Accélération et Optimisation de l'Implémentation Matérielle de l'Encodeur H.264/AVC par le Parallélisme des Données

MESSAOUDI KAMEL[1,2], BOURENNANE EL-BAY[1], TOUMI SALAH[2], TOUIZA MAAMAR [1]

[1] Laboratoire d'Electronique, Informatique et Image (LE2I)
Université de Bourgogne, BP 47 870, 21078 Dijon Cedex, France
[2] Laboratoire d'Etude et de Recherche en Instrumentation et en Communication d'Annaba (LERICA)
Université Badji Mokhtar, BP 12, 23000 Annaba, Algérie

[1] kamel.messaoudi@u-bourgogne.fr, ebourenn@u-bourgogne.fr, maamar.touiza@u-bourgogne.fr
[2] kamel.messaoud@univ-annaba.org, salah.toumi@univ-annaba.org,

Résumé - Dans ce papier, nous présentons une nouvelle solution d'implémentation matérielle de la norme H264/AVC basée sur le parallélisme des données. Cette nouvelle approche permet la réduction des ressources utilisées dans le FPGA, les besoins en mémoires internes ainsi qu'une réduction du temps de traitement d'un macrobloc de l'image. Une comparaison avec les propositions existantes est fournie à la fin de cet article et montre bien les avantages de notre implémentation.

Abstract – In this paper, we present a new hardware implementation solution of the H264/AVC based on data parallelism. Our approach allows the reduction of the occupied area in the FPGA, the required internal memory as well as the reduction of the execution time. A comparison with existing approaches is provided at the end of this work. This comparison demonstrates the advantages of our implementation.

1 Introduction

Pour répondre aux besoins du temps réel, d'un grand nombre d'applications multimédias et de traitement vidéo, des solutions d'implémentations matérielles sur des plateformes reconfigurables ont été proposées [1]. La plupart des solutions existantes sont basées sur des modules de traitement avec des entrées/sorties de 32 bits [2].

Comme norme de compression vidéo, H.264/AVC [3] est spécialement conçue pour remédier à plusieurs faiblesses des normes précédentes. Elle est basée sur un codage hybride fondé sur les blocs et sur une amélioration des concepts et outils existants, notamment pour les prédictions Intra-image et Inter-image. L'amélioration des performances est significative sur le taux de bits avec la même qualité visuelle, cependant la complexité algorithmique est bien plus élevée. En effet, par comparaison, l'encodeur H.264 est dix fois plus complexe que l'encodeur MPEG-2 pour un gain de 50% en taille de fichier [4].

Dans le présent article, nous allons montrer comment exploiter au mieux la largeur des données (128 bits) à la sortie des mémoires DDR2. Nous allons également montrer les avantages apportées par l'utilisation des bus de données de 128 bits comparée à des solutions basées sur des bus de 32 bits. En effet, dans l'encodeur H.264, les unités de base à traiter sont des blocs de 4×4 pixels, ce qui les rend plus appropriées à un traitement sur 128 bits.

Dans la suite de ce travail, nous introduisons la norme de compression vidéo H.264/AVC (section 2). Nous décrivons les deux méthodes de gestion de mémoire pour l'encodeur dans la section 3. Une implémentation matérielle, basée sur des données de 128 bits des modules de l'encodeur H.264 est données dans la section 4. Nous terminons ce travail par une conclusion.

2 L'encodeur H.264/AVC

Le standard H.264/AVC a été introduit en 2003, par l'UIT-T et l'ISO/IEC, afin de répondre aux besoins de la vidéo. Le document « Recommandation H.264 : Advanced Video Coding » [3] ne définit pas un codec à proprement parler, mais plutôt une syntaxe pour le flux vidéo compressé et une méthode pour décoder cette syntaxe et produire une séquence vidéo affichable. Ce standard permet le codage de vidéo au format chroma 4:2:0, avec des images progressives ou entrelacées, éventuellement combinées dans une même séquence. La couche VCL est définie pour représenter de manière efficace le contenu des données vidéo. Le concept de la couche de codage vidéo H.264/AVC est semblable à celui des autres normes telles que MPEG-2. Il s'agit d'un hybride de prédiction temporelle et de prédiction spatiale, allié à un codage par transformée. Le standard H.264/AVC est un codeur en boucle fermée [5]. Le codeur H.264 comprend le décodeur (en boucle) afin de réaliser la prédiction pour les blocs suivants ou bien l'image suivante (Figure 1). Les coefficients quantifiés subissent un rééchelonnement et une transformée inverse similaire à ceux du décodeur, aboutissant au reliquat de prédiction décodé. Ce reliquat est ajouté ensuite à la prédiction [2]. Le résultat de cette opération est transmis au filtre anti-blocs permettant d'éliminer certaines dégradations produites par le module de

quantification qui engendre des pertes d'informations [6]. L'utilisation d'algorithmes complexes au niveau de l'encodeur implique une charge plus importante des calculs [4].

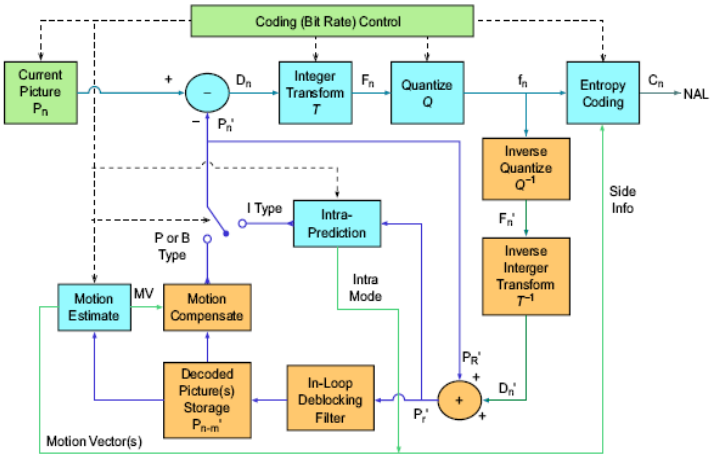

Figure 1 : Diagramme de l'encodeur H264 [7]

3 Besoins en mémoire de données de la norme H264/AVC

Pour une bonne gestion de mémoire, plusieurs paramètres doivent être pris en compte : le type et la capacité de mémoire, le temps lié à la latence, la cadence, …etc. Aujourd'hui, sur le même circuit FPGA (ou bien sur la même carte de prototypage), nous disposons de plusieurs types de mémoires. Sur une même carte, on trouve des mémoires externes de type DDR2/DDR3, des mémoires flash ou bien des mémoires de type SRAM, mais aussi des mémoires embarquées internes au FPGA (on chip RAM ou bien BRAM). Dans les récents circuits FPGA de la famille virtex5 de Xilinx par exemple, les mémoires BRAM sont reconfigurables, et offrent la possibilité de définir la taille des mots (8, 16, 32 et 64bits), la direction des données et le type de la mémoire, …etc. Dans ce cas, les mémoires sont données en Kbits et il appartient à l'utilisateur de les configurer selon les besoins de l'application à implémenter. Vue la limitation en mémoire interne des circuits FPGA, les implantations matérielles des algorithmes de la H264 exploitent plutôt les mémoires externes. Dans notre cas, nous avons utilisé la mémoire externe de type DDR2-SDRAM disponible sur la carte Xilinx ML501. Nous avons exploité deux solutions pour l'implantation de l'encodeur H.264 avec des entrées et sorties de 128 bits.

Dans la première solution, les modules de l'encodeur sont utilisés dans une architecture à base de processeur. Afin d'adapter nos accélérateurs matériels (128bits) au reste du système à travers le bus PLB, nous avons utilisé des mémoires tampon type FIFO. Ces FIFO ont des entrées sur 32 bits et des sorties sur 128bits. Elles servent à collecter les données avant leurs transferts vers les modules de traitement [8]. L'objectif ici est de faire une économie des mémoires tampons en intégrant des traitements parallèles des modules de traitement. La figure 2 montre un exemple d'adaptation des modules de l'encodeur H.264 avec le bus PLB.

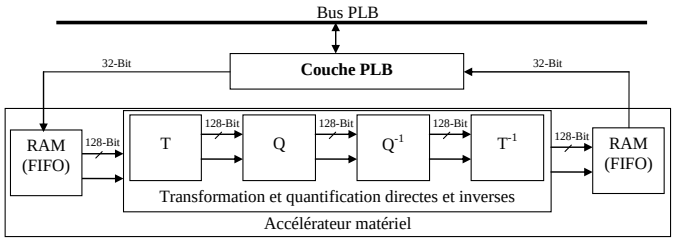

Figure 2 : Adaptation des modules de l'encodeur H.264 avec le bus PLB

Dans la deuxième solution, les modules de l'encodeur (accélérateurs matériels) sont reliés directement avec la mémoire externe à travers un contrôleur mémoire spécialement conçu pour les besoins de la norme H264. Dans [9], nous avons proposé un contrôleur mémoire qui joue le rôle de contrôleur de caches de données pour les différents modules de traitement. L'idée, dans ce cas, est d'exploiter au mieux la largeur des données (128 bits) à la sortie des mémoires DDR2 en évitant le bus de 32 bits. Cet organe d'adressage est constitué d'un contrôleur mémoire DDR2 et d'une partie intelligente chargée des calculs d'adresses en prédiction des besoins de chacun des modules de la H264. Ceci permet aussi de décharger l'encodeur de la partie adressage. La figure 3 montre l'architecture proposée de ce contrôleur.

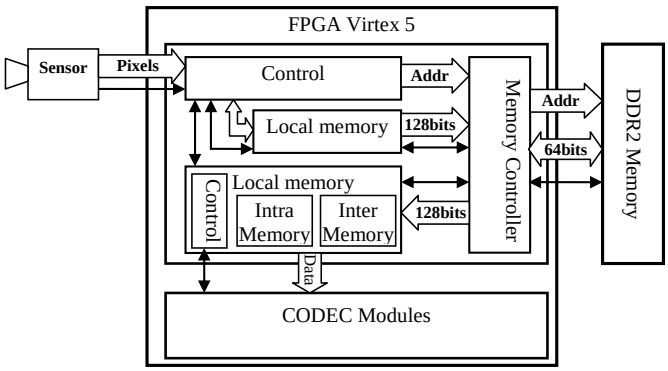

Figure 3 : Le contrôleur mémoire proposé pour l'encodeur H.264

4 Implémentations matérielles de l'encodeur H.264/AVC

La dépendance inter-bloc dans les modules de l'encodeur H264 nous empêche d'envisager plus de parallélisme sans surcoût matériel significatif. Pour l'implémentation matérielle des modules de traitement de l'encodeur H.264, nous proposons des architectures basées sur les principes de parallélisme de données et de pipeline. Nous avons utilisé des entrées/sorties sur 128 bits (bloc de 4×4 pixels) et nous avons comparé les performances obtenues avec celles basées sur des architectures à 32 bits [1].

Le module de prédiction Intra calcule, par comparaison des coûts, un seul mode pour le macrobloc 16×16pixels ou bien 16 modes pour les blocs 4×4pixels. Ensuite, les blocs résiduels (128 bits) sont calculés et transférés vers les modules de transformation et de quantification directe. Les sorties de ces deux modules sont envoyées vers l'encodage entropique ainsi qu'à la chaine de prédiction inverse. Cette dernière est

composée d'un module de quantification inverse, d'un module de transformation inverse et d'un module de reconstruction pour produire les blocs décompressés. Ces blocs reconstruits et non filtrés sont utilisés comme blocs de voisinage pour des futures prédictions Intra [10]. Après le filtrage, les blocs reconstruits sont utilisés pour la prédiction Inter.

Pour ce qui est de l'implantation de la chaîne de prédiction Intra, nous proposons des architectures matérielles basées sur des entrées/sorties de 128 bits. Cette chaîne est composée des modules suivants : Intra-prédiction, transformation et quantification (TQ), quantification et transformation inverses (IQIT) ainsi que le module de décision et de contrôle (Figure 4). Tous ces modules ont été séparés par des étages de pipeline. Tout cela facilité également la gestion de la mémoire pour les différents modules et la simplification des équations de traitement dans chaque module. Nous obtenons ainsi une augmentation du débit de données pour chaque bloc et chaque macrobloc sans l'augmentation des ressources utilisées dans le FPGA. Le tableau 1 résume les résultats de la synthèse de ces architectures en utilisant la plateforme Virtex5-ML501 et donne aussi une comparaison avec les résultats des implémentations, proposées dans [1], basées sur des entrées/sorties de 32 bits. On devrait s'attendre à ce que le surcoût en ressources matérielles soit multiplié par un facteur de 4 lors du passage d'une implantation 32 bits à celle de 128 bits (4×32). Avec notre approche, les ressources consommées n'ont augmenté que de 25% pour les LUT et de 100% pour les slices de registres alors que les BRAM n'ont pas changé. Pour cette modeste augmentation en ressources utilisées, nous avons amélioré le temps de traitement d'un macrobloc de 43% (687 au lieu de 1200 cycles d'horloge).

Pour le filtre anti-blocs, nous proposons une nouvelle implémentation matérielle avec une méthode de gestion de la mémoire selon l'ordre de filtrage donné par Chen et al [6]. L'implémentation du filtre est réalisée selon une stratégie à base de 5 filtres directionnels, avec des données de 32 et de 128 bits. L'utilisation des filtres élémentaires permet d'éviter les circuits de transpositions. Les résultats de synthèse sont mentionnés sur le tableau 2.

Nous pouvons déduire des deux tableaux que notre proposition a les avantages suivants : réduction de l'usage des mémoires tampons dans les modules ; élimination des circuits de transposition dans le filtre ; possibilité de traitement en parallèle sans l'augmentation de ressources matérielles ; possibilité de traitement en pipeline, ce qui augmente la fréquence des traitements. Ici, la fréquence est limitée par l'unité de contrôle de l'ensemble de la chaîne de traitement (module Mode decision sur la figure 4).

En passant d'une implantation de 32 bits à 128 bits, cela ne cause qu'une légère augmentation des ressources en faveur d'une réduction très significative du nombre de cycles nécessaires au traitement d'un macrobloc. Dans l'exemple du filtre, le nombre de cycles d'horloge nécessaires au traitement d'un macrobloc est ramené à 55 cycles au lieu de 115 cycles, ce qui représente un gain de plus que 100%.

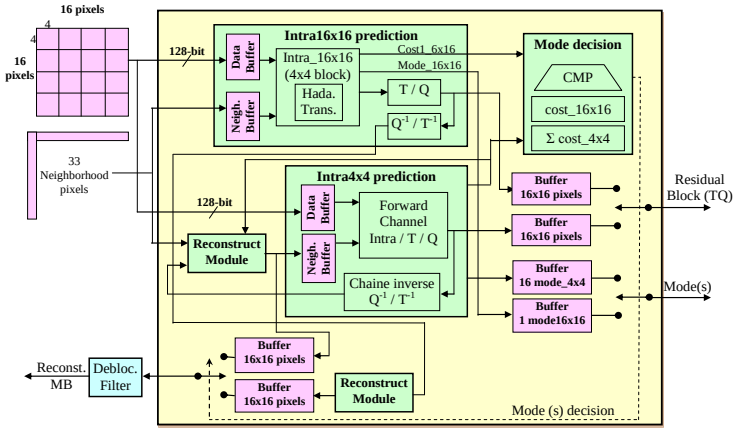

Figure 4 : Implémentation matérielle du module Intra

Tab 1 : Résultants de synthèse des différents modules élémentaires de la chaîne de prédiction Intra

	Ressources / Module	silice registres	LUTs	BRAM /ROM	DSP48E	Fréquence max (MHz)	Nom. Cycles d'horloge	Possibilité de pipeline
Nos implémentations (128 bits)	Intra4×4	1070	1584	2	/	157.921	15/bloc	Oui
	Intra16×16	1603	1672	/	/	233.100	150/MB	Oui
	Transfo. Nombre entier	1150	776	/	/	457.247	5/bloc, 20/MB	Oui
	Quantification	355	943	ROM 64×42bit	16	366.569	4/bloc, 19/MB	Oui
	Quantification inverse	451	739	ROM 64x15bit	16	366.569	4/bloc, 19/MB	Oui
	Transformation inverse	1,187	912	/	/	457.247	5/bloc, 20/MB	Oui
	Total de la chaîne intra	5,075	5,403	4 RAM, 2 ROM	32	155.618	687	
Le travail dans [1] (32 bits)	Intra4×4	450	601	2	/	238,903	/	Non
	Intra16×16	Non programmé						
	Transfo. Nombre entier	101	105	/	/	443,321	/	Non
	Quantification	35	130	ROM 64×42bit	1	308,471	/	Non
	Quantification inverse	26	93	ROM 64x15bit	1	371,913	/	Non
	Transformation inverse	776	715	/	/	304,507	/	Non
	Total de la chaîne intra	2536	4376	4 RAM, 2 ROM	2	167,653	1200	Non

Tab 2 : Résultats de synthèse du module de filtrage (deblocking filter)

Ressources / Module	silice registres	LUTs	BRAM /ROM	DSP48E	Fréquence max (MHz)	Nom. Cycles d'horloge	Possibilité de pipeline
Filtre proposé (128 bits)	6416	7701	5	/	184,417	55-71/MB	Oui
Filtre proposé (32 bits)	2846	5404	3	/	189,139	115/MB	Non

5 Conclusion

Dans cet article, nous avons montré l'utilité de la parallélisation par les données dans le cas particulier de la H264. En effet, dans l'encodeur H.264, nous avons souvent besoin de traitements de blocs (4×4) d'où l'intérêt de travailler sur des tailles de données de 128 bits au lieu du 32 bits habituellement. Cette proposition nous a permis d'avoir des gains en ressources utilisées, en mémoire nécessaire et une augmentation de la fréquence des traitements ainsi qu'une réduction du temps d'exécution (nombre de cycles d'horloge nécessaire pour le traitement d'un macrobloc) par l'exploitation du pipeline (soit un gain de 43% pour le module Intra et plus que 100% pour le module du filtre anti-blocs). Tous ces résultats ont été confirmés par la synthèse et par l'implantation sur la carte ML501 de Xilinx. En perspective à ce travail, nous expérimentons actuellement la reconfiguration dynamique sur les différents modules.

Références

[1] Site: http://hardh264.sourceforge.net/H264-encoder-manual.html.

[2] C. W. Ku, C. C. Cheng, G. S. Yu, M. C. Tsai et T. S. Chang, "A High-Definition H.264/AVC Intra-Frame Codec IP for Digital Video and Still Camera Applications", IEEE Trans. on Circuits and Systems for Video Tech.16 (8) (2006) 917-928.

[3] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, "Draft of Version 4 of H.264/AVC (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding)", 2003.

[4] T. Wiegand, G. J. Sullivan, G. Bjøntegaard et A. Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. On Circuits and Systems for Video Technology 13(7), pp.560-576, 2003.

[5] Ralf Schäfer, Thomas Wiegand et Heiko Schwarz, "H.264/AVC la norme qui monte", UER–RevueTechnique, (2003) pp 1-10.

[6] Z. CHEN, W. UAO, U. WANG, M. ZHANG et W. ZHENG, "A performance optimized architecture of deblocking filter for H .264/AVC", The Journal of China Universities of Posts and Telecommunications 14 (2007) 83-88.

[7] Iain E. G. Richardson, "H.264 and MPEG-4 Video Compression - Video Coding for Next-generation Multimedia", The Robert Gordon University, Aberdeen, UK, Edition WILEY, 2003.

[8] K. messaoudi, M. Touiza, E.B. Bourennane et S. Toumi, "Hardware/Software Co-Design with MicroBlaze Soft-Core Processor for the Integer Transform Algorithm Used in the H.264 Encoder", International Review on Computers and Software (I.Re.Co.S), Vol. 5. n. 3, pp. 348-354, May 2010.

[9] K. messaoudi, E.B. Bourennane, S. Toumi, E. kerkouche et O. Labbani, "Memory Requirements and Simulation Platform for the Implementation of the H.264 Encoder Modules", IEEE International Conference on Image Processing Theory, Tools and Applications IPTA'10, pp133-137, Juillet 2010.

[10] C. W. Ku, C. C. Cheng, G. S. Yu, M. C. Tsai, T. S. Chang, "A High-Definition H.264/AVC Intra-Frame Codec IP for Digital Video and Still Camera Applications", IEEE Trans. on Circuits and Systems for Video Tech.16 (8) (2006) 917-928.