priese, J. Klieber, R. Lakmann, V. Rehrmann, and R. Schian. New results on traffic sign recognition. In *IEEE Intelligent Vehicles Symposium*, pages 249–254, October 1994.

[PM88] P. Perona and J. Malik. A network for multiscale image segmentation. In *IEEE ISCAS Conf.*, volume 3, pages 2565–2568, June 1988.

[PM90] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7) :629–639, 1990.

[PMB98] F. Paillet, D. Mercier, and T.M. Bernard. Making the most of 15k lambda2 silicon area for a digital retina PE. In T.M. Bernard, editor, *Proc. SPIE, Vol. 3410, Advanced Focal Plane Arrays and Electronic Cameras*, Zürich, Switzerland, May 1998.

[PMB99] F. Paillet, D. Mercier, and T.M. Bernard. Second generation programmable artificial retina. In *IEEE ASIC/SOC Conference*, pages 304–309, September 1999.

[Pre04] F. Precioso. *Contours actifs paramétriques pour la segmentation d'images et vidéos*. PhD thesis, Université de Nice, 2004.

[PSR04] D. Panyasak, G. Sicard, and M. Renaudin. A current shaping methodology for lowering em disturbances in asynchronous circuits. *Microelectronics journal*, 35 :531–540, June 2004.

[PWK00] I Pollak, A. S. Willsky, and H. Krim. Image segmentation and edge enhancement with stabilized inverse diffusion equations. *IEEE Transactions on Image Processing*, 9(2) :256–266, 2000.

[Ren00a] M. Renaudin. Asynchronous circuit and systems : A promising alternative. *Microelectronic engineering*, 54 :133–149, 2000.

[Ren00b] M. Renaudin. Etat de l'art sur la conception des circuits asynchrones : perspectives pour l'intégration des système complexes. Technical Report ISRN TIMA-RR-02/12-02-FR, 2000.

[Rez04] A. Rezzag. *Synthèse Logique de Circuits Asynchrones Micropipeline.* PhD thesis, Institut National Polytechnique de Grenoble, 2004.

[Ric06] J. Richefeu. *Détection et analyse du mouvement sur système de vision à base de rétine numérique.* PhD thesis, Université Paris VI, 2006.

[RJL96] U. Ringh, C. Jansson, and K. Liddiard. Readout concept employing a novel on chip 16 bit adc for smart ir focal plane arrays. In *SPIE*, volume 2745, pages 99–110, April 1996.

[RP98] V. Rehrmann and L. Priese. Fast and robust segmentation of natural color scenes. In *ACCV*, pages 598–606, 1998.

[RR00] T. Roska and A. Rodriguez. Review of cmos implementations of the cnn universal machine-type visual microprocessors. In *IEEE International Symposium on Circuits and Systems*, pages 120–123, 2000.

[Sli04] K. Slimani. *Une méthodologie de conception de circuits asynchrones à faible consommation d'énergie : application au microprocesseur MIPS.* PhD thesis, Institut National Polytechnique de Grenoble, 2004.

[SRSR04] K. Slimani, Y. Remond, G. Sicard, and M. Renaudin. Tast profiler and low energy asynchronous design methodology. In *Proceedings of 14th International Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 268–277, september 2004.

[Sut89] I.E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6) :720–738, 1989.

[TB98] G. S. Taylor and G. M. Blair. Reduced complexity two-phase micropipeline latch controller. *IEEE Journal of Solid State Cicuits*, 33(10), 1998.

[TH99] A. Torralba and J. Herault. An efficient neuromorphic analog network for motion estimation. *IEEE Transactions on circuits and systems I : special issue on bio-inspired processors and CNNs for vision*, 49(2), February 1999.

[vBKR+91] K. van Berkel, J. Kessels, M. Roncken, R. Saeijs, and F. Schalij. The vlsi-programming language tangram and its translation into handshake circuits. In *European Conference on Design Automation*, pages 284–289, 1991.

[vGBvB+98] H. van Gageldonk, D. Baumann, K. van Berkel, D. Gloor, A. Peeters, and G. Stegmann. An asynchronous low-power 80c51 microcontroller. *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 96–107, 1998.

[Wal64] C. Wallace. A suggestion for a fast multiplier. *IEEE Transactions on Electronic Computers*, pages 14–17, 1964.

[Wit83] A. P. Witkin. Scale-space filtering. In *International Joint Conference on Artificial Intelligence*, pages 1019–1021, 1983.

[WS92] D. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *Computer Vision, Graphics, and Image Processing : Image Understanding*, 55 :14–26, 1992.

[YBA96] K. Y. Yun, P. A. Beerel, and J. Arceo. High-performance asynchronous pipeline circuits. In *Proceedings of the 2nd International Symposium on Advanced Research in Asynchronous Circuits and Systems*, page 17, 1996.

[YFG99] D.X.D. Yang, B. Fowler, and A. El Gamal. A nyquist-rate pixel-level adc for cmos image sensors. *IEEE Journal of Solid State Circuits*, 34(3) :348–356, 1999.

[Zav81] B. Zavidovique. *Contribution à la vision des robots.* PhD thesis, Université de Besançon, 1981.

[ZY96] S. C. Zhu and A. L. Yuille. Region competition : Unifying snakes, region growing, and bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9) :884–900, 1996.