Dix-septième colloque GRETSI, Vannes, 13-17 septembre 1999

# La Reconfiguration dynamique des FPGA : Que doit-on en attendre ?

Didier DEMIGNY, Ryad BOURGUIBA, Lounis KESSAL, Michel LECLERC

ETIS, UPRESA CNRS 8051
ENSEA/UCP 6 av. du Ponceau, 95014 Cergy Pontoise Cedex, France
demigny@ensea.fr, bourguiba@ensea.fr
kessal@ensea.fr, leclerc@ensea.fr

**Résumé** – Nous discutons brièvement du choix de la technologie pour le traitement des images temps réel (ASIC, FPGA, DSP). Partant du constat que les FPGA sont souvent sous-exploités, nous analysons quantitativement l'amélioration du rendement silicium qui peut être obtenue en exploitant la reconfiguration dynamique. Nous mettons en évidence des limites à son utilisation liées à la puissance de calcul, à l'application envisagée et à la technologie. Finalement nous présentons le cas de l'architecture ARDOISE pour une application de traitement d'images. Nous montrons alors l'intérêt de disposer d'outils de synthèse de haut niveau pour le placement et l'ordonnancement des algorithmes de l'application.

**Abstract** – We briefly discuss the choice of the technology for real time image processing (ASIC, FPGA, DSP). We quantitavely studie the reduction of the silicium size which can be obtained by the use of the dynamical reconfigurability of the FPGA. We present the case of ARDOISE architecture for an image processing application. We show the need of high level synthesis tools for the placement and the schedule of the application algorithms.

## 1 Introduction

Dans le cadre de l'action incitative "systèmes reconfigurables et traitement d'images" a été entrepris la conception d'une architecture mixte FPGA/DSP baptisée ARDOISE. L'ambition d'ARDOISE est de pouvoir enchaîner les exécutions d'un flot d'algorithmes temps réel de segmentation d'images sur une même structure physique en reconfigurant plusieurs fois les FPGA au cours du traitement d'une même image [1]. Le principal intérêt d'un système reconfigurable est de pouvoir changer à tout moment les algorithmes exécutés par le matériel, rappelons que cela est fait depuis bien longtemps par les microprocesseurs. La reconfiguration dynamique n'a donc de sens que si elle s'accompagne de performances (à coûts identiques) bien supérieures à celles des structures programmées classiques. Or, on constate que les processeurs conventionnels ou les DSP (Pentium, C60, C80, SHARC) sont capables d'exécuter un alogrithme de traitement d'images relativement complexe - *filtre de Deriche* - en temps réel [3], ce qui semble discréditer quelque peu la technologie FPGA. On remarque cependant que les processeurs sont alors en limite d'utilisation : boucle interne de deux ou trois cycles optimisée au cycle près ; toute modification même minime remet en cause la faisabilité globale sous contrainte temps réel. D'autre part, ce résultat est obtenu pour des fréquences de travail supérieures d'un facteur dix par rapport aux solutions FPGA, les problèmes de dissipation de puissances en limitent alors l'usage dans les systèmes embarqués. Il semble que si les solutions programmées se placent en concurrence des solutions FPGA, c'est aussi parce que ces derniers sont souvent sous-employés comme nous le montrerons dans ce papier. La reconfiguration dynamique (RD) est une solution architecturale et technologique qui ne permet pas des gains en vitesse de traitement mais qui réduit la complexité de circuit nécessaire à une application donnée. Par une approche quantitative, qui s'appuie sur les travaux initiaux de H. Guermoud [2], nous mettons en évidence le gain potentiel et surtout les limitations d'utilisation liées d'une part à la complexité de l'application et d'autre part à la vitesse de configuration de la technologie.

## 2 Analyse quantitative des performances de la RD

On considère l'exécution d'un ensemble d'algorithmes sur une architecture mono ou multi FPGA. On envisage le mode de configuration statique *sans et avec tampon d'image* et le mode de configuration dynamique. Le traitement s'applique à un flot de données d'entrées constituant un bloc (flot de pixels constituant une image par exemple). Bien qu'illustrer par le traitement d'images, cette analyse est entièrement transposable à d'autres champs applicatifs.

### 2.1 Quelques notations

L'indice $s$ est réservé aux configurations statiques et $d$ aux configurations dynamiques. $G$ est le nombre de portes équivalentes utilisées pour l'implantation complète du traitement. Cette mesure permet de s'affranchir des disparités entre les complexités des blocs logiques des différents types de FPGA. $F_e$ est la fréquence d'échantillonnage des données ( fréquence pixel caméra dans le cas du traitement d'images). $F_t$ est la fréquence limite d'utilisation de la

technologie. Nous supposons que la fréquence limite de la technologie est liée à la rapidité des mémoires. Ceci semble légitime en traitement d'images compte tenu de l'importance des accès mémoires. Il est cependant possible de considérer $F_t$ comme étant la fréquence de calcul interne des FPGA qui peut être estimée entre la moitié et le tiers de la vitesse donnée par le fabricant. $T$ est la durée d'acquisition d'un bloc de donnée auquel est appliqué l'ensemble du traitement. Lorsqu'on utilise la RD, une suite de configurations est appliquée aux FPGA pour réaliser l'ensemble du traitement sur le bloc de donnée pendant la durée $T$. $\alpha$ est le taux d'échantillons utiles dans le bloc. Ce paramètre permet par exemple de prendre en compte les durées de retour ligne et trame en traitement d'images. $V_c$ est la vitesse de reconfiguration : nombre de portes équivalentes reconfigurées par seconde. Cette mesure ne dépend que du type de FPGA. On notera $G_g$ le gain relatif en portes équivalentes apporter par le mode reconfiguration dynamique par rapport au mode statique.

## 2.2 Configuration statique sans tampon d'images

Ce mode d'utilisation classique nous sert de référence pour discuter l'apport de la RD. L'absence de tampon d'image implique que la fréquence de traitement est égale à la fréquence d'échantillonnage des données et qu'on ne peut effectuer aucun traitement significatif pendant l'acquisition de données non utiles (retours ligne, trame).

Soit $G_s$ le nombre de portes utilisées. On définit la puissance utile $Pu$ nécessaire à l'application temps réel comme :

$$Pu_s = G_s \, \alpha \, F_e \tag{1}$$

$Pu$ est proportionnelle au "bops" *bit operation per second* définit par J. Vuillemin et à l'avantage d'une meilleure indépendance au type de FPGA sans toutefois permettre des équivalences immédiates avec des mesures de puissances utilisées pour les $\mu$processeurs.

On définit de même la puissance maximale $Pm$ offerte par l'architecture :

$$Pm_s = P_s \, F_t \tag{2}$$

Ces deux mesures permettent d'introduire un rendement $\eta_s$ ou taux d'utilisation du matériel :

$$\eta_s = \frac{Pu_s}{Pm_s} = \frac{\alpha \, F_e}{F_t} \tag{3}$$

Améliorer le rendement $\eta_s$ suppose qu'on puisse diminuer le nombre de portes utilisées en augmentant la fréquence de traitement, ce qui est impossible en configuration statique sans tampon pour les raisons invoquées au début de ce paragraphe. Le tableau 1 illustre le cas d'images de taille $512^2$ et $1024^2$ avec dans ce dernier cas l'acquisition de deux pixels simultanés. La période trame valant toujours $40ms$. Nous supposons dans cet exemple, que la fréquence limite de la technologie, liée à la rapidité des mémoires, est de $25ns$.

TAB. 1: Rendement en configuration statique

| image | $\alpha$ | $F_e$ | $F_t$ | $\eta_s$ |
|---|---|---|---|---|
| $512^2$ | 0,665 | 10 Mhz | 40 Mhz | 16,6% |
| $1024^2$ | 0,665 | 20 Mhz | 40 Mhz | 33,2% |

## 2.3 Configuration statique avec tampon d'images

L'adjonction d'un tampon d'images permet de travailler à fréquence maximale même pendant les temps morts de l'acquisition et donc de réduire la durée du calcul. On remarquera que cela ne permet pas de réduire le nombre de portes $G_s$ utilisées,.sauf dans le cas de traitements identiques successifs qui peuvent être alors repliés sur le même matériel. C'est en fait un cas de RD où l'identité des traitements rend inutile une reconfiguration. Un autre intérêt est de pouvoir traiter plusieurs images dont les pixels sont entrelacés ( plans d'images couleurs, stéréovision,...). Le tampon d'images sera bien évidemment indispensable à l'utilisation de la RD. Notons aussi qu'il est fonctionnellement indispensable dans le cas d'accès aléatoires aux pixels de l'image d'origine.

## 2.4 Configuration dynamique

La RD a pour but de réduire le nombre de portes utilisées dans un rapport $G_g$.

$$G_g = \frac{G_s}{G_d} \tag{4}$$

On fait l'hypothèse qu'un parallélisme de donnée ($\beta$) peut être exploité et que le nombre de portes utilisées évolue proportionnellement à $\beta$. En général, cette dernière hypothèse est pessimiste parce que certains calculs peuvent être mis en commun. Le nombre de configurations $C$ nécessaire au traitement complet est alors tel que :

$$G_g = \frac{C}{\beta} \tag{5}$$

On suppose ici que le partitionnement est idéal, que les traitements de chaque configuration ont la même durée et que les besoins initiaux en débit mémoire sont uniformément répartis sur les différentes configurations. Ceux-ci sont alors réduits dans le rapport $G_g$.

La durée $D$ d'un traitement effectué à la fréquence $F_t$ vaut alors :

$$D = \frac{\alpha \, F_e \, T}{\beta \, F_t} \tag{6}$$

### 2.4.1 Reconfiguration non masquée

Le respect de la contrainte temps réel, en tenant compte des temps de reconfiguration non masqués, impose alors :

$$C \left[ \frac{\alpha \, F_e}{\beta \, F_t} + \frac{G_d}{V_c \, T} \right] \leq 1 \tag{7}$$

L'égalité correspond à l'utilisation complète de la durée $T$ et donc à la plus grande réduction du nombre de portes.

La figure 1 présente l'évolution de $C$ en fonction du nombre de portes $G_d$ pour deux technologies ( Xilinx 4000

$V_c = 45.10^4$ et Atmel 40K $V_c = 45.10^6$ ) et différentes valeurs de $\beta$ (1,2,4) avec les paramètres définis précédemment pour une image $512^2$.

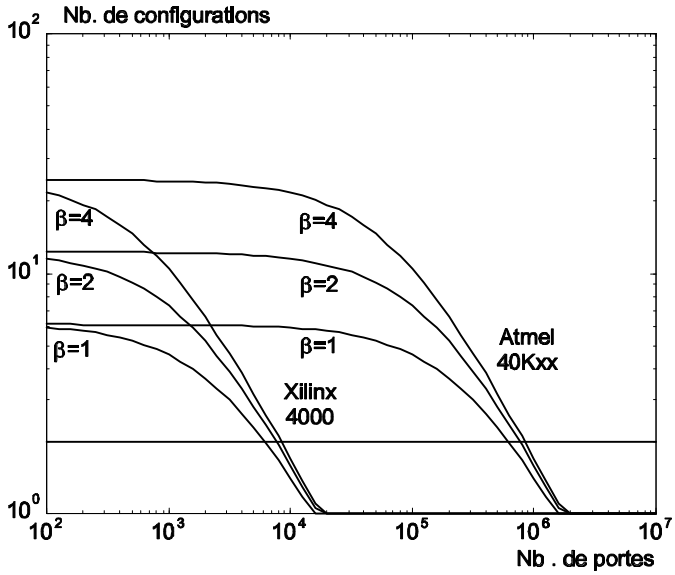


FIG. 1: Nombre de configurations en fonction du nombre de portes $G_d$

La ligne horizontale délimite la zone d'intérêt ($C > 2$). Augmenter $\beta$ réduit la durée du traitement d'une configuration et conduit, à nombre de portes fixé, à augmenter le nombre de configurations pour effectuer le traitement complet. On constate l'importance d'une technologie adaptée ; la faible vitesse de configuration de la famille Xilinx 4000 réduit l'utilisation de la RD à un cas ou le nombre de portes est très faible, valeur bien en dessous des besoins de traitement d'images.

La puissance utile reste inchangée par rapport à la configuration statique. C'est la puissance maximale qui a été réduite par la diminution du nombre de portes et donc le rendement augmenté. On obtient :

$$Pu = \frac{\alpha F_e G_d}{\frac{\alpha F_e}{F_t} + \frac{\beta G_d}{V_c T}} \tag{8}$$

Ce résultat est particulièrement intéressant. Il montre qu'il existe une limite en puissance utile $Pu_{\max}$ pour l'utilisation de la RD dépendant de l'application et de la vitesse de configuration. On notera également que le parallélisme de donnée réduit $Pu_{\max}$, ce qui peut définir un critère pour le partionnement.

$$Pu_{\max} = \frac{\alpha F_e T V_c}{\beta} \tag{9}$$

Ce résultat se traduit aussi par une correspondance entre la complexité de l'application (nombre de portes statiques maximum) et le produit durée de bloc × vitesse de configuration.

$$G_{s\max} = T V_c \tag{10}$$

La figure 2 montre l'évolution du gain $G_g$ en nombre de portes en fonction de la puissance utile $Pu$ avec les mêmes paramètres que précédemment. On observe l'existence de la limite asymptotique de $Pu$ discutée ci-dessus. La RD est utilisable jusqu'à 90% de $Pu_{\max}$ ; le temps de configuration est alors de 10% du temps total de traitement. Pour la famille Atmel 40kxx, une réduction d'un facteur 5 du nombre de portes peut être obtenue pour des applications nécessitant initialement 45000 portes.

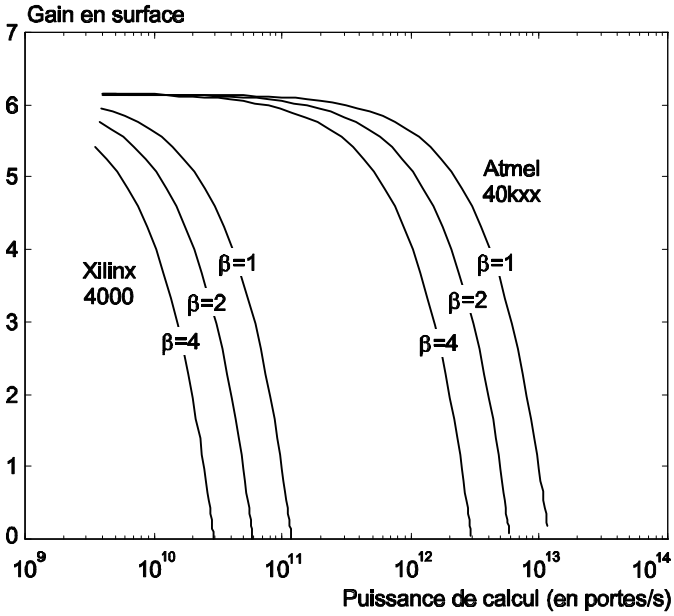


FIG. 2: Gain en surface en fonction de la puissance de calcul nécessaire à l'application

S'il est évident que la vitesse de configuration est un critère essentiel pour l'efficacité de la RD, l'équation 10 montre l'importance de la durée de bloc. Prenons l'exemple d'une application en commande rapprochée vectorielle de machines électriques. Typiquement la fréquence de commutation des onduleurs est de 20 Khz correspondant à $T = 50\mu s$. Dans ce cas $G_{s\max} = 2250$ portes, ce qui est bien inférieur à la complexité de ce genre d'application. La RD n'est pas utilisable dans ce cas.

Le rendement de l'architecture utilisant la RD a pour expression :

$$\eta_d = \frac{1}{1 + \frac{\beta G_d F_t}{\alpha F_e T V_c}} \tag{11}$$

La figure 3 montre qu'une amélioration significative du rendement peut être obtenue grâce à la RD.

Il reste à examiner en quoi le masquage des temps de reconfiguration modifie ces résultats. Deux types de solutions peuvent être employées :

- une solution architecturale qui consiste à doubler le nombre de circuits et par un mécanisme de ping-pong à effectuer le traitement sur un circuit pendant que l'autre est en cours de configuration ;
- une solution technologique qui consiste à modifier la structure des FPGA de façon à effectuer simultanément chargement de la configuration suivante et traitement ; le basculement étant instantané en fin de traitement.

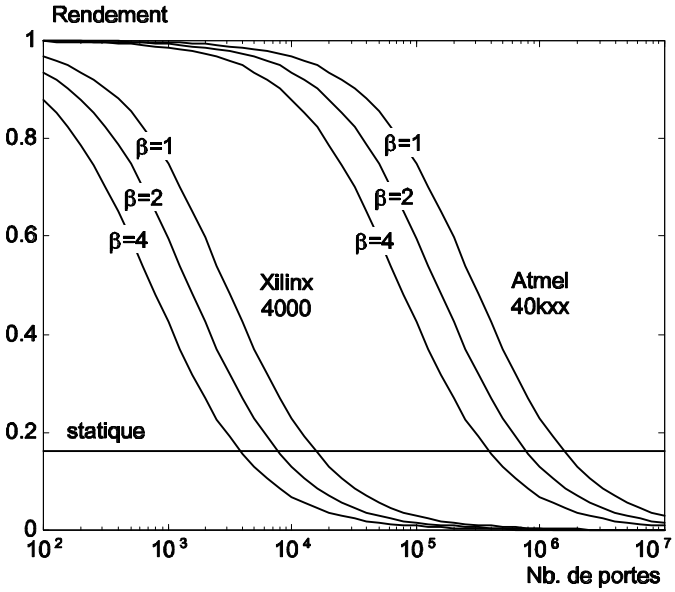


FIG. 3: Amélioration du rendement apportée par la RD

### 2.4.2 Reconfiguration avec masquage ( solution architecturale )

Deux cas sont à envisager suivant que la durée d'un traitement élémentaire est inférieure ou supérieure au temps de chargement d'une nouvelle configuration. L'identité est atteinte pour :

$$\frac{\alpha\, F_e\, T}{\beta\, F_t} = \frac{G_d}{V_c} \qquad (12)$$

où $G_d$ est le nombre de portes des circuits effectuant le traitement ( identique au nombre de portes des circuits en cours de configuration ).

Si la durée de traitement est supérieure à la durée de chargement d'une nouvelle configuration, la contrainte temps réelle impose :

$$C \left[ \frac{\alpha\, F_e}{\beta\, F_t} \right] \geq 1 \qquad (13)$$

La puissance utile maximale correspond à l'égalité (12) et est identique au cas non masqué (9). Le gain $G_g$ est alors indépendant du nombre de portes et reste toujours inférieur au gain obtenu sans masquage. Le facteur 2 est dû au doublement du nombre de circuits. Il faut aussi un minimum de 4 configurations pour qu'on puisse parler de reconfiguration dynamique.

$$G_g = \frac{F_t}{2\,\alpha\, F_e} \qquad (14)$$

Si la durée de chargement de configuration est supérieure à la durée de traitement, la contrainte temporelle impose :

$$C \left[ \frac{G_d}{V_c\, T} \right] \geq 1 \qquad (15)$$

On montre alors aisément que la puissance utile reste constante et égale à sa valeur maximale. Augmenter le nombre de portes n'est d'aucune utilité. L'identité (12) fixe donc le nombre maximum de portes pour lequel la RD est utilisable. La reconfiguration avec masquage utilisant une solution de type ping-pong ne présente donc aucun intérêt.

### 2.4.3 Reconfiguration avec masquage ( solution technologique )

les résultats précédents s'appliquent encore dans ce cas sauf au niveau du gain $G_g$ qui est cette fois supérieur au cas non masqué.

La figure 4 montre l'évolution de $G_g$ dans les 3 cas de reconfiguration en fonction du nombre de portes pour $\beta = 1$.

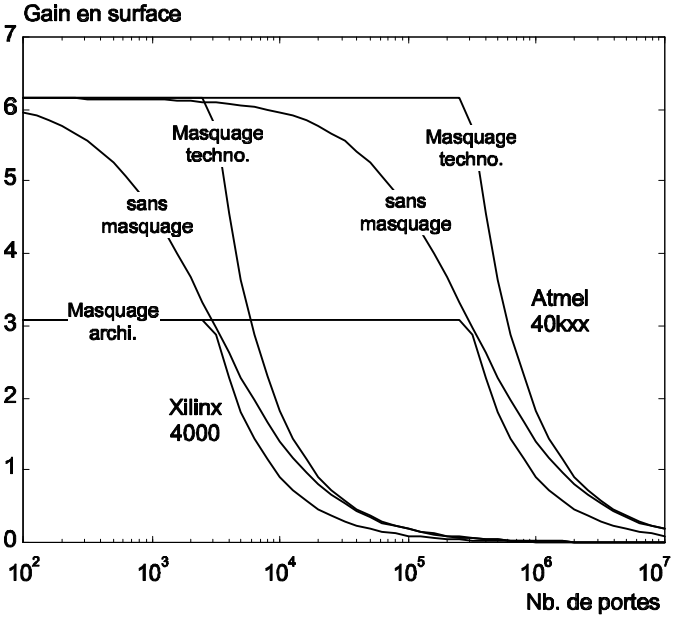


FIG. 4: Influence des stratégies de configuration sur le gain en surface

[1] Riad Bourguiba, Didier Demigny, Mamoud Karabernou, and Lounis Kessal. Designing a new architecture for real time image analysis with dinamically cpnfigurable fpgas. In *Proc. IEEE - IMACS Computational Engineering in Systems Applications*, pages 739–743, Nabeul Hammamet, Tunisia, may 1998. IEEE Systems Man and Cybernetics.

[2] H.Guermoud, S. Weber, Y. Bervillier, and E. Tisserand. Reconfiguration dynamique des fpga-sram pour l'optimisation d'architectures matérielles dédiées aux traitement d'images temps réel. *Journées Adéquation Algorithme Architecture*, pages 61 – 68, 1998.

[3] L. Lacassagne and P. Garda. Exécution temps réel des détecteurs de contours de deriche par des profcesseurs risc d'architectures. *Journées Adéquation Algorithme Architecture*, pages 251 – 258, 1998.