ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE ÉLECTRIQUE
M.Ing.

PAR
SERIGNE MBAYE FALLO DIA

CONCEPTION ET RÉALISATION D'UN FILTRE À DÉCIMATION PARALLÉLISÉ SOUS FORME DE NOYAU PROGRAMMABLE

MONTRÉAL, LE 08 NOVEMBRE 2006

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. François Gagnon, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Claude Thibeault, codirecteur
Département de génie électrique à l'École de technologie supérieure

M. Jean Belzile, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Jean-Marc Lina, membre du jury
Département de génie électrique à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 02 OCTOBRE 2006

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

CONCEPTION ET RÉALISATION D'UN FILTRE À DÉCIMATION PARALLÉLISÉ SOUS FORME DE NOYAU PROGRAMMABLE

Serigne Mbaye Fallo Dia

SOMMAIRE

Plusieurs applications radio utilisent de plus en plus de hautes fréquences d'échantillonnage qui permettent de numériser directement les signaux radio fréquence (RF). Ceci permet de réduire les composants analogiques qui sont coûteux et peu flexibles. Une autre tendance dans les dispositifs modernes de télécommunication, en l'occurrence les transmetteurs multi-mode, bénéficie aussi de cette numérisation en haute fréquence. Idéalement, tous les modes d'opération supportés doivent utiliser le même étage frontal de traitement analogique et numérique. De ce fait, avec la faisabilité de convertisseurs analogique/numérique utilisant des fréquences d'échantillonnage de l'ordre des GHz (AT84AS008GL de Atmel, Max108 de Maxim et ADC08D1500 de National Semiconductor), l'unité de traitement numérique est poussée à fonctionner à une fréquence de l'ordre des GHz aussi. Cependant, l'implémentation CMOS d'un filtre à décimation fonctionnant à cette fréquence peut poser problème. Une solution à ce problème est de faire du traitement parallèle.

D'un autre coté, l'évolution rapide de la microélectronique a mis à la disposition des concepteurs de FPGA comme d'ASIC des puces intégrant de plus en plus de portes, permettant d'intégrer des systèmes de plus en plus complexes sur une même puce. Ainsi, en gardant un processus de conception identique à ce qui se fait à l'heure actuelle, il faudrait augmenter la taille des équipes de développement et ceci malgré l'évolution des outils et des méthodes. Toutefois, il est démontré qu'au-delà d'une dizaine d'ingénieurs travaillant sur une même puce, l'efficacité n'évolue plus de manière proportionnelle avec l'augmentation de la taille de l'équipe. Ainsi, il apparaît une faille de productivité. Une méthodologie possible pour exploiter cette faille est le *design-reuse*, permettant de réutiliser dans un autre contexte des modules déjà conçus.

Ainsi, l'objectif principal de ce mémoire est de concevoir et de réaliser un filtre à décimation qui offre la possibilité d'opérer à une haute fréquence d'échantillonnage de l'ordre des GHz avec une complexité réduite. Ce filtre doit être réalisé sous forme de noyau programmable avec les moyens technologiques disponibles tels que les FPGA. Trois techniques ont été étudiées. La première et la deuxième technique consistent à utiliser respectivement les filtres RIF et les filtres RII. La troisième, quand à elle, utilise les filtres Cascaded Integrator and Comb (CIC) où plusieurs structures ont été analysées et évaluées.

CONCEPTION AND REALIZATION OF A POLYPHASE DECIMATION FILTER CORE

Serigne Mbaye Fallo Dia

ABSTRACT

One of the trends in modern telecommunication devices is to push the analog/digital boundary as close to the antenna as possible, thus reducing the need of large and costly analog components. This means that we prefer either digitizing intermediate frequency (IF) signal at the highest frequencies practical or even directly digitizing radio frequency signal. Another current trend, namely the use of multi-mode transceivers, also benefits from digitizing the signal at a high IF. Ideally, all the supported operating modes would need only a single analog and digital signal processing (DSP) front-end. With the feasibility of GHz sampling frequency analog to digital converter based on time interleaving (AT84AS008GL of Atmel, Max108 of Maxim and ADC08D1500 of National Semiconductor), the DSP is pushing toward GHz rate for decimation. However, CMOS implementation of digital downconversion (or decimation) at such a high frequency could be a problem. One solution to this problem is to use parallel processing. A wise way to implement parallel structure in DSP is polyphase decomposition.

Silicon technology has progressed to allow chip with tens of millions of transistors. This not only promises new levels of integration onto a single chip, but also allows more features and capabilities in reprogrammable technology. Keeping the same conception process as what is done actually, the number of designers must be increased. But it has been proven that over a ten of engineers working on the same chip, the efficiency is no longer growing proportionally with the number of engineers. So, the exploitation of the chip resources, in a reasonable design time, is practically impossible. A solution to this problem is the design-reuse.

Thus, the goal of this project is to build a parameterized, flexible and very high speed firmware digital decimation filter VHDL core with reasonable complexity. The target device is an FPGA since they are appropriate choices for demanding real-time signal processing, as they typically include dedicated memory blocks and multipliers that greatly enhance the computing capabilities. Hence, three techniques are considered. The first one consists of using FIR filters. The second one uses IIR filters and the third one the Cascaded Integrator and Comb filter (CIC). A number of various architectures are evaluated and compared and the best one is implemented in VHDL.

REMERCIEMENTS

Ce mémoire a été réalisé au sein du LACIME (Laboratoire de communication et d'intégration de la microélectronique) de l'École de technologie supérieure, en collaboration avec la chaire Ultra-Electronics TCS en communication sans fils.

Je remercie les professeurs François Gagnon et Claude thibeault pour leur support et leurs conseils qui m'ont permis de mener à bien le projet.

Je remercie aussi tous mes collègues du laboratoire LACIME, en particulier Marie-Eve Grandmaison, Stéphane Cormier et Benoît Châtelain avec qui j'ai eu des discussions qui m'ont permis de trouver des solutions ou inspiré de nouvelles idées.

Je tiens à remercier tous mes amis (es) pour leur soutien moral dans les moments difficiles, particulièrement Djiby et Badou.

Je remercie chaleureusement mes chers parents, mes frères et sœurs qui n'ont cessé de m'encourager dans mon cheminement et ce même de l'autre coté de l'atlantique.

Je remercie infiniment Mame Fama pour son soutien moral et surtout sa patience durant toute la période de ce travail.

Finalement, merci à toutes les personnes qui ont contribué de prés ou de loin à mener à terme ce travail.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

Le développement spectaculaire des circuits intégrés, de plus en plus complexes et puissants, permet une croissance importante des applications en télécommunication sans fil. Notamment, la radio mobile, les radars, la vidéo conférence numérique... Ces nouvelles applications nécessitent, généralement, de transmettre une quantité importante d'information à haute vitesse. De plus, l'utilisation de hautes fréquences d'échantillonnage permet de remplacer plusieurs composants analogiques qui sont coûteux et peu flexibles car l'unité de traitement numérique est rapprochée de l'antenne. Ceci a pour but d'augmenter la flexibilité au niveau de la programmation et de la configuration *hardware* pour pouvoir supporter plusieurs standards de communication. Cependant, le traitement numérique de ces signaux en haute fréquence s'avère complexe car les dispositifs de traitement numérique des signaux disponibles ne peuvent assurer une haute fréquence d'échantillonnage de l'ordre des gigahetz (GHz). Un autre défi pour les communications sans fil est la réduction de la puissance consommée. Une des solutions est de réduire la tension d'alimentation [1]. Cette solution engendre une augmentation des délais ce qui limite à son tour la fréquence d'opération. Une solution efficace à ce problème est le traitement numérique multicadence.

En effet, la technique de filtrage multicadence [2] a été introduite dans le but de réduire la vitesse de calcul dans les filtres numériques. Cette technique inclut les opérations de décimation et d'interpolation qui permettent de modifier la fréquence d'échantillonnage. L'introduction du filtrage multicadence a suscité l'intérêt de plusieurs chercheurs tels que R.E. Crochiere et L.R. Rabiner qui ont été les premiers à présenter une conception optimale en terme de minimisation du nombre de multiplicateurs dans les filtres numériques à Réponse Impulsionnelle Finie (RIF) décimateurs multiétage [3] [4]. Ils ont apporté des contributions au plus haut niveau dans le domaine de filtrage numérique en général et le filtrage multicadence en particulier. Cette approche [3] [4], qui consiste à réduire le nombre de multiplicateurs dans les filtres FIR, a ensuite inspiré Peled et Liu [5] à remplacer les multiplicateurs par des additionneurs. Dans la même optique,

Goodman et Carry ont démontré qu'un choix judicieux et adéquat des coefficients des filtres demi bande décimateurs et interpolateurs peut être un moyen efficace pour la réalisation de filtres performants [6].

En 1981, un article de E.B Hogenauer [7] a introduit de nouveaux filtres. Cet article présente une méthode efficace et économique pour la conception des filtres multicadences décimateurs et interpolateurs. Cette méthode consiste à mettre en cascade deux blocs. Le premier est constitué d'étages d'intégrateurs ; le second est constitué d'étages de différentiateurs. Ces filtres sont appelés Cascaded Integrator and Comb (CIC). Les filtres CIC sont plus performants que ceux décrits par Peled et Liu [5]. Le grand avantage des filtres CIC réside dans le fait que leur réalisation ne nécessite ni multiplicateurs ni stockage de coefficients. Cet avantage représente un excellent moyen d'augmenter la vitesse de traitement tout en diminuant la taille des filtres. Toutefois, l'architecture telle que présentée ici, n'est pas appropriée pour des applications à hautes fréquences. Ainsi, vu l'efficacité des filtres CIC, beaucoup de chercheurs se sont mis à les étudier pour contribuer à l'augmentation de la fréquence de fonctionnement de ces derniers.

Y. Djadi, et al. [8] ont conçu un filtre numérique à décimation ou à interpolation programmable basé sur la structure des filtres CIC. Le filtre peut être configuré en décimateur ou en interpolateur. Le facteur de décimation ou d'interpolation est programmable et peut prendre n'importe quelle valeur entre 10 et 256. L'ordre du filtre est fixé à 5. Les résultats de simulation ont montré que la fréquence maximale de fonctionnement de ce filtre pouvait atteindre 50 MHz.

Alan Kwentus, et al. [9] ont conçu et fabriqué un filtre CIC à décimation programmable avec un ordre fixé à 6. Le facteur de décimation est programmable et peut prendre n'importe quelle valeur puissance de 2 située entre 2 et 1024. La fréquence maximale de fonctionnement de ce filtre est de 250 MHz pour une technologie CMOS 0.8 micron.

Les étages d'intégrateurs et de différentiateurs ont été implémentés en utilisant l'arithmétique *carry-save* dans l'optique d'augmenter la fréquence de fonctionnement. Kei-Yong Khoo et al [10] ont aussi présenté une architecture efficace du premier intégrateur basée sur le *carry-save*.

Hyuk Jun Oh et Yong H. Lee [11] ont utilisé un polynôme interpolateur d'ordre 2 suivis d'un filtre CIC décimateur pour réduire la perte dans la bande passante causée par le filtre CIC. Cependant, l'atténuation dans la bande coupée est un peu dégradée. Pour remédier à cela, une méthode d'optimisation, qui suggère de mettre en cascade le polynôme d'interpolateur d'ordre 2 avec un filtre RIF programmable, à été présentée dans [12].

Une autre façon d'implémenter les filtres CIC décimateurs a été présentée par Yonghong Gao et al [13]. Dans cette méthode, le facteur de décimation R est décomposé en M facteurs de puissance de 2. La fonction de transfert peut être réécrite comme le produit de M filtres FIR identiques et de petit ordre. La fréquence d'échantillonnage de cette implémentation diminue d'étage en étage par un facteur de 2. Le premier étage est crucial à la fréquence de fonctionnement d'où un décomposition polyphasée.

Il existe aussi des *cores* (noyaux programmables) commerciaux de filtres CIC décimateurs complètement programmables mais leur fréquence maximale de fonctionnement tourne autour de 200 MHz.

Tel que décrit ci dessus, plusieurs travaux ont été réalisés dans le but d'avoir des filtres décimateurs rapides et efficaces. Cependant, beaucoup de limitations sont associées à ces méthodes. Par exemple, pour la méthode utilisée dans [9] et [13], le facteur de décimation est subdivisé en M facteurs de puissance 2. Pour la référence [8], le facteur de décimation peut prendre n'importe quelle valeur située entre 10 et 256 ce qui est un peu limité. La largeur binaire des entrées/ sorties n'est pas programmable non plus. Pour

les produits commerciaux, comme le CIC de Xilinx, ils sont généralement très intéressants du point de vue programmabilité mais sont très limités en fréquence. Ainsi, dans le but de surpasser ces limitations, le filtre décimateur doit être complètement programmable avec une grande fréquence de fonctionnement. Pour y arriver, le traitement parallèle s'avère incontournable car il permet non seulement d'augmenter la vitesse de traitement mais aussi une réduction de la puissance consommée.

L'objectif principal de notre recherche est d'étudier, de concevoir et de réaliser en VHDL un noyau de filtre à décimation complètement programmable et capable de recevoir des données échantillonnées à 1.2 GHz sur des puces ne supportant que des cadences de 150 à 200 MHz. Les spécifications détaillées que le noyau doit respecter sont les suivantes : largeur binaire des entrées variable entre 8 et 16 bits; largeur binaire des sorties variable entre 16 et 24 bits; le facteur de décimation variable entre 2 et 16384. Il doit aussi supporter un nombre d'entrées parallèles maximal de 8.

Ainsi, ce mémoire de 5 chapitres est organisé comme suit : le premier chapitre introduit d'une façon brève mais concise les concepts de base des filtres à décimation. Le deuxième chapitre présente les spécifications globales que le noyau doit respecter ainsi que le choix de l'architecture à implémenter. L'architecture des filtres CIC décimateurs étant choisie, le troisième chapitre porte sur les paramètres de conception de ces derniers. Pour avoir le filtre à décimation sous forme de noyau programmable, le quatrième chapitre porte sur l'implémentation en VHDL et la réalisation sur FPGA. Enfin, pour montrer la contribution de ce noyau dans le domaine du traitement numérique en haute cadence, le cinquième chapitre présente une comparaison au niveau des ressources matérielles requises et de la fréquence maximale de fonctionnement de ce noyau avec ceux disponibles commercialement.

CHAPITRE 1

CONCEPTS DE BASE DES FILTRES À DÉCIMATION

1.1 Introduction

Le réajustement de la fréquence d'échantillonnage suivant le signal d'intérêt est une tâche assez fréquente dans le traitement numérique du signal. Les deux principales raisons de ce réajustement de fréquence sont : la performance et le coût. Par exemple, en audio, trois fréquences d'échantillonnage sont actuellement utilisées : 32 KHz pour la diffusion, 44.1 KHz pour le disque compact numérique et 48 KHz dans la bande sonore numérique. Des systèmes, qui utilisent plusieurs fréquences d'échantillonnage différentes, sont connus sous le nom de systèmes multicadences. Ces systèmes, comparés aux systèmes traditionnels, offrent de meilleures performances à un coût réduit [14]. Ce réajustement de fréquence peut se faire de deux façons : la première façon est de passer d'une grande fréquence d'échantillonnage à une fréquence d'échantillonnage beaucoup plus petite (décimation) à l'aide d'un filtre à décimation. La deuxième façon, consiste à transformer une petite fréquence d'échantillonnage en une beaucoup plus grande (interpolation) à l'aide d'un filtre à interpolation. Ainsi, dans ce chapitre, les concepts de base du traitement numérique multicadence du signal et quelques structures de filtres à décimation sont présentés. La priorité est accordée à la décimation plutôt qu'à l'interpolation.

1.2 Traitement numérique multicadence du signal

La conversion d'une fréquence d'échantillonnage d'un signal à une fréquence plus grande ou plus petite peut se faire par deux méthodes différentes. La première méthode consiste à utiliser un CNA pour convertir le signal numérique en signal analogique et le rééchantillonner à la fréquence désirée en utilisant un CAN. Cette méthode augmente les

ressources du système et introduit de l'erreur de quantification à chaque fois qu'un CAN ou un CNA est utilisé. La deuxième méthode consiste à faire la conversion dans le domaine numérique en utilisant un facteur de décimation ou d'interpolation. Cette deuxième méthode est plus avantageuse que la première parcequ'elle n'utilise que des filtres numériques. Chester a fait la comparaison entre les filtres analogiques et les filtres numériques [15]. Il explique que les filtres numériques, par rapport à la surface occupée et la puissance, sont plus efficaces que les filtres analogiques dans le cadre des applications qui demandent une phase linéaire, une bonne atténuation dans la bande arrêtée et une petite déviation dans la bande passante. Un autre aspect qui rend attrayant la méthode purement numérique est la flexibilité des filtres numériques par rapport à une réponse programmable ou adaptive.

1.3 Décimation

La façon la plus simple de décimer une séquence $x(n)$ par un facteur de décimation R est de choisir chaque $R^{iéme}$ donnée de la séquence et supprimer le reste. Si $x(n) = [\ldots\ldots, x(-1), x(0), x(1), x(2), \ldots\ldots\ldots]$, la séquence décimée est donnée par $[\ldots\ldots, x(-2R), x(-R), x(0), x(R), x(2R), \ldots\ldots]$ et peut s'écrire sous forme d'équation comme suit [16] :

$$x_R(n) = (\downarrow R)x(n) = x(Rn) \qquad \text{avec} \quad n = 0, \pm 1, \pm 2, \ldots\ldots\ldots \qquad (1.1)$$

Considérons le cas simple d'une décimation par un facteur de 2 et essayons de déduire la relation entre les spectres de la séquence originale et décimée :

$$x_2(n) = [\ldots., x(-2), x(0), x(2), x(4), \ldots\ldots.]$$

Posons $v(n)$, le produit de la séquence $x(n)$ par $h(m)$ défini par l'équation (1.2). Le résultat est donné par l'équation (1.3).

$$h(m) = \begin{cases} 1, & m = 2n \\ 0, & m \neq 2n \end{cases} \tag{1.2}$$

$$v(n) = [....., x(-2), 0, x(0), 0, x(2), 0, x(4), ...] \tag{1.3}$$

$v(n)$ peut être écrit en fonction de $x(n)$ comme suit [16] :

$$v(n) = \frac{1}{2}\left[x(n) + (-1)^n x(n)\right] \tag{1.4}$$

Étant donné que -1 = $e^{-j\pi}$, alors l'équation (1.4) devient :

$$v(n) = \frac{1}{2}\left[x(n) + e^{-j\pi n} x(n)\right] \tag{1.5}$$

1.3.1 Analyse spectrale

L'une des propriétés de la transformée de Fourier dit qu'un décalage par ω_0 dans le domaine fréquentiel entraîne une multiplication par $e^{j\omega_0 n}$ dans le domaine temporel. Cette propriété sur le décalage fréquentiel est représentée par l'équation suivante :

$$x(n)e^{j\omega_0 n} \quad \xleftrightarrow{TF} \quad X(\omega - \omega_0) \tag{1.6}$$

Ainsi, en utilisant cette propriété, la transformée de Fourier discrète de l'équation (1.5) est donnée par :

$$V(\omega)=\frac{1}{2}[X(\omega)+X(\omega+\pi)] \tag{1.7}$$

Maintenant, essayons de trouver la relation entre les spectres de $v(n)$ et $x_2(n)$. La transformée de Fourier discrète de $x_2(n)$ est donnée par [16]:

$$\begin{aligned} X_2(\omega) &= \sum_{n=-\infty}^{\infty} x_2(n)e^{-j\omega n} \\ &= \sum_{n=-\infty}^{\infty} v(2n)e^{-j\omega n} \\ &= \sum_{m=-\infty}^{\infty} v(m)e^{-j\frac{\omega}{2}m} \qquad (1.8) \\ &= V\left(\frac{\omega}{2}\right) \qquad (1.9) \end{aligned}$$

Dans l'équation (1.8), l'indice m est pair et égal à $2n$. Ceci veut dire que seules les composantes paires de $v(m)$ sont prises en compte. Alors, on pourrait se poser des questions sur la validité de cette équation. Mais, comme les composantes impaires de $v(m)$ sont nulles, alors le fait de ne pas les tenir en compte n'influence en rien l'équation (1.8). En combinant les équations (1.7) et (1.9), le spectre du signal décimé en fonction du signal original peut s'écrire comme suit [16] :

$$X_2(\omega)=\frac{1}{2}\left[X\left(\frac{\omega}{2}\right)+X\left(\frac{\omega}{2}+\pi\right)\right] \tag{1.10}$$

Cette équation montre que le spectre du signal décimé n'est rien d'autre que la somme des spectres rétrécis et décalés en fréquence du signal original. La figure 1 illustre bien ce processus [16].

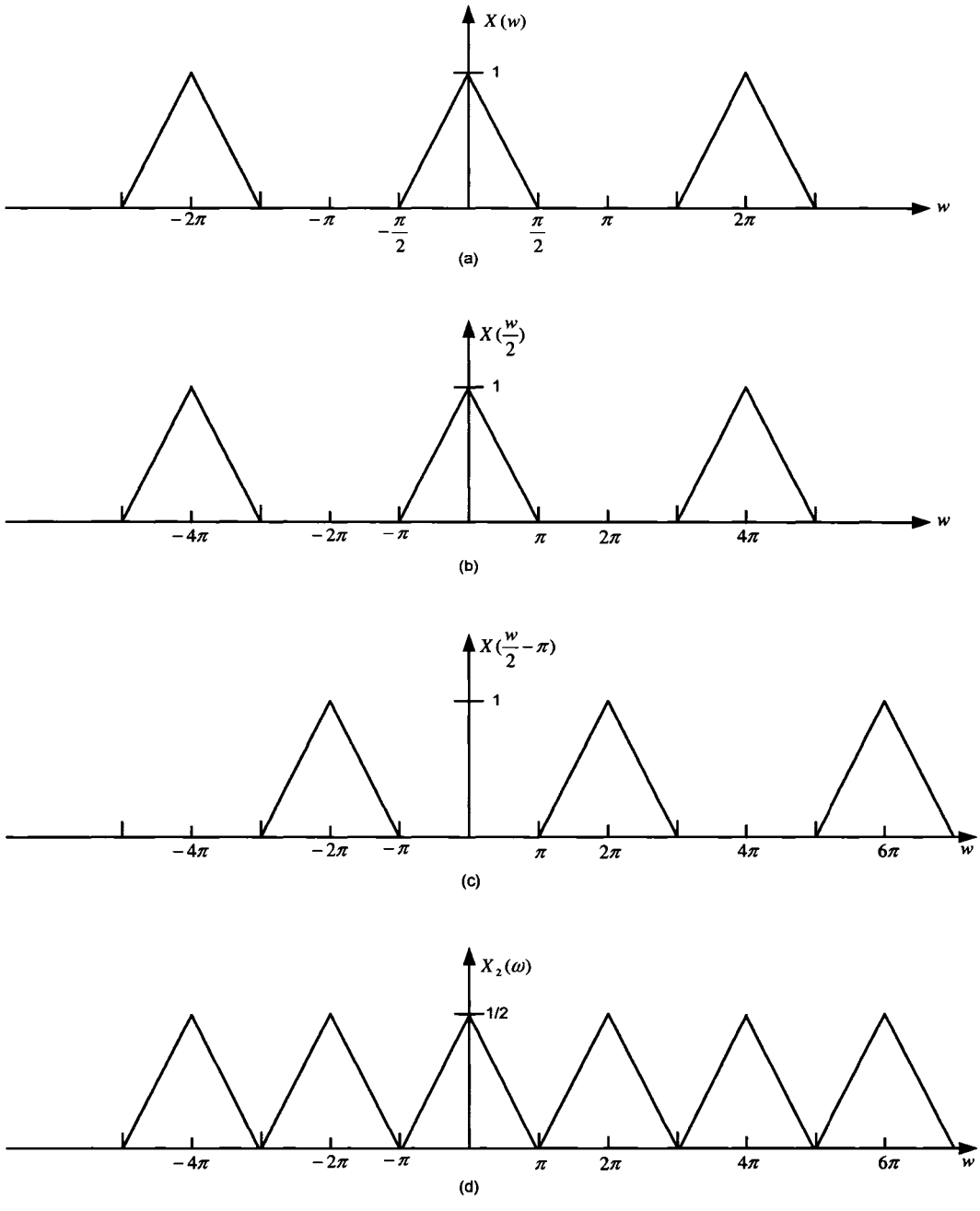
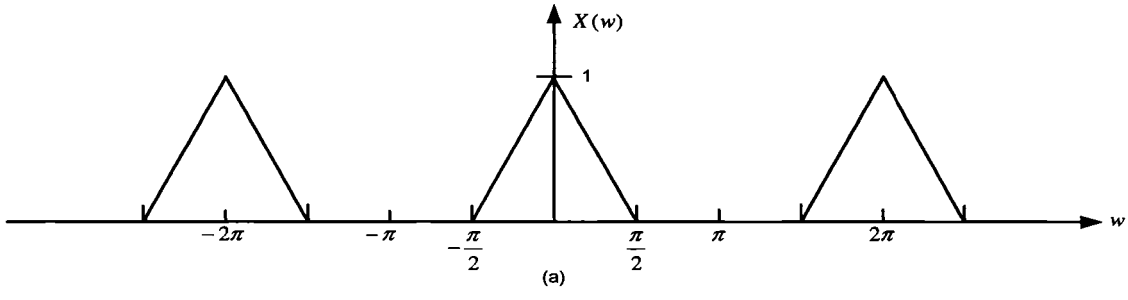


Figure 1 Obtention du spectre du signal décimé à partir du spectre du signal original

La figure 1(a) montre le spectre du signal original. Le premier terme de l'équation (1.10), $X\left(\frac{\omega}{2}\right)$, est présenté dans la figure 1(b). Il est intéressant de remarquer que la fréquence est divisée par 2. Ceci se traduit par une multiplication par 2 sur l'axe horizontal d'où un étirement du spectre. Le deuxième terme de l'équation (1.10), $X\left(\frac{\omega+2\pi}{2}\right)$, est présenté dans la figure 1(c). Cette figure est obtenue en décalant en fréquence la figure 1(b) par 2π. Le spectre du signal décimé, présenté dans la figure 1(d), est la somme des figures 1(b) et 1(c) avec une amplitude de 1/2.

1.3.2 Effet de chevauchement

Le spectre du signal original et du signal décimé sont repris respectivement à la figure 2 (a) et (b). La figure 2(b) est obtenue en étirant chaque triangle de la figure 2(a) jusqu'à ce que sa largeur soit doublée. Nous pouvons observer que les triangles de la figure 2(b) sont à la limite du chevauchement. Ceci veut dire que les triangles sont à la limite de la distorsion de repliement (*aliasing*).



(a)

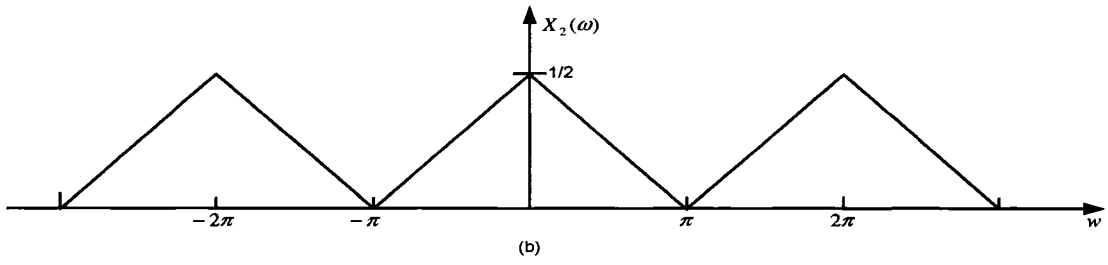
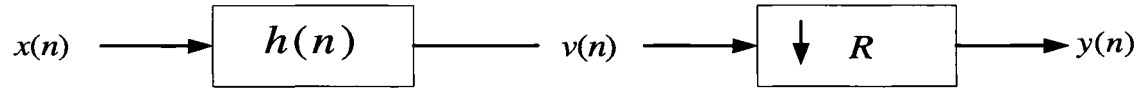
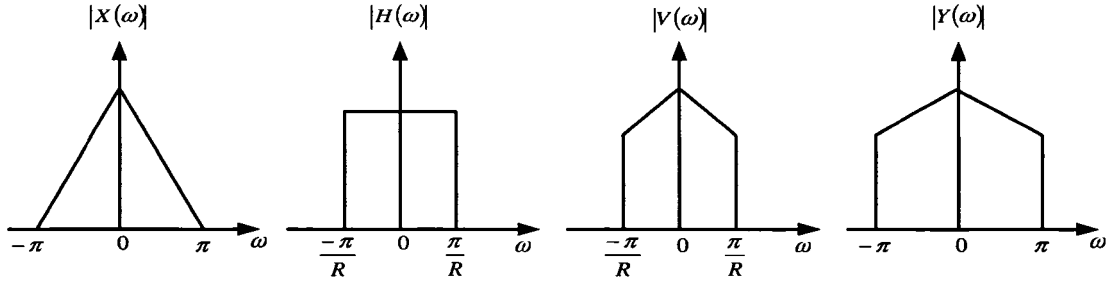


Figure 2 Spectres du signal original et du signal décimé

La figure 2 montre qu'il peut y avoir du chevauchement dans le spectre du signal décimé si le spectre du signal d'entrée n'est pas dans la plage $\left[\frac{-\pi}{R},\frac{\pi}{R}\right]$. Le chevauchement dans le spectre du signal décimé rend impossible la reconstruction exacte du signal d'entrée. Pour éviter ce problème, un filtre passe bas doit être utilisé pour limiter la largeur de bande du signal d'entrée. Ceci est illustré à la figure 3. Celle-ci montre bien l'utilité du filtre passe bas dans le processus de décimation. Le signal à la sortie du filtre passe bas $|V(\omega)|$ est limité en fréquence entre $\frac{-\pi}{R}$ et $\frac{\pi}{R}$. Le facteur de décimation, quand à lui, étend entre $-\pi$ et π le spectre présentée à son entrée en choisissant chaque $R^{iéme}$ échantillon. Le spectre du signal décimé est $|Y(\omega)|$. Cette figure montre bien que, sans l'utilisation du filtre passe bas pour limiter en fréquence le signal d'entrée, la décimation crée du chevauchement pour les signaux dont le spectre se situe en dehors de l'intervalle $\left[\frac{-\pi}{R};\frac{\pi}{R}\right]$.



(a) bloc diagramme du filtre passe-bas $h(n)$ et du facteur de décimation R



(b) relation entre le spectre d'entrée du filtre passe-bas et le spectre de sortie du facteur de décimation

Figure 3 Décimation par R dans le domaine fréquentiel

La méthode utilisée pour aboutir à l'équation (1.10) peut être généralisée pour une décimation avec un facteur R. Ceci donnera la relation générale entre le spectre du signal original et celui du signal décimé. Cette équation générale est donnée par [16] :

$$X_R(\omega) = \frac{1}{R}\sum_{k=0}^{R-1} X\left(\frac{\omega}{R} + \frac{2\pi k}{R}\right) \tag{1.11}$$

Cette équation peut être étendue dans le domaine Z en faisant les équivalences suivantes :

$$\omega \rightarrow z$$

$$\frac{\omega}{R} \rightarrow z^{1/R}$$

$$\omega R \rightarrow z^{R}$$

$$\omega + \frac{2\pi}{R} \rightarrow z e^{j2\pi/R}$$

Ainsi, l'équation (1.11) s'écrit, dans le domaine Z, comme suit [16] :

$$X_R(z) = \frac{1}{R}\sum_{k=0}^{R-1} X\left(z^{1/R} e^{j2\pi k/R}\right) \tag{1.12}$$

1.4 Structure des filtres décimateurs

Dans un processus de décimation, il est montré que le facteur de décimation R doit être précédé par un filtre passe-bas avec une fréquence de coupure qui est égale à $\frac{\pi}{R}$ pour limiter la largeur de bande du signal d'entrée. La figure 4 montre la structure de forme directe d'un filtre décimateur [16].

Le filtre $H(z)$ d'ordre M permet de limiter la largeur de bande du signal d'entrée. Ce filtre est constitué de M multiplicateurs. Si la fréquence du signal d'entrée est f_s, alors le filtre exécute $M \times f_s$ multiplications par cycle. Cette structure peut être améliorée en utilisant la décomposition polyphasée. Cette technique est illustrée au chapitre 2.

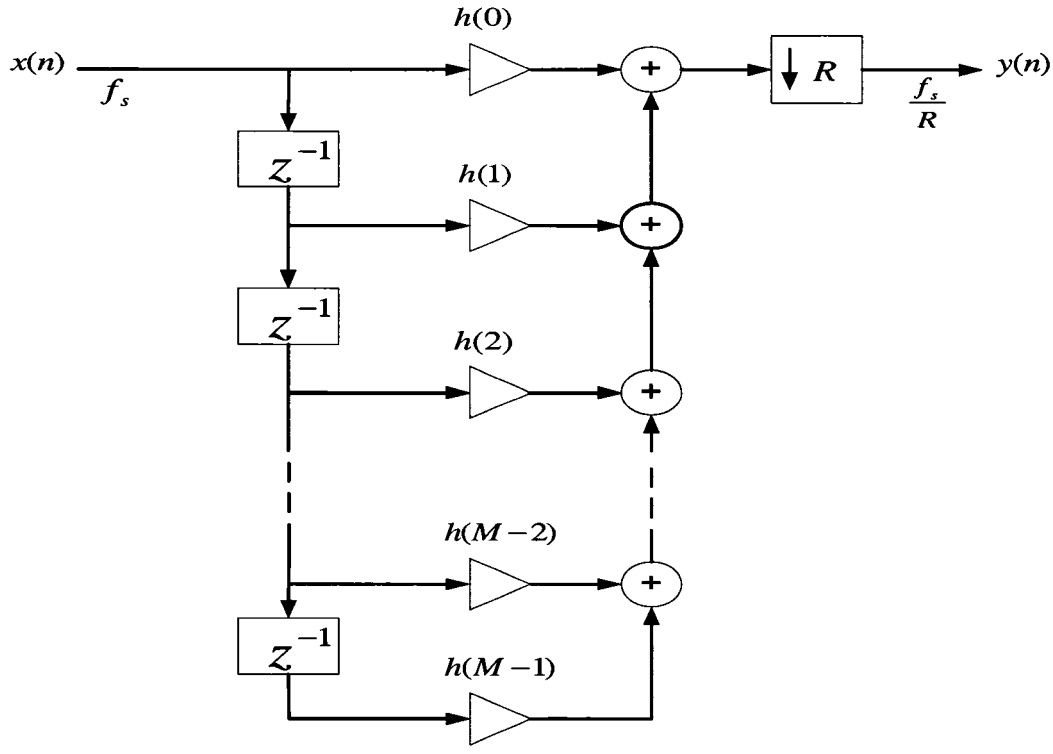


Figure 4 Structure de forme directe d'un filtre décimateur

1.5 Applications des filtres décimateurs

Les filtres décimateurs peuvent être utilisés dans plusieurs domaines d'application. Entre autres, leur utilisation dans les récepteurs radio numériques est très populaire. Un récepteur radio numérique effectue une décimation et une démodulation d'un signal RF à bande étroite incorporé dans un paquet de fréquences assignées à d'autres services. L'architecture traditionnelle d'un récepteur radio qui effectue cette tâche est présentée à la figure 5 [14].

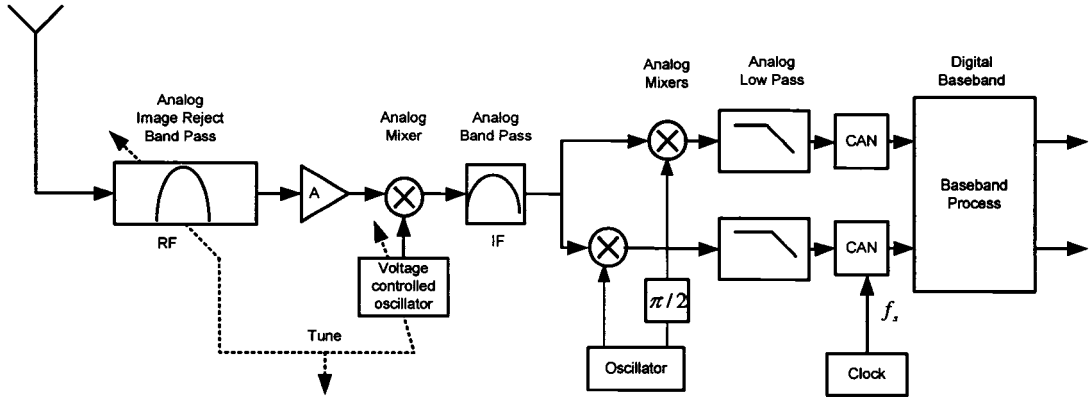


Figure 5 Architecture d'un récepteur radio standard

Cette architecture standard effectue deux translations de fréquence. Le signal d'entrée passe à travers un filtre passe bande qui rejette les images, puis par un amplificateur. Le signal RF amplifié est ensuite translaté en une fréquence intermédiaire IF pour limiter la largeur de bande du signal. Pour pouvoir être traité en bande de base, le signal à la sortie du filtre IF est retranslaté en fréquence par des mixeurs en quadrature qui sont suivis par des filtres analogiques qui effectuent le contrôle final sur la largeur de bande du signal. Chaque signal en quadrature est converti en numérique par une paire de convertisseurs analogiques numériques (CAN). La sortie des CAN est traitée par une unité de traitement numérique qui permet de faire la synchronisation, l'égalisation, la démodulation, la détection et le décodage de canal.

Cependant, le déséquilibre du gain et de la phase des mixeurs en quadrature, les filtres analogiques en bande de base et les CAN sont à l'origine de la diaphonie qui peut exister entre le signal en phase et le signal en quadrature. Les CAN injectent du DC dans le signal en bande de base et les filtres analogiques introduisent de la distorsion au niveau du délai de groupe.

Ces problèmes numériques, causés par les composantes analogiques, peuvent être résolus en utilisant des algorithmes adaptatifs. Ceci augmenterait la complexité du récepteur. Au lieu de réparer ces problèmes analogiques, il est beaucoup plus intéressant de numériser le plus tôt possible sur la chaîne de réception. Ainsi, en plus d'éviter la dégradation des performances due au déséquilibre des signaux en phase et en quadrature qui est provoqué par la tolérance des composantes analogiques, d'éviter le coût des mixeurs et d'éviter la distorsion au niveau du délai de groupe causé par les filtres analogiques, cette technique permet d'augmenter la flexibilité des composantes. Ceci permet d'avoir des filtres avec des largeurs de bandes et des facteurs de décimations programmables.

La figure 6 [14] présente l'architecture d'un récepteur radio de deuxième génération. La conversion analogique numérique se fait maintenant au niveau de l'étage IF. Dans cette architecture, les CAN doivent fonctionner avec une plus grande fréquence d'échantillonnage. Il est aussi intéressant de remarquer que la translation en bande de base se fait par un filtre et un décimateur numérique. Un autre avantage de cette translation numérique en bande de base est que les filtres numériques sont conçus de façon à avoir des réponses en phases linéaires.

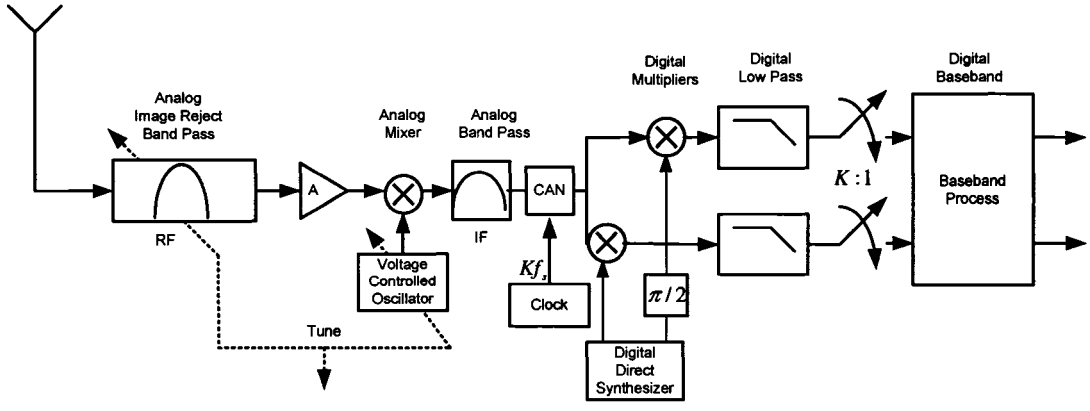


Figure 6 Architecture d'un récepteur radio de deuxième génération

1.6 Conclusion

Les concepts de base de la décimation ont été présentés dans ce chapitre. Ainsi, l'analyse temporelle et fréquentielle du signal décimé ont été faites. La structure standard des filtres à décimation ainsi que leur application dans les récepteurs radio ont été décrits. Le chapitre suivant présentera les spécifications du noyau et le choix de l'architecture à implémenter.

CHAPITRE 2

SPÉCIFICATIONS ET CHOIX DE L'ARCHITECTURE

2.1 Introduction

La compréhension des besoins du client, en termes des spécifications du noyau programmable, est très importante pour un concepteur de circuits logiques. Cette compréhension lui permet de choisir l'architecture la plus apte à répondre aux spécifications désirées. Ainsi, dans ce chapitre, les différentes spécifications auxquelles doit répondre le noyau programmable et une comparaison des architectures matérielles sont présentées.

2.2 Spécifications

Cette section expose les spécifications que le module du noyau programmable doit remplir, notamment au niveau des paramètres programmables et de la reconfigurabilité. Le module conçu doit être configurable c'est-à-dire conçu pour résoudre un problème général, portable c'est-à-dire conçu pour différentes technologies et pour différents outils, déboguable c'est-à-dire vérifié avec un haut niveau de confiance et finalement lisible c'est-à-dire documenté clairement au niveau des applications et restrictions.

2.2.1 Langage et technologie pour l'implémentation

Le noyau doit être programmé en langage de type HDL (Hardware Description Language) pour faciliter la compréhension et permettre l'emploi de variables de configuration. Le langage de programmation choisi pour la réalisation du noyau programmable est le VHDL (Very High Speed Integrated Circuit Hardware Description Language) avec un code entièrement synthétisable. Tous les paramètres de

configurations du noyau sont programmables avant synthèse à l'exception du facteur de décimation qui peut changer de valeur après la synthèse. En ce qui concerne la technologie, le noyau est conçu pour cibler les plates-formes de type FPGA (Field Programmable Gate Array).

2.2.2 Ressources et vitesse

Le noyau doit être conçu efficace avec une complexité acceptable. Pour ce faire, les ressources matérielles limitées telles que la logique de calcul et de contrôle ainsi que la mémoire doivent être minimisées. En ce qui concerne la fréquence d'opération, elle doit être maximisée parce que l'objectif principal est de pouvoir filtrer un signal échantillonné à une fréquence de 1.2 GHz. La latence, délai en cycles d'horloge entre l'entrée du premier échantillon et la première sortie du filtre, est négligée au détriment de la fréquence d'opération. Ceci veut dire que la priorité est accordée à la fréquence d'opération et même si l'augmentation de cette dernière entraîne une augmentation de la latence.

2.2.3 Paramètres fixes

Le module du filtre reçoit ses données d'entrée de façon parallèle. La sortie est produite de manière sérielle. Le module doit posséder un port d'entrée et un port de sortie, chacun ayant une largeur spécifiée par l'utilisateur. Le module a aussi une entrée pour l'horloge `clk` et une pour un signal d'activation de l'horloge `nd`. Il doit aussi posséder un mécanisme pour la synchronisation avec des modules extérieurs, tel qu'un signal pour lui indiquer le début d'une séquence et un signal pour informer que les résultats sont valides.

2.2.4 Paramètres programmables

Comme le but est de réaliser un noyau de filtre à décimation parallélisée qui est configurable, portable, déboguable et lisible, alors tous ses paramètres doivent être programmables. Les différents paramètres sont énumérés et détaillés dans les sous-sections suivantes.

2.2.4.1 Ordre du filtre

L'ordre du filtre dépend entièrement des spécifications requises. Ainsi, il faut établir une plage d'ordre du filtre qui permet de couvrir plusieurs applications. De ce fait, l'ordre minimal du filtre permet de connaître la limite inférieure et l'ordre maximal la limite supérieure. Un bon choix de l'ordre permet de minimiser les ressources logiques ce qui pourrait minimiser la complexité.

2.2.4.2 Largeur des mots binaires

Les ressources logiques utilisées par le noyau ainsi que la précision des calculs dépendent entièrement de la largeur des mots binaires. De ce fait, pour avoir une complexité acceptable et une fréquence d'opération efficace, un bon choix de largeurs des mots binaires est nécessaire.

La largeur des ports en entrée comme en sortie doit pouvoir être choisie par l'utilisateur comme bon lui semble. Cependant, vu que le noyau doit pouvoir filtrer un signal échantillonné à 1.2 GHz, la largeur maximale des mots binaires à l'entrée ne peut pas dépasser 16 bits. Ceci est dû au fait que pour l'instant et pour quelques années encore, aucun convertisseur analogique numérique ne peut fournir en sortie plus que 16 bits avec une telle fréquence d'échantillonnage. Ainsi la largeur des mots binaires à l'entrée est programmable entre 8 et 16 bits.

Dans les filtres à décimation, la largeur des mots binaires à la sortie est souvent plus grande que celle à l'entrée ce qui permet d'éviter les débordements à l'intérieur du filtre. Cependant, la largeur à la sortie est tellement grande qu'il faut la tronquer où l'arrondir ce qui influence beaucoup la précision des résultats. En regardant les besoins de plusieurs applications et pour pouvoir toutes les couvrir, la largeur des mots binaires à la sortie doit être programmable entre 16 et 24 bits.

2.2.4.3 Largeur des bus de données internes

Les largeurs des différents bus de données internes doivent être définies comme paramètres de manière à conserver ou à améliorer la précision des résultats. Pour ce faire, des mécanismes d'augmentation de la largeur des mots binaires pour empêcher le débordement et/ou pour réduire la largeur, tels que l'arrondi ou la troncation, doivent être employés. Ainsi il serait intéressant de créer des fonctions qui réalisent ces mécanismes.

2.2.4.4 Facteur de décimation

Le facteur de décimation est principalement constitué d'un simple compteur qui permet de réduire la fréquence d'échantillonnage. Ce qui veut dire que la fréquence à la sortie du noyau est donnée par la fréquence d'échantillonnage divisée par le facteur de décimation. Une attention particulière mérite d'être portée au facteur de décimation car ce dernier est strictement lié à la largeur des bus à l'intérieur du filtre. Une augmentation du facteur de décimation entraîne une augmentation de la largeur des bus pour prévenir le débordement ce qui entraîne à son tour une augmentation des ressources logiques et une diminution de la fréquence d'opération. Pour couvrir une bonne partie des applications, le facteur de décimation doit être programmable entre 4 et 1024. Il serait aussi attrayant si le facteur de décimation est reconfigurable ce qui veut dire qu'il peut être programmable après synthèse. Pour ce faire, il faut utiliser une architecture

compacte c'est-à-dire une architecture dont la variation du facteur de décimation n'entraîne aucun changement au niveau de l'architecture elle-même. Ceci est faisable en réservant un port d'entrée pour le chargement du facteur de décimation après synthèse.

2.2.4.5 Type de facteur de décimation

Le noyau doit accepter deux types de facteur de décimation qui sont : fixe ou programmable. Une décimation est dite fixe si le facteur de décimation est choisi seulement avant synthèse. Une décimation est dite programmable si le facteur de décimation est modifiable avant et après synthèse. Donc le noyau doit offrir à l'utilisateur le choix d'un facteur de décimation fixe ou programmable. Si l'utilisateur choisit un facteur de décimation programmable, alors il doit spécifier sa valeur maximale pour pouvoir déterminer la largeur maximale des bus internes qui permet d'éviter le débordement.

2.3 Comparaison des architectures matérielles

Dans le cadre de la réalisation d'un noyau programmable avec une complexité réduite, le choix d'une bonne architecture peut faire une très grande différence au niveau de l'implémentation. Les spécifications requises doivent guider le concepteur à faire un bon choix. Ainsi, trois techniques sont étudiées. La première technique consiste à utiliser des filtres numériques RIF. La deuxième technique utilise des filtres numériques RII. La troisième technique, quand à elle, utilise des filtres Cascaded Integrated and Comb (CIC). Avec ces trois techniques, plusieurs structures sont analysées et évaluées. Cependant, il faut noter que ces différentes structures sont comparées seulement au niveau de la réalisation matérielle. Ainsi, après comparaison, on pourra faire le choix de l'architecture qui répond le mieux aux spécifications.

2.3.1 Filtres numériques à réponse impulsionnelle finie (RIF) à décimation

Cette section présente la théorie des filtres RIF à décimation ainsi que leurs structures de réalisations.

2.3.1.1 Théorie des filtres RIF

Un filtre RIF à coefficients constants est un système linéaire invariant dans le temps. La sortie d'un filtre FIR d'ordre L, quand la série $x[n]$ est présentée à son entrée, est donnée par l'équation (2.1) :

$$y[n] = x[n] \otimes f[n] = \sum_{k=0}^{L-1} f[k]x[n-k] \tag{2.1}$$

avec $f[k]$, k allant de 0 à $L-1$, les L coefficients du filtre. Ces L coefficients correspondent à la réponse impulsionnelle du filtre RIF. La transformée en z de l'équation (2.1) donne :

$$Y(z) = F(z)X(z) \tag{2.2}$$

$F(z)$ représente la fonction de transfert du filtre RIF qui est donnée par :

$$F(z) = \sum_{k=0}^{L-1} f[k]z^{-k} \tag{2.3}$$

La structure de forme directe d'un filtre RIF d'ordre L est présentée à la figure 7. Elle est constituée de délais, d'additionneurs et de multiplicateurs.

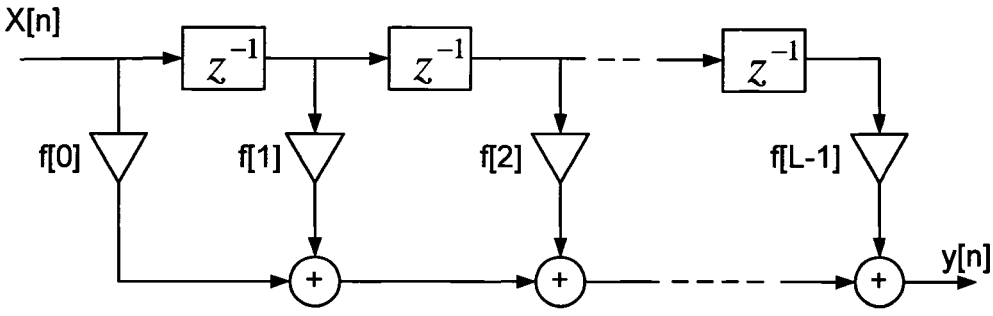


Figure 7 Structure de forme directe d'un filtre RIF d'ordre L

Cette structure de forme directe peut subir une certaine variation pour donner la structure transposée présentée à la figure 8. La structure transposée est, en général, la structure d'implémentation préférée pour les filtres RIF. Les avantages de cette structure résident dans le fait que les entrées n'ont plus besoin de registres supplémentaires pour leur décalage. Cette structure n'a pas besoin non plus d'étages de pipeline supplémentaires pour les additionneurs pour augmenter la fréquence de fonctionnement [16].

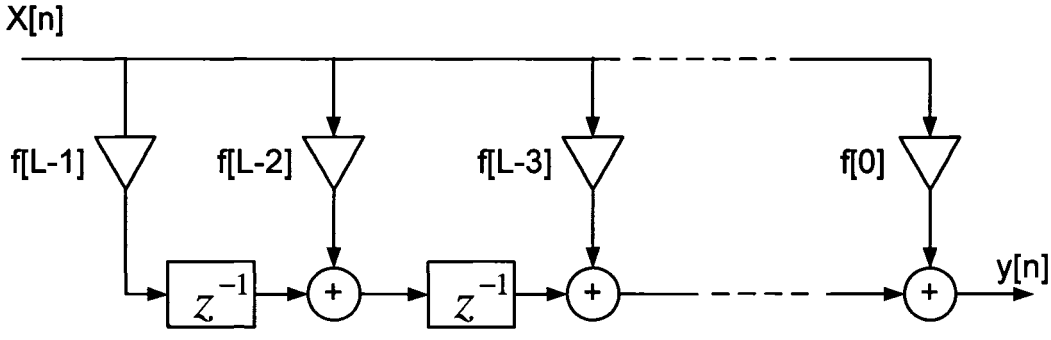


Figure 8 Structure transposée d'un filtre RIF d'ordre L

Le centre de la réponse impulsionnelle d'un filtre RIF est un point important de symétrie. Si ce point est considéré comme le point 0, alors la fonction de transfert s'écrit :

$$F(z) = \sum_{k=\frac{-(L-1)}{2}}^{\frac{(L-1)}{2}} f[k] z^{-k} \tag{2.4}$$

La réponse en fréquence est donnée en posant $z = e^{j\omega T}$ dans l'équation (2.4) :

$$F(\omega) = F\left(e^{j\omega T}\right) = \sum_{k} f[k] e^{-j\omega kT} \tag{2.5}$$

La réponse en phase du filtre est donnée par :

$$\Phi(\omega) = \arctan\left(\frac{\Im(F(\omega))}{\Re(F(\omega))}\right) \tag{2.6}$$

Maintenir l'intégrité de la phase sur une plage de fréquence est très désirée dans beaucoup d'applications comme en communication ou en traitement de l'image. Ceci veut dire que le filtre conçu doit avoir une réponse en phase linéaire. Cette linéarité est mesurée par le délai de groupe définit comme suit :

$$\tau(\omega) = \frac{d\Phi(\omega)}{d\omega} \tag{2.7}$$

Un filtre RIF a une réponse en phase linéaire sur une plage de fréquence si le délai de groupe est constant sur cette plage de fréquence. Le délai de groupe d'un filtre RIF est constant si et seulement si la réponse en fréquence $F(\omega)$ est purement réelle ou purement imaginaire. Ceci veut dire que la réponse impulsionnelle doit être symétrique de façon paire ou impaire, c'est-à-dire :

$$f[n] = f[-n] \quad \text{où} \quad f[n] = -f[-n] \tag{2.8}$$

Cette propriété de symétrie peut être utilisée pour réduire le nombre de multiplicateurs dans la structure directe de L à $\frac{L}{2}$.

2.3.1.2 Structure de réalisation des filtres RIF à décimation

La décomposition polyphasée du filtre RIF ou RII est très importante dans l'implémentation d'un filtre à décimation ou à interpolation. Prenons, comme exemple, la décomposition polyphasée de la structure du filtre FIR présentée à la figure 1. Le filtre FIR est suivi d'un facteur de décimation R. Ceci veut dire qu'une sortie sur R est choisie à la sortie du filtre. Donc, il n'est pas nécessaire de calculer toutes les sommes de produits $x[n]f[n-k]$ de la convolution. Pour éviter ces calculs inutiles, les identités de Noble, présentées à la figure 9, sont utilisées [16].

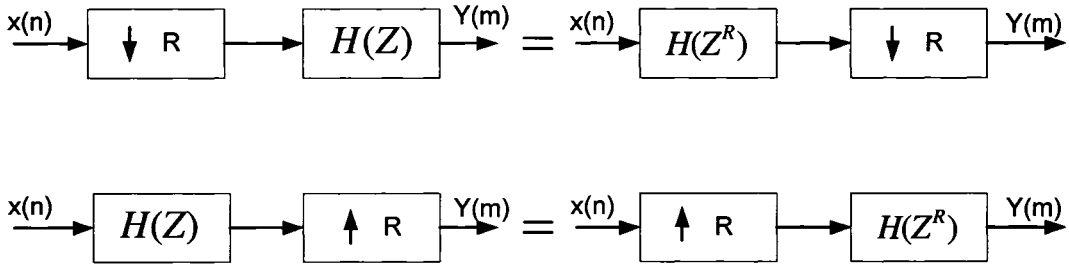


Figure 9 Identités de Noble

Ces identités montrent que $x[0]$ est multiplié par $f[0]$, $f[R]$, $f[2R]$, ... seulement. Les coefficients du filtre sont aussi multipliés par $x[R]$, $x[2R]$, ... Seulement, d'où l'intérêt de diviser la séquence d'entrée en R séquences comme suit :

$$x_0[n] = \{x[0], x[R], ...\}$$
$$x_1[n] = \{x[1], x[R+1], ...\}$$

$$x_{R-1}[n] = \{x[R-1], x[2R-1], ...\} \tag{2.9}$$

et le filtre $f[n]$ en R séquences comme suit :

$$f[n] = \sum_{r=0}^{R-1} f_r[n]$$

$$f_0[n] = \{f[0], f[R], ...\}$$

$$f_1[n] = \{f[1], f[R+1], ...\}$$

$$f_{R-1}[n] = \{f[R-1], f[2R-1], ...\} \tag{2.10}$$

La figure 10 montre l'architecture polyphasée d'un filtre FIR décimateur. Cette architecture est R fois plus rapide que celle du FIR suivi du décimateur.

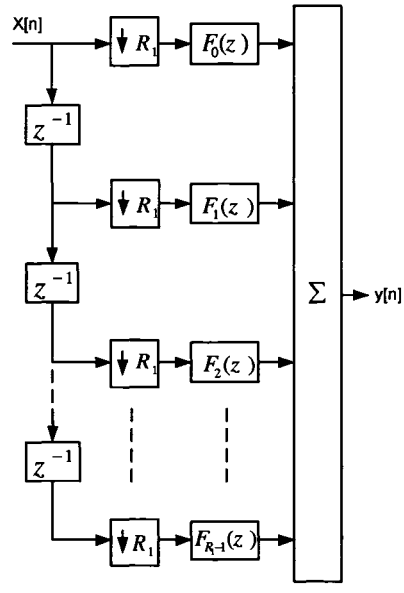


Figure 10 Architecture polyphasée d'un filtre RIF avec un facteur de décimation R_1

Comme indiqué dans le chapitre 1, tout processus de décimation est précédé d'un filtre passe bas avec une fréquence de coupure égale à $\frac{\pi}{R}$ pour limiter la largeur de bande du signal. Ainsi, si le facteur de décimation est grand, alors la largeur de bande du filtre est étroite. Sa bande de transition est préférablement étroite aussi. Ceci nécessite un filtre RIF de grand ordre. L'équation aux différences d'un filtre RIF d'ordre L avec des coefficients symétriques autour de $\frac{L}{2}$, est donnée par [17] :

$$y(n) = f\left(\frac{L}{2}\right)x\left(n-\frac{L}{2}\right)+\sum_{k=0}^{\frac{L}{2}-1} f[k][x(n-k)+x(n-L+k)] \tag{2.11}$$

Cette équation montre que le filtre a besoin de stocker $\frac{L}{2}+1$ coefficients, d'effectuer $\frac{L}{2}+1$ multiplications et L additions par sortie. Donc, un filtre RIF de grand ordre entraîne un grand nombre de multiplications. Malgré le gain R en vitesse par rapport à la structure de forme directe, la décomposition polyphasée du filtre RIF est très limitée pour des applications qui utilisent de hautes fréquences d'échantillonnage.

2.3.2 Filtres numériques à réponse impulsionnelle infinie (RII) à décimation

Comparés aux filtres numériques à réponse impulsionnelle finie (RIF), les filtres RII ayant le même ordre offrent des performances beaucoup plus intéressantes. Ceci est dû au fait que les filtres RII ont des boucles de rétroaction de la sortie sur l'entrée. Ces boucles leur permettent de réaliser les zéros et les pôles de la fonction de transfert d'où la réalisation de fonctions de filtrage beaucoup plus sélectives que les RIF. Les avantages des filtres RII sont : leur design standard en utilisant les prototypes de filtres analogiques est bien compris; des fonctions de filtrage très sélectives peuvent être

réalisées avec des filtres de petits ordres ce qui permet de réduire la complexité. Pour les mêmes spécifications, les filtres RII requièrent moins de calculs comparés aux filtres RIF. Cependant, les filtres RII présentent des caractéristiques indésirables qui sont : la non linéarité de la réponse en phase c'est-à-dire qu'il est difficile d'obtenir une réponse en phase linéaire. Un filtre passe tout pourrait être utilisé pour compenser, ce qui contribuerait à l'augmentation de la complexité. Les boucles de rétroaction peuvent non seulement introduire de l'instabilité mais rendent difficiles les techniques pipelines qui permettent d'augmenter la fréquence de fonctionnement. Ainsi, pour mieux comprendre les tenants et les aboutissants de ces filtres, leur théorie, leur implémentation et des architectures destinées à fonctionner en haute vitesse seront présentés dans cette section.

2.3.2.1 Théories des filtres RII

Les filtres RII sont des filtres récursifs c'est-à-dire qu'ils ont des boucles de rétroaction ce qui fait qu'ils ont une réponse impulsionnelle infinie. La figure 11 montre la forme canonique d'un filtre RII.

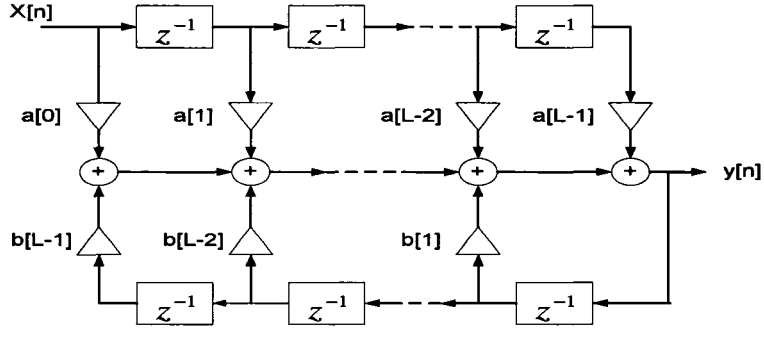


Figure 11 Forme canonique du RII

La fonction de transfert de la forme canonique du RII est donnée :

$$F(z) = \frac{\sum_{l=0}^{L-1} a[l] z^{-1}}{1 - \sum_{l=1}^{L-1} b[l] z^{-1}} \tag{2.12}$$

L'équation aux différences, quand à elle, est donnée par :

$$y(n) = \sum_{l=0}^{L-1} a[l] x[n-l] + \sum_{l=1}^{L-1} b[l] y[n-l] \tag{2.13}$$

L'équation aux différences montre que la sortie d'un filtre RII dépend non seulement des L entrées précédentes mais aussi des $L-1$ sorties précédentes. Pour étudier les propriétés les plus importantes des filtres RII, écrivons la fonction de transfert en fonction des pôles $p_{\infty l}$ et des zéros p_{0l} et en remplaçant z par e^{jwT}.

$$F(w) = |F(w)| e^{\theta(w)} = \frac{\prod_{l=0}^{L-2} \left(p_{0l} - e^{jwT} \right)}{\prod_{l=0}^{L-2} \left(p_{\infty l} - e^{jwT} \right)} \tag{2.14}$$

En analysant l'équation (2.14), nous pouvons déduire les propriétés suivantes :
Un zéro sur le cercle unité, c'est-à-dire $p_0 = e^{jw_0 T}$ produit un zéro dans la fonction de transfert à la fréquence w_0.

Un pôle sur le cercle unité, c'est-à-dire $p_\infty = e^{jw_0 T}$, produit un gain infini dans la fonction de transfert à la fréquence w_0.

Pour avoir une phase linéaire, c'est-à-dire un délai de groupe constant, tous les pôles et les zéros doivent être symétriques au cercle unité donc à $z = 0$.

Donc, en résumé, pour qu'un filtre RII soit stable et ait une réponse en phase linéaire, il faut que tous les zéros soient symétriques au cercle unité et tous les pôles à $z = 0$. Cependant, en réalité, un filtre RII a des zéros et des pôles simples sur le cercle unité. Ceci fait que les filtres RII ne peuvent avoir que des réponses en phase approximativement linéaire. Pour ce faire, un filtre passe tout est utilisé. Ce dernier a un gain unité et un gain en phase non nul qui est utilisé pour assurer la linéarisation dans la bande passante du filtre.

2.3.2.2 Méthode de calcul des coefficients

Habituellement, le calcul des coefficients des filtres RII n'est pas effectué d'une manière directe. Il se fait en utilisant une fonction modèle (gabarit). Celle-ci est une fonction réelle définie sur l'axe des fréquences. Les fonctions modèles, telles que les fonctions de Butterworth, Tchebychev et les fonctions elliptiques, sont des fonctions connues pour leurs propriétés de sélectivité.

Ces fonctions modèles peuvent être choisies comme étant le carré du module de la fonction de transfert à obtenir. Cependant, un obstacle apparaît à leur utilisation au niveau du calcul des coefficients des filtres numériques. En fait, ces fonctions modèles ne sont pas périodiques alors que les fonctions à obtenir ont la période f_e. Pour remédier à cela, il faut établir une correspondance entre l'axe réel et $[0, f_e]$. Une telle correspondance est fournie par une transformation conforme dans le plan complexe et qui respecte les contraintes suivantes :

- Transformer une fraction rationnelle de la variable complexe s en une fraction rationnelle de la variable complexe z.
- Conserver la stabilité

Une transformation simple qui utilise la règle trapézoïdale pour faire l'intégration d'une fonction temporelle continue, connue sous le nom de transformation bilinéaire, est obtenue en faisant correspondre un élément du plan s à un élément du plan z en respectant la relation suivante [16] :

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) \tag{2.15}$$

Trois types de fonctions modèles, à savoir Butterworth, Chebychev et elliptiques, sont très intéressants à investiguer :

La fonction de Butterworth d'ordre N est donnée par [18] :

$$\left|F(w)\right|^2 = \frac{1}{1+(\frac{w}{w_s})^{2N}} \tag{2.16}$$

L'ordre N du filtre est déterminé à partir de son gabarit. Soit à réaliser un filtre dont la réponse en fréquence est supérieure ou égale à 1-δ_1 dans la bande [0, f_1] et inférieure ou égale à δ_2 dans la bande $\left[f_2, \frac{f_e}{2}\right]$. Ainsi N a pour expression [19] :

$$N \geq \frac{\log\left(\frac{1}{\delta_2\sqrt{2\delta_1}}\right)}{\log(tg(\pi f_2 T)) - \log(tg(\pi f_1 T))} \tag{2.17}$$

Cette fonction de transfert, donnée par l'équation 2.16, est N fois dérivable à $w = 0$ ce qui la rend douce autour de 0 Hz.

Les fonctions de Chebychev de type I ou II sont déduites du polynôme de Chebychev $V_N(w) = \cos(N\cos(w))$. Ce polynôme force les pôles du filtre à être sur une ellipse. Ainsi les fonctions de Chebychev de type I et II sont données respectivement par [18] :

$$|F(w)|^2 = \frac{1}{1+\varepsilon^2 V_N^2\left(\frac{w}{ws}\right)} \tag{2.18}$$

$$|F(w)|^2 = \frac{1}{1+\left(\varepsilon^2 V_N^2\left(\frac{w}{ws}\right)^{-1}\right)} \tag{2.19}$$

La fonction elliptique, quand à elle, est définie selon les termes de la solution de la fonction elliptique jacobéenne, $U_N(w)$. Elle est donnée par [18] :

$$|F(w)|^2 = \frac{1}{1+\varepsilon^2 U_N^2\left(\frac{w}{ws}\right)^{-1}} \tag{2.20}$$

L'ordre N du filtre numérique correspondant se calcule aussi à partir de son gabarit et est donné par [19] :

$$N \cong 1.076\log\left(\frac{2}{\delta_2\sqrt{\delta_1}}\right)\log\left(\frac{f_e}{\Delta f}\frac{4}{\pi}\sin\left(2\pi\frac{f_1}{f_e}\right)\right) \tag{2.21}$$

où δ_1 représente l'ondulation dans la bande passante, δ_2 l'ondulation dans la bande coupée, f_1 la fréquence de coupure, f_2 la fréquence de début de bande affaiblie, $\Delta f = f_1 - f_2$ la bande de transition et f_e la fréquence d'échantillonnage.

En se basant sur ces différentes fonctions modèles, les conclusions suivantes peuvent être tirées selon les spécifications données :

Tableau I

Comparaison des différentes fonctions modèles

	Ordre du filtre	Caractéristiques De la bande passante	Caractéristiques De la bande coupée	Caractéristiques De la bande de transition
Petit	Elliptique			
Moyen	Chebychev			
Grand	Butterworth			
Équiripple		Elliptique Chebychev I	Elliptique ChebychevII	
Plate		Butterworth ChebychevII	Butterworth ChebychevI	
Étroite				Elliptique
Moyenne				Chebychev I et II
Large				Butterworth

Puisque c'est la réalisation d'un filtre qui fonctionne à une très grande fréquence d'échantillonnage qui nous intéresse, un aperçu sera donné uniquement sur les structures directes car ces dernières utilisent le minimum d'opérations élémentaires. Ceci les rend plus rapides comparées aux autres structures. Cependant, le prix à payer est la grande sensibilité aux erreurs d'arrondi dans le cadre d'un arithmétique à point fixe. Les structures directes de type I et II sont présentées aux figures 12 et 13.

La structure directe de type I calcule d'abord le dénominateur puis le numérateur. Ainsi elle utilise deux blocs de multiplicateur. Un bloc de multiplicateur pour les coefficients du dénominateur et un autre bloc de multiplicateur pour ceux du numérateur.

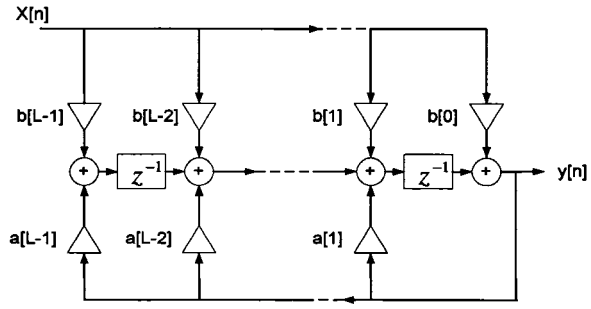


Figure 12 Structure directe de type I

La structure directe de type II n'est autre que la transposée de la structure directe de type I. Cependant, elle présente une particularité intéressante : chaque y[n] ou x[n] est multiplié par tous les coefficients ce qui fait que le numérateur et le dénominateur se calculent simultanément. Ceci simplifie beaucoup la mise en œuvre de la multiplication car au lieu de deux blocs de multiplicateur, cette structure n'en utilise qu'un seul. Cette structure directe de type II est plus rapide que celle de type I mais nécessite beaucoup plus de ressources que cette dernière.

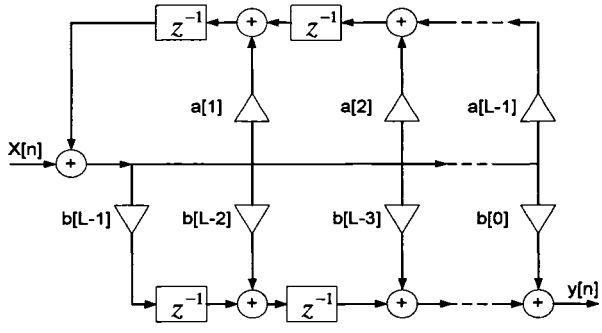


Figure 13 Structure directe de type II

2.3.2.3 Structures de réalisation des filtres RII décimateurs

Dans un processus de décimation, si le facteur de décimation se situe après le filtre passe bas, alors le résultat d'un certain nombre de calculs n'est pas utilisé à la sortie. Pour éviter ces calculs inutiles, on utilise des systèmes qui ne font des calculs que pour les sorties désirées. Les filtres, issus de ces systèmes, ne produisent pas de sortie pour chaque signal d'entrée. Ceci nous incite à croire que la décimation ne peut être effectuée que par des filtres non récursifs. Après tout, un filtre non récursif n'a pas besoin de ses sorties antérieur pour calculer la prochaine sortie, ce qui nous permet d'éliminer certaines sorties. Un filtre récursif, quand à lui, a besoin de ses sorties antérieures pour calculer la prochaine sortie. Donc, un filtre récursif, pour son propre profit, a besoin de toutes ses sorties même si le facteur de décimation ne les utilise pas toutes.

Les filtres récursifs peuvent, en effet, être utilisés comme des filtres décimateurs. La plupart des structures développées pour les filtres non récursifs décimateurs sont applicables aux filtres récursifs décimateurs. Cependant, le fait d'avoir accès directement à la réponse impulsionnelle des filtres non récursifs facilite leur décomposition polyphasée. Pour les filtres récursifs, la décomposition polyphasée est possible en utilisant d'autres techniques car l'accès à la réponse impulsionnelle n'est pas direct.

Martinez et Parks [20] ont proposé une classe de filtres récursifs à décimation qui regroupe les avantages des filtres RIF et ceux des filtres elliptiques. Ces filtres ont leurs zéros sur le cercle unité, des déviations dans la bande passante et dans la bande coupée. L'idée est d'avoir une fonction de transfert avec un dénominateur qui n'a que des puissances de z^R, R étant le facteur de décimation. Cette fonction de transfert est donnée par :

$$F(z)=\frac{\sum_{k=0}^{L} a[k]z^{-k}}{1+\sum_{k=1}^{K} b[k]z^{-R}}=\frac{N(z)}{D(z)} \tag{2.22}$$

Dans la bande passante $|F(z)|$ oscillera autour de 1 au moins $k+1$ fois. Dans la bande coupée, $|F(z)|$ oscillera autour de $\delta_s/2$ au moins $L+1$ fois. Quand la fonction de transfert de l'équation (2.22) est implémentée avec une structure de forme directe, la sortie $y(n)$ peut s'écrire comme suit :

$$\begin{aligned} y(n) &= -\sum_{k=1}^{K} b(k)\,y(n-kR) + a(L/2)x(n-L/2) \\ &\quad + \sum_{k=0}^{L/2-1} a(k)[x(n-k)+x(n-L+K)] \end{aligned} \tag{2.23}$$

L'équation (2.23) montre que cette structure nécessite $(K+1+L/2)$ multiplications par sortie. Mais, comme une sortie sur R est calculée, alors le nombre de multiplications par seconde est de $(K+1+L/2)/R$ mults/s. Le taux d'addition est de $(K+L)/R$ adds/s. Le nombre de coefficients à stocker est de $(K+1+L/2)$.

La question, maintenant, est de savoir comment concevoir un filtre récursif avec un dénominateur ne contenant que des puissances de z^R et comment déterminer leur ordre. Dans [21], un algorithme itératif pour le design de filtres récursifs à rebonds égalisés (*equiripple filter*) avec tous ses zéros sur le cercle unité est présenté. Il fonctionne comme suit : si δ_p (l'ondulation dans la bande passante), f_p (La fréquence de coupure), f_c (début de la bande coupée) et l'ordre du filtre sont donnés, alors la valeur de δ_s (l'ondulation dans la bande coupée) est choisie de telle sorte que le filtre ait le comportement d'un filtre à rebonds égalisés. Un filtre récursif avec un dénominateur ne

contenant que des puissances de z^R peut être conçu en utilisant cet algorithme. La seule différence dans l'algorithme est que le module au carré du dénominateur est de la forme :

$$D\left(e^{j2\pi f}\right)D\left(e^{-j2\pi f}\right)=\sum_{k=0}^{K}\tilde{b}(k)\cos(2\pi Rkf) \tag{2.24}$$

Si le numérateur $N\left(e^{j2\pi f}\right)$ contrôle l'ondulation dans la bande coupée et est égale à 1 à $f=0$, alors une valeur $M_1 \approx L/2$ peut être donnée par :

$$M_1=\frac{\cosh^{-1}(1/\delta_s)}{\cosh^{-1}(X_0)} \tag{2.25}$$

avec $X_0=(3-\cos 2\pi f_c)/(1+\cos 2\pi f_c)$. Si le numérateur $N\left(e^{j2\pi f}\right)$ contrôle l'ondulation dans la bande coupée mais a la valeur 1 à $f=f_p$, alors une valeur $M_2 \approx L/2$ peut être donnée par :

$$M_2=\frac{\cosh^{-1}(1/\delta_s)}{\cosh^{-1}(X_p)} \tag{2.26}$$

avec

$$X_p=\left[\frac{X_0+1}{2}\right]\cos(2\pi f_p)+\left[\frac{X_0-1}{2}\right]$$

Une approximation de L, l'ordre du numérateur, peut être obtenue comme suit :

$$L_0=M_1+M_2 \tag{2.27}$$

Pour ce qui concerne l'ordre K du dénominateur, il doit être choisi de façon à avoir un grand nombre de pôles pour respecter l'ondulation désirée dans la bande passante. Un bon point de départ est de choisir $L = L_0$ et $K = 2$ ou $K = 4$ et de jouer avec les valeurs de L et K pour contrôler respectivement l'ondulation dans la bande arrêtée et celle dans la bande passante. Le tableau II présente, pour les mêmes spécifications, une comparaison entre un filtre FIR décimateur, un filtre elliptique décimateur et le filtre RII avec dénominateur à puissance de z^R.

Tableau II

Comparaison entre FIR décimateur, elliptique décimateur et RII avec dénominateur à puissance de z^R

	Facteur de décimation	Ordre du filtre	Multiplication par seconde	Nombre de coefficients
FIR (un étage de décimation)	$D = 20$	$M = 652$	16.4	327
Elliptique (un étage de décimation)	$D = 20$	$N = 7$	11	11
RII à puissance z^R (un étage de décimation)	$D = 20$	$L = 116$ $K = 4$	3.15	63
FIR (deux étages de décimation)	$D_1 = 10$ $D_2 = 2$	$M_1 = 34$ $M_2 = 72$	3.65	55
Elliptique (deux étages de décimation)	$D_1 = 10$ $D_2 = 2$	$N_1 = 3$ $N_2 = 7$	6.1	16
RII à puissance z^R (deux étages de décimation)	$D_1 = 10$ $D_2 = 2$	$L_1 = 26 \quad K_1 = 1$ $L_2 = 10 \quad K_2 = 6$	2.1	27

Le tableau II montre le gain du filtre RII avec dénominateur à puissance de z^R par rapport aux autres structures. Donc il est le plus intéressant pour des applications à hautes fréquences d'échantillonnages. Cependant, sa structure telle que définie, ne nous permet pas de filtrer un signal de l'ordre des GHz. La question qui se pose à ce stade est : est-ce possible d'utiliser la technique pipeline dans la structure du RII avec dénominateur à puissance de z^R . En fait, dans n'importe quelle architecture de filtre, la section en amont peut être en pipeline pour augmenter son débit en rajoutant des registres ce qui entraîne une augmentation de la latence. La technique pipeline ne peut pas, en général, être utilisée dans la boucle de rétroaction parcequ'elle change le délai de cette dernière. Ceci, en conséquence, change la fonctionnalité de la boucle de rétroaction. 'Scattered lookahead' est une technique qui transforme un filtre à pôle avec une boucle de rétroaction de délai K en un filtre à pole et zéro avec une boucle de rétroaction dont le délai est un multiple de K. Ceci permet d'utiliser la technique pipeline, pour les additionneurs et les multiplicateurs dans la boucle de rétroaction, par K. La boucle de rétroaction désirée est générée en prenant chaque pôle du filtre original et générer $K-1$ pôles/zéros qui ont la même module que le pôle original et localisés à $z_n = pe^{j\pi n/K}$. Les $K-1$ pôles ajoutés seront annulés par les zéros de la section en amont du filtre. L'exemple suivant illustre bien cette technique.

Soit un filtre à pôles d'ordre 2 avec $p_1 = 0.5$ et $p_2 = 0.25$. La fonction de transfert est donnée par :

$$F(z) = \frac{1}{1 - 3/4z^{-1} + 1/8z^{-2}}$$

La technique 'Scattered lookahead' introduit deux étages de pipelines additionnels en ajoutant une paire de pôles/zéros à $0.25e^{\pm j2\pi/3}$ et $0.5e^{\pm j2\pi/3}$. Cette addition de pôles/zéros à cette location transforme la fonction de transfert comme suit :

$$F(z) = \frac{1}{1+0.5z^{-1}+0.25z^{-2}} \times \frac{\left(1+0.5z^{-1}+0.25z^{-2}\right)\left(1+0.75z^{-1}+0.5625z^{-2}\right)}{\left(1+0.75z^{-1}+0.5625z^{-2}\right)\left(1-0.75z^{-1}+0.125z^{-2}\right)}$$

$$= \frac{1+1.25z^{-1}+1.1875z^{-2}+0.4687z^{-3}+0.1406z^{-4}}{1-0.5469z^{-3}+0.0527z^{-6}}$$

$$= \frac{512+640z^{-1}+608z^{-2}+240z^{-3}}{512-280z^{-3}+27z^{-6}}$$

Cette technique permet d'augmenter le débit du filtre mais son point faible est qu'elle augmente la complexité de la section en amont du filtre.

Les filtres RII avec dénominateur à puissance de z^R sont très intéressants mais le prix de leur pipeline et la grande largeur des coefficients les rendent très limités pour des applications à hautes fréquences d'échantillonnage.

2.3.3 Filtres CIC décimateurs

Généralement, dans un filtre numérique, le nombre de multiplication à faire par seconde est exprimé par $M_r = Kf_e$ où f_e représente la cadence à laquelle se font les calculs donc la fréquence d'échantillonnage. Le paramètre K dépend du type de filtre, de son ordre et de ses performances. Dans ce contexte, Eugène B Hogenauer [7] a présenté en 1981 des filtres particulièrement efficaces. Ce sont les filtres Cascaded Integrator and Comb (CIC). L'absence de multiplicateur, le non stockage des coefficients du filtre et la simplicité d'implémentation permettent aux filtres CIC de jouer un rôle très important dans les applications exigeant de hautes fréquences d'échantillonnage comme les communication sans fil où les signaux sont échantillonnées directement en RF ou IF. Dans les applications à bande étroite comme la radio mobile, des taux de décimation de l'ordre de 1000 sont souvent requis. Une autre application des filtres CIC est dans le domaine des convertisseurs ΣΔ [22].

2.3.3.1 Description générale

Les filtres CIC [7] sont des filtres utilisés dans le domaine de traitement multicadence du signal. Ils sont utilisés pour augmenter (interpolation) ou diminuer (décimation) le rapport d'échantillonnage. Ils se basent sur le fait qu'une élimination parfaite de pole/zéro peut être obtenue. Ceci est possible en utilisant seulement une arithmétique exacte comme le complément à deux. Une attention particulière sera accordée aux filtres CIC décimateurs.

La structure de base des filtres CIC décimateurs (CIC d'ordre 1) [7] indiquée à la figure 14 montre que ces filtres sont constitués essentiellement de deux blocs en cascade. Le premier bloc est composé d'un étage d'intégrateur numérique et le second d'un étage de différentiateur. Ces deux blocs sont séparés par un facteur de décimation R. La figure 15 montre une structure équivalente à la structure de base, obtenue à l'aide des identités de Nobles.

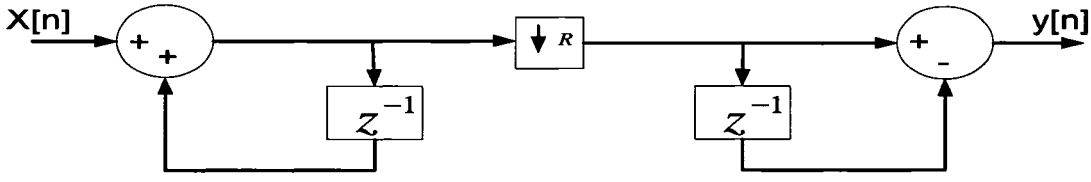


Figure 14 Structure d'un filtre CIC d'ordre 1

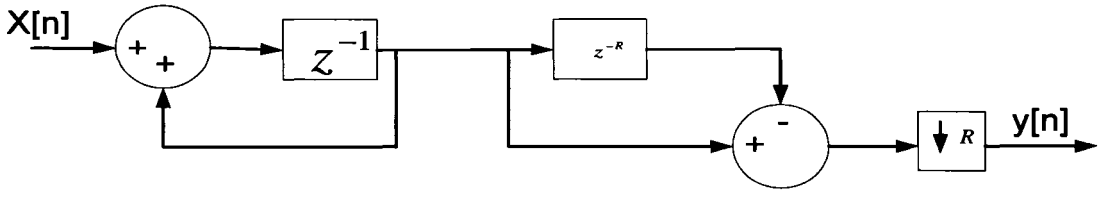


Figure 15 Structure équivalente à la structure d'un filtre CIC d'ordre 1

Un intégrateur est réalisé par un simple filtre à un seul pôle avec une contre réaction unitaire. Les intégrateurs des filtres CIC opèrent à la fréquence d'échantillonnage f_s. Ils peuvent être considérés comme des accumulateurs. La figure 16 représente un simple intégrateur.

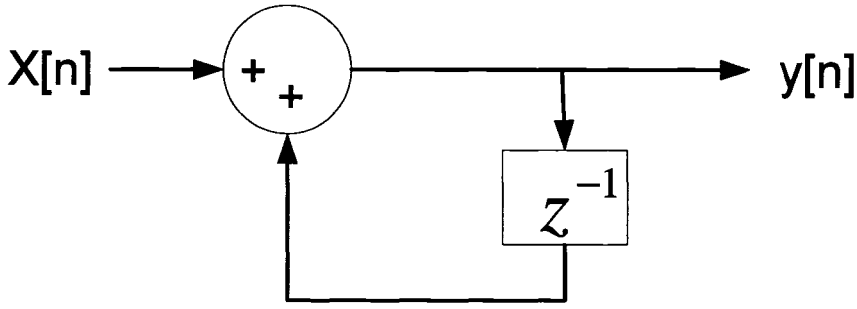


Figure 16 Schéma d'un simple intégrateur

L'équation aux différences de l'intégrateur numérique est donnée par :

$$y(n) = y(n-1) + x(n) \tag{2.28}$$

ce qui donne une fonction de transfert sous la forme suivante :

$$H_I(Z) = \frac{1}{1 - Z^{-1}} \tag{2.29}$$

Les différentiateurs des filtres CIC, quand à eux, fonctionnent à une fréquence beaucoup plus basse f_e et qui est égale à $\frac{f_s}{R}$ avec R le facteur de décimation. Les différentiateurs utilisent un paramètre de délai différentiel noté M et qui sert à contrôler la position des zéros dans la réponse fréquentielle du filtre. La figure 17 représente un différentiateur.

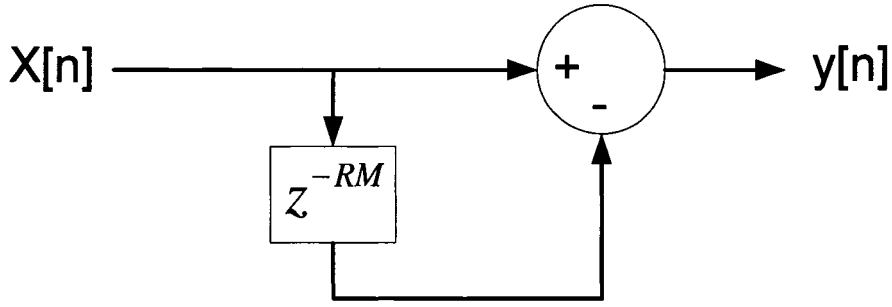


Figure 17 Schéma d'un simple différentiateur

L'équation aux différences du différentiateur est donnée par :

$$y(n) = x(n) - x(n - RM) \tag{2.30}$$

ce qui donne une fonction de transfert sous la forme suivante :

$$H_C(Z) = 1 - Z^{-RM} \tag{2.31}$$

avec R le facteur de décimation et M le délai différentiel.

Le filtre CIC d'ordre N est constitué alors de N intégrateurs en cascade qui fonctionnent à la fréquence d'échantillonnage f_s, suivi d'un facteur de décimation R qui réduit la fréquence de f_s à $\frac{f_s}{R}$ puis de N différentiateurs en cascade qui fonctionnent à la fréquence $\frac{f_s}{R}$. La figure 18 présente la structure d'un filtre CIC d'ordre N.

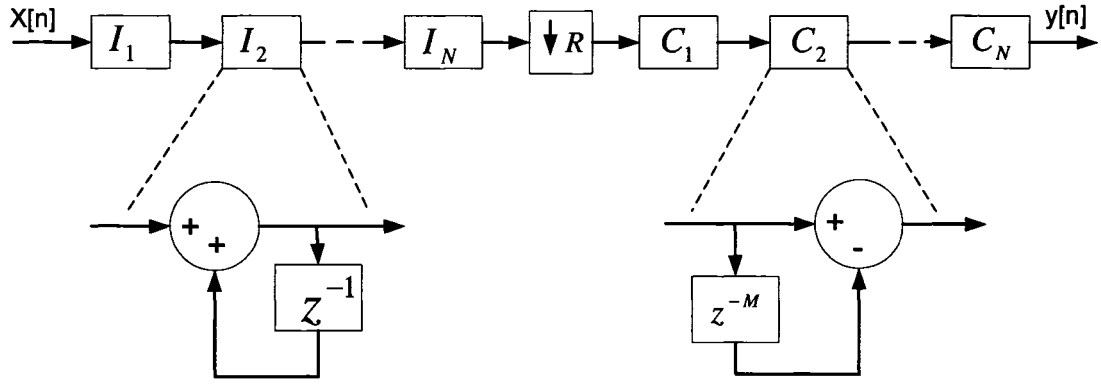


Figure 18 Structure d'un filtre CIC décimateur d'ordre N

La fonction de transfert du CIC décimateur d'ordre N est donnée par le produit de la fonction de transfert de l'intégrateur (2.29) par la fonction de transfert du différentiateur (2.31) :

$$H(Z) = H_I(Z) \times H_C(Z) = \left(\frac{1}{1-Z^{-1}}\right)^N \times \left(1-Z^{-RM}\right)^N \tag{2.32}$$

$$H(z) = \left(\frac{1-Z^{-RM}}{1-Z^{-1}}\right)^N$$

La division euclidienne du numérateur par le dénominateur de l'équation (2.32) donne

$$\frac{1-Z^{-RM}}{1-Z^{-1}} = 1 + Z^{-1} + Z^{-2} + Z^{-3} + \ldots\ldots\ldots\ldots + Z^{-RM+1} \tag{2.33}$$

Donc la fonction de transfert de l'équation (2.32) peut s'écrire sous la forme suivante :

$$H(z)=\left[\sum_{K=0}^{RM-1}Z^{-K}\right]^{N} \tag{2.34}$$

L'équation (2.34) montre qu'un filtre CIC d'ordre N n'est rien d'autre que N filtres FIR en cascade. L'intérêt de cette équivalence réside dans le fait que le filtre est stable et la phase est linéaire ce qui entraîne un délai de groupe constant. Ces propriétés sont primordiales pour le filtrage.

2.3.3.2 Réalisation des filtres CIC décimateurs

Le développement spectaculaire de la microélectronique a rendu les FPGA ``Field Programmable Gate Array`` très populaire dans le domaine de traitement du signal. Cependant, la vitesse des opérations élémentaires de ces derniers est encore limitée à environ 500 MHz ce qui rend le filtrage à hautes fréquences d'échantillonnage difficile. La structure régulière du filtre CIC décimateur ne permet pas de filtrer un signal échantillonné avec une fréquence de l'ordre des GHz. Il faut donc trouver une architecture de réalisation appropriée qui utilise les filtres CIC décimateurs et qui permettra aux FPGA d'implémenter un filtre à haute vitesse. Différentes architectures sont présentées dans les sections suivantes.

- Structure pipeline

La structure pipeline est l'une des premières structures utilisée pour augmenter la vitesse des filtres CIC [7]. Cette technique consiste à couper les longues chaînes d'additionneurs en insérant des registres (P0, P1, P2, ...) entre les additionneurs. Ceci donne un gain en vitesse mais augmente le temps de chargement de toutes les bascules du circuit (la latence). La figure 19 montre la structure pipeline d'un filtre CIC décimateur d'ordre 2.

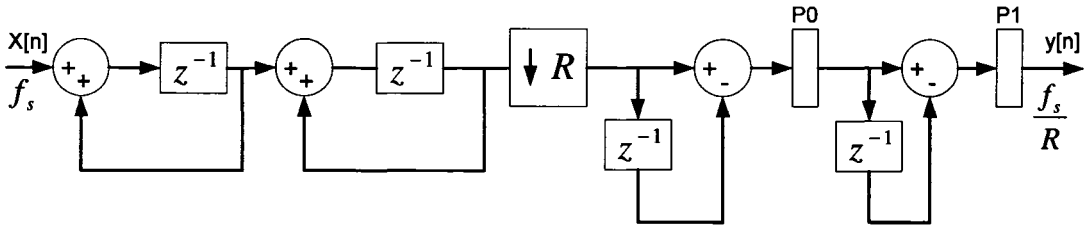


Figure 19 Structure pipeline d'un CIC décimateur d'ordre 2

Cette structure est très facile à implémenter mais reste limitée en terme de rapidité. Ceci fait que son utilisation pour des applications à haute vitesse devient impossible.

- Structure parallèle

Le traitement parallèle a pour but d'augmenter la rapidité d'exécution d'un processus. En traitement numérique du signal, le traitement parallèle est la technique qui permet d'atteindre la plus grande vitesse d'exécution pour les circuits arithmétiques tels que les multiplieurs et les additionneurs [23].

Le temps nécessaire à l'exécution d'une addition parallèle dépend du temps de propagation de la retenue. Tous les bits des opérations sont transmis simultanément aux opérateurs parallèles ce qui réduit le temps de propagation de la retenue d'où une augmentation de la vitesse de calcul. Le résultat reste disponible après la détection et la correction du dépassement.

La première approche, pour paralléliser la structure régulière du filtre CIC décimateur, est de considérer des entrées parallèles de l'intégrateur [19]. La figure 20 montre le concept de parallélisme.

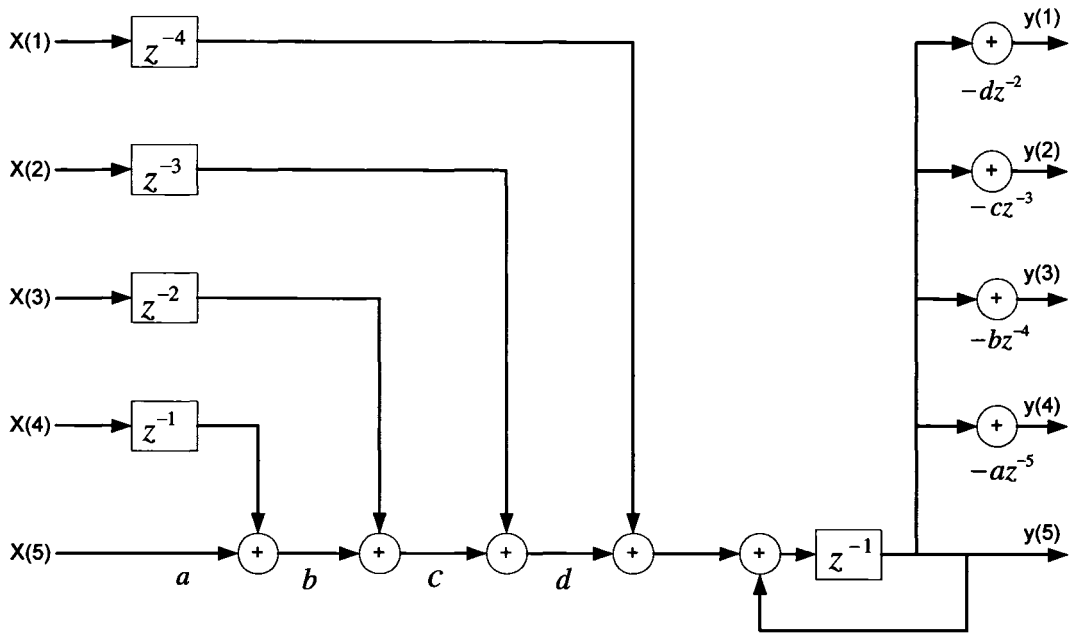


Figure 20 Structure parallèle de l'intégrateur du CIC décimateur

En fait, les 5 entrées X (1), X (2), ... X (5) sont générés simultanément et additionnées une à une de façon à obtenir une somme totale qui sera ensuite intégrée par l'intégrateur parallèle. Afin de retrouver les images des entrées, il faut soustraire de la sortie de l'intégrateur parallèle, les valeurs X (1), X (2), ... X (5). Des accumulateurs sont utilisés pour le stockage des résultats intermédiaires. Ensuite vient le facteur de décimation et le différentiateur.

En regardant la structure parallèle de la figure 20, la première chose qui apparaît est la longue chaîne d'additionneurs qui est étroitement lié aux nombres d'entrées parallèles. Ceci limite la vitesse d'exécution.

- Structure parallèle améliorée

Cette structure permet de couper la longue chaîne d'additionneurs en utilisant des additionneurs en arbre au lieu qu'ils soient en série comme le montre la figure 21 [19]. Cette structure réduit nettement la latence. Elle passe de P+2 à $\log_2(P)$, P étant le nombre d'entrées parallèles.

La complexité d'une architecture de traitement numérique de signal est définie comme étant la quantité de matériel nécessaire pour réaliser cette architecture sous un ensemble de contraintes données. Pour la structure parallèle et la structure parallèle améliorée, les paramètres principaux de complexité sont le nombre d'additionneurs et la taille des mémoires.

Dans le cadre de l'architecture parallèle, l'addition se fait à un résultat précédent disponible dans un registre. La mise en mémoire de ce résultat est ensuite effectuée. Ce qui fait que, si le nombre d'entrées parallèles est P alors la complexité est donnée par :

$$C = P(P+2)\times b \tag{2.35}$$

b représente la largeur binaire des entrées.

Quand à la structure améliorée présentée à la figure 15, la complexité est donnée par :

$$C = P\log_2(P+2)\times b \tag{2.36}$$

Les équations (2.35) et (2.36) montrent bien que la complexité de la structure parallèle améliorée est nettement réduite ce qui engendre automatiquement une amélioration de la latence.

Cependant, cette structure montre que son implémentation occuperait une surface CMOS assez grande. Cette surface augmenterait en fonction du nombre de branches

parallèles. L'occupation d'une grande surface CMOS entraîne une consommation de puissance importante. Cette structure réduit la complexité mais n'est pas une solution économique. Donc, il faut se retourner vers d'autres techniques beaucoup plus intéressantes et économiques.

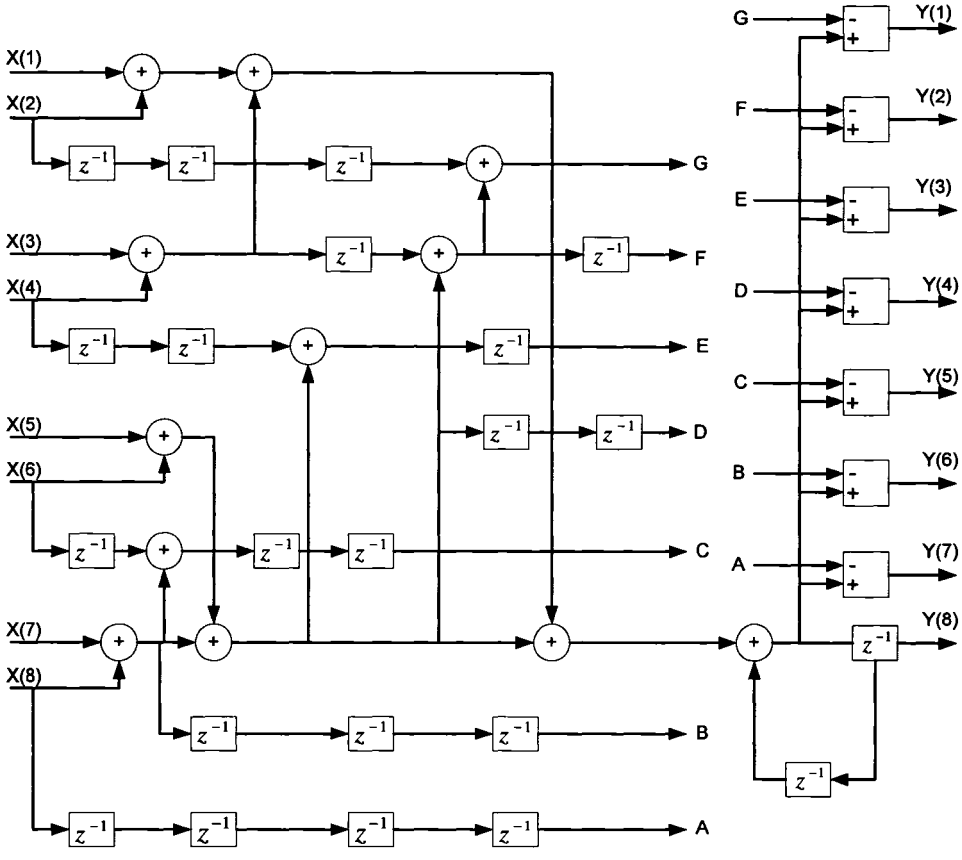


Figure 21 Structure parallélisée améliorée avec 8 entrées

➢ Structure parallèle utilisant la décomposition polyphasée

La fonction de transfert d'un filtre CIC d'ordre N avec un facteur de décimation R est donnée par l'équation (2.32) et s'écrit comme suit :

$$H(z)=\left(\frac{1-Z^{-RM}}{1-Z^{-1}}\right)^{N}$$

Cette fonction de transfert peut être décomposée en deux fonctions de transfert [19]. La première fonction de transfert est d'ordre N_1 avec un facteur de décimation R_1 et la deuxième est d'ordre N_2 avec un facteur de décimation R_2. Cependant, pour avoir le même taux de décimation, R doit absolument être le produit de R_1 par R_2. Ainsi, la fonction de transfert peut s'écrire comme suit :

$$H_P(z)=\left(\frac{1-z^{-R_1}}{1-z^{-1}}\right)^{N_1}\left(\frac{1-z^{-RM}}{1-z^{-R_1}}\right)^{N_2}=H_1(z)\times H_2(z^{R_1}) \tag{2.37}$$

avec

$$H_1(z)=\left(\frac{1-z^{-R_1}}{1-z^{-1}}\right)^{N_1} \tag{2.38}$$

$$H_2(z)=\left(\frac{1-z^{-R_2M}}{1-z^{-1}}\right)^{N_2} \tag{2.39}$$

Étant donné qu'un filtre CIC d'ordre N n'est rien d'autre que N filtres FIR en cascade, alors en utilisant l'équation (2.34), l'équation (2.38) peut s'écrire comme suit :

$$H_1(z)=\left(1+z^{-1}+z^{-2}+.....+z^{-(R_1-1)}\right)^{N_1} \tag{2.40}$$

L'équation (2.40) montre que $H_1(z)$ équivaut à N_1 filtres FIR en cascade. Ainsi, une partie de la fonction de transfert du filtre CIC est transformée en un filtre FIR. La fonction de transfert $H_1(z)$, qui est un filtre FIR, peut être décomposée en structure polyphasée et donne :

$$H_1(z) = \sum_{i=0}^{R_1-1} z^{-i} E_i(z^{R_1}) \tag{2.41}$$

$E_i(z)$ représente la composante polyphasée avec i le nombre de phases ou de branches parallèles. En d'autres termes, i est le nombre d'entrées parallèles. Le choix de N_1 et de R_1 doit être judicieux. Le tableau III présente les valeurs des coefficients des composantes polyphasées pour différents valeurs de N_1 et de R_1.

Tableau III

Coefficients des composantes polyphasées

N_1	2			3		
R_1	2	4	8	2	4	8
$E_0(z)$	(1, 1)	(1, 3)	(1, 7)	(1, 3)	(1, 12, 3)	(1, 42, 21)
$E_1(z)$	(2, 0)	(2, 2)	(2, 6)	(3, 1)	(3, 12, 1)	(3, 46, 15)
$E_2(z)$		(3, 1)	(3, 5)		(6, 10, 0)	(6, 48, 10)
$E_3(z)$		(4, 0)	(4, 4)		(10, 6, 0)	(10, 48, 6)
$E_4(z)$			(5, 3)			(15, 46, 3)
$E_5(z)$			(6, 2)			(21, 42, 1)
$E_6(z)$			(7, 1)			(28, 36, 0)
$E_7(z)$			(8, 0)			(36, 28, 0)

Le tableau III montre que tous les coefficients sont des entiers. Ceci est très important du point de vu réalisation car les composantes polyphasées ne nécessitent pas de multiplications. La figure 22 montre un exemple de réalisation pour $E_0(z) = 1 + 3z^{-1}$.

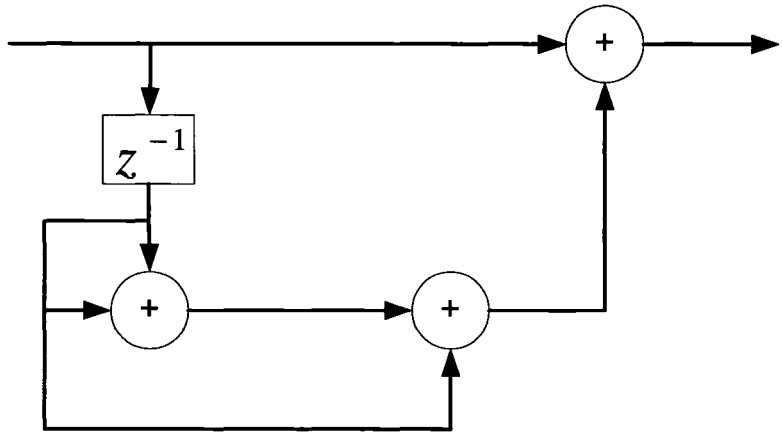


Figure 22 Exemple de réalisation pour $E_0(z) = 1 + 3z^{-1}$

La structure parallèle utilisant la décomposition polyphasée présente beaucoup d'avantages par rapport aux autres structures. Elle ne nécessite pas de multiplications ni de stockage des coefficients. Sa structure est compacte ce qui permet de varier le facteur de décimation sans changer la structure globale. Elle effectue une décimation multiétage qui permet de réduire considérablement la complexité par rapport à une décimation à un seul étage. Elle est aussi facile à implémenter. Ainsi, la structure parallèle du CIC utilisant la décomposition polyphasée est l'architecture retenue pour l'implémentation du filtre décimateur et est présentée à la figure 23.

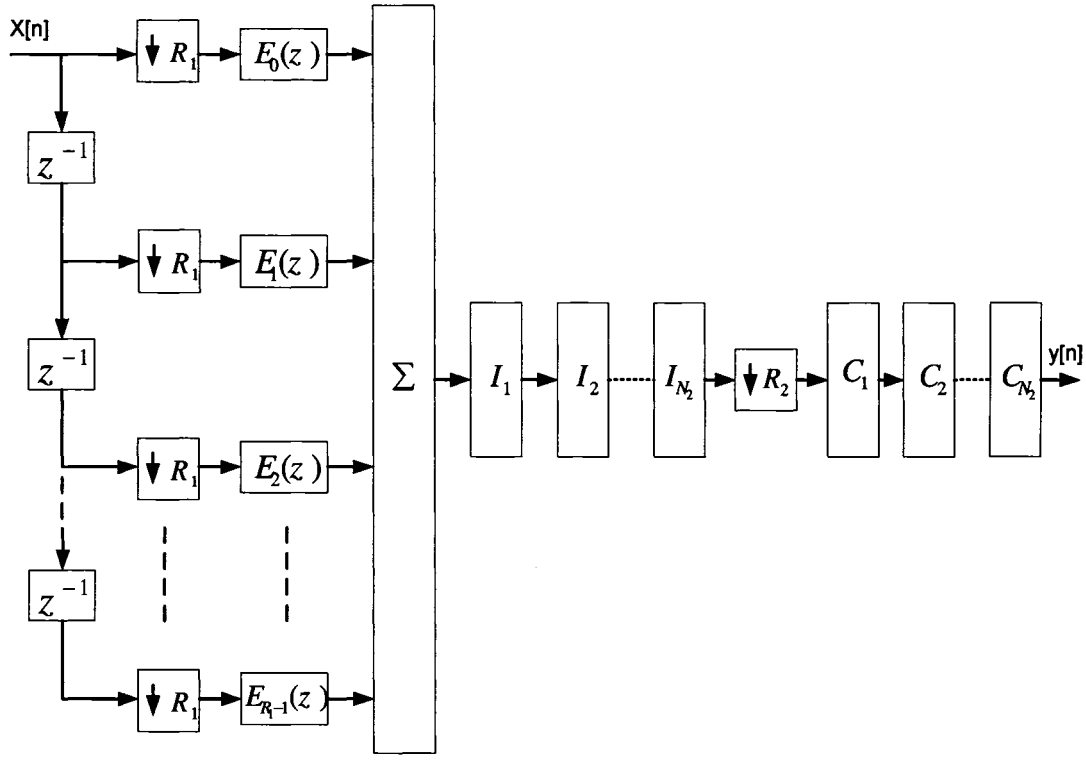


Figure 23 Structure parallèle du CIC utilisant la décomposition polyphasée

La structure du CIC polyphasé est composée de deux étages de décimation. Le premier étage de décimation est un filtre FIR qui fonctionne à la fréquence d'échantillonnage divisée par le nombre de branches parallèles. Le deuxième étage de décimation est un filtre CIC régulier dont les intégrateurs fonctionnent à la même fréquence que le filtre FIR polyphasé et les décimateurs, quand à eux, fonctionnent à la fréquence des intégrateurs divisée par le deuxième facteur de décimation R_2. Cette structure est très intéressante pour les applications qui utilisent de hautes fréquences d'échantillonnage car elle est facile à implémenter et est capable de filtrer un signal échantillonné à la fréquence $f_s = f_{filtre} \times R_1$. Ceci veut dire que si la fréquence du filtre est f_{filtre} = 150 MHz et le nombre de branches parallèles est R_1 = 8, alors le filtre peut filtrer un signal échantillonné à la fréquence f_s = 150 MHz * 8 = 1.2 GHz.

2.4 Conclusion

Ce chapitre à été consacré aux spécifications du noyau et à la recherche d'une architecture qui répond le mieux aux spécifications désirées.

Dans un premier temps, une étude a été accomplie sur les filtres numériques à réponse impulsionnelle finie (RIF). La théorie de base des ces filtres et différentes structures de réalisation ont été présentées. Ces dernières sont assez intéressantes mais sont très limitées en vitesse. De ce fait, une autre approche a été présentée. Elle consiste à utiliser une décomposition polyphasée qui permet de paralléliser l'entrée. Cette technique augmente considérablement le débit du filtre. Malgré cette augmentation de débit, le filtre reste toujours limité pour des applications en hautes fréquences.

Dans un deuxième temps, les filtres numériques à réponse impulsionnelle infinie (RII) sont présentés. La théorie de base des ces filtres et différentes structures de réalisation ont été présentées. Ces dernières sont plus rapides que celles des filtres RIF mais sont aussi limitées pour des applications à hautes fréquences.. De ce fait, une autre approche a été présentée. Elle consiste à utiliser des filtres numériques à réponse impulsionnelle infinie avec un dénominateur ne contenant que des puissances de z^R. Les inconvénients de cette approche sont la grande sensibilité des coefficients et l'augmentation de la complexité de la section en amont du filtre. Ainsi, l'ensemble des techniques décrites utilisant les filtres RII ne permet pas de répondre adéquatement aux spécifications très exigeantes requises pour le filtrage à haute fréquence.

Les filtres CIC ont été présentés en dernier. La structure de base des filtres CIC est en pipeline. L'utilisation de cette structure en haute fréquence est impossible. Par conséquent, comme le parallélisme est le meilleur moyen d'augmenter la vitesse de traitement, il fallait trouver une architecture parallèle. Ainsi, trois architectures parallèles ont été présentées. La première architecture est une structure parallèle qui présente une

longue chaîne d'additionneur qui augmente au fur et à mesure que le nombre d'entrées parallèles augmente. L'amélioration de cette structure parallèle constitue la deuxième architecture. Cette dernière occupe une très grande surface CMOS. L'occupation d'une grande surface CMOS entraîne une consommation de puissance importante. En fin, une structure parallèle utilisant la décomposition polyphasée a été présentée. Elle est avantageuse car en plus de la simplicité d'implémentation, elle permet d'atteindre des performances que les architectures classiques n'offrent pas. De ce fait, c'est la structure retenue pour la réalisation du noyau programmable.

CHAPITRE 3

PARAMÈTRES DE CONCEPTION DES FILTRES CIC DÉCIMATEURS

3.1 Introduction

Comme indiqué dans les chapitres précédents, la structure des filtres CIC a été présentée en 1981 par Hogenauer [7]. Actuellement, des récepteurs numériques commerciaux utilisent cette structure. Cependant, l'implémentation des filtres CIC décimateurs demande l'analyse de plusieurs aspects. Ainsi, dans ce chapitre, la structure du filtre CIC décimateur et les différents aspects à prendre en considération durant l'implémentation sont présentés en détail.

3.2 Description et analyse mathématique du filtre CIC décimateur

La structure des filtres CIC décimateurs telle que proposée par Hogenauer est présentée à la figure 24. Le filtre est constitué de N intégrateurs et de N différentiateurs séparés par un facteur de décimation R. Les données arrivent à la fréquence d'échantillonnage f_s et sont traitées par les intégrateurs. Ceci veut dire que les intégrateurs fonctionnent à la fréquence f_s. Le facteur de décimation R permet de choisir une sortie sur R du dernier intégrateur qui sera traitée par les différentiateurs. Ceci veut dire que les différentiateurs fonctionnent à la fréquence $\frac{f_s}{R}$. Donc, la fréquence d'échantillonnage à la sortie du filtre CIC décimateur est réduite et est égale à $\frac{f_s}{R}$.

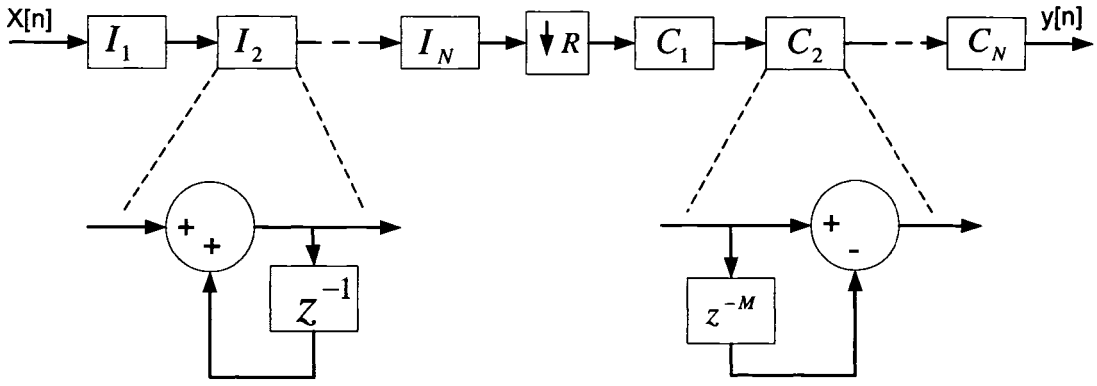


Figure 24 Structure du filtre CIC décimateur d'ordre N

La fonction de transfert de cette structure est donnée par :

$$H(z)=\left(\frac{1-Z^{-RM}}{1-Z^{-1}}\right)^{N} \tag{3.1}$$

Cette implémentation utilise $2N$ additionneurs et $4N+2$ registres : il y a un additionneur dans chaque étage, deux registres sont donc utilisés pour les deux entrées de chaque additionneur et deux registres additionnels pour les sorties de l'intégrateur et du différentiateur. Ce nombre de calcul est très minime comparé aux autres structures de réalisation de filtre à décimation.

3.3 Analyse fréquentielle du filtre CIC décimateur

Analysons le filtre CIC décimateur dans le domaine fréquentiel. Posons $x_1(n)$ la sortie du N^{ieme} intégrateur et $x_2(n)$ la sortie de la décimation c'est-à-dire l'entrée du premier différentiateur. Si la transformée de Fourier de $x\ (n)$ est $X(w)$, alors $x_1(n)$ dans le domaine fréquentiel est donné par :

$$X_1(w) = \left(\frac{1}{1-e^{-jw}}\right)^N .X(w) \tag{3.2}$$

$x_2(n)$ est donné par

$$X_2(w) = \frac{1}{R}\sum_{k=0}^{R-1} X_1\left(\frac{w-k.2\pi}{R}\right)$$

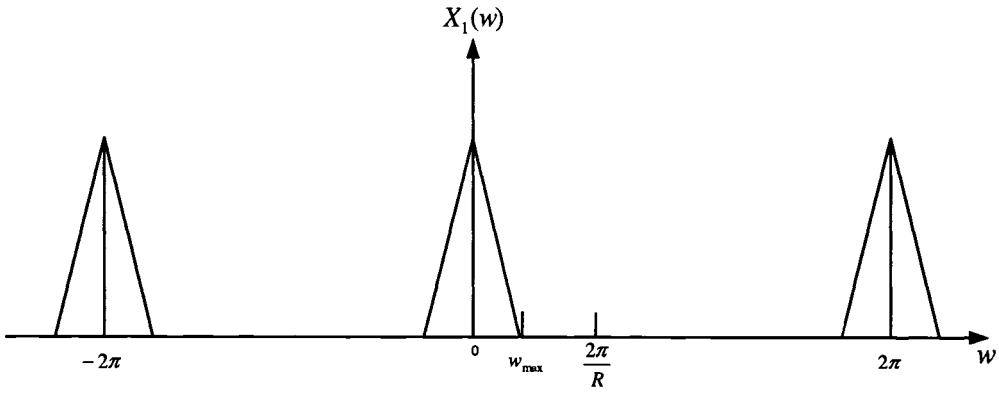
$$= \frac{1}{R}\sum_{k=0}^{R-1}\left[\left(\frac{1}{1-e^{-j\left(\frac{w-k.2\pi}{R}\right)}}\right)^N .X\left(\frac{w-k.2\pi}{R}\right)\right]$$

$$= \frac{1}{R}\left\{\left[\frac{1}{1-e^{-j\left(\frac{w}{R}\right)}}\right]^N .X\left(\frac{w}{R}\right)+\left[\frac{1}{1-e^{-j\left(\frac{w-2\pi}{R}\right)}}\right]^N .X\left(\frac{w-2\pi}{R}\right)+.....\right\} \tag{3.3}$$

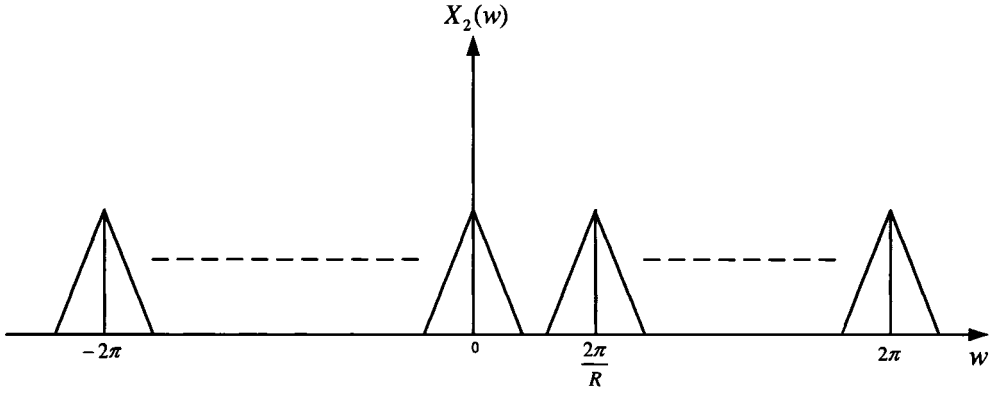
Si $X(w)=0$ quand $w \succ \frac{\pi}{R}$ ou $2w_{\max} \prec \frac{2\pi}{R}$, $w \in \left[-\frac{\pi}{R},\frac{\pi}{R}\right]$, alors l'équation (3.3) peut se simplifier pour s'écrire comme suit :

$$X_2(w) = \frac{1}{R}\left(\frac{1}{1-e^{-j\frac{w}{R}}}\right)^N .X\left(\frac{w}{R}\right) \tag{3.4}$$

L'équation (3.4) montre qu'à part le premier terme, les autres termes ne contribuent pas dans le calcul de $X_2(w)$ dans l'intervalle $\left[-\frac{\pi}{R},\frac{\pi}{R}\right]$. Ceci est bien illustré par la figure 25.



(a) Spectre du signal d'entrée



(b) Spectre du signal de sortie après décimation

Figure 25 Comparaison du spectre du signal avant et après décimation

La figure 25 montre qu'il est important, dans un filtre à décimation, d'éliminer les composantes hautes fréquences du signal d'entrée pour éviter qu'il y ait des chevauchements. Ceci veut dire qu'il faut que la fréquence maximale w_{max} du signal d'entrée soit inférieure à $\frac{\pi}{R}$ (théorème de Nyquist). Dans un filtre CIC décimateur, cette élimination de composantes en hautes fréquences du signal d'entrée est réalisée par les intégrateurs et plus leur nombre augmente plus l'atténuation est grande. Si le nombre

d'intégrateur N est assez grand, alors la simplification de l'équation (3.3) est applicable dans le cadre des filtres CIC. Cependant, les intégrateurs du filtre CIC ne sont pas stables d'où l'amplification des composantes en basse fréquence. Mais, comme un filtre CIC n'est rien d'autres que des filtres FIR en cascade, alors il est stable. Ceci est expliqué par le fait que l'étage de différentiateur ait une structure de filtre passe haut ce qui lui permet d'atténuer les composantes en basse fréquence d'où la stabilité du filtre CIC [7].

Calculons maintenant la réponse en fréquence du filtre CIC au complet :

$$Y(w) = \left(1 - e^{-jw}\right)^N . X_2(w)$$

$$= \frac{1}{R} \frac{\left(1 - e^{-jw}\right)^N}{\left(1 - e^{-j\frac{w}{R}}\right)^N} . X\left(\frac{w}{R}\right)$$

$$= \frac{1}{R} \left[\frac{e^{-j\frac{w}{2}}}{e^{-j\frac{w}{2R}}} . \frac{\left(e^{j\frac{w}{2}} - e^{-j\frac{w}{2}}\right)}{\left(e^{j\frac{w}{2R}} - e^{-j\frac{w}{2R}}\right)} \right]^N . X\left(\frac{w}{R}\right)$$

$$= \frac{1}{R} . \left(\frac{\sin\frac{w}{2}}{\sin\frac{w}{2R}} . e^{-jw\left(\frac{1}{2} - \frac{1}{2R}\right)} \right)^N . X\left(\frac{w}{R}\right)$$

$$= \frac{1}{R} . \left(\frac{\sin\frac{w}{2}}{\sin\frac{w}{2R}} \right)^N . e^{-jN\left(\frac{1}{2} - \frac{1}{2R}\right)w} . X\left(\frac{w}{R}\right) \qquad (3.5)$$

L'équation (3.5) montre bien la réponse en amplitude et en phase du CIC au complet. En augmentant l'ordre du filtre N, la réponse en fréquence de l'amplitude montre que l'atténuation dans la bande passante augmente. La réponse en amplitude d'un filtre CIC avec différents ordres est présentée à la figure 26.

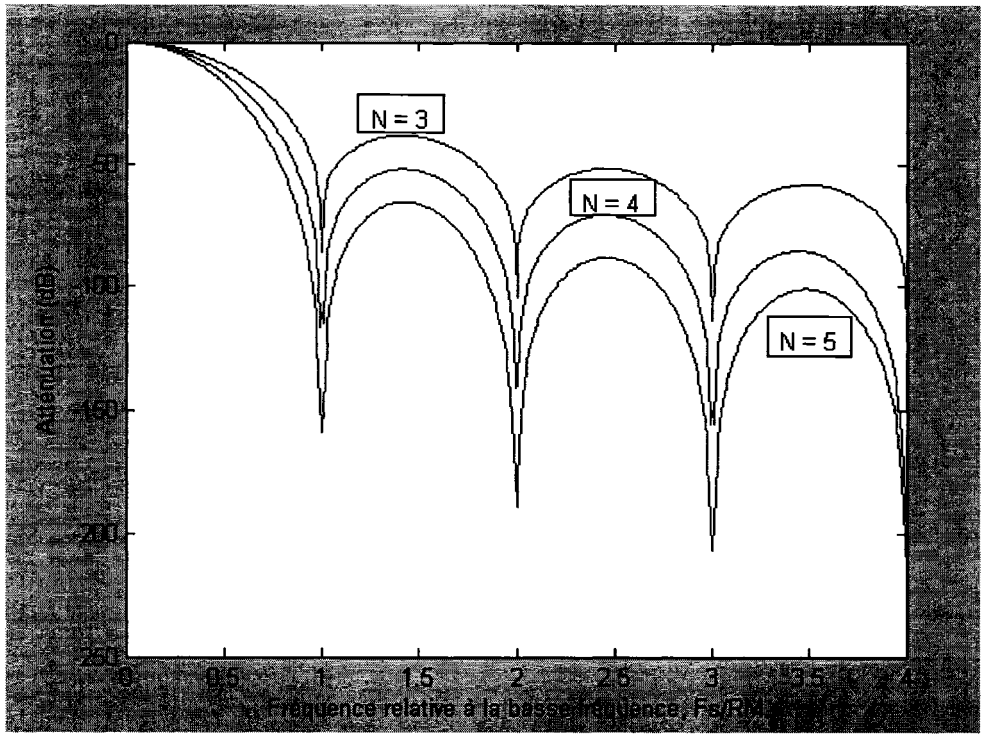


Figure 26 Réponse en amplitude du CIC avec différentes valeurs de N

La figure 26 montre que le filtre CIC décimateur est un filtre passe bas dans l'intervalle [0, 1]. Posons $[0, f_c]$ la bande passante et $[f_A, 1]$ la bande image/recouvrement qui a la même largeur que la bande passante (f_c est la fréquence de coupure relative à la basse fréquence). f_A est donné par [Hogenauer] : $f_A = 1 - f_c$. Ainsi, en augmentant l'ordre du filtre CIC, l'atténuation dans la bande image/recouvrement est meilleure mais la perte dans la bande passante augmente. Les tableaux IV et V présentent respectivement l'atténuation dans la bande passante et celle dans la bande image/recouvrement en

fonction de l'ordre du filtre [7]. Dans ces tableaux, il est considéré que le facteur de décimation est assez grand pour permettre la simplification de la réponse en amplitude du filtre CIC $P(f)$ de l'équation (3.6) [7],

$$P(f) = \left[\frac{\sin \pi M f}{\sin \frac{\pi f}{R}} \right]^{2N} \tag{3.6}$$

en $P_s(f)$ de l'équation (3.7)

$$P_s(f) = \left[RM \frac{\sin \pi M f}{\pi M f} \right]^{2N} \tag{3.7}$$

f est la fréquence relative à la basse fréquence et M le délai différentiel. L'erreur faite en simplifiant $P(f)$ par $P_s(f)$ est inférieure à 1 dB si $RM \geq 10$, $1 \leq N \leq 7$ et $0 \leq f \leq \frac{255}{256M}$ [7].

Tableau IV

Atténuation dans la bande passante

Largeur de bande relative à la basse fréquence (f_c)	L'atténuation (en dB) dans la bande passante à la fréquence (f_c) en fonction de l'ordre du filtre N					
	1	2	3	4	5	6
1/128	0.00	0.00	0.00	0.00	0.00	0.01
1/64	0.00	0.01	0.01	0.01	0.02	0.02
1/32	0.01	0.03	0.04	0.06	0.07	0.08

Tableau IV (suite)

Largeur de bande relative à la basse fréquence (f_c)	L'atténuation (en dB) dans la bande passante à la fréquence (f_c) en fonction de l'ordre du filtre N					
	1	2	3	4	5	6
1/16	0.06	0.11	0.17	0.22	0.28	0.34
1/8	0.22	0.45	0.67	0.90	1.12	1.35
1/4	0.91	1.82	2.74	3.65	4.56	5.47

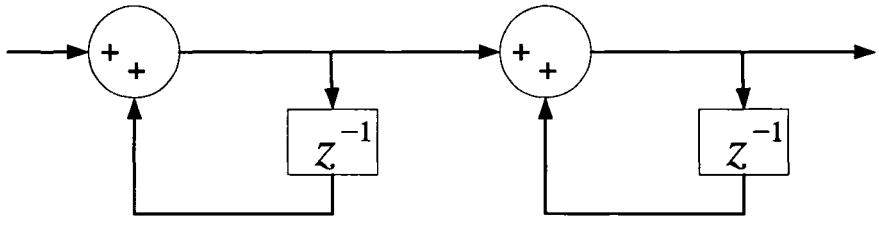
Tableau V

Atténuation dans la bande image/recouvrement

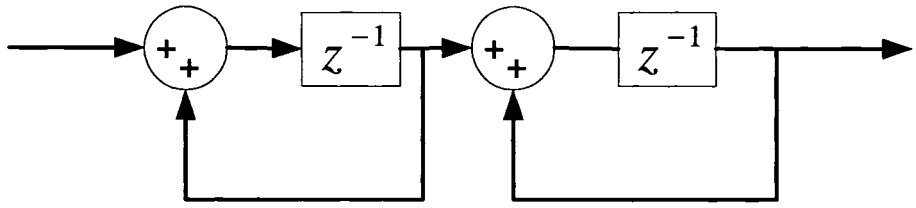
Délai différentiel (M)	Bande passante relative	Atténuation (en dB) dans la bande image/recouvrement à la fréquence f_A					
		1	2	3	4	5	6
1	1/128	42.1	84.2	126.2	168.3	210.4	252.5
1	1/64	36.0	72.0	108.0	144.0	180.0	215.9
1	1/32	29.8	59.7	89.5	119.4	149.2	179.0
1	1/16	23.6	47.2	70.7	94.3	117.9	141.5
1	1/8	17.1	34.3	51.4	68.5	85.6	102.8
1	1/4	10.5	20.9	31.4	41.8	52.3	62.7
2	1/256	48.1	96.3	144.4	192.5	240.7	288.8
2	1/128	42.1	84.2	126.2	168.3	210.4	252.5
2	1/64	36.0	72.0	108.0	144.0	180.0	216.0
2	1/32	29.9	59.8	89.6	119.5	149.4	179.3
2	1/16	23.7	47.5	71.2	95.0	118.7	142.5
2	1/8	17.8	35.6	53.4	71.3	89.1	106.9

3.4 Structure pipeline des intégrateurs

Pour augmenter le débit des filtres CIC décimateurs, le design doit être fait de façon à pouvoir utiliser une grande fréquence de fonctionnement. La fréquence maximale de fonctionnement est donnée par le délai maximal entre deux registres adjacents sensibles au même front d'horloge. Dans un filtre CIC décimateur, les intégrateurs fonctionnent à une fréquence R fois supérieure à celle des différentiateurs. Donc la fréquence de fonctionnement des intégrateurs donne la fréquence de fonctionnement du filtre CIC au complet. Ainsi, augmenter le débit d'un filtre CIC revient à augmenter la fréquence de fonctionnement des intégrateurs. La structure pipeline des intégrateurs, comparée à celle régulière présentée par Hogenauer, permet d'augmenter la fréquence de fonctionnement des intégrateurs. La figure 27 montre la structure régulière et pipeline des intégrateurs [8]. Il est intéressant de constater que la structure pipeline utilise les mêmes ressources matérielles que la structure régulière. Ceci veut dire que la structure pipeline des intégrateurs ne nécessite pas de registres additionnels. Ainsi, comme le délai entre deux registres adjacents détermine la fréquence maximale, alors plus il est court plus la fréquence est grande. Dans la structure régulière, il existe deux additionneurs entre deux registres adjacents alors qu'il en existe qu'un seul dans la structure pipeline d'où sa préférence par rapport à celle régulière. La structure pipeline n'est pas nécessaire dans la partie différentiateur du CIC car la fréquence est beaucoup plus basse comparée à celle des intégrateurs.



(a) structure régulière



(b) structure pipeline

Figure 27 Structure régulière et pipeline du CIC d'ordre 2

3.5 Augmentation de la largeur des registres sans troncation

Dans un filtre récursif idéal à pôle simple, l'addition de deux nombres binaires peut provoquer un débordement si la largeur binaire de la somme dépasse celle réservée dans l'implémentation du CIC. Étant donné que les N premiers étages d'un filtre CIC sont constitués d'intégrateurs avec des rétroactions unitaires, alors la largeur binaire de leur sortie peut augmenter sans limite si les données d'entrée sont statistiquement indépendantes. Ceci provoque du débordement dans les registres des intégrateurs qui peut être résolu en respectant les deux conditions suivantes [7] :

- Utiliser une arithmétique complément à deux ou une autre arithmétique qui permet le passage du plus grand nombre positif au plus grand nombre négatif.
- L'amplitude maximale utilisée pour tous les étages de calculs doit être supérieure ou égale à l'amplitude maximale à la sortie du filtre CIC au complet.

Si ces deux conditions sont réunies, alors le débordement dans les étages intermédiaires n'affecte pas les résultats finaux. Ceci est expliqué dans les sections suivantes.

3.5.1 Représentation en complément à deux

Une représentation ou codification d'un nombre est la façon selon laquelle il est décrit sous forme binaire. La représentation des nombres sur un ordinateur est indispensable pour que celui-ci puisse les stocker et les manipuler. Il s'agit donc de prédéfinir un nombre de bits et la manière de les utiliser pour que ceux-ci servent le plus efficacement possible à représenter l'entité. La représentation en complément à deux d'un nombre positif est obtenue en soustrayant de sa valeur absolue 2^n où n représente la largeur binaire du nombre. Ceci consiste, pratiquement, à inverser tous les bits du nombre positif et y ajouter 1. Le bit le plus significatif (le bit de signe) est 1, ce qui représente un nombre négatif. La figure 28 présente la sortie d'un compteur de 4 bits en représentation décimale non signée et sa conversion en complément à deux. Il faut remarquer que si la sortie de l'additionneur dépasse 7 en décimal, ceci crée un débordement dans la représentation en complément à deux. Ainsi, une troncation est effectuée en supprimant les bits qui sont au delà du MSB ce qui permet le passage entre le plus grand nombre positif et le plus grand nombre négatif.

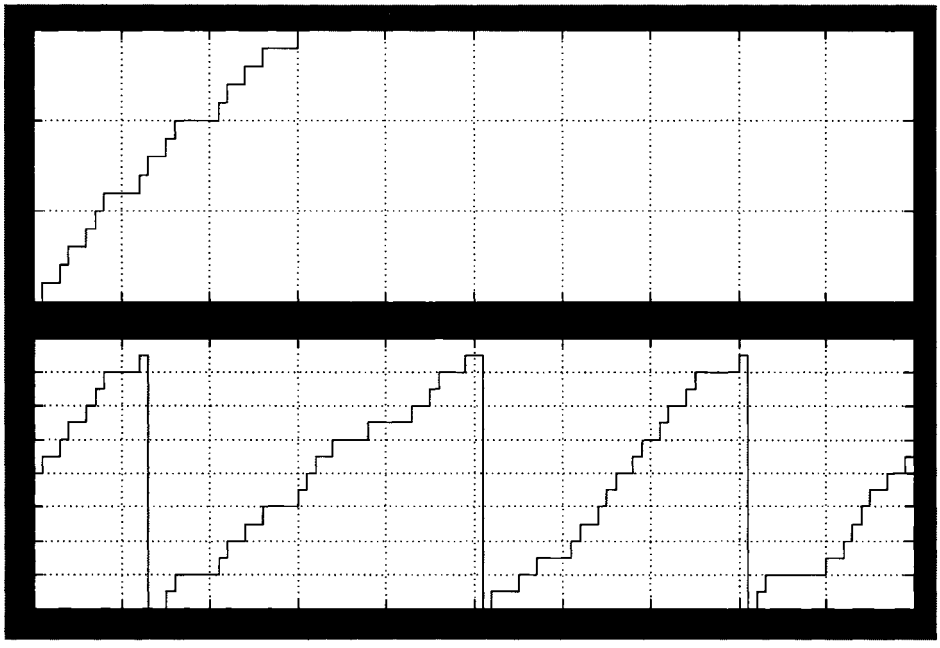


Figure 28 Sortie d'un compteur de 4 bits en décimal non signé et en complément à deux

3.5.2 Calcul de la largeur maximale des registres

La largeur maximale des registres se calcule en se basant sur le gain des intégrateurs. Ce calcul est très important dans l'implémentation d'un filtre CIC car il permet de trouver la largeur maximale des registres qui permet de négliger les débordements à l'intérieur des étages intermédiaires. Ceci sera expliqué plus tard.

Dans un filtre CIC décimateur, les intégrateurs fonctionnent à une fréquence R fois plus rapide que celle des différentiateurs. Ceci fait que chaque intégrateur a un gain de R. Alors, un CIC d'ordre N a un gain maximal G_{max} qui est donné par [7] :

$$G_{max} = R^N \tag{3.8}$$

Ainsi, si la largeur binaire du signal d'entrée est B_{in}, alors la largeur binaire maximale $B_{\max}$, qu'il faut à la sortie du filtre, est donnée par [7] :

$$B_{\max} = [N \log_2 R + B_{in}] \tag{3.9}$$

où le bit le moins significatif du signal d'entrée est le bit 0 de $B_{\max}$ et le résultat de l'opérateur $[x]$ est l'entier le plus petit mais pas inférieur à x lui-même .

L'équation (3.9) peut être utilisée pour calculer la largeur binaire maximale à la sortie du filtre CIC décimateur polyphasé. Si la largeur binaire du signal d'entrée est B_{in1}, alors la largeur binaire $B_{\max\_FIR}$ à la sortie du FIR polyphasé est donné par :

$$B_{\max\_FIR} = [N_1 \log_2 R_1] + B_{in1} \tag{3.10}$$

Comme la sortie du FIR polyphasé est l'entrée du CIC régulier, alors la largeur binaire à l'entrée du CIC régulier est $B_{\max\_FIR}$. La largeur maximale $B_{out\_CIC\_poly}$ à la sortie du CIC polyphasé est donné par :

$$B_{out\_CIC\_poly} = [N_2 \log_2 (R_2 M)] + B_{\max\_FIR} \tag{3.11}$$

Le deuxième facteur de décimation R_2 est égale à $\frac{R}{R_1}$. L'équation (3.11) devient alors :

$$\begin{aligned} B_{out\_CIC\_poly} &= \left[N_2 \log_2 \left(\frac{R}{R_1} M \right) \right] + B_{\max\_FIR} \\ &= [N_2 (\log_2 (RM) - \log_2 (R_1))] + B_{\max\_FIR} \end{aligned}$$

$$= [N_2(\log_2(RM) - \log_2(R_1))] + [N_1 \log_2 R_1] + B_{in1}$$
$$= \underbrace{[N_2 \log_2 R] + B_{in1}}_{B_{max}} - ([N_2 \log_2 R_1] - [N_1 \log_2 R_1]) \quad (3.12)$$

L'équation (3.12) montre que la largeur maximale du CIC décimateur polyphasé est plus courte que celle du CIC régulier de $([N_2 \log_2 R_1] - [N_1 \log_2 R_1])$ bits. Ceci est dû, principalement, à la décimation multi-étage et sera expliqué plus tard. Cette réduction de la largeur des registres est très intéressante car elle augmente la vitesse des additionneurs et diminue la complexité.

3.5.3 Largeur des registres du CIC décimateur

La largeur maximale des registres d'un filtre CIC décimateur est strictement liée au facteur de décimation R et à l'ordre du filtre N. Une fois ces deux paramètres déterminés, le gain G_{max} est fixe. Ainsi, si $B_{max}-1$ est le bit le plus significatif à la sortie du filtre, la largeur du registre à la sortie est suffisante. De ce fait, l'amplitude maximale utilisée pour tous les étages de calcul est supérieure à l'amplitude maximale à la sortie du filtre CIC au complet, ce qui est une des conditions à respecter. Ceci étant fait, si l'arithmétique complément à deux est utilisée, ce qui est la deuxième condition à respecter, alors le débordement dans les étages intermédiaires n'affecte pas les résultats à la sortie du filtre. La figure 29 présente le comportement de débordement dans un compteur binaire en complément à deux.

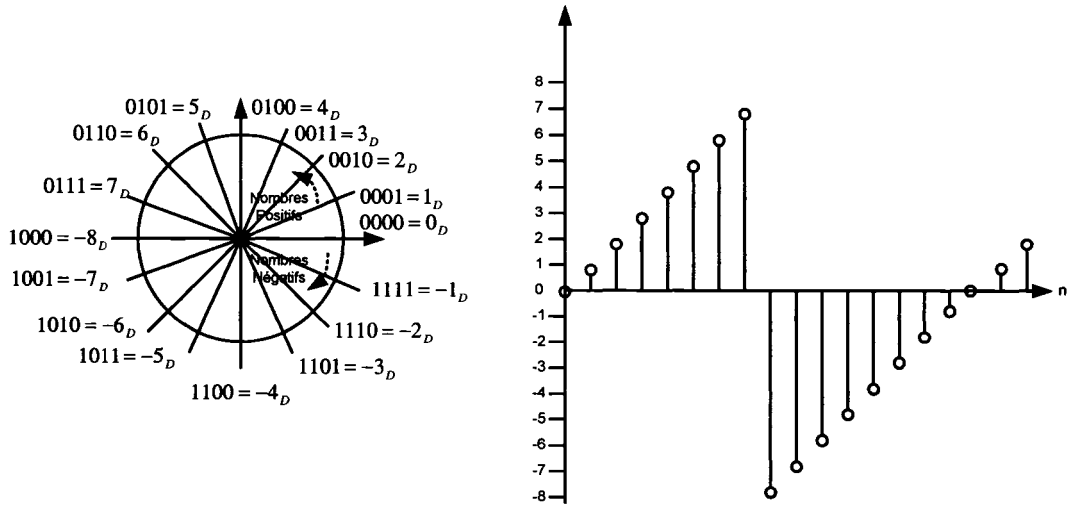


Figure 29 Comportement de débordement dans un compteur binaire en complément à deux

Le débordement est périodique. Il est intéressant de remarquer que la différence entre les points du compteur (sur le cercle) est correcte même si le compteur a subi des débordements. Comme exemple, examinons la réponse transitoire d'un filtre CIC décimateur avec un délai différentiel $M = 4$. Ici, le nombre 1 est ajouté à l'accumulateur à chaque coup d'horloge et le différentiateur qui suit l'accumulateur fait la différence entre l'entrée présente et celle formée il y'a 4 coups d'horloge au paravent. La séquence de sortie est présentée au tableau VI, où nous pouvons observer que la sortie du différentiateur est correcte malgré le débordement dans l'accumulateur.

Donc, si l'accumulateur est suffisamment large et que le filtre CIC est implémenté avec une arithmétique complément à deux, alors la sortie du CIC ne sera pas affectée par les débordements dans les étages intermédiaires.

Tableau VI

Réponse transitoire d'un filtre CIC décimateur avec $M = 4$

X (n) voulu	X (n) Débordement	Opération	X (n-4) voulu	X (n-4) Débordement		Y (n) voulu	Y (n) débordement
0	0	-	0	0	=	0	0
1	1	-	0	0	=	1	1
2	2	-	0	0	=	2	2
3	3	-	0	0	=	3	3
4	4	-	0	0	=	4	4
5	5	-	1	1	=	4	4
6	6	-	2	2	=	4	4
7	7	-	3	3	=	4	4
8	-8	-	4	4	=	4	-12 = 4
9	-7	-	5	5	=	4	-12 = 4
10	-6	-	6	6	=	4	-12 = 4
11	-5	-	7	7	=	4	-12 = 4
12	-4	-	8	-8	=	4	4
15	-3	-	9	-7	=	4	4
14	-2	-	10	-6	=	4	4
15	-1	-	11	-5	=	4	4

3.6 Erreur de troncation ou d'arrondi

L'implémentation des filtres CIC décimateur exige que tous les registres aient la largeur binaire B_{max} donnée par l'équation (3.9) pour que les débordements n'affectent pas le résultat final. Ainsi, la sortie du filtre a une largeur binaire égale à B_{max}. Cependant, en

pratique, la largeur binaire de la sortie est prédéfinie par l'utilisateur et elle est en générale très inférieure à B_{max}. Pour réduire cette largeur binaire B_{max} à la largeur binaire spécifiée par l'utilisateur, la troncation ou l'arrondi est utilisée et ceci introduit de l'erreur qui va jouer sur la précision des résultats. La troncation ou l'arrondi peut être appliquée à l'entrée de chaque étage ce qui réduirait progressivement la largeur binaire. Dans ce cas, il y aura $2N+1$ sources d'erreur de troncation ou d'arrondi qui vont contribuer à l'erreur observée à la sortie du filtre. Les $2N$ sources d'erreur proviennent des registres d'entrée de chaque étage et une source d'erreur de plus qui provient du registre à la sortie du filtre.

Pour calculer l'erreur produite par la troncation ou l'arrondi, supposons que les sources d'erreur produisent du bruit blanc qui est non corrélé avec le signal d'entrée et les autres sources d'erreur et que l'erreur à la source j a une probabilité de distribution uniforme. Il sera démontré plus tard que l'erreur statistique provoquée par la troncation ou par l'arrondi est la même à l'exception des registres d'entrée et de sortie. Étant donné que la troncation est plus facile à implémenter, elle est utilisée pour le calcul de l'erreur.
L'erreur introduite par la j^{ieme} source est donnée par :

$$E_j = 2^{B_j} \tag{3.13}$$

avec B_j le nombre de LSB supprimés à la j^{ieme} source. Comme l'erreur a une distribution uniforme, alors sa moyenne est donnée par :

$$\mu_j = \frac{1}{2} E_j \tag{3.14}$$

et la variance donnée par :

$$\sigma_j^{\,2} = \frac{1}{12} E_j^{\,2} \tag{3.15}$$

Dans un filtre CIC d'ordre N, le nombre de sources d'erreur est égale à $2N+1$ et la j^{ieme} source d'erreur se propage à partir du j^{ieme} étage jusqu'au dernier étage de différentiateur. Ainsi, pour connaître la contribution de cette source d'erreur à l'erreur totale observée à la sortie du filtre, il faut calculer la fonction de transfert du filtre CIC à partir du j^{ieme} étage jusqu'au dernier étage de différentiateur. Cependant, il existe deux plages de calculs possibles pour cette fonction de transfert. La première plage est [1, N] et la deuxième [$N+1$, $2N$].

Dans la première plage, la fonction de transfert est donnée par [7] :

$$H_j(z) = H_I^{N-j+1} H_C^N = \sum_{k=0}^{(RM-1)N+j-1} h_j(k) z^{-k} \quad , \quad j = 1, 2, \ldots, N \tag{3.16}$$

Une autre façon d'écrire cette fonction de transfert est d'utiliser l'expansion binomiale des termes de l'étage intégrateur et l'étage différentiateur :

$$H_j(z) = \left[\sum_{l=0}^{N} (-1)^l \binom{N}{l} z^{-RMl} \right] \left[\sum_{v=0}^{\infty} \binom{N-j+v}{v} z^{-v} \right] \tag{3.17}$$

Le produit des deux termes de l'équation (3.17) donne :

$$H_j(z) = \sum_{l=0}^{N} \sum_{v=0}^{\infty} (-1)^l \binom{N}{l} \binom{N-j+v}{v} z^{-(RMl+v)} \tag{3.18}$$

En posant $k = RMl + v$, avec l un entier dans l'intervalle $[0, \frac{k}{RM}]$, l'équation (3.18) devient :

$$H_j(z) = \sum_{k\geq 0}\left[\sum_{l=0}^{k/RM}(-1)^l \binom{N}{l}\binom{N-j+k-RlM}{k-Rlm}\right]z^{-k} \tag{3.19}$$

En comparant les équations (3.19) et (3.16), $h_j(k)$ est donné par :

$$H_j(k) = \sum_{l=0}^{k/RM}(-1)^l \binom{N}{l}\binom{N-j+k-RlM}{k-RlM} \tag{3.20}$$

Dans la deuxième plage, la fonction de transfert est donnée par :

$$H_j(z) = H_C^{2N+1-j} = \left(1 - z^{-RM}\right)^{2N+1-j}, \qquad j = N+1,.....,2N \tag{3.21}$$

En faisant, une expansion binomiale, l'équation (3.21) devient :

$$H_j(z) = \sum_{k=0}^{2N+1-j} h_j(k) z^{-kRM} \quad , \quad j = N+1,.....,2N \tag{3.22}$$

avec $h_j(k)$ qui est donné par l'équation (3.23) :

$$h_j(k) = (-1)^k \binom{2N+1-j}{k}, \qquad j = N+1,.....,2N \tag{3.23}$$

Ainsi, en résumé, la fonction de transfert du j^{ieme} étage jusqu'au dernier étage de différentiateur est donné par :

$$H_j(z)=\begin{cases} H_I^{N-j+1}H_C^N = \sum_{k=0}^{(RM-1)N+j-1} h_j(k)z^{-k}, j=1,2,.....,N \\ H_C^{j-N} = \sum_{k=0}^{2N+1-j} h_j(k)z^{-kRM}, j=N+1,.....,2N \end{cases} \quad (3.24)$$

avec

$$h_j(k)=\begin{cases} \sum_{l=0}^{(k/R)}(-1)^l \binom{N}{l}\binom{N-j+k-RlM}{k-RlM}, j=1,2,.....,N \\ (-1)^k \binom{2N+1-j}{k}, j=N+1,.....,2N \end{cases} \quad (3.25)$$

La figure 30 montre la propagation de l'erreur de la j^{ieme} source d'erreur.

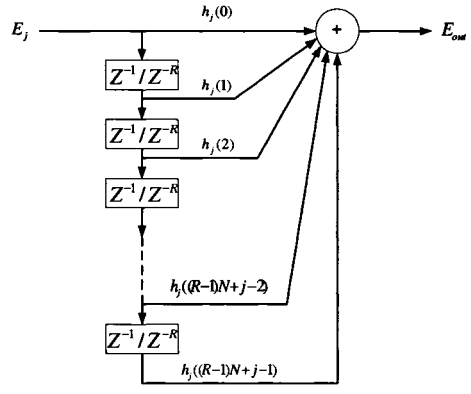


Figure 30 Propagation de l'erreur de la j^{ieme} au dernier étage de différentiateur

Cette fonction de transfert permet de calculer la moyenne et la variance de l'erreur introduite par la j^{ieme} source à la sortie du filtre. La moyenne de l'erreur est donnée par [7] :

$$\mu_{out} = \mu_j D_j$$

$$\text{avec} \quad D_j = \begin{cases} \sum_k h_j(k), j = 1,2,.....,2N \\ 1, j = 2N+1 \end{cases} \tag{3.26}$$

qui représente le gain de l'erreur moyenne de la j^{ieme} source. La variance de cette erreur à la sortie est donnée par :

$$\sigma_{out}{}^2 = \sigma_j{}^2 F_j{}^2$$

$$\text{avec} \quad F_j{}^2 = \begin{cases} \sum_k h_j{}^2(k), j = 1,2,.....,2N \\ 1, j = 2N+1 \end{cases} \tag{3.27}$$

qui représente le gain de la variance de l'erreur de la j^{ieme} source.

Le gain de l'erreur moyenne de la j^{ieme} source présentée à l'équation (3.26) peut être simplifiée sous la forme suivante :

$$D_j = \begin{cases} (RM)^N, j = 1 \\ 0, j = 2,3,.....,2N \\ 1, j = 2N+1 \end{cases} \tag{3.28}$$

Cette équation montre que le gain de l'erreur moyenne est nul partout sauf à la première et la dernière source d'erreur. Ceci veut dire que l'erreur moyenne est nulle à l'intérieure des étages intermédiaires. Alors, étant donné que l'erreur moyenne due à l'arrondi est nulle, la différence entre l'erreur moyenne due à la troncation et celle due à l'arrondi n'affecte pas l'erreur moyenne totale μ_{out} à l'exception de la première et de la dernière source d'erreur. Donc, le choix entre la troncation ou l'arrondi n'affecte que la première et la dernière source d'erreur. En ce qui concerne la variance de l'erreur, celle

créée par la troncation est identique à celle créée par l'arrondi. Ainsi, la moyenne totale de l'erreur du signal de sortie et sa variance, dues à la troncation sont données respectivement par [7]:

$$\mu_{out} = \sum_{j=1}^{2N+1} \mu_{outj} = \mu_{out1} + \mu_{out(2N+1)} \tag{3.29}$$

$$\sigma_{out}{}^{2} = \sum_{j=1}^{2N+1} \sigma_{outj}{}^{2} \tag{3.30}$$

3.7 Calcul de la largeur des registres avec troncation

En pratique, la largeur du registre de sortie est tronquée pour répondre aux spécifications désirées. Si l'utilisateur spécifie B_{out} comme largeur binaire de sortie, alors le nombre de LSB à supprimer à la sortie est donné par :

$$B_{2N+1} = B_{\max} - B_{out} + 1 \tag{3.31}$$

Comme c'est la variance de l'erreur qui affecte toutes les sources d'erreur introduites par la troncation ou l'arrondi, elle sera utilisée comme paramètre de design pour calculer le nombre de LSB à supprimer dans chaque étage de calcul. La variance de l'erreur provoquée par la troncation effectuée à l'équation (3.26) est donnée par :

$$\sigma^2_{out(2N+1)} = \sigma^2_{2N+1} \tag{3.32}$$

En faisant toute la troncation au registre de sortie, la largeur binaire maximale $B_{\max}$ est traînée à travers tous les étages de calcul pour n'en choisir à la fin qu'une infime partie de cette largeur. Ceci est considéré comme du traitement inutile. Ainsi, ce serait intéressant de pouvoir tronquer étage par étage. Pour ce faire, Hogenauer utilise les

critères suivants : la somme des $2N$ sources d'erreur à la sortie est inférieure ou égale à σ^2_{2N+1}. Cette somme d'erreur est également distribuée aux $2N$ sources d'erreur ce qui veut dire :

$$\sigma^2_{outj} \leq \frac{1}{2N}\sigma^2_{2N+1}, \qquad j = 1, 2, \ldots\ldots\ldots\ldots, 2N \tag{3.33}$$

Supposons que le nombre de LSB supprimé à la j^{ieme} source est B_j, alors l'erreur de troncation est 2^{B_j} et sa variance $\sigma_j^2 = \frac{1}{12}2^{2B_j}$. La valeur de cette variance à la sortie est calculée en utilisant l'équation (3.27) :

$$\sigma^2_{outj} = \frac{1}{12}2^{2B_j}F_j^2 \tag{3.34}$$

Le remplacement de σ^2_{outj} par sa valeur dans l'équation (3.33) donne :

$$2^{2B_j}F_j^2 \leq \frac{6}{N}\sigma^2_{2N+1} \tag{3.35}$$

$$B_j \leq -\log_2 F_j + \log_2 \sigma_{2N+1} + \frac{1}{2}\log_2 \frac{6}{N}$$

B_j doit être un entier. Alors l'équation (3.35) devient :

$$B_j = \left[-\log_2 F_j + \log_2 \sigma_{2N+1} + \frac{1}{2}\log_2 \frac{6}{N}\right] \tag{3.36}$$

Cette équation permet de choisir le nombre de LSB à supprimer à chaque étage du filtre CIC décimateur.

3.8 Conclusion

La structure du filtre CIC décimateur a été présentée, dans un premier temps, dans ce chapitre. La réponse fréquentielle montre que les intégrateurs amplifient les composantes à basse fréquence et atténuent celles à hautes fréquences. Ces intégrateurs sont instables mais leur connections en série avec les différentiateurs assure la stabilité du filtre. Dans un deuxième temps, les détails d'implémentation ont été présentés. Il fallait calculer la largeur des registres de chaque étage de filtrage. Cependant, comme la troncation introduit de l'erreur, il fallait adopter des critères pour les calculs des différentes largeurs binaires. Les deux critères présentés par Hogenauer ont été utilisés et sont :

- Utiliser une arithmétique complément à deux ou une autre arithmétique qui permet le passage entre le plus grand nombre positif et le plus grand nombre négatif
- L'amplitude maximale utilisée pour tous les étages de calculs doit être supérieure ou égale à l'amplitude maximale à la sortie du filtre CIC au complet.

Ainsi, dans l'implémentation du filtre CIC décimateur, l'arithmétique complément à deux est utilisée pour répondre au premier critère. La largeur maximale du registre de sortie B_{max} est calculée et est en même temps la largeur maximale des registres intermédiaires, ce qui répond au deuxième critère.

CHAPITRE 4

IMPLÉMENTATION DU FILTRE EN VHDL ET SA RÉALISATION SUR FPGA

4.1 Introduction

Dans ce chapitre, l'implémentation du filtre CIC décimateur en utilisant le langage VHDL est présentée. Cette implémentation est basée sur une représentation à point fixe. Le diagramme bloc du filtre est subdivisé en plusieurs sous-blocs et le codage au niveau RTL est effectué pour chaque bloc. Après le codage RTL, les différents sous-blocs sont interconnectés et une vérification du CIC décimateur au complet, pour différentes configurations, est effectuée. Finalement, quelques considérations pour la synthèse et l'optimisation sont présentées.

4.2 Architecture du CIC décimateur polyphasé

L'architecture du CIC décimateur polyphasé à implémenter a déjà été présentée à la section 2.3.2.2. Pour faciliter la compréhension des différentes opérations qu'effectue le filtre CIC décimateur et les différents aspects à prendre en considération pour le codage en VHDL, cette architecture est reprise à la figure 31. À partir d'ici, on utilisera l'expression « noyau programmable » pour designer le filtre CIC décimateur polyphasé.

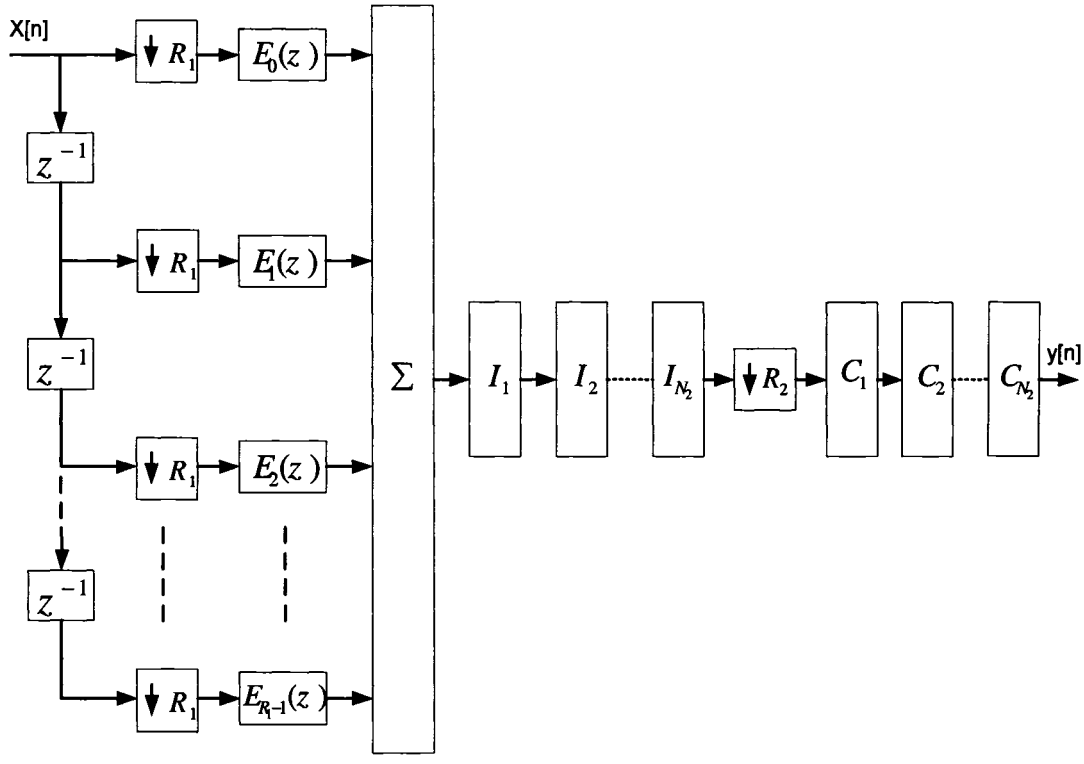


Figure 31 Architecture parallèle du CIC décimateur polyphasé

Avec cette architecture, le noyau programmable est constitué de deux principaux étages de décimation interconnectés de manière sérielle. Le premier étage, placé à l'entrée du noyau programmable, est un filtre FIR polyphasé. Le deuxième étage est un filtre CIC décimateur régulier. Les composantes polyphasées du filtre FIR et les intégrateurs du filtre CIC décimateur régulier fonctionnent à la même fréquence. Par contre, les différentiateurs du CIC décimateur régulier fonctionnent à une fréquence plus basse d'un facteur de R_2 .

4.3 Particularités des différentes configurations

L'utilisateur du noyau programmable peut choisir les valeurs des différents paramètres. Cependant, son choix a une influence directe sur l'architecture matérielle qui sera

implémentée. Tous les paramètres sont programmables avant synthèse à l'exception du facteur de décimation qui peut l'être après synthèse. Ainsi, ce facteur de décimation peut être fixe ou programmable.

4.3.1 Cas d'un facteur de décimation fixe

Si le facteur de décimation est fixe, alors sa valeur ne peut être choisie qu'avant synthèse. Ainsi, le calcul de la largeur maximale des registres sera basé sur cette valeur. L'avantage de cette configuration réside dans le fait que toute la largeur binaire des registres est utilisée pour les différents calculs à faire. Cependant, cette configuration est trop limitée du point de vue de la reconfigurabilité.

4.3.2 Cas d'un facteur de décimation programmable

Si le facteur de décimation est programmable, alors sa valeur peut être choisie avant et après synthèse. Dans ce cas, l'utilisateur doit spécifier la valeur maximale que peut prendre le facteur de décimation après synthèse. Cette valeur maximale sera utilisée pour calculer la largeur maximale des registres. De ce fait, si un facteur de décimation, de valeur outre que la valeur maximale, est utilisé, alors toute la largeur binaire des registres n'est pas utilisée. Ceci veut dire qu'on peut utiliser des largeurs de 40 bits alors qu'on en a besoin que des largeurs de 30 bits. Cependant, cette configuration est reconfigurable ce qui est très important dans le design d'un noyau programmable.

4.4 Implémentation en VHDL

La conception du noyau programmable en VHDL a été divisée en trois niveaux. D'abord, on retrouve le niveau supérieur qui permet à l'utilisateur de choisir les valeurs des différents paramètres. Ensuite, le niveau intermédiaire qui est divisé en trois *PROCESS* : un pour le filtre RIF polyphasé, un pour le CIC décimateur régulier et un

pour le module de contrôle. Enfin, on retrouve au plus bas le niveau, les sous-modules qui sont constitués d'additionneurs, de soustracteurs et d'un compteur.

4.4.1 Niveau supérieur

Le niveau supérieur du noyau programmable établit l'interface avec l'usager. C'est à ce niveau que tous les paramètres programmables peuvent être choisis à l'aide de *generics*. Le nombre d'entrées parallèles et l'ordre du CIC décimateur régulier dictent respectivement le nombre de composantes polyphasées et le nombre d'étages d'intégrateurs et de différentiateurs à utiliser. Étant donné que les coefficients des composantes polyphasées sont fixes pour un nombre d'entrées parallèles donné, alors un *case-is* est utilisé pour choisir les coefficients qu'il faut pour le nombre d'entrées parallèles choisi. En ce qui concerne les étages d'intégrateurs et de différentiateurs, c'est une boucle *for-loop* qui s'exécute.

De nombreuses fonctions s'activent à la compilation avant la synthèse et permettent de fixer adéquatement les largeurs des mots binaires ainsi que les sous-modules créés au niveau inférieur. Toutes ces fonctions sont regroupées dans un package appelé fonctions_utiles et est présenté à la figure 32. La figure 33 montre le symbole du noyau programmable et la figure 34 les paramètres programmables

```
PACKAGE fonctions_utiles IS

FUNCTION int_2_std_logic_vector (value, bitwidth : INTEGER )
   RETURN std_logic_vector;
FUNCTION bitsneededtorepresent( a_value : INTEGER )
   RETURN INTEGER;

FUNCTION extend (vector: std_logic_vector; bits: INTEGER)
```

```
    RETURN std_logic_vector;

FUNCTION std_logic_vector_2_var (vect: std_logic_vector)
    RETURN std_logic_vector;

END fonctions_utiles;
```

Figure 32 Package fonctions_utiles

Fichier : CIC_poly.vhd

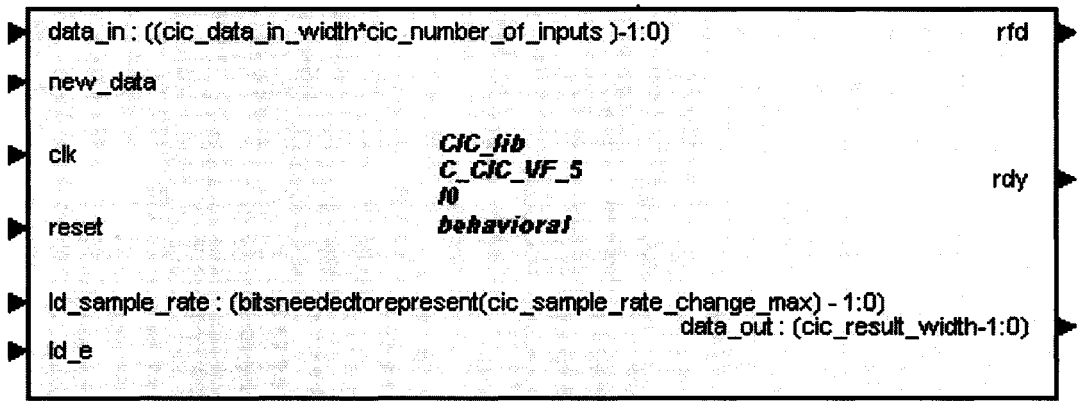


Figure 33 Symbole du noyau programmable

GENERIC:

cic_data_in_width	INTEGER	Largeur binaire de l'entrée du noyau programmable
cic_number_of_inputs	INTEGER	Nombre d'entrées parallèles(R_1)
cic_width_fir_poly	INTEGER	Largeur binaire maximale des bus du FIR polyphasé

cic_stages_cic_reg	INTEGER	L'ordre du CIC régulier (nombre d'intégrateurs et de différentiateurs)
cic_sample_rate_change_type	INTEGER	Type de décimation (1 pour programmable et 2 pour fixe)
cic_sample_rate_change	INTEGER	Facteur de décimation du CIC régulier (R_2)
cic_sample_rate_change_max	INTEGER	Facteur de décimation maximale du CIC régulier dans le cas d'un type de décimation programmable
cic_differential_delay	INTEGER	Délai différentiel des différentiateurs
cic_width_max	INTEGER	Largeur binaire maximale des bus du CIC régulier
cic_result_width	INTEGER	Largeur binaire de la sortie du noyau
cic_latency	INTEGER	Latence de la sortie (permet de retarder la sortie du noyau

Figure 34 Paramètres programmables

Les ports d'entrée et de sortie du noyau programmable ainsi que leur définition sont présentés dans la table 1.

Tableau VII

Ports d'entrée/sortie du noyau et leur définition

Nom du port	Direction	Description
clk	entrée	Horloge
data_in	entrée	Port d'entrée de données du noyau
new_data	Entrée	Signal d'activation de l'horloge *clock enable.* Quand ce signal est actif, les échantillons de donnés présents sur le port data_in sont chargés dans le filtre
reset	entrée	Permet d'initialiser le noyau
ld_sample_rate	entrée	Permet de charger la nouvelle valeur du facteur de décimation si l'option 1 est choisie pour le cic_sample_rate_change_type. Ce port est optionnel et est disponible seulement pour l'option programmable du cic_sample_rate_change_type.
ld_e	entrée	Ce signal est associé avec le port ld_sample_rate. La valeur du port ld_sample_rate est prise en compte par le noyau quand ld_e actif coincide avec un front montant du clock. Comme le port ld_sample_rate, ce port est disponible seulement pour l'option programmable du cic_sample_rate_change_type.
rfd	sortie	Indique si le noyau peut accepter une nouvelle donnée sur Sur le port d'entrée data_in
rdy	sortie	Indique la disponibilité d'une sortie sur le port de sortie data_out
data_out	sortie	Port de sortie du noyau

Les ports *nd (new data), rfd (ready for data) et rdy (ready)* sont utilisés pour contrôler les entrées et les sorties du noyau. Le signal *rfd*, qui est actif haut, indique au système

que le filtre est prêt à accepter des données en entrée. Quand le signal *nd* est actif, il signale au noyau la disponibilité d'une nouvelle entrée sur le port d'entrée *data_in*. Le signal de sortie *rdy* indique la disponibilité d'une nouvelle sortie sur le port *data_out*.

Les signaux de l'interface du noyau peuvent être utilisés de la manière suivante : L'utilisateur du système doit d'abord attendre que *rfd* = 1. Ceci signale qu'une nouvelle entrée peut être traitée par le filtre. La nouvelle entrée est placée sur le port d'entrée *data_in* et *nd* devrait être en mode actif (*nd* = 1) pour un cycle d'horloge. *nd* actif indique au noyau que les données sur le port *data_in* doivent être traitées. Ainsi, le filtre traite les données au front montant de l'horloge qui coïncide avec *nd* = 1. Les données à la sortie du filtre peuvent être récupérées si le noyau assigne la valeur 1 à *rdy*. Ce signal *rdy* peut être utilisé comme *clock enable* par le bloc en bande de base qui récupère les sorties du filtre.

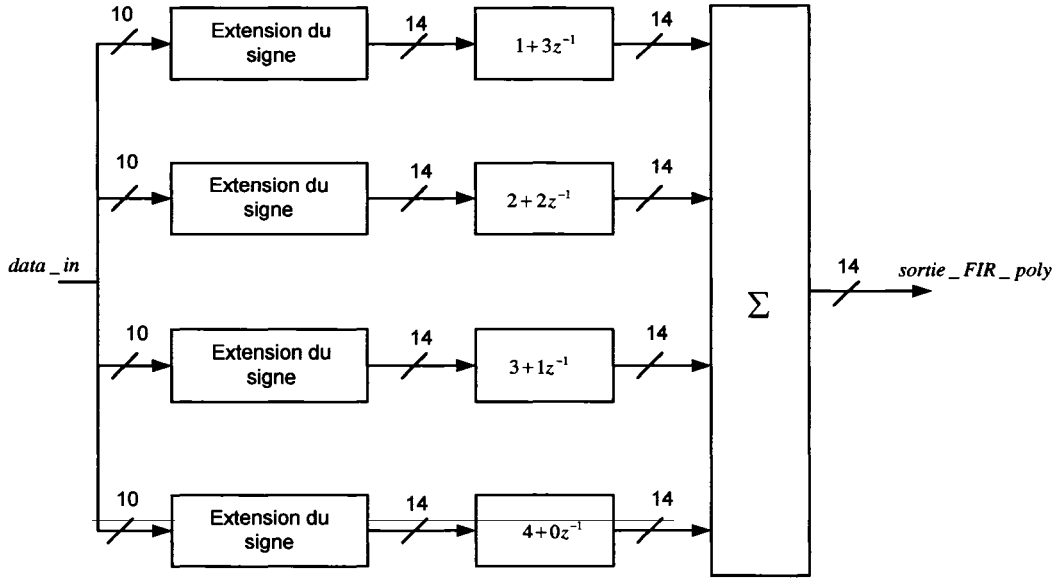
Supposons les deux cas de configurations suivantes :

Cas 1 :

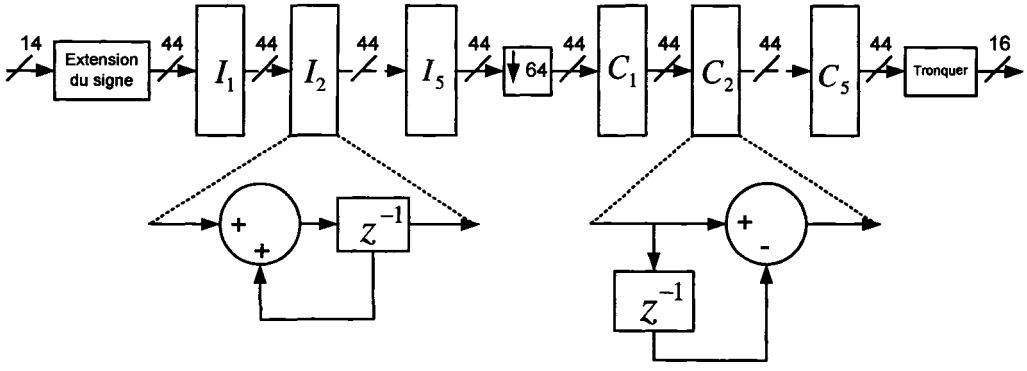
Paramètres	Valeur
cic_data_in_width	10
cic_stage_cic_reg	5
cic_number_of_inputs (R_1)	4
cic_width_fir_poly	$2\log_2(R_1)+10 = 14$ bits
cic_sample_rate_change_type	2 (c'est-à-dire fixe)
cic_sample_rate_change (R_2)	64
cic_differential_delay	1
cic_width_max	$5\log_2(R_2M)+14 = 44$ bits
cic_result_width	16 bits
cic_latency	1

```
ENTITY C_CIC_VF IS
 GENERIC (
        cic_data_in_width              : INTEGER: = 10;
        cic_number_of_inputs           : INTEGER: = 4;
        cic_width_fir_poly             : INTEGER: = 14;
        cic_stages_cic_reg             : INTEGER: = 5;
        cic_sample_rate_change_type    : INTEGER: = 1;
        cic_sample_rate_change         : INTEGER: = 64;
        cic_sample_rate_change_max     : INTEGER: = 64;
        cic_differential_delay         : INTEGER: = 1;
        cic_width_max                  : INTEGER: = 44;
        cic_result_width               : INTEGER: = 16;
        cic_latency                    : INTEGER: = 1);
```

La figure 35 montre l'architecture générée pour cette configuration:



(a) FIR polyphasé



(b) CIC régulier

Figure 35 Architecture générée pour la configuration du cas 1

Une fois cette architecture implémentée, une valeur du facteur de décimation supérieure à celle spécifiée dans le *generic* ne peut pas être utilisée car le calcul de la largeur binaire des bus de données est basé sur cette dernière.

Cas 2 :

Le deuxième cas de configuration est la même chose que le premier, sauf que le type de décimation est programmable. Supposons maintenant que le nombre d'entrées parallèles est égale à 4 et que le CIC régulier doit supporter un facteur de décimation maximale de $R_2 = 256$. Ceci veut dire que le filtre doit supporter un facteur de décimation total $R = R_1 R_2 = 4 * 256 = 1024$. Le filtre FIR polyphasé généré pour cette configuration est le même que celui du cas 1 mais, par contre, le CIC régulier généré est différent et est présenté à la figure 36.

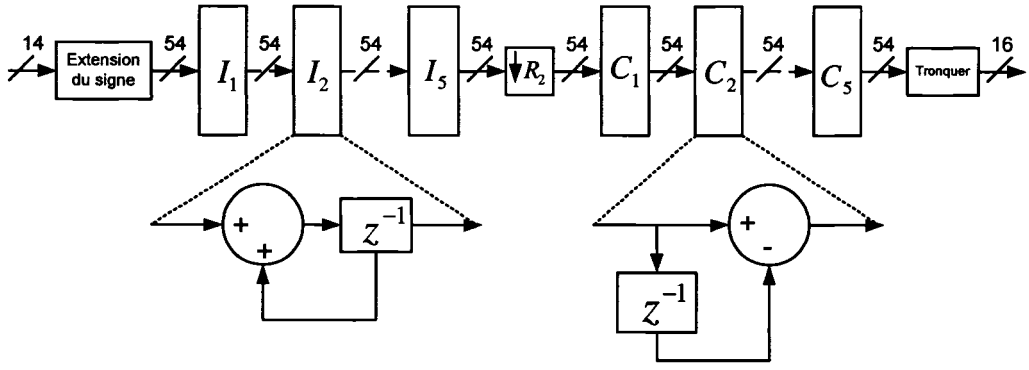


Figure 36 CIC régulier généré pour le deuxième cas de configuration

4.4.2 Niveau intermédiaire

Le niveau intermédiaire contient trois *PROCESS* différents pour ses trois différents blocs qui sont : le FIR polyphasé, le CIC régulier et le module de contrôle.

4.4.2.1 FIR polyphasé

Le nombre d'entrées parallèles du FIR polyphasé peut être 2, 4 ou 8. Comme il est impossible d'avoir des entrées qui changent dynamiquement, le port $data\_in$, qui est l'entrée du noyau programmable et du FIR polyphasé, a une largeur binaire de $(R_1 \times CIC\_data\_in\_width)$ bits qui permet d'accommoder le nombre d'entrées parallèles. Cette largeur binaire sera séparée en R_1 entrées parallèles de largeurs binaires $CIC\_data\_in\_width$ bits chaque. Ces R_1 entrées parallèles de largeurs binaires $CIC\_data\_in\_width$ bits chaque sont les entrées parallèles du FIR polyphasé. Le module du FIR polyphasé possède des signaux de contrôle local qui sont *fir_poly_new_data* et *fir_poly_rdy*. Le signal *fir_poly_new_data* joue le rôle de *clock enable* de l'horloge du module FIR polyphasé. Le signal *fir_poly_rdy*, quand à lui,

indique la disponibilité d'un échantillon sur la sortie du FIR polyphasé. Ainsi, en résumé, le FIR polyphasé fonctionne comme suit : Quand une nouvelle entrée est placée sur le port d'entrée *data_in*, le signal *fir_poly_new_data* est en mode actif (*fir_poly_new_data* = 1) pour un cycle d'horloge. *fir_poly_new_data* actif indique au FIR polyphasé que les données sur le port *data_in* doivent être traitées. Ainsi, le filtre traite les données au front montant de l'horloge qui coïncide avec *fir_poly_new_data* = 1. Les données à la sortie du FIR polyphasé peuvent être récupérées si la valeur 1 est assignée au signal *fir_poly_rdy*. Ce signal est utilisé comme *clock enable* par le bloc d'intégrateurs qui récupère les sorties du FIR polyphasé.

4.4.2.2 CIC régulier

La sortie du FIR polyphasé est présentée à l'entrée du CIC régulier qui est constitué d'un bloc d'intégrateur, d'un facteur de décimation et d'un bloc de différentiateurs. Ainsi, deux *PROCESS* sont utilisés pour le traitement des données dans les blocs d'intégrateurs et de différentiateurs.

Le bloc d'intégrateurs possède des signaux de contrôle qui sont : *integ_new_data* et *integ_rdy*. Étant donné que le bloc d'intégrateur fonctionne à la même fréquence que le FIR polyphasé, alors son signal *integ_new_data* est actif quand *fir_poly_rdy* l'est. Ainsi, le signal présent à l'entrée du bloc d'intégrateur est traité et la valeur 1 est assignée au signal *integ_rdy* qui indique la disponibilité de la sortie du bloc d'intégrateurs.

Le facteur de décimation, qui est réalisé à l'aide d'un compteur, permet de choisir la sortie du bloc d'intégrateurs qui sera présentée à l'entrée du bloc de différentiateurs. Ainsi, si une entrée valable est présentée à l'entrée du bloc de différentiateurs, son signal de contrôle *comb_new_data* est actif. De ce fait, le signal est traité et la valeur 1 est assignée au signal de contrôle *comb_rdy* ce qui indique la disponibilité de la sortie du bloc de différentiateurs. Cette sortie est tronquée et retardée en fonction du délai spécifié

dans *cic_latency* du *generic*. Le résultat est placé dans le port de sortie *data_out* pour donner le résultat final du noyau programmable.

4.4.2.3 Module de contrôle

Le module de contrôle permet de faire le contrôle global du noyau. Il permet de séquencer les signaux de contrôle local du FIR polyphasé et du CIC régulier. Il est essentiellement constitué d'un compteur dont la taille dépend de la valeur du facteur de décimation du CIC régulier.

4.4.3 Sous-modules

Le noyau programmable est constitué d'additionneurs, de soustracteurs, de délais, d'intégrateurs, de différentiateurs et d'un compteur. Cependant, les trois sous-modules de base sont l'intégrateur, le différentiateur et le compteur.

4.4.3.1 Intégrateur

Un intégrateur numérique n'est rien d'autre qu'un accumulateur. Il est constitué d'un additionneur et d'une ligne à délai qui permet de réaliser la boucle de rétroaction. La figure 37 montre un bloc d'intégrateur selon *System generator* de Xilinx. b est l'entrée et q la sortie . L'équation réalisée par ce bloc est la suivante :

$$q(n) = q(n-1) + b(n) \tag{4.1}$$

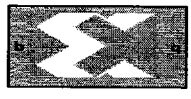


Figure 37 bloc d'intégrateur

L'équation (4.1) montre que la réalisation d'un intégrateur en VHDL est très facile. Ainsi, pour générer N blocs d'intégrateurs en cascade, la boucle *for-loop* est utilisée.

4.4.3.2 Différentiateur

Un différentiateur numérique est réalisé à l'aide d'une ligne délai et d'un soustracteur. La figure 38 montre un bloc de différentiateur avec un délai différentiel $M = 1$ selon *System generator* de Xilinx. Si l'entrée du bloc est $x(n)$ et la sortie $y(n)$, alors l'équation réalisée par ce bloc est la suivante :

$$y(n) = x(n) - x(n-1) \tag{4.2}$$

avec $a = x(n)$ et $b = x(n-1)$. En VHDL, la boucle *for-loop* est utilisée pour générée N blocs de différentiateurs en cascade.

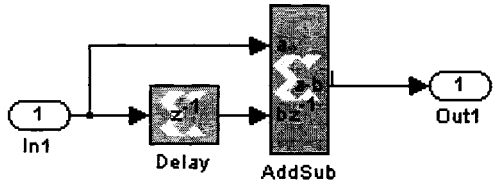


Figure 38 Bloc de différentiateur

4.4.3.3 Compteur

Le facteur de décimation du CIC régulier, R_2, est réalisé à l'aide d'un compteur. Le compteur est incrémenté à chaque fois qu'une sortie est disponible sur le port de sortie du bloc d'intégrateurs. Ainsi, quand le compteur atteint la valeur de R_2, le signal de contrôle *comb_new_data* est mis à 1 pour permettre au bloc de différentiateur de

récupérer la sortie du bloc d'intégrateurs. Étant donné que le facteur de décimation R_2 peut changer de valeur après la synthèse, on peut se poser la question à savoir qu'est ce qui arrive si le facteur de décimation change de valeur alors que le compteur n'a pas encore atteint l'ancienne valeur de R_2. Dans ce cas, le compteur continue à compter jusqu'à ce qu'il atteigne l'ancienne valeur de R_2 et s'initialise à zéro. À partir de ce moment, il compte pour atteindre la nouvelle valeur de R_2.

4.5 Vérification du modèle VHDL

La vérification est extrêmement importante dans un processus de design utilisant les FPGA ou les ASIC. Ainsi, concevoir un circuit, c'est réfléchir à la manière dont il sera décomposé : quelle sera la structure, comment vont se propager les données, où sont les éléments critiques en vitesse, etc, sont autant des questions auxquelles le designer doit répondre pendant la phase de conception. Cette phase est aussi celle durant laquelle est écrit le code VHDL du design. La grande facilité avec laquelle on peut créer des *testbench* (bancs de test) pour les modules VHDL est telle qu'on observe souvent les designers tester les modules au fur et à mesure qu'ils les décrivent. Nous distinguerons donc la simulation de bas niveau, qui permet de tester plus ou moins exhaustivement chaque module ou sous-module, de la simulation de haut niveau, qui a pour objectif de valider le design complet tel qu'il est décrit dans la spécification. Cette méthodologie *bottom-up* (de bas en haut) avec une validation des sous-modules et modules peut diminuer le nombre total de test à faire et engendrer un taux de confiance suffisant [24].

Les modules et les sous-modules ont été validés en utilisant des *testbenchs* spécialisés qui isolent l'entité à tester. De ce fait, la complexité de l'exercice de conception diminue, ce qui permet d'envisager de tester plus complètement la fonction. Les modules ou les sous-modules sont testés de manière presque complète, ce qui augmente la confiance de leur fonctionnalité. Cette technique a été utilisée pour tester le FIR

polyphasé et le CIC régulier. Pour la validation de la fonction à haut niveau, il suffit de vérifier les interconnexions et le timing. La figure 39 montre la vérification à haut niveau du noyau programmable.

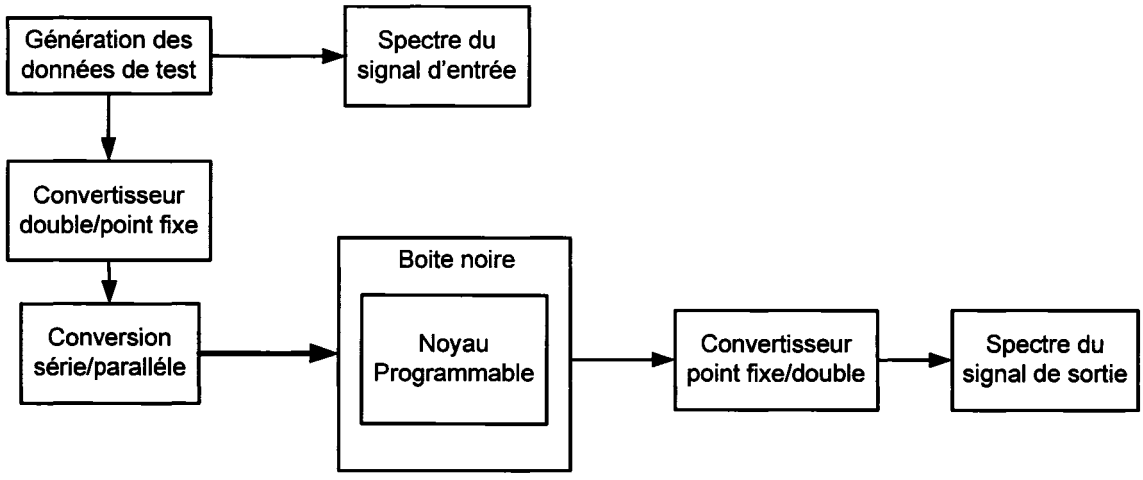
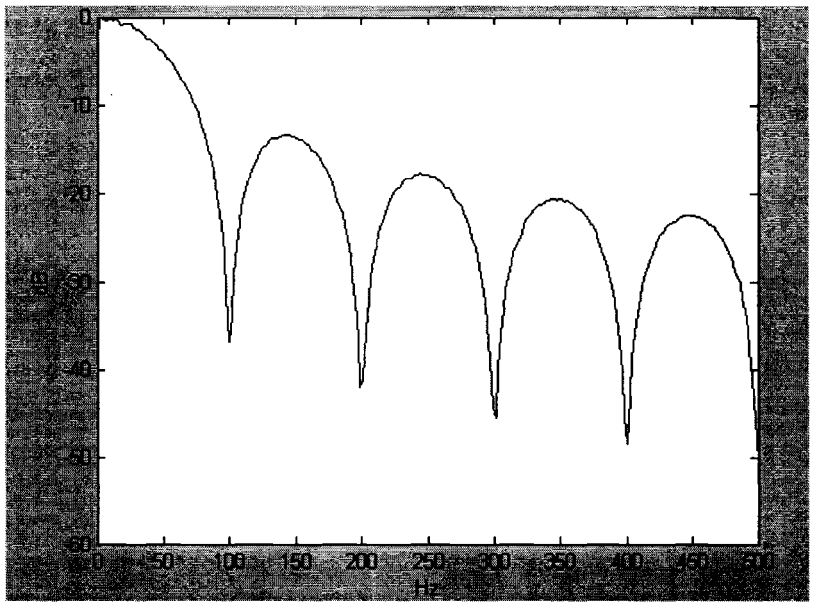
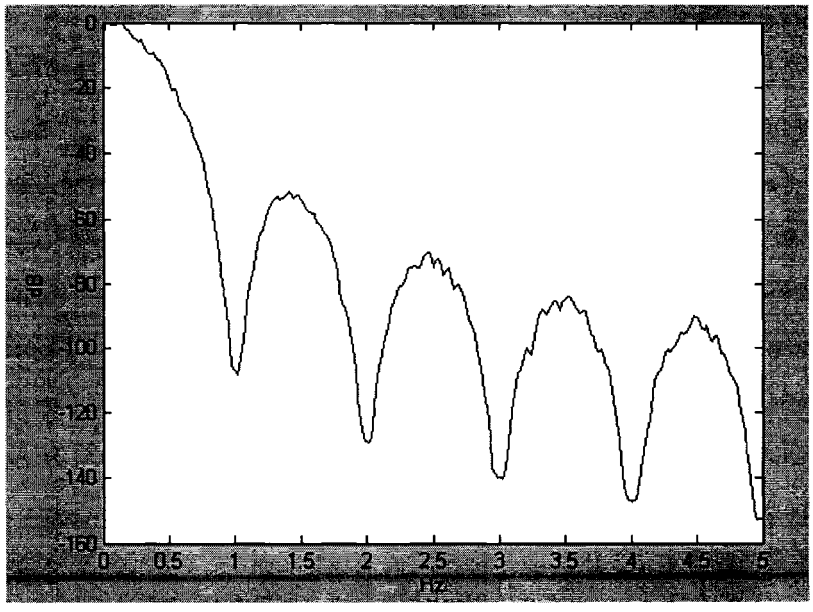


Figure 39 Vérification à haut niveau du noyau programmable

Dans la figure 39, le code VHDL du noyau programmable, à l'aide de Modelsim, a été encapsulé dans une boîte noire pour pouvoir être utilisé dans un modèle *Simulink*. Les données de test générées sont de type double. Le convertisseur double/point fixe permet de convertir les données de test en une arithmétique binaire point fixe complément à deux. Pour interfacer les données avec le bloc du noyau programmable, un convertisseur série parallèle est utilisé. En comparant les spectres du signal d'entrée et de sortie, nous pouvons dire si le noyau fonctionne bien ou pas à haut niveau. La figure 40 montre les spectres des signaux d'entrée et de sortie avec un facteur de décimation $R = 100 (R_1 = 4, R_2 = 25)$ et d'ordre $N = 4$. La figure 40 (a) montre le spectre du signal d'entrée avec une largeur de bande de 100 Hz. La figure 40 (b) montre le spectre normalisé du signal de sortie avec une largeur de bande de 1 Hz. Ceci montre bien que le signal d'entrée a été décimé par 100. Il est intéressant de remarquer l'atténuation de 48 dB (à peu près) sur le spectre du signal décimé.



(a) Spectre d'entrée



(b) Spectre de sortie

Figure 40 Spectres d'entrée et de sortie du noyau programmable avec $R = 100$ et $N = 4$

Pour pouvoir dire que le noyau programmable fonctionne correctement, il faudrait, pour les mêmes spécifications, comparer sa réponse en fréquence avec celle du CIC régulier. Pour ce faire, on utilise, comme référence, le bloc CIC de simulink qui est disponible sous la rubrique *Xilinx blockset.* La structure de vérification est présentée à la figure 41.

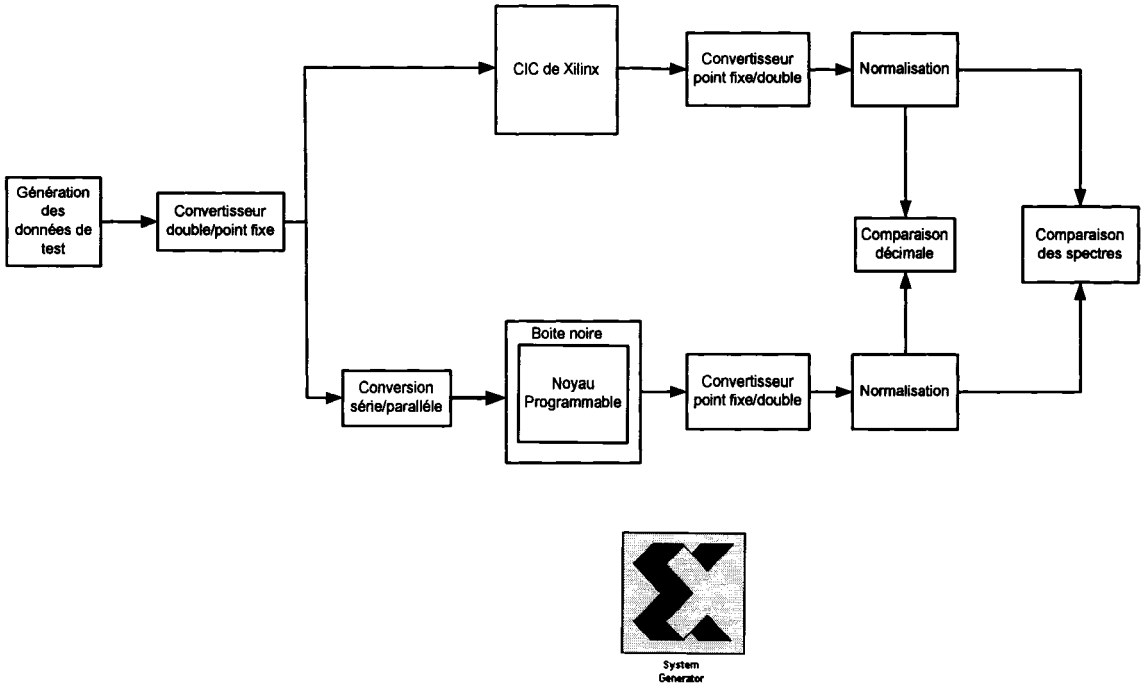
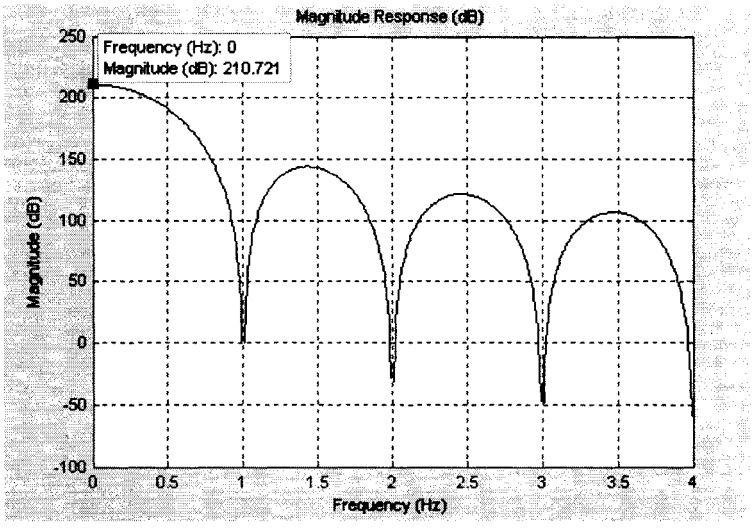


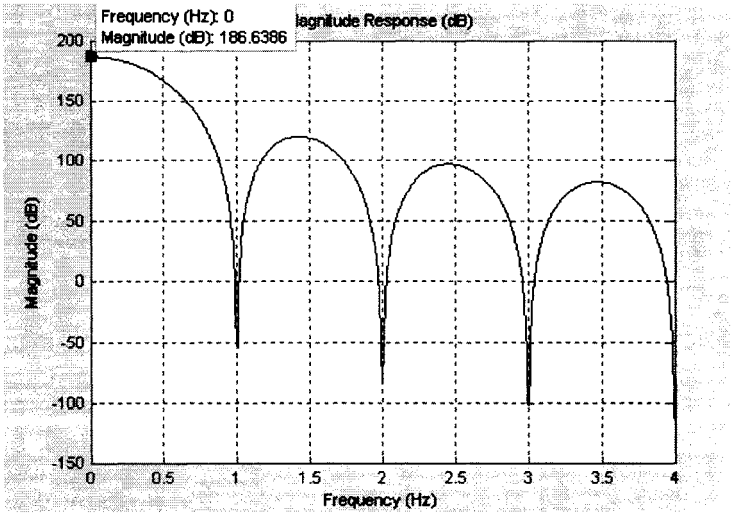
Figure 41 Comparaison du noyau programmable avec le CIC régulier de Xilinx

Dans la figure 41, les données de test sont générées par un générateur de données aléatoire. Le convertisseur double/ point fixe convertit les données de type double en binaire point fixe complément à deux pour que ces dernières puissent être traitées par les filtres. Le convertisseur série/parallèle permet d'interfacer les données qui arrivent de façon sérielle avec les entrées parallèles du noyau. Aux sorties des filtres, les données sont converties en format double par les convertisseurs point fixe/double. À ce stade, vu que les deux filtres n'ont pas le même gain DC, les données sont normalisées pour pouvoir être comparées. Le bloc comparaison des spectres contient du code Matlab qui

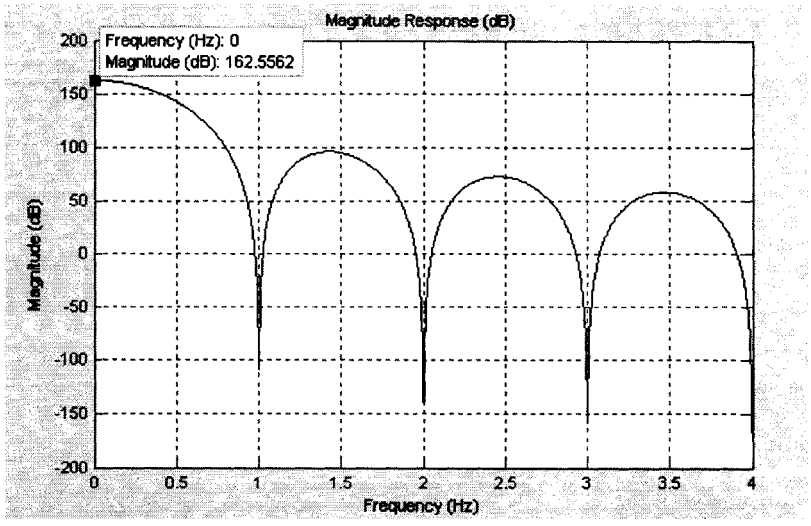
permet de tracer les spectres de sorties sur la même figure. De ce fait, on pourra les comparer. La figure 42 montre, pour $f_s = 128$ Hz, $R = 128$, $N = 5$ et $M = 1$ les spectres du CIC régulier de Xilinx et du noyau programmable avec 2, 4 et 8 entrées parallèles.



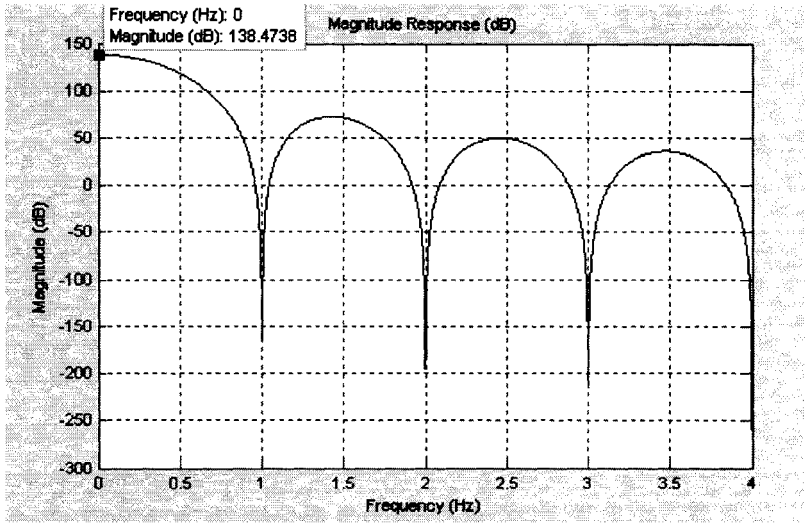
(a) CIC régulier de Xilinx



(b) Noyau programmable avec deux entrées parallèles



(c) Noyau programmable avec quatre entrées parallèles



(d) Noyau programmable avec huit entrées parallèles

Figure 42 Spectres non normalisés du CIC régulier de Xilinx et du noyau programmable

La première remarque à faire sur les spectres présentés à la figure 42 est la différence sur le gain DC. Le gain DC d'un CIC régulier est donné par l'équation suivante :

$$G(dB) = 20\log\left((RM)^N\right) \tag{4.3}$$

En remplaçant les inconnus de l'équation (4.3) par leur valeur, le gain DC du CIC régulier présenté à la figure 10 (a) donne :

$$G(dB) = 20\log\left((128\times1)^5\right) = 210.72 \text{ dB}$$

En faisant une décomposition polyphasé du CIC régulier, le gain DC se transforme et s'écrit comme suit :

$$G(dB) = 20\log(R_1) + 20\log\left((R_2M)^N\right) \tag{4.4}$$

Ainsi, pour $R_1 = 2$ c'est-à-dire deux entrées parallèles, $R_2 = 64$ et le gain DC du CIC polyphasé présenté à la figure 10 (b) est donné par :

$$G(dB) = 20\log(2) + 20\log\left((64\times1)^5\right) = 186.63 \text{ dB}$$

Pour $R_1 = 4$ c'est-à-dire quatre entrées parallèles, $R_2 = 32$ et le gain DC du CIC polyphasé présenté à la figure 10 (c) est donné par :

$$G(dB) = 20\log(4) + 20\log\left((32\times1)^5\right) = 162.55 \text{ dB}$$

Pour $R_1 = 8$ c'est-à-dire huit entrées parallèles, $R_2 = 16$ et le gain DC du CIC polyphasé présenté à la figure 10 (d) est donné par :

$$G(dB) = 20\log(8) + 20\log\left((16\times1)^5\right) = 138.47 \text{ dB}$$

Les résultats théoriques coïncident bien avec les résultats expérimentaux présentés à la figure 10. Cependant, il faut noter que le gain DC du CIC polyphasé est plus petit que celui du CIC régulier. Ceci est dû au fait que la décimation du CIC polyphasé se fait sur deux étages alors que celle du CIC régulier se fait sur un seul étage. L'équation (3.12) montrait que la largeur binaire maximale des bus de données du CIC décimateur polyphasé est plus courte que celle du CIC régulier de $([N_2 \log_2 R_1] - [N_1 \log_2 R_1])$ bits. Cette diminution de la largeur binaire maximale du CIC polyphasé est due à la diminution de son gain DC. En normalisant le gain DC à 1, les spectres du CIC régulier et du noyau programmable sont présentés à la figure 43.

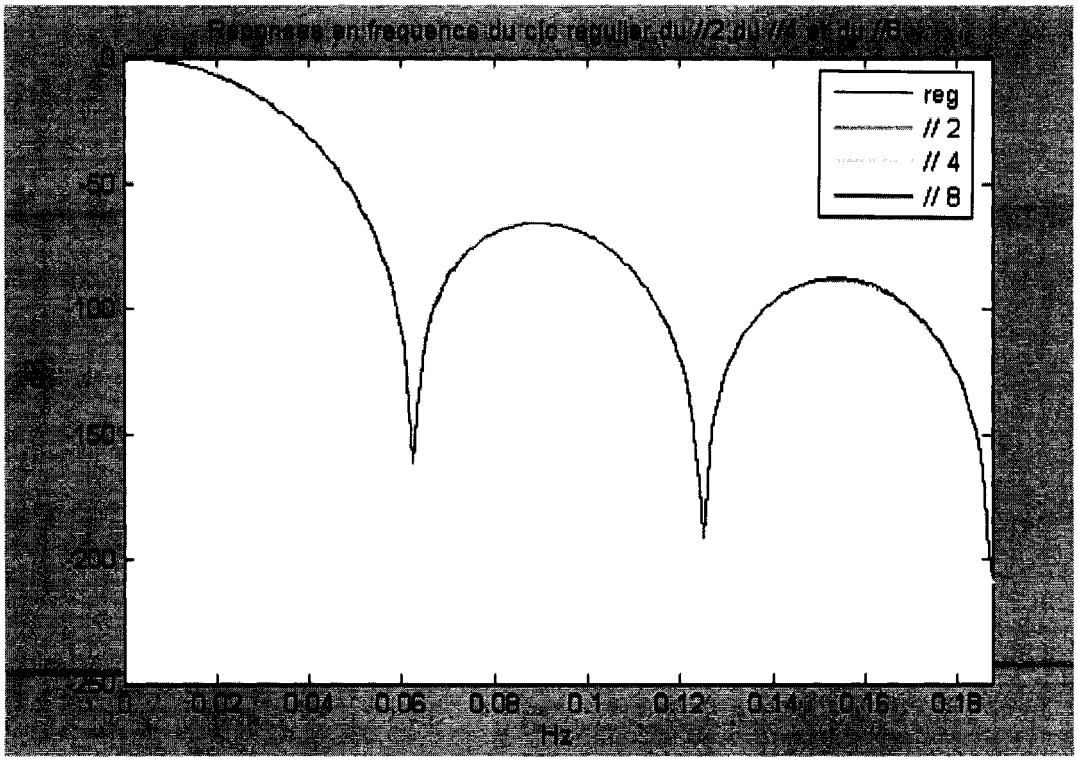


Figure 43 Comparaison des spectres normalisés du CIC régulier et du noyau programmable avec 2, 4 et 8 entrées parallèles.

Finalement, pour terminer la vérification du noyau programmable, les caractéristiques des filtres CIC décimateurs tels que présentés dans l'article de base de Hogenauer sont comparés à ceux du noyau programmable. Le tableau VIII présente la comparaison au niveau de l'atténuation dans la bande passante et le tableau IX la comparaison au niveau de l'atténuation dans la bande image/recouvrement.

Tableau VIII

Atténuation dans la bande passante du CIC régulier et du CIC polyphasé avec 2, 4, et 8 entrées parallèles

Largeur de bande relative à la basse fréquence (f_c)	Filtre CIC	Atténuation (en dB) dans la bande passante à la fréquence (f_c) en fonction de l'ordre du filtre N					
		1	2	3	4	5	6
1/128	CIC_régulier	0.00	0.00	0.00	0.00	0.00	0.01
	CIC_2-phase	0.00	0.00	0.00	0.00	0.00	0.00
	CIC_4-phase	0.00	0.00	0.00	0.00	0.00	0.00
	CIC_8-phase	0.00	0.00	0.00	0.00	0.00	0.00
1/64	CIC_régulier	0.00	0.01	0.01	0.01	0.02	0.02
	CIC_2-phase	0.00	0.00	0.01	0.01	0.01	0.02
	CIC_4-phase	0.00	0.00	0.01	0.01	0.01	0.02
	CIC_8-phase	0.00	0.00	0.01	0.01	0.01	0.02
1/32	CIC_régulier	0.01	0.03	0.04	0.06	0.07	0.08
	CIC_2-phase	0.01	0.02	0.04	0.05	0.07	0.08
	CIC_4-phase	0.01	0.02	0.04	0.05	0.07	0.08
	CIC_8-phase	0.01	0.02	0.04	0.05	0.07	0.08

Tableau VIII (suite)

Largeur de bande relative à la basse fréquence (f_c)	Filtre CIC	Atténuation (en dB) dans la bande passante à la fréquence (f_c) en fonction de l'ordre du filtre N					
		1	2	3	4	5	6
1/16	CIC_régulier	0.06	0.11	0.17	0.22	0.28	0.34
	CIC_2-phase	0.05	0.11	0.16	0.22	0.28	0.33
	CIC_4-phase	0.05	0.11	0.16	0.22	0.28	0.33
	CIC_8-phase	0.05	0.11	0.16	0.22	0.28	0.33
1/8	CIC_régulier	0.22	0.45	0.67	0.90	1.12	1.35
	CIC_2-phase	0.22	0.44	0.67	0.89	1.12	1.34
	CIC_4-phase	0.22	0.45	0.67	0.89	1.12	1.34
	CIC_8-phase	0.22	0.44	0.67	0.89	1.12	1.34
1/4	CIC_régulier	0.91	1.82	2.74	3.65	4.56	5.47
	CIC_2-phase	0.91	1.82	2.73	3.64	4.56	5.47
	CIC_4-phase	0.91	1.82	2.73	3.64	4.56	5.47
	CIC_8-phase	0.91	1.82	2.73	3.64	4.55	5.46

Tableau IX

Atténuation dans la bande image/recouvrement du CIC régulier et du CIC polyphasé avec 2, 4, et 8 entrées parallèles

Délai différentiel (M)	Bande passante relative	Filtre CIC	Atténuation (en dB) dans la bande image/recouvrement à la fréquence f_A					
			1	2	3	4	5	6
1	1/128	CIC_régulier	42.1	84.2	126.2	168.3	210.4	252.5
		CIC_2-phase	42.1	84.2	126.2	168.3	210.4	252.5
		CIC_4-phase	42.1	84.2	126.2	168.3	210.4	252.5
		CIC_8-phase	42.1	84.2	126.2	168.3	210.4	252.5
1	1/64	CIC_régulier	36.0	72.0	108.0	144.0	180.0	215.9
		CIC_2-phase	36.0	72.0	108.0	144.0	180.0	215.9
		CIC_4-phase	36.0	72.0	108.0	144.0	180.0	215.9
		CIC_8-phase	36.0	72.0	108.0	144.0	180.0	215.9
1	1/32	CIC_régulier	29.8	59.7	89.5	119.4	149.2	179.0
		CIC_2-phase	29.8	59.7	89.5	119.4	149.2	179.0
		CIC_4-phase	29.8	59.7	89.5	119.4	149.2	179.0
		CIC_8-phase	29.8	59.7	89.5	119.4	149.2	179.0
1	1/16	CIC_régulier	23.6	47.2	70.7	94.3	117.9	141.5
		CIC_2-phase	23.6	47.2	70.7	94.3	117.9	141.5
		CIC_4-phase	23.6	47.2	70.7	94.3	117.9	141.5
		CIC_8-phase	23.6	47.2	70.7	94.3	117.9	141.5
1	1/8	CIC_régulier	17.1	34.3	51.4	68.5	85.6	102.8
		CIC_2-phase	17.1	34.3	51.4	68.5	85.6	102.8
		CIC_4-phase	17.1	34.3	51.4	68.5	85.6	102.8
		CIC_8-phase	17.1	34.3	51.4	68.5	85.6	102.8

Tableau IX (suite)

Délai différentiel (M)	Bande passante relative	Filtre CIC	Atténuation (en dB) dans la bande image/recouvrement à la fréquence f_A					
			1	2	3	4	5	6
1	1/4	CIC_régulier	10.5	20.9	31.4	41.8	52.3	62.7
		CIC_2-phase	10.5	20.9	31.4	41.8	52.3	62.7
		CIC_4-phase	10.5	20.9	31.4	41.8	52.3	62.7
		CIC_8-phase	10.5	20.9	31.4	41.8	52.3	62.7
2	1/256	CIC_régulier	48.1	96.3	144.4	192.5	240.7	288.8
		CIC_2-phase	48.1	96.3	144.4	192.5	240.7	288.8
		CIC_4-phase	48.1	96.3	144.4	192.5	240.7	288.8
		CIC_8-phase	48.1	96.3	144.4	192.5	240.7	288.8
2	1/128	CIC_régulier	42.1	84.2	126.2	168.3	210.4	252.5
		CIC_2-phase	42.1	84.2	126.2	168.3	210.4	252.5
		CIC_4-phase	42.1	84.2	126.2	168.3	210.4	252.5
		CIC_8-phase	42.1	84.2	126.2	168.3	210.4	252.5
2	1/64	CIC_régulier	36.0	72.0	108.0	144.0	180.0	216.0
		CIC_2-phase	36.0	72.0	108.0	144.0	180.0	216.0
		CIC_4-phase	36.0	72.0	108.0	144.0	180.0	216.0
		CIC_8-phase	36.0	72.0	108.0	144.0	180.0	216.0
2	1/32	CIC_régulier	29.9	59.8	89.6	119.5	149.4	179.3
		CIC_2-phase	29.9	59.8	89.6	119.5	149.4	179.3
		CIC_4-phase	29.9	59.8	89.6	119.5	149.4	179.3
		CIC_8-phase	29.9	59.8	89.6	119.5	149.4	179.3

Tableau IX (suite)

Délai différentiel (M)	Bande passante relative	Filtre CIC	Atténuation (en dB) dans la bande image/recouvrement à la fréquence f_A					
			1	2	3	4	5	6
2	1/16	CIC_régulier	23.7	47.5	71.2	95.0	118.7	142.5
		CIC_2-phase	23.7	47.5	71.2	95.0	118.7	142.5
		CIC_4-phase	23.7	47.5	71.2	95.0	118.7	142.5
		CIC_8-phase	23.7	47.5	71.2	95.0	118.7	142.5
2	1/8	CIC_régulier	17.8	35.6	53.4	71.3	89.1	106.9
		CIC_2-phase	17.8	35.6	53.4	71.3	89.1	106.9
		CIC_4-phase	17.8	35.6	53.4	71.3	89.1	106.9
		CIC_8-phase	17.8	35.6	53.4	71.3	89.1	106.9

Les tableaux VIII et IX montrent que, pour les mêmes spécifications, le noyau programmable possède les mêmes caractéristiques que le CIC régulier. Ceci nous permet de conclure que le noyau programmable fonctionne bien. La comparaison, aux niveaux des ressources utilisées et de la fréquence maximale de fonctionnement du noyau programmable avec les noyaux qui existent sur le marché sera l'objet du chapitre 5.

4.6 Considération pour la synthèse et l'optimisation

Synthétiser un design consiste à traduire une description textuelle d'une fonction en une interconnexion de modules physiques. Décrire une fonction en VHDL pour la rendre correctement synthétisable n'est pas trivial et doit répondre à une certaine rigueur si on veut garantir une certaine qualité de résultat car le design synthétisable va se transformer en silicium et soumis à toutes sortes d'influences externes (températures, tension,

humidité .....). Il est évident que ce n'est pas parce qu'une fonction peut être décrite en VHDL qu'elle sera synthétisable.

4.6.1 Choix du Package

Le VHDL permet de définir un nombre illimité de types de signaux, avec leur arithmétique et leur logique propre. Cependant, dans l'intérêt de la clarté du code, la logique standard, celle définie dans le package IEEE 1164 est priorisée. Ce package contient la description du type élémentaire des signaux logiques.

4.6.2 Opérateurs

Les synthétiseurs fournissent un certain nombre de packages qui contiennent souvent une description de fonctions arithmétiques plus ou moins complexes. Ces modules sont généralement optimisés pour une technologie particulière avec des critères tels que la surface, la consommation ou la vitesse. Ainsi, pour garder la portabilité du code VHDL, qui est un des buts de notre travail, les fonctions linéaires (additions, soustractions) sont implémentées avec le package *IEEE.std_logic_signed*.

4.6.3 Généricité

La généricité permet de décrire un code paramétrable dont la configuration peut être choisie soit au moment de la synthèse, soit dans l'instanciation des composants. Cette possibilité, bien que formidable en théorie, crée souvent des problèmes lors de la synthèse. Les synthétiseurs ne comprenant souvent qu'un sous-ensemble du VHDL, il arrive parfois, surtout pour les outils bon marché, que ceux-ci refusent le code contenant des paramètres génériques. Les outils plus évolués, quand à eux, acceptent normalement la généricité mais celle-ci demande des manipulations supplémentaires. Ainsi, pour

limiter les paramètres génériques dans ce travail, les fonctionnalités simples sont décrites de manière directe (c'est-à-dire sans généricité).

4.6.4 Code concurrent ou séquentiel

Le VHDL permet de décrire le code de deux manières distinctes : de manière concurrente ou séquentielle. Le code séquentiel correspond aux *PROCESS* et le code concurrent à la logique hors des *PROCESS*. Le code séquentiel a tendance à créer des *PROCESS* de plus en plus gros avec des *if* et des *case* imbriqués, rendant le code moins robuste et illisible. De plus, cette complexité logique engendre souvent des difficultés lors de la synthèse car l'outil étant incapable de regrouper efficacement les fonctions. Étant donné que les fonctionnalités décrites de manière concurrente sont souvent découpées en petites fonctions élémentaires plus proches du matériel [Thierry Schneider], le code concurrent est utilisé là où c'est possible.

4.6.5 Optimisation du timing

L'optimisation du *timing* constitue souvent le premier pas pour augmenter les performances du circuit final. Pour cela, on décrit les contraintes temporelles du circuit pour que le synthétiseur sache où concentrer son effort. Étant donné que le design est synchrone, en spécifiant une contrainte de fréquence, le synthétiseur dispose du temps maximum à allouer à la logique combinatoire.

4.6.6 Partage de ressources

Un design VHDL comprend généralement trois parties différentes : des mémoires, des fonctions combinatoires et des fonctions arithmétiques. Dans ce dernier cas, on peut économiser beaucoup de surface et de consommation si on parvient à partager des ressources matérielles entre plusieurs processus. Le gain en surface obtenu en optimisant

manuellement une fonction, par rapport à une optimisation automatique, ne justifie pas une telle dépense de temps. Les synthétiseurs effectuent efficacement cette tâche. Par contre, les générations actuelles d'outils de synthèse rencontrent des difficultés pour détecter certaines duplications de fonction. Pour éviter ce cas de figure, toutes les fonctions arithmétiques sont décrites de façon explicite.

4.7 Conclusion

La réalisation et la vérification du noyau programmable ont été l'objet de ce chapitre. Ainsi, le code VHDL est divisé en trois grands *PROCESS* : un pour le FIR polyphasé, un pour le CIC régulier et un pour le contrôle. Chaque *PROCESS* est vérifié par un *testbench* spécial qui permet de l'isoler. Le noyau programmable au complet est vérifié, premièrement, au haut niveau en comparant ses spectres d'entrée et de sortie pour voir si le code VHDL fonctionne bien. En prenant le CIC régulier de Xilinx qui existe sur Matlab/Simulink comme référence, sa réponse en fréquence est comparée à celles du noyau programmable pour 2, 4 et 8 entrées parallèles. Pour terminer la vérification, les caractéristiques de base du CIC régulier présenté par Hogenauer sont comparés à ceux du noyau programmable. Cette dernière vérification nous a permis de conclure que le noyau programmable fonctionne bien. Pour terminer, étant donné que l'un des buts finaux de ce travail est d'avoir un noyau synthétisable, quelques considérations pour la synthèse ont été présentées.

CHAPITRE 5

COMPARAISON AVEC DES *CORES* COMMERCIAUX

5.1 Introduction

L'évolution rapide de la microélectronique met à la disposition des concepteurs de FPGA comme d'ASIC des puces intégrant de plus en plus de portes, permettant d'intégrer des systèmes de plus en plus complexes sur une même puce. Ainsi, en gardant un processus de conception identique à ce qui se fait à l'heure actuelle, il faudrait augmenter la taille des équipes de développement et ceci malgré l'évolution des outils et des méthodes. Toutefois, il est démontré qu'au-delà d'une dizaine d'ingénieurs travaillant sur une même puce, l'efficacité n'évolue plus de manière proportionnelle avec l'augmentation de la taille de l'équipe. Ainsi, comme le montre la figure 44, il apparaît une faille de productivité [24]. Sans évolution de la méthode de conception, l'exploitation des ressources disponibles sur une puce, en un temps de design raisonnable, devient impossible. Une méthodologie possible pour exploiter cette faille est le *design-reuse*, permettant de réutiliser dans un autre contexte des modules déjà conçus. Ceci explique la disponibilité de plusieurs noyaux (*core*) sur le marché. Ainsi, le but de ce chapitre, dans un premier temps, est de présenter les ressources utilisées et la fréquence d'opération du CIC polyphasé. Dans un deuxième temps, un aperçu sur les *cores* disponibles sur le marché sera présenté. Enfin, une comparaison des *cores* de CIC au niveau des ressources utilisées et de la fréquence d'opération sera faite.

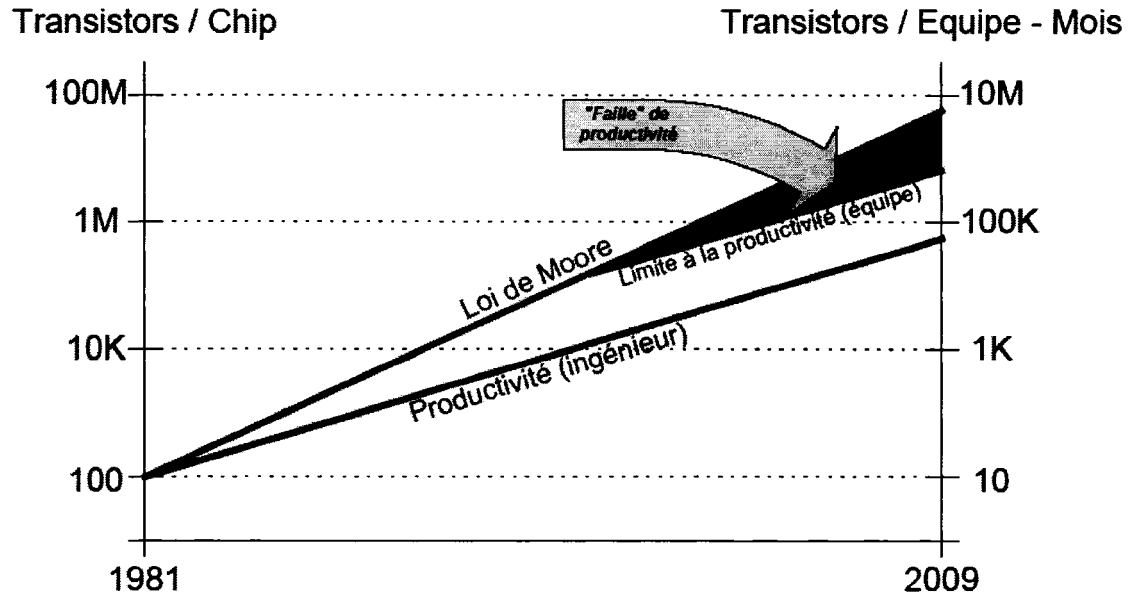


Figure 44 Faille de productivité [24]

5.2 Estimation des ressources requises par le CIC polyphasé

La plate-forme choisie pour tester l'implémentation est une puce programmable de la famille VirtexII Pro de Xilinx (Xilinx 2004). Ces FPGA possèdent des ressources logiques sous forme d'unités de base nommées *slices*. Chaque *slice* contient notamment deux générateurs de fonctions à quatre entrées, une chaîne de retenue, des multiplexeurs et deux registres. Chaque générateur de fonction peut être programmé soit en une table de conversion (LUT, Look Up Table) à quatre entrées, soit en 16 bits de mémoire ou soit comme un registre à décalage de 16 bits. De plus, le FPGA possède des ressources dédiées telles que les multiplicateurs ayant une largeur d'entrée de 18 bits (BMULT). Une bascule peut être activée environ au centre de la logique de multiplication lorsque l'option « pipeline » est sélectionnée. Cette option permet d'accélérer la fréquence d'horloge et introduit un cycle de latence. De la mémoire embarquée est aussi disponible sous forme de *Block RAM* (BRAM) de taille 18 kbits. Ces blocs sont configurables en

mode simple ou double ports et utilisent des mots d'entrée de 1, 2, 4, 18 ou 36 bits. Par exemple, on peut avoir une RAM 18x1024 ou 36x512.

Traditionnellement, les résultats en termes de ressources requises pour l'implémentation dans ces puces sont exposés par le nombre de *slices*, de blocs de mémoire RAM (BRAM) et de multiplicateurs dédiés (BMULT). Des processeurs PowerPC sont aussi présents dans les VietexII Pro. Cependant, les blocs mémoire RAM, les multiplicateurs dédiés et les processeurs PowerPC ne sont pas utilisés par le CIC polyphasé.

Pour l'étape de la synthèse du code VHDL, le logiciel Synplify Pro 7.7 de Synplicity a été employé. Aucun paramètre dans l'interface usager de Synplify n'a été sélectionnée parmi les options disponibles : *pipelining, retiming, ressource sharing, modular design et FSM compiler.* Ces optimisations n'occasionnent pas de modifications sur les résultats finaux, mais augmente le temps requis pour effectuer la synthèse.

Pour la plupart des scénarios de vérification, le FPGA choisi est le XC2VP20-7. Le suffixe -7 indique la performance sur le plan de la vitesse. En synthétisant avec un suffixe différent, on obtient une contrainte d'horloge plus restrictive. Lorsqu'on choisit un autre FPGA de la famille VirtexII Pro suffisamment grand pour contenir le design, on obtient les mêmes besoins en ressources. La seule contrainte imposée concerne la période de l'horloge qui est fixée à 5 ns (200Mhz) dans le cas d'une puce -7. Le rapport de synthèse nous donne les ressources sous forme du nombre de LUT, de registres, de BMULTS et de BRAM.

Finalement, les étapes de placement et de routage ont été effectuées avec l'outil de Xilinx ISE v7.1i avec les options suivantes sélectionnées :

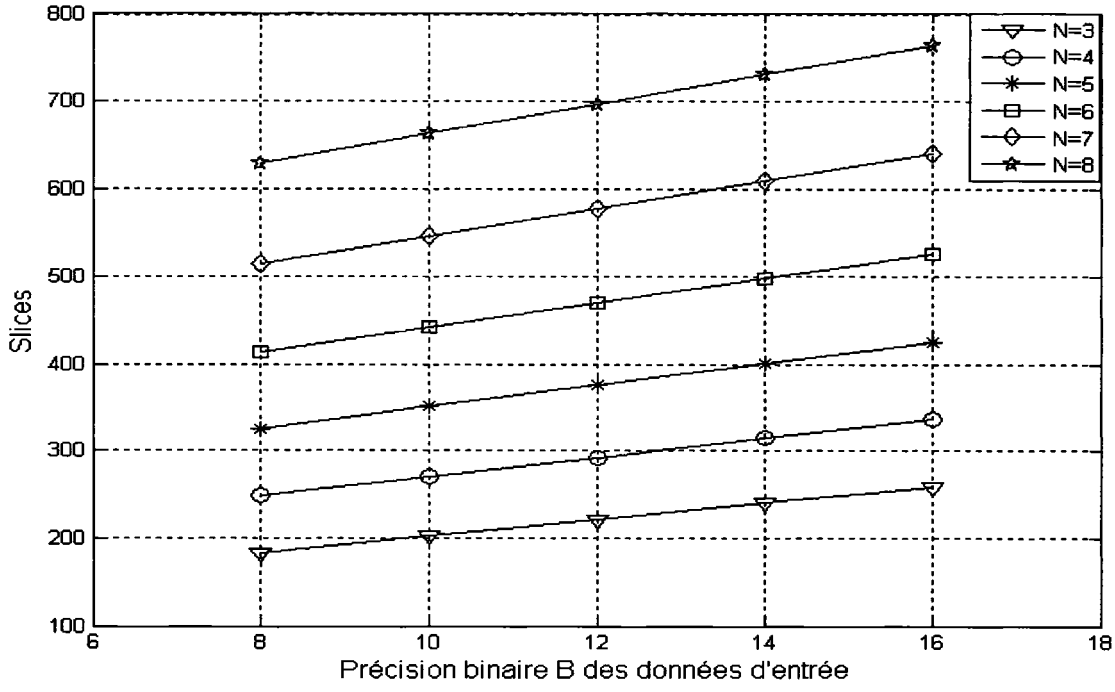
- *optimization Strategy : speed*
- *Perform Timing-Driven Packing : yes*
- *Place & Route Effort Level : high*

Tableau X présente les ressources, en terme de *slices*, requises par le *core* en fonction de l'ordre N du filtre et de la largeur binaire B des données d'entrée. Ce tableau est généré pour un filtre CIC polyphasé avec un facteur de décimation $R = 32$. La représentation graphique de ce tableau est présentée à la figure 45.

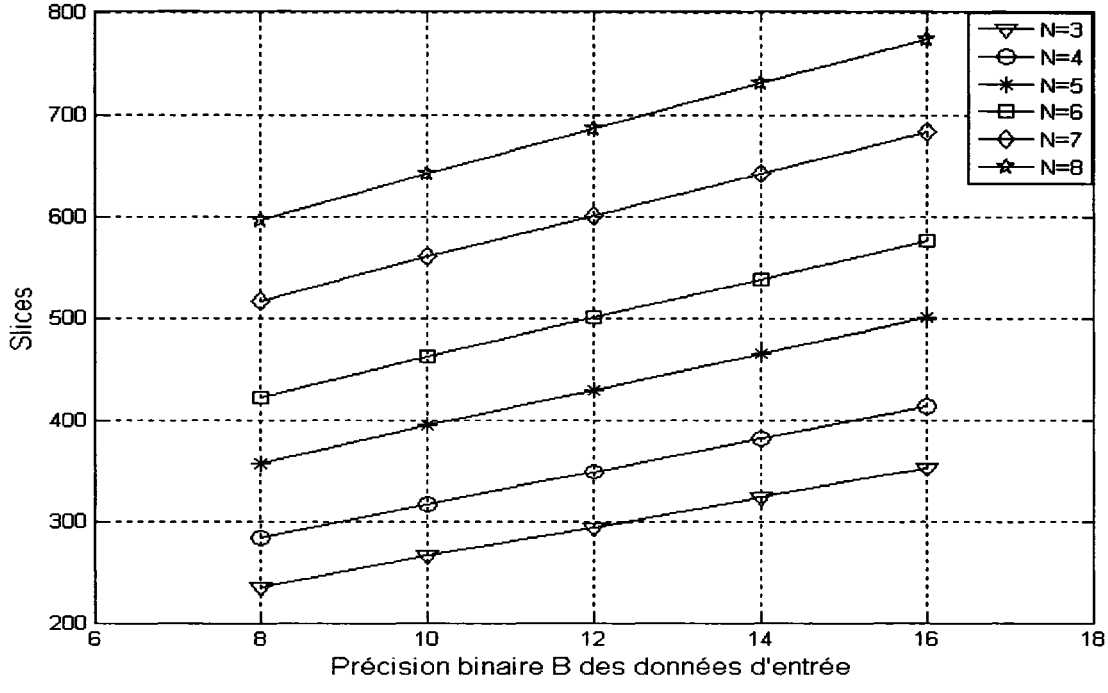
Tableau X

Nombre de *slices* du *core* en fonction de l'ordre N du filtre et de la largeur binaire B des données d'entrée; $R = 32$.

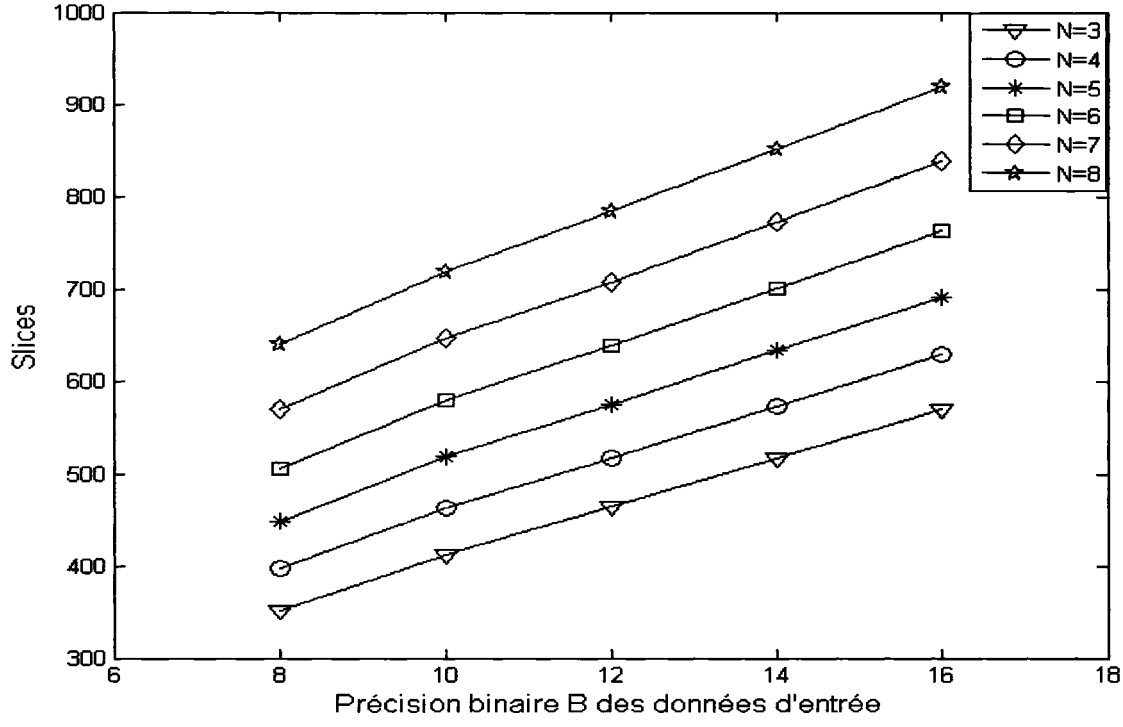
N	Nombre de *slices*					
		B = 8	B = 10	B = 12	B = 14	B = 16
3	CIC_2-phase	184	203	222	241	260
	CIC_4-phase	235	266	294	323	353
	CIC_8-phase	351	413	465	518	570
4	CIC_2-phase	249	271	293	315	337
	CIC_4-phase	283	317	349	382	414
	CIC_8-phase	397	464	518	573	629
5	CIC_2-phase	326	351	376	401	426
	CIC_4-phase	358	395	430	465	500
	CIC_8-phase	448	519	575	634	692
6	CIC_2-phase	415	443	471	499	527
	CIC_4-phase	422	462	500	538	576
	CIC_8-phase	507	580	640	701	764
7	CIC_2-phase	516	547	578	609	640
	CIC_4-phase	517	560	601	642	683
	CIC_8-phase	571	647	709	774	839
8	CIC_2-phase	629	663	697	731	765
	CIC_4-phase	596	642	686	730	774
	CIC_8-phase	641	720	785	853	920



(a) CIC_2-phase



(b) CIC_4-phase



(c) CIC_8-phase

Figure 45 Nombre de slices en fonction de l'ordre *N* du filtre et de la précision binaire B des données d'entrée

Le nombre de *slices* occupés, lorsque le placement est effectué, est plus élevé que la moitié du plus grand nombre entre les LUT et les registres obtenus à la synthèse. Un *slice* comporte deux LUT et deux registres, mais l'outil de placement ne prend pas toutes les ressources disponibles dans chaque *slice* parce qu'il a suffisamment d'espace pour s'étendre dans la puce. On peut donc affirmer que la quantité de *slices* présentée est le pire cas de ressources requises. En réalité, moins de slices seraient nécessaires dans le cas de l'utilisation du design dans un FPGA plus rempli.

Les tableaux XI, XII et XIII présentent les ressources, en terme de *slices*, requises par le *core* en fonction de l'ordre *N* du filtre et de la largeur binaire B des données d'entrée.

Ces tables sont générées pour un filtre CIC polyphasé avec un facteur de décimation $R = 16$, $R = 48$ et $R = 4096$ respectivement.

Tableau XI

Nombre de *slices* du *core* en fonction de l'ordre N du filtre et de la largeur binaire B des données d'entrée; $R = 16$.

N	**Nombre de slices**					
		B = 8	B = 10	B = 12	B = 14	B = 16
3	CIC_2-phase	171	190	209	228	247
	CIC_4-phase	209	240	269	298	327
	CIC_8-phase	338	402	453	507	558
4	CIC_2-phase	217	239	261	285	307
	CIC_4-phase	252	287	318	350	382
	CIC_8-phase	365	430	485	541	594
5	CIC_2-phase	288	313	340	364	390
	CIC_4-phase	301	339	373	408	443
	CIC_8-phase	412	480	537	596	652
6	CIC_2-phase	350	378	406	434	463
	CIC_4-phase	357	397	434	472	510
	CIC_8-phase	444	512	573	639	694
7	CIC_2-phase	442	474	505	536	567
	CIC_4-phase	418	461	501	542	583
	CIC_8-phase	496	570	634	703	761
8	CIC_2-phase	519	553	587	621	655
	CIC_4-phase	485	531	574	618	662
	CIC_8-phase	529	606	672	740	812

Tableau XII

Nombre de *slices* du *core* en fonction de l'ordre N du filtre et de la largeur binaire B des données d'entrée; $R = 48$.

N	Nombre de slices					
		B = 8	B = 10	B = 12	B = 14	B = 16
3	CIC_2-phase	199	218	237	256	275
	CIC_4-phase	247	278	307	336	365
	CIC_8-phase	363	427	478	532	583
4	CIC_2-phase	283	305	327	349	371
	CIC_4-phase	315	349	381	413	445
	CIC_8-phase	428	494	549	604	661
5	CIC_2-phase	366	391	416	441	466
	CIC_4-phase	376	413	448	483	518
	CIC_8-phase	486	557	613	673	731
6	CIC_2-phase	461	489	517	545	573
	CIC_4-phase	465	505	543	581	619
	CIC_8-phase	551	625	685	746	809
7	CIC_2-phase	593	624	655	686	717
	CIC_4-phase	566	609	650	691	732
	CIC_8-phase	647	723	786	850	914
8	CIC_2-phase	715	749	783	817	851
	CIC_4-phase	679	725	769	813	857
	CIC_8-phase	726	805	869	940	1005

Tableau XIII

Nombre de *slices* du *core* en fonction de l'ordre N du filtre et de la largeur binaire B des données d'entrée; $R = 4096$.

N		Nombre de slices				
		B = 8	B = 10	B = 12	B = 14	B = 16
3	CIC_2-phase	333	352	371	390	409
	CIC_4-phase	371	402	431	460	489
	CIC_8-phase	500	563	615	668	721
4	CIC_2-phase	479	501	523	545	567
	CIC_4-phase	514	548	580	612	644
	CIC_8-phase	627	694	748	804	859
5	CIC_2-phase	674	699	724	749	774
	CIC_4-phase	687	724	759	794	829
	CIC_8-phase	797	867	925	983	1040
6	CIC_2-phase	883	911	939	967	995
	CIC_4-phase	890	930	968	1006	1044
	CIC_8-phase	975	1047	1108	1170	1231
7	CIC_2-phase	1147	1178	1209	1240	1271
	CIC_4-phase	1123	1166	1207	1248	1289
	CIC_8-phase	1202	1278	1341	1406	1470
8	CIC_2-phase	1419	1453	1487	1521	1555
	CIC_4-phase	1386	1432	1476	1520	1564
	CIC_8-phase	1428	1509	1576	1644	1711

Les tableaux ci-dessus montrent bien que le traitement parallèle qu'effectue le CIC polyphasé est très efficace du point vue ressources matérielles. Pour un peu plus de *slices*, le nombre d'entrées parallèles peut être doublé ou quadruplé. Ainsi il est

intéressant de remarquer que le nombre de *slices* nécessaires évolue plus rapidement avec une augmentation de l'ordre du filtre ou du facteur de décimation qu'avec une augmentation de la largeur binaire des données d'entrée.

5.3 Fréquence d'horloge

Les outils utilisés pour l'estimation des ressources requises permettent d'estimer la fréquence maximale de fonctionnement du *core*. Elle correspond à la fréquence de l'horloge unique qui dépend du taux d'entrées des échantillons dans le module. La contrainte imposée sur la période de l'horloge est de 5 ns (200 Mhz) dans le cas d'une puce -7. Aucun placement manuel n'a été effectué puisque le *core* doit être utilisable sans manipulation particulière par l'usager. Par contre, de meilleures performances de fréquence d'horloge ou de ressources pourraient être obtenues en contraignant le design dans des régions. Le tableau XIIII présente les performances du *core*. La fréquence de fonctionnement du *core* est f_{in}. Étant donné que ses entrées sont parallèles, alors il peut recevoir des données échantillonnées à la fréquence f_s. La fréquence f_s est obtenue en multipliant la fréquence de fonctionnement du *core* f_{in} par le nombre d'entrées parallèles, c'est-à-dire le nombre de phases.

Tableau XIV

Performance du CIC polyphasé dans un Virtex2P (-07 speed grade); N = 5, R = 48.

Performance f_{in} / f_s (MHZ)					
	B = 8	B = 10	B = 12	B = 14	B = 16
CIC_2-phase	300 / **600**	289 / **578**	285 / **570**	275 / **550**	266 / **532**
CIC_4-phase	208 / **832**	208 / **832**	206 / **824**	203 / **812**	202 / **808**
CIC_8-phase	180 / **1440**	173 / **1384**	173 / **1384**	165 / **1320**	165 / **1320**

5.4 Aperçu des *cores* disponibles sur le marché

Il existe plusieurs compagnies sur le marché qui fournissent des *cores*. Le fabriquant de FPGA Xilinx offre tout genre de *core* dans sa librairie *Logicore* accessible à l'aide de l'outil CORE Generator. Ce sont aussi ces modules qui sont instanciés dans le FPGA lorsque le *toolbox* de Simulink, Xilinx System Generator, est utilisé pour la conception. Parmi les *Logicores*, on retrouve un module de CIC reconfigurable. Cette section porte sur la performance et les ressources matérielles requises par ce core optimisé pour les Virtex et Virtex-II.

Les compagnies Lattice Semiconductor Corporation et Intersil proposent aussi, chacune, un *core* de CIC qui possède plusieurs paramètres configurables. En revanche, dans les sections suivantes, nous limiterons notre analyse au produit de Xilinx, puisque nous avons des données quantitatives permettant de le comparer au *core* CIC polyphasé.

5.4.1 Présentation des cores de CIC

La compagnie Intersil a présenté en juillet 2004 un *core* de filtre à décimation appelé HSP43220 [25]. Le HSP43220 est un filtre à décimation optimisé pour le filtrage des signaux à bande étroite. Il est implémenté en deux étages de décimation. La figure 46 montre le diagramme bloc du HSP43220.

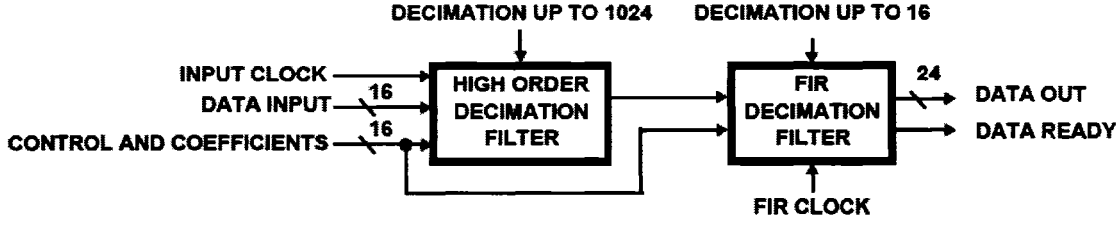
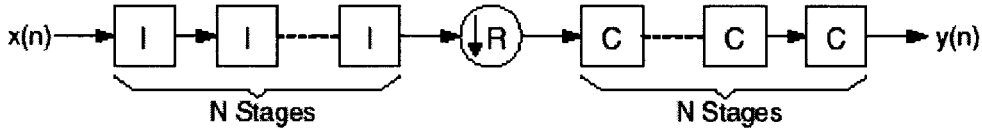


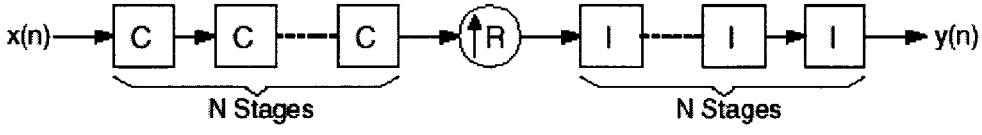
Figure 46 Diagramme bloc du HSP43220

Le premier étage de décimation est constitué d'un filtre à décimation de grand ordre (CIC d'ordre 5) qui utilise une technique de décimation efficiente permettant d'atteindre un facteur de décimation maximale de 1024 et une atténuation maximale de 96 dB. Le deuxième étage de décimation est constitué d'un filtre FIR à décimation avec un nombre de coefficients symétriques maximal de 512. Le facteur de décimation maximale du filtre FIR à décimation est égal à 16. La combinaison des deux facteurs de décimation donne le facteur de décimation total du filtre qui est égal à 16384. Le HSP43220 offre la possibilité de contourner le filtre à décimation de grand ordre ou le filtre FIR à décimation pour de la flexibilité additionnelle. Cependant, le HSP43220 est très limité du point de vue de la programmabilité. La largeur binaire des sonnées d'entrée et de sortie est fixée, respectivement, à 16 et 24 bits en complément à deux. La largeur binaire des coefficients du filtre FIR à décimation est fixée à 20 bits. Sa fréquence maximale de fonctionnement est de 33 MHz.

La Compagnie Lattice Semiconductor Corporation a présentée, en octobre 2005, un *core* de CIC [26]. Ce noyau peut être configuré pour être un CIC à décimation ou à interpolation. La figure 47 montre le CIC décimateur et interpolateur.



(a) CIC décimateur



(b) CIC interpolateur

Figure 47 CIC décimateur et interpolateur

Ce *core* est complètement paramétrable et offre les possibilités suivantes :

- Largeur binaire des données d'entrée : entre 1 et 32 bits
- Choix du type de filtre : décimation ou interpolation
- Facteur de décimation ou d'interpolation : fixe ou programmable
- Nombre d'étages (ordre du filtre) du CIC : entre 1 et 8
- Délai différentiel : entre 1 et 4
- Facteur de décimation : entre 2 et 16384
- Jusqu'à 4 canaux d'entrée pour la décimation ou l'interpolation

La largeur binaire des données de sortie de ce *core* n'est cependant pas programmable. Elle dépend de la largeur maximale de bus qui permet aux débordements à l'intérieur du circuit de ne pas affecter les résultats à la sortie du filtre. La fréquence maximale de fonctionnement de ce *core* est de l'ordre de 200 MHz.

5.4.2 Présentation du Logicore de CIC et des ressources utilisées

La compagnie Xilinx a présenté, en mars 2002, un *core* de CIC complètement paramétrable et qui offre les possibilités suivantes :

- Largeur binaire des données d'entrée : entre 1 et 32 bits
- Choix du type de filtre : décimation ou interpolation
- Facteur de décimation ou d'interpolation : fixe ou programmable
- Nombre d'étages (ordre du filtre) du CIC : entre 1 et 8
- Délai différentiel : entre 1 et 2
- Facteur de décimation varie entre 8 et 16383
- Jusqu'à 2 canaux d'entrée pour la décimation seulement

Pour assurer une grande fréquence de fonctionnement, le CIC décimateur est implémenté en utilisant une architecture pipeline présentée à la figure 48 [27]. Les registres de pipeline P0, P1, P2 et P3 permettent de couper la longue chaîne d'additionneurs causée par les différentiateurs en cascade. Ceci permet d'avoir un seul additionneur entre deux registres. Donc le chemin critique de cette architecture est constitué d'un additionneur. Cette technique permet d'augmenter la fréquence de fonctionnement du filtre.

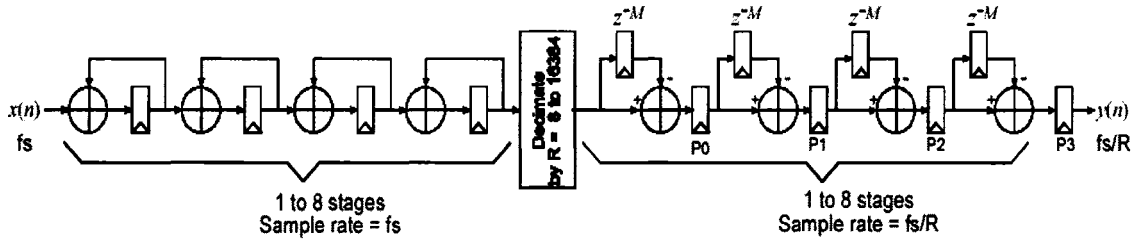


Figure 48 Architecture pipeline du CIC décimateur de Xilinx [27]

En échangeant le bloc d'intégrateurs avec celui de différentiateurs, comme présenté à la figure 49, le *core* devient un CIC interpolateur.

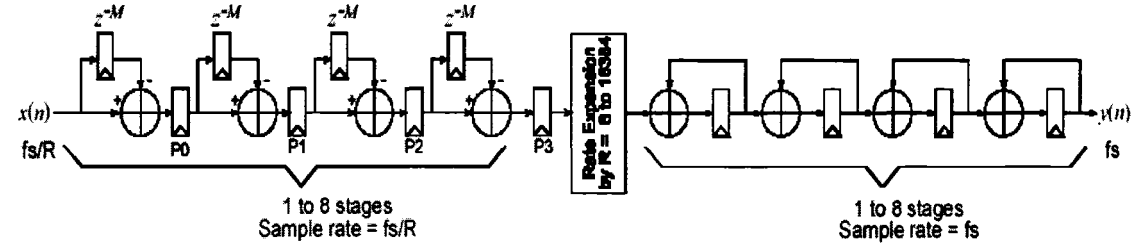


Figure 49 Architecture pipeline du CIC interpolateur de Xilinx [27]

La largeur binaire des données de sortie de ce *core* n'est pas programmable non plus. Elle dépend de la largeur maximale de bus qui permet aux débordements à l'intérieur du

circuit de ne pas affecter les résultats à la sortie du filtre. Dans le cas d'une configuration à deux canaux, c'est-à-dire deux entrées différentes, la plupart des ressources matérielles utilisées est multiplexée dans le temps. À cause de ce multiplexage temporel, la fréquence de chaque donnée d'entrée est réduite d'un facteur de 2 pour le cas de deux canaux. Ceci veut dire que si la fréquence maximale du CIC décimateur est de 100 MHz, une décimation avec un seul canal peut supporter des données qui arrivent à 100 MHz. Dans le cas d'une décimation avec deux canaux, le filtre supporte deux données qui arrivent chacune à 50 MHz.

Les tableaux XV, XVI, XVII et XVIII présentent, pour les mêmes configurations, les ressources utilisées par le *Logicore* de CIC et celles utilisées par le CIC polyphasé.

Tableau XV

Comparaison du *Logicore* de CIC versus le CIC polyphasé pour R = 16

N	**Nombre de slices**				
		B = 8		B = 12	B = 16
3	Xilinx_CIC	100		118	136
	CIC_2-phase	171		209	247
	CIC_4-phase	209		269	327
	CIC_8-phase	338		453	558
4	Xilinx_CIC	156		180	204
	CIC_2-phase	217		261	307
	CIC_4-phase	252		318	382
	CIC_8-phase	365		485	594
5	Xilinx_CIC	224		254	284
	CIC_2-phase	288		340	390
	CIC_4-phase	301		373	443
	CIC_8-phase	412		537	652

Tableau XV (suite)

N	Nombre de slices				
		B = 8		B = 12	B = 16
6	Xilinx_CIC	304		340	376
	CIC_2-phase	350		406	463
	CIC_4-phase	357		434	510
	CIC_8-phase	444		573	694
7	Xilinx_CIC	396		438	480
	CIC_2-phase	442		505	567
	CIC_4-phase	418		501	583
	CIC_8-phase	496		634	761
8	Xilinx_CIC	500		548	596
	CIC_2-phase	519		587	655
	CIC_4-phase	485		574	662
	CIC_8-phase	529		672	812

Tableau XVI

Comparaison du *Logicore* de CIC versus le CIC polyphasé pour R = 32

N	Nombre de slices				
		B = 8		B = 12	B = 16
3	Xilinx_CIC	111		129	147
	CIC_2-phase	184		222	260
	CIC_4-phase	235		294	353
	CIC_8-phase	351		465	570

Tableau XVI (suite)

N	Nombre de slices				
		B = 8		B = 12	B = 16
4	Xilinx_CIC	182		206	230
	CIC_2-phase	249		293	337
	CIC_4-phase	283		349	414
	CIC_8-phase	397		518	629
5	Xilinx_CIC	256		286	316
	CIC_2-phase	326		376	426
	CIC_4-phase	358		430	500
	CIC_8-phase	448		575	692
6	Xilinx_CIC	360		396	432
	CIC_2-phase	415		471	527
	CIC_4-phase	422		500	576
	CIC_8-phase	507		640	764
7	Xilinx_CIC	461		503	545
	CIC_2-phase	516		578	640
	CIC_4-phase	517		601	683
	CIC_8-phase	571		709	839
8	Xilinx_CIC	598		646	694
	CIC_2-phase	629		697	765
	CIC_4-phase	596		686	774
	CIC_8-phase	641		785	920

Tableau XVII

Comparaison du *Logicore* de CIC versus le CIC polyphasé pour R = 48

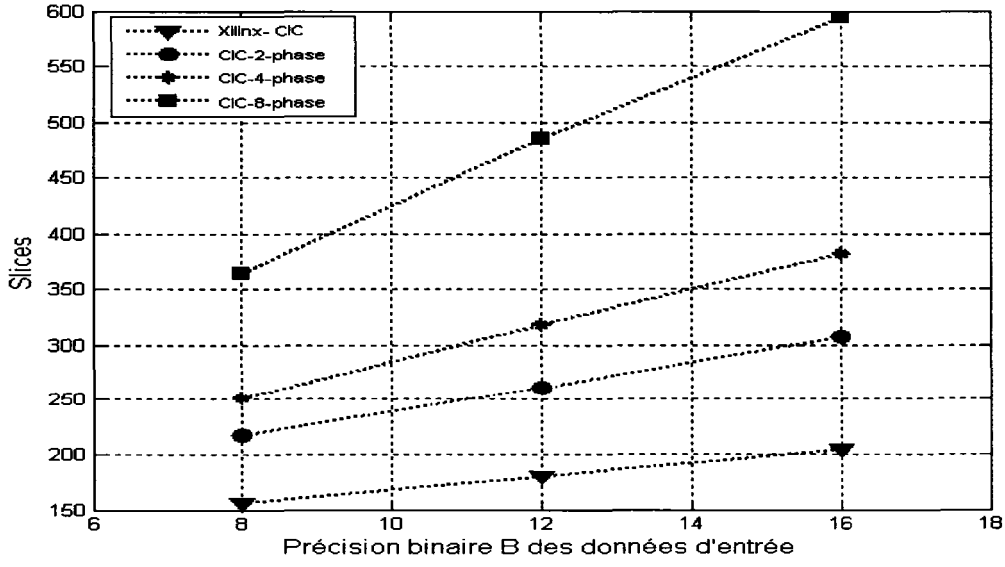
N	Nombre de slices				
		B = 8		B = 12	B = 16
3	Xilinx_CIC	120		138	156
	CIC_2-phase	199		237	275
	CIC_4-phase	247		307	365
	CIC_8-phase	363		478	583
4	Xilinx_CIC	194		218	242
	CIC_2-phase	283		327	371
	CIC_4-phase	315		381	445
	CIC_8-phase	428		549	661
5	Xilinx_CIC	286		316	346
	CIC_2-phase	366		416	466
	CIC_4-phase	376		448	518
	CIC_8-phase	486		613	731
6	Xilinx_CIC	396		432	468
	CIC_2-phase	461		517	573
	CIC_4-phase	465		543	619
	CIC_8-phase	551		685	809
7	Xilinx_CIC	524		566	608
	CIC_2-phase	593		655	717
	CIC_4-phase	566		650	732
	CIC_8-phase	647		786	914
8	Xilinx_CIC	646		694	742
	CIC_2-phase	715		783	851
	CIC_4-phase	679		769	857
	CIC_8-phase	726		869	1005

Tableau XVIII

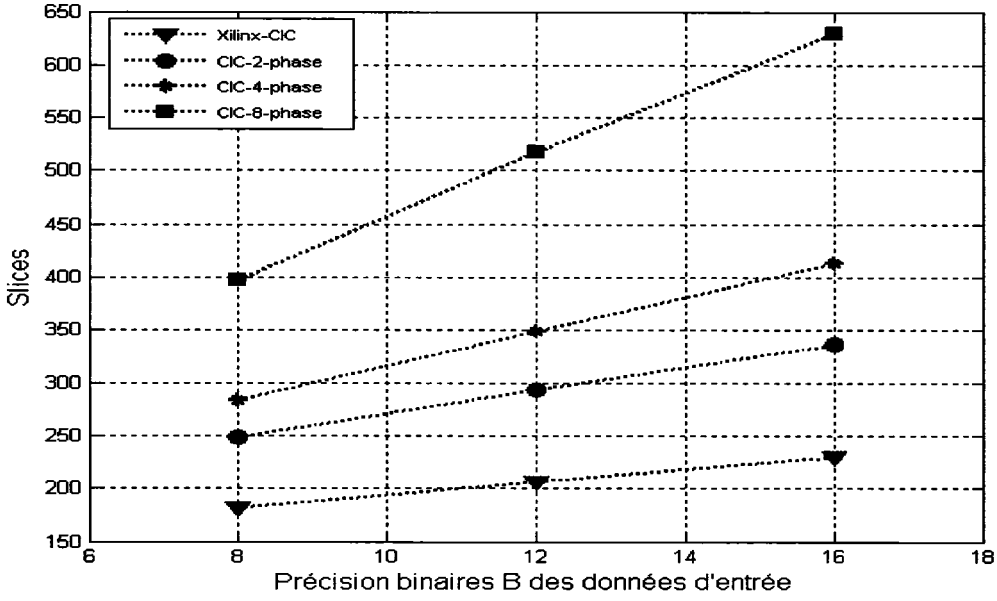
Comparaison du *Logicore* de CIC versus le CIC polyphasé pour R = 4096

N	Nombre de slices				
		B = 8		B = 12	B = 16
3	Xilinx_CIC	214		232	250
	CIC_2-phase	333		371	409
	CIC_4-phase	371		431	489
	CIC_8-phase	500		615	721
4	Xilinx_CIC	354		378	402
	CIC_2-phase	479		523	567
	CIC_4-phase	514		580	644
	CIC_8-phase	627		748	859
5	Xilinx_CIC	890		940	990
	CIC_2-phase	674		724	774
	CIC_4-phase	687		759	829
	CIC_8-phase	797		925	1040
6	Xilinx_CIC	1246		1306	1366
	CIC_2-phase	883		939	995
	CIC_4-phase	890		968	1044
	CIC_8-phase	975		1108	1231
7	Xilinx_CIC	1662		1732	1802
	CIC_2-phase	1147		1209	1271
	CIC_4-phase	1123		1207	1289
	CIC_8-phase	1202		1341	1470
8	Xilinx_CIC	2138		2218	2298
	CIC_2-phase	1419		1487	1555
	CIC_4-phase	1386		1476	1564
	CIC_8-phase	1428		1576	1711

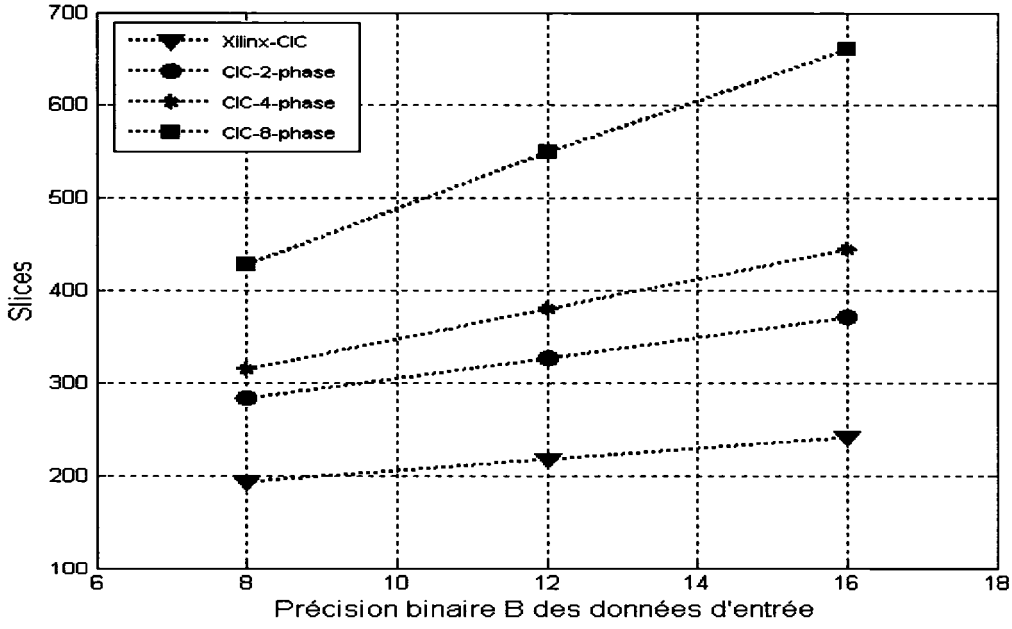
Les tableaux ci-dessus résument les différentes ressources utilisées par le *Logicore* et les comparent à des configurations semblables du CIC polyphasé. Les ressources présentées sont en fonction de l'ordre du filtre et de la précision binaire des données d'entrée. Il est important de remarquer que le CIC polyphasé nécessite toujours un peu plus de ressources que le CIC de Xilinx sauf dans le cas ou l'ordre du filtre et le facteur de décimation sont grands. Ceci est bien illustré par le tableau 9 pour N égale à 6, 7 ou 8. Ces résultats sont expliqués par le gain en largeur binaire du CIC polyphasé par rapport au CIC de Xilinx. En fait, comme expliqué à la section 3.5.2, la largeur maximale des bus du CIC décimateur polyphasé est plus courte que celle des bus du CIC régulier de $([N_2 \log_2 R_1] - [N_1 \log_2 R_1])$ bits. À titre d'exemple, si $N_1 = 2$ et $N_2 = 6$ alors la largeur binaire des bus du CIC_2-phase, CIC_4-phase et CIC_8-phase est plus courte que celle des bus du CIC régulier de 4, 8 et 12 bits respectivement. Ceci est dû, principalement, à la décimation multi-étage. Cette réduction de la largeur des registres est très intéressante car diminue les ressources requises par le CIC polyphasé. Pour mieux voir les ressources utilisées par ces deux noyaux programmables, les figures 50 et 51 présentent, pour quelques paramètres sélectionnés les résultats sous forme graphique.



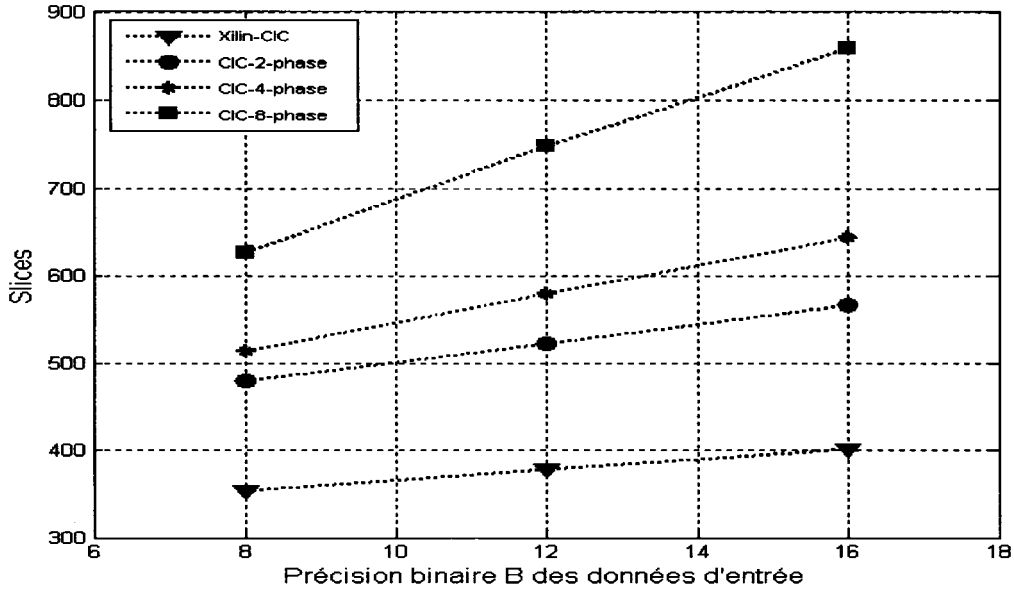
(a) R = 16


Xilinx-CIC
CIC-2-phase
CIC-4-phase
CIC-8-phase
Slices
650
600
550
500
450
400
350
300
250
200
150
6
8
10
12
14
16
18
Précision binaires B des données d'entrée

(b) R = 32

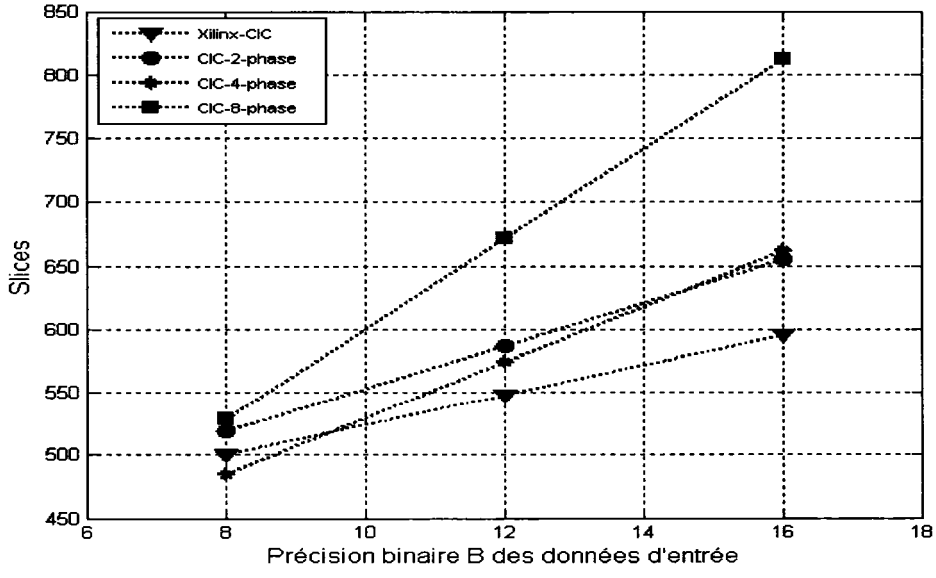

Xilinx-CIC
CIC-2-phase
CIC-4-phase
CIC-8-phase
Slices
700
600
500
400
300
200
100
6
8
10
12
14
16
18
Précision binaire B des données d'entrée

(c) R = 48

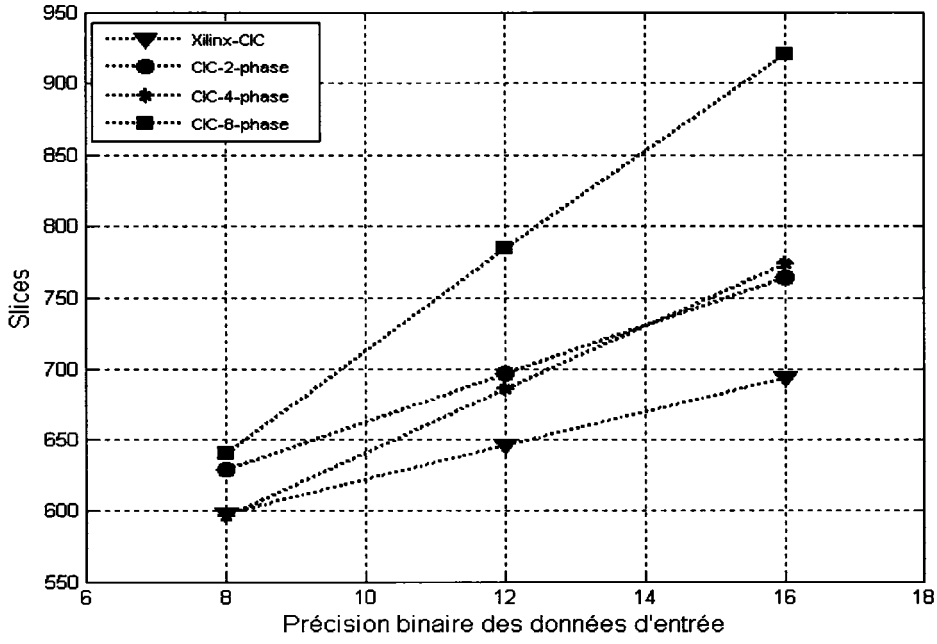


(d) R = 4096

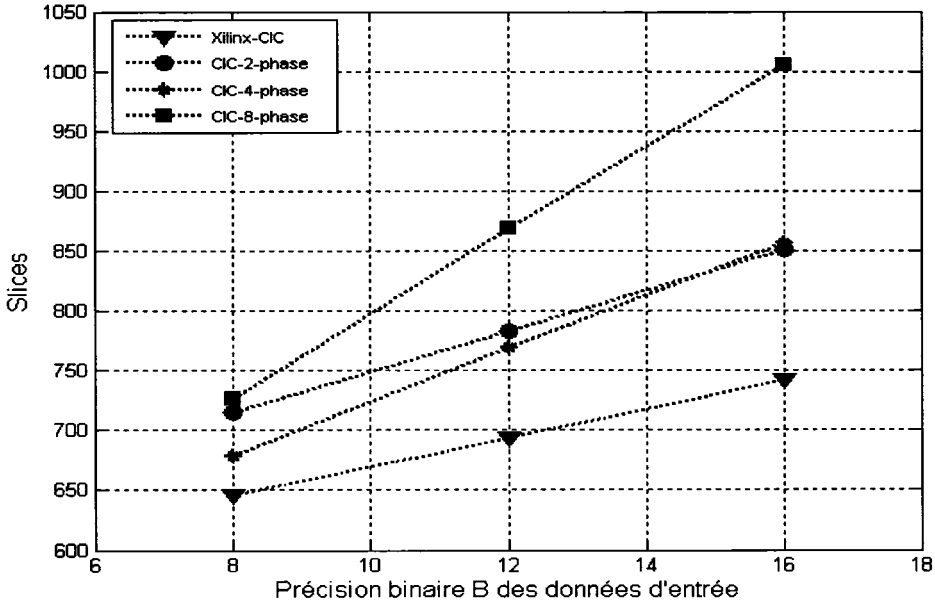
Figure 50 Comparaison des ressources FPGA requises par le CIC polyphasé et par le *Logicore* de CIC de Xilinx pour différentes précisions binaires des données d'entrée avec N = 4



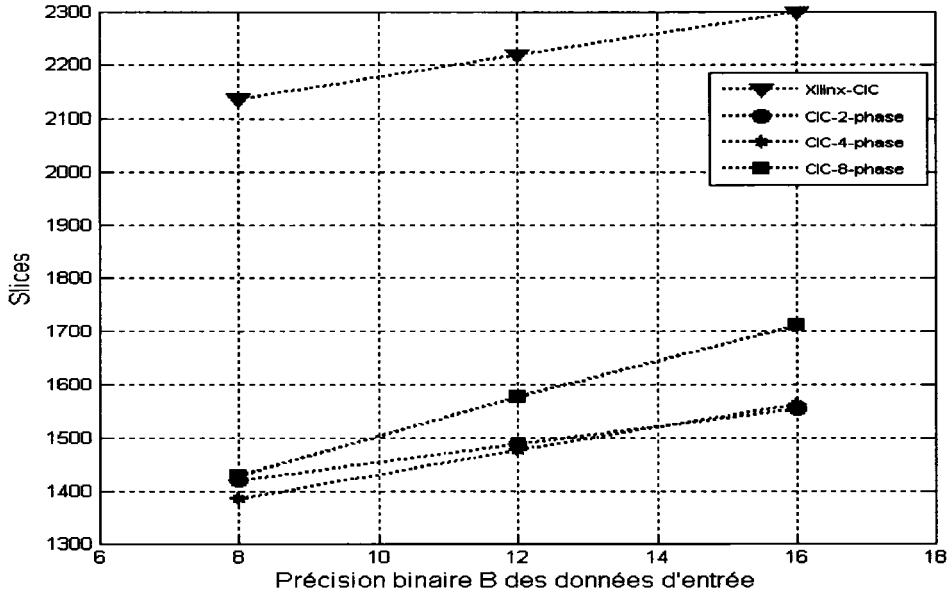
(a) R = 16


Xilinx-CIC
CIC-2-phase
CIC-4-phase
CIC-8-phase
Slices
Précision binaire des données d'entrée

(b) R = 32


Xilinx-CIC
CIC-2-phase
CIC-4-phase
CIC-8-phase
Slices
Précision binaire B des données d'entrée

(c) R = 48



(d) R = 4096

Figure 51 Comparaison des ressources FPGA requises par le CIC polyphasé et par le *Logicore* de CIC de Xilinx pour différentes précisions binaires des données d'entrée avec N = 8

5.4.3 Comparaison au niveau de la fréquence de fonctionnement

La comparaison des deux noyaux, au niveau de la fréquence de fonctionnement, est présentée par le tableau XIX. Ainsi, nous pouvons remarquer que le CIC polyphasé peut recevoir des données échantillonnées à des fréquences 2, 4 ou 8 fois plus grandes que celles du CIC de Xilinx pour juste un peu plus de *slices*. Par exemple, pour N = 4, R = 48 et B = 16, le CIC de Xilinx peut recevoir des données échantillonnées à la fréquence $f_s = 191$ MHz pour 242 *slices* alors que le CIC_4-phase peut recevoir des données à la fréquence $f_s = 936\,\text{MHz}$ pour 445 *slices*. Donc, le CIC_4-phase peut recevoir des données échantillonnées à une fréquence 4 fois plus grande que celle du CIC de Xilinx pour moins que le double des ressources requises par ce dernier.

Tableau XIX

Comparaison entre le CIC de Xilinx et le CIC_polyphasé au niveau de la fréquence de fonctionnement pour N = 4 et R = 48

Device	Performance f_{in} / f_s (MHZ)			
		B = 8	B = 12	B = 16
Virtex-E -08 speed grade	Xilinx_CIC	162/**162**	162/**162**	162/**162**
	CIC_2-phase	166/**332**	165/**330**	155/**310**
	CIC_4-phase	174/**696**	166/**664**	161/**644**
	CIC_8-phase	165/**1320**	146/**1168**	145/**1160**
Virtex-II -06 speed grade	Xilinx_CIC	208/**208**	203/**203**	191/**191**
	CIC_2-phase	259/**518**	248/**496**	234/**468**
	CIC_4-phase	260/**1040**	248/**992**	234/**936**
	CIC_8-phase	208/**1664**	194/**1552**	188/**1504**

5.5 Conclusion

L'estimation des ressources, la fréquence de l'horloge et une comparaison du CIC polyphasé avec les noyaux commerciaux ont été l'objet de ce chapitre. Ainsi du point de vue ressources, le CIC polyphasé présente des résultats très intéressants. Par exemple, le CIC_8-phase d'ordre N = 8, R = 4096 et B = 16 bits occupe 4% des ressources d'un VirtexII Pro (xc2vp100). Du point de vue de la fréquence de fonctionnement, le CIC polyphasé est capable de supporter des données échantillonnées à des fréquences de l'ordre des GHz. Ceci veut dire que le CIC polyphasé est capable de filtrer des données numériser directement en IF ou RF. Enfin, la comparaison avec les noyaux commerciaux a montré que le CIC polyphasé tire bien son épingle du jeu par rapport au compromis fréquence de fonctionnement / ressources requises. Pour R = 48, N = 5 et B = 8, le CIC polyphasé peut recevoir des données échantillonnées à une fréquence 8 fois

plus grande que celle du CIC de Xilinx pour moins que le double des ressources de ce dernier.

CONCLUSION

La conception des filtres numériques à décimation utilisant de hautes fréquences d'échantillonnage n'est pas une tâche facile. Pour cela, le point de départ a été basé sur des techniques qui permettent d'accroître la vitesse de calcul et de diminuer le nombre d'opérations arithmétiques. Ceci nous a poussé à nous orienter vers les structures parallèles qui sont réputées efficaces en traitement numérique de hautes cadences. Étant donné que la multiplication nécessite plus de capacité mémoire que l'addition, les structures qui en utilisent moins ou presque pas sont privilégiées. En tenant compte de ces critères, trois méthodes de conception de filtre multicadence ont été étudiées d'une façon détaillée.

La première méthode que nous avons considérée consiste à utiliser les filtres numériques FIR. Toutefois, la structure régulière de ces filtres ne permet pas d'utiliser de grandes fréquences d'échantillonnage. Ainsi, pour accélérer la vitesse de calcul des ces filtres, la décomposition polyphasée est utilisée. Cette technique fonctionne bien si le facteur de décimation est petit. Tout processus de décimation utilise un filtre passe bas avec une fréquence de coupure égale à $\frac{\pi}{R}$ pour limiter la largeur de bande du signal. Ainsi, si le facteur de décimation est grand, alors la largeur de bande du filtre est étroite. Sa bande de transition est préférablement étroite aussi. Ceci nécessite un filtre FIR de grand ordre ce qui engendre à son tour un grand nombre de multiplications effectuer. Dans ce cas de figure, le filtre FIR polyphasé ne peut pas fonctionner avec de grandes fréquences d'échantillonnage.

La deuxième méthode présentée dans ce travail consiste à utiliser les filtres RII. Ces filtres RII sont intéressants dans la mesure ou, pour les mêmes spécifications, ils utilisent un ordre beaucoup plus petit que celui des filtres FIR. Ceci veut dire qu'ils nécessitent moins de calculs. En plus, les filtres RII ayant le même ordre que les filtres RIF offrent

des performances beaucoup plus intéressantes. Cependant, la structure régulière des ces filtres ne nous permettent pas d'utiliser de grandes fréquences d'échantillonnage. Pour augmenter la vitesse de calcul, la technique de pipeline s'impose. Ceci n'est pas tâche facile car un filtre récursif a besoins de ses sorties antérieurs pour calculer la prochaine sortie. Ainsi, pour se faire, la technique 'Scattered lookahead' qui transforme un filtre à pôle avec une boucle de rétroaction de délai K en un filtre à pole et zéro avec une boucle de rétroaction dont le délai est un multiple de K est utilisée. Cette technique permet de pipeliner les additionneurs et les multiplicateurs dans la boucle de rétroaction. Cette technique permet d'augmenter le débit du filtre mais son point faible est qu'elle augmente la complexité de la section en amont du filtre.

La troisième méthode, quant à elle, consiste à utiliser les filtres CIC. Leur structure de base présente déjà un avantage qui réside dans le fait que leur réalisation ne nécessite aucune opération de multiplication. Toutefois, cette structure ne permet pas d'atteindre de hautes fréquences d'échantillonnage. Cette limitation provient principalement des complications dans l'enchaînement des calculs qu'apporte la mise en cascade d'étages. Il fallait donc trouver des structures parallèles surtout pour le premier étage. Ces considérations ont conduit à trois architectures parallèles. La première présente l'agencement des données en parallèle avec une latence de P+2. Cette architecture a été améliorée en effectuant une addition en arbre. Cette amélioration, qui est la seconde architecture parallèle présentée, a permis de réduire la latence à $\log_2(P)$. Malgré que la complexité ait été réduite, cette structure reste encombrante et moins flexible. La troisième architecture parallèle considérée utilise une décomposition polyphasée partielle et une décimation multiétage. La décimation multiétage permet de réduire la complexité tandis que la décomposition polyphasée partielle permet de supporter de grandes fréquences d'échantillonnage. La combinaison de ces performances fait la force de cette structure d'où son choix pour l'implémentation.

Les résultats de simulation obtenus ont confirmés l'efficacité de cette architecture. Elle présente de meilleures performances avec une complexité nettement réduite pour les hautes vitesses. Il faut souligner que l'implémentation de cette architecture est simple.

Ainsi, en résumé, le noyau programmable conçu est capable d'opérer à haute fréquence d'échantillonnage de l'ordre de 1.2 GHz en utilisant des outils de programmation disponibles tels que les FPGA. Du point de vu de la programmabilité, tous ses paramètres peuvent être définis par l'utilisateur. Ainsi, ce *core* peut être utilisé efficacement pour une démodulation quadratique ou pour des modulateurs sigma-delta ou ceux basés sur l'interlacement temporel, ainsi que pour plusieurs applications qui nécessitent une haute vitesse de traitement. Sur ce, nous pensons avoir atteint nos objectifs.

Publication

L'article [33], qui a été issu des travaux présentés dans le cadre de cette recherche, résume l'essentiel du contenu de ce mémoire et a été présenté a la conférence « International IEEE-NEWCAS Conference ».

BIBLIOGRAPHIE

[1] Chandrakasan, A.P.,Brodersan, R.W. (Apr.1995). Minimizing power consumption in digital CMOS circuits Proceedings of the IEEE, 83(4), 498-523.

[2] Rabiner, L., Gold, B. (c1975). Theory and application of digital signal processing (Prentice-Hall)

[3] Crochiere, R., L Rabiner, (Oct 1975). Optimum FIR digital filter implementations for decimation, interpolation, and narrow-band filtering. Acoustics, Speech, and Signal Processing 23(5), 444-456.

[4] Rabiner, L., Crochiere, R. (Oct 1975). A novel implementation for narrow-band FIR digital filters. Acoustics, Speech, and Signal Processing, 23(5), 457-464.

[5] Peled, A., Bede Liu. (Dec 1974). A new hardware realization of digital filters.Acoustics, Speech, and Signal Processing, 22(6), 456-462

[6] Goodman, D., Carey.M. (Apr 1977). Nine digital filters for decimation and interpolation. Acoustics, Speech, and Signal Processing, 25(2), 121-126

[7] Hogenauer, E. (Apr1981). An economical class of digital filters for decimation and interpolation. Acoustics, Speech, and Signal Processing, 29(2), 155-162

[8] Y.Djadi, T.A.Kwasniewski, C. Chan and V. Szwarc. A high throughput programmable decimation and interpolation filter. Proceeding of the 1994 international conference on Signal Processing Applications and Technology, pp.1743 – 1748

[9] A. Kwentus, O. Lee, and A.N. Wilson, Jr., A 250 Msample/sec Programmable Cascaded Integrator-Comb Decimation Filter, in Proceeding VLSI Signal Processing IX, pp.231- 240, Oct-Nov. 1996

[10] Kei-Yong Khoo, Zhan Yu and Alan N. Wilson, Jr., Efficient High-Speed CIC Decimation filter, Proceeding Eleventh Annual IEEE International ASIC Conference, pp. 251-254, June 1998.

[11] H.J.Oh, S.Kim, G.Choi, and Y.H.Lee, On the Use of Interpolated Second-order Polynomials for Efficient Filter Design in Programmable Downconversion, IEEE Journal on selected Areas in communications, Vol. 17, No. 4, pp. 551-560, April1999

[12] H.J. Oh and Y.H. Lee, Multiplierless FIR Filters Based on cyclotomic and Interpoladed Second-order Polynomials with Powers-of-two Coefficients, in Proceeding Midwest. Symp. Circuits Syst, Sacramento, CA, Aug. 1997

[13] Yonghong Gao, Lihong Jia and Hannu Tenhunen, An Improved Architecture and Implementation of Cascaded Integrator-Comb Decimation Filters, 1999 IEEE Pacific Rim Conference on communications, Computers and Signal Processing (PACRIM 1999), pp. 317-32s0, Victoria, B.C., Canada, August 1999.

[14] Frederic. J. Harris (c2004). Multirate Signal Processing For Communication SYSTEMS (Prentice Hall PTR)

[15] David B. Chester, Digital IF Filter Technology for 3G Systems: An Introduction, IEEE Communications Magazine, pp. 102-107, February 1999.

[16] Tamal Bose (c2004). DIGITAL SIGNAL AND IMAGE PROCESSING (WILEY)

[17] H. Martinez, T. Parks. New class of recursive digital filters for decimation Acoustics, Speech and Signal Processing, IEEE International Conference on ICASSP 78. Volume 3, Apr 1978, pages 508-511

[18] U.Meyer-Baese (c2004). Digital Signal Processing With Field Programmable Gate Arrays (springer Second Edition)

[19] Mourid. Btisam., Filtre à décimation parallélise, Ecole de Technologie Supérieure (Canada), 2003, 86 pages

[20] H. Martinez, T. Parks. A Class of Infinite-Duration Impulse Response Digital Filters for sampling Rate Reduction, IEEE transactions on Acoustics, Speech and Signal Processing 26(4), 154-162 (1979)

[21] H. Martinez, T. Parks. Design of recursive digital filters with optimum magnitude and attenuation poles on the unit circle, IEEE transactions on Acoustics, Speech and Signal Processing, Volume 26, Issue 2, Apr 1978 pages 150-156

[22] Motorola: Data sheet, DSPADC16 16-Bit Sigma-Delta Analog-to-Digital Converter

[23] Krukowski,A., Kale,I. (September 1996.). Decomposition of IIR transfer functions into parallel arbitrary-order IIR subfilters. IEEE Nordic Signal Processing Symposium (NORSIG 96), Dipoli Convension Center and Department of Electrical Engineering Espoo, Finland

[24] Ludovic, Loiseau (2001). Le Design-reuse ("Réutilisabilité") Pour la Conception de FPGA à partir du Langage VHDL, École polytechnique de Montréal, Département Génie Electrique.

[25] Intersil HSP43220 Decimating Digital Filter Development Software Data Sheet, July 2004 File Number: 2486.9

[26] Lattice Semiconductor Corporation Cascaded Integrator-Comb (CIC) Filter User's Guide, October 2005

[27] Xilinx Cascaded Integrator-Comb (CIC) Filter V3.0, Product Specification, March 2002

[28] Shuni, Chu., Burrus, C. (Nov 1984). Multirate filter designs using comb filters. Circuits and Systems, IEEE Transactions, 31 (11), 913-924

[29] Proakis, J. G., Manolakis, Dimitris G. (1996). Digital signal processing: principles, algorithms and applications (Prentice-Hall 3^{rd} ed)

[30] Vaidyanathan, P.P. (1993). Multirate systems and filters banks (PTR Prentice Hall)

[31] Krukowski, A., Kale, I. (July 2001). Polyphase Filter Design with Reduced Phase Non-Linearity, presented at 5th WSES/IEEE World Multiconference on circuits, Systems, Communications & Computers CSCC 2001, Rethymnon, crete (Greece)

[32] Brambilla, M., Guidi, D., Liberali, V. (29 Sept-2 Oct 1999). High speed FIR filters for digital decimation. Integrated Circuits and Systems Design, 1999. Proceedings. XII Symposium, 124-127

[33] Dia, S M F., Thibeault, C., & Gagnon, F. (2006). A Very High Speed and Efficient CIC Decimation Filter Core. Paper presented at the 4^{th} International IEEE-NEWCAS Conference, 2006.