De l'algorithme à l'architecture : définition d'une architecture multi-FPGAs reconfigurable dynamiquement.

Gilles Millon, Michel Doussot, Michel Roussel

LAM - équipe image
LTI - IUT de Troyes
9, Rue de Québec
BP 396 10026 Troyes cedex
tél/fax : 03 25 42 46 43
e-mail : Gilles.Millon@univ-troyes.fr

RESUME

Cette étude est réalisée dans le cadre des réunions de travail du GDR PRC ISIS du CNRS (opération 6.3). L'objectif du groupe de travail est d'étudier l'exploitation de la reconfigurabilité des FPGAs pour le traitement d'image bas niveau en temps réel. L'architecture qui résulte de notre étude est composée de plusieurs FPGAs dédiés au calcul, à la gestion des mémoires et à la communication entre les différents modules. Nous proposons ainsi une alternative aux solutions ASICs ou aux gros systèmes multi-processeurs, toujours couteux et lourds à mettre en oeuvre, pour le traitement d'image en temps réel.

ABSTRACT

This study is carried out in the frame of the working meetings of GDR PRC ISIS (operation 6.3). The aim of the working group is to study the exploitation of the FPGA reconfigurability for real time low level image processing.
The architecture resulting from our study consists of several FPGAs dedicated to calculation, memory management and communication between various modules. Thus we propose an alternative to ASICs solutions or large multi-processors system, that are always expensive and hard to set up, for real time image processing.

1 Introduction

Le traitement des images en temps réel se heurte au débit de l'information fournie par la caméra, qui dépasse 6 Moctets par seconde. Ainsi, les systèmes à mettre en oeuvre pour traiter ces données reposent souvent sur des machines multi-processeurs encore difficiles à programmer, ou bien sur des solutions ASICs où doit intervenir un fondeur. Le coût financier et/ou temporel rend ces solutions inadaptées pour des industriels désireux de dévelolpper des systèmes simples, peu couteux et destinés à de très petites séries.

Une alternative consiste à opter pour les FPGAs qui présentent une simplicité de mise en oeuvre et un coût maîtrisé pour une efficacité correspondant aux besoins des industriels. En outre, la reconfigurabilité de ces circuits à mémoire SRAM permet d'implanter différentes applications sur la même architecture, lui donnant ainsi une polyvalence très appréciable. Cette reconfigurabilité peut aussi être vue comme un moyen de tester et valider l'implantation matérielle d'algorithmes de traitement d'image bas niveau. Dans tous ces cas, nous parlons de reconfigurabilité statique.

L'idée que souhaite développer le groupe de travail s'appuie sur la reconfigurabilité dynamique des FPGAs . Il s'agit alors de décomposer une chaîne de traitement en plusieurs algorithmes qui viendront successivement configurer l'architecture pour réaliser le traitement complet (figure 1). Le principe de traitement permet d'extraire un certain nombre de caractéristiques pour définir l'architecture. Notamment, nous pourrons définir la largeur des bus de données et d'adresses, le type et la taille des mémoires ainsi que les caractéristiques des FPGAs et l'organisation de leur utilisation.

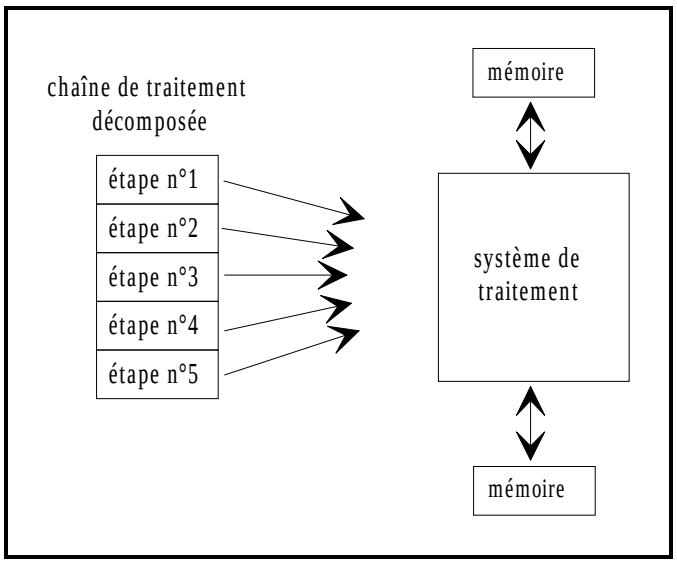

Figure 1: *Principe de la reconfiguration dynamique.*

Pour mener à bien cette étude, le groupe s'est proposé de l'associer à une chaîne de traitement concrète, développée par l'équipe de Didier Demigny de l'ENSEA[1], qui réalise la détection et la fermeture de contours en temps

réel, suivie de l'étiquetage des régions délimitées par les contours fermés. Cependant, l'architecture proposée doit pouvoir supporter aux besoins d'autres algorithmes de traitement d'image bas niveau en temps réel pour exploiter la polyvalence offerte par la reconfigurabilité du système.

2 Les flux de données

L'exploitation de la reconfigurabilité nous conduit à disposer de 2 plans mémoire dans la mesure où les données à traiter par un algorithme sont les données traitées par l'algorithme précédent (figure 1). Il faut noter que la cadence des images délivrées sera restreinte puisque le flux vidéo est délaissé lorsque les données sont traitées par les étapes 2, 3, 4, etc. Les plans mémoire sont de la taille des images standard (512 x 512) puisqu'il s'agit ici de traitement bas niveau où le résultat d'un traitement demeure une image. L'étude des flux de données transitant d'un plan mémoire à l'autre via le système nous permet de définir les caractéristiques de l'architecture.

2.1 Le traitement d'images entrelacées.

Les caméras standard fournissent une image entrelacée que nous numérisons. Le traitement de ces données, synchronisé avec le flux vidéo, conduit à une organisation matérielle particulière dans la mesure où deux pixels spatialement décalés d'une ligne sont temporellement décalés d'une trame[2]. Il s'agit alors de mémoriser des trames sources ou résultats et de calculer en parallèle les pixels homologues des deux trames (figure 2).

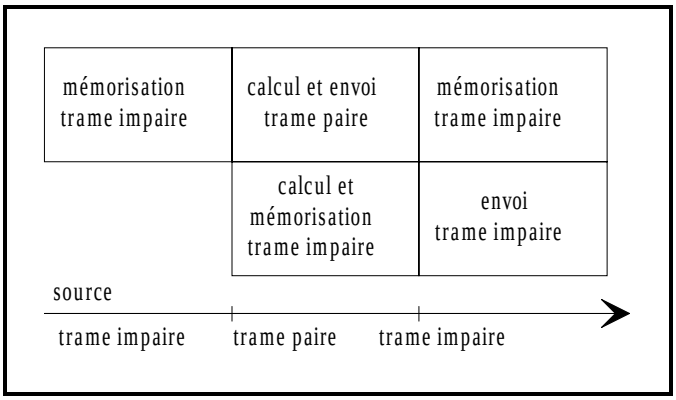

Figure 2 : Principe de traitement des images entrelacées.

Lorsqu'une trame est délivrée par la caméra, elle est mémorisée dans un premier plan. Lorsque la trame suivante se présente, la précédente est relue en mémoire. Nous disposons alors des deux pixels homologues dans chacune des trames. Le calcul est mené en parallèle sur chacune d'elles et fournit deux pixels résultats simultanément. Deux cas peuvent alors se présenter :

- si nous travaillons dans le cadre de la reconfiguration dynamique, les deux résultats sont mémorisés;
- sinon l'un des résultats est envoyé au système de réception tandis que l'autre est mémorisé dans un second plan mémoire. Ce second résultat sera délivré au système de réception tandis que la première trame de l'image suivante sera mémorisée dans le premier plan.

Dans le cadre de la décomposition, les calculs intermédiaires pourront être menés indépendamment de la synchronisation vidéo. Cela permet à la fois d'accéder aux données de façon non-entrelacée et de travailler à des fréquences supérieures. Le gain de temps ainsi obtenu nous rapprochera alors de la cadence vidéo standard de 25 images par seconde.

2.2 Le traitement en flot de données.

Le balayage par ligne de l'image n'autorise pas un accès aléatoire aux données dans l'image. Les algorithmes travaillent donc systématiquement sur des fenêtres locales et de taille limitée.
Plusieurs idées se présentent pour disposer des données des lignes précédant celles des pixels courants.

- Mémoriser ces données dans des FIFOs internes au FPGA. Cette proposition, qui mobilise beaucoup de ressources logiques dans le circuit, est à écarter dans l'état actuel de la technologie des circuits.
- Lire les données dans la mémoire conventionelle. La gestion de l'adressage pour simuler la FIFO sera, là encore, exigeante en ressources logiques et ceci d'autant plus qu'il faudra accéder à plusieurs lignes précédentes dans l'image. Le problème majeur reposera en fait sur la fréquence d'accès à la mémoire qui augmentera très rapidement.
- Implanter des circuits FIFO ligne autour du FPGA. Nous avons retenu cette solution qui limite la difficulté pour accéder à ces données mais il faut noter une augmentation significative des besoins en entrées/sorties sur le FPGA pour acceuillir ces FIFOs.

En ce qui concerne la disponibilité des pixels dans une ligne, la mémorisation est intégrée dans le FPGA par des registres.

2.3 Les algorithmes à implanter.

La chaîne de traitement étudiée est constituée de plusieurs algorithmes[1],[4],[5].

- Le filtrage de Nagao a été modifié pour se prêter à une implantation matérielle. Ce filtre travaille sur une fenêtre 5x5 décomposée en 9 voisinages 3x3. Pour chacun d'eux, nous calculons la somme et l'étendue des niveaux de gris des pixels. Nous affectons alors au pixel central de la fenêtre 5x5 la somme liée à l'étendue minimale parmi les voisinages 3x3. Il faut donc disposer d'au moins 4 FIFOs sur 8 bits pour accéder aux données nécessaires. Les données résultantes sont des pixels sur 12 bits. Le bruit dans l'image est éliminé tandis que le contraste caractéristique des contours est, lui, réhaussé. Ceci prépare bien la détection des contours qui suivra.

- Le calcul des gradients horizontaux et verticaux permet d'en déduire le gradient (13 bits) et sa direction (2 bits)

qui caractérisent les contours. Le traitement s'effectue sur une fenêtre 2x2 qui correspond alors au besoin d'une FIFO sur 12 bits.

- La suppression des non-maxima locaux dans la direction du gradient associée à un double seuillage fournit des contours d'épaisseur un pixel et trie les pixels parmi 3 classes. Pour les gradients supérieurs au seuil haut, les pixels sont considérés comme des contours dont nous sommes sûrs. Pour les gradients inférieurs au seuil bas, les pixels sont classés comme des points de fond dans l'image. Enfin, les gradients situés entre les deux seuils sont considérés comme des points de contours potentiels, appelés crêtes, qui seront retraités par la suite pour les définir comme des contours ou du fond. Cet algorithme travaille sur une fenêtre 3x3 qui nécessite par conséquent deux FIFOs sur 15 bits. Les données résultant de ce traitement sont sur 2 bits.

- La phase suivante améliore les contours obtenus en traitant les cas simples de discontinuité. Cet algorithme travaille sur une fenêtre 3x3. Les FIFOs seront ici sur 15 bits dans la mesure où l'algorithme a besoin de l'information gradient. Les pixels demeurent sur 2 bits.

- L'algorithme qui suit, reconstruit les contours en connexité 4 et travaille sur une fenêtre 2x2 en utilisant, là encore, l'information gradient.

- Pour préparer la fermeture des contours, une nouvelle classe de pixels est créée: l'extrémité. C'est un contour sûr n'ayant qu'un seul voisin de type contour sûr. Cet algorithme travaille sur une fenêtre 3x3 et fournit des pixels sur 2 bits.

- La fermeture des contours s'effectue en deux phases. La première trie les pixels parmi le fond ou les contours et étiquette les pixels crêtes pour lesquels la décision ne peut être prise. La seconde phase résoud les équivallences entre crêtes pour obtenir des contours fermés, soit un bit par pixel. Cette étape nécessite une mémoire supplémentaire pour les équivalences entre crêtes.

- La dernière étape réalise l'étiquetage des régions délimitées par les contours fermés. Cet algorithme est similaire à celui de la fermeture des contours. Il travaille sur une fenêtre 5x3 et nécessite deux FIFOs.

3. L'architecture.

L'étude des algorithmes de la chaîne nous fournit donc le type et la taille des mémoires et des données manipulées. Cela nous permet de définir les caractéristiques de notre architecture.

3.1 Estimation des entrées/sorties.

Classiquement, les pixels sont codés sur 8 bits mais le traitement de ceux-ci fournit des informations de taille supérieure. Nous étendrons donc les données à 16 bits. Les FIFOs doivent donc être au nombre de 4 sur 16 bits. Les bus d'adressage des plans mémoire (512x512) sont portés à 18 bits. Les circuits disponibles travaillent sur 8 bits. Nous les piloterons de manière indépendante pour donner un maximum de souplesse à notre architecture. Ainsi nous diposons de 4 plans image (512x512x8 bits) et 8 FIFOs 512x8 bits.
Si la gestion indépendante des modules permet d'exploiter au mieux l'architecture, elle est synonyme d'un nombre d'entrées/sorties incompatible avec celui dont disposent même les plus gros circuits (figure3)[3].

Nature des entrées/sorties	Nombre
bus d'adresses des mémoires	4 x 18
signaux de contrôle des mémoires	4 x 3
bus de données des mémoires (i/o séparées)	4 x 16
bus de données des FIFOs	6 x 16
signaux de contrôle des FIFOs	8 x 5
signaux de synchronisation de la vidéo, divers	10
Total	**294**

Figure 3: *Estimation des entrées/sorties pour le FPGA.*

3.2 Définition de l'architecture multi-FPGAs.

A la vue de l'estimation des entrées/sorties nécessaires (figure 3), il faut nous diriger vers une architecture multi-FPGAs. Si la gestion des mémoires demeure synchrone vis à vis du traitement des données, elle reste la plupart du temps indépendante du traitement de l'information proprement dit. Nous avons donc des FPGAs de gestion mémoire et des circuits dits de calcul. Pour le cas où les ressources en CLBs d'un FPGA ne suffiraient pas pour implanter un algorithme, nous avons réfléchi à une architecture multi-FPGAs au niveau du traitement des données. Pour cela nous associons à l'architecture, un FPGA dit de communication qui pourra répartir les données vers chacun des FPGAs de traitement. La figure 4 montre l'architecture résultant de notre étude.
Il faut noter que plusieurs configurations de travail sont alors possibles puisque l'on peut les faire travailler :

- avec les mêmes algorithmes et des données différentes (cas de la duplication d'algorithme pour le traitement des images entrelacées);
- avec des algorithmes et des données différents. Il s'agit ici du cas de la concaténation des algorithmes dans la décomposition de la chaîne pour réduire le nombre d'étapes, et augmenter par là, la cadence des images délivrées;
- avec des algorithmes différents et des données identiques pour le cas où l'on souhaiterait comparer deux traitements et choisir le meilleur résultat.

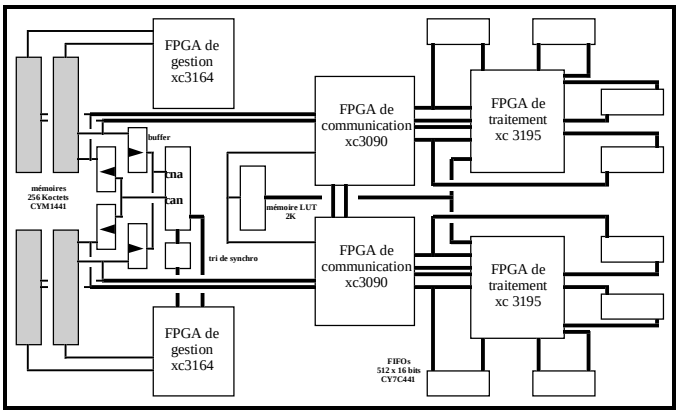

Figure 4 : *Architecture multi-FPGA proposée.*

3.3 Implantation.

L'architecture proposée a été réalisée et le filtrage de Nagao ainsi que le calcul du gradient et de sa direction ont été décrits en VHDL [6],[7]. La simulation après routage a permis de valider cette implantation. Le filtrage de Nagao est implanté pour une partie sur le XC3195 et pour l'autre partie sur le XC3090. Pour le XC3195, le multiplexage des données vers un opérateur unique a été préféré à la duplication de l'algorithme liée au traitement d'image entrelacée. Nous avons pour contrainte supplémentaire une fréquence de fonctionnement doublée pour respecter le temps réel. L'implantation a été modifié vis à vis de la description naturelle de l'algorithme pour réduire les besoins en CLBs. Notamment, les calculs de somme et d'étendue ont été menés sur un seul voisinnage 3x3, au lieu des 9, avec une sauvegarde des résultats intermédiaires dans les FIFOs. Ceci nous a conduit à utiliser les 8 FIFOs disponibles autour du circuit. 389 CLBs sont utilisés pour 484 disponibles. La fréquence de fonctionnement atteint 22.5 Mhz pour la partie devant fonctionner à 20 Mhz et 11.6 Mhz pour 10 Mhz.
La seconde partie de l'algorithme a été implantée sur le XC3090 dans la mesure où elle n'utilise pas de FIFO. Nous utilisons alors 296 CLBs pour 320 disponibles sur le circuit, avec des fréquences de fonctionnement de 25.1 Mhz pour 20 Mhz et 11.1 Mhz pour 10 Mhz.

L'implantation du gradient sur le second XC3195 met en oeuvre deux FIFOs sur 8 bits et occupe 346 des 484 CLBs avec une fréquence de fonctionnement de 25.1 Mhz pour 20 Mhz comme contrainte de temps réel.
Tout ceci représente donc une étape dans la décomposition de la chaîne de traitement.

4. Conclusion.

L'étude des flux de données a permis de définir une architecture multi-FPGAs pour l'exploitation de la reconfigurabilité de ces circuits. L'implantation des premiers algorithmes de la chaîne étudiée est une première étape pour valider cette architecture en mettant en évidence sa souplesse pour s'adapter aux algorithmes que l'on souhaite implanter. Nous pouvons souligner ici l'adéquation réalisée entre l'algorithme et l'architecture par la définition de l'architecture en fonction des caractéristiques de l'algorithme et par la modification de l'algorithme pour s'adapter à l'architecture (cas du filtrage de Nagao).
L'architecture proposée permet de recevoir une chaîne de traitement d'image telle que celle développée par l'équipe de D. Demigny.
Les circuits disponibles actuellement sur le marché voient leur reconfigurabilité trop lente pour une utilisation en dynamique. La série XC6200 de XILINX autorisera des reconfigurations rapides et partielles de l'ordre de la milliseconde. L'implantation des algorithmes ne permet actuellement que de valider l'architecture proposée.
Un projet appelé ARDOISE (Architecture Reconfigurable Dynamiquement Orienté Image et Signal Embarquable) regroupant une dizaine d'équipe du GDR PRC ISIS est lancé pour poursuivre l'étude. Les différentes études d'architecture réalisées par les équipes de l'opération 6.3 pourront alors servir de base de travail pour ce projet.

Références bibliographiques:

[1] Quesne J.F. ″Vision robotique: architecture data-flow pour le traitement des images en temps réel″, Thèse de doctorat de l'université de Paris-Sud(centre d'Orsay) , 1992.

[2] Doussot M. ″ *Etude, réalisation et applications d'opérateurs à la cadence vidéo utilisant des architectures rogrammables*″, Thèse de doctorat de l'université de Reims Champagne Ardenne, 1994.

[3] Xilinx ″ *The programmable gate array data book*″ 1995.

[4] Demigny D., Devars J., Kessal L. and Quesne J.F. ″*Implantation temps réel du filtre de lissage d'image de Nagao*″, Traitement du signal, vol. 10, n°4, 1993.

[5] Jaber J. ″*Définition et validation d'une architecture électronique rapide d'étiquetage et de caractérisation d'objets dans une image*″, Thèse de doctorat de l'Université de Nancy I, 1993.

[6] Airiau R., Bergé J.M., Olive V. and Rouillard J. ″*VHDL du langage à la modélisation*″, Presses polytechniques et universitaires romandes, 1990.

[7] VIEWlogic ″*VHDL reference manual*″, 1994.