

# Filière Systèmes industriels

Orientation Infotronics

# Travail de Bachelor Diplôme 2023

Cesalli Maxime

Audiophile 2-way splitter for active loudspeakers

-  *Professeur*  
François Corthay
-  *Expert*  
Dario Biner
-  *Date de la remise du rapport*  
18.08.2023



## Travail de diplôme | édition 2023 |

Filière  
*Systèmes industriels*

Domaine d'application  
*Infotronics*

Professeur responsable  
*François Corthay*  
*francois.corthay@hevs.ch*

## Audiophile 2-way splitter for active loudspeakers

 Diplômant/e      Maxime Cesalli

### Objectif du projet

L'objectif de ce projet réside dans la conception d'un diviseur à deux voies destinées aux haut-parleurs actifs. Cette réalisation s'appuiera sur l'utilisation de filtres FIR à phase linéaire.

### Méthodes | Expériences | Résultats

Ce diviseur à deux voix, également appelé crossover, sera intégré dans une FPGA. Celle-ci recevra les données audio au format I2S, les décodera puis les filtrera à l'aide de deux filtres FIR à phase linéaire : un passe-bas pour les basses et les médiums, et un passe-haut pour les aigus. Ensuite, les données seront retransmises au format I2S. Ainsi, les données seront converties en un signal analogique, amplifié et envoyé vers un haut-parleur. Chaque transducteur du haut-parleur recevra ainsi des signaux filtrés, offrant à la fois une précision spectrale et temporelle, ce qui contribuera considérablement à une restitution sonore d'une fidélité exceptionnelle.

De plus, ce système est entièrement configurable grâce à une interface graphique conviviale. Les utilisateur·rice·s ont la possibilité de modifier facilement le type de filtre en fonction de leurs besoins. Cela permet une personnalisation aisée du traitement audio en fonction des préférences de l'utilisateur·rice.

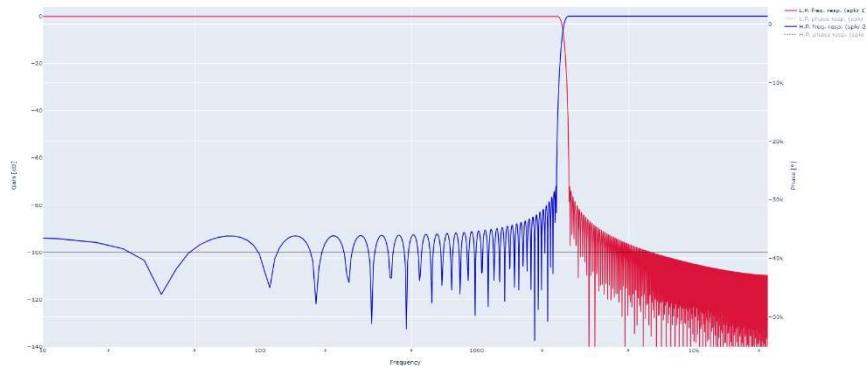


Diagramme de Bode d'un crossover FIR à phase linéaire 600 taps

## Table des matières

<b>1</b>	<b>INTRODUCTION .....</b>	<b>6</b>
<b>2</b>	<b>BASES THÉORIQUES .....</b>	<b>7</b>
2.1	L'IMPORTANCE D'UN BON CROSSOVER .....	7
2.2	POURQUOI UNE MAUVAISE SÉPARATION DES FRÉQUENCES EST-ELLE UN PROBLÈME ? .....	7
2.3	POURQUOI UNE PHASE NON LINÉAIRE EST-ELLE UN PROBLÈME ? .....	8
2.4	QU'EST-CE QU'UN FILTRE FIR (FINITE IMPULSE RESPONSE)?.....	9
2.5	QU'EST-CE QU'UNE FIELD-PROGRAMMABLE GATE ARRAY (FPGA)?.....	10
2.6	LES AVANTAGES D'UNE FPGA DANS LE TRAITEMENT DU SIGNAL.....	10
<b>3</b>	<b>ÉTAT DE L'ART.....</b>	<b>11</b>
<b>4</b>	<b>VUE D'ENSEMBLE DU PROJET .....</b>	<b>12</b>
4.1	CAHIER DES CHARGES.....	12
4.2	SCHÉMA BLOC DU SYSTÈME .....	12
4.3	SCHÉMA BLOC ARCHITECTURE FPGA .....	12
<b>5</b>	<b>ANALYSE ET DÉVELOPPEMENT I2S .....</b>	<b>13</b>
5.1	PROTOCOLE I2S .....	13
5.2	DÉCODEUR I2S .....	14
5.3	ENCODEUR I2S .....	15
<b>6</b>	<b>FILTRES FIR.....</b>	<b>16</b>
6.1	LES COEFFICIENTS.....	16
6.2	ARCHITECTURE PARALLÈLE.....	17
6.2.1	<i>Description :</i> .....	17
6.2.2	<i>Analyse des performances :</i> .....	18
6.3	ARCHITECTURE PARALLÈLE SYMÉTRIQUE .....	19
6.3.1	<i>Description.</i> .....	19
6.3.2	<i>Analyse des performances :</i> .....	19
6.4	DE FILTRE SIMPLE À CROSSOVER .....	21
6.4.1	<i>Problématique</i> .....	21
6.4.2	<i>Passe-haut soustrait au signal original</i> .....	21
6.4.3	<i>Partager le même registre à décalage</i> .....	21
6.5	ARCHITECTURE SÉRIE .....	22
6.5.1	<i>Description.</i> .....	22
6.5.2	<i>Analyse des Performances.</i> .....	23
6.6	ARCHITECTURE SÉRIE-PARALLÈLE .....	24
6.7	UTILISATION DE LA RAM POUR LE REGISTRE A DÉCALAGE .....	24
6.7.1	<i>Description.</i> .....	24
6.7.2	<i>Qu'est-ce qu'une RAM ?</i> .....	24
6.7.3	<i>Comment utiliser une RAM dans ce projet ?</i> .....	25
6.7.4	<i>Comment utiliser une RAM comme registre a décalage ?</i> .....	25
6.7.5	<i>Architecture Complète.</i> .....	27
6.7.6	<i>Analyse des performances.</i> .....	29
6.8	SYNTHESE DES RÉSULTATS .....	30
<b>7</b>	<b>IMPORTATION DE NOUVEAUX COEFFICIENTS DANS LA FPGA .....</b>	<b>31</b>
7.1	COMMUNICATION ORDINATEUR FPGA .....	31
7.2	PROTOCOLE RS-232 .....	31

7.3	DÉCODAGE RS-232 .....	32
7.4	LOGIQUE DE DÉCODAGE.....	33
7.5	ÉCRITURE DES COEFFICIENTS DANS LA RAM.....	34
7.6	ARCHITECTURE DU FILTRE UTILISANT LES COEFFICIENTS DANS LA RAM.....	36
7.7	SCRIPT PYTHON .....	38
7.7.1	<i>Marche à suivre</i> .....	38
7.7.2	<i>Importation permanente de nouveaux filtres dans la FPGA</i> .....	39
7.8	ANALYSE DES RÉSULTATS.....	41
<b>8</b>	<b>CONCLUSION .....</b>	<b>43</b>
8.1	CONCLUSION .....	43
8.2	AMÉLIORATIONS FUTURES.....	43
<b>9</b>	<b>DURABILITÉ.....</b>	<b>44</b>
<b>10</b>	<b>TABLE DES ILLUSTRATIONS .....</b>	<b>45</b>
<b>11</b>	<b>BIBLIOGRAPHIE.....</b>	<b>47</b>
<b>12</b>	<b>ANNEXES.....</b>	<b>48</b>
12.1	SCHÉMA FIR AVEC RAM UNIQUEMENT ÉCHANTILLONS .....	48
12.2	SCHÉMA FIR AVEC RAM ÉCHANTILLONS & COEFFICIENTS .....	49
12.3	LIEN DU REPOSITORY GITHUB .....	49
12.4	MESURES RÉELLES SUR LES HAUT-PARLEURS .....	50

Filière / Studiengang <b>SYND</b>	Année académique / Studienjahr <b>2022-23</b>	No TB / Nr. BA <b>IT/2023/82</b>
Mandant / Auftraggeber <input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input type="checkbox"/> Etablissement partenaire <i>Partnerinstitution</i>	Etudiant / Student <b>Maxime Cesalli</b>	Lieu d'exécution / Ausführungsort <input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input type="checkbox"/> Etablissement partenaire <i>Partnerinstitution</i>
Travail confidentiel / vertrauliche Arbeit <input type="checkbox"/> oui / ja <input checked="" type="checkbox"/> non / nein	Expert / Experte (données complètes) <b>Dario Biner</b> Stenheim, Route du Rhône 10, 1963 Vétroz	

## Titre / Titel

**Audiophile 2-way splitter for active loudspeakers**

## Description / Beschreibung

Most loudspeakers are passive. The amplified audio signal is brought to all speakers, yet matched to each of their frequency range with the help of analog LC filters : lowpass for the bass, highpass for the tweeter and possibly bandpass for the mid-range speaker. For active loudspeakers, the line-level audio signal is limited to the bands associated to each of the speakers and amplified separately. This removes the distortions induced by the analog filters.

The aim of this project is to design a 2-way audio splitter (lowpass and complementary highpass) based on FIR-filters. The circuit is to be implemented inside an FPGA, with an I2S digital audio input and two I2S outputs. The system is first to be implemented with fixed filter coefficients and later with the possibility to download the coefficients into the FPGA.

## Objectifs / Ziele

- Prepare a VHDL design environment
- Implement an I2S decoder and encoder, test them on the bass channel of the loudspeaker
- Implement a digital lowpass filter, insert it between decoder and encoder and test it
- Implement the complementary highpass filter and then both on the loudspeaker
- Propose a method to download the filter coefficients from a PC into the system

## Signature ou visa / Unterschrift oder Visum

Responsable de l'orientation /  
*Leiter der Vertiefungsrichtung:*

<sup>1</sup> Etudiant / Student:

## Délais / Termine

Attribution du thème / Ausgabe des Auftrags:  
**15.05.2023**

Présentation intermédiaire / Zwischenpräsentation:  
**19 - 20.06.2023**

Remise du rapport final / Abgabe des  
*Schlussberichts:*  
**18.08.2023, 12:00**

Expositions / Ausstellungen der Diplomarbeiten:  
**25.08.2023 – HEI**  
**28.08.2023 – Monthey**  
**31.08.2023 – Visp**

Défense orale / Mündliche Verfechtung:  
**Semaine/Woche 36 (04-07.09.2023)**

<sup>1</sup> Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.

Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.



---

## 1 Introduction

---

La plupart des haut-parleurs du marché sont passifs, cela signifie qu'un signal audio amplifié leur arrive directement puis est ensuite divisé, en fonction des fréquences correspondantes aux transducteurs émettant le son, par des filtres LC analogiques : un filtre passe-bas pour les basses et un filtre passe-haut pour les aigus, cela s'appelle un Crossover. Les haut-parleurs actifs eux reçoivent individuellement chaque bande de fréquence préalablement filtrée puis amplifiée. Le fait de filtrer le signal avant l'amplification rend possible l'utilisation de filtres plus performants et complexes. Cela permet aussi de se débarrasser des distorsions produites par un filtre analogique. L'objectif de ce projet est de concevoir un diviseur audio à 2 voies (passe-bas et passe-haut) basé sur des filtres FIR (Finite Impulse Response) à phase linéaire. Le circuit doit être implémenté à l'intérieur d'une Field-Programmable Gate Array (FPGA), avec une entrée/sortie audio numérique I2S. Le système va d'abord être implémenté avec des coefficients de filtre fixes, puis avec la possibilité de les télécharger dans la FPGA depuis un ordinateur.

Ce document constitue une synthèse exhaustive des recherches entreprises et des résultats obtenus tout au long du projet. Il débutera par une exposition des fondements théoriques nécessaires, suivie d'une vue d'ensemble globale du projet. Une analyse du protocole I2S sera ensuite présentée, suivie d'un examen détaillé des différentes architectures de filtres développées, mettant en évidence leurs avantages et performances respectifs. Enfin, le processus d'importation de nouveaux coefficients dans le système sera expliqué en détail.

## 2 Bases Théoriques

### 2.1 L'importance d'un bon crossover

Comme cité précédemment la majorité des haut-parleurs du marché fonctionnent avec un crossover analogique. Cette approche implique une mauvaise séparation des fréquences et des déphasages non linéaires qui perturbe la propreté du signal.

### 2.2 Pourquoi une mauvaise séparation des fréquences est-elle un problème ?

La mauvaise séparation des fréquences peut entraîner une situation où les deux haut-parleurs (basses et aigus) produiront simultanément le même son. Deux sources émettent un signal identique à partir de différents points peut causer des interférences en fonction de la position de l'auditeur·ice dans l'espace. Ces interférences peuvent être de deux types : constructives, dans le cas où les deux signaux se superposent et génèrent un signal avec une amplitude deux fois plus importante que celle prévue, ou destructives, dans le cas où les deux ondes sonores sont déphasées de 180 degrés et annulent le signal.

Cela implique qu'en fonction de la position de l'auditeur·ice dans la pièce, l'amplitude sonore de certaines fréquences peut varier, ce qui perturbe l'expérience d'écoute musicale. Une séparation plus précise des fréquences réduit la plage dans laquelle de telles interférences peuvent se produire. La plupart des crossovers disponibles sur le marché présentent une atténuation variant entre 6 dB/octave (40 dB/décade) et 48 dB/octave (160 dB/décade). Cependant, le crossover développé dans ce projet possède une atténuation de 315 dB/octave, équivalant à 1050 dB/décade.

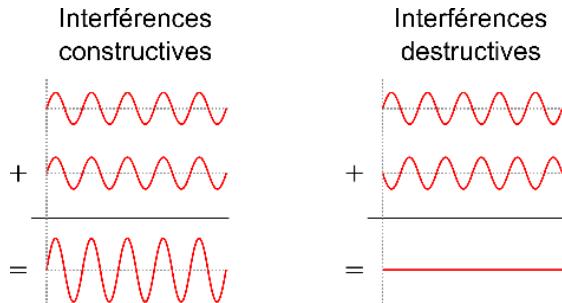


Figure 1: exemple d'interférence [1]

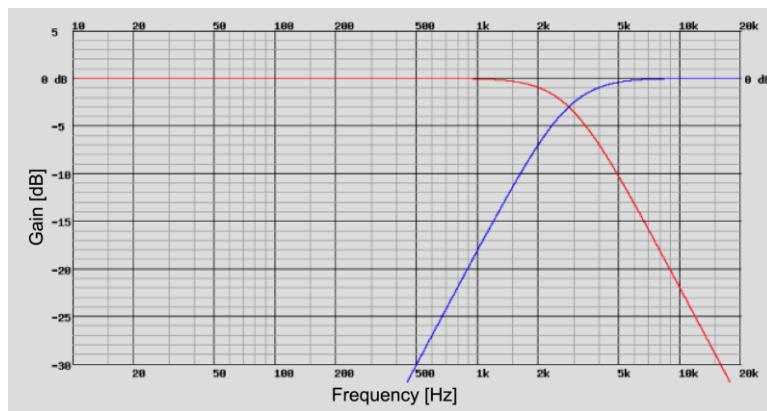


Figure 2 : Crossover analogique ordre 2 [2]

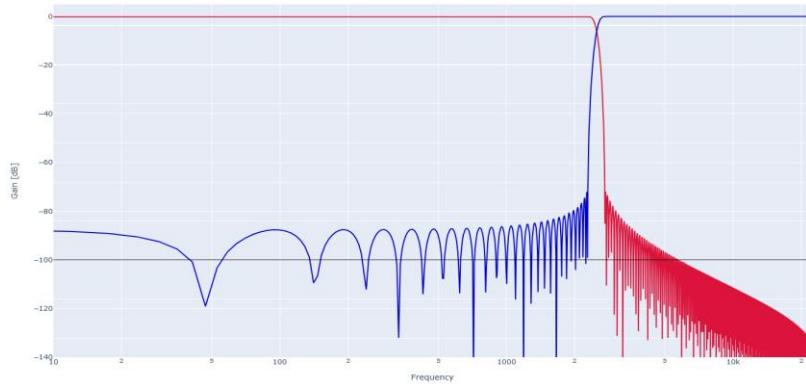


Figure 3: Crossover FIR 500 taps

## 2.3 Pourquoi une phase non linéaire est-elle un problème ?

Chaque filtre engendre un déphasage entre le signal à son entrée et celui à sa sortie, induisant ainsi un retard temporel du signal de sortie par rapport à celui d'entrée. Néanmoins, la durée du retard induit par les filtres n'est pas toujours uniforme. Par exemple, dans un filtre IIR(Infinit Impulse Response), le décalage temporel ne varie que très peu jusqu'à ce que la fréquence de coupure soit atteinte. À ce point, le décalage varie rapidement, puis au-delà de la bande passante la phase n'a plus d'importance. Cette situation crée donc un décalage inégal du signal. En revanche, un filtre à phase linéaire maintient un délai constant indépendamment de la fréquence de sa bande passante. Cette constance du délai temporel facilite grandement le réglage du système, ainsi que la fidélité du son.

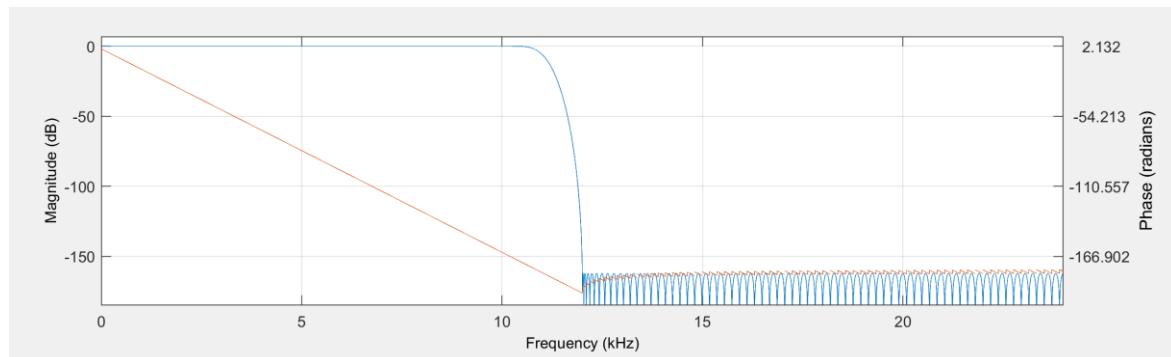


Figure 4 : Filtre FIR à phase linéaire

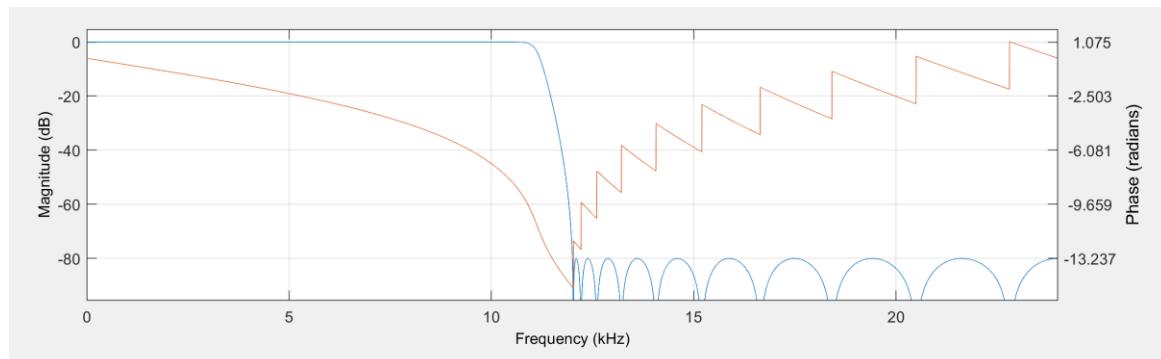


Figure 5 : filtre IIR à phase non linéaire

## 2.4 Qu'est-ce qu'un filtre FIR (Finite Impulse Response)?

Dans le domaine du traitement du signal, un filtre FIR est un élément fondamental. Son rôle est d'altérer un nombre fini d'échantillons d'entrée pour générer un échantillon de sortie. Pour ce faire, un registre à décalage enregistre un certain nombre d'échantillons (Taps), chaque échantillon est multiplié par un coefficient spécifique, les résultats sont ensuite sommés pour générer la sortie finale.

Les coefficients du filtre FIR peuvent être réglés pour obtenir diverses opérations de filtrage, telles que la réduction du bruit, la séparation de fréquences ou la mise en évidence de certaines caractéristiques spectrales. Un filtre FIR est un élément polyvalent et puissant, ce qui est adéquat dans le cadre de la conception d'un Crossover.

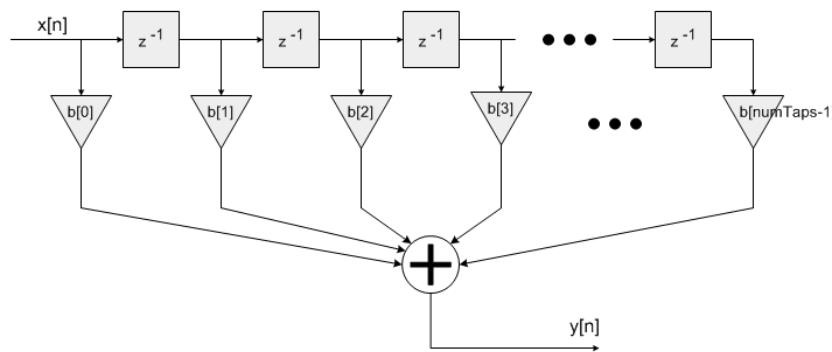


Figure 6 : schéma de principe d'un filtre FIR [3]

La propriété de phase linéaire est valide uniquement quand les coefficients du filtre sont symétrique. Cette symétrie implique que le coefficient initial ( $b[0]$ ) est égal au coefficient final ( $b[n]$ ), et de même, que le coefficient secondaire ( $b[1]$ ) équivaut au coefficient avant-dernier ( $b[n-1]$ ), et ainsi de suite. La disposition symétrique des coefficients garantit une réponse en phase linéaire, une caractéristique essentielle dans ce projet.

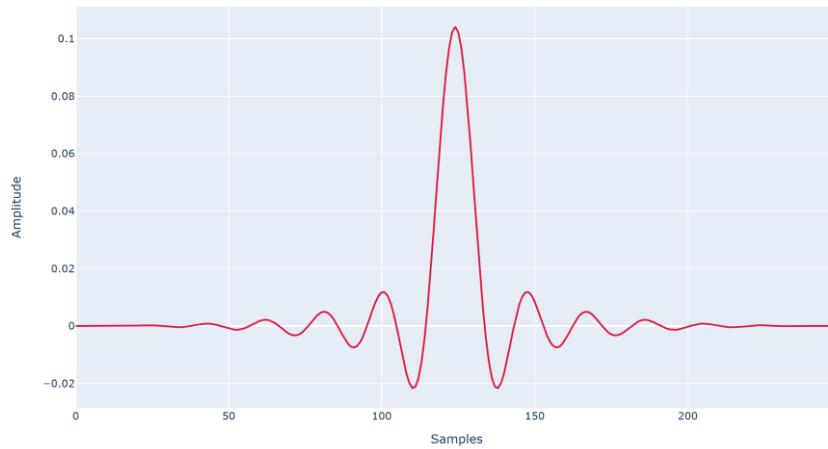


Figure 7 : réponse impulsionnelle d'un filtre FIR symétrique

Ce qui rend le filtre FIR particulièrement intéressant, c'est que sa réponse impulsionnelle correspond au graphique formé par ses coefficients. C'est pourquoi, dans la réponse impulsionnelle du filtre ci-dessus (Figure 7), on peut clairement observer la symétrie de ces coefficients.

## 2.5 Qu'est-ce qu'une Field-Programmable Gate Array (FPGA)?

Une FPGA est un circuit intégré programmable constitué d'un réseau de blocs logiques tels que des portes logiques, des bascules et des multiplexeurs, reliés entre eux par des chemins programmables. Cette structure permet de concevoir des circuits numériques personnalisés en programmant les blocs logiques et les connexions au moyen de langages de description matérielle tels que VHDL (Very high-speed integrated circuit Hardware Description Language) ou Verilog. Dans le cadre de ce projet, le langage de programmation utilisé est VHDL.

Une différence fondamentale entre une FPGA et un microcontrôleur réside dans la manière dont le code créé est exécuté. Dans un microcontrôleur, le code est exécuté séquentiellement, chaque ligne de code étant analysée puis exécutée de manière séquentielle. En revanche, dans une FPGA, le code créé est traduit en configuration matérielle, c'est-à-dire en câblage qui interconnecte les différents blocs logiques. Par conséquent, le code dans une FPGA ne s'exécute pas de manière séquentielle, mais plutôt en parallèle, exploitant ainsi efficacement les ressources disponibles pour des opérations simultanées et parallèles.

## 2.6 Les avantages d'une FPGA dans le traitement du signal

L'utilisation d'une FPGA présente l'avantage majeur de permettre d'effectuer un grand nombre d'opérations complexes en un temps constant et prévisible. Cette caractéristique permet un contrôle essentiel pour parvenir à un traitement efficace du signal. En le comparant à des systèmes plus conventionnels, tels qu'un Raspberry PI exécutant un logiciel de traitement du signal numérique (DSP), la différence est notable. Dans le cas du Raspberry PI, le système d'exploitation gère les ressources de calcul, le temps de traitement du signal n'est donc jamais constant. Cette variation rend le système difficile (voire impossible) à contrôler avec précision en termes de temps, ce qui peut compromettre l'efficacité du traitement du signal. En revanche, la FPGA assure une exécution constante et fiable des opérations, contribuant ainsi à garantir la qualité et la performance du traitement du signal.

### 3 État de l'art

Actuellement, de nombreux travaux de recherche se penchent sur le domaine des filtres à phase linéaire, explorant divers aspects de leur mise en œuvre.

Une étude intitulée "Linear phase mixed FIR/IIR crossover networks: design and real-time Implementation"[4] se consacre à examiner différentes approches d'implémentation. Elle étudie les filtres FIR (Finite Impulse Response) qui assurent une phase linéaire, mais peuvent engendrer des retards significatifs, ainsi que les filtres IIR (Infinite Impulse Response) qui peuvent approximer une phase linéaire tout en minimisant les retards. Cette étude envisage également une combinaison des deux types de filtres afin de tirer profit des avantages inhérents à chacun d'eux, ouvrant ainsi la voie à une meilleure optimisation des performances.

Un débat reste ouvert dans le milieu scientifique sur l'impact des distorsions de phases lors de l'écoute de signaux audio.

Dans un papier intitulé "The Audibility of Loudspeaker Phase Distortion" (AES E-Library), Mr Richard Greenfield, Dr Malcolm Hawksford [5] examinent l'audibilité de la distorsion de phase des haut-parleurs. Ils étudient comment la distorsion de phase introduite par les crossovers et les transducteurs peut affecter la perception sonore, en particulier la localisation spatiale et la clarté sonore. L'article met en évidence le lien entre la distorsion de phase et la perception des auditeurs·ices, soulignant que même de petites quantités de distorsion de phase peuvent altérer significativement la qualité sonore perçue.

Une thèse de doctorat intitulée "Aural Phase Distortion Detection"[6] analyse en profondeur les mécanismes de distorsion de phase dans les systèmes audio et leurs effets perceptibles. L'étude explore les aspects neurophysiologiques de la perception du signal sonore, y compris le phénomène de "phase-locking"[7], qui joue un rôle crucial dans la localisation spatiale des sons. Ce lien entre la phase des signaux audio et la perception auditive met en évidence l'importance d'une linéarité de phase dans les crossovers et les systèmes de haut-parleurs.

En conclusion, la recherche dans le domaine des crossovers à phase linéaire met en évidence l'importance cruciale de maintenir une linéarité de phase pour garantir une reproduction sonore précise et cohérente. Les avancées dans la compréhension des mécanismes perceptuels, tels que le "phase-locking", renforcent davantage l'importance de la linéarité de phase dans les crossovers et les systèmes audio pour une expérience d'écoute optimale.

## 4 Vue d'ensemble du projet

### 4.1 Cahier des charges

- Implémenter un encodeur/décodeur I2S dans la FPGA et le tester sur le canal des basses du haut-parleur
- Implémenter un filtre passe-bas digital utilisant différentes méthodes pour sa mise en œuvre
- Implémenter le filtre passe-haut complémentaire et faire le test avec les deux canaux
- Proposer une méthode pour importer les coefficients du filtre dans la FPGA

### 4.2 Schéma bloc du système

Le signal audio analogique d'entrée parvient par le biais de deux connecteurs RCA. Ensuite, un convertisseur analogique-numérique effectue un échantillonnage du signal à une fréquence de 96 kHz (48 kHz par canal), puis transmet les données à la FPGA via le protocole I2S. La FPGA vas ensuite séparer ce signal en fonction des fréquences qui le composent pour créer un signal I2S contenant un canal aigu et un canal grave. Les données vont ensuite être reconvertis en deux signaux analogiques qui pourront ensuite être amplifiés séparément.

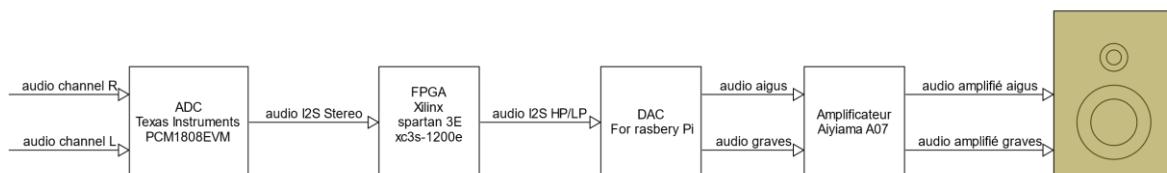


Figure 8: schéma bloc du système.

### 4.3 Schéma bloc architecture FPGA

Au sein de la FPGA le processus démarre par la désérialisation du signal par le bloc I2S décodeur. Par la suite un des deux canaux vas être sélectionné par un démultiplexeur. Cette opération est réalisée par un signal externe, il permet de configurer le signal pour le haut-parleur gauche ou droite sans devoir reprogrammer l'ensemble. La donnée sélectionnée passe ensuite dans les deux filtres. Les informations qui en sortent sont ensuite serialisées et envoyées à l'extérieur de la FPGA.

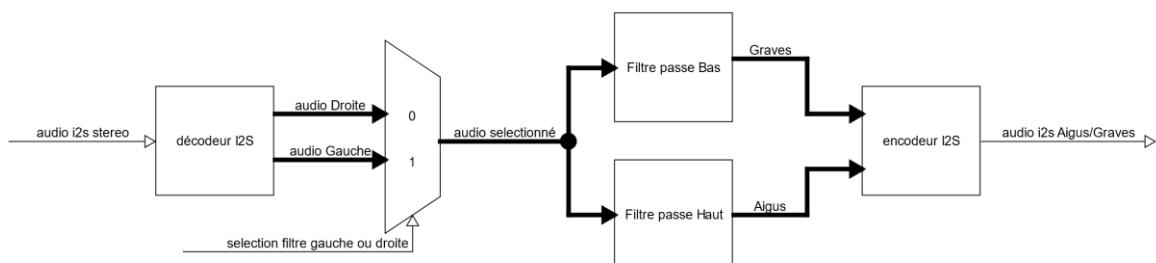


Figure 9 : Schéma bloc architecture FPGA

## 5 Analyse et Développement I2S

### 5.1 Protocole I2S

La première étape afin de pouvoir manipuler les signaux audio transmis est de comprendre le protocole qui les transporte. Le protocole I2S est une interface série synchrone utilisée principalement pour la transmission des données audio numériques. Il a été développé par Philips Semiconductor (maintenant NXP Semiconductors) et est largement adopté dans l'industrie audio. Le protocole I2S permet la transmission simultanée des données audio et des signaux de synchronisation, garantissant ainsi une reproduction précise du son. Le protocole I2S se compose de trois signaux : la ligne de données (Data line), la ligne de synchronisation des horloges (Bit Clock line) et la ligne de synchronisation des canaux (Word Select line).

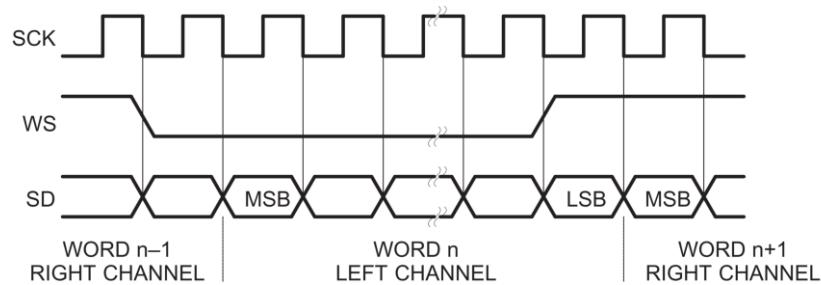


Figure 10: chronogramme d'un transfert de données I2S[8]

La Figure 10 illustre l'une des particularités du transfert I2S, à savoir que les données binaires ne changent pas en synchronisation avec le signal Word Select, mais plutôt décalées d'une période d'horloge. Cette particularité a généré un ensemble de défis durant le développement de ce projet.

## 5.2 Décodeur I2S

Le décodeur I2S parallélise les données sérielles qu'il reçoit. Pour ce faire un compteur se décrémente à chaque flanc montant du bit clock. La valeur de ce compteur est ensuite soumise à un processus de multiplexage, en fonction de l'état du signal Word Select, préalablement décalé d'une impulsion d'horloge pour se conformer aux spécifications de la norme I2S. La valeur obtenue par le compteur détermine la position d'enregistrement de la ligne de données. Pour illustrer, si la valeur du compteur est 24 et que le signal Word Select est à 0, alors la valeur de la ligne de données est extraite et sauvegardée en tant que 24<sup>e</sup> bit de la donnée correspondant au canal Droit.

Une fois que nos deux ensembles de données sont prêts, ce qui se produit au moment du flanc montant de notre signal Word Select décalé, une impulsion est générée pour signaler aux blocs suivants que de nouvelles données sont disponibles et peuvent être lues.

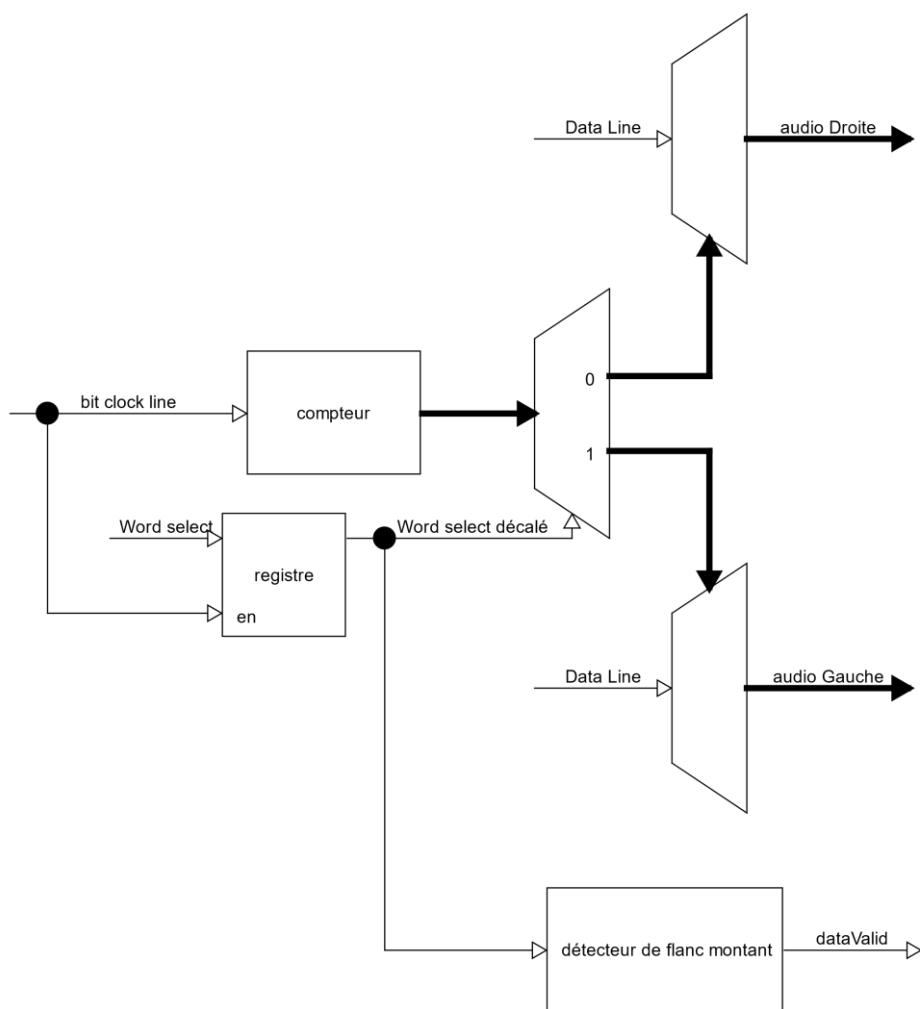


Figure 11: schéma interne du décodeur I2S

Pour finaliser la phase de déserialisation, il est impératif d'implémenter un registre supplémentaire. Ce registre agit comme un mécanisme de contrôle en autorisant la sortie des données uniquement lorsque le signal "dataValid" est en état logique élevé (1). Cette fonctionnalité est cruciale, car en dehors de ces instants particuliers, les données sont en cours de mise à jour et peuvent afficher des valeurs transitoires.

## 5.3 Encodeur I2S

L'encodeur I2S reçoit en entrée des données parallèles et les convertit en données sérielles. Il reçoit les informations de synchronisation depuis l'encodeur, cela permet de s'adapter à n'importe quelle fréquence de transmission, tant qu'elle ne s'approche pas trop de la fréquence du clock interne de la FPGA. Le principe de fonctionnement est l'exact inverse du décodeur. On y retrouve un décompteur qui va sélectivement parcourir chaque bit de notre donnée afin de le transmettre sur la data line. Le Word Select, lui, va se faire décaler à l'aide d'une bascule D puis, venir sélectionner laquelle des données, droite ou gauche ou, dans notre cas, aigus ou graves, vas se faire serialiser. La réinitialisation du compteur s'effectue de manière simultanée avec chaque transition d'état du signal Word Select décalé.

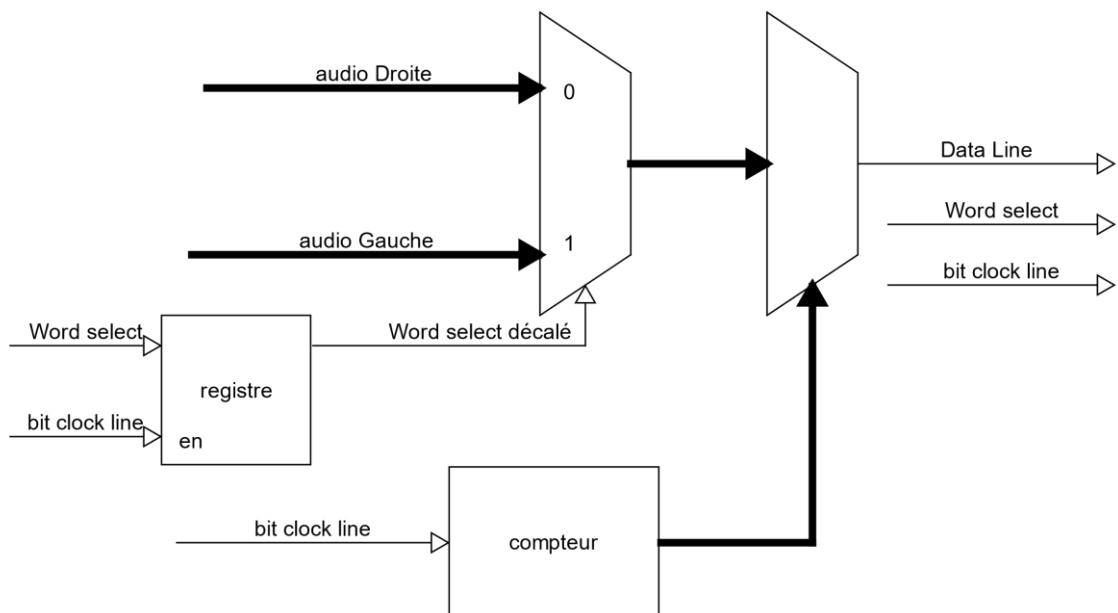


Figure 12: schéma interne de l'encodeur I2S

## 6 Filtres FIR

Dans la section qui suit, une analyse des différentes architectures des filtres implémentés durant le développement de ce projet va être effectuée. Cette analyse commencera par une explication de la manière dont sont obtenus les coefficients des filtres. Puis continuera en abordant les architectures de filtre, des plus simples aux architectures les plus élaborées. Pour chaque architecture, nous examinerons en détail leur performances, ainsi que leur avantages et inconvénients respectifs.

### 6.1 Les coefficients

Pour obtenir des filtres pertinents il est essentiel d'avoir des coefficients appropriés. Dans le cadre de ce projet, les coefficients sont issus de [LinFIRXover](#)[9], une plateforme en ligne qui permet de configurer des Crossovers pour jusqu'à 4 transducteurs. Tous les paramètres de configuration sont ajustables selon les besoins spécifiques du projet. Cette ressource a été mise à disposition par Demion Arnaud, Collaborateur scientifique HES, et elle se révèle cruciale pour garantir l'adéquation des coefficients et l'efficacité des filtres mis en place.

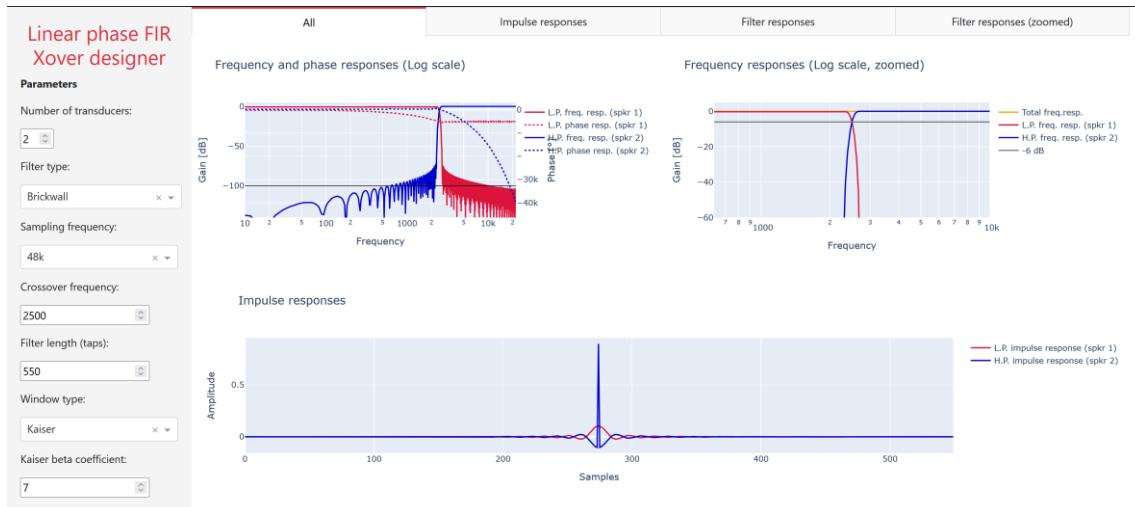


Figure 13: LinFIRXover home page

Cependant, il est important de prendre en considération que les coefficients d'un filtre FIR demeurent toujours inférieurs à 1. Cette particularité revêt une importance particulière dans le contexte d'une FPGA qui travaille avec des types de données restreints à des nombres entiers, tels que ceux manipulés par la bibliothèque "ieee-numeric-std" [10] en VHDL. En conséquence, il devient impératif d'effectuer une conversion vers des nombres entiers. Pour cela, une méthode simple basée sur une multiplication est appliquée. En supposant que les coefficients doivent être codés sur 32 bits, il faut donc les multiplier individuellement par  $2^{32}$ . Cependant, il est primordial de souligner que le résultat final du filtre doit être divisé par cette même valeur pour éviter des résultats excessivement grands.

Pour faire cette conversion un autre logiciel s'est avéré crucial pour la réalisation de ce projet. Il s'agit d'un script python[11] permettant de sélectionner directement les fichiers téléchargés depuis [LinFIRXover](#), et de les convertir en un tableau VHDL. De plus, il offre la flexibilité de choisir le nombre de bits pour la conversion ainsi que le type de données (Hexadécimal, Binaire, Octal). Cet outil a été développé et mis en open source par Dimitar Marinov, un contributeur du site vhdlwhiz.com. Son article[12] sur la création d'un filtre FIR sur FPGA as été d'une grande aide pour les premières étapes de ce projet.

## 6.2 Architecture parallèle

### 6.2.1 Description :

La première architecture explorée adopte une approche simplifiée, elle consiste à réaliser le modèle théorique du filtre FIR. En y ajoutant une division à la sortie de l'additionneur comme discuté dans le chapitre précédent. Cette architecture effectue toutes les opérations en parallèle. Le filtre tout en entier est résolu en une période d'horloge, ce qui implique que le nombre de multiplicateurs est égal au nombre de Taps (dans ce cas nombre de coefficients). Chaque multiplicateur génère en sortie une donnée binaire dont la taille en nombre de bits est la somme des deux entrées. À titre d'exemple, si les échantillons sont codés sur 32 bits et les coefficients sur 16 bits, la taille de la sortie sera de  $16 + 32 = 48$  bits.

Ensuite, il est nécessaire d'ajouter chaque résultat du multiplicateur. Cependant, la sortie de l'additionneur ne peut pas rester limitée à 48 bits, car l'objectif est de ne pas perdre d'informations. Par exemple, si le nombre de coefficients est de 8 et que chaque entrée de l'additionneur représente la valeur maximale encodable sur 48 bits, la valeur de sortie sera huit fois supérieure à ce qui pourrait être encodé en 48 bits. Afin de préserver ces informations, il est impératif d'augmenter la taille de la sortie de l'additionneur. Dans le cas où le nombre de taps est de 8, ce qui équivaut à  $2^3$ . Il est alors nécessaire d'ajouter 3 bits à la donnée de sortie, soit  $48 + 3 = 51$  bits. Ainsi, l'additionneur comptera un ensemble d'entrées de 48 bits et une sortie de 51 bits, permettant de conserver l'intégrité des données sans perte d'information.

Une étape finale demeure à franchir pour conclure le processus : la sortie de notre filtre doit contenir le même nombre de bits que son entrée. Il est donc nécessaire de conserver 32 bits parmi les 51 de notre additionneur. Théoriquement, il faudrait préserver les 32 bits de poids fort. Cependant, étant donné qu'il est peu probable que toutes les valeurs en entrée de l'additionneur soient maximales, plusieurs bits de poids forts deviennent redondants. Par conséquent, il est essentiel de sélectionner une plage spécifique pour conserver les informations nécessaires. Bien que cette sélection puisse être automatisée, cette option n'a pas été implémentée dans le cadre de ce projet. Par conséquent, la plage appropriée a été choisie manuellement après une série de tests.

Il est évident que le nombre de bits dans le système augmente rapidement. Dans le simple exemple exposé précédemment, le système comporte 8 multiplicateurs 48 bits et un additionneur 51bits. De tels opérateurs sont très gourmands en ressources, la complexité du filtre est donc très vite limitée par la taille de la FPGA.

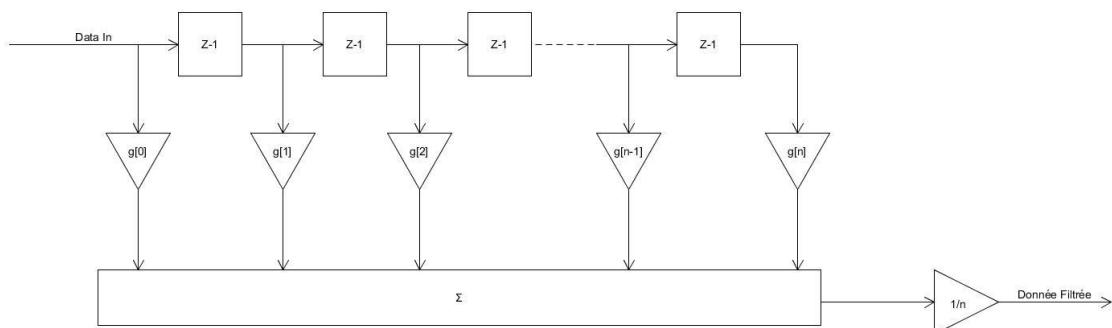


Figure 14: schéma de principe filtre FIR parallèle

### 6.2.2 Analyse des performances :

L'architecture Parallèle telle que présentée ci-dessus ne permet pas la création de filtres assez complexes pour satisfaire aux exigences de ce projet. Cependant la rapidité d'exécution de ce genre de processus peut être exploitée dans d'autres domaines.

Des tests de performances ont été effectués. Les résultats montrent qu'en utilisant une FPGA spartan 3E xc3s-500E, executant un seul filtre avec des coefficients 16 bits, la taille maximale atteinte est un filtre de 33 taps.

Parallèle	
33 taps	
<u>utilisation :</u>	
flip flops	13%
LUTs	71%
Slices	95%

Figure 15: utilisation ressources FPGA pour architecture Parallèle

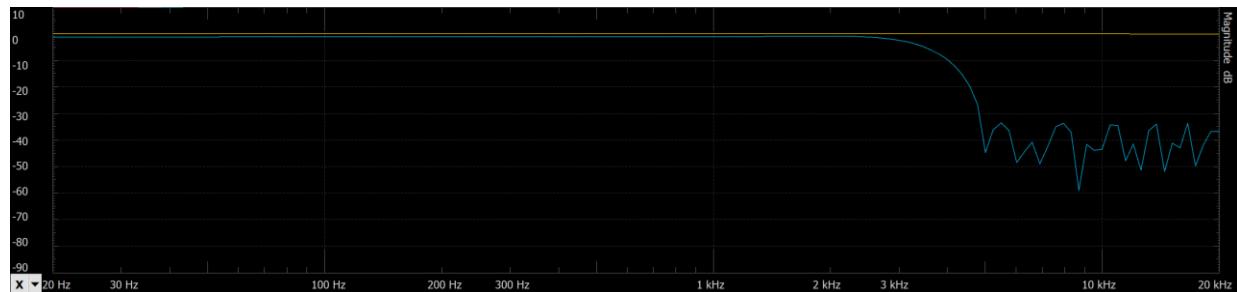


Figure 16 : diagramme de Bode filtre FIR 33taps

## 6.3 Architecture Parallèle symétrique

### 6.3.1 Description

Pour optimiser davantage l'architecture parallèle, il est possible de tirer parti de la nature symétrique des coefficients. En effet, pour garantir une phase linéaire dans notre filtre, les coefficients présentent une symétrie. Cela signifie qu'il existe deux occurrences de chaque coefficient, à l'exception du coefficient central qui est unique. Dans ce type de filtre, le premier et le dernier échantillon sont multipliés par la même valeur. Par conséquent, il est possible de sommer le premier et le dernier échantillon, puis de multiplier cette somme par le premier coefficient, étant donné qu'il est équivalent au dernier coefficient. Cette approche réduit le nombre de coefficients à enregistrer de moitié, ainsi que le nombre de multiplications, ce qui contribue à une optimisation significative de l'architecture.

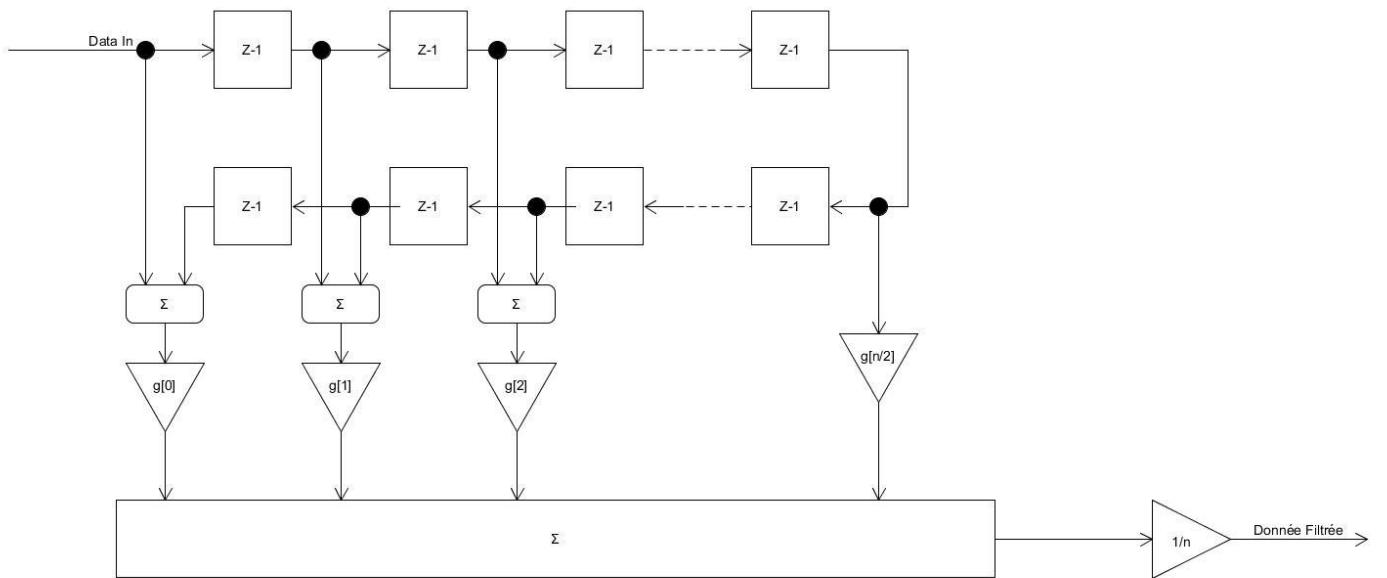


Figure 17: schéma de principe filtre FIR parallèle symétrique

### 6.3.2 Analyse des performances :

Cette considération implique que, pour un nombre de taps identique, les ressources de la FPGA seront sollicitées deux fois moins intensément. Cette hypothèse peut être vérifiée par le biais de mesures concrètes :

Parallèle avec symétrie	
33 taps	
<u>utilisation :</u>	
flip flops	13%
LUTs	33%
Sclices	49%

Figure 18 : utilisation ressources FPGA pour architecture Parallèle symétrique 33 taps

Comme le tableau ci-dessus (Figure 18) le montre, la consommation en termes de LUTs et de sclices a diminué de manière notable pour obtenir le même résultat. Cependant, il est important de noter que le nombre de flip-flops est demeuré constant. Cette situation s'explique par le fait que les flip-flops sont principalement associés au registre à décalage, dont l'utilisation est demeurée inchangée.

Cela implique également qu'il est possible de concevoir un filtre comportant un plus grand nombre de taps. En gardant les mêmes paramètres que pour l'architecture Parallèle simple, le nombre maximal de taps atteint est de 55.

Parallèle avec symétrie	
55 taps	
<u>utilisation :</u>	
flip flops	21%
LUTs	66%
Sclices	94%

Figure 19 : utilisation ressources FPGA pour architecture Parallèle symétrique 55 taps

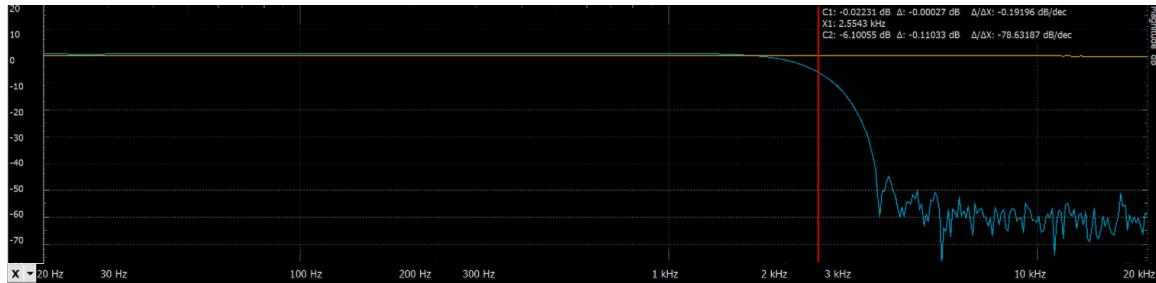


Figure 20 : diagramme de Bode filtre FIR 55taps

## 6.4 De filtre simple à Crossover

### 6.4.1 Problématique

La transition d'un filtre simple à un crossover consiste à créer deux filtres distincts : un filtre passe-haut et un filtre passe-bas, tous deux partageant la même fréquence de coupure. Cependant, cette méthode n'est pas optimale car elle conduit à la duplication inutile de registres à décalage, effectuant exactement la même tâche simultanément. Cela entraîne une utilisation inefficace des ressources de la FPGA. Dans les sections suivantes, deux approches alternatives seront présentées pour remédier à ce problème.

### 6.4.2 Passe-haut soustrait au signal original

La première approche alternative consiste à éviter la création d'un second filtre passe-haut et plutôt à soustraire la sortie du filtre passe-bas du signal d'origine. En effet, lorsque le filtre passe-bas permet le passage complet du signal, la soustraction de la sortie du passe-bas avec le signal d'origine résulte en une valeur nulle, éliminant ainsi les basses fréquences. À l'inverse, lorsque la sortie du filtre passe-bas est presque nulle dans les hautes fréquences, le signal d'origine demeure inchangé, laissant ainsi passer les hautes fréquences.

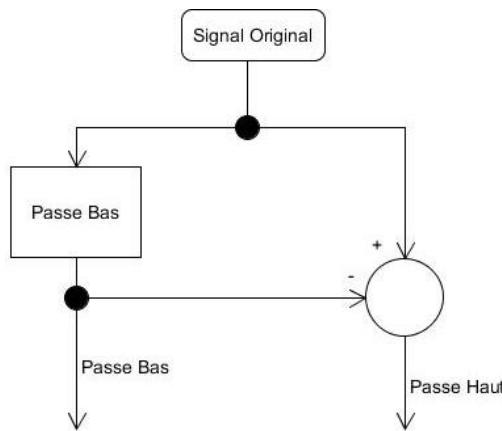


Figure 21 : schéma bloc de Passe-haut soustrait au signal original.

Cette approche se révèle très efficace en termes d'utilisation des ressources. Cependant, elle présente un inconvénient majeur : la réponse en fréquence du filtre passe-haut est exactement l'inverse de celle du filtre passe-bas. Ceci entraîne la perte d'un avantage considérable. Car lorsque les filtres sont indépendants, il est possible d'ajuster leurs coefficients pour adapter leur réponse en fréquence, et ainsi corriger les imperfections observées sur le matériel réel. Cette approche ne sera donc pas utilisée dans ce projet.

### 6.4.3 Partager le même registre à décalage

Une approche alternative consiste à regrouper les deux filtres au sein d'un même bloc partageant un registre à décalage commun. Chaque paire de registres subit ensuite deux multiplications, une pour le filtre passe-bas et l'autre pour le filtre passe-haut. Cette réorganisation élève le nombre total de multiplicateurs au niveau des taps du filtre. Bien que cette approche double le nombre de multiplicateurs, elle préserve le nombre de registres à décalage. De plus, elle préserve l'indépendance des filtres, garantissant ainsi leur fonctionnement distinct. C'est donc cette approche qui sera adoptée pour la suite du projet.

## 6.5 Architecture Série

### 6.5.1 Description

Afin de s'adapter aux les contraintes liées à l'utilisation des ressources, il est envisageable de développer une architecture qui exploite un unique multiplicateur réutilisé en boucle. Pour mettre en œuvre cette approche, les données du registre à décalage sont dirigées vers le bloc logique responsable du calcul à l'aide d'un démultiplexeur. Ce démultiplexeur est contrôlé par un compteur synchronisé avec l'horloge de la FPGA. Les données issues du calcul du multiplicateur sont ensuite acheminées vers un accumulateur. Celui-ci additionne sa sortie, décalée d'une période d'horloge, avec son entrée, permettant ainsi l'accumulation des valeurs générées par le multiplicateur. Lorsque le compteur atteint sa valeur finale, la sortie de l'accumulateur reflète la somme de tous les échantillons multipliés par leurs coefficients respectifs. Cette approche permet ainsi d'obtenir le même résultat que celui obtenu avec un nombre de multiplicateurs équivalent à la moitié du nombre de taps. Cela apporte un gain de ressources très important, car dans les architectures précédentes la majeure partie des ressources étaient utilisées par les multiples blocs multiplicateurs.

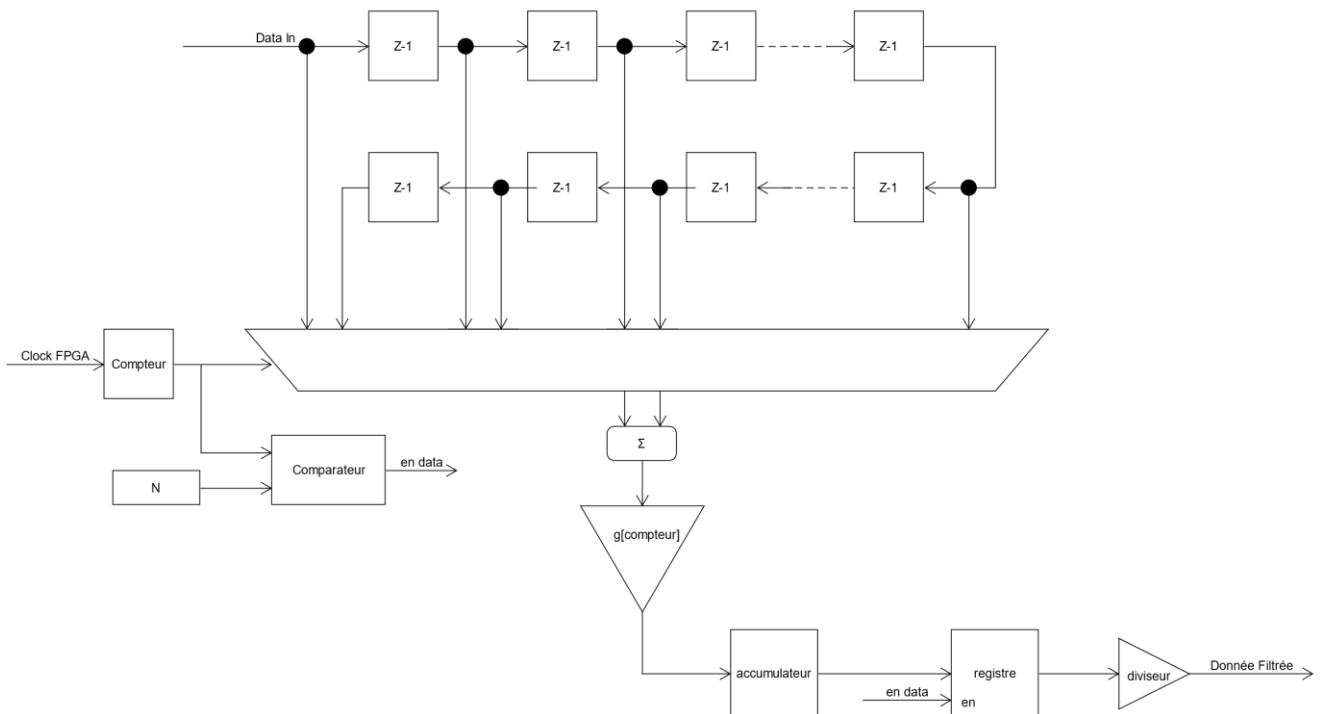


Figure 22 : schéma de principe filtre FIR série

### 6.5.2 Analyse des Performances

Cette méthode permet ainsi de réduire considérablement les ressources nécessaires pour effectuer l'opération de filtrage. Cependant, elle engendre une augmentation significative du temps de calcul. Désormais, une période d'horloge est équivalente à une opération de multiplication. Les ressources multiplicatives nécessaires sont réduites de "nombre de taps" à "un", mais le temps de calcul augmente lui de "1" à "nombre de taps".

Cependant, cette approche présente des avantages notables. La fréquence d'horloge de la FPGA étant de 66 MHz et les échantillons étant reçus à une fréquence de 48 kHz, cela offre une marge de 1375 périodes d'horloge pour exécuter le calcul.

Il est donc important maintenant de prendre en compte le temps de calcul dans nos analyses de performance. Voici un tableau récapitulatif (Figure 23) de l'utilisation de la FPGA xc3s-1200e (l'équivalent plus puissant de la xc3s-500e) réalisant un crossover complet (passe-haut passe-bas) avec des coefficients de 32 bits.

	t calc [μs]	t total [μs]	% de temps tot	flip flops[%]	LUTs[%]	Sclices[%]
coeffs 32 bits 200 taps	4.6	20.8333333	22.08	39	46	87
coeffs 32 bits 225 taps	5.16	20.8333333	24.768	44	50	95

Figure 23 : utilisation ressources FPGA pour architecture Série 200 et 225 taps.

Comme le montre la Figure 23 la contrainte majeure provient toujours des ressources limitées de la FPGA, plutôt que du temps disponible pour effectuer les calculs. Cette situation découle du registre à décalage, qui représente la principale source d'utilisation des ressources au sein de cette architecture.

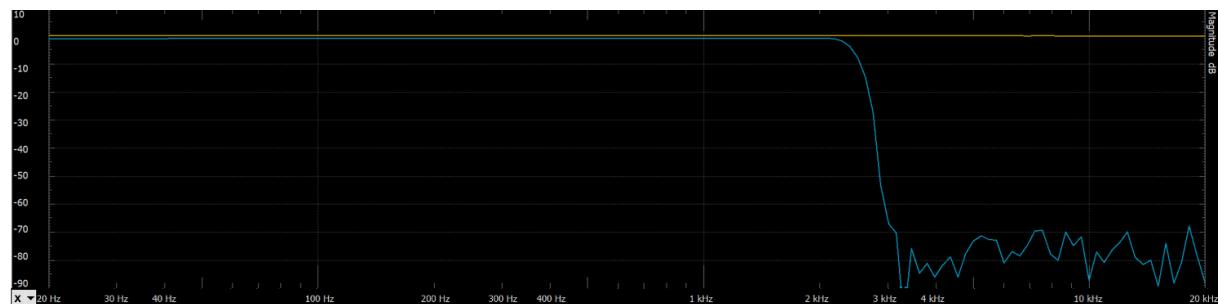


Figure 24 : diagramme de Bode passe-bas 225 taps.

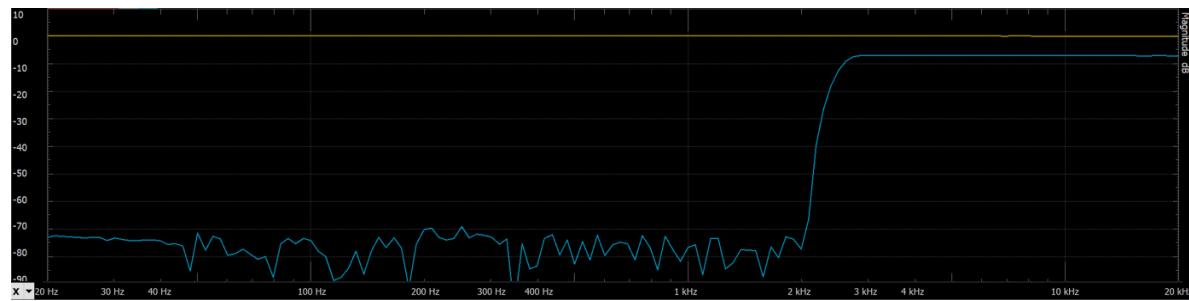


Figure 25 : diagramme de Bode passe-haut 225 taps.

Comme on peut le voir sur le diagramme de Bode (figure 25), la réponse en fréquence du passe-haut ne se stabilise pas à 0 dB, mais à -6 dB. Cette configuration est intentionnelle afin de corriger le rendement du tweeter (le transducteur des aigus). En effet, étant donné que le tweeter présente un meilleur rendement, pour un même signal électrique il produit un son plus intense. Pour équilibrer les performances des deux transducteurs, il a été nécessaire de réduire l'intensité du canal aigu.

## 6.6 Architecture Série-parallèle

L'architecture série-parallèle offre l'avantage d'exécuter plusieurs opérations de multiplication en une seule période d'horloge, ce qui peut être particulièrement avantageux lorsque le temps est limité. Cependant, dans notre situation, la limitation n'est pas temporelle, et par conséquent, cette architecture n'a pas été mise en œuvre ni testée. Néanmoins, il est intéressant de prendre connaissance de son existence et de ses capacités.

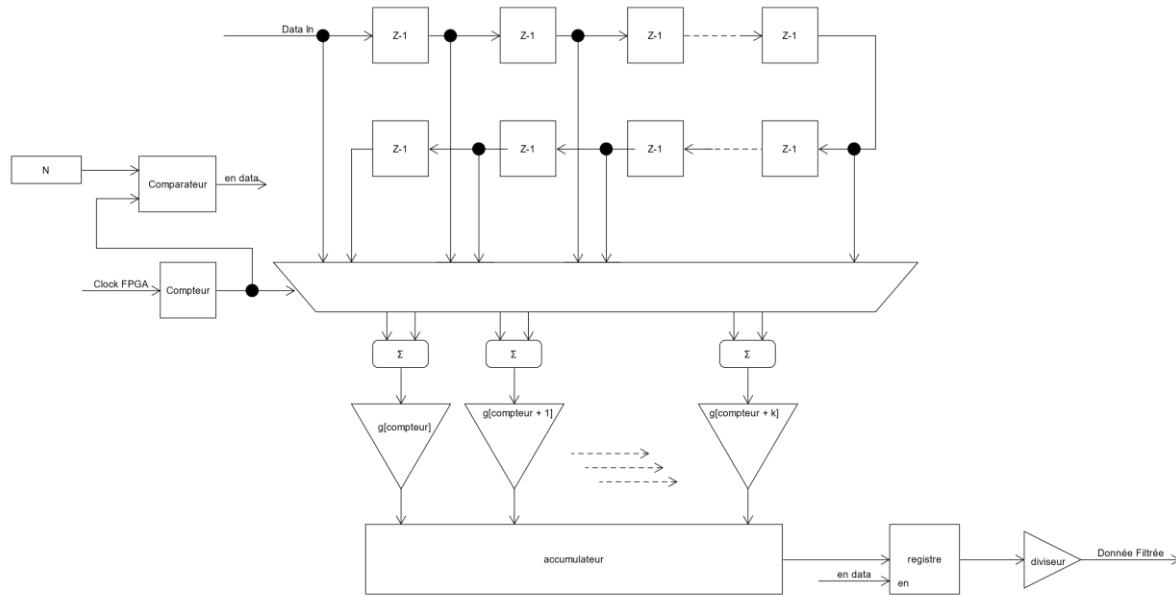


Figure 26: schéma de principe filtre FIR parallèle / série

## 6.7 Utilisation de la RAM pour le registre à décalage

### 6.7.1 Description

Afin d'améliorer davantage les performances, il est essentiel de réduire la charge de travail de la FPGA. Actuellement, la majeure partie de cette charge provient du registre à décalage. Une solution alternative serait d'opter pour l'écriture directe des valeurs des échantillons dans la RAM. En effet, la RAM offre une capacité de stockage élevée avec un accès extrêmement rapide. Cette approche aurait le potentiel de considérablement alléger la charge de travail de la FPGA, tout en maintenant intactes les fonctionnalités essentielles du système.

### 6.7.2 Qu'est-ce qu'une RAM ?

Une RAM, ou Random Access Memory, est une mémoire volatile utilisée dans les systèmes informatiques pour stocker temporairement les données avec une grande vitesse d'accès. Contrairement aux mémoires de stockage permanent telles que les disques durs, la RAM assure un temps d'accès constant pour toutes les données qui y sont stockées. Grâce à sa rapidité et à sa capacité de traitement en parallèle, la RAM joue un rôle crucial dans l'amélioration des performances des systèmes électroniques et informatiques. Dans le contexte du projet actuel, l'utilisation d'une RAM contribue de manière significative à l'optimisation des ressources de la FPGA tout en favorisant un traitement efficace du signal audio.

### 6.7.3 Comment utiliser une RAM dans ce projet ?

Dans le cadre de ce projet, la mémoire RAM interne de la FPGA sera utilisée. Pour accéder à cette mémoire, plusieurs informations sont nécessaires, notamment l'adresse à laquelle l'accès doit être effectué, les données à écrire en mode d'écriture, les données à lire en mode de lecture, ainsi qu'un signal "write enable" qui, lorsqu'il est en niveau logique élevé, active le mode d'écriture de la RAM, tandis qu'un niveau logique bas active le mode de lecture. La mémoire RAM présente dans notre FPGA est de type "dual port", ce qui signifie qu'il est possible d'accéder simultanément à deux adresses différentes. Cette caractéristique sera avantageuse pour les étapes suivantes. Cependant, pour l'écriture et la lecture des coefficients, un seul port de la RAM sera utilisé.

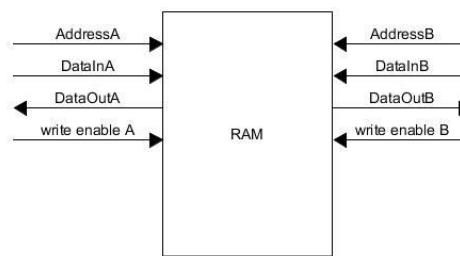


Figure 27: schéma de principe bloc RAM

### 6.7.4 Comment utiliser une RAM comme registre à décalage ?

Afin de comprendre le principe de ce qui va suivre, voici un exemple simplifié d'utilisation de la RAM comme registre à décalage dans lequel la RAM va être utilisée comme registre à décalage pour cinq échantillons (équivalent d'un filtre 5 Taps).

#### 6.7.4.1 Écriture dans la RAM

À l'arrivée de chaque nouvel échantillon, un compteur est incrémenté. La valeur de ce compteur est utilisée comme adresse pour stocker les données dans la mémoire RAM. Lorsque le compteur atteint sa valeur maximale, il est remis à zéro. Par conséquent, le nouvel échantillon D5 occupera la place précédemment occupée par l'échantillon D0, et ce processus se répétera pour les échantillons suivants.

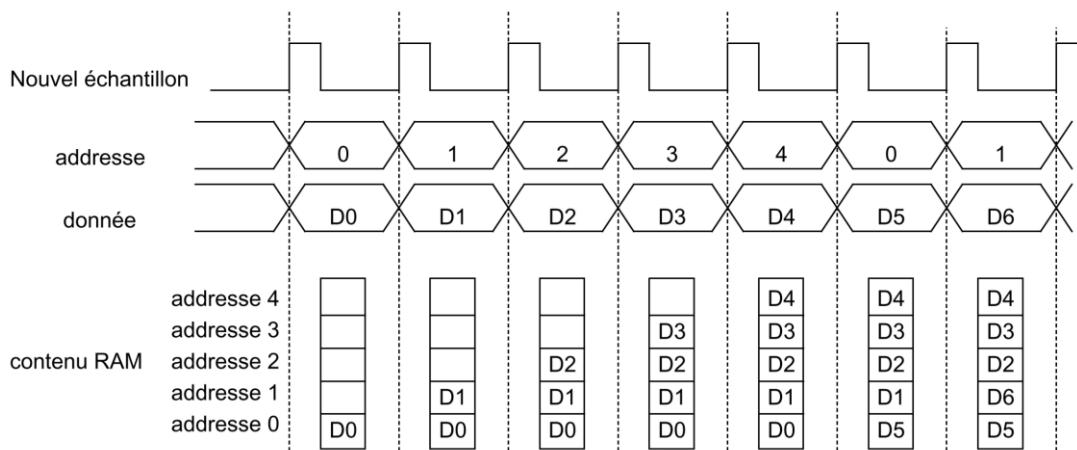


Figure 28 : chronogramme écriture RAM

#### 6.7.4.2 Lecture simple de la RAM

La lecture de la mémoire RAM débute uniquement lorsque celle-ci est pleine. Une lecture complète de la RAM est effectuée à chaque fois qu'un nouvel échantillon apparaît. Lors de la première lecture, il suffit de lire les données depuis l'adresse 0 jusqu'à l'adresse 5. Cependant, pour les lectures suivantes, il est nécessaire de déplacer le point de départ de la lecture afin de maintenir un ordre chronologique correct. Ainsi la deuxième lecture, démarre à l'adresse 1 et se termine à l'adresse 0. Ce même principe est appliqué pour les lectures ultérieures : la troisième lecture démarre à l'adresse 2 et se termine à l'adresse 1, et ainsi de suite. En revanche, cette méthode de lecture ne tire pas pleinement parti de l'optimisation consistant à additionner les coefficients initial et final pour réduire le nombre de multiplications et de coefficients à stocker. Dans cette approche, l'avantage de la symétrie du filtre n'est pas pleinement exploité.

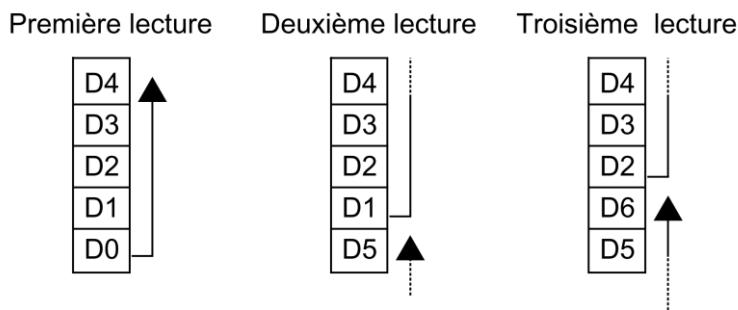


Figure 29 : illustration de lecture simple RAM

#### 6.7.4.3 Lecture symétrique de la RAM

Pour synchroniser la lecture de la mémoire RAM avec l'architecture de calcul du filtre symétrique, il est nécessaire de lire les coefficients de manière symétrique. Lors de la première lecture, il requiert de lire les données D0 et D4 afin de les additionner, puis de procéder à leur multiplication. Ensuite, les données D1 et D3 doivent être lues. Enfin, les deux compteurs d'adresse pointeront vers la même case mémoire en l'occurrence D2. Dans ce cas, seule l'une des deux valeurs pointées est retenue et est ensuite multipliée par le coefficient correspondant. Cette synchronisation garantit que les opérations de multiplication et d'addition sont effectuées en respectant la symétrie du filtre.

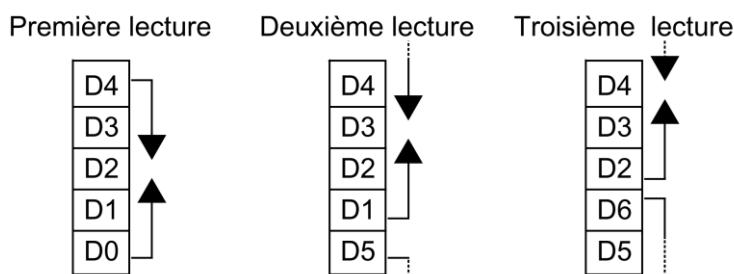


Figure 30 : illustration de lecture symétrique de la RAM

### 6.7.5 Architecture Complète

Le schéma illustré dans la Figure 31 représente schéma bloc de la lecture et l'écriture de la mémoire RAM, ainsi que le processus de calcul du filtre. Cependant, ce schéma a été largement simplifié pour une meilleure compréhension. Il convient de noter que dans cette représentation conceptuelle, tous les accès en lecture et en écriture sont considérés comme instantanés, contrairement à la réalité où une période d'horloge est nécessaire entre la demande de lecture de la RAM et le changement de sa sortie. De plus, il est supposé que les compteurs et décompteurs se réinitialisent automatiquement. Les compteurs et décompteurs de lecture démarrent automatiquement avec la valeur appropriée. Une version plus complète de ce schéma est disponible en annexe(12.1)

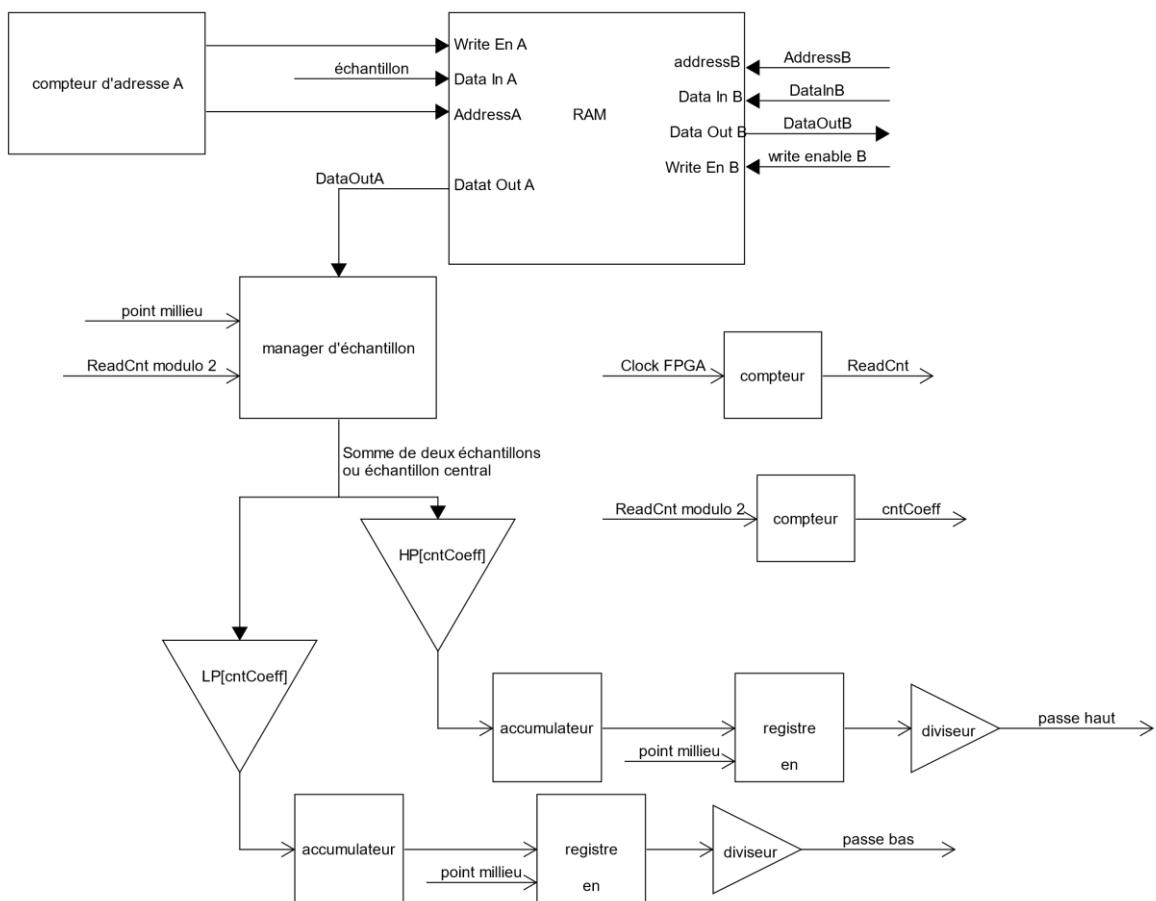


Figure 31 : schéma de principe Filtre avec RAM

Ci-dessous (Figure 32) se trouve un chronogramme qui offre une vue détaillée du processus d'écriture d'un coefficient suivi de la lecture symétrique de la mémoire RAM. Ce chronogramme présente toutes les valeurs essentielles nécessaires pour compléter le schéma illustré dans la Figure 31. Il est important de noter que ce chronogramme suit les mêmes simplifications que celles présentes dans le schéma précédent (Figure 31).

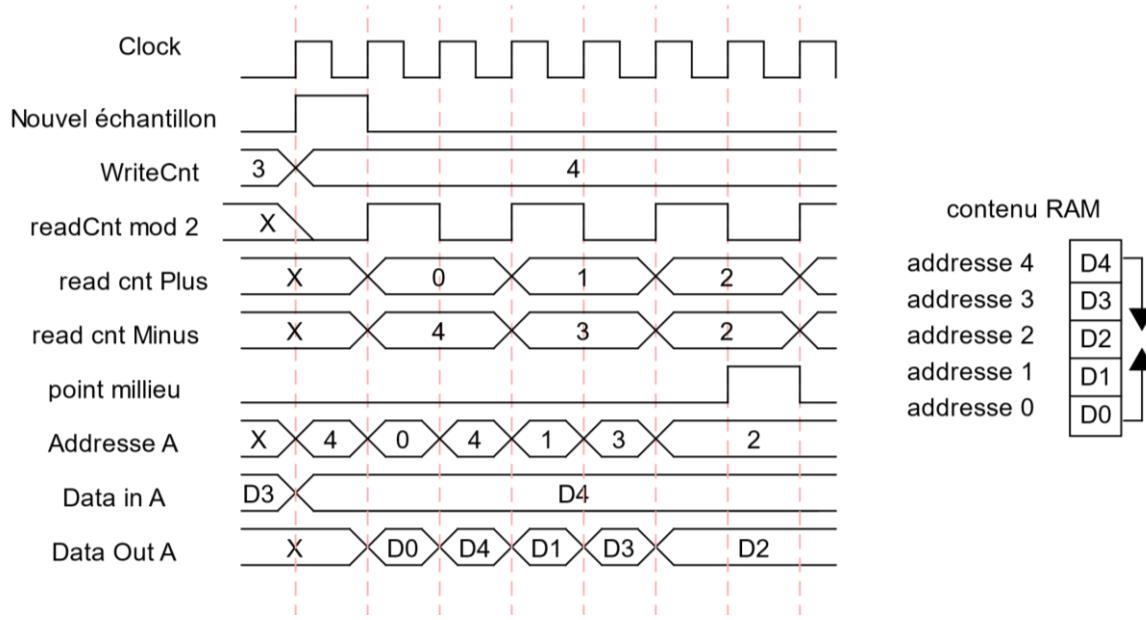


Figure 32 : chronogramme de la lecture/écriture de la RAM

### 6.7.6 Analyse des performances

Maintenant que la partie la plus gourmande en ressources du système a été substituée par la RAM, un gain de performances important est attendu.

	t calc [μs]	t total [μs]	% de temps tot	flip flops[%]	LUTs[%]	Sclices[%]
500 taps symétrique	15	20.8333333	72	3	4	7
650 taps symétrique	19.8	20.8333333	95.04	5	16	26

Figure 33 : utilisation ressources FPGA pour architecture Série ave RAM 500 et 650 taps.

L'amélioration des performances est confirmée, comme le montre la Figure 33. La contrainte actuelle provient désormais de l'intervalle de temps entre l'arrivée de deux échantillons plutôt que des ressources de la FPGA. La valeur maximale atteindre est maintenant de 650 taps par filtre.

Cependant il serait possible d'atteindre des valeurs de taps plus importantes. Une erreur de ma part m'a fait penser que le nombre de bits de sortie de la RAM était limité. Cependant il ne l'est pas. Tous les exemples de performance présentés, fonctionnant avec la RAM, sont suboptimaux. Car la lecture et l'écriture des échantillons se font en deux étapes de 16 bits chacune, au lieu de 32 bits en une seule étape. Cela, en plus de réduire les performances de moitié, a grandement complexifié le travail fourni. N'ayant remarqué cette erreur qu'à la fin de mon travail, celle-ci persiste dans le rendu final.

Néanmoins, il est important de noter qu'il n'est pas conseillé de travailler avec des filtres comportant un grand nombre de taps. En effet, à chaque tap supplémentaire, le décalage temporel entre l'entrée et la sortie augmente. Bien que cela ne soit généralement pas un problème lors de l'écoute de musique, cela pourrait présenter des inconvénients pour des musiciens par exemple. Un décalage significatif entre leur action et le son amplifié pourrait être perturbant.

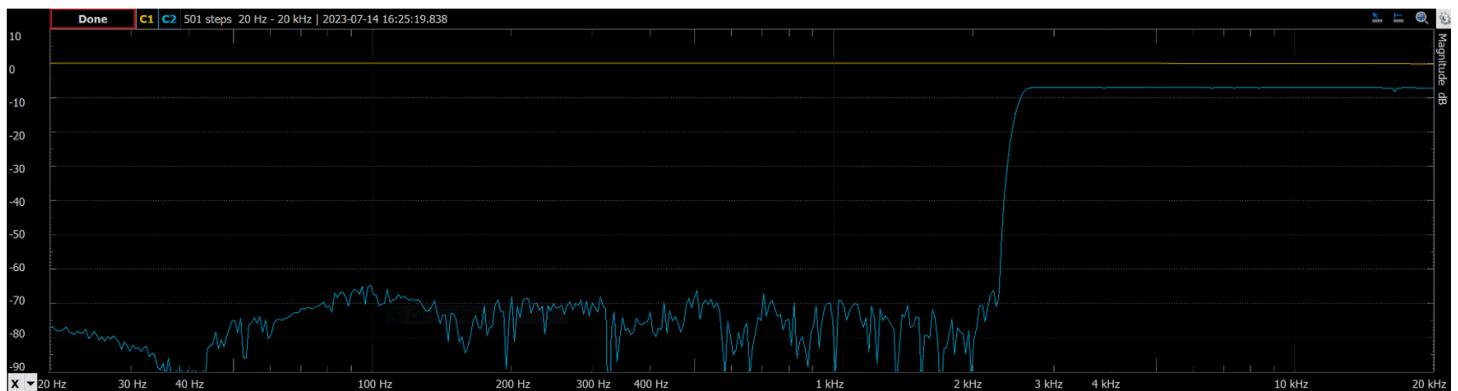


Figure 34 : filtre passe-haut 500 taps



Figure 35: filtre passe-bas 500 taps

## 6.8 Synthèse des résultats

Pour conclure cette section sur le développement des différentes architectures, voici un tableau récapitulatif présentant la taille maximale atteinte par chaque architecture. Afin d'assurer une comparaison équitable, toutes les données de ce tableau sont mesurées en utilisant la FPGA la plus puissante parmi les deux (xc3s-1200e) utilisées dans ce projet.

Architectures	nombre de taps max	limitations	sclices [%]	utilisation du temps a disposition [%]
Paralèle simple	63	Resources	97	0.72
Paralèle symétrique	116	Resources	99	0.72
série	225	Resources	95	24
série avec RAM	650	Temps	26	95

Figure 36 : tableau récapitulatif architectures

Voici une expérience théorique qui consiste à chercher la taille d'une FPGA qui serait capable de créer un filtre de la même performance que le filtre le plus performant atteint. C'est-à-dire deux filtre 650 taps, utilisant les deux premières architectures effectuent la totalité du calcul en une seule période d'horloge.

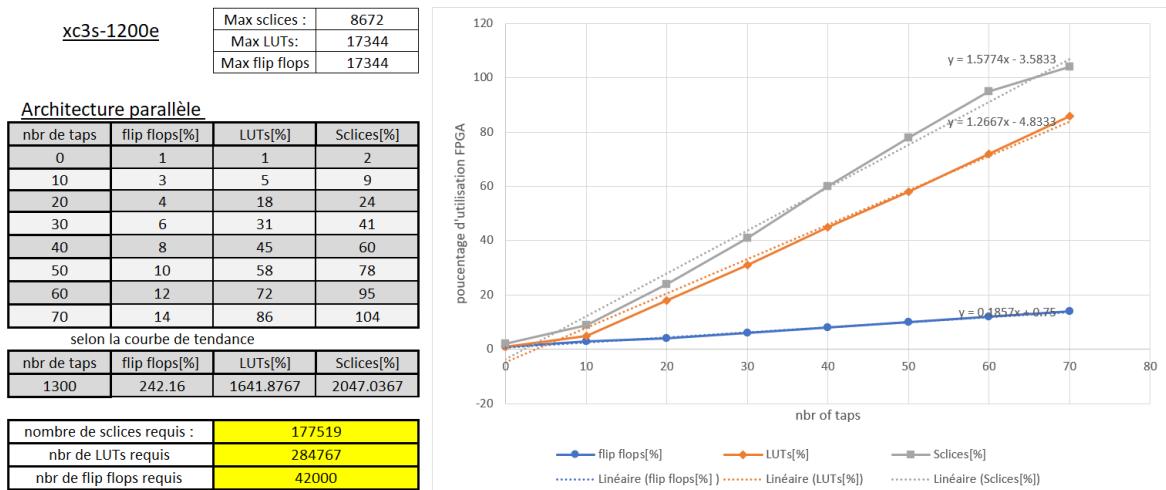


Figure 37 : extrapolation taille FPGA pour architecture parallel (2\*650 taps)

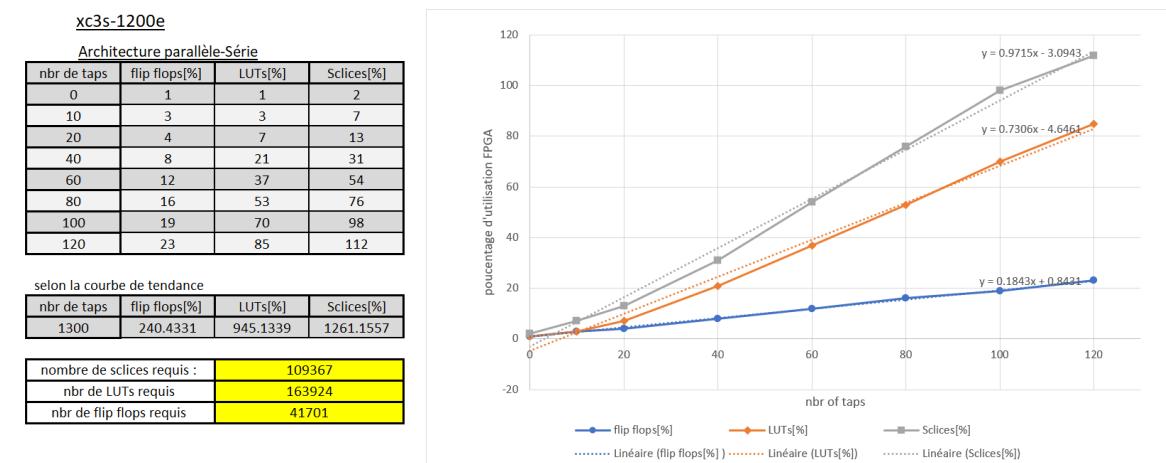


Figure 38: extrapolation taille FPGA pour architecture parallel (2\*650 taps)

## 7 Importation de nouveaux coefficients dans la FPGA

Un autre objectif majeur de ce projet était de permettre la modification des coefficients du filtre à partir d'un ordinateur, sans nécessiter une reprogrammation complète du système. Cette fonctionnalité apporte une grande flexibilité au système, car les filtres implémentés dans la FPGA peuvent être adaptés en fonction des besoins spécifiques de l'utilisateur·rice. Par exemple, deux des systèmes créés dans le cadre de ce projet pourraient être combinés pour être adaptés à un haut-parleur actif à quatre transducteurs. Dans ce scénario, le premier système utiliserait un filtre passe-bas et un filtre passe-bande, tandis que le deuxième système utiliserait un filtre passe-bande et un filtre passe-haut. Cette approche permettrait une personnalisation avancée du système audio en fonction des caractéristiques souhaitées. La section suivante détaillera la solution mise en œuvre pour répondre à cette exigence.

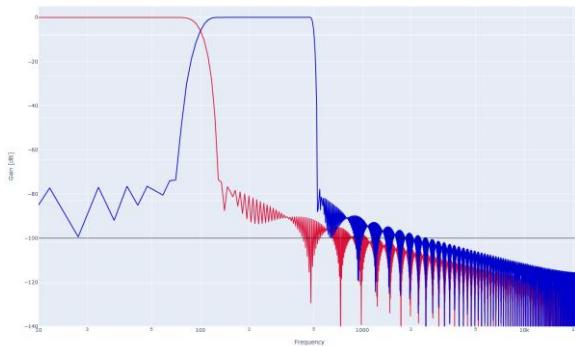


Figure 39 : deux premiers filtres

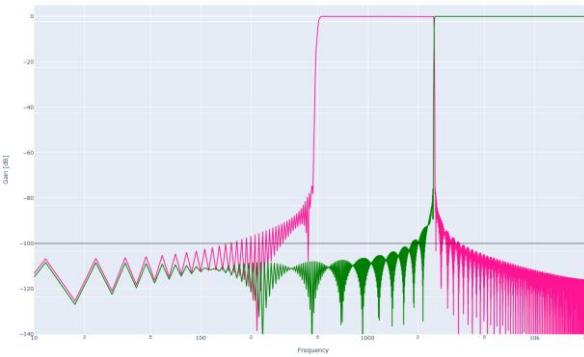


Figure 40 : deux seconds filtres

### 7.1 Communication ordinateur FPGA

Pour établir une communication entre l'ordinateur et la FPGA, il est nécessaire de sélectionner un protocole de communication. Cette communication peut être réalisée de manière filaire ou sans fil. Dans le contexte de ce projet, la communication filaire est privilégiée en raison de la disponibilité des connecteurs sur la carte de développement utilisée, laquelle propose des connecteurs SUB-D mâle et femelle conformes au standard de communication RS-232. De plus, ces connecteurs sont reliés à un circuit intégré MAX3224E[13], qui assure la conversion des tensions du protocole RS-232 (+12V à -12V) aux niveaux de tension acceptés par la FPGA (3,3V à 0V). Par conséquent, ce protocole a été choisi pour la communication entre l'ordinateur et la FPGA dans ce projet.

### 7.2 Protocole RS-232

Le protocole RS-232 est l'une des premières normes de communication série établies pour les équipements électroniques. Il a été largement utilisé pour établir des connexions entre différents appareils tels que des ordinateurs, des modems ou des imprimantes et autres dispositifs électroniques. Ce protocole présente une particularité majeure : il est asynchrone, contrairement au protocole I2S. Cela signifie que le signal transmis n'est pas accompagné d'un signal d'horloge pour synchroniser le récepteur. Ainsi, pour que deux appareils puissent communiquer, ils doivent préalablement convenir d'une vitesse de communication, exprimée en baud par seconde (Baud/s). Le baud par seconde indique le nombre de symboles transmis par seconde. Dans le contexte du protocole RS-232, un symbole équivaut à 1 bit, c'est-à-dire à une valeur logique 1 ou une valeur logique 0. Par conséquent, la valeur de baud par seconde est équivalente à la valeur de bits par seconde.

## 7.3 Décodage RS-232

Pour interpréter les données provenant du protocole RS-232 à l'entrée de la FPGA, un bloc de décodage est nécessaire pour convertir ces données sérielles en données parallèles, représentées sur 7 bits dans ce cas. Dans cette démarche, un bloc de décodage a été réalisé à l'aide d'une simple machine d'état. La transmission des données se fera en utilisant des caractères ASCII, où chaque paquet de 7 bits de données correspondra à un caractère alphabétique.

Comme le protocole utilisé n'est pas synchrone, il ne transporte pas les informations de synchronisation avec lui. Il est donc nécessaire de les générer. Dans ce cas, le transfert de données s'effectue à une fréquence de 9600 Hz, ce qui signifie qu'un signal avec la même fréquence est nécessaire pour lire les données au bon moment. Pour réaliser cela, il suffit de diviser la fréquence de l'horloge de la FPGA par la fréquence de transfert, ce qui donne :  $\frac{66 \times 10^6}{9.6 \times 10^3} = 6875$ . Ainsi, un compteur s'incrémentera à chaque flanc montant de l'horloge de la FPGA et, une fois que ce compteur aura atteint la valeur de 6875, une lecture des données sera effectuée.

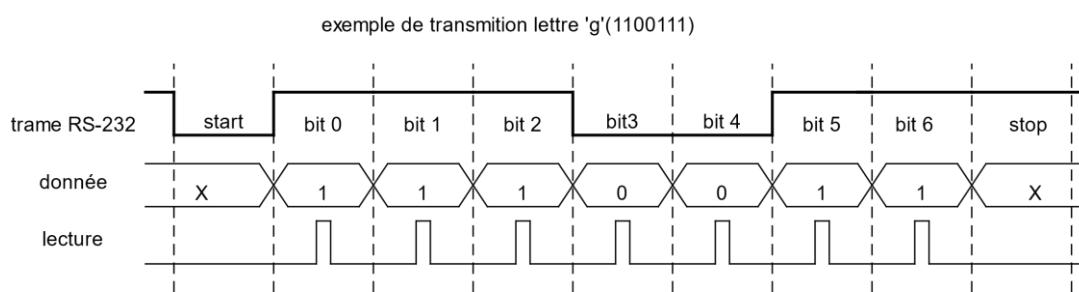


Figure 41 : trame RS-232 transmettant la valeur ascii 'g'

Comme illustré dans la Figure 41X, la lecture des bits ne se produit pas directement au moment du changement de valeur binaire. Cela permet à la donnée de se stabiliser avant d'être lue. Pour ce faire, une fois que le flanc descendant indiquant le début du transfert est détecté, un compteur commence à s'incrémenter. Cependant, sa condition d'arrêt sera réglée à une fois et demi la valeur calculée précédemment. Cela crée le décalage nécessaire pour lire des valeurs stables.

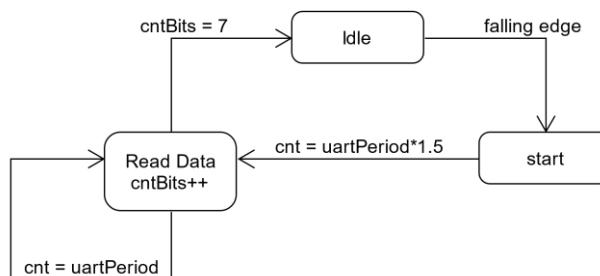


Figure 42 : machine d'état du décodage RS-232

## 7.4 Logique de décodage

Une fois le moyen de communication établi, il est nécessaire de développer une méthode pour convertir les caractères reçus en valeurs numériques exploitables par la FPGA. Lors de la transmission, trois types de données distincts doivent être décodés : le nombre de taps des nouveaux filtres à importer, tous les coefficients de ces filtres, ainsi que leur gain. En effet, en fonction du nombre de taps, il est crucial de pouvoir ajuster le gain appliqué au système afin d'éviter tout dépassement de capacité, comme expliqué précédemment (Chapitre 6.2.1).

Pour cela, un protocole simple a été défini : pour réécrire les coefficients, l'envoi des caractères "N", "E", puis "W" est requis, suivis du nombre de taps encodé en hexadécimal sur 4 caractères. Une fois le nombre de taps transmis, le caractère "T" doit être envoyé pour passer à l'étape suivante. Cette étape consiste en la réception de l'ensemble des coefficients. Les coefficients sont envoyés en alternance, commençant par le premier coefficient du filtre passe-haut, suivi du premier coefficient du filtre passe-bas, puis du deuxième coefficient du filtre passe-haut, et ainsi de suite.

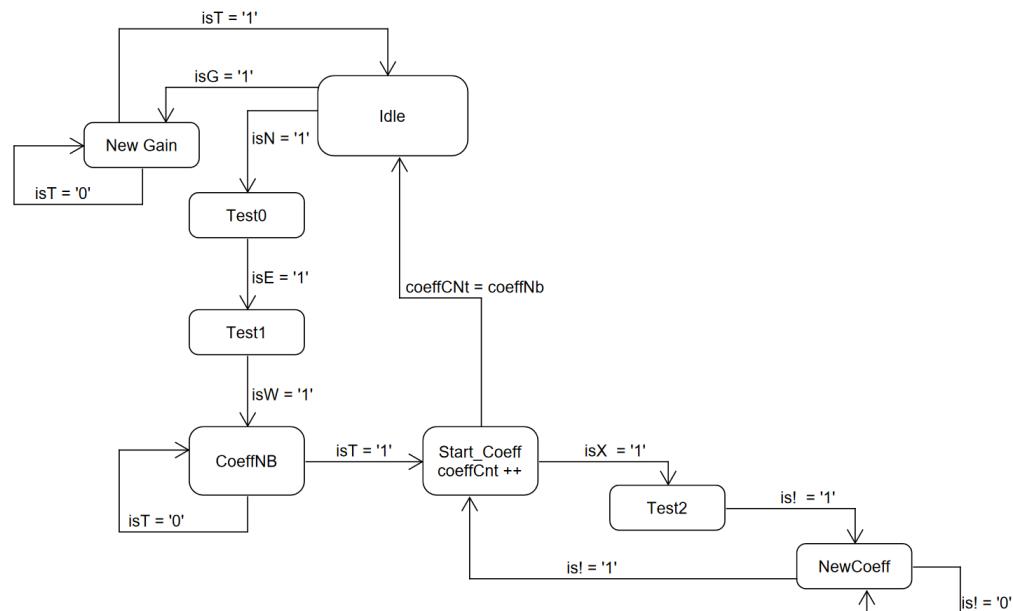


Figure 43 : machine d'état de la logique de décodage

Les coefficients sont codés en hexadécimal sur 8 caractères chacun. Chaque nouveau coefficient commence par le caractère "x", suivi du caractère "!", puis les caractères hexadécimaux représentant la valeur du coefficient sont envoyés, et enfin le coefficient est terminé par le caractère "!". Par exemple, le coefficient 3'735'928'559 serait encodé comme suit : "X !DEADBEEF !". Une fois que le nombre de coefficients correspondant au nombre de taps requis est reçu, le système passe en mode "idle" et est prêt pour de nouvelles réceptions.

La variation du gain peut être ajustée sans nécessiter la réécriture de tous les coefficients. Cette flexibilité permet de réaliser des tests plus rapidement pour déterminer le gain optimal. La variation du gain s'effectue selon une progression exponentielle de puissance de deux. Par exemple, si le gain passe de 7 à 8, le signal sera deux fois plus grand, tandis que si il passe de 7 à 9, il sera multiplié par quatre. Pour écrire un nouveau gain, le caractère "G" doit être envoyé, suivi de la valeur du gain encodée de la même manière que le nombre de taps, c'est-à-dire en utilisant 4 caractères hexadécimaux. Enfin, pour finaliser l'écriture, le caractère "T" doit être envoyé afin de retourner en état "idle" et d'être prêt pour de nouvelles réceptions.

La conversion des caractères ASCII en une valeur numérique exploitable par la FPGA est réalisée par une séquence de conditions. Par exemple, si l'une de ces conditions est remplie, supposons que `isC = '1'` et que c'est le deuxième caractère reçu, alors une variable temporaire prendra la valeur du caractère "C" en hexadécimal, soit 1100. Cette valeur binaire sera ensuite écrite sur les bits 11 à 8 de notre valeur, permettant ainsi une transition facile des caractères ASCII vers une représentation hexadécimale.

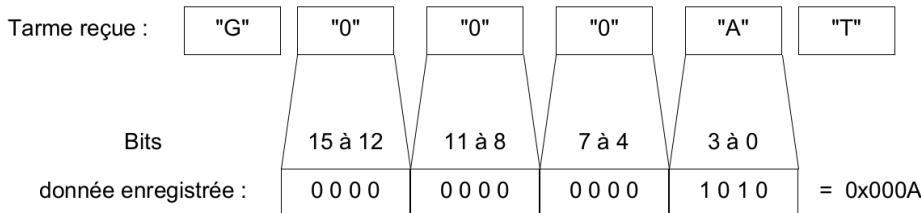


Figure 44 : décodage de char en Hex 16bits

```

if currentState = state_newGain then
    cnt2 <= cnt2 + 1;
    if is0 = '1' then temp2 <= to_unsigned(0,temp2'length);
    elsif is1 = '1' then temp2 <= to_unsigned(1,temp2'length);
    elsif is2 = '1' then temp2 <= to_unsigned(2,temp2'length);
    elsif is3 = '1' then temp2 <= to_unsigned(3,temp2'length);
    elsif is4 = '1' then temp2 <= to_unsigned(4,temp2'length);
    elsif is5 = '1' then temp2 <= to_unsigned(5,temp2'length);
    elsif is6 = '1' then temp2 <= to_unsigned(6,temp2'length);
    elsif is7 = '1' then temp2 <= to_unsigned(7,temp2'length);
    elsif is8 = '1' then temp2 <= to_unsigned(8,temp2'length);
    elsif is9 = '1' then temp2 <= to_unsigned(9,temp2'length);
    elsif isA = '1' then temp2 <= to_unsigned(10,temp2'length);
    elsif isB = '1' then temp2 <= to_unsigned(11,temp2'length);
    elsif isC = '1' then temp2 <= to_unsigned(12,temp2'length);
    elsif isD = '1' then temp2 <= to_unsigned(13,temp2'length);
    elsif isE = '1' then temp2 <= to_unsigned(14,temp2'length);
    elsif isF = '1' then temp2 <= to_unsigned(15,temp2'length);
    -- condition to go out of the state_newGain
    elsif isT = '1' then
        currentState <= state_idle;
    end if;

    if cnt2 /= 0 and cnt2 <=4 then
        if cnt2 = 1 then gain(15 downto 12) <= temp2;
        elsif cnt2 = 2 then gain(11 downto 8) <= temp2;
        elsif cnt2 = 3 then gain(7 downto 4) <= temp2;
        elsif cnt2 = 4 then gain(3 downto 0) <= temp2;
    end if;
end if;

```

Figure 45 : code VHDL du décodage char en Hex 16 bits

## 7.5 Écriture des coefficients dans la RAM

Afin de tirer parti des données reçues pour le calcul du filtre, une décision a été prise : enregistrer les coefficients reçus dans la RAM. En utilisant le deuxième port de la RAM, laissé inexploité dans la section précédente, il est possible de réaliser une lecture parallèle des coefficients et des échantillons.

Pour enregistrer les coefficients reçus via le protocole série, il suffit d'incrémenter un compteur à chaque nouveau coefficient reçu. Il est également nécessaire de mettre un décalage (offset) sur ce compteur afin de ne pas écrire par-dessus les échantillons du registre à décalage. Ensuite, ce compteur est utilisé comme adresse d'écriture dans la RAM. Les données à écrire sont fournies par le bloc logique de décodage. Ainsi, chaque coefficient reçu est écrit dans la RAM à une adresse spécifique calculée en fonction de l'état actuel du compteur.

Cette approche est efficace, mais elle présente un défi lors du démarrage du système, car la RAM est initialement vide. Pour résoudre ce problème, un bloc spécifique a été créé pour gérer l'écriture initiale des coefficients. Ce bloc contient un tableau pré-rempli avec des échantillons. Lors du démarrage, ce bloc écrit tous les coefficients dans la RAM, assurant ainsi le bon fonctionnement du système dès le démarrage.

Cependant, il convient de noter que cette approche signifie que l'importation de nouveaux coefficients via le protocole RS-232 n'est pas permanente. Lorsque le système redémarre, les coefficients importés seront perdus. Afin de conserver de nouveaux coefficients de manière durable, il est nécessaire de les copier manuellement dans un fichier VHDL (Marche à suivre au point 7.7.2) et de reprogrammer l'ensemble de la FPGA. L'importation des coefficients depuis l'ordinateur offre un moyen rapide de prototyper différents filtres, mais elle ne permet pas de changer la configuration du filtre de manière permanente.

Un nouveau défi se profile alors, impliquant la coordination de trois blocs qui accèdent au même port de la RAM. Il est donc essentiel d'établir des mécanismes de coordination pour éviter tout conflit. Pour cela, il est nécessaire de définir des priorités. La plus élevée est attribuée au bloc d'initialisation, car il est crucial pour le bon démarrage du système. Ensuite, le bloc RS-232 bénéficie d'une priorité supérieure par rapport au bloc de filtrage, afin de permettre l'interruption du filtrage lorsque l'utilisateur·rice souhaite reprogrammer des coefficients. Dans le même temps, lorsqu'une nouvelle série de coefficients est importée, le bloc responsable de l'encodage I2S est arrêté pour éviter toute émission de signaux qui pourrait endommager les transducteurs pendant le transfert. Les deux blocs prioritaires ont chacun un signal « en » pour indiquer au Manager à quel moment ils veulent accéder à la RAM.

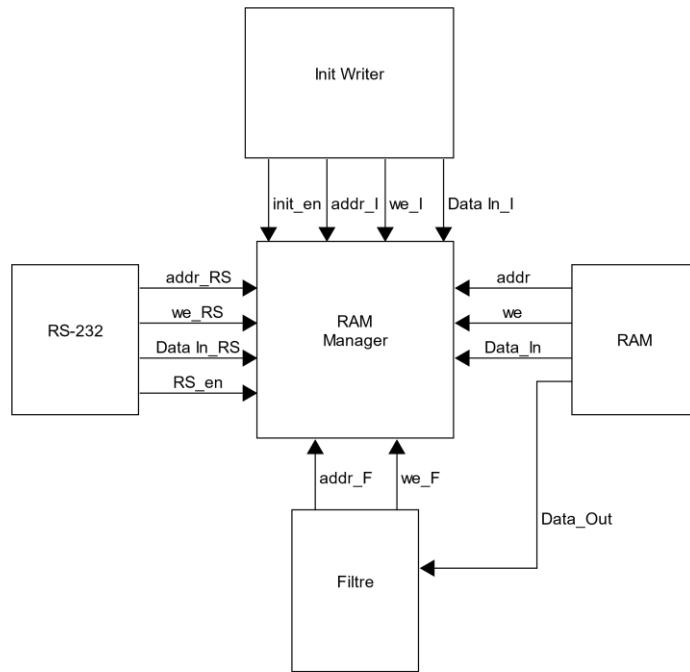


Figure 46 : schéma de principe de la gestion de la RAM

## 7.6 Architecture du filtre utilisant les coefficients dans la RAM

Le bloc responsable du filtrage a été adapté pour lire directement les coefficients depuis la RAM. Contrairement à la lecture des échantillons, la lecture des coefficients est beaucoup plus simple. Il s'agit simplement de parcourir la RAM en suivant l'adresse de début des coefficients jusqu'à l'adresse de fin. Cette simplicité de lecture découle du fait que les coefficients sont enregistrés en alternance dans la RAM. En effet, lors de la lecture, il suffit de récupérer le premier et le dernier échantillon, et simultanément les premiers coefficients pour les filtres passe-bas et passe-haut. Ainsi, tous les éléments nécessaires au calcul du filtre sont disponibles et le temps d'accès à l'information n'a pas doublé bien que l'on demande deux fois plus d'information que pour l'architecture précédente. Une version plus détaillée de ce schéma se trouve en annexe (12.2).

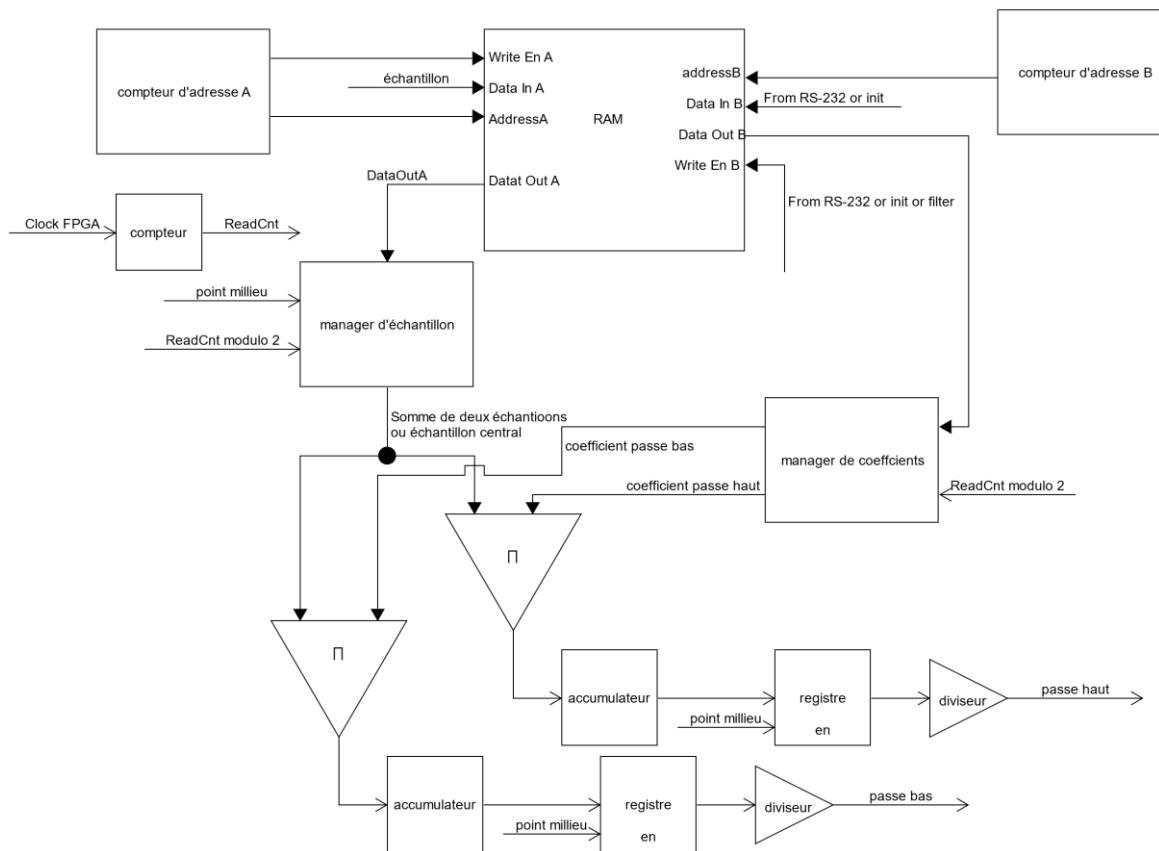


Figure 47 : schéma de principe Filtre avec RAM pour coefficients et échantillons

Voici un chronogramme détaillé qui illustre le processus d'écriture d'un échantillon dans la RAM, suivi de la lecture simultanée des coefficients et des échantillons. Il est important de noter que ce chronogramme suit les mêmes simplifications que celles présentes dans le schéma donné dans la Figure 31.

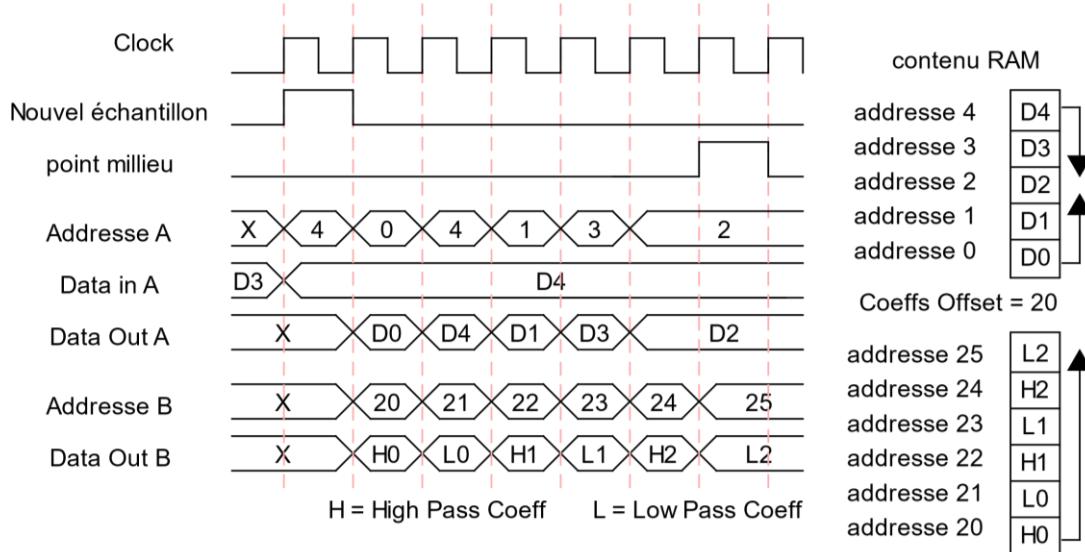


Figure 48 : chronogramme des accès à la RAM pour échantillons et coefficients

## 7.7 Script python

Le script Python a été développé pour créer une interface utilisateur·rice simplifiant l'importation des coefficients dans la FPGA. Une partie de ce script a été générée en collaboration avec ChatGPT[14]. Ce script gère la conversion des valeurs décimales en valeurs entières hexadécimales et arrange les données de manière à les préparer pour la communication RS232. Cette interface permet aux utilisateurs·rice·s de télécharger des filtres depuis l'interface LinFIRXover et de les importer directement dans la FPGA. Les étapes pour réaliser ce transfert sont expliquées dans la section suivante.

### 7.7.1 Marche à suivre

#### 7.7.1.1 LinFirXover

- Ouvrez LinFIRXover.
- Configurez vos filtres selon vos besoins.
- Une fois vos filtres configurés, allez dans l'onglet "File Format" en bas à gauche.
- Sélectionnez l'option "Text - Decimal".
- Cliquez sur le bouton "Download Filters".

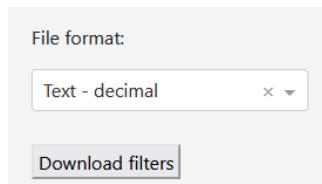


Figure 49 : download filters image

- Le fichier téléchargé contient deux documents texte, chacun contenant les valeurs des coefficients de chaque filtre.

#### 7.7.1.2 Python

- Après avoir téléchargé votre filtre, ouvrez le fichier "FIR\_Lin\_Xover\_Programmer.exe". Vous serez dirigé vers une fenêtre semblable à celle-ci :

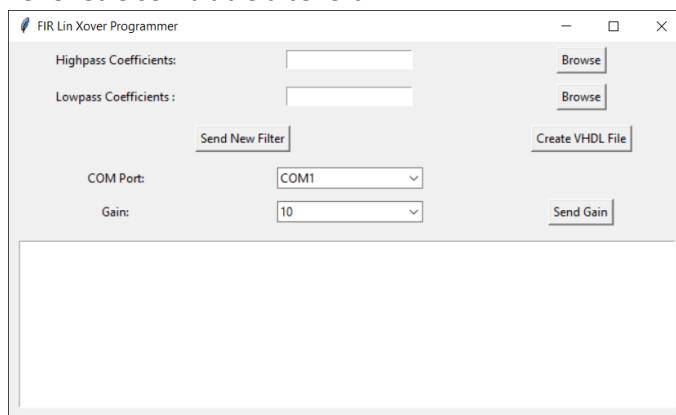


Figure 50 : logiciel FIR\_Lin\_Xover\_Programmer

- Ensuite, sélectionnez vos deux fichiers en utilisant les boutons "Browse".
- Une fois les deux fichiers sélectionnés, choisissez le port COM sur lequel la communication série doit avoir lieu. Ensuite, cliquez sur le bouton "Send New Filter". Si tout a été configuré correctement, une barre de progression apparaîtra. Une fois la progression terminée, la FPGA contiendra les nouveaux filtres.

### 7.7.2 Importation permanente de nouveaux filtres dans la FPGA

Pour importer de nouveaux filtres de manière permanente dans la FPGA, une reprogrammation est nécessaire. Le script Python est capable de générer un fichier que vous pouvez directement intégrer à l'intérieur d'un bloc de l'architecture VHDL. Voici une marche à suivre détaillée pour ce processus :

- Commencez par ouvrir le logiciel "FIR\_Lin\_Xover\_Programmer.exe".
- Sélectionnez les filtres que vous souhaitez importer.
- Cliquez sur le bouton « Create VHDL File ». Cela générera un fichier VHDL nommé "CoeffWriter\_Archi1.vhd" dans le même emplacement que vos fichiers texte.
- 

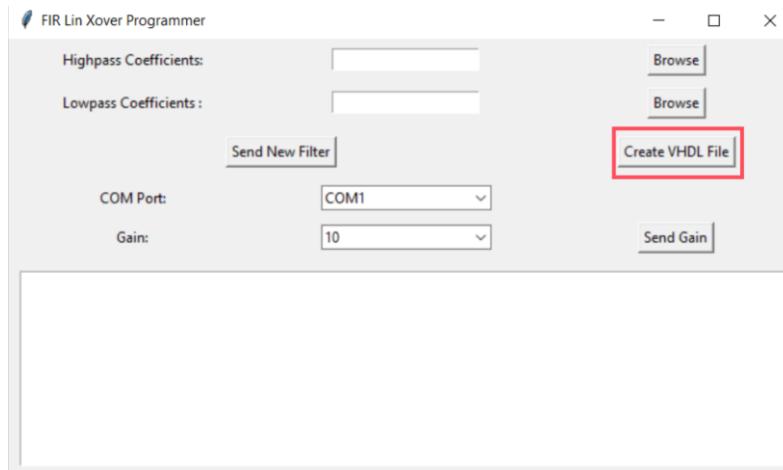


Figure 51 : create VHDL file highlight.

Nom	Statut	Modifié le	Type	Taille
<input checked="" type="checkbox"/> CoeffWriter_Archi1.vhd	✓	17/08/2023 10:30	Mentor Graphics H...	13 Ko
<input type="checkbox"/> high_pass_2500hz_speaker_2_Fs_48.0kHz_650taps_decimal.txt	✓	01/08/2023 17:40	Document texte	13 Ko
<input type="checkbox"/> low_pass_2500hz_speaker_1_Fs_48.0kHz_650taps_decimal.txt	✓	01/08/2023 17:40	Document texte	13 Ko

Figure 52 : emplacement CoeffWriter\_Archi1.vhd

- Copiez ce fichier et collez-le dans le projet VHDL sous : VHDL-> Splitter -> hdl. Un message apparaîtra, cliquez sur "Remplacer le fichier dans la destination" dans la fenêtre de confirmation.

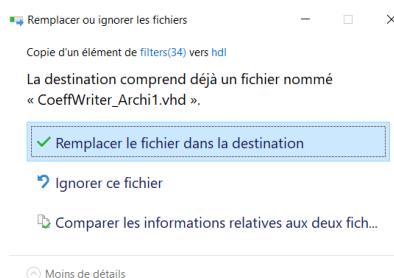


Figure 53 : remplacez CoeffWriter\_Archi1.vhd dans la destination

- Une fois le fichier copié, double-cliquez sur "splitter.bat" dans le dossier "VHDL" (Notez qu'il faut être connecté au VPN de l'école pour accéder à la licence).

Splitter	13/08/2023 14:00	Dossier de fichiers
Splitter_test	13/08/2023 14:00	Dossier de fichiers
splitter.bash	13/08/2023 14:00	Fichier source Bash
<input checked="" type="checkbox"/> splitter.bat	13/08/2023 14:00	Fichier de comma... 2 Ko

Figure 54 : emplacement splitter.bat

- Après avoir ouvert HDL Designer, accédez à l'onglet "Board" situé en bas à droite de la fenêtre, puis sélectionnez le projet "FPGA\_splitter".

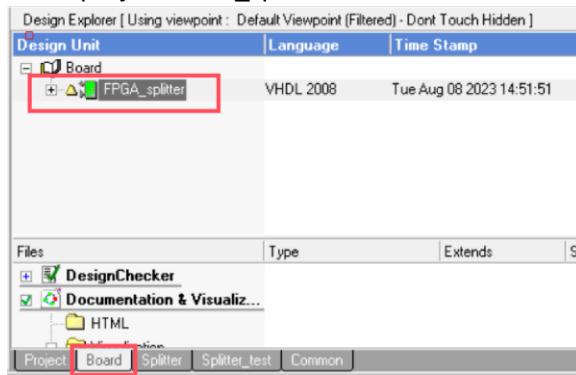


Figure 55 : HDL Designer home page

- Ensuite, allez en haut à gauche du fichier ouvert, puis modifiez la valeur de FILTER\_TAP\_NB pour correspondre à la valeur du filtre que vous souhaitez importer. Assurez-vous que cette valeur est impaire. Si elle est paire, soustrayez y 1 pour la rendre impaire.

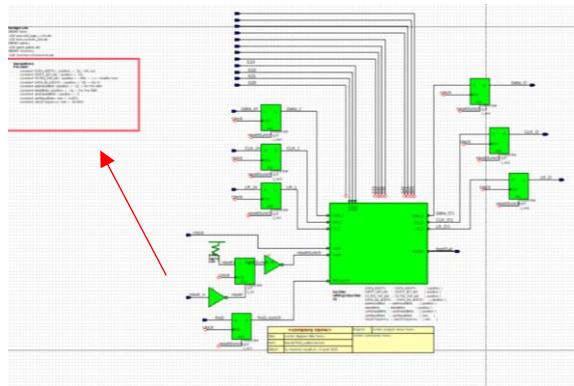


Figure 56 : FPGA\_splitter plan large

```
constant DATA_WIDTH : positive := 32; --i2s out
constant COEFF_BIT_NB : positive := 32;
constant FILTER_TAP_NB : positive := 499; -- <<- modify here
constant DATA_IN_WIDTH : positive := 32; -- i2s In
constant addressBitNb : positive := 12; -- for the RAM
constant dataBitNb :positive := 16; -- For the RAM
constant uartDataBitNb : positive := 7;
constant uartBaudRate: real := 9.6E3;
constant clockFrequency: real := 66.0E6;
```

Figure 57 : emplacement FILTER\_TAP\_NB

- Une fois que vous avez terminé ces étapes, suivez la procédure standard pour charger une nouvelle configuration dans la FPGA.

## 7.8 Analyse des résultats.

Cette section, bien que fonctionnant correctement dans la plupart des cas, présente certaines limitations en raison de problèmes non résolus. L'un des filtres fonctionne comme prévu, mais le deuxième présente des problèmes tels que le maintien d'une bande coupée à -55 dB au lieu de descendre jusqu'à -90 dB, comme prévu. Étrangement, cela ne se produit pas sur tous les filtres importés. De plus, il est impossible de reprogrammer ce filtre en mode passe-haut. La reprogrammation en différents modes passe-bas et passe-bande fonctionne sans problème. Cependant, la raison pour laquelle la reprogrammation en mode passe-haut ne fonctionne pas reste inconnue.

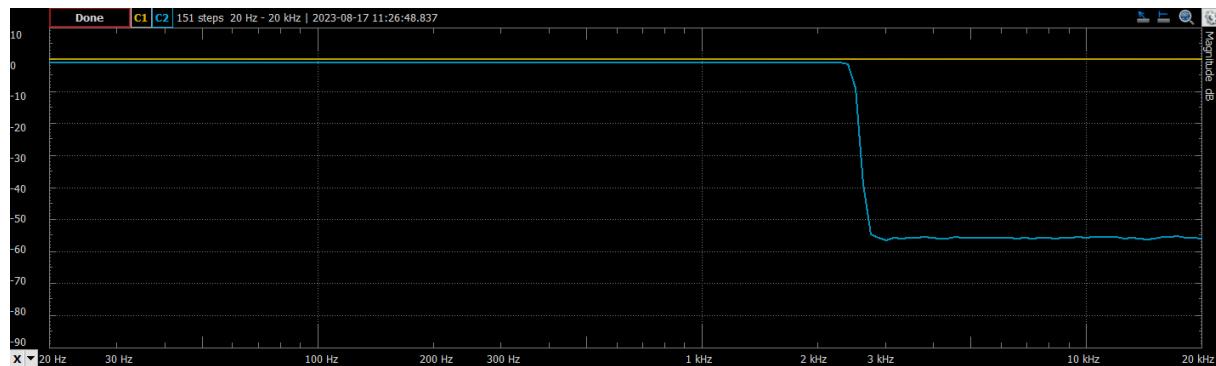


Figure 58 : filtre original passe-bas 500 taps

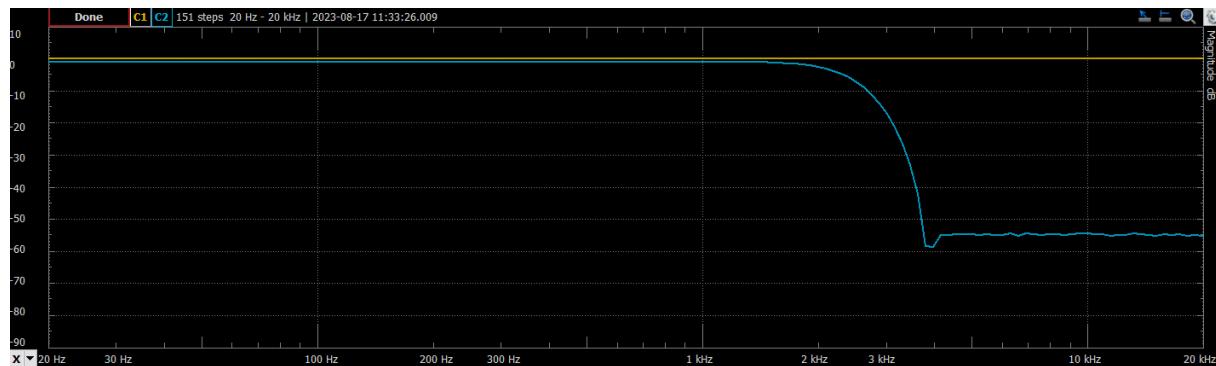


Figure 59 : reprogrammation RS-232 filtre passe bas 100 taps

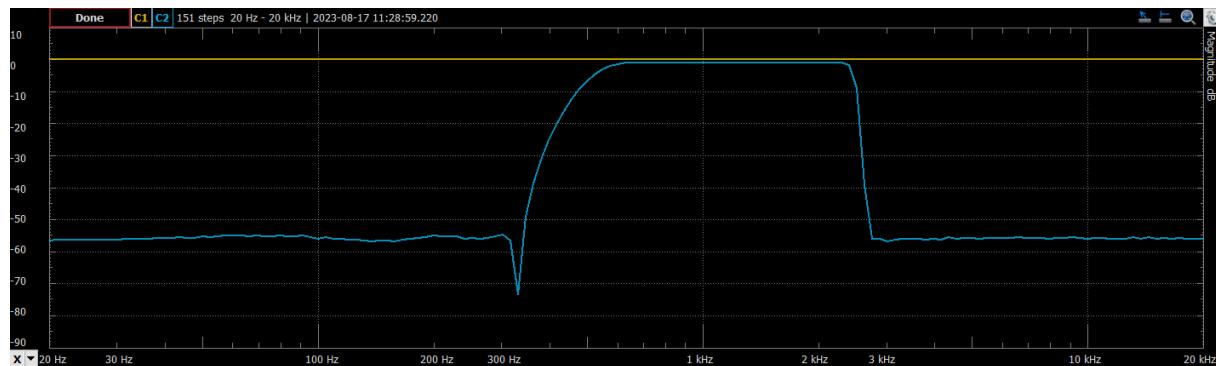


Figure 60 : reprogrammation RS-232 filtre passe bande 600 taps

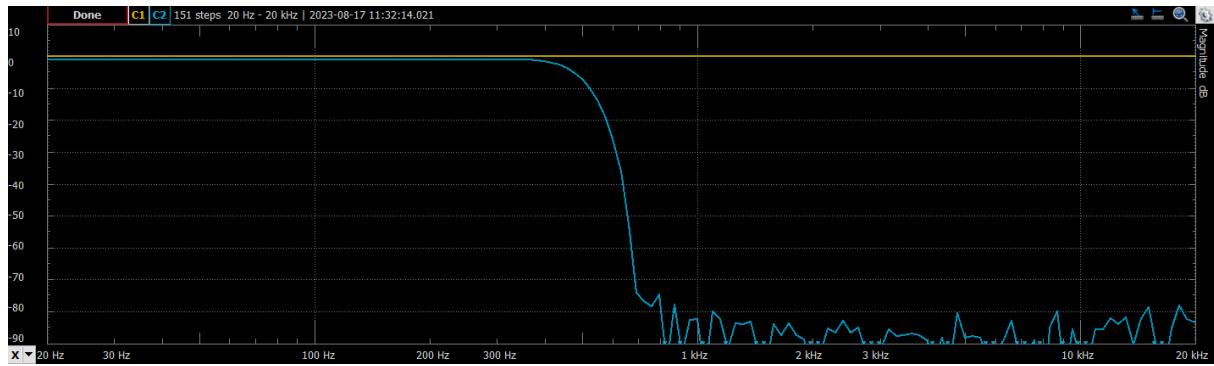


Figure 61 : reprogrammation RS-232 filtre passe bas 600 taps (FC = 500Hz)

Comme le montrent les figures ci-dessus(Figures 59, 60, 61), les reprogrammations des filtres passe-bas et passe-bande fonctionnent correctement. Cependant, lors d'une reprogrammation en passe-haut, le filtre prend la forme d'un passe-bas. Cette erreur reste incompréhensible car en théorie, les coefficients pour créer un filtre passe-bas n'existent plus dans le système, et pourtant la réponse en fréquence montre clairement un comportement passe-bas.

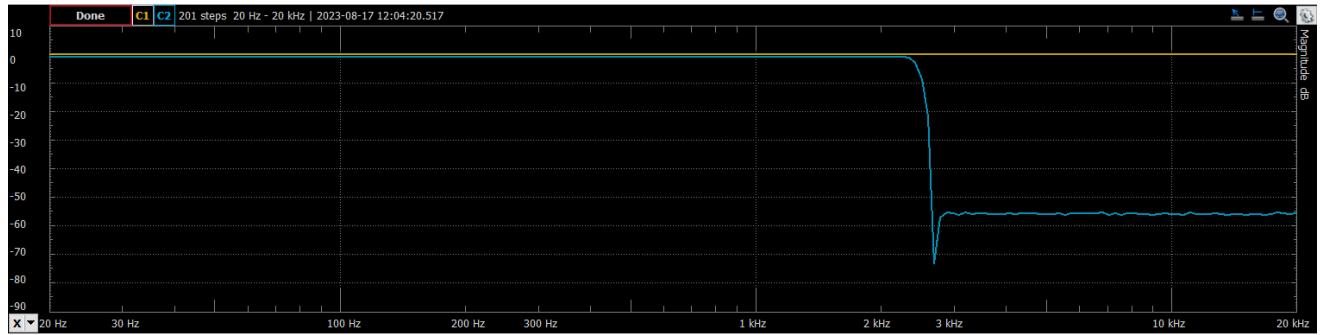


Figure 62 : Diagramme de Bode après une reprogrammation RS-232 passe haut.

---

## 8 Conclusion

### 8.1 Conclusion

En résumé, ce projet a été couronné de succès. L'approche itérative consistant à explorer différentes architectures de filtre, en quête d'optimisation croissante, a été à la fois stimulante et gratifiante. Cette progression a abouti à la réalisation d'un filtre performant et économique en ressources, répondant ainsi pleinement à l'objectif initial. Par la suite, la mise en place de la communication série RS-232 pour l'importation des coefficients s'est avérée fructueuse, offrant une simplicité et une précision remarquables pour le développement de filtres adaptables à tous types de systèmes.

Toutefois, malgré ces réalisations, des limites ont été identifiées, notamment des problèmes non résolus lors de la reprogrammation de certains filtres, des comportements inattendus dans certaines circonstances ainsi qu'une utilisation non optimale de la RAM. Ces défis techniques ouvrent la voie à des opportunités futures d'optimisation et d'amélioration.

L'ensemble de ce projet a permis de démontrer la faisabilité et le potentiel d'un système de traitement audio flexible et reconfigurable, ouvrant ainsi la porte à des développements ultérieurs pour relever ces défis et étendre les capacités de cette solution innovante.

### 8.2 Améliorations futures

Pour parfaire ce projet, certaines modifications ou ajouts pourraient être envisagés. Tout d'abord, il serait pertinent d'automatiser la recherche du gain optimal du filtre à implémenter, plutôt que de le déterminer par une série de tests. Cette optimisation est cruciale pour éviter des dépassements de capacités indésirables dans le signal de sortie, pouvant endommager les transducteurs.

Ensuite, il serait bénéfique d'explorer l'utilisation de la RAM avec des accès en 32 bits, voire en 64 bits, afin d'optimiser les temps de calcul. L'implémentation d'accès en 64 bits pourrait permettre de traiter deux données simultanément en parallèle, offrant ainsi des gains significatifs en performances.

Par ailleurs, il conviendrait de résoudre le problème du filtre présentant des comportements imprévisibles pour garantir un fonctionnement fiable et cohérent du système.

Enfin, une amélioration intéressante serait d'intégrer la possibilité de retarder l'un des deux signaux via l'interface RS-232, ce qui permettrait de synchroniser d'éventuels délais introduits par les transducteurs. Ces évolutions contribueraient à rendre le système encore plus performant, robuste et adaptable aux besoins des utilisateur·rice·s.

Maxime Cesalli : \_\_\_\_\_

Remerciements :

François Corthay

Arnaud Deimon

Relecture : Laurent Cesalli, Nina Balabbas, ChatGPT

## 9 Durabilité

Les Objectifs de Développement Durable (ODD), également connus sous le nom d'Objectifs mondiaux, sont un ensemble de 17 objectifs adoptés par les Nations Unies en 2015 dans le cadre de leur Agenda 2030 pour le développement durable[15]. Ces objectifs visent à guider les efforts mondiaux vers un avenir plus juste, inclusif et durable d'ici à 2030. Chaque objectif cible des domaines clés tels que l'éradication de la pauvreté, la lutte contre les inégalités, la protection de l'environnement et la promotion de la paix et de la justice. Les ODD reconnaissent l'interconnexion des défis mondiaux et visent à mettre en place une approche holistique qui englobe les dimensions économiques, sociales et environnementales du développement. Les 17 objectifs couvrent une gamme de problématiques, de la santé et de l'éducation à l'égalité des sexes, en passant par la préservation de la biodiversité et la promotion de l'accès à l'eau potable et à l'énergie propre. Ils encouragent la collaboration internationale et la coopération entre les gouvernements, les entreprises, la société civile et les citoyens pour créer un avenir durable pour tous. Les ODD représentent un cadre essentiel pour orienter les politiques et les actions à travers le monde afin de relever les défis les plus urgents de notre époque et de bâtir un monde plus équitable, prospère et respectueux de la planète.

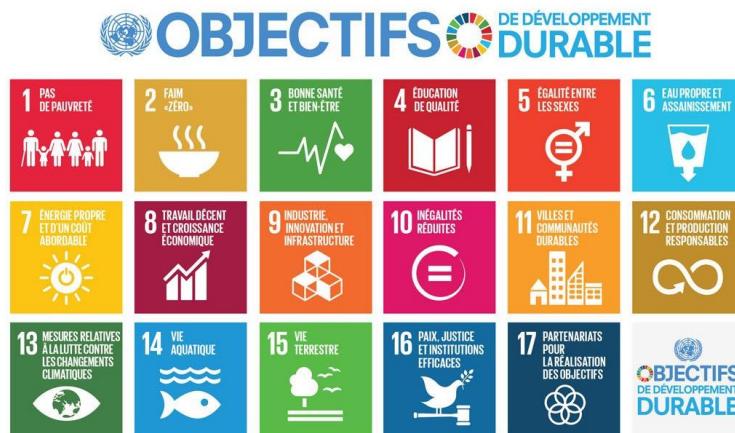


Figure 63 : liste des ODD

Mon projet s'inscrit dans le cadre des objectifs de développement durable de la manière suivante :

**Objectif 9 : Industrie, Innovation et Infrastructure** : Mon projet concrétise des concepts d'ingénierie et d'innovation en concevant un diviseur audio à 2 voies basé sur des filtres FIR symétriques, contribuant ainsi au développement de l'industrie et des infrastructures innovantes.

**Objectif 5 : égalité entre les sexes** : En écrivant ce rapport de manière inclusive, je contribue à promouvoir l'égalité entre les sexes en veillant à ce que les différentes identités et perspectives soient prises en compte et respectées. Cela reflète mon engagement envers un monde plus inclusif et équitable, où chacun·e peut participer activement et bénéficier de projets et d'opportunités, indépendamment de son genre.

**Objectif 17 : Partenariats pour la Réalisation des Objectifs** : Ma collaboration avec ChatGPT [14] pour la conception d'une partie du script Python et la relecture de ce rapport témoigne de l'esprit de collaboration et de partenariat essentiel pour la réalisation des Objectifs de Développement Durable, en unissant différentes compétences pour un impact positif.

En intégrant ces perspectives dans mon projet, je contribue ainsi aux objectifs plus larges de développement durable, en alignant mes efforts techniques avec des initiatives mondiales pour un avenir meilleur.

## 10 Table des illustrations

FIGURE 1: EXEMPLE D'INTERFÉRENCE [1] .....	7
FIGURE 2 : CROSSOVER ANALOGIQUE ORDRE 2 [2] .....	7
FIGURE 3: CROSSOVER FIR 500 TAPS .....	8
FIGURE 4 : FILTRE FIR A PHASE LINÉAIRE .....	8
FIGURE 5 : FILTRE IIR A PHASE NON LINÉAIRE .....	8
FIGURE 6 : SCHÉMA DE PRINCIPE D'UN FILTRE FIR [3] .....	9
FIGURE 7 : RÉPONSE IMPULSIONNELLE D'UN FILTRE FIR SYMÉTRIQUE .....	9
FIGURE 8: SCHÉMA BLOC DU SYSTÈME.....	12
FIGURE 9 : SCHÉMA BLOC ARCHITECTURE FPGA.....	12
FIGURE 10: CHRONOGRAMME D'UN TRANSFERT DE DONNÉES I2S[8] .....	13
FIGURE 11: SCHÉMA INTERNE DU DÉCODEUR I2S .....	14
FIGURE 12: SCHÉMA INTERNE DE L'ENCODEUR I2S.....	15
FIGURE 13: LINFIRXOVER HOME PAGE .....	16
FIGURE 14: SCHÉMA DE PRINCIPE FILTRE FIR PARALLÈLE .....	17
FIGURE 15: UTILISATION RESSOURCES FPGA POUR ARCHITECTURE PARALLÈLE .....	18
FIGURE 16 : DIAGRAMME DE BODE FILTRE FIR 33TAPS .....	18
FIGURE 17: SCHÉMA DE PRINCIPE FILTRE FIR PARALLÈLE SYMÉTRIQUE .....	19
FIGURE 18 : UTILISATION RESSOURCES FPGA POUR ARCHITECTURE PARALLÈLE SYMÉTRIQUE 33 TAPS.....	19
FIGURE 19 : UTILISATION RESSOURCES FPGA POUR ARCHITECTURE PARALLÈLE SYMÉTRIQUE 55 TAPS .....	20
FIGURE 20 : DIAGRAMME DE BODE FILTRE FIR 55TAPS .....	20
FIGURE 21 : SCHEMA BLOC DE PASSE-HAUT SOUSTRAIT AU SIGNAL ORIGINAL. ....	21
FIGURE 22 : SCHÉMA DE PRINCIPE FILTRE FIR SÉRIE .....	22
FIGURE 23 : UTILISATION RESSOURCES FPGA POUR ARCHITECTURE SÉRIE 200 ET 225 TAPS.....	23
FIGURE 24 : DIAGRAMME DE BODE PASSE-BAS 225 TAPS. ....	23
FIGURE 25 : DIAGRAMME DE BODE PASSE-HAUT 225 TAPS. ....	23
FIGURE 26: SCHÉMA DE PRINCIPE FILTRE FIR PARALLÈLE / SÉRIE .....	24
FIGURE 27: SCHÉMA DE PRINCIPE BLOC RAM .....	25
FIGURE 28 : CHRONOGRAMME ÉCRITURE RAM .....	25
FIGURE 29 : ILLUSTRATION DE LECTURE SIMPLE RAM .....	26
FIGURE 30 : ILLUSTRATION DE LECTURE SYMÉTRIQUE DE LA RAM .....	26
FIGURE 31 : SCHÉMA DE PRINCIPE FILTRE AVEC RAM .....	27
FIGURE 32 : CHRONOGRAMME DE LA LECTURE/ÉCRITURE DE LA RAM .....	28
FIGURE 33 : UTILISATION RESSOURCES FPGA POUR ARCHITECTURE SÉRIE AVE RAM 500 ET 650 TAPS.....	29
FIGURE 34 : FILTRE PASSE-HAUT 500 TAPS.....	29
FIGURE 35: FILTRE PASSE-BAS 500 TAPS .....	29
FIGURE 36 : TABLEAU RÉCAPITULATIF ARCHITECTURES.....	30
FIGURE 37 : EXTRAPOLATION TAILLE FPGA POUR ARCHITECTURE PARALLEL (2*650 TAPS).....	30
FIGURE 38: EXTRAPOLATION TAILLE FPGA POUR ARCHITECTURE PARALLEL (2*650 TAPS) .....	30
FIGURE 39 : DEUX PREMIERS FILTRES      FIGURE 40 : DEUX SECONDS FILTRES .....	31
FIGURE 41 : TRAME RS-232 TRANSMETTANT LA VALEUR ASCII 'G' .....	32
FIGURE 42 : MACHINE D'ÉTAT DU DÉCODAGE RS-232 .....	32
FIGURE 43 : MACHINE D'ÉTAT DE LA LOGIQUE DE DÉCODAGE .....	33
FIGURE 44 : DÉCODAGE DE CHAR EN HEX 16BITS.....	34
FIGURE 45 : CODE VHDL DU DÉCODAGE CHAR EN HEX 16 BITS.....	34
FIGURE 46 : SCHÉMA DE PRINCIPE DE LA GESTION DE LA RAM .....	35
FIGURE 47 : SCHÉMA DE PRINCIPE FILTRE AVEC RAM POUR COEFFICIENTS ET ÉCHANTILLONS .....	36
FIGURE 48 : CHRONOGRAMME DES ACCÈS A LA RAM POUR ÉCHANTILLONS ET COEFFICIENTS .....	37
FIGURE 49 : DOWNLOAD FILTERS IMAGE.....	38
FIGURE 50 : LOGICIEL FIR_LIN_XOVER_PROGRAMMER .....	38
FIGURE 51 : CREATE VHDL FILE HIGHLIGHT.....	39
FIGURE 52 : EMPLACEMENT COEFFWRITER_ARCHI1.VHD .....	39

---

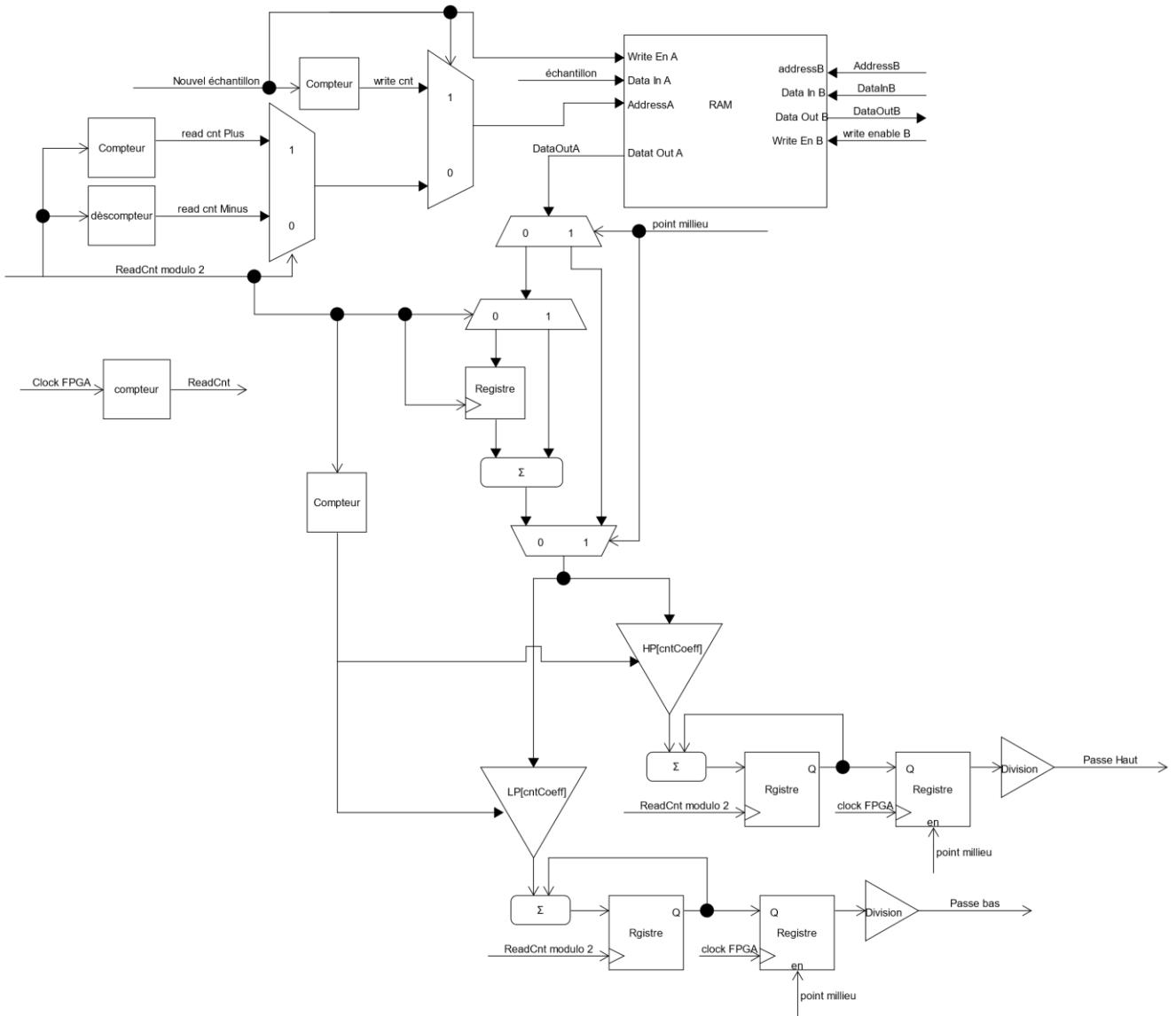
FIGURE 53 : REMPLACEZ COEFFWRITER_ARCII1.VHD DANS LA DESTINATION .....	39
FIGURE 54 : EMPLACEMENT SPLITTER.BAT .....	40
FIGURE 55 : HDL DESIGNER HOME PAGE .....	40
FIGURE 56 : FPGA_SPLITTER PLAN LARGE .....	40
FIGURE 57 : EMPLACEMENT FILTER_TAP_NB .....	40
FIGURE 58 : FILTRE ORIGNAL PASSE-BAS 500 TAPS.....	41
FIGURE 59 : REPROGRAMMATION RS-232 FILTRE PASSE BAS 100 TAPS .....	41
FIGURE 60 : REPROGRAMMATION RS-232 FILTRE PASSE BANDE 600 TAPS.....	41
FIGURE 61 : REPROGRAMMATION RS-232 FILTRE PASSE BAS 600 TAPS (FC = 500Hz) .....	42
FIGURE 62 : DIAGRAMME DE BODE APRÈS UNE REPROGRAMMATION RS-232 PASSE HAUT.....	42
FIGURE 63 : LISTE DES ODD.....	44

## 11 Bibliographie

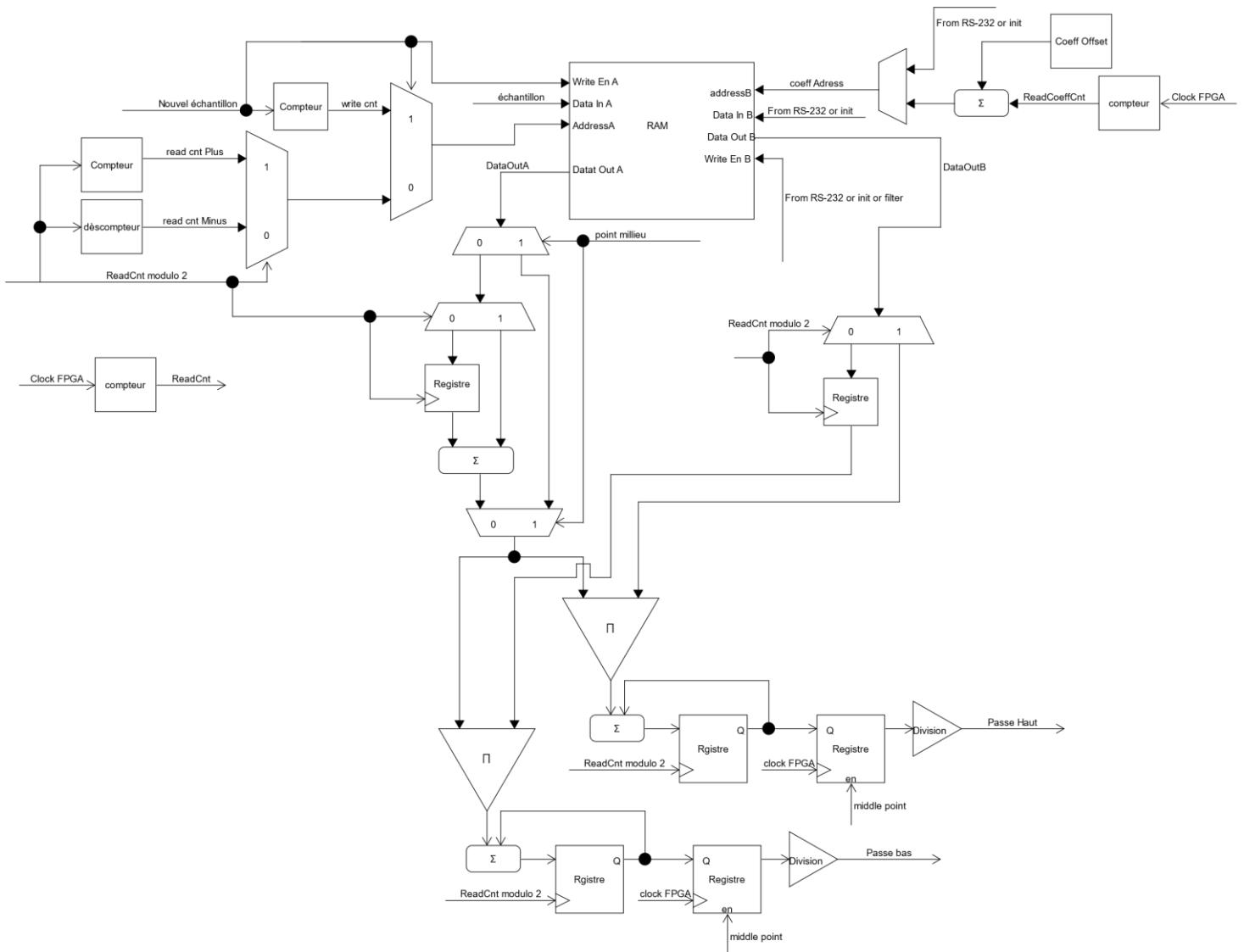
- [1] « Interférences et diffraction d'une onde - Comment ça marche ? ». [https://fr.science-questions.org/comment\\_ca\\_marche/155/Interferences\\_et\\_diffraction\\_d\\_une\\_onde/](https://fr.science-questions.org/comment_ca_marche/155/Interferences_et_diffraction_d_une_onde/) (consulté le 17 août 2023).
- [2] « SubAudio - Les filtres passif et actif pour enceintes ». <https://www.subaudio.org/filtrage.html> (consulté le 17 août 2023).
- [3] « Finite Impulse Response (FIR) Filters ». [https://www.keil.com/pack/doc/CMSIS/DSP/html/group\\_\\_FIR.html](https://www.keil.com/pack/doc/CMSIS/DSP/html/group__FIR.html) (consulté le 17 août 2023).
- [4] L. Palestini, P. Peretti, S. Cecchi, F. Piazza, A. Lattanzi, et F. Bettarelli, « Linear phase mixed FIR/IIR crossover networks: Design and real-time implementation », présenté à Audio Engineering Society - 123rd Audio Engineering Society Convention 2007, janv. 2012.
- [5] M. Hawksford et R. Greenfield, « THE AUDIBILITY OF LOUDSPEAKER PHASE DISTORTION », mars 1990.
- [6] K. Daisuke, « Aural Phase Distortion Detection ». [En ligne]. Disponible sur: <https://dl.icdst.org/pdfs/files/933a77b1bec45d2ff9594581de23c868.pdf>
- [7] « APA Dictionary of Psychology ». <https://dictionary.apa.org/> (consulté le 17 août 2023).
- [8] NXP Semiconductors, « I2S bus specification ». [En ligne]. Disponible sur: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwj11P2W2eOAAxVBgP0HHc1KDgEQFnoECA0QAQ&url=https%3A%2Fwww.nxp.com%2Fdocs%2Fen%2Fuser-manual%2FUM11732.pdf&usg=AOvVaw3h7BCIXbY7WJR2ZsQCltqi&opi=89978449>
- [9] « LinFIRXover ». <http://195.162.163.143:8050/> (consulté le 17 août 2023).
- [10] « iee numeric std ». [https://redirect.cs.umbc.edu/portal/help/VHDL/numeric\\_std.vhdl](https://redirect.cs.umbc.edu/portal/help/VHDL/numeric_std.vhdl) (consulté le 17 août 2023).
- [11] D. H. Marinov, « Coefficient-Translator ». 2 février 2022. Consulté le: 17 août 2023. [En ligne]. Disponible sur: <https://github.com/DHMarinov/Coefficient-Translator>
- [12] D. Marinov, « Part 2: Finite impulse response (FIR) filters », *VHDLwhiz*, 10 février 2022. <https://vhdlwhiz.com/part-2-finite-impulse-response-fir-filters/> (consulté le 17 août 2023).
- [13] D. Analog, « MAX3224E ». [En ligne]. Disponible sur: [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjJ0afK1-WAAxXu3QIHHTH8A\\_EQFnoECBEQAAQ&url=https%3A%2Fwww.analog.com%2Fmedia%2Fen%2Ftechnical-documentation%2Fdata-sheets%2FMAX3224E-MAX3245E.pdf&usg=AOvVaw3DyfiqdO1gtNdcz\\_4NR7If&opi=89978449](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjJ0afK1-WAAxXu3QIHHTH8A_EQFnoECBEQAAQ&url=https%3A%2Fwww.analog.com%2Fmedia%2Fen%2Ftechnical-documentation%2Fdata-sheets%2FMAX3224E-MAX3245E.pdf&usg=AOvVaw3DyfiqdO1gtNdcz_4NR7If&opi=89978449)
- [14] « ChatGPT ». <https://chat.openai.com> (consulté le 17 août 2023).
- [15] J. Bodiguel, « Les Etats membres de l'ONU adoptent un nouveau programme de développement audacieux », *Développement durable*, 25 septembre 2015. <https://www.un.org/sustainabledevelopment/fr/2015/09/25/les-etats-membres-de-lonu-adoptent-un-nouveau-programme-de-developpement-audacieux/> (consulté le 17 août 2023).

## 12 Annexes

### 12.1 Schéma FIR avec RAM uniquement échantillons



## 12.2 Schéma FIR avec RAM échantillons & coefficients



## 12.3 Lien du repository GitHub

[https://github.com/ChaiseM/Bachlor\\_Project\\_Maxime\\_Cesalli](https://github.com/ChaiseM/Bachlor_Project_Maxime_Cesalli)

## 12.4 Mesures réelles sur les Haut-parleurs

