Ecole Nationale Supérieure Louis-Lumière

Mémoire de fin d'étude

Traitement du signal audionumérique à partir des modulations sigma-delta et PCM

Paul Payen de La Garanderie

Directeur interne : Mohammed Elliq

Directeur externe : Olivier Tremois

Rapporteur : Laurent Millot

15 Juin 2015

Résumé

Dans ce mémoire, on présente les principes de fonctionnement des modulations PCM et sigma-delta appliqués au domaine de l'audionumérique. On détaille le fonctionnement des structures employées couramment par l'intermédiaire d'une tranche de console numérique équipée d'une fonction de filtrage (passe bas), de gain et de mélange (pour éventuellement en combiner plusieurs), en plus des convertisseurs analogique/numérique et numérique/analogique. Chacun de ces éléments est étudié en double, dans un cas en exploitant une modulation PCM, et dans l'autre cas en exploitant une modulation sigma-delta. Il est aussi question des structures permettant de passer d'une modulation PCM à une modulation sigma-delta et *vice versa*.

En plus du détail du fonctionnement théorique de chacun de ces éléments, on en fait une réalisation pratique. Cette réalisation pratique aborde dans un premier temps le domaine de l'électronique pour la réalisation des convertisseurs analogique/numérique et numérique/analogique. On peut alors créer un système matériel basé sur un FPGA pour traiter le signal audio à partir des éléments théoriques qui ont été développés et enfin, on peut aborder le problème de la réalisation de l'interface utilisateur en posant les bases de la communication entre les logiciels d'une part et les matériels d'autre part. La structure utilisée pour intégrer l'ensemble de ces composants logiciels et matériels est la plateforme de développement ZedBoard. L'ensemble des dispositifs audio étant regroupés sur la même structure physique il est plus facile d'en réaliser une comparaison, tant du point de vue de la complexité de réalisation, que de la complexité des éléments matériels de base auxquels ils font appel.

Mots clés

PCM, sigma-delta, DSD, convertisseur, numérique, analogique, FPGA, DSP, électronique

Abstract

In this thesis, we show working principles about PCM and sigma-delta modulations applied to digital audio. We detail the functioning of main digital structures by making a channel strip with filter (low-pass), gain, and mixing (to combine some), in addition to analog to digital converters and digital to analog converters. We study each of these elements in two cases: first by using the PCM modulation and secondly by using the sigma-delta modulation. We also talk about structures which permit to convert the sigma-delta modulation into PCM and vice versa.

Adding to details of theoretical functioning of each of these element, we do a practical realization of them. This realization, in a first part, enter in the electronic domain to realize analog to digital and digital to analog converters. Then, we can create some hardware on an FPGA to process the audio signal from theoretical elements we developed and finally, we talk about the problem of the realization of the user interface by giving basics of communication between software and hardware. We use the ZedBoard development board to integrate all the developed hardware and software. Because all of these audio systems are on the same physical structure it is easier to compare them in terms of the complexity of the realization and complexity of basic hardware cells needed.

Keywords

PCM, sigma-delta, DSD, converter, digital, analog, FPGA, DSP, electronic

I - Introduction

Le matériel audionumérique ne cesse de se développer et chaque jour apparaissent de nouveaux produits répondant soit à de nouveaux besoins soit cherchant à améliorer l'existant. L'amélioration d'un produit répond soit à un critère technique (moins de bruit de fond, moins de latence, moins de distorsion, …), soit à un critère d'ordre plus général (plus convivial, moins cher, moins encombrant, …). Ce mémoire va s'intéresser plus particulièrement à un critère d'évolution d'ordre technique : le codage du signal audio sous forme numérique.

D'un point de vue utilisateur il existe un codage largement répandu : le PCM. Le PCM est l'abréviation de Pulse Code Modulation signifiant modulation d'impulsion codée et ce système est la base du codage sur les CD audio. Il en existe plusieurs variantes permettant de changer la résolution (16 bits, 24 bits, 32 bits flottant, …) ou la fréquence d'échantillonnage (44.1 kHz, 48 kHz, 96 kHz, 192 kHz, …). Ce codage n'est pas le seul utilisé, il existe aussi le codage DSD pour Direct Stream Digital signifiant flux numérique direct et se basant sur la modulation sigma-delta. Ces deux termes peuvent sembler inconnus, mais le DSD est à la base du SACD (Super Audio CD) qui est un concurrent du DVD-Audio et le sigma-delta est à la base de la quasi-totalité des convertisseurs analogique/numérique (ADC, Analog to Digital Converter) et des convertisseurs numérique/analogique (DAC, Digital to Analog Converter) utilisés dans le domaine de l'audio en remplacement des convertisseurs exploitant le PCM.

De nos jours, le PCM étant utilisé par défaut pour la réalisation des traitements audio, il paraît intéressant de se poser la question de la pertinence d'une généralisation du recours à la modulation sigma-delta dans l'ensemble de la chaine audionumérique.

Le premier chapitre cherchera à répondre à la problématique du point de vue de l'utilisateur. Le deuxième chapitre exposera les éléments de traitement du signal pertinents. Et enfin, dans le troisième chapitre, on abordera la mise en œuvre pratique d'un dispositif audio. Dans ces trois chapitres on cherchera à élaborer deux "tranches de console" : l'une d'elle employant la modulation PCM (méthode classique) et l'autre employant la modulation sigma-delta. Elles seront composées d'un convertisseur analogique/numérique, d'une égalisation, d'un gain, d'un mélangeur (si on envisage d'utiliser plusieurs tranches) et enfin d'un convertisseur numérique/analogique.

II - Deux types de modulation

Dans ce premier chapitre, on présentera une description des deux types modulations PCM et sigma-delta du point de vue de l'utilisateur. Après un bref rappel historique, on présentera le fonctionnement de ces deux modulations. On terminera par la présentation de leurs applications dans le cas d'une console numérique.

1. Historique [1]

C'est à partir du début des années 1970 que le numérique est introduit dans les studios d'enregistrement et les deux premiers dispositifs à adopter le numérique sont les enregistreurs et les réverbérations. L'enregistrement en numérique permet de palier aux problèmes de dégradation des supports d'enregistrement et de la diminution de la qualité des enregistrements obtenus lorsque des opérations de lecture et/ou d'enregistrement se succèdent notamment durant le processus de mixage. Les réverbérations numériques constituent, quant à elles, un complément voir une alternative aux systèmes de réverbération à plaque et à ressort ; elles offrent de nouvelles possibilités dans la réalisation d'espaces sonores.

Les premiers enregistrements numériques ont eu lieu en parallèle des enregistrements analogiques, et en général ce sont ces derniers qui sont utilisés pour produire le mixage final. C'est finalement à partir des années 1980 avec l'arrivée du CD-Audio que le numérique se démocratise.

L'arrivée des réverbérations numériques précède l'arrivée des stations de travail audionumériques (parfois appelées DAW pour Digital Audio Workstation). Sur ces stations de travail, il devient possible de faire du montage de fichiers sons numériques, comme avec de la bande magnétique auparavant, et on peut, en plus, commencer à réaliser les premiers traitements audio. Tout ceci se développe aussi grâce à la démocratisation de l'informatique qui a lieu en parallèle. Pour ce qui est du traitement numérique, il a lieu soit directement sur des consoles de mixage numériques, soit au moyen de logiciels qui s'installent sur des ordinateurs.

Le numérique se démocratise entièrement dans les années 1990 avec l'apparition du "Home Studio". Aujourd'hui, toute la production audio utilise les technologies du numérique, mais on continue à utiliser l'analogique souvent par choix esthétique et non plus pas par nécessité.

L'évolution du numérique dans les studios est à mettre en relation avec l'évolution des technologies. Le PCM et le sigma-delta étant deux façons différentes de représenter le signal audio, les procédés nécessaires au travail de l'audio pour ces deux modes de représentation sont complètement différents. Les premiers éléments théoriques mathématiques posant les bases des ADC et des DAC ont été posés durant la première moitié du XX^{ème} siècle et exploitent le PCM pour représenter le signal analogique sous forme numérique. Il faudra toutefois attendre les années 1950 pour que soient mis au point ces convertisseurs. D'autres systèmes de modulation sont aussi explorés à partir des années 1950, comme la modulation DPCM (Différential Pulse Code Modulation ou modulation d'impulsion codée différentielle) où l'on ne code plus la valeur de l'échantillon, mais sa différence avec l'échantillon précédent. Enfin, les premiers convertisseurs dits sigma-delta sont produits à partir des années 1960.

Figure 1 : Photographie du "datrac" (PCM 50 kHz/11 bits) développé par Bernard M. Gordon en 1954 (48 cm x 38 cm x 68 cm, 68 kg) (d'après [1], figure 1.17 p.1.22) .

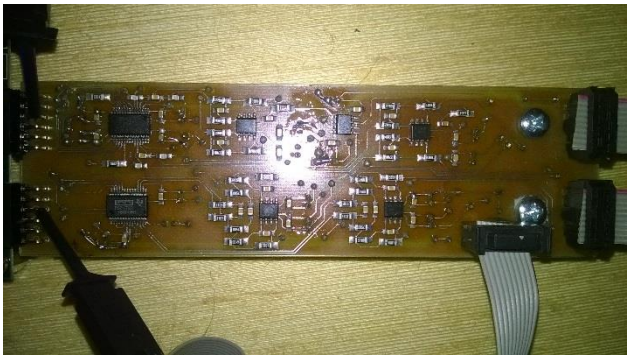

Figure 2 : Photographie du convertisseur sigma-delta (2,8224 MHz/1 bit) réalisé dans le cadre de ce travail (4,3 cm x 14,6 cm x 1,5 cm).

Le PCM s'est imposé pour effectuer les traitements parce qu'il est plus simple à conceptualiser et parce que sa forme le rapproche beaucoup de celle du signal analogique ; ce qui n'est pas le cas des autres types de modulation. Néanmoins, ces dernières se sont développées pour remédier à des problèmes techniques ponctuels dans la chaîne PCM. C'est le cas de la modulation sigma-delta qui permet de réduire le coût de fabrication des convertisseurs tout en apportant une amélioration considérable de leurs performances.

2. Modulation PCM [2] [3]

La modulation PCM consiste à prélever des valeurs (des échantillons) du signal d'entrée régulièrement dans le temps. Ces valeurs peuvent correspondre aux tensions délivrées par le dispositif audio (microphone, table de mixage, processeur d'effets, …) ou à des nombres numériques (lors de la requantification par exemple). Dans un logiciel audio, la forme d'onde correspond au tracé de l'ensemble des échantillons de la modulation PCM.

On prélève dans un premier temps les échantillons de la modulation PCM de façon régulière dans le temps (opération appelée échantillonnage). Le nombre d'échantillons par seconde est nommé fréquence d'échantillonnage. Idéalement, cette opération consiste à obtenir un nuage de points (les échantillons) tels qu'il existe une seule courbe reliant tous ces points et ayant un contenu fréquentiel compris entre 0 Hz et la moitié de la fréquence d'échantillonnage, appelée fréquence de Nyquist. Contrairement aux idées reçues, le signal PCM n'est ni un signal en créneaux (marches d'escaliers) ni un signal dont il faut relier les points avec des segments de droites : dans la modulation PCM, il n'y a "rien" entre deux points consécutifs. Pour que l'on puisse reconstituer ce "rien" lors de la conversion numérique/analogique, il faut prélever suffisamment de points lors de l'opération d'échantillonnage. Pour peu que l'on y ait fait attention, on peut théoriquement reconstituer parfaitement le signal lors du retour en analogique. En effet, l'augmentation de la fréquence d'échantillonnage ne permet pas d'améliorer la reproduction des fréquences déjà présentes. Concrètement, le passage de 48 kHz à 96 kHz ne permet pas d'améliorer la reproduction des fréquences comprises entre 0 Hz et 24 kHz, en revanche avec une fréquence d'échantillonnage de 96 kHz on peut en plus reproduire les fréquences comprises entre 24 kHz et 48 kHz. Modifier la fréquence d'échantillonnage revient donc à modifier la gamme des fréquences reproductibles.

Par ailleurs, les échantillons ont une certaine intensité ou encore amplitude qui doit se situer dans l'intervalle imposé par le convertisseur. Le convertisseur décompose cet intervalle en sous-intervalles et fait correspondre à chaque sous-intervalle une valeur numérique unique, opération que l'on appelle quantification. Le nombre de bits utilisé pour coder la valeur numérique est appelé résolution. La Figure 3 illustre le passage d'un signal analogique à un signal numérique PCM 5 bits. Le convertisseur numérique/analogique, quant à lui, associe à chaque valeur numérique (mesure d'une valeur analogique) une valeur unique dans chaque

sous-intervalle. L'écart entre cette valeur et la valeur analogique initiale correspond à l'erreur de quantification ou encore au bruit de fond intrinsèque du codage.

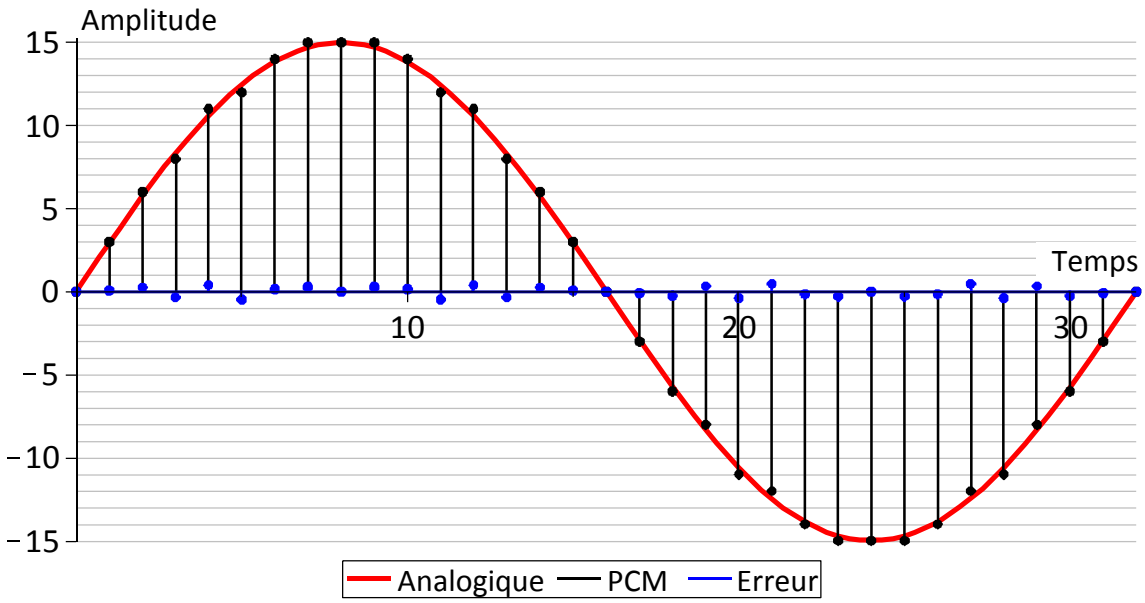

Figure 3 : Signal Analogique avec sa représentation en PCM 5 bits et l'erreur de quantification.

La dynamique du codage correspond au rapport entre la valeur la plus élevée et la valeur la plus faible, celle-ci étant associée au bruit de fond. Pour augmenter la dynamique du codage, il faut augmenter le nombre de valeurs possibles, donc le nombre de bits utilisés pour la quantification. En première approximation, on peut estimer que la dynamique du codage (en dB) comme étant égale à 6 fois le nombre de bits de quantification (on calculera plus loin une expression plus précise). Ainsi, la dynamique d'un CD-Audio, qui emploie un codage sur 16 bits, est de l'ordre de 96 dB (on calculera une valeur plus précise plus loin).

On a étudié les deux paramètres déterminant ce qu'il est possible d'enregistrer et de reproduire en PCM, cependant il existe d'autres facteurs pouvant limiter les performances du convertisseur d'un point de vue pratique, malgré l'utilisation d'une fréquence d'échantillonnage plus importante ou d'un pas de quantification plus fin. C'est le cas du phénomène de repliement spectral ou encore de la gigue d'horloge. On parlera uniquement du phénomène de repliement spectral (appelé aliasing en anglais), aussi bien pour le codage PCM que pour le codage sigma-delta. N'importe quelle fréquence au-delà de la fréquence de Nyquist est prise en compte par le convertisseur analogique/numérique, mais sera

retranscrite par le convertisseur par une fréquence entre 0 Hz et la fréquence de Nyquist. Cette fréquence apparente correspond à la valeur absolue de l'écart entre la vraie fréquence et le plus proche multiple de la fréquence d'échantillonnage. Par exemple, pour une fréquence d'échantillonnage de 48 kHz, une fréquence de 35 kHz donnera une fréquence apparente de $48 - 35 = 13$ kHz ; ainsi la fréquence de 35 kHz s'est repliée pour donner une fréquence apparente de 13 kHz, ce phénomène s'apparente à celui qui consiste à voir la roue d'une voiture changer de sens de rotation quand elle accélère. Toutes les fréquences réelles retranscrites par la même fréquence apparente interfèrent, par conséquent, il faut absolument éliminer toutes fréquences supérieures à la fréquence de Nyquist avant de procéder à l'échantillonnage.

3. Modulation sigma-delta [3]

La modulation sigma-delta consiste à numériser un signal en tenant compte de l'erreur de quantification. La valeur d'un échantillon numérique dépend de la totalité du signal qui est entré dans le convertisseur (pas seulement du signal actuel correspondant à l'échantillon comme dans le cas de la modulation PCM) ainsi que de tous les échantillons numériques déjà acquis. Grâce à la modulation sigma-delta, on va pouvoir réaliser des quantificateurs beaucoup plus précis simplement en augmentant la fréquence d'échantillonnage.

Concrètement on mesure l'écart entre le signal analogique et le signal numérisé puis on accumule cet écart. Si au moment de récupérer l'échantillon cette accumulation est positive, alors on récupère un bit numérique ayant la valeur 1 et, si cette accumulation est négative, alors on récupère un bit numérique ayant la valeur 0. Ces convertisseurs fonctionnent typiquement 64 fois plus vite qu'en PCM. On parle de facteur de sur-échantillonnage, ce qui signifie que pour réaliser un système équivalent à 44,1 kHz en PCM, on utiliserait une fréquence d'échantillonnage de 2822,4 kHz ou encore 2,8224 MHz en sigma-delta.

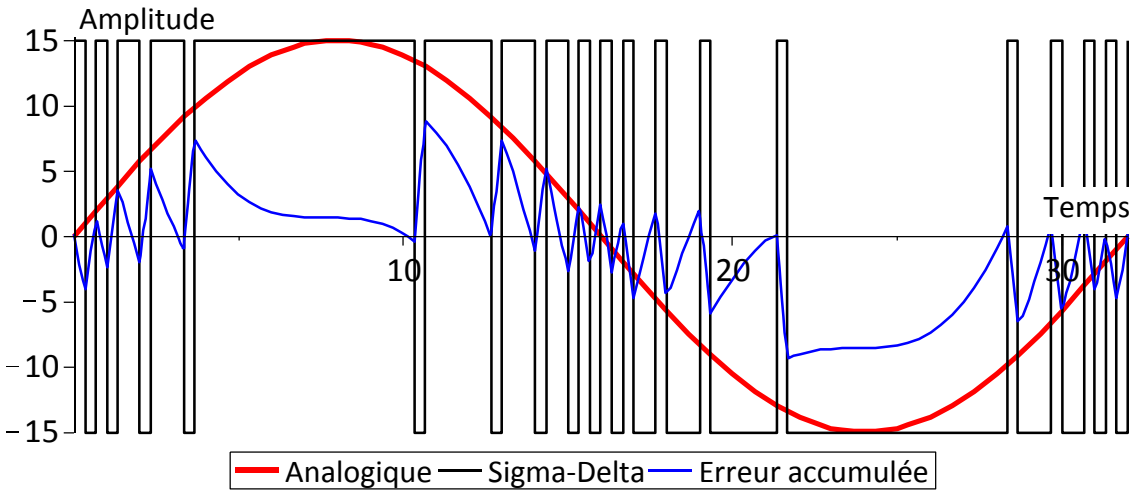

Figure 4 : Signal Analogique avec sa représentation en Sigma-Delta 1 bit et l'erreur accumulée.

Il s'agit bien là d'une convention car il existe toutes sortes de structures de convertisseur et un convertisseur ayant une fréquence d'échantillonnage plus importante n'a pas forcément de meilleures performances. En plus de la fréquence d'échantillonnage, il faut tenir compte de l'ordre du convertisseur c'est-à-dire de l'ordre du filtre qui le compose, qui se trouve être un intégrateur pour les convertisseurs d'ordre 1. L'accumulateur est le filtre le plus simple à fabriquer et c'est aussi le plus simple à comprendre mais, plus l'ordre du filtre utilisé sera important, meilleur sera le convertisseur. Les convertisseurs utilisés dans le domaine de l'audio utilisent généralement des filtres d'ordre 4 voire 5.

Il est beaucoup plus compliqué de définir la dynamique d'un convertisseur sigma-delta car elle dépend à la fois de la bande passante du signal et de l'ordre du convertisseur. En fait réaliser un convertisseur sigma-delta c'est réaliser un système qui va rajouter un bruit de fond au signal de façon à le rendre carré. Le problème de réalisation du convertisseur sigma-delta réside dans le report des fréquences composant ce bruit de fond le plus possible dans les ultra-sons de façon à ce qu'elles ne soient pas gênantes. De cette façon un convertisseur peut se révéler très médiocre sitôt que l'on considère des fréquences pour lesquelles il n'est pas prévu, même si on reste en dessous de la fréquence de Nyquist.

Il existe aussi des convertisseurs sigma-delta multi-bits, c'est-à-dire utilisant plusieurs bits au lieu d'un seul, comme ceux qui ont été considérés jusque-là. Comme on verra plus loin, dans ces convertisseurs on insère un quantificateur (identique à celui utilisé en PCM) à la sortie

de l'accumulateur et on verra qu'il existe un lien entre la fréquence d'échantillonnage et la résolution. Ces convertisseurs nécessitent une fréquence d'échantillonnage plus faible pour avoir la même qualité en termes de dynamique.

Dans le cas du modulateur sigma-delta, on a vu qu'il existe trois paramètres qui permettent de qualifier les limites de ce que l'on va pouvoir enregistrer et reproduire. De la même façon qu'en PCM on peut aussi avoir du repliement spectral dans les convertisseurs sigma-delta. Si l'on considère le même cas que pour le PCM, on constate qu'un convertisseur sigma-delta ayant un facteur de sur-échantillonnage de 64 n'est pas sensible au phénomène de repliement spectral pour une fréquence de 35 kHz. Il faut en réalité une fréquence d'au moins $2822{,}4 - 20 = 2802{,}4$ kHz avant d'observer le phénomène de repliement spectral. Les microphones sont bien incapables d'enregistrer de telles fréquences et on peut considérer qu'il n'y a pas de phénomène de repliement spectral dans les convertisseurs sigma-delta.

4. Principe de fonctionnement d'une console numérique

On peut maintenant établir la structure des convertisseurs utilisés. Comme pour des raisons de coût on utilise le sigma-delta pour les convertisseurs et le PCM pour les traitements audio, il faut donc insérer des convertisseurs sigma-delta/PCM et PCM/sigma-delta entre les ADC et DAC d'une part et les processeurs pour le traitement d'autre part. Etant donné que jusqu'à présent on utilise uniquement un traitement au format PCM, tous les convertisseurs analogique/numérique du commerce intègrent aussi le convertisseur sigma-delta correspondant. Il existe cependant quelques convertisseurs du commerce qui acceptent aussi d'envoyer ou de recevoir directement les informations au format sigma-delta en plus des informations au format PCM.

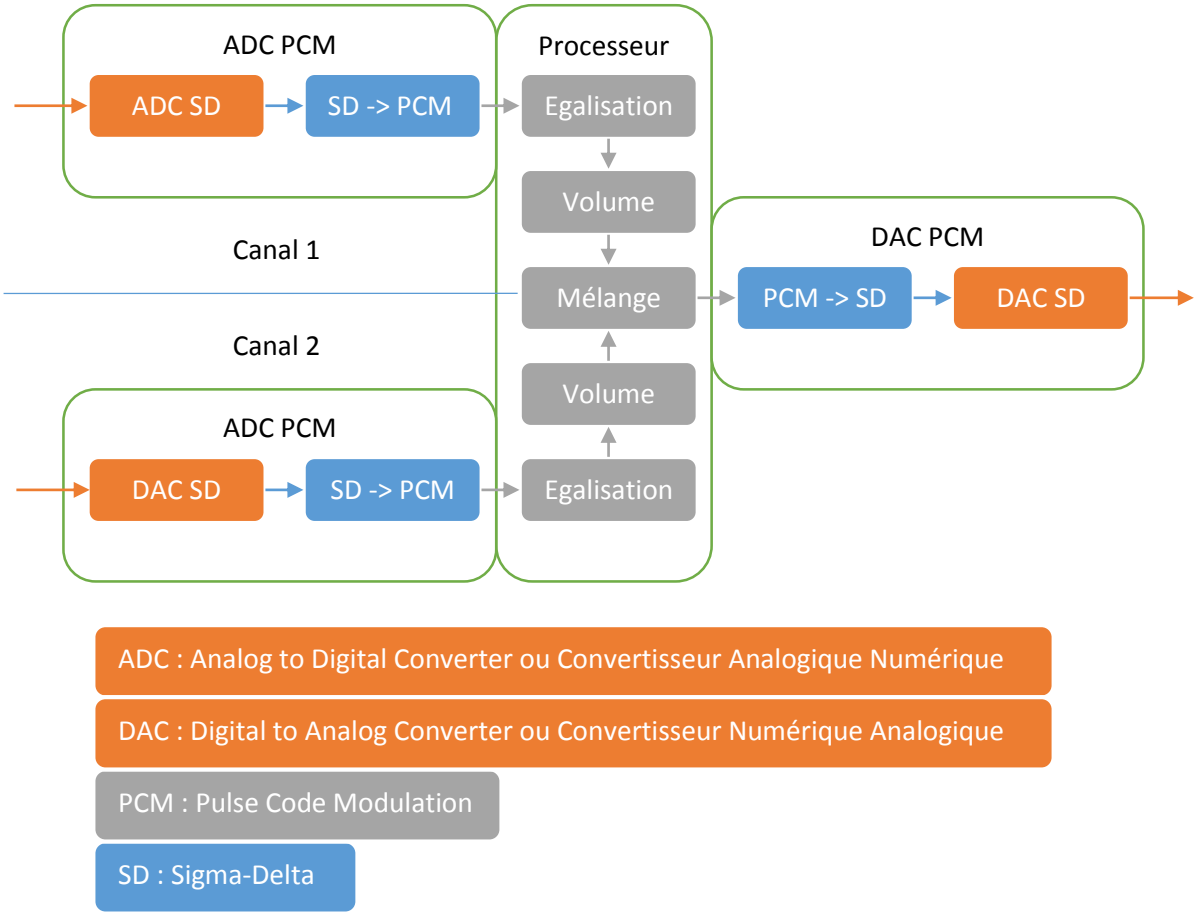

Figure 5 : Principe de fonctionnement d'une console audionumérique.

Le traitement au format PCM peut être réalisé soit par des processeurs spécialisés en traitement numérique du signal appelés DSP pour Digital Signal Processor (processeur de traitement du signal numérique), ce qui est le cas dans les consoles numériques, soit par des processeurs généralistes comme ceux équipant les ordinateurs. Il n'existe pas de différence en termes de qualité audio, la seule différence se situe au niveau du coût et de la vitesse d'exécution. Les DSP sont très rapides et peu chers pour peu que l'on traite de l'information audio, mais ils ne sont pas adaptés pour faire fonctionner un système d'exploitation ou un logiciel audio. En revanche il est possible d'ajouter des cartes à base de DSP (UAD2, carte Pro Tools HDX, PowerCore, …) dans un ordinateur pour effectuer les traitements audio sur celles-ci, mais elles n'accélèrent pas directement le système d'exploitation ou le logiciel audio.

D'un point de vue structurel le son d'un canal circule à travers trois composants successifs : le convertisseur analogique/numérique PCM, puis le processeur et enfin le convertisseur numérique/analogique PCM. Chaque liaison entre chaque couple de

composants introduit un retard supplémentaire d'un échantillon audio. Le processeur met un certain temps à faire le calcul et ce temps doit être un nombre entier d'échantillons pour que les convertisseurs numérique/analogique restent synchronisés avec les convertisseurs analogique/numérique. On obtient donc une latence minimale de 3 échantillons. Cette latence se calculant en nombre d'échantillons, il devient évident que plus la fréquence d'échantillonnage est importante, plus on pourra espérer avoir une faible latence.

Au niveau du temps de calcul, tous les traitements de base, comme le réglage du volume, l'égalisation ou encore le réglage de la dynamique sont relativement rapides à effectuer. Cependant, certains effets peuvent demander plusieurs échantillons pour le calcul, comme par exemple tous les traitements à base de transformée de Fourier (égaliseur linéaire par rapport à la phase, certaines réverbérations à convolution, …), et introduisent alors une latence plus importante dans le dispositif.

On voit ainsi que la structure au niveau des composants de la console numérique influe directement sur ses performances en termes de latence. Si la latence devient trop importante on peut essayer d'augmenter la fréquence d'échantillonnage, mais comme dans ce cas-là on a aussi davantage de calculs à effectuer, il est judicieux, voire nécessaire, d'utiliser des composants plus rapides.

5. Conclusion

Pour conclure, aujourd'hui, la modulation sigma-delta est présente dans la quasi-totalité des convertisseurs audio récents alors que la modulation PCM est presque exclusivement utilisée dans le domaine du traitement du signal audio. La modulation sigma-delta s'est imposée dans les convertisseurs car elle apporté une simplification de leur structure et donc une diminution des coûts. Cependant, son emploi a exigé la conception de convertisseurs pour passer d'un type de modulation à un autre afin de conserver la compatibilité avec ce qui existait auparavant.

Après avoir étudié les différences entre les deux types de modulations, il sera question, dans le prochain chapitre, de la possibilité de réaliser le traitement du signal audio numérique en utilisant directement la modulation sigma-delta.

III - Conversion et traitement du signal en PCM et en sigma-delta

Dans ce chapitre, on étudiera dans un premier temps le fonctionnement des deux modulateurs sigma-delta et PCM, puis comment calculer la plage dynamique accessible pour chacun d'eux. On étudiera ensuite leur intégration dans les convertisseurs analogique/numérique, numérique/analogique, sigma-delta/PCM et PCM/sigma-delta. Enfin, on observera les éléments théoriques en vue de la réalisation des dispositifs de gain, de mélange et de filtrage passe-bas pour chacun des deux types de modulations.

1. Principe de fonctionnement des modulateurs sigma-delta et PCM

a) Modulateur PCM

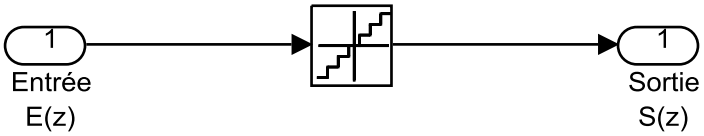

Figure 6 : Schéma de principe d'un modulateur PCM.

Le modulateur PCM (cf. Figure 6) est composé uniquement d'un quantificateur. Ce quantificateur étant un composant non-linéaire il est très difficile de réaliser une étude quantitative des caractéristiques globales du modulateur.

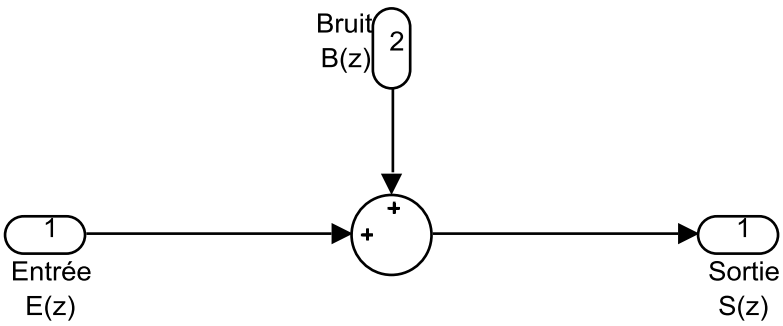

Figure 7 : Modèle linéaire équivalent à un modulateur PCM.

La Figure 7 montre comment linéariser le modulateur en remplaçant le quantificateur par une source de bruit blanc afin de pouvoir calculer la sortie en fonction des deux entrées :

$$S(z) = E(z) + B(z)$$

Par identification on obtient les fonctions de transfert H_{Signal}^{PCM} et H_{Bruit}^{PCM} caractérisant les évolutions de E et de B à travers le système.

$$H_{Signal}^{PCM}(z) = 1 \qquad\qquad H_{Bruit}^{PCM}(z) = 1$$

Ainsi, la sortie du modulateur est le résultat de l'addition du signal d'entrée avec un bruit blanc.

b) Modulateur sigma-delta d'ordre 1

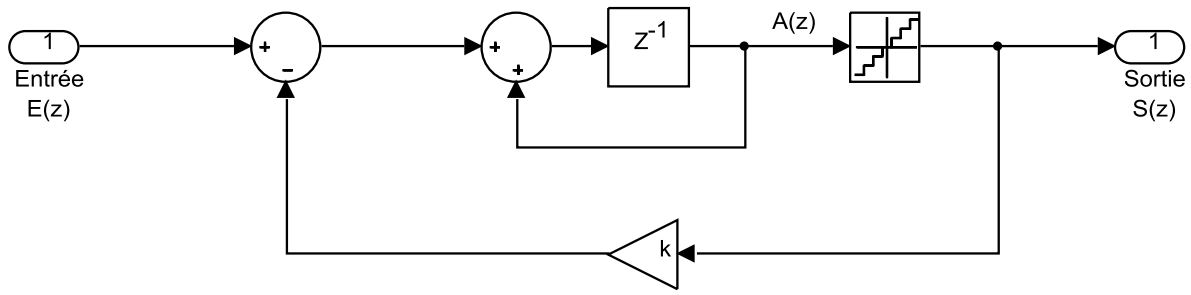

Figure 8 : Schéma de principe d'un modulateur sigma-delta d'ordre 1.

Comme illustré en Figure 8, le modulateur sigma-delta d'ordre 1 réalise la différence entre le signal d'entrée et le signal de sortie multiplié par le facteur k. Cette erreur est accumulée grâce à la boucle de rétroaction locale faisant intervenir un retard unitaire noté z^{-1} ; ensuite, l'erreur cumulée fait l'objet d'une quantification. Le gain k est calculé de manière à compenser les écarts en amplitude entre l'entrée et la sortie. Dans le cas d'une quantification sur 1 bit, on regarde le signe du nombre en sortie de l'accumulateur et la valeur de k correspond à la valeur maximale du signal d'entrée. Il faut noter qu'on peut utiliser cette structure aussi bien avec une entrée numérique qu'analogique. Dans le cas d'une entrée analogique, on remplace l'accumulateur par un intégrateur ; le gain k correspond à un convertisseur numérique analogique ayant les mêmes valeurs extrêmes que le signal analogique et le même nombre de bits que le quantificateur. Dans le cadre de ce mémoire, on supposera que la quantification se fait sur un seul bit.

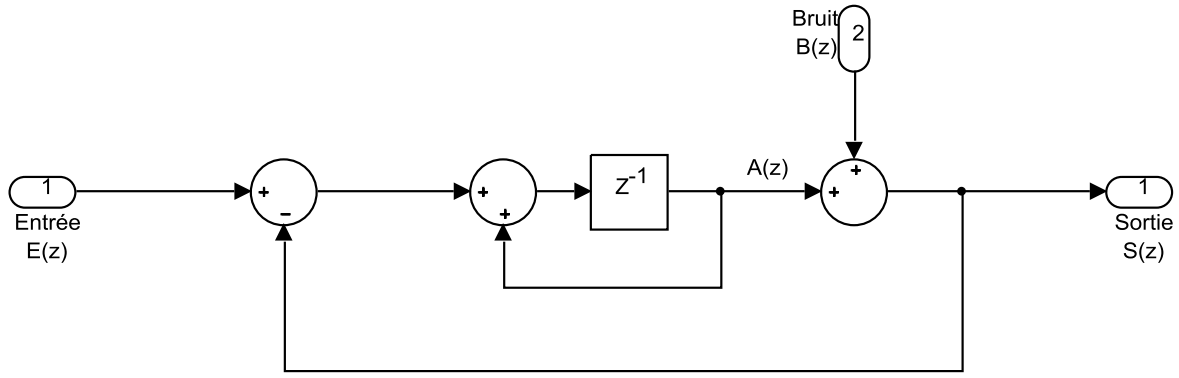

Figure 9 : Modèle linéaire équivalent à un modulateur sigma-delta d'ordre 1.

Comme pour le modulateur PCM, sur la Figure 9, on a linéarisé le modulateur sigma-delta en remplaçant le quantificateur par une source de bruit blanc et en choisissant un retour unitaire ; ce qui donne :

$$\begin{cases} S(z) = A(z) + B(z) \\ A(z) = z^{-1} \cdot \big(A(z) + E(z) - S(z)\big). \end{cases}$$

D'où, après élimination de $A(z)$:

$$S(z) = z^{-1} \cdot E(z) + (1 - z^{-1}) \cdot B(z).$$

Par identification, on obtient les fonctions de transfert H_{Signal}^{SD1} et H_{Bruit}^{SD1} caractérisant les évolutions de E et de B à travers le système :

$$H_{Signal}^{SD1}(z) = z^{-1} \qquad \text{et} \qquad H_{Bruit}^{SD1}(z) = 1 - z^{-1}.$$

En utilisant les fréquences normalisées, on peut tracer les courbes correspondantes ; ce qui est fait en Figure 10.

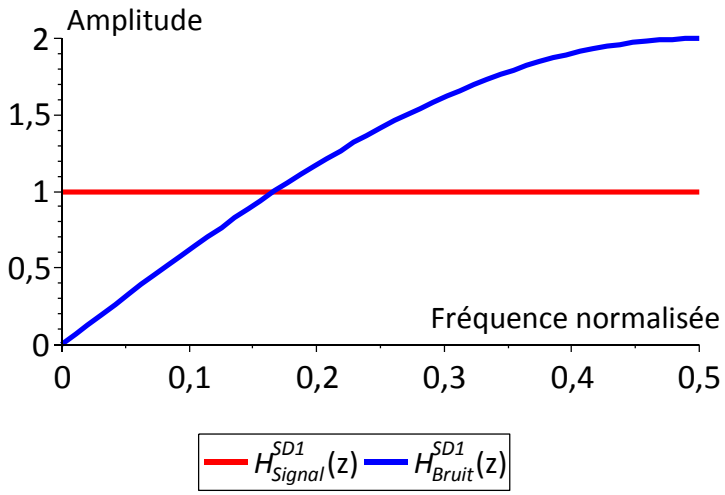

Figure 10 : Représentation des fonctions de transfert du signal et du bruit de fond en fonction de la fréquence normalisée.

En utilisant un modulateur de ce type, on constate que, si le signal d'entrée n'est pas filtré, le niveau du bruit de fond présent en sortie augmente avec la fréquence du fait du filtre $1 - z^{-1}$ qui se comporte comme un filtre passe-haut. La dynamique du modulateur dépend donc de la gamme de fréquences que l'on veut retranscrire (20 Hz à 20 kHz, pour le signal audio).

2. Dynamique pour un signal sinusoïdal [4]

a) Modulateur PCM

Pour le détail des calculs, on se réfèrera à l'Annexe 1.

Considérons un modulateur PCM linéaire quantifiant sur N_b bits l'intervalle $[-V_{max}, V_{max}]$ et ayant une fréquence d'échantillonnage de $N_o \cdot F_s$ où $N_o = 2^r$ est le facteur de sur-échantillonnage (r étant un entier positif) et F_s la fréquence d'échantillonnage cible. On cherche à donner la dynamique de ce modulateur pour un signal sinusoïdal ayant une fréquence inférieure à f_0.

Pour ce type de signal, la dynamique est donnée par la formule :

$$Dyn_0^{PCM} = 20 \cdot log_{10}\left(\frac{V_{max}}{\sqrt{2}}\right) - 10 \cdot log_{10}\left(P_{Bruit}^{f_0,PCM}\right),$$

où $P_{Bruit}^{f_0}$ correspond à la puissance du bruit de fond se situant sur la même plage de fréquences que le signal. On cherche à calculer dans un premier temps ce bruit de fond.

On constate que ce modulateur découpe l'intervalle $[-V_{max}, V_{max}]$ en 2^{N_b} sous-intervalles de même largeur :

$$\Delta = \frac{2 \cdot V_{max}}{2^{N_b}} = V_{max} \cdot 2^{1-N_b}.$$

Comme on a pu voir précédemment, l'erreur de quantification peut être considérée comme un bruit blanc évoluant dans l'intervalle de valeurs $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$ et sa puissance est donnée par la formule :

$$P_{Bruit}^{f_0,PCM} = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \int_{-f_0}^{f_0} \left|H_{Bruit}^{PCM}(f)\right|^2 df.$$

On obtient $H_{Bruit}^{PCM}(f)$ à partir de $H_{Bruit}^{PCM}(z)$ en substituant $e^{j \cdot \frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}}$ à z, avec j le nombre complexe tel que $j^2 = -1$. On a donc :

$$H_{Bruit}^{PCM}(f) = H_{Bruit}^{PCM}\left(e^{j \cdot \frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}}\right) = 1.$$

On a donc :

$$P_{Bruit}^{f_0,PCM} = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \int_{-f_0}^{f_0} 1 \, df = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \cdot 2 \cdot f_0.$$

La puissance de bruit de fond $P_{Bruit}^{f_0,PCM}$ est donc donnée par la formule :

$$P_{Bruit}^{f_0,PCM} = \frac{\Delta^2 \cdot f_0}{6 \cdot N_o \cdot F_s}.$$

On peut maintenant remplacer $P_{Bruit}^{f_0,PCM}$ par son expression dans la formule permettant de calculer la dynamique :

$$Dyn_0^{PCM} = 20 \cdot log_{10}\left(\frac{V_{max}}{\sqrt{2}}\right) - 10 \cdot log_{10}\left(P_{Bruit}^{f_0}\right),$$

$$Dyn_0^{PCM} = 20 \cdot log_{10}\left(\frac{V_{max}}{\sqrt{2}}\right) - 10 \cdot log_{10}\left(\frac{\Delta^2 \cdot f_0}{6 \cdot N_o \cdot F_s}\right).$$

En utilisant les lois usuelles de calcul sur les logarithmes, on obtient l'expression suivante :

$$Dyn_0^{PCM} = 20 \cdot log_{10}\left(\frac{V_{max}}{\Delta}\right) - 10 \cdot log_{10}\left(\frac{f_0}{3 \cdot F_s}\right) + 10 \cdot log_{10}(N_o),$$

et en remplaçant Δ et N_o par leurs expressions, on a :

$$Dyn_0^{PCM} = 20 \cdot log_{10}(2^{N_b-1}) - 10 \cdot log_{10}\left(\frac{f_0}{3 \cdot F_s}\right) + 10 \cdot log_{10}(2^r).$$

En sachant que $20 \cdot log_{10}(2) \simeq 6{,}02$ et que $10 \cdot log_{10}(2) \simeq 3{,}01$, on obtient finalement :

$$Dyn_0^{PCM} = 6{,}02 \cdot (N_b - 1) + 4.77 - 10 \cdot log_{10}\left(\frac{f_0}{F_s}\right) + 3{,}01 \cdot r.$$

On constate alors que multiplier par 2 la fréquence d'échantillonnage, c'est-à-dire augmenter r d'une unité, puisque r correspond au facteur de sur-échantillonnage, est analogue à ajouter 0,5 à N_b. On peut donc écrire :

$$Dyn_0^{PCM} = 6{,}02 \cdot (N_b - 1 + 0{,}5 \cdot r) + 4.77 - 10 \cdot log_{10}\left(\frac{f_0}{F_s}\right).$$

On peut donc dire que multiplier la fréquence d'échantillonnage par 2 est équivalent à rajouter 0,5 bits à la précision du quantificateur pour peu que l'on considère un signal d'entrée évoluant dans la même plage de fréquences.

On peut calculer la dynamique d'un modulateur PCM, ayant les caractéristiques suivantes (cas du CD-Audio) :

- quantification sur 16 bits ;
- facteur de sur-échantillonnage de $1 = 2^0$ à la fréquence cible de 44100 Hz ;
- fréquence maximale du signal à numériser de 22050 Hz ;

ce qui conduit à une dynamique pour le modulateur PCM égal à :

$$Dyn_0^{PCM} \simeq 98\ dB.$$

b) Modulateur sigma-delta d'ordre 1

Pour calculer la dynamique du modulateur sigma-delta, on procède de façon analogue au modulateur PCM.

Considérons un modulateur sigma-delta d'ordre 1 reprenant les caractéristiques de fréquence d'échantillonnage du modulateur PCM précédent et quantifiant sur 1 bit. On redonne la formule de calcul de la dynamique :

$$Dyn_0^{SD1} = 20 \cdot log_{10}\left(\frac{V_{max}}{\sqrt{2}}\right) - 10 \cdot log_{10}\left(P_{Bruit}^{f_0,SD1}\right).$$

Ce modulateur découpe l'intervalle $[-V_{max}, V_{max}]$ en deux parties $[-V_{max}, 0]$ (pour les valeurs négatives) et $[0, V_{max}]$ (pour les valeurs positives). On a :

$$\Delta = V_{max}.$$

On substitue à nouveau $e^{j\cdot\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}}$ à z dans la fonction de transfert $H_{Bruit}^{SD1}(z)$ du bruit de fond pour le modulateur sigma-delta :

$$H_{Bruit}^{SD1}(z) = 1 - z^{-1},$$
$$H_{Bruit}^{SD1}(f) = 1 - e^{-j\cdot\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}}.$$

Par définition de l'exponentielle complexe, on obtient :

$$H_{Bruit}^{SD1}(f) = 1 - \left[\cos\left(-\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}\right) + j\cdot\sin\left(-\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}\right)\right].$$

En remarquant que $\cos(-x) = \cos(x)$ et $\sin(-x) = -\sin(x)$, on obtient finalement :

$$H_{Bruit}^{SD1}(f) = \left[1 - \cos\left(\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}\right)\right] + j\cdot\sin\left(\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}\right).$$

Calculons maintenant $\left|H_{Bruit}^{SD1}(f)\right|^2$:

$$\left|H_{Bruit}^{SD1}(f)\right|^2 = \left|\left[1 - \cos\left(\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}\right)\right] + j\cdot\sin\left(\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}\right)\right|^2,$$
$$\left|H_{Bruit}^{SD1}(f)\right|^2 = \left[1 - \cos\left(\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}\right)\right]^2 + \sin^2\left(\frac{2\cdot\pi\cdot f}{N_o\cdot F_s}\right).$$

Après développement du premier terme :

$$\left|H_{Bruit}^{SD1}(f)\right|^2 = 1 - 2 \cdot \cos\left(\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}\right) + \cos^2\left(\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}\right) + \sin^2\left(\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}\right),$$

et sachant que $\cos^2(x) + \sin^2(x) = 1$, on obtient au final :

$$\left|H_{Bruit}^{SD1}(f)\right|^2 = 2 - 2 \cdot \cos\left(\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}\right).$$

Dans le cadre de la modulation sigma-delta, on a un facteur de sur-échantillonnage important (égal ou supérieur à 64, en général), ce qui implique que $\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}$ soit proche de 0, d'où le choix d'un équivalent à $\cos(x)$ quand x est proche de 0 :

$$\cos\left(\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}\right) \underset{x \to 0}{\sim} 1 - \frac{1}{2} \cdot \left(\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}\right)^2.$$

En combinant ce résultat au résultat précédent, on peut écrire :

$$\left|H_{Bruit}^{SD1}(f)\right|^2 = 2 - 2 \cdot \left(1 - \frac{1}{2} \cdot \left(\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}\right)^2\right),$$

ce qui donne après développement :

$$\left|H_{Bruit}^{SD1}(f)\right|^2 = \left(\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}\right)^2.$$

On peut maintenant calculer la puissance de bruit de fond pour des fréquences inférieures à f_0 en remplaçant $\left|H_{Bruit}^{SD1}(f)\right|^2$ par son expression :

$$P_{Bruit}^{f_0,SD1} = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \int_{-f_0}^{f_0} \left|H_{Bruit}^{SD1}(f)\right|^2 df,$$

$$P_{Bruit}^{f_0,SD1} = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \int_{-f_0}^{f_0} \left(\frac{2 \cdot \pi \cdot f}{N_o \cdot F_s}\right)^2 df.$$

En sortant les constantes de l'intégrale, on a :

$$P_{Bruit}^{f_0,SD1} = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \cdot \left(\frac{2 \cdot \pi}{N_o \cdot F_s}\right)^2 \int_{-f_0}^{f_0} f^2 \, df,$$

et on peut maintenant chercher une primitive de f^2 :

$$P_{Bruit}^{f_0,SD1} = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \cdot \left(\frac{2 \cdot \pi}{N_o \cdot F_s}\right)^2 \cdot \left[\frac{f^3}{3}\right]_{-f_0}^{f_0}.$$

On obtient alors :

$$P_{Bruit}^{f_0,SD1} = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \cdot \left(\frac{2 \cdot \pi}{N_o \cdot F_s}\right)^2 \cdot \left(\frac{{f_0}^3}{3} - \frac{(-f_0)^3}{3}\right).$$

Mais, sachant que $(-1)^3 = -1$, on peut écrire :

$$P_{Bruit}^{f_0,SD1} = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \cdot \left(\frac{2 \cdot \pi}{N_o \cdot F_s}\right)^2 \cdot \frac{2 \cdot {f_0}^3}{3}.$$

En réalisant des regroupements, on obtient l'expression finale :

$$P_{Bruit}^{f_0,SD} = \frac{\Delta^2}{12} \cdot \frac{\pi^2}{3} \cdot \left(\frac{2 \cdot f_0}{N_o \cdot F_s}\right)^3.$$

D'après cette expression, on peut maintenant calculer la dynamique du modulateur sigma-delta :

$$Dyn_0^{SD1} = 20 \cdot log_{10}\left(\frac{V_{max}}{\sqrt{2}}\right) - 10 \cdot log_{10}\left(P_{Bruit}^{f_0,SD1}\right),$$

$$Dyn_0^{SD1} = 20 \cdot log_{10}\left(\frac{V_{max}}{\sqrt{2}}\right) - 10 \cdot log_{10}\left(\frac{\Delta^2}{12} \cdot \frac{\pi^2}{3} \cdot \left(\frac{2 \cdot f_0}{N_o \cdot F_s}\right)^3\right).$$

De la même façon qu'avec le modulateur PCM, en utilisant les lois usuelles de calcul sur les logarithmes, on obtient :

$$Dyn_0^{SD1} = 20 \cdot log_{10}\left(\frac{V_{max}}{\Delta}\right) - 20 \cdot log_{10}\left(\frac{\pi \cdot \sqrt{2}}{6}\right) - 30 \cdot log_{10}\left(\frac{f_0}{F_s}\right) + 30 \cdot log_{10}\left(\frac{N_o}{2}\right).$$

De même que pour le modulateur PCM, on remplace Δ et N_o par leurs expressions :

$$Dyn_0^{SD1} = 20 \cdot log_{10}(2^{N_b - 1}) - 20 \cdot log_{10}\left(\frac{\pi \cdot \sqrt{2}}{6}\right) - 30 \cdot log_{10}\left(\frac{f_0}{F_s}\right) + 30 \cdot log_{10}\left(\frac{2^r}{2}\right),$$

ce qui donne :

$$Dyn_0^{SD1} = 6{,}02 \cdot (Nb - 1) + 2.61 - 30 \cdot log_{10}\left(\frac{f_0}{F_s}\right) + 9{,}03 \cdot (r - 1).$$

Pour le modulateur sigma-delta, on constate que le doublement de la fréquence d'échantillonnage, c'est-à-dire ajouter 1 au facteur de sur-échantillonnage r, est analogue à ajouter 1,5 à N_b. D'où :

$$Dyn_0^{SD1} = 6{,}02 \cdot \big(Nb - 1 + 1{,}5 \cdot (r-1)\big) + 2.61 - 30 \cdot log_{10}\left(\frac{f_0}{F_s}\right).$$

Pour le modulateur sigma-delta d'ordre 1, doubler la fréquence d'échantillonnage est équivalent à ajouter 1.5 bit à la précision du quantificateur.

Calculons la valeur numérique de la dynamique du modulateur sigma-delta d'ordre 1 décrit ci-après est :

- quantification sur 1 bit ;
- facteur de sur-échantillonnage de $64 = 2^6$ à la fréquence cible de 44100 Hz ;
- fréquence maximale du signal à numériser de 22 050 Hz ;

ce qui conduit à une dynamique pour le modulateur sigma-delta de :

$$Dyn_0^{SD1} \simeq 57\ dB.$$

Comme ce modulateur n'est donc pas capable d'égaler la qualité du CD-Audio pour un signal sinusoïdal, il faudrait augmenter davantage le facteur de sur-échantillonnage. Une autre solution consisterait à augmenter le nombre de bits de précision équivalent au doublement de la fréquence d'échantillonnage, ce qui est possible en augmentant l'ordre du convertisseur. C'est pour cette raison que les convertisseurs fonctionnent avec un facteur de sur-échantillonnage de 64 voire 128 mais pour un filtre d'ordre de 4 voire 5. Néanmoins, pour des raisons de simplification, lorsque l'on parlera de modulateur sigma-delta dans la suite du document, sauf mention explicite, on se réfèrera au modulateur d'ordre 1 quantifiant sur 1 bit étudié précédemment.

3. Convertisseurs

a) Analogique vers numérique

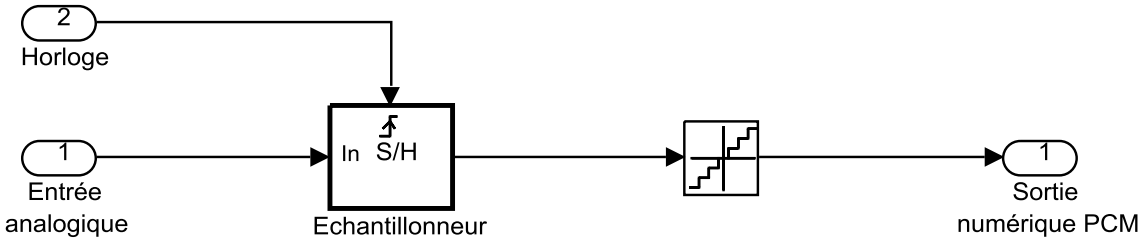

Figure 11 : Fonctionnement d'un convertisseur analogique numérique PCM.

Les modulateurs présentés précédemment ne tiennent pas compte de la nature (analogique ou numérique) du signal d'entrée. Dans le cas du modulateur PCM, on va insérer l'échantillonneur avant le quantificateur (cf. Figure 11) ; pour le modulateur sigma-delta, on insère l'échantillonneur entre le quantificateur et le départ pour la rétroaction (cf. Figure 12).

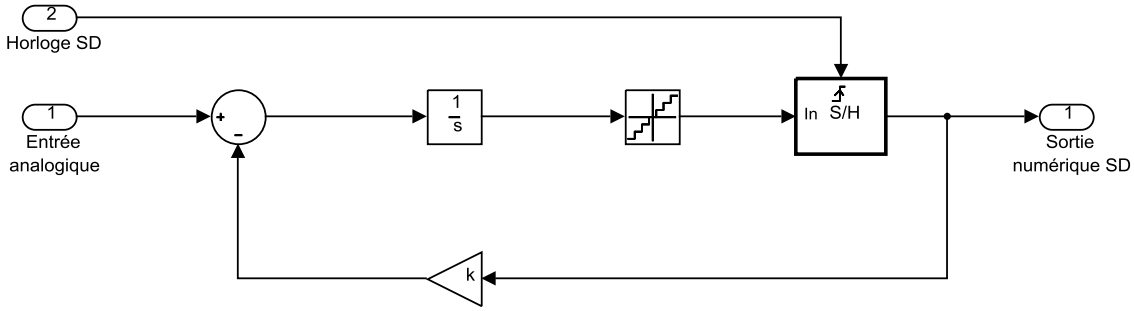

Figure 12 : Fonctionnement d'un convertisseur analogique numérique sigma-delta.

b) Numérique vers analogique

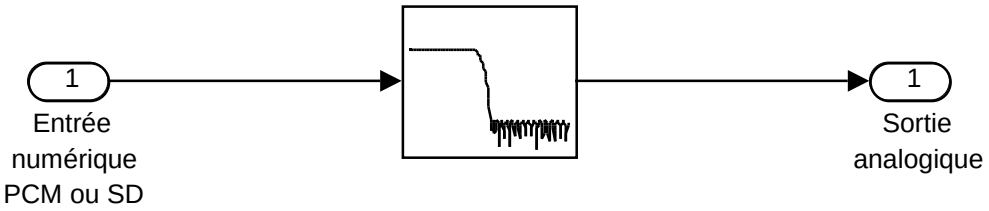

Figure 13 : Fonctionnement d'un convertisseur numérique/analogique PCM et sigma-delta.

Dans les deux cas (PCM et sigma-delta), on doit réaliser un filtrage passe-bas (cf. Figure 13). Dans le cadre de la modulation PCM, ce filtre est appelé filtre de lissage et sert à reconstituer le signal analogique à partir du nuage de points numériques. Pour la modulation sigma-delta, le filtre permet de supprimer le bruit de fond qui se situe dans les hautes fréquences, bruit qui peut perturber le fonctionnement des équipements audio analogiques.

c) Sigma-delta vers PCM

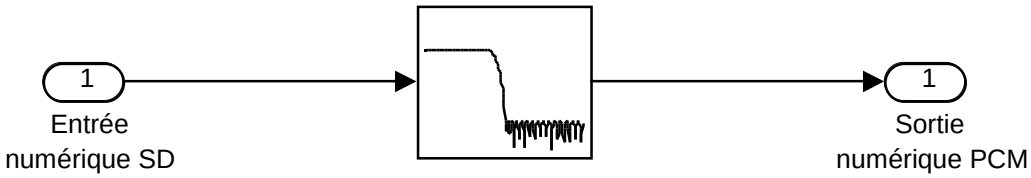

Figure 14 : Fonctionnement d'un convertisseur sigma-delta vers PCM.

Pour passer de la modulation sigma-delta à la modulation PCM (cf. Figure 14), il faut donc filtrer le bruit de fond qui se situe dans les hautes fréquences, de la même façon que si on cherchait à revenir en analogique, sauf que cette fois on va utiliser un filtre numérique.

d) PCM vers sigma-delta

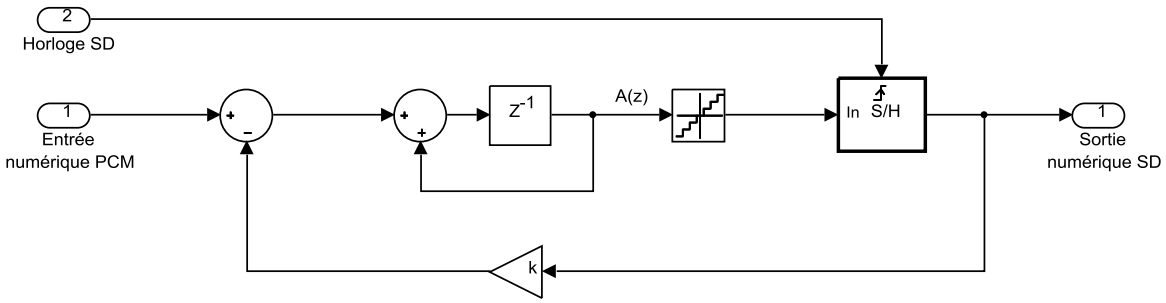

Figure 15 : Fonctionnement d'un convertisseur PCM vers sigma-delta.

Pour passer de la modulation PCM à la modulation sigma-delta (cf. Figure 15), il suffit d'utiliser le modulateur sigma-delta avec une entrée numérique. Pour faire correspondre la fréquence d'échantillonnage du signal d'entrée PCM, beaucoup plus élevée (typiquement on multiplie par 64), à celle du signal de sortie sigma-delta, on recopie les échantillons PCM autant de fois que nécessaire (cf. Figure 16).

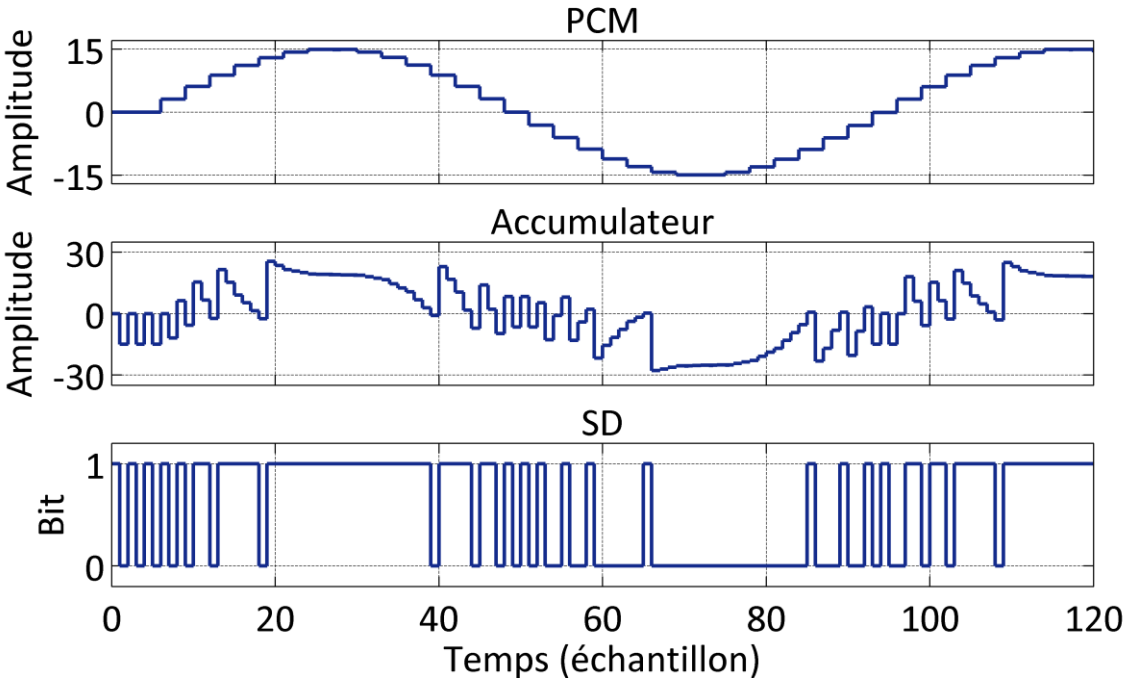

Figure 16 : Simulation d'une conversion PCM vers sigma-delta avec une fréquence d'échantillonnage de sortie 3 fois supérieure à celle d'entrée.

4. Dispositif de gain

a) Gain en PCM

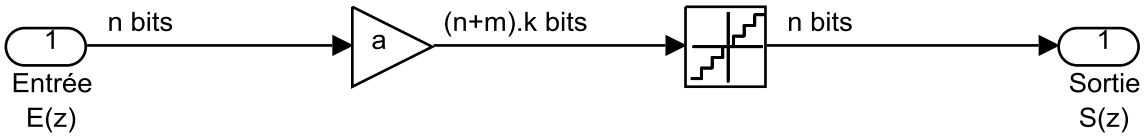

Figure 17 : Schéma de principe du gain sur une modulation PCM.

Si on considère que les nombres binaires sont codés en virgule fixe (comme pour le CD-Audio, par exemple), que les échantillons sont codés sur n bits signés et que le gain à appliquer est codé sur m.k bits, c'est-à-dire m+k bits au total dont m bits avant la virgule (représentant la partie entière) et k bits après (représentant la partie fractionnaire) (cf. Annexe 2 pour les fondamentaux du calcul en binaire), alors à la sortie d'un multiplieur, on obtient des échantillons codés sur (n+m).k bits. Comme on a donc augmenté le nombre de bits, ce qui peut éventuellement poser des problèmes pour le dispositif suivant dans la chaîne

puisque celui-ci attend un nombre codé sur n bits, on requantifie donc les échantillons (cf. Figure 17), en supprimant les m bits de poids fort par saturation et les k bits de poids faible par un arrondi quelconque ou par une troncature.

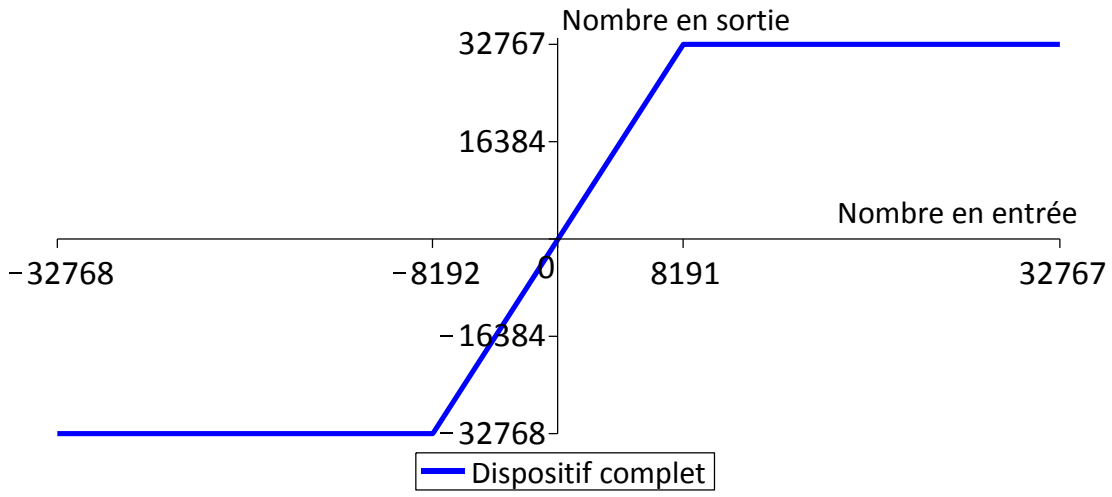

Figure 18 : Illustration du phénomène de saturation pour une multiplication de facteur 3 d'échantillons codés sur 16 bits.

Sur la Figure 18, les échantillons sont codés sur 16 bits et possèdent des valeurs comprises entre -32768 et 32767. Si on les multiplie par 3, c'est-à-dire par un nombre codé sur 2 bits, on obtient des échantillons compris entre -98304 et 98301, ce qui nécessite 18 bits. Pour la requantification, on ne garde que les 16 bits de poids faible en réalisant une saturation ; cela signifie que tous les nombres à l'entrée du quantificateur inférieurs à -32768 seront remplacés par -32768 et tous ceux supérieurs à 32767 seront remplacés par 32767, ce qui correspond à des valeurs de -8192 et 8191 à l'entrée du dispositif de gain. Ce phénomène de saturation est susceptible d'apparaître seulement lorsque le gain devient supérieur à 1, c'est-à-dire lorsqu'on augmente le volume sonore.

b) Gain en sigma-delta

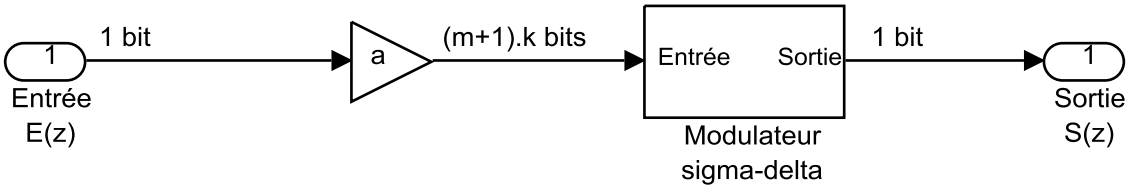

Figure 19 : Schéma de principe du gain sur une modulation sigma-delta.

Dans le cas de la modulation sigma-delta, de la même façon que pour la modulation PCM, il faut à nouveau moduler le signal obtenu à la sortie de l'opérateur de gain pour obtenir un signal sur 1 bit (cf. Figure 19). Pour réaliser correctement la multiplication, il faut considérer que l'unique bit formant la modulation sigma-delta correspond aux valeurs numériques +1, dans le cas où il est à l'état haut, et -1, dans le cas où il est à l'état bas, et on peut ainsi remplacer le gain a par un multiplexeur 2 vers 1 ayant pour entrées –a et a, et comme sélecteur d'entrée la modulation sigma-delta (voir en Figure 20).

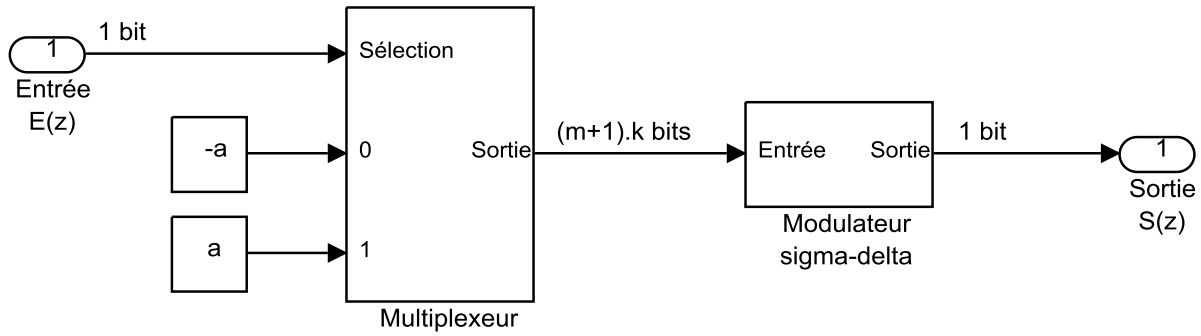

Figure 20 : Schéma de principe du gain sur une modulation sigma-delta en utilisant un multiplexeur.

De cette manière, on s'affranchit complétement de l'utilisation d'un multiplieur qui est coûteux à réaliser d'un point de vue matériel. La Figure 21 est une simulation de l'application d'un gain sur une sinusoïde, en utilisant ce procédé.

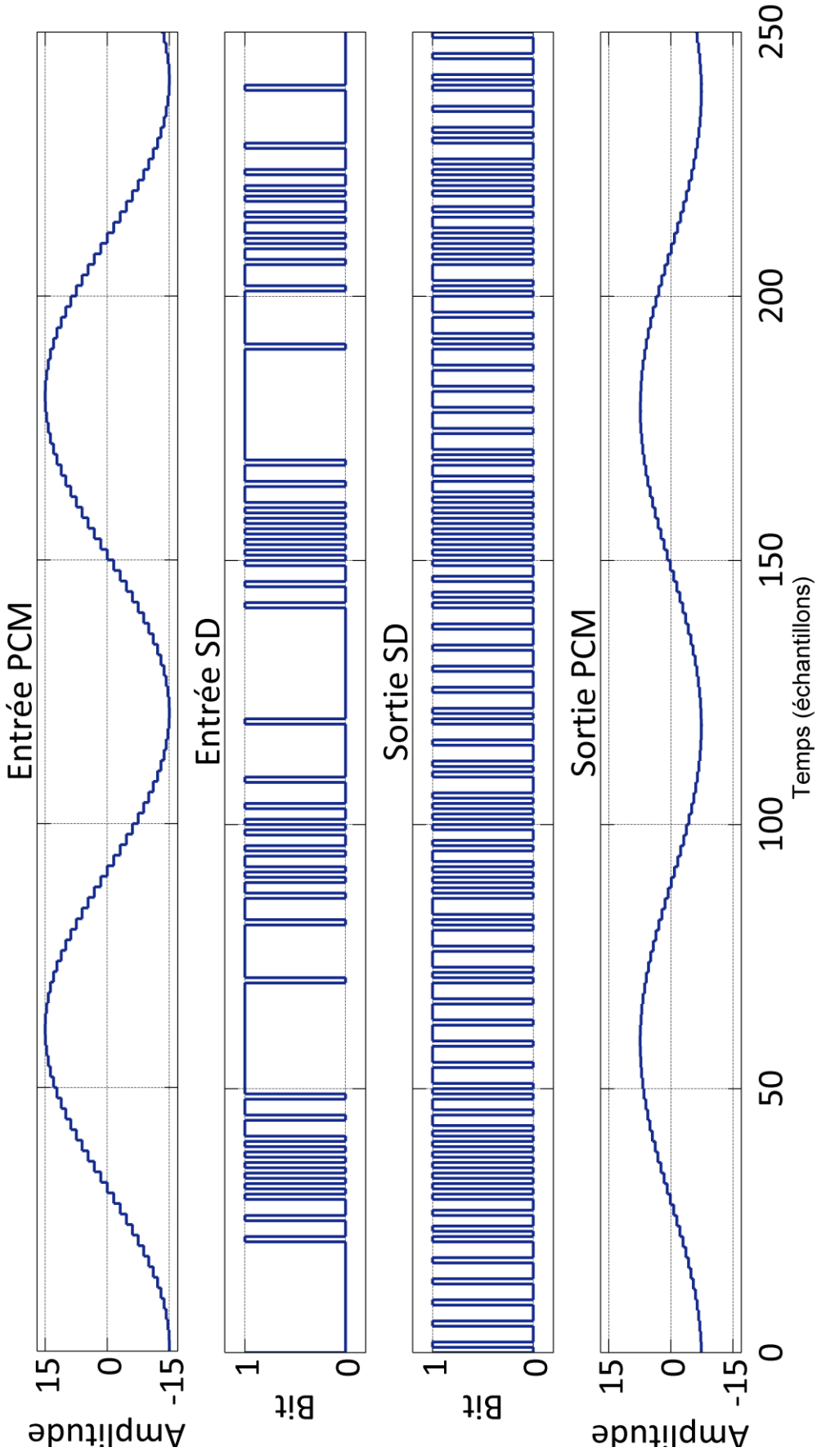

Figure 21 : Simulation d'un gain de 0,5 (-6 dB) sur une modulation sigma-delta générée à partir d'une modulation PCM et un facteur de sur-échantillonnage de 2.

5. Dispositif de mélange (ou de sommation)

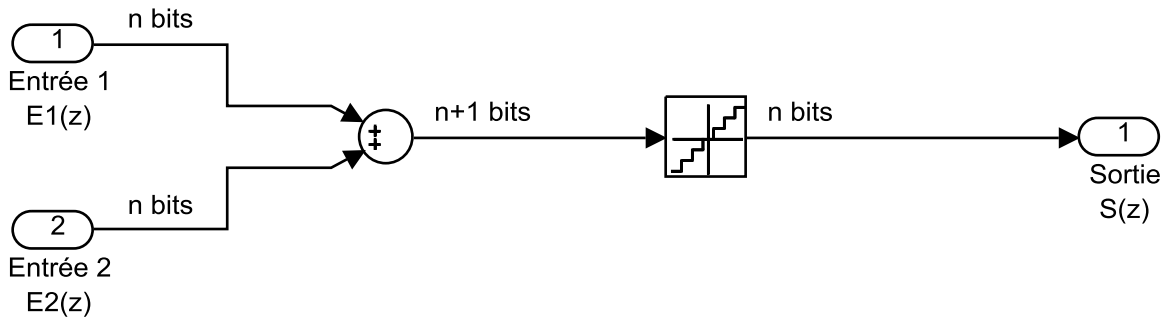

Figure 22 : Schéma de principe du mélange sur une modulation PCM.

Lorsque l'on somme, comme sur la Figure 22, deux échantillons codés sur n bits provenant de deux canaux, on obtient un échantillon codé sur n+1 bits et d'une façon plus générale, lorsqu'on somme 2^m échantillons de n bits, on obtient un échantillon codé sur n+m bits. Comme on a pu le voir dans le cas du dispositif de gain, il faut requantifier les échantillons sur n bits en ne conservant que les n bits de poids faible en réalisant une saturation.

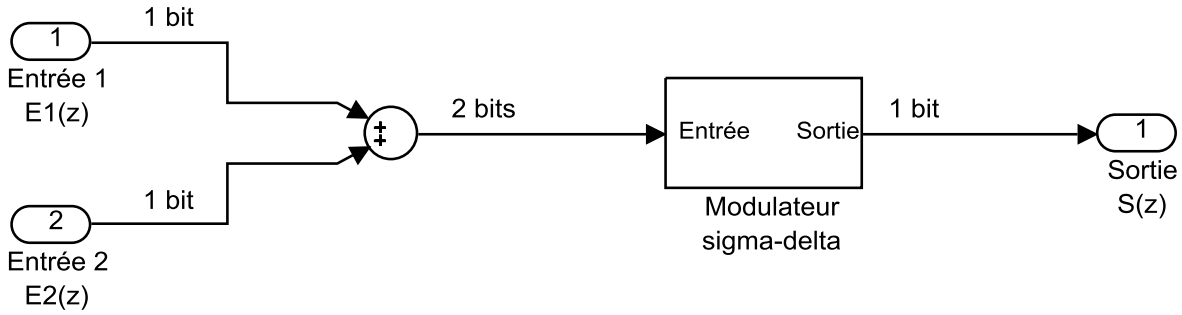

Figure 23 : Schéma de principe du mélange sur une modulation sigma-delta.

De même qu'en PCM, la somme de 2^m modulations sigma-delta donne des échantillons codés sur m+1 bits et on a donc recours à nouveau à un modulateur sigma-delta pour avoir des échantillons codés sur 1 bit en sortie [5]. La Figure 23 correspond au cas où m=1 et la Figure 24 est une simulation de la somme de deux sinusoïdes par ce procédé.

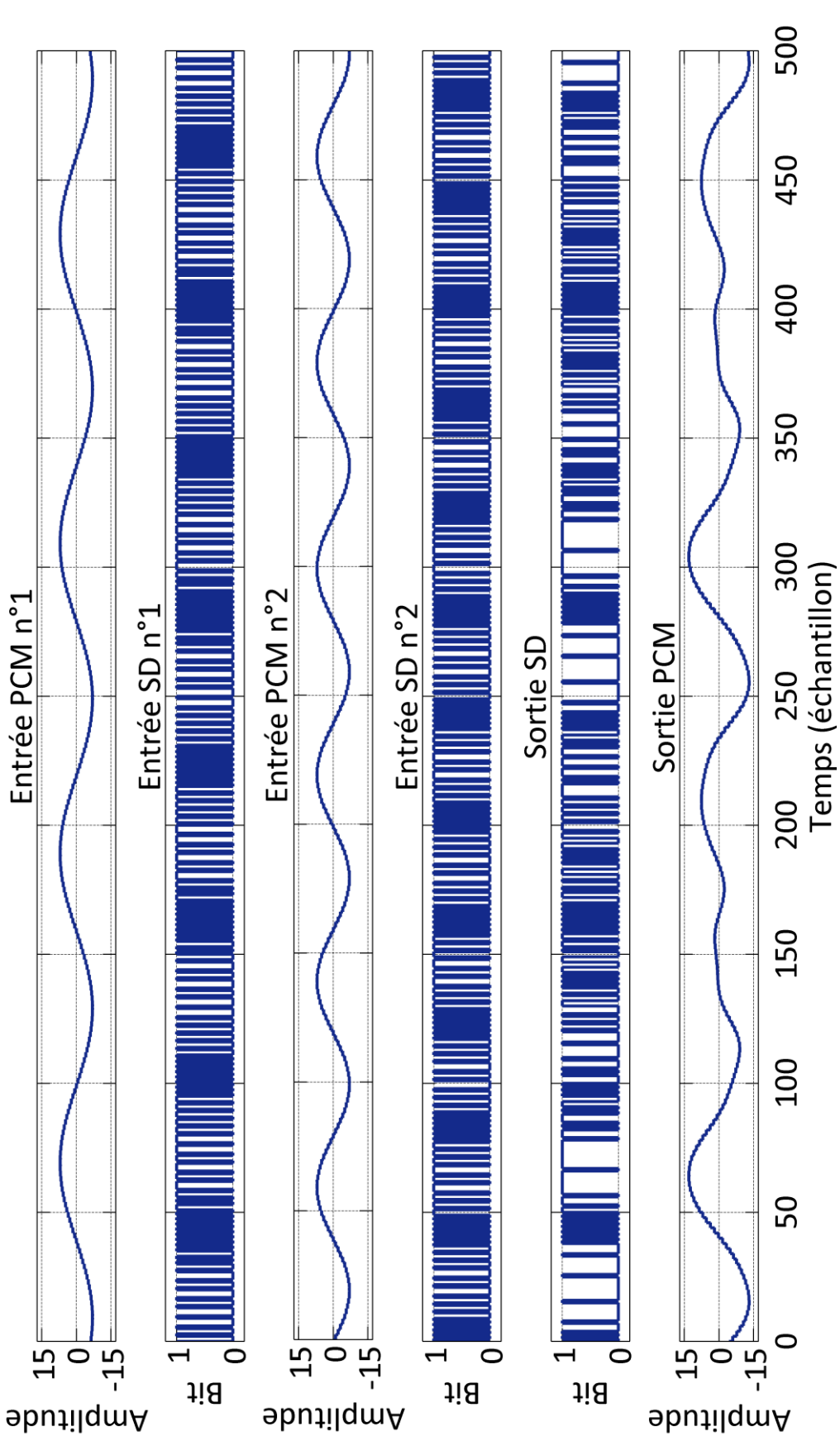

Figure 24 : Simulation de la sommation de deux modulations sigma-delta générée à partir de deux modulations PCM et un facteur de sur-échantillonnage de 2.

6. Dispositif de filtrage passe-bas

a) Modulation PCM

On rappelle que le coefficient a est codé sur m.k bits, tout comme le coefficient 1-a.

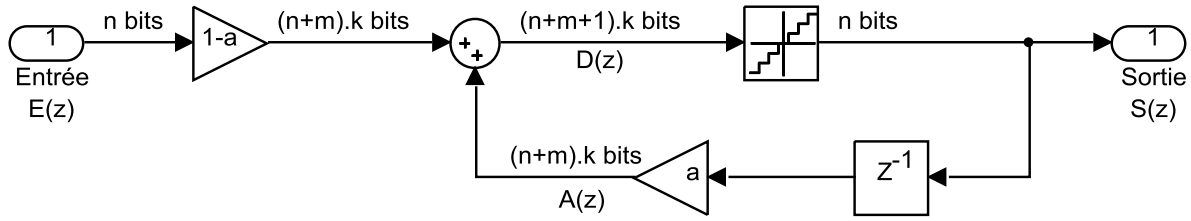

Figure 25 : Schéma de principe d'un filtre passe-bas utilisant la modulation PCM.

L'application d'un filtre passe-bas sur une modulation PCM consiste à sommer une version amplifiée d'un facteur $1-a$ avec le signal de sortie correspondant à l'échantillon précédent et amplifiée par un coefficient a, comme indiqué sur la Figure 25. Pour calculer la sortie en fonction de l'entrée, on peut supprimer le quantificateur, car pour assurer la stabilité du filtre a doit être inférieur à 1 et donc l'erreur ne s'amplifie pas :

$$\begin{cases} S(z) = D(z) \\ D(z) = (1-a)\cdot E(z) + A(z), \\ A(z) = a\cdot z^{-1}\cdot S(z) \end{cases}$$

d'où :

$$S(z) = \frac{1-a}{1-a\cdot z^{-1}}\cdot E(z).$$

En posant $H_{LP}(z) = \frac{S(z)}{E(z)}$, on obtient :

$$H_{LP}^{PCM}(z) = \frac{1-a}{1-a\cdot z^{-1}}.$$

C'est le coefficient a qui permet de choisir la fréquence de coupure. Son expression en fonction de f_0 et F_s est donnée ci-après (pour son calcul on se réfèrera à l'Annexe 3) :

$$a = 2 - \cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right) - \sqrt{\left(\cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right) - 2\right)^2 - 1}.$$

En utilisant cette expression, on peut tracer un réseau de courbes de réponse du filtre passe-bas obtenues en faisant varier les fréquences de coupure et d'échantillonnage.

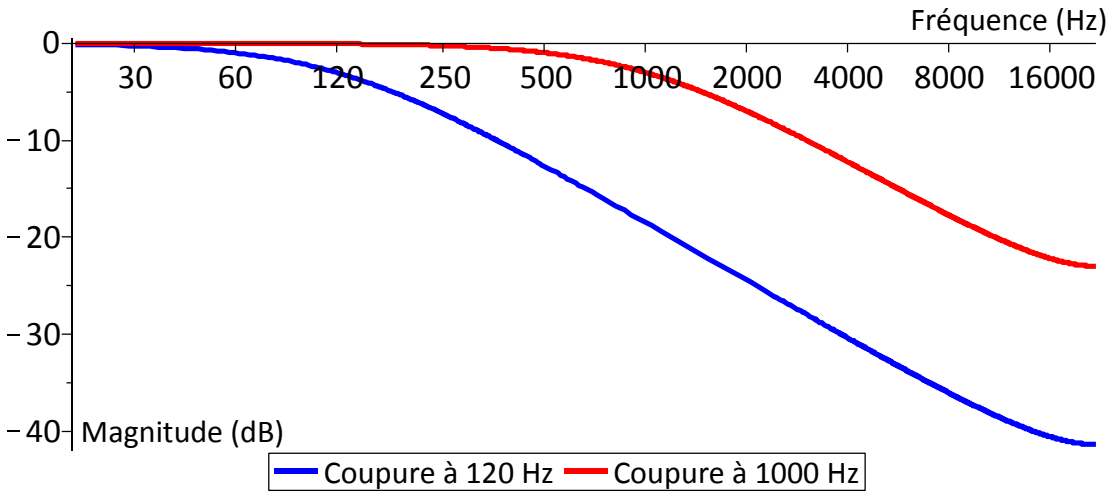

Figure 26 : Représentation de la courbe de réponse du filtre passe-bas pour des fréquences de coupure de 120 Hz ($a = 0{,}9830486376$) et 1000 Hz ($a = 0{,}8674169988$) et une fréquence d'échantillonnage de 44,1 kHz.

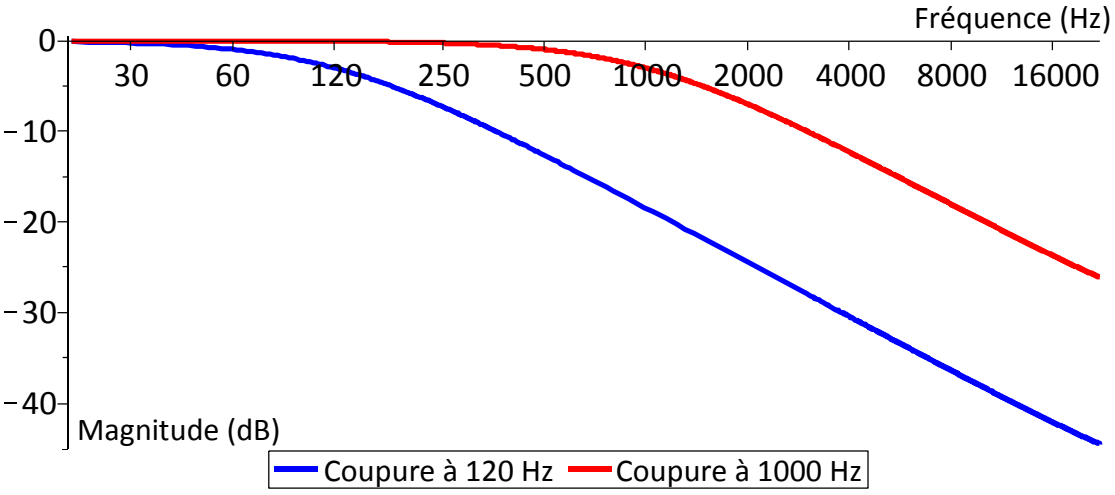

Figure 27 : Représentation de la courbe de réponse du filtre passe-bas pour des fréquences de coupure de 120 Hz ($a = 0{,}9921768621$) et 1000 Hz ($a = 0{,}9366678916$) et une fréquence d'échantillonnage de 96 kHz.

Sur la Figure 26, on remarque que la courbe de réponse ressemble à cette d'un filtre en plateau (shelving filter en Anglais), mais cet artefact est lié au fait que la courbe de réponse du filtre en fréquence est symétrique par rapport à la fréquence de Nyquist. Ceci est confirmé

par la disparition de cet artefact lorsqu'on adopte une fréquence d'échantillonnage de 96 kHz (Figure 27).

b) Modulation sigma-delta [6]

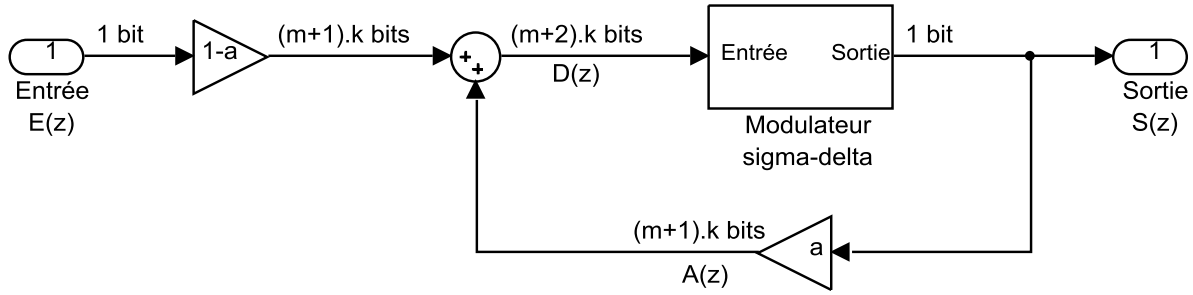

Figure 28 : Schéma de principe d'un filtre passe-bas utilisant la modulation sigma-delta.

Pour la modulation sigma-delta, on construit un filtre passe-bas autour du modulateur sigma-delta, comme sur la Figure 28. Etant donné que le modulateur sigma-delta introduit aussi une rétroaction, il faut développer la boîte le contenant pour calculer la fonction de transfert de l'intégralité du système (cf. Figure 29).

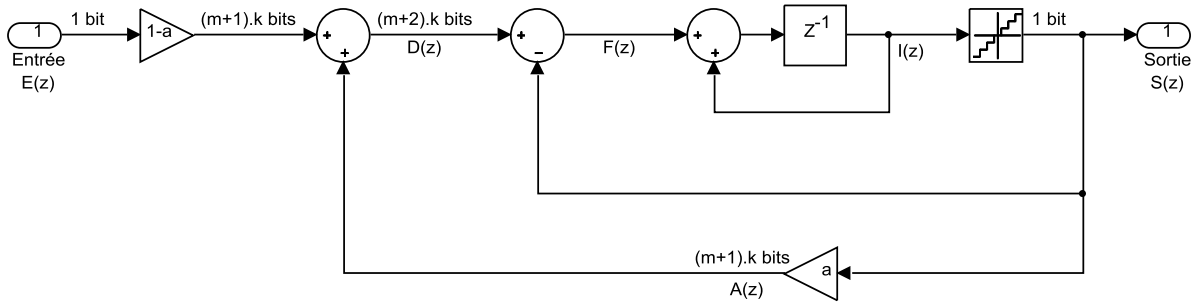

Figure 29 : Schéma de principe d'un filtre passe-bas utilisant un modulateur sigma-delta d'ordre 1.

De la même façon que pour le PCM, on ne tient pas compte du quantificateur pour calculer la fonction de transfert décrivant la sortie en fonction de l'entrée. En utilisant les notations indiquées sur le schéma, on obtient le système d'équations suivant :

$$\begin{cases} D(z) = (1-a)\cdot E(z) + a\cdot S(z) \\ F(z) = D(z) - S(z) \\ I(z) = z^{-1}\cdot (F(z) + I(z)) \\ S(z) = I(z) \end{cases}$$

On peut réécrire le système de la façon suivante :

$$\begin{cases} D(z) = (1-a)\cdot E(z) + a\cdot S(z) \\ D(z) = F(z) + S(z) \\ F(z) = \dfrac{1-z^{-1}}{z^{-1}}\cdot S(z) \end{cases}$$

Après substitution de $F(z)$ dans la deuxième ligne par son expression donnée dans la troisième ligne, on obtient :

$$\begin{cases} D(z) = (1-a)\cdot E(z) + a\cdot S(z) \\ D(z) = \dfrac{1-z^{-1}}{z^{-1}}\cdot S(z) + S(z) \end{cases},$$

soit encore :

$$\begin{cases} D(z) = (1-a)\cdot E(z) + a\cdot S(z) \\ D(z) = \dfrac{S(z)}{z^{-1}} \end{cases}.$$

D'où les expressions suivantes :

$$(1-a)\cdot E(z) + a\cdot S(z) = \frac{S(z)}{z^{-1}},$$
$$(1-a)\cdot z^{-1}\cdot E(z) = S(z)\cdot(1-a\cdot z^{-1}).$$

Après simplification, on obtient l'expression :

$$\frac{S(z)}{E(z)} = \frac{(1-a)\cdot z^{-1}}{1-a\cdot z^{-1}}.$$

On peut poser $H_{LP}^{SD1}(z) = \frac{S(z)}{E(z)}$, ce qui donne au final :

$$H_{LP}^{SD1}(z) = \frac{1-a}{1-a\cdot z^{-1}}\cdot z^{-1}.$$

On obtient bien la fonction de transfert correspondant à un filtre passe-bas. Le z^{-1} en facteur traduit le retard d'un échantillon entre l'entrée et la sortie, dû à la présence du bloc z^{-1} au niveau de l'accumulateur sur la chaîne directe. De même qu'en PCM, le coefficient a permet de changer la fréquence de coupure du filtre et son expression est identique :

$$a = a_1 = 2 - \cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right) - \sqrt{\left(\cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right) - 2\right)^2 - 1}.$$

On peut aussi tracer les courbes de réponses du filtre pour différentes fréquences de coupure et pour une fréquence d'échantillonnage de 2,8224 MHz

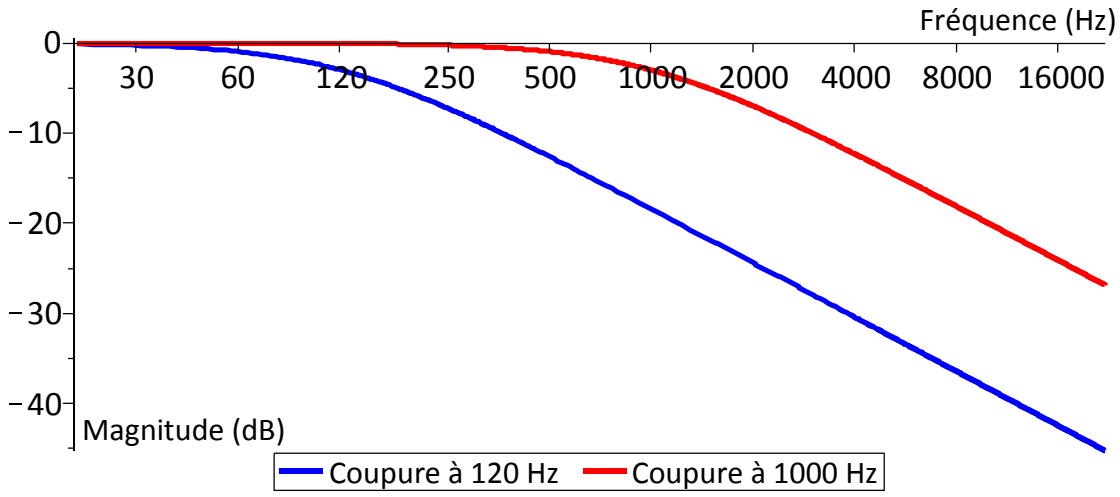

Figure 30 : Représentation de la courbe de réponse du filtre passe-bas pour des fréquences de coupure de 120 Hz ($a = 0{,}9997317078$) et 1000 Hz ($a = 0{,}9977762705$) et une fréquence d'échantillonnage de 2,8224 MHz.

On constate, sur la Figure 30, qu'il n'y a plus les éventuels problèmes du PCM lorsqu'on approche des fréquences les plus aigües que l'on peut entendre. Ceci est lié à la fréquence d'échantillonnage beaucoup plus importante des modulateurs sigma-delta, de la même façon que le passage en 96 kHz permet d'amoindrir le problème en PCM.

7. Conclusion

Dans ce chapitre, on a vu la façon dont sont structurés les modulateurs sigma-delta et PCM ainsi que la façon de réaliser les 4 convertisseurs présentés. Par ailleurs, on a constaté qu'il était envisageable de réaliser les traitements d'une modulation sigma-delta et que comparativement aux traitements d'une modulation PCM, ils sont plus difficiles à concevoir et à étudier.

Après avoir défini la façon de réaliser tous ces systèmes, on expose dans le chapitre suivant, leur réalisation pratique, sauf pour le filtre passe-bas, non encore réalisé par manque de temps.

IV - Implémentation d'une chaîne PCM et sigma-delta

Dans un premier temps, il sera question de présenter l'environnement de travail. Plus tard, on présentera les différents éléments par couches successives. Dans la première couche, on traitera de la partie électronique via la réalisation des convertisseurs. La deuxième couche sera consacrée à présenter une méthode d'implémentation des algorithmes initialement étudiés. Dans la troisième couche, on abordera la communication entre les parties logicielles et matérielles. La quatrième couche concernera l'interface entre l'interface utilisateur et les parties évoquées dans la troisième couche.

1. Présentation de la ZedBoard [7]

La ZedBoard est une plateforme de développement, c'est-à-dire un système qui permet de développer un système électronique et informatique rapidement et simplement, tout en permettant au développeur d'accéder à tous les éléments sur ces deux parties et de les modifier (schéma électroniques, code informatique, …). En effet, lorsque l'on utilise un ordinateur ou une table de mixage, on cherche à rendre conviviale l'utilisation du matériel et du logiciel en évitant à l'utilisateur de se pencher sur les parties techniques impliquées dans la réalisation de ce système. Dans le cas d'une table de mixage, l'utilisateur n'a pas besoin de savoir si c'est plutôt tel ou tel composant électronique qui est utilisé pour effectuer le traitement demandé, c'est seulement le rapport signal sur bruit et/ou le taux de distorsion, par exemple, qui peuvent l'intéresser. Il est de la responsabilité du concepteur de choisir les composants électroniques appropriés pour réaliser la fonction désirée.

La ZedBoard est centrée autour d'un composant fabriqué par la société Xilinx : le Zynq XC7Z020-1CLG484. Ce composant appartient à la famille Zynq-7000 qui sont des AP SoC (All Programmable System on Chip ou système sur puce entièrement programmable). Ce système est bâti autour de deux parties distinctes programmables mais pouvant communiquer entre-elles au moyen de canaux privilégiés. La première partie est un processeur double cœur Cortex-A9 MPCore conçu par la société ARM identique à celui qui équipe certaines tablettes ou téléphones intelligents (smartphones). Ce processeur peut être programmé de façon

logicielle, c'est-à-dire par des lignes de codes s'exécutant chronologiquement et exploitant les instructions disponibles sur ce processeur. La deuxième partie s'articule autour d'un FPGA (Field Programmable Gate Array ou réseau de portes logiques programmable in-situ) Artix-7 conçu par la société Xilinx. Ce FPGA peut être programmé de façon matérielle, c'est-à-dire que l'on va configurer les portes logiques, pour définir leurs fonctions, et réaliser les connexions adéquates entre-elles. Ainsi, on peut à la fois programmer un logiciel sur le processeur et un matériel sur le FPGA, et, concrètement, on programmera les dispositifs étudiés précédemment sur le FPGA et on les contrôlera à partir d'un logiciel sur le processeur.

Afin de réaliser notre système, la ZedBoard met à la disposition du développeur 512 Mo de mémoire vive dynamique (DDR3) qui sera affectée aux logiciels, un port pour une carte SD pour stocker des informations, un port HDMI pour la visualisation, un port USB pour brancher des périphériques informatiques (souris, clavier, clé USB, …), un port réseau pour communiquer avec d'autres systèmes et enfin des ports d'extensions Pmod qui seront utilisés par les ADC et les DAC. La ZedBoard propose aussi d'autres fonctionnalités qui n'ont pas été utilisées dans le cadre de ce mémoire.

2. Liaisons électroniques entre composants numériques

a) Bus I2C [8]

Le bus I2C (Inter Integrated Circuit ou entre les circuits intégrés), appelé parfois TWI (Two Wire Interface ou interface à deux fils), est un bus série de communication qui permet de relier facilement des systèmes électroniques très divers. Prévu initialement pour des équipements domestiques, il est aussi utilisé dans des systèmes électroniques complexes. Ce bus permet à différents composants numériques d'échanger des données à la vitesse standard allant jusqu'à 100 kbits/s en mode standard, 1 Mb/s en mode rapide, 3,2 Mb/s en mode très rapide et 5 Mb/s en mode ultra rapide. La spécificité de cette liaison est de relier tous les composants sur deux fils en utilisant des adresses et une communication de type maître-esclave. Ce type de communication permet d'introduire la notion de hiérarchisation entre les composants, le maître initie la communication et la gère, tandis que les esclaves se contentent de répondre aux ordres du maître et ne peuvent pas initier de communication,

cependant chaque composant peut devenir maître à tout moment s'il n'y en a pas et redevenir esclave pour libérer la place.

Sur la Figure 31 on constate que la liaison est constituée de deux lignes au niveau électronique : la ligne SDA (Serial DAta) transporte les données et la SCL (Serial CLock) qui transporte l'horloge permettant de valider les données présentes sur SDA. Dès qu'un composant veut communiquer avec un autre il doit réaliser une condition START, qui sera captée par l'ensemble des composants esclaves de la liaison, puis émettre l'adresse du composant auquel est destinée l'information. L'adresse comporte deux parties : l'adresse à proprement parler codée sur 7 bits ainsi qu'un bit qui permet de spécifier l'opération (lecture ou écriture) que l'esclave devra exécuter. Une fois ces 8 premiers bits transmis l'esclave doit accuser réception de l'information, et si c'est le cas, on va pouvoir transmettre toutes les données par octets successifs dont l'esclave devra à chaque fois accuser bonne réception. A la fin de la transmission, le maître exécutera une condition STOP pour signaler la fin de la transmission et la libération de la liaison pour les autres maîtres. La Figure 32 illustre le principe de ce fonctionnement. Il est à noter que dans le cas de l'opération de lecture, les données sont dirigées de l'esclave vers le maître, et c'est donc au maître d'accuser bonne réception de l'information : s'il ne le fait pas cela met fin au transfert.

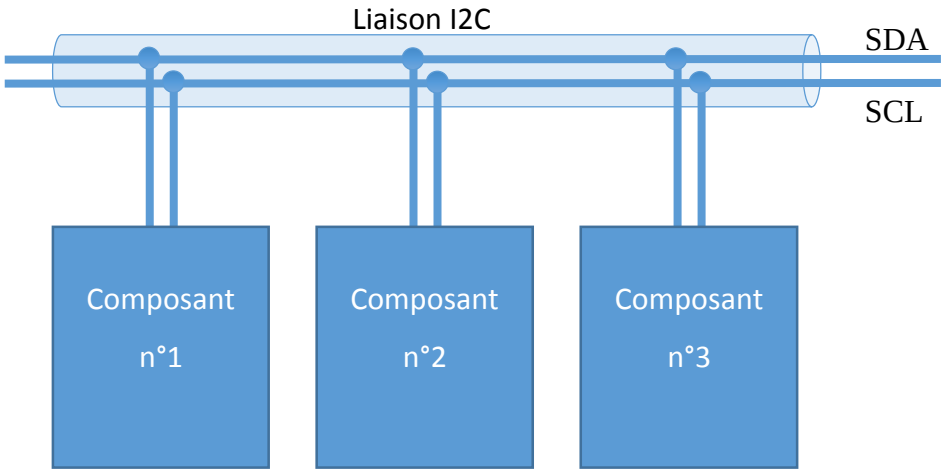

Figure 31 : Agencement des composants sur une liaison I2C.

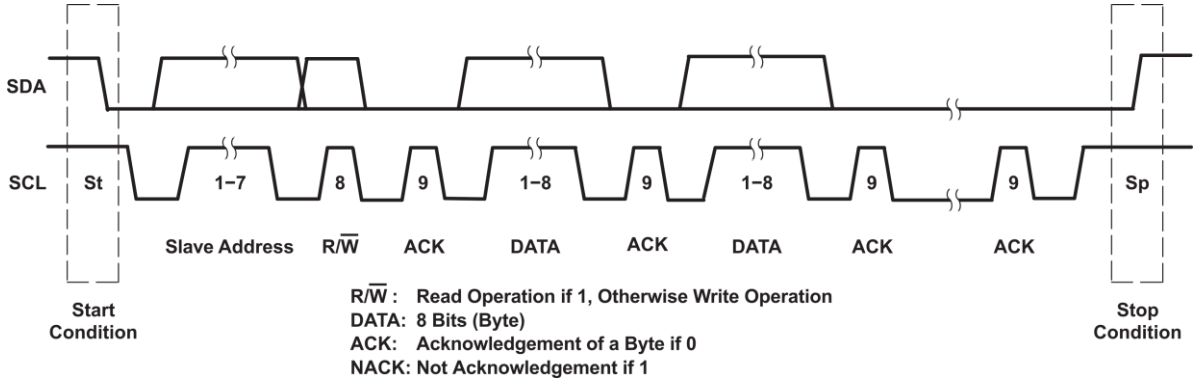

Figure 32 : Structure d'une transmission I2C. D'après [9], Figure 28, page 20.

b) Bus I2S [10]

Le bus I2S (Inter Integrated circuit Sound ou son entre circuit intégré) est un bus série de communication entre divers systèmes électroniques dédié au transport de signaux audionumériques. Le son est une donnée de flux, c'est-à-dire que, périodiquement, on va devoir transmettre une quantité d'information constante. Par exemple, dans le cadre du CD-Audio on transmet 44100 fois par seconde deux fois 16 bits. Dès lors, il va devenir compliqué de réaliser des liaisons où plusieurs composants doivent pouvoir communiquer entre eux lorsqu'ils le désirent, avec en plus des destinataires d'information qui changent en permanence, c'est pourquoi la liaison I2S s'effectue dans un seul sens et entre deux composants uniquement. La majorité des produits audio fabriqués (en terme de quantité industrielle et non pas en terme produit différent) possédant deux canaux, pour la reproduction stéréophonique, par exemple, la liaison I2S est une liaison bi-canal. Il n'y a pas de débit spécifié afin de pouvoir s'adapter à toutes les résolutions et à toutes les fréquences d'échantillonnage, et il revient au concepteur du système de vérifier que les composants peuvent bien communiquer entre eux.

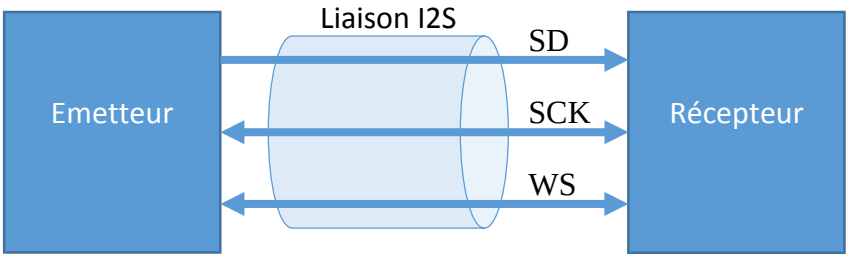

Figure 33 : Agencement des composants sur une liaison I2S.

Cette liaison s'articule autour de 3 lignes électriques (cf. Figure 33), la ligne SD (Serial Data) transmettant les données des deux canaux, la ligne SCK (Serial ClocK) donnant l'horloge qui permet de valider les données et la ligne WS (Word Select) qui permet de dire si c'est le premier (WS = 0) ou le deuxième canal (WS = 1) qui est en cours d'émission. Ces noms ne sont pas normalisés et changent en fonction des constructeurs et des composants. Le sens des flèches indique le sens dans lequel les informations circulent. Les données vont forcément de l'émetteur au récepteur, tandis que les deux autres signaux peuvent être générés de façon liées, soit par l'émetteur soit par le récepteur. C'est le cas, par exemple, d'un DSP qui synchronise l'ensemble des convertisseurs, et bien qu'il reçoive les informations audio des convertisseurs analogique/numérique, c'est lui qui génère le signal d'horloge et de sélection du canal transmis.

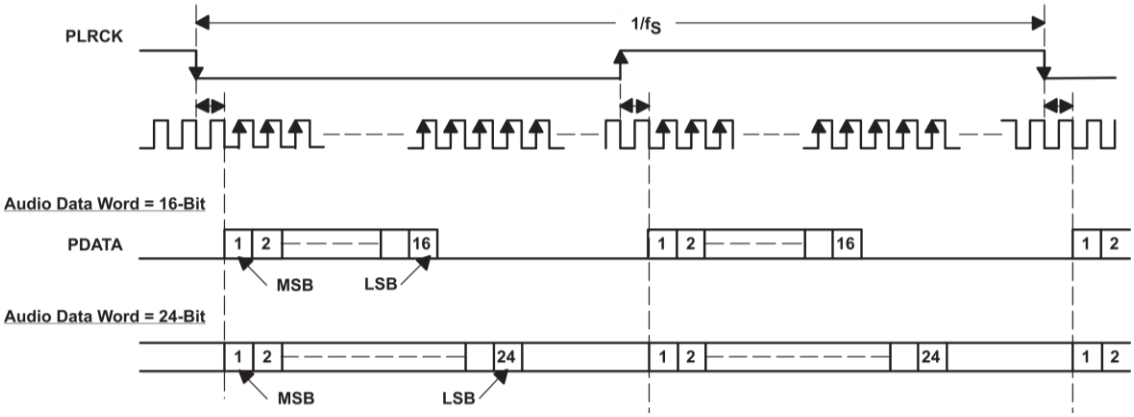

Figure 34 : Structure d'une transmission I2S, PLRCK correspond à WS, PBCK correspond à SCK, PDATA correspond à SD. D'après [9], Figure 27, page 18.

Sur la Figure 34, on constate que les canaux de gauche et de droit son transmis alternativement et que l'on met la durée d'un échantillon à transmettre les deux canaux. Il y a par ailleurs 1 cycle de retard (au sens de l'horloge SCK) entre le changement de canal sur WS

et la transmission du premier bit correspondant à ce canal sur SD. On remarque aussi que le nombre de bits par échantillon n'est pas normalisé, si ce n'est qu'il doit être égal pour les deux canaux. Par ailleurs il existe la liaison TDM (Time Division Multiplexing ou multiplexage par division du temps ou plus simplement multiplexage temporel) qui assouplit les règles de l'I2S et qui peut transmettre un nombre indéfini de canaux, on indique généralement le nombre de canaux après le sigle TDM (par exemple TDM 64 pour une liaison à 64 canaux).

c) Bus DSD

Le bus DSD n'est pas normalisé comme les deux précédents, mais c'est celui qu'on utilisera pour transmettre les données au format sigma-delta. Cette liaison sert aussi à transmettre des données de flux, sauf que comme on n'a qu'un seul bit à transmettre, on a, *a priori*, plus besoin de l'horloge de canal qui sert à indiquer le premier bit de chaque canal. On va donc supprimer cette ligne et ajouter à la place une ligne pour transmettre un deuxième canal.

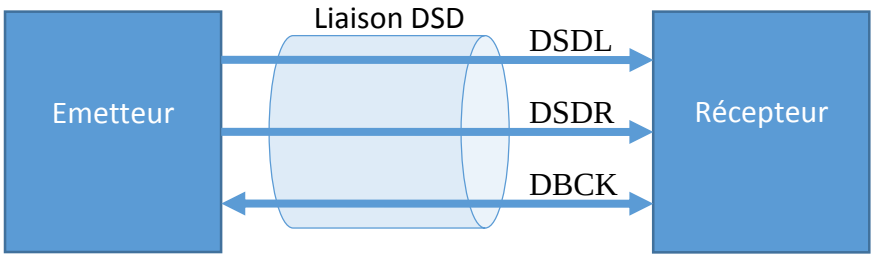

Figure 35 : Agencement des composants sur une liaison DSD.

Sur la Figure 35, la liaison DSD se compose de 3 lignes : la ligne DSDL (DSD Left) servant à transmettre les données du canal gauche, la ligne DSDR (DSD Right) servant à transmettre les données du canal droit et la ligne DBCK (DSD Bit ClocK) qui transmet l'horloge validant les données. La Figure 36 montre un exemple d'évolution au cours du temps des informations transitant dans le bus.

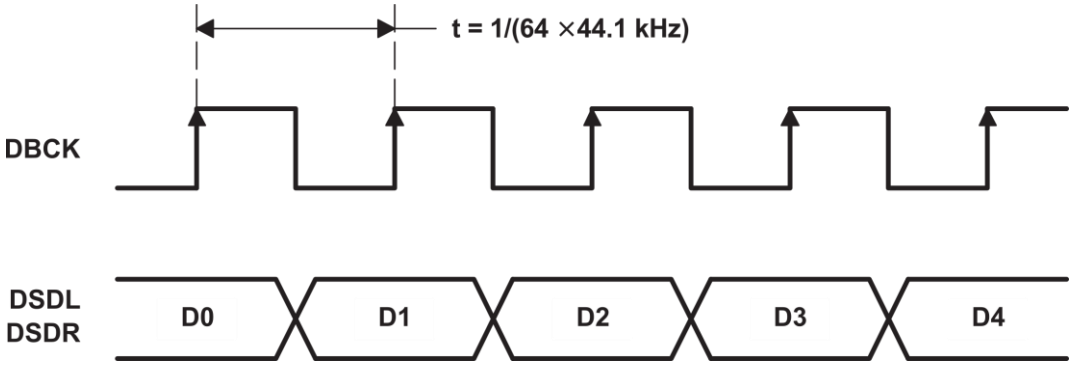

Figure 36 : Structure une liaison DSD où D0, D1, D2, D3 et D4 représentent les échantillons successifs. D'après [9], Figure 38, page 39.

d) Bus AXI-4 [11]

Le bus AXI-4 (Advanced eXtensible Interface version 4 ou interface extensible avancée version 4) fait partie des bus spécifiés dans l'AMBA (Advanced Microcontroller Bus Architecture) qui est standardisé par la société ARM. Contrairement aux autres liaisons, le bus AXI ne sert pas à relier des composants électroniques entre eux, mais des composants à l'intérieur des circuits intégrés entre eux, on parle de SoC (System on Chip ou système sur puce). Dans notre cas, on s'en servira pour relier les différents composants matériels programmés dans le FPGA au processeur ARM qui héberge les programmes de commande (tous deux situés dans le même circuit intégré). On ne s'intéressera pas au fonctionnement de ce bus car il en existe plusieurs variantes, avec des ports optionnels en fonction des besoins, et on se contentera d'utiliser les outils déjà fournis par les fabricants des composants.

3. Convertisseurs sigma-delta à partir de composants électroniques de base

a) Convertisseur analogique/numérique

Le convertisseur analogique/numérique réalisé reprend les éléments du modulateur sigma-delta décris précédemment et la Figure 37 montre le schéma du circuit réalisé. Le premier étage de ce montage réalise l'opération de sommation du signal d’entrée et du signal de sortie. Le deuxième étage effectue l'opération d’intégration. Dans le troisième étage on trouve le quantificateur réalisé par un simple comparateur. Le quatrième étage réalise la fonction d’échantillonneur et est constitué d'une simple bascule D. La bascule D est un composant électronique qui recopie l'état logique présent à l'entrée de donnée D (Data) sur la sortie principale, baptisée Q, lorsqu’il y a un front montant sur l’entrée d'horloge Cp (Clock pulse). La sortie complémentée, baptisée $\overline{Q}$, correspond au complément de Q : si Q vaut 1 alors $\overline{Q}$ vaut 0 et si Q vaut 0 alors $\overline{Q}$ vaut 1. La rétroaction se fait par $\overline{Q}$ car dans le modulateur on réalise une soustraction entre l’entrée et la sortie, même si le premier étage du montage réalise une addition.

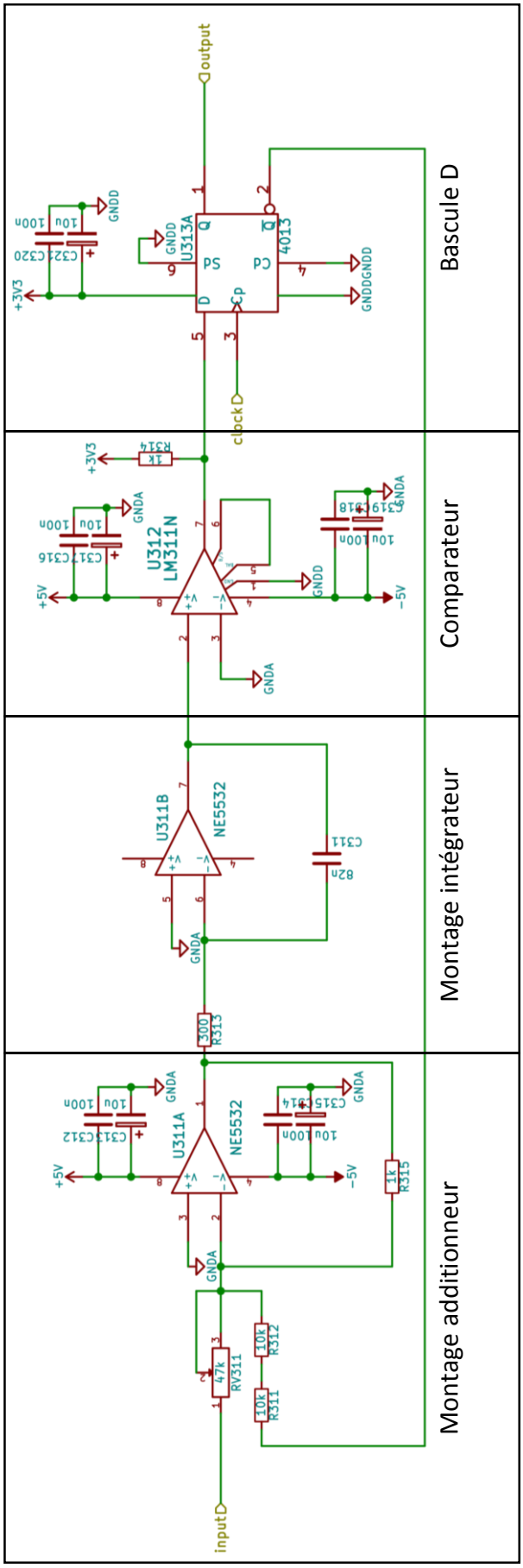

Figure 37 : Schéma électronique du modulateur sigma-delta.

b) Convertisseur numérique/analogique

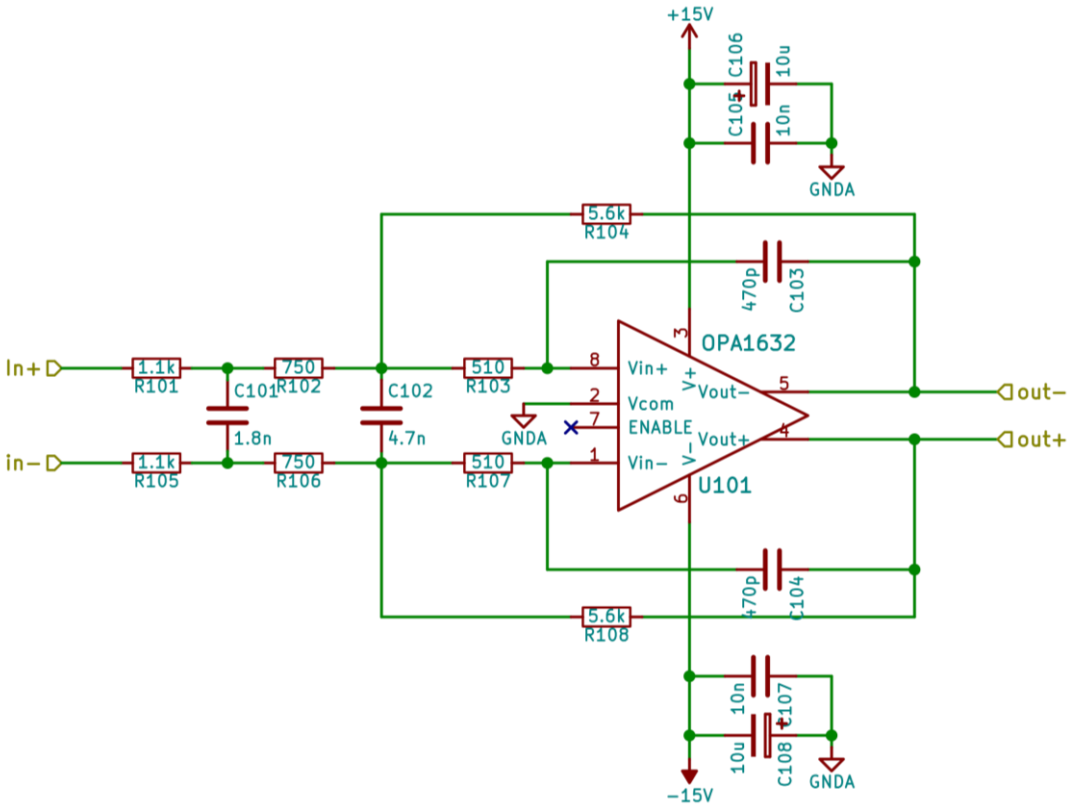

Figure 38 : Filtre passe-bas d'ordre 3 utilisé.

Pour réaliser la conversion numérique/analogique, on peut réaliser un filtrage passe-bas d'ordre 6 en utilisant 2 filtres passe-bas d'ordre 3 comme celui présenté en Figure 38. Le calcul des éléments des filtres a été effectué au moyen d'un outil de synthèse de filtre analogique à partir d'un gabarit établi à partir d'un cahier des charges [12]. La fréquence de coupure a été choisie de façon à laisser une bande passante de 50 kHz, ce qui est analogue au SACD qui utilise aussi la modulation sigma-delta. Pour le premier filtre on connectera l'entrée In- à la masse et l'entrée In+ directement au composant Zynq. On notera l'absence du signal d'horloge, ce qui se justifie car d'une part il n'y a qu'un seul bit à transmettre donc il n'y a pas besoin d'indiquer le premier bit de la transmission, et d'autre part le signal d'horloge sert aussi à indiquer quand le bit transmis est valide et son absence implique que le signal transmis a la bonne forme (ce qui est le cas dans l'implémentation que j'ai réalisée).

4. Convertisseurs sigma-delta à partir de puces spécifiques

a) Convertisseur analogique/numérique [13]

Pour réaliser le convertisseur analogique/numérique, on peut utiliser le composant de référence PCM4202 conçu et fabriqué par la société Texas Intruments. Ce composant intègre deux convertisseurs analogique/numérique (pour réaliser de la stéréophonie, par exemple) avec des entrées différentielles. La Figure 40 montre le câblage des entrées et des sorties de ce convertisseur. Le signal audio analogique entre dans le connecteur P2 (broches 3 et 4 pour le canal gauche et broches 7 et 8 pour le canal droit) puis entre dans l'étage d'entrée (voir Figure 39) servant à adapter en tension le signal audio. En effet celui-ci entre avec un niveau de crête de +22 dBu, soit une plage de tension allant de -9,76 V à 9,76 V, et le convertisseur accepte une plage de tension de 0 V à 6 V. Par ailleurs, on notera que le signal en entrée a une tension moyenne de 0 V, alors que le convertisseur en a une de 3 V, d'où l'existence d'une entrée supplémentaire sur l'étage d'entrée permettant au convertisseur de lui indiquer cette tension de référence. Quant au connecteur P1, il réalise l'interface entre le convertisseur et le composant Zynq. Ce connecteur étant relié à une des interfaces Pmod (interface permettant d'ajouter du matériel externe à la ZedBoard), il doit se conformer à ce standard, c'est-à-dire la possibilité de transférer des données quelconques sur les broches 1 à 4 et 7 à 10, les broches 5 et 11 correspondent à la masse et les broches 6 et 12 permettent à la ZedBoard de fournir une alimentation de 3,3 V à la carte, fonction qui n'est pas utilisée car on utilise une alimentation dédiée. Les broches 1 à 3 seront affectées au bus DSD (ou I2S si on choisit ce mode de transmission), les broches 4 et 7 servent à configurer le format de la transmission (DSD dans notre cas) et les broches 8 à 10 permettent de modifier la fréquence d'échantillonnage (2,8224 MHz pour l'application qui est réalisée).

L'étage d'entrée est constitué d'un amplificateur opérationnel entièrement différentiel U2 réalisant la fonction amplification. Il n'est pas nécessaire dans ce cas de préciser s'il est inverseur ou non, car il suffit d'inverser la connexion des fils en sortie de l'amplificateur opérationnel pour inverser la polarité du signal. L'entrée Vcom de cet amplificateur sert à recevoir la tension générée par le convertisseur afin d'aligner correctement la tension de sortie par rapport à la plage de fonctionnement du convertisseur. La documentation, fournie par Texas Instruments, précise qu'on ne doit pas directement brancher cette ligne de référence à l'amplificateur mais qu'on doit passer par un autre

amplificateur opérationnel pour suivre la tension et isoler le convertisseur de l'amplificateur opérationnel du signal. C'est la fonction que réalise U3A.

Il reste l'entrée CLK à droite du schéma qui permet de recevoir l'horloge globale du système (qui n'a rien à voir avec les horloges des données), et par manque de place sur l'interface Pmod du convertisseur analogique/numérique la ligne est tirée depuis le module Pmod du convertisseur numérique/analogique.

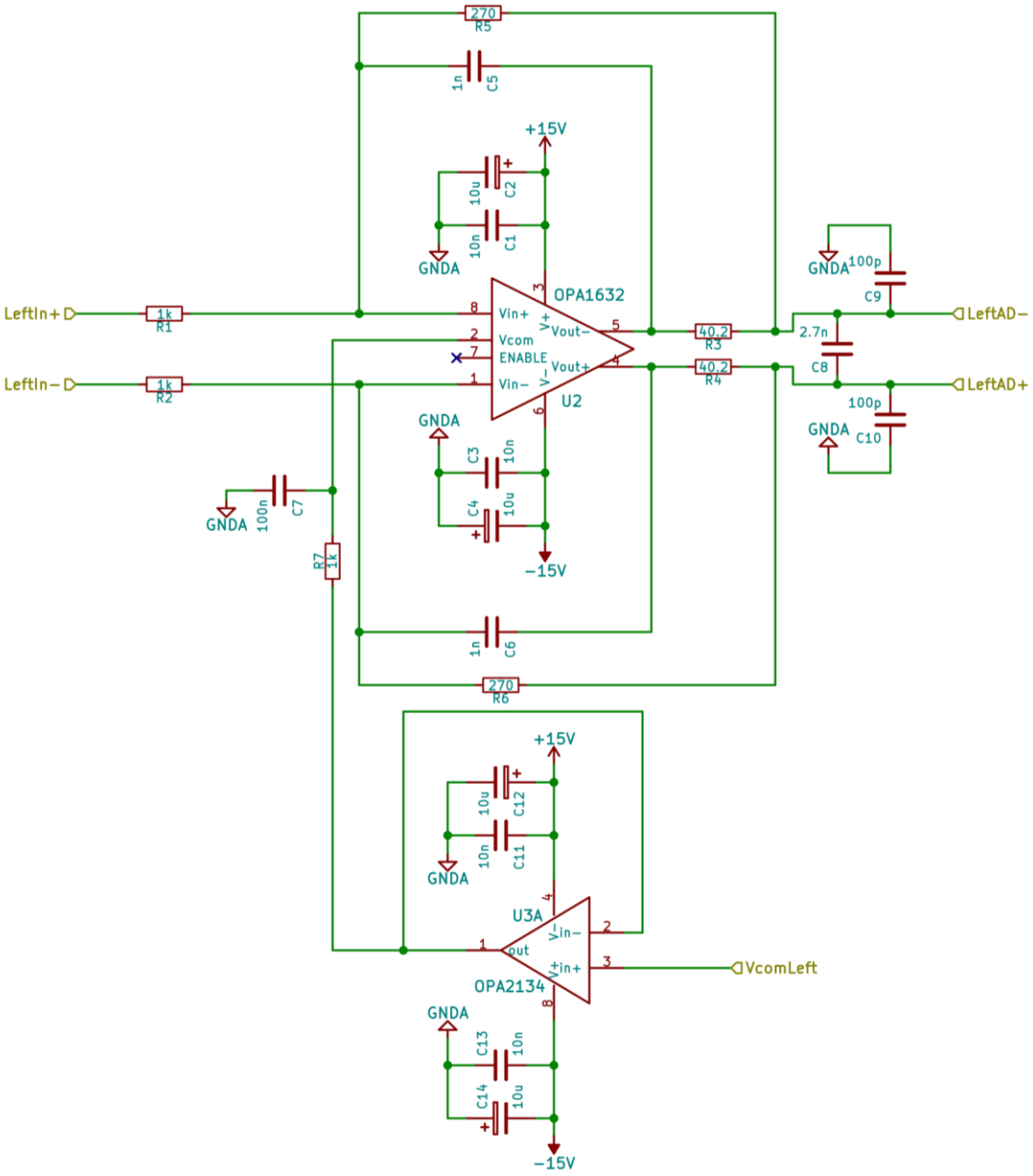

Figure 39 : Etage d'entrée du convertisseur analogique/numérique.

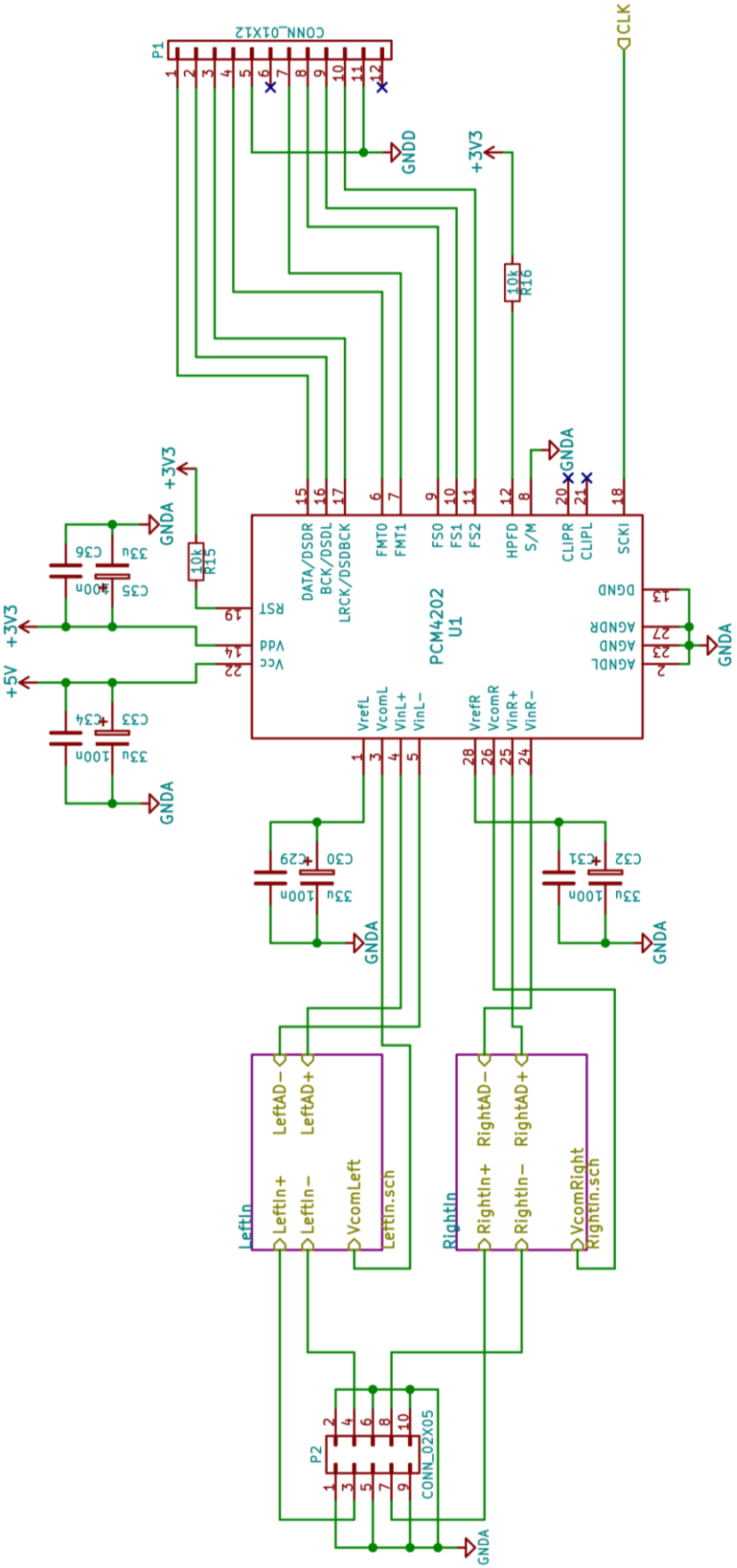

Figure 40 : Convertisseur analogique/numérique basé sur le PCM4202.

b) Convertisseur numérique/analogique [9]

La Figure 42 montre la réalisation d'un convertisseur numérique/analogique basé sur le composant DSD1793 de Texas Instruments. Le port P101 correspond à l'interface Pmod qui permet de relier le convertisseur à la ZedBoard. Les broches 1 et 2 de ce port correspondent au bus I2C permettant de paramétrer le convertisseur numérique/analogique. La broche 3 sert à transmettre l'horloge globale du système et on constate qu'elle est reliée à la sortie CLK pour être reliée au convertisseur analogique/numérique. Les broches 5, 7 et 8 sont affectées au bus DSD.

La documentation du convertisseur spécifie qu'il faut relier les sorties du convertisseur à des filtres qui sont précisés dans cette même documentation et la Figure 41 reprend ce schéma.

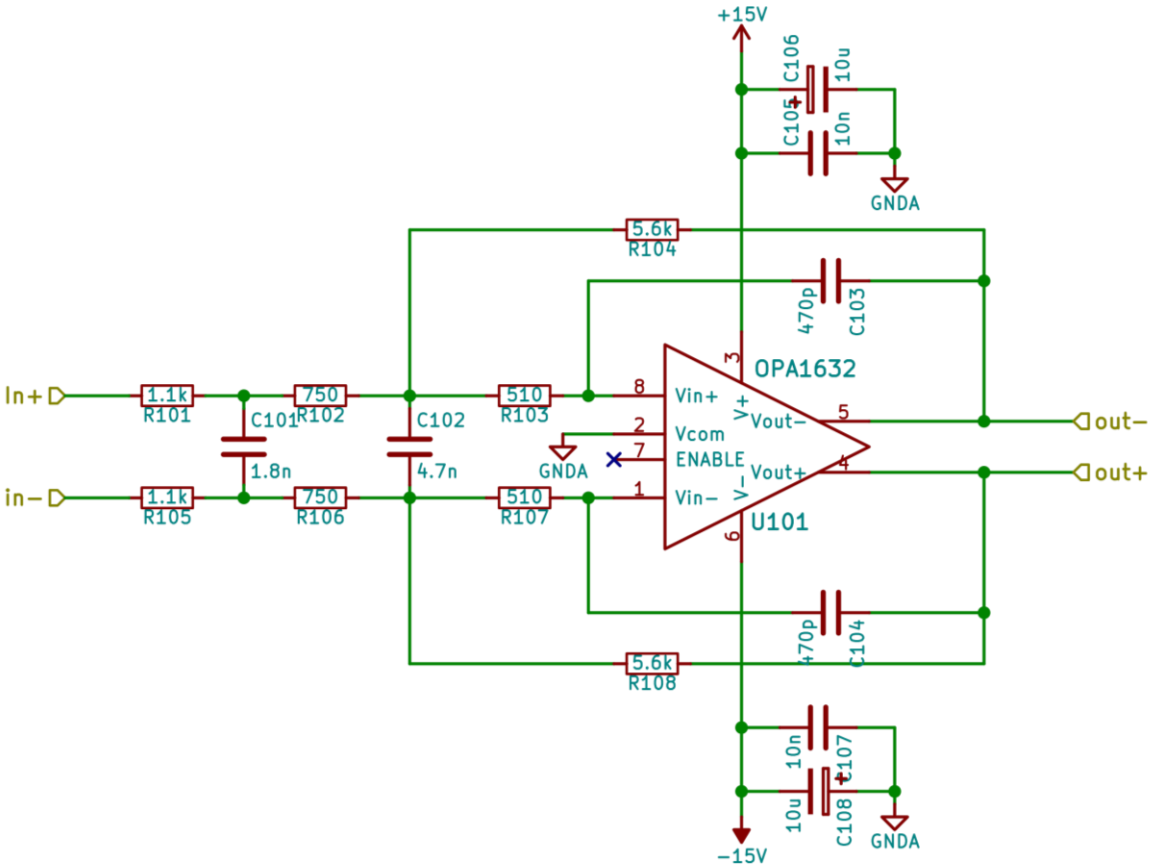

Figure 41 : Etage de sortie du convertisseur numérique/analogique.

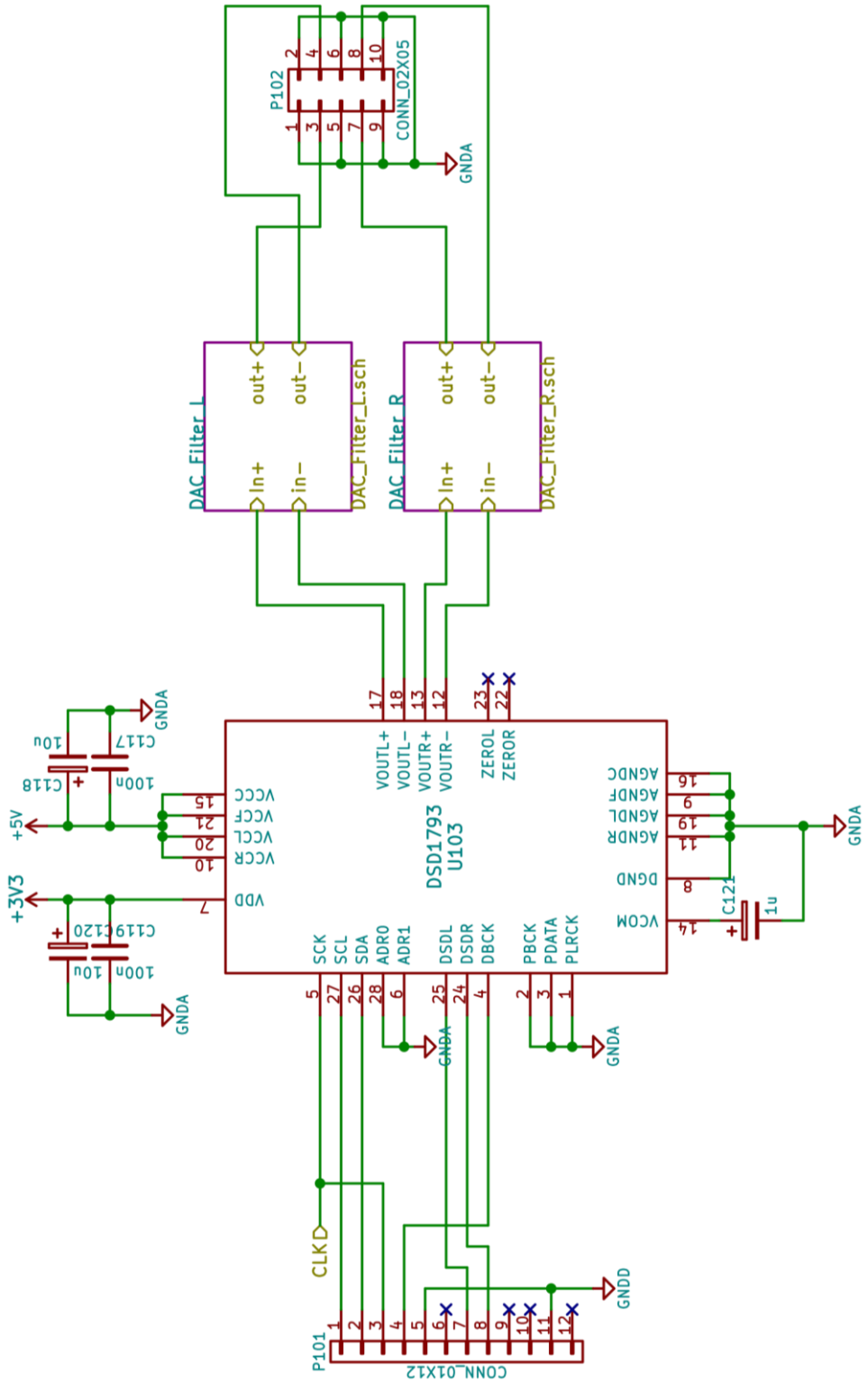

Figure 42 : Convertisseur numérique/analogique basé sur le DSD1793.

5. Conversion entre sigma-delta et PCM

a) Convertisseur sigma-delta vers PCM

Pour réaliser le convertisseur sigma-delta vers PCM, on doit réaliser un filtre qui supprime toutes les fréquences au-delà de la fréquence de Nyquist de la modulation PCM à obtenir. Dans le cas présent, on souhaite convertir une modulation sigma-delta ayant une fréquence d'échantillonnage de 2,8224 MHz en une modulation PCM ayant une fréquence d'échantillonnage de 44100 Hz (c'est-à-dire, 64 fois plus faible). On doit donc filtrer toutes les fréquences au-delà de 22050 Hz qui sont présentes dans la modulation sigma-delta. Réaliser un filtre passe-bas efficace à une fréquence d'échantillonnage de 2,8224 MHz est très coûteux en calcul, c'est-à-dire en place au sein du FPGA, on réalise donc deux filtrages successifs en passant par une fréquence d'échantillonnage intermédiaire 16 fois inférieure à la modulation sigma-delta (c'est-à-dire, 4 fois supérieure à celle de la modulation PCM à obtenir), c'est-à-dire 176,4 kHz. Sur la Figure 43, qui montre le schéma-bloc réalisé, le multiplicateur de fréquence d'échantillonnage génère cette fréquence d'échantillonnage intermédiaire à partir de l'horloge de la modulation PCM (on se réfèrera à l'Annexe 4 pour un détail des blocs utilisés).

L'étage d'entrée est un multiplexeur qui sert à convertir le bit de la modulation sigma-delta qui vaut 0 ou 1 en les nombres respectifs -1 et 1 qui vont pouvoir être utilisés par les opérateurs suivants.

Le premier filtre est basé sur une structure de filtre CIC (Cascaded Integrator-Comb filter ou filtre en peigne par intégrations successives) et a l'avantage de n'utiliser que des additionneurs et des soustracteurs qui sont simples à réaliser. Ce filtre (cf. Figure 44) consiste à réaliser des accumulations successives (5 sur cette implémentation) en réalisant un enroulement quand on dépasse leurs valeurs maximales ; puis on réalise un sous-échantillonnage (d'un facteur 16, dans notre cas) et enfin, on dérive pour "compenser" les effets des intégrateurs [14].

En deuxième position, on trouve un filtre FIR (Finite Impulse Response ou réponse impulsionnelle finie) qui permet de couper de façon très efficace les hautes fréquences avant le sous-échantillonnage pour atteindre la fréquence de 44,1 kHz. Il devient possible d'utiliser ce filtre à cet endroit car la fréquence d'échantillonnage du signal d'entrée a été abaissée d'un

facteur 16 par le filtre CIC et par conséquent, on a plus de temps pour effectuer les calculs. La réponse impulsionnelle du filtre est montrée en Figure 45 et sa réponse en fréquence est donnée en Figure 46. On notera que ce filtre rehausse les hautes fréquences dans la bande passante afin de compenser les effets du filtre CIC.

L'étage de sortie effectue un sous-échantillonnage d'un facteur 4 permettant d'atteindre la fréquence d'échantillonnage cible de 44,1 kHz, puis réalise une conversion au format 32 bits flottant, qui est utilisé dans le reste de l'implémentation pour les échantillons PCM, et enfin effectue une multiplication pour réaliser une correction du niveau de sortie de l'ensemble des filtres. Il est à noter que le sous-échantillonneur est constitué d'un additionneur qui additionne 4 échantillons en sortie du filtre FIR pour former un échantillon de la modulation PCM. Pour ce qui est de la représentation en 32 bits flottants, les valeurs extrêmes pour les échantillons sont -1 et 1 ; le format permet de coder des nombres plus grands mais au-delà de ces valeurs, on considèrera qu'il y a saturation.

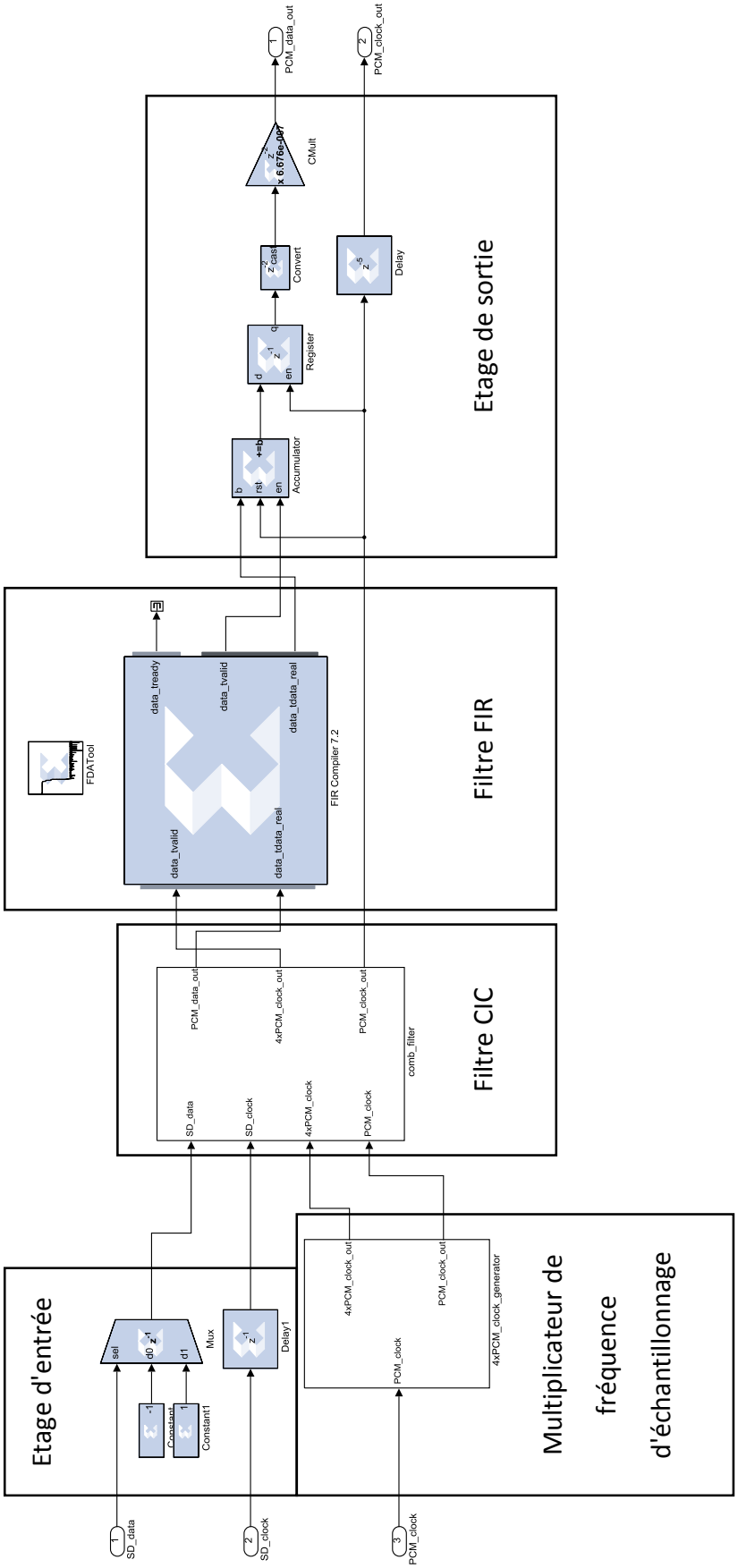

Figure 43 : Enchaînement des filtres dans le convertisseur sigma-delta vers PCM.

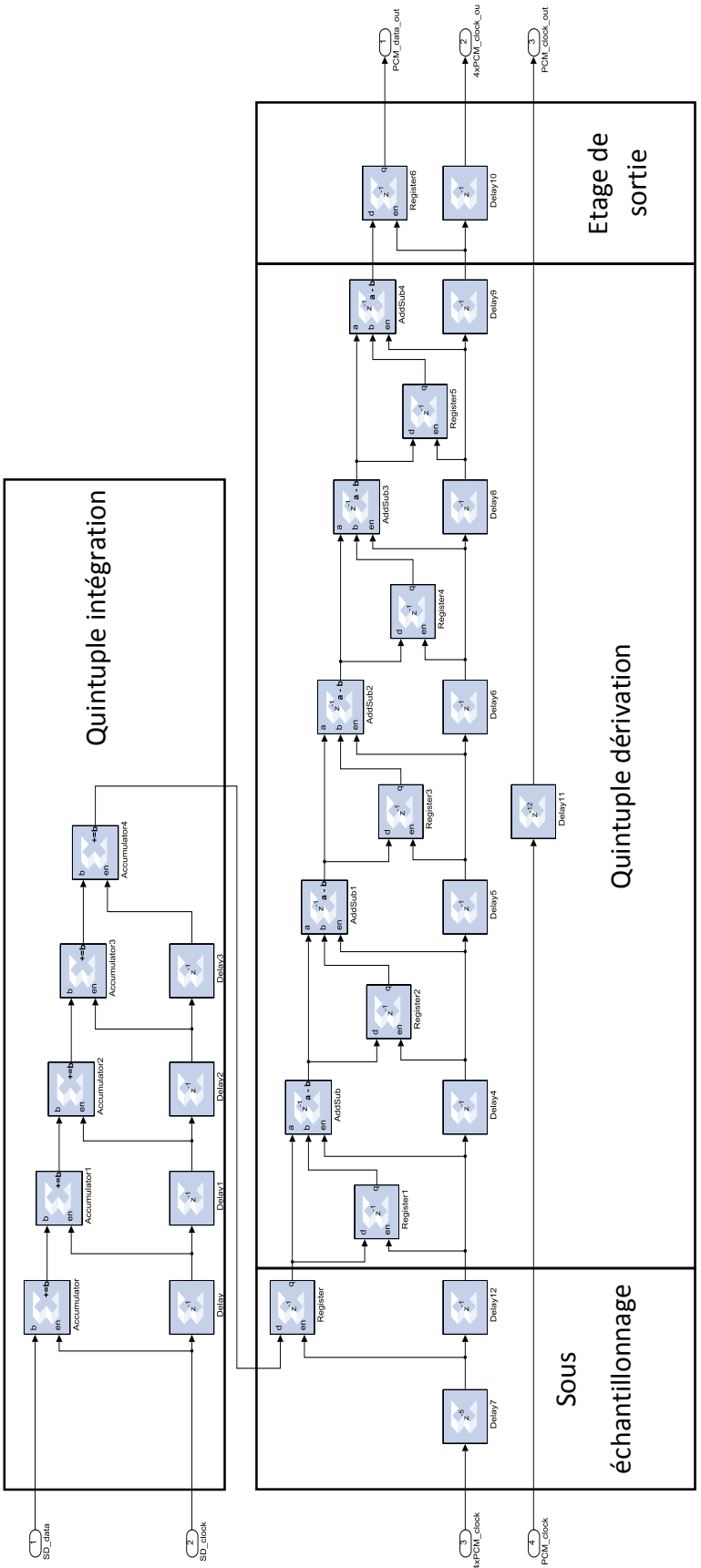

Figure 44 : Réalisation du filtre CIC.

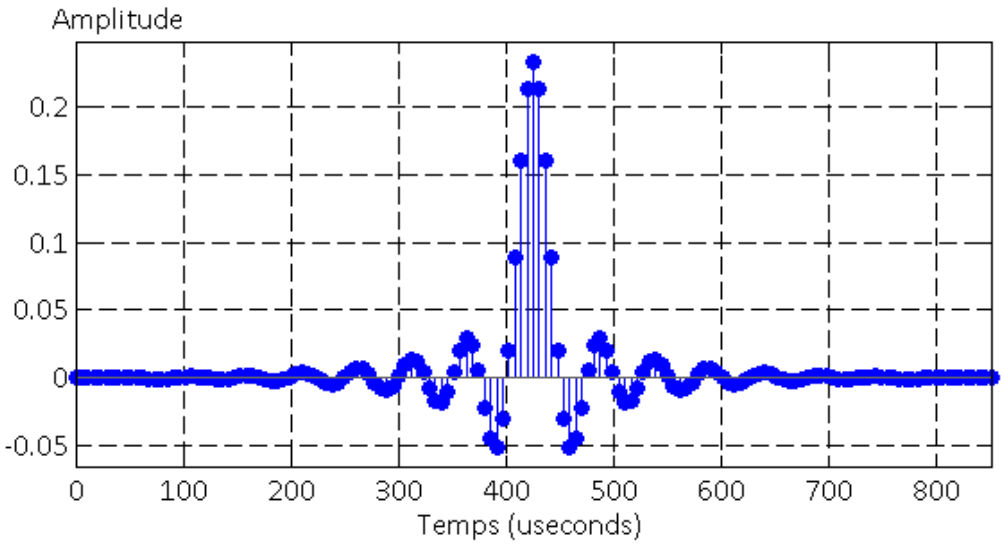

Figure 45 : Réponse impulsionnelle du filtre FIR.

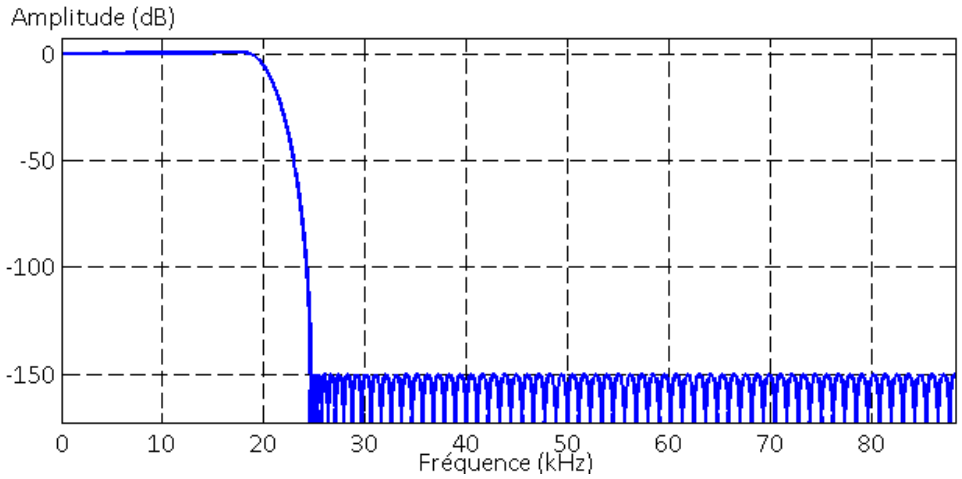

Figure 46 : Réponse fréquentielle du filtre FIR.

b) Convertisseur PCM vers sigma-delta, ordre 1

Pour réaliser ce modulateur (cf. Figure 47), il suffit de suivre le schéma général présenté précédemment. Il y a toutefois deux astuces qui sont utilisées. La première sert à convertir le bit sigma-delta, valant 0 ou 1, en nombre -1 et 1 directement dans le soustracteur. Pour cela, on choisit un opérateur qui va pouvoir faire l'addition et la soustraction et dont l'opération est commandée par le signal de sortie. Sur l'entrée invariante, on relie l'étage d'entrée et sur l'entrée dont le signe peut être modifié on relie la constante 1. La deuxième astuce se situe au niveau de l'absence d'échantillonneur ; cette opération est directement réalisée par le registre dans l'accumulateur.

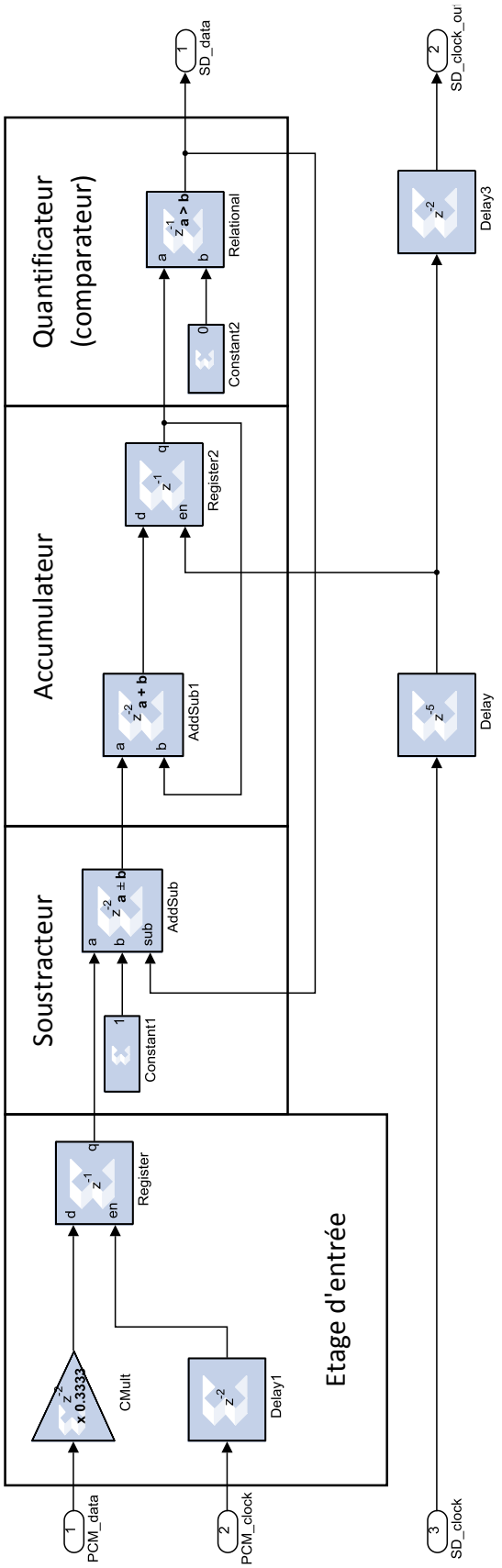

Figure 47 : Réalisation du modulateur permettant de passer d'une modulation PCM à une modulation sigma-delta.

c) Convertisseur PCM vers sigma-delta, ordre 5

La Figure 48 présente le modulateur sigma-delta qui sera implémenté. Il faut noter qu'il y a une erreur sur le schéma, en effet les blocs z^{-1} ne se situent non pas sur la chaine directe, mais sur les rétroactions qui font office d'accumulateur pour chaque étages.

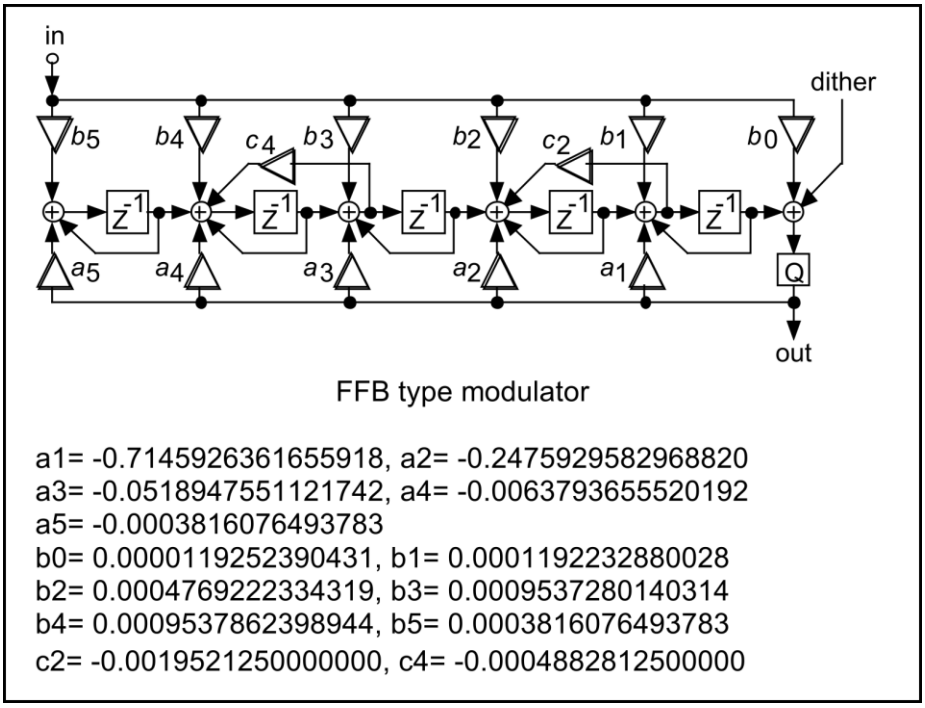

Figure 48 : Schéma du modulateur sigma-delta d'ordre 5 réalisé. D'après [15], Figure 2, page 2.

Sur la Figure 49 on peut voir le code C permettant de réaliser le modulateur. Normalement, le langage de programmation C est utilisé pour programmer des logiciels, mais la société Xilinx fournit un outil pour transformer ce code C logiciel en structure matérielle, et le rôle des instructions commençant par #pragma est d'indiquer au compilateur comment transformer certaines parties du code. Les deux premières lignes et la dernière servent à délimiter la fonction C réalisant le modulateur dans le cas où il y aurait plusieurs fonctions différentes dans le code source (ce qui n'est pas le cas ici).

En rouge sont notés les mots clés du langage et dans le cas présent il n'est pas nécessaire de les connaitre pour les comprendre. En vert sont notés les formats utilisés pour représenter les nombres en binaire ; DataIn_t correspond à des nombres en 32 bits flottant car il s'agit des échantillons PCM ; DataOut_t correspond à des nombres sur un seul bit car il s'agit des échantillons sigma-delta ; Coef_t correspond aux coefficients à appliquer dans le

modulateur et ils sont codés en virgule fixe s0.34 bits (1 bit de signe, 0 bit avant la virgules et 34 bits après la virgule) ; enfin, Path_t correspond aux calculs intermédiaires et utilise le format s4.43 (1 bit de signe, 4 bits avant la virgule et 43 bits après la virgule).

Lors de la définition de la fonction, on définit les variables Din et Dout grâces auxquelles on va pouvoir récupérer les échantillons entrant dans le modulateur et retourner les échantillons sigma-delta calculés. La lettre N entre crochet après Dout signifie qu'il y a N échantillons sigma-delta (64 dans le cas étudié) pour un unique échantillon PCM. Les lignes 5 à 16 définissent les valeurs des constantes des multiplications sur le schéma.

Sur le schéma, on constate que l'entrée est directement multipliée par les constantes b et on a besoin d'effectuer cette opération uniquement à chaque fois que l'on reçoit un nouvel échantillon ; c'est pourquoi on définit R1 à la ligne 17 pour retenir le résultat des 6 multiplications qui sont réalisées à la ligne 20.

Les lignes 22 à 35 servent à réaliser la boucle du modulateur. Dans un premier temps, on définit FB qui va retenir le résultat des 5 multiplications de la sortie par les coefficients a, cette opération est réalisée en ligne 29. Ensuite, on définit I à la ligne 23 qui va retenir le résultat de chacun des sommateurs après le calcul de chaque échantillon sigma-delta et cette opération est réalisée aux lignes 30 et 32 ; l'indication static signifie qu'on ne réinitialise pas I à chaque nouvel appel de la fonction (c'est-à-dire à chaque nouvel échantillon PCM) ; last sert à retenir la valeur de l'échantillon sigma-delta que l'on vient tout juste de calculer et ce calcul s'effectue à la ligne 34. On notera la présence de deux boucles OutputLoop et IntegratorLoop. OutputLoop sert à répéter le calcul N fois pour calculer les N échantillons sigma-delta successifs à calculer pour chaque échantillon PCM. IntgratorLoop sert à calculer le résultat de chacun des 6 sommateurs. C'est à la ligne 34 qu'on affecte à Dout le résultat des échantillons calculés. Enfin, la ligne 36 sert à mettre fin à la fonction puisqu'on a calculé tous les échantillons sigma-delta.

```
1  | void ordre5(DataIn_t Din, DataOut_t Dout[N])
2  | {
3  | #pragma HLS INTERFACE ap_fifo depth=64 port=Dout
4  | #pragma HLS INTERFACE ap_vld register port=Din
5  |         const Coef_t A[6] = {
6  |                       0,                  -0.7145926361655918,
7  |                     -0.2475929582968820, -0.0518947551121742,
8  |                     -0.0063793655520192, -0.0003816076493783};
9  |         const Coef_t B[6] = {
10 |                       0.0000119252390431,  0.0001192232880028,
11 |                       0.0004769222334319,  0.0009537280140314,
12 |                       0.0009537862398944,  0.0003816076493783};
13 |         const Coef_t C[6] = {
14 |                       0,                    0,
15 |                     -0.0019521250000000,  0,
16 |                     -0.0004882812500000,  0};
17 |         Path_t R1[6];
18 |         InputLoop:for(int i=0; i<6; i++)
19 | #pragma HLS PIPELINE
20 |               R1[i] = (CvtDataIn_t) (Din/3.f) * B[i];
21 |
22 |         Path_t FB[6];
23 |         static Path_t I[7] =  {0};
24 |         static DataOut_t last = 0;
25 |
26 |         OutputLoop:for(int j=0; j<N; j++) {
27 | #pragma HLS PIPELINE II=16
28 |               IntegratorLoop:for(int i=5; i>0; i--) {
29 |                     FB[i] = last ? A[i] : (Coef_t) -A[i];
30 |                     I[i] += R1[i] + I[i+1] + FB[i] + C[i] * I[i-1];
31 |               }
32 |               I[0] = I[1] + R1[0];
33 |               last = I[0] > 0 ? (DataOut_t)1 : (DataOut_t)0;
34 |               Dout[j] = last;
35 |         }
36 |         return;
37 | }
```

Figure 49 : Extrait du code C permettant de réaliser le modulateur PCM vers sigma-delta d'ordre 5.

6. Gain

a) PCM

L'opération de gain sur une modulation PCM (cf. Figure 50) se résume à une simple multiplication entre la valeur de l'échantillon PCM et la valeur du gain à appliquer. Il est à noter une fois de plus qu'on travaille en 32 bits flottants et que par conséquent chaque opérateur retourne un nombre en 32 bits flottants. On n'a donc pas besoin de supprimer de bits de poids fort ou de poids faible.

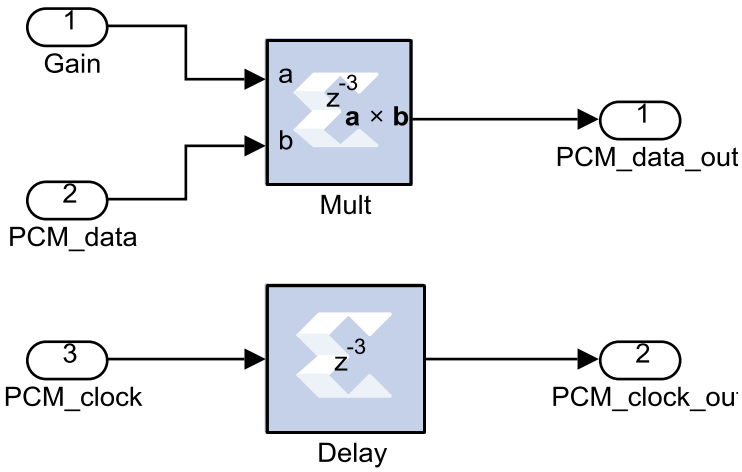

Figure 50 : Réalisation de l'opération de gain sur une modulation PCM.

b) Sigma-delta

La Figure 51 montre l'opération de gain sur une modulation sigma-delta. On retrouve la forme détaillée précédemment avec l'opération de gain à proprement parler au sein de l'étage d'entrée et le retour sur un bit unique à travers un modulateur d'ordre 1.

L'étage d'entrée se compose d'un multiplexeur qui fait office de multiplieur du gain par le bit de la modulation sigma-delta, étant donné que la modulation sigma-delta ne peut prendre comme valeur 1 (bit à 1) ou -1 (bit à 0), le résultat de la multiplication est soit la valeur du gain, soit son opposé. Ce gain est codé sur 16 bits ; ce qui lui donne comme valeurs extrêmes 0 et 65535.

Le modulateur sigma-delta est légèrement différent de celui présenté précédemment, mais reprend la même structure. Il faut noter que dans ce modulateur la constante sur l'additionneur vaut 65535, ce qui correspond à la valeur maximale du gain et implique qu'on ne peut pas amplifier le signal. Si on veut pouvoir le faire, il faut une valeur de gain supérieure à la valeur de cette constante (on peut soit diminuer cette constante, soit augmenter le nombre de bits pour coder la valeur du gain).

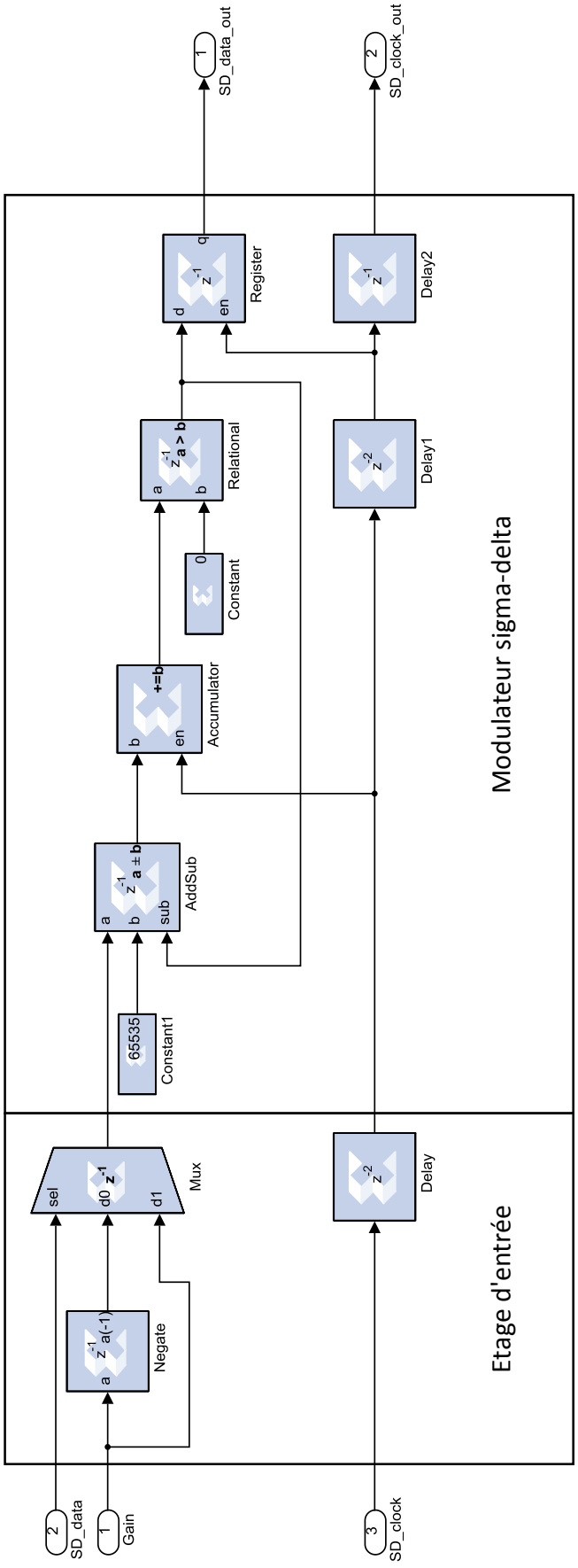

Figure 51 : Réalisation d'un gain sur une modulation sigma-delta.

7. Mélange

a) PCM

Sur la Figure 52, on peut voir une façon de réaliser un mélange de deux signaux au format PCM. Comme il n'est pas évident que les deux signaux arrivent de façon synchrone, on place des registres qui vont stocker les valeurs des échantillons et synchroniser les deux canaux. L'additionneur est activé lorsqu'un échantillon arrive sur le premier canal et par conséquent si on ne branche pas de modulation sur ce canal, les données du deuxième canal ne pourront pas transiter.

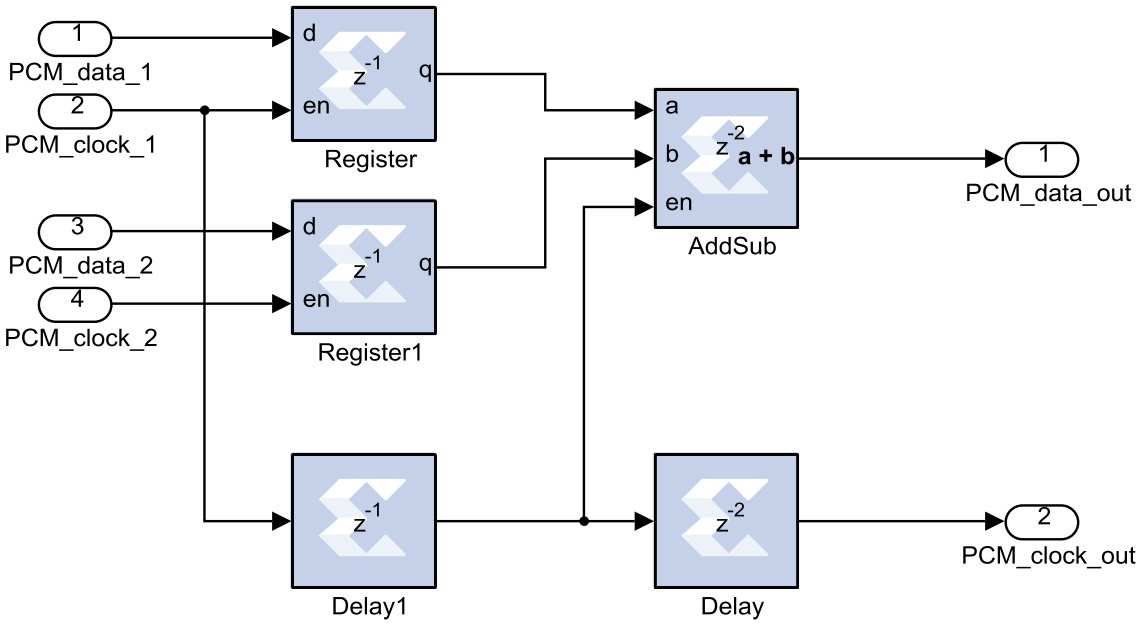

Figure 52 : Réalisation de l'opération de gain sur une modulation PCM.

b) Sigma-delta

L'opération de mélange appliquée à une modulation sigma-delta (cf. Figure 53) se compose d'un étage d'entrée qui met en forme les modulations sigma-delta et les synchronise sur l'horloge du premier canal, comme dans le cas de la modulation PCM. En deuxième position, on trouve la sommation et enfin le modulateur sigma-delta. Malgré le fait qu'en modulation sigma-delta, lorsqu'on somme deux canaux ayant pour valeur 1 on trouve 2, on ne place pas une constante égale à 2 sur la rétroaction mais bien égale à 1. En effet, lorsqu'on utilise la modulation sigma-delta, il faut regarder la valeur moyenne de l'amplitude de la somme des deux échantillons. On peut avoir 3 valeurs : 2 (1+1), 0 (1-1 et -1+1) et -2 (-1-1), ce qui en amplitude donne deux cas pour obtenir 2 et deux cas pour obtenir 0, soit une moyenne de 1, d'où la valeur de la constante dans la rétroaction.

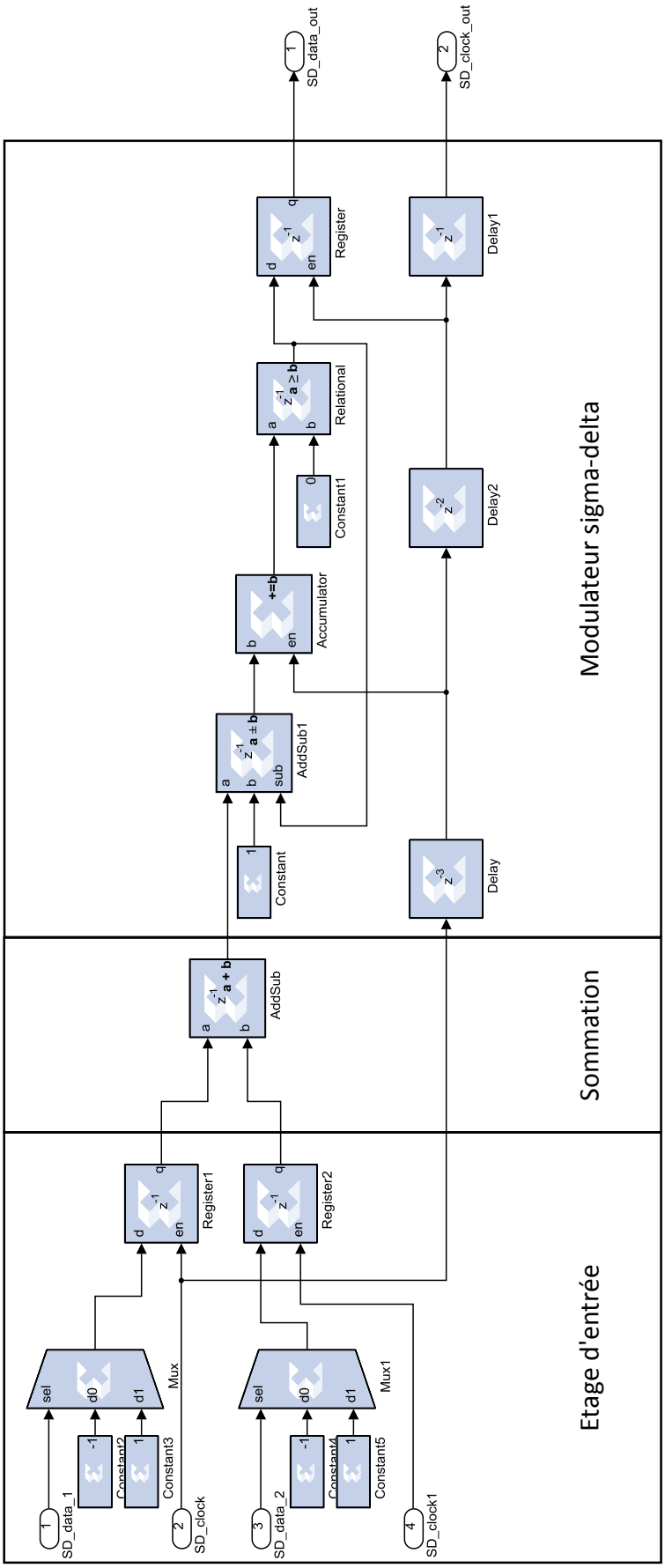

Figure 53 : Réalisation de l'opération de mélange sur une modulation sigma-delta.

8. Communication entre les dispositifs et contrôle de leurs paramètres

a) Communication entre le logiciel et le matériel

Les paramètres du matériel sont commandés directement à partir d'une adresse mémoire qui est donnée par le logiciel qui réalise l'implémentation des composants sur la ZedBoard. Pour des raisons de sécurité, le noyau Linux interdit à l'utilisateur d'écrire directement dans ces emplacements, mais, il existe une astuce qui consiste à faire correspondre à ces emplacements inaccessibles des emplacements mémoire accessibles à l'utilisateur et c'est le rôle de la fonction C mmap().

Pour illustrer le fonctionnement de mmap(), on va étudier l'initialisation du convertisseur analogique/numérique. Pour réaliser l'interface entre le processeur et l'interface Pmod, on a inséré un module AXI GPIO (General Purpose Input Output) qui permet de contrôler depuis le processeur l'état haut ou bas des broches du composant Zynq qui sont reliées au module Pmod et donc aux broches de paramétrage du convertisseur. On a donc un espace mémoire qui correspond à l'état des broches de paramétrage du convertisseur et en modifiant les données qu'il contient on va pouvoir modifier le paramétrage de ce convertisseur. Ce module AXI GPIO est configuré de telle sorte à disposer de 5 bits configurés en sortie (par rapport au composant Zynq) dont le bit 0 correspond à la broche FMT0 du convertisseur, le bit 1 à FMT1, le bit 2 à FS0, le bit 3 à FS1 et le bit 4 à FS2. D'après la documentation du convertisseur et par rapport à l'application qu'on veut réaliser, il faut programmer FMT0 à 1, FMT1 à 1, FS0 à 1, FS1 à 0 et FS2 à 1. Ces deux informations conduisent à écrire le nombre binaire 00010111 dans la case mémoire correspondant à l'interface AXI GPIO.

Le code C réalisant cette opération est donné en Figure 54. Les lignes commençant par le mot clé #define (en violet) permettent d'associer un nom avec un nombre afin de pouvoir faire référence à ce nombre plusieurs fois dans le texte en évitant de l'inscrire tout autant de fois, et cela permet d'améliorer la lisibilité du code. La ligne 1 correspond à la valeur de départ de la plage mémoire occupée par l'interface AXI GPIO. La ligne 2 correspond à la taille de cette plage mémoire et ces deux valeurs sont données par le logiciel qui permet de programmer le composant Zynq. Les lignes 4 à 8 décrivent les différents emplacements mémoire par l'intermédiaire desquels on va pouvoir dialoguer avec le composant AXI GPIO. Ces valeurs sont

données par la documentation de l'interface et on utilisera uniquement l'emplacement qui permet de modifier les données en sortie, qui est donné en ligne 4. La ligne 10 permet de créer une variable base_ptr qui va retenir l'emplacement mémoire dans lequel on peut écrire et qui est associé à l'emplacement mémoire de l'interface AXI GPIO (mais dans lequel on ne peut pas écrire directement). Par la suite, on définit deux fonctions, l'une entre les lignes 12 et 15, et l'autre entre les lignes 17 et 25.

La première fonction PCM4202_write_gpio() sert à écrire dans l'emplacement mémoire associé la valeur contenue dans la variable data qui est donnée en paramètre de cette fonction. La deuxième fonction PCM4202_init() sert à réaliser l'initialisation de mmap() pour les utilisations futures de l'interface AXI GPIO et elle en profite pour écrire la valeur permettant au convertisseur de fonctionner. La ligne 19 consiste justement à appeler la fonction mmap() pour créer une nouvelle association avec l'espace mémoire de l'interface AXI GPIO en précisant ses caractéristiques et le nouvel espace mémoire est stocké dans la variable base_ptr. Sur la ligne 21, on définit une variable i qui stocke la valeur à écrire sur les broches du convertisseur et on appelle la première fonction définie dans le code avec le paramètre i pour inscrire cette valeur dans l'espace mémoire associé qui va se répercuter dans l'espace mémoire de l'interface GPIO, puis sur les broches du composants Zynq et enfin sur les broches concernées du convertisseur numérique/analogique PCM4202.

```
1  | #define GPIO_ADDR_BASE 0x41210000
2  | #define GPIO_ADDR_SIZE 0x00010000
3  |
4  | #define GPIO_OFFSET_DATA  0x00000000
5  | #define GPIO_OFFSET_TRI   0x00000004
6  | #define GPIO_OFFSET_GIER  0x0000011C
7  | #define GPIO_OFFSET_IER   0x00000128
8  | #define GPIO_OFFSET_ISR   0x00001200
9  |
10 | static char *base_ptr = 0;
11 |
12 | void PCM4202_write_gpio(int data)
13 | {
14 |     *((int *) (base_ptr + GPIO_OFFSET_DATA)) = data & 0x000000FF;
15 | }
16 |
17 | int PCM4202_init(int fd)
18 | {
19 |     base_ptr = (char *) mmap(NULL, GPIO_ADDR_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, GPIO_ADDR_BASE);
20 |
21 |  int i = 0b00010111;
22 |  PCM4202_write_gpio(i);
23 |
24 |     return 0;
25 | }
```

Figure 54 : Extrait du code C permettant d'initialiser le convertisseur analogique/numérique PCM4202.

Ce code pourrait s'apparenter à un pilote (ou driver) (même s'il n'en constitue pas un d'un point de vue de Linux et de Linaro) dans la mesure où il décrit les interactions possibles entre le logiciel et le matériel. Dès qu'on aura besoin de modifier le paramétrage du convertisseur, on utilisera les fonctions définies précédemment et on n'aura plus besoin de savoir que le convertisseur PCM4202 est relié au processeur par une interface AXI GPIO. On parle alors de couches successives et on a réalisé la couche de communication entre le logiciel et le matériel.

b) Intégration dans l'environnement de Pure Data [16]

Après avoir réalisé la couche de communication avec le matériel, on va réaliser la couche qui va faire le lien avec l'interface utilisateur Pure Data pour la modification du gain pour une modulation sigma-delta. Ce code source est présenté en Figure 55 et ne sera pas expliqué plus en détail. Dans ce code, on définit 3 fonctions, la première des lignes 1 à 5, la deuxième des lignes 7 à 26 et la troisième des lignes 28 à 33.

La première fonction pdDSDGain_set() est appelée par Pure Data dès que l'utilisateur demande la modification de la valeur du gain. Cette fonction possède deux paramètres ; le paramètre x correspond à l'objet Pure Data ; le paramètre gain correspond à la nouvelle valeur de gain qu'il faut transmettre au matériel, *via* la couche de communication. La ligne 3 permet d'enregistrer la nouvelle valeur de gain dans la variable gain contenue dans x. On multiplie cette valeur par 65535 car la valeur de gain, donnée en paramètre, utilise un codage en virgule flottante compris entre 0 et 1 et on a vu précédemment que pour le gain en sigma-delta, il faut une valeur comprise entre 0 et 65535 et codée sur 16 bits. La ligne 4 permet de dire à la couche de communication que l'on veut modifier le gain pour le canal associé à l'objet Pure Data.

La deuxième fonction pdDSDGain_new() permet l'initialisation de l'objet Pure Data. On y retrouve en ligne 9 la création de l'objet Pure Data, en ligne 12 l'initialisation de la couche de communication, en lignes 15 à 21 le choix du canal pour appliquer le gain (dans l'exemple suivant on a réalisé deux fois le dispositif de modification de gain pour une modulation sigma-delta) et en ligne 23 la création d'une broche pour relier l'objet à d'autres objets.

La dernière fonction pdDSDGain_setup() est obligatoire, c'est celle que Pure Data appelle quand l'utilisateur veut créer l'objet et si cette fonction n'existe pas, Pure Data retourne un message d'erreur à l'utilisateur. La ligne 30 sert à indiquer à Pure Data quel type d'objet il doit utiliser en interne et que la fonction pdDSDGain_new() qui a été définit juste avant permet de réaliser l'initialisation. La ligne 32, couplée à la ligne 23, permet d'indiquer que lorsqu'il y a une modification de la valeur sur la broche de l'objet graphique (qui correspond au nouveau gain à appliquer), on doit appeler la fonction pdDSDGain_set() qui a été définie en premier.

```
1  |  void pdDSDGain_set(t_pdDSDGain *x, t_floatarg gain)
2  |  {
3  |      x->gain = gain * 65535.f;
4  |      FPGA_DSD_setGain(x->channel, x->gain);
5  |  };
6  |
7  |  void *pdDSDGain_new(t_symbol* s, int argc, t_atom *argv)
8  |  {
9  |      t_pdDSDGain *x = (t_pdDSDGain*)pd_new(pdDSDGain_class);
10 |
11 |      int fd = MMAP_open();
12 |      if(FPGA_PCM_open(fd))
13 |          return NULL;
14 |
15 |      x->channel = 1;
16 |      if(argc)
17 |      {
18 |          x->channel = atom_getint(argv);
19 |          if( (x->channel != 1) & (x->channel != 2) )
20 |              x->channel = 1;
21 |      }
22 |
23 |      inlet_new(&x->x_obj, &x->x_obj.ob_pd, gensym("float"),
gensym("set"));
24 |
25 |      return (void*)x;
26 |  }
27 |
28 |  void pdDSDGain_setup(void)
29 |  {
30 |      pdDSDGain_class = class_new(gensym("pdDSDGain"),
(t_newmethod)pdDSDGain_new, 0, sizeof(t_pdDSDGain), CLASS_NOINLET, A_GIMME, 0);
31 |
32 |      class_addmethod(pdDSDGain_class, (t_method)pdDSDGain_set,
gensym("set"), A_FLOAT, 0);
33 |  }
```

Figure 55 : Extrait du code C permettant de réaliser l'interface entre la couche de communication et l'interface utilisateur Pure Data pour la modification de la valeur de gain pour une modulation sigma-delta.

9. Conclusion

Dans ce chapitre, on a vu que la réalisation d'un système de traitement audio complexe peut se décomposer en couches successives plus simples et indépendantes. Par ailleurs, la réalisation des dispositifs reprend fidèlement les structures théoriques établies dans le chapitre précédent, auxquelles on peut ajouter quelques optimisations dépendantes du matériel. Enfin la communication entre les parties logicielles et matérielles s'effectue par de simples écritures en mémoire par l'intermédiaire de pilotes.

V - Conclusion générale

A travers ce mémoire on a pu avoir un aperçu d'une partie des systèmes numériques permettant de traiter le signal audio, depuis sa numérisation, jusqu'à son retour sous forme analogique. Pour cela on a réalisé différents convertisseurs analogique/numérique et numérique/analogique, ainsi qu'une partie des principaux traitements utilisés dans cette industrie : le gain, le mélange et le filtrage passe bas. Par ailleurs cette réalisation s'est accompagnée d'une réflexion sur l'utilisation de deux types de modulation permettant de représenter un signal analogique sous forme numérique : le sigma-delta et le PCM. On a pu voir qu'il était possible de réaliser des traitements sous l'une et l'autre des représentations, même si l'utilisation de la modulation sigma-delta n'en est qu'à ses balbutiements et que les processeurs qu'on utilise sont difficilement capables de traiter ce type d'information. Cependant l'utilisation de cette modulation pourrait permettre de réduire l'encombrement des composants et ainsi le coût de leur fabrication, tout en améliorant la latence introduite par le système de traitement.

Pour que la modulation sigma-delta se développe il faudrait pouvoir réaliser l'ensemble des traitements élémentaires qu'il est possible d'effectuer à partir de la modulation PCM, cela incluant, en plus des traitements déjà évoqués précédemment, tous les filtres (passe-haut et passe-bas du premier et du second ordre, paramétriques, shelve, …), le compresseur et la réverbération. Le problème du compresseur soulève la question de la mesure et de la représentation des signaux pour la visualisation et donc du montage son. Enfin il faudrait pouvoir s'assurer que la modulation sigma-delta puisse rivaliser en termes de qualité sonore avec la modulation PCM.

VI - Bibliographie

1. **Kester, Walt.** Analog-Digital Conversion. *Chapter 1 : Data Converter History.* 2004. http://www.analog.com/library/analogDialogue/archives/39-06/Chapter%201%20Data%20%Converter%20History%20F.pdf.

2. **Millot, Laurent.** *Traitement du signal audiovisuel : applications avec Pure Data.* s.l. : Dunod, 2008.

3. **Smith, Steven.** The Scientist and Engineer's Guide to Digital Signal Processing. 1999. http://www.dspguide.com.

4. *An Overview of Sigma-Delta Converters.* **Pervez Aziz, Henrik Sorensen, Jan Van Der Spiegel.** 1, 1996, IEEE Signal Processing Magazine, Vol. 13, pp. 61-84.

5. *Digital Audio Effects Applied Directly on a DSD Bitstream.* **Josh Reiss, Mark Sandler.** Naple : DAFx, 2004. 7th International Conference on Digital Audio Effects (DAFx'04). pp. 372-377.

6. *Bit-stream signal processing and its application to communication systems.* **H. Fujisaka, R. Kurata, M. Sakamoto, M. Morisue.** [éd.] IEEE. 3, 2002, IEEE Proc.-Circuits Devices Syst., Vol. 149, pp. 159-166.

7. **Griffin, Rich.** *Zynq Workshop for Beginners.* s.l. : Silica, 2014.

8. **NXP.** UM10204. *I2C-bus specification and user manual.* 2014. http://www.nxp.com/documents/user_manual/UM10204.pdf.

9. **Texas Intruments.** DSD1793 datasheet. 2006. http://www.ti.com/lit/ds/symlink/dsd1793.pdf.

10. **Philips Semiconductors.** I2S bus specification. 1986. https://sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf.

11. **ARM.** AMBA AXI and ACE Protocol Specification. 2011. https://capocaccia.ethz.ch/capo/raw-attachment/wiki/2014/microblaze14/AXI4_specification.pdf.

12. **.** (Sample) 3rd order Multiple Feedback Low-pass Filter Design Tool - Result -. *OKAWA Electronic Design.* 2009. http://sim.okawa-denshi.jp/en/Multiple3tool.php.

13. **Texas Intruments.** PCM4202 datasheet. 2004. http://www.ti.com/lit/ds/symlink/pcm4202.pdf.

14. **Sarioglu, Guner Refik.** *Sigma-Delta Signal Processing of Low Level Signals With Simplified Hardware.* Ohio State University. 2001. Master of Science.

15. *Investigation of Practical 1-bit Delta-Sigma Conversion for Professional Audio Applications.* **Hiroshi Takahashi, Ayataka Nishio.** 2001.

16. **Zmölnig, IOhannes M.** HOWTO write an External for Pure Data. 2014. http://iem.at/pd/externals-HOWTO/pd-externals-HOWTO.pdf.

Annexe 1 : Bruit de quantification et dynamique

1. Puissance du bruit de quantification

Considérons le cas d'un quantificateur découpant l'intervalle de tension $[-V_{max}, V_{max}]$ en 2^{N_b} intervalles de même largeur $\Delta = \frac{2 \cdot V_{max}}{2^{N_b}} = V_{max} \cdot 2^{1-N_b}$ où N_b correspond au nombre de bits du quantificateur. Comme le bruit de quantification correspond à l'erreur que réalise le quantificateur sur chaque intervalle, sa tension u_{err} est donc comprise dans l'intervalle $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$ et il peut être assimilé à un bruit blanc (pour peu que le signal d'entrée soit suffisamment important) de moyenne nulle et sa densité de probabilité p est uniforme sur cet intervalle.

$$p(u_{err}) = \begin{cases} \frac{1}{\Delta} & \text{si } u_{err} \in \left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right] \\ 0 & \text{sinon} \end{cases}$$

Autrement dit, à un instant donné, on suppose que la tension peut prendre n'importe quelle valeur dans l'intervalle $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$ avec la même probabilité $\frac{1}{\Delta}$.

On peut calculer la puissance σ_e^2 du bruit de fond en intégrant l'ensemble des puissances possibles multipliées par leur probabilité d'existence (ce calcul est identique à celui de la variance de p) :

$$\sigma_e{}^2 = \int_{-\infty}^{\infty} u^2 \cdot p(x) du\,,$$

$$\sigma_e{}^2 = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \frac{u^2}{\Delta} du.$$

Après intégration, on obtient :

$$\sigma_e{}^2 = \frac{\Delta^2}{12}.$$

Par définition, la répartition spectrale P_{BB} du bruit blanc est constante sur l'intervalle $[0, N_o \cdot F_s]$ où Fs correspond à la fréquence d'échantillonnage cible et $N_o = 2^r$ le facteur de sur-échantillonnage du convertisseur avec r un entier naturel quelconque. On peut donc écrire :

$$P_{BB}(f) = \frac{{\sigma_e}^2}{N_o \cdot F_s},$$
$$P_{BB}(f) = \frac{\Delta^2}{12 \cdot N_o \cdot F_s},$$

où f est la fréquence considérée.

Ainsi, on peut calculer la densité spectrale de puissance de bruit P_{Bruit} dans le modulateur :

$$P_{Bruit}(f) = P_{BB}(f) \cdot |H_{Bruit}(f)|^2,$$
$$P_{Bruit}(f) = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \cdot |H_{Bruit}(f)|^2,$$

avec $H_{Bruit}(f)$ la fonction de transfert du bruit de fond pour chacun des modulateurs PCM et sigma-delta.

La puissance de bruit de fond $P_{Bruit}^{f_0}$ sur la gamme de fréquence [0, f_0] se calcule par intégration :

$$P_{Bruit}^{f_0} = \int_{-f_0}^{f_0} P_{Bruit}(f)\, df\,,$$
$$P_{Bruit}^{f_0} = \int_{-f_0}^{f_0} \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \cdot |H_{Bruit}(f)|^2\, df.$$

Après avoir sorti les facteurs constants de l'intégrale, on obtient :

$$P_{Bruit}^{f_0} = \frac{\Delta^2}{12 \cdot N_o \cdot F_s} \int_{-f_0}^{f_0} |H_{Bruit}(f)|^2\, df.$$

2. Dynamique d'un système numérique

On s'intéressera uniquement au cas où le signal d'entrée est sinusoïdal, sa puissance est donc définie par la relation :

$$P_{sin}(V) = \frac{V^2}{2},$$

où V correspond à la tension maximale de la sinusoïde.

La dynamique Dyn d'un signal correspond au rapport de la puissance de signal sur la puissance de bruit exprimé en décibels. Dans la suite chaque fois que l'on parlera de dynamique on se réfèrera toujours à un signal de référence sinusoïdal. Ainsi, on peut écrire que :

$$Dyn(V) = 10 \cdot log_{10}\left(\frac{P_{sin}(V)}{P_{Bruit}^{f_0}}\right),$$

$$Dyn(V) = 20 \cdot log_{10}\left(\frac{V}{\sqrt{2}}\right) - 10 \cdot log_{10}\left(P_{Bruit}^{f_0}\right).$$

Comme la dynamique Dyn0 d'un convertisseur correspond à la dynamique que l'on obtient pour un signal d'amplitude maximale, on a donc :

$$Dyn_0 = 20 \cdot log_{10}\left(\frac{V_{max}}{\sqrt{2}}\right) - 10 \cdot log_{10}\left(P_{Bruit}^{f_0}\right).$$

Annexe 2 : Calcul en binaire

1. Représentation des nombres

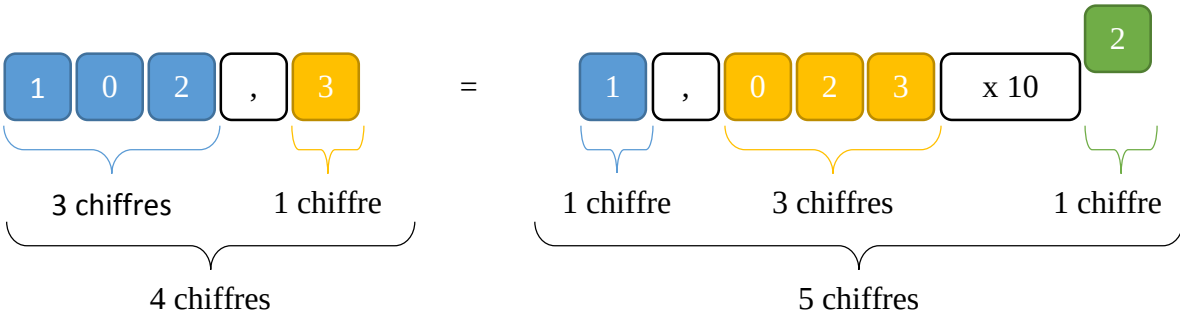

Figure 56 : Représentation du nombre 102,3 en virgule fixe à gauche et en virgule flottante à droite.

En mathématique il existe deux façons de représenter les nombres (cf. Figure 56) : la représentation en virgule fixe et la représentation en virgule flottante. La représentation en virgule fixe consiste à représenter chaque nombre à l'aide d'un nombre connu à l'avance de chiffres avant et après la virgule. La représentation en virgule flottante, plus couramment appelée écriture scientifique, consiste à représenter tous les nombres en utilisant un seul chiffre non nul avant la virgule et en indiquant un exposant qui va permettre de retrouver le nombre de départ.

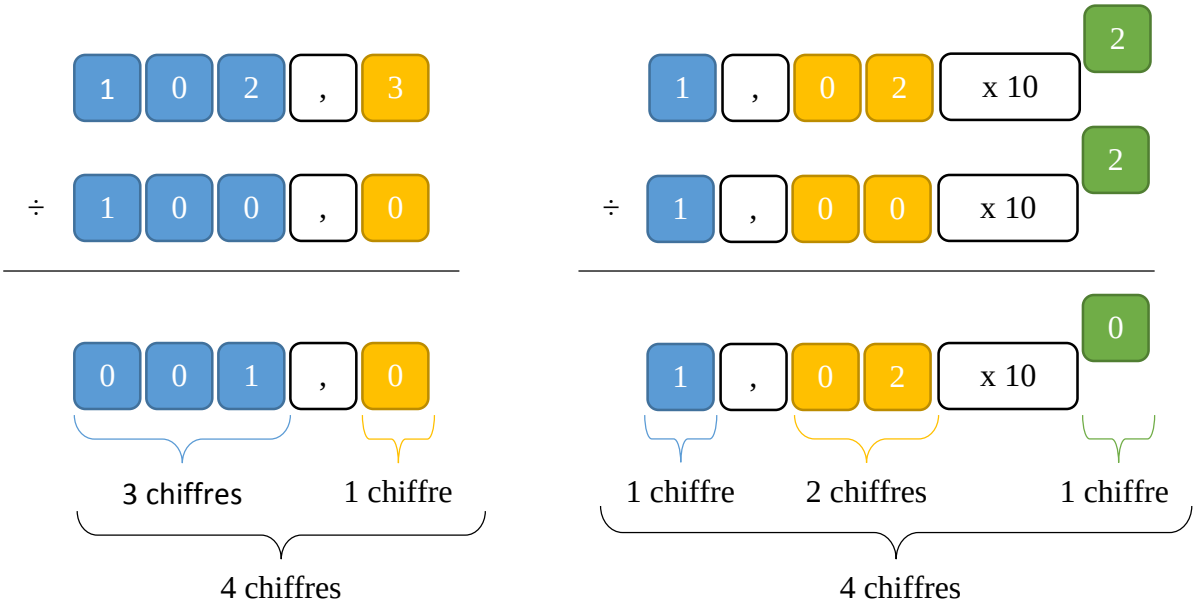

Figure 57 : Division du nombre 102,3 par 100 avec le plus de précision possible et en utilisant le même nombre de chiffres : en virgule fixe à gauche et en virgule flottante à droite.

Il peut sembler aberrant d'utiliser la notation en virgule flottante étant donné qu'elle demande plus de nombres ; en effet, si on veut écrire 102,3 le plus précisément possible en virgule flottante et en utilisant le même nombre de chiffres qu'en virgule fixe, on l'écrira $1{,}02 \times 10^2$, mais l'intérêt de la virgule flottante est de pouvoir représenter des nombres très petits et très grands avec la même précision. En effet, si on divise par 100 le nombre précédent, comme sur la Figure 57, la représentation en virgule flottante devient plus précise.

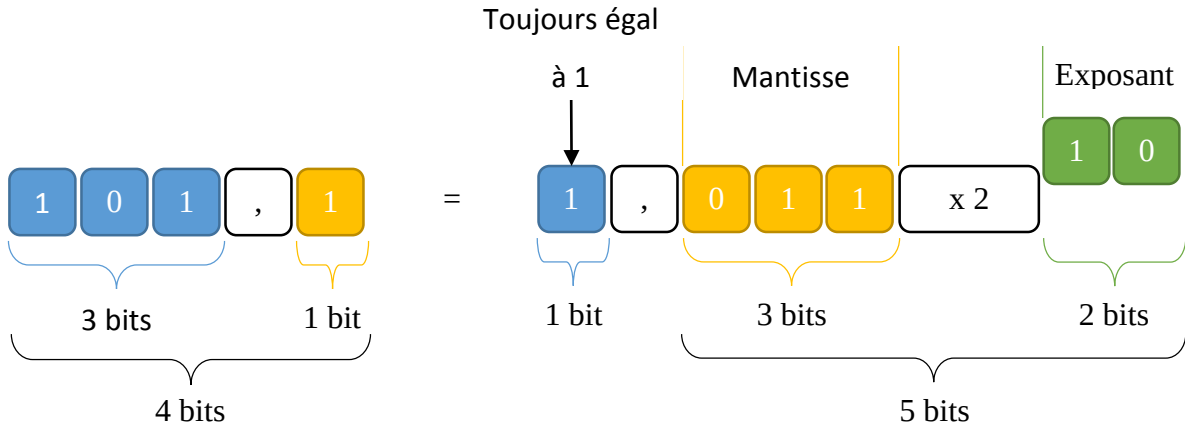

Figure 58 : Représentation du même nombre binaire en virgule fixe à gauche et en virgule flottante à droite.

On a étudié cette représentation dans le cadre des nombres décimaux. Pour ce qui est du binaire on suit le même principe, mais maintenant chaque chiffre, qu'on appelle bit (contraction de binary digit : élément binaire), peut seulement prendre la valeur 0 ou 1 et on remplace le facteur 10 (représentant la base de numération en binaire) avant l'exposant par un 2 (représentant la base de numération en binaire), comme indiqué sur la Figure 58. Dans la représentation en virgule flottante, le bit avant la virgule, étant obligatoirement non nul, il est toujours égal à 1, et ne rentre donc pas en compte dans le calcul du nombre total de bits. On utilisera la notation 3.1 bits pour désigner un nombre positif de 4 bits composé de 3 bits avant la virgule et de 1 bit après, et lorsque l'on écrit s3.1 bits cela désigne un nombre positif ou négatif de 5 bits composé d'un bit de signe, de 3 bits avant la virgule et d'un bit après la virgule. Pour ce qui est des nombres en virgule flottante, on utilisera par la suite le format simple précision décrit dans la norme IEEE 754 et qui reprend les principes exposés ci-dessus.

2. Addition et multiplication en virgule fixe

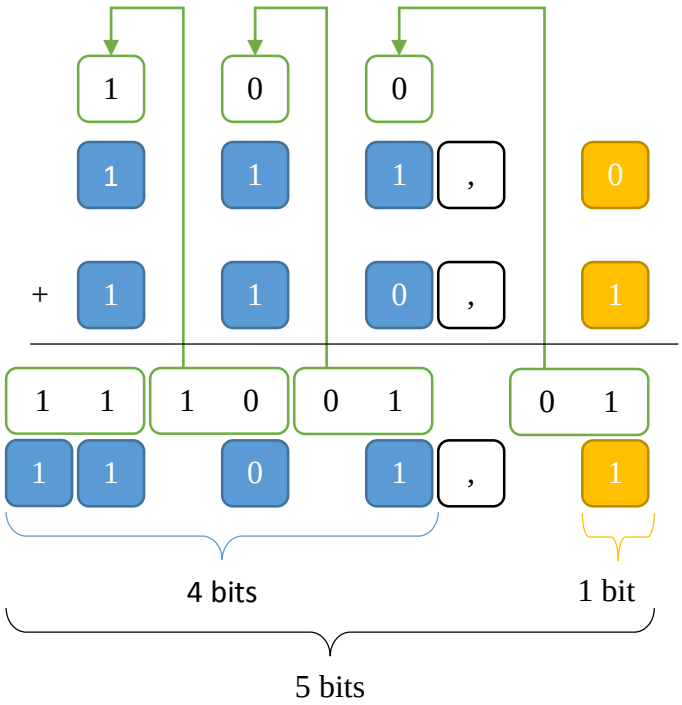

Figure 59 : Réalisation d'une addition avec des nombres binaires à virgule fixe.

On réalise une addition en binaire de la même manière qu'en décimal, comme le montre la Figure 59. En décimal on utilise les 10 chiffres de 0 à 9 et lorsque l'addition de deux nombre dépasse 9 on pose le chiffre de droite, celui des unités, et on retient le chiffre de gauche, celui des décimales. Il suffit d'appliquer le même principe en binaire : on utilise les deux chiffres 0 et 1 et si le résultat de l'addition dépasse 1, il faut deux bits pour l'écrire et on pose le bit de droite puis on retient le bit de gauche. On constate aussi qu'il faut n+1 bits pour écrire le résultat de l'addition de deux nombres de n bits.

De la même manière, sur la Figure 60, pour réaliser une multiplication en binaire, il suffit de multiplier successivement le premier nombre par chacun des chiffres composant le second nombre, en tenant compte du décalage lié à la position de ce chiffre pour réaliser l'addition finale. Etant donné qu'en binaire on multiplie uniquement par 0 ou 1, il suffit alors d'ajouter ou non ce premier nombre décalé par rapport à la position du bit considéré du deuxième nombre. Pour écrire le résultat, il faut donc n+m bits, avec n et m les nombres de bits de chacun des deux nombres.

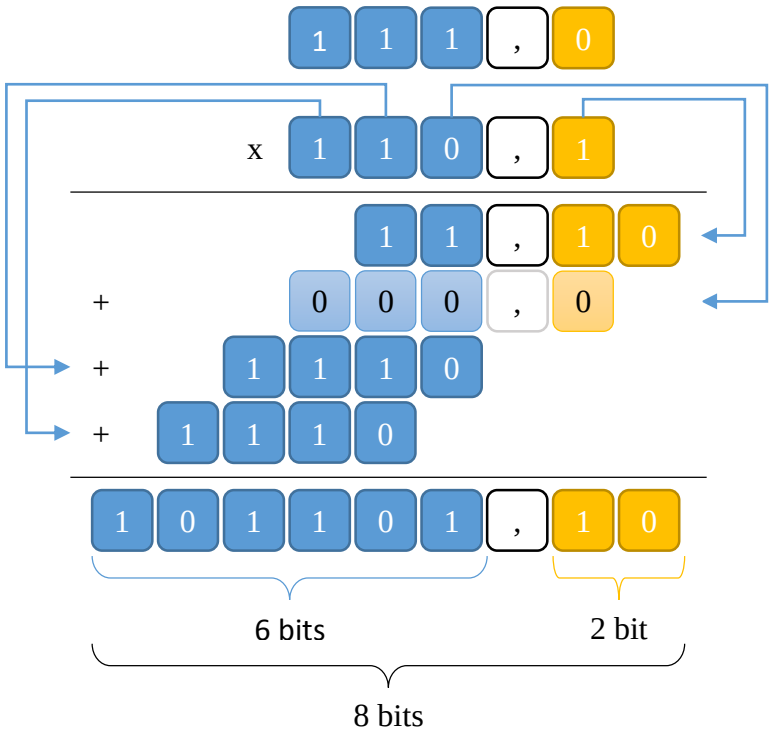

Figure 60 : Réalisation d'une multiplication avec des nombres binaires à virgule fixe.

3. Conversion des nombres à virgule fixe

a) Suppression des bits de poids faible

Lorsque l'on supprime des bits de poids faible, comme en Figure 61, on peut soit réaliser un arrondi, soit une troncature. Il existe trois arrondis : l'arrondi au supérieur, l'arrondi au plus proche et l'arrondi à l'inférieur. L'arrondi au supérieur consiste à prendre le nombre immédiatement plus grand en valeur absolue que l'on peut coder dans le nouveau système ; l'arrondi à l'inférieur consiste à prendre le nombre immédiatement plus petit en valeur absolue ; l'arrondi au plus proche consiste à réaliser un arrondi au supérieur si le dernier bit retiré est 1 et un arrondi à l'inférieur sinon. La troncature consiste à supprimer purement et simplement les bits de poids faible dans la mise en forme du nouveau nombre.

	Arrondi au supérieur	Arrondi à l'inférieur	Arrondi au plus proche	Troncature
1 0 1 , 1	1 1 0	1 0 1	1 1 0	1 0 1
1 0 1 , 0	1 1 0	1 0 1	1 0 1	1 0 1
- 1 0 1 , 1	- 1 0 1	- 1 1 0	- 1 1 0	- 1 0 1
- 1 0 1 , 0	- 1 0 1	- 1 1 0	- 1 0 1	- 1 0 1

Figure 61 : Exemple d'arrondi et de troncature pour différents nombres.

b) Suppression des bits de poids fort

Lorsqu'on supprime des bits de poids fort on peut soit réaliser un enroulement, soit réaliser une saturation. L'enroulement consiste à ne pas tenir compte des bits de poids fort supprimés, c'est-à-dire qu'on se contente de récupérer les bits intéressant. Sur la Figure 62, le phénomène d'enroulement est décrit par la courbe bleue. Dans le nouveau système de codage, on constate que lorsqu'on dépasse d'une unité la plus grande valeur possible on obtient la plus petite valeur nouvellement codable ; de la même manière, lorsqu'on retranche une unité à la plus petite valeur que l'on peut coder, on obtient finalement la plus grande. La saturation, quant à elle, consiste à remplacer les nombres trop grands ou trop petits pour être codés avec moins de bits par les nouvelles valeurs extrêmes possibles.

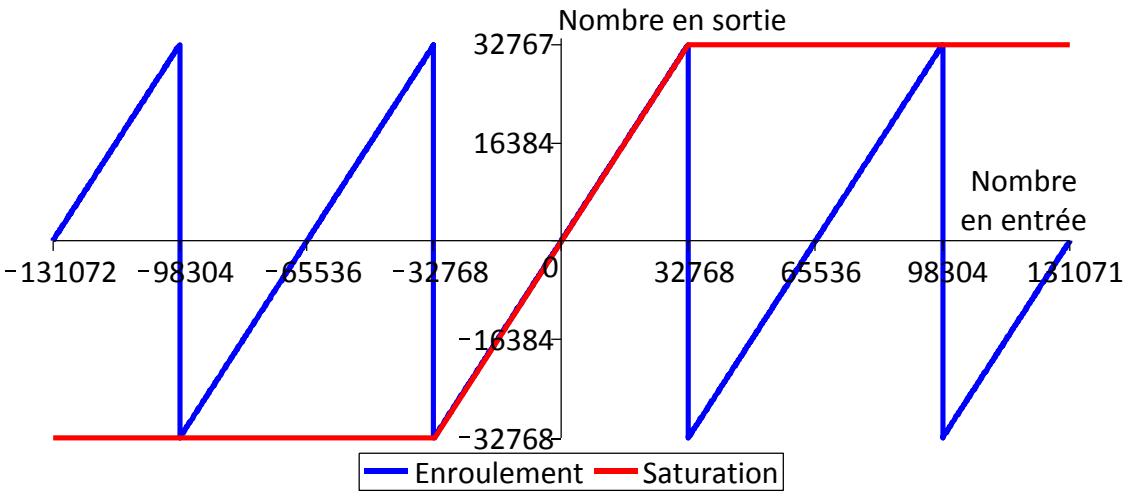

Figure 62 : Illustration de l'enroulement et de la saturation lors du passage de 18 bits à 16 bits.

4. Addition et multiplication en virgule flottante

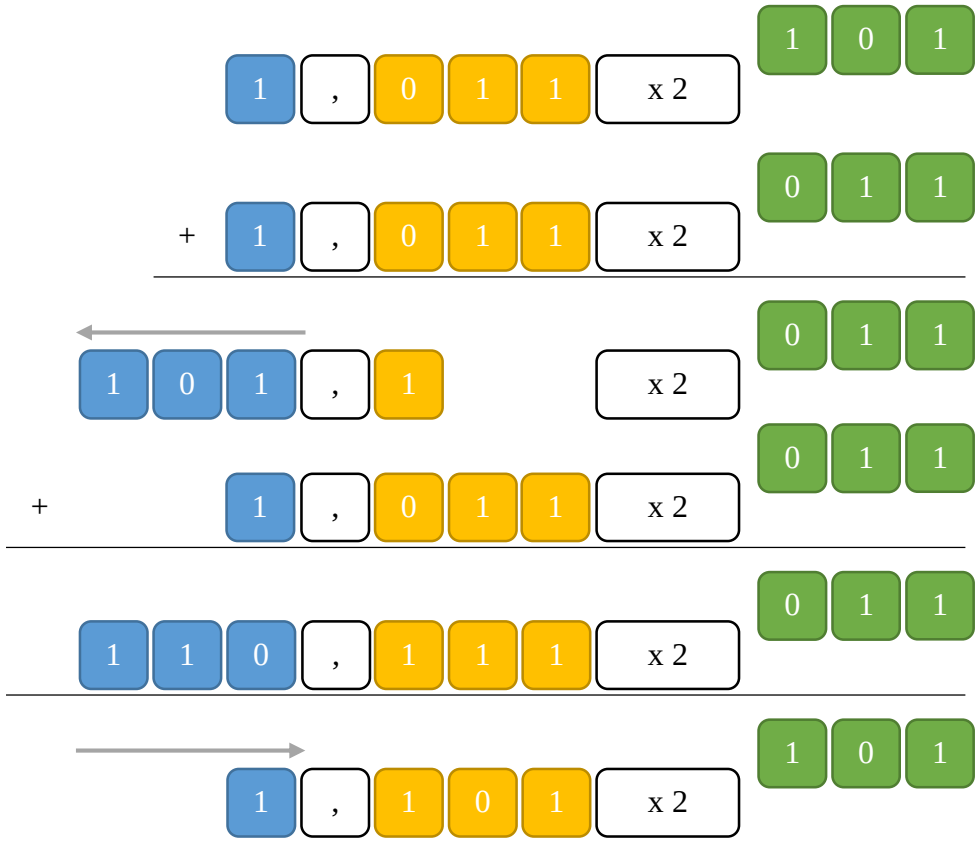

Figure 63 : Réalisation d'une addition avec des nombres à virgule flottante.

Pour effectuer l'addition en virgule flottante, comme sur la Figure 63, on adopte une représentation des deux nombres telle que les deux exposants soient égaux et, comme les deux nombres ont alors le même format de type virgule fixe, on réalise simplement l'addition de ces deux nombres. Pour obtenir le résultat final, il suffit de convertir le résultat obtenu en virgule flottante et en lui ajoutant l'exposant commun des deux premiers nombres (après modification de leur représentation).

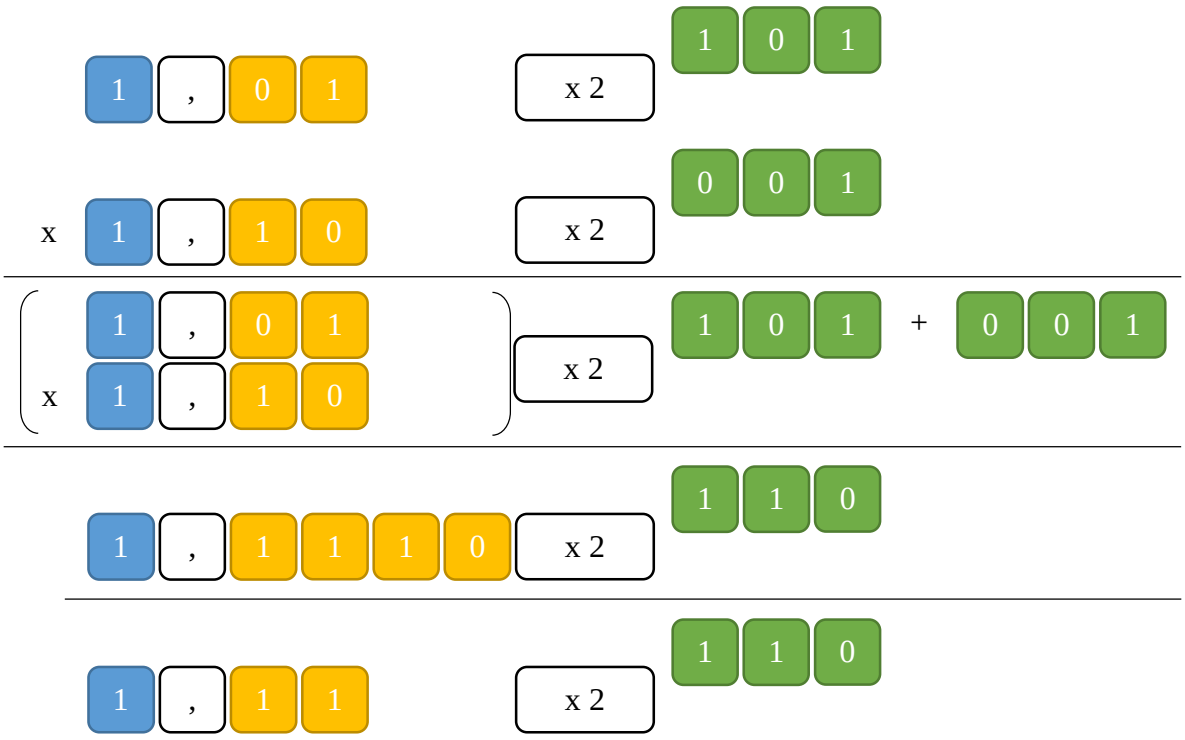

Figure 64 : Réalisation d'une addition avec des nombres à virgule flottante.

Pour la multiplication (cf. Figure 64), il suffit de multiplier les deux mantisses entre elles, puis d'ajouter les exposants entre eux et enfin de convertir le résultat en nombre à virgule flottante, si besoin est en réalisant un arrondi.

Annexe 3 : Filtre passe bas du premier ordre

L'expression de la fonction de transfert H_{LP} d'un filtre passe bas du premier ordre est :

$$H_{LP}(z) = \frac{1-a}{1-a\cdot z^{-1}}.$$

En substituant à z l'expression $e^{j\frac{2\cdot\pi\cdot f}{F_s}}$, on peut calculer $|H_{LP}(f)|^2$:

$$|H_{LP}(f)|^2 = \left|\frac{1-a}{1-a\cdot e^{-j\frac{2\cdot\pi\cdot f}{F_s}}}\right|^2,$$

$$|H_{LP}(f)|^2 = \frac{(1-a)^2}{\left(1-a\cdot\cos\left(\frac{2\cdot\pi\cdot f}{F_s}\right)\right)^2+\left(a\cdot\sin\left(\frac{2\cdot\pi\cdot f}{F_s}\right)\right)^2}.$$

Si f_0 est la fréquence de coupure à -3 dB, alors on a $|H_{LP}(f_0)|^2 = \frac{1}{2}$, d'où :

$$\frac{(1-a)^2}{\left(1-a\cdot\cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)\right)^2+\left(a\cdot\sin\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)\right)^2} = \frac{1}{2},$$

$$\left(1-a\cdot\cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)\right)^2+\left(a\cdot\sin\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)\right)^2 = 2\cdot(1-a)^2,$$

$$1-2\cdot a\cdot\cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)+a^2\cdot\left(\cos^2\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)+\sin^2\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)\right) = 2-4\cdot a+2\cdot a^2.$$

On sait que $\cos^2(x)+\sin^2(x)=1$, ce qui amène à la résolution de l'équation du second degré suivante :

$$a^2+2\cdot a\cdot\left(\cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)-2\right)+1=0.$$

Le calcul du discriminant Δ conduit à :

$$\Delta = \left(2\cdot\left(\cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)-2\right)\right)^2-4,$$

$$\Delta = 2^2\cdot\left[\left(\cos\left(\frac{2\cdot\pi\cdot f_0}{F_s}\right)-2\right)^2-1\right].$$

On vérifie bien que le discriminant est positif pour $0 < f_0 < \frac{F_s}{2}$:

$$-1 < \cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) < 1,$$

$$-3 < \cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - 2 < -1.$$

Comme $\cos\left(\frac{2 \cdot \pi \cdot f}{F_s}\right) - 2$ est négatif, il ne faut pas oublier de changer le sens des inégalités lors de l'élévation au carré :

$$9 > \left(\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - 2\right)^2 > 1,$$

$$8 > \left(\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - 2\right)^2 - 1 > 0,$$

$$32 > 2^2 \cdot \left[\left(\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - 2\right)^2 - 1\right] > 0.$$

L'équation du second degré a donc deux solutions réelles distinctes a_1 et a_2 :

$$a_1 = \frac{-2 \cdot \left(\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - 2\right) - \sqrt{2^2 \cdot \left[\left(\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - 2\right)^2 - 1\right]}}{2},$$

$$a_1 = -\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) + 2 - \sqrt{\left(\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - 2\right)^2 - 1}\,.$$

De la même manière on obtient :

$$a_2 = -\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) + 2 + \sqrt{\left(\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - 2\right)^2 - 1}.$$

D'après les critères de stabilité de la transformée en z, il faut absolument que le coefficient a soit compris entre -1 et 1 pour assurer la stabilité du filtre. La Figure 65 représente les valeurs que prennent a_1 et de a_2 en fonction de la fréquence normalisée $\frac{f_0}{F_s}$.

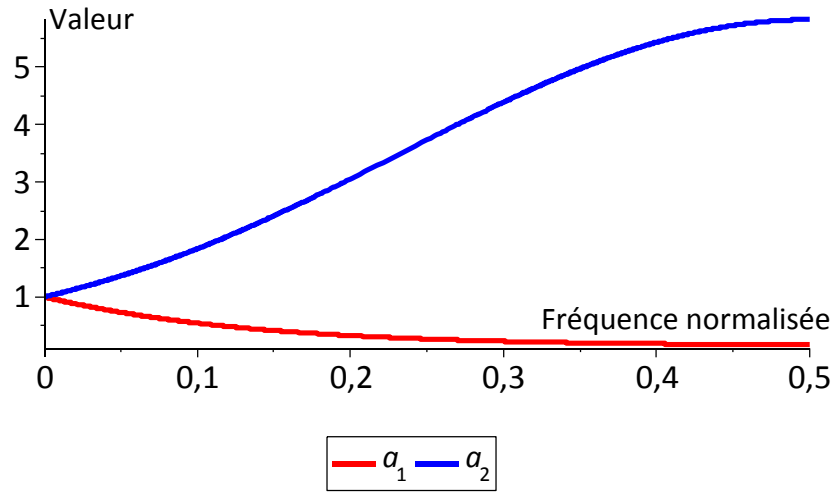

Figure 65 : Tracé de a_1 et de a_2 en fonction de la fréquence normalisée.

Puisque le coefficient doit-être compris entre -1 et 1, pour calculer la valeur de a en fonction de la fréquence de coupure f_0 et de la fréquence d'échantillonnage F_s, il faut utiliser l'expression suivante :

$$a = a_1 = 2 - \cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - \sqrt{\left(\cos\left(\frac{2 \cdot \pi \cdot f_0}{F_s}\right) - 2\right)^2 - 1}.$$

Annexe 4 : Introduction à System Generator for DSP

1. Bloc Delay

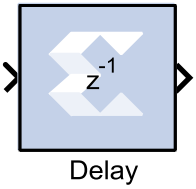

Figure 66 : Bloc réalisant la fonction de délai.

Ce bloc permet de retarder le signal d'un nombre d'échantillons correspondant à l'opposé du nombre indiqué en exposant du terme z. Dans le cas précis de la Figure 66, on réalise un retard d'un échantillon. On utilise ce bloc essentiellement pour garder deux informations synchrones, comme par exemple le signal d'horloge avec sa donnée ; en effet, les blocs suivants introduisent un retard lié au temps de calcul et ce retard est aussi indiqué par l'exposant du terme z.

2. Bloc Mux

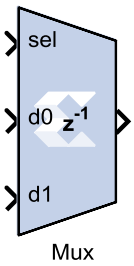

Figure 67 : Bloc réalisant la fonction de multiplexage.

Le multiplexeur (Mux) a pour fonction de faire transiter vers la sortie un signal parmi ceux présents en entrée (d0 et d1, data0 et data1) ; le choix du signal à aiguiller vers la sortie s'effectue au moyen de signaux de sélection (sel, selection). Dans le cadre de ce mémoire, on

utilisera ce bloc pour convertir le bit de la modulation sigma-delta, pouvant prendre la valeur 0 ou 1, en nombre numérique pour réaliser les calculs, pouvant prendre, généralement, la valeur -1 ou 1.

3. Bloc Accumulator

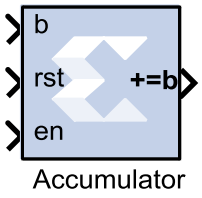

Figure 68 : Bloc réalisant la fonction d'accumulation.

L'accumulateur permet d'additionner à la valeur de sortie actuelle la valeur de l'entrée b. L'entrée en (enable) permet de dire à l'accumulateur quand il doit réaliser l'addition. L'entrée rst (reset) permet de remettre à 0 l'accumulateur.

4. Bloc AddSub

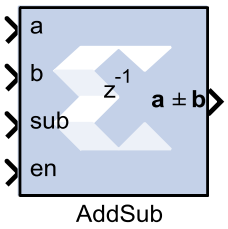

Figure 69 : Bloc réalisant les opérations d'addition ou de soustraction.

L'opérateur AddSub permet de réaliser au choix une opération d'addition ou de soustraction entre les entrées a et b. L'entrée sub (substract) permet de modifier l'opération exécutée même après avoir programmé le système, et si elle n'est pas présente, cela signifie que l'opération est fixée à l'avance et qu'on ne peut pas la modifier après la programmation. On se servira de cette entrée pour réaliser l'opération de multiplexage sur l'entrée b lors de la rétroaction sur les modulateurs sigma-delta pour convertir le bit en la valeur numérique b

ou -b. L'entrée en (enable) permet de dire si on doit effectuer cette opération ou si on doit mémoriser la valeur en sortie.

5. Blocs Mult et CMult

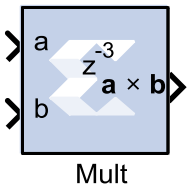

Figure 70 : Blocs réalisant l'opération de multiplication.

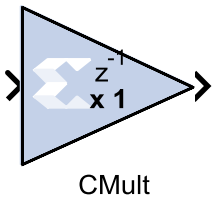

Figure 71 : Blocs réalisant l'opération de multiplication par une constante.

Le bloc Mult (Multiply) permet d'effectuer l'opération de multiplication de deux nombres a et b. Le bloc CMult a pour fonction de multiplier le nombre d'entrée par une constante (égal à 1 sur la Figure 71) ; l'intérêt de ce bloc est d'optimiser l'espace matériel utilisé par l'opération de multiplication.

6. Blocs Negate et convert

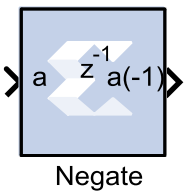

Figure 72 : Blocs réalisant l'opération de négation.

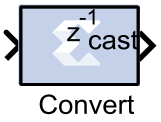

Figure 73 : Blocs réalisant la fonction de changement de représentation.

Le bloc Negate permet de renvoyer l'opposé du nombre a présent en entrée. Le bloc Convert permet de passer d'un mode représentation (binaire) des nombres à un autre (en virgule fixe ou en virgule flottante selon les besoins imposés par les dispositifs qui le suivent) ; il servira dans l'étage de sortie du convertisseur sigma-delta vers PCM.

7. Bloc Register

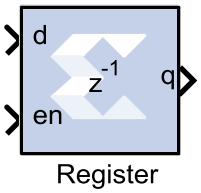

Figure 74 : Bloc réalisant la fonction de mémorisation.

Ce bloc permet de mémoriser l'information présente sur l'entrée d (data) lorsque l'entrée en (enable) est activée.

8. Blocs Relational et Constant

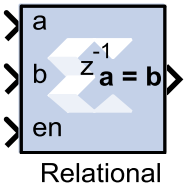

Figure 75 : Blocs réalisant l'opération de comparaison.

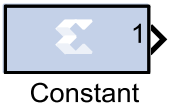

Figure 76 : Blocs réalisant la fonction de génération d'une constante.

Le bloc Relational permet de renvoyer la comparaison de deux nombres a et b. Le bloc Constant génère une valeur constante susceptible d'être exploitée à travers les blocs qui suivent. Concrètement, on utilisera l'association de ces deux blocs pour effectuer la comparaison de la valeur de l'accumulateur à 0 afin de réaliser la quantification du signal dans les modulateurs sigma-delta.

9. Blocs FDATool et FIR Compiler

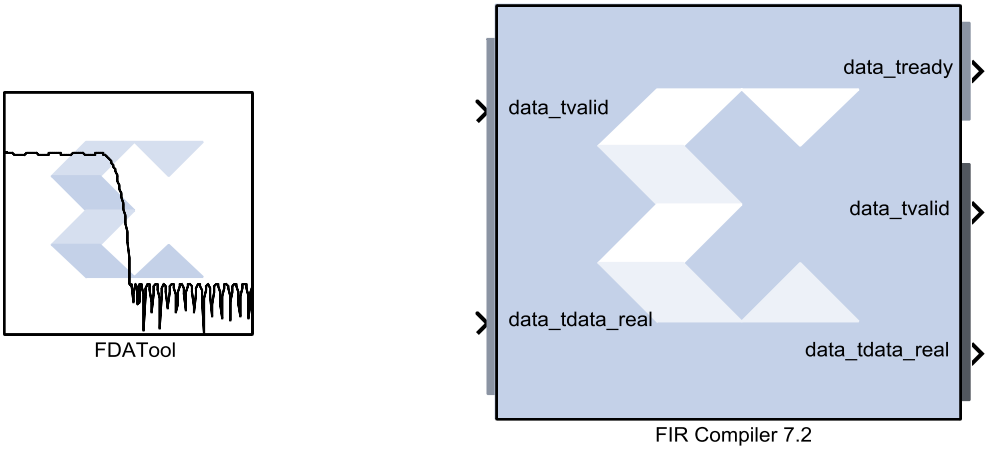

Figure 77 : Ensemble de blocs réalisant la fonction de filtrage par un filtre FIR.

Ces deux blocs sont généralement associés. FDATool est un utilitaire permettant de générer des filtres à réponse impulsionnelle finie appelés FIR. Le bloc FIR Compiler permet d'effectuer les calculs correspondant. L'entrée data_tvalid permet de signaler au bloc que les données présentes sur son autre entrée data_tdata_real sont valides. Pour signifier qu'il est à nouveau prêt à recevoir des données nouvelles en entrée, le bloc envoie vers sa sortie data_tready un signal. Etant donné qu'on traite de l'information audio en temps réel, on n'utilise pas ce signal, et on doit réaliser des tests pour vérifier qu'il est bien capable de traiter les échantillons d'entrée au rythme imposé. Les deux sorties data_tvalid et data_tdata_real ont, quant à elles, des fonctions identiques aux deux entrées, si ce n'est qu'elles véhiculent le signal traité.