Architecture temps réel d'analyse temps/fréquence large bande

Sébastien TREGARO[1], Emmanuel BOUTILLON[2], Christian ROLAND[2]

[1]Rubisoft, 83-87 avenue d'Italie, 75013 Paris, FRANCE

[2]UBS, Lab-STICC CNRS UMR 3192
Centre de recherche, BP 92116, 56321 Lorient Cedex, FRANCE
Université Européenne de Bretagne, France
sebastien.tregaro@univ-ubs.fr, emmanuel.boutillon@univ-ubs.fr
christian.roland@univ-ubs.fr

Résumé – Pour une application d'analyse de signaux radar, nous avons implanté au sein d'un FPGA, un banc de filtres polyphasés permettant une décomposition temps/fréquence du signal reçu. Pour tenir les contraintes de temps réel, 4 filtres polyphasés fonctionnant en parallèle ont été implantés. L'ordre de sortie des échantillons du convertisseur analogique numérique étant différents de l'ordre de consommation des données dans les bancs de filtres, un entrelaceur a dû être mis au point. Dans ce papier, nous présentons la méthode suivie pour concevoir cet entrelaceur avec pour contrainte une complexité matérielle minimale. La méthode proposée utilise une représentation en 4 dimensions du problème d'entrelacement, puis la projection de cette solution sur une structure de mémoire parallèle.

Abstract – In the context of a radar identification application, we have implemented a polyphase filter bank in a FPGA to generate a time/frequency image of the received signal. In order to cope up with the real time constraints, 4 polyphase filters have been implemented. Since the order of arrival of the 4 input samples differs from the the order required by the polyphase filters, an interleaver has been developped. In this paper, we present a method to build the interleaver with a minimal memory size. The proposed method is based first on a solution in a 4-dimension space, then the projection of this solution on a parallel memory.

1 Introduction

Dans le contexte d'une application temps réel d'analyse de signaux radar, nous avons implanté sur un FPGA, un banc de filtres polyphasés permettant une décomposition en temps et en fréquence d'un signal de bande utile de 500 MHz avec une résolution temporelle de 96 ns et une résolution fréquentielle de 10.1 MHz.

Schématiquement, le signal RadioFréquence est en premier lieu transposé à une fréquence intermédiaire de 1 GHz avant d'être échantillonné sur 8 bits à une fréquence F_e de 1,33 GHz. La première étape (non décrite dans cet article) consiste à ramener le signal en bande de base et à créer en parallèle les voies en phase et en quadrature (voie I et Q). Le passage d'un signal réel à un signal complexe permet de diviser par deux la fréquence des échantillons qui passe ainsi à $F_s = F_e/2$ soit 666,67 MHz. L'étape suivante consiste à effectuer une analyse temps/fréquence en temps réel. Le principe est le suivant : il s'agit de diviser les 666.67 MHz de bande du signal en 64 sous-bandes par un banc de filtres puis de mesurer l'énergie dans chaque sous-bande. Comme la largeur de bande du signal en sortie de chaque filtre est de 666,67/64 soit 10,1 MHz, il est possible d'effectuer une décimation d'un facteur 64 avec un faible repliement de spectre. La résolution temporelle pour chacun des filtres est de $64/F_s = 96$ ns. La qualité du filtre dépend de sa taille, nous avons choisi un filtre de 128 coefficients comme étant un bon compromis entre performance et complexité. La technique du filtre polyphase permet, par l'utilisation d'une TFR, de réduire significativement la complexité de l'opération de filtrage [1].

L'implantation matérielle de cette chaîne d'analyse temps-fréquence est relativement complexe. En effet, le FPGA ciblé fonctionne à une fréquence d'horloge F_{clk} 4 fois inférieure à la fréquence F_s, soit $f_{clk} = 166.7$ MHz. Pour tenir compte des contraintes du temps réel, il faut donc dupliquer par 4 l'architecture du banc de filtres polyphasés, chacun étant capable de traiter un bloc de 128 données en 64 cycles d'horloge. Il se pose alors un problème d'entrelacement des données. En effet, l'ordre d'arrivée des données dans le FPGA (4 données consécutives) étant différent de l'ordre de traitement par les bancs de filtres polyphasés, un entrelaceur est nécessaire.

Dans ce papier, nous nous focalisons sur le problème de l'entrelaceur. Nous présentons une démarche originale permettant d'aboutir à un entrelaceur de taille mémoire minimale. La démarche proposée consiste à trouver une solution dans un espace mémoire non contraint (de dimension quatre), puis de projeter la solution trouvée sur un espace mémoire réel, constitué de bancs mémoire (soit un espace à deux dimensions, une pour le numéro du banc et l'autre pour l'adresse dans le banc mémoire).

L'article est divisé en 4 parties. Tout d'abord, nous rappelons le principe du filtre polyphasé et le cahier des charges

de l'entrelaceur par rapport aux contraintes de notre application. Ensuite, nous présentons un état de l'art sur les techniques d'entrelaceur. Enfin, en constatant que l'état de l'art ne propose pas de solution satisfaisante à notre problème, nous présentons une nouvelle démarche originale de conception d'entrelaceur. Finalement, la conclusion résume les résultats et présente les perspectives.

2 Fonctionnement du filtre polyphasé

Le banc de filtres polyphasés permet d'effectuer le traitement suivant sur un bloc constitué des 128 derniers échantillons complexes :

$$X_q(n) = \sum_{k=0}^{127} h(k) e^{-2\pi j \frac{kq}{64}} \times u(n-k) \quad (1)$$

Où $q = 0..63$ est l'indice de la sous-bande, $h(k)$ est la réponse impulsionnelle du filtre prototype réel (passe bas de largeur de bande 1/64) et les $\{u(n-k)\}_{k=0..127}$ représentent le bloc des derniers 128 échantillons reçus à l'instant n. Il est possible de factoriser le calcul des 64 sous bandes en deux étapes. La première étape correspond au calcul, pour $k = 0..63$ de :

$$u_f(k) = h(k)u(n-k) + h(k+64)u(n-(k+64)) \quad (2)$$

La seconde consiste à calculer la transformée de Fourier du vecteur $U_f = \{u_f(k)\}_{k=0..63}$ afin d'obtenir $X : X = TFR(U_f)$. Il s'agit d'un filtre polyphasé classique.

Pour fonctionner en temps réel, à chaque cycle d'horloge 2 données complexes doivent alimenter chacun des 4 bancs de filtres polyphasés mis en parallèle. La trame est ainsi découpée en paquets de 256 données complexes consécutives (4 fois 64 données) $u(256p)...u(256p+255)$, où p est l'indice du paquet. Au cycle d'horloge $256p + k$ (k variant de 0 à 63) les données $u(256p - k)$ et $u(256p - (64 + k))$ sont utilisées par le premier filtre pour le calcul du $k$ième élément du vecteur d'entrée $U(k)$ de la première TFR. De façon identique, à ce même instant, les couples $\{u(256p - (64 + k)), u(256p - (128 + k))\}$, $\{u(256p - (128 + k)), u(256p - (192 + k))\}$ et $\{(u(256p - (192 + k)), x(256p - (256 + k))\}$ sont respectivement utilisés pour le calcul du k^{ieme} élément de la deuxième, de la troisième et de la quatrième TFR (voir figure 1).

De ce séquencement, deux remarques s'imposent. Premièrement, deux filtres polyphases consécutifs partagent une entrée en commun. Il suffit donc de fournir 5 données aux filtres polyphases. Le dernier filtre polyphasé nécessite les informations $u(256p - (256 + k))$, soit encore $u(256(p-1) - k)$. Il suffira donc de mémoriser dans une *FIFO* les 64 derniers échantillons du paquet précédent pour les obtenir dans l'ordre voulu. En résumé, à chaque cycle d'horloge, en mettant à part le cas du dernier échantillon qui vient d'être traité, 4 données sont lues dans les mémoires avant l'arrivée de 4 nouvelles données.

3 État de l'art de l'entrelacement

Il existe une littérature très riche sur les entrelaceurs et leurs réalisations matérielles, particulièrement, dans le domaine des codes correcteurs d'erreur. Brièvement, on peut considérer 3 classes d'entrelaceurs. Les premiers sont les entrelaceurs "continus" qui permettent de permuter l'ordre temporel d'une donnée. Ces entrelaceurs ont été étudiés dans les années 90, principalement pour le cas des turbo-codes convolutifs mais sans aborder l'aspect parallélisme qui nous concerne. L'autre cas est celui des entrelaceurs à deux dimensions, toujours pour les turbo-codes en bloc nécessitant un parallélisme de décodage. Il faut alors trouver une structure mémoire permettant la lecture sans conflit des données entre l'ordre naturel et l'ordre entrelacé. Un état de l'art de ces méthodes est disponible dans [3]. Il existe aussi une abondante littérature sur le placement des données dans des bancs de mémoire et la définition d'un réseau de permutations pour permettre la lecture sans conflit des lignes, colonnes, diagonale etc. de tableaux.

Fort de cet état de l'art, il eut été aisé de construire une solution basée sur deux structures mémoires multiplexées, chaque structure contenant 4 bancs mémoires. Pendant que la première structure mémorise 4 données entrantes par cycle d'horloge, la seconde fournit 4 données dans le bon ordre aux filtres polyphasés. Une fois le bloc traité, le rôle des deux structures est échangé. Toutefois, nous avons été contraints de ne pas retenir cette solution car elle multiplie par 2 la quantité de mémoire par rapport à la taille mémoire minimale.

4 Mémoire entrelacement 4D

Notre objectif est de trouver une solution de taille mémoire minimale (soit un bloc de 256 données) pour effectuer l'opération d'entrelacement. Pour respecter cette contrainte, il faut que les emplacements des 4 données lues du bloc courant soient immédiatement remplacées par les 4 nouvelles données du bloc suivant. Nous proposons dans cette section une solution théorique basée sur une mémoire fictive de dimension 4. Nous montrons ensuite comment projeter cette solution sur une structure mémoire réelle. Enfin, nous présentons les résultats d'implantation.

4.1 Solution théorique à 4 dimensions

Nous allons d'abord décrire une solution théorique utilisant un hypercube de dimension 4 et de largeur de coté 4 (soit $4^4 = 256$ cases mémoires). Pour alléger les notations, l'échantillon d'index $(256p - k)$ sera simplement représenté par l'index k dans la suite de l'article. Les données sont modélisées sous la forme d'un hypercube à 4 dimensions nommées (k_3, k_2, k_1, k_0), avec $k_i \in \{0, 1, 2, 3\}, i = 0..3$. Sur la figure 2, ces quatre dimensions correspondent à la décomposition de la donnée k en base 4.

À l'instant initial, les données courantes du paquet p de 256 valeurs sont mémorisées dans l'ordre naturel. Autrement dit, la

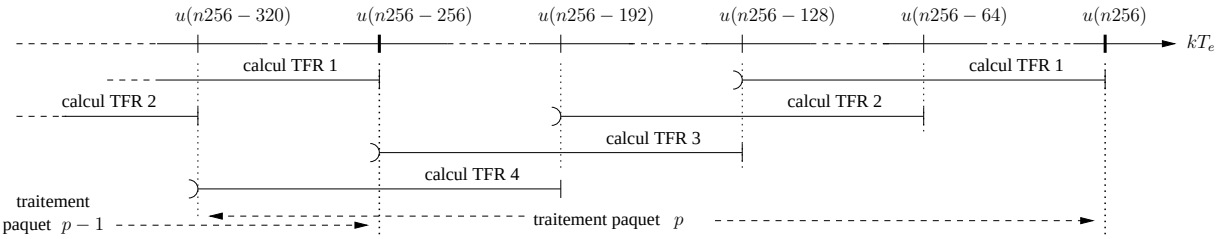


FIG. 1 – Séquencement temporel des TFR et des FIR

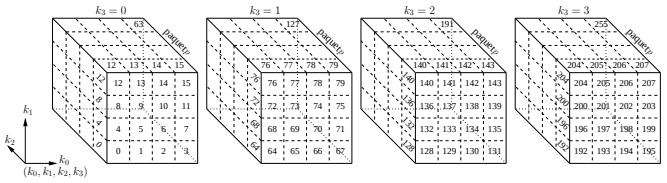


FIG. 2 – Placement dans l'hypercube des données à l'instant initial

donnée k se trouve à l'adresse (k_3, k_2, k_1, k_0) de la mémoire 4D, avec (k_3, k_2, k_1, k_0) vérifiant : $k = (k_3, k_2, k_1, k_0)_4$, ou encore, $k = 4^3k_3 + 4^2k_2 + 4^1k_1 + 4^0k_0$.

Pendant le traitement des 256 données du bloc p, le premier cycle d'horloge, le système lit les données en commençant par les adresses 0, 64, 128 et 192. Au coup d'horloge suivant, les adresses 1, 65, 129 et 193 sont lues et ainsi de suite. Il s'agit donc d'une lecture des données en parallèle selon la dimension k_3. Durant ce cycle de traitement du premier paquet, les 4 premiers coefficients du paquet $(p+1)$ suivant arrivent : ils se placent naturellement dans les cases mémoires venant d'être lues. Ainsi, durant le traitement de ce premier paquet, les données sont lues en parallèle selon la dimension k_3, et en série selon les dimensions k_0, k_1 et k_2. À la fin de l'itération, le nouveau paquet entrant se trouve donc en mémoire selon l'ordre (k_0, k_3, k_2, k_1) comme indiqué sur la figure 3. Nous avons donc une rotation du repère en 4D pour le nouveau paquet de données entrant.

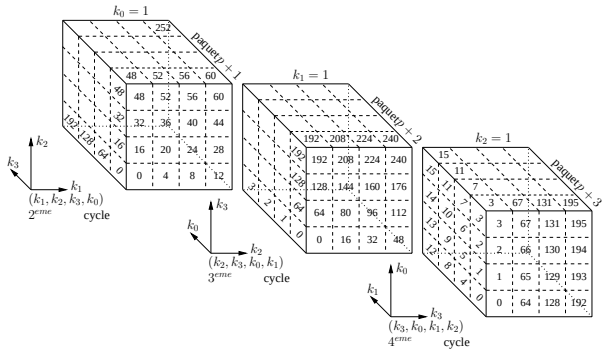


FIG. 3 – Placement dans l'hypercube des données pour les trois paquets suivants

Pour le traitement du paquet $(p+1)$, les données seront lues en parallèle selon la dimension k_2 pour tenir compte du nouvel agencement des données. Il en sera de même pour l'écriture des données. Ainsi, à la fin du traitement de ce paquet, les données du paquet suivant $(p+2)$ seront placées dans l'ordre (k_1, k_0, k_3, k_2). À l'issue du traitement du paquet $(p+3)$, quatre rotations de base auront eu lieu et les données sont à nouveau dans l'ordre naturel comme à l'instant initial. La périodicité des dimensions de lecture et d'écriture est donc de 4. En supposant l'existence d'une mémoire de dimension 4, la solution proposée permet un mécanisme d'accès en mémoire simple et de périodicité égale à $4 \times 64 = 256$ cycles d'horloge.

Il reste maintenant, à partir de cette solution basée sur une mémoire fictive de dimension 4, de trouver une solution pour un banc mémoire réel.

4.2 Projection sur les bancs mémoires

L'architecture de l'entrelaceur, inspirée de [2], est constituée par un réseau de rotations en entrée pour permuter les échantillons avant écriture dans les bancs mémoires, des bancs mémoires double ports, chacun ayant un générateur d'adresse dédié et enfin, d'un réseau de rotation en sortie. La figure 4 donne le schéma synoptique de l'entrelaceur.

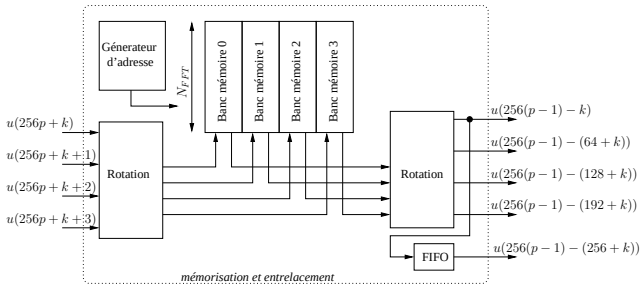
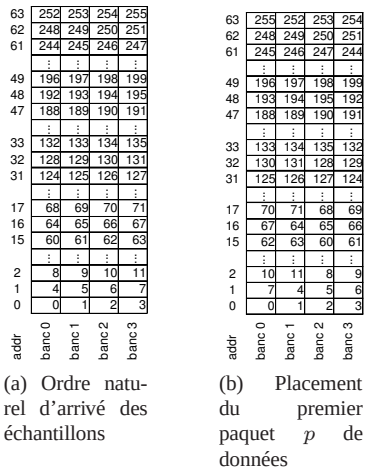


FIG. 4 – Schéma synoptique de l'entrelaceur

Les échantillons entrants arrivent dans l'ordre naturel selon la représentation de la figure 5(a). Nous avons ensuite défini un placement initial des données permettant de lire sans conflit les données dans chacune des 4 dimensions. Ce placement se déduit automatiquement de notre modélisation à l'aide d'hypercubes. Il est représenté en 4 dimensions par la figure 2, les données sont transposées à seulement deux dimensions selon la figure 5(b).

Le placement des données k du premier paquet p se fait comme suit : soit $k = (k_3, k_2, k_1, k_0)_4$, alors la donnée k sera placée dans le banc mémoire $(k_0 + k_1 + k_2 + k_3)\%4$ à l'adresse $(k_3, k_2, k_1)_4$, avec $k\%4$ représentant la valeur de k modulo 4.



(a) Ordre naturel d'arrivé des échantillons

(b) Placement du premier paquet p de données

FIG. 5 – Ordre naturel d'arrivée des échantillons et premier placement en mémoire

Ainsi les données k = 0, 1, 2, 3 se trouvent respectivement dans les bancs 0, 1, 2 et 3 à l'adresse 0. Les données $k = 0, 4, 8, 12$ se trouvent respectivement dans les bancs 0, 1, 2 et 3 aux adresses 0, 1, 2 et 3. Les données $k = 0, 16, 32, 48$ se trouvent respectivement dans les bancs 0, 1, 2 et 3 aux adresses 0, 4, 8 et 12 et enfin, les données $k = 0, 64, 128, 192$ se trouvent dans les bancs 0, 1, 2 et 3 aux adresses 0, 16, 32 et 48.

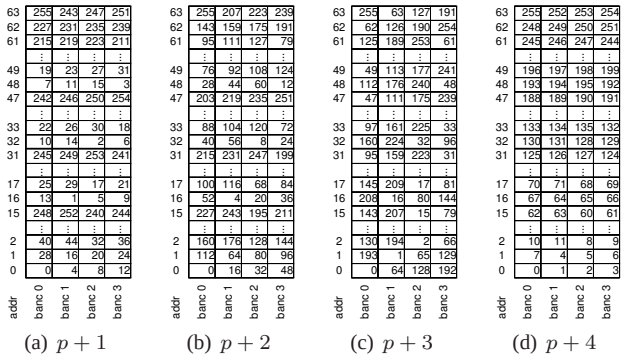
Lors du traitement du premier bloc, les données peuvent ainsi être lues en parallèle dans la dimension 3 (k_3) sans conflit. Il en sera de même pour le deuxième bloc et les suivants.

Le numéro du banc mémoire où doit se placer une donnée k est toujours donné par la somme des dimensions k modulo 4 : $n_{banc} = (k_0 + k_1 + k_2 + k_3)\%4$. Par contre l'adresse mémoire d'une donnée est fonction du paquet considéré. Le contrôle des permutations et de l'adresse d'écriture dans les bancs mémoires découle naturellement du placement mémoire et du séquencement 4D. Au cycle $k = (k_3, k_2, k_1)_4$ du traitement du bloc p, l'entrelaceur d'entrée effectue une rotation à droite des entrées de $\pi(k) = 4 - (k_3 + k_2 + k_1)\%4$ positions et l'entrelaceur de sortie une rotation des sorties de $\pi^{-1} = (k_3 + k_2 + k_1)\%4$ positions. Le banc mémoire i sera adressé à l'adresse $a(p, k, i)$, avec $i = 0..3$ l'indice du banc, $p = 0..3$, l'indice du paquet et k le numéro du cycle dans le paquet ($k = (k_3, k_2, k_1)_4$). Pour calculer $a(p, k, i)$, on calcule d'abord $\overline{k}_0^i = (3 + i - (k_3 + k_2 + k_1))\%4$ afin de déterminer l'indice du point qui sera situé dans le banc i, i.e. $(k_3, k_2, k_1, \overline{k}_0^i)_4$. On en déduit, en fonction de la règle de permutation des dimensions en fonction de p, que l'adresse d'écriture sera donnée par :

$$a(p, k, i) = (rot_p(k_3, k_2, k_1, \overline{k}_0^i)_4)/4 \qquad (3)$$

avec rot_p la permutation circulaire à droite de p positions des indices du quadruplet.

La figure 6(d) montre le placement mémoire des quatre paquets en respectant cette loi d'entrelacement. Il y a bien une périodicité de 4 dans le placement des données.



(a) $p+1$ (b) $p+2$ (c) $p+3$ (d) $p+4$

FIG. 6 – placements en mémoire des paquets suivants

4.3 Résultats d'implantation

L'implantation du banc de filtres complet a été effectuée et validée sur la cible FPGA de Xilinx (Virtex2 pro XC2VP70-6-FF1704). La fréquence de fonctionnement de 170,2 MHz est conforme aux spécifications demandées à savoir supérieure à 166,7 MHz. Le taux d'occupation est d'environ 17% (11436 LUTs) pour cette partie du récepteur numérique.

5 Conclusion

Dans le cadre de la réalisation d'un banc de filtres polyphasés, nous avons étudié le problème d'entrelacement en parallèle et en continue de paquets de données. Pour trouver une solution optimale à ce problème nouveau, nous avons tout d'abord construit une solution dans un espace "non contraint" (ici de dimension 4) avant de projeter la solution sur une structure matérielle constituée, en entrée, d'un réseau de rotation, d'un banc de mémoires accessible en parallèle et d'un deuxième réseau de rotation en sortie. La démarche présentée dans le papier peut aisément être généralisée. En effet, nous pouvons l'utiliser avec d'autres paramètres de parallélisme et taille de TFR pour des filtres polyphasés mais il est aussi possible d'étendre cette méthodologie à d'autres types applications.

Références

[1] M. Bellanger. *Traitement numérique du signal*. DUNOD, 1998.

[2] C. Verdier, E. Boutillon, A. Lafage, A. Demeure. *Access ans alignment of arrays for a bidimensional parallel memory*. International Conference on Application Specific Array Processors, ASAP'1994, pages 346-356, San Francisco (USA), 1994.

[3] E. Boutillon, C. Douillard, G. Montorsi, *Iterative Decoding of Concatenated Convolutional Codes : Implementation Issues*. Proceeding of the IEEE, vol. 95, n°6, june 2007.