Contraintes de réalisation d'un démonstrateur mixte FPGA- ADSP 21160 sur une application de Télécommunication : récepteur DVB-RCS

M. TORRES[*], A. VERGONJANNE, V. MEGHDADI, J.P. CANCES et R. SALVETAT[*]

GESTE-ENSIL, Université de Limoges, Parc ESTER, B.P. 6804, 87068 Limoges

anne@ensil.unilim.fr

[*]ARBOS Ingénierie 32, rue du Général Dejean 11400 Castelnaudary

matthieu.torres@arbos-dsp.com

Résumé – Le développement d'algorithmes de télécommunications débute par une phase de simulation informatique permettant de prévoir les performances des routines mises en œuvre. Cependant cette simulation n'est pas suffisante pour la validation des algorithmes qui devront nécessairement être implantés en temps réel sur un démonstrateur hardware. Nous avons expérimenté la migration de la phase de simulation vers un démonstrateur mixte FPGA-DSP au travers d'une application dans le domaine de la réception satellite (récepteur DVB-RCS) pour dégager les problèmes rencontrés lors du passage sur démonstrateur hardware. Nous avons mis l'accent sur les problèmes de répartition hardware, de flux de données, d'optimisation et de test. Ces de ces problèmes liés à l'architecture n'apparaissent que sur une maquette proche du système final. L'utilisation d'un rack multiprocesseur généraliste permettrait des cycles de développement plus court, mais dans notre cas aurait masqué les contraintes de flux de données et d'optimisation.

Abstract *– This article deals with the implementation phase of an embedded system. Specific problems connected to mixed architectures have been evaluated in a telecommunication application case. We have studied a QPSK demodulator (DVB-RCS) implemented on a platform combining a floating-point DSP and a FPGA. We have pointed out constraints related to hardware partitioning, data streams, code optimization and system testing. Some of these architecture problems appear only on models close to the final system. The use of a general purpose multiprocessor rack, allows shorter development stage, but in our case it would have masked data stream and code optimization problems.*

1. Introduction

Le développement d'un système embarqué se compose généralement d'une phase de simulation et d'une phase de prototypage. La simulation permet de développer l'algorithme de l'application à implanter et de valider son principe de fonctionnement. Le prototypage consiste à faire migrer cet algorithme sur un démonstrateur afin de vérifier sa faisabilité technique et d'évaluer ses performances. C'est dans cette phase de prototypage que peuvent se poser diverses contraintes, liées à la précision des calculs, au temps-réel, au test du système… L'implantation d'un algorithme temps-réel peut s'effectuer en utilisant un système surdimensionné permettant de contourner certains problèmes comme la vitesse d'exécution ou les ressources mémoire. Une autre approche consiste à travailler sur une maquette plus proche du système final pour ne pas masquer les problèmes d'architecture.

Dans le secteur des télécommunications, les systèmes s'orientent vers l'utilisation conjointe de FPGAs et de processeurs. Afin d'évaluer les problèmes spécifiques à ce type d'architecture, nous nous sommes intéressés à la phase d'implantation d'un algorithme de télécommunication sur une plateforme combinant un DSP à virgule flottante et un FPGA (SHARC HammerHead d'Analog Devices et SPARTAN II de Xilinx). L'application test est un démodulateur QPSK (norme DVB-RCS) pour lequel nous cherchons à estimer le débit maximal pouvant être obtenu avec l'architecture retenue.

2. Description de l'application

La modulation utilisée est définie par la norme DVB-RCS (Digital Video Broadcast-Return Channel Satellite) :

- modulation QPSK
- accès multiple de type MF-TDMA
- paquets avec un préambule de 48 symboles
- taille des paquets de 16, 56 et 188 octets
- erreur fréquentielle maximum à compenser : 10^{-2}Ts
  (Ts: Temps symbole)

L'objet de notre étude est la réalisation du récepteur numérique. Nous ne nous intéressons pas à la partie

analogique de détection du signal. L'entrée du démonstrateur accepte les voies I et Q en bande de base.

Avant de pouvoir effectuer la démodulation proprement dite, différentes opérations doivent être exécutées dans le récepteur (figure 1) :

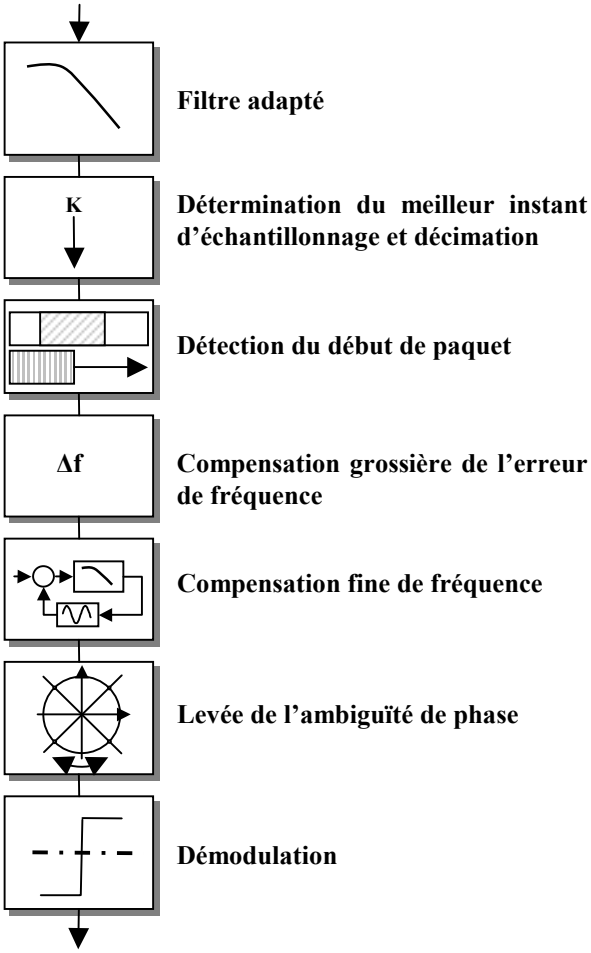

FIG. 1 : le récepteur numérique

- Le premier étage dans un récepteur numérique est le filtre adapté. Ce filtre, identique à celui de l'émetteur, est un FIR en racine de cosinus surélevé.

- Le signal d'entrée étant suréchantillonné d'un facteur K, un algorithme recherche le meilleur échantillon parmi les K et effectue la décimation.

- Une séquence d'apprentissage connue du récepteur est émise comme préambule au paquet de données. Le récepteur calcule la corrélation entre la séquence et le signal reçu décimé sur une fenêtre glissante et en déduit le début du paquet.

- Le signal reçu est entaché d'une erreur de fréquence due aux différences entre les oscillateurs locaux de l'émetteur et du récepteur et au mouvement relatif éventuel entre émetteur et récepteur. L'erreur fréquentielle est compensée en deux temps. La première compensation grossière réalisée en utilisant l'approche de Marco-Luise [1] permet d'attaquer un étage basé sur une boucle à verrouillage de phase (PLL).

Le temps d'accrochage de la PLL pouvant gêner la démodulation des premiers symboles, Sollenberger [2] propose d'utiliser une PLL supplémentaire qui parcourt le paquet du milieu vers le début. Ainsi, en initialisant cette seconde PLL avec l'état de la première au milieu du paquet, cette seconde PLL est accrochée d'office. Elle parcourt le paquet jusqu'au début et corrige la phase du signal pour les tous premiers symboles. Nous avons ainsi une estimation de phase cohérente tout au long du paquet. Ce principe est illustré sur le schéma ci dessous (figure 2).

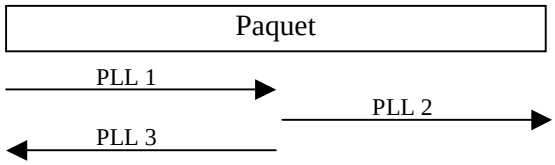

FIG. 2 : la méthode de Sollenberger

- L'ambiguïté de phase inhérente à la modulation MPSK est levée en utilisant l'intercorrélation avec le préambule.

- Le dernier étage détermine le quadrant dans lequel chaque échantillon complexe se place pour effectuer la démodulation QPSK.

Les algorithmes correspondant aux différents blocs du récepteur ont été mis au point en langage C dans la phase de simulation informatique. La figure 3 présente les résultats de simulation comparés aux résultats théoriques. Le TEB est calculé en fonction du rapport signal à bruit, pour différentes tailles de paquets. On remarque que pour les paquets de longueurs 188 octets, l'écart avec les courbes théoriques est négligeable. Par contre pour les paquets courts, l'écart est plus important. Ceci se justifie par le fait que les boucles à verrouillage de phase ont du mal à s'accrocher quand les paquets ne sont pas suffisamment longs.

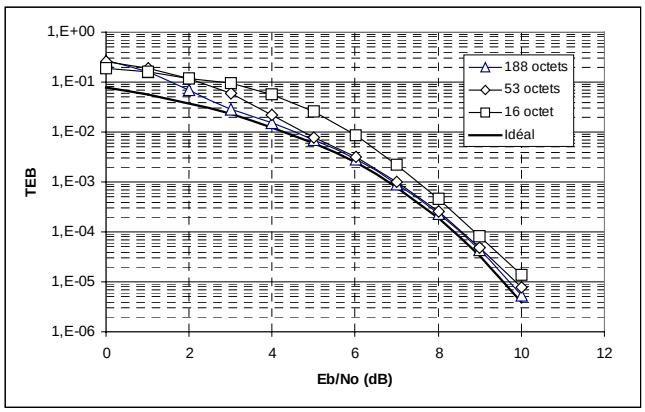

FIG. 3 : résultats de la simulation

3. Migration sur la cible hardware

3.1 Architecture

Nous avons retenu des circuits de génération récente : le FPGA XC2S50 (famille SPARTAN II de Xilinx) et le processeur ADSP-21160 (famille SHARC HammerHead d'Analog Devices). L'architecture SIMD du DSP est bien adaptée au traitement des nombres complexes, et son contrôleur DMA non intrusif permet d'échanger des données sans interrompre les calculs du cœur.

La répartition des tâches entre les deux circuits est généralement basée sur les critères suivants [3] : opérations de calculs intensif et régulier sur le FPGA, algorithmes à branchements multiples et forte précision (virgule flottante) sur le processeur. C'est pourquoi on rencontre souvent les FPGA en début de chaîne de réception. Nous avons choisi d'implanter le filtre en racine de cosinus surélevé dans le FPGA. Nous avons pu constater que la réalisation du filtrage par le DSP consommerait environ 50% des ressources du processeur. La figure 4 présente un diagramme blocs du démonstrateur réalisé.

3.2 Mise en œuvre du FPGA

Depuis quelques années, on a accès à des blocs IP (Intellectual Property) qui permettent d'implanter des fonctions dans un FPGA sans connaître les détails de l'architecture du composant. Le filtre à été implanté en utilisant les blocs IP fournis par Xilinx. (Logicore : DA FIR Filter 5.0). Les contraintes de dimensionnement que nous avons eu à prendre en compte sont la précision du calcul, et le degré de parallélisme du filtre. Le ratio entre l'horloge de cadencement du filtre et l'horloge d'échantillonnage détermine le degré de parallélisme de l'implémentation, dicté par le compromis entre la taille occupée sur le silicium et la fréquence maximum d'utilisation. Le ratio de 3:1 que nous avons sélectionné nous permet d'économiser de la place dans le FPGA tout en conservant de la marge sur la vitesse d'exécution.

Transfert des données entre le FPGA et le DSP

Nous recherchons le débit maximum que peut traiter notre système, ce qui amène une forte contrainte au niveau du flux de données entre le DSP et le FPGA. Les données filtrées sont passées au DSP au travers d'un canal DMA. Le contrôleur DMA du SHARC est non-intrusif, c'est à dire qu'il nous permet d'assurer l'échange des données indépendamment du travail du cœur, sans ralentir les calculs. Cette caractéristique est primordiale pour manipuler des flux de données à haut débit.

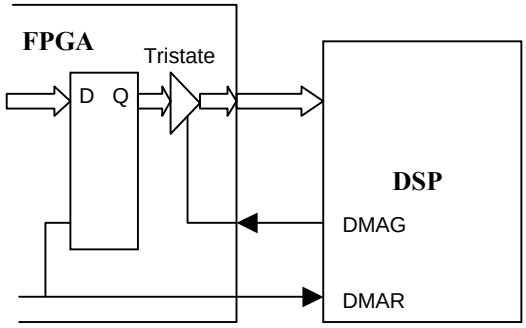

FIG. 5 : La boite à lettres DMA

Nous avons mis en place un mécanisme de " boite à lettres " dans la FPGA, de manière à interfacer le contrôleur DMA du SHARC en mode " HANDSHAKE " (figure 5). Ce mode utilise un signal de requête DMA généré par le périphérique, appelé DMAR, plus un signal d'acquittement produit par le contrôleur DMA, DMAG.

DMAR est généré par le FPGA à partir de l'horloge d'échantillonnage. Ce signal contrôle le verrouillage de l'échantillon en provenance du filtre, de manière à le tenir prêt en attendant l'acquittement DMA. DMAG devient actif dès que le contrôleur DMA du SHARC reprend le contrôle du bus. Ce signal commande alors le positionnement de l'échantillon sur le bus (grâce à des drivers à trois états implantés dans le FPGA).

Ce processus permet de charger 64 bits en un cycle bus, répartis automatiquement sur deux mots de 32 bits consécutifs dans la mémoire interne du processeur. Nous allons ainsi profiter de l'architecture SIMD du DSP qui va pouvoir travailler simultanément sur les voies I et Q.

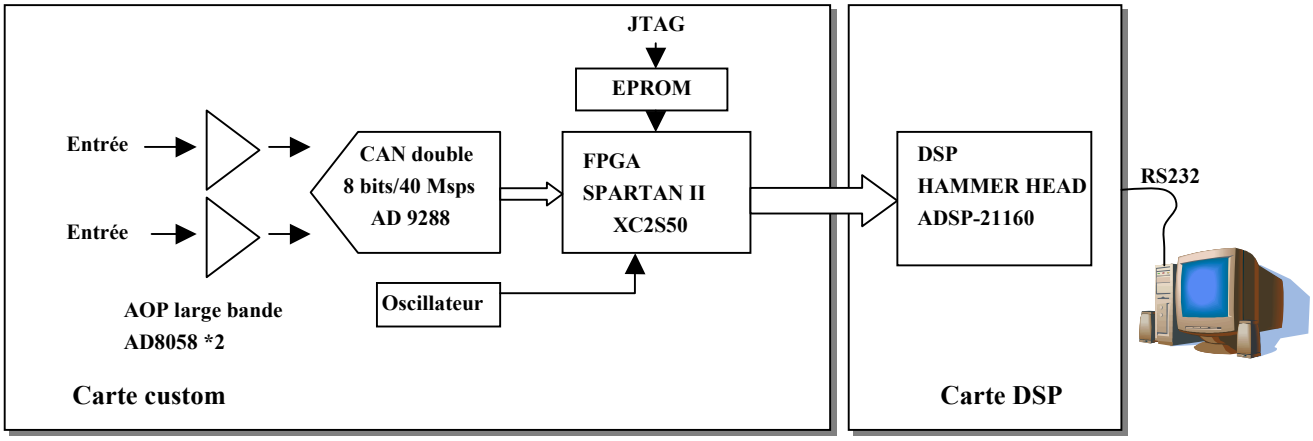

FIG.4 : schéma blocs du système

3.3 Mise en œuvre du DSP

L'implantation directe sur le DSP des fonctions C que nous avions écrites pour la simulation a montré une limite importante au niveau de la vitesse d'exécution. Les routines on été réécrites en assembleur et optimisées, afin d'atteindre la pleine puissance du processeur. Diverses techniques ont permis d'optimiser le code :

- L'architecture SIMD consiste à dupliquer les unités de calcul d'un processeur. Dans notre cas, chaque instruction peut s'appliquer à deux données simultanément, ce qui offre une possibilité de traitement accélérée des nombres complexes.

- L'architecture Harvard modifiée (deux mémoires, deux bus et un cache d'instructions) couplée au SIMD multiplie les opérations simultanées. Cette instruction tirée de la boucle de corrélation illustre l'efficacité du coeur :

```
f8=f0*f4, f12=f8+f12, f0=dm(i3,m2), f4=pm(i8,m10); // (mode SIMD activé)
```

Nous réalisons ici deux multiplications, deux additions et quatre accès mémoire 32 bits à virgule flottante en un seul cycle d'horloge.

- L'utilisation de tables a permis d'accélérer l'évaluation des fonctions trigonométriques. Il a fallu prêter attention à ne pas dégrader la précision de l'arctangente utilisée dans le calcul de la phase d'un échantillon complexe. Si on stocke les valeurs de Arctg dans une table à pas constant, on obtient une très bonne précision en fin de table et une mauvaise précision au début. Nous utilisons donc une échelle logarithmique pour étaler au mieux cette précision.

Ces calculs optimisés sont imbriqués au sein d'un programme C qui se charge des opérations n'ayant pas de contraintes de rapidité, comme l'initialisation du système ou l'envoi des résultats à un PC.

3.4 Évaluation des performances

La méthode de test qui nous a paru la plus rapide à mettre en place consiste à remplir la RAM de la carte DSP avec les paquets démodulés, pour ensuite les rapatrier sur un PC. Nous avons choisi la liaison RS232 pour sa simplicité de mise en œuvre aussi bien du coté DSP (émulation logicielle par des I/O) que sur le PC. Nous avons programmé des applications graphiques nous permettant de reprogrammer les cartes, générer des données de test, récupérer les paquets résultats et calculer le TEB (Taux d'Erreur Binaire).

Cette manipulation nous à permis de déterminer le débit symboles maximal de notre système, en conservant un TEB cohérent avec nos simulations : 410 Ksymboles/s.

Le calcul du TEB a été effectué par comparaison des paquets émis et des paquets démodulés. Nous avons donc réservé les premiers octets (après le préambule) pour numéroter les paquets. Le problème est qu'une erreur au niveau de l'octet d'identification fausse complètement le TEB en provocant la comparaison de deux paquets différents. Il nous a donc fallu rejeter les paquets suspectés d'être mal identifiés suivant des critères stricts. Nous avons choisi de doubler l'octet d'identification, et de n'accepter que les paquets pour lesquels le numéro est cohérent avec les précédents. Sachant que nous incrémentons les numéros à l'émission, nous n'acceptons que les paquets ayant des numéros consécutifs.

4. Conclusion

Nous avons présenté la phase de migration d'une application de télécommunications sur une plate-forme mixte DSP-FPGA en s'intéressant aux différentes contraintes à prendre en compte lors de cette étape.

L'utilisation d'un système de type rack de processeurs aurait masqué les problèmes de gestion du flux de données et d'optimisation du code. Une approche plus équilibrée peut être envisagée par l'emploi de cartes de prototypage qui évite de masquer les contraintes d'architecture. La principale limitation de cette méthode réside dans la difficulté des outils de développement à prendre en compte les architectures mixtes et multiprocesseurs.

Références

[1] Marco Luise and Ruggero Reggiannini, "Carrier Frequency Recovery in All-Digital Modems for Burst-Mode Transmissions," IEEE trans. on Commun., vol 43, no. 2/3/4, pp 1169-1178, February/March/April 1995.

[2] Nelson Sollenberger and Justin C.I. Chuang, "Low-Overhead Symbol Timing and Carrier Recovery for TDMA Portable Radio Systems," IEEE trans. on Commun., vol 38, no. 10, pp 1886-1892, October 1990.

[3] M. Guillaud, A. Burg, L. Mailaender, B. Haller, M. Rupp, E. Beck, "From Basic concepts to Real-Time Implementation: Prototyping WCDMA Downlink Receiver Algorithms – A case study", Proc. 34th Asilomar Conf. on Signals, Systems, and Computers, Oct. 29 – Nov. 1, 2000.