Architecture SoC-FPGA pour la mesure temps réel par traitement d'image. Conception d'un système embarqué : imageur CMOS et Circuit Logique Programmable.

Lionel Lelong

Année 2005

THESE

Présentée devant

L'UNIVERSITE JEAN MONNET DE SAINT-ETIENNE

Pour obtenir le grade de

DOCTEUR

Spécialité : électronique

Par
Lionel LELONG

Achitecture SoC-FPGA pour la mesure temps réel par traitement d'image. Conception d'un système embarqué : imageur CMOS et Circuit Logique Programmable.

Directeur de thèse : Gérard Jacquet Co-directeur : Guy Motyl

Soutenue le 7 décembre 2005 devant le jury composé de :

- JURY -

Elbey BOURENNANE	Professeur, Université de Bourgogne (Dijon)	Rapporteur
Serge WEBER	Professeur, Université Henri Poincaré (Nancy 1)	Rapporteur
Tanguy Risset	Professeur, INSA de Lyon (laboratoire CITI)	Examinateur
Gérard Jacquet	Professeur, Université Jean Monnet	Encadrant
Guy Motyl	Maitre de conférences, Université Jean Monnet	Encadrant

TABLE DES MATIERES

Introduction

Les applications embarquées font appel à des algorithmes pour traiter différents types de données (traitement du signal et des images, compression audio ou vidéo) et utilisent des algorithmes de plus en plus sophistiqués dont la complexité n'a cesse de croître. Du fait de cet accroissement de la complexité, ces applications, et en particulier celles de traitement d'images ou de vision, demandent toujours de plus en plus de puissance de calcul. Cet aspect s'avère encore plus crucial, voire critique, lorsque l'implémentation de ces algorithmes nécessite un traitement des données en temps réel. Il s'agit donc de respecter des contraintes temporelles imposées entre deux occurrences successives d'un signal d'entrée (période) ou entre un signal d'entrée et un signal de sortie (latence). Actuellement, les algorithmes de traitement d'images présentant une contrainte temps réel vidéo voient leurs implémentations s'orienter vers des architectures numériques dédiées. Les solutions architecturales habituellement utilisées sont : des ordinateurs de types PC, des cartes électroniques à base de circuits intégrés ou de processeurs spécialisés (DSP) ou à architecture multi-composant (processeurs spécialisés, des circuits de type VLSI et/ou des FPGAs). Le choix de l'architecture est fonction de nombreux critères fortement dépendants de l'application. Les solutions adoptées en termes de techniques et de technologie, dépendent très fortement du contexte et des contraintes de ces systèmes. Un autre facteur important sur le choix d'une solution est bien sûr celui de la capacité du système d'embarquer plus ou moins de ressources matérielles.

Intégrer une grande puissance de traitement dans un système embarqué implique généralement des architectures hétérogènes et parallèles pour rechercher les meilleurs compromis performances, taille et consommation. Ces architectures peuvent être réalisées avec des composants offrants une grande flexibilité comme les DSPs et/ou des composants offrants de bonnes performances comme les ASICs et les circuits logiques programmables (FPGAs). Pour réaliser de tels systèmes, deux approches sont possibles : répartir le système sur plusieurs composants ou tout intégrer dans un unique composant. Cette dernière approche repose sur le concept de SoC, abréviation pour *System On-Chip*.

Aujourd'hui, les circuits logiques programmables offrent aux concepteurs la possibilité de réaliser des applications numériques relativement complexes dans un unique circuit en fournissant performance et flexibilité. En effet, les dernières générations de circuits configurables disposent, en plus des traditionnelles cellules logiques, de blocs mémoires RAM, de multiplieurs câblés, d'entrées/sorties dédiées aux communications rapides et de cœurs de processeurs embarqués. Utilisés auparavant comme circuit de prototypage pour la réalisation d'ASIC ou comme circuit d'interface pour des processeurs dédiés aux calculs, les circuits logiques programmables offrent, aujourd'hui, une flexibilité et un flot de conception rapide, sûr et bon marché. L'optimisation des algorithmes à l'architecture du composant (parallélisme, *Look Up Tables*, etc.) augmente les performances jusqu'à dépasser celles d'implémentation sur DPS et ASICs. De plus, la description d'une architecture optimisée en code synthétisable VHDL avec une approche hiérarchique et modulaire apporte une bonne flexibilité au système. Le FPGA est le circuit logique programmable le plus couramment utilisé de nos jours, de part ses capacités, sa vitesse et sa grande flexibilité. Ces architectures mixent généralement des composants matériels et logiciels travaillant en concurrence afin de répondre le plus efficacement à la contrainte temporelle imposée. Des études de cas montrent en effet que de tels SoCs présentent actuellement une structure architecturale des plus adéquates en terme de performances et d'efficacité d'implantation malheureusement au détriment d'une méthodologie d'implantation plus complexe et longue.

Dans le cadre de cette étude, nous avons étudié les possibilités d'un système embarqué pour une application d'estimation de mouvement dans des séquences d'images. L'application présentée est basée sur une technique de mesure de mouvement pour la mécanique des fluides appelée Particles Images Velocimetry (PIV). Elle consiste à détecter le mouvement de particules illuminées entre une paire images successives. Cette détection est réalisée grâce à un algorithme de corrélation. Ce type d'application requiert des puissances de calcul importantes (proportionnelles à N^4 sans simplification des calculs) ainsi qu'une capacité à manipuler de grandes quantités de données. Les travaux menés en mécanique des fluides au sien du laboratoire TSI ont permis de décrire ces traitements sous forme d'algorithmes réguliers avec un parallélisme au niveau des calculs. Les circuits logiques programmables apparaissent bien adaptés à ce type d'application. D'autres travaux en électronique, antérieure aux nôtres, ont permis de concevoir une architecture générique appelée Round-About pour des applications de mesures de vitesse de particules dans un écoulement. Dans un souci de généricité, la conception de l'architecture a été pensée pour répondre à une classe d'applications plus vaste : la mesure de paramètres physiques par traitement d'images.

Le travail que nous présentons dans ce mémoire vise la transposition de l'architecture Round-About vers une architecture SoC-FPGA, dans un but de mettre en place une plate-forme de prototypage rapide et évolutive. Cette transposition consistera à décrire en langage VHDL l'architecture et à l'adapter aux contraintes physiques de notre prototype. Dans le cadre de ce travail, nous utiliserons une carte de développement FPGA de la société Altera pour simplifier la conception

et le développement du système. Associé à un capteur d'images et à une mémoire externe (SRAM), le FPGA APEX 20K200 permet de constituer un système complet d'acquisition et de traitement embarqué et dédié. En ce qui concerne le capteur d'images électronique, notre choix s'est porté sur les deux technologies, dominant actuellement le marché, les capteurs d'images CCD ou CMOS. Grâce à l'évolution des technologies submicroniques, les capteurs CMOS possèdent, aujourd'hui, une qualité d'image (sensibilité, bruit) égale celle des capteurs CCD qui permet leur utilisation dans les applications de traitement d'images embarquées et temps réel.

Ce mémoire de thèse se découpe en quatre parties, organisée en chapitres.

La première partie expose le contexte applicatif de cette étude. Il s'agit de l'étude d'écoulement en mécanique des fluides, plus particulièrement la visualisation et l'estimation des mouvements. Le premier chapitre présente succinctement les techniques de mesures utilisées en mécanique des fluides, pour développer la technique retenue depuis la mise en évidence des mouvements jusqu'au traitement des images. Le deuxième chapitre examine les dispositifs expérimentaux mis en œuvre pour la visualisation en mécanique des fluides au sein du laboratoire TSI dans le cadre de l'étude d'écoulement de jets d'eau, mais aussi ceux proposés sur le marché par le monde industriel. Puis, nous terminerons par l'approche de mesure temps réel qui a motivé cette étude ainsi qu'une bibliographie des systèmes développés par la communauté scientifique.

La deuxième partie présente le contexte technologique de cette étude. Elle est divisée en deux chapitres (numéroté 3 et 4), chacun étant dévolu à l'un des circuits de notre système : les capteurs d'images et les systèmes monopuces. Pour les premiers, nous nous limiterons à la présentation des capteurs optiques à base de semi-conducteur que sont les capteurs CCD et CMOS. De ces présentations, nous choisirons le type de capteurs le mieux adapté aux contraintes de l'application dans l'optique de réaliser un prototype. Le quatrième chapitre nous permet d'introduire le concept de système embarqué monopuce. L'approche système sur puce sera d'abord présentée, de sa définition jusqu'à ces ressources matérielles et logicielle en passant par ces flots de conceptions. Puis, nous exposerons notre approche des systèmes embarqués monopuce : la conception *System on-a-Programmable-Chip*. Après avoir définie cette approche, une description des différents composants pouvant constitués l'architecture interne d'un SoPC est présentée. Puis, les méthodologies de conception associées et le support physique choisi (les FPGAs) sont détaillés.

La troisième partie développe l'architecture Round-About de sa conception à son fonctionnement. D'abord, le cinquième chapitre est consacré à la démarche que nous avons suivi pour aboutir à cette architecture, des concepts architecturaux jusqu'à la définition d'une architecture spécifique pour la mesure de paramètres physiques par traitement d'images. L'architecture ainsi obtenue est composée de modules dédiés à une fonctionnalité pour former une architecture modulaire. Ce chapitre se termine sur les présentations des deux implémentations physiques de l'architecture Round-About en détaillant les points de l'architecture qui ont du être adaptés pour répondre aux

contraintes. Dans le sixième chapitre, une description fonctionnelle précise expose l'architecture de façon hiérarchique dans le cas d'une implémentation SoC-FPGA. Les communications entre modules seront détaillées avant de décomposer chaque module en structures plus petites.

La quatrième partie décrit la réalisation et les perspectives. Ce septième chapitre développe le travail que nous avons fait pour aller de la description fonctionnelle à la réalisation d'un prototype du système en passant par l'adéquation des algorithmes à l'architecture du module de calcul. Après la présentation matérielle et logicielle du prototype, l'implémentation SoC-FPGA de l'architecture Round-About sera exposée, ainsi que les résultats obtenus tant en occupation surfacique que temporels. De plus, un compte-rendu de l'utilisation du prototype sur des bulles d'air en mouvement dans de l'eau démontera la validité de nos choix. Nous terminerons ce chapitre et ce mémoire sur les évolutions à apporter au prototype mais aussi à l'architecture Round-About, et aussi sur ces perspectives futures.

Chapitre 1

Techniques de mesure

1.1 - Etude des écoulements en mécanique des fluides

1.1.1 Contexte

L'évolution actuelle de la mécanique des fluides conduit à réaliser des mesures de vitesse de plus en plus complètes et de plus en plus fines. En effet, la sophistication des modèles numériques de prévision des écoulements nécessite l'acquisition de mesures en deux dimensions voire en trois dimensions permettant la validation de ces modèles. De plus, des cadences élevées de traitement s'avèrent de plus en plus nécessaires car on doit traiter maintenant des problèmes intrinsèquement instationnaires (en plus de leur caractère turbulent) comme par exemple l'injection de carburant dans un moteur [Peters 00], la modification de fonctionnement d'une aile d'avion en fonction de la position des volets [Gad-el-Hak 00], le déplacement et la diffusion d'un nuage de polluant dans l'atmosphère [Blackadar 98].

La mécanique des fluides est une science qui associe étroitement théorie et expérimentation. En effet, les équations décrivant les phénomènes de mécanique des fluides sont très complexes, et les expériences visent à la compréhension de ces lois [Zaouali 04]. On s'intéresse pour cela aux déplacements du fluide en étudiant les trajectoires et les vitesses correspondantes en tout point de la zone d'étude.

Parmi les différentes techniques de mesure, on distingue deux grandes familles. La première consiste en la mesure ponctuelle de vitesse tandis que la seconde, mettant à profit les techniques d'imagerie et d'optique, apportent une vision globale (deux ou trois dimensions) de l'écoulement par mesure multipoints au même instant.

Sur la base des méthodes de mesures ponctuelles, une carte des vitesses peut être obtenue par déplacement du point de mesure, à l'aide de platines de déplacement linéaire micrométriques. La

représentation obtenue, bidimensionnelle ou tridimensionnelle ne permet cependant pas l'obtention d'un champ instantané de vitesses. Les points faibles des techniques de mesures ponctuelles comme le fait de perturber l'écoulement ont conduit au développement de l'imagerie appliquée à la visualisation d'écoulements comme l'anémométrie laser à effet Doppler qui effectue des mesures ponctuelles non intrusives [Ben Aissia 02].

L'imagerie apporte au mécanicien des fluides un moyen souple et rapide pour l'interprétation qualitative des phénomènes observés. Sur le plan quantitatif, l'étude de deux images successives permet d'obtenir une carte de vitesses instantanées, base de tout traitement statistique (moyenne, écart type, etc.). D'autres techniques telles que l'ombroscopie, l'interférométrie ou l'holographie permettent l'accès à d'autres grandeurs.

1.1.2 Techniques de mesures de vitesse sur des écoulements

En dynamique, la vitesse constitue la grandeur physique la plus "recherchée" avec la position et l'accélération. Ce sont les paramètres les plus importants dans la caractérisation de l'aérodynamique d'un écoulement. De ce fait, plusieurs techniques ont fait l'objet de nombreuses études ayant comme objectif le développement de méthodes de mesure non intrusives autres que les moyens de mesure traditionnels (capteurs physiques tels que les sondes de pressions et les anémomètres à fil chaud). Actuellement, on utilise deux techniques de mesure permettant la détection de la vitesse : la technique *Laser Doppler Velocimetry* (LDV) et la technique *Particle Image Velocimetry* (PIV).

Ces techniques de mesure sont utilisées pour l'étude d'une grande variété d'écoulement. Diverses structures tourbillonnaires sont susceptibles d'apparaître dans ces écoulements. Pour pouvoir les caractériser et distinguer, on calcule une grandeur adimensionnelle, valeur du nombre de Reynolds ou Re.

1 - Laser Doppler Velocimetry

La technique de mesures locales de vitesse par Vélocimétrie Laser à effet Doppler (LDV) est une technique optique qui permet de mesurer à chaque instant la vitesse moyenne de l'écoulement en un point de mesure [Balint 86]. C'est une technique locale au sens où les résultats obtenus sont la vitesse du fluide en un point et à différents instants. Elle consiste à faire croiser deux faisceaux laser monochromatiques et cohérents sur un point de l'écoulement (Figure 1-1).

A la croisée de ces faisceaux se crée, dans un volume appelé volume de mesure, un réseau de franges d'interférences. Ainsi, une particule diffuse une lumière d'intensité variable suivant qu'elle traverse une frange brillante ou sombre de ce réseau. La vitesse de la particule est déterminée en calculant le temps pris par la particule pour changer de franges d'interférence. Cette technique présente l'avantage d'avoir une bonne résolution spatiale (de 1 à 2 millimètres dans la plus grande dimension) et de posséder une grande précision de mesure (de 0,1 à 1%) [Adrian 86].

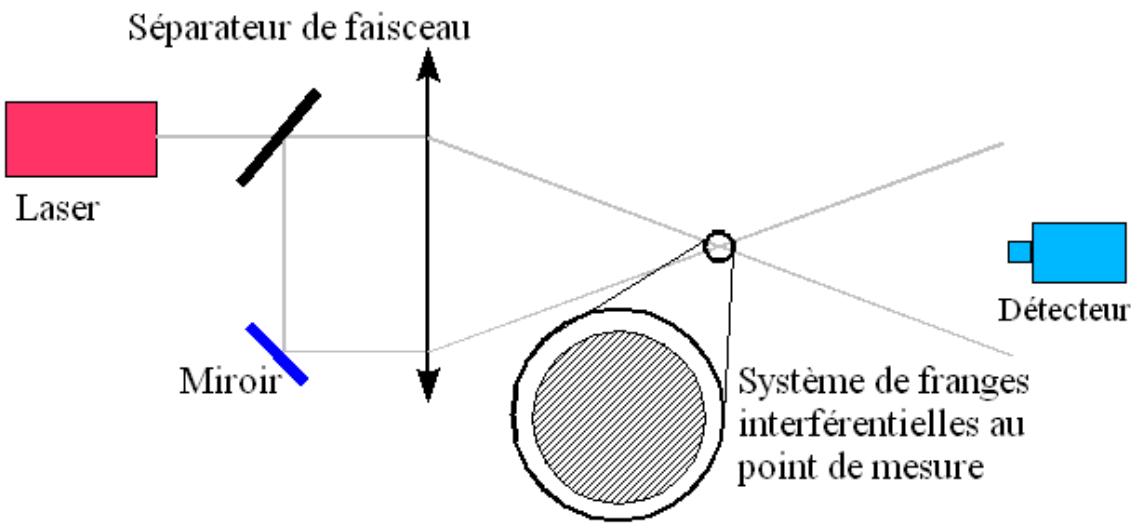


Figure 1-1 : Schéma d'un système type de mesure de déplacement par LDV.

L'inconvénient inhérent à cette méthode est de ne fournir une information sur la vitesse de l'écoulement qu'en un point (mesures ponctuelles). Ainsi, pour obtenir une mesure multipoints, c'est à dire une carte des vitesses de l'écoulement, il faut réitérer la mesure sur l'ensemble des points désirés. Le temps nécessaire à l'obtention de cartes de vitesses devient très rapidement prohibitif (plusieurs heures). Cet inconvénient a grandement contribué à la recherche d'autres techniques permettant l'obtention directe et simultanée de plusieurs points de mesures.

2 - Particle Image Velocimetry

Historiquement, les premiers résultats obtenus concernaient simplement la visualisation d'un écoulement par injection de traceurs le plus souvent diffus. Clayton et Massey [Clayton 67] donnent une revue de ces premières visualisations. Mais les véritables mesures de déplacements réalisés en PIV ont vu le jour avec la tomographie laser [Schon 79]. Cette technique de visualisation permet d'éclairer un plan de l'écoulement à étudier. Une nappe de lumière destinée à éclairer une tranche de l'écoulement est générée à l'aide d'un laser et d'un jeu de lentilles. L'éclairage laser est retenu du fait de son caractère énergétique et de la faible divergence de son faisceau. L'écoulement est ensemencé avec de très petites particules réfléchissantes servant de traceurs, de densité proche de celle du fluide, afin que l'on puisse considérer que les particules suivent l'écoulement étudié sans le perturber. Un capteur (appareil photo, caméra à film, caméra CCD ou imageur CMOS) placé face au plan de lumière acquiert de façon séquentielle des images de l'écoulement (Figure 1-2).

Les images obtenues correspondent à des clichés quasi instantanés et successifs des particules dans l'écoulement, séparés d'un intervalle de temps connu. Le principe de base de la PIV est de déterminer la projection, dans le plan de mesure, du vecteur déplacement. Lorsque l'écoulement est considéré comme bidimensionnel et le capteur strictement parallèle au plan laser, alors la projection mesurée est égale au déplacement du fluide, au grandissement optique près.

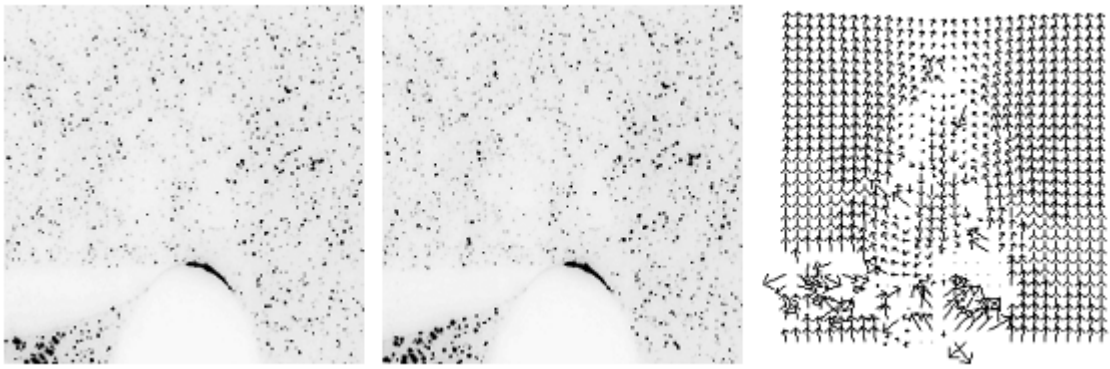


Figure 1-2 : Vélocimétrie par Image de Particules appliquée au sillage à l'aval d'un cylindre dans l'eau avec des fenêtres 64×64 se chevauchant à 75% sur chaque axe, Re=1000 [Fayolle 96].

Le calcul de la vitesse d'une particule localisée en un point $\mathbf{M_0\ (X_0,\ Y_0)}$ revient à mesurer la distance qu'elle a parcourue pendant l'intervalle de temps **dt**. Le vecteur vitesse de la particule est alors défini par la relation suivante :

$$V(M_0,t)=\left[\left(\frac{d(X)}{dt}\right)_{M_0},\left(\frac{d(Y)}{dt}\right)_{M_0}\right] \tag{1-1}$$

Dans la section suivante, nous détaillons la technique de mesure par Vélocimétrie par Images de Particules.

1.2 - Vélocimétrie statistique par images de particules

La technique de PIV est une méthode statistique de mesure de vitesse. Elle est instantanée et bidimensionnelle. Son principe général consiste à enregistrer des images de particules (souvent désignées par les termes marqueurs ou traceurs) à des instants très proches. La comparaison de deux images successives permet de remonter localement au déplacement du fluide et ainsi d'accéder au champ des vitesses à un instant donné. Cette mesure peut se décomposer en trois étapes : la visualisation des écoulements, l'acquisition des images, et enfin l'extraction de l'information de vitesse.

1.2.1 Visualisation des écoulements

Cette étape nécessite tout d'abord d'ensemencer correctement le fluide que l'on désire étudier, puis de générer un éclairage adapté aux conditions expérimentales.

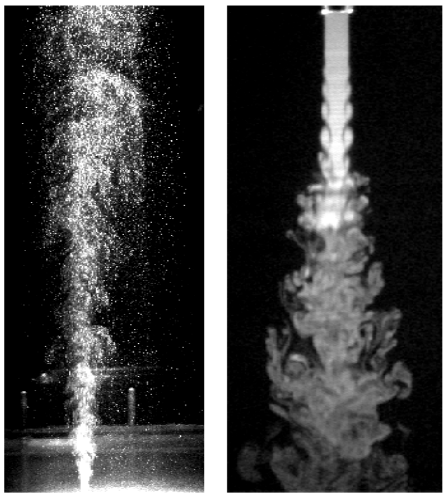
1 - Les traceurs

L'ensemencement consiste à rendre visible les mouvements du fluide en observant le mouvement de matériaux étrangers adéquats (traceurs) ajoutés au fluide en mouvement. Par conséquent, on ne mesure pas directement la vitesse de l'écoulement mais plutôt celle des traceurs en suspension dans

l'écoulement. De plus, la nature du traitement et la qualité de la mesure dépendent très fortement du type de traceurs injectés ainsi que de leur concentration. L'ensemencement dépend des conditions expérimentales : on distingue les traceurs continus (colorants, fumées, etc.) qui donnent en général des informations qualitatives, et les traceurs individualisés (particules, solides, bulles, etc.) qui donnent accès à des grandeurs quantitatives. Il y a une grande diversité de substances susceptibles de jouer le rôle de traceur. A titre d'exemples, nous citons des traceurs trouvés dans la littérature (Tableau 1-1) :

Nature du traceur	Diamètre	Fluide étudié Vitesse	Référence
Traceurs liquides			
Gouttelettes d'huile légère (aérosol)	~ 1 à 2 µm	Air 1,73 m/s	[Meynart 83]
Lait (suspension colloïdale)	~ 3 à 20 µm	Eau 1,6 mm/s	[Grobel 91]
Traceurs gazeux			
Bulles d'air		Eau de 0,52 à 3 m/s	[Stefanini 92]
Traceurs solides			
Fumée d'encens	1 µm	Jet d'air 1,04 m/s, Re=830	[Zaouali 04]
Billes de verre	~ 9 µm	Jet d'eau 0,2 m/s, Re=1015	[Coudert 02]

Tableau 1-1 : Quelques types de traceurs utilisés pour la visualisation des écoulements.



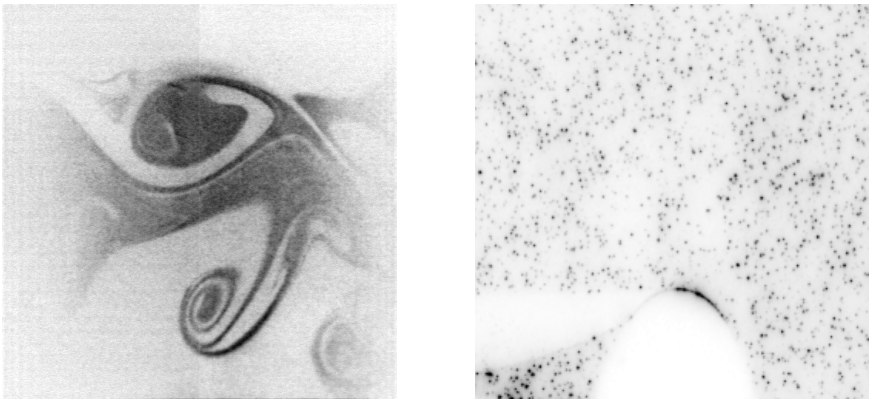
a) b)

Figure 1-3 : Etudes réalisées sur :
a) jet d'eau, billes de verre, Re=1015,
b) jet d'air, fumée d'encens, Re=1800.

La Figure 1-3 illustre les deux types de jets et les traceurs utilisés pour les étudier. Les traceurs sont souvent solides dans les gaz et les liquides, mais on peut aussi utiliser comme traceurs des bulles gazeuses dans les liquides, ou des gouttelettes liquides dans les gaz. Dans ce cas, le traceur est présent naturellement dans l'écoulement. On parle alors de traceurs intrinsèques. On trouve également un certain nombre de traceurs qui interagissent avec la lumière laser (particules fluorescentes, réaction photochimique, etc.). L'ensemencement d'un écoulement se résume à deux choix : l'un relatif aux tailles des traceurs, l'autre à la concentration de traceurs dans l'écoulement.

Le choix de la taille des traceurs est basé sur les deux principaux aspects suivants : l'aspect mécanique des fluides et l'aspect optique. Pour la mécanique des fluides, les traceurs doivent évidemment être de petite taille afin de ne pas perturber l'écoulement et de masse volumique la plus proche possible de celle du fluide porteur. L'aspect optique, en première approche, est contradictoire avec le précédent. En effet, plus la taille des traceurs est importante, plus ils réfléchissent de lumière. En réalité, d'autres phénomènes interviennent, en particulier des phénomènes de masquage lorsque la concentration en traceurs est importante. Le choix de la taille des traceurs est donc soumis à un compromis. Le plus souvent, on fera appel à des traceurs de tailles comprises entre 1 microns (dans l'air) à 10 µm (dans l'eau).

La concentration en traceurs est l'un des facteurs les plus importants pour réussir des mesures par PIV. Deux grands types de méthodes peuvent être distingués suivant la concentration du traceur. Lorsque cette concentration est suffisamment faible, chaque traceur éclairé par un faisceau de lumière cohérente peut être traité individuellement (Figure 1-4b). Dans le cas contraire, les traceurs ne sont pas individualisés mais traités comme une entité globale (Figure 1-4a). Il est alors possible de mesurer les champs de vitesse en utilisant la Vélocimétrie par Suivi de Frontière (VSF) [Fayolle 96], mais le rapport signal/bruit est beaucoup plus faible qu'avec la PIV.



a) b)

Figure 1-4 : Exemples d'utilisation de traceurs :

a) traceurs diffus composés d'huile vaporisée, incorporés à un écoulement d'air ;
b) traceurs ponctuels composés de particules à base de matières plastiques pour la visualisation du sillage d'un cylindre dans l'eau.

Dans le premier cas, la concentration doit être choisie pour que les images individuelles des traceurs soient visibles. Il est donc essentiel d'ajuster avec le plus grand soin la concentration en traceurs. Comme la PIV est une technique statistique, il est nécessaire d'avoir suffisamment d'informations à l'intérieur d'une image c'est à dire un nombre de traceurs suffisamment important. En pratique, une concentration de 10^{10} à 10^{11} traceurs par m³ est couramment utilisée [Coudert 02, Fayolle 96, Riou 99].

2 - Plan lumineux

Lorsque l'on éclaire un écoulement avec un éclairage direct, l'image observée représente la projection d'une scène éclairée en trois dimensions sur un plan. Une autre approche consiste à n'éclairer qu'un plan de l'écoulement et l'image bidimensionnelle obtenue décrit précisément la scène définie par le plan. Cette technique appelée "tomographie" permet de visualiser de façon plus précise une partie du phénomène. La géométrie du plan laser dépend du montage optique utilisé. Le rayonnement laser est utilisé pour ses qualités de collimation et de son caractère énergétique. Un faisceau laser passant à travers une lentille cylindrique forme une nappe de lumière d'une épaisseur de l'ordre du millimètre. Cet éclairage "découpe" une tranche de l'écoulement dans la direction de propagation (Figure 1-5). L'appareillage de prise de vues est disposé perpendiculairement à la nappe de lumière [Schon 92].

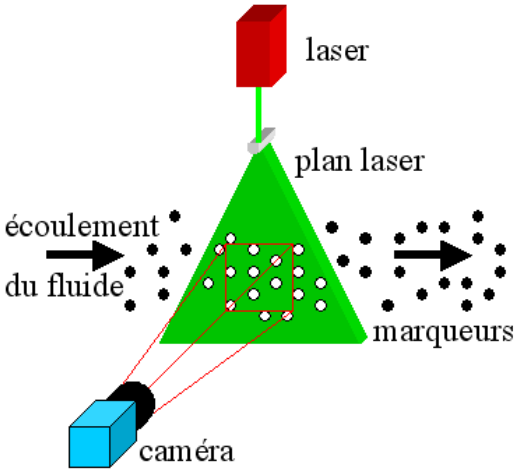


Figure 1-5 : Nappe laser et positionnement de la caméra.

La technique de tomographie est associée à l'ensemencement du fluide puisque ce sont les propriétés de réflexion des traceurs qui sont exploitées. Les deux types de traceurs, diffus et ponctuels peuvent être utilisés, seule la technique de traitement des images change. Le choix des traceurs doit cependant être fait de telle sorte que la nappe laser puisse pénétrer dans la totalité du champ d'étude évitant ainsi les zones d'ombre.

Puisque les traceurs sont en mouvement, il est nécessaire de réaliser un effet stroboscopique pour figer leurs images, et ainsi éviter le filé de l'image du au mouvement. Cet effet peut être obtenu de deux manières différentes : soit en utilisant des sources lasers pulsées qui réalisent d'elles-mêmes

l'effet stroboscopique, soit en utilisant des sources lasers continues couplées à des systèmes d'obturation.

Laser pulsé Les lasers pulsés (rubis, Nd-Yag, vapeurs de cuivre) peuvent délivrer des impulsions très courtes (< 20 ns) dont la période peut être réglée. L'énergie lumineuse délivrée à chaque impulsion est très supérieure à celle d'un laser continu pendant le même intervalle de temps. L'aspect énergétique de l'éclairage est intéressant pour les petits champs et les phénomènes rapides car leur observation conduit à utiliser des particules de plus en plus petites. La puissance de l'éclairage devient alors un problème capital auquel le laser pulsé apporte une solution.
Cependant, les lasers pulsés ont le plus souvent une faible cadence de répétition (< 100 Hz). Aussi, pour obtenir deux impulsions très rapprochées l'une de l'autre, deux lasers pulsés sont utilisés. Leurs faisceaux sont alignés de sorte qu'ils ne forment plus qu'un seul faisceau laser. Ce système se nomme un laser à deux impulsions. L'intervalle de temps entre deux impulsions laser est commandé par un boîtier de synchronisation. La valeur de l'intervalle peut aller de 0 à quelques dizaines de millisecondes (cette dernière valeur étant inférieure à la fréquence de répétition d'une impulsion).

D'un point de vue pratique, l'utilisation de lasers pulsés est délicate du fait des importantes énergies mises en jeu. De plus, ce sont des systèmes lourds et difficilement transportables (encombrement, fragilité, nécessité d'un circuit hydraulique de refroidissement, etc.). Leur prix est par ailleurs relativement plus élevé que celui de sources continues de moindre puissance. Pour toutes ces raisons, l'utilisation du laser pulsé reste limitée.

Laser continu La mise en oeuvre d'un système de tomographie laser est facilitée par l'utilisation de sources continues. Les possibilités d'exposition des images (impulsions à durée et périodicité réglable) ne sont plus conditionnées par la source lumineuse mais par un système d'obturation pilotable. Deux types d'obturation peuvent être distingués. Il y a d'une part, les obturateurs agissant sur le faisceau laser et d'autre part, les obturateurs agissant sur les systèmes mêmes d'acquisition d'images (interne ou externe).

Pour obturer le faisceau laser, un obturateur mécanique ou un déflecteur acousto-optique est utilisé. Pour obturer le capteur de la caméra, un obturateur électronique ou mécanique est utilisé. Par exemple, un obturateur électronique est intégré à la majorité des caméras. Les obturateurs permettent de créer plusieurs obturations du faisceau laser. Ces obturations peuvent être effectuées à différents intervalles de temps les unes des autres, ainsi qu'avec une durée différente. Ceci permet d'obtenir un codage sur les images acquises. Par exemple, une particule apparaît à l'image sous la forme d'un trait puis d'un point.
L'inconvénient majeur réside dans le fait que la puissance lumineuse acquise sur les images est faible. Toutefois, il est possible d'utiliser des caméras avec un intensificateur de lumière ce qui apporte une solution d'obturation satisfaisante [Zara 96]. Il permet tout d'abord un réglage précis du temps

d'exposition des caméras par une simple commande électronique, jusqu'à des temps de l'ordre de la dizaine de nanosecondes. Cependant, l'intensificateur introduit un bruit supplémentaire sur les images [Zara 97].

Les lasers continus ont bien sûr le gros inconvénient de délivrer une énergie instantanée bien moindre que les lasers pulsés. Toutefois, leur utilisation permet une souplesse de réglage des temps d'exposition que les lasers pulsés ne peuvent donner.

1.2.2 Acquisition des images

Le principe général de l'acquisition des images repose sur l'enregistrement de deux images successives qui sont numérisées puis stockées en mémoire pour être traitées numériquement dans une phase ultérieure. Ces traitements permettent alors de calculer, via des algorithmes décrits dans le paragraphe 1.2.3, les champs de vecteurs vitesse de l'écoulement. Nous allons tout d'abord détailler les différents types de capteurs possibles, puis nous exposerons comment les mettre en oeuvre.

1 - Capteurs de prise d'images

Les premiers dispositifs utilisés pour l'acquisition des images ont été les caméras à film et les appareils photographiques, actuellement ils sont quasiment supplantés par les capteurs d'images électroniques.

Les caméras à film permettent d'acquérir des images à plus grande vitesse que les caméras électroniques (caméra à film de la société Cordin [Cordin-web] : de 35000 à 25 millions d'images/s). Cependant, les traitements du film (développement puis numérisation) restent fastidieux avant le traitement des données proprement dit. Aussi, elles ne sont utilisées que dans des cas très spécifiques.

Les caméras électroniques sont majoritairement utilisées du fait de leur simplicité de mise en oeuvre. Elles permettent une exploitation directe des images par des outils informatiques. Aujourd'hui, deux sortes de capteurs électroniques sont disponibles sur le marché : les capteurs CCD (*Charge Coupled Device*) et CMOS (*Complementary Metal Oxyde Semiconductor*). Les principes fondamentaux de ces technologies de capteurs d'images seront abordées et détaillées dans la partie B chapitre 3.

2 - Contrôle de l'acquisition

Exposition du capteur La majorité des caméras électroniques possède un obturateur électronique intégré (*shutter*). Sinon, on peut toujours ajouter un obturateur optique au système d'acquisition. Il est alors possible d'exposer le capteur de deux façons différentes : la mono-exposition et la multi-exposition.

<u>Mono-exposition :</u> l'exposition du capteur est réalisée une seule fois : une particule apparaît sous la forme d'un point ou d'un trait (Figure 1-6).

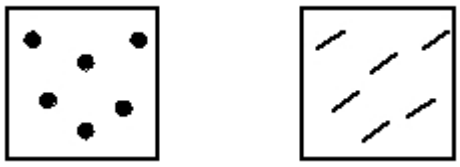


Figure 1-6 : Illustration de la mono-exposition.

Pour que chacune des particules apparaisse comme un trait et non comme un point, il suffit de choisir le temps d'exposition du support d'enregistrement suffisamment long. La longueur et la direction de chacun de ces traits renseignent sur le module et la direction du vecteur déplacement du fluide en ces points. Connaissant la valeur du temps d'exposition du capteur, il est aisé de remonter au champ de vitesse.

Cette méthode présente l'avantage de fournir un ensemble de mesures de vitesses à partir d'une seule image de l'écoulement. Ceci permet son exploitation même lorsque la vitesse du fluide est très importante. De plus, les résultats obtenus par cette méthode présentent de bonnes précisions [David 92].

Multi-exposition : l'exposition du capteur est réalisée plusieurs fois sur la même image : une particule apparaît sous la forme soit d'un trait pointillé, soit d'un trait-point, soit d'une toute autre combinaison de points et de traits (Figure 1-7). Dans le cas de la bi-exposition, en utilisant des temps d'exposition très courts, les traces des particules n'apparaissent plus comme des traits continus mais comme deux points. La distance entre ces deux points fournit l'information de déplacement de la particule pendant le temps d'inter-exposition. Pour obtenir une combinaison de trait-point, il faut exposer le support d'enregistrement deux fois mais avec des temps d'exposition différents (long-court).

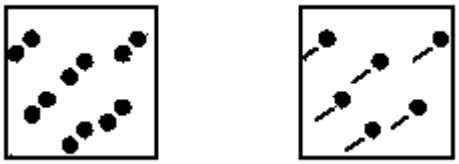


Figure 1-7 : Illustration de la bi-exposition.

Rapidité d'acquisition Suivant l'évolution des technologies, les résolutions des capteurs sont en constante augmentation (7,1 Mégapixel). De tels capteurs sont limités en taux de transfert pixel. Cela impose un temps inter-image minimum non modifiable, donc une cadence d'acquisition fixe. Cependant, il existe des techniques qui permettent de réduire ou contrôler au mieux ce temps d'inter-exposition. Elles consistent soit à réduire le nombre de pixels acquis soit à multiplier le nombre de plan de stockage.

Accès aléatoire aux pixels (*Random Access*) : avec le développement des capteurs de type CMOS, il est apparu la possibilité d'accéder à l'information de chaque pixel par un adressage en X et en Y de la matrice du capteur. Cette fonction permet de ne lire qu'une zone définie voire plusieurs zones dans une même image par l'utilisateur de la matrice. Elle permet donc de définir des régions d'intérêt et d'éliminer la ou les parties des images ne présentant aucun intérêt. Ceci offre l'avantage d'une réduction majeure du temps de transfert de l'image.

Caméras possédant un double plan de stockage : des caméras récentes dédiées à la PIV par inter-corrélation intègrent au moins un plan mémoire supplémentaire pour stocker le plus rapidement possible une première exposition afin de réaliser la seconde dès que possible. Ainsi, le temps d'inter exposition peut être raccourci à quelques dizaines de nanosecondes (la caméra DiCam Pro de la société PCO [PCO-web] permet maintenant d'obtenir une paire d'images rapprochées de 500 ns avec un taux de répétition de 7 images par seconde).

Multi-capteurs CCD : Certaines caméras possèdent un tableau de CCD. Le capteur principal est constitué de plusieurs matrices CCD distincts côte à côte. Chaque matrice est illuminée successivement. Plusieurs images sont donc acquises à une fréquence élevée, puis les images sont transférées. Dans le même ordre d'idée, certaines configurations possèdent plusieurs capteurs CCD disposés sur un arc de cercle. L'image est distribuée sur ces différents capteurs à l'aide d'un miroir tournant. La fréquence des images est de l'ordre de quelques millions d'images par seconde sur une dizaine d'images (le système IMACON 468 de Hadland Photonics [Honour 01] permet de réaliser jusqu'à huit images successives séparées de 10 ns à partir de huit capteurs CCD indépendants).

Multi-caméras : Une autre solution est une configuration utilisant un système composé de plusieurs caméras dont les images sont recalées par un calibrage préalable. Un système bi caméra a été développé au sein du laboratoire TSI qui permet d'obtenir des paires d'images rapprochées de quelques centaines de nanosecondes (largement suffisant pour bon nombre d'applications). Ces deux caméras sont disposées en face l'une de l'autre. Elles sont focalisées sur le même plan laser à mi-distance entre les deux caméras. Cette configuration présente l'avantage de ne pas avoir à développer un produit spécifiquement dédié. Seuls des algorithmes de traitement spécifique des images sont à utiliser [Riou 99].

1.2.3 Le traitement d'image pour la PIV

A partir du moment où les images ont été correctement enregistrées, il existe tout un cheminement permettant de transformer deux images successives (couple d'images mono-exposées) en

un champ de vecteurs vitesse. Nous développons ici les différentes étapes et algorithmes conduisant à l'obtention des champs de déplacements.

Pour multiplier les points de mesures, chacune de ces techniques nécessite une décomposition des images en multiples zones d'interrogation [Adrian 91]. Les différentes techniques employées pour effectuer la mesure instantanée de vitesse sont toutes basées sur des méthodes statistiques inspirées de la corrélation. Suivant le type d'images (mono-exposées, bi-exposées) nous utilisons, soit des techniques d'inter corrélation, soit des techniques d'auto corrélation.

Il est important de noter qu'il existe une condition de fonctionnement pour toutes ces techniques. La mesure étant statistique, elle est moyennée spatialement sur l'ensemble de la zone d'interrogation. Cela signifie que les mesures de PIV ne sont valables que si le déplacement est homogène à l'intérieur de la zone d'interrogation. Cette idée de base de la PIV étant rappelée, nous allons maintenant décrire la méthode de décomposition des images en zones d'interrogation, appelée encore partitionnement.

1 - Partitionnement des images

Chaque image est d'abord divisée en petites régions rectangulaires que l'on nomme fenêtres d'interrogation ou imagettes. A l'intérieur de ces imagettes, une mesure ponctuelle est réalisée (Figure 1-8). Le procédé utilisé par la majorité des auteurs pour positionner les zones d'interrogation de manière régulière est un pavage de chacune des images par des zones d'interrogation jointives et de mêmes dimensions.

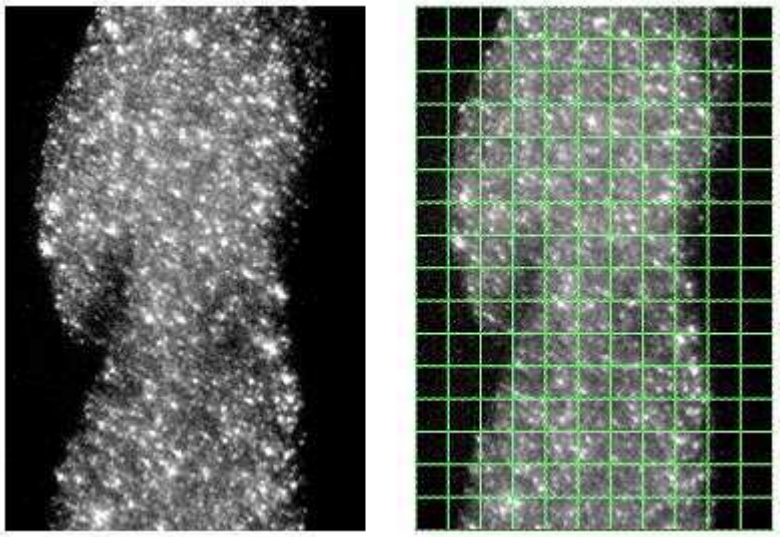


Figure 1-8 : Découpage d'une image de jet d'eau de 704×1024 pixels en fenêtres de 64×64, soit 176 fenêtres d'interrogation.

A taille d'images fixe, la taille des imagettes (en pixels) détermine le nombre total de vecteurs qui seront calculés pour un couple d'images. Cette taille est fixée selon la vitesse de déplacement des particules dans l'écoulement ainsi que selon la fréquence d'acquisition de la caméra. Dans de

nombreux cas, des fenêtres de 32×32 pixels suffisent à avoir un bon compromis entre résolution spatiale et résolution dynamique (étendue de l'échelle des vitesses mesurables). De plus, comme la corrélation est souvent effectuée à l'aide de transformées de Fourier rapides (*Fast Fourier Transform* ou FFT), les imagettes sont choisies de forme carré et de taille en puissance de 2. Les tailles les plus souvent rencontrées dans la littérature sont : 16×16, 32×32 et 64×64 pixels.

Quelques auteurs proposent d'utiliser un partitionnement avec un recouvrement des imagettes adjacentes (suivant les deux axes). Cette procédure n'augmente pas la quantité d'information concernant la résolution spatiale (le nombre de pixels étant le même) mais permet d'avoir plus de vecteurs vitesse pour un même couple d'images. Ainsi un recouvrement de 50% sur chaque axe permet d'obtenir approximativement quatre fois plus de vecteurs [Willert 91]. Cette technique de définition du partitionnement est utilisée sur des écoulements présentant des vitesses moyennes globalement différentes dans toute l'image ou pour réduire le nombre de mesures fausses obtenues avec un partitionnement jointif.

En revanche, il existe d'autres possibilités de définir un partitionnement. En particulier, il est possible que les imagettes n'occupent pas respectivement la même place dans l'image. Cette méthode peut être utilisée dans le cas où l'écoulement moyen est connu [Westerweel 97, Lecordier 99].

A partir des fenêtres d'interrogation, les vitesses sont déterminées avec l'algorithme approprié selon le type des images. Pour déterminer un vecteur de déplacement, nous utilisons successivement deux algorithmes : un algorithme pour déterminer l'image de corrélation, puis un algorithme pour rechercher et mesurer la position du maximum de corrélation.

2 - Carte de corrélation

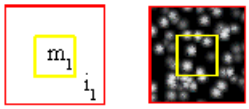
Le premier algorithme utilisé permet de déterminer la corrélation sur chaque fenêtre d'interrogation. Dans le cas où les images de particules seraient des images mono exposées, l'inter corrélation entre deux images est réalisée. Dans le cas où les images de particules seraient des images multi-exposées, l'auto-corrélation de l'image est réalisée. Ce dernier cas revient à corréler l'image avec elle-même.

La corrélation des deux images peut être réalisée de deux façons. Soit directement dans l'espace image : ce type de corrélation est appelé corrélation directe. Soit en passant par l'espace des fréquences : ce type est alors appelé corrélation par transformée de Fourier.

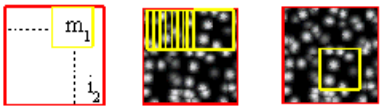
Ces 2 méthodes sont basées sur l'hypothèse restrictive que l'écoulement est localement uniforme (à l'échelle de la fenêtre de calcul : taille de l'imagette). Le résultat de l'opération de corrélation est représenté sous forme d'une imagette résultat en niveaux de gris (à valeur dans l'ensemble des réels) appelée imagette de corrélation. Cette imagette de corrélation possède une zone de valeur plus élevée qui correspond à la zone du maximum de corrélation entre les 2 imagettes i_1 et i_2 (Figure 1-9).

i_1 i_2

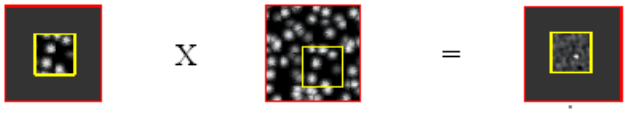
le couple d'imagettes extraites des deux images



le motif m_1 choisi dans l'imagette i_1



opération de corrélation directe : le motif m_1 est déplacé sur l'imagette i_2, afin de rechercher la correspondance des particules du motif avec les particules de l'imagette



i_c

le motif m_1 de l'imagette i_1 est trouvé dans l'imagette i_2 pour le maximum de la fonction f_c. Ce maximum est représenté sur l'imagette i_c par le point blanc.

Figure 1-9 : Opération de corrélation directe entre 2 imagettes avec un facteur de réduction r=0,5.

Corrélation Directe Un motif m_1 est créé à l'intérieur de la première imagette i_1. Ce motif m_1 est déplacé sur i_2 de manière à trouver la meilleure correspondance entre les particules de i_1 et celles de i_2. Cette correspondance est réalisée pour le maximum de la fonction de corrélation f_c telle que :

$$f_c(i,j) = \frac{\sum_x \sum_y s_1(x,y) \times s_2(x-i, y-j)}{\sqrt{\sum_{x,y} s_1(x,y)^2} \sqrt{\sum_{x,y} s_2(x-i, y-j)^2}} \qquad (1\text{-}2)$$

où s_1 représente le niveau de gris au point M de coordonnées (x,y) du motif m_1 extrait de la première imagette i_1 et s_2 celui dans la seconde imagette i_2 avec un déplacement de (i,j) pixels.

L'image résultante i_c est dite : imagette de corrélation. A chaque position du motif m_1 sur i_2, la fonction f_c est calculée. La valeur de f_c est reportée dans l'imagette de corrélation sous forme de niveaux de gris à valeur dans l'ensemble des réels. La taille de l'imagette de corrélation est en relation directe avec la taille du motif.

Sur l'imagette de corrélation i_c de la Figure 1-9, le facteur de réduction est de 0,5. Le déplacement maximal mesurable est inférieur au quart de la taille de l'imagette i_1. Si le déplacement est supérieur au quart de l'imagette i_1, le pic de corrélation est en dehors de l'imagette i_c. Le déplacement ne serait donc pas représenté sur la fenêtre de corrélation.

Corrélation binaire Une méthode similaire à la corrélation directe consiste à utiliser des images binaires à la place d'images en niveaux de gris. Cette réduction de la quantification des images représente une perte d'information. Cependant, comme la PIV fonctionne surtout par reconnaissance de la forme des particules et non par niveaux de gris, cette perte d'information peut être acceptable face à la réduction de la complexité des calculs dans le cas où la rapidité est essentielle. Cette technique de mesure de déplacement est connue sous le nom de corrélation binaire et présente les mêmes avantages que les méthodes par corrélation en niveaux de gris décrites précédemment.

L'utilisation d'images binaires permet de remplacer l'opérateur de multiplication de la corrélation directe par l'opérateur logique de comparaison (Ou exclusif inversé : XNOR), la fonction de corrélation binaire f_{cb} est alors définie par l'équation suivante :

$$f_{cb}(i,j) = \sum_x \sum_y s_1(x,y) \quad XNOR \quad s_2(x-i, y-j) \tag{1-3}$$

Le pic de corrélation correspond à la position pour laquelle le nombre de pixels identiques entre les deux imagettes est maximum. Ce maximum correspond à la position la plus probable du motif m_1 dans l'imagette i_2, soit au déplacement le plus probable entre les deux acquisitions successives.

Corrélation par Transformée de Fourier La fonction de corrélation f_c peut être estimée en passant dans le domaine fréquentiel sous la forme :

$$f_c = TF^{-1}\left(TF(s_1) \times TF(s_2)^*\right) \tag{1-4}$$

Les transformées de Fourier des imagettes sont multipliées dans le domaine des fréquences (Figure 1-10). Puis le retour au domaine spatial fournit l'imagette de corrélation i_c à niveau de gris réel.

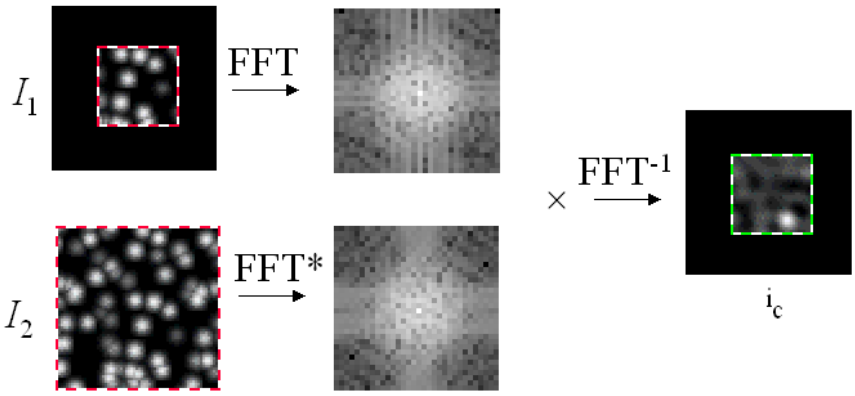


Figure 1-10 : Opération de corrélation par FFT entre 2 imagettes.

L'utilisation de l'algorithme *Fast Fourier Transform* (FFT) impose le partitionnement des images en fenêtres dont la taille est une puissance de 2.Comme pour l'opérateur de corrélation Directe,

à chaque taille de fenêtre correspond un déplacement moyen. Wilbert et Gharid [Wilbert 91] préconisent un déplacement ne devant pas dépasser le tiers de la taille des imagettes.

3 - Détermination de la position du pic de corrélation

Quelle que soit l'opération de corrélation choisie le résultat se présente sous la forme d'une imagette à niveaux de gris (imagette de corrélation) possédant un maximum (pic de corrélation). La localisation du pic de corrélation dans l'imagette de corrélation permet de déterminer le déplacement entre les deux prises de vues. Dans le cas d'une inter-corrélation, ce pic est le pic principal de l'image (Figure 1-11).

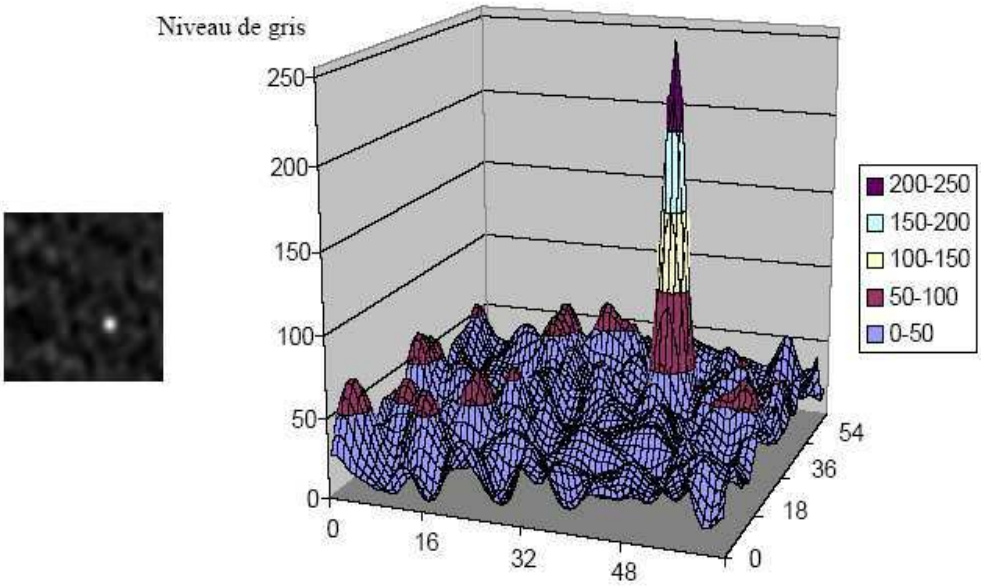


Figure 1-11 : Représentation 2D et 3D d'une imagette de corrélation.

Le vecteur déplacement a pour origine le centre de l'imagette de corrélation et pour extrémité le maximum du pic de corrélation. Le pic de corrélation est tout d'abord localisé en déterminant la zone où le niveau de gris est maximal dans la fenêtre de corrélation.

Dans sa forme initiale, la localisation du pic de corrélation se fait au pixel près. Dans la zone de niveau de gris maximal, on recherche le pixel situé au centre de cette zone par l'utilisation d'une grille dont le pas est la taille d'un pixel. Néanmoins, il est possible de mesurer des déplacements avec des précisions inférieures à la taille du pixel. Les opérateurs subpixels développés reposent sur l'interpolation du pic de corrélation par une fonction donnée (gaussienne, parabole…).

Trois types de méthodes sont possibles pour estimer le déplacement avec une précision inférieure au pixel [Lourenco 95, Coudert 98] :

- la recherche du centroïde du pic de corrélation : le maximum du pic de corrélation correspond au barycentre géométrique des points du pic pondérés par leur niveau de gris,
- l'ajustement par les moindres carrés de la forme du pic de corrélation par une fonction continue, et recherche du maximum de cette fonction,

- l'interpolation du pic de corrélation par une fonction (gaussienne ou parabolique), et recherche du maximum de cette fonction.

4 - Post-traitement des champs de vecteurs

La PIV est une technique de mesure instantanée : toutes les informations sont échantillonnées au même instant. Il est par conséquent probable que certaines régions n'aient aucune signification physique. Pour remédier à ce problème, il existe des méthodes mathématiques qui permettent de valider ou non les champs de vecteurs. Il n'y a pas de méthode unique permettant d'effectuer une telle validation : par exemple, il est possible d'agir sur la hauteur et la largeur du pic de corrélation ou encore sur le module de la vitesse en délimitant les vitesses mini et maxi au delà desquelles les vitesses calculées seront erronées... Enfin, il est nécessaire de calculer un champ de vitesse moyen à partir de l'ensemble des champs instantanés de vecteurs calculés sur chaque couple d'images ainsi que les propriétés statistiques (moyenne, variance, écart-type, coefficient de corrélation, etc.). Le nombre de champs instantanés nécessaire au calcul du champ moyen dépend évidemment des conditions expérimentales.

1.3 - Conclusion

La technique PIV repose essentiellement sur l'aptitude du capteur d'image à capter les positions initiale et finale des particules contenues dans un plan de l'écoulement afin de pouvoir calculer les champs de vecteurs correspondants. La qualité des mesures dépend donc de la qualité des images enregistrées. Pour cela, il faut conjuguer un bon ensemencement avec une bonne illumination.

Lors d'une mesure de déplacement, l'inter-corrélation permet d'obtenir directement une information sur le sens de déplacement et la vitesse en deux dimensions. Il y a quelques années, l'utilisation d'images mono-exposées nécessaire pour cette technique rendait difficile son application à l'étude des écoulements rapides. Mais, le développement des systèmes multi-capteurs et de caméras spécifiques à la PIV permet aujourd'hui de réaliser des paires d'images mono-exposées très rapprochées dans le temps (dizaines de nano-secondes). Nous avons donc retenu la méthode d'inter-corrélation pour la suite de ce travail de thèse.

Cette technique est, en raison du volume de calcul nécessaire, généralement réalisées en temps différé sur des stations de travail. Il est cependant très intéressant d'obtenir un traitement en temps réel des images. L'aspect traitement temps réel apporte les avantages de l'exploitation immédiate des mesures de vitesse. Deux approches sont possibles : l'une qui consiste à accélérer les calculs par des processeurs dédiés et l'autre à développer des systèmes de vision (caméra + processeur) pour des mesures à la cadence d'acquisition.

Chapitre 2

Plates-formes expérimentales

Dans ce chapitre, nous présentons divers montages expérimentaux scientifiques et commerciaux utilisés pour faire des mesures en utilisant la Vélocimétrie par Image de Particule (PIV).

2.1 - Système expérimental classique

2.1.1 Principe

Pour exploiter tous les avantages de la technique de PIV, un système doit pouvoir obtenir des champs de vecteurs précis et quasi-instantanés. La précision du contrôle de synchronisation et l'activation séquentielle des composants principaux du système sont des éléments essentiels. La précision de la commande doit être complétée par un système de traitement flexible et à l'architecture ouverte pour offrir un accès facile aux données brutes et traitées.

Un système typique de PIV se décompose en 4 groupes de fonctions :

- le système de contrôle,
- la visualisation de l'écoulement,
- l'acquisition d'images,
- l'analyse et l'affichage des images.

La Figure 2-1 expose les interactions entre ces groupes de fonctions. Ces groupes de fonctions regroupent divers composants matériels et/ou logiciels. En fonction de la conception de la plate-forme expérimentale, plusieurs fonctions peuvent être regroupées au sein d'un même composant comme, par exemple, un ordinateur pour le système de contrôle et l'analyse des images.

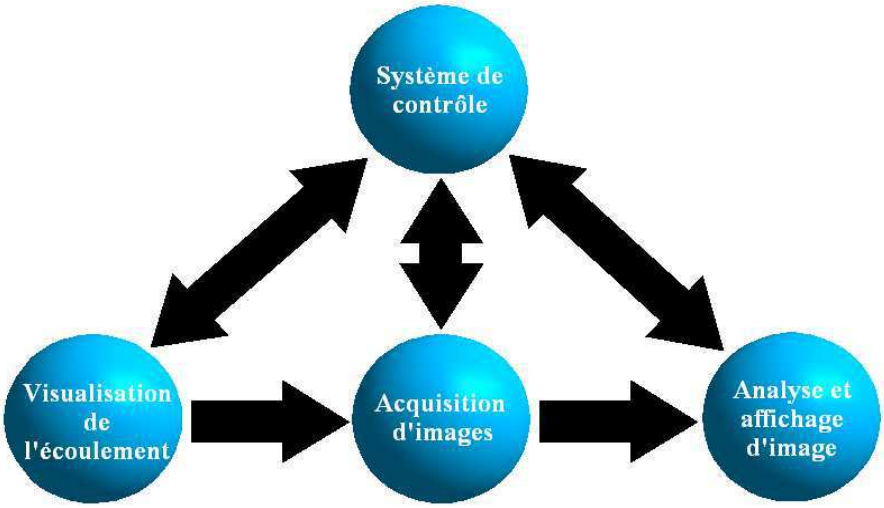


Figure 2-1 : Diagramme d'un système typique de PIV.

2.1.2 Le dispositif du laboratoire

La Figure 2-2 montre notre dispositif expérimental pour la mesure de mouvement par PIV. Elle illustre l'installation expérimentale pour visualiser l'évolution de l'écoulement d'un jet d'eau, ensemencé de particules, dans de l'eau. Cette plate-forme expérimentale peut se décomposer en trois éléments distincts : le système générateur de mouvement, le système d'éclairage et le système de vision.

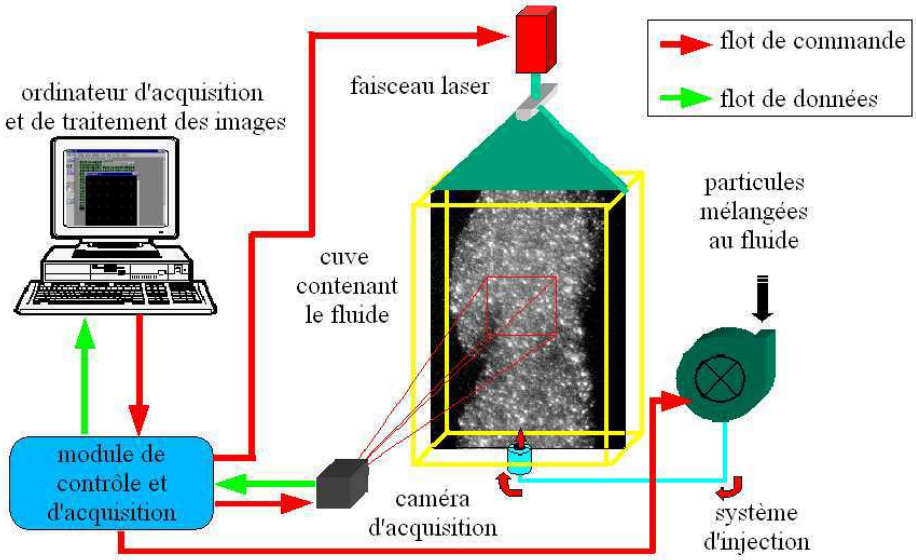


Figure 2-2 : Schéma de notre installation expérimentale de PIV mono caméra.

Le système générateur de mouvement est constitué :

- d'une cuve en plexiglas afin de permettre les visualisations,
- d'un dispositif d'injection d'eau situé au centre de la base de la cuve,
- d'un système composé d'une pompe d'alimentation et de réservoirs d'eau et de particules.

Le système d'éclairage est une source d'illumination de type laser continu à Argon associée à un système optique. Composé de lentilles, le système optique permet d'obtenir une tranche laser verticale d'une épaisseur de l'ordre du millimètre. Elle délimite ainsi le plan d'étude du mouvement des particules dans l'axe de l'écoulement.

Le système de vision se compose d'une chaîne d'acquisition des images et d'un ordinateur standard. La chaîne d'acquisition est constituée d'une ou deux caméras (stéréovision) couplées à des cartes d'acquisition PCI. Les caméras sont positionnées sur un axe perpendiculaire à la tranche laser afin d'obtenir une image nette des particules d'ensemencement de l'écoulement. Les cartes d'acquisition permettent de piloter la ou les caméras (un système de synchronisation est nécessaire pour l'utilisation de plusieurs caméras [Zara 97]) et de récupérer les images pour un stockage temporaire en mémoire vive. Puis, ces images sont transférées vers la mémoire de stockage de l'ordinateur. Toutefois, ceci limite la taille des séquences au nombre d'images pouvant être mémorisées et transférées par la carte d'acquisition. Les dernières générations de cartes d'acquisition ont la possibilité de mémoriser directement les images fournies par la caméra dans la mémoire vive du PC. Cela permet une grande rapidité d'acquisition et une grande capacité de stockage.

L'ordinateur, associé aux logiciels adéquats, contrôle l'acquisition, le système de synchronisation et gère le stockage et le traitement des images. Ce dispositif expérimental ne permet pas l'acquisition et le traitement des données en même temps. C'est-à-dire que nous réalisons les mesures de vitesses en temps différés. Après avoir réalisé l'expérience (acquisition et stockage des images de particules), l'ordinateur effectue les calculs et stocke les résultats sous forme d'images de champ de vecteurs. Nous utilisons pour cela un logiciel de traitement développé par le laboratoire TSI en langage C : Wima.

2.2 - Systèmes d'expérimentations industriels

Il existe de nombreuses sociétés de part le monde fournissant des systèmes de vision plus ou moins complets. Pour notre application de mesure du mouvement en mécanique des fluides, nous pouvons classer ces industriels en 2 catégories : les fabricants de caméras et systèmes d'acquisition et les fabricants de plates-formes expérimentales complètes. Dans la suite de ce paragraphe, nous ne parlerons que des sociétés qui commercialisent des systèmes industriels dédiés à la PIV.

Les principaux fabricants proposent tous des systèmes complets comprenant un ensemble matériel et logiciel intégré. Comme l'illustre la Figure 2-3, ces systèmes s'articulent autour d'ordinateurs permettant la gestion et le contrôle de la totalité de la plate-forme expérimentale. Les logiciels permettent l'automatisation de nombreuses opérations sur le système complet de PIV. En plus des composants tels que le laser, le générateur de jet, ces systèmes utilisent des boîtiers de contrôle pilotables par ordinateur. Des caméras couplées avec des cartes d'acquisition PCI complètent les

systèmes en transférant les images en temps réel vers la mémoire de l'ordinateur. Les caméras sont en majorité des caméras de type CCD voire *Intensified* CCD pour des mesures en faible luminosité. Depuis quelques années, les caméras de type CMOS apparaissent dans ces systèmes essentiellement pour deux raisons. Les caméras CMOS apportent à la fois une haute résolution (matrice de 2k×2k) et une vitesse d'acquisition supérieure aux CCD (jusqu'à 2000 images par seconde).

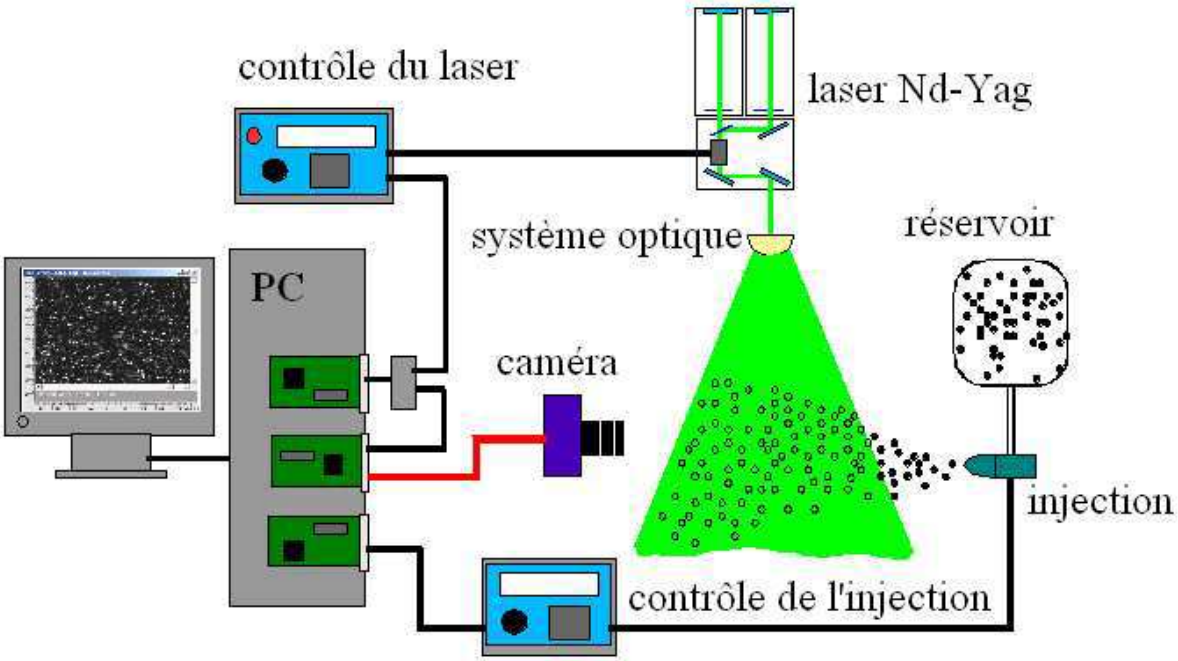


Figure 2-3 : Schéma de principe des systèmes industriels dédiés à la PIV.

Les ordinateurs sont fournis avec des logiciels performants. Ils gèrent l'acquisition, le traitement et l'analyse des images. Les logiciels sont entièrement paramétrables de l'acquisition de séries d'images à l'automatisation des séquences de mesures (de l'acquisition des données jusqu'à la présentation du résultat final). Le progiciel fournit le traitement en direct et l'affichage des champs de vitesse en 2D. Ces champs de vitesse peuvent être visualisés au-dessus des images de particules pour une vérification visuelle. A partir des champs instantanés de vitesse, il permet une analyse statistique détaillée des phénomènes. Un mode d'opération de traitement en temps différé permet au système d'utiliser la vitesse maximale d'acquisition de la caméra.

Ces systèmes complets offrent une grande facilité d'utilisation et comportent, en général, la possibilité d'effectuer des mesures avec plusieurs caméras ou des enregistrements de séries d'images.

Actuellement, quatre sociétés proposent, en France, des systèmes entièrement dédiés aux mesures par Particle Image Velocimetry :

- la société LaVision [LaVision-web] ;
- la société Dantec [Dantec-web] ;
- la société TSI [Tsi-web] ;
- La société.HAMAMATSU [Hamamatsu-web].

Ces systèmes industriels sont performants mais doivent être utilisés avec des caméras appropriées et des ordinateurs dernière génération, généralement fournis par l'industriel. De plus, ces sociétés développent leurs propres types de liaisons ou d'interfaces hauts débits, ce qui leur assurent un transfert des images en temps réel. Mais cette fonctionnalité ne réalise que l'acquisition en temps réel, le traitement des données est effectué a posteriori. Le traitement est alors en temps différé. Cela est dû aux débits très élevés des dernières générations de caméras ainsi qu'à leurs hautes résolutions spatiales.

2.3 - Approche temps réel d'un système de PIV

Un système est dit temps réel lorsqu'il est capable d'absorber toutes les informations d'entrée sans qu'elles soient trop vieilles pour l'intérêt qu'elles présentent, et par ailleurs, de réagir à celles-ci suffisamment vite pour que cette réaction ait un sens. Bien que l'évolution relative du système temps réel et de son environnement soit entièrement asynchrone, la réaction d'un système temps réel doit sembler instantanée selon la granularité de temps du changement d'état de l'environnement. En résumé, nous pouvons dire qu'un système temps réel doit être prévisible (prédictible), les contraintes temporelles pouvant s'échelonner.

Il est évident que la structure de ce système dépendra de ces contraintes temporelles. On peut diviser les systèmes temps réel en deux catégories :

- les systèmes dits à contraintes souples ou molles (*soft real time*). Ces systèmes acceptent des variations dans le traitement des données de l'ordre de la seconde car cela ne met pas en péril le fonctionnement correct de l'ensemble du système. Ces systèmes se rapprochent fortement des systèmes d'exploitation classiques à temps partagé.
- les systèmes dits à contraintes dures (*hard real time*) pour lesquels une gestion stricte du temps est nécessaire pour conserver l'intégrité du service rendu.

Les systèmes à contraintes dures doivent répondre à trois critères fondamentaux :

- Le déterminisme logique : les mêmes entrées appliquées au système doivent produire les mêmes effets.
- Le déterminisme temporel : une tâche donnée doit obligatoirement être exécutée dans les délais impartis.
- La fiabilité : le système doit être disponible. Cette contrainte est très forte dans le cas d'un environnement embarqué car les interventions d'un opérateur sont très difficiles ou même impossibles. Cette contrainte est indépendante de la notion de temps réel mais la fiabilité du système sera d'autant plus mise à l'épreuve dans le cas de contraintes dures.

Le traitement du signal et des images utilise des algorithmes de plus en plus sophistiqués dont la complexité ne cesse de croître. Si certains traitements peuvent être effectués en temps différé (hors ligne), d'autres nécessitent un traitement en temps réel des données. C'est le cas de la mesure et du contrôle par traitement d'images qui nécessitent des opérations en temps réel sur un flux élevé de données (images hautes résolutions à haut débit). Beaucoup de ces algorithmes de traitement d'images ne peuvent être réalisés en temps réel sur un ordinateur. Bien que les stations de travail actuelles deviennent de plus en plus rapides et puissantes, le temps de calcul lui aussi devient de plus en plus important. De plus, les stations sont gouvernées par leur système d'exploitation, ce qui rend le temps de calcul non constant. De nos jours, l'évolution des performances des circuits électroniques à haute densité permet de réaliser des systèmes embarqués afin d'atteindre le traitement en temps réel pour un faible coût.

La contrainte temps réel agit sur les possibilités de conception et influence fortement les choix d'algorithmes. Trois éléments caractérisent le traitement d'images en temps réel : une quantité de données à stocker et à traiter non négligeable, un débit élevé d'acquisition et de traitement et un choix des algorithmes à mettre en oeuvre. De plus, le parallélisme des algorithmes permet d'estimer le coût de calcul en tenant compte de la disponibilité des ressources mémoires et de communications. Il appartient donc au concepteur d'exploiter le pipeline à bas niveau pour satisfaire la contrainte temps réel sur la partie calcul.

L'approche Temps Réel pour la PIV présente des avantages sur deux aspects : l'un pour la partie expérimentation et l'autre pour le traitement des images. Disposer d'un système temps réel apporte plusieurs avantages à l'expérimentation. Le premier avantage est relatif à la commande en temps réel du système de vision en fonction des observations réalisées sur l'écoulement. A partir des mesures de vitesse en temps réel, il est alors possible de réguler le système lui donnant naissance. Un autre avantage de ce type de mesures se situe dans la visualisation en ligne des champs de vecteurs vitesse. Ce type de visualisation peut permettre à un opérateur d'observer directement les résultats des modifications sur les paramètres d'une expérience de PIV. Il peut ainsi réaliser une mesure optimale sur le phénomène à observer.

Sur l'aspect calcul des vecteurs, la parallèlisation implicite des algorithmes permet d'envisager des calculs de PIV en parallèle, et donc une diminution des temps de traitement. Une spécificité des algorithmes de mesure sur des images réside dans le fait que, dans la plupart des cas, les mesures sont reproduites dans plusieurs régions de l'image. L'image peut alors être découpée en zones (nommées imagettes), de ce fait la parallèlisation des calculs est implicite, chaque imagette analysée donnant naissance à un point de mesure. Les mesures étant généralement indépendantes, le traitement de plusieurs imagettes peut être réalisé de manière simultanée, par le même algorithme ou par des algorithmes différents.

En conclusion, le terme temps réel signifie pour cette application de mécanique des fluides :

- la visualisation en direct des phénomènes,

- le calcul des mesures à la cadence d'acquisition des images.

2.4 - Systèmes temps réel développés par le milieu scientifique

Beaucoup des scientifiques travaillent à développer des techniques de traitement d'images qui permettent d'extraire de nombreux paramètres de mesure des images. D'autres optimisent le système de vision dans sa globalité, en ayant pour but d'accélérer les performances et temps de calcul. Puisque ce travail de thèse porte sur le développement d'un système de vision pour la PIV, c'est cette catégorie de travaux que nous allons aborder dans ce paragraphe. Ce paragraphe fait une présentation des principaux travaux scientifiques qui ont développé des systèmes de traitement dédiés à la PIV. Ces travaux sont présentés dans l'ordre chronologique de publication.

Le premier travail présenté [Meinhart 93] n'est pas un système temps réel, mais c'est un des travaux le plus significatif dans le domaine scientifique. Les auteurs proposent un système pour accélérer les calculs de corrélation et l'obtention des champs de vecteurs.

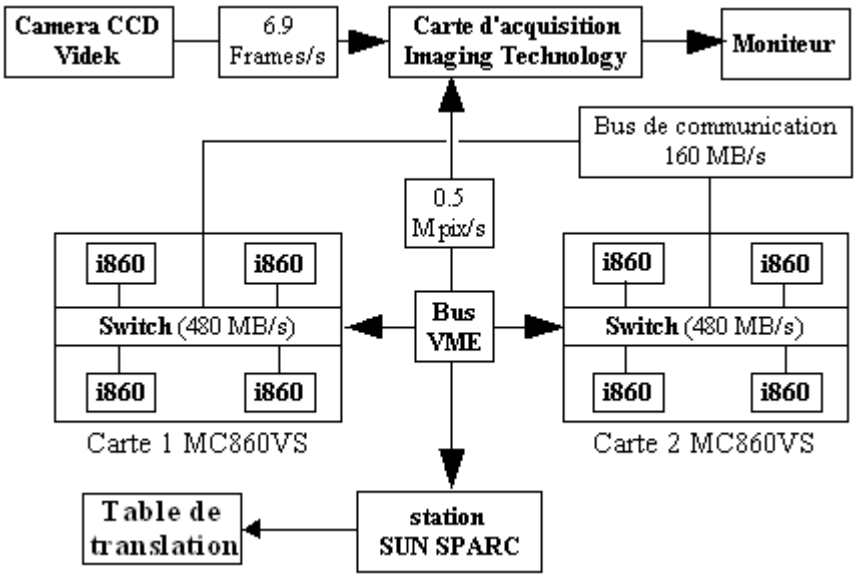


Figure 2-4 : Schéma de l'architecture de Mienhart.

Le système est basé sur un ordinateur contenant une carte d'acquisition couplée à une caméra CCD et de deux cartes multiprocesseurs pour le traitement des données. Avec 8 processeurs Intel i860, le système peut calculer plus de 100 vecteurs par seconde, avec des fenêtres d'observation de taille 128×128, soit 0,5 images 512×512 par seconde. Une réduction de la taille des fenêtres d'observation n'augmente pas les performances du système puisque les fenêtres sont complétées par la valeur 00 pour atteindre le format 128×128.

Le second travail présenté ici [Dubois 01b] se trouve être le travail précurseur à ce travail de thèse au sein du Laboratoire Traitement du Signal et Instrumentation. Il présente un système

électronique multicartes hétérogène (DSP et FPGA) dédié aux traitements d'images en temps réel. Ce système, se plaçant entre la ou les caméras d'acquisition et l'ordinateur, est composée de plusieurs unités de calcul indépendantes mais aussi d'une unité de répartition des données, et d'une unité de commande. Le système a été validé avec des paires d'images binaires de résolution 512×512 pixels. Avec des fenêtres de taille 32×32 soit 256 points de mesure par images, la performance atteinte est de 28000 vecteurs par seconde soit plus de 100 images par seconde avec un total 4 unités de calcul.

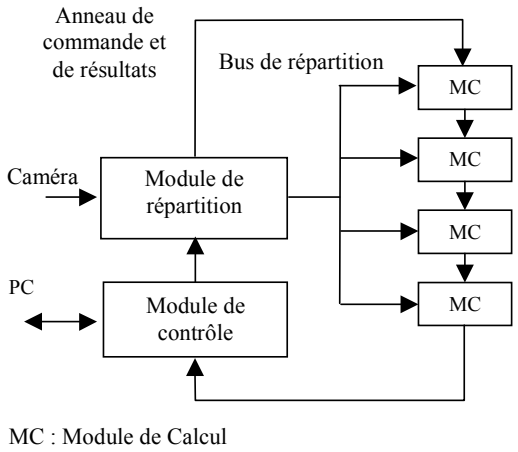


Figure 2-5 : Organisation de l'architecture *Round-About*.

Le dernier travail présenté dans ce paragraphe [Maruyama 01, Fujiwara 03] propose une autre approche. Leur système se compose d'une carte FPGA, à insérer dans un ordinateur, connectée à une caméra CCD. La carte comporte un seul FPGA couplé avec 8 bancs mémoires. Ceci permet de réaliser le traitement en temps réel par la méthode d'inter corrélation. Avec leur implémentation, le nombre de cycles machine pour atteindre le traitement en temps réel est de 470 cycles ce qui correspond à une fréquence de travail de 62,3 MHz. Leur système travaillant à la fréquence de 66 MHz, ces performances satisfont les conditions de traitement en temps réel.

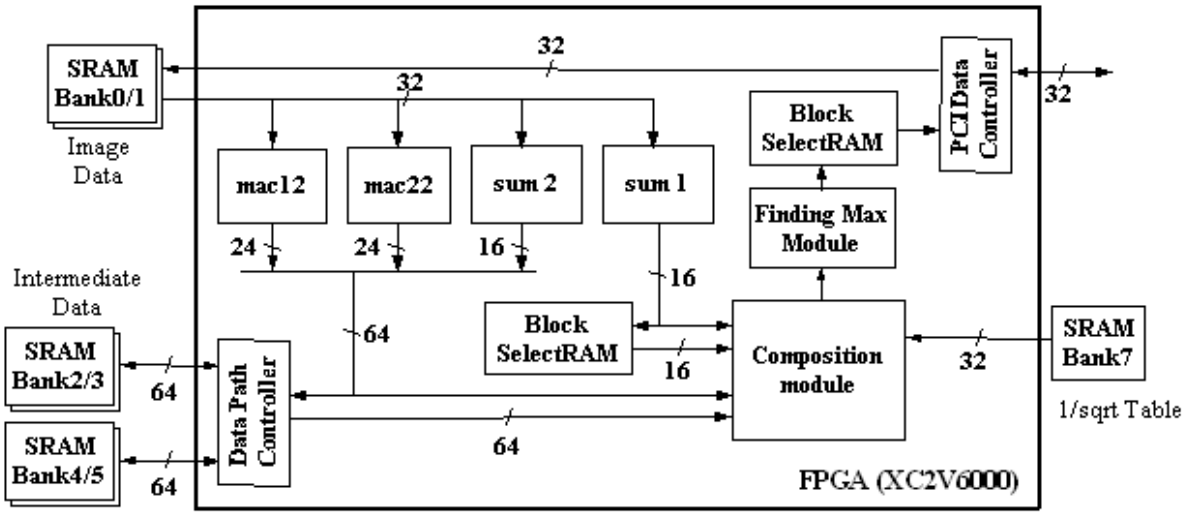


Figure 2-6 : Schéma de l'architecture de Fujiwara.

Ce travail de thèse reprend l'approche système monopuce et FPGA des travaux précédents et des travaux de thèse de Julien Dubois. En plus, nous proposons d'inclure dans l'architecture modulaire *Round-About* la gestion de l'acquisition des images pour des capteurs CMOS. Tout ceci a pour but de réaliser le traitement des images en temps réel avec un système de vision compact et re-programmable.

2.5 - Apports de l'électronique numérique

Les systèmes, exposés dans ce chapitre sont tous plus ou moins performants à divers points de vue (systèmes dédiés, acquisitions temps réel, calculs sur d'importantes quantités de données, etc.). Les performances atteintes n'étaient pas envisageables il y a encore quelques années. Mais, leur mise en œuvre reste encore rigide et ne permet pas leur adaptation simple aux nouvelles technologies sans une nouvelle phase de conception.

C'est dans ce but que nous avons développé notre propre système de vision. Grâce à l'évolution des technologies CMOS, diminution de la taille de gravure (0,18µm) et densité d'intégration élevée (Loi de Moore), nous disposons maintenant de capteurs d'images CMOS facilement intégrables et compatibles avec d'autres circuits électroniques (DSPs, FPGAs et microcontrôleurs) dont la qualité d'image (sensibilité, bruit) égale celle des capteurs CCD. Ces applications industrielles exploitent l'image, et plus particulièrement son contenu, puis transmettent soit sa représentation, soit les résultats d'analyse/traitement, ce qui nécessite une puissance de calcul importante alliée à des moyens de communication. Ce travail, initialement réalisé par des DSP, peut avantageusement être effectué par des systèmes monopuce (plus largement appelé SoCs) qui, non seulement présentent les atouts d'une grandes puissance de calcul, haute intégration, faible consommation (les rendant ainsi complémentaires des capteurs d'image CMOS APS pour la réalisation de capteurs "intelligents" ou de systèmes complets), mais encore, peuvent aisément s'adapter aux divers besoins des concepteurs.

Chapitre 3

Les capteurs d'images

Le principe du capteur d'images est d'échantillonner l'image du plan focal dans les deux directions X et Y. Le point élémentaire d'échantillonnage est le "pixel" (contraction des mots anglais : *picture element*). Ce point élémentaire est une surface de silicium rectangulaire ou carrée. On entend par les mots capteur d'image ou imageur, le système de prise d'image complet. Ce système est constitué d'une matrice de pixels disposés, en général, orthogonalement dans les deux directions et du système électronique capable de lire et de transmettre les données issues des pixels.

Le pixel est le lieu où se situe le système de conversion des photons en électrons. Le détecteur devra être capable ensuite de transformer ces électrons en un signal (tension ou courant) pour être traité par un ensemble de circuits électroniques. Pour cela, le pixel effectue une opération d'échantillonnage qui consiste à prélever la valeur du signal pour des valeurs discrètes de l'espace.

Actuellement, le marché des capteurs d'images, dans le milieu industriel ou scientifique, appartient aux technologies CCD (*Charge Coupled Device*) et CMOS (*Complemantary Metal Oxyde Semiconductor*). Le principe de base de ces technologies est le même. Des photons incidents viennent exciter des électrons se trouvant dans du silicium. La différence entre ces deux technologies réside dans le mode de transfert (ou de lecture) de l'information induite vers la sortie du circuit. Les capteurs CCD ont la particularité de transférer l'information, sous forme de charges analogiques, de pixel en pixel. Les capteurs CMOS classiques transmettent des tensions ligne par ligne à travers des bus-colonnes.

Ce chapitre est entamé avec une explication sommaire des phénomènes physiques qui sous-tendent le fonctionnement des capteurs d'images silicium. Les deux technologies sont ensuite présentées d'un point de vue général, en portant une attention particulière sur les principaux critères de performances qui les définissent, de manière à permettre une meilleure compréhension des caractéristiques souhaitables pour un capteur d'images dédié à notre application. Une présentation détaillée des différents types de capteurs CMOS, ainsi que leurs caractéristiques propres, est exposée.

Une présentation de l'utilisation d'imageurs CMOS en traitement d'images complète ce chapitre puisque nous avons fait ce choix de technologie. Nous conclurons ce chapitre sur la définition d'un capteur CMOS dédiés à notre application.

3.1 - Captation lumineuse à l'aide de semi-conducteur

3.1.1 L'effet photoélectrique

Dans le cas particulier des cristaux semi-conducteurs tels que le silicium, deux bandes d'énergie jouent un rôle fondamental : la bande de valence et la bande de conduction. Ces deux bandes sont séparées par une bande d'énergie interdite, appelée couramment "gap" ou *bandgap*. Dans le cas particulier du silicium, cette bande a une largeur $E_{gap} = 1,12\ eV$ [Sze 01]. La bande de valence correspond aux électrons dits de valence ou liés, assurant les liaisons entre les atomes du cristal, et la bande de conduction correspond aux électrons excités de conduction ou libres. Le passage d'un électron de la bande de valence à la bande de conduction peut être réalisé suivant divers processus de génération, et en particulier par effet thermique ou par absorption d'un rayonnement électromagnétique.

Nous rappelons que l'énergie d'un photon de longueur d'onde λ est donnée par la relation :

$$E_{photon} = \frac{hc}{\lambda} \tag{3-1}$$

Avec : $h = 6,625\times10^{-34}\ J.s$: la constante de Planck, $c = 3\times10^{8}\ m.s^{-1}$: la célérité de la lumière dans le vide et λ : la longueur d'onde du rayonnement électromagnétique (en mètre).

La condition pour que le photon soit absorbé par le cristal est que son énergie E_{photon} soit supérieure à la largeur E_{gap} de la bande interdite [Streetman 99], c'est-à-dire que la longueur d'onde λ du rayonnement doit être inférieure à $\lambda_{coupure}$: longueur d'onde de coupure donnée par la relation suivante [Bhattacharya 96] :

$$\lambda_{coupure}(\mu m) = \frac{1,24}{E_{gap}(eV)} \tag{3-2}$$

Dans le cas du silicium, la longueur d'onde de coupure est approximativement égale à 1,1 μm.

L'électron ainsi excité peut alors quitter la bande énergétique de valence qu'il occupe pour se retrouver dans la bande de conduction, en libérant une charge positive ou "trou", et crée ainsi une paire électron/trou. Ces trous et ces électrons sont libres de se déplacer dans le semi-conducteur et peuvent participer aux courants de conduction ou de diffusion. Ces charges, désignés porteurs en excès ou photoélectrons, sont alors libres de circuler dans le matériau. Le mécanisme d'interaction entre le photon et l'électron dans la zone de déplétion est illustré à la Figure 3-1. Le phénomène d'absorption du photon (1) et de séparation des charges par le champ électrique (2) sont représentés sur le

diagramme de bandes d'énergie en a), alors que b) représente l'accumulation de charges en excès dans le semi-conducteurs.

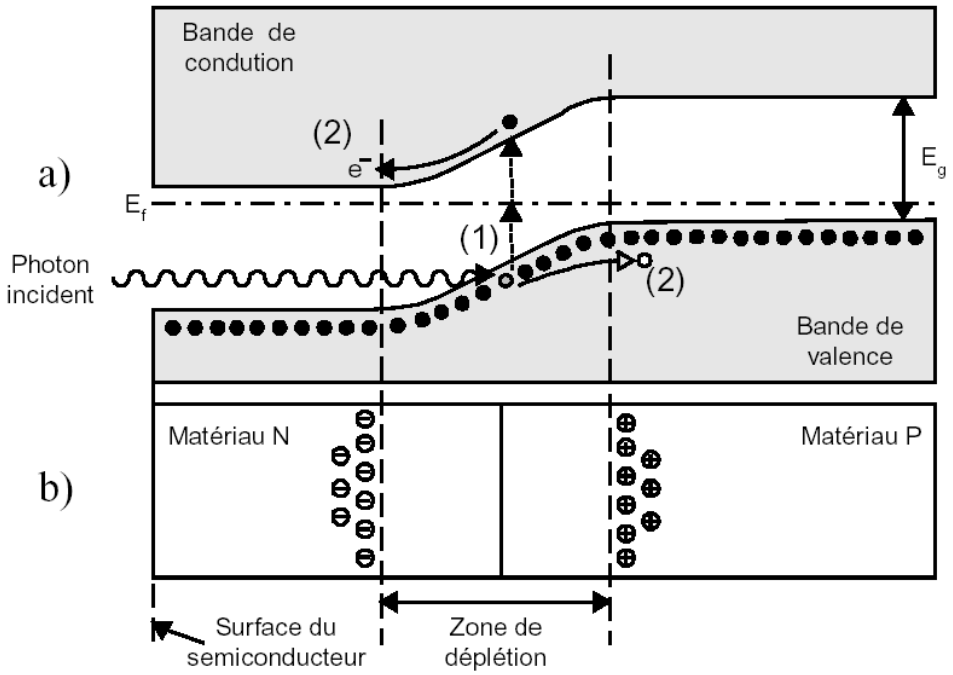


Figure 3-1 : Absorption d'un photon dans une jonction P-N.

Afin de collecter les électrons générés par l'effet photoélectrique ou de préserver leur distribution spatiale, un champ électrique doit être généré à l'intérieur du matériau. Il peut être induit dans une jonction P-N (photodiode) ou dans une jonction induite qui se forme sous la grille d'une structure MOS (photoMOS). Ces dispositifs permettent de tirer profit de ce phénomène de manière à mesurer une intensité lumineuse.

3.1.2 Eléments de base des capteurs optiques à semi-conducteur

Le principe de la **photodiode** consiste à permettre à des photons de pénétrer dans un semi-conducteur afin qu'ils génèrent des paires électrons-trous à l'intérieur de la zone de déplétion d'une jonction P-N polarisée en inverse. Le champ électrique présent dans la zone de déplétion sépare alors rapidement les deux porteurs en excès. Ceci crée alors un courant de dérive à travers la jonction P-N (Figure 3-2) qui est directement proportionnel au nombre de photons excitateurs pénétrant dans le semi-conducteur. Il est alors possible de déduire l'intensité du signal lumineux au moyen de senseurs électriques en mesurant l'une des deux variables suivantes : le courant de dérive généré par le signal lumineux (photocourant), ou la charge accumulée d'un côté de la jonction P-N au cours d'une période déterminée [Weckler 67]. Il est à noter que les charges accumulées sont de type opposé à celles forcées par la tension inverse aux bornes de la diode. L'accumulation de charges se traduit donc par une diminution de la tension aux bornes de l'élément photosensible.

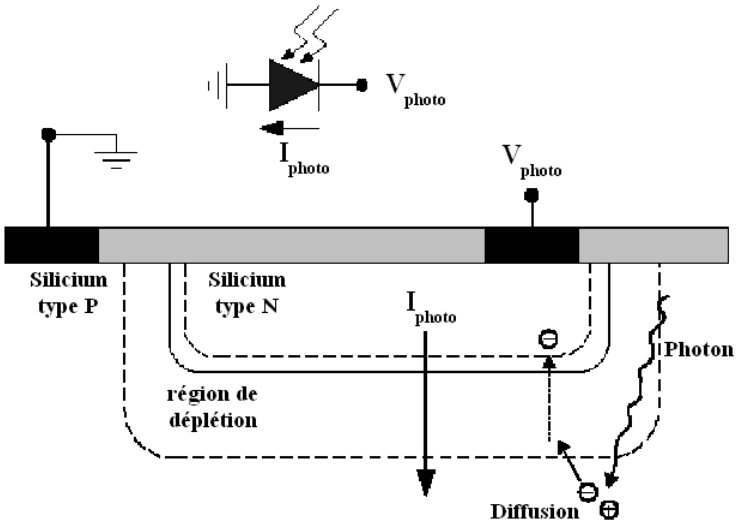


Figure 3-2 : Schéma photodiode P-N.

Le **photocondensateur** est l'autre élément de détection lumineuse fréquemment utilisé pour les capteurs d'images. Le condensateur en question est constitué d'une plaque de conducteur transparent séparée d'un substrat semi-conducteur par une mince couche d'oxyde isolant. Par analogie avec le transistor dit à *Metal Oxyde Semiconductor* (MOS) qui partage la même structure, on y fait régulièrement référence à l'aide des appellations photoMOS, phototransistor ou transistor à photogrille. En appliquant une tension à la grille (en anglais *gate*) du phototransistor (Figure 3-3), une zone de déplétion se forme dans le semi-conducteur, à l'image de ce qui se produit lors de la formation d'un canal sous la grille d'un transistor MOS. La grille conductrice et l'isolant étant transparent, les photons sont libres de pénétrer la surface du substrat semi-conducteur. Les charges libérées par les interactions entre photons et électrons dans le substrat peuvent alors être accumulées dans la zone de déplétion, située sous la grille, que l'on nomme puits de potentiel. Le principal avantage du photoMOS est que, grâce au fait que ses bornes sont électriquement isolées, aucun courant ne circule lorsque la tension est appliquée pour générer la zone de déplétion. Ceci lui procure donc une meilleure sensibilité à de très faibles niveaux d'éclairage, puisque le signal ne se confond avec aucun courant présent par défaut (nommé courant de noir).

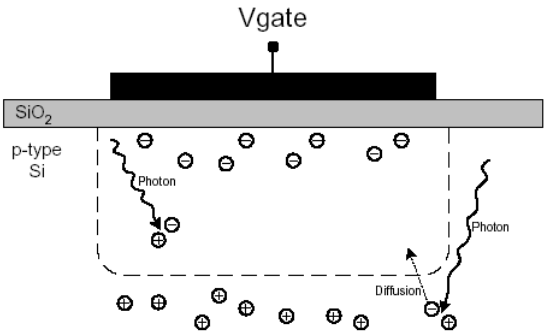


Figure 3-3 : Schéma d'une capacité MOS.

Dans les deux cas, photodiode et photocondensateur, les paramètres qui définissent les performances du dispositif (sensibilité, nombre de charges générées par photon incident, intensité du courant, réponse spectrale, courant de noir) dépendent principalement du substrat et du dopage du semi-conducteur. Le concepteur d'un capteur optique utilisant ces photo-senseurs dans une technologie donnée n'a donc que très peu de contrôle sur ces performances.

En réalité, la sensibilité d'un semi-conducteur à une longueur d'onde est maximale lorsque l'énergie photonique est légèrement supérieure à l'énergie qui sépare ses bandes énergétiques de valence et de conduction. En effet, si l'énergie du photon est trop largement ou trop faiblement supérieure à celle qui sépare les bandes énergétiques du matériau, l'absorption du photon a de très fortes probabilités de se produire respectivement trop en surface ou trop en profondeur dans le semi-conducteur pour que les charges générées soient accumulées par l'élément photosensible [Blansky 00]. Le premier cas est d'autant plus vrai lorsqu'un photoMOS est utilisé, car le photon risque de se faire absorber directement par la photogrille.

Heureusement, le silicium présente de bonnes caractéristiques dans la plage des fréquences associées à la lumière visible. L'énergie qui sépare les bandes énergétiques du silicium, soit 1,11eV, permet de capter des photons dont la longueur d'onde est inférieure à environ 1,1µm. Ceci couvre généralement adéquatement tout le spectre de lumière visible, qui s'étend d'environ 390 à 770nm, du bleu au rouge, respectivement (Figure 3-4).

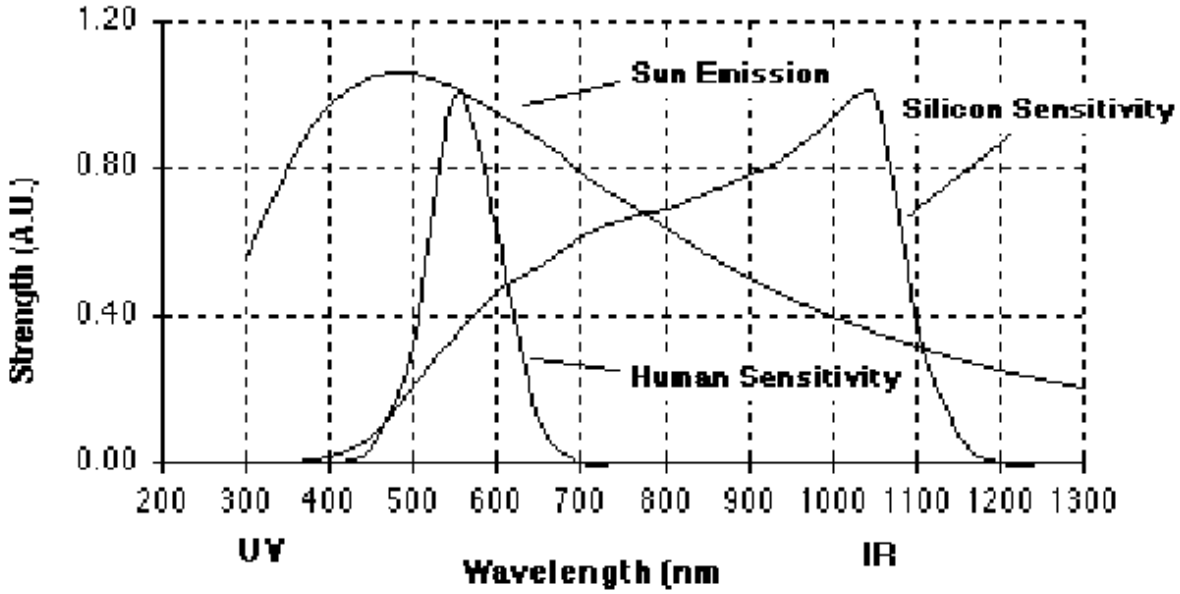


Figure 3-4 : Réponse spectrale pour le silicium, le soleil et l'œil humain (*spectral sensitivity*).

3.2 - Introduction aux capteurs d'images intégrés

3.2.1 Critères de performances

Avant de présenter les principales technologies utilisées pour la réalisation de capteurs d'images, il convient de définir les principaux critères d'évaluation sur lesquels nous avons une influence. Les performances d'un capteur d'images se regroupent en différentes catégories [Goy 02]. Ces catégories sont les paramètres géométriques (résolution, taille des pixels, taux de remplissage) [Chen 00], électriques (dynamique, bruit, vitesse de lecture) [Meynants 01] et physiques (sensibilité, rendement quantique, réponse spectrale) [Catrysse 00, Catrysse 02]. Dans le cadre de ce travail de thèse, nous ne parlerons que des deux premières catégories de performances. La dernière étant liée à la structure silicium du pixel, elle n'est paramétrable que par les concepteurs de capteurs silicium.

1 - Paramètres géométriques

Résolution Tout système d'imagerie se caractérise principalement par la résolution de l'image obtenue. La résolution détermine le nombre de points par unité de surface, exprimé en points par pouce (PPP, en anglais DPI pour *Dots Per Inch*). La résolution permet ainsi d'établir le rapport entre le nombre de pixels d'une image et la taille réelle de sa représentation sur un support physique. Par exemple, une résolution de 300 dpi signifie donc 300 colonnes et 300 rangées de pixels sur un pouce carré ce qui donne donc 90000 pixels sur un pouce carré. Mais pour un capteur d'image, cette résolution est dépendante du système optique associé. C'est pour cela que l'on parle plutôt de **définition** d'un capteur d'image.

La définition d'un capteur est le nombre de points (pixel) constituant l'image, c'est-à-dire sa dimension physique ou informatique (le nombre de colonnes de l'image que multiplie son nombre de lignes). Elle s'exprime en nombre de pixels. Plus la définition est élevée, plus le confort visuel est grand (les détails sont bien définis), mais plus le capteur est coûteux. La tendance aux capteurs à millions de pixels se porte principalement sur les capteurs CMOS puisque, contrairement aux CCD, ils ne sont limités que par les paramètres électriques qui s'améliorent avec les réductions de la finesse de gravure de la longueur du canal minimale des transistors (entre 0,35 et 0,18 µm). Bien que les appareils grand public offrent des définitions toujours plus élevées (jusqu'à 7 Mégapixels), il faut préciser qu'il existe bon nombre d'applications, par exemple des caméras de surveillance, ne nécessitant pas une très forte définition. De plus, une image grande définition entraîne une masse de données d'autant plus importante à traiter. Donc, le choix de la définition se fait sur la base d'un compromis entre la qualité de l'image et la quantité de données à traiter en sortie du capteur.

Taille des pixels La taille des pixels est un paramètre moins crucial que la résolution. Puisque la taille des circuits intégrés augmente, cela permet de réduire le coût du circuit, et donc de faciliter la fabrication et la diffusion en grande série. Une fois encore, les avancées de la densité d'intégration de l'électronique bénéficient aux capteurs CMOS. Il est à noter que l'amélioration de la finesse de gravure ne pénalise pas la fabrication des capteurs d'images CMOS. Au contraire, des dimensions plus fines permettent d'obtenir des capacités parasites réduites qui améliorent grandement les performances statiques (gain, bruit) et dynamiques (vitesse de lecture). De plus, la taille des pixels est souvent imposée par la quantité d'électronique intégrée au sein du pixel et par la taille de la zone photosensible.

Facteur de remplissage Le facteur ou taux de remplissage (en anglais *fill factor*) représente la configuration spatiale à l'intérieur du capteur, c'est-à-dire le rapport entre la surface totale du pixel (sa taille) et la surface photosensible (généralement la surface de la photodiode). La différence entre ces deux surfaces est celle occupé par le routage, ou par les parties électroniques servant au transport ou au traitement de l'information. Un taux de remplissage peu élevé peut caractériser une sensibilité faible (la surface photosensible est petite) ou un pixel de taille importante (beaucoup d'électronique intégrée au sein du pixel). Ce facteur est généralement compris entre 50 à 95% pour des capteurs d'images basiques. Mais, ce taux diminue fortement lorsqu'un traitement ou une fonction spécifique est associé à la photodiode au sein du pixel. Divers techniques existent pour améliorer ce facteur. La plus courante est l'usage de microlentilles [Nussbaum 97].

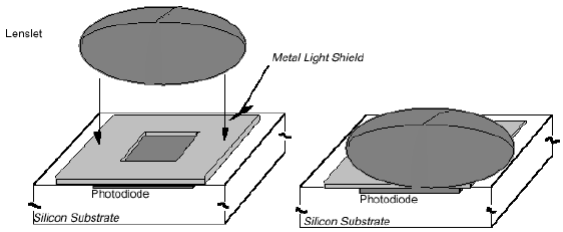


Figure 3-5 : Positionnement de microlentille sur une photodiode.

Il s'agit de microlentilles de taille comparable au pixel qui sont installées à quelques micromètres de la surface sensible du pixel (Figure 3-5). Le facteur de remplissage équivalent d'un tel système est proche de la perfection, c'est-à-dire 100%. Cependant, ce système a ses propres limites, notamment en ce qui concerne les pixels situés sur le bord de la matrice. Dans certains cas particuliers (capteurs d'images CMOS), on pourra considérer que les charges générées dans la zone active du pixel, c'est-à-dire entre les transistors, contribuent au courant photonique, soit parce qu'elles diffusent vers la photodiode, soit parce qu'elles modifient les caractéristiques électriques des transistors qui les entourent. Cette technique sera abordée dans la section 3.3 – Les capteurs d'images de type CMOS.

2 - Paramètres électriques

Dynamique La dynamique est la plage de luminosité dans laquelle le capteur est capable de mesurer correctement l'information. Elle est en fait comprise entre les deux niveaux extrêmes (du signal) que peut traiter le capteur. Le premier est le niveau de saturation du capteur, le second est le niveau de bruit. Les capteurs d'images ont donc des problèmes de sensibilité en lumière faible, et des problèmes de saturation en lumière forte. Pour augmenter la dynamique (évitant ainsi la saturation), des capteurs d'images CMOS à conversion logarithmique ont été développés [Kavadias 00]. Mais ce mode logarithmique présente des inconvénients qui seront exposés dans la section 3.3 - Les capteurs d'images de types CMOS.

Temps de lecture Le temps de lecture peut être important selon l'application envisagée. Il existe des applications dans lesquelles la rapidité est primordiale, on parle ainsi de vidéo rapide [Bouffaut 96]. On peut distinguer la vitesse de lecture des pixels, d'une ligne, et d'une trame. Bien que la vitesse de lecture des pixels soit sensiblement constante pour un circuit (de l'ordre de quelques Méga pixels par seconde à plusieurs dizaines), la vitesse de trame (ou taux de rafraîchissement de l'image) dépend de la définition de la trame lue. Dans le cas des CCD, la question n'est pas fondamentale, puisque cette définition est fixe. Il peut toutefois y avoir une ambiguïté en ce qui concerne les capteurs CMOS à pixels actifs, puisque l'utilisation du fenêtrage ou du sous échantillonnage augmentera considérablement la vitesse de trame. La définition et la vitesse de lecture sont donc deux paramètres inversement proportionnels. Le temps de lecture sera considéré comme le temps mis entre l'instant de déclenchement de la capture de l'image et l'instant où l'image est complètement sortie du capteur.
Généralement, la vitesse du circuit sera limitée en sortie par le multiplexeur et/ou les convertisseurs.

Bruit fixe Le bruit fixe est le niveau de bruit invariant dans le temps associé aux disparités entre les réponses de différents pixels. Ce type de bruit, particulier aux capteurs d'images électroniques, mérite une attention particulière. Les différences entre les caractéristiques des différents éléments au travers de la matrice (non uniformité dans la géométrie, l'épaisseur de couche, le niveau ou le profil du dopage, sur la définition de la grille), qui apparaissent au moment de la fabrication, procurent des réponses différentes d'un pixel à l'autre (différence dans la sensibilité, le niveau de saturation, et la génération du courant de noir). En conséquence, la variation de la réponse des cellules photosensibles se traduit par une image irrégulière en sortie, même sous une illumination homogène. Ces irrégularités sont fonction de l'espace, et non du temps. Elles se répètent donc d'une image à l'autre. Pour cette raison, ce bruit est désigné sous l'appellation de bruit à motif fixe, traduction de l'anglais *Fixed Pattern Noise* ou FPN. Comme elles ne sont pas prévisibles avant la fabrication du circuit, il faut donc recourir à des méthodes de compensation pour pouvoir les atténuer a posteriori.

Pour cela, des techniques de calibration permettent de compenser ces défauts comme, par exemple, les techniques de double échantillonnage. Ces techniques seront détaillées dans le paragraphe 3.3.3 - Techniques de double échantillonnage.

3.2.2 Technologie Charge Coupled Device

Le concept *Charge Coupled Device* (CCD) est né en 1970 dans les laboratoires **Bell Lab.** avec les travaux des ingénieurs W.S. Boyle et G. Smith [Boyle 70]. Initialement utilisé pour réaliser des lignes à retard, des filtres analogiques, etc, le concept CCD a rapidement évolué vers une utilisation pour des applications de capteurs d'images. Ce détecteur a atteint une grande maturité alliant un très haut niveau de performances et une importante utilisation dans toutes les applications d'imagerie aussi bien dans le domaine scientifique que grand public. Cette section présente le fonctionnement et les caractéristiques d'un tel capteur. Bien qu'il ne soit pas le sujet de ce travail de thèse, nous ne pouvons passer sous silence le capteur CCD qui demeure la référence dans le monde de l'acquisition d'images numériques. Pour une étude plus approfondie, nous vous conseillons l'ouvrage suivant [Theuwissen 96].

1 - Principe

Le pixel CCD est basé sur une capacité MOS (Métal Oxyde Semi-conducteur) dont la grille (*gate*) est polarisée. Cette tension de polarisation provoque une zone de déplétion appelée "puit" sous la grille. L'arrivée de photons sur le silicium va créer des paires électron-trou, paires qui vont s'accumuler durant un certain temps, appelé temps d'intégration, dans ce "puit".

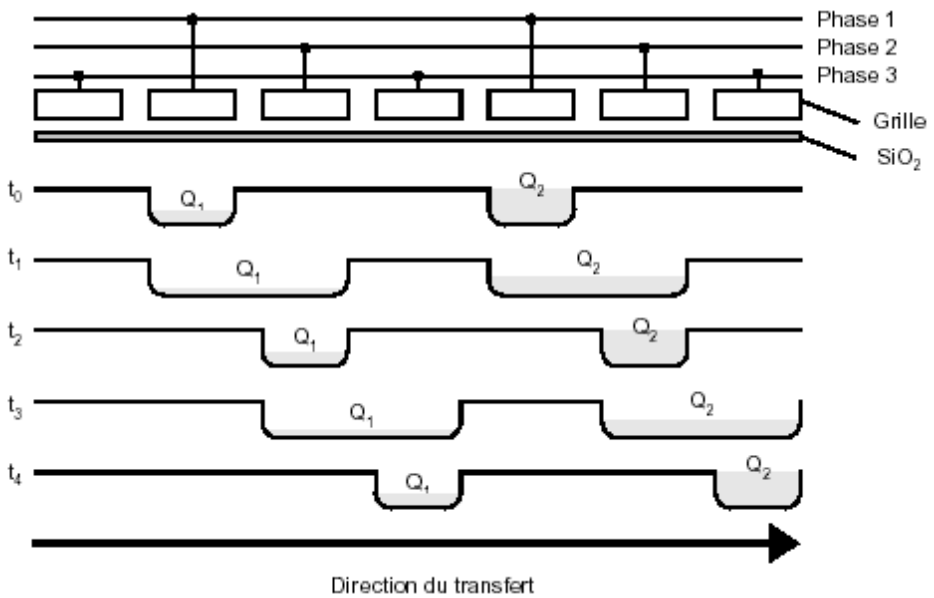


Figure 3-6 : Mécanisme de transfert de charge.

Les charges sont transportées par un dispositif à couplage de charge sous l'influence des tensions appliquées aux grilles des capacités MOS. Ces grilles sont regroupées en phases pour réduire

au minimum le nombre de tensions nécessaires. Selon la configuration, les capteurs peuvent être pilotés par quatre, trois, deux phases ou par un système d'horloges variables. Chacun de ces systèmes a ses avantages et ses inconvénients.

En jouant sur ces tensions, nous allons modifier la position des puits, et donc forcer les charges à se déplacer. La Figure 3-6 présente le mécanisme permettant d'arriver à cette fin. Sur ce schéma, les charges sont initialement intégrées sous les grilles formées par le signal **Phase 1**. Par la suite, l'activation appropriée des différents signaux **Phase** de 1 à 3 permet, en quelques étapes ($\mathbf{t_0}$ à $\mathbf{t_4}$ sur la figure), de former et de détruire des puits de potentiel afin de contraindre le déplacement des charges ($\mathbf{Q_x}$) dans la direction de transfert voulue. Ces charges arrivent séquentiellement (pixel par pixel) sur un amplificateur, puis sortent du circuit. Reste alors à mettre un circuit, pour la conversion analogique numérique dans le cas d'une sortie numérique, ou pour la mise en forme des signaux (synchronisation, voltage) dans le cas d'une sortie analogique. En décalant les charges lentement à la verticale et rapidement à l'horizontale, l'image est présentée à l'extérieur du circuit intégré sous forme d'un signal vidéo analogique.

2 – Configurations d'un capteur CCD

L'organisation la plus simple des pixels CCD dans un imageur à semi-conducteurs est une ligne d'éléments photosensibles (Figure 3-7a). La lecture des charges se fait grâce à un registre à décalage juste à côté des pixels. Après l'intégration des charges, les pixels sont vidés en même temps de façon parallèle vers le registre à décalage comme indiqué par les flèches. Une nouvelle période d'intégration peut alors commencer et dans le même temps, les charges sont transférées par le registre à décalage vers la sortie en série. Le registre à décalage est protégé de la lumière incidente pour éviter la perturbation des charges.

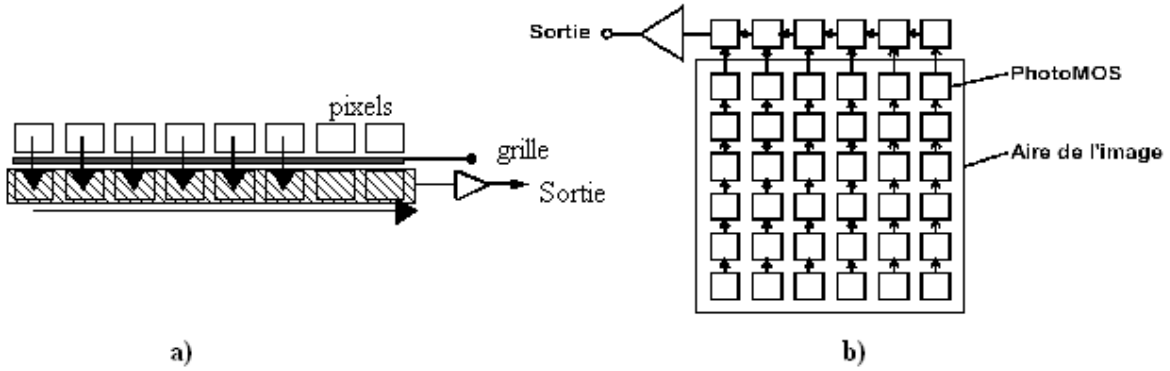


Figure 3-7 : Architecture d'un capteur CCD linéaire a) ou matriciel b).

L'autre organisation des pixels est la matrice d'éléments photosensibles (Figure 3-7b). Ce type de capteur possède le gros avantage d'utiliser toute la surface du pixel pour l'intégration de charges, ce qui lui donne une grande sensibilité. Néanmoins, il va falloir déplacer ces charges accumulées dans les pixels avant de les convertir en tension électrique. Ce déplacement peut atteindre une quinzaine de

millimètres, ce qui n'est pas instantané. Mais, sans l'utilisation d'un obturateur, cette technique possède l'inconvénient de polluer l'image par des photons parasites pendant le transfert de charge (ou lecture). Cet effet est appelé le *smearing* en français traînée ou traînage. Pour éviter cet effet indésirable, l'image (les charges) peut être envoyée dans une zone mémoire proche. Cette zone de stockage intermédiaire permet, aussi, de découpler le temps d'intégration des charges du temps de transfert des données. C'est sur la définition de cette zone de stockage que nous pouvons classer les différents types de capteurs CCD : les imageurs à transfert de trame, à transfert interligne ou les imageurs *full-frame* (sans zone de stockage).

Le **transfert de trame** consiste à définir une zone de stockage de taille similaire à la trame du capteur. A la fin du temps d'intégration, les charges sont transférées ligne par ligne dans une zone équivalente mais protégée de la lumière par un film opaque. Les charges sont ensuite converties pixel par pixel pour donner le signal vidéo. Cette technique présente l'avantage de séparer la zone photosensible et la zone de stockage (Figure 3-8), ce qui ne diminue pas la sensibilité, mais la surface totale du capteur CCD est alors multipliée par deux.

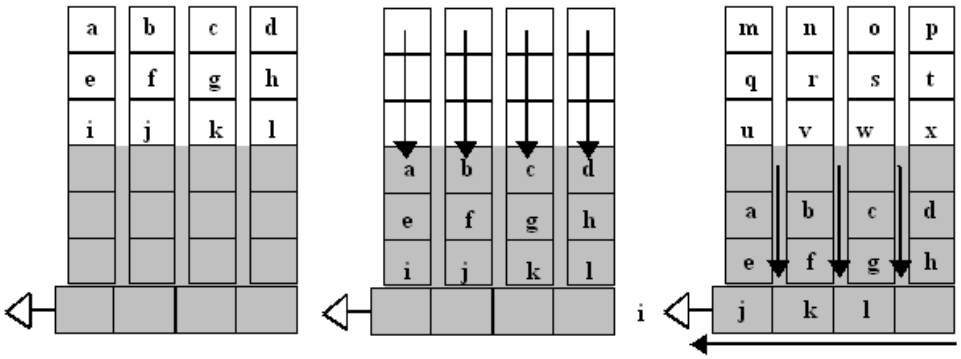


Figure 3-8 : Principe d'un capteur à transfert de trame.

Pour le **transfert interligne**, la zone de transfert se trouve entre les lignes (Figure 3-9), et les charges n'ont plus que quelques microns à parcourir pour être sauvegardés. Le phénomène de smearing est donc théoriquement supprimé. Par contre, le pourcentage de surface photosensible recule à environ 50%, ce qui réduit d'autant la sensibilité du capteur, ayant pour résultat un faible rapport d'ouverture.

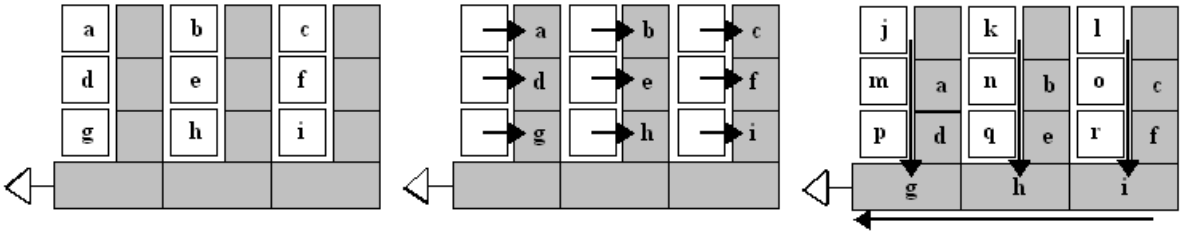


Figure 3-9 : Principe d'un capteur à transfert interligne.

Les capteurs CCD pleine trame ou *full-frame* se présentent comme une matrice de pixels CCD. Ces pixels servent à la fois pour l'intégration des photons, mais aussi pour le transport des charges accumulées vers la sortie de la matrice. Ce type de capteur possède une grande sensibilité et une excellente définition. Ces capteurs présentent deux inconvénients : le temps entre deux acquisitions est relativement long et le *smearing* pollue l'image acquise. L'utilisation d'un obturateur avec ces capteurs élimine cette pollution.

Chacune de ces technologies a son domaine d'application. Toutefois, les CCD *full-trame* ne sont plus conservées que pour les applications en très faible lumière (astronomie, biologie) où l'on peut accepter des temps de pose importants (de une à plusieurs secondes). Tandis que les CCD à transfert interligne autorisent par contre des temps d'intégration très courts (inférieurs à la milliseconde), ce qui permet de figer des scènes en mouvement.

Type de capteur	**Avantages**	**Désavantages**
CCD à transfert de trame	bonne résolution horizontale rapport d'ouverture élevé possibilité d'obturateur	capteur de grande taille niveau élevé de *smearing*
CCD à transfert interligne	capteur de petite taille niveau faible de *smearing*	faible rapport d'ouverture
CCD full-frame	Grande sensibilité Bonne résolution de l'image rapport d'ouverture élevé	Temps de lecture long niveau élevé de *smearing*

Tableau 3-1 : Vue d'ensemble des avantages et inconvénients des différents types de matrice CCD.

3 - Modes de balayage

Les premiers capteurs CCD ont été conçus pour délivrer des images entrelacées (*interlaced scan*). D'une part, cela correspondait à la norme du signal vidéo format télévision (standard CCIR). D'autre part, cela permettait de garder une surface photosensible suffisante sur le circuit. Ce mode de balayage permet d'augmenter la résolution verticale de l'image sans augmenter la largeur de la bande passante du système de transmission. Par définition, une trame (ou image) est faite de deux demi-trames entrelacées (lignes paires et impaires). Les demi-trames sont lues l'une après l'autre durant une période de lecture de trame. Cela permet d'éviter l'effet de scintillement des images vidéo.

A l'opposé de l'entrelacement, il y a le balayage progressif (*progressive scan*). Toutes les lignes sont parcourues à chaque passage, et chaque trame est exactement identique. Le résultat direct est qu'il est possible d'exposer simultanément tous les pixels du capteur. Le balayage progressif est une technique utilisée dans les applications où une résolution verticale plus élevée est d'importance primordiale et où, par exemple, la largeur de bande moyenne de transmission n'impose aucune limite à

la largeur de bande pour l'information visuelle. Les nouvelles générations de capteurs CCD sont systématiquement déclinées en une version *progressive scan*. Les demandes sont en effet fortes de la part du monde industriel et scientifique, pour acquérir des images avec une résolution maximale. Une autre raison est que les transferts d'images numériques apportent une alternative au transfert analogique standard, qui ne supporte pas les images non entrelacées.

4 - Contrôle du temps d'intégration

L'entrelacement des images pose un problème d'images floues dans le cas de mouvements. En effet une partie des lignes est exposées au temps T, les autres au temps T+1, l'image obtenue est par conséquent la somme de deux demi images acquises à deux instants différents.

Un problème similaire peut survenir si tous les pixels du capteur ne sont pas exposés simultanément, dans ce cas l'image résultante est composée d'une somme de pixels n'ayant aucune cohérence temporelle.

Le temps d'intégration variable et l'exposition simultanée des pixels sont gérés sur les capteurs numériques par un signal *reset* (mise à zéro) commun à tous les pixels, et par une fonction de *shutter* électronique. La fonction *reset* réinitialise tous les pixels dans un état sans charge, et permet l'accumulation des charges à un même instant. La fonction *electronic shutter* bloque à un instant donné l'accumulation des charges, en détournant les nouvelles charges créées par les photons afin que celles-ci ne rejoignent par les puits d'accumulation. Il permet de vider complètement les pixels sans détruire les charges de la zone de stockage. Ces deux fonctions permettent la prise de images avec un temps d'intégration variable, au même sens que dans la photographie argentique. Cette technique joue le rôle d'un obturateur électronique au sein même du capteur. Le *shutter* électronique permet d'éliminer les défauts inhérents aux CCD comme le *blooming*.

La conjonction des fonctions *progressive scan* et *shutter* électronique, permet de garantir une cohérence temporelle de tous les pixels d'un capteur CCD.

5 - Bilan

La technologie CCD offre des performances très intéressantes en détection et transfert de signaux avec un niveau de bruit relativement faible et uniforme. Il s'agit aussi d'une technologie mature en termes de rendement et de performances. Ces performances ont en général atteint des niveaux proche de leurs limites théoriques et n'ont subi aucune amélioration significative depuis des années [Kozlowski 99]. Cependant, quelques points rendent cette technologie non désirable dans certaines applications. Premièrement, un sous échantillonnage de la matrice est impossible. En effet, on ne peut avoir accès à la valeur de chaque pixel qu'en accédant à tous les pixels des lignes et colonnes qui le précédent dans la matrice. De plus, les dispositifs CCD possèdent une consommation plutôt grande. Cela est due à deux facteurs : des charges capacitives très importantes des lignes de commandes et plusieurs niveaux de tensions. Ces différents niveaux de tensions sont nécessaires pour une bonne efficacité du transfert des charges de chaque site vers la sortie.

Finalement, les circuits de ce type ont besoin d'un procédé de fabrication spécial, ce qui rend leur intégration à des systèmes plus complexes sur une seule puce (*System on-Chip*) impossible, ou tout au moins inefficace [Fossum 97]. Mais, aujourd'hui, avec les technologies CMOS, nous disposons de capteurs qui égalent les performances des capteurs CCD [Meynants 98a] tout en proposant de nouvelles fonctionnalités comme, par exemple, l'accès aléatoire aux pixels.

3.2.3 Technologie Complementary Metal Oxyde Semiconductor

Bien que cette technologie date des années 60 (la longueur minimal du canal était alors de 10µm) [Wanlass 67] et que l'effet photoélectrique des circuits de type *Complementary Metal Oxyde Semiconductor* ou CMOS soit connu et étudié depuis longtemps, elle n'a été exploité industriellement que récemment. Ceci est dû principalement à la "légendaire" grande taille des pixels CMOS (ne permettant, compte tenu du coût, que des matrices de faible résolution), et à leur faible rapport signal/bruit.

La forte densité d'intégration, liée à l'arrivée des technologies sub-microniques, a permis de mieux maîtriser les technologies et d'atteindre des pixels de quelques micromètres de côté. De taille équivalente aux pixels des capteurs CCD, les capteurs CMOS atteignent des résolutions intéressantes de plusieurs millions de pixels. Ces capteurs d'images fabriqués en technologie CMOS ouvrent des horizons et des perspectives de conception nouvelles [Fossum 97].

1 - Principe

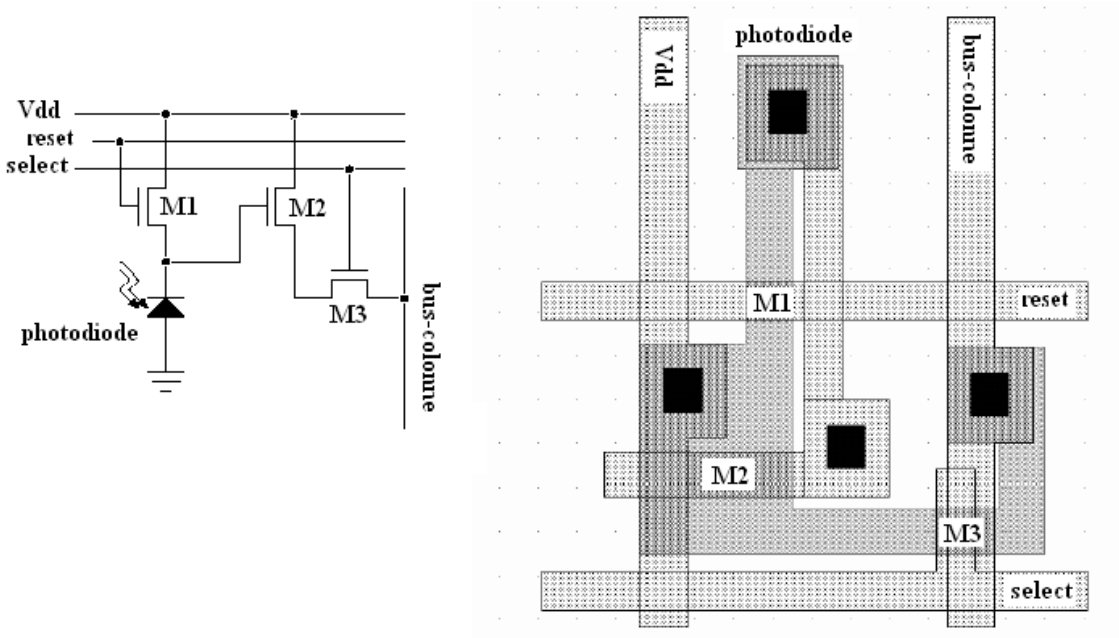


Figure 3-10 : Schéma électrique et *layout* d'un pixels CMOS [Meynants 98a].

Le capteur d'image CMOS ou CIS (pour *CMOS Image Sensor*) est un composant basé sur la technologie des semi-conducteurs. L'arrivée de photons sur le silicium va créer des charges électriques, qui vont s'accumuler dans le silicium durant un certain temps, appelé temps d'intégration.

Un capteur d'image CMOS est formée de pixels mesurant quelques microns, disposés en une matrice de Y lignes et de X colonnes. Chaque pixel est composé d'une partie photoréceptrice (généralement une photodiode) et de transistors jouant le rôle d'interrupteurs (Figure 3-10).

Un maillage en X et Y permet en fermant les transistors voulus de lire la tension du pixel. Cette tension est proportionnelle à l'intensité lumineuse reçue. Les capteurs d'image CMOS sont organisés comme des mémoires. La matrice de pixels est associée à un circuit de lecture. La Figure 3-11 présente une structure générale de ce circuit de lecture. Chaque pixel est adressé par une ligne et une colonne commandées par des séquenceurs, via des décodeurs d'adresses. Les colonnes sont le plus souvent appelées "bus-colonne" car elles véhiculent les données des pixels jusqu'au circuit de sortie. Les données de la matrice sont donc accessibles aléatoirement, par adressage [Meynants 98b]. Des amplificateurs de colonne et un système de multiplexage améliorent la dynamique et aiguillent les données vers le ou les convertisseurs analogiques numériques (ADC). Avant la conversion, le signal analogie peut être amplifié, voire quelquefois prétraité (gain et *offset* réglables, traitement anti-FPN, etc.).

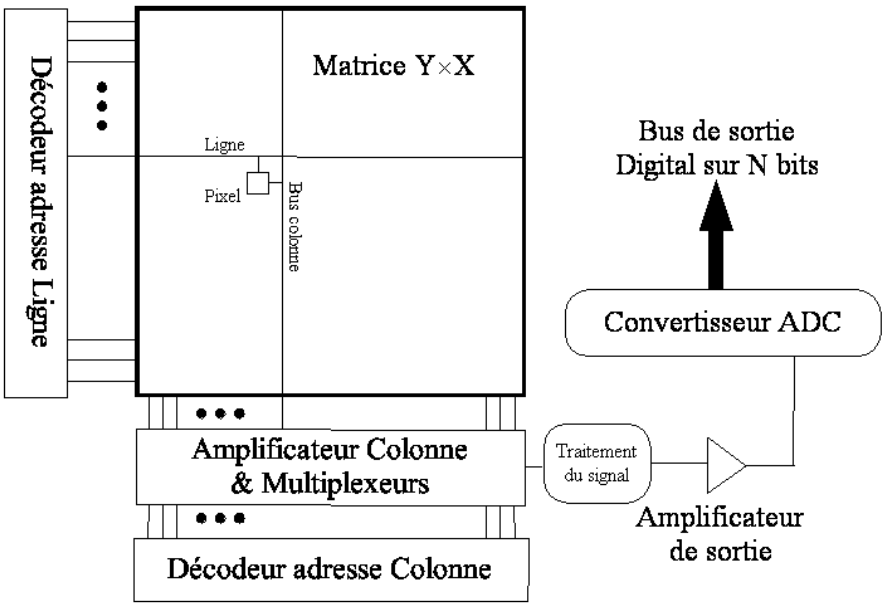


Figure 3-11 : Schéma blocs d'un imageur CMOS.

Les principales innovations des capteurs d'images CMOS concernent le nombre de transistors affectés à chaque pixel, et surtout leurs rôles. Le problème de base à résoudre est le caractère bruité du signal lu au niveau du pixel. On distingue donc trois familles de capteurs CMOS en fonction de leur type de pixels : passifs, actifs et digitaux.

L'intérêt des pixels passifs est leur forts taux de couverture de la partie photosensible du capteur (facteur de remplissage proche de 100%), mais le rapport signal/bruit est mauvais. Les pixels actifs sont les plus répandus. Ils sont caractérisés par la présence au sein du pixel d'un amplificateur suiveur qui amplifie localement le signal, mais aussi améliore le signal en jouant sur des *offsets* de tensions. Bien sûr, tout cela prend de la place, et tend à réduire la surface photosensible. Pour éviter cet "effet de bord", il existe plusieurs solutions technologiques qui seront détaillées dans la section suivant. Les pixels digitaux sont des pixels actifs numérisant et mémorisant les données directement au sein même du pixel. Ils ont été développés pour augmenter de façon considérable la fréquence d'acquisition des images. Ce type de pixel présente l'avantage de fortement réduire le bruit spatial fixe. Mais, sa taille reste un facteur limitant à une utilisation pour un grand nombre d'applications.

2 - Apport des capteurs d'images de types CMOS

Indépendamment de l'architecture, les capteurs CMOS ont typiquement une qualité d'image inférieure (sensibilité plus faible à la lumière) et un niveau plus élevé de bruit (un grand courant de noir) que les capteurs CCD. Les capteurs CMOS ont cependant d'autres avantages par rapport aux capteurs CCD. La prise d'image par une puce alliant la conversion analogique numérique, de la mémoire et un traitement embarqué n'est possible qu'avec les capteurs de type CMOS.

L'attrait des capteurs CMOS actifs augmente avec le nombre de pixels. Cette technologie est devenue la référence pour les capteurs de grande taille. Ceci est principalement dû à leur plus faible bruit de lecture à des fréquences au dessus de 10MHz, grâce à l'amplification possible à l'intérieur même du pixel. De plus, le fait d'effectuer la conversion directement sur la puce améliore la qualité du signal, en éliminant le bruit qui s'introduit dans les interconnexions entre composants. Le transfert de l'information à l'extérieur de la puce de façon numérique est bien sûr mieux immunisé au bruit que le transfert analogique. La vitesse de lecture est aussi améliorée, puisque la distance qui sépare les photo-détecteurs et les circuits de lecture est réduite, et que la charge capacitive des plots est éliminée.

Toutefois, les dispositifs CMOS étant particulièrement sujet à de fortes disparités de leurs caractéristiques technologiques. Le bruit spatial fixe constitue une sévère limite aux performances d'un capteur. Dans la plupart des cas, des traitements, réduisant ce type de bruit, sont utilisés pour procurer une image de qualité acceptable. Heureusement, le bruit se répétant d'une image à l'autre, des moyens efficaces d'amélioration d'image existent.

3.2.4 Choix technologique

Nous avons fait le choix de développer notre propre système d'acquisition répondant aux contraintes et caractéristiques qu'exige l'application visée. Pour cela, nous devons d'abord choisir la technologie du capteur d'image. Le Tableau 3-2 présente une liste non exhaustive des principales caractéristiques des capteurs d'image à semi-conducteur et leurs équivalents pour l'œil humain. Pour les capteurs d'images, les paramètres essentiels sont la sensibilité, le rapport signal sur bruit ou SNR (*Signal to*

Noise Ratio), le bruit spatial fixe ou FPN (*Fixed Pattern Noise*), le courant d'obscurité, la non-linéarité, le facteur de remplissage (*Fill Factor*) et l'efficacité quantique (*Quantum Efficiency*).

Contrairement aux capteurs CCD, les procédés de fabrication CMOS standard n'ont pas été optimisés pour les caractéristiques optiques de leurs éléments, mais plutôt pour la réalisation des circuits numériques ou mixtes. De ce fait, les capteurs CMOS ont une qualité d'image inférieure et un niveau plus élevé de bruit que les capteurs CCD. D'ailleurs, ils ont un bruit spatial fixe plus élevé puisque les données de l'image sont lues par différentes chaînes de buffers et d'amplificateurs. Mais, des circuits de réduction des bruits sont ajoutés sur la puce avant la sortie des données du capteur. Les capteurs CMOS ont cependant des avantages par rapport aux capteurs CCD. D'abord, un capteur CMOS ne requiert qu'une alimentation de 3,3 Volts ou 5 Volts, au lieu de 15 Volts typiquement pour le CCD ce qui réduit la consommation.

Critère	**Œil**	**CCD**	**CMOS APS**
Réponse spectrale	400-700 nm	400-1000 nm	400-1100 nm
Dynamique	Au total, de l'ordre de 120 dB En pratique : 60 dB	~ 70 dB	en moyenne entre 50 et 70 dB
Limite de noir	0,001 lux	Typ : 0,1 lux < 0,0001 possible	Typ : 0,1 lux 0,001 possible
Fréquence maximum de trame	15 Hz	10 kHz	10 MHz voire plus
Résolution	120 millions de cônes récepteurs	en moyenne 3 Méga pixels	en moyenne 3 ou 4 Méga pixels (max. 7,1)
Taille du pixel	2-3 µm	3-10 µm	3-10 µm
Taux de remplissage	-	90-100%	de 25 à 80%
Bruit de lecture et FPN	-	20 électrons rms	250 électrons rms
Taille du plan focal	3 cm	de 1 mm à la taille du wafer	de 1 mm à la taille du wafer
Consommation en énergie électrique	-	typiquement 500 mW	typiquement 50 mW
Forme des données	influx nerveux	8-10 bits	8-14 bits
Temps de lecture	Instantané	Durée d'une trame	Accès aléatoire aux pixels
Accès à l'information	5 millions de nerfs	série	Série ou parallèle

Tableau 3-2 : Récapitulatif des principales caractéristiques de différents capteurs d'image.

De plus, on peut intégrer sur un même morceau de silicium la partie capture de l'information et la partie traitement de l'information. L'intégration à très haute densité de fonctions complexes peut rendre le système complètement autonome. Cette association est plus connue sous le nom de capteur intelligent (*smart sensor*). La simplicité de conditionnement du signal numérique est aussi un facteur déterminant car son intégration à l'intérieur d'un système est facilitée. C'est le cas du capteur CMOS à pixel actif (*Actif Pixel Sensor*) qui délivre typiquement un signal de sortie relativement élevé avec une impédance de sortie faible. Grâce à ce type d'architecture, les accès multiples à l'image sont possibles. On peut alors faire du fenêtrage simple et rapide, du sous échantillonnage ou de la lecture directe, multiple et aléatoire d'une scène. Comme chaque pixel est indépendant, il peut être commandé séparément et réaliser une intégration de charge différente par rapport au pixel voisin. Mais, la non uniformité de l'image (bruits) est accentuée par ces circuits intégrés dans le pixel. Ce capteur ne requiert pas de signaux de contrôles précis pour aiguiller les photocharges. La cadence d'acquisition n'est plus limitée par le transfert des charges mais elle est restreinte par les performances du capteur.

Notre choix s'est porté sur la technologie CMOS en raison de ses avantages et de ses potentialités. L'évolution de la microélectronique numérique contribue fortement à son développement. En effet, la forte demande d'intégration pour obtenir des mémoires de capacité supérieure et des circuits plus complexes fonctionnant à des fréquences de plus en plus élevées, entraîne une diminution de la lithographie silicium (finesse de gravure) depuis les années 70. Cette évolution suit la loi empirique de Moore qui veut que le nombre de transistors dans un circuit double tous les 18 mois.

3.3 - Les capteurs d'images de types CMOS

Les capteurs d'images CMOS peuvent être divisés en trois grandes catégories selon les éléments qui composent leurs cellules photosensibles. Nous faisons communément référence à ces capteurs au moyen d'acronyme : PPS pour *Passive Pixel Sensor*, APS pour *Active Pixel Sensor* et DPS pour *Digital Pixel Sensor*. Après leur description, nous présentons ensuite l'inconvénient majeur des capteurs de types CMOS et les techniques pour le réduire. Cependant, le lecteur doit noter que cette section n'est qu'un bref exposé sur les capteurs CMOS. Nous terminerons cette section par une présentation des deux types de capteurs d'images CMOS : les imageurs et les rétines artificielles.

3.3.1 Structure des Pixels

Cette section expose les trois différentes approches pour définir la structure d'un pixel CMOS. Nous décrirons, en plus, le cas particulier des pixels actifs à conversion logarithmique puisque ce type de pixel est actuellement assez répandu sur le marché.

1 - Approche pixel passif (*Passive Pixel Sensor*)

Le terme passif est utilisé pour des pixels sans amplification, ni *buffer*. Le concept de pixel passif est présenté dans la Figure 3-12. Il consiste en une photodiode et un transistor de sélection. Pour activer le pixel, le transistor de sélection est activé une première fois pour remettre la photodiode à la tension de reset. Puis, la photodiode collecte les charges produites par la lumière durant la période d'intégration. Pour lire le pixel, le transistor de sélection est de nouveau activé. La quantité de charge requise pour remettre la photodiode à la tension de reset est alors envoyée vers un amplificateur par le bus-colonne. Cette lecture détruit donc la valeur du pixel. La lecture d'une image complète se fait généralement rangée par rangée. L'amplificateur de chaque colonne convertit la charge reçue en une tension proportionnelle à cette charge.

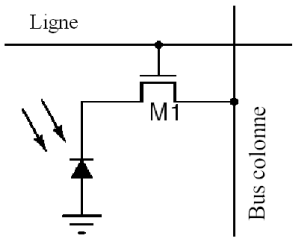


Figure 3-12 : Circuit d'un pixel passif.

Comme un seul et unique transistor est exigé, le facteur de remplissage est maximal, voire égal à celui de certains capteurs CCD. Parfois, un second transistor de sélection est ajouté pour permettre un vrai adressage en X et Y de la matrice. Les principaux problèmes du pixel passif sont ses niveaux de bruit de lecture importants. L'augmentation du bruit de lecture est due à la grande distance qui sépare les pixels de leur amplificateur. Le pixel passif ne fonctionne pas correctement avec de larges tailles de matrice et/ou des vitesses rapides de lecture pixel.

Les capteurs passifs sont très peu développés à cause de leurs niveaux élevés de bruits et de diaphonie entre les pixels (effets d'éblouissements) [El Gamal 98]. Des travaux [Fujimori 00] au MIT ont développé une technique de réduction des niveaux de bruit (surtout le FPN) des pixels passifs. Cette réduction permet d'obtenir un niveau de bruit comparable à ceux des pixels actifs annoncés dans la littérature. Grâce à un prototype, un imageur CMOS de 256×256 pixels passifs, ils ont démontré qu'une architecture de type *column-parallel differential* et un circuit de double échantillonnage corrélé permettent de retirer les effets de *smearing* du courant parasite qui est le défaut majeur des pixels passifs.

2 - Approche pixel actif (*Active Pixel Sensor*)

Contrairement au pixel passif, où le signal véhiculé est constitué de la charge accumulée à une borne de la diode, le pixel actif peut transmettre à sa sortie n'importe quel type de signal (tension, courant) selon le choix du concepteur. Ceci est du à un second transistor qui a pour fonction d'amplifier localement le signal de la photodiode. Ce second transistor joue le rôle d'amplificateur suiveur.

Le pixel actif standard comporte de 3 à 4 transistors (Figure 3-13) dont un pour l'amplification locale du signal. Le pixel est composé d'une photodiode pour détecter la lumière incidente, d'un transistor de reset (M1), d'un amplificateur suiveur (M2) et d'un switch de sortie pour la sélection de ligne (M3). Le transistor de sélection de ligne offre l'adressabilité, et ainsi le registre de lecture du bus-colonne sera disponible à d'autres lignes quand le transistor n'est pas ouvert. Afin de s'assurer que seulement un pixel est utilisé sur la ligne sélectionnée, les lignes sont séquentiellement envoyées vers la sortie au moyen du signal de sélection de ligne.

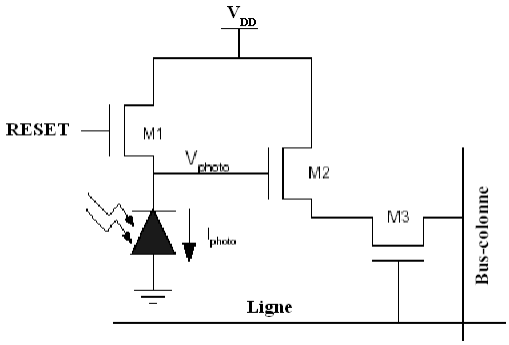


Figure 3-13 : Circuit d'un pixel actif à 3 transistors.

Le pixel APS est maintenant le pixel standard pour les capteurs d'images CMOS [Nixon 96]. Il fournit des performances raisonnables sans exiger un grand nombre de transistors pour sa mise en oeuvre. Cependant, le bruit spatial fixe reste un problème. Il provient principalement de la chute de tension de seuil présente dans la tension de sortie. La tension de seuil des transistors varie dans la puce, faisant apparaître le bruit spatial fixe dans l'image finale. Le bruit spatial fixe est décrit en détail dans le paragraphe 3.3.2 Bruits ou Non uniformité des images.

De plus, le fait de rajouter des transistors au sein du pixel tend à réduire significativement la surface photosensible. Cela diminue le facteur de remplissage du capteur. Mais des techniques existent pour pallier ce problème. La plus répandue est celle qui consiste à mettre des petites lentilles convergentes (*microlens*) sur les pixels. Une autre technique a été développée à l'IMEC [Imec-web], en Belgique, et commercialisée par la société FillFactory [fillfactory-web]. Elle constitue une approche originale du problème. Il s'agit de contrôler le dopage du substrat du capteur afin de rediriger les charges vers l'élément sensible. On peut ainsi considérer que la plus grande partie des paires électrons-trous créées par la lumière contribue au courant photonique. La différence de dopage entre les différentes couches crée un puit de potentiel sous l'électronique du pixel, qui sert à stocker les électrons. Ceux-ci se déplacent alors par diffusion dans toute la surface du circuit et sont captés par la diode la plus proche (diode à puits N [Dierickx 97]).

3 - Cas particulier : APS en mode Logarithmique

Les capteurs CMOS présentés jusqu'ici offrent une plage dynamique optique maximale d'environ 50 à 70dB. Ceci reste encore très loin de celle du système visuel humain, qui est de l'ordre d'environ

120dB. Le secret d'une telle performance provient principalement du fait que l'oeil possède une réponse logarithmique du signal de sortie par rapport à la lumière incidente. Dans les faits, la sensibilité de la rétine est très importante dans les cas de faible luminosité et diminue avec l'augmentation de l'intensité lumineuse incidente.

Plusieurs approches visant à imiter un tel comportement à l'aide d'un capteur à semi-conducteur ont été publiées. Le pixel de la Figure 3-14 possède une tension de sortie logarithmiquement dépendante du photocourant polarisant le transistor M1 en région de faible inversion [Ricquier 94, Scheffer 97]. Ce pixel est similaire au pixel APS standard, excepté qu'il ne possède pas de signal *reset*. Au niveau du pixel, le transistor de *reset* du pixel est connecté pour fonctionner comme une diode. Les deux autres transistors ont les mêmes rôles (un suiveur et un sélecteur).

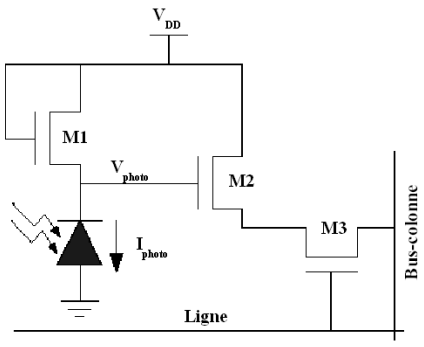


Figure 3-14 : Circuit d'un pixel actif en mode logarithmique.

Pour fonctionner comme une diode, la grille du transistor est reliée au drain. Cette diode équivalente est donc en série avec le générateur de courant constitué par la photodiode. En considérant V_{tr} la tension à ses bornes, le courant traversant ce transistor monté en diode est alors de la forme :

$$I = I_S \left(e^{\frac{q(V_{tr})}{nkT}} - 1 \right) \tag{3-3}$$

La tension aux bornes de ce transistor est donc proportionnelle de manière logarithmique avec le courant qui le traverse :

$$V_{tr} = \frac{nkT}{q} \ln\left(\frac{I}{I_S} \right) \tag{3-4}$$

Enfin, la tension de sortie aux bornes de la photodiode est :

$$V_{photo} = V_{DD} - \frac{nkT}{q} \ln\left(\frac{I}{I_S} \right) \tag{3-5}$$

L'avantage du pixel APS Logarithmique est qu'il fournit une dynamique plus importante que le pixel APS standard, à condition que la plage de luminosité soit limitée par la plage électrique des transistors, et non pas par la taille des puits de la photodiode.

Le mode logarithmique présente cependant des inconvénients. Du fait de son comportement logarithmique, les informations en sortie ont une dynamique très petite dans la plupart des scènes observées, ce qui cause des problèmes de quantification et de rapport signal/bruit. En effet, ce comportement est utile dans les scènes très fortement contrastées, mais le signal a une dynamique faible dans les scènes à contraste moyen. C'est pour cela que ce mode logarithmique est souvent un mode optionnel disponible sur les capteurs CMOS standard (mode linéaire). Le principal défaut du mode logarithmique provient du transistor fonctionnant en diode. Comme le transistor est en régime d'inversion faible, ses variations de la tension de seuil causent un grand bruit spatial, qu'il convient de corriger. Généralement, ces compensations se font a posteriori, en dehors du capteur. Cependant, quelques chercheurs [Ni 01] propose une structure analogique de compensation du bruit spatial dans la puce sur chaque bus-colonne.

4 - Approche pixel digital (*Digital Pixel Sensor*)

De récents travaux [El Gamal 99, Yang 99] à l'Université de Stanford utilisent une autre approche de design de pixel. Le pixel digital est un pixel actif possédant un convertisseur analogique numérique (ADC) directement au coeur même du pixel. La dynamique de conversion des prototypes est à l'heure actuelle de 8 bits. Tous les convertisseurs opèrent en parallèle. Les données numériques sont stockées dans des petites mémoires dynamiques (une par pixel). Ce qui revient à lire la matrice de l'imageur comme une mémoire conventionnelle (Figure 3-15).

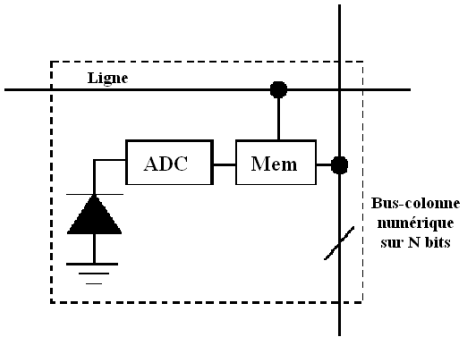


Figure 3-15 : Schéma de principe d'un pixel digital.

Le pixel digital présente de sérieux avantages par rapport aux pixels analogiques comme l'APS. Cela permet de réduire l'utilisation de circuits analogiques connexes à la matrice et d'éliminer les bruits de lecture des bus-colonnes (FPN). Avec un ADC et une mémoire par pixel, l'imagerie instantanée (*snap-shot*) massivement parallèle devient accessible, éliminant ainsi les goulots d'étranglement analogiques des imageurs standard [Lim 03]. Cela profite aux applications d'imagerie grande vitesse et permet de réaliser des implémentations performantes pour des applications vidéo standard telles que l'estimation de mouvement [Lim 01, Liu 03]. L'inconvénient principal du pixel digital est sa grande taille due au plus grand nombre de transistors par pixel. Le taux de remplissage est de l'ordre de 30% voire inférieur. Avec leur prototype en technologie CMOS 0,18µm, les auteurs

[Kleinfelder 01] obtiennent un taux de 15% avec des pixels digitaux de taille 9,4µm. Ces pixels sont constitués de 37 transistors, dont 3 forme une DRAM de 8 bits.

3.3.2 Bruits ou Non uniformité des images

Une limitation significative des performances d'un capteur CMOS est la non uniformité entre les pixels. Celle-ci est présente dans les images noires, mais aussi dans les scènes sous illumination normale. Cette non uniformité ou irrégularité des images d'un capteur CMOS est issue de problèmes technologiques rencontrés lors la fabrication du circuit. Elle se caractérise par une dispersion du comportement des pixels. Ces défauts peuvent être regroupés en deux catégories : le courant de noir (conséquence d'un défaut de la photodiode) et le bruit. La thèse de H. Tian [Tian 00] présente une analyse du bruit dans les imageurs CMOS. Ce bruit est constitué d'une composante spatiale (dominante) et d'une composante temporelle (plus ou moins importante d'une image à une autre).

Le courant de noir est dû aux fuites de charges dans la photodiode, qui se décharge même dans l'obscurité. Cette décharge se fait par le substrat à travers la diode. Ce courant peut être modélisé par un générateur de courant, en plus de celui de la diode, qui débite un courant constant. Son effet est de limiter la sensibilité de la photodiode. Rapporter au courant de la photodiode et au pas de quantification du signal (256 niveaux de gris), ce courant de noir reste relativement faible. Il ne pause problème que dans les applications en faible lumière (par exemple en astronomie).

Le bruit spatial fixe (BSF ou FPN pour *Fixe Pattern Noise*) provient des dispersions du process sub-micronique rencontrées lors de la fabrication des composants. Des éléments identiques, comme les pixels d'un capteur, ont finalement des caractéristiques électriques sensiblement différentes. Elles sont à l'origine du bruit spatial. Ce bruit est surtout présent dans les transistors amplificateurs suiveurs des pixels et particulièrement dans les amplificateurs des bus-colonnes. La valeur du bruit n'est pas prévisible. Mais comme l'indique son nom, il est fixe par rapport au temps. Il est donc facile à réduire voire à éliminer grâce à des structures électroniques appropriées. Elles ont pour principe d'effectuer une lecture différentielle pour s'affranchir de l'offset en cause de l'erreur, et ainsi de limiter son influence sur la lecture du signal. Les techniques de réduction du FPN sont exposées dans le paragraphe suivant.

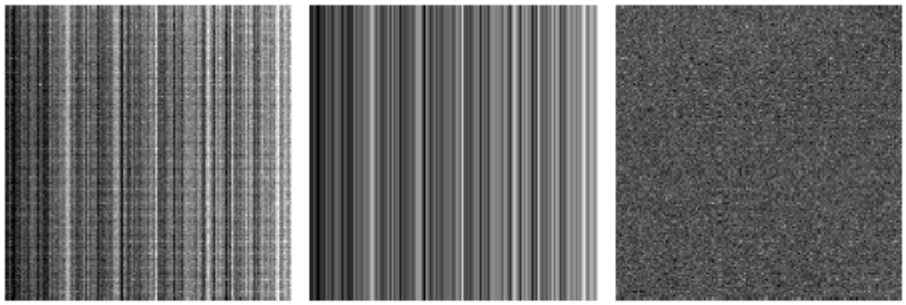
Le bruit temporel est dû aux densités spectrales de bruit des différents éléments du pixel. Ces densités spectrales s'ajoutent aux variations d'offsets lors de la lecture. On distingue différents types de bruit temporel :

- le bruit de grenaille (*shot noise*) créé par les courants présents dans la photodiode,
- le bruit en 1/f produit par la photodiode et l'électronique du pixel,
- et le bruit thermique venant de l'agitation thermique des électrons dans le matériau.

Les bruits dominants sont les bruits de grenaille et thermique.

On note cependant que les bruits temporels sont négligeables par rapport au bruit spatial fixe. Le but de ce travail de thèse n'étant pas l'étude des bruits dans les capteurs CMOS, nous ne développerons pas plus ce sujet. De nombreuses études ont été et sont menées dans ce domaine.

A cause des bruits spatiaux fixes, des différences de réponse existent entre les différents pixels, spatialement. Ce bruit se caractérise par des imperfections sur l'image (Figure 3-16). Nous pouvons classer ces imperfections en deux types facilement identifiables sur une image : le bruit pixel et le bruit colonne. Le bruit de type pixel provient des dispersions inter pixels (photodiode, transistors de *reset* et suiveur). Le bruit de type colonne provient des dispersions des amplificateurs des bus-colonnes, qui ont des gains différents. Le bruit de type colonne est (légèrement) plus important que le bruit de type pixel dans les capteurs de type APS à 3 transistors [Goy 02]. Notons aussi que le bruit colonne, à cause de sa structure, est très observable, alors que le bruit de type pixel est peu détectable par l'œil.



a) b) c)

Figure 3-16 : Somme de 50 images obtenues dans l'obscurité avec un APS 256×256 pixels : a) bruit fixe total, b) bruit fixe de colonne, et c) bruit fixe de pixel.

Pour atténuer ces bruits, il existe des structures électroniques de réduction du bruit. Elles consistent en un procédé de double mesure (double échantillonnage). Ce qui permet de ne s'intéresser qu'à la décharge effective de la photodiode. Les conditions initiales n'influent donc pas sur le résultat. La réduction du bruit peut être effectuée de deux manières : soit par Double Echantillonnage Corrélé ou soit par Double Echantillonnage Non Corrélé.

3.3.3 Techniques de double échantillonnage

Plusieurs techniques existent pour compenser le bruit spatial fixe. Bon nombre d'entre elles impliquent le concept d'une prise d'image de référence de noir [Moini 99], et puis de la soustraire aux autres images pour obtenir des images moins bruitées. Cette technique procure des résultats intéressants en éliminant le décalage entre les pixels, mais nécessite une mémoire supplémentaire, de la taille de l'image entière, uniquement pour mémoriser les données de correction. Le double échantillonnage est une technique d'élimination du bruit au sein du capteur durant l'acquisition des images. L'opération consiste à prélever le photovoltage et à le stocker dans un condensateur. Quand on

remet à zéro le pixel, on soustrait le photovoltage résultant de la remise à zéro du photo-voltage de l'image. Ceci revient au même que soustraire une image de référence de noir à la scène capturée. Généralement, cela est fait à la base de chaque bus-colonne par un circuit analogique spécifique [Mendis 97].

Le double échantillonnage corrélé (CDS en anglais *Correlated Double Sampling*) est une mesure qui consiste à prendre deux tensions sur une même exposition du pixel [El Gamal 98, Fujimori 00]. C'est-à-dire que l'on réalise deux lectures successives d'un même pixel pendant une même lecture de trame. Les deux échantillons sont corrélés, puisque le bruit et l'offset perturbent de manière identique la décharge. Une première tension est prise au début de l'intégration des photons. C'est la tension effective aux bornes de la photodiode à la fin du *reset*. Sa valeur réelle est Vreset - offset. Après le temps d'exposition (ou d'intégration), une deuxième tension est relevée. La mesure de la décharge est alors définie comme la différence entre ces deux mesures. Compte tenu des instants de mesure, le CDS élimine le bruit en kT/C présent lors de la pré charge de la photodiode. Cette mesure est précise, mais pose des problèmes de mise en œuvre. En effet, il faut mémoriser une information analogique sur un temps assez long (plusieurs millisecondes), ce qui ne peut pas se faire sans perte. En pratique, le CDS est donc utilisé principalement dans les pixels actifs à photo condensateur.

Pour pallier les problèmes rencontrés pour mettre en œuvre le CDS, une alternative a été développé : le double échantillonnage non corrélé (NCDS ou en anglais *Non Correlated Double Sampling* ou *Double Sampling*). Cette technique consiste à soustraire de la valeur lue en fin d'intégration le niveau de reset de la trame suivante. Elle soustrait deux niveaux de tension qui ont lieu pendant le temps où le pixel est accédé. Il n'y a donc pas de stockage de tension, et sa mise en œuvre s'en trouve ainsi grandement simplifiée. L'erreur causée par un offset non constant perturbe donc la mesure ; c'est pour cela que le NCDS est moins précis. Le double échantillonnage non corrélé permet une mise en œuvre bien plus aisée, surtout compte tenu du temps de mémorisation plus court (par rapport au CDS).

3.3.4 Types de capteurs d'image en technologie CMOS

Nous pouvons diviser les capteurs d'image CMOS en deux types distincts : les imageurs et les rétines. D'une manière générale, tout capteur fournissant une image perçue brute ou prétraitée est appelé imageur. De même, tout capteur générant une information (traitée) autre que l'image acquise est appelé rétine.

1 – Imageur CMOS

Un imageur CMOS est un capteur simple qui fournie en sortie l'image de la scène observée. Les technologies employées pour ces capteurs sont les technologies sub-microniques courantes largement amorties (c'est le cas également des rétines artificielles), et sur lesquelles il est possible d'intégrer des fonctionnalités classiques d'ASIC. Le fait de pouvoir intégrer une grande variété de circuits sur le

capteur lui-même permet d'améliorer (prétraiter) l'image perçue. Les circuits analogiques sont disposés le long du flux de données entre la réception photonique et la présentation des données en sortie sous format numérique. Nous pouvons distinguer 3 classes de circuits sur un imageur CMOS : les circuits d'adressage, les circuits de traitement du signal et les circuits de sortie. La première classe de circuits est les circuits de lecture associés à la matrice de pixels. De plus en plus, ces circuits permettent de modifier l'adressage de l'image pour obtenir des fonctions comme la taille variable, la position variable ou la résolution variable (*binning*) de l'image. Cela rend le capteur très versatile. La coexistence de ces caractéristiques aboutit à la création de matrices entièrement adressables, permettant d'accéder aléatoirement (*Random Access*) aux informations de l'image brute. La deuxième classe effectue un traitement du flux de données comme l'amplification, la conversion ou le prétraitement. Généralement, les imageurs CMOS disposent de circuits permettant la réduction du bruit, le contrôle des gains et offsets ainsi que d'un ou plusieurs convertisseurs analogique numérique. Quand à la dernière, elle gère la sortie des données. Les circuits gérant cette sortie dépendent de la façon dont est lue la matrice de pixel. Il y a deux manières de lire une matrice de pixels : la lecture série et la lecture parallèle. La première ne nécessite pas l'ajout d'électronique complexe après la conversion et elle est facile à mettre en œuvre. Malheureusement, la vitesse de lecture de l'information est inversement proportionnelle à la taille de la matrice. En règle générale, cette méthode de lecture s'utilise pour une matrice de petite taille. A l'inverse, le système à lecture parallèle est très rapide car on lit plusieurs pixels dans un même cycle d'horloge. Cela se traduit par l'ajout de mémoires (après conversion analogique numérique) pour stocker l'image lue avant de la transmettre à l'extérieur, soit par la multiplication des ADC et donc des sorties pixels. Mais cela nécessite l'accroissement de la surface de silicium.

Les capteurs CMOS de type APS (*Active Pixel Sensors*) sont actuellement les imageurs les plus répandus, car ils ont des performances en acquisition maintenant comparables aux capteurs CCD, tout en assurant des coûts de fabrication plus faibles et une plus grande souplesse. De plus, des imageurs CMOS APS (intégrant le capteur optique et la mise en forme numérique du signal vidéo) sont maintenant disponibles chez plusieurs sociétés. Ces capteurs s'intègrent alors parfaitement avec des processeurs de traitement (FPGA, DSP) ce qui permet de réaliser des caméras adaptables ou dédiées aux applications visées (concept de caméra intelligente).

2 – Rétine artificielle CMOS

Une rétine artificielle est un circuit VLSI monolithique où cohabitent intimement des opérateurs de transduction et de traitement de l'image. Il s'agit donc un capteur dit "intelligent" (en anglais s*mart sensor*). Les rétines sont des composants qui combinent les propriétés photoélectriques de la technologie CMOS avec ses caractéristiques électroniques. Cette approche consiste à intégrer tout ou une partie des traitements à réaliser sur le capteur lui même ou au plus proche du capteur en technologie CMOS. Cette intégration, sur un même substrat silicium, de dispositifs d'imagerie assurant

simultanément la capture de l'image et un traitement de cette image est apparue dans les années 1980 [Mahowald 88]. A la même époque, en France, Patrick Garda [Garda 88] proposa une rétine silicium basée sur des calculs entre des images binaires. L'idée est d'implanter au sein même du pixel un traitement qui va permettre de fournir non pas l'image perçue, mais des informations pertinentes sur l'image perçue. Comme le résume Thierry Bernard [Bernard 97a], "Analyser l'image là où elle est acquise pour n'en retenir et en transmettre qu'un extrait pertinent pour la tâche de vision en cours, tel est le principe des rétines artificielles".

De tels circuits permettent aussi de traiter directement l'information lumineuse, en modifiant le signal utile qui sera transmis. Ce traitement bas niveau de l'image a pour but d'alléger le calcul à effectuer par la suite. Le système de vision, classiquement composé d'un capteur et de périphériques de traitement des données (calculateurs) s'intègre ainsi dans un unique composant, afin de rendre les calculs plus rapides. La tâche du calculateur devient alors la supervision de l'algorithme de perception et sa commande. Cela permet de s'affranchir des problèmes de bande passante et de consommation d'un système sur carte.

Bien sur, le facteur de remplissage s'en ressent car cela implique en général plus de transistors qu'un imageur CMOS APS classique. Cependant, des solutions ingénieuses peuvent être trouvées. Cela dit, le facteur de remplissage est ici moins critique que pour les capteurs d'images classiques. Il en est d'ailleurs rarement fait état dans la littérature. En effet, le fait que ces circuits ne fournissent qu'une information sur l'image fait que l'influence d'une résolution moins grande est nettement moins gênante. Cela est également dû au fait que ces circuits seront plutôt utilisés dans des applications où l'information sera interprétée par un système et non visualisée directement (vision active, robotique, etc.).

Ce concept, que l'on peut décliner de différentes manières, offre de multiples avantages. Ces capteurs intelligents intégrés dans des petites caméras seront dons utilisés pour certaines applications particulières (détection, robot mobile, machine de vision, etc.) où l'information prétraitée permettra de simplifier la chaîne de traitement en aval ou d'accélérer sa réaction vis-à-vis de systèmes de vision classiques comprenant un capteur de vision en amont d'un processeur de traitement.

3.4 - Utilisation des capteurs CMOS en Traitement d'Image

Du fait de leur nature, les capteurs d'images CMOS ne sont pas cantonnés à un rôle de prise d'images comme les capteurs CCD. Les capteurs d'images CMOS ont trouvés leur essor dans les systèmes de vision dédiés à l'imagerie. Dans le traitement d'image, on distingue deux niveau : le traitement bas niveau (*low-level*) et le traitement haut niveau (*high-level*). Le traitement bas niveau est aussi appelé prétraitement. La séparation entre ces deux niveaux est mince et sujet à discussion ; le critère le plus répandu est la structure des données après traitement. Le traitement d'image bas niveau convertie une image en une autre image, par exemple le changement de contraste, la réduction de

bruit, ou calcul des caractéristiques simples de l'image d'origine, comme les contours, les histogrammes, etc. Tandis que le traitement d'image haut niveau interprète les données images généralement prétraitées, par exemple pour de la reconnaissance d'objet.

En raison de la grande quantité de données à traiter à tous les niveaux du traitement d'image, la puissance et la durée exigées par les calculs sont très élevées. Le choix de la stratégie d'implantation dépend fortement de la complexité de l'algorithme à mettre en oeuvre. La complexité de l'algorithme est classée selon trois types : les traitements simples, les traitements spatiaux et les traitements spatio-temporels.

Il existe de très nombreuses applications d'imagerie reposant sur l'utilisation d'imageurs ou de rétines CMOS. Le but est d'intégrer une fonction ou un algorithme au plus proche des cellules photosensibles afin de diminuer les temps de calcul et de ne transmettre que ce qui est important. Nous ne ferons pas une liste exhaustive des divers travaux dans ce vaste domaine, mais plutôt une présentation s'appuyant sur des exemples significatifs dans notre contexte applicatif : l'estimation de mouvement ou l'estimation directe du champ de vecteurs vitesse.

Les filtres d'estimation de mouvement sont l'application la plus évoluée au sein du capteur silicium. 'Evolué' dans le sens où l'information fournie en sortie de la matrice n'est plus une image mais un ensemble de primitives qui donnent une mesure du flot optique. L'élément de sortie d'un pixel est ici le vecteur vitesse qui donne une direction et une amplitude de vitesse. Cette estimation du flux optique est très importante dans de nombreuses applications où la perception du mouvement permet d'avoir accès à toutes sortes d'informations.

3.4.1 L'implantation *Off-chip*

Les systèmes de vision classiques séparent la partie traitement de l'acquisition de l'image. Cette approche permet d'utiliser toute la puissance de calcul des processeurs actuels. Le rôle du capteur ou imageur CMOS se restreint dans ce cas à transformer une image optique en un tableau de valeurs numériques exploitables par un processeur.

Les capteurs CMOS APS (*Active Pixel Sensors*) sont maintenant utilisés de façon industrielle, car ils ont des performances en acquisition maintenant comparables aux capteurs CCD, tout en assurant des coûts de fabrication plus faibles et une plus grande souplesse. Grâce aux technologies employées, des caméras vidéo en un seul circuit (intégrant le capteur optique et la mise en forme du signal vidéo) sont maintenant disponibles chez plusieurs sociétés. Ces capteurs intègrent parfois au niveau même du pixel des fonctions de correction de caractéristiques.

[Röwekamp 97] présente une architecture *smart sensor* pour l'estimation de mouvement (flot optique) pour la navigation autonome de véhicule. Leur système est caractérisé par un traitement temps réel sur des images hautes résolutions. Le cœur du système est un ASIC couplé à un imageur

CMOS et de la mémoire externe pour le stockage temporaire. Bien que basé sur un ASIC, le système est très flexible. Il dispose d'interfaces entrée/sortie standard permettant la connexion de divers imageurs et circuits de traitement d'image de haut niveau (caractérisation du mouvement). Suivant une méthodologie *top-down*, ils ont réalisé l'ASIC en utilisant le langage de description VHDL pour simplifier la conception et les tests (simulations temporelle et fonctionnelle). Les auteurs ont choisi l'algorithme de Horn et Schunck [Horn 81]. Les vecteurs vitesses sont itérativement calculés à partir des dérivés spatio-temporels de l'image. L'avantage principal de cette implémentation est qu'il traite les données image séquentiellement, pixel par pixel (*pixel pipeline*). Dans [Röwekamp 98], les auteurs présentent un prototype qui calcule un champ de 128×128 vecteurs vitesses à la fréquence de 50 images par seconde.

[Turenne 00] propose une approche basée sur un imageur CMOS spécialisé, de la mémoire RAM, un FPGA et un PC embarqué. Le PC embarqué est utilisé pour la gestion des données et leur envoi vers une station de travail. Il apporte au système la flexibilité des applications à distance (contrôle, optimisation des communications, réseaux, etc.). L'imageur fournit l'information temporelle non corrélée sur le mouvement d'un pixel. Le FPGA calcule le vecteur vitesse réel de clusters de pixel, basé sur l'information temporelle délivrée par l'imageur. Ce système fournit une bonne approximation du mouvement, en l'occurrence, la taille et l'orientation du mouvement dans l'image. Cette approche peut réaliser le calcul en temps réel de 30 images de mouvement par seconde.

Arias-Estrada [Arias-Estrada 98] propose une approche alternative pour le calcul de mouvement : l'utilisation d'un co-processeur de type FPGA. Dans l'article [Arias-Estrada 01], il présente une architecture pour la convolution générique en temps réel d'un masque et d'une image. Cette architecture est dédiée au traitement d'image de bas niveau rapide. L'architecture basée sur un FPGA permet l'implémentation d'un module efficace et compact pour traiter les convolutions. L'architecture est conçue pour réduire au minimum le nombre d'accès à la mémoire contenant les images. Elle est basée sur des modules parallèles contenant des opérations internes en pipeline afin d'améliorer leurs exécutions et performances. Cette architecture est implémentée dans un FPGA, mais elle peut l'être aussi dans des circuits VLSI dédiés.

3.4.2 L'implantation *On-chip*

Plusieurs stratégies d'implantation de processeurs de traitement au sein d'une rétine artificielle existent : auprès de chaque pixel, au niveau de chaque colonne ou au niveau de la matrice. Le choix de la stratégie dépend fortement de la complexité de l'algorithme à mettre en oeuvre. Les rétines dédiées à l'estimation du mouvement donnent l'information du déplacement qui peut être représenté par un champ de vecteurs. Ces vecteurs vitesses donnent une direction et une amplitude de vitesse. Bien que

les calculs soient tous basés sur les caractéristiques spatio-temporelles des images, à savoir l'évolution des illuminations des pixels au cours du temps, ces circuits peuvent se regrouper en trois catégories selon qu'ils sont basés sur des filtres, sur les contrastes, ou qu'ils s'appuient sur des calculs numériques.

1 – Filtres spatio-temporels

Les rétines basées sur les filtres spatio-temporels ou rétines bio-inspirées sont axées autour d'une structure commune : l'utilisation d'interactions au niveau des pixels. Elles utilisent le gradient des pixels à travers des réseaux résistifs [Stocker 98], ou des réseaux de filtres inspirés des rétines biologiques [Higgins 00, Moini 99, Sicard 99]. Ces implantations intègrent généralement soit un algorithme (existant), soit la résolution d'équations différentielles sous forme de calculs analogiques. Un réseau interconnecte les pixels à travers les transconductances, modélisant des résistances et/ou des capacités. Mais, leurs utilisations et le grand nombre de transistors nécessaires donnent aux pixels un taux de remplissage faible (de l'ordre 7 %). Les informations issues de ces rétines bio-inspirées sont des réponses à des filtres, donc analogiques, qu'il faut pouvoir stocker et/ou traiter de manière parallèle. La conception demande donc des blocs supplémentaires (mémoires analogiques, convertisseurs) permettant une interface plus aisée. Pour ces raisons de mise en œuvre, les circuits de ce type rencontrés dans la littérature ont toujours des résolutions faibles, par exemple 7×7 pixels [Stocker 98]. Certains de ces circuits bio-inspirés donnent une information de mouvement globale plus simple à mesurer [Higgins 00]. De manière plus générale, les rétines basées sur les réseaux de filtres sont complexes à réaliser dès qu'on prend en considération l'aspect parallèle analogique des sorties. Pratiquement, il n'existe pas de circuit de ce type traitant des images fortement texturées à une résolution suffisante, où des objets en mouvement constitueraient les centres d'intérêts.

2 – Contrastes ou variations de gradient

Une autre catégorie de rétines estimant le mouvement utilise des éléments caractéristiques de l'image : les contrastes, caractérisés par un fort gradient. Deux approches existent : le suivi des fronts de contraste ou le suivi d'impulsions générées par des gradients forts. Ce stimulus est alors cherché dans une direction qui est celle du mouvement. Ce type de mesure est intuitif et facilement compréhensible en considérant une scène "minimaliste" composée d'un objet se déplaçant sur un fond uniforme. Cependant, cette mesure perd de son efficacité et ne trouve plus sa place lorsque l'information scénique est très riche. Une forte texture donne en effet de nombreux déclenchements de ces structures, entraînant une grande difficulté d'interprétation. Aussi, l'hypothèse d'illumination constante est faite, à ceci vient s'ajouter le fait qu'une telle structure demande une surface conséquente pour effectuer une estimation en deux dimensions. [Higgins 98] propose en effet une telle architecture en prenant en compte les mouvements horizontaux et verticaux, mais pas les mouvements diagonaux.

L'autre approche basée sur le contraste consiste à extraire des impulsions à partir des variations du gradient de l'image. Excepté les travaux précurseurs de Tanner et Mead [Tanner 86], la plupart des réalisations utilisent le photorécepteur adaptatif développé par Tobi Delbrück [Delbrück 95]. Nous

pouvons cité les articles de Kramer et al [Kramer 97] ainsi que Deutchmann et Koch [Deutchmann 98]. Cette structure a une propriété de filtrage des stimuli qui donne une réponse aux variations de l'illumination, ce qui permet d'extraire les hautes fréquences, caractéristiques de l'image. L'image texturée est donc convertie en contours (hautes fréquences sous forme d'impulsions) dont le suivi est plus simple. La structure fonctionne en temps continu, la tension de sortie donne donc un signal à tout instant, couvrant une grande dynamique (6 décades). Cette structure a cependant quelques inconvénients : la nature adaptative des informations crée l'effet d'une égalisation des gradients qui peut causer des erreurs lors de la reconnaissance, et la surface occupée par les capacités n'est pas négligeable. Aussi, le nombre d'éléments de la structure (transistors et capacités) en fait une solution coûteuse en surface. Enfin, les détecteurs sont orientés, ce qui ne leur confère qu'une sensibilité deux dimensions (orthogonale).

3 – Calculs numériques

La dernière catégorie regroupe les rétines basées sur des structures de calculs numériques. Les rétines de ce type sont intéressantes car elles peuvent effectuer des calculs de différentes sortes sur les images. En effet, la plupart de ces réalisations reposent sur des machines parallèles intégrant des éléments de calcul dans les pixels. Plusieurs de ces rétines, associant phototransduction, unité arithmétique et logique, mémoire et registres ont été réalisées [Komuro 97]. L'exemple le plus intéressant du point de vue de la résolution et des résultats est la rétine PVLSAR [Bernard 97b, Paillet 99].

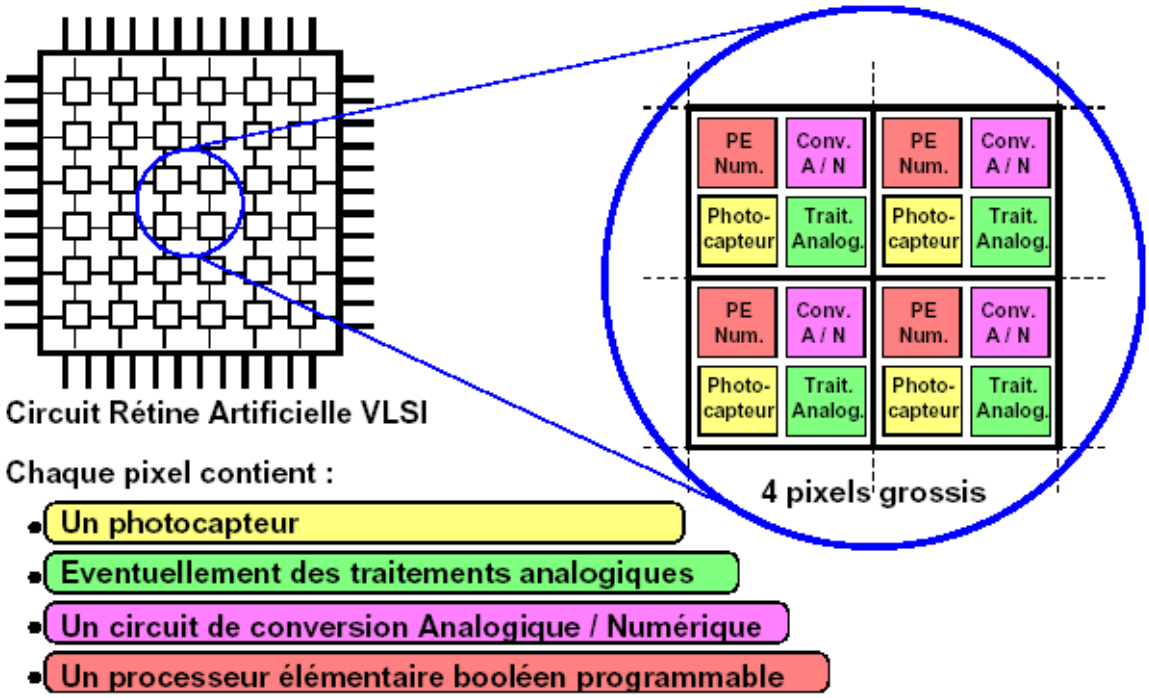


Figure 3-17 : Schéma de principe de quatre photosites de PVLSAR [Paillet 01].

Un processeur élémentaire est situé dans chaque photosite (Figure 3-17), et effectue des calculs booléens (AND, OR, XOR et copie) sur des images binaires. Cependant, l'accumulation des

informations au cours du temps permet de travailler sur des niveaux de gris, en extrapolant des seuils à partir des différents temps d'exposition.

Le photosite de cette rétine est composé d'un photorécepteur, d'un convertisseur analogique numérique et de fonctionnalités de calcul. Comme nous l'avons dit précédemment, le défaut de ce type de structures SIMD est qu'il est nécessaire de les piloter avec des séquences de commande extérieures, ce qui impose donc l'utilisation d'un séquenceur externe. De plus, le nombre d'instructions devient très conséquent lorsque des opérations complexes sont réalisées.

Une autre approche consiste à transposer un calcul numérique fortement itératif réservé aux processeurs numériques (FPGA, DSP) en une structure de calcul électronique. Une rétine, suivant cette approche, a été conçue pour estimer le mouvement [Navaro 03] avec comme principe d'acquérir une image et de la mémoriser sous forme binaire au sein même des pixels. Sa structure est celle d'un capteur DPS. Cette estimation de mouvement est basée sur une méthode robuste et efficace de mise en correspondance de blocs de pixels, comprenant une phase de précodage des pixels suivi d'une recherche de ce codage dans une fenêtre de destination potentielle. Dès l'acquisition de l'image, les valeurs analogiques présentes sur les photodiodes sont comparées, et le résultat des comparaisons est stocké dans une mémoire située dans chaque pixel. Cette étape correspond à la phase de codage des pixels. Ensuite, un bloc de calcul numérique parallèle effectue la phase de recherche des codes. Ce bloc de calcul est disposé en bout des bus-colonnes de la matrice, et traite les codes lors de leur transfert vers la sortie de la rétine.

3.5 - Capteur d'images CMOS dédié à notre application

Notre objectif est de concevoir un système de vision dédié à notre application : l'étude des déplacements du fluide en étudiant les trajectoires et les vitesses correspondantes en tout point de la zone d'étude. Pour réaliser ce système, nous devons choisir quel concept (caméra intelligente ou rétine artificielle) nous allons suivre. Cela permettra de disposer d'un ensemble d'acquisition et de traitement d'images, simple à mettre en œuvre et facilement transportable pour la visualisation d'écoulements.

3.5.1 Notre application : mesure de vecteurs vitesse dans des écoulements

La technique PIV repose essentiellement sur l'aptitude de l'imageur à estimer les positions initiale et finale des particules contenues dans un plan de l'écoulement afin de pouvoir calculer les champs de vecteurs correspondants (cf. Chapitre 1). La qualité des mesures dépend donc de la qualité des images enregistrées. Cela se justifie par le souhait d'un faible taux de mesures erronés en fin de traitement.

La méthode de mesure de vitesse que nous avons retenues se décomposent en cinq étapes (Figure 3-18) : le partitionnement de deux images successives en petites zones d'intérêt, l'extraction de

couples motif-imagette de ces zones d'intérêt, le déplacement du motif dans l'imagette et leur corrélation, la détection de la position du maximum de corrélation et la restitution des mesures sous forme de champs de vecteurs.

La corrélation ou la mise en correspondance de blocs ou de points n'effectue pas de détection de mouvement, mais estime directement le champ de vecteurs vitesse. Cette méthode consiste à comparer les intensités sur une fenêtre (ou motif) prise dans la première image avec une fenêtre de même taille dans la seconde image. Un voisinage de recherche est défini dans la seconde image. Un compromis doit être trouvé pour le choix de la taille du bloc et de la fenêtre. De plus, l'utilisation d'un algorithme de corrélation complexe donne de meilleurs résultats (précision et fiabilité), mais nécessite un nombre de calculs importants, essentiellement des multiplications et des additions. La complexité de ces calculs est proportionnelle à M^4 (où M est la taille de la fenêtre).

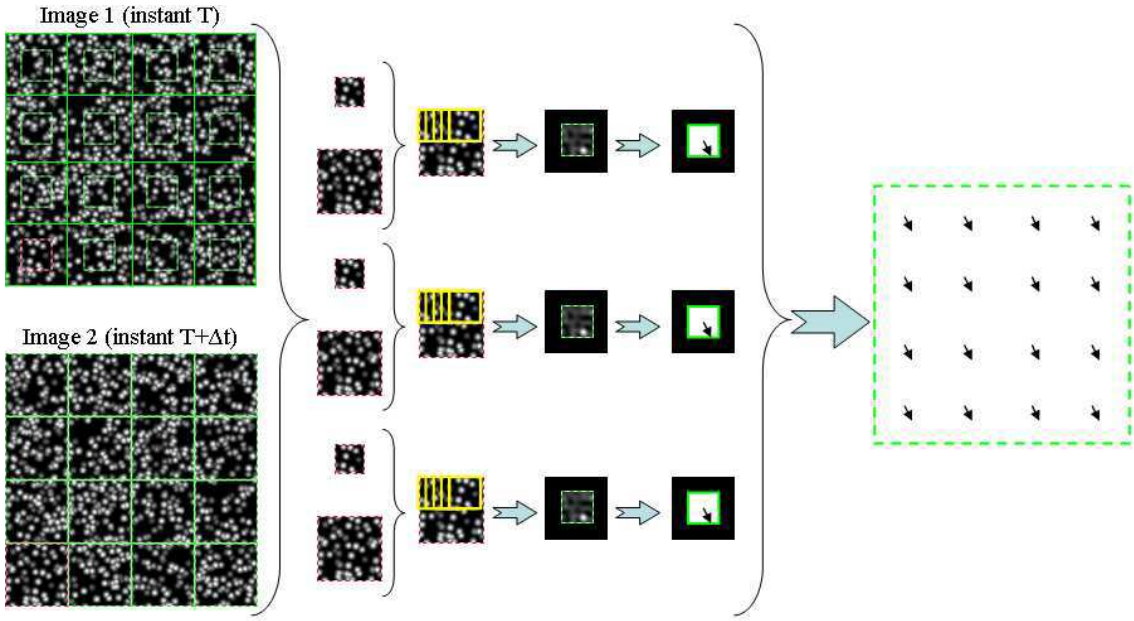


Figure 3-18 : Méthode de mesure de vitesse.

3.5.2 Etats de l'art : rétines électroniques et corrélation

L'implantation de dispositifs électroniques dans des rétines CMOS, destinés à réaliser une mesure de corrélation entre deux fonctions 2D, a donc déjà fait l'objet de plusieurs recherches et réalisation. L'idée de réalisation d'une rétine d'inter-corrélation n'est pas nouvelle et a été développée par J. E. Tanner et C. Mead en 1984 [Tanner 84], les avancées technologiques dans le domaine de la lithographie CMOS ont permis de remettre au goût du jour cette idée en lui apportant des modifications importantes.

Dans cette partie, nous aborderons les diverses approches dans le domaine des rétines en matière de mesure de corrélation entre deux fonctions. Ce que nous entendons ici ce sont des rétines réalisant une opération définie par :

$$f_c = \sum_{x=1}^{N} \sum_{y=1}^{M} g(x,y) \times h(x,y) \tag{3-6}$$

où g(x,y) et h(x,y) sont deux fonctions à deux dimensions et f_c correspond au coefficient de corrélation entre les deux fonctions. Ces rétines intègrent donc des opérations de multiplication et d'addition au sein même de la matrice de pixels.

1 - une multiplication analogique par pixel

De Weerth a proposé en 1992 une structure de rétine implantant dans chaque pixel une multiplication analogique entre le photocourant issu de chaque photodiode et une tension générée par un réseau de résistances [Deweerth 92]. Chaque pixel est constitué d'une paire différentielle aux bornes de laquelle est appliquée une différence de potentiel dont la valeur dépend des coordonnées du pixel. Le courant issu de la photodiode est réparti sur une des deux sorties du capteur proportionnellement à un facteur dépendant de la différence de potentiel. Finalement la différence de courant donne la position du centre de l'objet. Une rétine de 160×160 pixels a été réalisée suivant ce principe en technologie BiCMOS de 2µm.

2 - multiplication d'une valeur binaire par une valeur analogique

Tanner et Mead [Tanner 84] ont proposé en 1984 une rétine permettant la détection de la direction d'un mouvement par l'utilisation d'une mesure de corrélation. L'architecture est reproduite sur la Figure 3-19. Les photorécepteurs sont constitués par des photodiodes fonctionnant en intégration. A un instant donné, l'image acquise est mémorisée dans un point mémoire.

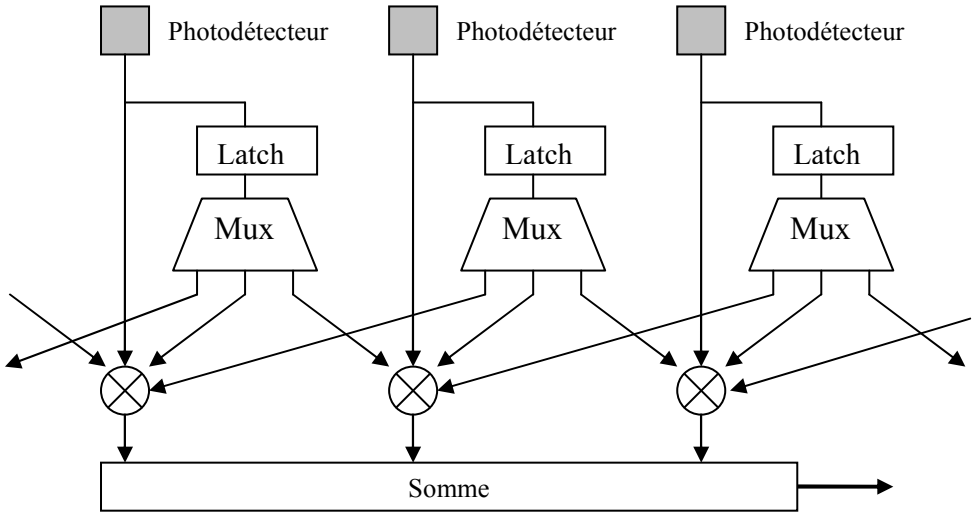


Figure 3-19 : Architecture de la rétine de Tanner et Mead.

Cette image mémorisée est par la suite corrélée avec l'image suivante de trois manières différentes : avec un décalage d'un pixel sur la droite, sans décalage, avec un décalage d'un pixel sur la gauche. La rétine donne donc trois mesures de corrélations. A partir de ces trois valeurs, il est alors possible de déterminer si l'image projetée sur la rétine a subit un décalage vers la droite, la gauche ou

aucun décalage. La multiplication est assurée par un aiguillage du courant piloté par la donnée issue du point mémoire. La sommation est très simplifiée par le fait que les signaux de sortie sont des courants.

3 - rétines à image gravée

Cette rétine a été réalisée au Laboratoire LE2I du Creusot. La zone photosensible est constituée d'une unique photodiode sur laquelle a été gravée une image (un coq) par dépôt de métal (Figure 3-20). En se basant sur la description décrite dans [Goria 99], la photodiode mesure donc la quantité de lumière d'une image projetée sur la rétine et modulée par l'image gravée. On retrouve ici un principe proche du corrélateur optique par ombroscopie. Outre la simplicité de son architecture, l'originalité de ce capteur résulte dans l'utilisation d'un algorithme de génération d'image en demi ton, pour générer l'image gravée sur la photodiode. La sortie la rétine donne alors une mesure de corrélation entre les deux images avec une erreur de mesure inférieure à 1% pour une image de dimension 256×256 pixels.

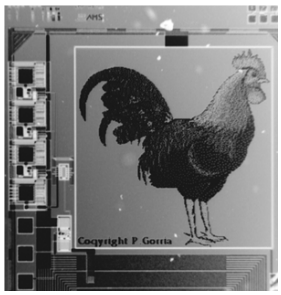


Figure 3-20 : Rétine à image gravée du Laboratoire LE2I du Creusot.

4 - rétines à masques

Les résultats de l'étude précédente ont abouti à la conception d'une rétine permettant de programmer l'image de référence [Bellach 03]. L'objectif de cette rétine est de déterminer le degré de corrélation de deux images. Le calcul de la corrélation entre deux images se fait ainsi : une image de référence est mémorisée, ensuite les variations entre l'image courante et l'image mémorisée se lit sur les sorties analogiques du circuit. Cela rend possible la reconnaissance d'un objet à partir de la mesure du coefficient de corrélation entre une image binaire (appelée masque) et une image analysée [Aubreton 04].

Dans une première phase dite phase de programmation, l'image de référence est binarisée puis mémorisée dans le plan mémoire de la rétine. Ensuite vient la seconde phase dite phase d'analyse où une image est projetée sur la rétine, un système électronique simple intégré dans la rétine fournit alors de manière instantanée deux signaux analogiques représentant le degré de ressemblance entre l'image préalablement binarisée et l'image courante. Une simple opération de seuillage de l'image de référence fournit deux masques : le premier appelé masque positif, correspond à la partie de l'image dont le niveau de gris est supérieur au seuil de binarisation, ce seuil correspond à la valeur moyenne des niveaux de gris de l'image de référence. Le deuxième masque négatif correspond au complément du masque positif. Une fois la phase de programmation terminée, l'image à analyser est projetée sur la

rétine qui fournit instantanément deux signaux analogiques : l'un appelé Iblanc qui correspond à la somme des courants issus des pixels appartenant à la partie blanche du masque positif et l'autre appelé Inoir qui correspond à la somme des courants issus des pixels appartenant à la partie blanche du masque négatif.

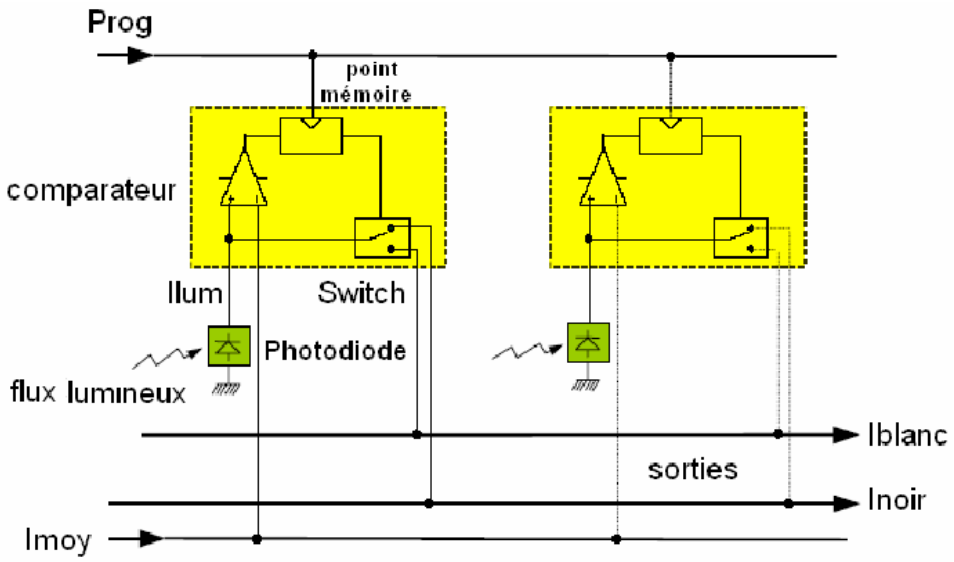


Figure 3-21 : Schéma fonctionnel du pixel [Bellach 03].

3.5.3 Notre Choix

Les contraintes liées à notre application, la mesure de paramètres physique par traitement d'image, vont nous permettre de choisir l'approche technologique (rétine électronique ou imageur CMOS couplé à un circuit de traitement du signal). Cet objectif est de réaliser un ensemble d'acquisition et de traitement d'images temps réel, simple à mettre en œuvre, facilement transportable pour la visualisation d'écoulements et performant.

En résumé, les contraintes de notre application sont que :

- les images aient une bonne résolution ;
- le capteur possède une excellente sensibilité ;
- le temps d'intégration soit constant et paramétrable ;
- les données soient disponibles pour le traitement dans un temps réduit ;
- les traitements des images et l'affichage des résultats s'effectuent en temps réel.

La résolution de l'image requise pour notre application est très forte car la méthode de traitement est statistique. Plus nous avons d'information sur les images sources, plus nous augmentons la précision des mesures. Mais, les prises de vue se font sous des conditions lumineuses faibles, sur des mouvements très rapide de nombreux objets de petites tailles rendu visibles par leurs propriétés de réflexion de la lumière (par exemple : des bulles d'air dans de l'eau à des vitesses comprises entre 0,52

et 3 m/s). Tout ceci impose de disposer d'un capteur d'excellente sensibilité pour éviter de perdre des informations et pour ne pas avoir à faire de fastidieux réglages optiques de compensation.

Pour effectuer la corrélation, il nous faut disposer d'une paire d'images successives. Pour cela, il faut pouvoir contrôler le temps entre deux acquisitions. Ce temps dépend de deux caractéristiques d'un capteur : le temps d'intégration et le temps de lecture de la matrice. Le temps d'intégration doit être identique en tout pixel de la matrice, c'est-à-dire que les images doivent posséder une cohérence temporelle. Pour le temps de lecture de la matrice, il doit être le plus rapide possible sans engendrer de pertes d'informations. Actuellement, dans les capteurs CMOS, le problème est que ces deux temps sont interdépendant l'un de l'autre et que leur contrôle est souvent limité voire inexistant. Pour pallier ce problème, l'utilisation de capteurs CMOS rapides permet un pseudo-contrôle du temps inter image mais cela ne permet plus le respect des contraintes de traitement temps réel (goulot d'étranglement au niveau des accès mémoires et de leur gestion).

Pour développer notre système de vision, l'approche rétine artificielle semble apporter des avantages mais certaines limitations la rendent coûteuse en temps et difficile à réaliser. L'intégration de notre système sur rétine artificielle assure une compacité et une vitesse de traitement incomparable avec les systèmes conventionnels. Mais la fiabilité de ces traitements n'est pas certaine car des contraintes technologiques, comme la dispersion des tensions de seuil des transistors MOS, font que certains algorithmes perdent en précision par rapport à leur implémentation sur d'autres systèmes. Dans le cas de notre application, l'approche rétine artificielle limiterait la résolution spatiale (compromis entre la résolution spatiale et la complexité des traitements à réaliser) et restreindrait la programmabilité du système. Dernier point limitatif, bien que de nombreuses rétines artificielles réalisent des fonctions similaires à notre application, elles sont le fruit de longue recherche scientifique et technologique ce qui n'est pas notre cas.

La deuxième approche (imageur CMOS couplé à un circuit de traitement) semble plus réaliste et appropriée à notre objectif. Elle apporte une flexibilité et une programmabilité nécessaire pour mettre au point un prototype fonctionnel de notre système embarqué. De plus, une étude précédente [Dubois 01b] ayant développée, pour la même application, un système électronique fait de plusieurs circuits de traitement couplés à une caméra CCD, cette approche permet de reprendre et de continuer ces travaux. Enfin, elle nous permettra de définir si l'ensemble de notre système embarqué peut être réalisé dans une rétine artificielle ou si seulement une partie est intégrable sur un capteur d'image CMOS.

Nous allons donc développé un système embarqué compact et temps réel, composé d'un imageur CMOS APS possédant une qualité d'image équivalente aux capteurs CCD associé à un processeur de traitement performant au plus proche du capteur.

Chapitre 1

Systèmes embarqués monopuces

Notre but est d'exploiter l'image, et plus particulièrement son contenu, puis de transmettre les résultats d'analyse/traitement, ce qui nécessite une puissance de calcul importante alliée à des moyens de communication et de mémorisation conséquents. Initialement réalisé par des stations de travail, ce travail peut avantageusement être effectué par des systèmes électroniques embarqués.

Par définition, un système embarqué est un système inclus dans un autre système. Donc, le domaine des systèmes embarqués est très vaste : il dépasse largement le cadre de ce travail de thèse. Ici, le terme « système embarqué » désigne tout sous-système électronique spécifique à un domaine d'application et devant être intégré dans un système complet tel qu'un appareil, un instrument, une plate-forme expérimentale ou un véhicule.

Un système embarqué est divisé en deux parties : une partie logicielle et une partie matérielle. Les systèmes électroniques embarqués possèdent des contraintes particulières par rapport aux autres systèmes électroniques :

- un encombrement réduit (transportable et compact),
- une consommation minimale (ressources limitées en énergie : piles, batteries),
- des contraintes de temps (applications temps réel),
- une bonne fiabilité (robustesse).

Envisager l'intégration sur une même puce d'un système embarqué complet permet de satisfaire une partie des contraintes qui leurs sont imposées. L'évolution des technologies sub-microniques fait que la partie matérielle est de plus en plus constituée de systèmes monopuces ou SoC (pour *System on-a-Chip*) contenant des processeurs (câblés ou logiciels) pour exécuter le logiciel embarqué, des composants spécifiques et parfois même des éléments analogiques. Les systèmes monopuces présentent les atouts d'une grande puissance de calcul, haute intégration, faible consommation (les rendant ainsi complémentaires des capteurs d'image CMOS APS pour la réalisation

de capteurs "intelligents" ou de systèmes d'imagerie complets), mais ils peuvent aussi aisément s'adapter aux divers besoins des concepteurs.

Notre travail de thèse s'intéresse à ce type de système embarqué, et tout particulièrement aux systèmes monopuces. Dans la première partie de ce chapitre, nous présentons les systèmes sur puce de façon générale. La deuxième partie relate notre approche de conception des SoCs : les SoPCs ou *Systems on a Programmable Chip*.

4.1 - Système sur puce

Les applications embarquées appartiennent à un domaine en très forte expansion. Néanmoins, elles sont de plus en plus soumises à de fortes contraintes fonctionnelles (telle que la puissance de calcul et des communications performantes). D'un autre côté, le progrès technologique permet une capacité d'intégration de centaines de millions de transistors sur une seule puce de silicium. Ce progrès technologique permet une meilleure localisation des ressources, c'est-à-dire le regroupement dans un minimum de circuits de l'ensemble du système. L'architecture associée à un flot de conception efficace traduit le potentiel de la technologie en performance et capacité. Ce potentiel permet d'exploiter au mieux les parallélismes de l'application, et aussi de spécialiser l'architecture. Le parallélisme implique l'utilisation de plusieurs processeurs ou composants de calcul (multiprocesseur), la localité implique un minimum de circuits voire un seul, et la spécialisation implique de la dédier à une application spécifique ou une famille d'applications comme la mesure par traitement d'image. Quant au flot de conception, manipuler un grand volume de ressources revient à remonter le niveau d'abstraction et proposer une méthodologie systématique pour le passage de ce niveau à une implémentation optimale.

A la différence des systèmes à processeurs classiques [Culler 98, Patterson 04], les systèmes monopuces sont dédiés à des applications spécifiques et taillés sur mesure pour satisfaire seulement les besoins de l'application visée.

4.1.1 Définition

Un système sur puce, encore appelé système monopuce ou SoC (pour *System on-a-Chip*), désigne l'intégration d'un système complet intégrant plusieurs composants complexes et hétérogènes sur un seul circuit ou puce de silicium. Ce terme est devenu très populaire dans le milieu industriel malgré l'absence d'une définition standard. Certains considèrent qu'un circuit complexe est automatiquement un SoC, mais cela inclurait probablement chaque circuit existant aujourd'hui.

Une définition plus appropriée d'un système monopuce serait : un système complet sur une seule pièce de silicium, résultant de la cohabitation de nombreuses fonctions déjà complexes en elles-mêmes telles que des processeurs, des DSP (*Digital Signal Processors*), différents types de mémoires,

des bus ou tout autre réseau de communication, des convertisseurs, des blocs analogiques, etc. Le système monopuce doit comporter au minimum une unité de traitement de logiciel (ou CPU) et doit dépendre d'un minimum de (voire d'aucun) composants externes pour exécuter sa tâche. Ce type de système doit être quasi autonome en fonctionnant indépendamment d'éléments externes à la puce. En conséquence, il nécessite à la fois une partie matérielle (*hardware*) et une partie logicielle (*software*) comme un système embarqué. La partie matérielle peut être décomposée en deux couches :

- la couche basse contenant les principaux composants utilisés par le système : CPU, mémoires et périphériques ;
- la couche de communication matérielle embarquée sur la puce.

Bien que les réseaux de communication soient composés d'éléments standard, il est souvent nécessaire d'ajouter des couches d'adaptation entre ces réseaux et les composants de la première couche.
La partie logicielle est aussi découpée en couches :

- la couche de l'application : le programme ;
- la couche de gestion de ressources permettant d'adapter l'application à l'architecture.

Cette dernière couche permet d'isoler le logiciel embarqué de l'architecture matérielle. Ce lien avec le matériel se fait via des pilotes d'entrées/sorties et d'autres contrôleurs de bas niveau comme pour une station de travail et son système d'exploitation.

Pour optimiser un système embarqué, il faut adapter les ressources matérielles et logicielles aux besoins spécifiques de l'application. Un système monopuce étant la cohabitation de ces ressources sur une même puce, notre travail s'est orienté vers la définition et la réalisation d'un SoC à partir d'un système dédié à une application spécifique existante.

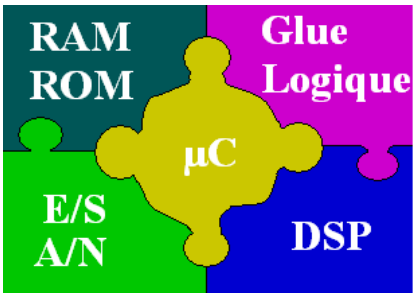


Figure 4-1 : Principe d'un système monopuce typique.

La Figure 4-1 présente le principe des systèmes monopuce. Il se compose d'un coeur de processeur (µC : microcontrôleur ou µP : microprocesseur ou CPU) auquel on associe divers composants tel qu'un processeur de traitement du signal numérique (*Digital Signal Processor* ou autres), de la mémoire embarquée, et des périphériques tels qu'un contrôleur d'Entrées/Sorties. L'ensemble est assemblé par des éléments logiques ou adaptateurs de communication matériels (glue logique).

Il y a plusieurs raisons pour lesquelles la solution monopuce représente une manière attrayante pour implanter un système embarqué. Les technologies de fabrication (de plus en plus fines)

d'aujourd'hui permettent de combiner la logique et la mémoire sur une seule puce, réduisant ainsi le temps global des accès mémoires (goulet d'étranglement de tous systèmes embarqués). Si les besoins en mémoire de l'application ne dépasse pas la taille de la mémoire embarquée sur la puce, la latence de mémoire sera réduite grâce à l'élimination du trafic de données entre des circuits séparés ainsi que le nombre de broches du système et donc la consommation d'énergie.

Les principes de conception d'un système sur puce sont :

- la construction d'une architecture matérielle composée de blocs logiques standard (processeurs, mémoires), de blocs logiques spécifiques et de bus de communications,
- le développement de ressources logicielles.

Le but de la conception de SoC est que l'ensemble de l'architecture ne requiert l'utilisation que d'un seul circuit. Dans la pratique une architecture SoC peut également contenir un ensemble d'interfaces d'entrées/sorties vers d'autres circuits comme, par exemple, de la mémoire, des périphériques *off-chip* tels que des capteurs, et des interfaces de communication vers d'autres systèmes comme une station de travail. Le terme système sur puce désigne donc des architectures où toutes les parties essentielles des systèmes de calcul (ou traitement) ont été intégrées dans un seul et unique circuit spécialisé.

Comme les SoC sont conçus dans le but d'un usage limité à une catégorie d'applications, ils tendent à avoir besoin de moins de capacité de traitement qu'un ordinateur standard. Tandis qu'une station de travail moderne fonctionne à des fréquences de l'ordre de trois gigahertz, l'unité de traitement d'un SoC n'a besoin pour fonctionner que d'une centaine de mégahertz voire moins. Un coeur de processeur idéal de SoC fonctionne à la fréquence minimale requise pour accomplir correctement le travail à effectuer. En utilisant la plus basse fréquence permise, la consommation d'énergie et la température du circuit en sont réduites au strict minimum. Ceci permet aux SoC de fonctionner avec moins de dispositifs de refroidissement et de faire une meilleure utilisation de l'énergie, donc de satisfaire au mieux les contraintes d'un système embarqué.

4.1.2 Architectures d'un système monopuce

Les premiers SoC était des architectures monoprocesseurs qui contrôlaient des périphériques par des communications de type maître/esclaves. Le processeur est dédié au calcul et au contrôle de l'ensemble du système. La réaction aux événements était effectuée par le biais de routines de traitement d'interruptions (similaire aux programmes de microcontrôleur). Ces premiers systèmes embarqués ne pouvaient fournir que des fonctions simples ne requérant que peu de puissance de calcul. Leur architecture ne peut pas supporter les fonctionnalités requises pour les systèmes embarqués actuels à qui il est demandé non seulement d'effectuer du contrôle, mais aussi des calculs

complexes tels que ceux requis pour le traitement numérique du signal. Avec le progrès technologique et la capacité d'intégration de centaines de millions de transistors sur une seule puce, deux tendances architecturales ont émergé pour relever ce défi.

La première tendance s'est restreinte à l'utilisation d'architectures monoprocesseurs tout en améliorant considérablement les performances du CPU (fréquence, puissance de calcul) utilisé et par l'utilisation de co-processeurs spécialisés. Le CPU ou processeur central est dédié au contrôle de l'ensemble du système. Les co-processeurs ou processeurs annexes sont utilisés pour le calcul. Un tel processeur central se distingue par : une fréquence de fonctionnement très élevée, des structures matérielles spécialisées, un ensemble d'instructions sophistiquées, plusieurs niveaux de hiérarchie mémoire (plusieurs niveaux de caches), et des techniques spécialisées d'optimisation logicielle (nombre d'accès mémoire, taille, etc.). Dans cette direction, on peut citer : PowerPC, Intel Pentium 4, ST100, et beaucoup d'autres processeurs de type VLIW ou superscalaire. Dans de telles architectures, la communication est basée sur le principe maître/esclaves : le processeur central est le maître et l'ensemble co-processeurs et périphériques sont les esclaves. Les interfaces des co-processeurs sont généralement faites de registres transposés dans la mémoire du processeur central et peuvent lui produire des interruptions. Ces communications se font généralement via des bus partagés : chaque processeur pouvant disposer de son propre bus de communication. En termes de performance, de telles architectures - centrées autour d'un seul CPU - ont un inconvénient : la dégradation des performances engendrée par le fait que le processeur central gère la totalité du système de la communication au calcul.

La deuxième tendance s'est tournée vers des architectures multiprocesseurs. Ces architectures qui étaient réservées jusqu'à maintenant aux machines de calculs scientifiques tel que CRAY [Scott 96] ne le sont plus avec le progrès de la technologie (plusieurs millions de portes sur un seul circuit). Ainsi, l'idée de ceux qui ont choisi d'adopter cette tendance est la suivante : il s'avère de plus en plus difficile d'avoir des architectures monoprocesseurs assez rapides et flexibles alors que les architectures multiprocesseurs sur puce deviennent réalisables. Les principales raisons de l'attrait des architectures multiprocesseurs monopuces sont : le parallélisme, la réduction des temps (calculs et communications) et le temps de conception mieux maîtrisé. Alors que dans la première tendance, le goulot d'étranglement était les ressources en calcul, ici, il se situe plutôt au niveau des communications et de leur gestion. Ce sont elles qui définissent désormais l'architecture et non plus les ressources de calcul.

Un SoC à architecture multiprocesseur permet de réduire le temps de conception car il permet l'instanciation de plusieurs composants pré-validés sur une même puce. En outre, ces composants pré-validés peuvent être réutilisés dans d'autres SoCs voire pour d'autres applications. Seule la logique d'interconnexion entre composants et les communications sont à adapter voire à reconcevoir. Mais cet inconvénient tend à disparaître par l'utilisation de réseaux de communications complexes. Cependant,

un des inconvénients de tels SoCs est que les problèmes de synchronisation et de communication entre les différents processeurs surviennent. Ceci rappelle encore une fois l'importance de l'architecture de communication dans les systèmes multiprocesseurs.

Ces architectures multiprocesseurs intègrent des processeurs de divers types (CPU, DSP, ASIP, IP Core, etc.) et des réseaux de communications complexes (bus hiérarchiques, bus avec protocole TDMA, connexion point à point, structure en anneau et même des réseaux de communication par paquets). On parle même aujourd'hui de réseaux de composants monopuces (*Networked system on-a-Chip* contracté en *Networks on-Chip* ou NoC) [Jantsch 03]. Le principe revient à assembler des composants standard pour en faire un système complet comme on le ferait avec des cartes électroniques. Ainsi, le problème n'est plus tant de concevoir des composants efficaces mais surtout de les assembler efficacement afin d'obtenir une architecture qui respecte les contraintes de performance et de coût. Donc, la validation globale de l'ensemble hétérogène du système avant sa réalisation devient un élément primordial au niveau de la conception du système.

Pour résumer, l'architecture d'un système monopuce se compose typiquement d'un ou plusieurs coeurs de processeur, de périphériques de toutes sortes, des mémoires embarquées, le tout relier par un réseau de communication (Figure 4-2).

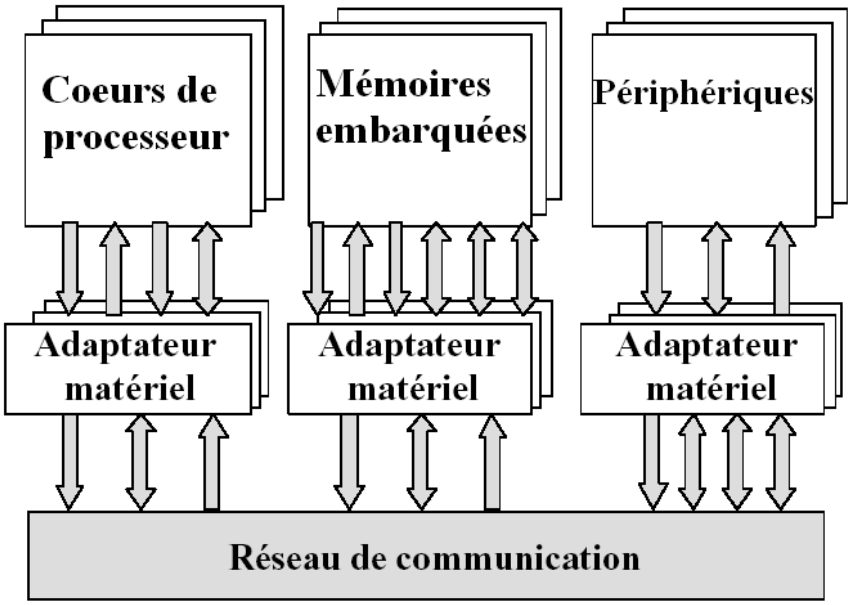


Figure 4-2 : Architecture d'un système monopuce.

4.1.3 La conception d'un SoC

Le principe de la conception de tel SoC reste le même ; il s'agit de générer une réalisation physique sous forme d'une puce en partant d'une spécification système. Par contre, les outils mis en œuvre et l'organisation du travail durant le processus de conception ont beaucoup évolué. Partant

d'une conception complètement manuelle où l'on dessinait les masques du circuit à réaliser sur du papier spécial, on est passé à une conception quasi automatique en partant d'une description du comportement du circuit sous forme d'un programme décrit dans un langage de description matériel de haut niveau [Jerraya 02].

La conception de SoC apporte donc des changements importants dans les flots de conception classiques (Figure 4-3). Notamment, la frontière entre le logiciel et le matériel n'est plus aussi nette : en effet avec les anciens systèmes, le matériel était déjà conçu lorsqu'il fallait concevoir le logiciel. Avec les systèmes monopuces, les deux doivent être conçus en même temps. Cela augmente la complexité de la conception, mais cela offre aussi plus de liberté : chaque composant peut être réalisé en logiciel, en matériel, ou de manière mixte.

Actuellement, la conception de SoC pose plusieurs défis dont les principaux sont :

- la réutilisation de composants existants qui n'ont pas été conçus spécialement pour être assemblés sur la même puce ;
- l'utilisation de processeurs programmables accompagnés, en plus du matériel, d'une couche logicielle ;
- la conception d'interfaces de communication hétérogènes mettant en œuvre des parties matérielles, logicielles et non numériques ;
- les difficultés grandissantes de la vérification des composants au sein de ces systèmes complexes.

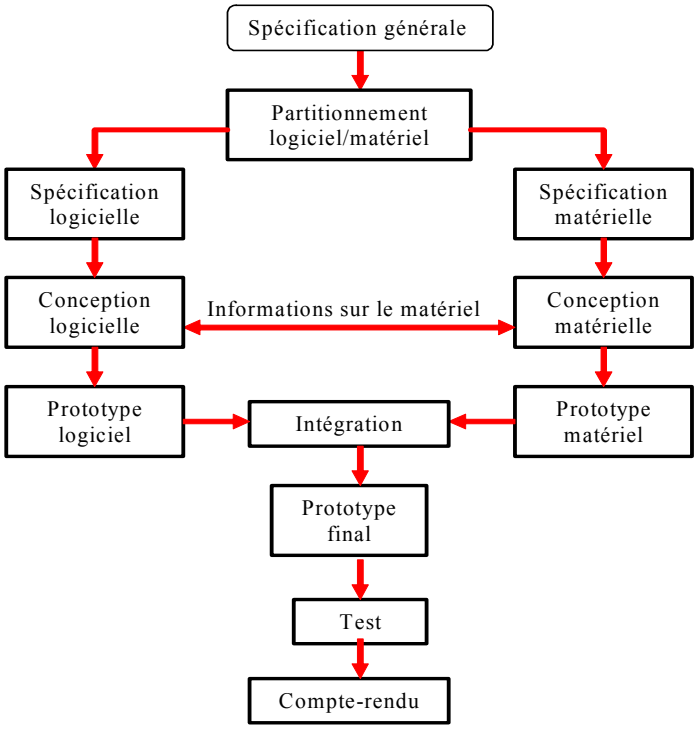


Figure 4-3 : Les flots de conception classiques pour les systèmes embarqués.

Depuis quelques années, nous assistons à la définition de méthodologies de conception permettant de réduire le fossé qui existe entre la productivité des concepteurs et la capacité d'intégration offerte par les technologies sub-microniques. Ainsi, les flots de conception autorisant la co-conception (*Co-Design*), la réutilisation de composant virtuel (ou IP) et la conception au niveau système réduisent de façon considérable les coûts de conception des systèmes complexes.

Ces méthodes reposent sur l'orthogonalisation de différents aspects qui permettent une plus large exploration de l'espace de solutions d'implantation [Balarin 97, Rowson 97, Keutzer 00]. L'orthogonalisation concerne : (1) la fonctionnalité et l'architecture, (2) les calculs et la communication et (3) les besoins/contraintes et les choix d'implantation. Cette dissociation requiert la définition de modèles, de niveaux d'abstraction, de méthodes permettant le passage d'un niveau de raffinement à un autre et de flots de conception adaptés.

Un flot de conception repose sur un ensemble d'étapes permettant de fournir, à partir d'une spécification initiale, l'implantation d'une application. La complexité de ce problème de transformation nécessite un processus de raffinement incrémental et la définition de modèles intermédiaires d'abstraction adéquats. Un flot de conception est constitué d'une suite d'alternances de phases d'exploration à un niveau d'abstraction et de raffinements entre niveaux d'abstraction. Cette gestion par palier permet de réduire la quantité d'information que le concepteur doit gérer à un niveau donné évitant ainsi les interactions avec les niveaux inférieurs (Figure 4-4).

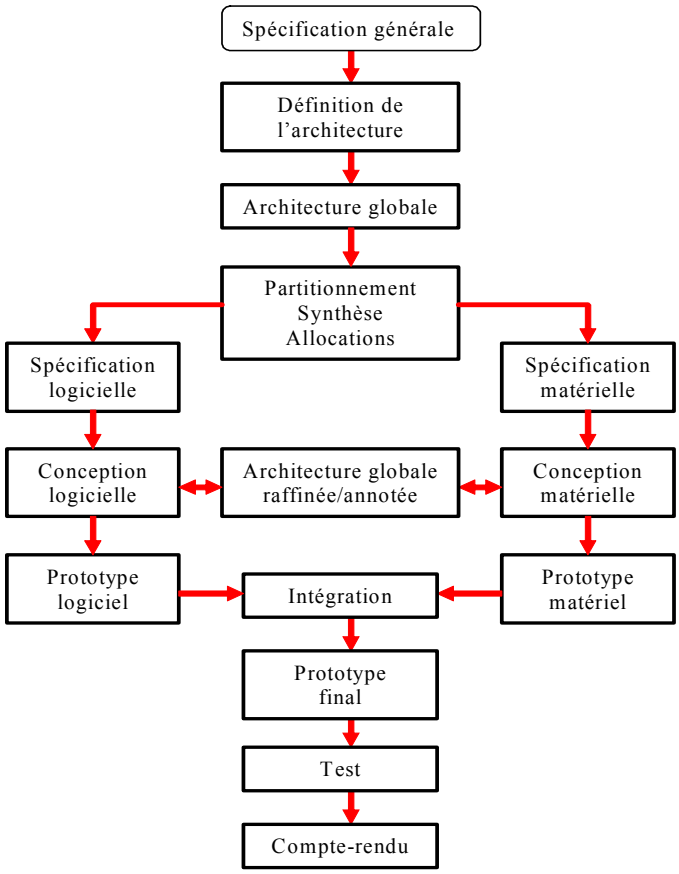


Figure 4-4 : Co-Design pour les systèmes embarqués.

Les nombreuses solutions apportées par la recherche académique ou industrielle peuvent être regroupées en trois familles [Cyr 01, Cai 03] : la synthèse à partir d'une spécification système fonctionnelle (*synthesis from specification or system synthesis*), la conception s'appuyant sur l'utilisation de plate-forme matérielle (*Platform-Based Design*) et la conception basée sur la réutilisation de composant (*Component-Based Design or IP assembly*). Nous ne détaillerons pas plus cette section sur les flots de conception puisque ce n'est pas l'objet de notre thèse. Nous conseillons pour approfondir ce thème de vous reporter aux références suivantes [Nuta Nicolescu 02, Jerraya 04].

4.1.4 Support physique pour les systèmes monopuces

Le développement de nouvelles architectures a pour but d'augmenter les performances et la rapidité des systèmes monopuces. Trois approches technologiques existent aujourd'hui pour l'implantation de systèmes monopuces performants, caractérisées par leur degré de flexibilité. La première est basée sur les processeurs généralistes. A l'opposé, la seconde se base sur la définition d'architectures complètement dédiées. Et, la solution intermédiaire consiste à employer des circuits programmables.

Nous évaluerons dans cette section les avantages et inconvénients des trois approches, en insistant sur trois critères : les performances des systèmes, leur coût financier et la facilité de développement des applications pour chacune d'entre elles.

1 - Les circuits à architectures généralistes

L'usage d'un processeur généraliste facilite la réalisation un SoC dans la majorité des cas grâce à leur grande flexibilité d'utilisation et de programmation. De plus, la puissance des processeurs généralistes s'accroît soit en augmentant la fréquence d'horloge soit en exploitant le parallélisme des calculs. Cela est réalisé dans ces processeurs en utilisant les techniques super scalaires (multiplication des unités fonctionnelles) et pipelines (décomposition d'une tâche en une suite d'instructions séquentielles, exécutées sur des ressources distinctes). Malgré la puissance en constante augmentation qu'ils délivrent, les processeurs généralistes sont souvent surdimensionné par rapport à la fonctionnalité logique à implémenter par logiciel, et/ou exigeantes en termes de performance (temps réel), de consommation ou de place. L'usage de circuits logiques programmables ou de circuits dédiées est alors une solution plus intéressante. Mais l'usage d'un processeur généraliste plus spécialisé comme un DSP peut aussi réduire considérablement le surdimensionnement tout en gardent la flexibilité. Leur coût financier est relativement important dans le cas d'applications spécialisées pour une large diffusion, où leur flexibilité n'est pas nécessaire [Rubini 97].

2 - Les circuits à architectures dédiées

A l'opposé de la solution des processeurs généralistes, cette solution consiste à proposer des architectures complètement dédiées à l'application en utilisant des circuits de type ASIC (*Application Specific Integrated Circuit*). Ces circuits sont apparus vers le début des années 80. Ceux-ci sont dédiés à une application, pour lesquelles ils fournissent la puissance de calcul et les performances requises [Ku 92]. Un ASIC peut mettre en application des conceptions simples mais également de grandes conceptions telles que des systèmes monopuces à multiprocesseur. Généralement, conçu pour une application unique et spécifique à un domaine (vidéo, télécommunication, traitement d'image, etc.), un ASIC peut, à cette fin, être adapté aux besoins de l'application comme la dissipation réduite de puissance, une puce très petite ou de grandes fréquences d'horloge.

Les circuits ASIC nécessitent un temps (donc un coût) de développement très important. Cependant, une fois mis au point, le circuit peut être produit en grande série avec un coût de production unitaire faible. Leur inconvénient majeur, si une large diffusion est visée, réside donc dans les temps de mise à disposition (plusieurs mois) parfois incompatibles avec les exigences du marché (notion de *Time-To-Market*). Les temps de développement sont encore accrus pour les systèmes multi-composants, car les interactions complexes entre eux et avec l'extérieur nécessitent lors de leur développement une phase de simulation prohibitive pour atteindre un degré de confiance suffisant.

Le développement d'un circuit ASIC est donc une solution pour la production en grande série de SoCs dédiés, si le temps de développement induit est acceptable, et si son coût, en terme de mise au point, est compensé par la taille du marché visé. L'utilisation de circuits ASIC est par contre incontournable si le facteur d'intégration est déterminant pour des raisons de place et/ou de puissance délivrée. Les inconvénients de l'ASIC sont la non reconfigurabilité, de longues phases de conception et des coûts élevés pour le prototypage ou la production en petite série.

3 - Les circuits à architectures programmables

Une solution intermédiaire entre les processeurs généralistes et les circuits complètement dédiés s'appuie sur la technologie en plein essor des circuits logiques programmables ou PLD (*Programmable Logic Device*), et en particulier la famille FPGA (*Field Programmable Gate Array*). Un PLD peut être programmé pour mettre en application une fonction simple comme de la logique combinatoire, mais il peut également mettre en application de grandes conceptions telles qu'un système embarqué sur une seule puce. Un PLD de type FPGA est une architecture générique se composant des blocs configurables de logique, de blocs de mémoire programmable et d'un réseau d'intercommunications programmable. Plus récemment, certains FPGAs possèdent des fonctions pré-cablées comme des multiplieurs et/ou des processeurs spécialisés implantés en hard (PowerPC, ARM) au sein même des réseaux de portes logiques. Ces circuits permettent de spécialiser une architecture à moindre coût, par programme, sans passer par le processus de fonderie long et coûteux des circuits

ASIC. Les calculs étant réalisés de manière câblée et spécialisée, des gains importants en performance peuvent être obtenus par rapport à une exécution sur processeur générique. Ce type de circuit est de plus reconfigurable à volonté. Ils peuvent donc être dédiés à une application de manière temporaire puis reprogrammés pour une toute autre application (flexibilité). De plus, il existe des circuits qui sont reconfigurables dynamiquement au fil de l'exécution (en totalité ou partiellement). Mais l'inconvénient de ces phases de reconfiguration est la surconsommation d'énergie qu'elles engendrent.

Le coût d'achat unitaire d'un circuit logique programmable est plus important que celui d'un circuit ASIC. De plus, à cause de leur architecture prédéfinie, les circuits logiques programmables sont, pour une application donnée, moins performante que les circuits ASIC complètement spécialisés. Il est cependant facile à reprogrammer, ce qui raccourcit les cycles de conception et permet d'effectuer des essais réels. L'avantage principal des circuits logiques programmables réside donc dans la rapidité de mise au point qu'ils procurent. Si en plus il est reprogrammable, la flexibilité offerte pour la définition incrémentale et l'adaptation des fonctionnalités de la puce est très importante. Ces circuits permettent donc une mise au point rapide et évolutive. Cela permet une mise à disposition très rapide du système embarqué. C'est pourquoi les circuits logiques reprogrammables sont souvent utilisés pour le prototypage rapide de systèmes et aux petits volumes de production. A cause de leur coût plus élevé que celui des circuits ASIC, il est ensuite possible de produire un circuit complètement dédié, reprenant les fonctionnalités du circuit logique programmable, une fois l'application complètement définie et stabilisée [Trézéguet 98].

Bien que ces composants soient entièrement numériques, il commence à y avoir des composants analogiques programmables sur le marché. On peut par exemple citer les pSoC de Cypress [Cypress-web] qui intègrent dans un même circuit un microcontrôleur, une zone de logique programmable et une zone analogique. La zone analogique comporte des blocs contenant des composants analogiques élémentaires (capacités, capacités commutées, résistances, amplificateurs, etc.) pour construire des fonctionnalités analogiques ou mixtes.

4.1.5 Ressources logicielles

Un système embarqué est un système numérique qui utilise au minimum un processeur pour la gestion des échanges et pour exécuter un logiciel dédié à la réalisation d'une fonctionnalité précise. Par le terme ressources logicielles, nous entendons l'ensemble des éléments de la partie logicielle d'un SoC ainsi que le processeur nécessaire a son exécution.

1 - Logiciel embarqué

Un logiciel embarqué (*embedded software*) est un programme informatique enfoui dans un système électronique. La différence essentielle avec un logiciel classique tient à sa complète intégration dans le

système. Historiquement, cette notion est antérieure à l'idée actuel du logiciel : le programmateur mécanique du lave-linge ou de l'arrosage automatique fait partie des logiciels embarqués ou dédiés.
Un logiciel embarqué est caractérisé par les critères suivants :

- il doit être ciblé au domaine d'action de l'application ;
- il doit être fiable et sécurisé pour un fonctionnement complètement autonome ;
- il doit avoir une durée de vie quasi infinie ;
- il doit être spécifique à l'architecture cible ;
- il doit être optimisé (taille, temps d'exécution, etc.).

Les habitudes de programmation pour les ordinateurs portent à croire que la compilation et l'édition de liens sont les seuls traitements à apporter au logiciel pour pouvoir le faire fonctionner sur une architecture cible. Dans le cas des systèmes embarqués, une autre étape est nécessaire avant la compilation : c'est l'adaptation du logiciel à l'architecture spécifique du circuit cible [Gauthier 03]. Dans la littérature, elle est souvent nommée : ciblage logiciel.

Aujourd'hui, la terminologie "logiciel embarqué" désigne tous composants *software* enfouis dans un système embarqué. Ces composants s'étendent des programmes simples jusqu'aux versions complexes et multi-usages que sont les systèmes d'exploitation (*Operating System* ou OS). Les systèmes embarqués peuvent être divisés en deux familles : les systèmes sans contrainte ou à faible contrainte temporelle et les systèmes avec contrainte temps réel. Cette séparation est fondamentale car elle déterminera complètement le choix des composants logiciels à utiliser. Dans le premier cas, nous pouvons envisager l'utilisation soit de programmes simples et séquentiels (programmes de type microcontrôleur), soit d'un système d'exploitation dérivé d'un système commercial comme Linux [Ficheux 02]. L'adaptation se situera principalement au niveau du développement de pilotes de périphériques et de l'optimisation du système en taille ou en performances. Dans le second cas, les contraintes matérielles nécessiteront l'utilisation de composants spécialisés, soit un logiciel spécifique, soit un système d'exploitation dit temps réel (*Real Time Operating System*).

Malheureusement, du point de vue purement logiciel, on ne dépasse pas le stade de la super boucle, c'est-à-dire la boucle infinie en concurrence avec les interruptions générées par le système cible. Il est néanmoins possible de mettre en œuvre un noyau Temps Réel. L'usage d'un noyau temps réel améliore les choses du point de vue de la conception logiciel (aspect multitâche) surtout si l'on a en plus des contraintes temporelles à respecter.

La contrainte principale est le plus souvent l'empreinte mémoire, c'est-à-dire la taille mémoire occupée par le logiciel. Il faut cependant faire la différence entre l'empreinte mémoire du logiciel embarqué et celle des données.

2 - Les langages de programmation utilisés

Pour des raisons évidentes de contraintes matérielles, le langage assembleur a longtemps été le choix de prédilection car il permettait à la fois d'optimiser la taille du code généré mais aussi ses performances. La gestion d'un projet complexe écrit en assembleur est cependant difficile et l'évolution des performances du matériel et des compilateurs permet aujourd'hui de se tourner vers des solutions plus confortables comme les langages C et C++. Historiquement, le langage C est également un langage permettant une programmation relativement proche du matériel, donc bien adaptée au logiciel embarqué.

Le choix entre ces différents langages de programmation est souvent imposé par l'architecture matérielle (type de processeur, taille de la mémoire programme) et les fonctions que doit réaliser le logiciel embarqué. En effet, nous n'utiliserons pas le même langage pour réaliser la gestion d'entrées/sorties que pour une interface visuelle homme/machine. Le choix imposé par les fonctions revient donc au concepteur du logiciel. En ce qui concerne le choix imposé par l'architecture matérielle, nous pouvons le classer selon trois catégories correspondant aux trois types d'architecture : les architectures monoprocesseur, les architectures processeur central + coprocesseurs spécialisés et les architectures multiprocesseurs.

Pour une architecture monoprocesseur dont la partie logicielle est constituée d'un seul programme, le choix du langage assembleur permet d'obtenir un code efficace et de petite taille. L'architecture des systèmes embarqués composés d'un processeur central et de quelques processeurs annexes contrôlés par le processeur central a besoin d'une composante logicielle répartie sur tous les processeurs. Le logiciel du processeur central est souvent décrit dans un langage de haut niveau tel que le C et les logiciels des processeurs annexes sont souvent trop spécifiques pour être entièrement décrit dans un langage de haut niveau, l'utilisation de langages assembleur est nécessaire. Pour pouvoir supporter conjointement les besoins en puissance et en flexibilité, les architectures des systèmes embarqués comprennent de plus en plus de processeurs, qui peuvent chacun se comporter en maître (multiprocesseur). Cet aspect multiprocesseur apporte de nombreuses alternatives pour le choix du langage : il peut y avoir un seul langage pour tous les processeurs (solution difficilement applicable lorsque les processeurs sont hétérogènes), ou, à l'opposé, il peut y avoir un langage par processeur (solution qui peut être plus coûteuse).

3 - Les systèmes d'exploitation embarqués

Si nous revenons à la notion de système d'exploitation telle qu'elle est communément admise, nous pouvons définir un OS comme un ensemble de programmes permettant :

- de gérer les ressources matérielles en assurant leurs partages ;
- d'assurer un ensemble de services en présentant aux utilisateurs une interface mieux adaptée à leurs besoins que celle de la machine physique.

La première réaction est de considérer qu'un système d'exploitation est a priori beaucoup trop complexe et surdimensionné pour correspondre aux caractéristiques et remplir les tâches d'un logiciel embarqué. Cela est vrai dans certains cas de spécificité extrême du logiciel à embarquer ou de fortes contraintes matérielles. Mais l'amélioration, sans cesse croissante, des performances du matériel permet dans un grand nombre de cas d'utiliser un système d'exploitation adapté au lieu d'un simple logiciel dédié [Kadionik 05].

L'utilisation d'un système d'exploitation comme "logiciel embarqué" présente de nombreux avantages. Il permet au développeur de l'applicatif embarqué de s'affranchir du travail d'adaptation très proche du matériel, ce qui diminue le temps de développement et donc les coûts. Si le système d'exploitation utilisé est suffisamment répandu, il permet aux applications industrielles et embarquées de bénéficier des mêmes avancées technologiques que les ordinateurs grand public. C'est ainsi qu'il est aujourd'hui possible d'utiliser dans des systèmes embarqués des protocoles de communication tels que TCP/IP, http, etc. De plus, les systèmes d'exploitation fournissent en général un environnement de développement facilitant la mise au point des programmes dans un contexte beaucoup plus accueillant et performant. En contrepartie, l'utilisation d'un véritable système d'exploitation pose le problème de la taille mémoire qui lui est nécessaire ; et c'est là son principal inconvénient. Il est bien évident que, si l'espace disponible est réduit à quelques centaines d'octets pour accomplir une tâche rudimentaire, un système d'exploitation ne s'imposera pas.

4.2 - Notre approche : la conception SoPC

La capacité de conception de systèmes numériques permet aujourd'hui de tout intégrer dans un même composant. On travaille donc au niveau système et non plus au niveau porte élémentaire ou schématique. Si les fonctionnalités du système embarqué sont implantées dans des composants spécifiques de type ASIC, on parle alors de Systèmes sur puce ou SoC. Si ces mêmes fonctionnalités sont implantées dans des composants logiques programmables de type FPGA, on parle alors de systèmes SoPC.

Notre travail porte sur la conception d'une architecture modulaire dédiée au traitement d'images en temps réel. Pour cela, l'architecture doit être la plus flexible possible. Comme nous l'avons vu précédemment (paragraphe 4.1.4 : Support physique pour les systèmes monopuces) seul les circuits programmables répondent à cette caractéristique.

4.2.1 Définition

Le terme SoPC, acronyme pour *System on a Programmable Chip*, est un système embarqué possédant une architecture dont les ressources (calcul, interconnexions, etc.) peuvent être modifiées

par programmation pour s'adapter au traitement à réaliser. Cette technologie correspond à l'intégration des ressources logicielles et matérielles d'un SoC sur une même puce de type circuit logique programmable.

Les circuits logiques programmables sont de natures variées. Le choix de la famille FPGA (*Field Progammable Gate Array*) à base de LUT (*Look Up Table*) est principalement justifié par la capacité d'intégration élevée de ce type de composants et la facilité de programmation qu'ils autorisent. Les FPGAs ouvrent une nouvelle voie dans la conception des systèmes embarqués temps réel. Un des points forts de ces circuits concerne leur reprogrammabilité infinie, qui confère au concepteur de systèmes une aisance comparable aux machines à processeurs. Ces composants permettent le développement d'applications simples jusqu'aux plus complexes tels que le traitement d'image.

4.2.2 Architecture interne d'un SoPC

L'architecture matérielle d'un système est caractérisée par une structure et une organisation. La structure est une représentation schématique et modulaire des composantes du système. Elle illustre aussi les interconnexions de ces modules et la direction des échanges d'informations entre ces derniers. L'organisation caractérise les relations de contrôle existantes entre les différents modules. Ainsi, un composant peut soit travailler en totale autonomie, soit opérer sous le contrôle d'un autre composant, alors appelé maître. Ce composant (ou esclave) peut ainsi offrir l'exécution de services aux maîtres. Cette section contient une étude des différents composants matériels d'une architecture multiprocesseur.

1 - Les composants de calculs

Un composant de calcul est un composant matériel permettant l'implémentation plus ou moins flexible d'un algorithme de contrôle et/ou de traitement de données. On distingue essentiellement deux sous-familles de composants de calculs : les processeurs matériels et les processeurs permettant l'exécution de logiciel. Ces composants de calcul peuvent être génériques ou dédiés à une fonctionnalité donnée. Leur utilisation permet l'exécution localisée d'une composante fonctionnelle. Leurs performances en terme de vitesse de traitement, leur coût en surface et leur consommation d'énergie sont des critères pouvant être considérés comme objectif de la conception ou au contraire comme contraintes.

Processeurs matériels Il s'agit de l'implémentation matérielle, encore appelée "logique câblée" d'un algorithme. Concrètement, un réseau de portes logiques (combinatoires et séquentielles) est utilisé pour implémenter de façon spécifique un comportement donné. Leur structure interne exhibe généralement la présence d'une partie contrôle composée d'une (rarement plusieurs) unité de contrôle réalisée par des machines d'états de Moore ou FSM (*Finite State Machine*), et une ou plusieurs unités de traitement constituant la partie opérative. Ces processeurs sont utilisés lorsque :

- l'algorithme à implémenter est définitivement arrêté (fonctionnalités, normes et standards fixés) ;
- les performances requises (puissance et vitesse de calcul, consommation, surface) pour l'exécution de cet algorithme ne peuvent être atteintes que par l'utilisation de ressources dédiées.

Ces implantations spécifiques apportent des réalisations performantes et sobres tant en consommation énergétique qu'en occupation surfacique. Les performances en vitesse et la surface requise sont estimables après la synthèse en portes logiques.

Ces composants sont très peu flexibles. Leurs fonctionnalités ne peuvent être changées de façon significative, seule une restriction de leur usage selon une configuration donnée peut être opérée. Leurs interfaces sont elles aussi souvent spécifiques. Il est donc nécessaire d'adapter ces interfaces lorsque le processeur doit être connecté à un réseau de communication qui lui est incompatible.

Processeurs de logiciel Il s'agit d'un cas particulier de processeurs matériels exécutant un algorithme d'interprétation d'un jeu d'instruction. Ces composants se décomposent en deux parties : la partie opérative elle-même composée d'une ou plusieurs unités fonctionnelles de complexité variable ; la partie contrôle en charge de séquencer les étapes de l'algorithme et de configurer la partie opérative. L'exécution de ces étapes est séquentielle car les ressources du processeur sont limitées.

Lorsque la vitesse d'exécution et la consommation énergétique requises ne nécessitent pas l'utilisation et la conception d'un processeur matériel spécifique, les processeurs de logiciel sont largement utilisés voire réutilisés. Leur usage peut aussi être dicté par la nécessité de flexibilité. Il s'agit de pouvoir changer les fonctionnalités du système afin de suivre les évolutions de standards et de normes en cours de développement. Mais aussi de pouvoir réutiliser l'ensemble d'une architecture matérielle en reportant au sein de ses processeurs de logiciels les adaptations nécessaires pour supporter de nouvelles applications. Ce type de processeur embarqué peut allier la souplesse du logiciel à l'accélération du temps d'exécution du matériel. Une fonctionnalité particulière peut être composée d'une partie logicielle couplée à une fonctionnalité matérielle dédiée : on a donc une conception conjointe matérielle/logicielle ou *Co-design*. Les processeurs logiciels sont réputés lents (moins performants) au regard de leurs contreparties matérielles.

Il est important de noter que le processeur de logiciel pour un SoPC peut être :

- soit implanté sous forme de composant VHDL : on parle de processeur *softcore* ;
- soit déjà implanté de manière fixe dans le FPGA : on parle de processeur *hardcore*.

Le choix entre ces deux possibilités se fait selon des critères :

- de performances au détriment de la flexibilité pour le processeur *hardcore* ;

- de flexibilité de mise à jour au détriment de performances moindres pour le processeur *softcore*.

Généralement, on privilégie les processeurs *softcore* pour s'affranchir des problèmes d'obsolescence et pour pouvoir bénéficier facilement des évolutions. Ce sont généralement des processeurs 16 ou 32 bits ayant une architecture de type Harvard avec un jeu d'instructions réduit RISC. Un processeur *softcore* est généralement lié à un fabriquant de circuit FPGA (comme Altera ou Xilinx) et ne peut être utilisé dans d'autres FPGA que celui pour lequel il est prévu. Soit il peut être libre : il est alors décrit en langage de description de matériel dont le code source peut être librement distribué et implanté dans n'importe quel circuit programmable FPGA. Nous pouvons cité les processeurs propriétaires NIOS et NIOS II d'Altera [Altera-web] et Microblaze de Xilinx [Xilinx-web]. Pour les processeurs libres, nous trouvons principalement le processeur Leon [Gaisler-web], le processeur OpenRisc [Opencores-web], le processeur F-CPU [f-cpu-web], etc. Pour les processeurs *hardcore*, leur type dépend des fabricants de CLP et de la famille du FPGA utilisé. Actuellement, les processeurs les plus répandus sont les PowerPC et les ARM.

La surface de silicium requis par un processeur de logiciel n'est pas en relation directe avec la complexité ou le volume de l'algorithme applicatif à exécuter. La surface occupée par le processeur peut donc être considérée comme constante, mais celles des mémoires en charge de contenir les instructions modélisant l'algorithme, ainsi que les données à traiter sont quant à elles, proportionnelles à la complexité algorithmique. C'est généralement le facteur limitant leur utilisation dans les SoCs à base de FPGA, car il y a peu de mémoire embarqué et que l'utilisation de mémoire externe entraîne une diminution des performances du processeur de logiciel. Cette limite tend à disparaître avec les dernières générations de FPGA qui sont dédiés pour l'utilisation de tels processeurs.

La consommation énergétique est proportionnelle à l'activité des processeurs. Or, un processeur de logiciel étant sensiblement lent, les concepteurs tendent à utiliser des fréquences d'horloges élevées, voire très élevées, augmentant ainsi considérablement la consommation. Pour limiter cette consommation, quelques processeurs de logiciel sont spécifiquement conçus pour les applications embarquées qui par définition ne disposent pas d'une source d'alimentation énergétique pérenne. La pratique veut qu'un processeur de logiciel soit énergétiquement plus gourmand que son homologue spécifique.

2 - Les composants de mémorisation

Les composants de mémorisation ou mémoires sont des ressources matérielles permettant de stocker pour des durées et des usages variables, les données à traiter, les résultats temporaires, ainsi que des informations de configuration telles que des programmes pour les processeurs.

Deux types de mémoire peuvent être associés à un FPGA : les mémoires "mortes" ou *Read-Only Memory* (ROM) pour les informations immuables et les mémoires "vives" ou *Random Access*

Memory (RAM) pour les informations variables. Ces mémoires présentent différents modes d'accès (séquentiel ou aléatoire) et caractéristiques de réponse à un accès (synchrone ou asynchrone). Les accès "séquentiels" sont caractérisés par la connaissance, à tout moment de l'emplacement du prochain accès tels que les *First-In First-Out memory* (FIFO). Quant aux accès "aléatoires", la mémoire ne peut pas anticiper l'emplacement du prochain accès et doit donc en être informée par une information supplémentaire : l'adresse. Si la mémoire répond aux fronts du signal d'horloge la séquencant, on parle alors d'accès synchrone. Sinon, si la mémoire répond au terme d'un délai fixe ou variable, elle sera désignée par le terme asynchrone.

Les performances en vitesses des mémoires se résument à la bande passante (ou débit d'information) qu'elles peuvent supporter. Puisque les horloges des composants de calcul et de communication sont de plus en plus rapides, il est de plus en plus courant d'imposer à ceux-ci des cycles d'attentes afin que les mémoires puissent répondre aux requêtes qui leur sont adressées. La métrique "vitesses des mémoires" dépend de trois facteurs :

- le temps d'accès en lecture/écriture ;
- la largeur du chemin de données ;
- des mécanismes internes (rafraîchissement, pagination) pouvant la rendre indisponible durant quelques cycles d'horloges.

Pour les mémoires comme pour tous les autres composants, l'activité (ou basculement) des transistors est consommatrice d'énergie. Ceux-ci sont actifs lors des accès et des éventuelles sessions de rafraîchissement. La consommation est principalement due à ces accès. La surface de silicium nécessitée par les mémoires représente le plus grand taux d'occupation surfacique pour les applications monopuces.

Pour un SoPC, ces composants mémoires peuvent être placés au sein du FPGA ou situés physiquement dans des circuits mémoires externes au FPGA. Ce choix est imposé au concepteur par à la fois les contraintes de l'application et la taille des mémoires disponibles à l'intérieur du circuit FPGA. Pour notre application, mesures par traitement d'images, nos besoins en mémoire sont proportionnels à la quantité d'information à traiter (images hautes définitions). Actuellement, les dernières générations de circuits FPGA possèdent une quantité de mémoire non négligeable (de l'ordre de 2 à 9 Mbits). Mais, son utilisation pose quelques inconvénients et limitations. L'inconvénient majeur est sa répartition dans l'architecture interne du FPGA. Le plus souvent, cette mémoire globale est découpée en unités de taille fixe en nombre plus ou moins important disposées entre les cellules d'éléments logiques dans des zones prédéfinies. Ce découpage impose une gestion lourde et complexe des communications entre elles et avec les autres composants. Les limitations de cette mémoire interne sont les possibilités restreintes de définition de composants mémoires (bus, contrôles, taille), son utilisation prioritaire et par défaut pour les processeurs de logiciel et la difficulté d'optimiser leur fonctionnement.

3 - Les composants de communication

Les liens reliant les composants de calcul ou de mémorisation ont eux aussi une variété de concrétisations physiques. Ces composants permettent l'implantation des liens logiques de communication reliant les composants de calcul ou de mémorisation. Ils sont composés de ressources de routage et de transport de l'information, mais aussi d'unité de décision de l'attribution de celles-ci lorsque leur usage est partagée par l'implantation de plusieurs liens logiques concurrents. Leurs utilisations peuvent être :

- l'échange de données entre interlocuteurs (composants de calcul et de mémorisation). On associe les termes chemin de données ou *data-path* à cette utilisation performante, mais très déterministe des composants de communication ;
- la synchronisation et la configuration/contrôle entre deux composants de calcul distants et concurrents. On parle alors d'utilisation non déterministe et fortement orientée contrôle de l'application.

Les performances d'échange de l'information dépendent de la topologie de ces composants et de la bande passante de chacun des sous composants impliqués dans la communication, du nombre de liens logiques se partageant les mêmes ressources, mais aussi du protocole d'échange. L'occupation surfacique de ces composants doit rester faible face à l'occupation de la logique câblée implantant les composants de calcul. Il faut donc prendre garde à ce que le choix d'une structure de ces composants ne vienne pas renverser cette relation d'ordre. Les fonctionnalités de ces composants sont très basiques, mais leur dimensionnement qui influe directement sur les performances de l'ensemble de l'application doit être soigné. La flexibilité d'un composant de communication peut être considérée sous deux angles :

- l'extensibilité ou l'aptitude du composant à supporter un nombre donné d'utilisateurs concurrents sans altérer ni ses fonctionnalités, ni ses performances globales (en anglais *scalability*) ;
- la compatibilité aux schémas de communication utilisés par l'application : point à point, multipoint, transferts en rafale, transferts non préemptifs (nécessaire pour les synchronisations par sémaphores).

Il existe de nombreux composants de communication. Nous exposerons que ceux qui sont les plus utilisés dans la conception d'architecture multiprocesseur dans les FPGAs. Ces modèles de communication sont la connexion point à point (de largeur variable), le bus partagé (extension de la connexion point à point) et les réseaux de communication (Figure 4-5).

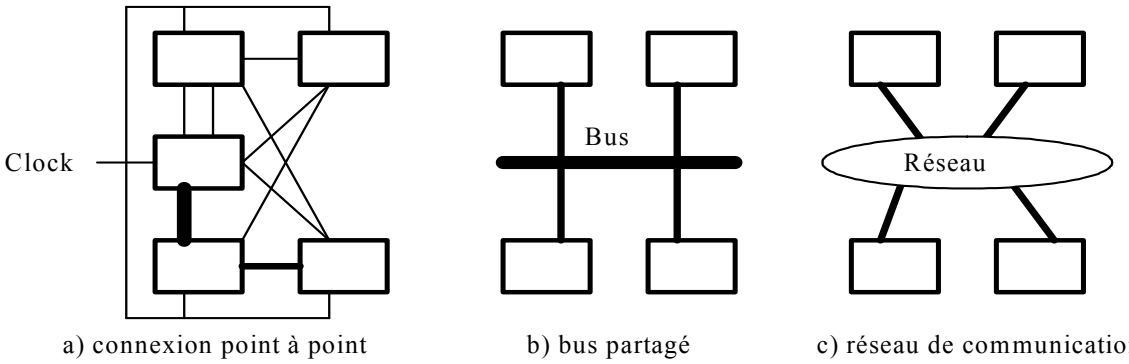


Figure 4-5 : Modèles de communication.

Les connexions point à point Il s'agit de l'utilisation de ressources de transmission spécifiques et dédiées à l'implémentation d'un lien entre deux composants. Le principal usage de ce schéma de communication prend place dans l'implémentation des communications lorsque l'application est orientée traitement de données et nécessite des latences faibles et des débits importants qu'une ressource partagée ne peut offrir. Les architectures cascadées (pipeline) en sont très friandes. La ressource de transmission étant dédiée à une unique communication, elle est toujours disponible pour cette dernière et peut ainsi lui offrir sa pleine bande passante. De plus, du fait de la simplicité du contrôle de telles connexions, les débits y sont élevés et les latences quasi-inexistantes.

Du fait de leur usage pour l'implémentation d'une unique communication, la surface de ces connexions est proportionnelle à leur nombre et la complexité des unités de contrôle est très réduite. De ce fait, ces connexions sont non extensibles et spécifiques à un protocole.

Bus partagé Il s'agit d'implanter plusieurs liens logiques de communication par un unique médium de communication à accès partagés : le bus. Ce médium permet la mise en place de connexions point à point ou multipoints entre un maître (unité initiant la communication) et un ou plusieurs esclaves. Le bus est alloué dynamiquement à une communication, plus précisément à un maître, en fonction d'une politique d'ordonnancement. Les échanges peuvent être cadencés par :

- une horloge globale (système) : bus globalement synchrone ;
- une horloge spécifique au bus : bus localement synchrone et globalement asynchrone ;
- par l'état du maître et des esclaves : bus asynchrone.

Les deux dernières catégories permettent de faire communiquer des éléments de l'architecture ayant des horloges indépendantes, alors que la première nécessite l'utilisation de la même horloge pour tous les composants connectés au bus. La structure de tels composants peut encore se décomposer en une partie contrôle et une partie chemin de donnée. Le chemin de donnée est constitué de ressources de transport généralement implémentées par des pistes ; plusieurs fonctionnalités incombent à la partie contrôle des composants de communication : l'attribution du chemin de donnée, la synchronisation des extrémités, le routage de l'information.

Ces bus sont généralement utilisés pour des systèmes ne nécessitant pas de large bande passante entre plusieurs composants de calculs. Ils sont donc employés dans les applications orientées contrôle, pour la configuration de composants de calcul (*data-path*).

Les vitesses de fonctionnement de ces bus en font des unités de communication très performantes. Cependant, cette caractéristique doit être revue à la baisse en fonction de l'activité et du nombre des communications se partageant cette unité. En effet, plus un bus est partagé plus la latence moyenne d'attribution est élevée. Lorsqu'ils ne sont pas disponibles pour une communication, on les dit "bloquants". Ils gèlent ainsi l'exécution de parties de l'application. En tant que goulot d'étranglement des performances globales, leur propre performance devient donc un critère limitant et doit en tant que telle, être soignée. Du fait du nombre restreint de ressources, les bus partagés consomment peu d'énergie. Cependant, ils peuvent pousser les composants qu'ils relient à consommer plus d'énergie. En effet, le partage de ressources implique inévitablement des cycles d'attente d'attribution durant lesquels de l'énergie est inutilement gaspillée. L'encombrement de cette solution est la plus réduit. Le partage intensif d'un nombre restreint de ressources offre le meilleur rapport coût/nombre de communications. La conception de ces composants doit sa complexité à la recherche de meilleures performances. Leur réutilisation se limite à configurer statiquement ces composants pour obtenir une bande passante et l'ordonnancement des communications adéquats. Ces composants de communication sont : non extensible, une augmentation du nombre de composants de calcul connectés réduit drastiquement les performances globales des communications; compatible uniquement avec une famille ou un jeu restreint de protocole.

Les réseaux de communication Les réseaux de communication se différencient des bus partagés par :

- leur nombre de ressources de transport et de routage supérieur à l'unité,
- la distribution du routage sur l'ensemble du réseau,
- la possibilité d'intégrer des services.

Ils prennent place dans la réalisation d'applications "temps réel" et "traitement du signal" nécessitant une grande bande passante, et une faible latence. Celles-ci mettant en oeuvre des calculs massivement parallèles, leurs nombreuses communications à haut débit ne peuvent être réalisées que par un réseau.

L'utilisation d'un nombre élevé de ressources de transport et de routage permet d'offrir de manière quasi-permanente la disponibilité des unités de transports. Les solutions à base de réseaux de communication offrent donc les meilleurs débits et les latences les plus basses. Le nombre de ressources utilisées augmente en proportion la consommation du réseau. De plus, les unités de contrôle et de routage essaimées sur toute la surface du réseau consomment inutilement lorsqu'ils ne sont pas sollicités. L'augmentation des ressources se traduit bien évidemment par un accroissement de

l'occupation surfacique. Mais si la surface nécessitée par les unités de contrôle et de routage est proportionnelle à leur nombre, il en va différemment pour les ressources de transport. Du fait de la complexité de leur structure et de leurs algorithmes de routage, les réseaux de communication sont de véritables challenges relevés par les concepteurs. Leurs caractères génériques et configurables nécessaires pour leur support d'un nombre variable de connexions sont pour l'essentiel des causes de cette complexité. Cependant, leur réutilisation est aisée par la philosophie de généricité qui a guidé leur conception. Le nombre de connexions supportées par ces réseaux est rendu facilement extensible par leur régularité structurelle.

4.2.3 Méthodologie de conception d'un SoPC

La complexité des systèmes embarqués pose le problème du choix de l'approche de conception rigoureuse parmi l'ensemble des méthodologies possibles.

Le *Co-design* fait partie intégrante de la méthodologie de conception d'un SoPC car les niveaux d'intégration sur silicium très importants des CLPs permettent embarquer des processeurs de logiciel performants dans d'architectures orientées multiprocesseur. Maintenant, la difficulté est de savoir comment implémenter une fonctionnalité dans un SoPC : sous forme *hardware* ou *software* ?

Le *Co-design* permet de concevoir en même temps à la fois le matériel et le logiciel pour une fonctionnalité à implémenter. De plus, il permet de repousser le plus loin possible dans la conception du système les choix matériels à faire contrairement à l'approche classique où les choix matériels sont faits en premier.

Pour la conception de SoPC, l'approche "schématique" au niveau portes logiques ou fonctionnalités de base RTL (*Register Transfer Logic*) est délaissée au profit d'une approche "textuelle". Pour cela, on utilise des langages de description de matériel comme le VHDL (*Very high speed integrated circuit Hardware Description Language*) ou le Verilog pour synthétiser une fonctionnalité numérique. Ils sont utilisés conjointement avec un synthétiseur et un simulateur. On utilise le terme synthétiseur au lieu de compilateur, car on génère au final dans la majorité des cas un fichier de programmation et non un exécutable. On synthétise un circuit numérique mais on ne le compile pas, bien que l'on ait une approche logicielle pour concevoir du matériel. Ces langages permettent de travailler avec un niveau d'abstraction plus grand laissant les basses besognes au synthétiseur. Ces langages de description de matériel sont aussi intéressants pour la facilité de modification et de réutilisation qu'ils offrent (*design reuse*). Ils permettent aussi une meilleure validation/vérification au cours de la conception grâce à leur possibilité de simulation (fonctionnelle et temporelle). Et ainsi, ils permettent de réduire le *Time To Market*.

Grâce à l'utilisation de ces langages, on a pu rapidement développer des bibliothèques de fonctionnalités que l'on appelle blocs IP (*Intellectual Property*). On peut ainsi voir la conception

SoPC comme un assemblage de blocs IP si bien que les langages de description de matériel sont un peu comme un langage assembleur vis-à-vis d'un langage plus évolué comme le langage C. L'architecture d'un SoPC peut être décomposée en sous-systèmes (modules, blocs IP) suivant une hiérarchie logique sur différents niveaux. Pour réaliser cela, une méthodologie dans la phase de conception de l'architecture du SoPC doit être utilisée. Il en existe deux :

- la méthodologie *top-down* part du niveau le plus abstrait (flux de données) pour aller à l'entité matérielle ou logicielle élémentaire.
- la méthodologie *bottom-up* part de l'entité matérielle ou logicielle élémentaire à implanter pour aller au système complet.

En conclusion, pour concevoir un SoPC, nous utiliserons une méthodologie de description *top-down* pour décrire notre système sous forme d'une hiérarchie de blocs IP (en VHDL) pour permettre la réutilisation de ces composants pour d'autres applications.

4.2.4 Description des circuits programmables

Nous décrivons dans ce paragraphe les différents composants programmables existants, afin d'en montrer la diversité, et d'expliciter l'évolution de ces technologies. Les capacités d'intégration, de plus en plus poussées au fur et à mesure de l'apparition des technologies et de l'évolution des architectures programmables permettent aujourd'hui d'intégrer des centaines de millions de portes logiques sur un même FPGA de type SRAM, reconfigurable à volonté. Les applications des différents composants ont également évolué avec la capacité d'intégration. Les circuits qui autrefois permettaient de remplacer quelques composants TTL, peuvent aujourd'hui émuler les processeurs généralistes des années 80.

Un circuit logique programmable ou PLD (*Programmable Logic Device*) regroupe, sur une même puce, des composants logiques configurables et des bascules. Des interconnexions programmables relient les ressources logiques entre elles. Les circuits n'ont pas de fonctionnalité prédéfinie, mais sont spécialisables suivant l'application visée. Des cellules de configuration mémorisent les fonctions réalisées par la logique et la manière dont celles-ci sont interconnectées pour fournir les résultats demandés. Les cellules de configuration et les interconnexions programmables sont formées à partir de points de programmation, élément le plus fin des circuits logiques programmables.

Ces circuits sont principalement caractérisés par la capacité d'intégration qu'ils proposent, la technologie utilisée, leur mode de programmation (outil externe, configuration à la volée), le mode de codage des fonctions, et les possibilités de reconfiguration (une ou plusieurs fois, reconfiguration

partielle ou non). Le diagramme de la Figure 4-6 propose une classification sommaire des circuits en fonction de ces différents paramètres.

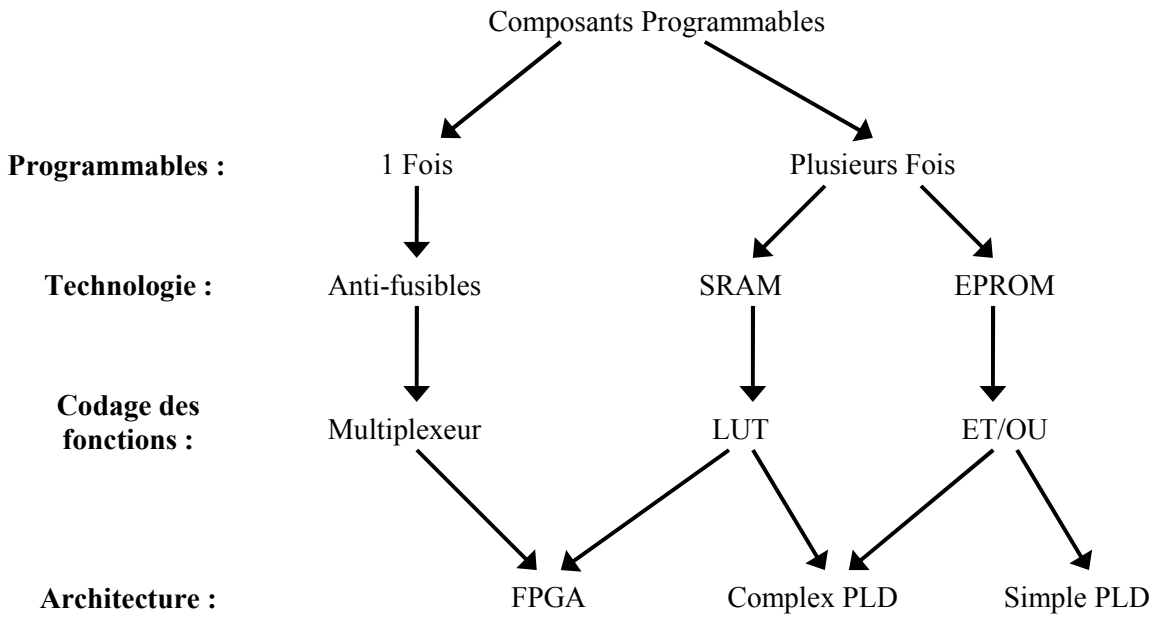


Figure 4-6 : Une classification des composants programmables.

Suivant la **technologie** utilisée pour réaliser les points de programmation, le circuit est **programmable** une fois (fusibles, anti-fusibles), reprogrammable à l'aide d'un outil externe (certaines EPROM, EEPROM, FLASH) ou reprogrammé à chaque mise sous tension (mémoire volatiles SRAM). Les mémoires volatiles sont par essence programmables sur site. Certains circuits à base de mémoires non-volatiles EEPROM ou FLASH incluent la circuiterie nécessaire pour une reprogrammation sur site.

Les modes de **codage des fonctions** représentent la manière dont les points de programmation sont agencés dans le circuit. Les points de programmation peuvent être regroupés de plusieurs manières pour constituer un élément de calcul programmable : en commandant un multiplexeur, en réalisant une mémoire utilisée comme table de vérité (*Look Up Table* ou LUT) ou assemblées en grille 2D pour réaliser des sommes de monômes de type ET/OU. Le mode de codage des fonctions est étroitement associé à l'architecture des circuits, par la suite seul le mode pour le FPGA sera détaillé.

L'**architecture** d'un circuit logique programmable est sa décomposition en éléments de base ou cellules logiques. La logique configurable est partitionnée en macro-cellules. Une macro-cellule implante une fonction logique combinatoire à une sortie (fonction booléenne), et peut éventuellement en stocker le résultat dans une bascule.

Trois grandes classes de circuits logiques programmables cohabitent aujourd'hui. Les circuits SPLD (*Simple* PLD) contiennent une ou deux dizaines de macro-cellules. Les premiers circuits

configurables PAL (*Programmable Array Logic*), apparus vers 1975, appartiennent à cette catégorie. Les circuits CPLD (*Complex* PLD) sont conceptuellement des regroupements de 2 à 32 SPLD interconnectés par une matrice d'aiguillage programmable de type *crossbar*. Enfin, les circuits FPGA sont composés d'un grand nombre de petites macro-cellules disposées en pavage et reliées par un réseau d'interconnexion régulier, souvent complété de ressources de routage supplémentaires spécifiques. L'évolution de l'architecture des circuits est due aux capacités d'intégration sans cesse croissantes du fait des progrès dans la technologie sub-micronique.

1 – Les circuits FPGA

Les circuits FPGA (*Field Programmable Gate Array*), inventés en 1985 (par la société Xilinx), sont les plus populaires actuellement. Cela est en partie dû à leur grande capacité d'intégration, inégalée dans le domaine des circuits programmables sur site. Les circuits FPGA reprennent l'idée des macro-cellules en présentant des cellules logiques programmables équivalent à des petits PAL complets. Mais ils intègrent également l'idée de tableau de cellules présentes dans les circuits ASIC prédiffusés (un FPGA utilisant la technologie anti-fusibles peut être vu comme une mer de portes) ou précaractérisées (chaque cellule d'un FPGA utilisant la technologie SRAM peut être vu comme une cellule standard de taille moyenne) (Figure 4-7).

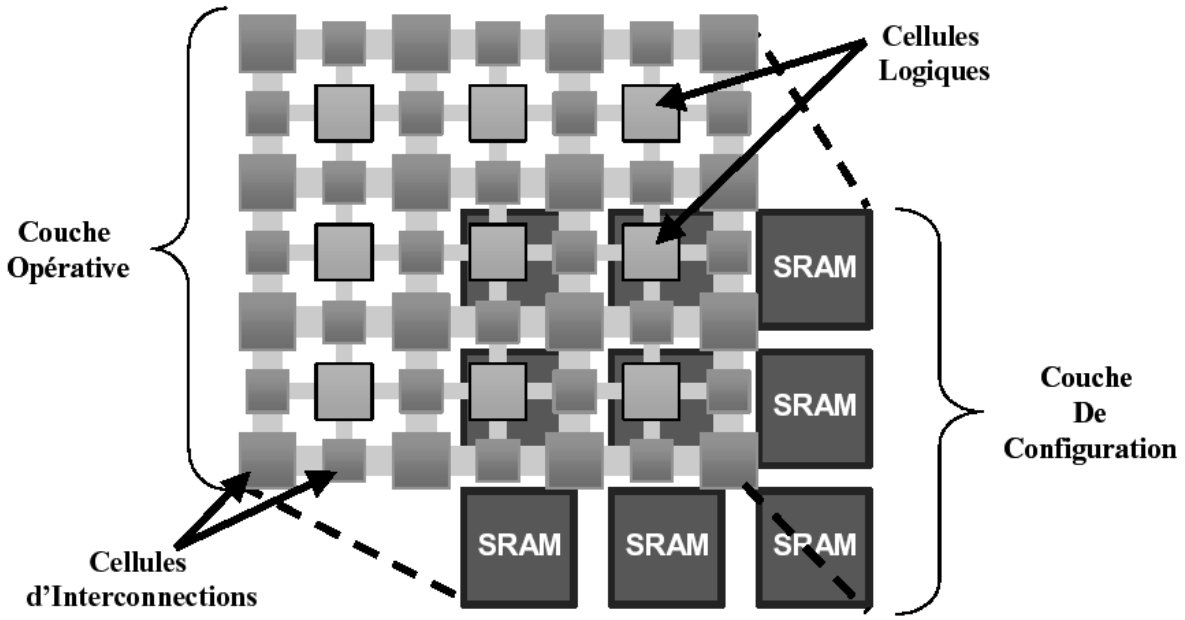


Figure 4-7 : Architecture de principe d'un FPGA SRAM.

Ces FPGA peuvent être programmés avec des outils informatiques, et évitent donc le processus long et coûteux de fonderie des circuits ASIC. Les circuits à anti-fusibles offrent de meilleures performances en délais que les circuits SRAM, à cause de la finesse de leur point de programmation. Par contre, ils ont l'inconvénient de ne pas être reprogrammables et de ne pas bénéficier, comme c'est le cas des circuits SRAM, des progrès réalisés dans le domaine de

l'intégration des mémoires volatiles. Nous nous intéressons donc ici uniquement aux circuits utilisant la technologie SRAM.

Les FPGAs SRAM sont composés de cellules logiques de type PAL, disposées généralement en matrice 2D, et interconnectées de manière plus ou moins régulière par des ressources de routage elles-mêmes programmables. Des blocs d'entrée/sortie, disposés à la périphérie du circuit, fournissent l'interface avec l'extérieur (Figure 4-8). Il existe deux types de FPGA dont l'architecture diffère dans la structure des blocs logiques et des interconnexions associées : le FPGA hiérarchique et le FPGA symétrique.

Le FPGA hiérarchique est composé de blocs logiques qui regroupent eux même une multitude d'autres blocs logiques plus élémentaires. Une zone centrale est réservée aux interconnexions entre les divers blocs élémentaires. Le FPGA symétrique a une structure totalement différente. Cette structure peut être représentée sous forme d'un tableau à deux dimensions : un tableau de blocs logiques où chaque bloc logique est encadré d'interconnexions. Suivant la catégorie de FPGA qui est choisie, l'intégration de la fonctionnalité obtenue est totalement différente, car d'une part pour le FPGA hiérarchique la concentration de blocs logiques est importante mais avec de grandes interconnexions, et d'autre part, pour le FPGA symétrique, la description est un éparpillement des blocs avec des interconnexions moins longues que la structure hiérarchique.

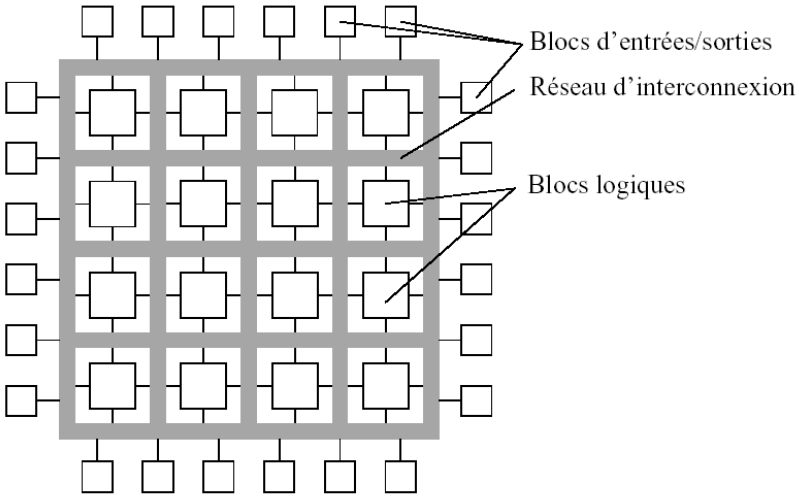


Figure 4-8 : Architecture conceptuelle des circuits FPGA.

La structure de routage est le plus souvent de type Manhattan dans les circuits à base de SRAM. Des connexions locales rapides entre cellules adjacentes sont disponibles ; des connexions plus globales, de longueur variable, avec des matrices d'interconnexion programmables permettant la diffusion de signaux au travers de canaux de routage intercalés entre les lignes et colonnes de cellules logiques. Des matrices d'interconnexion (*switching box*) complètement ou partiellement

programmables sont situées à l'intersection de chaque ligne et colonne. Leur programmation permet de définir les connexions entre les différentes ressources de routage adjacentes à l'intersection.

Les architectures à grain fin sont caractérisées par des cellules de très petite taille (LUT à deux entrées ou multiplexeur 4:1, une bascule) alors que les architectures à gros grain ont une cellule de base composée d'une ou deux LUT à 4-5 entrées et de une ou deux bascules. Une architecture à grain fin maximise le taux d'occupation des cellules mais en augmente le nombre. Cela nécessite des facilités de routage améliorées. A l'opposé, une architecture à gros grain autorise une intégration plus poussée de la logique dans les blocs.

Les capacités des circuits SRAM sont mesurées à la fois en nombre de portes logiques, et par le nombre de bits de configuration. Les équivalences en nombre de portes logiques avancées par les fournisseurs donnent une idée de la capacité d'intégration logique, mais sans prendre en compte la flexibilité de l'architecture de routage. Par exemple, pour des capacités annoncées de même ordre de grandeur, la taille de la mémoire de configuration peut varier énormément, ce qui reflète des différences notables entre les architectures en matière de ressources de routage spécifiques. Pour les circuits SRAM, il peut être intéressant de mesurer la capacité des circuits en nombre de cellules. Une mesure en terme de cellules logiques peut être considérée pour la prise en compte à la fois des ressources logiques et des ressources de stockage des circuits.

2 – Méthodologie de synthèse

Les techniques de conception mises au point reprennent les idées développées pour la réalisation de projets logiciels complexes : généricité du code, portabilité, découpage en module hiérarchisés, appel à des bibliothèques prédéfinies et réutilisables, etc. Le processus de génération de configuration s'apparente à une compilation (plus communément appelé synthèse). Elle se fait à partir d'un langage de spécification de haut niveau, masquant en partie les spécificités de l'architecture cible. Des descriptions de plus bas niveau sont générées pour définir plus finement le circuit, en prenant en compte les spécificités de l'architecture programmable cible, de manière analogue au processus de compilation de logiciel vers un processeur généraliste. Les méthodologies de conception sont descendantes (*Top-down*) et analogues à celles utilisées pour les circuits ASIC.

Le processus de synthèse VLSI peut être vu comme une série de transformations appliquées à des descriptions comportementales, structurelles puis physiques d'un circuit. Les descriptions comportementales décrivent les sorties du circuit en fonction de ses entrées. Les descriptions structurelles, elles, appréhendent le circuit en termes de composants et d'interconnexions. Enfin les représentations physiques sont caractérisées par des informations utilisables pour la fabrication d'un circuit intégré. Ces informations sont principalement de nature géométrique (dessin de masques).

Le cas de la synthèse FPGA est un peu différent de celui des circuits ASIC. Cela est dû à la spécificité de ce type de composants, fortement contraint par leur architecture prédéfinie. On retrouve, comme dans le cas général de la synthèse ASIC, les phases de haut niveau pour la spécification et la

représentation comportementale du circuit (qui sont indépendantes de la technologie), mais les étapes basses de génération de circuit et de dessin de masques sont remplacées. Le processus de synthèse utilise une méthodologie de conception descendante, divisée en phases. Le point d'entrée du processus de synthèse est une spécification comportementale du circuit, généralement concrétisée par une description à l'aide d'un langage de description matériel de haut niveau (par exemple VHDL) de l'application, de ses performances, de son interface, etc. A partir de cette spécification, les phases suivantes sont réalisées :

- La génération fonctionnelle, durant laquelle une description fonctionnelle est synthétisée. Le résultat est une description comportementale du circuit composée de directives de transfert de registres (RTL) et d'équations booléennes pour la partie combinatoire du circuit. Les outils de simulation comportementale permettent d'analyser le résultat de la synthèse et de le comparer à la spécification initiale.
- Durant la génération logique, une structure logique, implantant la description fonctionnelle, est produite. Le résultat est typiquement une représentation structurelle consistant en une liste de portes logiques interconnectées. La vérification symbolique de la logique et/ou simulation à l'aide de vecteurs de tests permet de valider le circuit obtenu par comparaison avec les résultats de la simulation comportementale.
- Conversion technologique : la génération d'une liste technologique correspond en fait soit à l'utilisation de portes prédéfinies extraites d'une bibliothèque, soit au découpage du circuit en fonctions logiques pouvant être implantées dans les cellules de base de l'architecture cible.
- Le placement de la liste technologique, c'est-à-dire, le choix d'un emplacement pour chaque composant sur la grille de cellules de base constituant le FPGA.
- Le routage des signaux entre les composants placés, reprenant les informations de connectivité de la liste technologique.

Le processus de synthèse automatisée peut donc être vu comme la succession de trois étapes majeures : (a) la synthèse de haut niveau ; (b) la synthèse logique et (c) le placement routage du circuit. Ce processus est illustré sur la Figure 4-9.

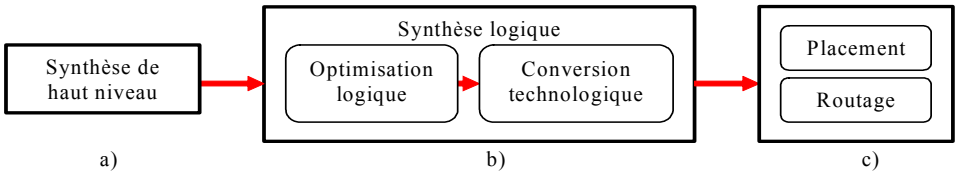


Figure 4-9 : Les 3 étapes de la synthèse d'un circuit FPGA.

Les outils de CAO tendent à offrir des procédures de synthèse de plus en plus automatisées dans le but de maîtriser la complexité des projets et de réduire les temps et les coûts.

4.2.4 Conclusion

L'utilisation des SoPCs nous permet de développer une architecture sous forme d'une hiérarchie de blocs IP (écrit en VHDL) pour bénéficier des facilités de développement et de test. Cela revient en fait à développer des composants matériels spécifiques : processeurs de calcul, unités de gestion mémoire externe et interne, adaptateurs de communication et unités d'interfaces Entrées/Sorties. De plus, les SoPCs, nous permettrons de développer la partie de gestion des communications intra puce sous forme logicielle pour bénéficier de la souplesse de la « logique programmée » exécutée par un processeur *softcore* embarqué dans le circuit cible. Notre choix s'est porté sur les processeurs *softcore* car ils nous assurent une indépendance vis-à-vis de l'obsolescence des circuits FPGA. Le lien entre les composants matériels spécifiques et le processeur *softcore* exécutant le logiciel embarqué sera établi soit en développant un driver logiciel soit en utilisant un adaptateur de communication matériel.

Chapitre 5

Une architecture de mesure par traitement d'images

Nous allons décrire dans ce chapitre la démarche qui nous a conduit à définir une architecture en adéquation avec l'application de mesure de vitesse par traitement d'images en temps réel. En s'appuyant sur les caractéristiques de notre application (Partie A) et nos choix technologiques (Partie B), nous avons abouti à la définition d'une architecture modulaire pouvant satisfaire les contraintes d'un système embarqué notamment le traitement en temps réel. Ce système est basé sur des concepts architecturaux qui permettent son adaptation aux besoins de l'application et aux évolutions de la technologie.

5.1 - Concepts architecturaux

Le choix d'une architecture est fonction de nombreux critères fortement dépendant de l'application et de ces contraintes comme la compacité ou l'adaptabilité. D'autres facteurs plus spécifiques à l'algorithme peuvent restreindre le nombre de solutions envisageables. Les trois principaux sont la complexité de l'algorithme, le volume de données à traiter et la vitesse de traitement nécessaire pour satisfaire les contraintes temps réel.

5.1.1 Les contraintes

Nous avons vu apparaître, ces vingt dernières années, de nombreuses réalisations de systèmes matériels dédiés à multitude d'applications du traitement d'images. Elles se distinguent par divers critères dont le principal est le domaine d'applications visé et les contraintes qui en découlent en terme de performances, mais aussi de complexité matérielle, de consommation, d'encombrement, etc.

Suivant les applications, les contraintes rencontrées sont de nature très différentes. En terme de performances, un système devant transformer à la volée un flux vidéo ne peut que respecter le débit trame, et une machine d'aide à la conduite, ne se conçoit que si elle est capable de fournir une réponse dans un temps compatible avec les contraintes mécaniques du système piloté. A l'inverse, d'autres applications sont beaucoup moins strictes en terme d'objectifs temporels, même si la réduction du temps de calcul reste une préoccupation constante.

La programmabilité de ces systèmes est également un critère discriminant. Certains systèmes vont appliquer systématiquement une transformation au flux de données entrant, en exécutant un algorithme prédéfini de manière répétitive. On parle alors de système dédié et leur mise en oeuvre se fait en câblant une version optimisée de l'algorithme. A l'inverse, même si la nature des données manipulées reste identique, il est nécessaire, dans d'autres cas, d'assurer une certaine flexibilité dans l'exécution des commandes et une programmabilité dans leur mise en oeuvre. C'est bien entendu le cas si l'on cherche à réaliser un système principalement axé sur le développement d'applications de traitement d'images, mais également si le système doit s'adapter à des variations importantes de l'environnement ou des objectifs du traitement. On parle alors plutôt de système spécialisé et l'adaptation au traitement d'images passe par une optimisation générique (modulaire) de l'ensemble du système. On constate d'ailleurs, avec l'évolution des performances des processeurs que, dans de nombreux cas, les systèmes spécialisés réalisés ne visent plus à dépasser les performances d'une station de travail, mais à réaliser de la manière la plus économique possible (au sens large) la transformation requise.

Le développement d'un système dédié demande beaucoup de temps et d'énergie. Nous avons donc décidé de ne pas concevoir une architecture dédiée uniquement à la mesure de vitesse par imagerie de particules mais plutôt une architecture spécialisée dans la mesure de paramètres physiques par traitement d'images. Ce choix nous permet d'ouvrir cette architecture à d'autres applications répondant aux mêmes caractéristiques que notre application. Le type d'applications que nous avons choisi présente trois caractéristiques. La première est relative aux flots parvenant au système et provenant du système. Le volume du flot de données d'entrée est important (images) et celui du flot de sortie est réduit (résultats de mesure). La seconde caractéristique est liée à la répartition spatiale des traitements. Généralement, les mesures sont relatives à des voisinages réduits dans l'image. Cela peut aller d'une zone définie par un pixel et ces plus proches voisins à des fenêtres de taille variable (zones ou Régions d'Intérêt ou RoI pour *Region of Interest*). Cela revient à partitionner l'image en zones indépendantes avant d'effectuer les calculs. Les traitements sont alors effectués indépendamment dans chaque zone de l'image en parallèle. La dernière, plus générale, concerne la faculté de s'adapter au traitement à réaliser et à sa complexité. Pour cela, l'architecture doit aisément permettre la modification du nombre d'éléments de calcul mais aussi de leur nature sans engendrer une restructuration de l'architecture dans son ensemble.

A partir de ces trois caractéristiques, nous allons définir une architecture en adéquation avec des applications de mesure de paramètres physiques par traitement d'images.

5.1.2 Définition d'une architecture spécialisée

Les performances d'une architecture sont conditionnées par la définition des éléments de calcul qui la composent mais aussi par leur interface d'entrées/sorties et le réseau d'interconnexion entre eux. Un quatrième point peut conditionner les performances d'une architecture : la mémoire. Nous en tiendrons compte ultérieurement. Mais, ce point dépend fortement des choix matériel en vue d'une implantation de l'architecture. Il nous faut donc avoir une réflexion sur :

- Les éléments de calcul ;
- l'interface de ces éléments de calcul (communications, échanges de données) ;
- l'organisation de ces mêmes éléments au sein de l'architecture ;
- et les topologies des interconnexions.

Pour pouvoir définir une architecture, système en adéquation avec notre application, il est nécessaire de faire un choix à ces quatre niveaux.

1 - Eléments de calcul

La définition de la structure interne d'un élément de calcul porte sur trois critères : le type, la granularité et les performances intrinsèques à l'élément de calcul.

Nous distinguons trois types d'éléments de calcul : fixe, configurable et programmable. Ces types d'éléments de calcul sont caractérisés par leur flexibilité. On désigne par fixe les éléments de calcul qui n'ont aucune flexibilité comme les ASIC. Ces éléments fixes sont dès le départ définis pour une ou plusieurs opérations non modifiable. Les éléments de calcul de type configurable apportent la flexibilité qui manque aux éléments de type fixe. Ces éléments sont généralement réalisés avec de la logique programmable. Nous disposons ainsi des avantages d'une structure câblée rapide mais modifiable selon les besoins. Mais, en contre partie, l'étape de configuration est statique (arrêt du système) et complexe. Enfin, pour une flexibilité totale, il existe les éléments de calcul de type programmable tels que les processeurs et les DSPs. Différents traitements peuvent être exécutés au gré des commandes de l'utilisateur. Tous ces traitements sont regroupés dans un programme modifiable à volonté stocké en mémoire interne de l'élément de calcul programmable. Nous souhaitons concevoir une architecture adaptable au traitement à réaliser et à sa complexité pour ne pas figer le système. Pour cela, les éléments de calcul doivent être de types configurable ou programmable. Les éléments de type programmable répondent au mieux à cette contrainte d'adaptabilité mais avec un coût en temps d'exécution et en surface significatif. Les éléments de type configurable apportent un gain à ce niveau puisqu'ils offrent l'utilisation de versions optimisées des algorithmes de traitement. Mais cette optimisation ne permet plus une totale adaptabilité de l'élément de calcul sans une reconfiguration. Le

choix entre l'un de ces types se fera en fonction des algorithmes à implanter, de leur complexité et du choix du support physique cible.

La mise en œuvre de l'élément de calcul s'effectue via un flot de commandes qui va séquencer les différentes opérations à réaliser. Plus le nombre et la complexité des opérations sont élevés plus les commandes le sont aussi. Pour simplifier leur séquencement, la granularité (nombre d'instructions) des commandes peut être augmentée. Notre application est orientée vers le traitement de flots de données. Cela nécessite peu de contrôle sur les flots de données à traiter, mais un volume important d'opérations arithmétiques à réaliser en temps réel. Pour cela, nous définissons des éléments de calcul réalisants des traitements sur des zones d'images indépendantes. Le contrôle de ces éléments de calcul se fait avec une granularité importante, via des macro-commandes déclenchant un ensemble d'opérations afin de réaliser le traitement des données en un minimum de temps. De plus, ces éléments de calcul doivent posséder une mémoire interne de taille significative afin de pouvoir stocker les instructions et une quantité de données suffisante. Cette mémoire de données doit permettre le traitement de portions d'images sans chargement intermédiaire.

Les performances de l'élément de calcul se caractérisent par : le nombre d'opérations réalisées par seconde (MPIS ou FLOPS), la taille et le type des opérandes manipulés et la taille de la mémoire interne. Notre application étant axée sur le traitement en temps réel d'une quantité de données relativement importante, les éléments de calcul doivent être puissants en terme de nombre d'opérations réalisées par seconde. Mais, des éléments puissants doivent demeurer actif le maximum de temps possible, sinon, les avantages de cette puissance seront perdus. Ces pertes sont essentiellement dues aux latences engendrer par la lecture séquentielle des données et la réception des commandes. Ceci est directement lié au réseau d'interconnexion et à l'architecture du système. De plus, comme ce traitement est effectué sur de petites zones d'intérêt, nous devons faire un compromis entre la puissance et le nombre d'éléments de calcul. Ce compromis est fortement dépendant de la nature du circuit cible (processeurs, DSPs ou FPGAs). Nous utiliserons des éléments de calcul assez puissants pour réaliser le traitement des zones d'intérêt de manière continue et en nombre suffisants pour pouvoir effectuer le traitement de l'ensemble des données (images) en temps réel.

2 – Interface entre un élément de calcul et l'extérieur

Un élément de calcul possède trois flots différents : le flot de données d'entrée, le flot de données de sortie et le flot de commandes. On distingue cinq modèles différents (Tableau 5-1) pour disposer ces flots autour de l'élément de calcul.

Nom du modèle	Description schématique	Avantages/Inconvénients
Modèle à flots séparés	Flot de commandes Elément de calcul Flot de données sortant Flot de données entrant	Avantages : Parallélisme, Très performant si les différents flots sont gérés simultanément Inconvénients : Nombre de connexions très importantes
Modèle de Von Neuman	Flot de commandes, de données entrant et de données sortant Elément de calcul	Avantages : Nombre de connexion réduit (un bus unique) Inconvénient : Parallélisme impossible, partage de la même bande passante
Modèle de Harvard	Flot de commandes Flot de données sortant Elément de calcul Flot de données entrant	Avantages : Répartition de la bande passante, Nombre de connexion réduit Inconvénient : Les flots de données ne peuvent pas être important
Modèle à flots de données d'entrée dominant	Flot de commandes Flot de données entrant Elément de calcul Flot de données sortant	Avantages : Répartition de la bande passante, Nombre de connexion réduit Inconvénient : Le flot de données sortant doit être réduit
Modèle à flots de données de sortie dominante	Flot de commandes Elément de calcul Flot de données entrant Flot de données sortant	Avantages : Répartition de la bande passante, Nombre de connexion réduit Inconvénient : Synchronisation entre les commandes et les données associées

Tableau 5-1 : Classification selon la séparation des flots.

Dans la catégorie d'applications que nous visons, il y a un grand déséquilibre entre les flots des données d'entrée et ceux des données de sortie. En entrée, Il s'agit d'images qui sont des supports de données contenant un volume important d'informations brutes et générées à cadence élevée. En sortie, ce sont des résultats de mesure, c'est-à-dire des valeurs codées sur quelques octets, qui présentent un volume beaucoup plus réduit. A ces flots, il faut ajouter les flots de commandes à forte granularité (macro-commandes). Mais, un traitement d'image peut être effectué par quelques commandes simples (lecture images, traitement, récupération des résultats) associées à des paramètres d'actions concis ce qui réduit l'importance des flots de commandes.

Les flots de données d'entrée sont donc prépondérants devant les autres flots (données de sortie et commandes). Pour ces raisons, un modèle de type Havard ou Von Neuman n'apparaît pas adapté. L'architecture de communication est donc basée sur un modèle où le flot d'entrée est isolé des autres. Cela nous laisse le choix entre deux modèles : celui à flots séparés et celui à flots de données d'entrée dominant. Pour affiner notre choix, les flots de résultats se limitant à quelques octets, nous pouvons les regrouper avec les flots de commandes et les intercaler entre deux commandes sans risque de conflit. Ce regroupement permet de mieux répartir la bande passante et de réduire le nombre de connections entre éléments. Ceci est réalisable avec le modèle à flots de données d'entrée dominant (Figure 5-1).

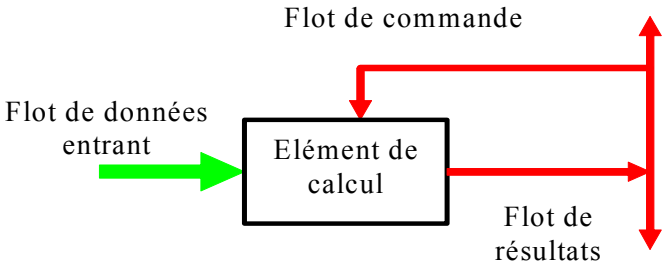


Figure 5-1 : Modèle à flot de données d'entrée dominant.

Ce choix va nous permette de gérer le système sans interférer avec les flots de données provenant de la caméra pour lesquels il y a de fortes contraintes à prendre en compte.

3 - Choix de la classe d'architecture

Nous allons maintenant décrire l'association des éléments de calcul en une architecture multiprocesseurs. Dès qu'une architecture comporte plusieurs éléments de calcul simultanément actifs, elle est dite parallèle. Diverses classifications ont été proposées pour les architectures ou machines parallèles. Leur discussion est largement hors cadre de ce travail de thèse. Nous utiliserons la plus célèbre, proposée par Flynn [Flynn 72]. Elle repose sur les caractéristiques du contrôle (ou flot de commandes) des traitements opérant sur le flot de données. Chaque catégorie de la classification de Flynn peut être déclinée suivant les cinq modèles d'éléments de calcul basés sur la séparation des flots décrits dans le Tableau 5-1.

Une architecture purement séquentielle ne comprend qu'un seul flot de commandes et ne manipule qu'une seule donnée à la fois; elle est dite SISD (*Single Instruction, Single Data*). Elle ne sera pas utilisée pour notre architecture puisqu'elle ne permet pas d'exploiter les parallélismes intrinsèques de notre application. Pour les architectures parallèles, on distingue trois classes : les architectures pipeline - dénomination largement préférée à MISD (*Multiple Instruction, Single Data*) - qui regroupent des architectures formées d'une séquence d'éléments de calcul indépendants placés sur le même flot de données exécutant chacun un flot de commandes distincts, les architectures SIMD (*Single Instruction, Multiple Data*) qui possèdent un flot de commandes unique qui pilote simultanément plusieurs éléments de calcul mais sur des données différentes et la classe des architectures MIMD (*Multiple Instruction, Multiple Data*), formées d'un assemblage d'éléments de calcul ayant chacun leur propre flot de commandes et manipulant leur flot de données.

Chacune de ces classes d'architectures présente des avantages et des inconvénients. A priori, l'organisation MIMD est la plus souple et permet d'employer des processeurs standard. A l'inverse, la distribution du contrôle entraîne un surcoût matériel et logiciel qui peut être pénalisant. Les organisations SIMD et pipeline présentent souvent, dans le cadre d'applications spécialisées, un rapport performances/coût supérieur.

Pour réaliser le choix d'une classe, nous allons procéder par élimination. Pour notre applicatif, nous avons deux sources de parallélismes : le parallélisme de données (les images subissent le même traitement) et le parallélisme de flux (données disponibles à la cadence vidéo voire supérieure). L'architecture MISD n'est pas non plus adaptée au type de traitements à réaliser. L'estimation de mouvement est appliquée sur des petites zones de l'image (traitement local) qui sont elles-mêmes en lignes de pixel avant traitement. Quand aux calculs, ils sont séquentiels. L'architecture MISD n'est donc pas retenue. Il demeure le choix entre les architectures de type SIMD et MIMD. L'architecture SIMD pourrait paraître très adaptée à des traitements identiques sur de multiples zones d'intérêt, mais des contraintes temporelles tel que la lecture séquentielle d'une image par la caméra et des limitations d'accès à la mémoire font qu'une architecture MIMD est préférée.

Aujourd'hui, la terminologie proposée par Flynn a été adaptée pour décrire un mode de fonctionnement avec une granularité de commandes plus importante : le terme d'instruction a été remplacé par celui de programme. Les différentes architectures seraient alors nommées SPSD, MPSD, SPMD et MPMD. En pratique, seuls les architectures SPMD et MPMD sont rencontrées. Cette terminologie semble bien adaptée aux applications de traitement d'images, donc à la notre.

Une architecture SPMD consiste en l'exécution d'un programme identique sur différents éléments de calcul ce qui impose un contrôle centralisé. Un flot de commandes unique relie tous ces éléments. Tandis qu'une architecture MPMD applique différents programmes sur ces éléments de calcul. Donc, chaque élément de calcul possède un contrôle distinct. L'ensemble de ce contrôle pourrait être centralisé, mais la complexité de son fonctionnement serait élevée et proportionnelle aux

nombres de programmes à gérer. C'est deux architectures peuvent avoir un fonctionnement synchrone ou asynchrone des éléments de calcul.

Notre programme ou algorithme est unique. Il consiste en une succession d'instructions séquentielles (lecture des données, traitement, transfert des résultats). De plus, il traite des zones d'intérêt indépendantes. Donc, ce programme peut être exécuté simultanément sur plusieurs éléments de calcul.

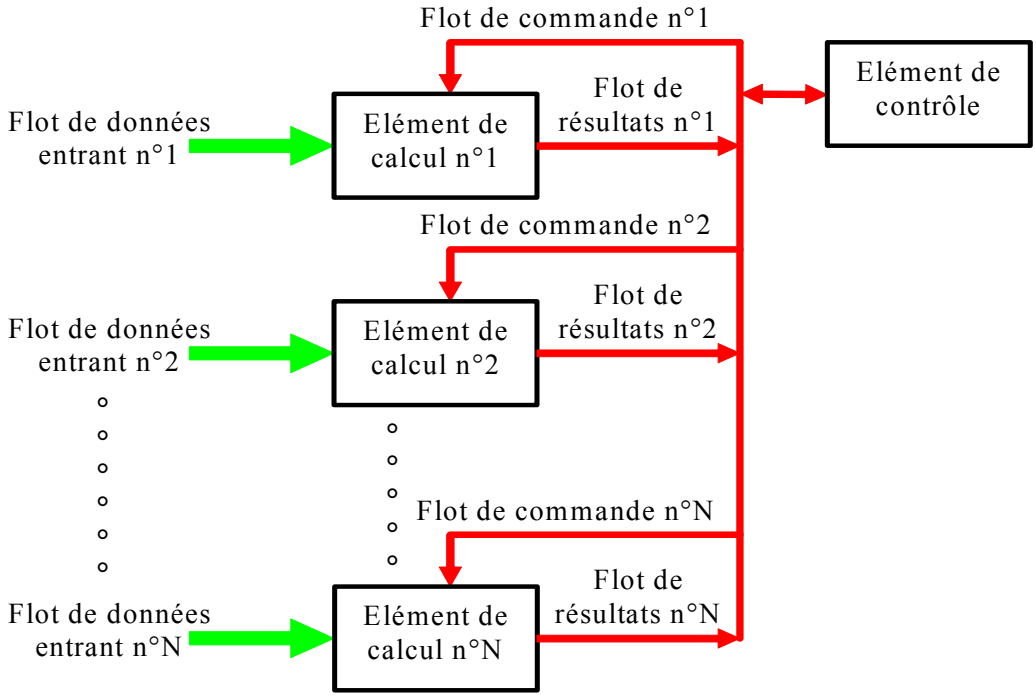


Figure 5-2 : Association de l'architecture MIMD et du modèle à flot de données d'entrée dominant.

Nous construirons une architecture MIMD avec un élément de contrôle unique (Figure 5-2), c'est-à-dire un fonctionnement SPMD. Ce choix d'unicité nous permet de faciliter le fonctionnement de l'architecture ainsi qu'une extensibilité du nombre d'éléments de calcul sans changement de l'élément de contrôle. Pour les différencier, chaque élément de calcul sera identifié par une adresse.

4 - Topologie des flots de données

La topologie caractérise la géométrie du réseau d'interconnexion dans le cas d'un système multiprocesseur (ou multi-éléments de calcul). Cette classification est complémentaire de celle de Flynn puisque la géométrie du réseau influe sur le mode de communication retenu et la complexité de la gestion des échanges entre les éléments.

En raison du choix du modèle à flots de données d'entrée dominant et d'une architecture MIMD, nous devons définir deux réseaux de communication : un pour le flot de données d'entrée, et un deuxième pour les deux autres flots. Cela nous laisse le choix entre définir une topologie identique pour ces deux réseaux ou une topologie pour chaque réseau de communication. Nous avons choisi de définir une topologie par réseau à cause de leur nature différente et des types de données qui sont à

transmettre. De plus, nous voulons garder la possibilité de modifier le nombre et la nature des éléments de calcul ce qui va influencer les choix de topologie. Ces choix peuvent s'effectuer au regard des topologies utilisées par les réseaux informatiques.

Topologie du flot de données d'entrée Une des principales caractéristiques du flot de données d'entrée est de provenir d'une source unique : la caméra et de devoir être dirigée efficacement vers un ou plusieurs éléments de calcul. Deux modes de transmission sont préconisés : un mode de transmission séquentiel (un groupe de données est dirigé vers un unique élément de calcul, puis un autre vers un deuxième élément, etc.) ou un mode de transmission parallèle (plusieurs éléments reçoivent simultanément les mêmes données). Ces deux modes de transmission sont utiles pour permettre le traitement de zones d'image qui peuvent être indépendantes ou se chevaucher.

Une structure de type bus reliant les différents éléments semble la plus appropriée pour satisfaire à cette contrainte de diffusion et pour réaliser ces deux modes de transmission de données sur plusieurs éléments (Figure 5-3).

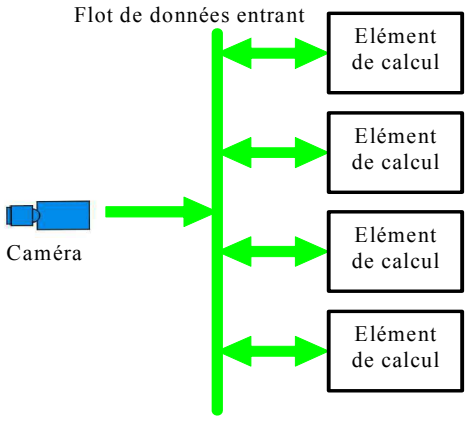


Figure 5-3 : Topologie du flot de données d'entrée.

Afin de permettre une évolution aisée du système, un module intermédiaire est intercalé entre la caméra et les éléments de calcul : le module de répartition des données. Ce module permet de réaliser l'interface avec un système d'acquisition (caméra) quelconque sans devoir apporter de modifications au reste du système (Figure 5-4). C'est celui-ci qui est relié au bus et qui distribue les données aux différents éléments de calcul en fonction des commandes de répartition.

L'introduction de ce module apporte de nouvelles possibilités architecturales. L'utilisation d'un seul bus de données partagé entre tous les modules de calcul n'est plus l'unique solution pour l'organisation des échanges de données. Dès lors, le module de répartition peut être connecté par un bus de données à un module de calcul ou un regroupement de modules de calcul (en fonction de leur nombre). Cela permet aussi de centraliser le contrôle des échanges en un seul maître : le module de répartition. Ainsi, quant une donnée est disponible, le module de répartition la transmet sur les différents bus en fonction des modules de calcul de destination.

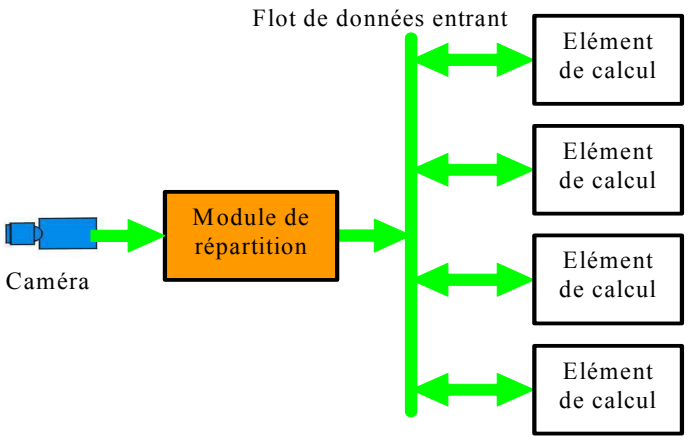


Figure 5-4 : Topologie du flot de données d'entrée avec Module de répartition.

Le choix entre ces possibilités architecturales est fortement contraint par le matériel. D'abord, le support physique retenu pour réaliser ces liaisons peut contraindre leur nombre et leur bande passante, et ainsi limiter le nombre de connexions possibles. De plus, l'implantation physique de l'architecture retenue peut avoir une grande influence. En fonction de la vitesse de traitement des données d'un élément de calcul, de la vitesse de transfert entre deux modules et de la fréquence d'acquisition de ces mêmes données, le nombre optimum de bus et leur bande passante peuvent être déterminés. La fréquence d'acquisition est fixe tout au long du fonctionnement et imposée par le capteur du système. En supposant que cette fréquence est suffisamment élevée pour fournis un flot de données en continue, le choix dépend donc du temps de transfert et du temps de calcul. Le Tableau 5-2 présente les différents cas.

Temps de transfert entre deux modules	**Temps de calcul**	**Quantité de bus de données**
faible	faible	plusieurs bus
faible	élevé	1 bus partagé
élevé	faible	plusieurs bus
élevé	élevé	de 1 à quelques bus partagés

Tableau 5-2 : Nombre de bus de données en fonction des temps de transfert et de calcul.

Cette topologie est assez flexible pour permettre l'ajout ou la suppression d'éléments de calcul (contrainte de modularité) sans modification de l'architecture.

Topologie du flot de commandes et de résultats Pour obtenir une communication fiable entre les éléments de calcul et l'unique élément de contrôle, un réseau de communication suivant une topologie en anneau a été retenue (Figure 5-5). Pour des raisons de simplicité de mise en oeuvre, cet anneau est unidirectionnel. Cette architecture est aisément modifiable puisque l'ajout ou la suppression d'un élément dans ce réseau ne modifie en rien la communication, seul le diamètre de

l'anneau est augmenté. De plus, sa structure simple permet une détection le nombre d'éléments de ce réseau à l'image de celle opéré dans les réseaux informatiques locaux.

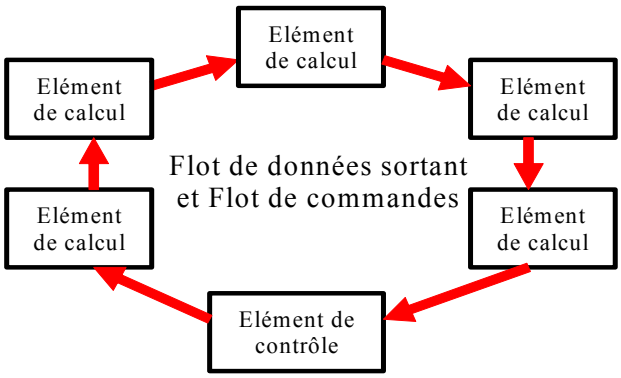


Figure 5-5 : Topologie en anneau unidirectionnel.

Elle permet, entre autre, une vérification a posteriori de la bonne réception et interprétation des commandes. En effet, les macro-commandes envoyées par l'élément de contrôle lui reviennent après un tour complet de l'anneau. Il peut ainsi savoir si les informations envoyées ont bien été reçues par le destinataire et s'il a pu les utiliser. Il faut cependant noter que cette topologie, simple à gérer, n'est efficace qu'avec un nombre d'éléments limité, sinon le temps de circulation d'une donnée peut devenir long et poser des problèmes de fonctionnement. Pour compléter cette topologie, il est nécessaire de prévoir une gestion des communications avec l'extérieur de l'anneau. D'une manière générale, chaque élément pourrait dialoguer avec l'extérieur, mais, tout comme la gestion des commandes, une centralisation de ces échanges semble plus adéquate. Et pour plus de clarté dans les communications, cette gestion avec l'extérieur est regroupée avec l'élément qui contrôle les échanges sur l'anneau. Cet élément est appelé "module de contrôle" (Figure 5-6).

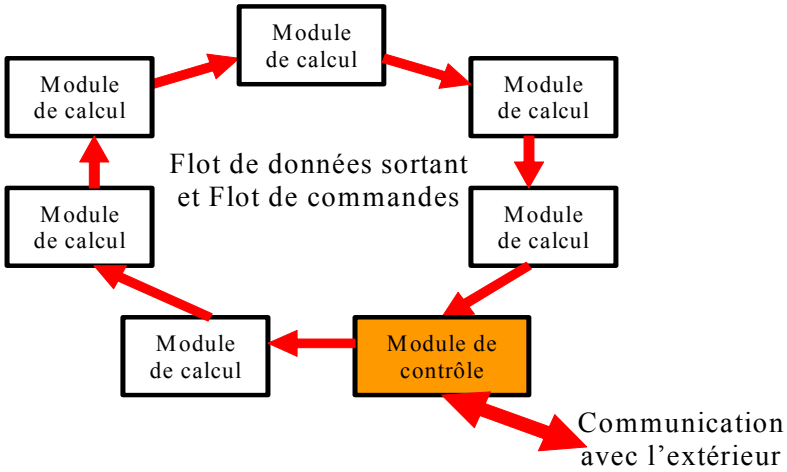


Figure 5-6 : Topologie en anneau avec un lien vers l'extérieur.

5.1.3 Architecture spécialisée obtenue

Ces deux topologies associées avec une architecture MIMD et le modèle à flot de données d'entrée dominant, nous permet de définir l'architecture suivante (Figure 5-7).

Nous retrouvons bien les caractéristiques citées précédemment : la séparation du flot dominant par rapport au deux autres et la flexibilité d'une architecture MIMD avec un contrôle centralisé.

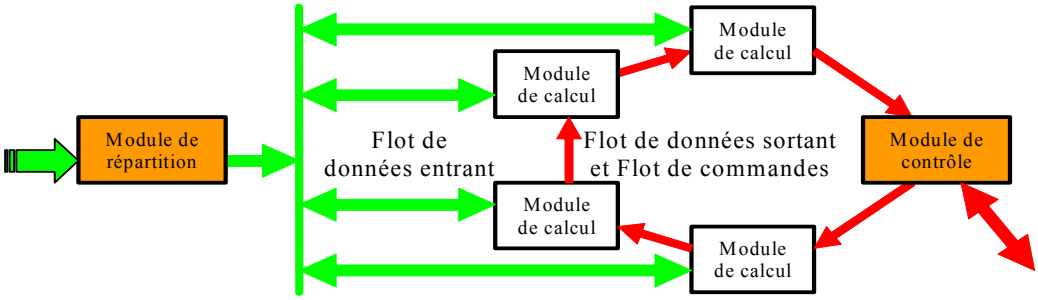


Figure 5-7 : Association des deux topologies dans une même architecture.

En l'état, cette architecture de base est utilisable, mais il apparaît nécessaire de lui apporter quelques modifications pour optimiser ses caractéristiques et performances. La première de ces modifications s'impose toute seule. Pour pouvoir mieux gérer le flot de données entrant et simplifier le contrôle de l'architecture, nous intégrons le module de répartition dans l'anneau. Ainsi, l'ensemble des commandes est regroupé dans un module unique et transmis par un seul réseau de communication. Cette option simplifie grandement l'architecture et sa gestion mais impose la conception d'un module de contrôle complexe et performant. La deuxième est plus un aspect pratique. Notre application actuelle ne nécessite aucun échange de données entre éléments de calcul. En effet, en raison du parallélisme intrinsèque des données d'entrée, chaque élément de calcul peut réaliser le traitement d'une zone de l'image sans avoir d'échange avec les autres éléments. Le bus de données d'entrée est alors utilisé de manière unidirectionnelle. La dernière est plus une remarque qu'une modification. L'organisation et la gestion de la mémoire ont été mises de côté dans la définition de cette architecture. Pour la définition de notre architecture, nous sommes partis sur une architecture à mémoire distribuée entre les différents modules qui la composent.

On obtient ainsi le schéma définitif de l'architecture (Figure 5-8). Cette architecture, de part son allure générale, a été prénommée Round-About (en français rond-point).

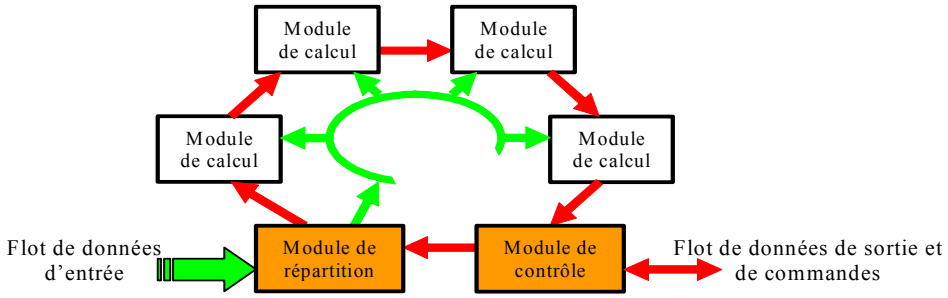


Figure 5-8 : Architecture Round-About sans échange de données entre les Eléments de Calcul.

5.1.4 Description globale de l'architecture

Cette partie vise à décrire en détail l'organisation et le fonctionnement de l'architecture Round-About indépendamment de toute implantation physique qui sera détaillée à la section suivante. L'architecture est organisée selon une approche hiérarchique et modulaire. Le niveau le plus élevé est appelé module. Il est ensuite décomposé en unités fonctionnelles. Il existe trois catégories de modules :

- Module de Calcul (MC) dont le nombre est variable,
- Module de Répartition (MR), un par source d'approvisionnement en données,
- Module de Contrôle (MCo), unique.

Les modules de calcul effectuent le traitement des données (imagette ou RoI). Ils regroupent un élément ou unité de calcul et deux unités en charge de la communication, une pour chaque flot. Seule l'unité de calcul est adaptable. Sa nature dépend du traitement à réaliser. Les autres unités permettent de réaliser l'interface avec les autres modules. Le ou les modules de répartition réalisent l'interface entre la ou les sources produisant les données (caméras, capteurs) et les modules de calcul via un bus unidirectionnel dit bus de données. Les données d'entrée sont ainsi réparties entre les différents modules de calcul avant traitement. La lecture de ces données étant séquentielle, le module de répartition intègre une unité de mémorisation temporaire pour désynchroniser le flot de données et les calculs plus rapides. Le module de contrôle agit en maître sur les autres modules, d'où son unicité. Il émet les commandes vers les autres modules et reçoit les résultats des modules de calcul. Les commandes et les résultats circulent sur l'anneau sous la forme de trames de taille fixe. Il faut noter que tous les modules ont une adresse propre. Cela permet au module de contrôle d'adresser les commandes aux modules désirés.

Cet ensemble de modules forme notre architecture Round-About spécialisée dans la mesure de paramètres physiques par traitement d'images en temps réel.

5.2 - Implantations physiques de l'architecture "Round-About"

Nous exposons, dans cette section, les implantations physiques de cette architecture réalisées. Ces diverses implantations ont apportées des modifications à l'architecture pour un bon fonctionnement du système physique.

5.2.1 Première implantation physique

Cette première réalisation est l'aboutissement des travaux de thèse de Julien Dubois [Dubois 01b] réalisés au Laboratoire TSI de Saint-Etienne. Ils décrivent l'implantation de l'architecture Round-About associant des unités de calcul à base de DSP et FPGA pour la mise en œuvre en temps réel d'outils de traitement d'images utilisés en PIV. Et ils ont abouti à la réalisation d'un prototype fonctionnent avec un caméra CCD analogique (Figure 5-9).

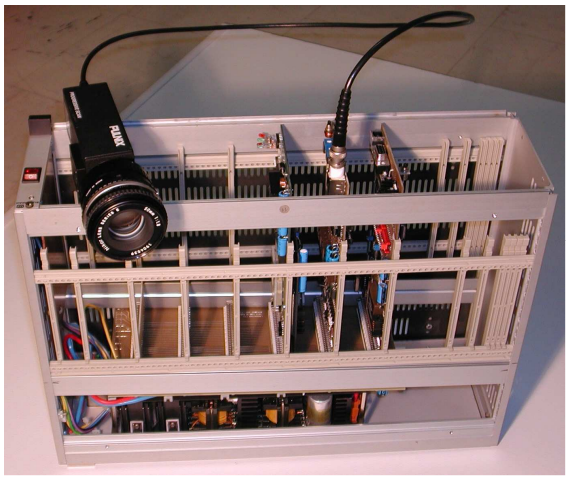


Figure 5-9 : Prototype fait de cartes électroniques [Dubois 01b].

1 – Implantation de deux algorithmes différents

L'architecture Round-About a été définie sur des concepts d'adaptabilité aux besoins de l'application et spécialisée dans la mesure de paramètres physiques par traitement d'images. Dans ces travaux de recherche, une étude d'implantation d'algorithmes de corrélation sur différents circuits cibles a été réalisée. De cette étude, deux algorithmes on été retenus pour l'application de mesures de vitesse par PIV. Le premier algorithme est un algorithme de corrélation par FFT sans normalisation (cf. Chapitre 1) appliqué sur de images à 256 niveaux de gris. Succession de multiplications et d'additions, cet algorithme nécessite l'utilisation d'un processeur de type DSP pour atteindre les performances voulues. Le deuxième algorithme est une adaptation de l'algorithme de corrélation directe pour réaliser une corrélation binaire par comparaison avec des opérateurs logiques. L'utilisation d'images binaires apporte une réduction de la quantité de ressources matérielles nécessaire pour satisfaire la contrainte de traitement en temps réel (cadences de traitement plus élevées).

Deux implantations du module de calcul ont donc été réalisées en adéquation avec les algorithmes retenus. Elles sont basées sur des circuits numériques de type DSP et FPGA.

2 – Adaptation du module de calcul aux deux implantations

Le module de calcul est divisé en trois blocs, chacun étant implanté sur un circuit différent (Figure 5-10). Le premier bloc réunit l'ensemble des fonctions de communication, de contrôle et de gestion des

échanges de données du module. Son implantation est réalisée avec un FPGA de type XC4036XLA de la société Xilinx. Le deuxième représente la mémoire. Des bancs de mémoires externes au FPGA permettent de désynchroniser le flot de données et la vitesse de calcul. Pour cela, ces mémoires sont découpées en deux zones indépendantes permettant des échanges simultanés entre, d'une part, le module de répartition des données et l'unité mémoire et d'autre part, l'unité mémoire et l'unité de calcul. Ces mémoires sont gérées par le FPGA.

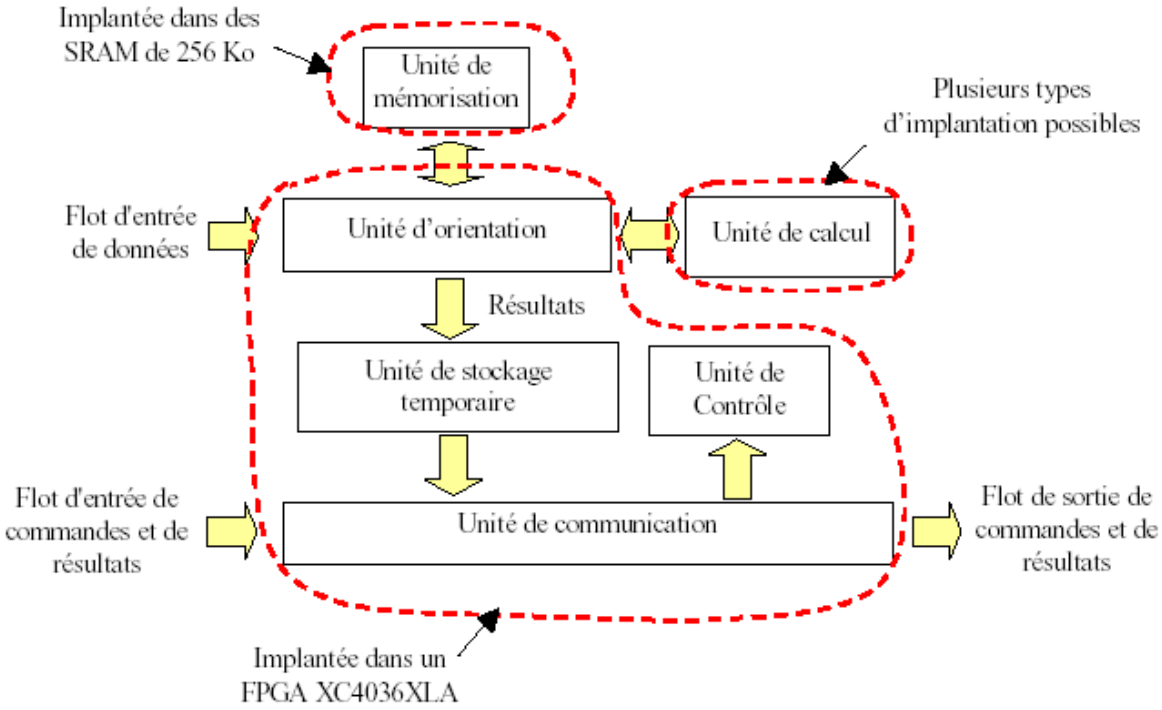


Figure 5-10 : Implantation du module de calcul.

Quant au troisième bloc, c'est l'unité de calcul. L'unité de calcul est l'unité qui s'adapte au traitement de l'application ou évolue pour améliorer les performances du système. En raison des deux algorithmes retenus, deux versions de l'unité de calcul ont été développées. Seule cette unité en charge du traitement est modifiée pour passer d'une version à l'autre. L'implantation de cette unité peut être soit interne au FPGA soit externe sur un circuit annexe (ici un DSP). Pour le traitement d'images à niveau de gris, l'unité de calcul comprend un circuit DSP de type TMS320C6201 qui est alimenté en données grâce à son port à grand débit (Port HPI) par le FPGA. Dans le cas du traitement d'images binaires, c'est le FPGA qui se charge des calculs en plus de ces autres rôles. Ceci est possible par l'adéquation de l'algorithme à l'architecture interne de FPGA (comparaison avec des opérateurs logiques et succession d'additionneurs). Il a été obtenu un module de calcul avec une architecture mixte pour répondre aux contraintes d'adaptations.

3 – Description du prototype

Le système réalisé est composé de cartes électroniques, une carte correspondant à un module de l'architecture. Chaque module est alors composé de divers circuits numériques réalisant chacun une fonction précise du module. La Figure 5-11 présente l'implémentation physique des différents modules du système.

L'implantation du module de calcul est répartie entre un FPGA, des bancs mémoire et un DSP comme présenté dans le paragraphe précédent.

Le module de répartition des données comporte un circuit (CPLD) et des mémoires tampon (FIFO). Le CPLD est de type EPM7128 de la société Altera. Il gère la répartition des données depuis les mémoires vers différents modules de calcul en fonction des commandes qu'il reçoit. Quand aux mémoires tampon, elle stocke le flux continu de données provenant d'une caméra CCD analogique dans l'attente de leur envoi sur le bus de répartition.

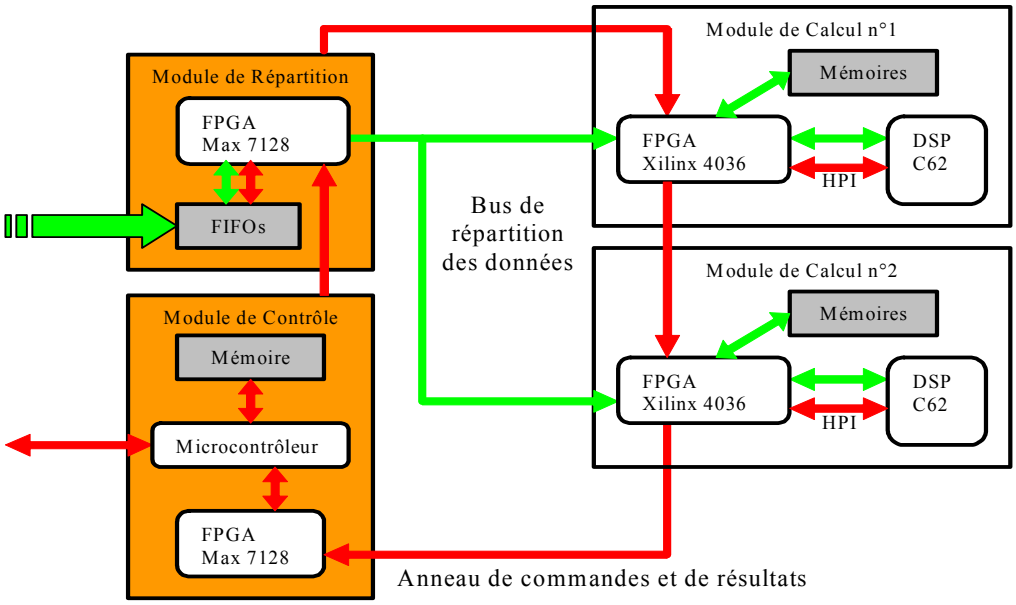


Figure 5-11 : Schéma du système Round-About multi-cartes.

Quant au module de contrôle, il est composé d'un microcontrôleur (SIEMENS) qui est le cœur de l'architecture et d'un CPLD (Altera EPM7128) pour la communication sur l'anneau. Associé à une mémoire locale, ce microcontrôleur organise le fonctionnement de l'architecture en gérant les commandes qui sont émises et reçues sur l'anneau par le CPLD. Cette gestion comprend l'ordonnancement des commandes, la désignation de leur destination et le contrôle de leurs bonnes exécutions. De plus, le microcontrôleur sert d'interface entre le système et un PC pour la transmission des résultats des calculs, mais aussi pour le contrôle en ligne.

5.2.2 Deuxième implantation physique

Avec les évolutions technologiques de la microélectronique, les FPGAs deviennent de plus en plus puissants, excédants aujourd'hui dix millions d'éléments de logique. L'intégration de coeurs de processeur au sein même du FPGA devient une possibilité. Par conséquent, l'architecture Round-About peut être entièrement implantée dans une seule puce. Cette évolution vers les circuits logiques configurables permet d'augmenter la modularité du système, mais aussi de bénéficier de la facilité d'intégration d'éléments décrits avec le langage VHDL. Cependant, les contraintes technologiques impliquent quelques modifications de plusieurs éléments de l'architecture. Ces modifications concernent principalement la mémorisation des images et la partie gérant l'acquisition.

Cette deuxième réalisation reprend l'application de la précédente réalisation en intégrant dans sa conception les évolutions technologiques récentes présentées dans les chapitres 3 – Les capteurs d'images et 4 – Systèmes embarqués monopuce.

1 - Adaptation pour l'utilisation d'imageur CMOS

Le remplacement de la camera CCD par un imageur CMOS permet une simplification de l'interface entre l'architecture Round-About et la source des images. Mais, cela implique quelques modifications du module de répartition des données.

Les capteurs CMOS présentent plusieurs avantages en considérant les caractéristiques d'intégration du système que nous voulons développé. En premier lieu, la gestion d'un capteur CMOS est facilitée par les niveaux de signaux numériques compatibles avec les autres circuits numériques. En second lieu, ce type de capteurs possède une interface numérique simplifiée et se pilote comme une mémoire. Enfin, les capteurs CMOS permettent un adressage aléatoire de petites fenêtres sur la matrice de pixel et donc une sélection directe des régions d'intérêt (RoI) de l'image à traiter.

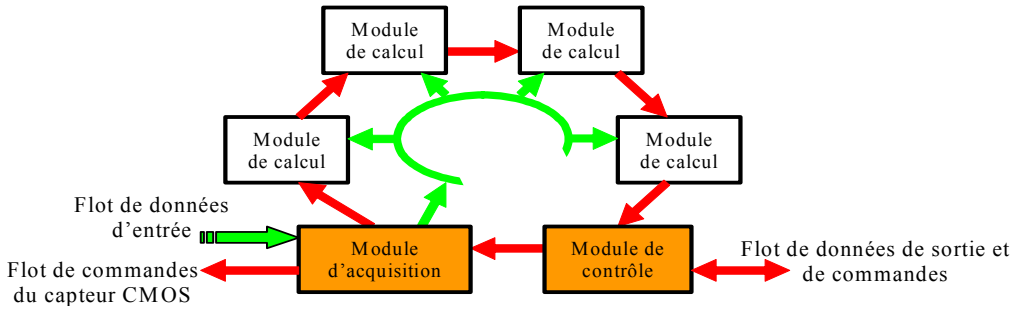


Figure 5-12 : Architecture Round-About modifiée pour l'utilisation d'imageur CMOS.

Avec l'utilisation de capteurs d'images CMOS, le module de répartition doit être adapté à ces nouvelles conditions. Ce module est remplacé par un module d'acquisition, qui gère toutes les commandes du capteur et reçoit le flot de données d'entrée (Figure 5-12). Semblable au module de répartition, le module d'acquisition est inséré dans l'anneau et toutes les commandes associées au

capteur CMOS (fenêtrage, configuration, prétraitement...) sont envoyées du module de contrôle par l'anneau.
Toutes les possibilités de ce type de capteurs apportent des avantages certains pour les applications de traitement d'image, la plus importante étant une réduction significative du flot des données d'entrée et donc une augmentation de vitesse de calcul.

2 - Adaptation vers un système sur une seule puce

Dans les systèmes électroniques classiques, les grandes dimensions des cartes électroniques entraînaient des problèmes électriques. Elles limitaient notamment la vitesse de communication. Cela pouvait conduire à des communications lentes entre des composants potentiellement très rapides. C'était particulièrement critique pour la communication entre le processeur et la mémoire : quelle que soit la vitesse du processeur, il devait aller lire ses instructions en mémoire. Avec les systèmes monopuce, la communication reste toujours un goulot d'étranglement, car elle est très consommatrice de surface, mais avec un facteur bien moindre. En effet, le fait d'avoir sur la même puce l'ensemble du système raccourcit les chemins de communication et facilite la construction d'architecture s'accordant aux localités de calcul de l'application. Ces facilités permettent ainsi souvent de réduire la quantité de mémoire globale du système. Les systèmes monopuce sont aussi moins encombrants et surtout, ils peuvent consommer moins : en effet les données doivent transiter par des chemins beaucoup moins longs, l'énergie nécessaire à cette transmission est donc plus faible.

Bien que l'architecture Round-About soit aisément transposable dans un système monopuce, des modifications sont nécessaires pour un fonctionnement optimal du système. Ces modifications portent sur l'intégration de l'ensemble de l'architecture dans un unique FPGA, la gestion de la mémoire et le contrôle du système.

Description hiérarchique Comme nous l'avons vu dans la section précédente, l'architecture Round-About est déjà découpée en blocs fonctionnels appelés modules. Ces modules regroupent les fonctions de communications, de contrôles, de mémorisations et de traitements. Pour une meilleur synthèse de cette architecture logique complexe, nous avons appliqué une description structurée : séparation de ces modules en unités, qui sont elles-mêmes divisées en blocs regroupent une fonction précise (cf. : Chapitre 6 – Description fonctionnelle des Modules). Un module se décomposera en quatre unités fonctionnelles : une unité de communication, une unité de contrôle, une ou plusieurs unités de traitement et une unité d'interface (Figure 5-13).

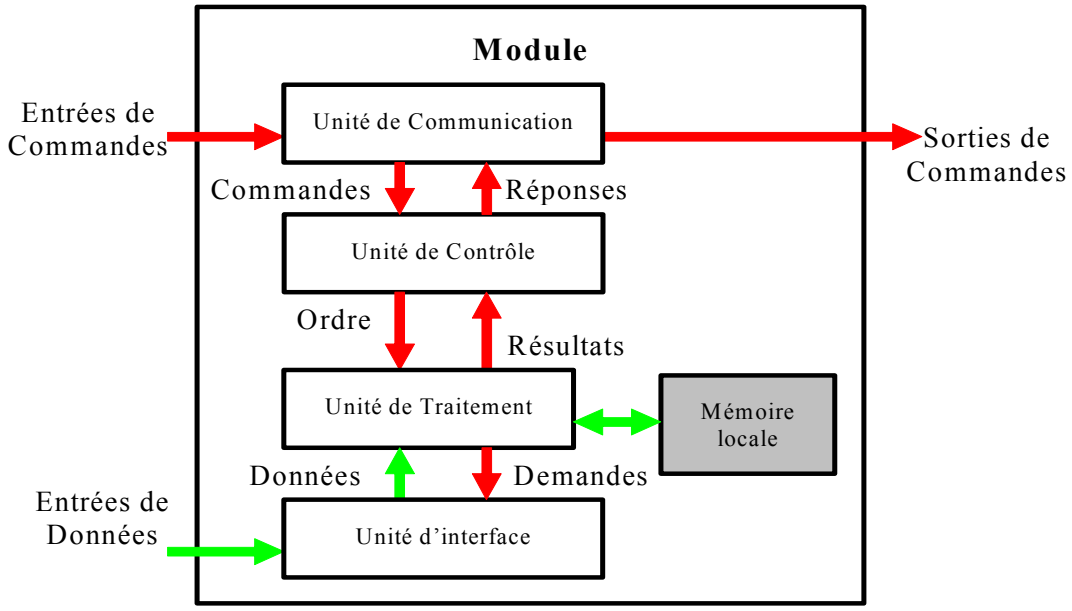


Figure 5-13 : Décomposition standard interne d'un module.

L'unité de communication regroupe les fonctions simples (émission et réception) de communications entrantes et sortantes du module avec l'extérieur. Elle transmet les commandes et attend les réponses de l'unité de contrôle. L'unité de contrôle, également appelée séquenceur contient un automate qui décrit l'enchaînement des actions que doit faire le module. Cette unité émet les ordres de travail et réceptionne les résultats d'une ou de plusieurs unités de traitement. Les résultats retournent à l'unité de communication pour être transmis à l'extérieur. L'unité de traitement contient les opérateurs de traitement de données utiles au module. Parmi ces opérateurs, on trouve les opérateurs arithmétiques et enfin les opérateurs logiques. Les ordres sont les actions demandées par l'unité de contrôle aux opérateurs de l'unité de traitement en fonction des commandes reçues par l'unité de communication. Les résultats sont des valeurs retournées par l'unité de traitement, qui conditionnent l'évolution du séquenceur, donc du module. Quand à l'unité d'interface, elle apporte les données nécessaires au traitement. Cette unité est optionnelle aux modules ne travaillant pas sur des données comme le module de contrôle. Elle regroupe les connectiques servant aux liaisons de données (multiplexeurs, démultiplexeurs et bus). Ce principe de conception revient à répondre aux questions suivantes : séparation contrôle/traitement, interface avec l'extérieur, synchronisation interne, dynamique du module (horloge) et la connectique.

Nous aboutissons à une architecture hiérarchique et modulaire, dont la décomposition facilite à la fois la synthèse mais aussi les tests de fonctionnement.

Gestion de la mémoire De nos jours, on est capable de fabriquer des systèmes de plus en plus complexes sur une même puce avec de la mémoire et de la logique. Cette évolution technologique est accompagnée par l'apparition d'applications dans les domaines tels que le traitement d'images qui manipulent des données très volumineuses et/ou fortement dépendantes. Elles exigent par

conséquent l'intégration de mémoires de différents types et tailles au sein du système. Il existe alors deux grands types d'architecture mémoire pour concevoir un système de traitement d'images :

- une mémoire globale partagée, accessible par tous les processeurs, dans laquelle l'image est intégralement présente,
- une mémoire distribuée, c'est-à-dire que l'image est découpée et répartie dans les mémoires de chaque processeur. La mémoire associée à un processeur peut être accessible aux autres processeurs (mémoire distribuée partagée), où non (dans ce cas, les processeurs émettent une demande pour recevoir un segment de l'image).

Notre choix s'est porté sur une architecture MIMD utilisée en mode SPMD (voir paragraphe 3 de la section 5.1.2). L'organisation de la mémoire et les méthodes d'accès à la mémoire par les processeurs sont la base d'une seconde classification des architectures MIMD [Blazewicz 00].

Architecture MIMD à mémoire partagée (*Shared memory*) : Destinées au départ aux architectures SIMD, les architectures à mémoire partagée sont maintenant portées sur des architectures MIMD exploitées alors en mode SPMD. Les architectures multiprocesseurs peuvent se partager une mémoire centralisée unique grâce à une ressource de communication. Cette ressource peut être soit un bus de données partagés, cas général, soit un réseau commuté [Guerrier 00]. Puisqu'il y a une seule mémoire principale avec un temps d'accès uniforme pour chaque processeur, ces architectures ont une méthode d'accès à la mémoire uniforme appelée UMA (*Uniform Memory Access*). Ces systèmes offrent un modèle de programmation général et commode permettant un partage simple des données, à travers un mécanisme uniforme de lecture et d'écriture des structures partagées dans la mémoire globale. Ils sont très utilisés pour les calculs parallèles sur des données volumineuses qui peuvent alors être échangées par mémoire partagée (applications de traitement d'image). Mais, la ressource de communication devient rapidement un goulot d'étranglement lorsque les requêtes des processeurs à la mémoire sont nombreuses et simultanées. Cela engendre donc un phénomène de contentions des processeurs qui est proportionnel à leur nombre dans l'architecture. Ce nombre est donc généralement limité et dépend de la technologie employée.

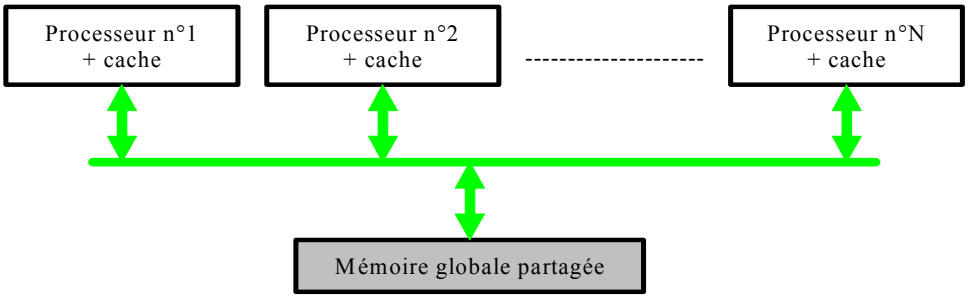


Figure 5-14 : Architecture à mémoire partagée centralisée.

Ces systèmes souffrent donc d'un handicap qui est la grande latence dans l'accès à la mémoire, ce qui rend leur flexibilité assez limitée. Cependant, ce type d'architectures à mémoire partagée centralisée reste de loin l'organisation la plus populaire actuellement dans les architectures multiprocesseur.

Architecture MIMD à mémoire distribuée (*Distributed memory*) : Ces systèmes sont constitués de plusieurs noeuds indépendants. Chaque noeud consiste en un ou plusieurs processeurs connectés à de la mémoire locale par un bus d'interconnexion. Les noeuds sont connectés entre eux en utilisant des technologies d'interconnexion extensibles (scalables) (Figure 5-15) comme un réseau de communication.

Ces architectures peuvent avoir deux méthodes d'accès à la mémoire distribuée : la méthode NUMA (*Non Uniform Memory Access*) et la méthode NORMA (*No Remote Memory Access*).

Dans une architecture à accès de type NUMA, tous les processeurs ont accès à tout l'espace mémoire de l'architecture mais avec un temps d'accès non uniforme (l'accès à la mémoire locale du processeur est nettement plus rapide que l'accès à la mémoire d'un processeur distant via le réseau). Ce temps d'accès dépend de la localisation géographique des données dans l'architecture. Bien que physiquement distribuée dans toute l'architecture, la mémoire est toujours vue, par tous les processeurs, comme un espace unique. Ce type d'architecture est dit à mémoire distribuée partagée (*Distributed Shared Memory*).

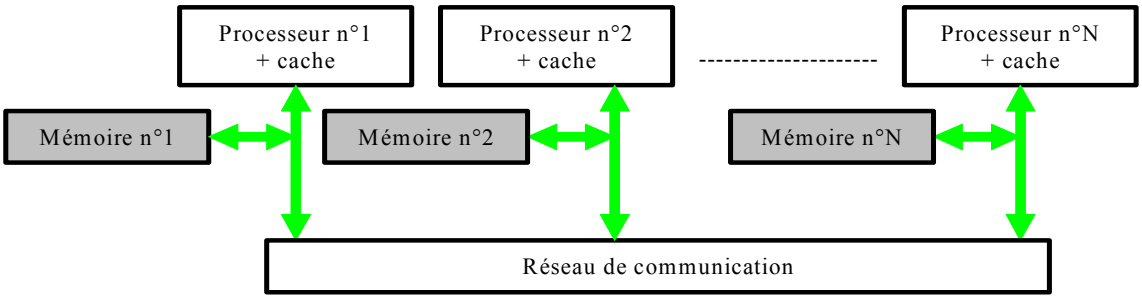


Figure 5-15 : Architecture à mémoire distribuée.

Dans une architecture à accès de type NORMA, les processeurs n'ont plus accès à toute la mémoire de l'architecture. Cela correspond aux architectures à mémoire physiquement distribuée, mais à accès non partagé. Chaque processeur est connecté à une mémoire locale (qui lui est privée), et il ne peut accéder à la mémoire locale d'un autre processeur. Les échanges de données se font de manière explicite sur un réseau de communication qui connecte les processeurs. Comme chaque processeur est connecté à sa mémoire, il peut fonctionner quasi indépendamment des autres. Elles ont l'intérêt d'être facilement extensibles (nombre de processeurs). Mais, elles sont plus difficiles à gérer puisque les données ne sont pas partagées.

La nature flexible de tels systèmes, les rend d'une très grande capacité de calcul. Mais, la communication entre des processeurs affiliés à des noeuds différents nécessite l'utilisation de modèles de communication par passage de messages impliquant un usage explicite de primitives du type Send/Recieve. En optant pour ce type de systèmes, le concepteur doit particulièrement faire attention à la distribution des données ainsi qu'à la gestion des communications (le transfert pose un important problème à cause des différents espaces d'adressage). Ainsi, les problèmes de gestion et d'accès sont relativement complexes dans les systèmes à mémoire distribuée.

Notre choix : Le choix entre ces deux architectures mémoire n'est pas du tout évident. Il est fortement contraint par le type de mémoire utilisé, le type de processeur (processeur à DMA pour *Direct Memory Access* par exemple), le réseau de connexion des processeurs, la bande passante des bus, mais aussi par la nature de l'application (nombre d'accès aux données, périodicité des accès, etc.).

Notre système a pour objectif de réaliser un traitement d'images en temps réel ce qui le soumet à de fortes contraintes temporelles. Le traitement d'images manipule des données très volumineuses et/ou fortement dépendantes ce qui exige beaucoup de mémoire et une gestion rigoureuse (accès et découpage). Par exemple, les ressources mémoire nécessaire pour une image de 1024×1024 pixels codés sur 10 bits sont d'environ 1,3 Mo. Le problème actuellement est le manque d'efficacité en terme de mémoires embarquées d'un système tout intégré sur FPGA. Pour pallier ce problème, le recourt à de la mémoire externe au FPGA de type RAM est donc nécessaire. Mais, la transposition en mémoire externe dans un environnement de type SoC n'est pas une tâche d'intégration immédiate et facile. C'est pourquoi, cette mémoire externe sera utilisée en tant que mémoire partagée.

Par conséquent, l'architecture Round-About a été modifiée pour répondre au mieux à ce type d'accès mémoire. Le module d'acquisition perd la fonction de gestion et envoi des données au profit d'un nouveau module : le module de mémorisation (Figure 5-16).

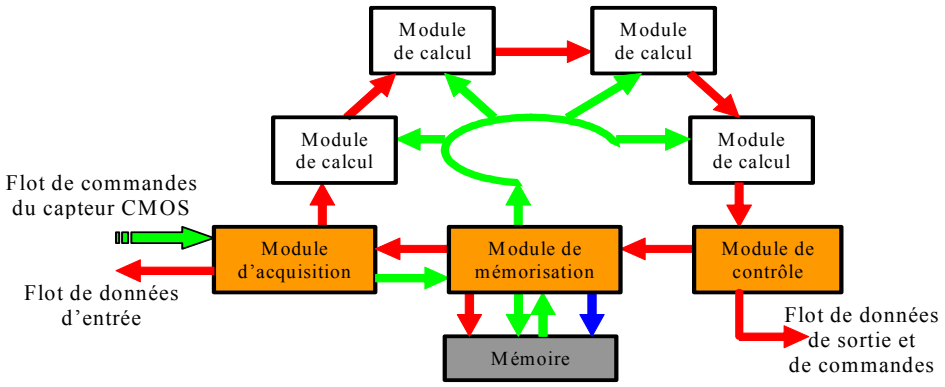


Figure 5-16 : Architecture Round-About de type MIMD à mémoire partagée.

Situé sur le flot de données entrant entre le module d'acquisition et les modules de calcul, son rôle est de stocker les images entrantes, puis, de diffuser ces données images sur le bus de données. En

gardant le même principe de commande par anneau, le module de mémorisation est inséré dans l'anneau. Chaque ordre de transfert mémoire est alors lancé par le module de contrôle et commandé par des macro-commandes envoyées sur l'anneau. L'introduction de ce module apporte deux avantages à l'architecture :

- ce système d'interface mémoire indépendant du reste de l'architecture permet d'augmenter la portabilité du système. Seul ce module change si la mémoire externe change.
- l'architecture présente une meilleure segmentation. Chaque module possède une fonctionnalité qui lui est propre.

Pour tenter d'atténuer le problème de goulot d'étranglement de communication engendré par ce type de mémoire, nous utilisons deux variantes architecturales. La première repose sur l'ajout de mémoire cache (registres) au niveau de chaque module de calcul. Nous diminuons ainsi la latence d'accès à la mémoire et réduisons les temps de calcul en rendant les données nécessaires au calcul disponibles. La seconde consiste à diviser l'accès à la mémoire partagée en tranches de temps égales (quantum de temps). C'est-à-dire que l'accès à la mémoire ne se fait plus sur interruptions mais par un cycle d'autorisations. Chaque quantum de temps est dédié à un type d'accès (écriture ou lecture). L'ordonnancement de ces accès permet de réaliser un multiplexage temporel en attribuant à chacun une durée fixe. Ce multiplexage temporel est détaillé dans le Chapitre 6, section 6.4 et paragraphe 6.4.1.

Comme notre architecture de type MIMD à mémoire partagée UMA est composée de plusieurs modules de calcul identique, nous pouvons la classer dans la catégorie SMP (*Symmetric Multi-Processor*) [Zomaya 96].

Organisation du flot de données Dans sa définition générale, l'architecture Round-About gère le flot de données en utilisant un bus unique partagé entre tous les modules de calcul. Mais, chaque implantation physique étant différente du point de vue de ces contraintes et ces avantages, l'organisation du flot de données doit être adapté au mieux pour répondre aux contraintes, ainsi qu'aux performances envisagées. Un système monopuce permet de supprimer les contraintes physiques et matérielles des interconnexions entre plusieurs circuits. Dans un FPGA, les connexions d'un module ne sont plus limitées par le nombre de pattes du circuit. Donc, pour permettre un accès plus rapide aux données, nous pouvons envisager de connecter chaque module de calcul au module de mémorisation par un bus de données unique. Nous implanterons donc cette solution, dans un premier temps, avec deux modules de calcul pour évaluer le fonctionnement global du système. Puis, en fonction des différents temps mesurés, nous affinerons cette solution pour qu'elle réponde aux contraintes de l'application, dont le traitement en temps réel. Cela nous permettra de prendre en

compte la longueur des liaisons requise par l'implantation physique puisqu'elle intervient dans les temps de transmission d'une donnée.

Contrôle de l'architecture Le module de contrôle a deux rôles au sein de l'architecture : il gère les échanges de données avec l'extérieur du système et organise le bon déroulement des commandes. La transposition de ce module dans un système monopuce n'est pas aisée car plusieurs choix s'offrent à nous. Lors de la première implantation physique, ce module était composé d'un CPLD et d'un microcontrôleur (cf. paragraphe 5.2.1). Le cœur du module est le microcontrôleur car ce type de processeur s'adaptait parfaitement à la gestion de notre application. C'est-à-dire qu'il était fait appel conjointement à des ressources logicielles micro-programmées et des ressources matérielles configurables.

Aujourd'hui, l'intégration de coeurs de processeur au sein même d'un FPGA devient une possibilité pour réaliser des systèmes monopuce numériques puisque les deux sont optimisés technologiquement. Pour intégrer une ressource logicielle micro-programmée dans l'architecture, nous avons recourt à un processeur embarqué (voir Chapitre 4 - paragraphe 4.2.2). Il nous reste alors le choix entre utiliser un cœur de processeur *softcore* ou *hardcore*. Il faut noter que les cœurs de processeur *hardcore* sont utilisables que sur des FPGAs dépendants, c'est-à-dire un système synthétisé pour un FPGA donné, ne peut être implanté que sur ce FPGA (même si l'autre FPGA est de la même famille). Pour cette raison, nous nous sommes porté vers l'utilisation de cœurs de processeur *softcore*. Ce choix répond aux besoins de notre architecture telles que une gestion simple, robuste et adaptable des communications, un ordonnancement des tâches (donc des commandes) modifiable et une communication avec l'extérieur facile.

Ce cœur de processeur est associé à deux blocs IP (émission et réception) pour assurer la communication avec le reste de l'architecture par l'intermédiaire de l'anneau. L'ensemble forme le module de contrôle.

3 - Synthèse

L'évolution de la complexité mesurée par l'équivalent de porte logique par circuit rend l'implantation complète de notre architecture sur une même puce possible. Cette possibilité engendre toute une série d'adaptations, ou modifications. Elles ont abouti à l'architecture suivante (Figure 5-17) basée sur l'architecture générique Round-About. Composée de cinq modules, elle répond aux contraintes de notre application de traitement d'images en temps réel [Lelong 03].

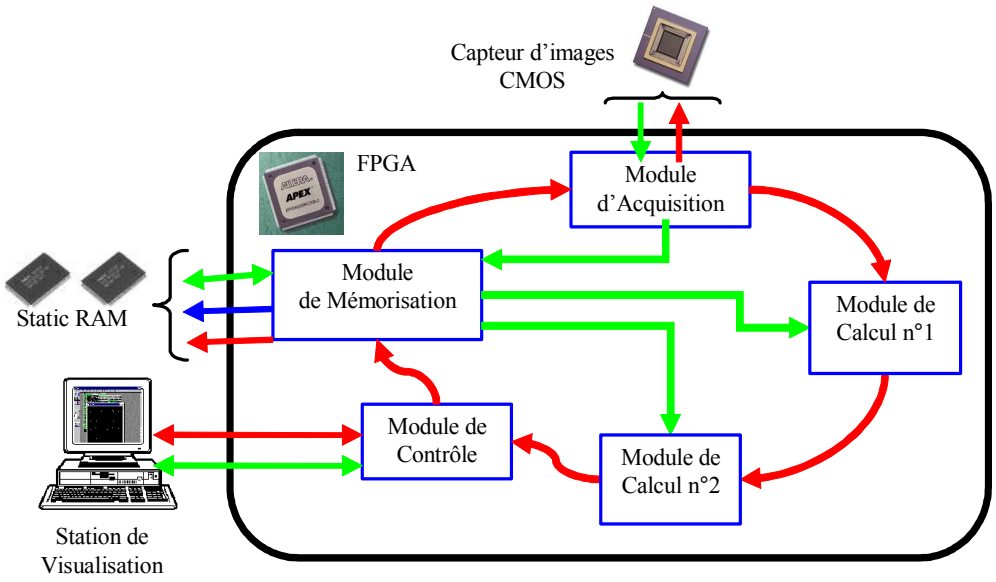


Figure 5-17 : Architecture Round-About dans un système SoPC-FPGA.

Afin de réaliser une validation de cette architecture adaptée, un prototype a été mis au point. Cette implantation sera développée plus en détails dans le Chapitre 7.

Ce prototype a permis de démontrer que la mise en œuvre en temps réel d'outils de traitement d'images utilisés en PIV était réalisable [Lelong 04b]. Il est constitué d'un capteur d'image CMOS de la société FillFactory [fillfactory-web] et d'une plateforme matérielle SoPC-FPGA de la société Altera [Altera-web]. Bien que le capteur utilisé soit moins performants (7 images 1280×1024 par seconde) que d'autres capteurs actuels, il possède des caractéristiques adaptées à notre application (qualité d'image équivalente à un capteur CCD, faible bruit et contrôle des gains et offsets). Quand à la plateforme matérielle, il s'agit d'une carte de développement Excalibur équipée d'un circuit FPGA APEX EP20K200 (8320 *Logic Elements* soit 200 000 portes et 104 Ko de RAM interne) et de 256 Ko de RAM statique. Elle permet l'implantation du cœur de processeur logiciel microprogrammé : NIOS.

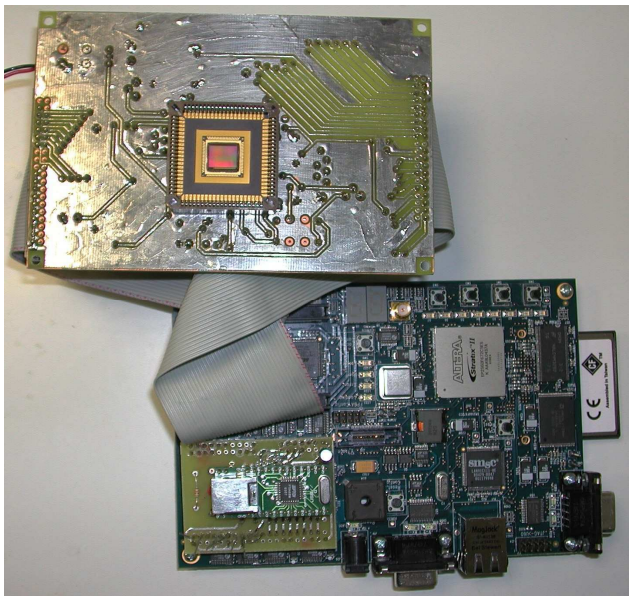


Figure 5-18 : Prototype constitué d'une plateforme FPGA couplé à un imageur CMOS.

Une évolution de ce prototype a été réalisée au début de l'année 2005 par le remplacement de la plateforme SoPC-FPGA par une version supérieure (Figure 5-18). Cette nouvelle plateforme est une carte de développement NIOS II spécialement conçue pour le développement d'applications utilisant le processeur d'Altera NIOS dans sa deuxième version. Elle comporte un FPGA Stratix II EP2S30 (33 880 LE et un total de 1,3 Mo de RAM interne) associé à 1 Mbyte de SRAM et 16 Mbyte de Single-Data-Rate SDRAM. Ce changement n'a demandé aucune nouvelle modification de l'architecture ou du système. Il a été fait pour permettre l'utilisation de la version supérieure du processeur NIOS nommé NIOS II et de nouvelles possibilités matérielles comme la communication Ethernet, de la mémoire SDRAM et une plus grande fréquence de fonctionnement.

Chapitre 6

Description fonctionnelle

Pour suivre les concepts d'architecture, le système a été conçu dans une méthodologie de conception en blocs IP regroupés dans des modules (Figure 6-1). Ceci permet la réutilisation de la partie principale de la conception sur différentes cibles.

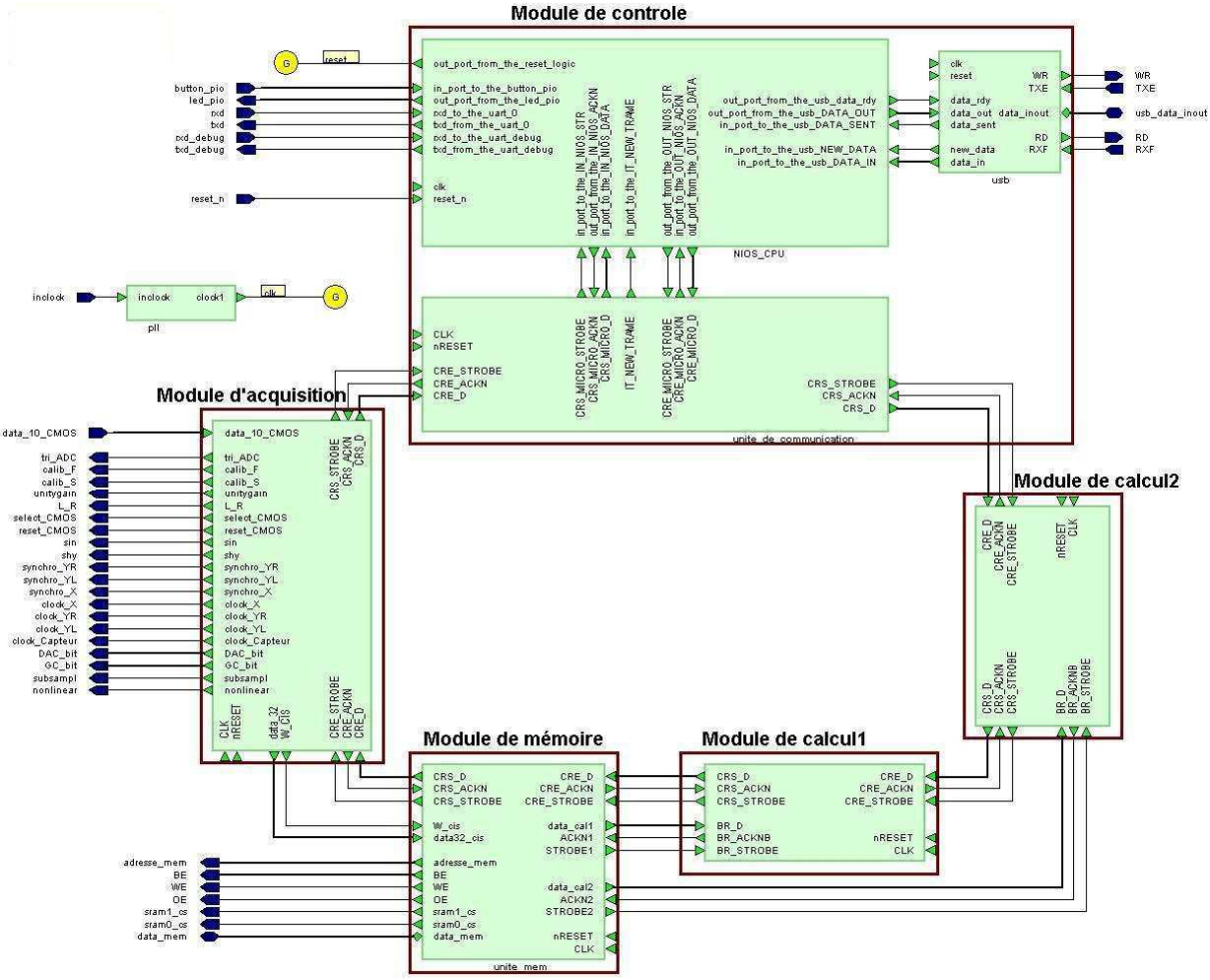


Figure 6-1 : Schéma blocs de l'architecture Round-About obtenu sous HDL Designer de la société Mentor Graphic.

Avant d'aborder le fonctionnement des différents modules constituant Round-About, nous allons détailler les échanges entre modules. Puis, les unités communes à tous les modules seront décrites, pour terminer par la description en détails de chaque module.

6.1 - Description des échanges entre modules

La nature même de l'architecture et plus particulièrement celle des modules dépend de la fonction qu'ils ont à réaliser. Pour ne pas modifier entièrement les modules à chaque modification du système ou des traitements, les deux réseaux d'échanges à savoir celui des commandes et des résultats et celui des données d'entrée, doivent être indépendants de la nature des modules qu'ils interconnectent.

Nous allons voir le fonctionnement de ces réseaux de communication ainsi que leur organisation permettant une indépendance vis à vis du reste des unités composant les modules.

6.1.1 Bus de transport des données

Comme son nom l'indique, ce bus sert uniquement aux transports des données. Il s'agit des données images provenant d'un capteur d'images ou d'une caméra et à destination des modules de calcul pour y être traitées. Pour notre architecture adaptée au SoC, nous avons défini deux types de bus de transport de données : le bus reliant le module d'acquisition au module de mémorisation et le bus allant du module de mémorisation à un module de calcul. Cette distinction a été faite car bien que leur rôle soit identique, la gestion de la communication est différente.

1 – Bus Module d'Acquisition – Module de Mémorisation

Dans ce cas, le bus transporte les données images provenant du capteur d'images jusqu'à la mémoire. Il sera unidirectionnel et devra transférer les données en flux continu à la vitesse du capteur. Pour réaliser ce transfert, un bus de données de largeur 32 bits (Data_32) associé à un signal de validation des données (W_CIS) relie le module d'acquisition et le module de mémorisation.

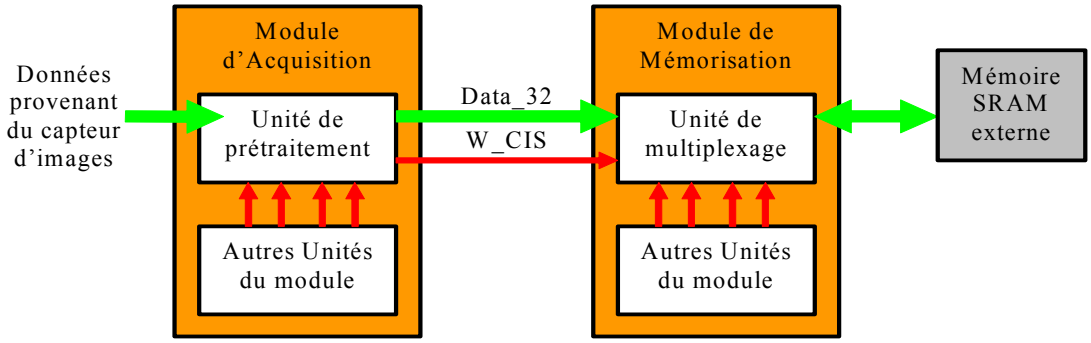


Figure 6-2 : Organisation du bus Module d'Acquisition – Module de Mémorisation.

Pour gérer la transmission, une unité spécifique a été intégrée dans chacun des modules : l'unité de prétraitement du module d'acquisition et l'unité de multiplexage du module de mémorisation (Figure 6-2). Ces unités assurent l'indépendance de la mise en œuvre de la transmission vis-à-vis du type de capteur d'images et de mémoire utilisés (format et vitesse des données fournies,…). De plus, ces deux unités permettent une désynchronisation entre les vitesses d'acquisition et d'écriture et la vitesse de transfert des données interne à l'architecture. Pour parfaire cette désynchronisation, ces unités sont pourvues de registres de stockage. Dans le cas de l'unité de prétraitement, ce registre sert aussi à la mise en forme des données sur 32 bits après leur prétraitement (seuillage, etc.). Cela engendre une latence dans le flux de données. Pour éviter que cette latence monopolise le module de mémorisation et l'accès à la mémoire externe, le signal W_CIS valide la donnée présente sur le bus comme une donnée à enregistrer dans le registre de stockage. Ce registre de l'unité de multiplexage permet la conservation des données valides dans l'attente de l'ordre d'écriture et de leur adresse mémoire.

Ce système nous permet de réaliser ainsi l'isolement du bus par rapport aux autres unités constituant les modules.

2 – Bus Module de Mémorisation – Module de Calcul

Ici, le bus fournit des données provenant de la mémoire à un module de calcul en fonction des commandes que reçoit le module de mémorisation. Elles sont réparties entre les modules de calcul au fur et à mesure de leur disponibilité. Ce transfert s'effectue à partir d'un bus de données de bande passante de 32 bits (B_D) suivant un mode de transmission asynchrone à l'aide de deux signaux de *handshake* (B_ackn pour *acknowledge* et B_strobe pour *strobe*). Pour gérer la transmission, chaque module de calcul possède une unité dédiée (Figure 6-3). Elle est nommé unité d'interface.

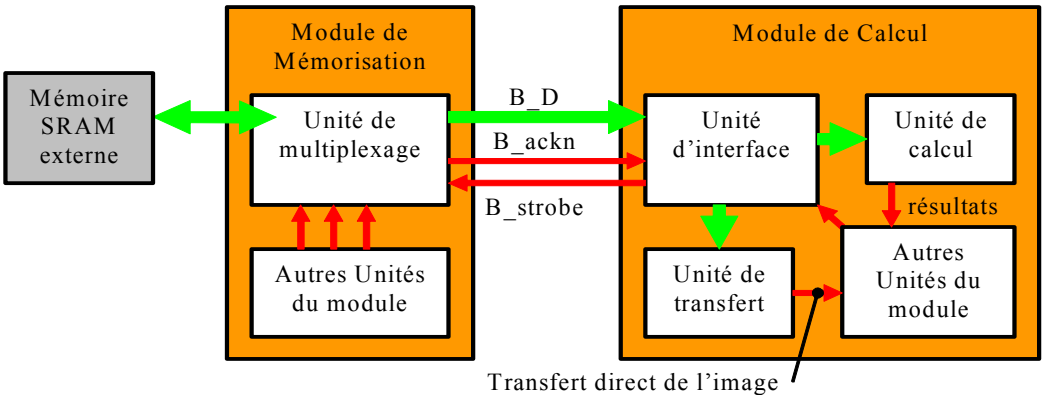


Figure 6-3 : Organisation du bus Module de Mémorisation – Module de Calcul.

Cette unité sert d'intermédiaire entre le bus de données et les unités destinataires des données (unité de calcul et unité de transfert). Elle gère l'orientation du flux de données et le *handshake* grâce à une machine d'états. Elle n'effectue aucun stockage temporaire. Comme précédemment, le transfert de données et sa gestion sont isolés des modifications internes des modules. Du côté du module de

mémorisation, l'utilisation d'un buffer dans l'unité de multiplexage découple la communication et la lecture des données dans la mémoire externe. De plus, il sert de verrou, tant que le module de calcul n'émet pas une demande (B_strobe) les données lues en mémoire reste dans le buffer. Le mode *handshake* à trois fils est généralement utilisé pour des transferts vers plusieurs modules de calcul simultanément (mode *multi-cast*), mais l'utilisation de gros FPGA rend possible l'usage d'un bus pour chaque module de calcul dans la limite d'un nombre raisonnable de modules au sein de l'architecture.

3 – Synthèse

Le flux de données entrantes suit donc un parcours simple divisé en deux phases : le trajet de la source à la zone mémoire et le trajet de la zone mémoire au registre de stockage avant traitement (Figure 6-4). L'organisation de ce système permet d'obtenir un réseau de transmission indépendant de la nature de la source de données et du type de mémoire externe utilisée. Il s'adapte facilement par simple modification des unités (prétraitement, multiplexage et interface) disposées le long du flux. Autre avantage, des modifications sur les autres unités des modules ne remettent aucunement en cause ni le système de transmission ni son fonctionnement.

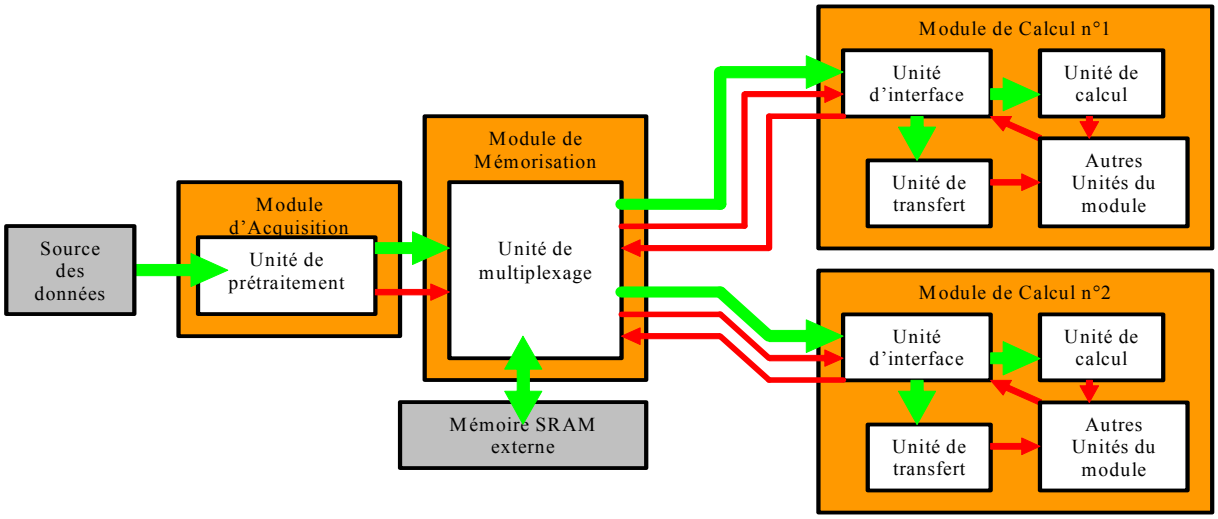


Figure 6-4 : Cheminement des données depuis leur source jusqu'au lieu où elles seront traitées.

Actuellement, chaque module de calcul est relié au module de mémorisation par un couple : bus de données et signaux de *handshake* qui lui est propre. Mais si un trop grand nombre de modules de calcul est nécessaire au bon fonctionnement de l'architecture, un bus pourra desservir un groupe de modules de calcul tout en gardant un principe similaire.

6.1.2 Réseau d'échanges de commandes et de résultats

Le réseau permettant la communication entre tous les modules de l'architecture se présente sous la forme d'un anneau unidirectionnel. Cet anneau sert à la circulation des commandes et des résultats (cf. : Chapitre 5).

1 - Organisation de l'anneau

L'anneau de communication est un ensemble de connexions point à point unidirectionnelles. Il est formé par un bus logique de 8 bits de largeur (CR_D) et de deux signaux de *handshake* (CR_strobe et CR_ackn). La bande passante du bus peut être adaptée en fonction des contraintes spécifiques au système cible. Tout comme le bus de données reliant le module de mémorisation et le module de calcul, l'anneau utilise un système de *handshake* identique pour s'assurer de la bonne transmission des données. Chaque module réceptionne le flux de commandes et de résultats du module qui le précède et le transmet au module qui le suit dans l'anneau. Pour se faire, chaque module comporte une unité de communication (Figure 6-5), leur enchaînement forme la continuité de l'anneau, donc des communications. Ces unités de communication et leur fonctionnement sont détaillés dans la section 6.2 - Parties communes dédiées à la communication. Elles permettent une unicité du protocole de communication puisque entre n'importe quelles unités de communication, la communication fonctionne de manière identique. De plus, elles sont un gage de l'uniformité des temps de transmission sur l'anneau.

La communication est initiée par le module de contrôle et retourne au module de contrôle après avoir parcouru l'anneau. Ce module génère les informations sous forme de trame de 6 octets. Ces trames permettent de transmettre les commandes aux autres modules, mais aussi à ces autres modules de transmettre des informations au module de contrôle. Ceci est rendu possible par une circulation permanente de trames dite vides sur l'anneau entre deux trames de commandes. Ces trames vides peuvent être remplies par les autres modules qui ont à transmettre des résultats de traitement ou des informations.

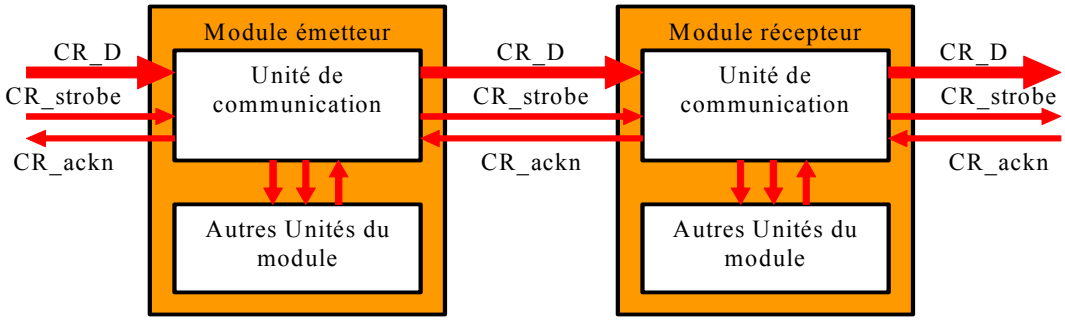


Figure 6-5 : Organisation de la communication entre deux modules.

Pour ne pas perturber la circulation des trames sur l'anneau, chaque module copie les trames qui lui sont destinées avant de les décoder. Dans le même laps de temps, cette même trame est transmise au module suivant de l'anneau. Pour ce faire, les modules possèdent une unité de décodage avec un buffer pouvant stocker une trame complète. Pour identifier les trames qui lui sont destinées, les modules comparent leur adresse (attribuée lors de la conception de l'architecture) à l'adresse figurant dans la trame qui est celle du module auquel est destiné les commandes. Ce type d'organisation permet une circulation ininterrompue des trames sur l'anneau, tout en ciblant les trames

à un unique destinataire. Tous les modules suivent ce mode de fonctionnement à l'exception du module de contrôle puisqu'il en est le gestionnaire. Celui-ci ne retransmet pas les trames qu'il reçoit, il les arrête soit pour déterminer si ses commandes ont bien été reçues, soit pour savoir si les trames vides qu'il a générées ont été remplies.

Tout comme le bus de transport des données, l'anneau est dissocié de la structure interne des modules pour permettre soit son évolution en un réseau de communication plus complexe soit l'indépendance de la communication vis-à-vis du rôle des modules (acquisition, mémorisation, calcul).

2 - Le protocole de gestion des trames au niveau des modules

Comme nous l'avons vu précédemment, les informations échangées entre les modules circulent sur l'anneau sous la forme de trames. Nous allons détailler dans les lignes qui suivent les protocoles mis en œuvre dans les modules pour extraire ces informations. Nous ferons la différence entre le protocole du module de contrôle et celui des autres modules.

Protocole pour le module de contrôle Le module de contrôle étant l'instigateur des communications, son protocole de gestion des trames est différent des autres modules. Le module de contrôle génère les trames en fonction des informations à émettre et de son séquenceur d'ordre. Deux possibilités peuvent être mis en œuvre pour cette génération soit un calcul de chaque trame à émettre soit un registre de toutes les trames de commandes dans une mémoire interne. L'une comme l'autre, ces solutions sont lourdes et consommatrices de ressources. Pour simplifier ce fonctionnement, une solution intermédiaire est préconisée. Les trames les plus émises sont disponibles en mémoire tandis que les autres sont calculées avant chaque envoi. Une fois la trame prête, elle est placée dans une file d'attente. Elle permet d'insérer cette trame de commande dans la circulation des trames de l'anneau. Cette circulation est due à l'émission régulière de trames vides par le module de contrôle. Ces trames ne sont pas adressées à un module spécifique mais au module de contrôle lui-même. Ces trames vides ont donc un temps de circulation fixe puisque tous les modules les transmettent au suivant.

Toutes les trames émises reviennent au module de contrôle grâce à la structure en anneau. Alors, deux cas se présentent. Si la trame reçue est marquée, elle a donc été enregistrée et décodée correctement par le module de destination. Dans le cas contraire, le module de contrôle place cette trame dans la file d'attente de l'émission pour effectuer un nouvel envoi.

Protocole pour les autres modules Pour chaque module, à la réception d'une nouvelle trame, un test sur l'adresse est réalisé. Il consiste à comparer cette adresse du destinataire contenue dans la trame à l'adresse du module réalisant le test. Si le test est négatif, la trame poursuit son chemin sur l'anneau. Dans le cas contraire, la trame est recopiée car elle peut comporter des informations complémentaires (adresses, nombre de mots à lire, etc.). Dans le même temps, un second test est réalisé. Il consiste à vérifier la disponibilité du module et de ces ressources pour exécuter la commande associée à la trame. Si c'est le cas, cette même trame est marquée comme reçue par le destinataire puis

transmise au module suivant. Si les ressources ou le module ne sont pas disponibles la trame est alors transmise sans marquage. Ce système de marquage rend plus robuste le protocole de commandes et l'administration de l'architecture par le module de contrôle.

Pour les trames vides, le protocole est similaire à celui d'une trame de commande. Mais avant sa réexpédition vers le module suivant, cette trame est complétée par le module uniquement s'il dispose d'un résultat ou d'une information à transmettre au module de contrôle. Lorsqu'un module remplit une trame, celui-ci remplace l'adresse du module de contrôle présente dans la trame vide par la sienne pour indiquer au module de contrôle d'où provient le résultat ou l'information. Cette modification de l'adresse empêche les autres modules d'utiliser cette trame pour envoyer leurs propres informations, et ainsi détruire celles déjà présentes. Lorsque cette trame est reçue par le module de contrôle, les informations sont extraites puis interprétées par le séquenceur. Dans le cas d'une trame de résultats, ils sont transmis vers l'extérieur de l'architecture soit vers un ordinateur de visualisation soit vers un système d'affichage.

3 – Définition d'une trame

Le format des trames est fixe. Cela permet d'obtenir une gestion identique et donc plus simple pour l'ensemble des trames par les modules. Chaque trame se compose de six octets (Figure 6-6). Le premier octet contient deux informations distinctes : l'adresse du module de destination (4 bits de poids fort) et le type de commande qu'il doit exécuter (4 bits de poids faible). Les quatre octets suivants, contenant des données associées à la commande, diffères suivant le type de la trame. Quand au dernier octet, il représente le marquage de la trame par le destinataire. Le module cible de la trame place dans cet octet, au départ à 0, l'information correspondant à son état actuel vis-à-vis de la trame reçue. Cette information d'état, destinée au module de contrôle, représente la réception correcte de la trame par le module destinataire, l'utilisation efficace de la commande et des informations associées (si la partie du module intéressée par la commande n'est pas déjà occupée) ou non. Actuellement, seuls ces trois états sont utilisés pour le bon fonctionnement de l'architecture Round-About. Mais, des états supplémentaires peuvent être ajoutés pour la réalisation de tests de l'architecture ou pour révéler des dysfonctionnements.

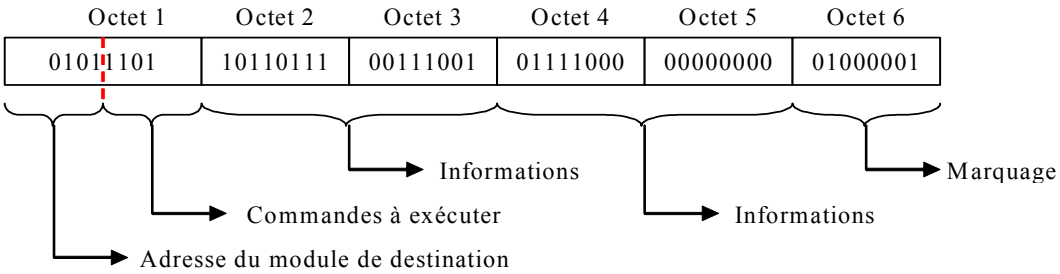


Figure 6-6 : Exemple de la structure d'une trame.

Ce format de trame a été choisi pour répondre au mieux à l'organisation et aux protocoles de communications de l'anneau. La trame de 6 octets est transmise par l'anneau octet par octet d'un

module à l'autre. Ce type de communication a été choisi pour permettre la réception d'un octet et l'envoi d'un autre par l'unité de communication des modules en même temps. Cela minimise et fixe le temps de passage d'un champ à l'intérieur d'un module tout en lui permettant de lire les informations qu'il contient.
Pour qu'il n'y ait aucun problème de remplissage ou de lecture des champs, nous avons choisi comme convention d'écrire les octets de droite à gauche c'est-à-dire de placer le LSB à droite et le MSB à gauche.

4 – Composition des trames utilisées dans l'architecture

Nous détaillerons les commandes et les informations associées nécessaires à un fonctionnement normal du système et les trames qui y sont associées. Un tableau (Tableau 6-1) récapitulatif des trames utilisées dans l'architecture est donné en fin de paragraphe.

Trames de commande adressées au module d'acquisition Elles sont au nombre de huit. Elles ont pour fonction la configuration du capteur et l'acquisition des images. Trois trames correspondent respectivement à la remise à zéro du module, à l'ordre d'acquérir un nombre défini d'images par le capteur et à la configuration d'éventuels prétraitements des données avant leur envoi vers le module de mémorisation. Quand aux autres, elles sont soit destinées à la définition de la position et de la taille de l'image qui va être acquise soit utilisées pour les réglages de paramètres du capteur tels que le gain, l'offset, etc.

Trames de commande adressées au module de mémorisation Elles sont au nombre de six. Nous pouvons regrouper ces trames en trois fonctions distinctes : remise à zéro du module, écriture de données en mémoire et lecture de ces données en mémoire. Pour la remise à zéro du module, la trame est identique à celle du module d'acquisition mais avec l'adresse du module de mémorisation. Pour la phase d'écriture, deux trames sont nécessaires : une pour définir la taille de mémoire à réserver (exprimée en nombre de mots de 32 bits : taille de la pagination de la mémoire externe) et une autre pour lancer l'écriture des données à l'adresse se trouvant dans la trame. L'espace mémoire réservée par la première trame correspond à l'espace mémoire pour stocker une image dont la taille est définie par les trames de configuration du module d'acquisition. Cette information a été placée dans une trame différente de l'ordre d'écriture car cette taille d'image ne varie pas ou très peu d'une écriture à une autre, donc il n'est pas nécessaire de la rappeler à chaque fois. L'adresse, associée à l'ordre d'écriture, est en fait l'adresse de début d'écriture des données ; les autres adresses sont calculées par le module (incrémentation de 1). L'espace mémoire nécessaire étant connu, l'écriture s'arrête après que le module ait écrit à l'adresse correspondant à la somme de l'adresse de début et la taille mémoire de la première trame. Pour le mode lecture, nous utilisons les deux mêmes trames que pour l'écriture avec le même fonctionnement du calcul de l'adresse, seule la commande change (lecture au lieu d'écriture). Mais dans ce cas, nous souhaitons lire que la zone de l'image que le module de calcul à besoin et non

l'image entière. Pour ce faire, la première trame ne donne plus la taille de l'image mais l'espace mémoire à lire qui correspond à la zone de l'image dont le module de calcul à besoin. Tout comme pour le mode écriture, cette information est séparée de la trame de commande car elle ne change que si la nature des traitements effectués par le module de calcul change. De plus, l'image a été mémorisée de manière séquentielle ligne par ligne. Donc, la zone de l'image utile pour les traitements n'est pas physiquement répartie au même endroit de la mémoire. Pour inclure ce décalage des données à lire, une troisième trame (modulo de lecture) est définie en mode lecture. Sur le même principe que les trames définissant les tailles, elle donne le nombre d'adresses inutiles entre deux lignes de la zone lue (ce nombre est proportionnel à la largeur d'une image). Ce chiffre permet au module de mémorisation de calculer toutes les premières adresses de chaque ligne de la zone à partir de celle fournie avec l'ordre de lecture.

Pour une définition plus générale, ces six trames constituent la base pour le bon fonctionnement du module. Mais l'architecture étant modulaire, le nombre de modules d'acquisition et de calcul est modifiable suivant les besoins du système. Si plusieurs modules d'acquisition sont nécessaires au système alors les deux trames du mode écriture seront multipliées autant de fois qu'il y a de modules d'acquisition. Cela permet ainsi de différencier les flux de données entrant par une trame et des informations différentes, donc de simplifier la gestion du système par le module de contrôle et d'éviter les conflits mémoires. Il en va de même pour le mode lecture et le nombre de modules de calcul, à la différence près que la trame de modulo de lecture reste commune à toutes les lectures puisque ce ne sont pas deux images que nous voulons traitées en même temps mais plusieurs zones d'une même image (parallélisme des données) sur différents module de calcul.

Trames de commande adressées au module de calcul Elles sont au nombre de trois. Elles permettent de réaliser soit le transfert direct des données pour un affichage de l'image soit les calculs de vecteurs vitesses dans la zone de l'image chargée dans le module. Elles servent essentiellement à avertir le module de calcul de l'arrivée de données sur le bus et à lui indiquer ce qu'il doit en faire. La trame de transfert direct permet une lecture directe des données image sans passer par le traitement c'est-à-dire un stockage temporaire de ces données dans l'attente d'une trame vide sur l'anneau. Cette commande est accompagnée du nombre de données à transférer, en nombre de mots de 32 bits. Cela évite au module de contrôle de compter les données transférées et d'envoyer une nouvelle trame pour stopper le transfert. Ainsi, le transfert s'arrête de lui-même quand la quantité de données est atteinte (même si des données arrivent encore par le bus) et libère le module de calcul qui peut alors exécuter une autre commande. Pour réaliser la corrélation, le module de calcul doit disposer de deux zones de deux images consécutives qui se trouvent en mémoire. Deux trames sont donc nécessaires pour les calculs. La première trame est utilisée pour préparer le module de calcul à recevoir les données de la première zone et de la stocker temporairement. La deuxième en plus de réceptionner et stocker les données de la seconde zone, déclenche les calculs lorsque toutes les

données sont disponibles. Comme pour le transfert direct, une information sur le nombre de données est associée à ces deux trames pour vérifier la bonne transmission des données par le bus (aucunes pertes ni interruption).

Trames en retour adressées au module de contrôle Elles sont au nombre de trois : une remplie par le module d'acquisition et deux par le module de calcul. Comme nous l'avons vu dans le paragraphe 2 - Le protocole de gestion des trames au niveau des modules, les trames vides sont à leur départ adressées au module de contrôle. Mais, si cette trame est remplie d'informations par un autre module alors l'adresse du module de contrôle, dans le premier champ de la trame, est remplacée par l'adresse du module qui dépose ces informations. Pour éviter une confusion par le module de contrôle entre deux trames envoyées par un même module mais pour deux raisons différentes, un mot de commande est associé à chaque trame vide utilisée en fonction des informations transmises. Le module d'acquisition remplit une trame vide pour signifier au module de contrôle qu'il a terminé l'acquisition du nombre d'images demandées dans la trame d'acquisition. Cette trame ne contient aucune information, elle est juste là pour prévenir de la libération de la ressource. Pour le module de calcul, deux trames différentes sont nécessaires pour que le module de contrôle puisse différencier les données brutes et les données traitées. Ces trames transmettent au module de contrôle soit l'image (transfert direct) soit les résultats de la corrélation sous la forme d'un mot de 32 bits réparti dans les quatre champs de la trame vide. Pour le transfert direct, le module de calcul se contente de découper le mot de 32 bits fournis par le module de mémorisation en 4 octets et de l'envoyer sur l'anneau. Quand aux résultats de la corrélation, seules les valeurs du déplacement en X et en Y, chacun sur 1 octet, sont transmises par les trames vides. Ces valeurs sont données par rapport au point supérieur droit de la zone de l'image qui a été traitée.

	Module d'Acquisition	**Module de Calcul**	**Module de Mémorisation**
Envoyées au module	reset start acquisition configuration prétraitement configuration CIS position_x position_y x_size y_size	transfert direct lecture imagette lecture motif+calcul	reset taille écriture position écriture+start taille lecture position lecture+start modulo de lecture
Envoyées par le module	fin de transfert	résultat transfert direct résultat corrélation	

Tableau 6-1 : Récapitulatif des trames.

5 – Séquencement de l'ensemble des trames

La mise en œuvre de ces trames peut être illustrée sur deux séquences : l'une pour l'acquisition et l'écriture en mémoire et l'autre pour la lecture, le traitement et l'envoi des résultats.

Pour la séquence gérant l'acquisition et l'écriture en mémoire, deux versions sont possibles : l'une incorporant la configuration du module d'acquisition et l'autre sans c'est-à-dire que la configuration a déjà été faite et ne nécessite aucun changement. Dans un fonctionnement général du système, la séquence avec configuration est effectuée au départ puis les autres séquences se font sans aucune modification de cette configuration. Nous allons décrire dans ce qui suit la séquence avec la configuration. Cette séquence doit débuter par les trames **reset** du module d'acquisition et du module de mémorisation pour éliminer toutes les données (configurations, adresses, tailles, etc.) encore stockées dans ces modules et qui correspondent à la séquence précédente. Une fois ceci fait, les différentes configurations du module d'acquisition doivent être initialisées. Les six trames contenant des informations sur la configuration de l'acquisition et du capteur d'images sont émises par le module de contrôle. Aucun ordre d'envoi n'est prédéfini, la seule contrainte est que toutes ces trames reviennent marquées comme « lu et exécuté » au module de contrôle pour que la séquence puisse continuer son déroulement. Dans l'attente de ces retours, le module de contrôle prépare le module de mémorisation pour le mode écriture par l'envoi des trames **taille écriture** et **position écriture+start**. Comme cela, le module de mémorisation est prêt à recevoir les données et à les écrire en mémoire. Une fois la configuration réalisée, le module d'acquisition reçoit la trame **acquisition** qui déclenche les commandes du capteur pour la prise du nombre d'images désiré. Quel que soit le nombre d'images acquises, elles sont rangées dans la mémoire externe les unes à la suite des autres. Une fois que le nombre d'images est atteint, le module d'acquisition se met en attente de nouvelles commandes et transmet par l'intermédiaire d'une trame vide au module de contrôle qu'il a terminé. A ce stade, la première séquence est terminée, le module de contrôle passe alors à la deuxième.

Cette deuxième séquence débute par la mise en attente du module de calcul concerné par le transfert de données depuis le module de mémorisation. Ceci se fait par l'émission d'une trame **transfert direct** ou **lecture imagette** qui indique à ce module quelles informations il va recevoir et ce qu'il doit en faire. Puis, le module de mémorisation réceptionne les trois trames associées à la lecture de la zone de l'image destinée à ce module de calcul. Ces trames sont **taille lecture**, **modulo de lecture** et **position lecture+start**. Dans cet ordre, elles permettent au module de mémorisation de savoir la taille de la zone, la position physique en mémoire de la première donnée et de pouvoir calculer les autres facilement. Chaque adresse est calculée pendant la transmission des données de l'adresse précédente. Ceci permet au module de mémorisation de toujours avoir l'adresse à disposition. La transmission des données entre ces deux modules se termine d'elle-même pour chaque module quand leur compteur interne indique que la quantité de données est atteinte. Une fois ce transfert fini, en fonction du travail que doit effectuer le module de calcul, le module de contrôle

continue ou fini la séquence. Si le module de calcul devait faire un transfert direct des données alors, pendant l'arrivée des données, les précédentes sont incorporées dans les trames vides qui parcourent en permanence l'anneau. Dès que toutes les données ont été transmises, le module de calcul est de nouveau disponible pour une autre tâche. Dans le cas ou le module de calcul doit réaliser un traitement des données (un calcul de corrélation entre deux zones de deux images), la trame **lecture motif+calcul** est envoyée au module de calcul pour qu'il se tienne prêt à recevoir la deuxième image et lance les calculs sitôt après. Dans le même temps, le module de mémorisation reçoit une nouvelle fois les trois trames de lecture mais avec des informations différentes. Vu qu'aucune trame vide n'est utilisée pour prévenir le module de contrôle de la fin de première lecture, il envoie la trame **lecture motif+calcul** sur l'anneau jusqu'à ce que le module de calcul la marque comme « lu et exécuté », puis il passe aux trames destinées au module de mémorisation. Une fois le calcul terminé, le module de calcul transmet ces résultats par l'intermédiaire d'une trame vide, et en faisant cela informe le module de contrôle qu'il est disponible.

6.2 - Parties communes dédiées à la communication

Comme tous les modules communiquent par l'anneau au moyen de trames identiques, la partie gérant cette communication est commune à toute l'architecture. De manière identique, chaque module décode ses trames et applique les actions appropriées à l'exception du module de contrôle puisse qu'il crée ces trames. La structure des parties communes des modules dédiées à la communication est montrée sur la Figure 6-7.

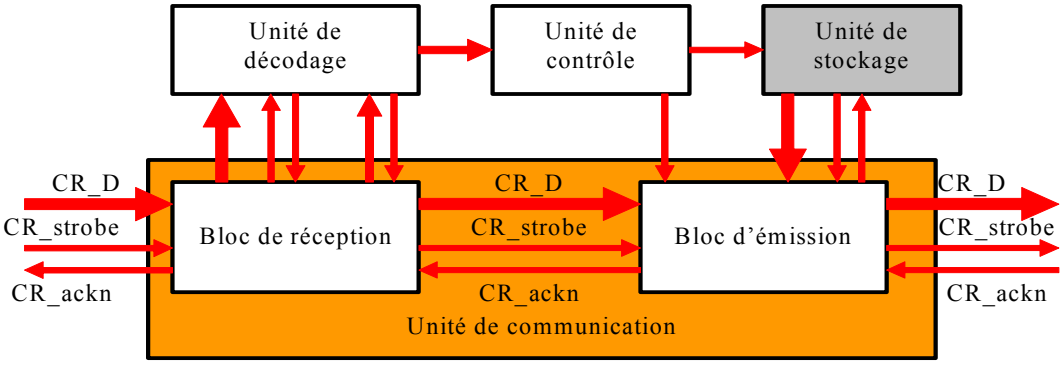


Figure 6-7 : Schéma des unités communes de communication.

Ces parties communes sont formées par quatre unités distinctes :

- l'unité de communication qui gère le flux des trames de l'anneau ;
- l'unité de décodage pour la transcription des ordres et des informations associées ;
- l'unité de contrôle qui exécute les ordres et envoie des commandes aux autres unités ;
- et l'unité de stockage pour les informations destinées aux trames vides.

Toutes ces unités ont une structure et un fonctionnement identique pour tous les modules à part leurs interfaces avec d'autres unités spécifiques au module (liaisons et leur gestion). Nous allons présenter chaque unité précitée en détail.

6.2.1 Unité de communication

L'unité de communication est divisée en deux blocs asynchrones, l'un reçoit les trames de l'anneau du module précédent, et l'autre envoie les mêmes trames au module suivant. Cette division en deux blocs (bloc de réception et bloc d'émission) est nécessaire pour assurer la réception d'un octet et la transmission d'un autre dans le même intervalle de temps. Le temps de transmission à l'intérieur du module est sensiblement réduit. En plus de leurs fonctions primaires (émission et réception), ces deux blocs ont d'autres rôles dans l'unité de communication.

1 – Bloc de réception

Sa fonction principale est de réceptionner l'octet de la trame qui arrive du module situé en amont sur l'anneau en vérifiant la bonne transmission par les signaux de *handshake* (voir 6.2.2 - Réseau d'échanges de commandes et de résultats). Si la transmission est correcte, cet octet est directement recopié dans l'unité de décodage et le bloc d'émission. La recopie dans le bloc d'émission est automatique pour chaque octet, tandis que celle dans l'unité de décodage est conditionnée par la correspondance de l'adresse de destination de la trame avec celle du module. Pour mettre en œuvre cette condition, le bloc de réception compte le nombre d'octets (modulo 6) qu'il réceptionne. Ce nombre permet de savoir quel champ de la trame est reçu. Le numéro de l'octet en cours dans la trame (de 1 à 6) est transmis à l'unité de décodage. Lors du comptage des octets dans la trame, le bloc de réception vérifie la valeur de l'octet 1. Si celle ci est nulle, c'est qu'il y a eu un problème dans la communication. En effet, le premier octet contenant l'adresse du module de destination et la commande ne peut avoir une valeur nulle même pour une trame vide. Dans ce cas de figure, le compteur de trames est arrêté jusqu'à rencontrer une donnée non nulle. Si il n'y a aucun problème, chaque premier octet est automatiquement recopié dans l'unité de décodage pour effectuer la comparaison des adresses. Si l'adresse est identique alors l'unité de décodage valide cette trame comme lui étant destinée par un signal (**adress_valid**) à le bloc de réception, sinon il invalide la trame. Dans le cas d'une trame validée, tous les octets sont recopiés dans l'unité de décodage pour être lus, sinon seul le prochain champ numéro 1 reçu sera recopié.

2 – Bloc d'émission

Ce bloc joue deux rôles important dans la communication. Le premier est d'émettre l'octet transmis par le bloc de réception au prochain module. Le deuxième est d'écrire des informations dans les octets qu'il envoie. Ce deuxième rôle ne concerne que le sixième champ des trames destinées au module et les trames vides. Pour les trames destinées au module, le bloc d'émission doit marquer le sixième octet

pour informer le module de contrôle sur l'état du module. La valeur à attribuer à ce marquage lui est indiquée par l'unité de contrôle via le signal **ressources_dispo**. Nous recensons trois marquages possibles : la trame n'a pas été lue (par défaut), la trame a été lue mais non traitée (le module est occupé, elle doit être retransmise) et la trame a été lues et traitée. Dans le cas des trames vides, le bloc d'émission effectue deux taches. La première est d'identifier les trames vides en vérifiant que leur adresse de destination correspond à l'adresse du module de contrôle. La deuxième tâche consiste à remplacer les octets des trames vides par les octets fournis par l'unité de stockage contenant les informations et les données à destination du module de contrôle et de les envoyer sur l'anneau.

6.2.2 Unité de décodage

Comme son nom l'indique, cette unité effectue le décodage des trames et envoie le cas échéant les informations nécessaires aux différentes unités du module. Son premier travail est de vérifier l'adresse de destination de chaque octet désigné comme le numéro 1 par le bloc de réception de chaque trame reçue par le module. Si l'adresse est valide, il transmet l'information au bloc de réception et le mot de commande situé dans le même octet est interprété. Les cinq octets de cette trame sont alors mémorisés temporairement au fur et à mesure de leur réception. En fonction du mot de commande, des signaux sont émis vers la ou les unités concernées pour qu'elles exécutent la commande. Le cas échéant, les données des quatre octets suivants sont dirigées vers l'unité destinataire. Toutes les opérations de cette unité sont cadencées par le numéro de l'octet de la trame. Des signaux de fin de décodage sont émis vers les unités de destination de la commande et des données. Leur positionnement temporel dépend des données associées. Par défaut, un signal de fin de décodage est généré entre le début et la fin du cinquième octet pour pouvoir appliquer le marquage adéquat sur le sixième octet avant son expédition sur l'anneau. Ce marquage est effectué par le bloc d'émission sur ordre de l'unité de contrôle du module.

6.2.3 Unité de contrôle

L'unité de contrôle est responsable du bon fonctionnement du module. Bien que commune à tous les modules, cette unité diffère d'un module à un autre de par son organisation. Seule sa fonction est identique. Cette unité prend en charge l'exécution globale de toutes les commandes. Elle gère le démarrage et l'arrêt de machines d'états des unités spécifiques au module, ainsi que l'ordre de marquage des trames et l'ordre d'envoi de trames à destination du module de contrôle, par l'intermédiaire de signaux envoyés vers les unités concernées. Ces machines d'état gèrent les opérations et leur séquencement du module. Après la réception de la commande du bloc de décodage, l'unité de contrôle envoie un signal de début à l'unité concernée par la commande et se met en attente du signal de retour signifiant la fin de l'exécution. Si cette fin d'exécution doit être transmise au

module de contrôle, l'unité de contrôle donne l'ordre à l'unité de stockage, par l'intermédiaire d'une impulsion, d'envoyer une trame d'information sur l'anneau.

Ainsi, grâce aux informations que lui envoient les autres unités, elle sait à chaque instant l'état dans lequel se trouve le module. Elle informe également à tout instant à le bloc d'émission si le module est en condition de traiter une nouvelle trame ou pas, par l'intermédiaire du signal **ressources_dispo**. Enfin, certains modules reçoivent une trame spécifique dénommée **reset**. Lors de la réception de cette trame, l'unité de contrôle donne l'ordre à toutes les unités possédant des registres de stockage de recharger leurs valeurs initiales.

6.2.4 Unité de stockage

L'unité de stockage a pour unique mission de transférer au bloc d'émission une trame contenant des informations destinées au module de contrôle. Le bloc d'émission se charge alors de l'émettre sur l'anneau lorsque une trame vide circule. Pour l'instant, seuls les modules d'acquisition et de calcul génèrent des trames à destination du module de contrôle. A terme, on peut supposer que d'autres modules pourront avoir des informations à communiquer alors cette unité est présente dans tous les modules de l'architecture (sauf le module de contrôle) même si elle est inutilisée. De plus, si cette unité est commune à tous les modules du système, c'est pour faciliter le développement de nouveaux modules.

Cette unité met en forme soit des données (image, résultat) soit des informations (fin d'un traitement, libération de ressources, etc.) arrivant d'autres unités du module. Elle génère aussi le mot de commande associé. Ce mot de commande est fonction de l'origine des données ou des informations. Elle reprend la mise en forme des trames en 6 octets et les transmet de la même façon que sur l'anneau au bloc d'émission. Le premier octet de cette trame contient l'adresse du module (module émetteur) et le mot de commande. Les 4 octets suivants servent aux données selon le format et la disposition déterminés par le mot de commande émis.

La suite de ce chapitre décrit les unités spécifiques qui composent chaque module de l'architecture Round-About. Nous avons défini l'ordre des sections en suivant le cheminement des données (acquisition, mémorisation et traitement). Pour conclure ce chapitre, nous présenterons le module de contrôle, cœur de l'architecture.

6.3 - Module d'Acquisition

Ce module réalise l'interface entre le module de mémorisation et une source d'images (caméra CCD ou CMOS) afin de mettre à disposition des modules de calcul les données dont ils ont besoin. Un module d'acquisition s'occupe que d'une unique source d'images. Dans le cas de multiple sources

(stéréovision), plusieurs modules d'acquisition sont intégrés au système en suivant les principes de l'architecture.

A partir des informations des trames qui lui sont destinées, ce module génère un ensemble de signaux de commande et de configuration nécessaires au capteur d'images pour l'acquisition des données. La récupération des données (pixels) est aussi effectuée par ce module qui peut leurs appliquer un prétraitement avant de les expédier au module de mémorisation via un bus spécifique pour y être enregistrées.

Le module d'acquisition est composé de plusieurs unités ayant chacune un rôle précis dans la suite des opérations à effectuer. En plus de l'ensemble servant à la communication, il contient trois unités spécifiques : deux pour l'interface avec le capteur d'images (l'unité de commande et l'unité de configuration) et une autre permettant la mise en forme des données (l'unité de prétraitement des données). La Figure 6-8 présentes une description schématique de l'organisation interne du module d'acquisition ainsi que les échanges de données entre les diverses unités.

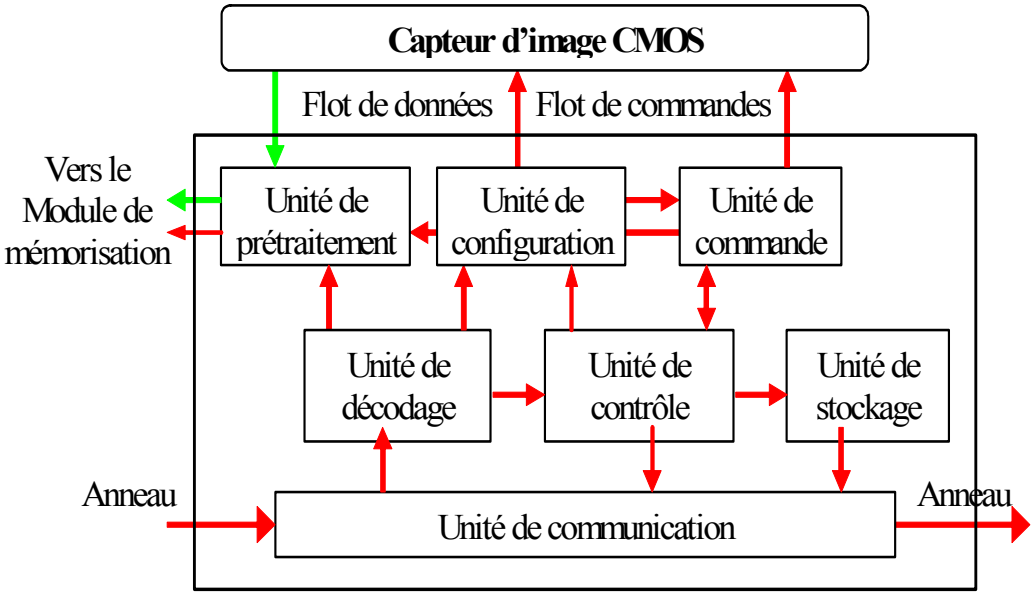


Figure 6-8 : Schéma du Module d'Acquisition.

6.3.1 Unité de commande

L'unité de commande produit tous les signaux dont à besoin le capteur d'images pour à la fois fonctionner correctement mais aussi pour fournir les images demandées. Une machine d'états gère les différentes phases du capteur (initialisation, définition des fenêtres, lecture de la matrice, etc.). Cette gestion par machine d'états permet de constituer une séquence complexe d'ordres de façon simple. Elle envoie différents signaux d'horloges, de synchronisation et d'activation tout en respectant les timings imposés par le capteur. Sur ordre de l'unité de contrôle, l'unité de commande déclenche l'acquisition d'une image en fonction des paramètres (taille et position de la ou des fenêtres) envoyés

par l'unité de configuration. A chaque fois qu'il est nécessaire, elle envoie à l'unité de prétraitement des pulses de début et de fin d'acquisition. Ils servent à différencier les pixels utiles de l'ensemble des pixels émis par le capteur. Enfin, elle signale la fin de chaque acquisition d'image à l'unité de contrôle par un pulse sur le signal **end_acqui_img**. Ce signal est utilisé par l'unité de contrôle pour dénombrer les images acquises puisque la trame **acquisition** indique le nombre d'images souhaitées. Une fois les images acquises, l'unité de contrôle passe l'ordre à l'unité de stockage d'émettre une trame de fin d'acquisition pour prévenir le module de contrôle de sa disponibilité.

6.3.2 Unité de configuration

L'unité de configuration est un ensemble de registres de stockage pour la configuration (calibrage, Offset) et les paramètres d'acquisition (taille et position de fenêtre) du capteur. Ces informations proviennent des trames de configuration transmises par l'unité de décodage. Pour cela, elle reçoit de l'unité de décodage la commande reçue et les valeurs associées. La commande permet de diriger de façon appropriée les valeurs vers les bons registres. Cela évite que des signaux à envoyer au capteur CMOS soient transmis à l'unité de commande. Tous ces signaux sont mémorisés sur ordre de l'unité de décodage. Des valeurs par défaut sont disponibles au sein de l'unité. L'utilisation des trames de configuration ne supprime pas ces valeurs, elle les remplace. Pour réinitialiser l'unité de configuration, l'unité de contrôle émet un signal d'initialisation après la réception par le module d'une trame **reset**. Ces signaux sont des états qui restent stables pendant toute la durée d'une acquisition. Il est possible de les modifier après chaque trame **acquisition**, mais, dans le cas général, un seul jeu de trames de configuration est utile pour faire plusieurs acquisitions d'images.

6.3.3 Unité de prétraitement

L'unité de prétraitement réceptionne les données fournies par le capteur. Puis, ces données sont sauvegardées temporairement avant leur envoie vers le module de mémorisation. Pour assurer une bonne réception et ne pas enregistrer des données en dehors des pixels à acquérir, l'unité de prétraitement est synchronisée par l'unité de commande qui lui fournit un signal d'horloge et des signaux de début et fin d'acquisition. Le capteur fournit l'image de manière séquentielle, pixel par pixel, ce qui permet à l'unité de ne pas avoir à sauvegarder l'image entière mais seulement les pixels à la volée. Ainsi, un prétraitement peut éventuellement leur être appliqué avant transfert vers le module de mémorisation. Actuellement, nous avons le choix entre la transmission directe des données sous le format du capteur (configuration par défaut) ou différents types de prétraitement tels que la binarisation avec un seuil paramétrable et la réduction de la quantification sur plusieurs bits. D'autres algorithmes de prétraitement peuvent être implantés mais pour notre application actuelle ceux-ci suffisent. Ces prétraitements sont activés et configurés par des informations issues de l'unité de décodage. La configuration du prétraitement est validée par un ordre de l'unité de décodage. Après

prétraitement, ces données sont envoyées au module de mémorisation. Un registre de sortie permet leur regroupement en mots de 32 bits (format du bus de transmission) et leur expédition à une fréquence fixe. En fonction du prétraitement appliqué, la fréquence d'envoie des données est variable. Pour éviter toutes pertes, l'unité de prétraitement fournie au module de mémorisation cette fréquence par l'intermédiaire d'un signal de synchronisation **W_CIS**.

6.4 - Module de Mémorisation

Ce module permet de gérer les flots de données entrant et sortant de la mémoire externe au FPGA. Cette gestion de la mémoire distribuée est déclenchée par des trames provenant du module de contrôle mais le séquencement des cycles d'écriture et de lecture est entièrement autonome.

Le module de mémorisation est composé de huit unités regroupées autour de trois tâches (Figure 6-9) :

- la communication avec l'anneau ;
- la gestion des accès en écriture et en lecture ;
- l'interface avec la mémoire externe et les bus de données des autres modules.

Ce regroupement permet une modification simple de ce module dans les cas de changement du système de communication, du type de mémoire externe ou de la gestion des cycles d'écriture/lecture. Actuellement, le module de mémorisation est interfacé avec de la mémoire de type SRAM pour *Static Random Access Memories* mais d'autres types de mémoire peuvent être envisagés.

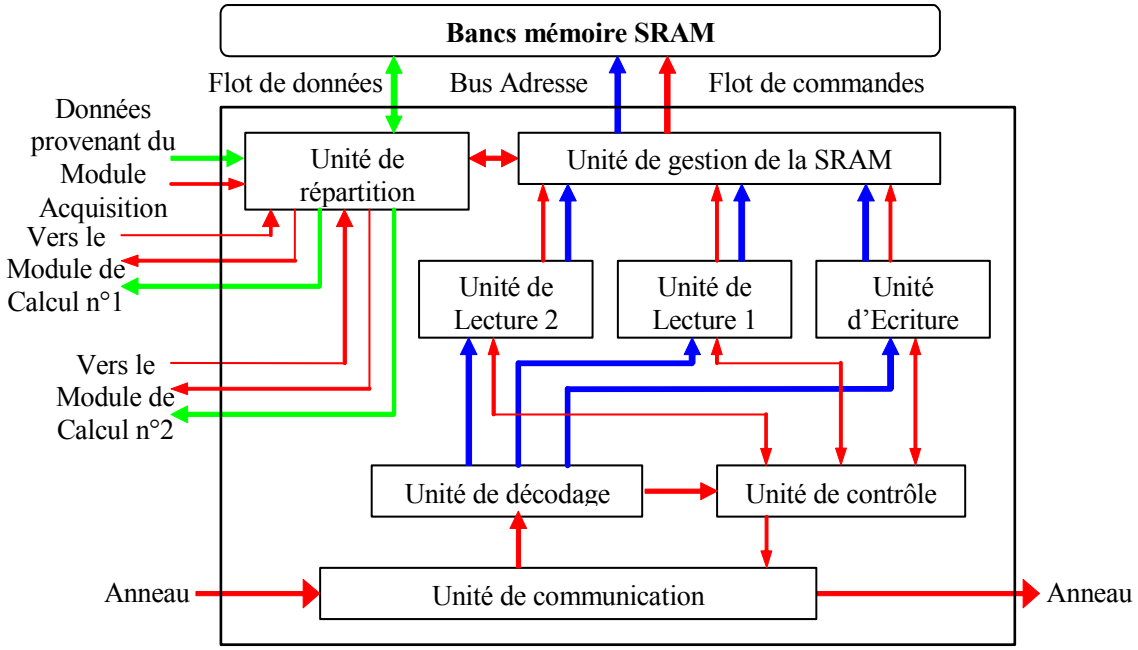


Figure 6-9 : Schéma du Module de Mémorisation.

Tout comme les autres modules, la communication est assurée par les unités de communication, de décodage, de contrôle et de stockage.

6.4.1 L'interface entre la mémoire et le système

Cette tâche incombe à deux unités : l'une gère les commandes et les adresses destinées à la mémoire SRAM et l'autre sert d'interface entre les différents bus de données et la liaison de données de la mémoire.

1 – Unité de gestion de la SRAM

L'unité de gestion de la SRAM émet les signaux de commande (écriture, lecture) en fonction du mode actif et des signaux d'état de l'unité de répartition. Elle transmet aussi l'adresse des données qui accompagne le mode actif à la mémoire. Une machine d'état, cadencée à la fréquence de la mémoire externe, organise le fonctionnement complexe de cette unité. Elle fonctionne en suivant le séquencement des opérations décrit dans le paragraphe suivant. Chaque mode est activé par un signal d'activation envoyé soit par l'unité d'écriture soit par l'unité de lecture. Mais, il n'est validé que si des données à écrire sont présentent dans l'unité de répartition ou que si les données de la lecture précédente ont bien été transférées au module de calcul. Pour ce faire, des signaux provenant de l'unité de répartition agissent comme des flags sur le déroulement de la machine d'états.

2 – Séquencement des opérations Ecriture/Lecture

L'unité de gestion de la SRAM génère en permanence, en interne grâce à sa machine d'états, un cycle Ecriture-Lecture1-Lecture2 réalisé sur 9 coups d'horloge. Ce cycle correspond à une architecture avec un module d'acquisition (Ecriture) et deux modules de calcul (Lecture1 et Lecture2) comme le présente la Figure 6-1. Si d'autres modules de ces types sont incorporés dans l'architecture alors le cycle évoluera. Par exemple, pour une architecture avec deux modules d'acquisition (stéréovision) et deux modules de calcul, nous aurons un cycle Ecriture1-Ecriture2-Lecture1-Lecture2. Le cycle d'opérations Ecriture/Lecture est nécessaire pour permettre la transparence de l'accès mémoire. Ce cycle joue le rôle de multiplexeur temporel des accès mémoire. Dans notre cas, la période de ce cycle correspond à la fréquence d'arrivée des données images qui est supérieure au temps d'accès mémoire. De ce fait, nous pouvons effectuer plusieurs opérations Lecture entre deux opérations Ecriture (Figure 6-10). Ce déséquilibre entre les opérations de lecture et d'écriture est nécessaire pour réduire le temps de repos pour les modules de calcul et ainsi les faire fonctionner en continu dès que les données sont disponibles, car ces modules ne sont généralement pas actifs durant l'acquisition des données.

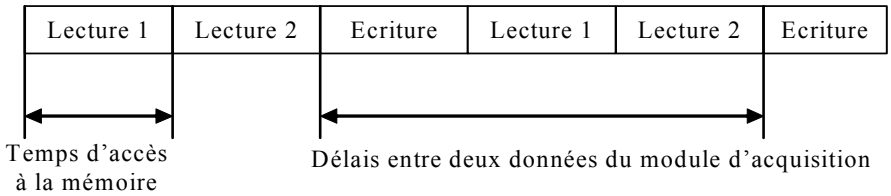


Figure 6-10 : Cycles d'écriture/lecture.

Pour limiter les contraintes de la mémoire partagée, un accès de multiplexage de temps est employé pour les opérations d'écriture et de lecture. Ceci est possible car la période d'accès à la mémoire RAM est supérieure au temps d'arrivée des données images du capteur.

3 – Unité de répartition des données

L'unité de répartition est une association entre des multiplexeurs de données et des verrous (*latchs*) pour l'orientation des données entrantes et sortantes du module et de la mémoire. Dans sa forme actuelle, l'unité de répartition est composée de trois registres et d'un multiplexeur 2 vers 1 (Figure 6-11). Pour limiter la multiplication de signaux de commande identique, le contrôle de l'unité de répartition est couplé à celle de la SRAM par l'unité de gestion de la SRAM. Cette structure permet de limiter les états de latence sur les bus de données ainsi qu'une désynchronisation temporelle entre la mémoire et les bus.

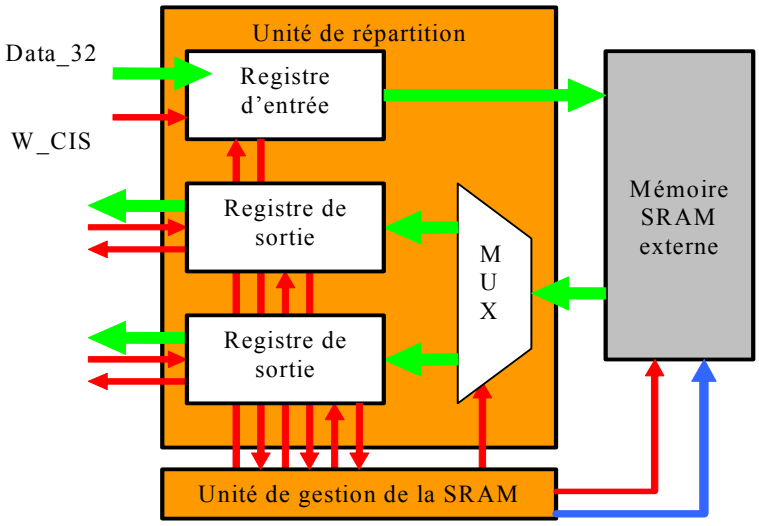


Figure 6-11 : Unité de répartition des données.

Un registre d'entrée sauvegarde les données présentes sur le bus lorsqu'il reçoit le signal de validation en provenance du module d'acquisition. La présence de données dans le registre est signalée à l'unité de gestion de la SRAM. Elles seront placées en mémoire sur ordre de cette même unité à l'adresse voulue par l'unité d'écriture. Pour les registres de sortie, le fonctionnement diffère. Il reçoit des données que si il est vide des données précédentes (un signal averti l'unité de gestion). Le transfert des données vers le module de calcul associé ne s'effectue que si ce module est prêt à les recevoir (signaux de *handshake* – voir paragraphe 6.1.1).

6.4.2 La gestion des modes écriture et lecture

La séparation des commandes lecture/écriture en unités distinctes permet de réaliser des accès mémoire aléatoires (écriture et lecture à différentes adresses pour différentes tailles de données). La

fonction principale de ces unités est l'adressage de la mémoire. Chaque unité a sa propre logique d'adresse.

1 – Unité d'écriture

L'unité d'écriture gère les paramètres de mémorisation (taille et adresse) et les ordres Ecriture pour l'unité de gestion de la SRAM. Les adresses ne sont pas transmises par l'intermédiaire de trames mais elles sont calculées après chaque accès mémoire. Pour les calculer, l'unité reçoit la première adresse et le nombre de données à écrire de l'unité de décodage. Ces deux paramètres sont enregistrés dans l'unité. Une unité d'écriture est associée à un unique module d'acquisition. Cela permet d'éviter les mélanges de données en mémoire, chaque source de données a donc une zone de mémoire qui lui est propre pour ses données. La gestion des données est alors autonome au niveau du module de contrôle.

2 – Unité de Lecture

L'unité de lecture joue le même rôle que l'unité d'écriture mais pour le mode de lecture. Chaque unité de lecture est associée avec un module de calcul ou à un groupe de module de calcul, dans les cas extrêmes. Le nombre de module de calcul est alors le même (ou un multiple) que le nombre d'unité de lecture. Cela permet d'éviter les goulots d'étranglement lors de l'accès à la mémoire externe, donc une gestion autonome des données de chaque Module de Calcul. Comme pour l'unité d'écriture, les adresses sont calculées après chaque accès mémoire. Pour les calculer, l'unité reçoit la première adresse, le nombre de données à lire et la taille du modulo de l'unité de décodage. La première adresse, le nombre de données et le modulo servent à calculer toutes les adresses de l'imagette en prenant en compte en particulier les passages d'une ligne à la suivante.

6.5 - Module de Traitement

Le module de calcul est le seul module s'occupant du traitement d'images. De ce fait, il est le module le plus sujet aux adaptations dans le cas de changement de traitement. Dans le cas de notre application, il réalise les opérations de corrélation afin de déterminer la position en X et en Y pour laquelle le nombre de pixels identique entre deux images est le plus élevé. Pour obtenir un module de calcul totalement adaptable sans avoir à reprendre sa conception en entier, nous l'avons construit en deux parties : l'une fixe (l'ensemble des communication) et l'autre flexibles (les traitements). L'organisation du module est de sept unités (Figure 6-12) : l'unité de communication, l'unité de décodage et l'unité de contrôle pour la communication avec l'anneau, l'unité de calcul et l'unité de transfert direct pour les étapes de traitement des images et l'unité d'interface qui gère les échanges de données avec le module de mémorisation. Il est à noter que ce module peut être implanté plusieurs fois dans le système (dans la limite des ressources du circuit FPGA) permettant ainsi d'accélérer le traitement en parallélisant les calculs sur différentes zones des images.

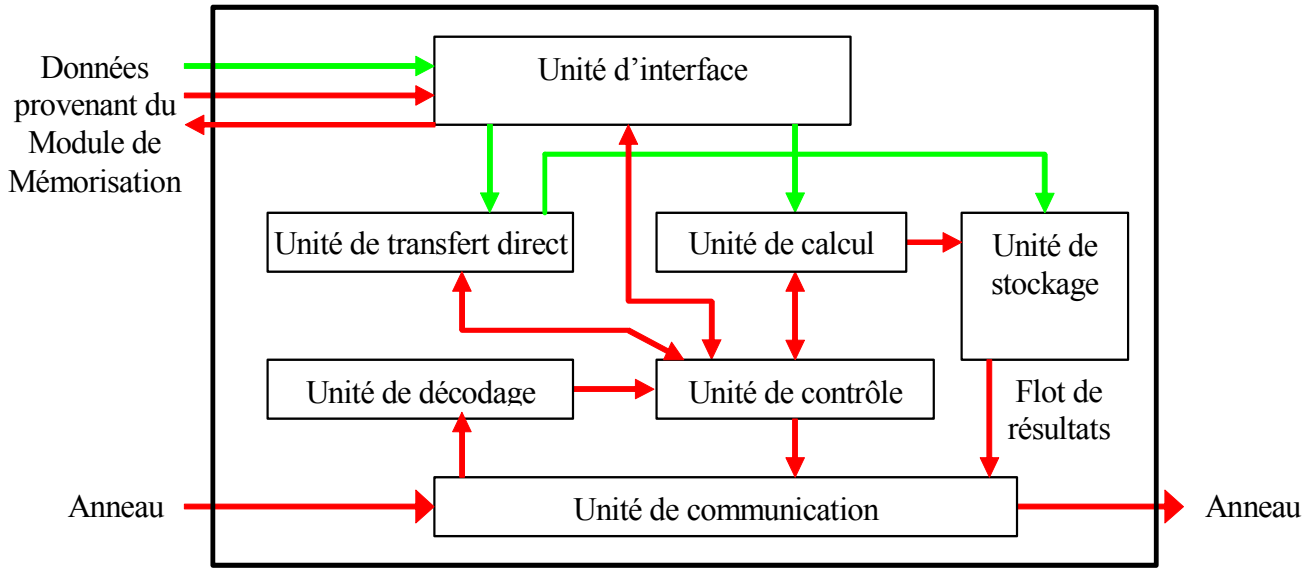


Figure 6-12 : Schéma du Module de Calcul.

Nous pouvons faire une remarque. Le module de calcul possède deux unités de traitement : l'unité de transfert direct et l'unité de calcul. L'unité de transfert direct a été conçue pour permettre une validation du bon fonctionnement de l'architecture dans une utilisation sans traitement. Puis, elle a été gardée dans la définition du module de calcul pour permettre la mise au point de la partie vision (configuration du capteur, éclairement et optiques) avant de faire des calculs.

Pour réduire le temps entre deux traitements, les résultats sont placés dans l'unité de stockage. Donc au cours d'un calcul, les résultats précédents sont stockés et envoyés de cette mémoire supplémentaire quand une trame vide se présente sur l'anneau.

6.5.1 Unité d'interface

Cette unité sert d'interface entre le module de mémorisation et le module de calcul et dirige les données en fonction des ordres provenant de l'unité de contrôle. Elle gère le bon transfert des données en utilisant le mécanisme de *handshake* (paragraphe 6.1.1). Son fonctionnement est des plus simple, cette unité est utilisée que pour un rôle de désynchronisation entre la transmission des données et leur utilisation pour les traitements. Le passage des données à travers cette unité est utilisé pour remettre en forme les données du format de communication (32 bits) au format image (8 bits en niveaux de gris ou 1 bit en binaire par pixel).

6.5.2 Unité de transfert direct

Cette unité n'intervient dans le module de calcul que pour valider la bonne réception ainsi que la bonne émission des données transitant par le module. Par conséquent, il s'agit simplement d'une recopie des données présentent à l'entrée du module de calcul (bus de données différent de l'anneau) vers l'anneau.

6.5.3 Unité de calcul

Cette unité est l'élément clé du système, elle sera présentée plus en détails dans le Chapitre 7. Nous présenterons ici que son aspect fonctionnel. Sa nature peut être différente suivant le type de traitement à réaliser et les performances à atteindre.

L'unité de calcul effectue les traitements qui lui sont implantés dans son ou ses cœurs de calcul selon les ordres fournis par l'unité de contrôle. Pour des accès rapides et simples et donc une augmentation des performances, deux mémoires internes entreposent les données en cours d'utilisation. Cette organisation permet de paralléliser au maximum les transferts avec les tâches de traitement pour limiter les temps d'attente. Tout ceci permet d'obtenir une unité de calcul dont les accès externes sont limités au strict minimum. Une fois les traitements réalisés, l'unité transmet les résultats à l'unité de stockage et signale son inactivité à l'unité de contrôle.

6.6 - Module de Contrôle

Le module de contrôle supervise les autres modules et se comporte en maître vis à vis d'eux. Il régule le fonctionnement du système. Il assure le séquencement des opérations en envoyant des commandes aux autres modules (modules de calcul, d'acquisition et de mémorisation). Il permet aussi le transfert des résultats provenant des modules de calcul vers le PC de visualisation. Le module de contrôle est composé de trois unités permettant de réaliser ces opérations (Figure 6-13).

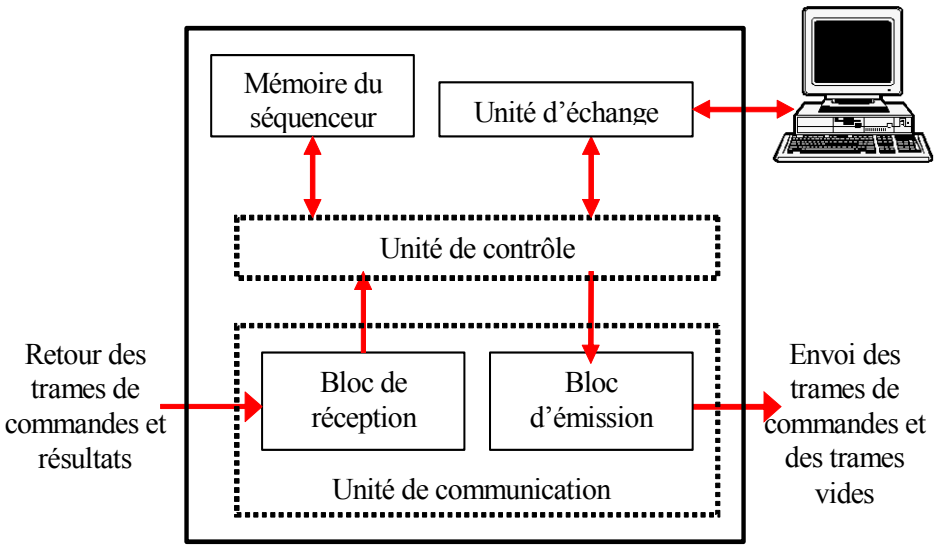


Figure 6-13 : Schéma du Module de Contrôle.

Le module de contrôle produit de nouvelles trames ainsi la structure de son unité de communication est différent des autres modules. L'anneau est ouvert et le module de contrôle doit recevoir toutes les trames émises pour vérifier la transmission correcte de la commande et pour

recevoir les résultats. L'unité de contrôle est le cœur du module. Associée à une mémoire, elle séquence l'ensemble des commandes dont ont besoin les autres modules pour fonctionner. Tandis que l'unité d'échange gère l'interface de l'architecture (donc du FPGA) avec l'extérieur du système.
La Figure 6-12 ne représente pas la réalité de l'implantation du module de contrôle dans un FPGA. Ce n'est qu'une description fonctionnelle de ce module. L'implantation réelle est présentée dans le Chapitre 7.

6.6.1 Unité de communication

L'unité de communication est légèrement différente de celles des autres modules, du fait qu'aucune trame arrivant depuis l'anneau n'est retransférée directement vers la sortie de l'anneau. Toutes les trames entrantes sont transmises à l'unité de contrôle (sauf les trames vides qui sont purement et simplement supprimées) qui les analyse, et les place éventuellement dans sa pile d'envoi selon des critères prédéfinis. Il n'y a donc aucun lien entre le bloc de réception et celui d'émission.

1 – Bloc de réception

Comme dans les autres modules, sa fonction principale est de réceptionner l'octet de la trame qui arrive et de vérifier la bonne transmission par les signaux de *handshake*. Il compte le nombre d'octet par trame (modulo 6) qu'il réceptionne tout en les mémorisant temporairement. Lors de ce comptage des octets, il vérifie toujours la valeur de l'octet 1 pour éviter les problèmes dans la communication. Si la transmission est correcte, une lecture de l'adresse est faite pour séparer les trames de commandes marquées et les trames de résultats des trames vides (adresse du module de contrôle). Si l'adresse dans la trame n'est pas identique à l'adresse du module alors la trame est recopiée dans l'unité de contrôle, sinon la trame vide est détruite. Pour avertir l'unité de contrôle de cette transmission, un signal est émit avec le premier octet. De plus, ce signal permet à l'unité de contrôle de ne pas attendre en continu d'éventuelle trames entrantes.

2 – Bloc d'émission

Ce bloc contrôle l'émission des trames sur l'anneau. Son rôle est d'incorporer les trames que lui envoie l'unité de contrôle dans le flot de trames vides qu'il génère en permanence sur l'anneau. Pour ce faire, ce bloc possède une mémoire tampon et un compteur d'octet reçu de la part de l'unité de contrôle. Deux systèmes de flag permettent l'insertion de trames dans le flot sans risquer de conflits. Le premier flag bloque toutes tentatives de nouvelle émission tant que le sixième octet de la trame actuellement émise n'est pas envoyé sur l'anneau. Et, le deuxième indique qu'une trame est disponible à l'envoie dans la mémoire tampon. La transmission entre l'unité de contrôle et le bloc d'émission suit le même schéma et protocole que la communication sur l'anneau.

6.6.2 Unité de contrôle

Cette unité est le cœur du système, le chef d'orchestre de l'architecture. Son rôle est d'envoyer les commandes sous forme de trames aux autres modules et d'attendre en retour leur réponse sous forme de commande marquée et de trames de résultats. Cette unité est en fait un séquenceur d'ordres et un interpréteur de réponses. Son démarrage ou activation peut provenir soit d'un ordre émit par l'ordinateur qui sert à la visualisation des résultats soit d'une action de l'utilisateur (pression sur un bouton). Pour éviter une conception fastidieuse, une mémoire, la mémoire du séquenceur, permet de répertorier toutes les trames de commandes dont à besoin l'architecture. Pour ne pas avoir une mémoire du séquenceur énorme, certaines trames ne sont pas répertoriées mais recalculées à chaque utilisation. Cela peut être le cas pour des trames peu utilisées ou pour des trames dont les informations sont à chaque fois différentes comme les trames contenant les adresses mémoires ou de configuration du capteur d'images.

En phase de fonctionnement, les trames stockées en mémoire du séquenceur ou recalculées sont transmises aux différents modules via l'anneau grâce au bloc d'émission de l'unité de communication. Pour cela les trames sont sélectionnées par l'unité de contrôle sur ordre du séquenceur, puis transférées aux modules via l'anneau. A l'entrée de l'anneau, le bloc de réception reçoit les trames en retour. Après vérification de leur contenu, seules les trames de commandes et de résultats sont transmises à l'unité de contrôle pour être interprétées. Dans le cas de trames de commandes, seul le sixième octet est regardé. En fonction du marquage, elle est soit retransmise à nouveau soit détruite. Dans le cas de retransmission des trames, le séquenceur les insère dans sa pile d'envoie. Cette pile stocke les trames à envoyées dans l'attente du bloc d'émission. Pour les trames de résultats, l'interpréteur en extrait le mot de commande et les données. Le mot de commande est décodé pour que l'unité de contrôle exécute les actions qui lui sont associées. Quant aux données, l'unité de contrôle les transfère au PC via l'unité d'échange.

6.6.3 Unité d'échange

L'unité d'échange ou d'interface PC réalise le dialogue avec l'ordinateur de visualisation. Ce dialogue s'effectue selon le protocole choisi lors de l'implantation. Divers protocoles sont disponibles comme la liaison série, USB (pour *Universal Serial Bus*). Bien qu'actuellement utilisée que pour envoyer les résultats au PC, l'unité d'échange permet également d'envoyer des informations à l'architecture mais surtout au module de contrôle. Cette liaison pourrait servir à la configuration des différents modules.
Côté ordinateur, un programme récupère les données pour soit les enregistrer dans un fichier, soit les afficher directement à l'écran sous forme d'image.

Chapitre 7

Utilisations et applications de l'architecture Round-About

Après la présentation de l'architecture Round-About (Chapitre 5) et sa description fonctionnelle (Chapitre 6), nous allons présenter, dans ce chapitre, les algorithmes de traitements d'images destinés à réaliser notre application en temps réel, la description de son implantation physique et ses performances, ainsi que les perspectives que nous envisageons.

7.1 - Les algorithmes utilisés

Pour réaliser un système adéquat à notre application, nous devons faire un choix d'algorithmes de traitement parmi ceux utilisés pour réaliser la mesure de vitesse par la méthode de PIV (voir Chapitre 1). Cette méthode consiste à corréler deux imagettes prises sur deux images consécutives et à rechercher le maximum de la fonction de corrélation pour déterminer le déplacement entre les deux instantanés (visualisé par un vecteur de déplacement).

L'implémentation de la corrélation d'images à niveaux de gris sur FPGA demande une grande quantité de ressources de calcul (multiplications et additions). La réduction de la quantification des images jusqu'à des images binaires simplifie cette implémentation. La corrélation binaire par comparaison est la solution que nous avons retenue.

7.1.1 Corrélation binaire par comparaison

1 – Principe

Cet algorithme peut être considérée comme une version modifiée des algorithmes d'inter-corrélation standard. En effet, lors d'une réduction de quantification à 2 niveaux, les opérations de multiplication convergent vers les opérateurs ET logique et OU exclusif. Le deuxième opérateur est préféré car il

permet une meilleure prise en compte des deux niveaux indifféremment. De plus, nous utiliserons l'opérateur OU exclusif inversé pour mieux dénombrer les pixels identiques par des additionneurs sur 1 bit. Ainsi, l'équation définissant l'opération de corrélation binaire est obtenue en remplaçant la multiplication par l'opération logique XNOR :

$$f_{cb}(i,j) = \sum_{x=0}^{M-1}\sum_{y=0}^{M-1}\left[s_1(x,y) \;\; XNOR \;\; s_2(x-i, y-j) \right] \tag{7-1}$$

Avec *i, j* Є [0, N-M]

Comme exposé dans le Chapitre 1, le calcul s'effectue sur un motif s1 (de taille M×M) extrait de la première imagette (de taille N×N) puis recherché dans la seconde imagette s2 (de taille N×N). Pour une image de taille N×N, le motif choisi a pour dimension M = N/2. Pour chaque position du motif dans l'image, une comparaison est réalisée entre les pixels du motif et ceux de la portion considérée. Lors de la comparaison de ces deux imagettes successives, l'identification du déplacement du motif se fait simplement en comptant les pixels identiques. La position du motif qui permet d'obtenir le plus grand nombre de pixels identiques correspond à la position la plus probable du motif dans la seconde image. Cette position permet de déduire directement le déplacement.

Plus précisément, chaque pixel *f(i,j)* de l'imagette de corrélation possède un niveau de gris égale au nombre de pixels identiques dénombrés pour cette position du motif dans l'imagette. La recherche du déplacement revient donc à rechercher le pixel dont le niveau de gris est le plus élevé de l'imagette de corrélation.

2 – Mise en œuvre par FPGA

La corrélation entre les pixels du motif et ceux de l'imagette est réalisée par M^2 portes logiques XNOR. Donc, le dénombrement des pixels identiques peut être effectué par un additionneur à M^2 entrées de 1 bit. Mais, un tel additionneur nécessite un grand nombre de portes logiques et ne satisfait pas les contraintes de temps réel. Pour le réaliser tout en minimisant les ressources matérielles et en respectant les contraintes, une structure pipeline est préférée. Elle consiste en un arbre d'additionneurs simples à deux entrées de 1 bit avec retenue. La Figure 7-1 présente cette structure sous forme d'une cascade d'additionneurs à deux entrées et une sortie dont la dynamique est multiplié par deux à chaque étage. Ces additionneurs sont constitués d'additionneurs 1 bit. Le nombre d'étages (*X*) se calcul en fonction du nombre de pixels au départ :

$$X = Log_2\left(M^2\right) \tag{7-2}$$

Dans le cas où *N=32* et *M=16*, nous obtenons une structure à 8 étages d'additionneurs. Cette structure en pipeline permet d'effectuer toutes les opérations de comparaison dans un même cycle d'horloge et le dénombrement des pixels identiques avec un nombre restreint de cycles d'horloge.

L'ensemble de cette structure permet de réaliser l'opération de corrélation pour une position du motif dans l'imagette. Donc, cette opération doit être réitérée pour les $(N\text{-}M+1)^2$ positions du motif afin de constituer tous les points de l'imagette de corrélation [Dubois 01a].

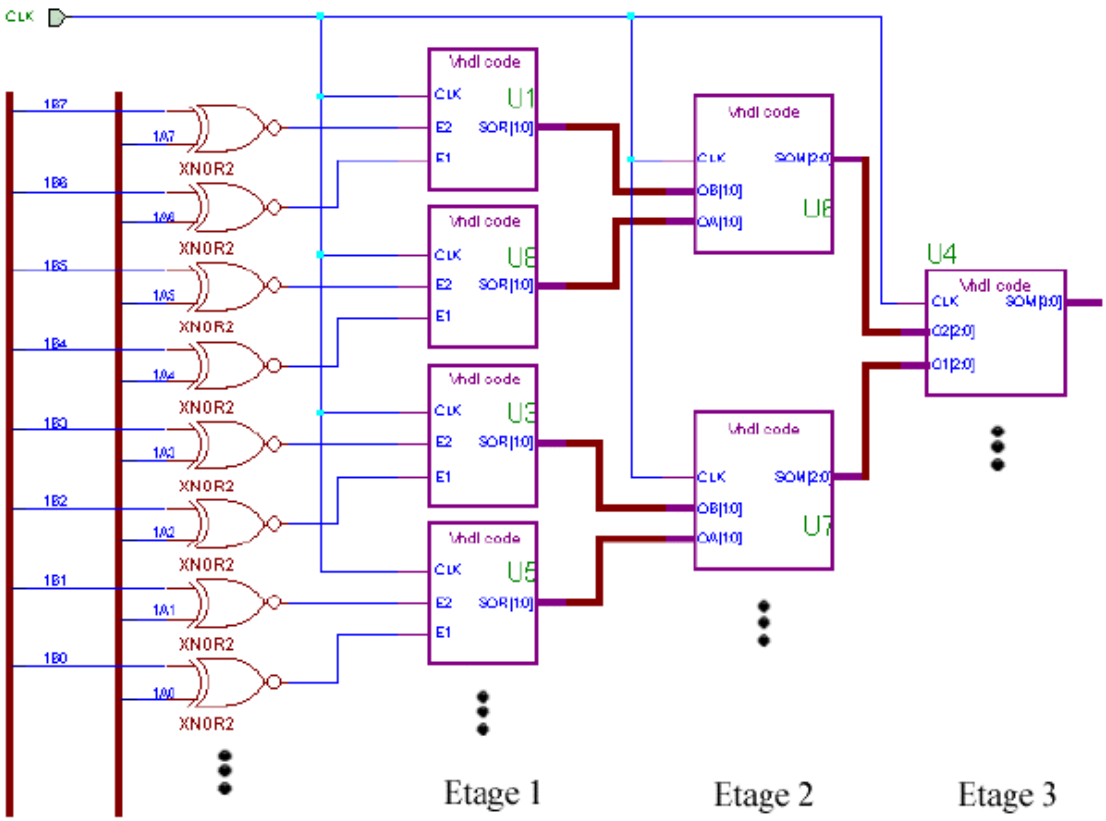


Figure 7-1: Schéma d'implémentation de la corrélation binaire par comparaison.

Cet opérateur de corrélation peut être répété plusieurs fois afin de paralléliser les $(N\text{-}M+1)^2$ calculs. Il est alors nécessaire de définir une architecture de communication à plus haut niveau pour approvisionner correctement ces opérateurs. Le détail de l'implémentation de cet opérateur au sein du module de calcul est présenté dans la section 7.3.

3 – Précision des calculs

La précision du calcul a été évaluée à partir de couples d'images de synthèse [Coudert 98, Dubois 01a, Lelong 04a, Lelong 05b] avec un déplacement sub-pixel déterminé. Chaque particule générée a un diamètre de 3 pixels et une forme gaussienne (le niveau de gris d'une particule suit une loi gaussienne sur les deux dimensions en X et Y). Ceci a été réalisé avec le logiciel de traitement d'image Wima développé au sein du laboratoire TSI de Saint-Etienne.

La mesure sub-pixel est faite par une interpolation parabolique du pic de corrélation. Des mesures sont réalisées à partir de paires d'images possédant un déplacement déterminé et pour des densités en particules de faibles à fortes. Les déplacements en X et en Y ont été choisis en fonction des erreurs liées à l'opérateur d'interpolation. Le déplacement en X (de 4,3 pixels) correspond à un

maximum d'erreur et en Y (de 2,6 pixels) à un minimum [Westerweel 97]. Une comparaison entre les différentes méthodes a été effectuée (Tableau 7-1).

Densité en particules	6 553	26 214	65 533
Corrélation directe sur des images à 256 niveaux de gris	X = 4,292 ± 0,035 Y = 2,605 ± 0,026	X = 4,270 ± 0,030 Y = 2,623 ± 0,024	X = 4,251 ± 0,030 Y = 2,640 ± 0,027
Corrélation directe sur des images binaires	X = 4,248 ± 0,125 Y = 2,637 ± 0,125	X = 4,230 ± 0,089 Y = 2,645 ± 0,102	X = 4,251 ± 0,076 Y = 2,655 ± 0,103
Corrélation par comparaison (XNOR) sur des images binaires	X = 4,243 ± 0,125 Y = 2,638 ± 0,117	X = 4,227 ± 0,089 Y = 2,645 ± 0,101	X = 4,225 ± 0,075 Y = 2,655 ± 0,102

Tableau 7-1 : Moyenne et écart type sur 256 vecteurs vitesse avec des imagettes de taille 32×32 (déplacement simulé en X = 4,3 et en Y = 2,6).

Afin d'évaluer la perte intrinsèque de précision due à la binarisation des images, la méthode de corrélation directe a été réalisée à la fois sur des images à niveaux de gris et des images binaires. Le Tableau 7-1 montre que, sur des imagettes 32×32, la corrélation par comparaison avec XNOR est moins précise que la méthode de corrélation directe réalisée sur des images à niveau de gris. Quoi qu'il en soit une précision de 0,1 pixel (bien que l'écart type soit multiplié par 4) est facilement atteinte en travaillant sur des images binaires en dépit de la perte d'information. Il est intéressant de remarquer qu'il y a peu de différence entre les deux méthodes de corrélation sur les images binaires. En fait, quelques particules peuvent disparaître après la binarisation de l'image, ce qui représente une perte d'information pour le calcul de vecteur. L'utilisation d'une imagette d'étude plus grande (64×64) peut permettre d'atténuer la perte de précision liée à la binarisation [Dubois 01a]. Celle-ci est réduite aux environs de 0.03 pixels, l'écart type est diminué d'un même facteur pour toutes les méthodes.

La précision obtenue est suffisante à de nombreuses applications de mécanique des fluides et justifie le choix de son utilisation pour développer un système embarqué temps réel.

7.1.2 Détection du déplacement

Pour une position donnée du motif, l'ensemble de ces lignes est corrélé, et le nombre de pixels identiques est déterminé par sommation. Ce nombre devient le niveau de gris d'un pixel de l'imagette de corrélation. Pour éviter d'avoir à mémoriser l'imagette de corrélation et ensuite de détecter son pixel maximum pour déterminer le déplacement, nous optons pour une méthode plus simple et facilement réalisable. Lorsque le nombre de pixels identiques a été déterminé pour une position, il est comparé avec une valeur mémorisée (soit une valeur de référence soit la plus forte valeur d'une position précédente). Si le nombre obtenu est supérieur, il remplace alors la valeur stockée. Avec cette valeur, nous mémorisons aussi les coordonnées du pixel correspondant. Cette valeur sert alors de

nouvelle référence pour les prochaines détections. Lorsque le motif a parcouru l'ensemble des positions possibles à l'intérieur de la seconde imagette, les coordonnées enregistrées, correspondant respectivement au maximum de l'imagette de corrélation, sont transmises comme résultats. La valeur de référence est remise à zéro après chaque corrélation entre un motif et une seconde imagette.

7.2 - Présentation de la plate-forme expérimentale

7.2.1 Description de la partie matérielle

Notre réalisation physique est basée sur un FPGA de type SoPC couplé à un capteur d'images de type CMOS. Ce système offre des avantages significatif : l'ensemble des signaux entre le capteur et le FPGA sont digitaux et chaque circuit peut être remplacé par une version plus performante sans engendrer des modifications importantes du système.

1 – Architecture du capteur d'images CMOS

Le capteur d'images CMOS, dénommé IBIS4 de la société FillFactory [FillFactory-web], est une matrice à pixel actif au format SXGA. Il a pour caractéristiques principales : une résolution de 1280×1024 pixels, un bon facteur de remplissage de 60%, et une taille de pixel de 7 µm. Bien que moins performants (en terme de fréquence image) que d'autres capteurs actuels, il possède des caractéristiques adaptées à notre application (qualité d'image équivalente à un capteur CCD, faible bruit et contrôle des gains et offsets). Ce capteur est composé de plusieurs éléments dont une matrice d'éléments photosensibles avec son circuit de lecture, un amplificateur de gain de sortie programmable, et un convertisseur analogique numérique (CAN) sur 10 bits *on-chip* (Figure 7-2). La fréquence d'acquisition des images est de 7 images pleine trame par seconde. Pour une matrice plus petite, 320×256 pixels, la fréquence image est de 112 images par seconde.

Trois registres composent le circuit de lecture de la matrice : deux pour les lignes (axe Y) et un pour les colonnes (axe X). Ils agissent comme des pointeurs pour désigner le pixel dont la valeur va être lue. Les deux registres Y fonctionnent de manière identique mais indépendante et pour un rôle différent. Ils servent de *shutter*, fonction de contrôle du temps d'intégration comme un obturateur d'appareil photographique. Ce type de *shutter* est appelé *shutter* roulant ou *Electronic Rolling Shutter* (ERS). Son principe est simple, il intègre la lumière sur une à plusieurs lignes pendant la lecture d'autres lignes. Un registre désigne la ligne qui va être activée pour la lecture. L'autre désigne la ligne dont les pixels vont être remis à zéro. Le décalage, en nombre de lignes, entre ces deux pointeurs représente le temps d'intégration des photons par les pixels. Ce temps d'intégration est, donc, un multiple du temps de lecture d'une ligne de l'image. Quand le mode ERS n'est pas activé, le temps d'intégration est égal au temps de lecture d'une image.

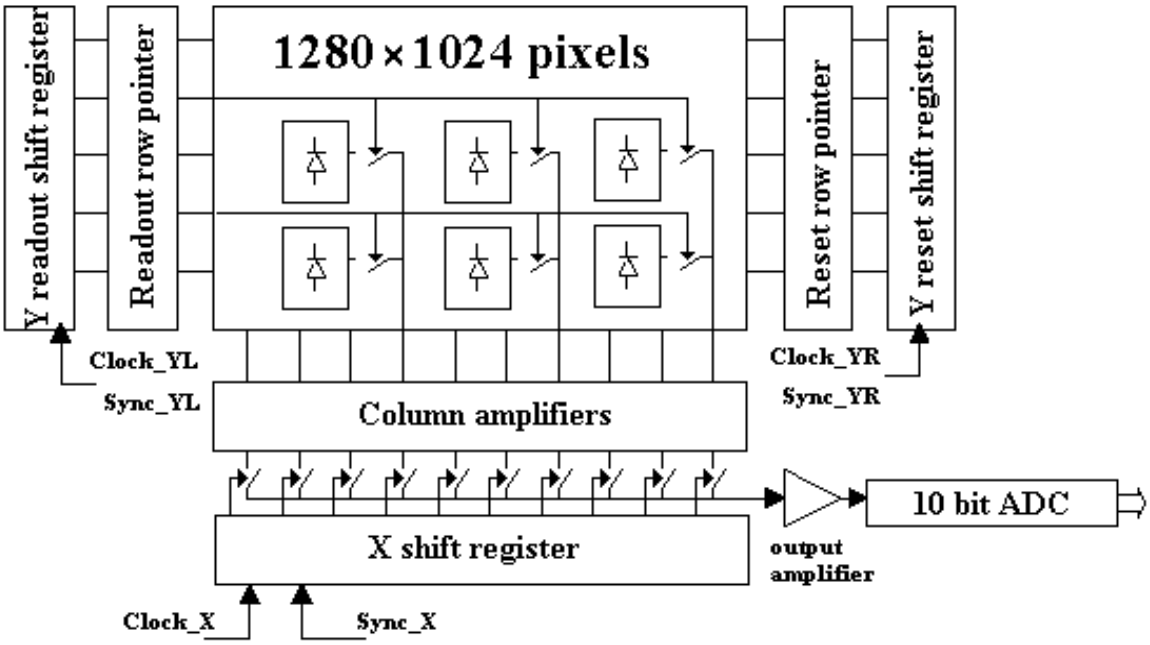


Figure 7-2: Schéma du capteur d'image IBIS4 (extrait de la *datasheet*).

2 – Architecture des circuits FPGAs

Pour ne pas devoir faire la fastidieuse étape de développement et réalisation de cartes électroniques à base de FPGAs, nous avons préféré l'utilisation d'une plateforme de développement SoPC-FPGA de la société Altera [Altera-web]. Il s'agit de la carte de développement Excalibur équipée d'un circuit FPGA APEX EP20K200 (200 000 portes logiques). Elle a été développée pour l'implantation de cœurs de processeur logiciel microprogrammé : NIOS. Ce processeur embarqué est détaillé dans le paragraphe 7.1.2.

Cette plateforme est développée autour d'un FPGA de la famille APEX couplé à de nombreux autres circuits numériques. Nous ne détaillerons pas tous les circuits présents sur la carte, nous n'exposerons que ceux que nous utilisons. Pour réaliser notre prototype, nous utilisons la mémoire externe associée au FPGA ainsi que les connecteurs permettant l'ajout de cartes filles. La carte possède 256 Ko de RAM statique, connectée directement au FPGA, découpée en mot de 32 bits avec un adressage sur 16 bits.

Les FPGAs de la famille APEX ont une architecture de type hiérarchique sur trois niveaux. Sur la Figure 7-3 apparaissent les éléments logiques de haut niveau (hiérarchiquement parlant) les MegaLAB (pour *Mega Logic Array Blocks*) et les ressources de routages propres à ce niveau qui permettent de relier les MegaLAB entres eux et avec les blocs d'entrées sorties (I/O). Les MegaLAB sont composés d'éléments plus petits qui sont les LAB (pour *Logic Array Blocks*) et d'un réseau d'interconnexions locales pour les échanges entre LAB et avec l'extérieur du MegaLAB. Il y a entre 10 et 24 LAB suivant les modèles dans un MegaLAB.

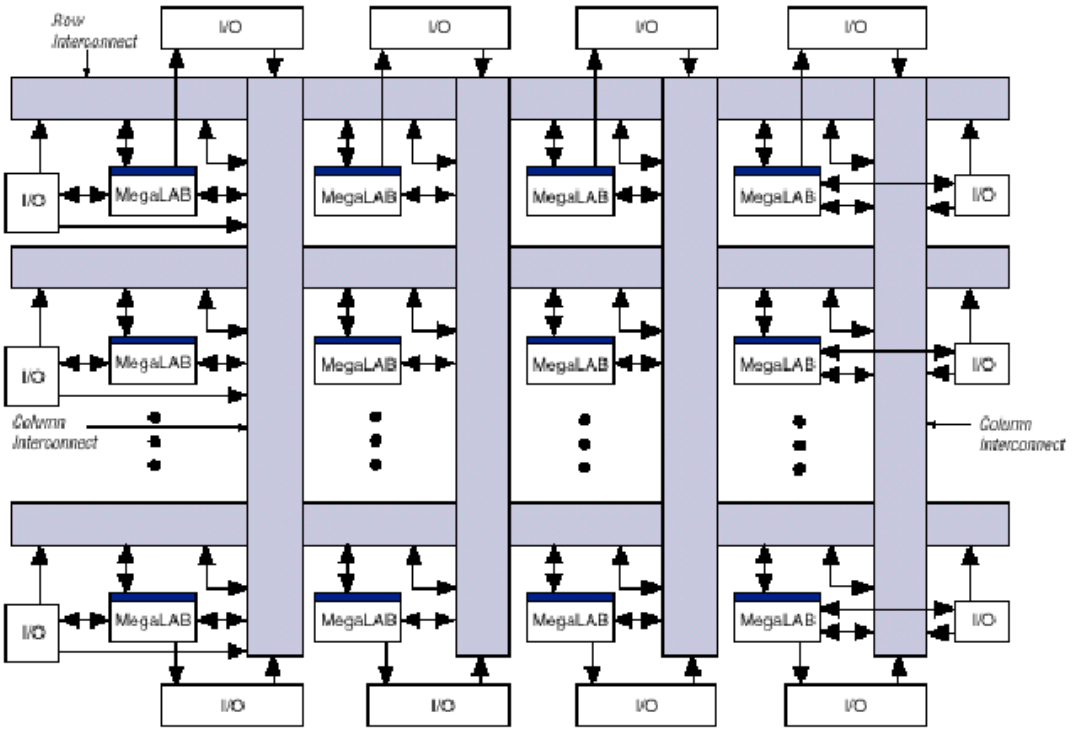


Figure 7-3 : Architecture hiérarchique d'un FPGA APEX [Altera-web].

Chaque LAB est lui-même constitué de 10 cellules logiques de base les LEs (pour *Logic Elements*) et d'un module ESB (pour *Embedded System Block*). Les éléments logiques (LEs) sont réalisés autour d'une LUT à quatre entrées, d'un registre de sortie, d'une chaîne de propagation rapide de la retenue et d'une logique de validation de l'horloge. Les modules ESB ont pour rôles de remplir deux missions : mémoires et synthèse de CPLD. De ce fait, le FPGA APEX possède un potentiel en :

- LUT : procédé de synthèse logique efficace pour l'arithmétique ;
- CPLD : procédé de synthèse efficace pour les machines d'états ;
- RAM : très utile en traitement du signal et pour les architectures pipelines.

L'APEX EP20K200EFC484-2X est composé de 8 320 Logic Elements soit l'équivalent de 200 000 portes logiques et un total de 104 Ko de RAM sous forme de modules ESB répartis dans chaque MegaLAB.

Au début de l'année 2005, nous avons réalisé l'évolution du prototype en remplaçant la plateforme SoPC-FPGA par une version supérieure. Cette nouvelle plateforme est une carte de développement NIOS II spécialement conçue pour le développement d'applications utilisant le processeur d'Altera NIOS. Elle comporte un FPGA Stratix II EP2S30 associé à 1 Mo de *Static* RAM et 16 Mo de *Single-Data-Rate* SDRAM. Ce changement n'a demandé aucune modification de l'architecture ou du système. Il a été fait pour permettre l'utilisation de la version supérieure du processeur embarqué NIOS nommé NIOS II et de nouvelles possibilités matérielles comme la communication Ethernet, de la mémoire SDRAM et une plus grande fréquence de fonctionnement.

Les FPGAs de la famille Stratix II ont une architecture de type hiérarchique sur trois niveaux comme les APEX mais suivant une conception différente. Sur la Figure 7-4 apparaissent les éléments logiques de haut niveau (hiérarchiquement parlant) les LABs (pour *Logic Array Blocks*). Ces LABs forment une matrice d'éléments logiques dans laquelle sont insérées des colonnes de blocs mémoire (M512 RAM, M4K RAM et M-RAM blocks) et de blocs DSP (pour *Digital Signal Processing*).

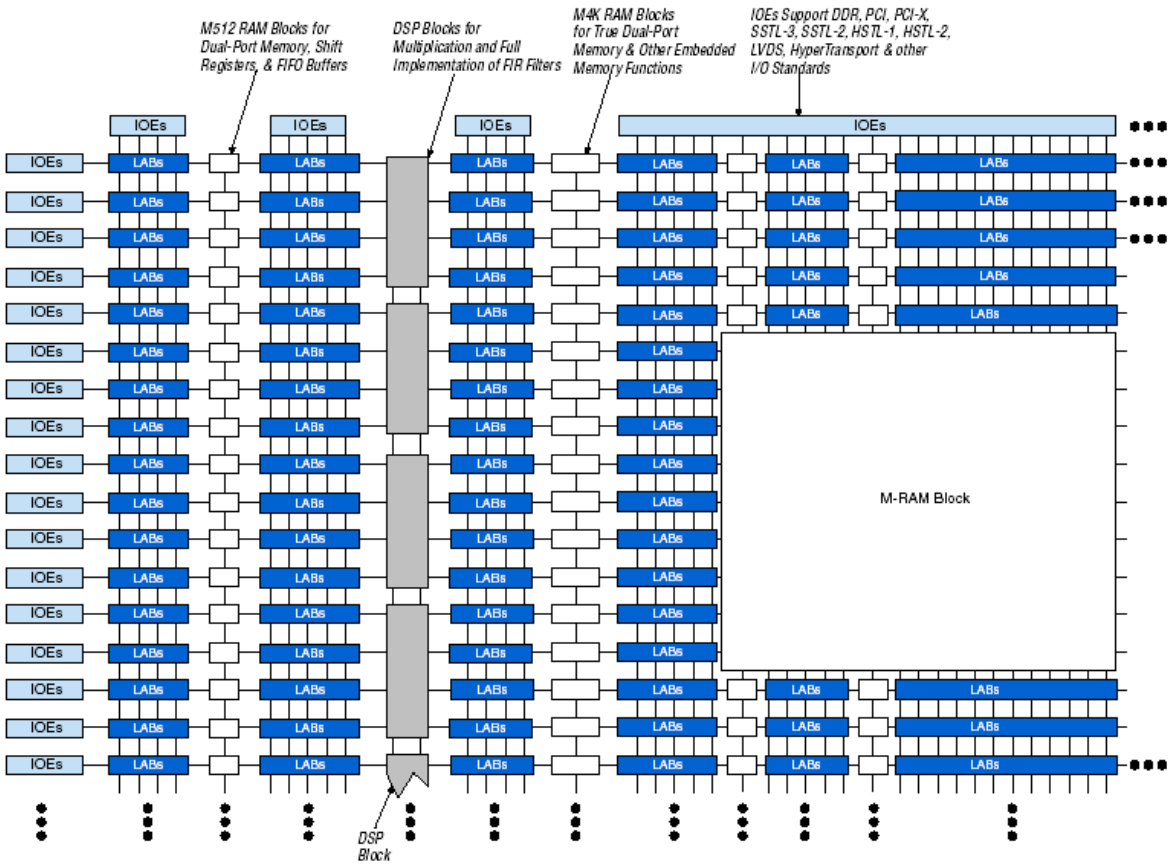


Figure 7-4 : Architecture hiérarchique d'un FPGA Stratix II [Altera-web].

Tout ceci est relié à des éléments regroupent les pattes d'entrée/sortie (IOEs) par un maillage d'interconnexions denses et rapides. Chaque LAB est composé de huit éléments plus petits, les ALMs (*Adaptive Logic Modules*), avec des ressources de routage locales vers les éléments externes (mémoires et DSP) mais aussi internes (entre ALM) du même LAB. Des réseaux de connexions distincts entre ALMs sont dédiés à la propagation de la retenu, à des chaînes arithmétiques et de registres partagées. Les ALMs sont réalisés autour d'une LUT à huit entrées, deux registres de sortie programmable, deux additionneurs complets ainsi que les réseaux de connexions précédents. La famille de FPGA Stratix II est performante pour l'implémentation d'applications nécessitant de grandes vitesses de travail alliées à des capacités arithmétiques importantes (multiplieurs, chaîne de propagation de retenue, etc.) comme le traitement du signal.

Le Stratix II EP2S60F672C5ES contient 24 176 ALMs, soit l'équivalent de 48 352 LEs, couplés à 36 blocs DSP. L'ensemble des blocs mémoire représente un total de 310,6 Ko.

7.2.2 Description de la partie logicielle

Le choix d'utiliser une plateforme SoPC-FPGA a été fait aussi pour ces possibilités d'implantation de processeur *softcore* embarqué dans le FPGA : le processeur NIOS de la société Altera.

1 – Architecture du processeur NIOS

Il s'agit d'un processeur de logiciel implanté sous forme de composant VHDL, donc par définition un processeur *softcore*. Comme tout processeur *softcore*, il est lié aux circuits FPGAs de la société Altera et ne peut être utilisé que avec les familles de FPGAs dédiées à la conception SoPC. Pour développer et programmer un processeur NIOS, la société Altera nous fournit une suite de logiciels (SOPC Builder) allant de l'environnement de développement de programme C/C++ à l'interface de programmation et débugage une fois le composant NIOS implémenté dans le FPGA.

De manière générale, le processeur NIOS se présente sous la forme d'un fichier *Top-Level Entity* regroupant tous les éléments, décris en langage VHDL, nécessaire à son fonctionnement. La Figure 7-5 expose l'ensemble des composants basiques d'un fichier *Top-Level Entity* obtenu pour un processeur NIOS standard. Le processeur NIOS peut être se composé d'un ou plusieurs CPU, de mémoires internes, d'un timer, de périphériques et de contrôleurs d'entrées/sorties variés.

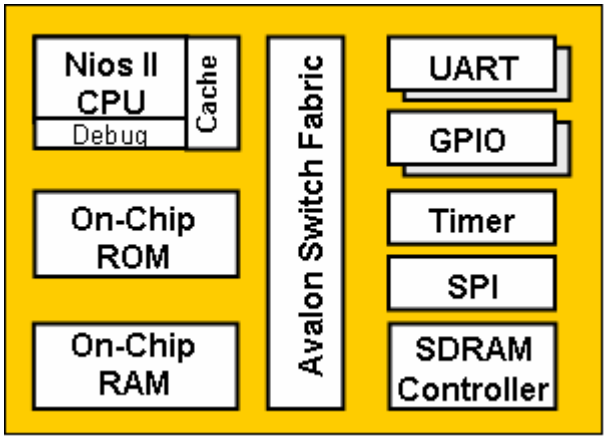


Figure 7-5 : Schéma blocs d'un processeur NIOS [Altera-web].

Le cœur du processeur NIOS est un microprocesseur à plusieurs niveaux de pipeline avec un jeu d'instructions réduit RISC. Il possède une architecture de type Harvard de 32 ou 16 bits avec un jeu d'instructions sur 16 bits. La Figure 7-6 expose l'organisation générale d'un cœur de processeur NIOS (ou NIOS CPU). Il est à noter que ce cœur dispose de mémoires caches séparées pour les instructions et pour les données dont la taille est configurable, d'un contrôleur d'interruption, d'un contrôleur d'exception, de registres et d'un ALU (unité arithmétique et logique ou *Arithmetic Logic Unit*).

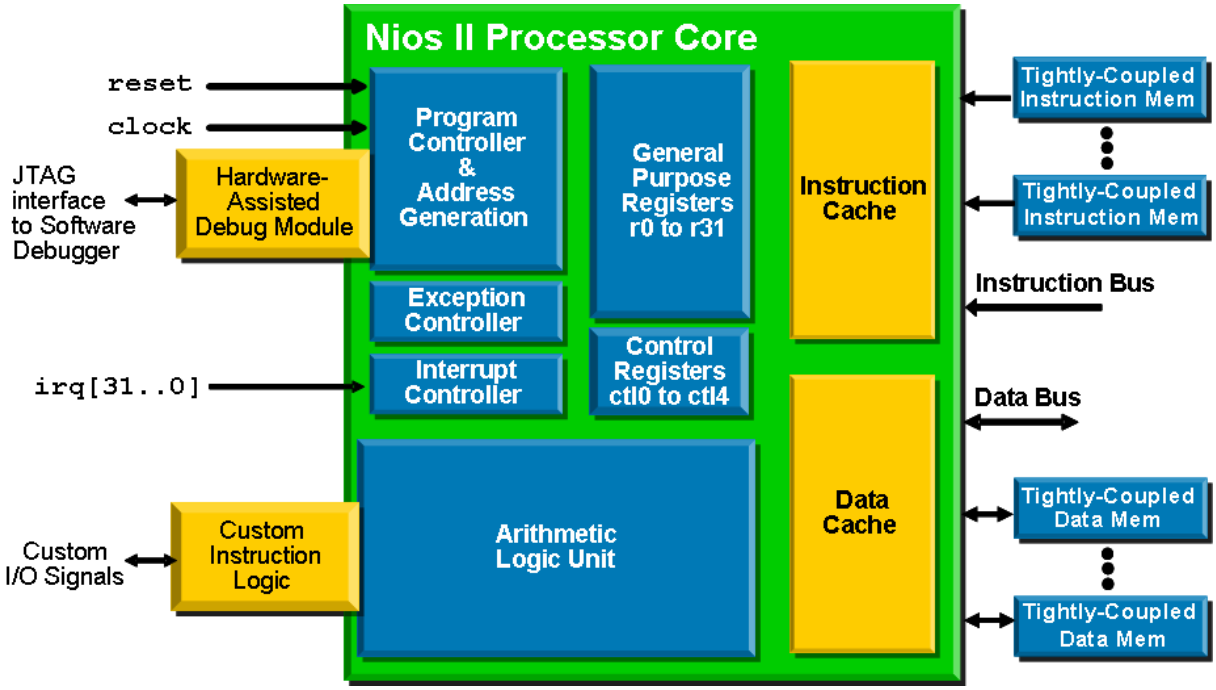


Figure 7-6 : Cœur du microprocesseur NIOS [Altera-web].

Il y a différents types de cœur de processeur NIOS envisageables. Ces possibilités vont du processeur rapide (jusqu'à six étages de pipeline) et performants mais gourmand en ressources matérielles jusqu'au petit processeur peu gourmand exécutant des fonctions simples. L'occupation des ressources matérielles dépend de ce choix, fait par le concepteur. En moyenne, un cœur peut occupé de 400 à 1800 éléments logiques pour des performances comprises entre 20 à 200 DMIPS pour des fréquences de travail comprises entre 125 et 175 MHz.

2 – Avantages et inconvénients

Le processeur embarqué NIOS présente des avantages significatifs pour notre système SoPC pour réaliser l'unité centrale du module de contrôle. Cette unité doit contrôler et gérer l'ensemble du fonctionnement du système. L'utilisation d'un tel processeur accentue l'aspect de modularité de l'architecture du fait de la facilité d'intégration et de modification de ces fonctionnalités (ajout d'éléments de même nature, programmation en langage C, etc.). De plus, des modifications du processeur n'entraîne pas obligatoirement des modifications de liens avec le reste de l'architecture, donc une éventuelle re-synthèse du système (gain de temps de conception). L'utilisation d'un processeur embarqué permet aussi de s'affranchir des latences engendrées par les communications entre un FPGA et un microprocesseur externe. Enfin, l'augmentation des fréquences de fonctionnement des FPGAs devrait rendre plus rapide l'exécution des instructions du processeur NIOS.

Cette solution est intéressante en terme de coût pour l'élaboration d'un prototype mais ce n'est pas forcément le cas lorsqu'il s'agit de réaliser une production (coût d'utilisation, performances globales, etc.). De plus, le processeur NIOS est un processeur propriétaire donc soumis à des droits de

licence et d'utilisation industrielle ce qui le rend beaucoup moins attractif dans le cas d'une production série. Enfin, point important, la compilation d'un processeur NIOS dure entre dix et quinze minutes d'où la nécessité de bien réfléchir aux modifications à apporter au programme avant de lancer une compilation.

Nous évaluerons le coût de l'utilisation d'un processeur NIOS dans l'architecture Round-About dans la section 7.4 – Résultats et performances.

7.3 - Implémentation matérielle des modules

Pour suivre les concepts de l'architecture, le système a été conçu dans une méthodologie de conception en blocs IP regroupés dans des modules (Figure 7-7), l'ensemble étant décrit en langage VHDL dans une optique de portabilité. Ceci permet la réutilisation de la partie principale de la conception sur différentes cibles.

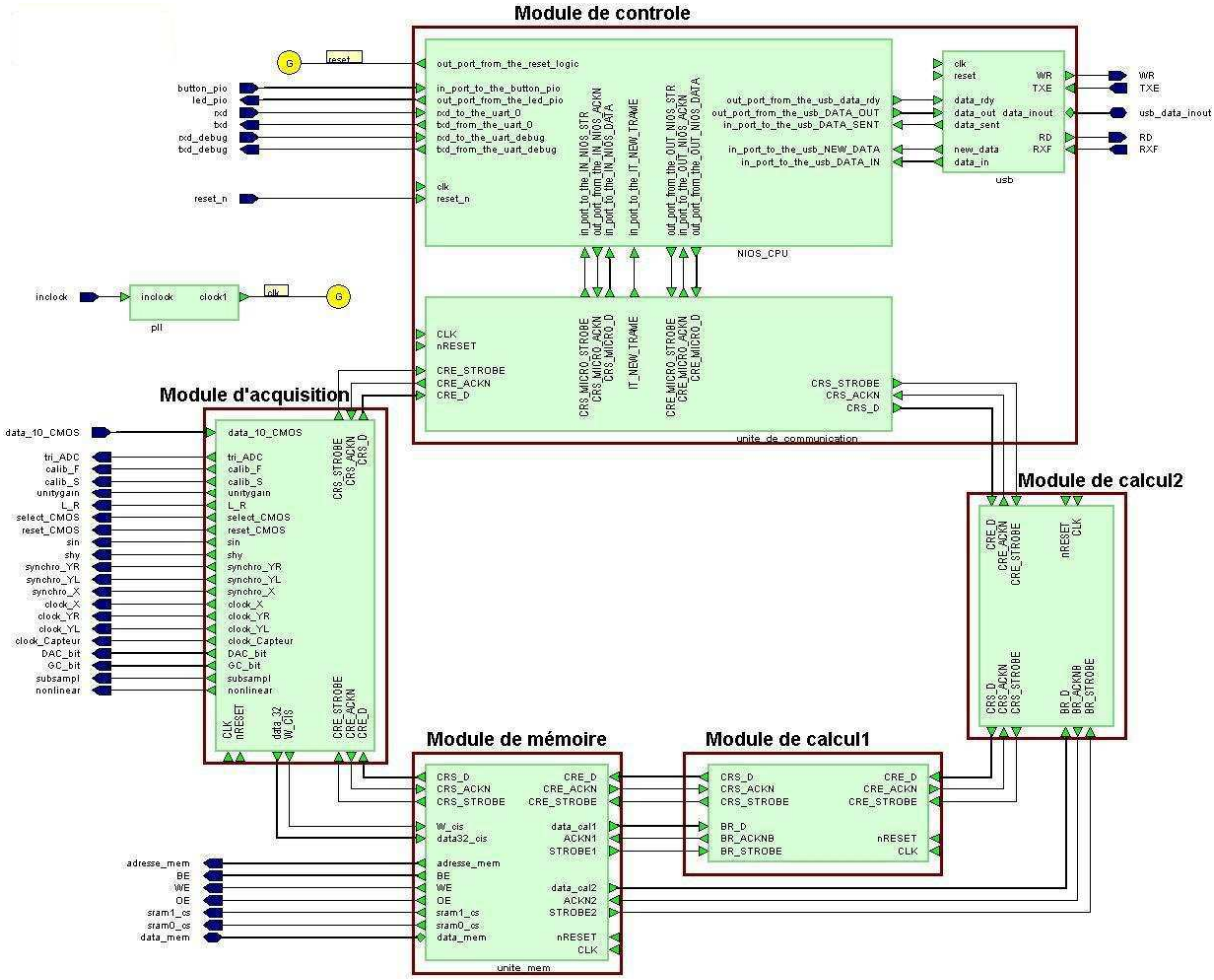


Figure 7-7 : Schéma blocs de l'architecture Round-About obtenu sous HDL Designer de la société Mentor Graphic [Mentor-web].

Le portage des descriptions fonctionnelles (Chapitre 6) en VHDL ne pose aucun problème conséquent sauf pour deux modules : les modules de calcul et de contrôle. Nous présentons, dans cette

section, les points posant problème. Pour le module de calcul, nous détaillerons l'implémentation de l'unité de calcul. Quant au module de contrôle, nous exposerons l'ensemble unité de contrôleur, mémoire du séquenceur et l'unité d'échange.

7.3.1 Module de Calcul

Au sein des modules de calcul, la seule unité sujette à des modifications en fonction des algorithmes est l'unité de calcul. Son fonctionnement et son implémentation dépendent des algorithmes, des contraintes de l'application et des possibilités du matériel.

Pour faciliter sa réalisation au sein de l'unité de calcul, la méthode de mesure basée sur les algorithmes de la section 7.1 est décomposée en plusieurs étapes :

- le stockage d'un motif et d'une imagette ;
- le positionnement du motif dans l'imagette ;
- la comparaison des pixels entre le motif et une portion de l'imagette et la sommation des pixels identiques ;
- la détection du déplacement par la détermination du maximum de la fonction de corrélation.

Nous réaliserons l'implémentation de la comparaison de motif de taille 16×16 pixels avec un imagette de taille 32×32. Cette unité est décomposée en trois blocs : le bloc de cache, le bloc de mesure et le bloc de supervision des calculs (Figure 7-8). Les blocs de cache et de mesure sont découpés en petites entités réalisant une fonction précise.

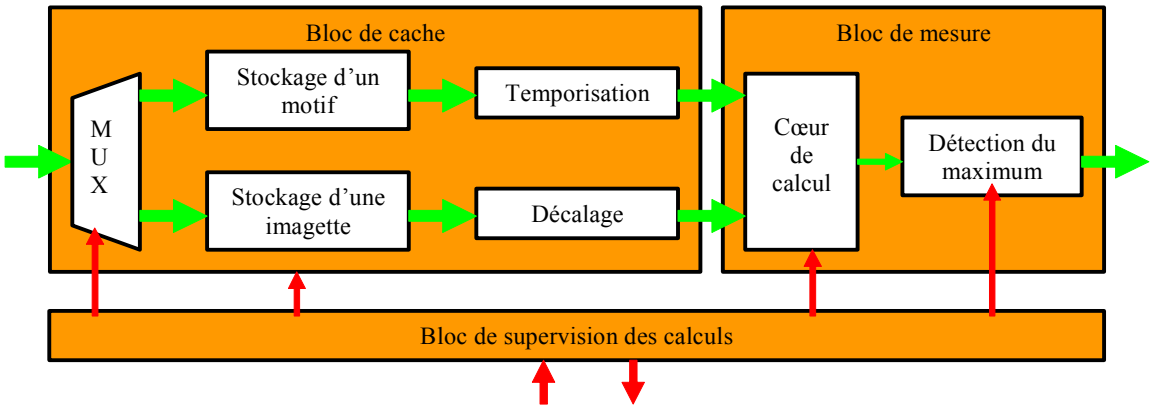


Figure 7-8 : Schéma de l'unité de calcul.

1 – Bloc de supervision des calculs

Le bloc de supervision des calculs est une machine d'états qui supervise le déroulement des opérations pour réaliser la comparaison d'un motif et d'une imagette. La machine d'état permet de synchroniser ces diverses opérations nécessaires au calcul de la position du maximum de vraisemblance entre le motif et l'imagette. Il reçoit les commandes de l'unité de contrôle du module de calcul comme enregistrement d'un motif, d'une imagette et de lancer les calculs. Il signale à cette même unité de

contrôle qu'il a bien enregistré les données et sa disponibilité pour d'autres calculs. Le bloc de supervision permet trois opérations :

- générer les adresses de lecture et d'écrire dans deux mémoires internes ;
- commander l'opérateur qui réalise le décalage ;
- commander la synchronisation de cœur de calcul et la détection du maximum.

2 – Bloc de cache

Ce bloc est subdivisé en cinq entités afin de gérer le flux de données de façon efficace et de permettre le déplacement du motif dans l'imagette. Ce bloc a pour fonction d'approvisionner en données les calculs. Pour ce faire, il dispose de deux mémoires caches réalisées par des registres, l'une pour le motif de taille 16×16 pixels et l'autre pour l'imagette 32×32. Ces deux mémoires sont décrites en VHDL. Elles peuvent donc être implémentées dans un FPGA en utilisant soit des éléments logiques soit de petites mémoires RAM internes (option de synthèse). Ce choix dépend des ressources matérielles disponibles mais aussi des vitesses de calcul obtenues avec l'une ou l'autre des solutions avec un type de FPGA donné. Ces résultats de temps sont fortement dépendants de l'architecture interne du FPGA cible. L'arrivée des données étant séquentielle, un multiplexeur, piloté par le bloc de supervision, oriente le flux de données vers sa mémoire de destination. Une fois les données du motif et de l'imagette stockées, le bloc de cache fournit ces données ligne par ligne au bloc de mesure. Seules les données nécessaires aux calculs d'une position du motif dans l'imagette sont acheminées. Pour ne transmettre que la zone de l'imagette correspondant au déplacement du motif, une entité gère le décalage de l'adressage de cette zone de l'imagette en fonction des ordres du bloc de supervision. Ce décalage engendre un retard des données de l'imagette. Pour que toutes les données arrivent en même temps au bloc de mesure, les données du motif sont retardées d'un nombre de cycle équivalent.

3 – Bloc de mesure

Il est composé du cœur de calcul et de la fonction de détection du maximum. Ce bloc permet d'effectuer la mesure de déplacement pour une position du motif dans l'imagette.

Le cœur de calcul Le cœur de calcul réalise la corrélation binaire par comparaison d'un motif, extrait d'une première imagette binaire, avec une seconde imagette pour une position donnée. Il s'agit de l'implémentation de l'algorithme adapté aux FPGAs du paragraphe 7.1.1.

Dans ce précédent paragraphe, nous avons exposé l'adaptation de l'algorithme de corrélation binaire par comparaison aux FPGAs de manière générale. Elle requiert $(N-M+1)^2$ étapes pour réaliser le calcul de la corrélation binaire pour un couple motif+imagette. Cette méthode réalise tous les calculs pour M^2 pixels en même temps. Mais, cela demande une grande quantité de ressources matérielles et engendre des latences au vue du nombre de calculs interdépendants à faire en parallèle. Pour répondre au mieux aux contraintes d'économie des ressources matérielles et de traitement en temps réel, ce calcul a été décomposé en $M\times(N-M+1)^2$ étapes. C'est-à-dire que pour chaque position

du motif dans l'imagette, la comparaison est effectuée ligne par ligne (soit *M* lignes par position) et non avec le motif entier. Nous réduisons ainsi la quantité de ressources matérielles nécessaires pour un calcul et augmentons le parallélisme des calculs puisque plusieurs couples de lignes peuvent être comparés dans le même intervalle de temps. Seul, l'accumulation des valeurs de sorties (une addition) prendra plus de temps en fonction du nombre de lignes comparées.

Pour mieux comprendre son fonctionnement, la Figure 7-9 détaille le fonctionnement de la comparaison ligne à ligne.

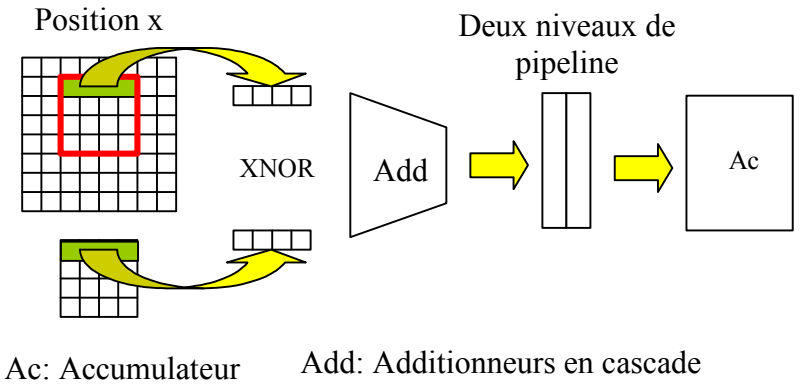


Figure 7-9 : Comparaison de lignes du motif et de la seconde imagette à deux instants successifs.

Chaque pixel d'une ligne est comparé en même temps. Puis, les résultats de ces comparaisons sont additionnés. Une fois les additions terminées, la valeur obtenue est transmise a un accumulateur et une nouvelle comparaison de ligne est lancée. L'accumulateur détermine la valeur du nombre de pixels identiques pour cette position du motif en additionnant les valeurs obtenues pour chaque ligne de *M* pixels du motif. Deux registres sont mis en série pour diminuer le temps entre deux comparaisons, tout en laissant l'accumulateur faire son travail. Lorsque la comparaison est terminée, l'accumulateur transmet la valeur du nombre de pixels identiques à la détection du maximum.

Détection du maximum Cette entité effectue la comparaison entre le nombre de pixels identiques pour une position et la valeur qu'il stocke. Lorsque ce nombre est le plus grand, il désigne la position du maximum de vraisemblance entre le motif et l'imagette. La position en X et en Y du motif au sein de l'imagette est alors mémorisée. Cette position est donnée par rapport au coin supérieur droit de l'imagette. Lorsque toutes les positions du motif dans l'imagette ont été parcourues, la position du maximum est alors transmise à l'unité de stockage pour être envoyée au module de contrôle.

7.3.2 Module de Contrôle

Le module de contrôle se présente comme un microprocesseur qui gère les communications sur l'anneau et les échanges entre l'architecture, donc le système, et l'extérieur. La description fonctionnelle du Chapitre 6 présente les fonctions du module de contrôle en dehors de toute implémentation physique. Au vue de cette description, le module de contrôle peut être réalisé avec un microcontrôleur associé à de la mémoire pour sa simplicité de programmation (langage C) et sa gestion des communications par interruptions.

L'évolution technologique de la microélectronique permet de disposer aujourd'hui de FPGAs de plus en plus puissants et pouvant intégrer des coeurs de microprocesseur au sein de leur matrice d'éléments logiques. Nous avons opté pour cette solution puisque nous souhaitons réaliser un système embarqué de type SoC. La Figure 7-10 présente une description schématique de l'implémentation réalisée. Le module de contrôle est toujours composé de trois unités mais l'unité de contrôle est un processeur *softcore* embarqué (le processeur NIOS) au sein du FPGA comme un composant décrit en langage VHDL.

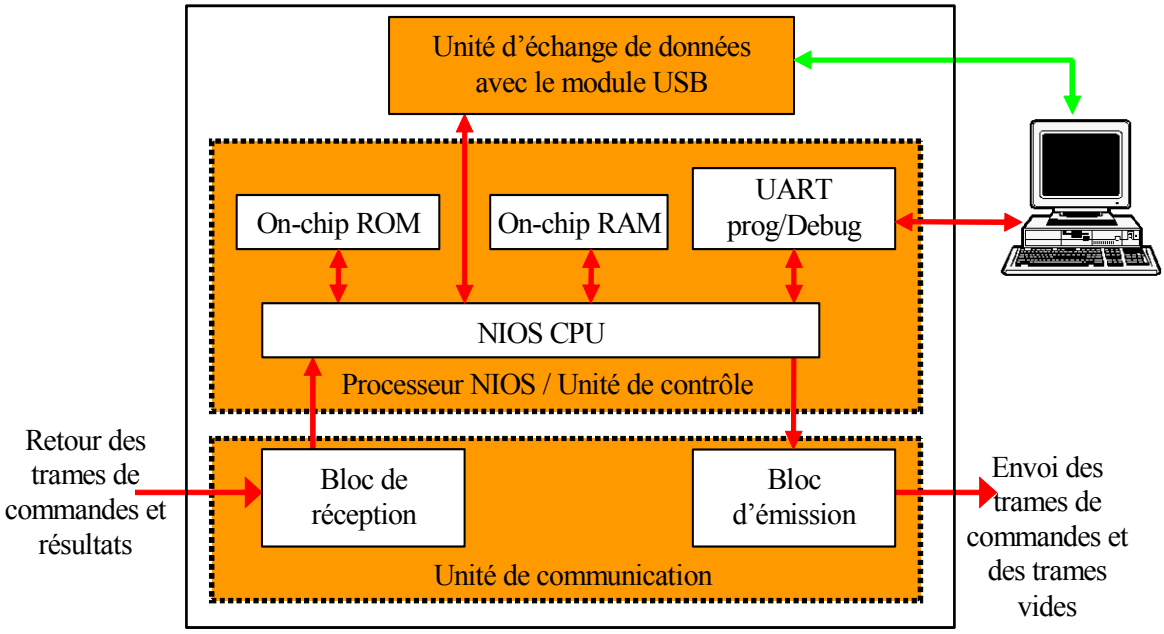


Figure 7-10 : Schéma de l'implantation du Module de Contrôle dans un FPGA.

L'unité de communication étant une adaptation de l'unité de communication commune à tous les modules, elle n'est pas intégrée au microprocesseur NIOS, dans un souci d'homogénéité. Elle reste décrite en langage VHDL sous forme de bloc IP mais adaptée aux spécifications du module de contrôle. En revanche, le reste des unités du module sont à adapter à l'implémentation physique choisie, c'est-à-dire au processeur NIOS.

1 – Unité de contrôle

L'unité de contrôle est réalisée à l'aide d'un processeur embarqué dans le FPGA. Le FPGA choisi est un FPGA de la société Altera, qui a la possibilité de contenir un microprocesseur embarqué NIOS décrit en langage VHDL comme exposé dans le paragraphe 7.2.2 précédent. Ce processeur NIOS réalise l'ensemble des fonctions de l'unité de contrôle. C'est-à-dire l'interprétation des trames arrivant au module de contrôle, le séquencement des trames à émettre sur l'anneau en fonction de son programme et l'envoi des résultats vers l'extérieur du système. Pour cela, le cœur de processeur NIOS exécute un programme écrit en langage C ou en assembleur. Pour éviter d'obtenir un processeur NIOS gourmand en ressources matérielles, et fastidieux à développer et à tester, nous avons développé trois programmes pour notre système embarqué. Chaque programme réalise un type de traitement sur les données. Il y a donc un programme pour le transfert direct des images en 256 niveaux de gris, un autre pour le transfert direct des images binaires et un dernier pour les calculs d'un champ de vecteurs vitesse.

Ce processeur NIOS est constitué d'un unique cœur de processeur (CPU) couplé à quelques composants. Le processeur NIOS incorpore donc dans sa description de la mémoire de type ROM pour son boot, de la mémoire de type RAM et l'UART pour la gestion de la programmation et du débugage. Cette mémoire RAM se répartie entre la mémoire de sauvegarde des registres et données programme pour le CPU, et la mémoire du séquenceur pour la mémorisation des trames à envoyer sur l'anneau. Le fait d'inclure la mémoire du séquenceur directement dans la description du processeur NIOS permet de réduire les temps d'accès aux trames de fonctionnement.

Dans la version actuelle, 9 Ko de mémoire sont allouées au processeur embarqué (un maximum de 13 Ko pour l'APEX20KE200) : 1Ko en tant que ROM et 8 Ko comme RAM. Cette taille mémoire ne permet pas de mémoriser toutes les trames. Seulement une quinzaine de trames sont stockées. Le reste des trames est recalculée par le processeur NIOS selon le protocole décrit dans le Chapitre 6 à la section 6.6 Module de contrôle. C'est le cas des trames de positionnement du pointeur d'adresse pour la relecture des motifs et des imagettes, envoyées au module mémoire.

2 – Unité d'échange

L'unité d'échange ou d'interface entre le module de contrôle et une station de travail est réalisée en VHDL. Physiquement, cette interface peut prendre deux formes : soit une liaison série soit une liaison USB. Chacune de ces liaisons se fait par l'intermédiaire de circuits dédiés à la gestion de cette communication avec un ordinateur (UART ou microcontrôleur USB). La liaison série a été utilisée au début de la réalisation car son unité d'échange était incluse dans le processeur NIOS comme un composant associé au CPU. Mais, cette liaison est aussi utilisée pour pouvoir débuguer le processeur NIOS et son programme, ce qui a pour inconvénient de reléguer le transfert de résultats au second plan. D'un point de vue pratique, nous avons décidé de séparer ces deux échanges. D'un côté, nous avons la liaison série pour le processeur NIOS (programmation et débugage), et d'un autre, une liaison

pour l'envoi de résultats. Pour cette seconde liaison, nous avons opté pour une liaison USB car le débit de données est important (de approximativement 800 Ko/s pour les résultats jusqu'à 10 Mo/s pour transmettre une image pleine trame du capteur). Un module électronique de liaison USB a donc été connecté au FPGA [optiminfo-web]. Ce module fournit une connexion USB 2 bidirectionnelle à une fréquence de 1 Mo/s. Côté station de travail, le driver fourni avec le module permet de récupérer ou d'envoyer des données à une fréquence de 1 Mo/s soit en développant notre propre logiciel soit en utilisant leur programme.

Dans l'état actuel des choses, cette unité d'échange n'est utilisée que pour envoyer les résultats à la station de travail. A terme, elle devrait permettre également d'envoyer des informations pour la configuration des différents modules. Côté ordinateur, un logiciel développé en C++ permet de récupérer les données et soit de les enregistrer dans un fichier, soit d'afficher directement les résultats à l'écran sous forme d'image binaire ou en niveaux de gris dans le cas d'un transfert direct, ou sous forme d'images de champs de vecteurs dans le cas de calculs de PIV (avec rafraîchissement permanent permettant de suivre le mouvement).

7.4 - Résultats et performances

Le fonctionnement a été validé par simulation sur le logiciel ModelSim [Model-web] de la société Mentor Graphics [Mentor-web]. A partir de fichiers *testbench* écrit en langage VHDL, chaque bloc, unité et module ont été simulés fonctionnellement et temporellement. Les unités communiquant avec des circuits extérieurs au FPGA (capteur d'images CMOS, mémoire SRAM et module USB) ont été simulées en utilisant les caractéristiques et timings extraits de leurs documentations respectives. Tous les éléments ont été validés séparément, puis module par module.

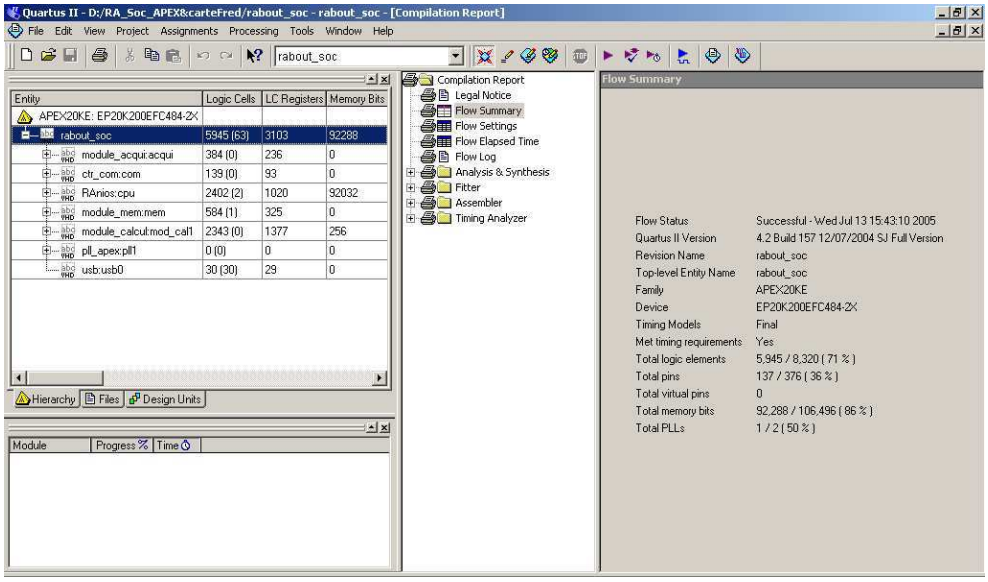


Figure 7-11 : Rapport de compilation du logiciel Quartus II.

Après simulation et validation, le système passe par les phases de synthèse et placement/routage à l'aide de l'outil Quartus II (version 4.2) [Quartus-web] de la société Altera [Altera-web]. Ce logiciel fournit, à la fin de ces phases, divers rapports et schémas. Parmi ces documents, le rapport de compilation (Figure 7-11) nous donne les quantités de ressources matérielles utilisées pour l'implémentation physique du système, tandis que la fonction *RTL viewer* (*Register Transfert Level*) présente une description hiérarchique par niveaux d'abstraction du système sous forme de schémas. Cette description correspond en tout point à la description fonctionnelle sous forme de blocs IP exposée dans le Chapitre 6.

7.4.1 Ressources matérielles

L'ensemble des résultats présentés dans cette section a été fait avec la même description VHDL de l'architecture Round-About. Cette architecture se compose d'un exemplaire unique de chaque module. Le module de contrôle est réalisé avec un processeur NIOS de première génération comme décrit dans le paragraphe 7.3.2 précédent. Par la suite, cette architecture Round-About sera dénommée « architecture simple ». Ce système a été synthétisé pour différentes cibles FPGAs. Il s'agit des FPGAs de type APEX EP20K200E et STRATIX II EP2S60F, présentées dans le paragraphe 7.2.2. Nous avons choisi ces deux cibles car elles ont l'avantage d'être les FPGAs des cartes de développement pour l'utilisation des processeurs embarqués NIOS, mais aussi parce que ces deux cibles présentent deux architectures internes complètement différentes, basées sur la même base : l'élément logique ou LE (Figure 7-12) équivalent à 24 portes logiques.

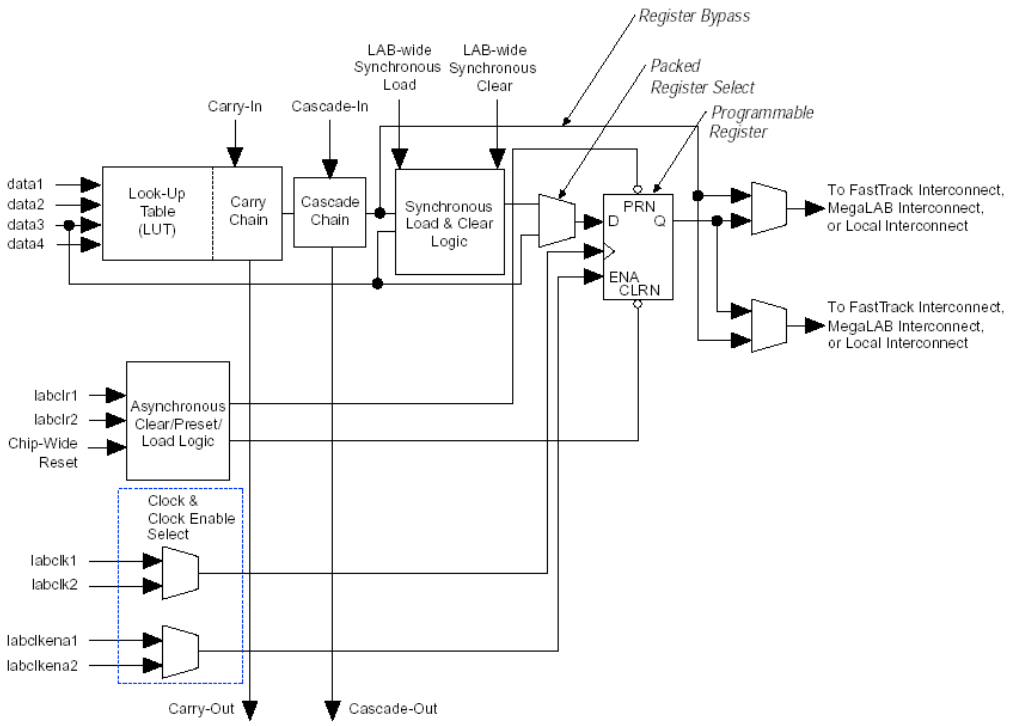


Figure 7-12 : Schéma bloc d'un élément logique [Altera-web].

Nous exposerons les résultats d'occupation en ressources matérielles sous deux formes : le pourcentage d'occupation du FPGA et le nombre d'éléments logiques (LEs) correspondant. Le coût global en ressources matérielles de notre système composé d'une architecture simple est détaillé dans le Tableau 7-2. Nous pouvons estimer le nombre d'éléments logiques à approximativement 4 000 et la quantité de mémoire interne à 91 kilobits pour notre architecture simple quelle que soit la cible FPGA. Il y a une différence de 762 LEs entre l'implémentation sur APEX et celle sur STRATIX II. Sachant que l'élément logique de base de ces deux FPGAS est identique. Nous pouvons en déduire que l'agencement des éléments logiques au sein du FPGA (l'architecture interne du FPGA) influence de façon significative le nombre d'éléments logiques utiles à l'architecture Round-About.

Nom du circuit	Pourcentage d'éléments logiques utiles (LE)	Quantité de mémoire interne utilisée (bit)
APEX EP20K200	50,69% 4 218 / 8 320	87,62% 93 312 / 106 496
STRATIX II EP2S30	5,84% 3 456 / 48 352	3,67% 93 312 / 2 544 192

Tableau 7-2 : Taux d'occupation de l'architecture Round-About avec 1 module de calcul.

Le Tableau 7-3 expose en détail la répartition des ressources matérielles en fonction du type de module au sein de l'architecture.

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="3">APEX EP20K200E</th><th colspan="3">STRATIX EP2S60F</th></tr>
<tr><th colspan="2">Nom</th><th>Eléments Logiques</th><th>Mémoire interne (en bit)</th><th>Eléments Logiques</th><th colspan="2">Mémoire interne (en bit)</th></tr>
<tr><td colspan="2">Module Acquisition</td><td colspan="2">384
4,61%</td><td>0</td><td colspan="2">252
0,52%</td><td>0</td></tr>
<tr><td colspan="2">Module Mémoire</td><td colspan="2">584
7,02%</td><td>0</td><td colspan="2">430
0,89%</td><td>0</td></tr>
<tr><td colspan="2">Module Calcul</td><td colspan="2">677
8,14%</td><td>1280
1,20%</td><td colspan="2">419
0,87%</td><td>1280
0,05%</td></tr>
<tr><td rowspan="3">Module Contrôle</td><td>µP NIOS</td><td>2402</td><td rowspan="3">30,92%</td><td>92032
86,42%</td><td>1668</td><td rowspan="3">3,56%</td><td>92032
3,62%</td></tr>
<tr><td>Unité Com.</td><td>139</td><td>0</td><td>49</td><td>0</td></tr>
<tr><td>Unité USB</td><td>32</td><td>0</td><td>7</td><td>0</td></tr>
</table>

Tableau 7-3 : Répartition des ressources matérielles de l'implémentation des différents modules.

Nous pouvons noter que l'essentiel des ressources est dédié au module de contrôle et plus particulièrement au processeur embarqué NIOS. Le processeur NIOS nécessite beaucoup de mémoire interne pour pouvoir stocker son programme à exécuter. Ce choix a été fait par nécessité puisque la totalité de la mémoire externe est utilisée pour les données images, le programme du processeur NIOS

ne peut être disposé qu'en interne. De plus, cette proximité du programme accélère son exécution par le CPU. Les autres modules ont besoin de beaucoup moins de ressources. Cela permet à notre architecture de répondre aux contraintes de modularité du système. La première est d'envisager l'emploi de plusieurs capteurs pour augmenter les cadences d'acquisition. Ainsi, l'architecture doit disposer d'autant de modules d'acquisition qu'il y a de capteurs d'images pour réaliser un système multi-capteurs. C'est rendu possible par le peu de ressources dont a besoin le module d'acquisition. Bien qu'il occupe plus de ressources matérielles, nous pouvons faire la même remarque avec le module de mémorisation. Plusieurs bancs mémoires externes pourraient ainsi être utilisés en parallèle. Cependant, cette extension des sources d'images ou de capacités mémoire est assujettie aux nombres de connexions disponibles entre le FPGA et l'extérieur.

Le module de calcul nécessite moins de 1000 LEs. Donc, nous pouvons associer plusieurs modules de calcul dans le même FPGA sans aucune difficulté. Cette association permettra d'accélérer les traitements des images par la parallélisation des calculs sur les couples motif imagettes dans le but de répondre à la contrainte temps réel. Mais, le besoin de mémoire interne (essentiellement pour la mémoire cache) de ce module limite leur nombre du fait de la quantité limitée de mémoire embarquée (en grande partie utilisée dans le processeur NIOS) mais aussi de sa distribution spatiale dans l'architecture interne des FPGAs. Plusieurs options pour implémenter ces mémoires caches du module de calcul sont possibles. Nous en avons choisi deux : soit diminuer l'utilisation la RAM interne soit synthétiser ces mémoires avec des éléments logiques. Nous avons exploré ces deux possibilités pour évaluer leur impact sur la quantité de ressources matérielles utiles pour implémenter le module de calcul. Les résultats nous sont fournis par le logiciel Quartus II après le choix d'options de synthèse adéquates (optimisation de la synthèse en terme de surface occupée ou en terme de vitesse interne, synthèse de mémoire en registre, etc.).

	Optimisation en terme d'utilisation de LEs		Optimisation en terme de vitesse		Aucune mémoire interne utilisée	
	Eléments Logiques	Mémoire interne (en bit)	Eléments Logiques	Mémoire interne (en bit)	Eléments Logiques	Mémoire interne (en bit)
Module de Calcul	677 8,14%	1280 1,20%	2343 28,16%	256 0,24%	2 775 33,35%	0
Total de l'architecture	50,69% 4 218 / 8 320	87,62% 93 312 / 106 496	70,72% 5 884 / 8 320	86,66% 92 288 / 106 496	75,91% 6 316 / 8 320	86,42% 92 032 / 106 496

Tableau 7-4 : Résultats d'implémentation sur le circuit APEX EP20K200E.

Le Tableau 7-4 présente la comparaison entre ces deux options et un module de calcul dans lequel la mémoire cache est entièrement réalisée avec de la RAM interne. Il montre aussi les

conséquences de leur utilisation sur l'implémentation de l'architecture simple dans un circuit FPGA APEX EP20K200E.

La libération de la totalité de la mémoire (gain de 100%) équivaut à une augmentation de 75,6% du nombre d'éléments logiques nécessaire au module de calcul. L'architecture simple occupe 25,22% de ressources en plus par rapport à la première implémentation. La solution intermédiaire (optimisation de la synthèse en terme de vitesse interne) ne transpose qu'une partie de la mémoire cache en éléments logiques. Une petite quantité de mémoire est utilisée pour réaliser le stockage du motif binaire de taille 16×16 pixels (soit 256 bits). Le gain en terme de capacité mémoire est de 80% pour une augmentation de 71% du nombre d'éléments logiques en plus pour le module de calcul, soit 1666 LEs pour 1024 bits mémoire. Pour l'architecture simple complète, cela représente un surcoût de 20% de LEs. La mémoire ainsi libérée permettra de disposer de plus de ressources pour le processeur NIOS, donc de développer son programme. Dans le cas d'un circuit FPGA APEX EP20K200E, la solution d'utiliser la mémoire interne est plus rentable que la solution éléments logiques. Nous pouvons disposer sept modules de calcul avec la première solution, deux avec la deuxième et d'un seul avec la solution tout en éléments logiques. C'est l'une des raisons qui nous a poussé à faire migrer notre système sur un FPGA disposant de plus de ressources matérielles.

7.4.2 Mesures et estimations des performances temporelles

L'ensemble de ces données a été obtenu avec le FPGA APEX EP20K200E couplé avec le capteur d'images CMOS IBIS4, dont les caractéristiques sont exposées dans le paragraphe 7.2.1 précédent. Le FPGA fonctionne à une fréquence de 50 MHz (soit une période de 20 nanosecondes) bien que la synthèse donne une fréquence maximale de fonctionnement de 61 MHz. Nous allons mesurer la vitesse de traitement de notre système. Avec ces résultats, nous pourrons évaluer les performances en terme de cadences de traitement, mais aussi démontrer que notre architecture répond aux contraintes de traitement en temps réel.

1 – Mesures

Pour mesurer les performances temporelles de notre système, nous nous baserons sur le traitement d'un motif binaire de taille 16×16 pixels et d'une imagette binaire 32×32. Pour réaliser ce traitement, le système doit acquérir deux imagettes 32×32 pixels et les mémoriser en mémoire externe. Cette phase d'acquisition n'a pas été mesurée car sa durée diffère en fonction des caractéristiques d'acquisition. Le Tableau 7-5 récapitule les durées des différentes phases pour réaliser le calcul d'un vecteur vitesse, en fait de la position du déplacement du motif au sein de l'imagette. Nous avons séparé le calcul en trois phases : le transfert des données de la mémoire SRAM jusqu'à la mémoire cache du module de calcul, le calcul en lui-même et le transfert du résultat jusqu'au module de calcul. L'ensemble est exécuté en 267 microsecondes depuis l'émission de la première trame de lecture jusqu'à l'arrivé de la trame de résultats (module de contrôle).

Fonctions	Temps
Transfert des données	160 µs
Calcul d'un vecteur	96 µs
Transfert résultat via USB	8 µs
Temps de latence entre deux calculs	3 µs
Total	267 µs

Tableau 7-5 : Durée pour le calcul d'un vecteur par l'architecture.

Les calculs en eux-mêmes ne représentent que 36% de cette durée, le reste, approximativement 2/3, est consacré aux échanges de données, images et trames de commandes. Les trois microsecondes de latence entre deux calculs sont dues au module de contrôle. Cela correspond au temps qu'il lui est nécessaire entre l'arrivée de la trame de résultats et l'envoi de la première trame pour lancer le prochain calcul.

La Figure 7-13 expose les détails de ces échanges sur le bus de données et sur l'anneau. Nous remarquons que les six trames de commandes nécessaires à ce traitement sont émises et retournées au module de contrôle dans cet intervalle de temps.

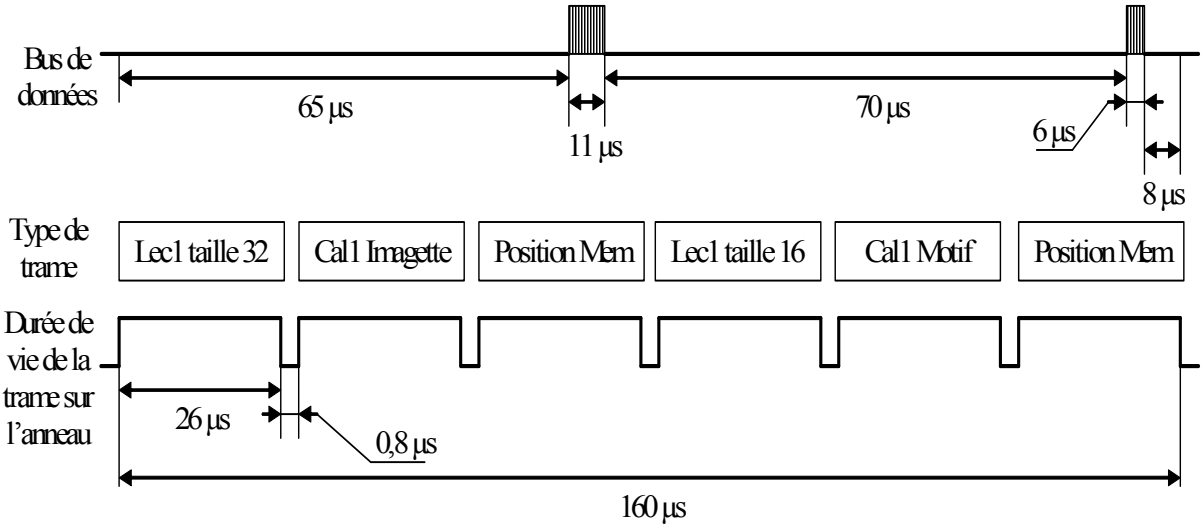


Figure 7-13 : Détails des échanges de données pour au cours du transfert des données.

Une trame met 26 microsecondes pour parcourir l'anneau. Ces 26 microsecondes sont mesurées depuis le départ du premier octet de la trame de l'unité d'émission jusqu'au retour du sixième et dernier octet de la trame à l'unité de réception. L'envoi de ces trames sur l'anneau se fait de manière séquentielle à cause du processeur NIOS et de sa gestion des interruptions. Lors de l'émission ou de la réception d'une trame, le processeur NIOS génère une interruption. Pour éviter des pertes

d'informations, l'interruption de la réception est prioritaire sur celle de l'émission, c'est-à-dire qu'elle interdit toute émission tant qu'elle n'est pas résolue. Cette gestion des interruptions engendre une latence sur l'anneau et provoque l'envoi séquentiel des trames de commandes du fait que le début de la réception de la trame circulant sur l'anneau commence avant le début de l'envoi de la suivante. Ceci se note par la présence d'une latence de 0,8 microsecondes entre la réception d'une trame et l'envoi de la suivante. Cet envoi séquentiel provoque une sous-utilisation des capacités du bus de données reliant le module de mémorisation au module de calcul. En effet, durant cette phase, le bus de données n'est actif que 10% du temps total. Le reste du temps, il est inactif. Ce temps d'inactivité pourrait être mis à profit pour transmettre de nouvelles données vers un autre module de calcul, mais cette transmission nécessite de nouvelles trames de commandes.

2 – Estimations

A partir de ces mesures, nous pouvons estimer approximativement le temps nécessaire aux calculs d'un nombre de vecteurs donné en fonction du nombre de modules de calcul présents sur l'anneau. Ce calcul s'effectue à partir de l'équation suivante, dans l'hypothèse où il n'y ait aucune latence dans la transmission des données :

$$Tps_{total} = Nb_{points} \times 17\mu s + \frac{Nb_{points}}{Nb_{M.\ calcul}} \times 96\mu s \qquad (7\text{-}3)$$

où Tps_{total} est le temps nécessaire aux calculs, Nb_{points} le nombre de vecteurs ou de points de mesures et $Nb_{M.\ calcul}$ le nombre de modules de calcul travaillant en parallèle. Les 17µs correspondent au temps de transfert de données (imagette et motif) entre la mémoire externe et la mémoire cache du module de calcul. Les 96 µs représentent le temps de calcul de la corrélation binaire par comparaison entre le motif et l'imagette.

Cette équation permet l'estimation du nombre de vecteurs vitesses pouvant être calculés en une seconde en fonction du nombre de modules de calcul dans l'architecture. Pour un module de calcul, nous obtenons 8 849 points de mesures par seconde.

De cette équation, nous pouvons également déduire le nombre maximum de modules de calcul qui peuvent mis en parallèle pour un fonctionnement à 50 MHz. Pour cela, nous résolvons l'égalité suivante :

$$Nb_{points} \times 17 = \frac{Nb_{points}}{Nb_{M.\ calcul}} \times 96 \qquad (7\text{-}4)$$

C'est-à-dire que le temps de calcul distribué pour un nombre donné de points de mesures ne peut être supérieur au temps de chargement des données. Nous obtenons donc un maximum de 5,65 modules de calcul pouvant être parallélisés dans l'architecture Round-About. Cela donne donc un maximum de 27 624 points de mesures par seconde en associant 5 modules de calcul.

Comme ces valeurs ont été réalisées à partir de mesures faites avec des imagettes 32×32 pixels, elles ne sont donc valables que dans ce cas de figure, pour d'autres tailles d'imagettes, il faudrait réaliser de nouvelles mesures de temps de calcul.

7.4.3 Performances de notre système

Un module de calcul a besoin de 267 microsecondes pour réaliser une mesure de déplacement (transferts et calculs) d'un motif binaire de taille 16×16 pixels dans une imagette binaire 32×32 soit 1 point de mesure. Pour des images 320×256 pixels soit 80 vecteurs ou points de mesures, nous pouvons estimer le temps total nécessaire à leur calcul pour un module de calcul à approximativement 80×267=21 360 µs. Cette valeur calculée à partir de mesures est à comparer avec les résultats expérimentaux suivant.

Pour valider notre système, nous avons calculé le temps nécessaire à la mesure de 80 points de mesures avec des couples d'images de taille 320×256 pixels. Pour simuler le mouvement de particules, nous utilisons une image de bruit blanc générée par ordinateur et disposée sur une roue en rotation. La Figure 7-14 présente un échantillon des images acquises par notre système. L'acquisition de deux images par le capteur et leur mémorisation sous forme binaire s'effectue en 19,5 millisecondes. Puis, la phase de mesure décrite précédemment prend 21,5 millisecondes. Cela fait un total de 41 millisecondes depuis le début de l'acquisition de la première image à l'affichage de ces vecteurs sur l'écran de l'ordinateur. Nous obtenons une cadence d'affichage des champs de vecteurs (Figure 7-14c) d'approximativement 24,39 images par seconde pour un module de calcul, soit 1951 vecteurs par seconde sur des images 320×256 pixels découpées en imagettes de taille 32×32 pixels.

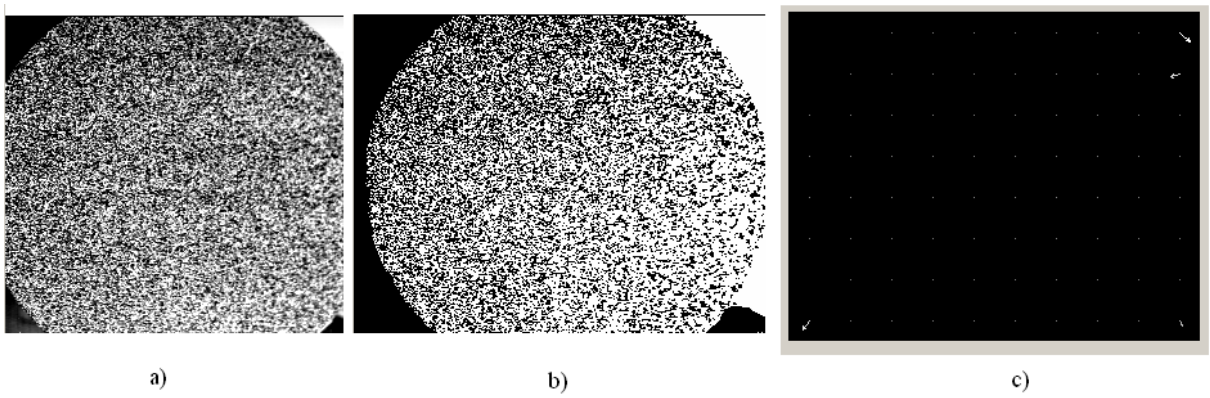


Figure 7-14 : Exemples d'images prises avec notre système :
a) image de la mire en niveaux de gris ; b) image de la mire en binaire ;
c) champ de 80 vecteurs vitesses du mouvement rotatif de la mire.

Si nous extrapolons ces résultats pour la même résolution des images, nous obtenons les performances présentées dans le Tableau 7-6. A cette résolution, le capteur d'image CMOS IBIS4 fournit 112 images par seconde. Avec 80 points de mesure par image, cela fait 8 960 points de mesure à traiter en une seconde.

Nombre de modules de calcul	Fréquence d'images traitées	Nombre de champs de vecteurs calculés
1	48 images/sec	24,39 images/sec
2	96 images/sec	48,78 images/sec
3	146 images/sec	73,17 images/sec
5	242 images/sec	121,95 images/sec

Tableau 7-6 : Extrapolations des performances du système embarqué Round-About pour des images de résolution 320×256 pixels (1/16 de la matrice).

Nous notons que pour atteindre le traitement en temps réel des images, il faut associer au moins 3 modules de calcul via l'architecture Round-About dans notre système embarqué. Il est à noter qu'avec 5 modules de calcul, le nombre maximum actuellement possible, le traitement de 200 images par seconde est réalisable.

Nous pouvons faire de même (Tableau 7-7) avec une résolution supérieure telle que 640×512 pixels (1/4 de la matrice) à une cadence de 28 images par seconde. Il y a 320 points de mesure par image, donc 8 960 points de mesure à calculer. Ce nombre est identique au précédent car si nous avons augmenté la résolution des images, nous avons aussi diminué leur cadence car le capteur d'image CMOS reste le même.

Nombre de modules de calcul	Fréquence d'images traitées	Nombre de champs de vecteurs calculés
1	12 images/sec	6,1 images/sec
3	36 images/sec	18,3 images/sec
5	60 images/sec	30,5 images/sec

Tableau 7-7 : Extrapolations des performances du système embarqué Round-About pour des images de résolution 640×512 pixels (1/4 de la matrice).

Le traitement temps réel est atteint aussi avec 3 modules de calcul. Cela est tout à fait normal puisqu'il y a le même nombre de points à calculer.

Dans le cas d'une utilisation à pleine résolution, le capteur fournit 7 images par seconde soit 8 960 points à calculer (1280 points de mesure par image). Nous obtenons un traitement en temps réel avec 3 modules de calcul (9 images traitées par seconde).

En conclusion, notre système embarqué Round-About effectue des mesures en temps réel avec 3 modules de calcul associés par l'architecture Round-About. Ce nombre est inférieur au nombre maximum estimé de modules de calcul, dans le paragraphe 7.4.2, pouvant être associés sans que le

temps de calcul global ne soit inférieur au temps de transfert des donnée. Cela tend à prouver que notre estimation est basée sur de bonnes hypothèses.

Pour démontrer ces performances sur des phénomènes physiques, une plate-forme expérimentale a été mise au point (Figure 7-15). Un rideau de bulles d'air est généré dans une cuve d'eau et illuminé par une tranche laser verte. Pour améliorer le contraste des bulles ainsi éclairées, le fond de la cuve est noir. Le système embarqué Round-About est relié à un ordinateur standard qui sert à la fois à la programmation du FPGA et à la visualisation des données soit sous forme d'images soit sous forme de champs de vecteurs vitesses.

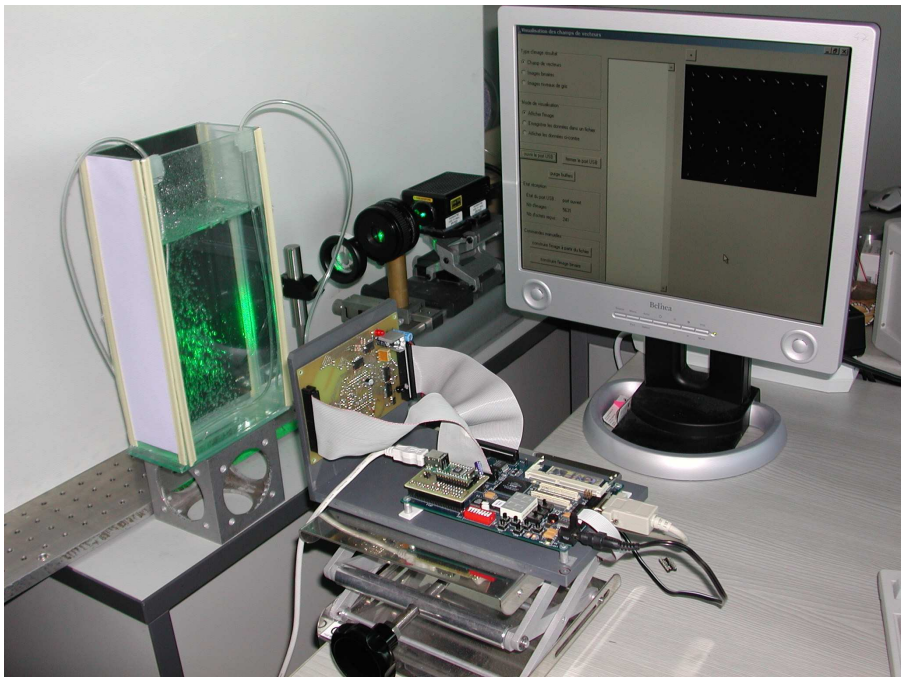


Figure 7-15 : Plate-forme expérimentale de test du système Round-About.

L'ensemble du système embarqué fonctionne à 50 MHz et réalise la corrélation binaire par comparaison sur des imagettes de taille 32×32 pixels avec un motif 16×16. Ce système réalise le traitement de 24 images 320×256 pixels en une seconde. Les mouvements des bulles quant elles traversent la nappe laser sont correctement restitués à l'écran. Notre système respecte donc les contraintes liées à l'application de mesure de vitesse en mécanique des fluides.

7.4.4 Synthèse et comparaisons

Pour récapituler, nous avons mesuré les performances suivantes :

- Un module de calcul a besoin de 267 microsecondes pour calculer le déplacement d'un motif binaire de taille 16×16 pixels dans une imagette binaire 32×32 soit 1 point de mesure.

- Pour des images 320×256 pixels soit 80 vecteurs ou points de mesures, le traitement se fait en 41,5 millisecondes soit une cadence de traitement de 48 images par seconde.

Toutes ces performances ont été mesurées sur un système embarqué composé d'un FPGA couplé à un capteur d'images CMOS avec une architecture Round-About simple, un seul module de calcul, pour une fréquence de travail de 50 MHz. Grâce à ces mesures, nous avons pu estimer à 5 le nombre maximum de modules de calcul à associer par l'intermédiaire de l'architecture Round-About pour satisfaire les contraintes d'une application de traitement en temps réel.

Comme l'architecture Round-About a déjà fait l'objet d'une première implémentation (voir Chapitre 5 section 5.2), une comparaison entre les performances des deux systèmes créés peut être faite. Les performances des deux systèmes sont réunies dans le Tableau 7-8.

	Système multi-cartes [Dubois 02] fonctionnant à 12 MHz	Système monopuce [Lelong 05a] fonctionnant à 50 MHz
Temps pour calculer 1 point de mesures par 1 module de calcul	411 µs	267 µs
Temps de traitement d'images 320×256 pixels soit 80 points de mesures	32,9 ms soit 30,4 images/sec	21,36 ms soit 46,8 images/sec
Temps de traitement d'images 512×512 pixels soit 256 points de mesures	105,2 ms soit 9,5 images/sec	68,35 ms soit 14,6 images/sec

Tableau 7-8 : Comparaison des résultats des deux implémentations physiques de l'architecture Round-About possédant 1 module de calcul.

Bien que le temps mis par le module de calcul pour calculer un point de mesure ait été diminué, les gains en images traitées en une seconde sont minimes. Cela est du, dans notre système actuel, aux latences dans l'accès au données qui double le temps de transfert des données par rapport à la durée des calculs.

7.5 - Perspectives et évolutions

Au vu de notre prototype fonctionnel réalisé et des résultats obtenus, nous pouvons identifier les points noirs du système et proposer une solution. Des évolutions de l'architecture peuvent aussi être envisagées pour améliorer l'architecture Round-About.

7.5.1 Optimisations de l'implémentation physique

Au vu des performances de notre prototype, nous pouvons identifier deux points à améliorer dans le but d'améliorer encore plus ce système. Le premier est l'unité de contrôle du module de

contrôle, le processeur NIOS. Le deuxième est l'adaptation du protocole de communication de l'anneau pour un fonctionnement plus rapide, voire temps réel.

1 – Le processeur embarqué NIOS

Nous avons vu dans le paragraphe 7.4.2 (Mesures et estimations des performances temporelles) que le processeur NIOS limitait de par son fonctionnement l'activité du module de calcul. Malgré son un jeu d'instructions réduit RISC, une fonction simple requiert beaucoup d'instructions, donc un grand nombre de cycles machine. Tous ceci prend du temps malgré une fréquence de fonctionnement de 50 MHz (cycle de 20 ns), temps qui est une des contraintes essentielles de notre système. Pour illustrer ce phénomène, nous savons que une trame de six octets met 26 microsecondes pour parcourir l'anneau, soit 4,333 microsecondes par octet (216,65 cycles). Un octet a besoin de 4 cycles d'horloge pour traverser l'unité de communication d'un module. Avec 3 modules sur l'anneau (le module de contrôle ne compte pas), un octet parcourt l'anneau en 12 cycles d'horloge. Donc, le processeur NIOS prend plus de 200 cycles d'horloge pour émettre et réceptionner un octet, ce qui est démesuré. Notre système devant réaliser le traitement des images en temps réel, donc fonctionner plus vite, le processeur NIOS représente un frein à cet objectif.

De plus, il a besoin d'une quantité d'éléments logiques largement supérieure aux autres unités et modules de l'architecture ainsi que de beaucoup de mémoire interne pour fonctionner plus rapidement. Ce besoin de ressources matérielles limite d'autant plus l'adaptabilité du système à l'application en terme de puissance de calcul mais aussi en terme d'ajout de composants extérieurs comme un autre capteur d'image.

De ces constations et pour mieux évaluer les contre-performances engendrées par l'utilisation du processeur NIOS, un module de contrôle entièrement décrit en langage VHDL, reprenant la description fonctionnelle du Chapitre 6, va remplacé le module de contrôle avec NIOS.

2 – Le protocole de communication de l'anneau

Le protocole de communication de l'anneau est une trame de six octets envoyés séquentiellement octet par octet (Chapitre 6 section 6-1). Ce protocole a été choisi pour sa simplicité de mise en œuvre et le peu de liens qu'il demande entre chaque module. Au cours du développement du prototype, nous avons noté que la totalité des six octets des trames pour faire fonctionner le système n'est pas nécessaire. Les octets numéro 4 et 5, destinés aux informations complémentaires du mot de commande, ne sont jamais utilisés. Ils ne servent qu'au timing du passage des trames dans un module. C'est-à-dire que le temps mis par ces octets pour traverser l'unité de communication est mis à profit par l'unité de décodage et de contrôle pour exécuter les instructions et ainsi pouvoir marquer le sixième octet.

En conclusion, un protocole, réduit à des trames de cinq voire quatre octets, peut être envisagé. Cela nécessitera une modification du fonctionnement de l'ensemble des unités gérant la

communication avec l'anneau, mais le gain en temps de communication améliorera les performances de l'ensemble du système.

7.5.2 Axes d'évolution de l'architecture

Aujourd'hui, la conception d'architectures de type SoC (ou SoPC) repose sur deux challenges : la mémoire et les communications. L'évolution de l'architecture Round-About passe donc par ces deux approches. La première, la gestion globale de la mémoire (mémoire partagée, distribué ou les deux) dans l'architecture, a pour objectif de diminuer l'influence des temps d'accès aux données. Quant à la deuxième, il s'agit de changer la topologie des communications pour les flots de commandes et de résultats pour augmenter le nombre de modules de l'architecture sans engendrer de latences. Par ces deux approches, nous souhaitons exploiter au mieux les multiples possibilités que l'architecture Round-About nous apporte.

1 – La gestion de la mémoire

L'utilisation d'un FPGA possédant une quantité de mémoire RAM interne significative pour nos besoins (traitement d'images en temps réel) permettrait de décrire l'architecture Round-About comme une architecture MIMD à mémoire distribuée et non plus partagée (voir Chapitre 5 paragraphe 5.2.2). Cette étape entraînerait des modifications de l'architecture actuelle telles que la disparition du module de mémorisation, et l'ajout d'unités de mémoire et de leur gestion dans les modules de calcul (Figure 7-16). Chaque module de calcul garde sa mémoire locale (bloc de cache) pour effectuer le calcul d'un point de mesure dans un temps minimum.

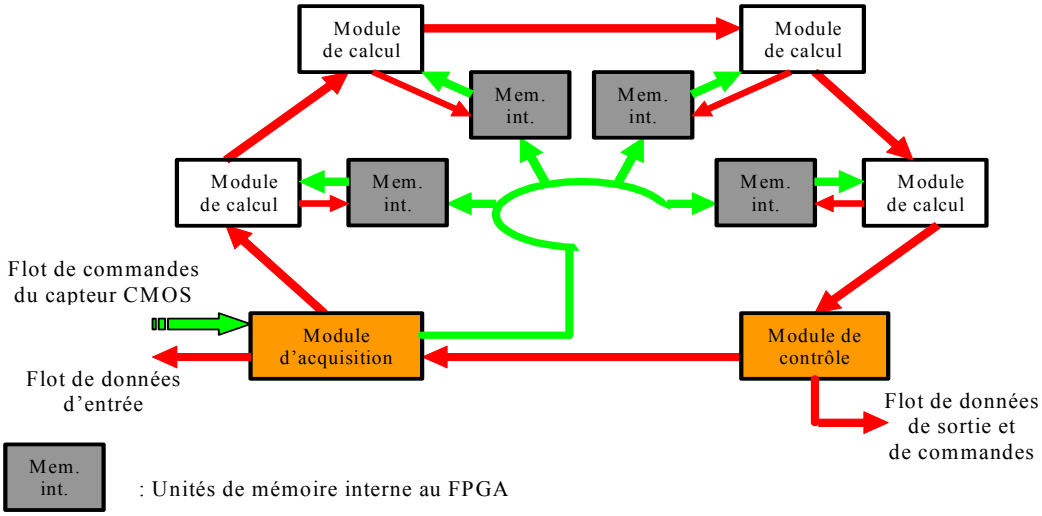


Figure 7-16 : Architecture Round-About de type MIMD à mémoire distribuée.

Dans le cas de notre application où les modules de calcul sont indépendants les uns des autres, une architecture mémoire à accès de type NORMA est requise. L'utilisation de mémoire distribuée à accès NORMA présente l'intérêt de ne plus limiter le nombre de modules de calcul. Mais, elle entraîne une gestion plus lourde de la répartition des données. En effet, chaque donnée, produite par le module

d'acquisition, doit être envoyée vers l'unité de mémoire du module de calcul qui en a besoin. Le fait que tous les modules de calcul soient identiques simplifie la répartition des données. Le temps de calcul étant le même quelque soit le module, les données peuvent être réparties de manière cyclique entre tous les modules de calcul comme un tourniquet ou *round-robin*.

Ces unités de mémoire pourraient implémenter sous deux formes : soit des FIFOs soit des mémoire RAM à double ports. Leur taille devra être calculée en fonction du temps de calcul d'un point de mesure par le module, c'est-à-dire le temps entre deux demandes de données, et le temps entre deux transferts d'imagettes par le module d'acquisition, donc le nombre de modules de calcul. Nous pouvons évaluer cette taille au minimum à l'équivalent de trois imagettes, un couple pour les prochains calculs et une imagette en cours de transmission.

Dans les cas la cadence d'acquisition des données est rapide (par exemple 300 images de taille 1024×1024 pixels par seconde) et que le nombre de modules de calcul soit au maximum des possibilités du FPGA, des pertes de données surviendront et le traitement en temps réel irréalisable. Pour s'adapter à ce type de contraintes, une architecture avec les deux types de mémoire, partagée et distribuée, peut être envisagée. La mémoire partagée sera utilisée pour la mémorisation des images provenant de la source (capteur ou caméra), tandis que la mémoire distribuée gérera les données (couples d'imagettes) à destination des modules de calcul et de leurs caches. Cette configuration permet de découpler l'acquisition des images, la répartition des imagettes et les calculs de points de mesure.

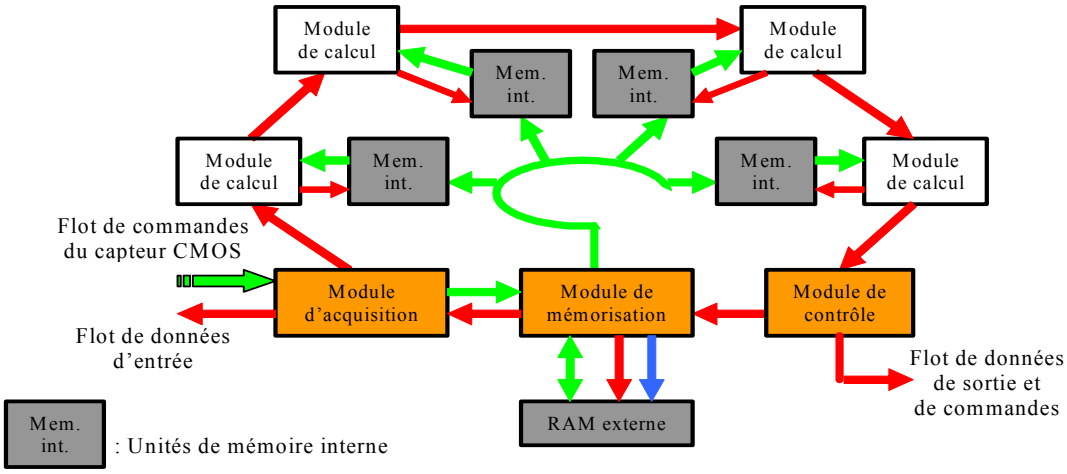


Figure 7-17 : Architecture Round-About de type MIMD à mémoire partagée distribuée.

L'architecture la plus réalisable sera d'utiliser de la mémoire externe pour la mémoire partagée et la mémoire interne pour la mémoire distribuée (Figure 7-17). Le choix d'utilisée de la mémoire externe pour la mémoire partagée est du aux contraintes auxquelles doit répondre cette mémoire en terme de vitesse d'accès et de quantité de données à stocker. Il restera à gérer ses accès en écriture et en lecture pour éviter toutes pertes de données, mais aussi pour satisfaire la contrainte temps réel. La solution qui consiste à faire un multiplexage temporel des accès semble être la meilleure, mais d'autres peuvent être explorées.

2 – La topologie des communications

Comme nous l'avons vu précédemment l'anneau de communication ralenti le fonctionnement du système. Bien que deux points noirs (le mécanisme d'arbitrage centralisé et le protocole) aient été identifiés, il est possible que cette topologie de communication soit à remettre en cause surtout si le nombre de modules augmente (goulot d'étranglement, temps de transmission très supérieur au temps de calcul). Dans ce cas de figure, d'autres topologies sont envisageables (maillée, tore, etc.) que l'anneau. Notre implémentation étant basée sur un système de type SoC, nous pouvons réalisé ces topologies par l'utilisation de réseaux sur puce (*Network-on-Chip* ou NoC).

Un réseau sur puce est un type d'interconnexion utilisant un modèle de communication distribué qui tente de répondre aux problèmes de latence, flexibilité et réactivité. Il s'agit de connecter les différents modules de l'architecture à travers un réseau. Cette approche a été déjà proposée pour les architectures de processeur parallèle [Lan 88]. Un NoC est conçu sur un modèle simplifié de trois couches qui sont la couche physique, la couche architecture-contrôle et la couche protocole [Benini 02]. La couche physique permet de définir les canaux de communication et la façon de les interconnecter entre eux (type *crossbar*, *switch* ou autres). La couche architecture-contrôle détermine la gestion des communications dans le réseau et leurs topologies. Quant à la couche protocole, elle définie les communications entre les nœuds du réseau et les modules. De nombreuses topologies sont disponibles, mais les topologies matricielles sont les plus répandues, pour des raisons d'implémentation (simplicité de routage des données) et de connexion (distance courtes de nœud à nœud). Cependant, cette architecture présente des temps de latence.

Notre système de communication en anneau par trame à forte granularité (Figure 7-18) est un réseau de communication basé sur le modèle *Token Ring* des réseaux informatiques de communication. Donc, il peut être considéré comme un réseau sur puce de topologie *ring*.

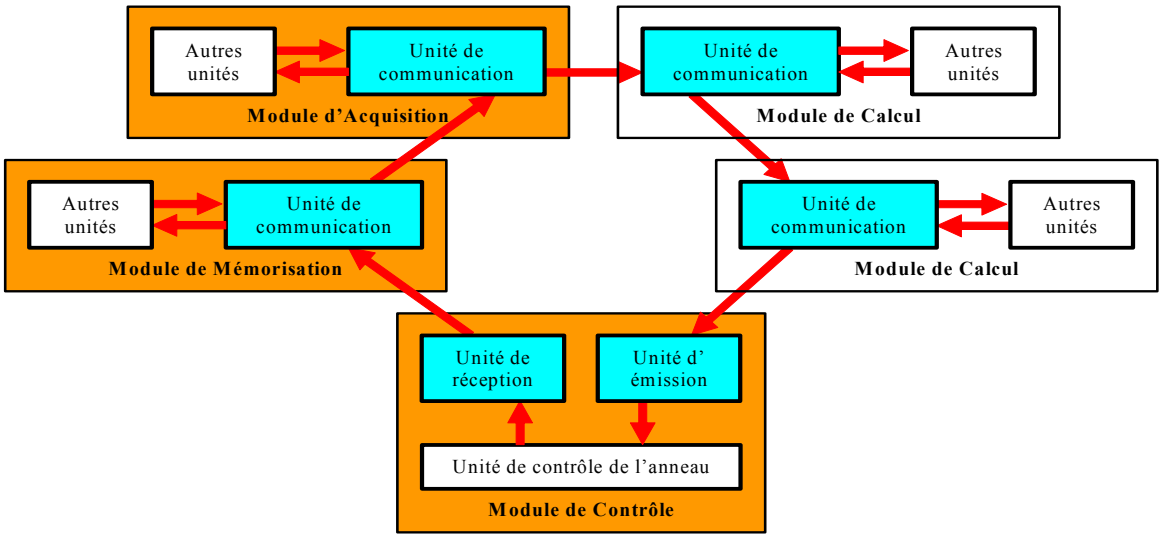


Figure 7-18 : Système de communication en anneau unidirectionnel par trames.

Mais, cet anneau limite de par sa nature le nombre de modules pouvant lui être connectés. Cela est du au temps de circulation des commandes qui croît avec ce nombre. Actuellement, ce nombre est d'une dizaine de modules : 5 modules de calcul, 1 module de contrôle, 1 module d'acquisition et 1 module de mémorisation.

Pour l'utilisation de plus d'une dizaine de modules, il faut repenser le système de communication de l'architecture. Plusieurs évolutions sont possibles. Soit, nous gardons le principe de transmission par trames et nous changeons la topologie du système. Soit, nous réorganisons l'architecture Round-About avec un nouveau système de communication pour les flots de commandes et résultats comme par exemple un réseau maillé 2D de *switch*. La Figure 7-19 illustre cet exemple avec l'architecture Round-About à mémoire partagée. Le principe de la communication par paquet peut être gardé avec cet exemple, mais, contrairement à l'anneau, chaque module reçoit seulement les trames qui lui sont destinées.

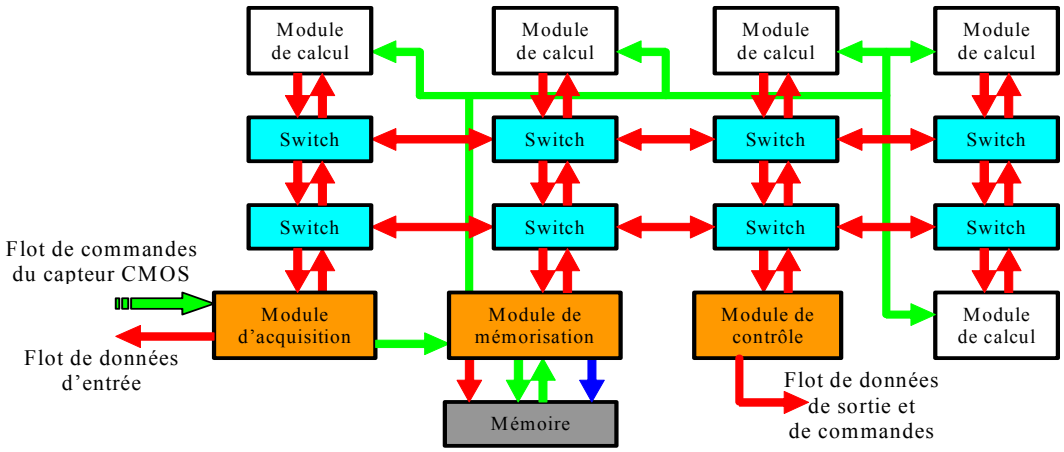


Figure 7-19 : Notre architecture adaptée à la topologie d'un réseau maillé 2D.

7.5.3 Adaptations de l'architecture à notre application

Notre application, présentée dans le Chapitre 1, est basée sur une technique de mesure de mouvement pour la mécanique des fluides appelée Particles Images Velocimetry (PIV). Elle consiste à détecter le mouvement par corrélation d'images. Trois points sont primordiaux pour une bonne détection du mouvement : la qualité des images, le temps entre les images et le temps d'intégration de chaque image. Ces points dépendent du système d'acquisition des images, donc non modifiable par notre architecture. Cependant, le premier point peut être modifié par l'utilisation d'algorithmes de prétraitement ou compensé par des algorithmes de traitement adaptés ; et le deuxième est contrôlable par l'utilisation de deux sources d'images.

1 – Contrôle du temps entre deux images

L'une de des principales contraintes de la PIV est le temps entre la prise de deux images. Cette durée doit être adapté à la vitesse des phénomènes observés. Plus elle est courte, plus la vitesse mesurée est

grande. Pour mieux contrôler l'intervalle de temps entre les deux images successives, il existe plusieurs solutions possibles : l'utilisation de système d'acquisition rapide, la stéréovision, etc… Dans le domaine de la PIV, la technique de stéréovision est la plus répandue bien qu'elle nécessite l'apport de modifications par rapport à la technique mono caméra [Coudert 02]. L'utilisation de deux sources d'images présente l'avantage de ne pas développer un système spécifiquement dédié à la PIV.

Dans notre cas, l'architecture sera composée de deux modules d'acquisition. Ainsi, le temps entre deux images sera défini par le temps séparant les deux trames de commande de l'acquisition, donc aisément modifiable. Une modification doit être apportée au module de mémorisation qui maintenant enregistre des couples d'images provenant de deux sources différentes. Le module de mémorisation possédera deux unités d'écriture distinctes, une pour chaque source, et son cycle temporel d'accès mémoire sera piloté par les deux acquisitions.

Figure 7-20 : Positionnement des deux systèmes d'acquisition préconisé.

Pour ne pas avoir à gérer la reconstruction des images de stéréovision par des algorithmes complexes, un positionnement des deux systèmes d'acquisition sur le même axe et l'une en face de l'autre permet la prise d'images du même plan de l'écoulement (Figure 7-20). Un simple algorithme d'effet miroir redonne les deux images successives nécessaire à la corrélation.

2 – Les algorithmes de traitement

La qualité des images influence de façon significative les mesures en sortie du système. Bien qu'un système de traitement en temps réel permette une correction des réglages de l'acquisition immédiatement, les images peuvent présenter des défauts non corrigibles comme par exemple une variation de l'intensité lumineuse ou le processus de digitalisation. Donc, nous devons adapter les algorithmes de traitement à la qualité des images pour minimiser le nombre de vecteurs aberrants tout en respectant la contrainte temps réel.

Actuellement, l'architecture Round-About travaille sur des images binaires car elles permettent une réduction du nombre de calcul à faire. Bien que la binarisation d'une image correspond à une perte d'information, en pratique, elle a peu d'influence en PIV puisque cette méthode fonctionne par reconnaissance de forme sauf dans le cas d'écoulement où les particules ne sont plus discernables après binarisation. De plus, la transformation d'images en niveaux de gris en images binaires est sensible à la qualité de l'image au départ et à la méthode de quantification utilisée. Pour l'instant, notre prototype dispose d'images de bonne qualité, d'où l'utilisation d'un simple algorithme de seuillage binaire sur le flot entrant de données avant leur mémorisation. Cette binarisation par seuillage est

réalisée par comparaison entre la valeur, en niveau de gris, du pixel et un seuil fixé par l'utilisateur du système. Mais, d'autres méthodes peuvent être envisagées tant dans le choix du seuil que dans la réduction de la quantification des niveaux des pixels.

Pour le choix du seuil, plusieurs méthodes existent en traitement d'images comme le calcul de l'histogramme de l'image. Mais, nous souhaitons inclure cette méthode à l'architecture Round-About. Donc, les méthodes retenues devront détecter le plus de formes de particules possibles pour une meilleure estimation du déplacement entre deux images successives, respecter les contraintes de traitement temps réel et occuper un minimum de ressources matérielles. Nous pouvons citer l'algorithme du filtre médian, l'algorithme de Niblack (une binarisation locale adaptative effectuée en prennent en compte les pixels voisins [Trier 95]) ou calculer le contraste moyen de l'image pour servir de seuil.

L'autre point est de ne plus réduire la quantification à seulement deux niveaux. Ce choix implique d'utiliser un autre algorithme de corrélation et non plus l'algorithme de corrélation binaire par comparaison. Pour l'architecture Round-About, cela revient à adapter l'unité de calcul au nouvel algorithme. Dans cet axe de recherche, nous pouvons citer les travaux suivants [Fillali 05a, Fillali 05b] qui proposent l'implémentation de l'algorithme optimisé de corrélation direct sur des images à 3 niveaux de gris sur un système constitué de FPGAs avec une approche temps réel.

7.5.4 Autres applications envisageables

Au départ, l'architecture Round-About a été conçue pour répondre aux contraintes liées à la mesure de vitesse de particules dans un écoulement [Dubois 01b]. Dans un souci de généricité, la conception de l'architecture a été pensée pour répondre à une classe d'applications plus générique : la mesure de paramètres physiques par traitement d'images. Pour étendre encore cette généricité, l'architecture a entièrement été modélisée sous forme de blocs IP hiérarchisés en langage VHDL [Lelong 04a]. Ce découpage permet une grande souplesse de conception, et ainsi l'ensemble de l'architecture est conçu indépendamment du circuit FPGA cible. L'architecture Round-About devient alors une architecture générique, modulable et paramétrable (nombre de modules) en fonction de l'application visée. Donc, d'autres applications peuvent être envisagées pour l'architecture Round-About.

Cette architecture a été bâtie autour de caractéristiques spécifiques à la mesure de paramètres physique par traitement d'image. Donc, nous ne pouvons envisager son utilisation que dans des applications répondant aux mêmes caractéristiques. Ces caractéristiques sont :

- un flot de données d'entrée est important (images) ;
- un flot de sortie est réduit (résultats de mesure) ;
- des traitements réalisés sur des zones réduites de l'image et non sur sa totalité (parallélisation des traitements) ;

- une gestion globale du système centralisée en un seul superviseur.

De plus, la décomposition en blocs IP permet la réutilisation immédiate de certains modules sans aucune adaptation pour différents traitements. De même, un second niveau de découpage à l'intérieur de chaque module permet de conserver une structure homogène entre les différents blocs. Cela offre deux avantages : tout d'abord une réutilisation des unités communes aux différents modules (essentiellement pour les communications), et également une distinction de la partie traitement proprement dite réduisant les modifications ou changements des blocs IP lors de l'adaptation d'un nouvel algorithme.

L'architecture Round-About a été utilisée pour l'implémentation d'un algorithme d'extraction de tables de service d'un flux MPEG2 [Fresse 05]. Cette application répondant aux trois caractéristiques précitées, les auteurs ont ainsi démontré que l'architecture Round-About peut servir à l'implémentation d'autres algorithmes dont l'application répond à ces contraintes.

D'autres applications répondant aussi à ce type de contraintes pourront être implémentées, après adaptation aux FPGAs, grâce à l'architecture Round-About comme le suivie d'objet, le contrôle qualité par vision, etc. L'objectif de l'utilisation de l'architecture Round-About pour une application demeure toujours la réalisation des traitements en temps réel.

Conclusion

Les travaux de recherche présentés dans ce mémoire ont été réalisés dans l'objectif de permettre la mesure en temps réel de paramètres physiques par traitement d'images embarqué sur un système dédié et compact. C'est-à-dire de concevoir une architecture électronique dans un système monopuce réalisant l'acquisition et le traitement d'images pour ne fournir à l'utilisateur qu'une représentation visuelle des mesures effectuées.

Pour illustrer le bon fonctionnement de cet ensemble, nous avons choisi une application d'estimation de mouvement utilisée au sein du laboratoire. Il s'agit de mesurer des vitesses de déplacement dans le cadre de la mécanique des fluides par la technique de la PIV (Vélocimétrie par Image de Particules). Cette technique de mesure choisie est basée sur des algorithmes d'inter-corrélation entre deux images à niveaux de gris (corrélation directe, Transformé de Fourier, etc.). Cette technique utilise un découpage des images en petites fenêtres d'étude ; un vecteur vitesse est calculé pour chaque couple de fenêtres. Il existe, ainsi, un parallélisme implicite des traitements à réaliser. En effet, chaque fenêtre d'étude peut être traitée indépendamment des autres. Comme l'algorithme doit être implémenté dans un système le plus compact possible, nous avons retenu la méthode de corrélation directe sur des images à niveaux de gris pour la simplicité des calculs (multiplications et additions). Cependant, le volume de calcul à faire est assez conséquent, donc peu enclin à être embarquée dans un système sous contrainte temps réel. La réduction de la quantification des images jusqu'à des images binaires permet de simplifier l'algorithme (les multiplications sont remplacées par de comparaison par opérateur logique OU EXCLUSIF). Cette corrélation binaire par comparaison est la solution que nous avons retenue. Bien que le passage à des images binaires représente une perte d'informations, les mesures de déplacement dans les imagettes conservent une précision suffisante pour l'application visée.

Pour pouvoir implémenter cette méthode de mesure par traitement d'images et exploiter les propriétés de l'algorithme retenu, une architecture, nommée *Round-About*, a été définie. C'est une architecture

générique et modulaire pour des applications de type flot de données d'entrée dominant par rapport aux flots de commandes et de données de sortie (résultats), plus réduit. Basée sur la classe MIMD pour exploiter le parallélisme implicite de la méthode de calcul et sur un concept de séparation des flots entrants et sortants, l'architecture *Round-About* se présente sous la forme d'un anneau de communication unidirectionnel reliant différents modules entre eux. Cet anneau est utilisé pour la circulation des commandes et des résultats. Les données, quant à elles, sont transférées via des bus de communication indépendants de l'anneau. Chaque module regroupe les unités nécessaire pour réaliser une fonction précise comme le contrôle, la gestion ou le calcul ainsi que les communications avec les autres modules. Nous avons donc spécialisé l'architecture *Round-About* au domaine d'applications retenues ce qui semble être une tendance actuelle. Cette description hiérarchique permet la modification du nombre et/ou de la fonction de ces modules sans changer l'architecture dans sa globalité. Cela permet de l'adapter aux besoins d'une application, mais aussi aux évolutions de la technologie des composants du système.

Notre architecture modulaire, dédiée au traitement d'images en temps réel, nécessite donc une puissance de calcul importante alliée à des moyens de communication et de mémorisation conséquents. Initialement réalisé par des stations de travail, ce travail peut avantageusement être effectué par des systèmes électroniques embarqués. La capacité des systèmes numériques actuels permet d'intégrer toute l'architecture *Round-About* dans un même composant. Donc, ce travail peut avantageusement être effectué par des systèmes monopuce (SoC) en fournissant de grandes puissances de calcul, une haute intégration et une faible consommation (les rendant ainsi complémentaires des capteurs d'image CMOS APS pour la réalisation de capteurs "intelligents" ou de systèmes de vision). De part sa flexibilité, nous avons opté pour un système monopuce autour d'un circuit logique programmable de type FPGA. Son utilisation nous permet de développer une architecture entrièrement sous forme d'une hiérarchie de blocs IP (écrit en VHDL) pour bénéficier de facilités pour le développement et le test. De plus, cela permet une grande souplesse d'utilisation de l'architecture *Round-About* en facilitant son adaptabilité globale (nombre et nature des modules modifiables).

Pour aboutir à une solution architecturale satisfassent les contraintes de l'application, nous avons adapté l'architecture *Round-About* à la technologie SoPC. Ces adaptations ont portées sur la gestion de la mémoire et la conception de certaines fonctions telles que la gestion et le contrôle de l'ensemble de l'architecture. L'architecture *Round-About*, ainsi modifiée, a été validée en utilisant un FPGA APEX 20K200 avec l'algorithme de corrélation binaire pour les traitements. Cela nous a permis d'évaluer les besoins en ressources matérielles de l'architecture (un peu plus de 50% d'éléments logiques et 90% de la mémoire interne), ainsi que ces performances (un module de calcul peut calculer 8 849 points de mesures par seconde). Nous avons couplé un capteur d'images CMOS au FPGA de manière à pouvoir mettre en œuvre une plate-forme expérimentale de PIV, et ainsi démontrer sur une expérience réelle le respect des contraintes fixées. Avec ce prototype, nous sommes

parvenu au traitement en temps réel des images fournis par le capteur et à visualiser ces résultats en live sur un écran d'ordinateur.

Grâce à l'application PIV, nous avons démontré que l'architecture *Round-About* répondait aux contraintes de traitement temps réel. L'adaptation de l'architecture aux caractéristiques matérielles de notre plateforme de référence, nous a permis de justifier nos choix de développement au vu des avantages d'utilisation de l'architecture *Round-About* (flexibilité et sa souplesse de modifications).

Les prochains travaux porteront sur l'optimisation de l'architecture en elle-même, mais aussi à son évolution par la recherche de meilleures performances et de nouvelles applications de traitement d'images temps réel.