République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Amar Thelidji- Laghouat

FACULTE: DE TECHNOLOGIE

DEPARTEMENT : D'ELECTRONIQUE

MEMOIRE DE MASTER

Réalisé par : Zougari Abd El Kader Djilali
Nagham Abd El Hafidh

DOMAINE : Science et Technologie

FILIERE : Electronique

OPTION : Electronique des systèmes embarqués

Thème

Implémentation d'un filtre numérique dans un FPGA pour le filtrage des signaux audio

Jury de soutenance :

Nom et Prénom	Grade	Qualité
Merah Lahcene	MCA	Encadrant
Ilyas Rougab	MCB	Président
Mourad Reguigue	MCB	Examinateur

Promotion : 2022/2023

Remerciement

En premier lieu, nous remercions ALLAH,
le tout-Puissant pour ses faveurs et ses grâces, de
nous avoir donné le courage et la patience durant ce travail
Nos remerciements et nos profondes gratitudes vont à notre
encadreur Merah Lahcen son soutien scientifique et psychique
était de valeur et essentiel tout au long de ce mémoire.
Nous remercions les honorables membres du jury pour
leur précieux temps accordé à l'étude de notre mémoire.
Adressons également nos remerciements, à
tous nos enseignants, qui nous ont enseigné
et qui par leurs compétences nous ont soutenu
dans la poursuite de nos études.

Dédicace

Dédiez mon diplôme et ma réussite

À mon cher père

aucune dédicace ne saurait être assez

éloquente pour exprimer ce qu'il mérite

pour ses sacrifices qu'il n'a jamais cessé de me donner

depuis ma naissance

À ma chère mère

mon bonheur permanent la plus belle

créature que dieu a créé sur terre,

Á cette source de tendresse, de patience et de générosité.

À le baume de l'âme et de la vie,

mes chères sœurs

À mes amis et à tous ceux qui me connaissent

Abd El Kader Djilali

Dédicace

Je dédie mon diplôme et ma réussite

A mon père,

que Dieu ait pitié de lui,

qui a toujours voulu me voir ce jour-là

A ma chère mère

ma force dans la vie qu'elle

ma pousse et elle ma encourager toujours d'avancer

pour le mieux, que Dieu la protège .

À mes chère frères mon soutien dans la vie

et mon bras droit

Merci à tous ceux qui ont contribué

à mes encouragements, de près ou de loin.

Abd El Hafidh

Sommaire

Sommaire

Liste des figures

Liste des figures

Liste des abréviations

Liste des abréviations

ACAP: Adaptive Compute Acceleration Platform.

ASIC: Application-Specific Integrated Circuits.

CAD : Computer-Aided design.

CI : circuit intégré.

CLB: Configurable Logic Bloc.

DCM: Digital Clock Manager.

DSP : digital signal processor.

EEPROM: electrically erasable programmable read-only memory.

ESL: electronic system-level.

fc: fréquence de coupure.

fe: fréquence échantillonnage.

FPGA: Field-Programmable Gate Array.

HDL: Hardware Description Language.

HDMI: High-Definition Multimedia Interface.

HLS: High Level Synthesis.

IDE: integrated development environment.

IOB : Input Output Bloc.

IP : propriété intellectuelle.

ISE : Integrated Software Environment .

LUT: Look Up table.

LVCMOS: Low voltage complementary metal oxide semiconductor.

LVDS: Low Voltage Differ- ential Signaling.

LVPECL: Low-voltage positive/pseudo emitter–coupled logic.

LVTTL: Low Voltage Transistor to Transistor Logic.

PCI: Payment Card Industry .

PL: logique programmable.

PS: processing system.

PSM: Programmable Switch Matrix .

RAM: random access memory.

RIF : Filtre à réponse impulsionnelle finie.

RII : Filtre à réponse impulsionnelle infinie.

RTL: Register Transfer Level.

SRAM: static random access memory.

USB: universal serial bus.

Verilog: verification and logic.

VHDL: VHSIC Hardware Description Language.

Introduction générale

Introduction générale

L'évolution rapide de la technologie a ouvert la voie à de nouvelles possibilités en matière de traitement du signal. Parmi les applications les plus courantes, le filtrage des signaux audio occupe une place prépondérante. Le filtrage numérique, en particulier, a gagné en popularité grâce à ses avantages en termes de flexibilité, de précision et de facilité de mise en œuvre.

Ce mémoire se concentre sur l'implémentation d'un filtre numérique dans un FPGA (Field-Programmable Gate Array) spécifiquement conçu pour le traitement des signaux audio. Les FPGA, en raison de leur nature programmable, offrent une solution polyvalente et performante pour la mise en œuvre de systèmes de filtrage audio en temps réel.

Le premier chapitre de ce mémoire se consacre à une introduction générale sur les filtres numériques. Nous explorerons les concepts fondamentaux du filtrage numérique, les différentes techniques de filtrage disponibles et leurs caractéristiques. Cette section permettra d'établir une base solide de compréhension pour la suite du mémoire.

Le deuxième chapitre sera consacré à l'étude approfondie des FPGA, en mettant l'accent sur la gamme de produits Xilinx. Nous examinerons les caractéristiques clés des FPGA, leur architecture interne, ainsi que les avantages. Une présentation détaillée des FPGA Xilinx permettra de mieux appréhender les possibilités offertes par ces composants.

Enfin, le dernier chapitre abordera les étapes d'implémentation d'un filtre numérique dans un FPGA. Nous passerons en revue les différentes phases du processus d'implémentation, de la conception du filtre à la programmation du FPGA en passant par la simulation et la vérification des performances. Cette section fournira un guide pratique pour la réalisation d'un système de filtrage audio efficace et optimisé.

En conclusion, ce mémoire vise à présenter une étude sur l'implémentation des filtres numériques dans les FPGA, en mettant l'accent sur leur application dans le domaine du traitement des signaux audio. Nous espérons que cette étude apportera des éclaircissements sur les avantages, les défis et les meilleures pratiques liés à l'utilisation des FPGA pour le filtrage des signaux audio, ouvrant ainsi de nouvelles perspectives dans le domaine de l'audio numérique.

CHAPITRE 1

Filtrage numérique

1.1. Introduction

Le filtrage numérique est une technique largement utilisée dans le domaine du traitement du signal pour manipuler et améliorer les signaux numériques. Il consiste à appliquer des opérations mathématiques sur un signal numérique afin d'en modifier certaines caractéristiques. Cela permet d'éliminer le bruit indésirable, de réduire la distorsion et d'extraire les informations pertinentes contenues dans le signal.

Il existe deux types de filtrage numérique couramment utilisés : le filtrage passe-bas et le filtrage passe-haut. Le filtrage passe-bas permet de laisser passer les fréquences basses du signal tout en atténuant les fréquences plus élevées. Cela peut être utile pour éliminer le bruit de haute fréquence. En revanche, le filtrage passe-haut permet de laisser passer les fréquences élevées tout en atténuant les fréquences plus basses. Cela peut être utile pour extraire des composantes de haute fréquence dans un signal.

Le filtrage numérique peut être réalisé à l'aide de différentes techniques telles que les filtres FIR (Finite Impulse Response) et les filtres IIR (Infinite Impulse Response). Les filtres FIR ont une réponse impulsionnelle de durée finie, tandis que les filtres IIR ont une réponse impulsionnelle de durée infinie. Chaque type de filtre a ses avantages et ses inconvénients, et le choix dépendra des spécificités de l'application.

Ce chapitre vise à fournir une introduction claire et concise au filtrage numérique, en couvrant les principes de base, les techniques de conception et les applications pratiques. Nous allons aborder leurs propriétés, leurs différents types et leur classification en tant que filtres à réponse impulsionnelle finie (RIF) ou à réponse impulsionnelle infinie (RII). Dans cette optique, nous intéresserons davantage aux filtres RIF, car ils sont plus stables et plus réalisables. De plus, nous avons également présenté des méthodes permettant la synthèse de ces filtres.

1.2. Définition

Le terme "filtre numérique" désigne un système utilisé pour altérer la répartition fréquentielle d'un signal numérique conformément à des spécifications données. Un filtre numérique peut être considéré comme un processus de calcul qui transforme un signal

numérique d'entrée (une séquence de nombres) en un signal numérique de sortie (une autre séquence de nombres) afin d'obtenir la modification souhaitée du signal. Le problème du filtrage numérique consiste donc à trouver l'équation qui régit cette transformation des signaux numériques, qui doit d'une part représenter la réponse fréquentielle spécifiée et d'autre part être réalisable en pratique. Cette transformation peut être implémentée sous forme de logiciel (algorithme) ou de matériel (circuits électroniques) [1].

1.2.1. Représentation d'un filtre numérique

Un filtre numérique peut ˆêtre represente en utilisant plusieurs types de spécifications :

- **Fonction de transformé en z**

Ce mode de représentation est le plus usuel. Il permet de lier l'entrée et la sortie dans le plan Z par Y(z) = H(z).X(z). On posera dans la suite [2]:

$$H(z) = \frac{N(z)}{D(z)} = \frac{\sum_{i=0}^{M} bi \, .Z^{-1}}{D(z)} \tag{1.1}$$

Où $N(z)$ est le polynôme du numérateur de la fonction de transfert, tandis que $D(z)$ est son d´dénominateur. N est ici l'ordre du filtre. Dans le cas où $H(z)$ possède des pôles, on parlera de filtres RII (pour Réponse Impulsionnelle Infinie).

Dans le cas où $D(z) = 1$, le filtre ne possède que des zéros. Cette famille de filtre correspond au cas des filtres RIF (pour Réponse Impulsionnelle Finie). Celle-ci n'a pas d''´équivalent en filtrage analogique, et nous verrons que ses propriétés en font une fonction très utilisée en traitement numérique du signal [2].

L''´équation 1.1 peut ´également ˆêtre représentée en mettant en avant les pôles et les zéros.

$$H(z) = b_0 \frac{\prod_{i=1}^{N} (z - z_i)}{\prod_{i=1}^{N} (z - p_i)} \tag{1.2}$$

Où pi sont les pôles et Zi sont les des zéros de H(z). On rappelle ici que la stabilité du filtre sera d´terminé par l'appartenance des pôles au cercle unité (i.e. |pi | < 1), et que des zéros appartenant au cercle unité caractériseront un filtre `a minimum de phase. La figure 1.1 montre plusieurs versions de représentations de $H(z)$. La forme directe (figure1.1. a) peut être d´composée en produit ou en somme de fonctions de transfert d'ordre inferieur,

généralement d'ordre 2. L''équation (1.1) et la figure 1.1.b représentent la forme parallèle, tandis que l''équation 1.4 et la figure 1.1.c représentent la forme cascade [2].

$$H(z) = \sum_{i=1}^{M} H_i(z) = \sum_{i=1}^{M} \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}} \quad (1.3)$$

$$H(z) = \prod_{i=1}^{M} H_i(z) = \prod_{i=1}^{M} \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}} \quad (1.4)$$

- **Réponses impulsionnelles**

La réponse impulsionnelle est la fonction en z inverse de H(z).

$$H(z) = \sum_{n=0}^{\infty} h(n) \cdot z^{-n} \quad (1.5)$$

Comme en filtrage analogique, la sortie d'un filtre $y(nT)$ est le résultat de la convolution du signal d'entrée représente de manière temporelle x(nT) avec la réponse impulsionnelle du filtre $h(nT)$. On a alors $y(nT) = x(nT) * h(nT)$, ou, si on fait abstraction de la période d''échantillonnage T :

$$y(n) = x(n) * h(n) = \sum_{k=0}^{\infty} x(k) \cdot h(n-k) = \sum_{k=0}^{\infty} x(n-k) \cdot h(k) \quad (1.6)$$

Dans le cas où x(n) est une impulsion $\delta(n)$, on retrouve bien $y(n) = h(n)$.

Selon les cas où h(n) est `a support infini ou fini, on retrouvera respectivement les deux types de filtres RII et RIF [2].

- **Equation aux différences**

Une transformation en z inverse de l''équation 1.1 permet d'aboutir à la forme suivante :

$$y(n) = \sum_{i=0}^{N} b_i \cdot x(n-i) - \sum_{i=0}^{N} a_i . y(n-i) \quad (1.7)$$

On identifie ici deux parties distinctes : une partie fonction de la valeur courante et des valeurs précédentes de l'entrée x(n), et une partie fonction des valeurs précédentes de la sortie y(n). Selon si les ai sont non nuls ou nuls, on parlera donc de filtres récursifs ou de filtres non récursifs [2].

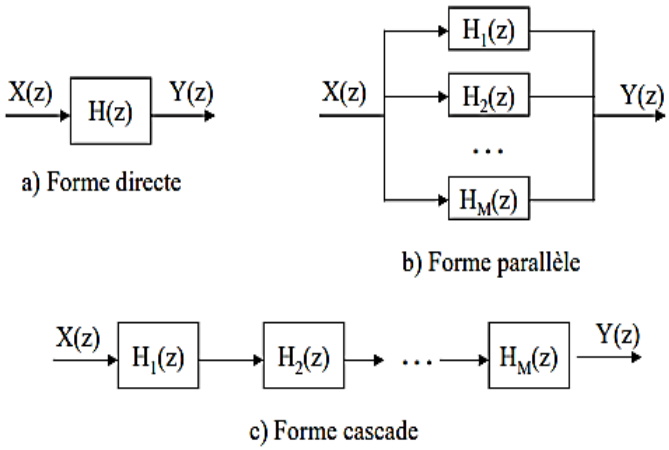

Figure 1.1. Représentations sous forme de fonctions de transfert en z.

1.3. Les différents types de filtres

Il existe quatre types de filtres fondamentaux selon la forme de leur réponses fréquentielles (RIF) (ou réponse impulsionnelle RI).

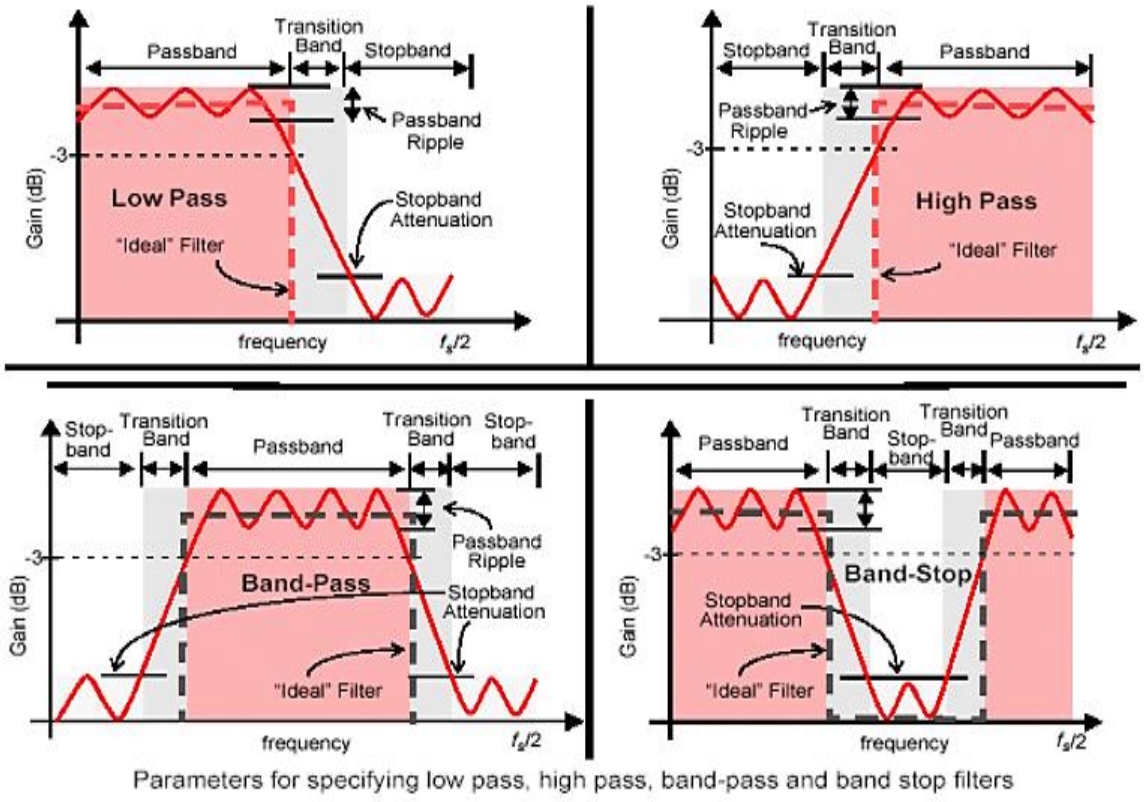

Figure 1.2. Les différents types des filtres (réponses fréquentiel).

1.3.1. Filtre passe-bas

Le filtre passe bas ne fait passer que les fréquences au-dessous de la fréquence de coupure et il atténue les autres (hautes fréquences). Sa réponse fréquentielle est définie par : [3]

$$H(f) = \begin{cases} 1 & \text{si} \quad |f| \leq fc \\ 0 & \text{ailleurs} \end{cases} \tag{1.8}$$

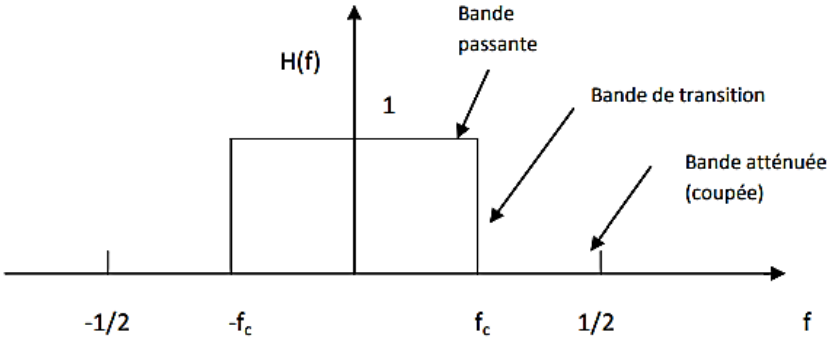

Figure 1.3. Filtre passe bas idéal.

1.3.2. Filtre passe-haut

Le filtre passe-haut permet de laisser passer les fréquences supérieures à la fréquence de coupure tout en atténuant les basses fréquences. En termes de réponse impulsionnelle, les filtres passe-haut amplifient les variations du signal. Ils peuvent être utilisés pour détecter un échelon de signal dans un circuit de déclenchement ou le front montant d'un signal d'horloge.

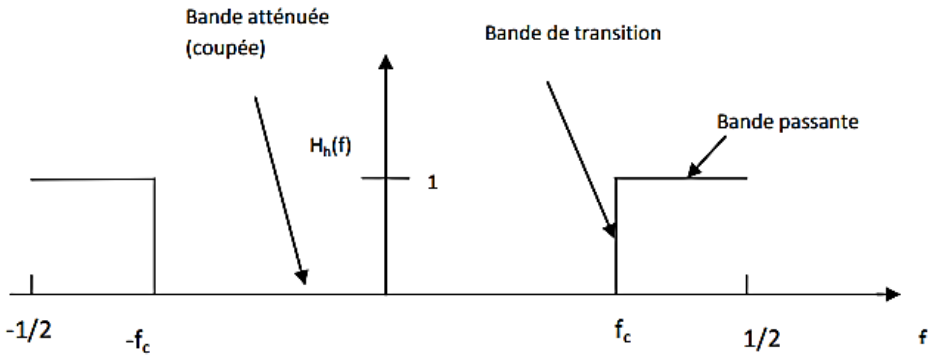

Figure 1.4. Filtre passe haut.

Sa réponse fréquentielle est définie par :

$$H_h(f) = \begin{cases} 1 & \text{si } |f| \geq f_c \\ 0 & \text{ailleurs} \end{cases} \tag{1.9}$$

1.3.3. Filtre coupe-bande

Le filtre coupe-bande est le complémentaire du passe-bande. Il atténue une plage de fréquences

$$H_c(f) = \begin{cases} 0 & \text{si } f_{c1} \leq |f| \leq f_{c2} \\ 1 & \text{ailleurs} \end{cases} \tag{1.10}$$

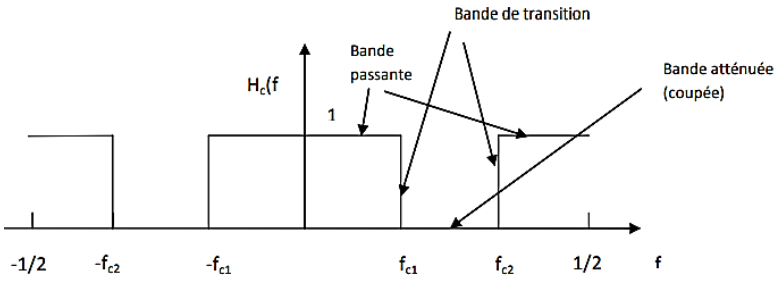

Figure 1.5. Filtre coupe bande.

La RI d'un filtre coupe bande peut être aussi obtenue en considérant un filtre passe bas en parallèle avec un passe haut [3].

1.3.4. Filtre passe-bande

Un filtre passe bande présente un gain supérieur pour une certaine bande de fréquences, il atténue tout ce qui est au-dessus ou au-dessous de cette bande, il est très utilisé dans le domaine des transmissions pour isoler le signal qu'on souhaite capter. On utilise les filtres passe-bandes spécialement pour la réception radio [3].

$$H_B(f) = \begin{cases} 1 & \text{si } f_{c1} \leq |f| \leq f_{c2} \\ 0 & \text{ailleurs} \end{cases} \tag{1.11}$$

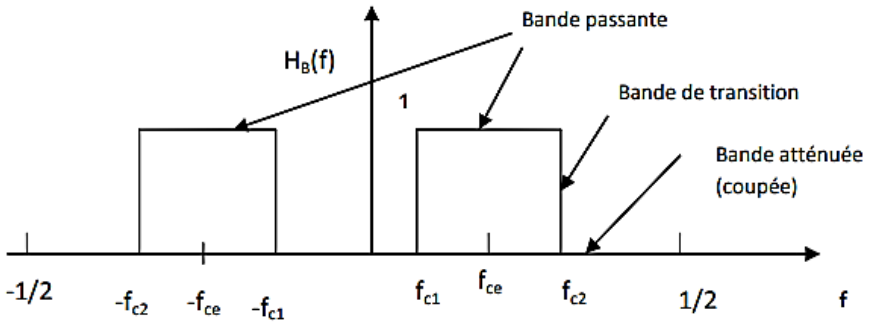

Figure1.6. Filtre passe bande [3].

1.4. Classification des filtres numérique

Les filtres numériques peuvent être classés de différentes manières en fonction de leurs caractéristiques et de leurs propriétés. Voici quelques classifications courantes des filtres numériques.

1.4.1. Filtre RIF

Les filtres numériques à réponse impulsionnelle finie sont caractérisés par une réponse impulsionnelle basée sur un nombre limité d'échantillons d'un signal, ils sont cependant très largement utilisés car ils possèdent des propriétés uniques (stabilité, flexibilité,…), ces filtres sont aussi connus par nom filtres non récursif.

De façon générale le filtre à réponse impulsionnelle fini est décrit par la fonction de transfert (1.12) et l'équation aux différences (1.13) suivantes **:** [4]

$$H(z) = \sum_{i=0}^{N-1} b_i \cdot z^{-i} \tag{1.12}$$

$$y(n) = \sum_{i=0}^{N-1} b_i \cdot x(n-i) = \sum_{i=0}^{N-1} h(i) \cdot x(n-i) \tag{1.13}$$

On remarque en exploitant l'´équation 1.12 que les coefficients bi du filtre sont également les valeurs de la réponse impulsionnelle h(n), qui se trouve donc être limitée dans le temps.

$$H(z) = \sum_{i=0}^{N-1} b_i \cdot z^{-i} \Leftrightarrow h(n) = \sum_{i=0}^{N-1} b_i . \delta(n-i) \quad (1.14)$$

$$h(n) = \begin{cases} b_n & \text{pour } 0 \leq n \leq N-1 \\ 0 & \text{ailleurs} \end{cases} \quad (1.15)$$

1.4.1.1. Les principales caractéristiques de filtre RIF

Les caractéristiques principales des filtres RIF sont les suivantes :

- Une bande de transition qui sera toujours plus large qu'un filtre RII ayant le même nombre de coefficients.
- Des méthodes de synthèse permettant de dériver n'importe quelle réponse fréquentielle.
- Une stabilité inhérente ($\sum_{n=0}^{N-1}|h(n)| < \infty$).
- Une plus grande stabilité numérique.
- Une phase qui peut-être exactement linéaire, par conséquent un temps de propagation de groupe constant et une absence de distorsion harmonique dans le signal.
- Une plus grande facilité d'implémentation dans un système numérique de traitement [5].

1.4.2. Filtre RII

On a vu dans la partie précédente qu'un filtre à réponse impulsionnelle finie est un filtre numérique ayant une forme très simple et qui s'implémente bien sur un microcontrôleur ou sur un PC. Néanmoins, pour effectuer des fonctions de filtrages complexes (avec un ordre élevé), le nombre de coefficient à déterminer et le nombre d'échantillon à stocké peut devenir

vite important. Pour ce faire, nous allons voir le filtre à réponse impulsionnelle infini (filtre RII) qui fait entrer en jeu les résultats des filtrages des échantillons précédents.

Nous posons donc directement la formule du filtre RII :

$$y(n) = \sum_{i=0}^{N} \left(b_i \cdot x(n-i)\right) - \sum_{j=1}^{M} \left(a_j \cdot y(n-j)\right) \quad (1.17)$$

Nous pouvons remarquer que pour calculer la valeur de l'échantillon après filtrage, nous faisons la somme pondéré des N échantillons précédent (jusque-là, rien ne change), mais nous rectifions cette somme par la somme des M résultats précédemment obtenus. Implicitement, cette formule rajoute de l'information contenue dans l'ensemble des échantillons depuis le premier jusqu'à l'échantillon courant.

Ce filtre est une formule récursive, on peut la considérer comme une discrétisation d'une équation différentielle linéaire [6].

1.4.2.1. Les principales caractéristiques de filtre RII

Les caractéristiques principales des filtres RII sont les suivantes :

-une bande de transition qui peut être étroite.

-une instabilité potentielle due `à des pôles de H (z) situ ´es en dehors du cercle unité .

-une complexité plus faible qu'un RIF `à s´électivité ´équivalente.

- une plus grande sensibilité numérique (quantification des coefficients, bruits de calculs) [5].

1.5. Structure des filtres RII et RIF

1.5.1. Structure de filtre RII

Le filtre RII peut être réalisé par deux structures : la structure direct (voir figure 1.7) et la structure cascade (voir figure 1.8) [4].

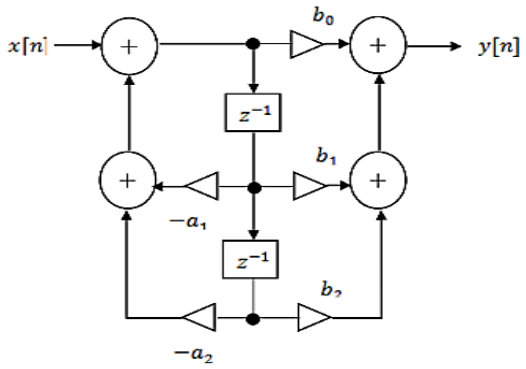

Figure 1.7 : Structure direct d'un filtre RII.

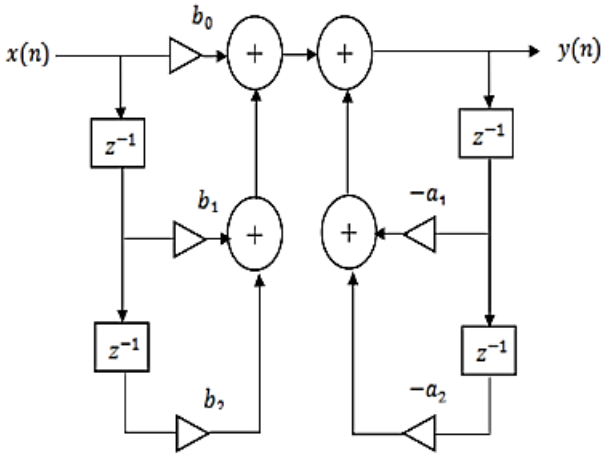

Figure 1.8 : Structure cascade d'un filtre RII

1.5.2. Structure de filtre RIF

Le filtre RIF peut être réalisé par deux façons : la structure directe (Voir figure 1.9) et la structure transposée (Voir figure 1.10) [4].

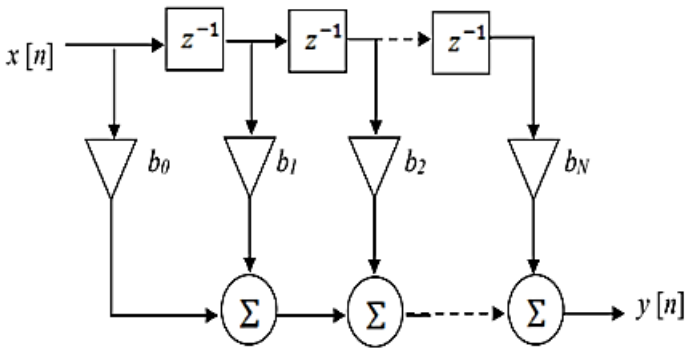

Figure 1.9. Structure directe d'un filtre RIF.

Ces structures nécessites N − 1 cases mémoire et a une complexité de calcul de Multiplications et N − 1 additions par échantillons de sortie.

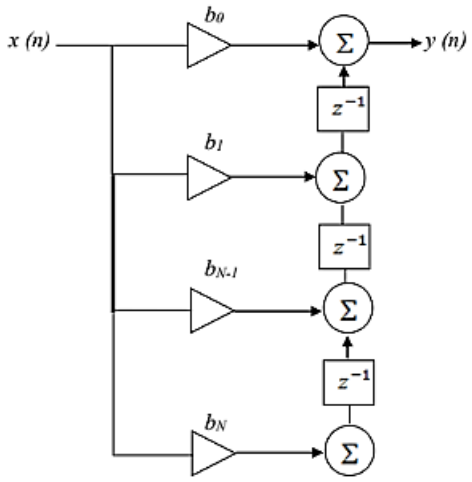

Figure 1.10 : Structure transposée d'un filtre RIF.

1.6. Synthèse des filtres RIF

On cite deux méthodes de synthèse de ce type de filtre :

1.6.1. Synthèse par la méthode de fenêtrage :

Cette méthode permet de calculer, à partir d'une réponse fréquentielle idéale, les coefficients du filtre RIF. Des fenêtres sont utilisées afin de limiter l'intervalle de la réponse impulsionnelle du filtre numérique [7].

1.6.1.1. Etape de synthèse par la méthode de fenêtrage

Les étapes qui interviennent dans la méthode de fenêtrage sont [7]:

Entrées : A synthétiser un filtre désiré d'ordre N, et de gabarit spécifié.

Etape 1 : A partir d'un gabarit idéal H(ω), on calcule la réponse impulsionnelle h(k).

$$h(k) = TF^{-1}H(\omega) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)e^{j\omega k}d\omega \qquad (1.18)$$

Cette réponse impulsionnelle est à durée infinie et non causale.

Etape 2 : La réponse impulsionnelle de durée finie $h_f(k)$ est calculée, le produit entre la fenêtre et la réponse impulsionnelle h(k) permet de limiter sa durée :

$$h_f(k) = h(k)w(k) \tag{1.19}$$

Où : w(k) est la fenêtre définie dans le domaine [-m, m] et N=2m.

Avec : m : longueur de filtre et N l'ordre du filtre.

Etape 3 : Le décalage temporel de la réponse impulsionnelle obtenue en étape 2, pour avoir un filtre causal.

$$h(k) = h_f(k-m) = h(k-m)w(k-m) \tag{1.20}$$

Etape 4 : Obtention de la fonction de transfert :

$$H(z) = \sum_{k=0}^{2m} h(k)z^{-k} \tag{1.21}$$

1.6.1.2. Différentes fenêtres : [8]

— Bartlett (ou triangulaire)

$$w(n) = \begin{cases} \frac{2}{N-1} & 0 \le n \le \frac{N-1}{2} \\ 2 - \frac{2n}{N-1} & \frac{N-1}{2} \le n \le N-1 \\ 0 & \text{sinon} \end{cases} \tag{1.22}$$

— Hanning

$$w(n) = \begin{cases} 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) & 0 \le n \le N-1 \\ 0 & \text{sinon} \end{cases} \tag{1.23}$$

— Hamming

$$w(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) & 0 \leq n \leq N-1 \\ 0 & \text{sinon} \end{cases} \quad (1.24)$$

— Blackman

$$w(n) = \begin{cases} 0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right) & 0 \leq n \leq N-1 \\ 0 & \text{sinon} \end{cases} \quad (1.25)$$

On pourra se reporter à la figure (1.12) pour une représentation temporelle de ces fenêtres, et à la figure (1.11) pour leur représentation spectrale.

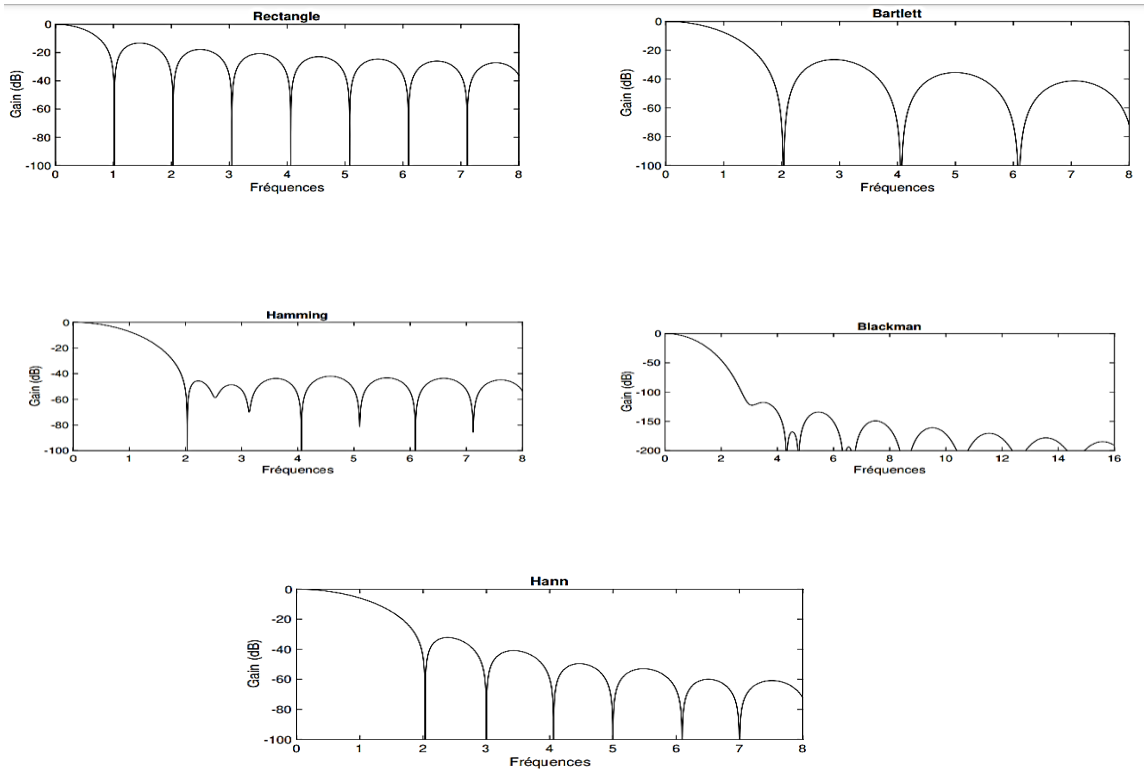

Figure 1.11. Représentation Spectral des fenêtres classiques.

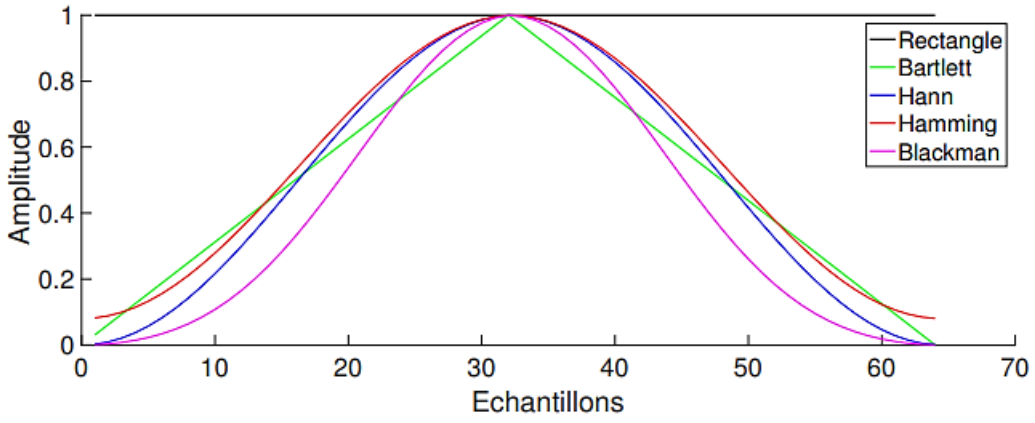

Figure 1.12. Représentation temporel des fenêtres classiques.

1.6.2. Synthèse par Méthode de l'échantillonnage fréquentiel

Lorsqu'une réponse impulsionnelle h(k) est de durée finie N, on sait que l'on peut échantillonner son spectre H(f) (ou H(w)) en au moins N points sans provoquer de repli dans la bande de base. Cette méthode de synthèse emploie la Transformée de Fourier Discrète inverse [9].

$$H(n) = H(f)|_{f=\frac{n}{N}}, \text{ pour } n = -\frac{N}{2}, \frac{N}{2} + 1 \tag{1.26}$$

La transformée de Fourier Discrète inverse fournit h(k) :

$$h(k) = \begin{cases} \frac{1}{N}\sum_{n=-N/2}^{N/2+1} H(n)e^{j2\pi\frac{n}{N}k} \\ 0. \text{ ailleurs.} \end{cases} \tag{1.27}$$

Cette méthode de synthèse ne garantit que les points fréquentiels H(n). Entre ces points, la valeur de H(f) n'est pas maîtrisée.

La structure de réalisation est alors obtenue par la transformée en z :

$$H(z) = \sum_{k=0}^{N-1} h(k)z^{-k} \text{ (Signal causal)} \tag{1.26}$$

$$H(z) = \sum_{k=-N/2}^{N/2-1} h(k)z^{-k} \text{ (Signal non causal)} \tag{1.27}$$

1.7. Conclusion

En conclusion, ce chapitre a cherché à fournir une introduction claire et concise au filtrage numérique, en couvrant les principes de base, les techniques de conception et les applications pratiques. Nous avons examiné les propriétés des filtres numériques , ainsi que leurs différents types et leur classification en tant que filtres à réponse impulsionnelle finie (RIF) ou à réponse impulsionnelle infinie (RII). Dans cette optique, nous sommes concentrés particulièrement sur les filtres RIF en raison de leur stabilité et de leur réalisabilité. De plus, nous avons également abordé les méthodes de synthèse de ces filtres, permettant ainsi de mettre en pratique les connaissances acquises. Grâce à ces méthodes, il est possible de concevoir et d'implémenter des filtres numériques adaptés aux spécifications requises. En explorant ces concepts, nous espérons que ce chapitre a pu vous fournir une base solide pour comprendre les fondements du filtrage numérique.

CHAPITRE 2

Les circuits FPGA et outils de conception

2.1. Introduction

Le calcul numérique fait référence à l'utilisation d'algorithmes mathématiques pour résoudre des problèmes complexes dans divers domaines tels que la finance, la physique, la chimie, la biologie et l'ingénierie. Il englobe l'application de méthodes numériques qui permettent d'effectuer des calculs sur des données numériques en utilisant des processeurs informatiques. L'un des dispositifs puissants de calcul est le FPGA. Un FPGA (Field-Programmable Gate Array) est un circuit intégré programmable, qui permet aux ingénieurs et développeurs de configurer son fonctionnement après sa fabrication. Contrairement aux circuits intégrés classiques, qui sont conçus pour une fonction spécifique, les FPGAs offrent une flexibilité en permettant la reprogrammation de la logique interne pour s'adapter à différentes applications.

Les FPGAs sont composés de blocs de logique programmable (comme des portes logiques, des registres, des multiplexeurs, etc.) ainsi que de connexions interconnectant ces blocs. La configuration de ces blocs et de leurs connexions peut être modifiée en utilisant des langages de description matérielle tels que VHDL (VHSIC Hardware Description Language) ou Verilog. Cela permet aux concepteurs de créer des circuits numériques personnalisés répondant aux besoins spécifiques d'une application.

2.2. Architecture de base d'un FPGA

Les circuits FPGA possèdent une structure matricielle de deux types de blocs (ou Cellules). Il y'a les blocs d'entrées/sorties (IOB; Input Output Bloc) et les blocs logiques programmables (CLB; Configurable Logic Bloc), comme le montre la figure 2.1 Le passage d'un bloc logique à un autre se fait par un réseau d'interconnexions programmable.

2.2.1. Les blocs logiques configurables

Les premiers FPGAs de Xilinx étaient basés sur le concept de de Configurable Logic Block (CLB), ce dernier contient une LUT de 3 entrées (Look Up table), une bascule, un multiplexeur et plusieurs autres éléments, mais ceux-ci sont typiquement les plus importants [10].

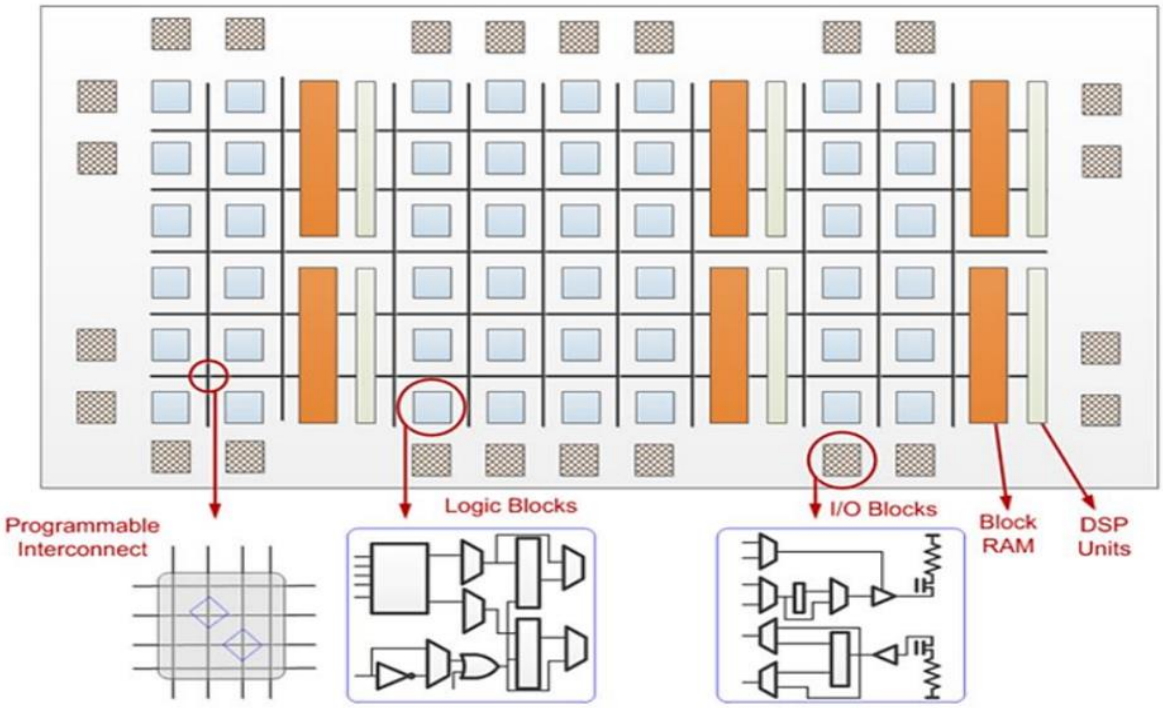

Figure 2.1. Architecture interne d'un FPGA.

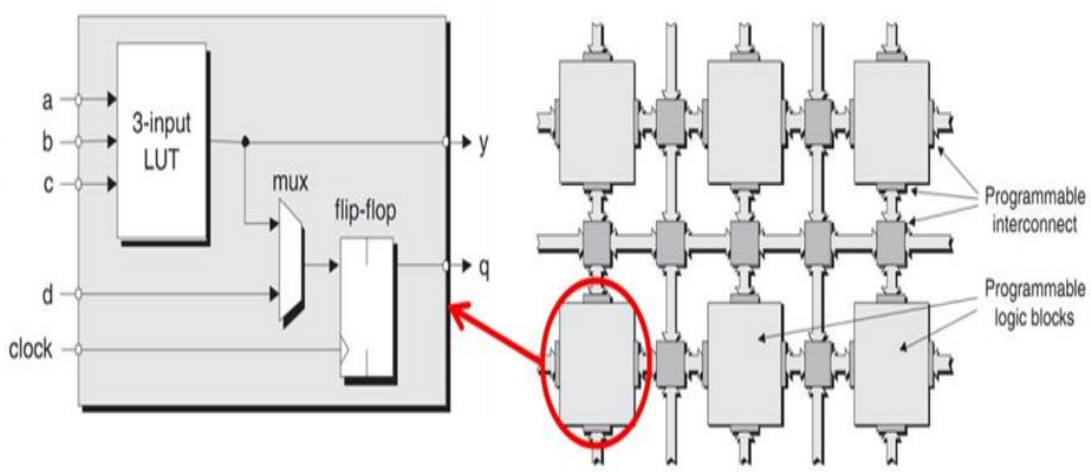

Figure 2.2. Schéma simplifié d'un bloc logique configurable.

Chaque FPGA contint un grand nombre des CLBs (iles) intégrés dans une (mer) d'interconnexion programmable, La fonction de la LUT est de stocker la table de vérité de la fonction combinatoire à implémenter dans les cellules mémoires.

2.2.2.1. La LUT

La LUT (Look Up Table) ou table de correspondance sert à implémenter des équations logiques ayant généralement 4 à 6 entrées et une sortie. Elle peut toutefois être considérée comme une petite mémoire, un multiplexeur ou un registre à décalage. Le registre permet de mémoriser un état (machine séquentielle) ou de synchroniser un signal (pipeline).

Les LUTs sont disposés en tant que cellules de mémoire SRAM, chaque cellule stockant un bit de la sortie de la table de vérité, par conséquent, une LUT de quatre entrées doit avoir 16 cellules de mémoire. En recherchant l'entrée (d'où le nom de la table de correspondance), la LUT délivre la sortie correspondante. En cascadant plusieurs LUTs, plusieurs tranches et plusieurs CLBs, n'importe quel circuit logique combinatoire ou séquentiel de n'importe taille peut être implémenté.

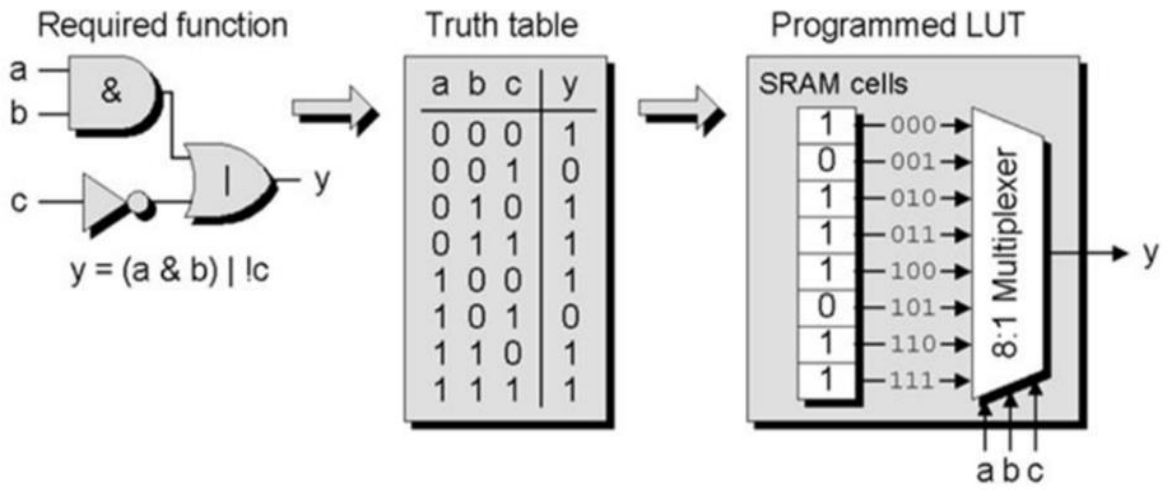

Figure 2.3. Schéma montrant le principe de fonctionnement d'un LUT.

2.2.2. Les blocs d'entrée/sortie

Les blocs d'entrée/sortie ou IOB (Input Output Bloc) permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signal bidirectionnel ou être inutilisé (état haute impédance).

Les IOB peuvent s'adapter à un grand nombre de standards de communication : LVTTL, LVCMOS, PCI, LVDS, LVPECL, etc. Ils sont regroupés en banques, et chaque banque à sa propre tension d'alimentation. Ceci permet de combiner des standards incompatibles entre eux sur le même circuit FPGA en utilisant des banques différentes [11].

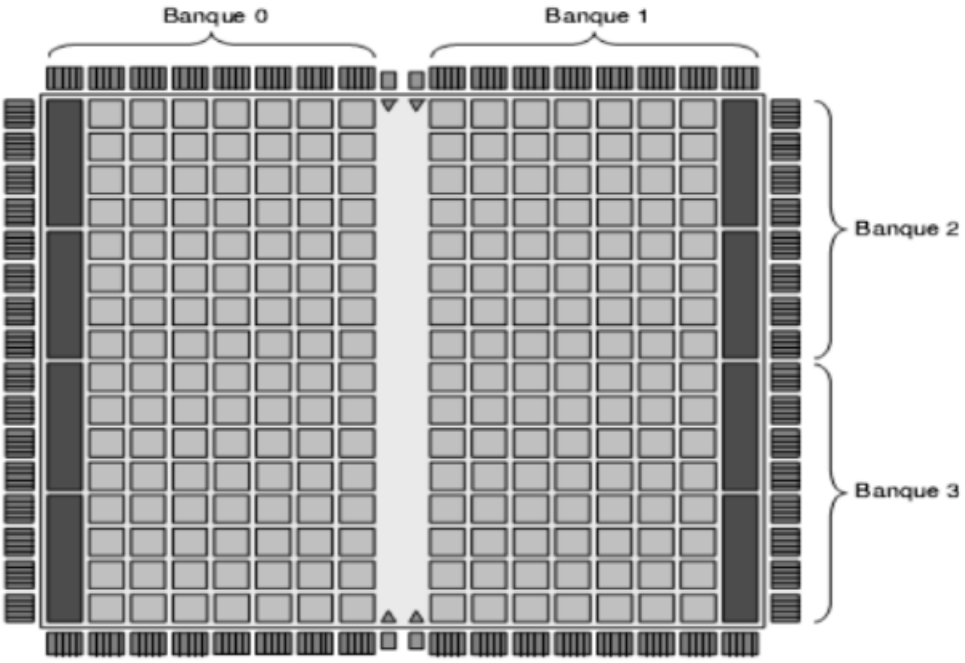

Figure 2.4. Distribution des blocs d'entrée /sortie (IOB) en banques.

2.2.3. Le réseau d'interconnexions dans un circuit FPGA

Le réseau d'interconnexions permet la réalisation des connexions (le routage) entre les différentes tranches au sein d'un même CLB, entre les différents CLBs, les DCMs, les multiplicateurs, la mémoire RAM et les IOBs utilisés pour la construction d'un circuit donné. Au sein d'un même CLB, les différentes tranches peuvent communiquer via des connexions rapides. Une matrice des connexions programmables (Programmable Switch Matrix- PSM) est associée à chaque CLB, IOB, DCM, à la mémoire RAM et au circuit multiplicateur. Le rôle de cette matrice est de réaliser une connexion programmée entre les conducteurs "entrants" et "sortants". La programmation de la connexion est établie à l'aide d'un mot de mémoire défini par la configuration. Grâce aux PSMs toute ressource peut donc accéder à

toute autre ressource, quel que soit l'endroit où elle se trouve dans le circuit FPGA. Les conducteurs "entrants" et "sortants" d'une matrice PSM sont disposés entre les lignes et les colonnes des CLBs [11].

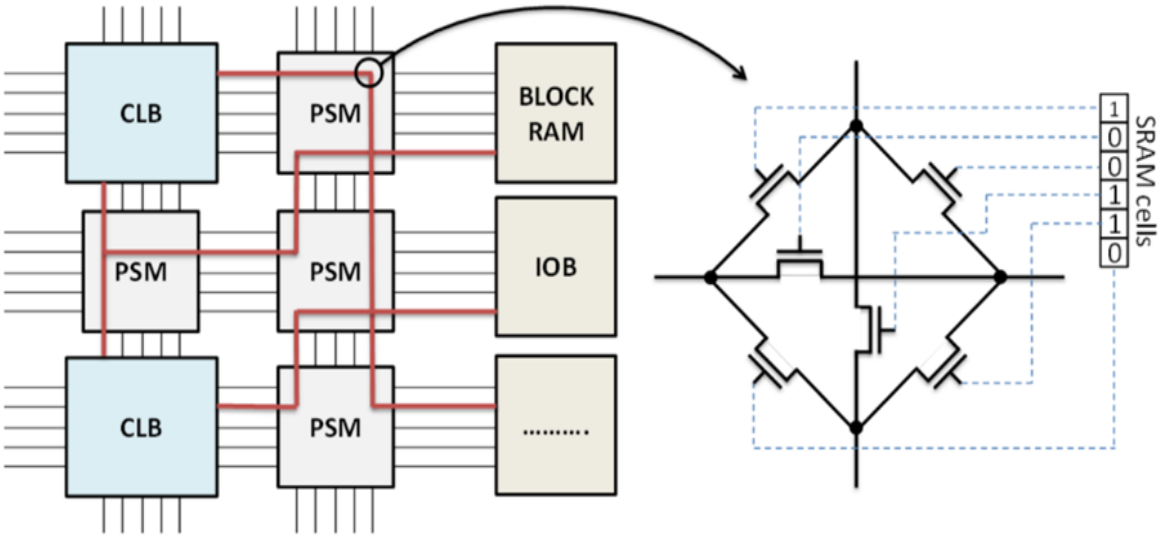

Figure 2.5. Schéma simplifié d'une matrice PSM [11].

2.2.4. Types d'interconnexion pour la configuration

La configuration d'un FPGA (Field-Programmable Gate Array) se réfère au processus de définir le comportement du circuit en programmant les blocs logiques et les connexions internes [12]. Il existe plusieurs méthodes de configuration utilisées dans les FPGAs, telles que les fusibles (fuse), les anti fusibles (antifuse) et la mémoire SRAM (Static Random-Access Memory) :

- Les fusibles sont des composants programmables utilisés dans certains FPGAs. Ils sont initialement connectés en série et, lorsque programmés, certains fusibles sont coupés (ou "brûlés") pour ouvrir des connexions spécifiques, ce qui modifie la fonctionnalité du circuit. Cette méthode est souvent utilisée pour les FPGAs de première génération.

- Les antis fusibles, en revanche, sont des composants programmables qui sont normalement fermés et s'ouvrent lorsqu'ils sont programmés. Cela permet de créer des connexions spécifiques pour configurer le FPGA. Les anti-fusibles offrent une programmabilité plus fiable et permanente par rapport aux fusibles.
- La mémoire SRAM est une autre méthode couramment utilisée pour la configuration des FPGAs modernes. Les FPGAs disposent d'une mémoire SRAM intégrée, qui stocke la configuration du circuit. Lors de la mise sous tension, la mémoire SRAM est chargée avec la configuration depuis un périphérique externe, tel qu'une mémoire Flash ou une mémoire EEPROM. Cette méthode offre une plus grande flexibilité car la configuration peut être modifiée à tout moment [13].

2.3. Les circuits FPGA de la gamme Xilinx

Xilinx (nom complet Xilinx, Inc.) est une entreprise américaine de semi-conducteurs. Inventeur du FPGA, Xilinx fait partie des plus grandes entreprises spécialisées dans le développement et la commercialisation de composants logiques programmables, et des services associés tels que les logiciels de CAO électroniques, blocs de propriété intellectuelle réutilisables et formation. Xilinx a été fondée en 1984 par Ross Freeman ; Bernie Vonderschmitt et Jim Barnett, dont le plan d'entreprise se résumait à la commercialisation de composants électroniques basé sur un concept alors nouveau : celui de la logique programmable.

Aujourd'hui, Xilinx propose une large gamme de circuits FPGA (Field-Programmable Gate Arrays) adaptés à différentes applications et besoins. Voici quelques-uns des circuits FPGA les plus connus de Xilinx :

1. Série Virtex : Les circuits FPGA de la série Virtex offrent des performances haut de gamme, une capacité d'intégration élevée et des fonctionnalités avancées pour les applications les plus exigeantes.
2. Série Kintex : Les circuits FPGA de la série Kintex offrent un équilibre entre performances élevées, consommation d'énergie réduite et coût compétitif, adaptés à une large gamme d'applications.
3. Série Artix : Les circuits FPGA de la série Artix offrent une performance élevée à un

coût abordable, adaptés aux applications grand public, industrielles et médicales.

4. Série Spartan : Les circuits FPGA de la série Spartan offrent une combinaison de performances, de flexibilité et de coût compétitif, adaptés à une variété d'applications.
5. Série Zynq : Les circuits FPGA de la série Zynq intègrent des cœurs de processeur ARM avec des ressources FPGA, offrant une plate-forme puissante pour les applications de traitement de signal, de vision par ordinateur et d'embarqué.

Il convient de noter que Xilinx propose également d'autres séries de circuits FPGA, telles que les séries CoolRunner et Versal, ainsi que les dernières architectures telles que les systèmes ACAP (Adaptive Compute Acceleration Platform).

Chaque série de circuits FPGA de Xilinx offre des caractéristiques, des performances et des capacités spécifiques. Pour plus de détails sur les circuits FPGA de Xilinx voir [14].

2.4. Avantages des circuits FPGA

Les FPGA offrent plusieurs avantages par rapport à d'autres types de circuits, tels que les ASIC (Application-Specific Integrated Circuits) ou les microcontrôleurs. Voici quelques-uns de ces avantages, avec des références pour approfondir :

- Flexibilité : Les FPGA sont programmables et reconfigurables, ce qui permet de les adapter à différentes fonctionnalités et applications. Cette flexibilité permet aux concepteurs de prototyper rapidement des systèmes, de les mettre à jour et de les adapter aux besoins changeants. [15].
- Puissance de traitement parallèle : Les FPGA peuvent exécuter plusieurs tâches simultanément grâce à leur architecture parallèle. Cela les rend efficaces pour le traitement de données massives et les applications nécessitant un haut débit de calcul. [16].
- Faible latence : Les FPGA offrent une latence réduite par rapport à d'autres circuits, tels que les microcontrôleurs, car ils exécutent les tâches directement en matériel plutôt qu'en utilisant un système d'exploitation. Cela les rend adaptés aux applications nécessitant des réponses en temps réel. [17].

- Adaptabilité : Les FPGA peuvent être reprogrammés pour s'adapter à de nouvelles exigences et fonctionnalités. Cela permet de prolonger la durée de vie d'un système et d'effectuer des mises à jour sans avoir à remplacer complètement le circuit.
- Prototypage rapide : Les FPGA permettent un développement et une validation rapides des conceptions matérielles, ce qui réduit le temps de mise sur le marché des produits. Les concepteurs peuvent itérer rapidement et effectuer des ajustements en fonction des besoins du projet [18].

2.5. Domaines d'application des FPGA

Les FPGA sont utilisés dans une variété de domaines d'application en raison de leur flexibilité, de leurs capacités de traitement parallèle et de leur adaptabilité. Voici quelques-uns des domaines d'application courants des FPGA (entre autres) :

- Télécommunications : Les FPGA sont utilisés dans les infrastructures de télécommunications pour des applications telles que le traitement des signaux, la compression vidéo, la modulation/démodulation, les interfaces de communication et la gestion du trafic [19].
- Aérospatiale et défense : Les FPGA sont largement utilisés dans l'aérospatiale et la défense pour des applications telles que le traitement radar, les communications sécurisées, les systèmes de navigation, le traitement d'images, la surveillance et le contrôle des drones [20].
- Systèmes embarqués : Les FPGA sont utilisés dans les systèmes embarqués pour la gestion des interfaces, le contrôle des périphériques, le traitement des signaux en temps réel, la fusion de capteurs, la robotique et les applications industrielles [21].
- Médical : Les FPGA sont utilisés dans les équipements médicaux pour le traitement des signaux biomédicaux, l'imagerie médicale, les systèmes de surveillance, les dispositifs de diagnostic et les dispositifs d'assistance à la vie [22].
- Automobile : Les FPGA sont utilisés dans les systèmes de conduite autonome, les systèmes de navigation, les systèmes de contrôle du moteur, les systèmes de vision et les applications de sécurité automobile [23].

2.6. Le VHDL

Le VHDL (VHSIC Hardware Description Language) est un langage de description matérielle largement utilisé dans l'industrie électronique pour concevoir et spécifier le comportement des circuits intégrés. Le VHDL a été développé à l'origine par le Département de la Défense des États-Unis pour répondre aux besoins de conception des systèmes VHSIC (Very High-Speed Integrated Circuits).

Le VHDL permet aux concepteurs de décrire les fonctionnalités et le comportement des circuits électroniques en utilisant une syntaxe textuelle structurée. Il offre une approche modulaire pour la conception de systèmes complexes, permettant de décrire les composants, les signaux, les connexions et les comportements temporels [24].

L'un des avantages du VHDL est sa capacité à représenter à la fois le comportement et la structure d'un circuit, ce qui facilite la simulation, la vérification et la génération automatique de code.

En utilisant le VHDL, les concepteurs peuvent définir les fonctionnalités du circuit, simuler son comportement, effectuer des vérifications de conformité et générer automatiquement le code de configuration pour les circuits FPGA.

Le VHDL est largement utilisé dans l'industrie électronique et est pris en charge par de nombreux outils de conception, notamment ceux proposés par des sociétés telles que Xilinx, Altera (maintenant Intel) et Lattice Semiconductor.

2.7. Outil de conception

Dans cette partie on se limite la discussion aux outils de conception de la gamme Xilinx. Xilinx propose plusieurs outils de conception pour faciliter le développement et la programmation des circuits FPGA. On présente quelques-uns des outils les plus couramment utilisés de Xilinx.

2.7.1. Vivado Design Suite

Vivado Design Suite est l'environnement de conception principal de Xilinx, offrant une suite complète d'outils pour la conception, la vérification, la synthèse et la mise en œuvre

des circuits FPGA. Vivado Design Suite est utilisé pour créer des conceptions matérielles, configurer les circuits FPGA et générer les fichiers de programmation.

Vivado a été introduit en avril 2012 et est un environnement de conception intégré (IDE) avec des outils de niveau système vers CI (circuit intégré) basés sur un modèle de données évolutif partagé et un environnement de débogage commun. Vivado comprend des outils de conception de niveau système électronique (ESL) pour la synthèse et la vérification d'IP (propriété intellectuelle) basées sur le langage C ; l'emballage conforme aux normes à la fois de l'IP algorithmique et RTL (Register Transfer Level) pour la réutilisation ; l'assemblage conforme aux normes de l'IP et l'intégration des systèmes de tous types de blocs de construction de systèmes ; ainsi que la vérification des blocs et des systèmes [26]

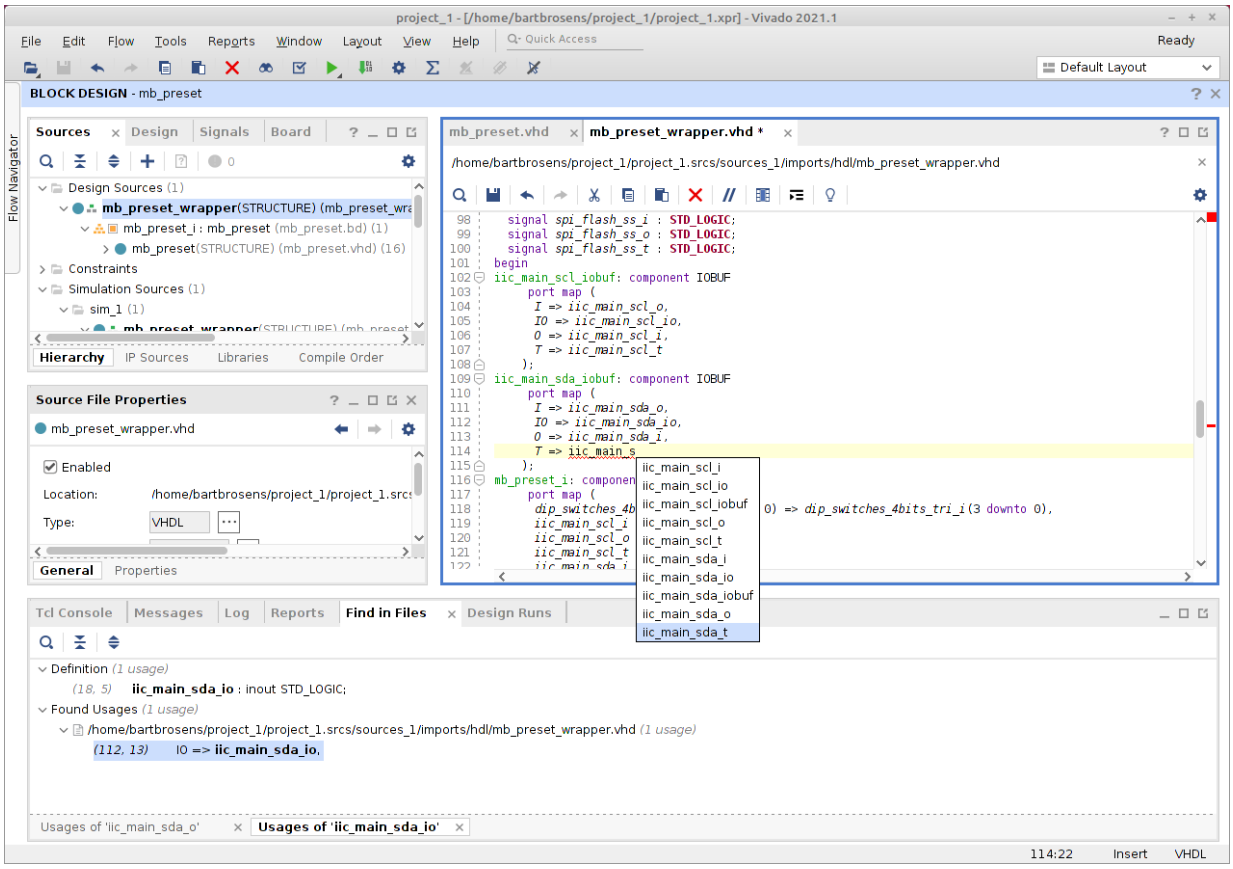

Figure.2.6. Vivado Design Suite (interface principal).

2.7.2. Xilinx ISE

Xilinx ISE : ISE (Integrated Software Environment) est un environnement de conception plus ancien, mais toujours utilisé par certains concepteurs FPGA. Il offre des

fonctionnalités de conception, de simulation, de vérification et de mise en œuvre pour les circuits FPGA Xilinx. ISE prend en charge plusieurs gammes de FPGA de Xilinx des générations anciennes, notamment Spartan-3, Spartan-6 et Spartan-7, Virtex-4, Virtex-5, Virtex-6, Virtex-7, etc.

2.7.3. PetaLinux

C'est un environnement de développement basé sur Linux spécifiquement conçu pour les systèmes embarqués sur les FPGA Xilinx. PetaLinux permet de créer et de personnaliser des systèmes d'exploitation Linux pour les applications FPGA.

2.7.4. Vitis

Environnement de développement unifié pour Xilinx Zynq, MPSoC, RFSoC, et ACAP. Cet outil permet de développer les parties logicielles et matérielles en utilisant des langages adaptés aux ingénieurs logiciels: C, C++, Python. Quelle que soit la cible, composant embarqué ou carte accélératrice à base de composant Xilinx intégrée dans un PC hôte, Vitis permet le développement et le déploiement d'applications utilisant toutes les capacités des composants Xilinx. Vitis dispose de toute une série de librairies 26 couvrant un large choix de domaines : Intelligence artificielle, traitement d'images, finance, sécurité, maths.

2.7.5. Vitis Model composer

Vitis Model Composer est un outil de conception basé sur les modèles qui permet une exploration rapide de la conception dans l'environnement MathWorks MATLAB® et Simulink®, et accélère le passage à la production sur les appareils AMD grâce à la génération automatique de code.

On peut concevoir des algorithmes DSP et itérer à travers eux en utilisant des blocs optimisés au niveau des performances et valider la correction fonctionnelle grâce à des simulations au niveau système. Vitis Model Composer transforme la conception en une implémentation de qualité de production grâce à des optimisations automatiques.

L'outil propose une bibliothèque de plus de 200 blocs HDL, HLS et AI Engine pour la conception et la mise en œuvre d'algorithmes sur les dispositifs Xilinx (AMD maintenant). Il permet également d'importer du code HDL, HLS et AI Engine personnalisé en tant que blocs dans l'outil [27].

2.8. Flot de conception

Chaque constructeur FPGA fournit ses propres outils de conception assistée par ordinateur (CAD : Computer-Aided design) pour aider les utilisateurs à concevoir leurs FPGA. Le flux de conception générique d'un FPGA comprend les étapes suivantes:

- Synthèse logique: C'est la phase où le concepteur décide quelle méthode de conception est appropriée à sa conception; en utilisant un langage de description matériel tel que le VHDL, une bibliothèque schématique ou un outil de conception de haut niveau tel que vitis Model composer [28].
- Simulation fonctionnelle: C'est l'étape de vérification du bon fonctionnement de la conception ou du code HDL écrit à l'aide d'un simulateur approprié tel qu'ISIM de Xilinx ou Modelsim de Mentor Graphics.
- Optimisation: C'est une étape importante dans la conception des FPGAs, ce processus est la projection de la conception sur le circuit physique, il a deux opérations principales; placement et routage. L'opération de placement détermine les blocs logiques au sein d'un FPGA pour ce qui devrait être utilisé pour implémenter le circuit. Il minimise intelligemment le câblage requis et maximise la vitesse du circuit. Une fois que les emplacements pour tous les blocs logiques dans un circuit ont été choisis, un routeur attribue les réseaux du circuit aux segments de routage sur le FPGA et détermine quels commutateurs programmables doivent être activés pour connecter les blocs logiques comme requis par le circuit [29].
- Simulation temporelle: Le but de cette étape est de calculer tous les retards sur chaque bloc logique et les liaisons filaires, puis fournir un rapport du temps (timing report) indiquant le débit que le design peut atteindre par rapport au matériel utilisé.
- Génération de Bitstream: C'est la dernière étape qui consiste à générer le fichier binaire pour configurer le FPGA.

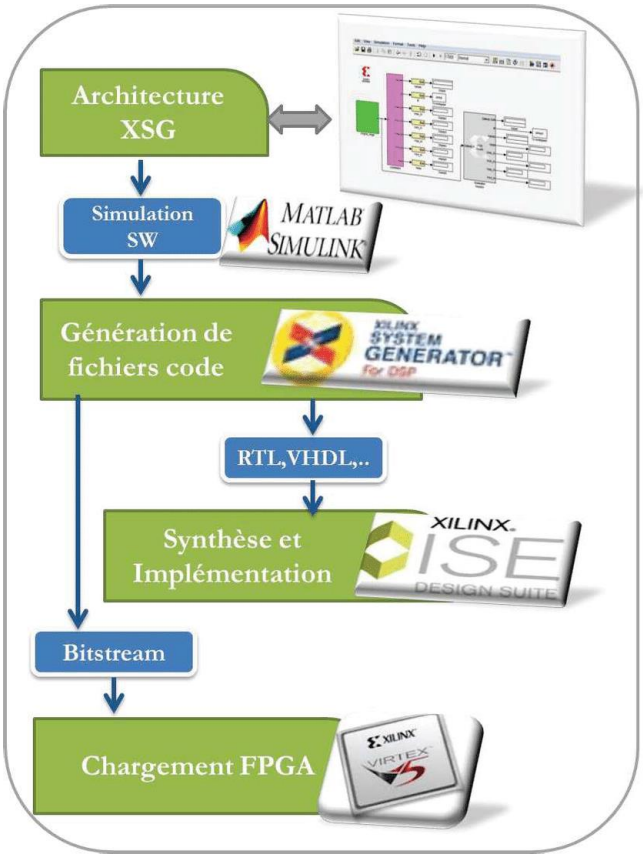

Figure.2.7. Flot de conception pour la configuration d'un FPGA.

2.9. Conclusion

Dans ce chapitre, nous avons exploré l'architecture des FPGA, en mettant en évidence les caractéristiques clés qui les distinguent des autres circuits intégrés programmables. Nous sommes concentrés sur les circuits FPGA de Xilinx, une entreprise leader dans ce domaine, et avons examiné en détail leur architecture interne, et comprit les blocs logiques configurables, les mémoires programmables et les interconnexions. En ce qui concerne les applications des FPGA, nous avons souligné leur polyvalence et leur adaptabilité, qui en font des composants essentiels dans divers domaines tels que les télécommunications, l'aérospatiale et la défense, les systèmes embarqués, le domaine médical et l'automobile. Des références spécifiques ont été fournies pour approfondir chaque domaine d'application.

En outre, le chapitre a abordé brièvement le langage VHDL (VHSIC Hardware Description Language), un langage de description matériel couramment utilisé pour spécifier et concevoir les circuits FPGA. Nous avons discuté de la syntaxe et de la sémantique du VHDL, ainsi que de son rôle dans la représentation du comportement et de la structure des circuits électroniques. Enfin, nous avons exploré les outils de conception associés aux FPGA, en mettant en évidence les principaux outils proposés par Xilinx, tels que Vivado Design Suite, Xilinx ISE, PetaLinux et Vitis. Ces outils offrent des fonctionnalités avancées pour faciliter la conception, la vérification, la synthèse et la mise en œuvre des circuits FPGA.

CHAPITRE 3

Simulation et implémentation d'un filtre RIF sur le FPGA

3.1. Introduction

Ce chapitre se concentrera sur les étapes clés de la simulation et de l'implémentation d'un filtre numérique RIF dans un circuit FPGA. Tout d'abord, nous simulerons le filtre à l'aide du logiciel Simulink, qui intègre la bibliothèque de l'outil ***Vitis code composer***. Cette bibliothèque offre une sélection de blocs prédéfinis spécifiquement conçus pour la réalisation des systèmes de traitement et comprit les filtres RIF. La simulation consistera à filtrer des signaux sinusoïdaux altérés par un bruit, puis à filtrer un signal audio bruité. Ensuite, nous procéderons à la génération automatisée de codes VHDL à partir de l'environnement SIMULINK/Vitis Code Composer. Le projet exporté sera évalué dans l'outil VIVADO, où d'autres extensions seront ajoutées pour assurer l'interfaçage des différents modules nécessaires à l'évaluation en temps réel du filtre conçu (convertisseur analogique/numérique, numérique/analogique, carte son, etc.).

3.2. Outils de conception et implémentation

L'objectif de cette section est de présenter les différents composants et l'outil de conception nécessaires à l'implémentation en temps réel du filtre souhaité sur un FPGA. Ainsi, cinq éléments essentiels sont nécessaires à cet effet. Tout d'abord, un générateur de fonction est requis pour produire le signal analogique à filtrer. Ensuite, un convertisseur analogique/numérique est utilisé pour effectuer la conversion du signal analogique en signal numérique. Le FPGA joue ensuite le rôle du filtre lui-même. Un convertisseur numérique/analogique est ensuite utilisé pour convertir le signal numérique filtré en signal analogique. Enfin, un oscilloscope est utilisé pour visualiser les résultats en temps réel.

3.3. Bref présentation de la carte Zedboard

La carte ZedBoard (figure.3.1) est une carte de développement basée sur un FPGA (*Field-Programmable Gate Array*) conçue par Xilinx. Elle offre une plateforme polyvalente pour le développement et la mise en œuvre de projets électroniques et informatiques.

La carte ZedBoard est équipée d'un FPGA Xilinx Zynq-7000, qui combine un

processeur ARM Cortex-A9 double cœurs (PS) et une matrice logique programmable (PL) dans un seul composant. Cette combinaison permet une grande flexibilité et des performances élevées pour une variété d'applications.

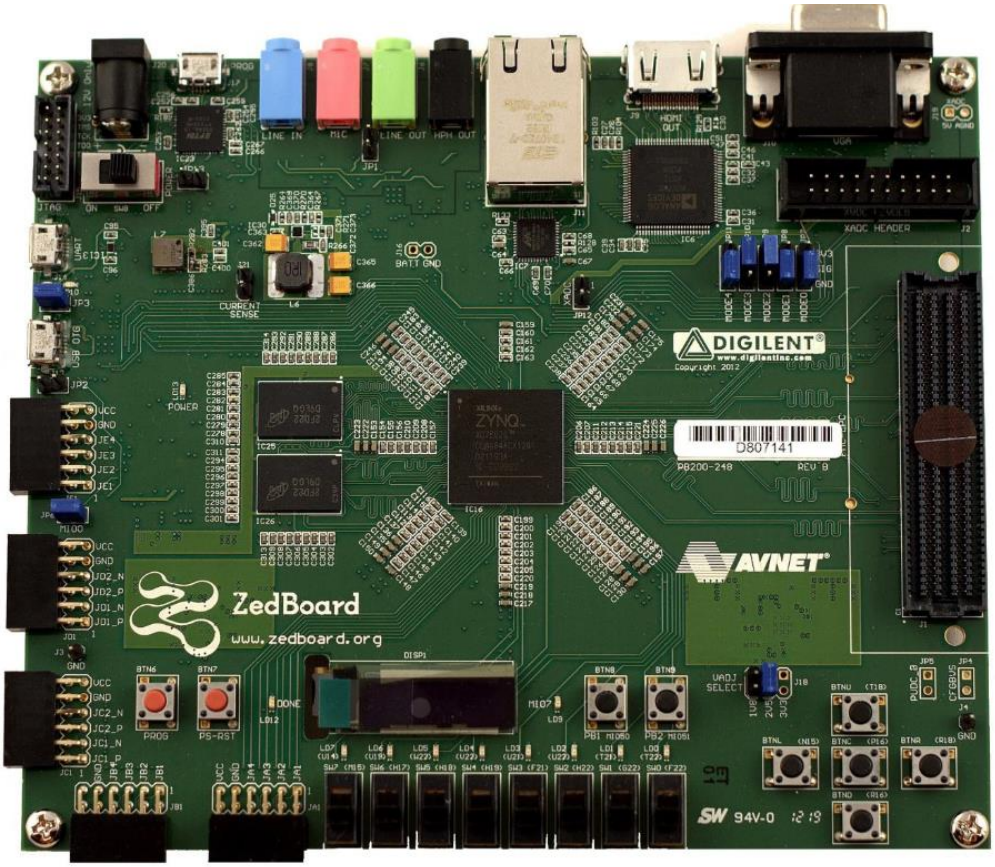

Figure.3.1. La carte d'évaluation ZedBoard.

La carte ZedBoard est dotée de connecteurs d'extension qui permettent de connecter différents périphériques et modules d'extension pour étendre les fonctionnalités de la carte. Elle offre également une connectivité Ethernet, USB, HDMI et d'autres interfaces courantes pour faciliter l'intégration dans des systèmes plus vastes. La ZedBoard est livrée avec une suite complète d'outils de développement, notamment le logiciel Vivado Design Suite de Xilinx, qui facilite la conception, la simulation et la programmation du FPGA. Il est également possible d'utiliser des langages de description matérielle tels que VHDL ou Verilog pour concevoir des circuits personnalisés.

Grâce à ses capacités de traitement puissantes, à sa flexibilité et à sa facilité d'utilisation, le ZedBoard est largement utilisé dans des domaines tels que l'embarqué,

l'informatique de pointe, la vision par ordinateur, l'audio-vidéo et bien d'autres encore.

Que ce soit pour l'apprentissage, la recherche ou le prototypage, la carte ZedBoard offre une plateforme robuste et performante pour développer et mettre en œuvre une large gamme de projets basés sur les FPGA.

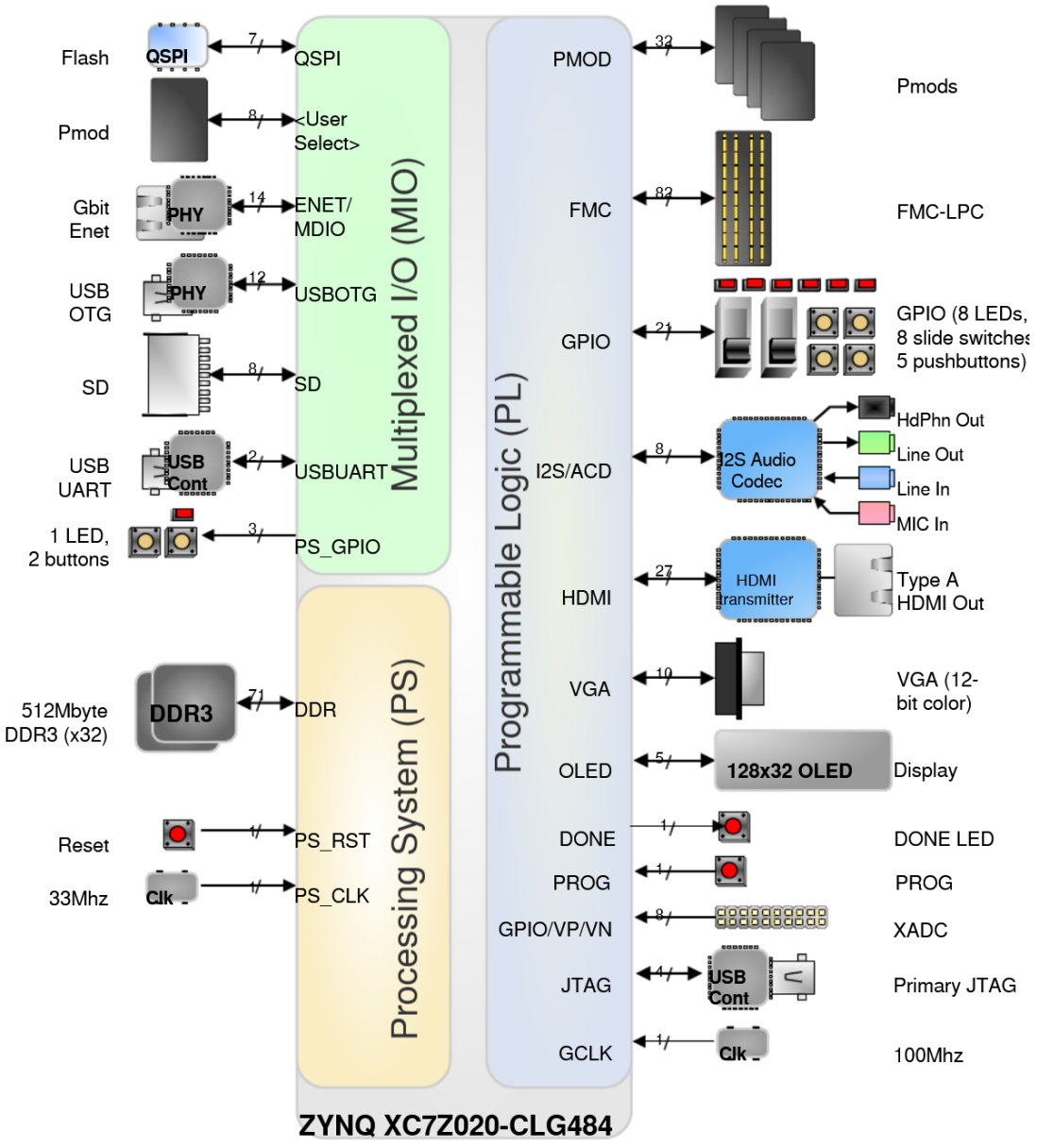

Figure.3.2. Diagramme de blocs de ZedBoard.

3.4. Les Modules ADC/DAC

Dans ce projet on va adopter deux modules pour assurer la conversion analogique-numérique et numérique-analogique, ils s'agissent des modules PMOD-AD1 et PMOD-DA2 (Peripheral Module) de DIGILENT Inc :

- **Le module PMOD-AD1** : Le module PMOD-AD1 comporte deux circuits convertisseurs Analogique/Numérique de type ADCS7476 [30] de Texas Instruments (figure.3.3) qui ont une résolution de 12 bits et supportent 1 Million d'échantillons par second.
- **Le module PMOD-DA2** : Le module PMOD-DA2 comporte deux circuits convertisseurs Numérique/Analogique de type DAC121S101 de *Texas Instruments* [31] (figure.3.4) qui ont une résolution de 12 bits.
-

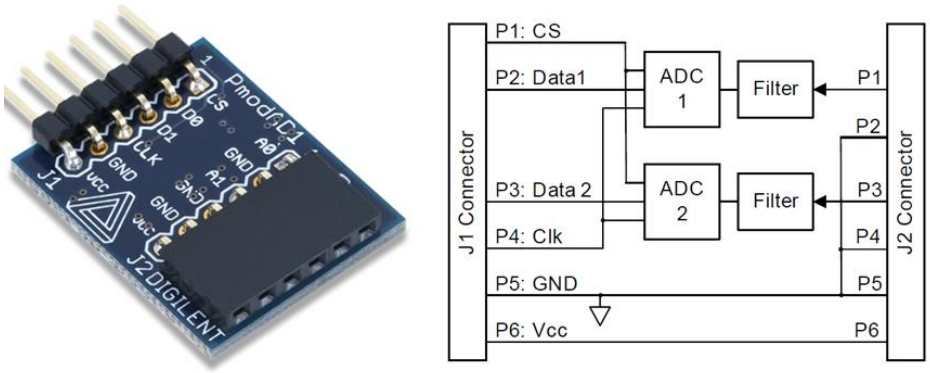

Figure.3.3. Le module PMOD-AD1 convertisseur Analogique/Numérique utilisé.

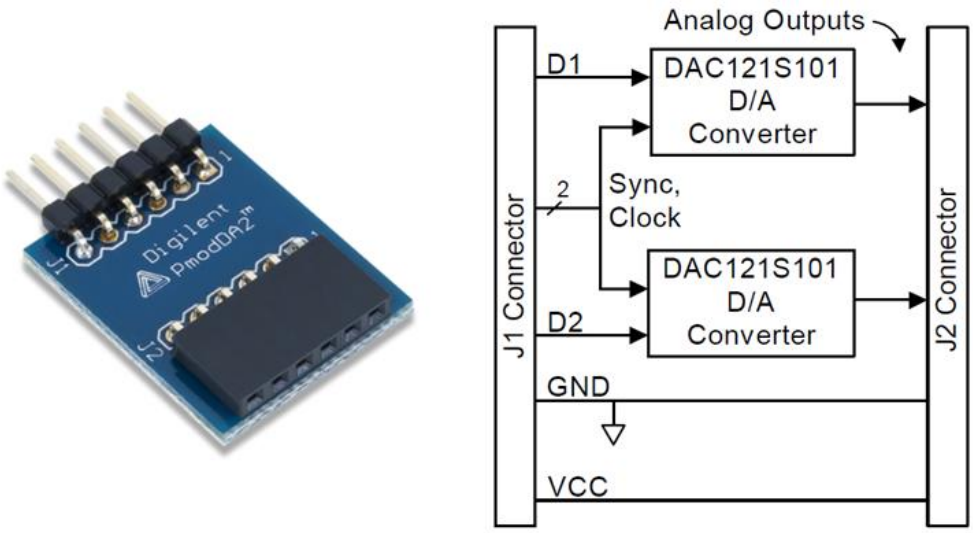

Figure.3.4. Le module PMOD-DA2 convertisseur Numérique/Analogique utilisé.

3.5. Conception et implémentation de filtre numérique

Xilinx Vitis Model Composer propose une bibliothèque complète de blocs préconfigurés et intégrés à l'environnement SIMULINK. Ces blocs peuvent être réutilisés pour concevoir des systèmes de traitement complexes sans nécessiter une connaissance préalable du langage de description matérielle.

Dans cette partie nous allons présenter les étapes de l'implémentation d'un filtre RIF passe-bas, l'implémentation sera effectuée suivant les étapes :

1- Création d'un projet sous SIMULINK et l'utilisation du bloc IP '*FIR Compiler 7.2*' à partir de la bibliothèque Vitis Model Composer.
2- La configuration de bloc Filtre rif on désignant quelques paramètres y compris la fréquence d'échantillonnage, la fréquence de coupure, l'ordre, etc.
3- La simulation de filtre par le filtrage des signaux ordinaires comme un signal sinusoïdale bruité, et des signaux audios altérés.
4- Après la simulation sous SIMULIK, nous passerons à l'exportation de système conçu vers l'outil VIVADO. Il s'agit de l'étape de la génération automatique du code VHDL équivalent ainsi d'autres fichiers nécessaires pour l'implémentation.
5- Les codes VHDL générés seront évalués à nouveau sous Vivado en simulant

3.5.1. Conception de filtre avec l'outil Vitis Model Composer

La figure 3.5 représente le système de filtrage conçu, il est constitués d'un :

1- Bloc *FIR Compiler 7.2* représentant le filtre numérique. Ce bloc une version d'un cœur IP (Propriété Intellectuelle) de traitement numérique du signal (DSP) fourni par Xilinx, il nous permet de générer des implémentations de filtres FIR efficaces et optimisées pour l'implémenter en FPGA. Il offre une interface (Figure.3.5) de haut niveau où on peut spécifier les caractéristiques souhaitées du filtre, telles que l'ordre du filtre, la fréquence de coupure et le type de filtre. Pour plus d'information sur ce bloc, voir [32]
2- Les blocs Gateway_In et Gateway_Out. Le bloc Gateway_In (Entrée de passerelle Xilinx) est un bloc qui sert d'interface d'entrée pour la communication de données

entre le système extérieur et le circuit intégré programmable. Il permet de recevoir des signaux ou des données provenant d'autres périphériques ou systèmes et de les transmettre à l'intérieur du circuit intégré pour un traitement ultérieur. Le bloc Gateway_Out (Sortie de passerelle Xilinx) est quant à lui un bloc qui agit comme une interface de sortie, permettant de transmettre des signaux ou des données depuis le circuit intégré programmable vers d'autres dispositifs ou systèmes externes. Il facilite la communication des résultats ou des informations traitées à partir du circuit intégré vers d'autres composants ou systèmes connectés.

3- Le bloc "Vitis Model Composer Hub" est un élément clé dans la configuration. Il sert de l'interface principale qui nous permet de sélectionner le matériel souhaité, l'emplacement de sauvegarde du code VHDL compilé et d'autres plusieurs paramètres.

4- Le bloc "FDATool" est un outil graphique avancé pour la conception et l'analyse de filtres numériques (Figure.3.7). Il nous permet de spécifier les caractéristiques du filtre souhaité, telles que la réponse en fréquence, les atténuations, les délais, etc. Il permet également de visualiser et d'analyser les caractéristiques des filtres conçus, tels que la réponse en fréquence, la phase, l'impulsion de réponse, etc. Ce bloc est utilisé pour définir les paramètres de notre filtre et les charger directement au bloc *FIR Compiler 7.2*.

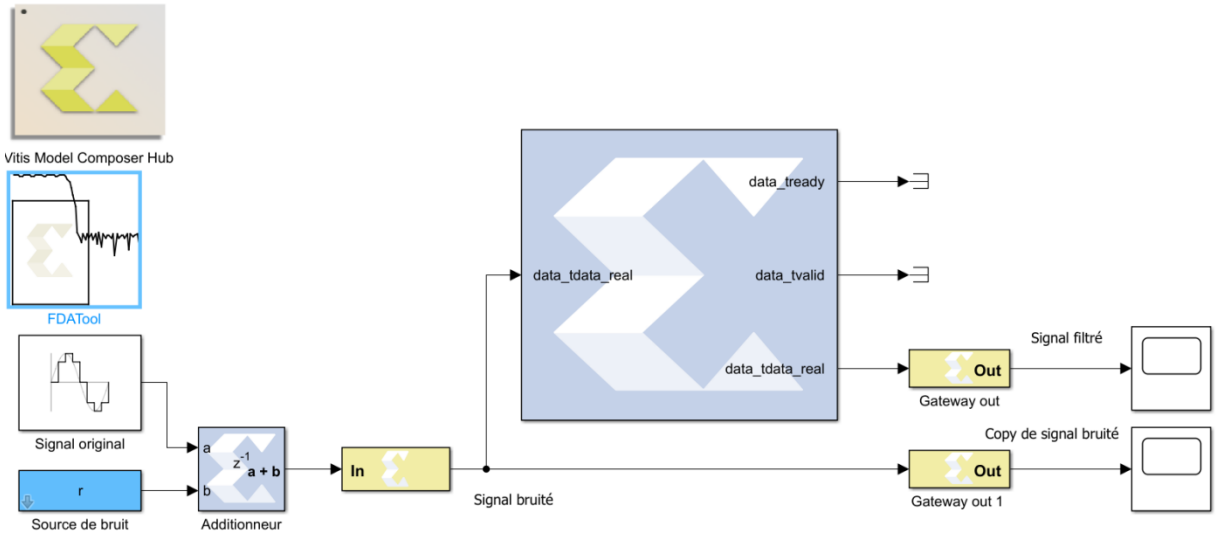

Figure.3.5. Le système de filtrage conçu sous SIMULINK/Vitis Model Composer.

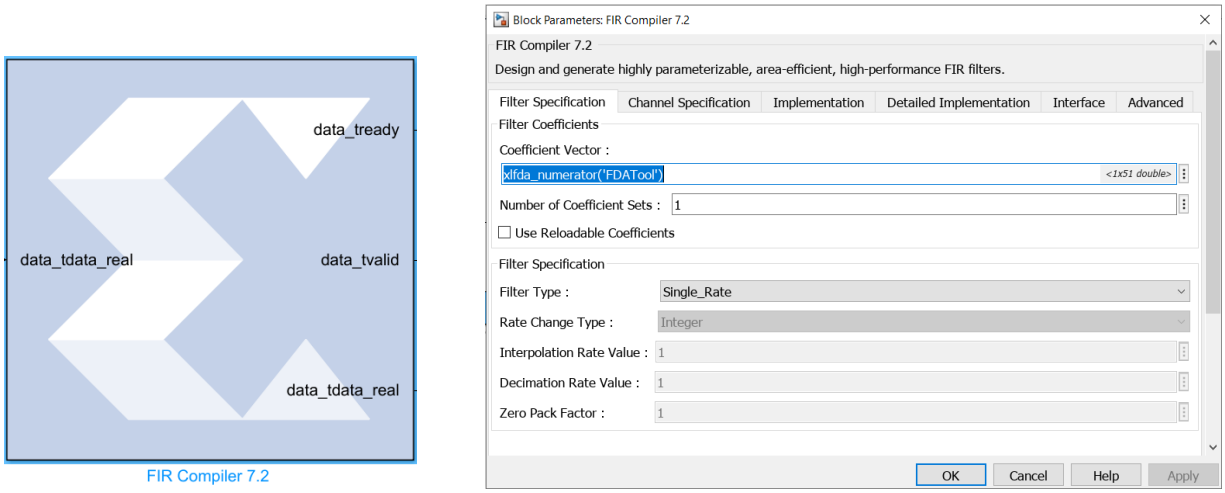

Figure.3.6. Le bloc FIR Compiler 7.2 avec l'interface de configuration.

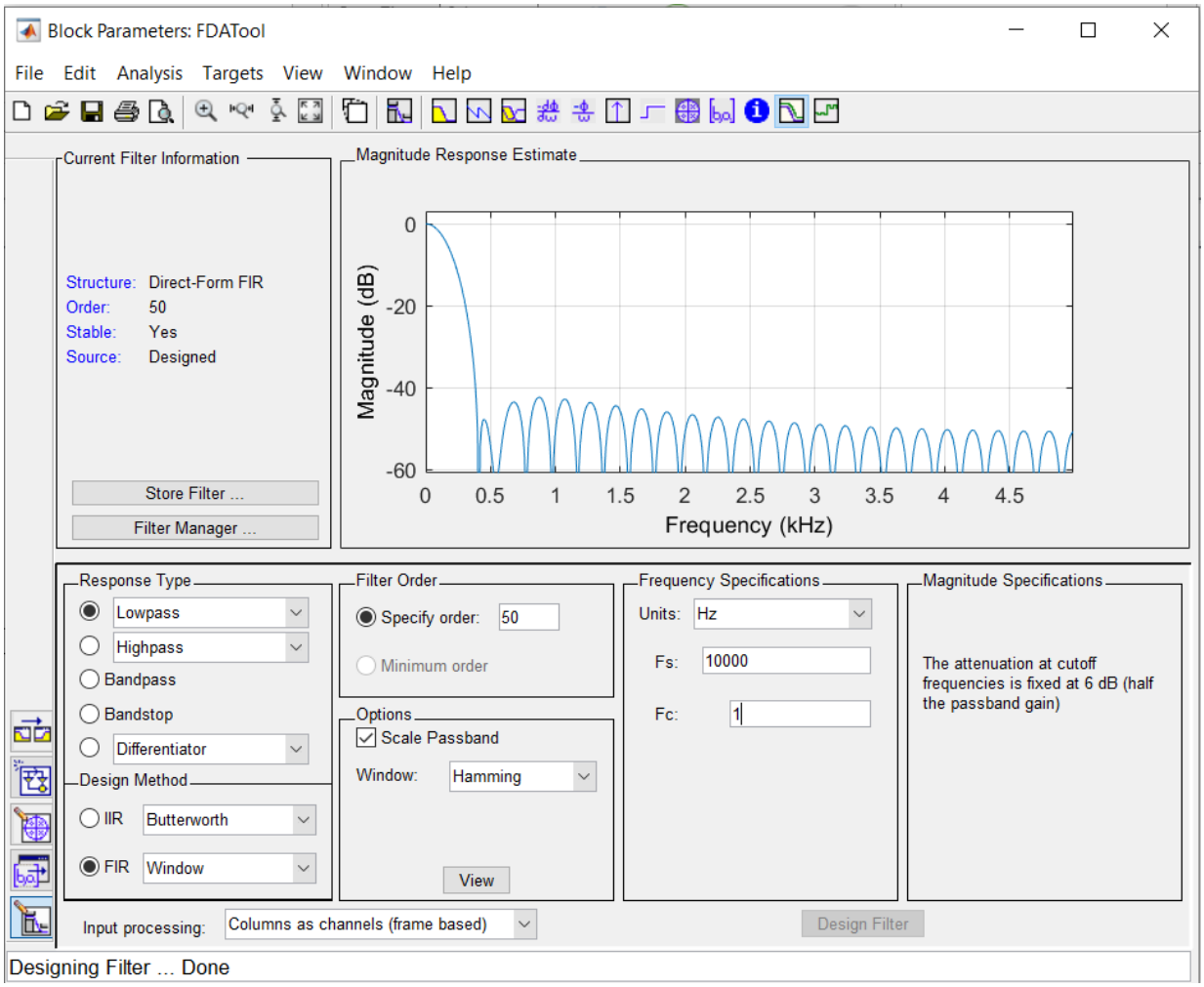

Figure.3.7. L'interface principale de bloc «FDATool".

3.5.1.2. Résultats de simulation de filtrage des signaux bruités

En choisissant les paramètres suivants : une fréquence d'échantillonnage $f_e = 10\ Khz$ et une fréquence de coupure $f_c = 2\ Khz$, le résultat du filtrage d'un signal sinusoïdal bruité est représenté dans la (Figure 3.8) Le résultat obtenu démontre clairement que le signal bruité est efficacement filtré.

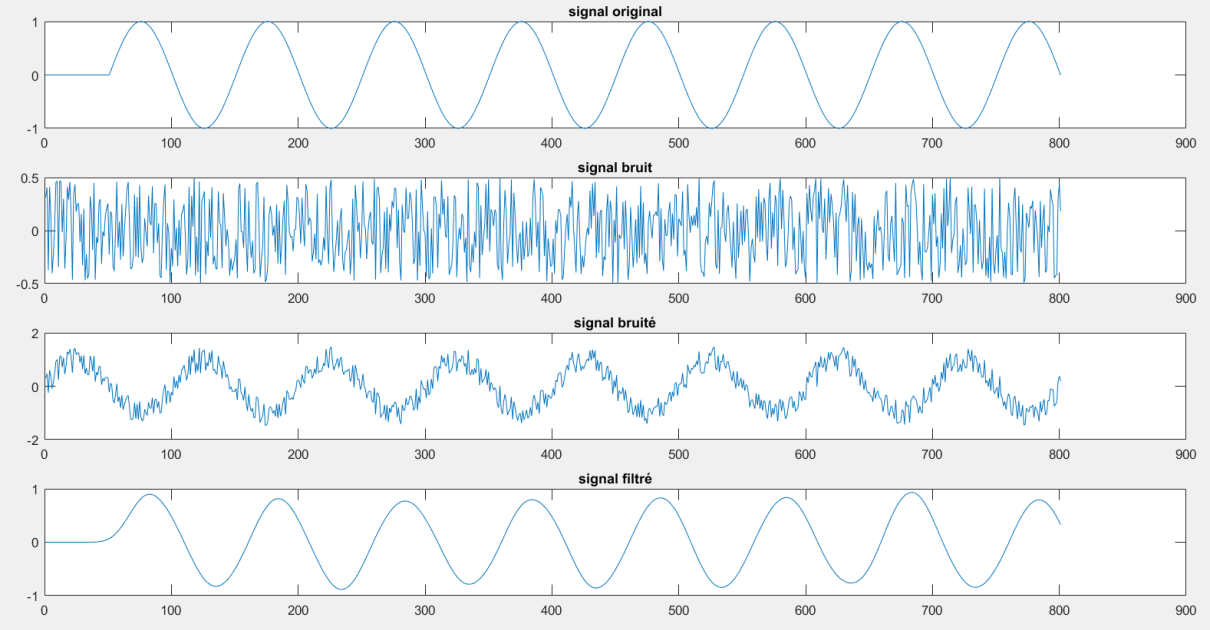

Figure.3.8. Résultat du filtrage d'un signal sinusoïdal bruité.

Dans la suite, les résultats de simulation de filtrage d'un signal audio bruité sont présentés, c'est parmi les applications les plus importantes de filtrage numérique. Le filtrage numérique des signaux audio est une méthode puissante pour améliorer la qualité sonore, éliminer les bruits indésirables et ajuster les caractéristiques fréquentielles d'un signal audio en utilisant des algorithmes de filtrage numérique.

Le filtrage numérique des signaux audio est largement utilisé dans diverses applications telles que le traitement audio professionnel, la musique, la radiodiffusion, la télécommunication et les systèmes de sonorisation. Il offre une flexibilité et une précision accrues par rapport aux techniques de filtrage analogique traditionnelles, permettant ainsi des manipulations plus précises et sophistiquées des signaux audio.

Dépend de caractéristiques spectrales de signal audio bruité (spectre de signal audio), en choisissant les paramètres suivants : une fréquence d'échantillonnage $f_e = 44{,}800\ Khz$

(fréquence d'échantillonnage des signaux audio usuels) et une fréquence de coupure $f_c = 10\ Khz$. La figure.3.9 présente le spectre du signal audio original ainsi que celui du signal bruité.

La figure.3.10. présente le signal original, le signal bruité, et le signal filtré. Il est clair que le signal audio filtré est identique à ce de l'original. D'autre part, lorsqu'on entend le résultat, on remarque que le signal audio perturbé a été bien filtré.

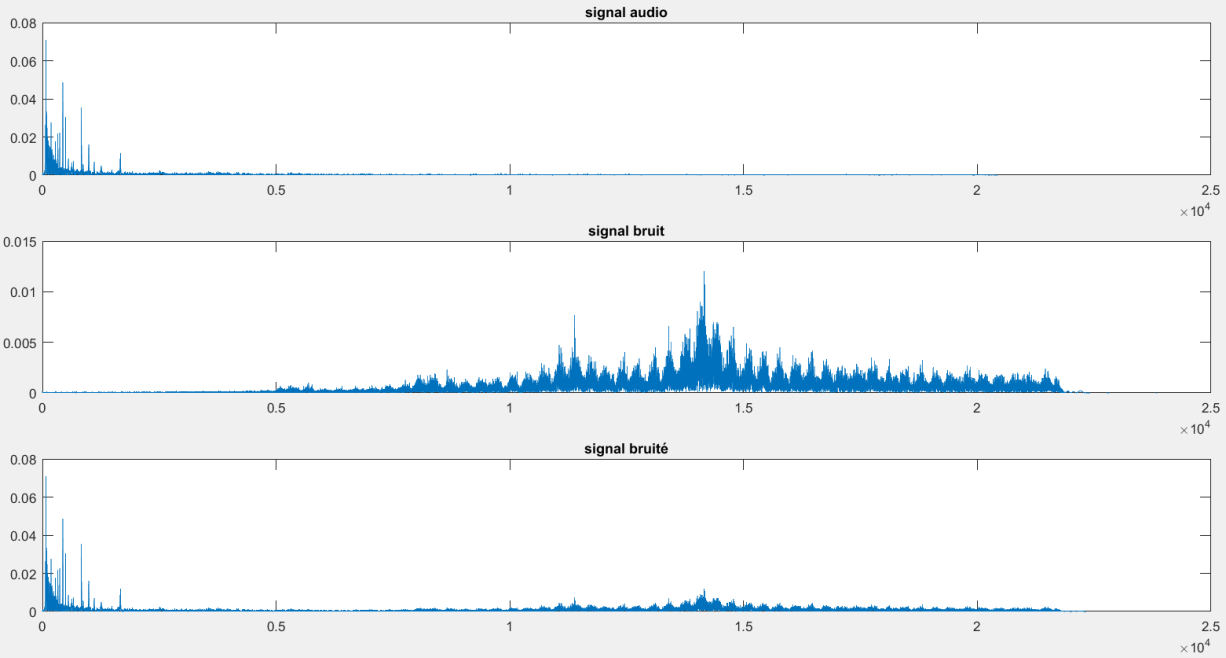

Figure.3.9. Le spectre du signal audio original, bruit, ainsi que celui du signal bruité.

3.5.2. Implémentation de filtre dans un FPGA

Dans cette partie, nous allons poursuivre les étapes d'implémentation du filtre dans un FPGA, ainsi que l'évaluation fonctionnelle des codes VHDL générés et l'évaluation en temps réel.

Pour générer les codes VHDL automatiquement à partir des blocs SIMULINK, on commence tous d'abord par la configuration de projet à partir de bloc *Vitis Model Composer Hub*. Nous commençons par définir la série de circuit FPGA souhaitée (dans ce cas, le circuit xc7z020clg484-1). Ensuite, nous spécifions le langage de descriptions matérielles (HDL) utilisé (dans ce cas, le VHDL) et l'emplacement où les codes VHDL sont générés figure.3.11.

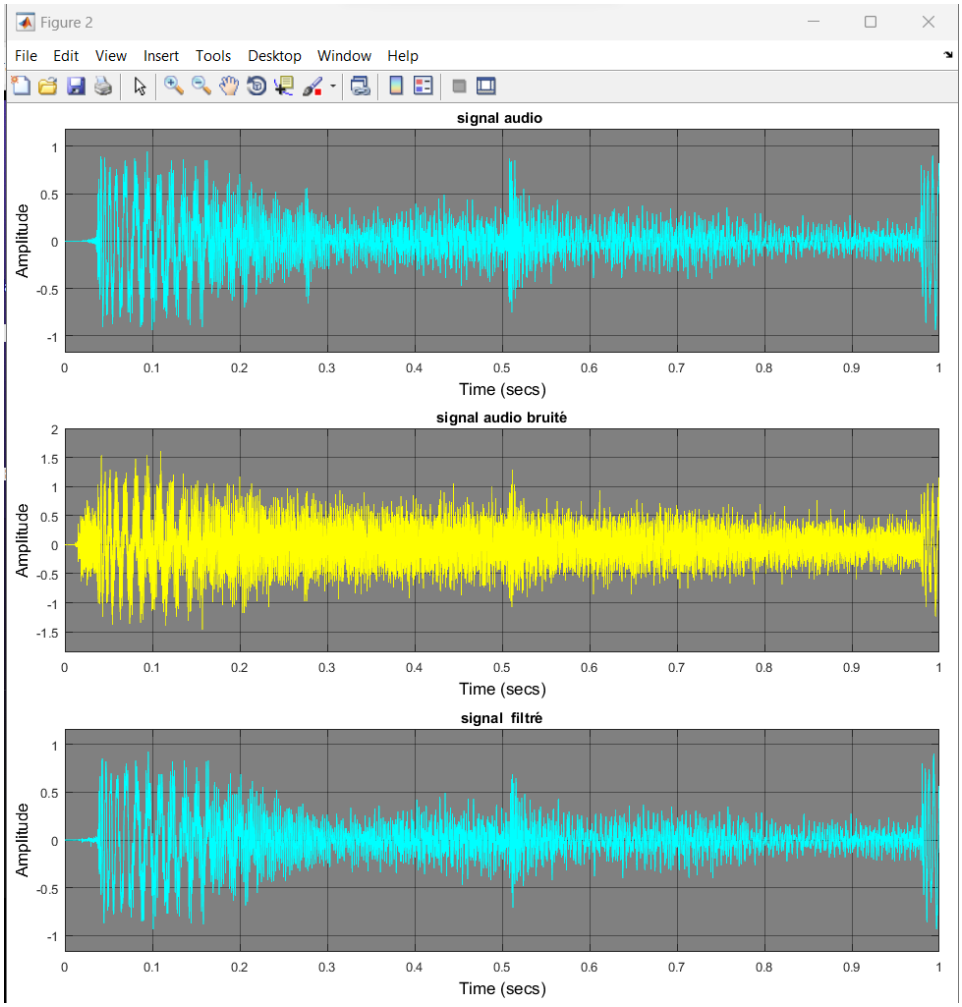

Figure.3.10. Le spectre du signal audio original, bruit, ainsi que celui du signal bruité.

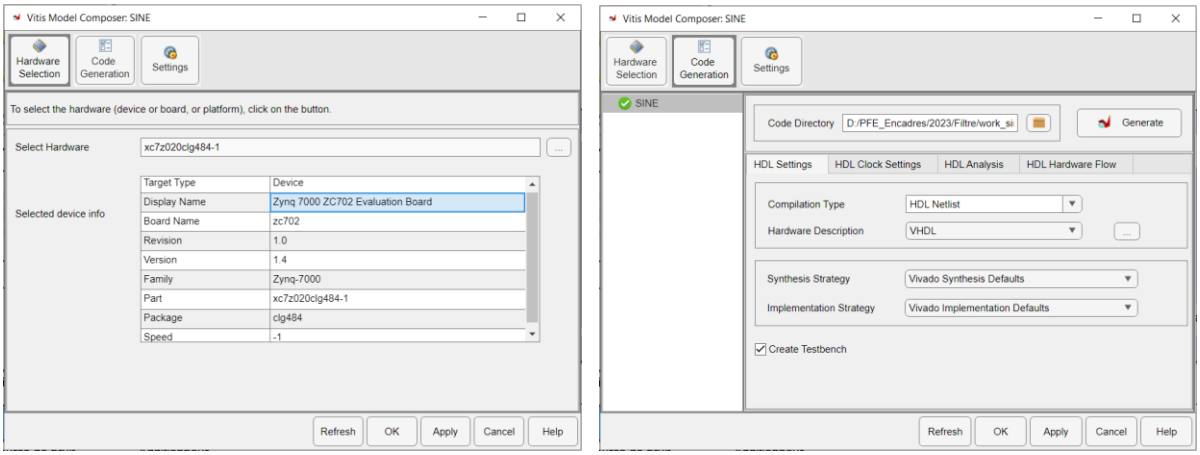

Figure.3.11. La configuration de projet à partir de bloc Vitis Model Composer.

L'étape suivante consiste à créer un projet sous Vivado et à lancer la simulation fonctionnelle pour vérifier le bon fonctionnement des codes VHDL générés. En conservant les mêmes paramètres précédemment mentionnés, le résultat de la simulation de filtrage d'un signal sinusoïdal bruité est représenté dans la figure 3.13, tandis que le résultat du filtrage d'un signal audio est représenté dans la figure 3.14. Les résultats obtenus confirment que les codes VHDL générés fonctionnent correctement.

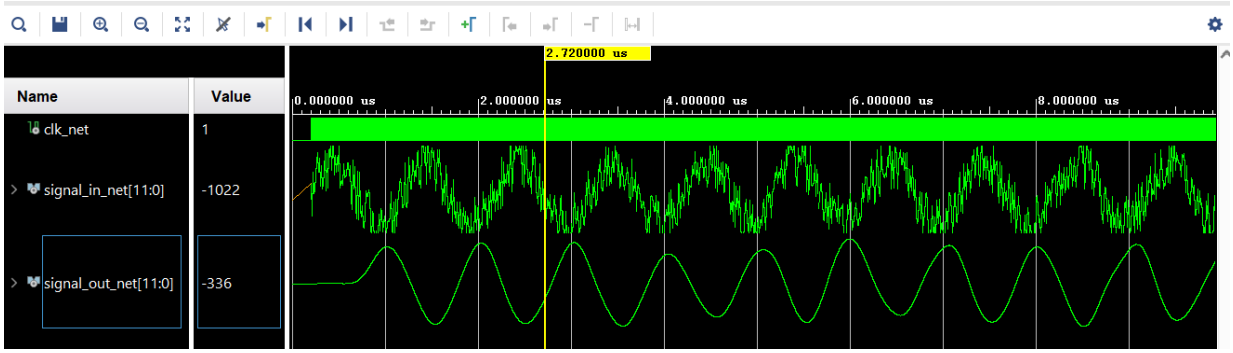

Figure.3.12. Le résultat de la simulation de filtrage d'un signal sinusoïdal bruité.

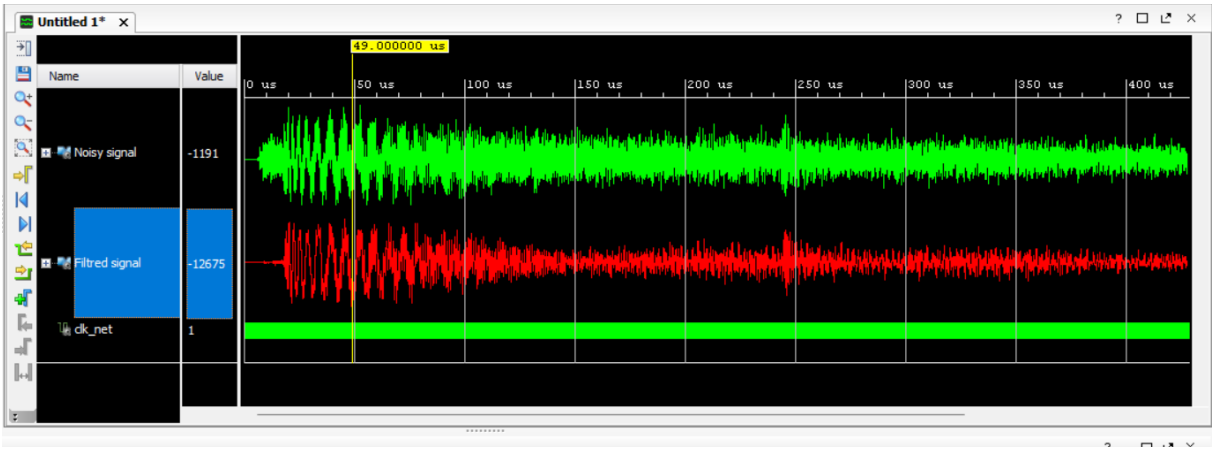

Figure.3.13. Résultat du filtrage d'un signal audio (signal bruité en vert, le signal filtré en rouge).

3.5.2.1. Filtrage d'un signal sinusoïdal bruité en temps réel

Dans cette partie, nous passerons à l'implémentation de notre filtre dans le circuit FPGA. Le circuit FPGA reçoit le signal bruité (analogique) après sa conversion en signal numérique au niveau du convertisseur PMOD-AD1. Après le traitement, le signal filtré (numérique) est ensuite converti en signal analogique au niveau du convertisseur PMOD-

DA2. Pour assurer l'interfaçage entre ces deux modules et le circuit FPGA, nous ajoutons des composants préalablement conçus en VHDL (drivers) au projet Vivado de notre filtre. La figure.3.14 représente le schéma synoptique de circuit complet. L'entité VHDL principale de circuit est présentée dans la figure.3.15.

Figure.3.14. Schéma de circuit complet (Filtre avec les modules PMOD-AD1 et PMOD-DA).

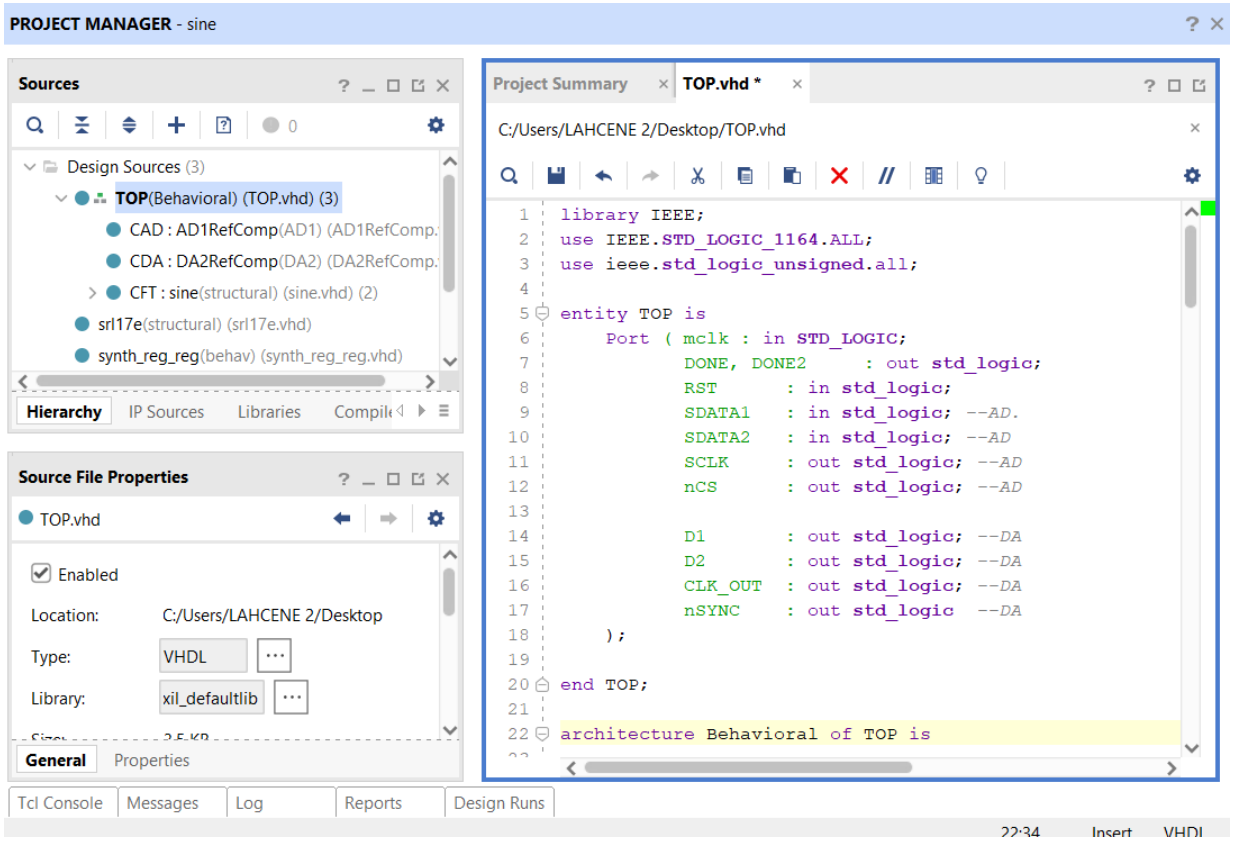

Figure.3.15. L'entité VHDL principale de circuit.

À partir de la (figure 3.13), pour le module PMOD-AD1 :

- Les signaux *SDATA1* et *SDATA2* sont les deux signaux numériques transmis en série provenant des deux CAN vers le circuit FPGA.
- Le signal *nCS* est utilisé pour synchroniser les deux CAN.
- Le signal *sclk* est issu du circuit FPGA pour contrôler les CAN (25 MHz).

Pour le module PMOD-DA2 :

- Les signaux *D1* et *D2* sont les deux signaux numériques transmis en série provenant du circuit FPGA vers le CNA.
- Le signal *CLK_OUT* est issu du circuit FPGA pour contrôler les CNA (25 MHz).
- Le signal *nSYNC* est utilisé pour synchroniser les deux CNA.

Après la vérification syntaxique des codes VHDL, nous passons à l'étape suivante, qui consiste à attribuer une adresse à chaque entrée/sortie du système conçu en fonction du circuit FPGA utilisé. Cette étape conduit à la création d'un fichier d'adressage, comme illustré dans la figure 3.16.

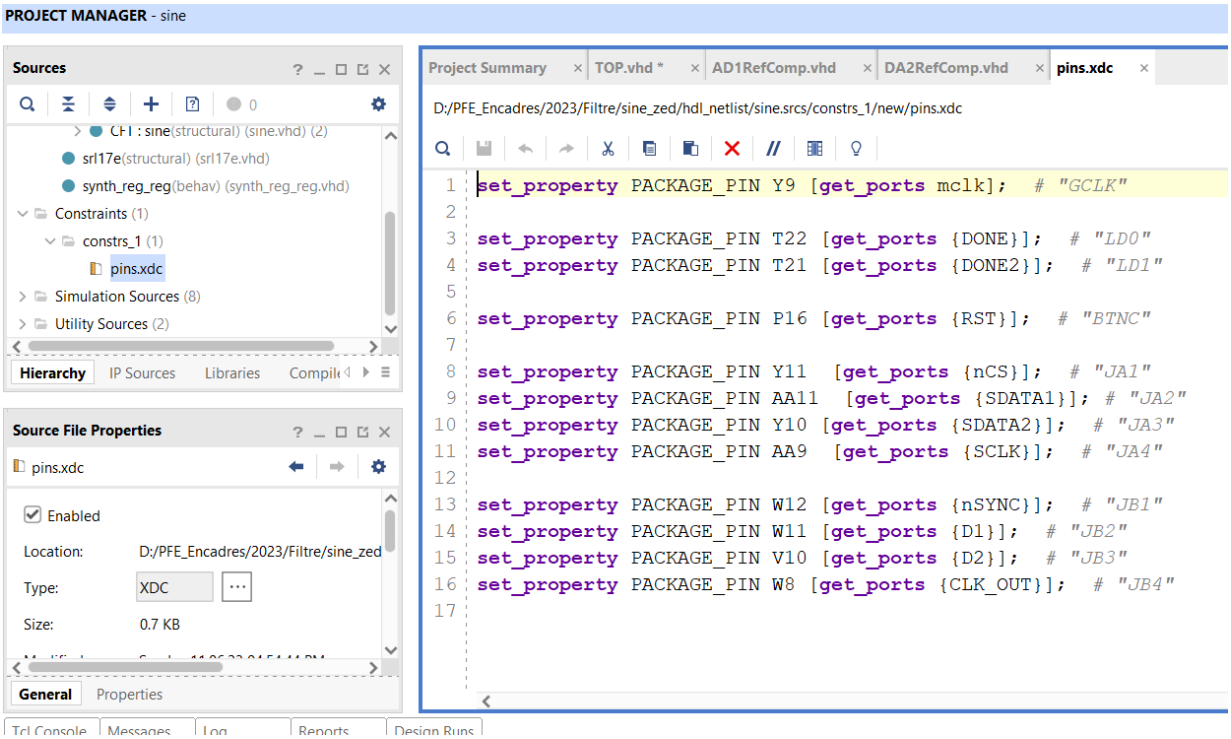

Figure.3.16 : Fichier d'adressage de chaque entrée/sortie de système.

La dernière étape consiste à générer le fichier de programmation pour configurer le circuit FPGA. Le résultat de filtrage d'un signal sinusoïdal en temps réel est représenté dans la figure suivante :

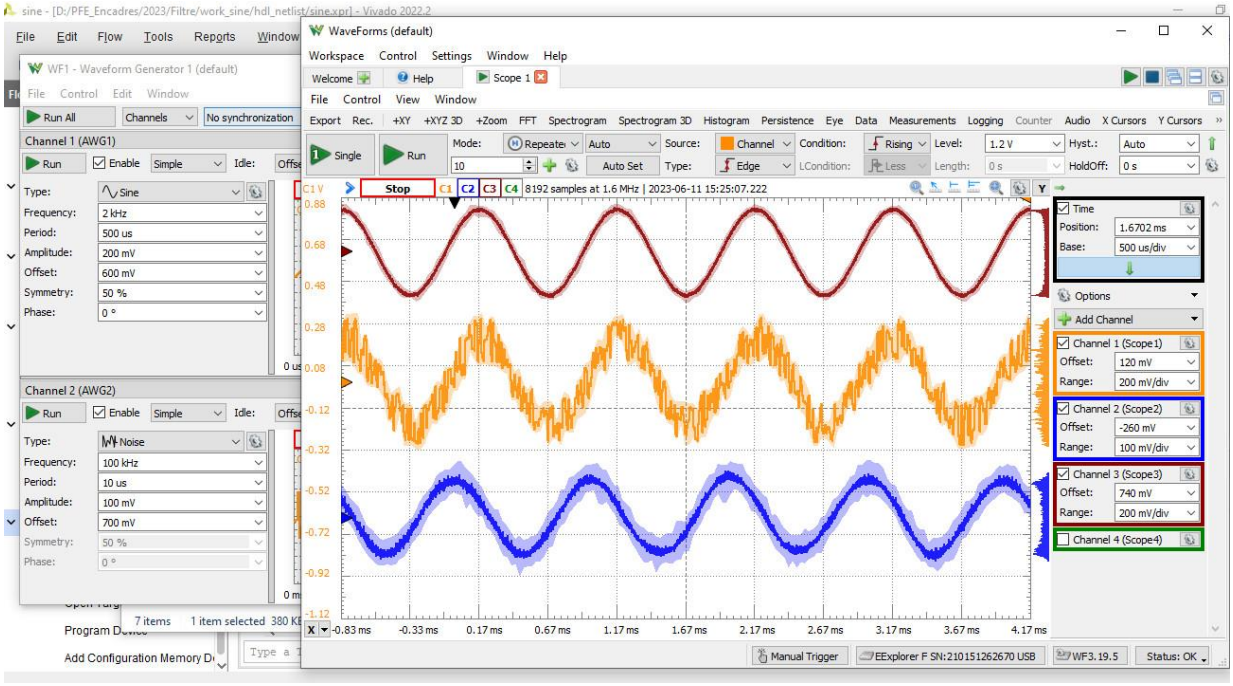

Figure.3.17 : Résultat de filtrage en temps réel d'un signal sinusoïdal bruité (bleu).

Le résultat obtenu pour le filtrage d'un signal sinusoïdal en temps réel montre bien le bon fonctionnement de notre filtre. Le schéma équivalent de circuit complet est également représenté dans la figure.3.18.

3.5.2.2. Filtrage d'un signal audio bruité en temps réel

La carte Zedboard est équipée d'une carte son contenant des convertisseurs 24 bits de la gamme ADAU1761 d'Analog Devices, ce qui permet l'enregistrement et la lecture stéréo à 48KHz. L'interfaçage entre le circuit FPGA et le ADAU1761 est assuré par l'utilisation d'un bloc IP (en VHDL) inspiré d'un projet réalisé par *Stefan Scholl* [33]. Nous avons réutilisé ce bloc IP dans notre projet, en apportant bien sûr quelques modifications.

Dans le circuit FPGA, les échantillons audio de 24 bits sont lus à partir de l'entrée de ligne bleue et sont fournis par *line_in_l* et *line_in_r* à la logique FPGA. Les échantillons audio peuvent être transmis à l'ADAU1761 pour être restitués sur la prise casque noire via les signaux d'entrée *hphone_r* et *hphone_l*. Les nouveaux échantillons (provenant de l'entrée) sont signalés par *new_sample* = 1 ou par le front montant de sample_clk_48k.

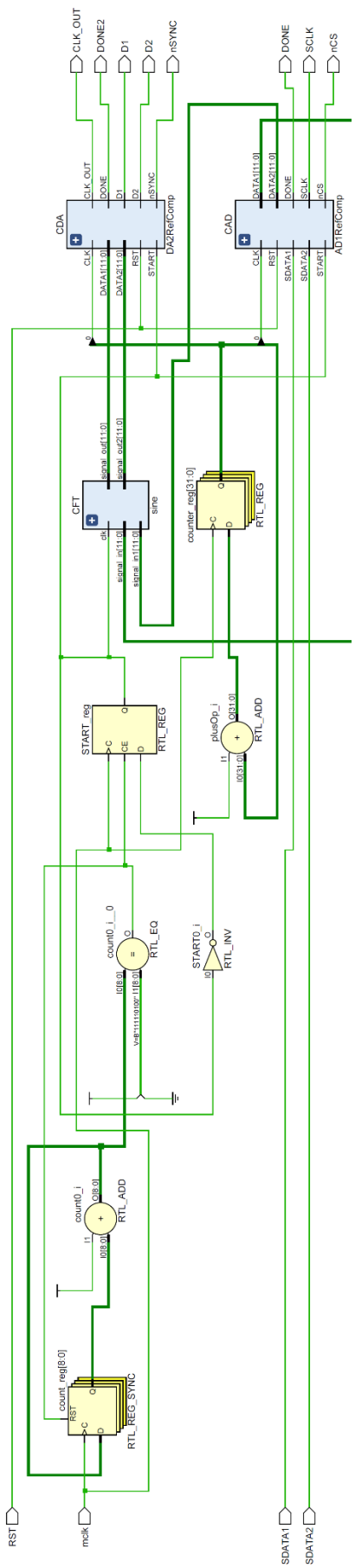

Figure.3.18 : Le schéma équivalent de circuit complet .

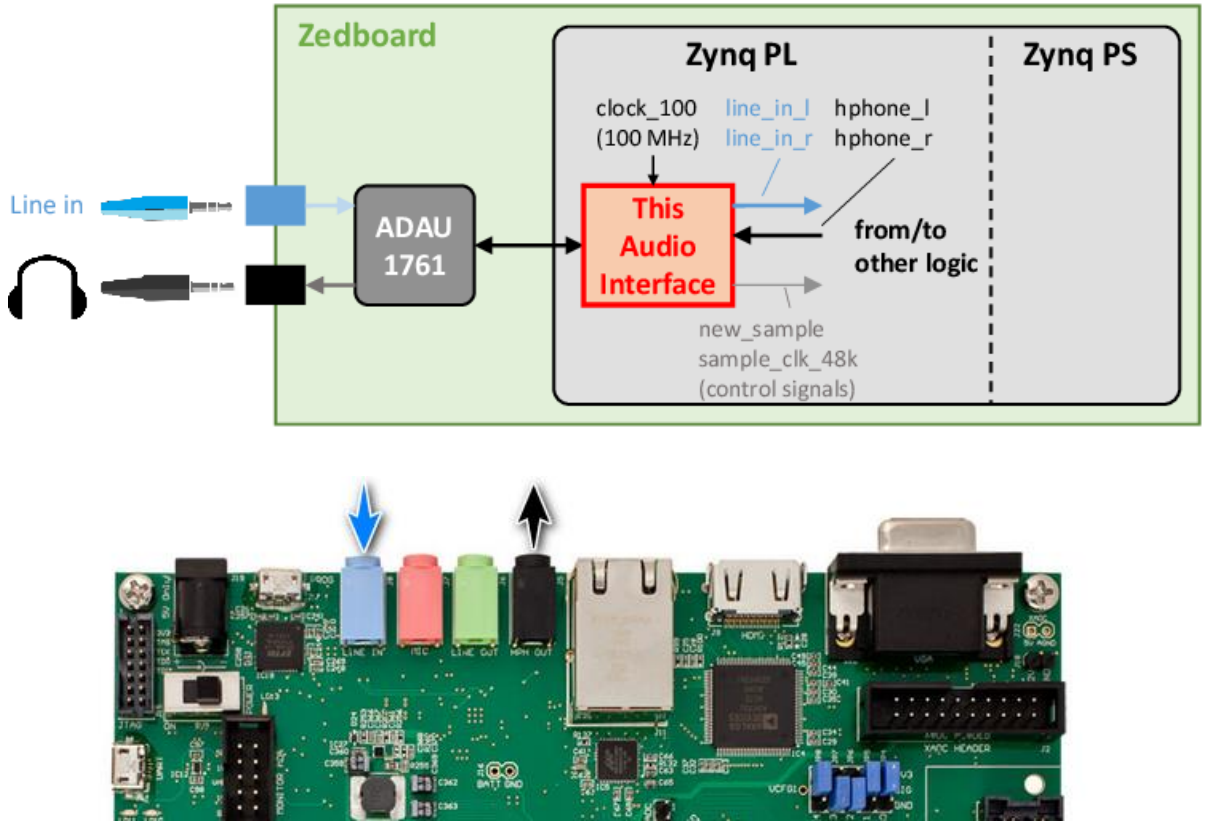

Figure.3.19 : L'interface audio de la carte Zedboard.

Une fois l'interface audio configurée avec le filtre numérique et les fichiers d'adressage ajoutés, le fichier de programmation du circuit FPGA est généré. Le résultat du filtrage d'un signal audio en temps réel peut être observé dans la figure 3.20. Il est évident que le signal audio est correctement filtré, ce résultat peut être confirmé en écoutant le signal filtré à travers les haut-parleurs. Le schéma équivalent de circuit complet est également représenté dans la figure 3.21.

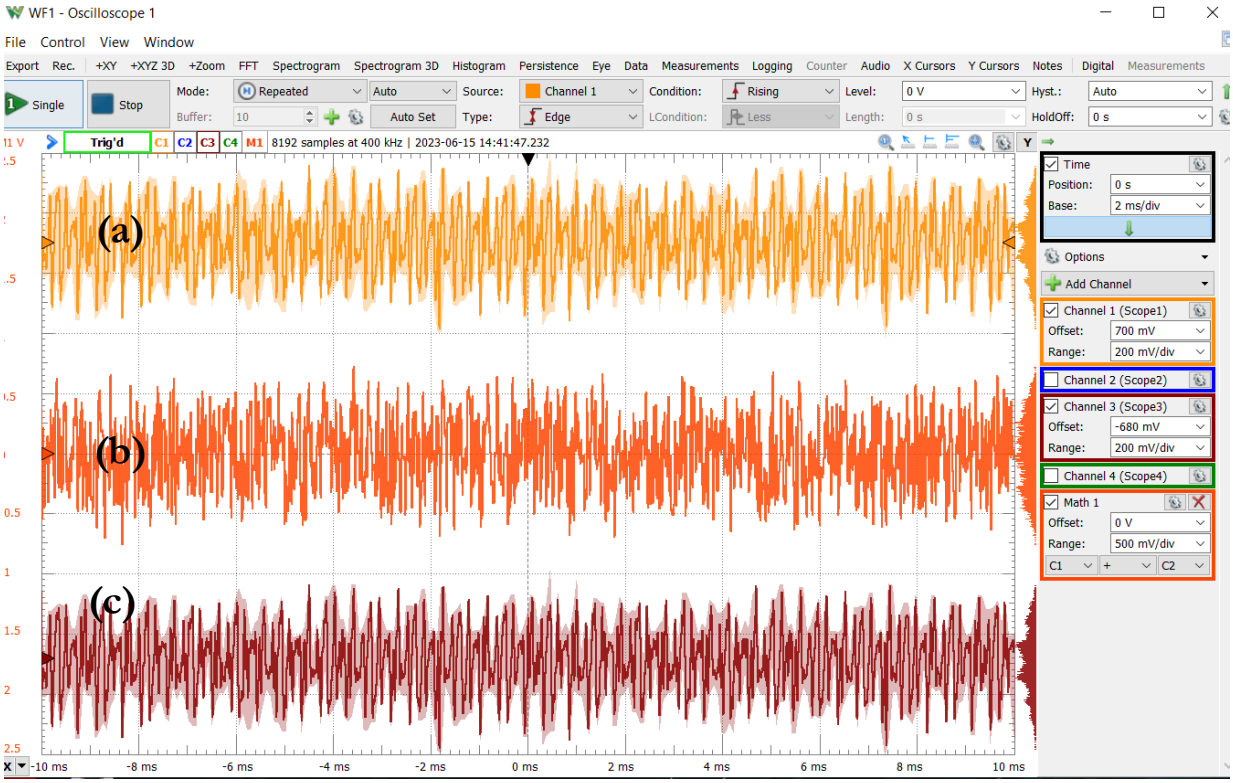

Figure.3.20 : Résultat de filtrage en temps réel d'un signal audio, (a : signal original, b : signal bruité, c : signal filtré).

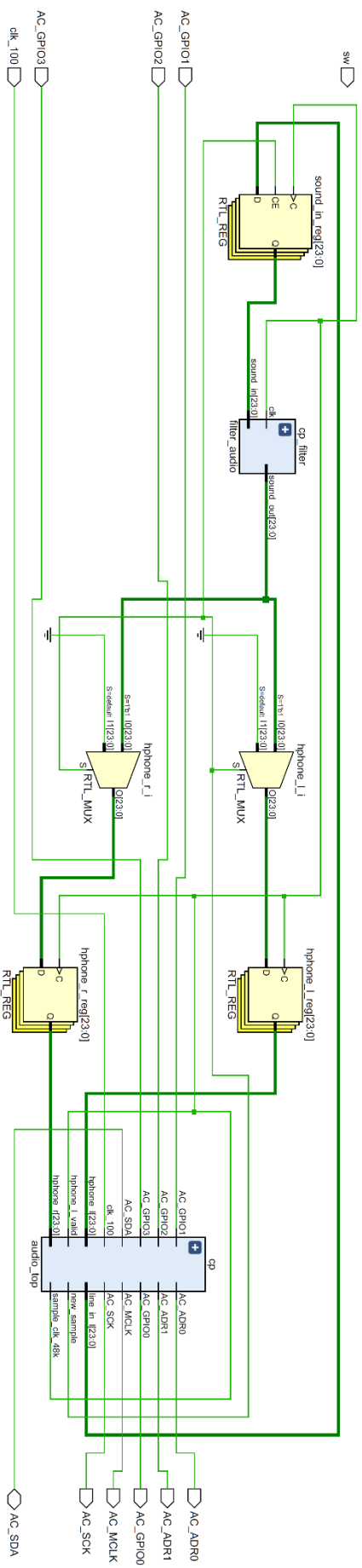

Figure.3.21. Le schéma équivalent de circuit complet pour le filtre d'un signal audio.

3.6. Conclusion

En conclusion de ce chapitre, nous avons atteint avec succès l'objectif de simuler un filtre numérique sous SIMULINK en utilisant l'outil Xilinx Vitis Model Composer et l'implémenter dans un FPGA. Le système de filtrage a été évalué à travers des simulations, au cours desquelles nous avons appliqué le filtre à un signal sinusoïdal bruité ainsi qu'à un signal audio. L'outil Vitis Model Composer s'est avéré être un outil précieux qui nous a grandement facilité la conception de systèmes numériques complexes. Sa convivialité et ses fonctionnalités avancées nous ont permis de modéliser, simuler et évaluer efficacement notre système de filtrage.

Nous avons décrit en détail les étapes d'implémentation du filtre sur la carte Zedboard, permettant ainsi de démontrer sa faisabilité pratique. De plus, nous avons présenté les résultats en temps réel obtenus pour le signal sinusoïdal bruité et le signal audio, mettant en évidence l'efficacité du filtre numérique dans la réduction du bruit.

Conclusion générale

Conclusion générale

Ce mémoire de fin d'étude a porté sur la simulation et l'implémentation d'un filtre RIF numérique dans un FPGA pour le filtrage des signaux audio. Nous avons abordé différents aspects liés aux filtres numériques, aux FPGA et au langage VHDL, en nous concentrant sur la gamme Xilinx et les outils de conception associés.

Dans le premier chapitre, nous avons exploré la théorie des filtres numériques, en étudiant les principes fondamentaux et les différentes techniques de filtrage. Nous avons acquis une compréhension approfondie des filtres RIF et de leur application dans le traitement des signaux audio.

Le deuxième chapitre a été consacré à la présentation des FPGA, du langage VHDL et des outils de conception, en mettant l'accent sur la gamme Xilinx. Nous avons examiné les fonctionnalités des FPGA et leur flexibilité pour l'implémentation de systèmes numériques complexes. De plus, nous avons exploré le langage VHDL comme moyen de décrire le comportement des circuits numériques et nous avons présenté les outils de conception Xilinx utilisés dans ce projet.

Dans le troisième chapitre, nous avons décrit en détail les étapes de la simulation du filtre numérique à l'aide de SIMULINK et de Xilinx Vitis Model Composer. Nous avons mis en évidence les avantages de ce dernier outil, qui permet une conception simplifiée sans nécessiter une expertise approfondie en VHDL. Nous avons ensuite présenté les résultats de la simulation, illustrant le filtrage réussi d'un signal sinusoidal bruité et d'un signal audio.

Les résultats obtenus ont démontré l'efficacité du filtre RIF numérique implémenté dans le FPGA pour le filtrage des signaux audio. Les signaux bruités ont été nettoyés de manière satisfaisante, améliorant ainsi la qualité de l'audio d'origine. Ces résultats confirment la faisabilité et l'applicabilité de notre approche.

En conclusion, ce mémoire a contribué à l'approfondissement des connaissances sur la simulation et l'implémentation de filtres RIF numériques dans les FPGA pour le filtrage des signaux audio. Il a fourni des résultats prometteurs et a ouvert la voie à des applications pratiques dans le domaine du traitement du signal audio. Des perspectives de recherche futures peuvent inclure l'optimisation des performances du filtre, l'exploration de techniques de filtrage avancées et l'extension de l'implémentation à d'autres types de filtres.

Bibliographie

Bibliographie

[1].Cottet, Francis. Traitement du signal. Dunod, 2005.

[2].Olivier SENTIEYS, *Traitement Numérique du Signal*. Polycopié de cour, Université de Rennes 1, lien : http://people.rennes.inria.fr/Olivier.Sentieys/teach/PolyTNS.pdf

[3].Mahieddine Imene Hammou Yamina, *Detection tt Localisation Dans Le Temps Des Differentes Porteuses A L'aide D'un Passe Bande Rif /Ondelette Continue.* Mémoire de Master. Université Abdelhamid Ibn Badis Mostaganem. 2020-2021.

[4].Taihi Fatiha & Benmaiza Samia, *Etude et implémentation d'un Filtre RIF adaptatif sous VHDL.* Mémoire de Master. Université SAAD DAHLAB de BLIDA. 2017-2018.

[5].Olivier Sentieys, Le filtrage numérique pour les nuls, ENSSAT – Université de Rennes 1, 28 mai 2008.

[6]. Le principe des filtres numériques, Sciences et Techniques, 24/04/2011, lien : http://www.ferdinandpiette.com/blog/2011/04/le-principe-des-filtres-numeriques/

[7].SLIMANI Amina &IRKI Nesrine. *Conception d'un Logiciel convivial pour la Synthèse des Filtres actifs et Filtres numériques*. Mémoire de Master. Université YAHIA FARES DE MEDEA.2021-2022.

[8].Matthieu Kowalski, *Traitement du signal*. Polycopié de cour. Université PARIS-SACLAY.

[9].Lien : http://dspace.univ djelfa.dz:8080/xmlui/bitstream/handle/123456789/941/chap1VF.pdf?sequence=8&isAllowed=y

[10]. BEN MAKHLOUFI Amar, *Implémentation des RNA sur FPGA en vue de commande de la MAS*. Mémoire de Master. Université de m'Sila. 2015-2016.

[11]. Lahcene Merah, *FPGA and VHDL*- Polycopié de cours, Université Amar Telidji de Laghouat, lien : http://dspace.lagh-univ.dz/handle/123456789/7783.

[12]. "Qu'est-ce qu'un FPGA ? Introduction à la programmation matérielle" - O'Reilly, lien : https://www.oreilly.com/library/view/microservices-for-enterprise/9780134642474/ch03.html

[13]. "Configuration des FPGA" - Xilinx: https://www.xilinx.com/fr/products/design-tools/fpga-design/configuration.html.

[14]. Le lien : https://www.xilinx.com/products/silicon-devices/fpga.html

[15]. S. M. Burns et al., "Benefits of FPGA Design Flexibility in Space borne Processing," 2019 IEEE Aerospace Conference.

[16]. A. Elhakeem et al., "High Performance Computing Using FPGA Accelerators," 2016 International Conference on High Performance Computing & Simulation.

[17]. J. Li et al., "An FPGA-based Low-Latency Motion Estimation Accelerator for H.265/HEVC Encoder," 2018 IEEE International Conference on Consumer Electronics-China.

[18]. J. P. Singh et al., "Rapid FPGA Prototyping of Real-Time Video Processing Systems," 2017 IEEE International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques.

[19]. G. Montavon et al., "FPGA-Based Implementation of 4G-LTE Baseband Transceiver," 2015 International Conference on Re-ConFigurable Computing and FPGAs.

[20]. S. S. Momeni et al., "FPGA Implementation of High-Speed Low-Power Pipelined LMS Adaptive Filter for Aerospace Applications," 2016 14th International Conference on Frontiers of Information Technology.

[21]. D. Munoz et al., "High-Level Synthesis-Based Design Space Exploration of Embedded FPGA Systems," 2019 32nd Symposium on Integrated Circuits and Systems Design.

[22]. E. T. Li et al., "A Novel Field-Programmable Gate Array (FPGA)-Based Smart Sensor for Real-Time Biomedical Signal Processing Applications," 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology.

[23]. R. Zhang et al., "FPGA-Based Real-Time Obstacle Avoidance System for Autonomous Vehicles," 2018 IEEE 4th International Conference on Computer and Communications.

[24]. P. J. Ashenden, "The Designer's Guide to VHDL," 3rd Edition, Morgan Kaufmann, 2008.

[25]. Xilinx Inc, Form 8-K, Current Report, Filing Date Apr 25, 2012". Secdatabase.com. Retrieved May 6, 2018.

[26]. EDN. "The Vivado Design Suite accelerates programmable systems integration and implementation by up to 4X, lien : https://www.edn.com/the-vivado-design-suite-accelerates-programmable-systems-integration-and-implementation-by-up-to-4x/

[27]. Vitis Model Composer Overview, AMD, lien : https://www.xilinx.com/products/design-tools/vitis/vitis-model-composer.html

[28]. System Generator for DSP, User Guide, Xilinx INC, UG640 (v11.4) December 2, 2009.

[29]. Place and Route for FPGAs, Western University, CANADA, http://www.eng.uwo.ca/

[30]. Digilent PmodAD1 Analog To Digital Module Converter Board Reference Manual, Revision: 04/12/05.

[31]. Digilent PmodDA2 Digital To Analog Module Converter Board Reference Manual, Revision: September 25, 2006.

[32]. AMD-Xilinx, FIR Compiler v7.2 LogiCORE IP Product Guide, PG149 October 26, 2022.

[33]. Stefan Scholl, Audio Interface for the Zedboard, Microelectronic Systems Design Research Group TU Kaiserslautern, Germany, March 2015.

Résumé

Résumé

L'objectif de ce travail consiste à mettre en œuvre un filtre numérique RIF dans un FPGA pour le filtrage des signaux audio bruités. Le filtrage numérique des signaux audio présente de nombreux avantages par rapport au filtrage analogique, tels que la flexibilité, la précision, la reproductibilité et la possibilité d'effectuer des filtrages complexes. Par ailleurs, un FPGA offre une flexibilité et une puissance de traitement élevées, ce qui en fait un choix populaire pour le filtrage numérique des signaux audio. Nous avons utilisé l'outil Vitis Model Composer pour la conception du filtre, cet outil convivial nous a permis de concevoir des systèmes complexes sans une connaissance approfondie des langages HDL. Après la simulation sous Simulink, des codes VHDL ont été automatiquement générés pour poursuivre le processus d'implémentation. Nous avons utilisé l'outil VIVADO pour la simulation fonctionnelle des codes VHDL générés et pour l'implémentation du filtre dans le circuit FPGA intégré à la carte ZEDboard. L'évaluation en temps réel du filtre implémenté, en filtrant des signaux sinusoïdaux bruités ainsi que des signaux audios, a démontré son efficacité.

Mots clés: RIF, FPGA, Vitis Model Composer, HDL, Simulink, VHDL, VIVADO, ZEDboard

Abstract

The objective of this work is to implement a digital RIF filter in an FPGA for filtering noisy audio signals. Digital filtering of audio signals offers several advantages over analog filtering, such as flexibility, precision, and the ability to perform complex filtering operations. Additionally, an FPGA provides high flexibility and processing power, making it a popular choice for digital filtering of audio signals. We used the Vitis Model Composer tool for filter design, which allowed us to design complex systems without in-depth knowledge of HDL languages. After simulation under Simulink, VHDL codes were automatically generated to proceed with the implementation process. We used the VIVADO tool for functional simulation of the generated VHDL codes and for implementing the filter in the FPGA circuit that is integrated into the ZEDboard. Real-time evaluation of the implemented filter, by filtering both noisy sinusoidal signals and audio signals, has demonstrated its effectiveness.

Key words: RIF, FPGA, Vitis Model Composer, HDL, Simulink, VHDL, VIVADO, ZEDboard.

ملخص

الهدف من هذا العمل هو تنفيذ مُرشِّح رقمي RIF في وحدة معالجة متكاملة قابلة للبرمجة (FPGA) لتنقية إشارات الصوت المُشوَشة. تقدم تقنية الترشيح الرقمي لإشارات الصوت العديد من المزايا عن مقارنة بالتقنية التناظرية، مثل المرونة والدقة وإمكانية والقدرة على تنفيذ عمليات تنقية معقدة. بالإضافة إلى ذلك، توفر وحدة المعالجة المتكاملة القابلة للبرمجة (FPGA) مرونة وقوة معالجة عالية، مما يجعلها خيارًا شائعًا لتصفية إشارات الصوت الرقمية. استخدمنا أداة Vitis Model Composer لتصميم المرشح، والتي سمحت لنا بتصميم أنظمة معقدة بدون الحاجة إلى معرفة معمقة للغات HDL. بعد المحاكاة ضمن برنامج Simulink، تم توليد كود ال VHDL تلقائيًا لمتابعة عملية التنفيذ. استخدمنا أداة VIVADO للمحاكاة الوظيفية لكود ال VHDL المُنشأ وتنفيذ المُرشِّح على دائرة ال FPGA المُتكامِلة على لوحة ZEDboard. أظهر التقييم في الوقت الآني للمُرشِّح المُنفَّذ، من خلال تنقية إشارات جيبية مشوَّشة وإشارات صوتية، فعاليته.

كلمات مفتاحية : RIF, FPGA, Vitis Model Composer, HDL, Simulink, VHDL, VIVADO, ZEDboard.