Les filtres numériques

1. Présentation

L'acquisition de signaux physiques, quels qu'ils soient, est une opération souvent délicate. Un simple convertisseur analogique-numérique permet de transformer une tension ou un courant en une donnée numérique. Cela suffirait si les valeurs physiques n'étaient pas étroitement associées à une multitude de signaux : autres signaux de mesure, signaux parasites tels que le bruit électronique, les décharges électrostatiques, les fluctuations de la tension d'alimentation. Tous ces signaux se combinent, s'interfèrent et viennent « polluer » le signal utile. Afin de capturer correctement le signal recherché, un convertisseur analogique-numérique est toujours associé à des convertisseurs, à des amplificateurs ou à des filtres. Le rôle de chacun est de mettre en forme les signaux pour les rendre « propres » : les convertisseurs transforment des fréquences, des courants, des impulsions en tension, les amplificateurs adaptent l'amplitude de la tension à la plage d'entrée du convertisseur analogique-numérique. Le filtrage permet, quant à lui, d'extraire une partie de l'information liée à un signal, d'éliminer des fréquences parasites indésirables.

Le filtrage peut s'appliquer à des signaux analogiques ou numériques. Quand on parle de filtrage de signaux de mesure, on pense immédiatement au filtrage analogique. Par exemple, le filtre anti-repliement, les filtres passe-bas de Bessel ou de Tchebychev, le filtre passe-bande utilisant un amplificateur opérationnel. Il existe bien d'autres familles de filtres à base de composants traditionnels : les filtres analogiques passifs RLC qui utilisent des résistances, des condensateurs, des inductances, les filtres passifs à résonateurs à base de quartz, de résonateurs céramiques ou à ondes de surface, les filtres passifs à lignes imprimées, les filtres analogiques actifs intégrant des amplificateurs opérationnels ou des transistors, les filtres analogiques à capacités commutées. Tous ces filtres se distinguent par une facilité de mise en œuvre, un fonctionnement à des fréquences qui peuvent atteindre quelques gigahertz. Le revers de la médaille réside dans la sensibilité de ces composants aux conditions externes (température, humidité,…). La non-maîtrise de leurs tolérances nuit également à la précision du filtrage…surtout dans le cas de filtres exigeants.

Pour s'affranchir des limites des composants traditionnels, il existe une alternative : les filtres numériques. Les judicieux assemblages de résistances, de capacités, d'inductances, d'amplificateurs opérationnels des filtres analogiques sont ici remplacés par des algorithmes de calcul implémentés dans des microprocesseurs DSP ou des composants spécifiques du type FPGA. Les filtres numériques travaillent sur des signaux numérisés et sont donc placés en aval des convertisseurs analogique-numérique. Le signal analogique d'entrée est d'abord échantillonné à un rythme élevé, à une fréquence qui est au moins le double de la fréquence maximale contenue dans le signal (sinon, une partie de l'information est perdue). Les échantillons obtenus sont ensuite numérisés et c'est sur ces échantillons numériques que travaillent les filtres numériques. Les algorithmes implémentés dans les filtres sont caractérisés par un ensemble de coefficients permettant de déterminer la valeur d'un échantillon de sortie en fonction des échantillons d'entrée et/ou de sortie précédents. En quoi un filtrage numérique est-il plus performant qu'un filtrage analogique ?

Tout d'abord, on n'a pas les erreurs liées aux non-linéarités des amplificateurs opérationnels ou aux mauvaises tolérances des résistances, des capacités, que l'on trouve sur les filtres analogiques. Les filtres numériques ne sont pas pour autant parfaits, leurs résultats sont entachés par d'autres sources d'erreurs. L'une d'entre elles est liée à la conversion analogique-numérique. La numérisation consiste à capturer régulièrement une valeur d'un

signal analogique pour la transformer en un nombre binaire. Lors de cette étape, on introduit une erreur d'arrondi, appelée bruit de quantification, dont la valeur crête est parfaitement connue.

Une autre source de bruit provient de l'architecture des microprocesseurs à virgule fixe. Prenons l'exemple d'un filtre avec des coefficients valant en théorie 0,0001 et 0,9999 pour deux d'entre eux et implémenté sur DSP (Digital Signal Processor) à virgule fixe. Ce genre de DSP autorise pour valeurs de paramètres 0, 0,1, 0,2…Il faut donc arrondir les coefficients de l'algorithme aux valeurs possibles du DSP. Cette opération implique une erreur supplémentaire : c'est le bruit de calcul. On retrouve le même phénomène avec les résultats des opérations de base (multiplication, addition, soustraction et division).

Contrairement aux erreurs issues de la non-linéarité des amplificateurs opérationnels ou des tolérances des composants, les bruits de quantification et de calcul sont parfaitement connus et ils restent stables. Les filtres numériques gagnent donc en précision et en reproductibilité. Il est même possible d'améliorer la précision en réduisant le bruit de quantification. Pour cela, il faut utiliser des microprocesseurs disposant d'un nombre de bits plus grand. Ainsi, le pas de quantification, donc l'erreur entre la valeur réelle et l'arrondi, seront plus petits. En ce qui concerne le bruit de calcul lié aux microprocesseurs à virgule fixe, une solution consiste à utiliser des FPGA à virgule flottante. Toutes les valeurs des coefficients d'un filtre numérique sont possibles sans arrondi, ce qui rend le bruit de calcul négligeable. En contrepartie, ces processeurs à virgule flottante requièrent une puissance de calcul plus importante, ralentissant les temps d'exécution. Avec les évolutions technologiques des composants, ce handicap s'estompe progressivement.

Le filtrage numérique apporte d'autres avantages par rapport aux filtres analogiques. Changer de gabarit d'un filtre (ondulation dans la bande passante, ordre du filtre, affaiblissement dans la bande de réjection) ou même de changer de filtre (passe-bas en passe-haut) revient à modifier les coefficients d'un algorithme ou l'algorithme en lui-même. Ces opérations s'effectuent par logiciel. Il n'est donc plus nécessaire d'enlever et d'assembler différemment résistances, capacités, amplificateurs opérationnels.

Filtrage numérique face au filtrage analogique

	Avantages	Inconvénients
Filtrage numérique (IIR et FIR)	• Insensibilité aux conditions extérieures (chaleur, humidité, etc.) • Pas de problème de dérive des composants • Nombre d'opérations illimité • Filtres non réalisables en analogique (par exemple filtres FIR) • Précision connue et contrôlée • Reproductibilité, sans réglage • Filtres modifiables par logiciel • Filtrage adaptatif (filtres FIR) • Systèmes intégrables • Systèmes adaptés aux grandes séries	• Performances du filtre directement proportionnelles à la puissance de l'unité de calcul (DSP ou FPGA) • Filtrage limité à 100 MHz • Nécessité d'un filtrage anti-repliement (analogique) à l'échantillonnage et à la restitution • Conversions analogique-numérique et numérique-analogique obligatoires • Alimentation nécessaire • Bande passante importante • Limitation pour les signaux forts et faibles
Filtrage analogique passif (avec composants discrets ou piézoélectriques)	• Réalisable jusqu'à des fréquences élevées (quelques GHz) • Pas d'alimentation • Systèmes adaptés aux grandes séries	• Faibles tolérances des composants • Systèmes difficilement intégrables
Filtrage analogique actif (filtres standards ou à capacité commutée)	• Fréquence de coupure programmable (filtres à capacité commutée) • Systèmes intégrables (filtres à capacité commutée)	• Filtrage limité à quelques MHz • Alimentation nécessaire • Excursion de la tension de sortie limitée • Nécessite des composants précis

Tableau 4 : comparaison entre le filtrage numérique et le filtrage analogique.

2. Comparaisons entre FIR et IIR

La fonction de transfert d'un filtre numérique est :

$$y(n) = \sum_{k=0}^{M} a_k .x(n-k) - \sum_{j=1}^{N} b_j .y(n-j)$$

a_k et b_j étant les coefficients, x l'échantillon d'entrée et y l'échantillon de sortie.

Il existe deux types de filtres numériques :

- Les FIR (Finite Impulse Response) qui sont stables donc pas de pôles (b_j=0).
- Les IIR (Infinie Impulsion Response) qui sont stables uniquement si les modules de leurs pôles sont supérieurs à 1.

Voici un tableau comparant ces deux types de filtres numériques :

Les filtres numériques : FIR ou IIR ?		
	Avantages	**Inconvénients**
Filtres Infinite Impulse Response	• Bien moins de calculs que pour un filtre FIR à performances équivalentes	• Vérifier la stabilité des filtres • Phase non linéaire, donc distorsion de phase
Filtres Finite Impulse Response	• Filtres toujours stables • Phase des signaux linéaire si symétrie des coefficients • Pas de distorsion de phase • Possibilité de réaliser toutes sortes de filtres (standards et exotiques)	• Beaucoup de calculs par rapport à un filtre IIR à performances équivalentes

Tableau 5 : comparaisons entre FIR et IIR.

3. Les FIR

Pour effectuer le gabarit d'un filtre numérique non-récursifs (FIR), il est nécessaire de connaître :

- ✓ Le taux d'ondulation dans la bande passante (δ_1) ;
- ✓ Le taux d'affaiblissement dans la bande rejetés (δ_2) ;
- ✓ La bande de transition $\Delta F = f_p - f_c$.

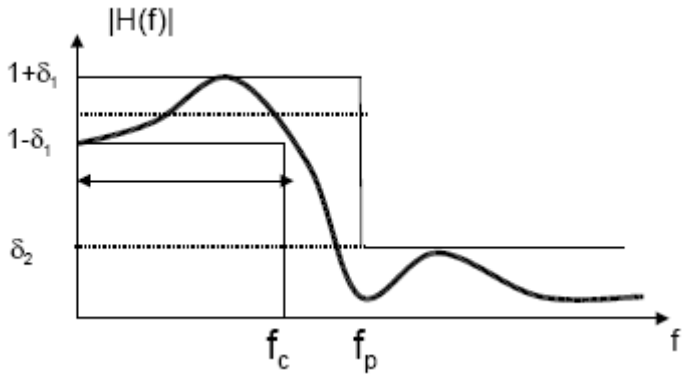

Figure 16 : Gabarit d'un FIR.

A partir de ces différents paramètres, il est possible de choisir une fenêtre appropriée. En effet, il existe plusieurs fenêtres mais 4 d'entre elles sont les plus répandues pour définir un FIR :

- La fenêtre rectangulaire ;
- La fenêtre de Hamming ;
- La fenêtre de Hanning ;
- La fenêtre de Blackman.

Fenêtre Rectangulaire :

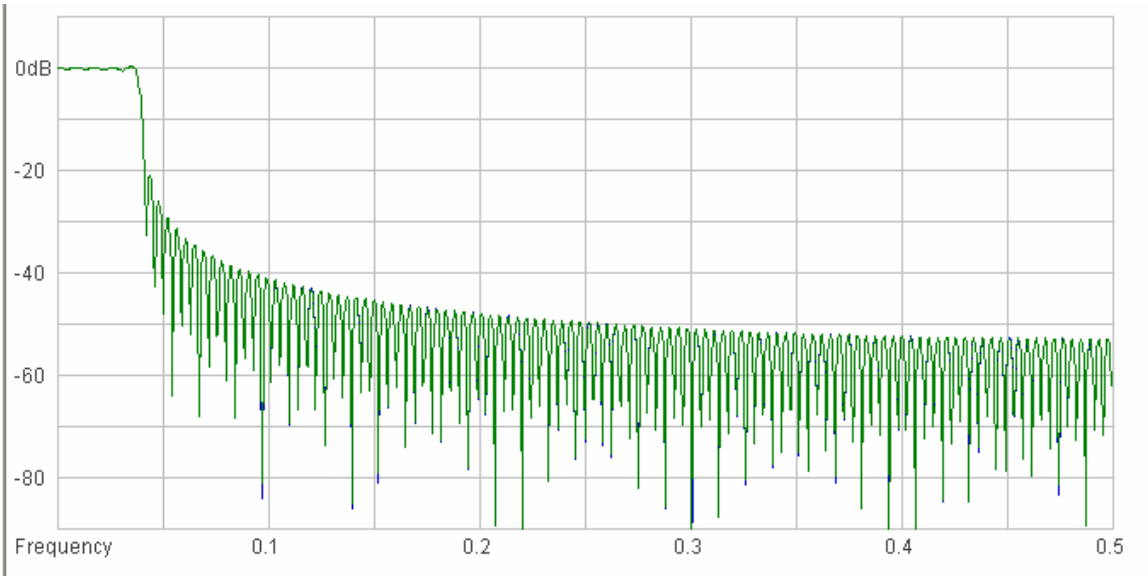

Figure 17 : Fenêtre Rectangulaire (256 samples à 250 kHz, fc=10 kHz)

- ✓ Bande de transition étroite (inférieure à 1 kHz).
- ✓ Atténuation stagnante dans la bande rejetée (-55 dB).

$$w[n] = 1$$

Fenêtre de Hamming :

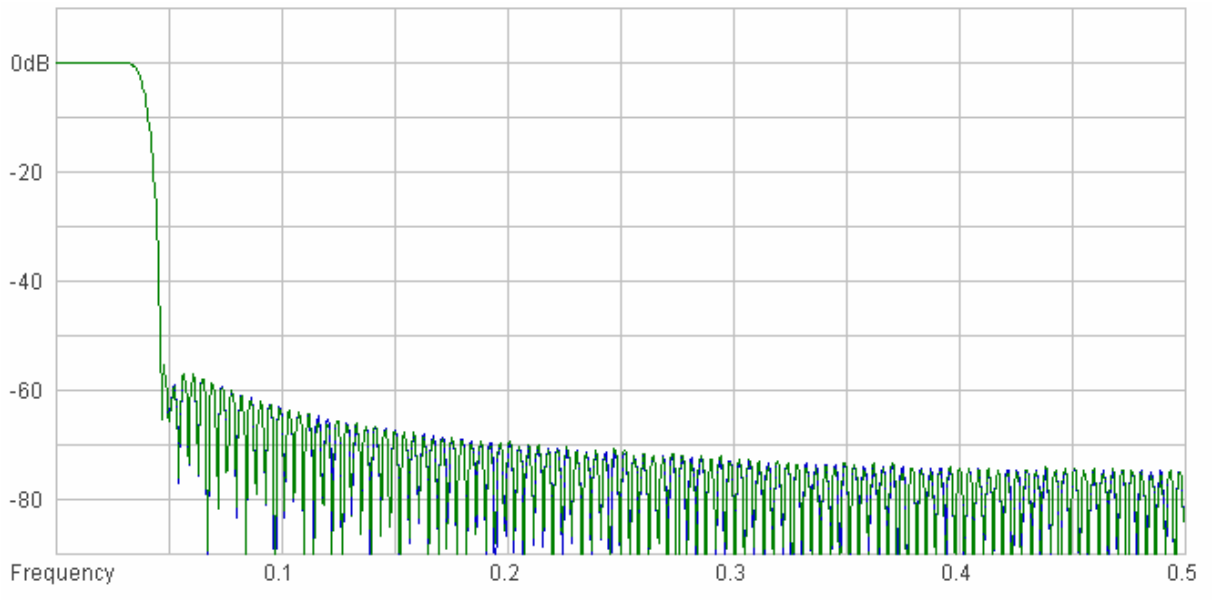

Figure 18 : Fenêtre de Hamming (256 samples à 250 kHz, fc=10 kHz)

- ✓ Atténuation stagnante dans la bande rejetée (-75 dB).
- ✓ Bande de transition étroite (inférieure à 2 kHz).

$$w[n] = 0.54 + 0.46\cos(2\pi \frac{n}{2M+1})$$

Fenêtre de Hanning :

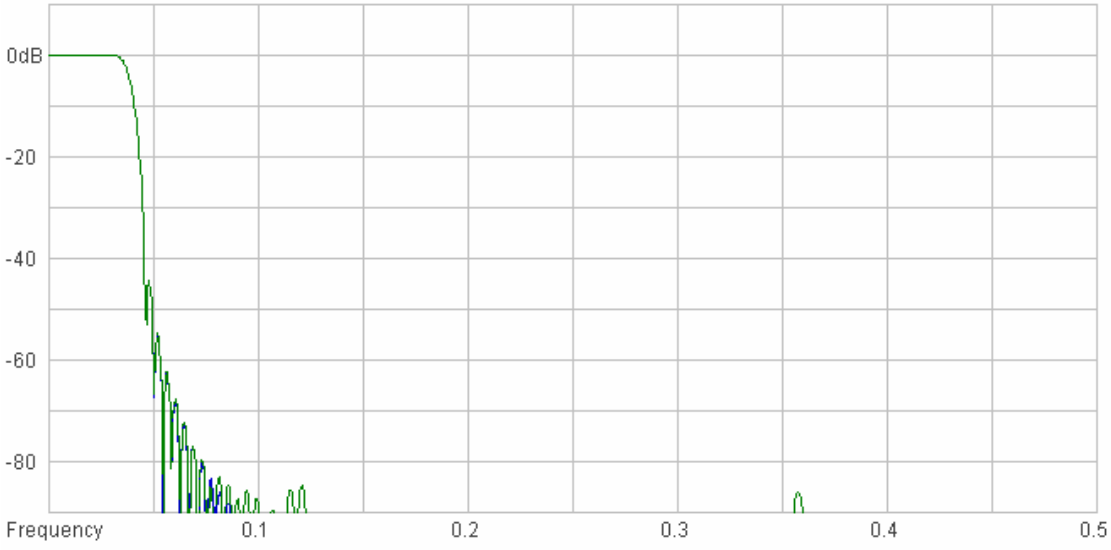

Figure 19 : Fenêtre de Hanning (256 samples à 250 kHz, Fc=10 kHz)

- ✓ Meilleure atténuation dans la bande coupée que la fenêtre de Hamming.
- ✓ Bande de transition plus grande (40 dB d'atténuation contre 60 dB pour la fenêtre de Hamming).

$$w[n] = 0.5 + 0.5\cos(2\pi\frac{n}{2M+1})$$

Fenêtre de Blackman :

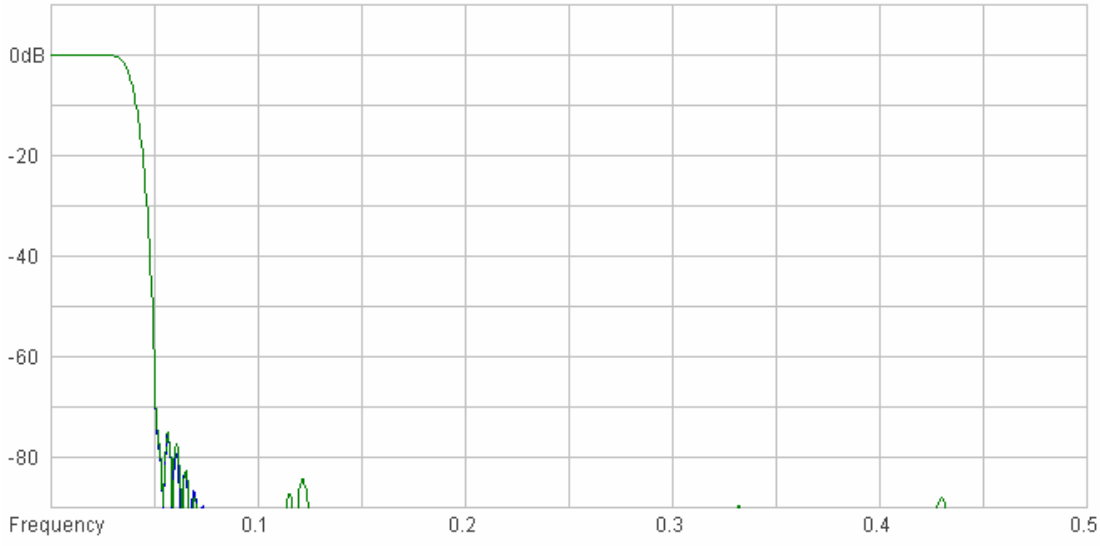

Figure 20 : Fenêtre de Blackman (256 samples à 250 kHz, Fc=10 kHz)

- ✓ Nette atténuation en bande rejetée.
- ✓ Faible bande de transition (2,5 kHz pour atteindre 60 dB d'atténuation contre 2 kHz pour la fenêtre de Hanning).

$$w[n] = 0.42 + 0.46\cos(2\pi\frac{n}{2M+1}) + 0.08\cos(2\pi\frac{n}{2M+1})$$

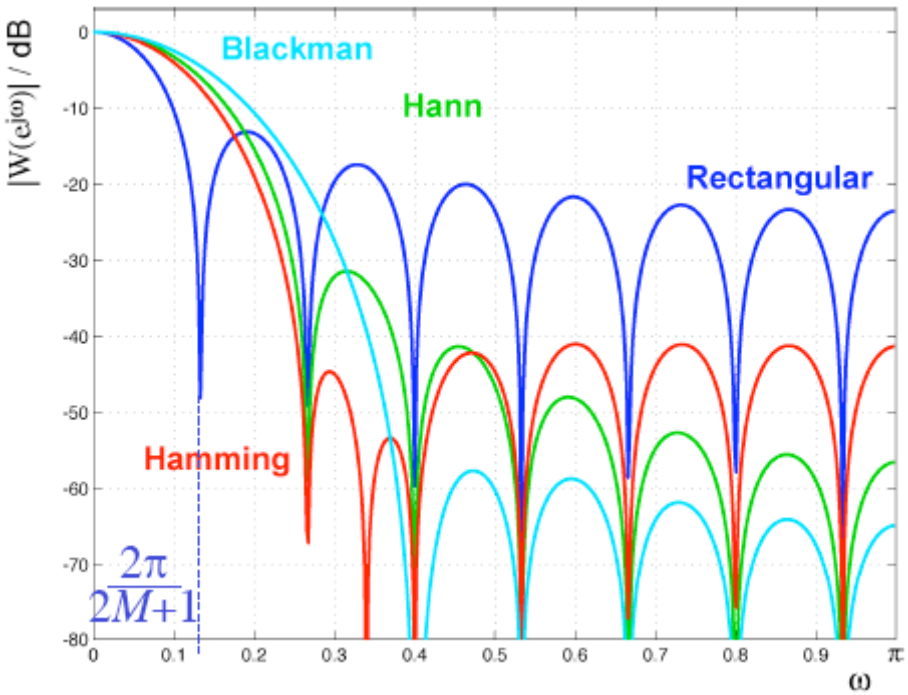

Figure 21 : Comparaisons des différentes fenêtres.

Chacune d'entre elles possède leurs propres avantages et inconvénients. Par exemple, si l'on veut un filtre ayant une faible bande de transition, peu importe l'atténuation, il est préférable d'utiliser une fenêtre Rectangulaire. En revanche, s'il est question d'une atténuation minimale de 30 dB, il vaut employer une fenêtre de Hanning. La fenêtre de Blackman correspond aussi à ce gabarit du filtre mais sa bande de transition est trop importante.

Type de fenêtre	Δf, largeur de transition (en fréquence réduite)	Ondulation en bande passante A_p (dB)	Atténuation du 1er lobe secondaire (dB)	Atténuation en bande atténuée A_a (dB)
Rectangu-laire	1.8/N	0.7416	13	21
Hann (Hanning)	6.2/N	0.0546	31	44
Hamming	6.6/N	0.0194	41	53
Blackman	11/N	0.0017	57	74

Tableau 6 : Principales caractéristiques des fenêtres.

Au plus le nombre d'échantillons utilisés dans le filtre est important, au plus la bande de transition est réduite. En revanche, le retard du premier échantillon valide s'accroît.

4. Calculs des coefficients des FIR

Les coefficients h(n) sont calculés grâce à la formule suivante :

$$h(n) = h_D(n).w(n)$$

avec : $h_D(n)$, réponse impulsionnelle du filtre (filtre passe-bas par exemple) et w(n), réponse impulsionnelle de la fenêtre.

Prenons par exemple un filtre passe-bas de réponse impulsionnelle :

$$h_D(n) = 2f'_c \sin c(f'_c.(n - N_{taps}))$$

ayant comme caractéristiques principales :

- 33 échantillons ;
- fréquence d'échantillonnage à 5 kHz ;
- fréquence de coupure à 2 kHz ;
- bande de transition de 0.25 kHz.

$f'_c = f_c + \Delta f = (2 + 0.25)/5 = 0.45$

Pour ce filtre-ci, la réponse impulsionelle de la fenêtre de Hamming est :

$$w[n] = 0.54 + 0.46\cos(2\pi\frac{n}{33})$$

Coefficient Values		
h(0)	0.9	h(0)
h(-1)	0.09755	h(1)
h(-2)	-0.09047	h(2)
h(-3)	0.07957	h(3)
h(-4)	-0.06606	h(4)
h(-5)	0.05136	h(5)
h(-6)	0.03689	h(6)
h(-7)	0.02399	h(7)
h(-8)	-0.01314	h(8)
h(-9)	0.00519	h(9)
h(-10)	0	h(10)
h(-11)	0.00277	h(11)
h(-12)	0.00372	h(12)
h(-13)	-0.00353	h(13)
h(-14)	0.00284	h(14)
h(-15)	-0.00209	h(15)
h(-16)	0.00155	h(16)

Tableau 7 : Valeurs des coefficients du filtre passe-bas calculé.

Calculer 16 coefficients à la main est faisable pour n'importe quel bipède, mais il est vrai que c'est une perte de temps. Certains ont développé des logiciels permettant de calculer tous ces coefficients. C'est le cas d'ALTERA qui fournit avec son logiciel Quartus II, un programme permettant certes, de calculer les coefficients des filtres mais aussi de générer des FIR en VHDL ou en Verilog. Le seul bémol de ce logiciel est que les filtres réalisés ne sont pas modifiables à souhait ; il n'est pas possible avec un seul de ces filtres de réaliser des filtres ayant des gabarits différents.

Figure 22 : Logiciel FIR Compiler MegaCore.

En cliquant sur « Parameterize », vous pouvez entrer les différentes caractéristiques de votre filtre. Dans cette première interface, il est possible de choisir le nombre de bits de vos échantillons ainsi que de vos coefficients. Vous pouvez également garder ou de tronquer le résultat final. Il est possible d'estimer la place allouée dans le FPGA en modifiant la famille désirée, la méthode de calcul (parallèle ou série), la sauvegarde des échantillons et des coefficients. Pour vos coefficients, ils peuvent être convertis en virgule fixe ou en virgule flottante. La réponse fréquentielle (« Frequency Response ») et la réponse impulsionnelle (« Time Response & Coefficients Values) sont visualisables sur cette même interface.

Les caractéristiques générales des filtres sont modifiables en cliquant sur « New Coefficient Set » ou sur « Edit Coefficient Set ».

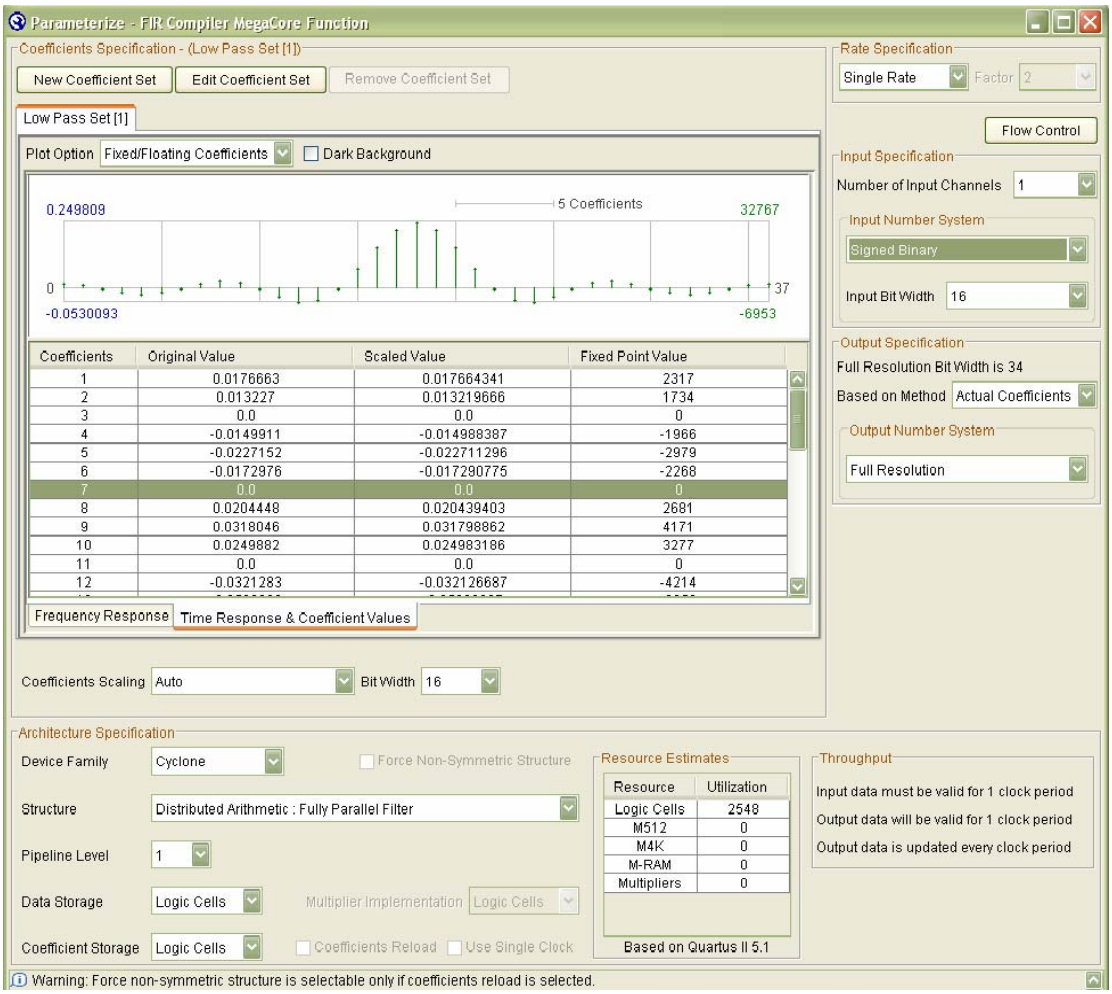

Figure 23 : Interface « Parameterize ».

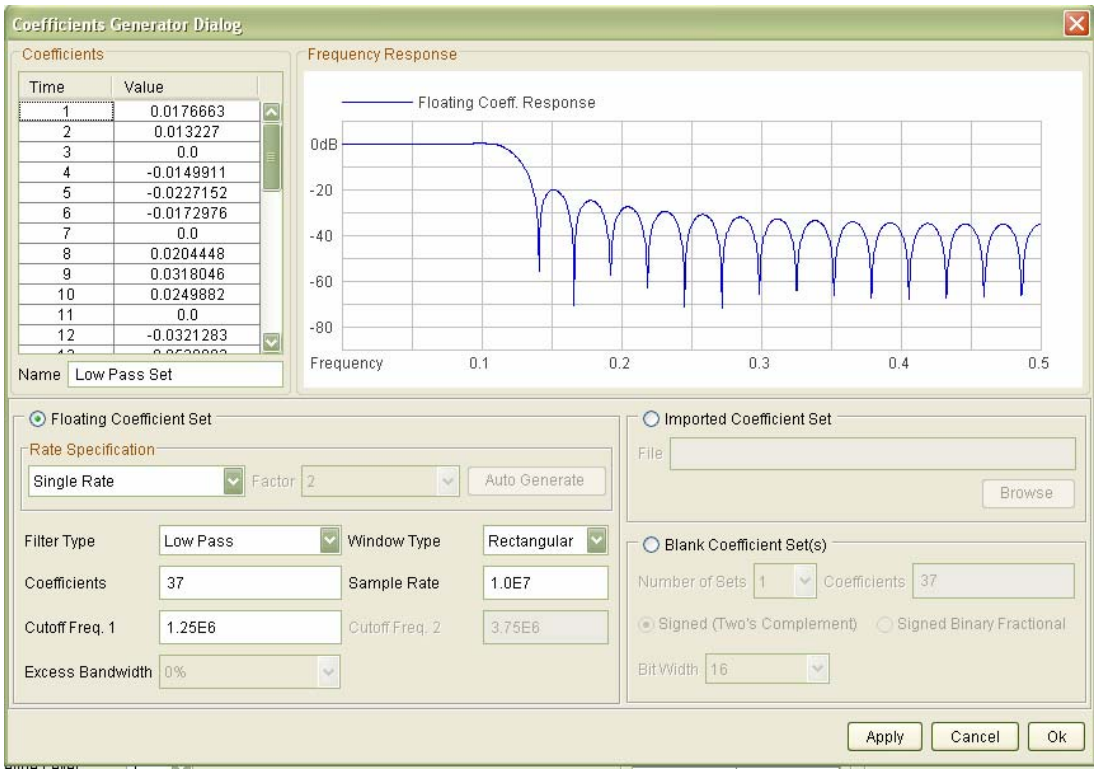

Figure 24 : Interface « Coefficients Generator Dialog »

Sur l'interface « Coefficients Generator Dialog », vous choisissez votre type de filtre (passe-bas, passe-haut, passe bande,…), vos fréquences de coupure, votre nombre d'échantillons, votre fréquence d'échantillonnage ainsi que votre type de fenêtre. Vous cliquez sur « Apply » et vous visualisez la réponse fréquentielle du filtre ainsi que les valeurs des coefficients. Une fois votre filtre clairement défini, il ne vous reste plus qu'à générer le fichier VHDL le représentant en appuyant sur « Generate » de la première interface du logiciel.

5. Méthodes de structure des filtres

Il existe trois méthodes de réaliser des FIR :

- FIR parallèle ;
- FIR série ;
- FIR utilisant les LUT (« Look-Up Table »).

a. FIR parallèle

Cette conception de filtre est très simple et ne nécessite qu'un seul coup d'horloge pour restituer le résultat. A chaque coup d'horloge, les échantillons sont décalés grâce à des registres à décalage. Pour chaque échantillon, un multiplieur est nécessaire. La résolution de sortie de ce multiplieur dépend du nombre de bits du coefficient et de l'échantillon. Si par exemple, le coefficient et l'échantillon sont de 16 bits chacun, la résolution du multiplieur sera alors de 32 bits. Un additionneur est nécessaire pour additionner les produits.

Par exemple, un FIR de 8 échantillons nécessite 8 multiplieurs, un registre à décalage de 8x16 bits et un additionneur (voir figure ci-dessous).

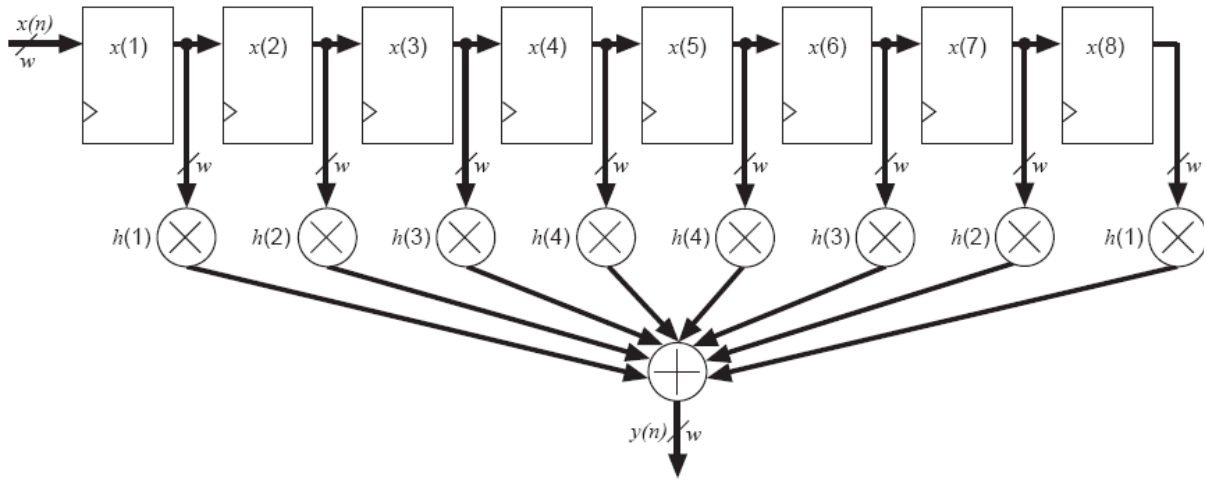

Figure 25 : FIR parallèle (1ère méthode).

Etant donné que les coefficients sont symétriques, il n'est guère nécessaire d'utiliser un multiplieur par échantillon. Il est préférable d'utiliser moitié moins de multiplieurs en sommant les échantillons ayant les mêmes coefficients puisque les multiplieurs nécessitent plus de places dans un FPGA qu'un additionneur.

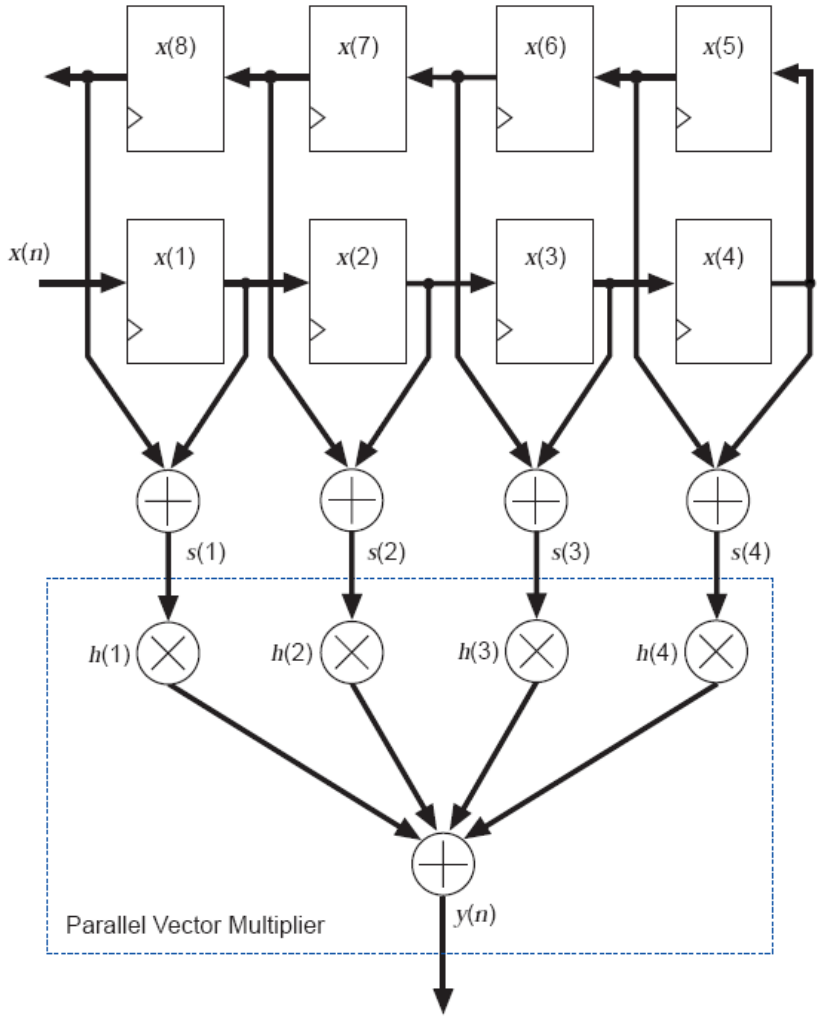

Figure 26 : FIR parallèle (2ème méthode).

b. *FIR série*

Le filtre série utilise nettement moins d'éléments logiques qu'un filtre parallèle. En effet, pour effectuer un FIR et ce quelque soit le nombre d'échantillons, vous n'avez besoin que d'un multiplieur et de deux additionneurs. Cela devient très intéressant lorsque vous utilisez des FIR utilisant quelques centaines d'échantillons. Le seul inconvénient majeur dans ce type de filtre est qu'il lui faut un certains temps pour obtenir le résultat.

Lorsqu'un échantillon arrive à l'entrée du FIR, il est tout d'abord, sauvegardé dans une RAM interne au FPGA, de là, s'ensuit une longue phase de calculs pour aboutir au résultat final. La première opération est une addition entre les deux échantillons ayant le même coefficient, étant donné que les coefficients sont soient symétriques, soient antisymétriques. Une fois utilisés, les échantillons sont par la suite déplacés d'une adresse pour le prochain calcul. L'addition terminée, la somme est multipliée par un coefficient sauvegardé dans une RAM. Tous les produits sont sommés entre eux afin d'avoir le résultat final. Ce résultat est obtenu une fois que tous les produits ont été sommés.

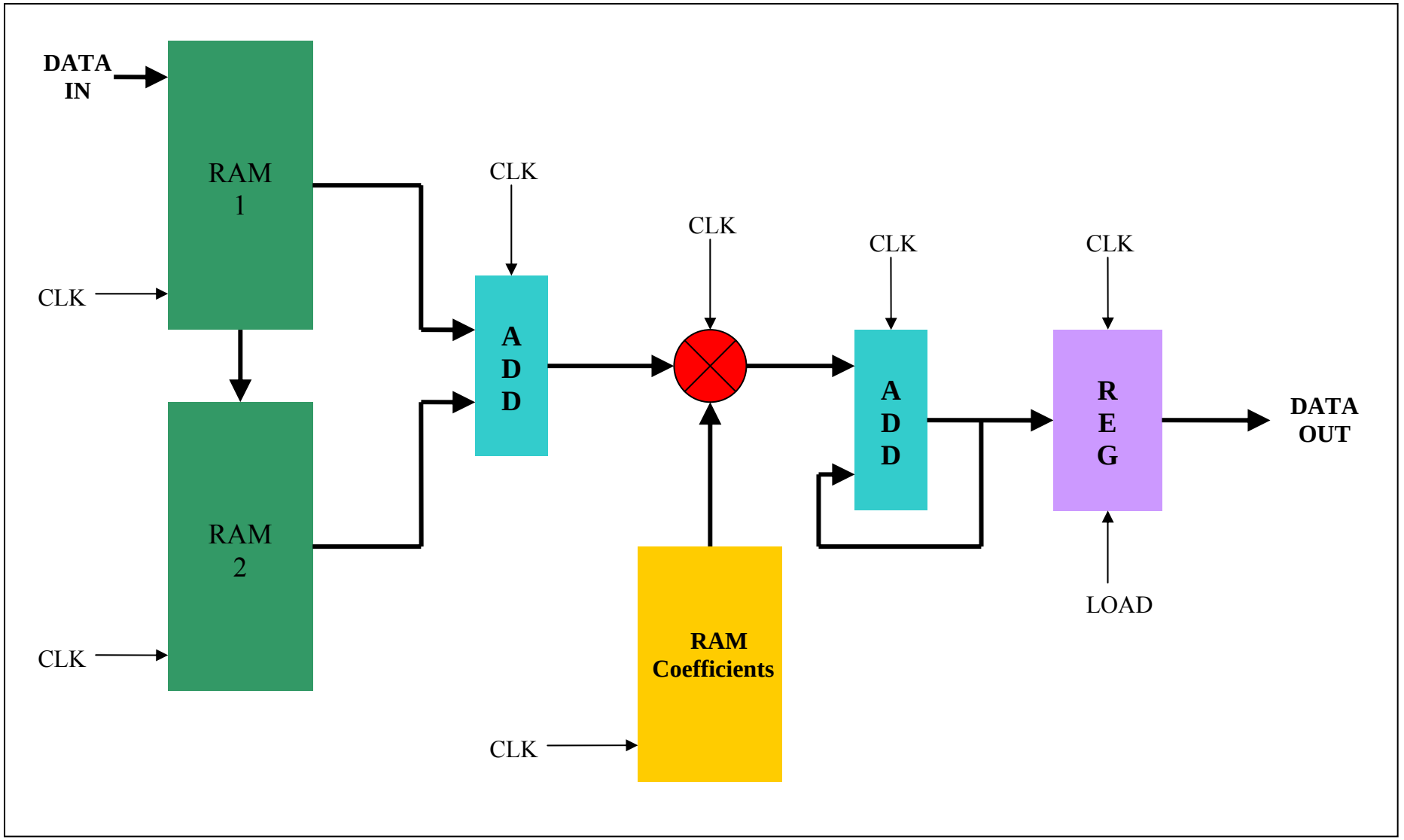

Figure 27 : FIR série.

Oui mais voilà, étant donné que toute cette phase de calcul est prévue pour deux échantillons à la fois, il faut donc un nombre de coups d'horloge minimal égal à la moitié des échantillons. Je m'explique, prenons l'exemple d'un FIR de 128 échantillons, il est nécessaire d'avoir au minimum 64 coups d'horloge pour obtenir le résultat du calcul. Lorsque vous travaillez avec un taux d'échantillonnage plus lent que la fréquence du microprocesseur, cela est parfait mais si ce n'est pas le cas, cette méthode sera très compliquée à employer…

Un filtre comme celui-ci nécessite 3 RAM internes dans le FPGA et 965 éléments logiques. A titre de comparaison, un filtre à structure parallèle utilise pas moins de 9800 éléments logiques…Un filtre série correspond tout à fait à la carte CPCI 228 car pour 16 voies analogiques, vous pouvez utiliser 16 filtres de cette structure dans le FPGA. De plus, la fréquence d'échantillonnage est nettement plus lente que la fréquence du FPGA. En effet, entre deux temps d'échantillonnage, il y a 128 coups d'horloge du microprocesseur, ce qui est assez pour effectuer un FIR à 128 voire 230 échantillons.

c. FIR à LUT

Les LUT sont utilisés pour remplacer les multiplieurs, gourmands en éléments logiques. Il est tout simplement composé d'une RAM plus ou moins grande selon la taille des entrées. Le seul inconvénient est qu'une des deux entrées doit être une constante.

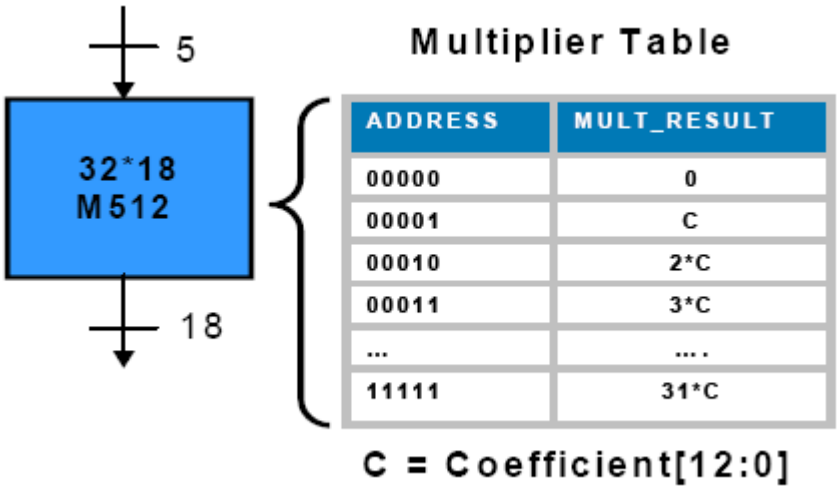

Figure 28 : LUT.

Dans la figure ci-dessus, la donnée est considérée comme l'adresse de la RAM. La RAM est initialement remplie des valeurs possibles en sortie. C est le coefficient et la donnée d'entrée est consistée de 5 bits. Si par exemple, la donnée est « 10011 » en binaire, soit 19 en décimal, le résultat sera donc la valeur inscrite à l'adresse « 10011 », donc « 19 x C ». L'inconvénient majeur est qu'il faut remplir la RAM des valeurs possibles, c'est-à-dire que vous faites les trois-quarts du travail du FPGA.

Il est également possible d'effectuer avec ce type de structures, des associations d'addition et de multiplications. Prenons l'exemple de l'équation suivante :

$$output = c_0.w + c_1.x + c_2.y + c_3.z$$

w, x, y et z sont des coefficients fixes et c_0, c_1, c_2 et c_3 sont les échantillons. Si dans l'ordre, ces quatre échantillons ont la valeur suivante, « 1011 », cela équivaut à la somme des coefficients w, y et z. Il y a 4 échantillons donc 16 résultats possibles. Il suffit juste de remplir la RAM correspondante à cette équation (voir figure ci-dessous).

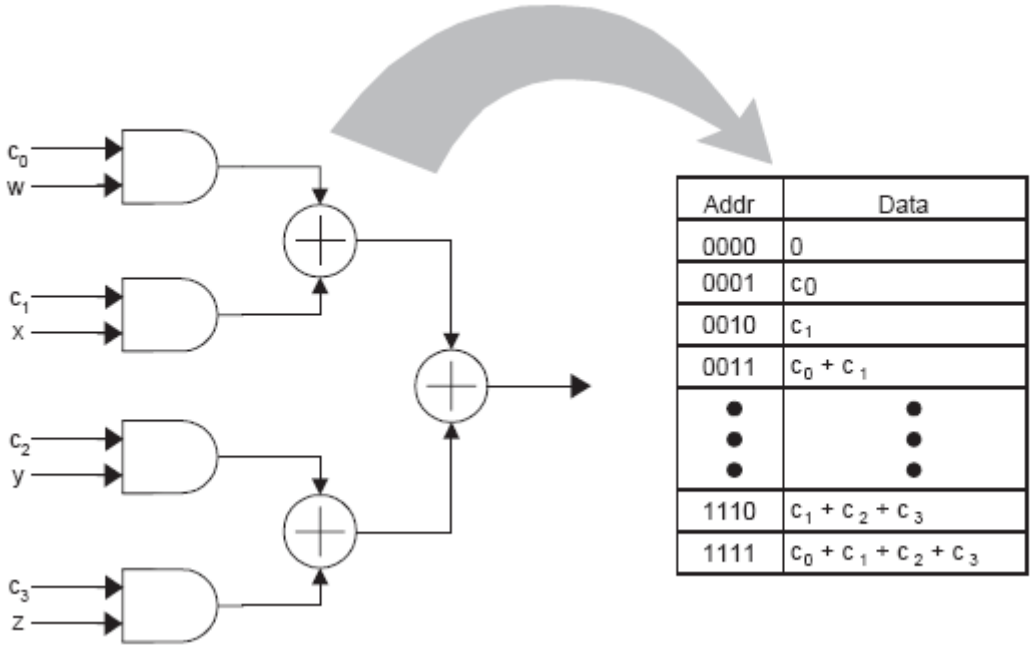

Figure 29 : LUT à 4 entrées.

Etant donné que les échantillons sont rarement composés d'un seul et unique bit, il est nécessaire de placer avant cette « table » un convertisseur parallèle-série par voie ainsi qu'un registre à décalage associé à un additionneur en sortie du filtre.

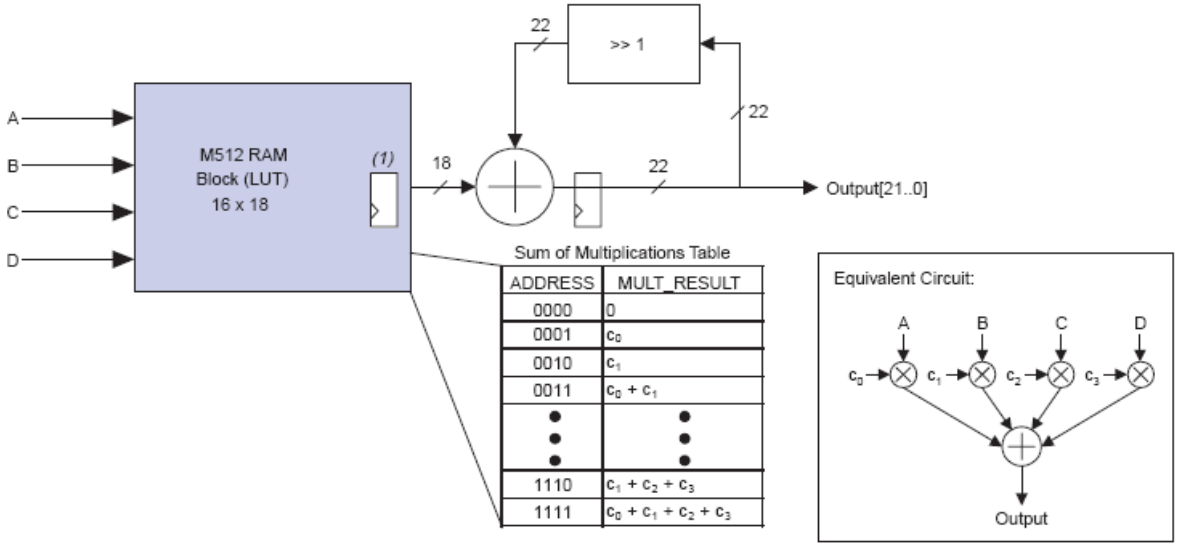

Figure 30 : FIR à 4 entrées.

4 échantillons pour un filtre numérique, c'est relativement peu si l'on veut un filtre relativement sélectif. Il est vrai que les coefficients sont symétriques donc on peut ajouter quelques additionneur en amont des convertisseurs parallèle-série, afin d'avoir un filtre à 8 échantillons. Cela reste relativement peu ! Si on double le nombre d'échantillons, c'est-à-dire 16, il faut pour cela, avoir une RAM de 256 x 20 bits et ainsi de suite. Imaginez un FIR de 64 échantillons, il faut une RAM avec un minimum d'espace adressable de 4 Go ! Impensable !

Il est donc nécessaire de séparer les échantillons par groupe de 4 et de sommer leurs résultats (voir figure 31).

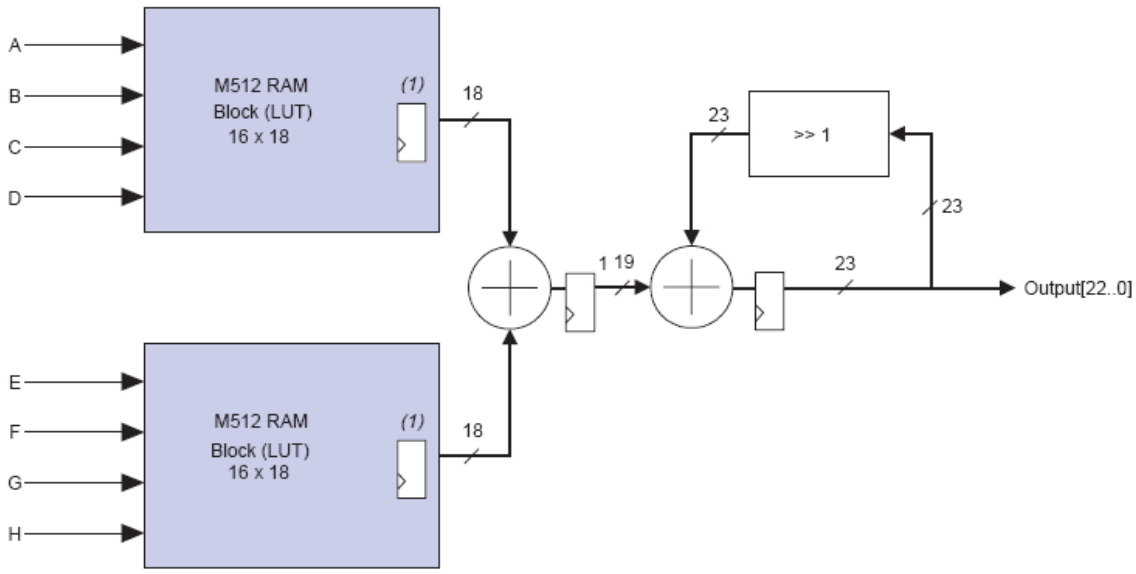

Figure 31 : FIR à 8 entrées.

Pour obtenir le résultat final, il faut attendre 17 coups d'horloge (nombre de bits de l'échantillon + « overflow » de la somme des deux échantillons). D'un point de vue, place occupée dans le FPGA, cela équivaut à un filtre série donc relativement faible. L'inconvénient majeur pour ce type de filtre est qu'il est nécessaire d'avoir au moins deux fois plus de RAM ainsi que d'éléments logiques pour effectuer un FIR avec des échantillons et coefficients SIGNES ! Cette structure ne conviendra pas à la carte CPCI228.

6. Modélisation d'un FIR série en VHDL

Avant de concevoir le filtre en VHDL, il faut réfléchir à la manière que l'on va procéder pour le réaliser. Pour cela, il vaut mieux commencer à concevoir les chronogrammes (voir figure 32) avec les différents signaux nécessaires. Ensuite, il faut songer aux contraintes constructeurs, c'est-à-dire celles du FPGA, comme par exemple le temps d'attente pour obtenir le résultat en sortie de RAM. Chez Altera, il est nécessaire d'attendre 2 cycles d'horloge après avoir placé l'adresse en entrée, pour avoir un résultat cohérent.

Sur la figure ci-après, les signaux ainsi que les bus forment le chronogramme principal du FIR à 128 échantillons. Tout y est représenté, des signaux servant aux calculs jusqu'à ceux utilisés pour déplacer les échantillons. A préciser également que tous les signaux sont synchrones (déclenchement sur front montant du signal d'horloge).

D'un point de vue VHDL, le multiplieur, les additionneurs, les RAMs et les registres ont tous été élaborés par le constructeur dans une librairie. Le FPGA utilisé par la carte CPCI228 est un Cyclone composé de 20 000 portes logiques et de 64 RAM de 4096 bits chacune. En récapitulant, on a besoin de 965 éléments logiques et de 3 RAM par filtre, ce qui nous fait donc 48 RAM et 15440 éléments logiques pour 16 filtres.

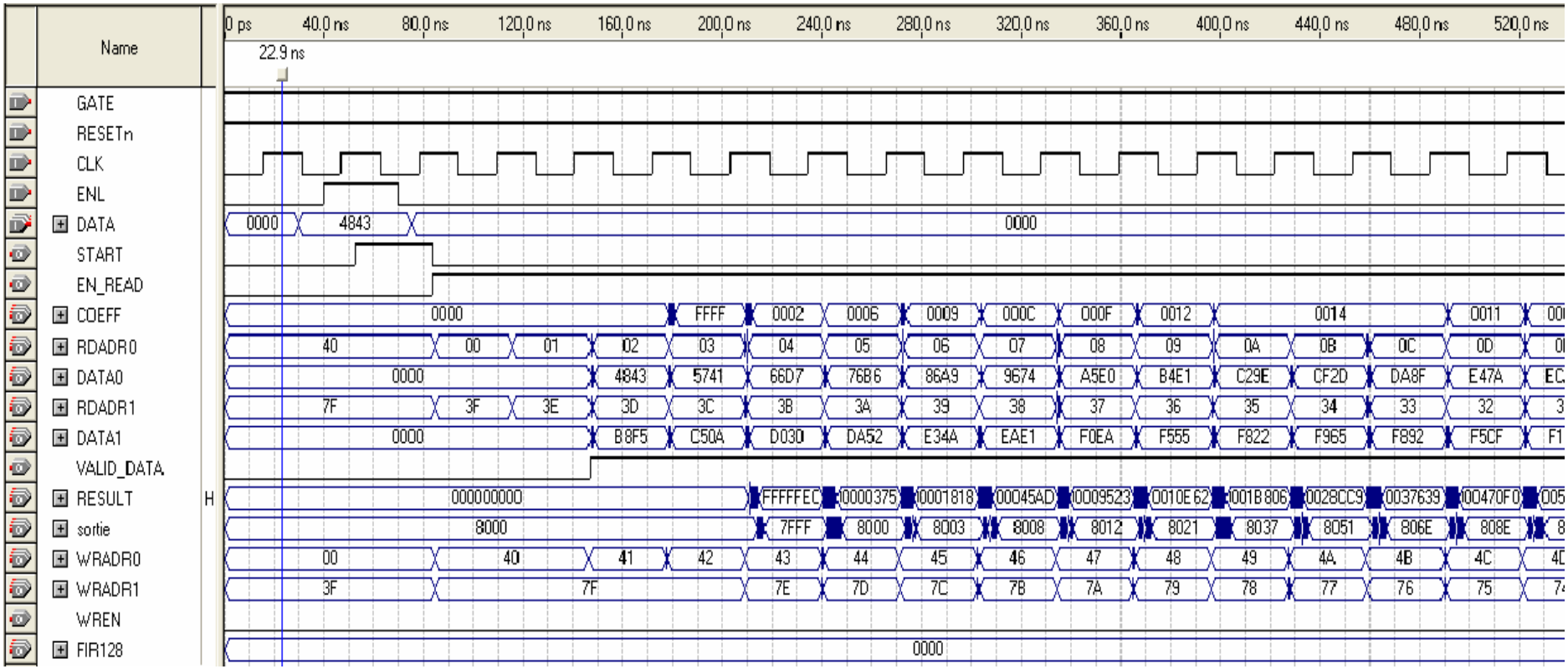

Figure 32: Chronogramme d'un FIR 128 échantillons.

```
ram0 : lpm_ram_dp
    GENERIC MAP(LPM_WIDTH=>16,LPM_WIDTHAD=>7,LPM_NUMWORDS=>128,LPM_FILE=>"ram0.mif")
    PORT MAP(wren=>WR0,data=>DATAA,wraddress=>wraddress0,wrclock=>CLK,rdclock=>CLK,rdaddress=>RDADR0,rden=>EN_READ,q=>DATA0);

ram1 : lpm_ram_dp
    GENERIC MAP(LPM_WIDTH=>16,LPM_WIDTHAD=>7,LPM_NUMWORDS=>128,LPM_FILE=>"ram1.mif")
    PORT MAP(wren=>WR1,data=>DATAB,wraddress=>wraddress1,wrclock=>CLK,rdclock=>CLK,rdaddress=>RDADR1,rden=>EN_READ,q=>DATA1);

multiplication : mult2
    PORT MAP(DATA0,DATA1,COEFF,CLR,CLK,VALID_DATA,RESULT,sortie);

ram_coeff : coeff_ram
    PORT MAP(CLK,EN_READ,ADR,"0000000000000000",COEFF);
```

RAM Mesures

Bloc Multiplieur

Bloc RAM Coefficients

Texte 11 : Déclaration des principaux composants du FIR 128.

```
retard0 : dff
    PORT MAP(enable,clk,'1','1',enable_mult);

retard1 : dff
    PORT MAP(clr,clk,'1','1',clear);

data_in<= (not DATA(15)) & (not DATA(15)) & DATA(14 downto 0);
datb_in<= (not DATB(15)) & (not DATB(15)) & DATB(14 downto 0);

adder : lpm_add_sub
    GENERIC MAP(LPM_DIRECTION=>"ADD",LPM_WIDTH=>17,LPM_REPRESENTATION=>"SIGNED",LPM_PIPELINE=>1)
    PORT MAP(dataa=>data_in,datab=>datb_in,aclr=>clear,result=>SUM,clken=>enable,clock=>clk);

mult : lpm_mult
    GENERIC MAP(LPM_WIDTHA=>17,LPM_WIDTHB=>16,LPM_WIDTHP=>34,LPM_WIDTHS=>34,LPM_REPRESENTATION=>"SIGNED",LPM_PIPELINE=>1,LPM_HINT=>"MAX
    PORT MAP(dataa=>SUM,datab=>COEFF,sum=>RESULT,clock=>clk,clken=>enable_mult,aclr=>clear,result=>RESULT);
```

Mise en forme des signaux

Addition

Multiplieur

Texte 12 : Bloc Multiplieur.

7. Résultats obtenus

J'ai réalisé 4 filtres FIR à 128 échantillons utilisant chacun une fenêtre différente : Rectangulaire, Hamming, Hanning et Blackman. La fréquence d'échantillonnage est la fréquence d'échantillonnage maximale, c'est-à-dire 250 kHz et la fréquence du FIR, 10 kHz. Voici quelques simulations à différents fréquences d'entrées…

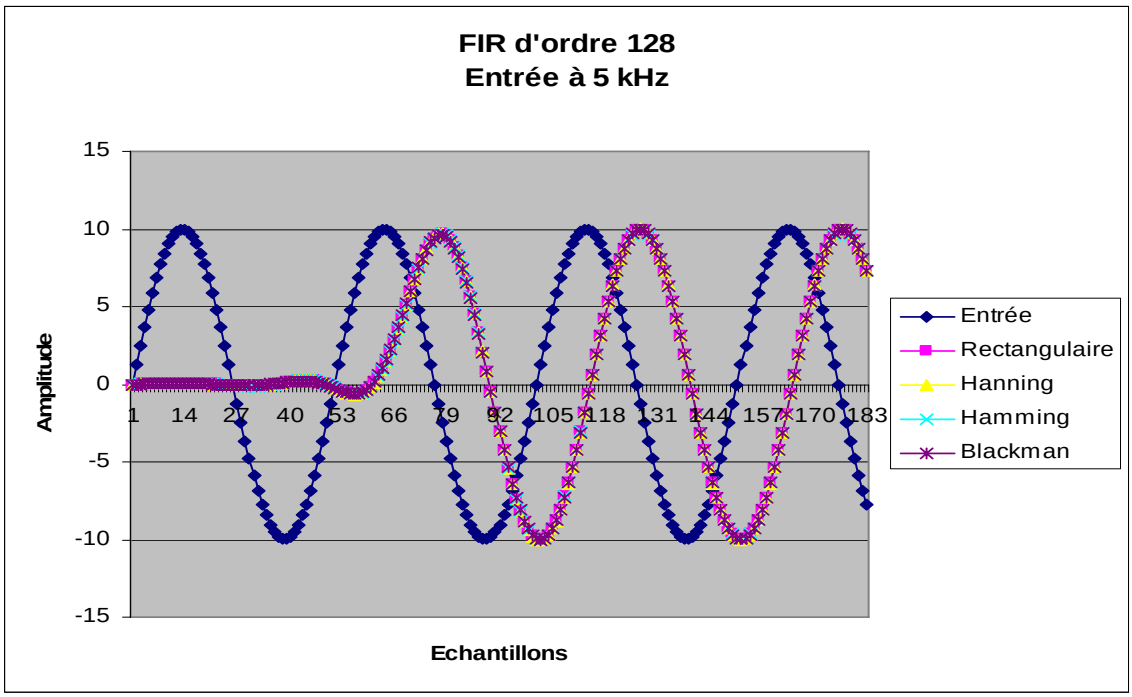

Figure 33 : Simulation du FIR 128 à 5 kHz.

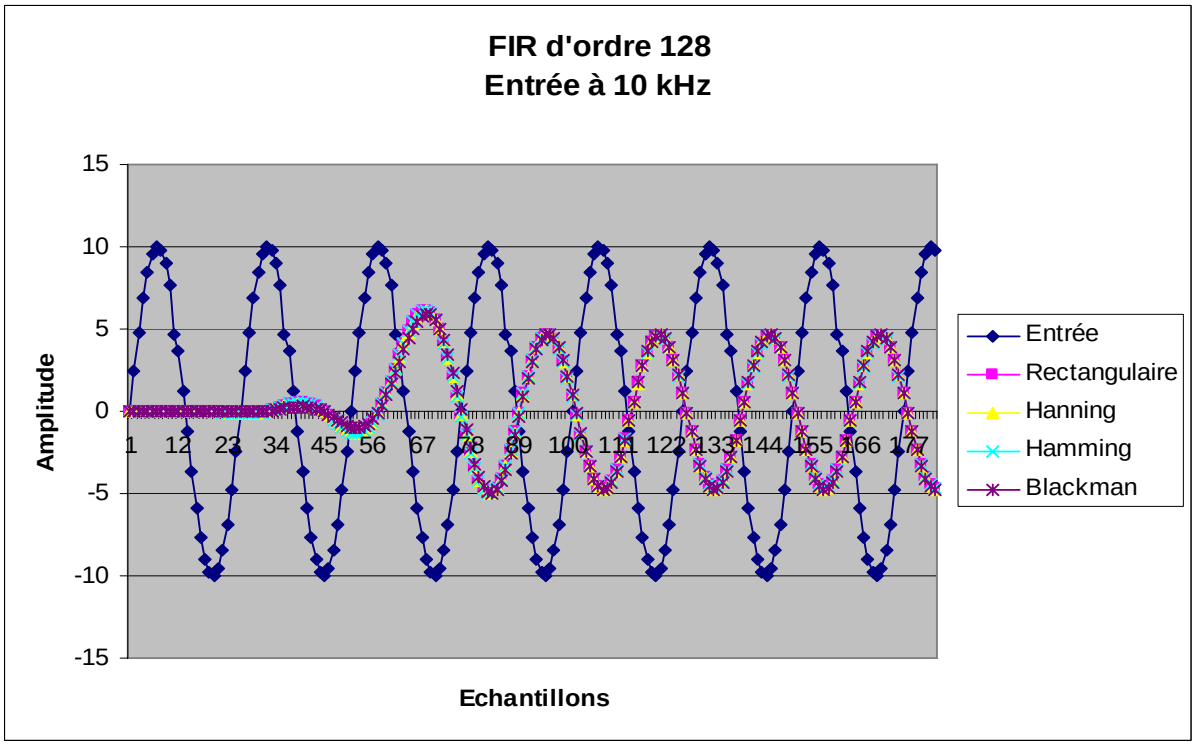

Figure 34 : Simulation du FIR 128 à 10 kHz.

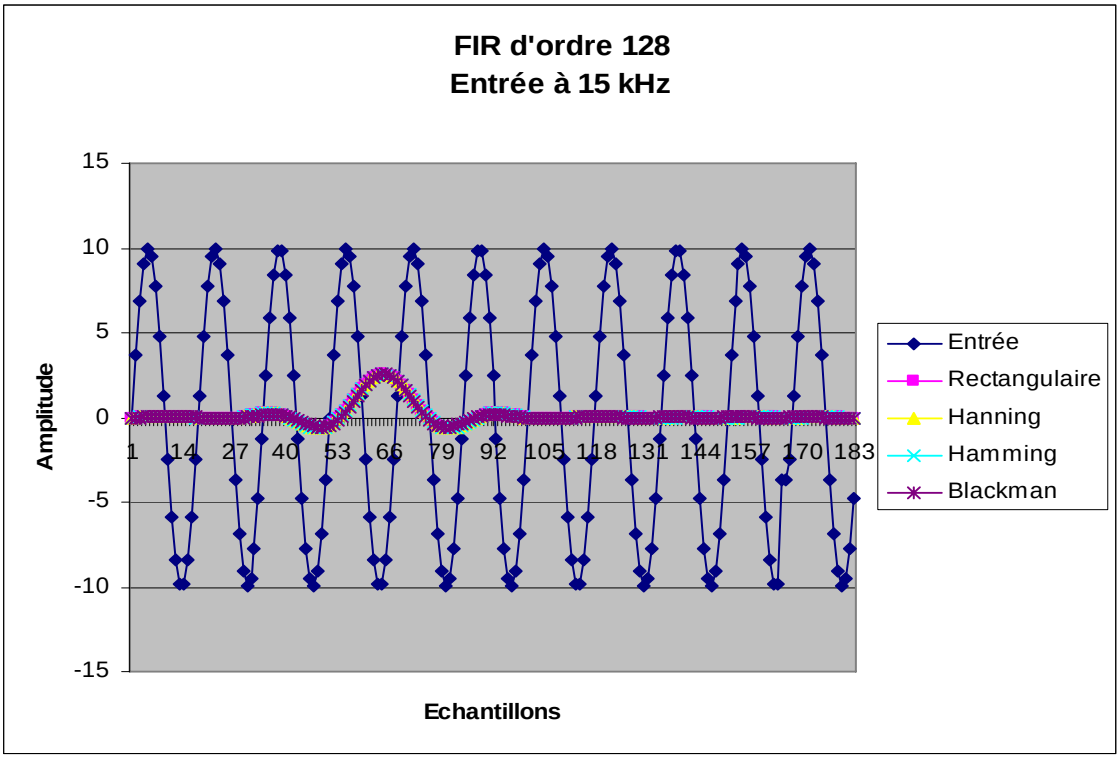

Figure 35 : Simulation du FIR 128 à 15 kHz.

Comme prévu, les 128 premiers échantillons de sortie sont inexploitables étant donné qu'ils ne servent uniquement qu'à remplir les RAM. On obtient donc un filtre plutôt sélectif comme le montre le tableau ci-dessous.

Fréquence du signal d'entrée	Atténuation
10 kHz	-6.3 dB
12.5 kHz	-22.6 dB
14 kHz	-40.5 dB
15 kHz	-60.5 dB

En effet, à 12.5 kHz, on a déjà une atténuation de plus de 22 dB, soit un signal de sortie 10 fois plus faible que le signal d'entrée.

Il reste donc au total, 15 RAM car 48 sont utilisées pour les filtres et une pour la table de motifs. Avec ces RAM restantes, il est possible de sauvegarder par moins de 60 filtres (à 128 échantillons) de gabarits tous différents. Cela sous entend qu'il est donc nécessaire de rajouter un ou plusieurs registres à la carte afin de paramétrer chaque filtre à un gabarit choisi…De plus, il n'est pas difficile d'insérer ces filtres dans le programme du FPGA initial. Il suffit juste de relier les échantillons de sortie du convertisseur série-parallèle à l'entrée du FIR et les différents signaux accompagnant ce filtre (horloge, validation de la donnée,…). Le signal de validation de la donnée, issue du convertisseur série-parallèle doit être retardé jusqu'à la fin du calcul du FIR.