Implantation du décodeur entropique de JPEG 2000 basée sur la reconfiguration dynamique des FPGAs

Sophie BOUCHOUX, El-Bay BOURENNANE

LE2I - Université de Bourgogne
Aile H - Sciences Mirande, B.P. 47870
21078 Dijon cedex, France
sbouchou@u-bourgogne.fr, ebourenn@u-bourgogne.fr

Résumé – Cet article a pour objet la mise en œuvre de la reconfiguration dynamique des FPGAs. L'application envisagée ici, est le décodeur entropique de JPEG 2000 et la procédure MQ-Décode. Des critères d'analyse pour les différentes implantations seront tout d'abord définis, puis nous présenterons la norme JPEG 2000 et l'architecture employée.

Abstract – This article is about dynamic reconfiguration of FPGA. The application implemented to illustrate this case is entropic decoder and arithmetic decoder of JPEG 2000. Some analysis criteria will be defined, JPEG 2000 norm described and the architecture used presented.

1 Introduction

Le standard de compression JPEG 2000 mis au point par le groupe JPEG a pour but d'offrir une alternative à l'actuel JPEG pour la compression d'images fixes. En effet, le standard actuel JPEG ne permet pas de palier à certains défauts ou artéfacts et n'est pas adapté au codage de certains types d'images. JPEG 2000 permet généralement d'obtenir une meilleure qualité d'image à taux de compression équivalent que JPEG. Nous avons choisi d'implanter une partie de cet algorithme sur une architecture à base de FPGA en utilisant la propriété de reconfiguration dynamique de ces derniers. La première partie de cet article est consacrée à la définition de critères d'analyse qui permettront d'évaluer les apports de la reconfiguration dynamique par rapport à la configuration statique. La seconde partie est consacrée à la description de l'algorithme JPEG 2000 et plus particulièrement à la partie que nous avons choisi d'implanter : le décodeur entropique. La troisième partie concerne la description de l'architecture utilisée, nommée ARDOISE. Enfin, la dernière partie présente les résultats obtenus par implantation du décodeur entropique en reconfiguration dynamique partielle sur l'architecture ARDOISE.

2 Critères d'analyse

2.1 Critères d'analyse des implantations statique et dynamique

L'analyse des implantations statique et dynamique va porter sur l'évaluation de deux critères [1], [2], l'un représente le coût de l'implantation et l'autre ses performances. Le coût sera désigné ici par le nombre de CLBs (Blocs Logiques Configurables) occupés par l'algorithme N_{CLBs_stat}. Le temps d'exécution T_{Exec_stat} nécessaire pour réaliser l'application permettra de définir le critère de performance (Perf.). De la même façon, nous définissons le coût dynamique et le critère de performance dynamique. A performances égales, l'apport de la reconfiguration dynamique sera donné par le rapport :

$$\frac{N_{CLB\_Stat} - N_{CLB\_Dyn}}{N_{CLB\_Stat}} \times 100\% \quad (1)$$

Afin d'évaluer le degré d'optimisation de l'architecture mise en oeuvre, nous utiliserons aussi le rendement spatio-temporel pour chacun des types d'implantation (statique et dynamique). Le rendement spatio-temporel est donné par la relation 2.

$$\rho_{Spat\_temp} = \sum_{i=1}^{M} (\rho_{Spat\_tache_i} . \rho_{Temp\_tache_i}) < 1 \quad (2)$$

avec

$$\rho_{Spat\_stat} = \frac{N_{CLBs\_actives}}{N_{CLBs\_dispo}} \quad (3)$$

et

$$\rho_{Temp\_stat} = \frac{T_{Exec\_stat}}{T_{Dispo}} \quad (4)$$

N_{CLBs_dispo} est le nombre de CLBs disponibles dans le FPGA et T_{Dispo}, le temps maximal dont on dispose pour effectuer le traitement (en imagerie standard, T_{Dispo} = T = 40 ms).

2.2 La reconfiguration dynamique dans le cas général

Dans certains cas, les traitements d'une application utilisant la configuration statique des FPGAs peuvent être exécutés en un temps très court et par conséquent, les FPGAs se mettent en attente de l'image suivante et restent inactifs. La figure 1 montre ainsi une application pouvant présenter ou non un pipeline et la durée de son exécution sur un FPGA classique en mode statique peut être inférieure à la durée d'acquisition de l'image : $M.N.T_{Iteration\_stat} < T$ (M étant le nombre de tâches à exécuter et N le nombre d'échantillons à traiter).

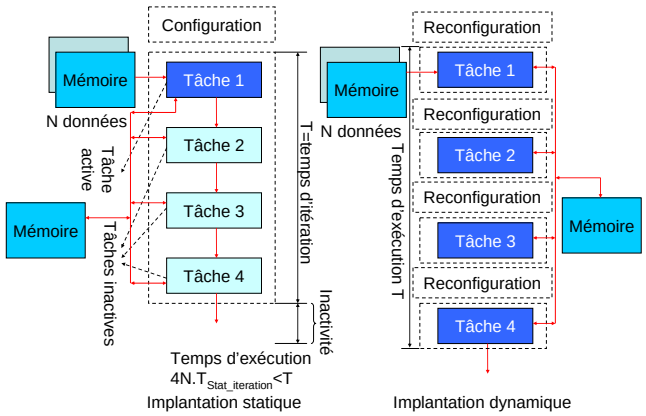

FIG. 1 – Implantation statique à rendement spatio-temporel inférieur à 1 et l'implantation dynamique correspondante

2.2.1 Discussion de l'implantation statique

Le coût et la performance sont respectivement donnés par les équations 5 et 6.

$$\begin{aligned} Cout_{Stat} &= N_{CLB\_stat} \\ &= \sum_{i=1}^{M} N_{CLB\_tache_i} + N_{CLB\_controle} \end{aligned} \quad (5)$$

$$\begin{aligned} Perf_{Stat} &= T_{Exec\_stat} \\ &= M.N.T_{Iteration\_stat} + T_{Inactivite} = T \end{aligned} \quad (6)$$

2.2.2 Discussion de l'implantation dynamique

$$\begin{aligned} Cout_{Dyn} &= N_{CLB\_dyn} \\ &= \max_{i=1}^{M}(N_{Tache_i\_CLB} + N_{CLB\_controle}) \end{aligned} \quad (7)$$

$$\begin{aligned} Perf_{Dyn} &= T_{Exec\_dynam} = T_{Exec\_stat} \\ &= M.N.T_{Iteration\_stat} + T_{Inactivite} \\ &= M.N.T_{Iteration\_dyn} + M.T_{Config} = T \end{aligned} \quad (8)$$

$$G_{CLB} = \frac{N_{CLB\_stat} - \max_{i=1}^{M}(N_{CLB\_tache_i})}{N_{CLB\_stat}} \times 100\% \quad (9)$$

La période d'itération dynamique nécessaire est telle que :

$$T_{Iteration\_dyn} = T_{Iteration\_stat} + \frac{T_{Inactivite}}{M.N} - \frac{T_{Config}}{N} \quad (10)$$

Le temps d'inactivité de l'implantation statique est exploité pour réduire la fréquence d'itération de l'implantation dynamique. Si la relation 11 est vérifiée, alors la fréquence d'itération dynamique peut même être plus faible que la fréquence d'itération statique.

$$\begin{aligned} &(T_{Inactivite})_{Stat} \geq M.T_{Config} \\ &\Rightarrow T_{Iteration\_dyn} \geq T_{Iteration\_stat} \end{aligned} \quad (11)$$

On parvient ainsi à obtenir une réduction de la surface et une diminution de la fréquence des traitements grâce à la reconfiguration dynamique.

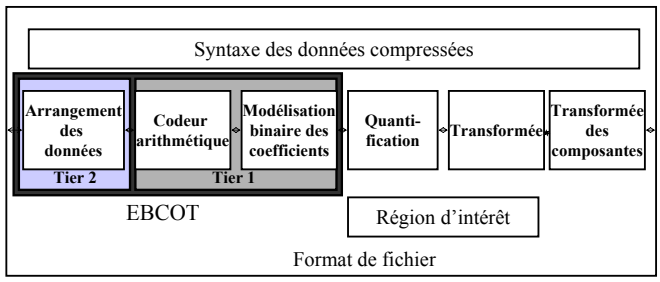

FIG. 2 – Les différents blocs de l'algorithme JPEG 2000

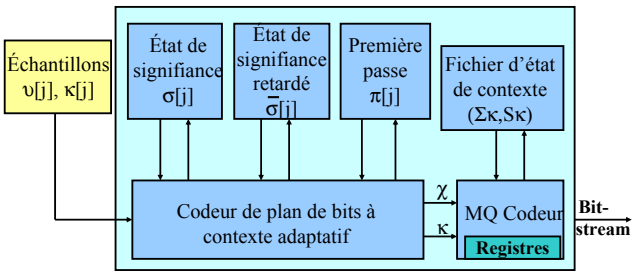

FIG. 3 – Schéma bloc de la partie Tier 1 de l'algorithme EBCOT

3 La norme JPEG 2000

La norme JPEG 2000 [3], [4] peut-être décomposée en plusieurs blocs successifs comme le montre la figure 2. Sur cette figure, les blocs sont disposés dans l'ordre « décodeur ». L'image d'origine est découpée en tuiles après la transformation des composantes. Toutes les tuiles subissent ensuite une transformation en ondelettes (transformation avec ou sans pertes), indépendamment les unes des autres. Tous les niveaux de résolution sont alors découpés en *code-block* de même taille (64×64, 32×32, ...). Les coefficients des *code-block* subissent une quantification et les coefficients quantifiés sont décomposés en plans de bits. En commençant par le plan de bits de poids fort, tous ces plans sont codés en fonction de leur « signifiance » et de leur contexte par trois passes successives. Les bits issus de ces différentes passes, ainsi que le contexte associé, sont ensuite envoyés à un codeur arithmétique (MQ-codeur identique à celui de JBIG2 [3]). Ces données codées sont ensuite mises en forme, en respectant la syntaxe définie dans la norme (*code-stream syntax*) pour former le *bit-stream* final. Ces trois derniers modules forment ce que l'on appelle EBCOT (*Embedded Block Coding with Optimisation Truncation*). L'EBCOT est lui-même composé de 2 parties distinctes : Tier 1 et Tier 2. Tier 1 regroupe la modélisation binaire des coefficients et le codeur arithmétique, et Tier 2 correspond à l'arrangement des données.

Nous allons nous intéresser plus particulièrement à la partie Tier 1 de l'EBCOT (Figure 3) et à la partie « décodeur ».

On trouve dans [3] les procédures correspondant à cette partie de l'algorithme (*Embedded Block Decoder*). Après initialisation, tous les éléments du *bit-stream* subissent les trois passes de décodage successivement : d'abord la passe de signifiance, puis la passe d'affinage et enfin la passe de nettoyage. Tous les plans de bits sont parcourus de la même façon. Les contextes, c'est-à-dire l'état de signifiance des 8 voisins du bit traité, sont initialisés puis modifiés au fil du décodage par la partie « décodeur de plan de bits à contexte adaptatif ». Il y a en tout 19 va-

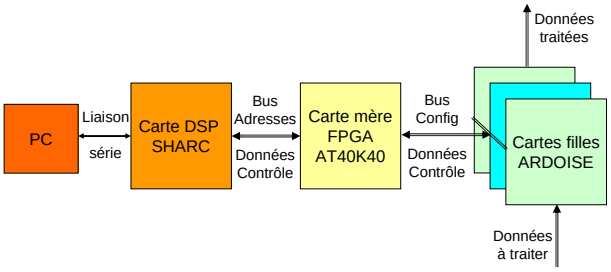

FIG. 4 – Les différents modules de l'architecture ARDOISE

leurs de contexte possibles : 9 pour la passe de signifiance (de 0 à 8), 5 pour le codage du signe (de 9 à 13), 3 pour la passe d'affinage (de 14 à 16) et 2 pour la passe de nettoyage (17 et 18). Dans toutes ces passes de décodage, une procédure est appelée régulièrement : la procédure MQ-Décode. Le codage arithmétique est basé sur la subdivision récursive de l'intervalle [0, 1) en fonction des symboles de la séquence d'entrée. La probabilité récursive de subdivision d'intervalle du codage Elias est la base de la procédure de codage arithmétique binaire. Quand le premier symbole de la séquence d'entrée est traité, l'intervalle est subdivisé en sous-intervalles associés aux symboles de l'alphabet. Chaque fois qu'un nouveau symbole est traité, le sous-intervalle associé est subdivisé de la même manière que l'intervalle initial. Cette procédure récursive cesse lorsque le dernier symbole de la séquence est traité.

4 L'architecture utilisée pour l'implantation : ARDOISE

Le projet ARDOISE a vu le jour au sein du GDR ISIS dans le but d'évaluer les performances d'une architecture dédiée au traitement des images en temps réel qui utiliserait la propriété de reconfiguration dynamique (totale ou partielle) des FPGAs [5]. Le FPGA choisit pour cette architecture est un FPGA ATMEL de la famille AT40K40. Cette famille de FPGAs a été choisie car elle permet des reconfigurations très précises (chaque CLB du FPGA est configurable indépendamment des autres contrairement à d'autres familles de FPGAs (par exemple XILINX) dont l'élément de configuration de base est la colonne de CLBs). Deux versions de l'architecture ARDOISE ont été réalisées. Actuellement, nous travaillons sur la seconde version. Nous disposons sur celle-ci d'une carte DSP SHARC Analog Devices, d'une carte mère à base d'un FPGA AT40K40 et de 3 modules carte fille (AT40K40 et 2 mémoires SRAM) (voir Figure 4).

La carte DSP sert d'interface entre la carte mère FPGA et le PC (via une liaison série). Elle est utilisée pour transmettre les fichiers de configuration des cartes filles et générer les signaux nécessaires. Le FPGA de la carte mère est chargé de gérer la configuration des différentes cartes filles. Sur la carte mère, on dispose d'une mémoire SRAM qui sert à stocker les fichiers de configuration reçus par le port série, ainsi qu'une mémoire Flash qui contiendra à terme le fichier de configuration du FPGA mère et les différentes configurations des cartes filles.

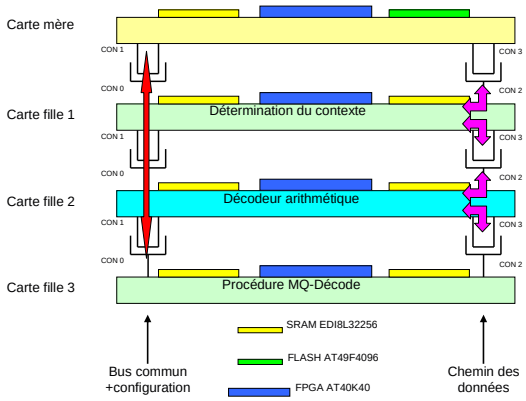

FIG. 5 – L'implantation des différentes parties du décodeur entropique sur l'architecture ARDOISE

5 Application de la reconfiguration dynamique

L'ensemble de l'algorithme JPEG 2000 nécessite beaucoup de ressources et de calculs. La proportion des besoins pour chacune des parties est de 70 % pour l'EBCOT, 20 % pour la transformation en ondelettes et les 10 % restants pour tout le reste des calculs [6]. La première partie de l'implantation concerne la procédure MQ-Décode [7]. Cette procédure est l'élément de base du décodeur entropique. Elle est utilisée une à plusieurs fois dans chaque passe de décodage. Cette partie de l'algorithme JPEG 2000 a été implantée sur une des 3 cartes filles de l'architecture ARDOISE.

La seconde partie de l'implantation concerne la détermination du contexte du symbole courant. En effet, la procédure de décodage utilise le voisinage du symbole à décoder. Cette partie de l'algorithme est elle aussi implantée sur une carte fille de l'architecture ARDOISE.

La dernière partie de l'algorithme du décodeur entropique de JPEG 2000 à implanter est la gestion des différentes passes de décodage qui interviennent dans l'algorithme EBCOT. Ces différentes passes de décodage sont également implantées sur une carte fille ARDOISE. L'implantation complète de ces 3 passes simultanément étant impossible sur un seul FPGA AT40K40, la reconfiguration dynamique de ce FPGA sera utilisée. Les différentes passes de décodage interviennent successivement pour chaque plan de bits : tout d'abord la passe de signifiance, puis la passe d'affinage et enfin, la passe de nettoyage, sauf pour le premier plan de bits signifiant qui est décodé uniquement par la passe de nettoyage. Ces différentes passes sont implantées les unes après les autres sur le même FPGA. Une autre configuration de gestion est alors nécessaire pour définir la configuration suivante à implanter. On utilise la mémoire interne du FPGA pour stocker les différentes informations nécessaires pour connaître le déroulement du décodage. Ces informations sont utilisées dans chacune des configurations, il faut donc que la reconfiguration du FPGA ne soit que partielle pour sauvegarder ces valeurs.

Au final, on utilise pour implanter le décodeur entropique de JPEG 2000 les 3 cartes filles de l'architecture ARDOISE. On peut voir sur la figure 5 la disposition des différentes parties implantées. Les FPGAs contenant les configurations du « MQ-Décodeur » et de la « détermination du contexte » sont confi-

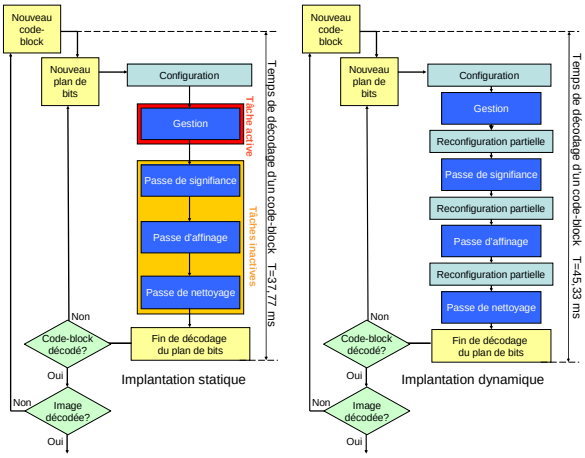

FIG. 6 – Représentation temporelle des fonctionnement en mode statique et en mode dynamique de l'implantation du décodeur entropique

gurées au début du décodage et le reste en permanence jusqu'à la fin. Le FPGA contenant les différentes passes de décodage est reconfiguré partiellement 4 fois pour chaque plan de bits à décoder.

La figure 6 représente la comparaison du fonctionnement des différentes configurations dans les cas statique et dynamique, au cours du temps.

La reconfiguration totale du FPGA ATMEL AT40K40 à la fréquence maximale de 33 MHz est effectuée en 337 μs. De plus, la fréquence de fonctionnement de ce FPGA est de 100 MHz [8]. La taille des *code-blocks* à décoder ici sera de 32×32 éléments. Pour chacune des trois passes de décodage, nous estimons le nombre maximal de cycles nécessaires pour traiter les 1 024 éléments. Si le traitement est effectué à la fréquence de 100 MHz, ce temps d'exécution est de 44,33 ms. Nous pouvons considérer que le temps total de reconfiguration du FPGA en reconfiguration dynamique pour les quatre reconfigurations partielles nécessaires est d'environ 1 ms. Nous obtenons donc pour la reconfiguration dynamique un temps total d'exécution de $Perf_{Dynam}$ = 45,33 ms pour chaque *code-block* 32×32. Dans le cas de la configuration statique, le temps d'exécution sera à peu de chose près identique à celui de la reconfiguration dynamique. En effet, nous nous trouvons ici dans le cas où il n'y a aucun parallélisme par les fonctions. Tous les éléments doivent être traités en respectant l'ordre de balayage défini dans la norme et le décodage de chaque élément est dépendant des éléments décodés précédemment. Le seul parallélisme que l'on peut avoir dans ce cas vient du fait que chaque *code-block* est indépendant des autres. Il est donc possible de traiter plusieurs *code-blocks* en même temps, à condition de disposer de la surface nécessaire pour implanter plusieurs fois le MQ-Décodeur dans le FPGA. Dans notre cas, la reconfiguration dynamique n'augmente pas le temps de traitement de manière significative par rapport au cas statique.

Nous pouvons calculer les coûts, performances et rendements obtenus dans chaque cas. Nous prenons comme temps de référence pour nos calculs, le temps T = 40 ms. Le tableau 1 récapitule ces résultats.

TAB. 1 – Comparaison des résultats obtenus pour les cas de configurations statique et dynamique sur ARDOISE

	Configuration statique	Configuration dynamique	Gain dynamique/statique
Coût (CLBs)	4 733	3 626	23,39 %
Performance	44,66 ms	45,33 ms	-1,5 %
ρ_{Spat}	0,5136	0,5246	2,14 %
ρ_{Temp}	1,12	1,13	0,9 %
ρ_{Spat_temp}	0,5752	0,5928	3,06 %

6 Conclusion

Nous avons montré ici l'implantation d'une partie de l'algorithme de décodage JPEG 2000 en utilisant la reconfiguration dynamique des FPGAs et comparé les coûts et les performances de cette méthode par rapport à l'implantation statique. Dans ce cas précis, la reconfiguration dynamique nous permet uniquement d'obtenir un gain en surface. Nous avons pour le moment étudié uniquement la reconfiguration dynamique des FPGAs de la famille ATMEL. Il serait également intéressant d'étudier la reconfiguration dynamique des FPGAs de la famille XILINX.

Références

[1] E. Bourennane, C. Milan, M. Paindavoine, S. Bouchoux, *Real Time Image Rotation using Dynamic Reconfiguration*, Real Time Imaging Journal 8, pp. 277 - 289, 2002.

[2] M. Wirthlin, B. Hutchings, *Improving Functional Density using Run-Time Circuit Reconfiguration*, IEEE Transactions on Very Large Scale Circuit Integration (VLSI) Systems, Vol. 6, No. 2, pp. 247 - 256, juin 1998.

[3] JPEG Group, *JPEG 2000 Image Coding System*, JPEG 2000 final commitee draft version 1.0, mars 2000.

[4] D. Taubman, M. Marcellin, *JPEG 2000 Image Compression Fundamentals and Practice*, Kluwer Academic Publishers, ISBN : 079237519X, novembre 2001.

[5] D. Demigny, M. Paindavoine, S. Weber, *Architecture Reconfigurable Dynamiquement pour le Traitement Temps Réel des Images*, Revue Techniques et Sciences de l'Information, Vol. 18, No. 10, pp. 1087 - 1112, 1999.

[6] D. Nicholson, P. Martinez, M. Iregi, J. Corral *Evaluation de la COmplexité Algorithmique de JPEG 2000*, Proceedings CORESA, 2000.

[7] J. Fagès, *JPEG 2000 - Principes, Implémentation et Evaluation*, Mémoire d'ingénieur CNAM, septembre 2000.

[8] ATMEL Corporation, *Data Sheet : 5K - 50K Gates Coprocessor FPGA with FreeRAM*.