Application de la reconfiguration dynamique des FPGAs : Décodeur arithmétique de JPEG2000

S. BOUCHOUX, E. BOURENNANE, J. MITERAN, M. PAINDAVOINE

Laboratoire LE2I, Université de Bourgogne, Aile des sciences de l'Ingénieur, B.P. 47870, 21078 Dijon Cedex
sbouchou@u-bourgogne.fr

Résumé – Cet article a pour sujet l'implantation d'une partie de l'algorithme du standard JPEG2000 (le décodeur arithmétique) dans un FPGA avec l'utilisation de la reconfiguration dynamique. L'implantation sera faite sur l'architecture ARDOISE.

Abstract *– This article is about implementation of a part of the algorithm of JPEG2000 (MQ-Decoder) in a FPGA with the use of dynamical reconfiguration. The implementation will be done on the ARDOISE architecture.*

Introduction

La nouvelle norme de compression JPEG2000, mise au point par le groupe JPEG dans le but de palier aux défauts de l'actuelle norme JPEG et de permettre l'obtention de meilleures performances (au niveau du taux de compression et de la qualité d'image) mais également d'avoir un plus grand nombre de fonctionnalités (régions d'intérêt, plusieurs types de décompression,…).Ce format a également été mis au point pour pouvoir s'adapter à différents types d'images : images médicales, texte, images naturelles,… Il doit également permettre d'encoder des images de grandes tailles. D'autres exemples d'applications où la compression doit être importante sans toutefois dégrader trop l'image reconstruite sont la photographie numérique, les images accessibles via Internet

Nous avons choisi d'implanter une partie de cette norme sur FPGA, et plus particulièrement sur la carte ARDOISE. Cette partie est développée en VHDL, après partitionnement de l'algorithme, de façon à pouvoir exploiter la reconfiguration dynamique des FPGAs sur la carte ARDOISE.

1. La norme JPEG2000

La norme JPEG2000 [1] peut-être décomposée en plusieurs blocs successifs comme le montre la Figure 1. Sur cette figure, les blocs sont disposés dans l'ordre « décodeur ».

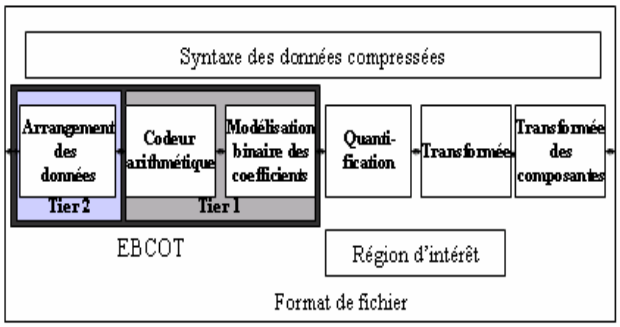

FIGURE 1 : Les différents blocs de JPEG2000.

L'image d'origine est découpée en tuiles après la transformation des composantes (voir figure 1). 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 (64x64, 32x32,…). Les coefficients de ces 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 (la passe de signification, la passe d'affinage et la passe de nettoyage). Les bits issus de ces différentes passes, ainsi que le contexte associé, sont ensuite envoyés à un codeur arithmétique (MQ coder identique à celui de JBIG2 [2]). 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 deux parties distinctes : Tier 1 et Tier 2 (voir Figure 2).

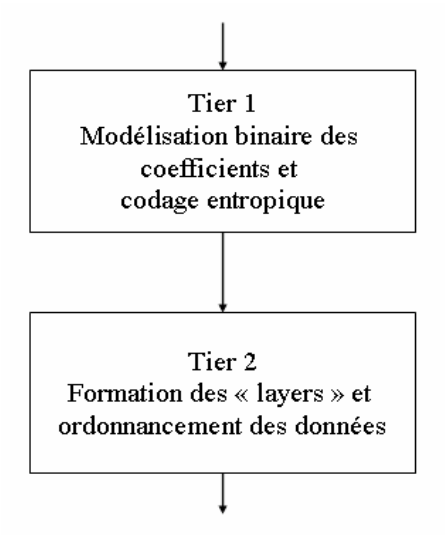

FIGURE 2 : Représentation des deux parties de l'EBCOT.

Tier 1 regroupe la modélisation binaire des coefficients et le codeur arithmétique, et Tier 2 correspond à l'arrangement des données. Ces données compressées forment alors un code-stream, respectant la syntaxe définie dans la norme. JPEG2000 permet plusieurs types de décompression à partir d'un même code-stream : progression par couche, par résolution,… Il est également possible de définir des régions d'intérêt où le taux de compression sera plus faible que pour le reste de l'image de façon à ne garder une bonne qualité d'image que dans cette partie. L'ordonnancement des données dans le code-stream permet une grande souplesse pour la reconstruction de l'image. En effet, à partir d'un même code-stream, il est possible d'obtenir plusieurs images reconstruites. On peut choisir de faire un décodage par région spatiale, par qualité, par résolution ou bien par composante. Il est également possible de faire subir à l'image reconstruite des transformations simples sans changer les données du code-stream, en particulier des rotations de 90° ou 180°.

2. Modélisation binaire des coefficients et décodeur arithmétique (Tier 1)

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

On trouve dans [2] les procédures correspondant à cette partie de l'algorithme (Embedded Block Decoder).

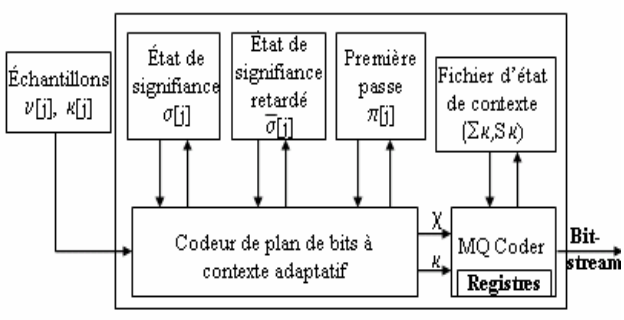

FIGURE 3 : Schéma bloc de la partie Tier 1 de l'EBCOT (codeur).

Après initialisation, tous les éléments du bit-stream subissent les trois passes de décodage successivement : d'abord la passe de signifiance, puis la passe de raffinement de magnitude et enfin la passe de nettoyage. Tous les plans de bits sont parcourus de la même façon (voir Figure 4).

FIGURE 4 : Ordre de balayage des plans de bits.

Les contextes, c'est-à-dire l'état de signifiance des 8 voisins du bit traité (voir Figure 5), 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 valeurs 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 de raffinage (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-Decode. Cette procédure est basée sur la probabilité des différents contextes obtenus dans les passes précédentes. Lorsque tout un code-block a été traité en codage, on trouve dans le code-stream des informations de début et de fin de bloc. Ceci permet un décodage en parallèle de plusieurs code-block.

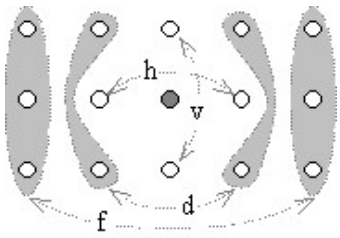

FIGURE 5 : Bits entrant en compte pour la détermination du contexte.

3. 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 [3]. Le FPGA choisit pour cette architecture est un FPGA ATMEL de la famille AT40K40 (équivalent 40 Kportes logiques). 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 carte 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 6).

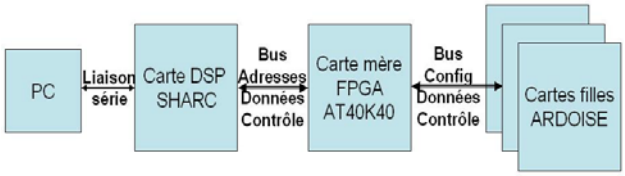

FIGURE 6 : Les différents modules de l'architecture ARDOISE.

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 programmé en mode série. Il 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.

Les données à traiter seront transférées en RAM par la liaison série et renvoyées sur le PC lorsque le traitement sera terminé ce qui permettra d'implanter successivement les différentes étapes du traitement et de les valider indépendamment les unes des autres.

4. Application de la reconfiguration dynamique

Cette partie de l'algorithme de JPEG2000 demande beaucoup de ressources au niveau du nombre d'opérations et des accès mémoire par bit traité [4]. Ce sont ces parties de l'algorithme JPEG2000 qui prennent le plus de temps au cours du traitement. En proportion, l'EBCOT nécessite 70% du temps de calcul total de l'algorithme de codage et la transformation en ondelettes 20%, les 10% restants correspondent à tous les autres traitements. Il est donc très important de bien maîtriser ces deux parties si l'on veut obtenir des résultats optimisés (en temps de traitement aussi bien qu'en efficacité).

La première partie de l'implantation concerne la procédure MQ-Decode [5]. En effet, cette procédure est l'élément de base de l'EBCOT. Elle est utilisée dans les différentes passes du décodage. Cette procédure comporte 3 phases distinctes : une première phase d'initialisation (utilisée une seule fois au début du décodage), la phase MQ-Decode en elle-même et, si nécessaire, une phase de renormalisation des différents registres. Cette partie de l'algorithme JPEG2000 est extrêmement « gourmande » au niveau des ressources utilisées car elle est composée d'une succession de tests et demande la manipulation d'un grand nombre de registres et de constantes. En effet, il faut avoir à disposition à tout moment les différentes valeurs relatives aux contextes. Il est donc judicieux de stocker ces valeurs à l'intérieur du FPGA, en RAM interne, et non en RAM externe, car les accès répétés aux mémoires externes sont coûteux aussi bien en temps qu'en ressources logiques et en consommation. Cette partie sera donc en permanence dans le FPGA. Elle restera fixe pendant que le reste du FPGA sera reconfiguré. L'implantation sur la carte ARDOISE de cette partie de l'algorithme nécessite 2 configurations. La première configuration correspond à l'initialisation du MQ-Decodeur, elle n'intervient qu'une fois au début des passes de décodage. La seconde correspond au décodeur en lui-même ainsi qu'à la renormalisation des registres. On conservera durant tout le traitement les valeurs des contextes. On recommence ensuite le décodage d'un nouveau symbole et ce jusqu'à ce que l'on ai décodé tout le « paquet » (voir Figure 7).

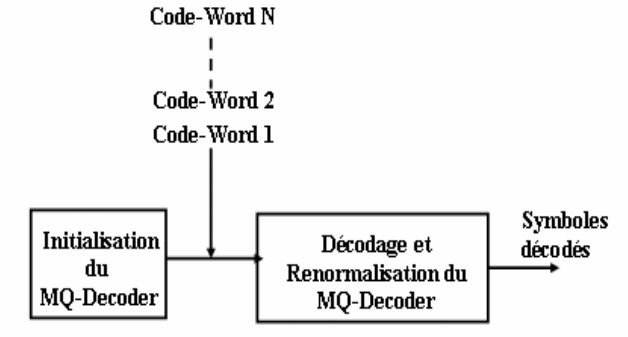

FIGURE 7 : Les différentes phases de configuration correspondant au MQ-Decoder.

Le nombre de symboles qu'il faut décoder est variable suivant l'image d'origine, la qualité recherchée et la compression réalisée. Chaque symbole dépend de ses 8 plus proches voisins, il faut donc garder en mémoire (interne ou externe suivant les cas) tous ces symboles ainsi que les contextes correspondants.

Les différents blocs de l'image compressée étant indépendants les uns des autres, on peut envisager de traiter le décodage de plusieurs symboles, appartenant chacun à des blocs différents, en mémorisant les informations les concernant (contextes, valeur du symbole, …) en mémoire interne avant de reconfigurer partiellement le FPGA pour finir le traitement.

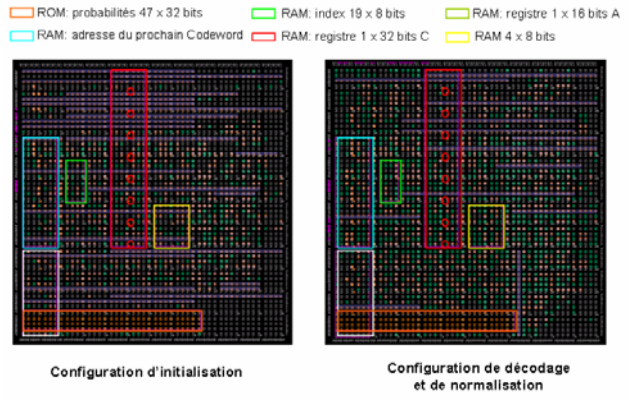

FIGURE 8 : Les deux configurations implantées dans le FPGA AT40K40.

Le MQ-Decoder a été implanté en reconfiguration dynamique (2 configurations) sur l'architecture ARDOISE (1362 CLBs utilisés sur 2304 (59%) et 21 blocs RAM utilisés sur 144 (15%) pour la première configuration et 1447 CLBs utilisés sur 2304 (63%) et 21 blocs RAM utilisés sur 144 (15%) pour la seconde configuration). Le reste de la partie Tier 1 de l'EBCOT, c'est-à-dire la modélisation binaire des coefficients (les trois passes de décodage successives), devra être implantée en réutilisant le MQ-Decodeur. Les paquets de données correspondant à chacun des code-block sont décodés les uns après les autres (sans interférences entre les différents paquets). Les données de ces paquets sont décodées dans une seule des trois passes successives et pour chaque passe, les données sont toutes examinées et toujours dans le même ordre. Les contextes sont initialisés au début du décodage de chaque code-block et évoluent ensuite en fonction des données décodées au fil des passes. Il faudra envisager de conserver l'implantation du MQ-Decodeur durant toute la partie décodage Tier 1 et utiliser la place restante dans le FPGA pour implanter en reconfiguration dynamique partielle les différentes passes de décodage les unes après les autres.

5. Conclusion et perspectives

A titre de comparaison au niveau de la surface utilisée dans le FPGA pour notre implantation, nous avons également implanté entièrement la partie MQ-Decoder dans un FPGA Xilinx Virtex II XC2V80 (équivalent à 80 Kportes logiques, soit environ le double du FPGA ATMEL AT40K40 (40 Kportes environ)). La surface occupée par ce traitement est de 491 slices (ou 982 CLBs Xilinx VirtexII) sur un total de 512, soit 95% de la surface du FPGA. Il serait toutefois intéressant d'étudier les possibilités de reconfiguration dynamique qu'offrent les FPGAs Xilinx, même si ceux-ci ne permettent pas un niveau de reconfiguration aussi précis que les FPGas ATMEL.

Références

[1] *JPEG2000 Image coding system* (JPEG2000 final commitee draft version 1.0, 16 mars 2000).

[2] David S. Taubman, Michael W. Marcellin. *JPEG2000 Image compression fundamentals, standards and practice* (Kluwer Academic Publishers; ISBN: 079237519X; Bk&Cd-Rom edition (November 1, 2001)).

[3] D. Demigny, M. Paindavoine, S. Weber : *Architecture Reconfigurable Dynamiquement pour le Traitement Temps Réel des Images*. Revue technique et Sciences de l'information, Numéro Spécial programmation des Architectures Reconfigurables. (1998).

[4] Didier Nicholson, Patrice Martinez, Marcela Iregui, Javier Corral. *Evaluation de la complexité algorithmique de JPEG2000* (Proceedings CORESA 2000).

[5] Jean-Marc Fages *JPEG2000 – Principes, implémentation et évaluation* (Mémoire d'ingénieur CNAM, 12 septembre 2000).