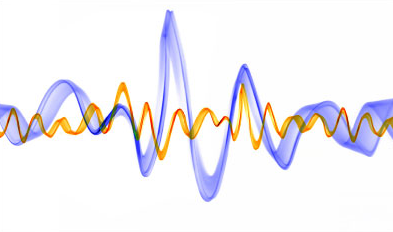
**TRAITEMENT DE SIGNAL**

**EN TEMPS REEL**



Réalisé par : Elhattami Amine

Encadré par : Pr. ElMhamdi Jamal

Table of Contents

[2 Vue d’ensemble 5](#_Toc369465011)

[2.1 Sujet du Projet 5](#_Toc369465012)

[2.2 Choix de la carte de développement 5](#_Toc369465013)

[3 Signal numérique audio 6](#_Toc369465014)

[3.1 Propriété physique d’un signal sonore 6](#_Toc369465015)

[3.2 Représentation numérique du son 8](#_Toc369465016)

[3.3 Fréquences d'échantillonnage: 12](#_Toc369465017)

[4 Interface audio lm4550 14](#_Toc369465018)

[4.1 Introduction 14](#_Toc369465019)

[4.2 Description des entrée/sortie du codec audio : 15](#_Toc369465020)

[4.3 Les registres internes du LM4550 : 16](#_Toc369465021)

[4.4 Description du fonctionnement 17](#_Toc369465022)

[4.4.2 Les sorties stéréo 17](#_Toc369465023)

[4.4.3 Les sorties mono 17](#_Toc369465024)

[4.4.4 Le loopBack analogique et numérique 17](#_Toc369465025)

[4.4.5 Réinitialisation(Reset) 18](#_Toc369465026)

[4.4.6 Protocole d’interface série de la trame AC-LINK 18](#_Toc369465027)

[4.4.7 Trame de sortie (SDATA\_OUT) 18](#_Toc369465028)

[4.4.8 Trame d’entrée (SDATA\_IN): 21](#_Toc369465029)

[4.5 Diagramme interne 26](#_Toc369465030)

[4.6 Description des registres interne au LM4550 27](#_Toc369465031)

[4.6.1 Sélection microphone (20h) 27](#_Toc369465032)

[4.6.2 Contrôle du gain du microphone (0Eh) 28](#_Toc369465033)

[4.6.3 Sélection d’entrée (1Ah) 29](#_Toc369465034)

[4.6.4 Contrôle du gain d’entrée (1Ch) 30](#_Toc369465035)

[4.6.5 Fréquence d’échantillonnage de l’entrée (2Ah) 31](#_Toc369465036)

[4.6.6 Contrôle de volume de sortie (18h) 32](#_Toc369465037)

[4.6.7 Sortie audio 3D (20h) 33](#_Toc369465038)

[4.6.8 Volume du Line-Out (02h) 34](#_Toc369465039)

[4.6.9 Contrôle de volume Headphone (02h) 35](#_Toc369465040)

[4.7 Branchement typique 36](#_Toc369465041)

[4.8 Implémentation du contrôleur sur l’FPGA 37](#_Toc369465042)

[4.8.1 Description des composants 37](#_Toc369465043)

[5 Traitement de signal : Filtrage 41](#_Toc369465044)

[5.1 Filtre à réponse impulsionnelle finie 41](#_Toc369465045)

[5.1.1 Description d’un FIR 41](#_Toc369465046)

[5.1.2 Propriétés d’un FIR 42](#_Toc369465047)

[5.1.3 Réalisation d’un FIR 42](#_Toc369465048)

[5.2 Filtre à réponse impulsionnelle infinie 42](#_Toc369465049)

[5.2.1 Description d’un IIR 43](#_Toc369465050)

[5.2.2 Propriétés d’un IIR 43](#_Toc369465051)

[5.2.3 Réalisation d’un IIR 43](#_Toc369465052)

[5.3 Implémentation d’un filtre FIR en VHDL 45](#_Toc369465053)

[5.3.1 Création du filtre sur matlab 45](#_Toc369465054)

[5.3.2 Génération du code vhdl depuis matlab 51](#_Toc369465055)

[5.3.3 Génération du code depuis Xilinx IP CORE GENERATOR 54](#_Toc369465056)

[6 Implémentation d’un filtre RII bi quad sur la carte de développement 59](#_Toc369465057)

[6.1 Introduction 59](#_Toc369465058)

[6.2 Contexte 59](#_Toc369465059)

[6.3 L'instanciation du module de filtrage 62](#_Toc369465060)

[6.4 Teste et vérification 64](#_Toc369465061)

[7 Les outils de développement 68](#_Toc369465062)

[7.1 La gestion de versions 68](#_Toc369465063)

[7.1.1 À propos de la gestion de version 68](#_Toc369465064)

[7.1.2 Les systèmes de gestion de version locaux 68](#_Toc369465065)

[7.1.3 Les systèmes de gestion de version centralisés 69](#_Toc369465066)

[7.1.4 Les systèmes de gestion de version distribués 70](#_Toc369465067)

[7.1.5 Quelque logiciels de gestion de version : 71](#_Toc369465068)

[7.2 Git 71](#_Toc369465069)

[7.2.1 Historique 71](#_Toc369465070)

[7.2.2 Rudiment de git 71](#_Toc369465071)

[7.2.3 Base de Git : 75](#_Toc369465072)

[7.2.4 Les branches avec Git 76](#_Toc369465073)

[7.3 GitHub 79](#_Toc369465074)

[7.3.1 GitHub c’est Quoi ? 79](#_Toc369465075)

[7.3.2 Statistiques 80](#_Toc369465076)

[7.3.3 Social coding : nouvelle tendance dans le développement logiciel 80](#_Toc369465077)

[7.3.4 Comment mettre tout ça en œuvre ? 81](#_Toc369465078)

**Préface**

Presque tous les équipements modernes sont équipés de matériel audio. Ce dernier permet d'enregistrer l’audio sous forme numérique, et le stocker pour une lecture ultérieure.

Cette information numérique peut être manipulé pour changer diffèrent aspects du signal audio lors de la lecture. Plusieurs "effets" appliqué aux signaux audio numériques sont utilisé dans la vie de tous les jours en raison de la flexibilité et de la fidélité du traitement de signal numérique ainsi que la robustesse et la rapidité du matériel électronique.

Le traitement de signal numérique en temps réel exige une combinaison de matériel efficace, et des algorithmes efficaces.

Ce rapport est le résultat d'un projet qui traite les exigences et la façon pour réaliser un système de traitement audio numérique en temps réel sur une carte de développement contenant un FPGA.

# Vue d’ensemble

## **Sujet du Projet**

Le traitement du signal numérique étudie les techniques de traitement (filtrage, compression), d'analyse et d'interprétation des signaux numérisés. À la différence du traitement des signaux analogiques qui est réalisé par des dispositifs en électronique analogique, le traitement des signaux numériques est réalisé par des machines numériques telles que des ordinateurs ou des circuits dédiés.

Ce projet consiste à réaliser un système de traitement de signal audio ainsi qu’une interface d’acquisition audio pour implémenter un système de filtrage audio utilisant des filtre a réponse impulsionnel finie et infinie.

## Choix de la carte de développement

Une des raisons principales pour l’utilisation de la carte de développement DIGILENT ATLYS SPARTAN 6 était la disponibilité d’une puce entrée/sortie audio tel que le LM5450 AC97 Audio Codec qui permettra la réalisation de l’interface audio avec des sources externe. L’FPGA et le LM4550 sont reliés par une simple interface numérique composée d'une horloge 12,288 MHz généré par le codec, des données série et un signal de synchronisation pour synchroniser les données entre le FPGA et codec.

La plateforme contient 4 entrées jack audio 1/8’’ et une dizaine de switches et de boutant poussoir permettant l’interaction avec l’FPGA et 8 leds pour la visualisation.

La carte propose aussi une interface USB pour la programmation et le débogage ainsi qu’un logiciel (ADEPT), qui facilite le transfert du programme synthétisé sans avoir recours à d’autre équipement matériel tel que JTAG un peu plus complexe.

# Signal numérique audio

## Propriété physique d’un signal sonore

Le signal sonore numérique est la représentation de sons naturels (vagues de vibration dans une moyenne d'air) comme un ensemble d'informations numériques (une série de chiffres). Le son est créé lorsque l'air est perturbé, généralement par un objet vibrant. L’objet vibrant provoque des ondulations pour varier la pression d'air. Très peu d'air se déplace, pendant que la variation de pression est propagée par la collision des molécules de l'air, semblable à la façon dont les ondulations se propagent à la surface d'eau. (Pohlmann 1993, 1-3) Ces «vagues» qui font varier la pression provoque le tympan de se déplacer d'avant et en arrière. Le mouvement est effectuée à partir de la membrane du tympan à un organe appelé cochlée qui est une série de petit os. La cochlée contient une série de plus de 10.000 poils de diffèrent tailles, qui converti ces vibrations a des impulsions nerveuse qui sont transmis au cerveau pour être décodé.

L'oreille traite deux caractéristiques du son: le volume et la hauteur. Les sons naturels se composent de plusieurs hauteurs, chacune à un volume différent. Une onde périodique simple, comme la montre la Figure 2-1, n'a qu'une seul hauteur. Le volume ou l'amplitude de l'onde sonore correspond à la quantité d'air déplacé par chaque oscillation ou cycle d'onde. La hauteur ou fréquence de l'onde correspond au nombre de cycles d'onde par seconde. Les poils de différentes tailles dans la cochlée de l'oreille répondent aux besoins spécifiques des fréquences présentes dans l'onde sonore. La quantité de vibration de cheveux et de l'intensité de l'influx nerveux résultante est proportionnelle à l'amplitude d’une fréquence particulière. Figure 2-2 illustre un exemple simple de deux fréquences dans un son unique combinée en autre terme une forme d'onde.

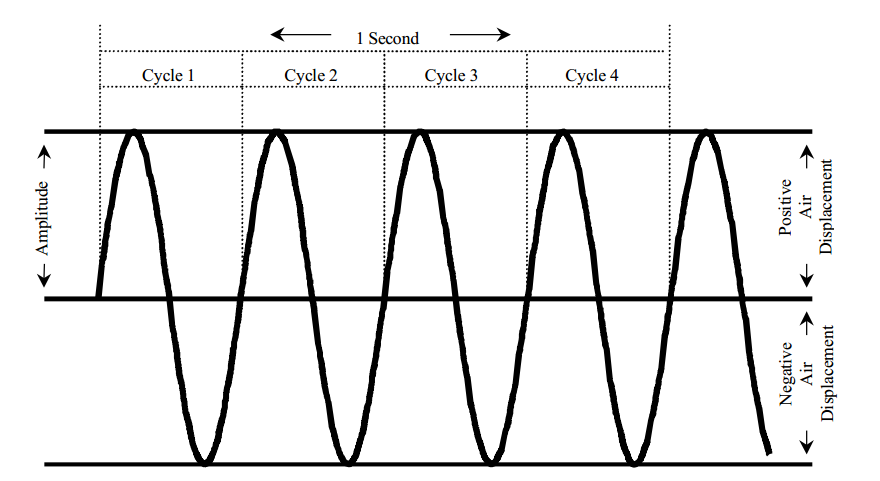


Figure ‑ : Amplitude et fréquence d’une onde cyclique simple

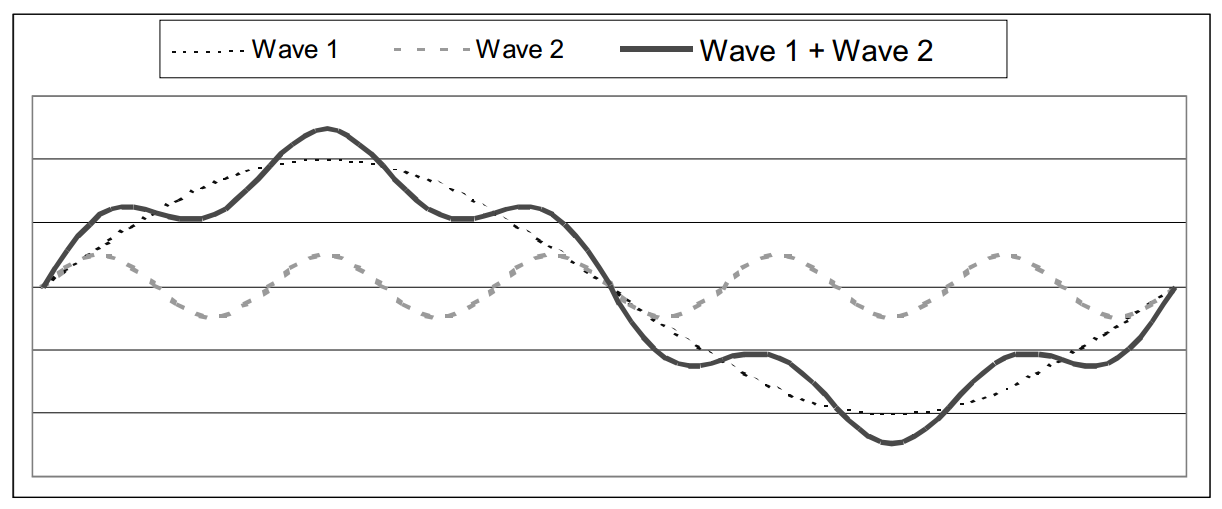


Figure ‑ : Combinaison de deux ondes de différentes fréquences

Le traitement du signal numérique sonore est l'analyse et la manipulation de données audio et de forme d'onde, ce qui peut devenir une tâche complexe. Comme beaucoup de représentations informatiques de phénomènes du monde réel, la modélisation des ondes sonores peuvent être effectués à n'importe quel niveau de complexité désiré. La complexité du signal acoustique est principalement due à la complexité du monde naturel. Comme L’illustre la Figure 2-3, un petit échantillon d'un signal de la voix humaine est très complexe et n'est pas complètement périodique. En outre, les propriétés physiques des objets dans une pièce et l’air lui-même déterminé quelles fréquences sont réfléchies où absorbés. La plupart des gens peuvent écouter un son les yeux bandés et déterminer la taille de la pièce. Des gens formés peuvent même déterminer les types de matériaux qui composent les murs.



Figure ‑ : Echantillon de forme d’onde de la voix humaine

La modification des propriétés telles que l'humidité et la température, peuvent affecter les caractéristiques acoustiques d'une pièce. En outre, les formes et les angles des parois d'une

chambre déterminent la direction vers lesquelles les ondes sonores rebondissent. Les ondes qui ont rebondis sur les surfaces peuvent entrer en collision, provoquant soit une addition ou l'effet d'annulation. Cet effet peut facilement reproduit si un petit objet est déposé dans un récipient d'eau. L'eau n’aura le même motif d'ondulation, à cause des vagues qui sont réfléchie par les extrémités du récipient.

La taille d'un objet dans une pièce peut bloquer ou plier des fréquences différentes. Par exemple: tenir un ordinateur portable entre vous et votre chaîne hi-fi bloquera les fréquences les plus élevées, mais vous serez toujours en mesure d'entendre les plus basse

Fréquences. La mesure dans laquelle un ordinateur peut analyser ou de générer des ondes sonores est donc dépendante de la complexité de l'environnement modélisé.

Générer un son d'un orchestre complet proche de la perfection exigerait des connaissances extrêmement détaillée des instruments et de l'environnement physique. Un tel modèle complexe devra alors suivi de la réflexion, de flexion et de l'interaction des ondes sonores dans la pièce.

## Représentation numérique du son

Pour un système numérique de traitement de l'audio, un procédé de conversion audio vers le domaine de l'information numérique est requise. Le format le plus commun de la représentation numérique des informations audio est Pulse Code Modulation.

Typiquement, les ondes sonores sont converties en une série de nombres (PCM) en tant que:

Une onde sinusoïdale simple, sera utilisée comme un exemple. Une telle onde serait générer par un objet vibrant selon un motif sinusoïdal (similaire au modèle fait par un coup de sifflet). La droite passante par le centre de la figure 2-4 représente la pression atmosphérique normale. Les portions la courbe en haut et en bas de la ligne centrale représente les changements positif et négatif de la pression.

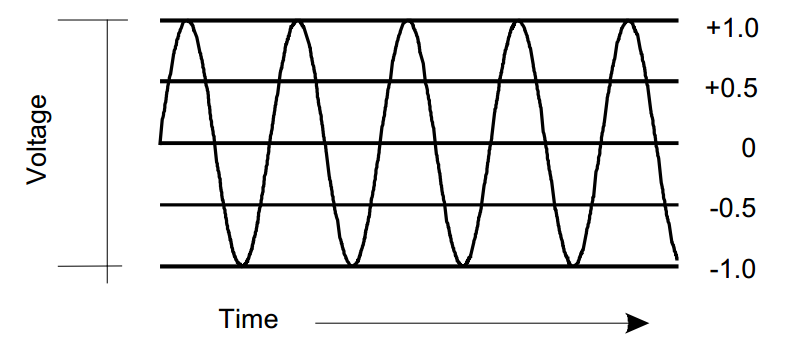


Figure ‑ : Onde sonore original

Ensuite, un microphone est utilisé pour convertir le signal audio (dans l'air) à un signal électrique. La gamme de sortie du microphone est de ± 1 volt Figure 2-5.

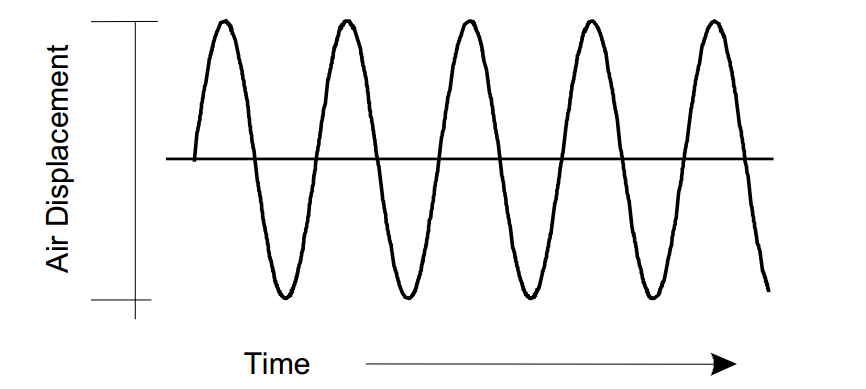


Figure ‑ : onde électrique

Le signal électrique analogique est ensuite converti en valeurs numériques par un dispositif appelé un convertisseur analogique-numérique. Un convertisseur analogique-numérique de 16 bits, qui présente une plage de -32768 à 32767, est illustré à la figure 2-6.

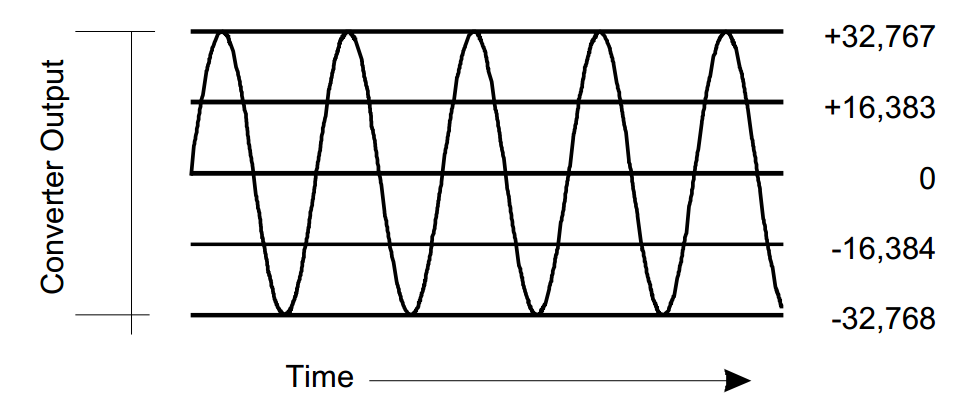


Figure ‑ :Sortie du convertisseur

En raison qu'un nombre infini de points de données ne peut pas être enregistré pour

caractériser la forme d'onde, un échantillon est prélevé à intervalles réguliers. Le nombre d'échantillons prélevés par seconde est appelée la fréquence d'échantillonnage. Sur la figure 2-4, 43 échantillons sont prélevés

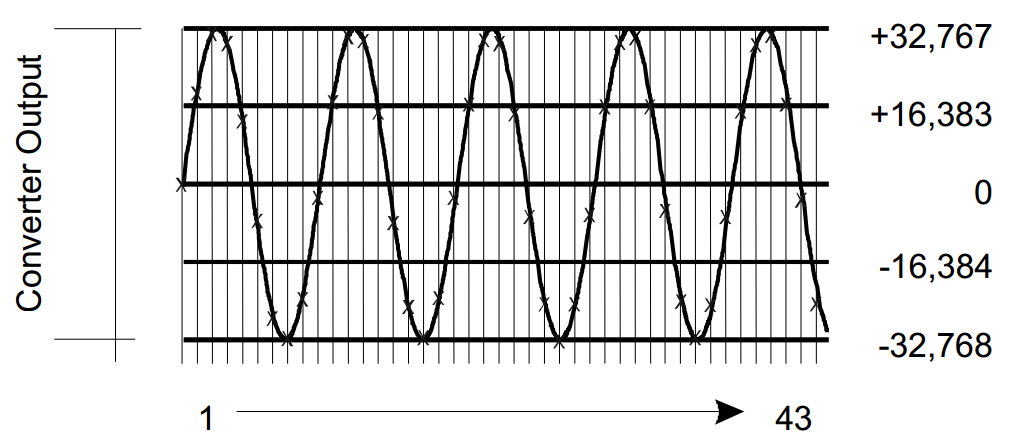


Figure ‑ : Echantillonnage de l’onde

La série résultante de 43 numéros représente la position de l'onde à chaque intervalle, comme le montre la Figure 2-8.

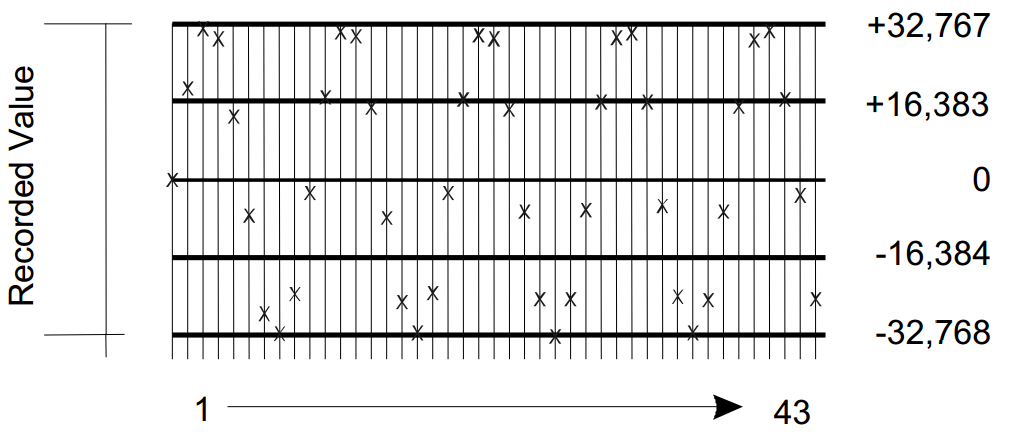


Figure ‑ : Représentation des valeurs échantillonnées

Ensuite la forme d'onde peut reconstruite en reliant les données (points). La forme d'onde résultante est illustrée dans la Figure 2-9.

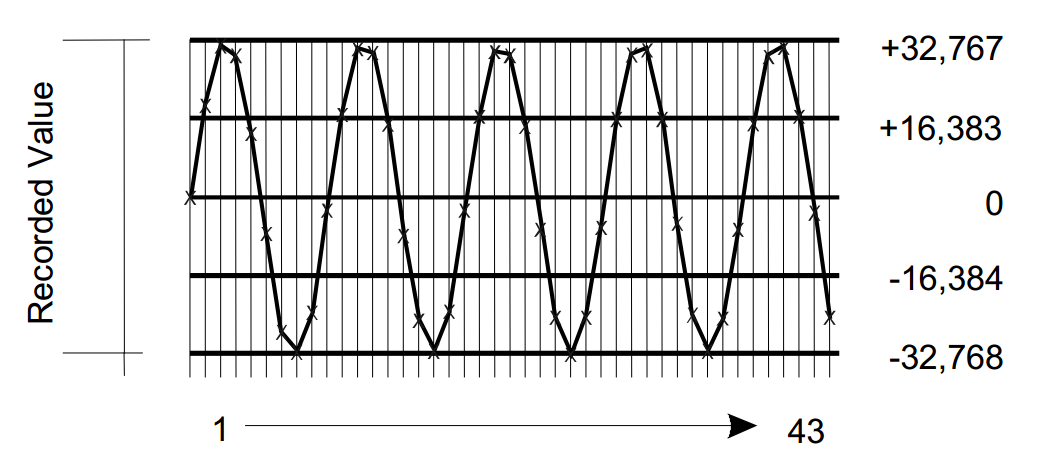


Figure ‑ : Reconstitution du signal

Notez qu'il existe certaines différences entre le signal original et celui reconstruit (Figure 2-4 et Figure 2-9):

Les valeurs que le convertisseur analogique-numérique génère sont des entiers

donc ils sont arrondis.

La reproduction précise de la forme de l'onde dépend du nombre d'échantillons enregistrés.

En général, toute série finie de nombres (représentation numérique) ne peut représenter qu’une onde analogique avec une précision finie. Cela est important de noter que la plupart des matériels audio ne produit pas reconstruite des formes d'onde dans un mode linéaire (comme l'a montré en Figure 2-9). Les caractéristiques électriques du convertisseur numérique-analogique, qui convertit les points de données correspondant à des niveaux de tension, aboutissent généralement à une forme d'onde courbé modérément lisse.

## Fréquences d'échantillonnage:

Lors de la conversion d'un son vers un signal numérique, une question importante est le nombre d'échantillons par seconde à prendre. Le taux d'échantillonnage nécessaire dépend de la fidélité requise.

Selon Nyquist et Shannon[[1]](#footnote-1), la fréquence d'échantillonnage détermine la fréquence maximal d'informations préservée. Pour créer une onde de fréquence de F nécessite 2F échantillons par seconde. Toutefois, la fréquence de Nyquist n’est pas suffisant dans la plupart des cas. Si une onde sinusoïdale de 500 Hz (cycles par seconde) est échantillonnée à une fréquence de 1000 Hz, il est possible que tous les échantillons puissent être pris lorsque le signal est à un "croisement zéro" échantillonner à des taux légèrement plus élevés que 2F peut causer certains effets "étranges" qui peut causer la variation de l’amplitude et ajouté bruit.

En général, plus le taux d'échantillonnage est élevé, plus les composantes de basse fréquence d’une onde seront reproduites.

Il est possible que la sélection d'un taux d'échantillonnage puisse avoir des répercussions plus importantes sur le système de traitement numérique.

Si la fréquence d'échantillonnage est trop faible, les acquisitions seront trop espacées et, de ce fait, si le signal original comporte des détails pertinents entre deux positions de capture, ceux-ci seront perdus.

Plus la fréquence d'échantillonnage est élevée, et plus la transmission coûte en puissance de traitement, en capacités de transmission, ou en espace de stockage.

Pour choisir une fréquence d'échantillonnage qui soit juste suffisante, il faut donc nécessairement déterminer la bande passante de l'information codée dans le signal à transmettre.

Pour la transmission de la parole avec une intelligibilité suffisante (on comprend tous les mots), on estime qu'une bande passante de 160 Hz à 3 500 Hz est suffisante.

Pour transmettre l'ensemble des signaux auditifs, y compris pour les personnes ayant l'ouïe la plus fine, on estime qu'une bande passante de de 20 Hz à 20 000 Hz est suffisante.

Si les signaux ayant la fréquence la plus élevée sont transmis, à plus forte raison, ceux de fréquence inférieure, décrits par plus d'échantillons, seront également transmis.

Le théorème de Shannon indique que toutes les fréquences du signal dont la différence avec un multiple de la fréquence d'échantillonnage est égale sont codées identiquement et impossibles à distinguer. Par conséquence, si la bande passante du signal est limitée à moins de la moitié de la fréquence d'échantillonnage, il peut être parfaitement reconstitué. Les fréquences supérieures à la moitié de la fréquence d'échantillonnage introduisent un recouvrement spectral également appelé repliement.

Pour échantillonner efficacement, il faudrait donc :

Limiter strictement la bande passante du signal à la partie qui code l'information

Choisir une fréquence d'échantillonnage égale à deux fois la fréquence supérieure de la bande passante.

Comme on ne peut pas limiter la bande passante rigoureusement, mais seulement atténuer fortement à partir d'une certaine fréquence, on doit en réalité :

Construire un filtre qui rejette le plus efficacement possible les fréquences au-delà de la limite supérieure de la bande passante du signal.

Choisir une fréquence d'échantillonnage supérieure à deux fois la fréquence supérieure de la bande passante, de telle sorte que les fréquences inutiles à l'information, mais présentes dans le signal, qui seront repliées sur le signal reconstitué, soient suffisamment atténuées par le filtre pour ne pas être gênantes.

# Interface audio lm4550

## Introduction

La carte de développement atlys Spartan 6 contient un contrôleur audio nationale Semiconductor LM4550 AC '97 audio Codec avec quatre 1/8 " prises audio pour line-out, sortie casque, line-in et microphone. Les données audio sont de 18 bits et la fréquence Échantillonnage maximale est de 48 kHz, l’enregistrement audio et le palyback, avec différents fréquences d’échenillage. Le microphone est mono tandis que toutes les autre entrées/sorties sont stéréo. La prise casque est entrainé par un amplificateur 50mW interne au codec.

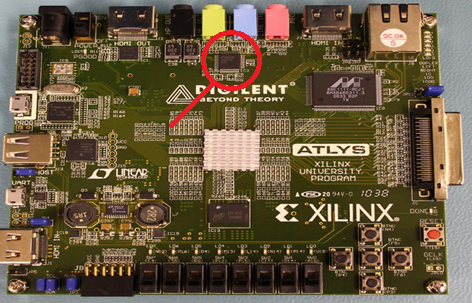


Figure ‑ : carte de développement atlys saprtan 6

## Description des entrée/sortie du codec audio :

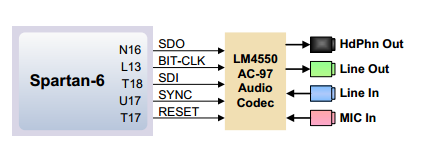


Figure ‑ : Interface du lm4550 avec l’FPGA

|  |  |  |
| --- | --- | --- |
| Nom du signal | Pin sur FPGA | Fonctionnalité |
| AUD-BITCLK | L13 | Sortie d’une horloge de 12.888 Mhz, généré par une demi-période du Crystal (XLT\_IN) de 24.576 Mhz. |
| AUD-SDI | T18 | (Serial Data In) entrée sur l’FPGA. SDI consiste d’une trame d’entrée AC97 qui contient à la fois la configuration et le data PCM audio. SDI est prélevé sur front montant de AUD-BITCLK. |
| AUD-SDO | N16 | (Serial Data Out) sortie de l’FPGA. SDO consiste d’un trame de sortie AC97, qui contient à la fois la configuration et le data du DAC. SDO est prélevé sur front descendant de AUD-BITCLK. |
| AUD-SYNC | U17 | SYNC définit les limites de trame de liaison AC97. Chaque trame dure 256 périodes de AUD-BIT-CLK. SYNC est normalement une impulsion positive 48kHz avec un rapport cyclique de 6,25% (16/256). SYNC est prélevé sur le front montant de AUD-BITCLK, et le codec prend le premier échantillon positif de SYNC repère  du début d'une nouvelle trame de liaison AC.  Si une impulsion de synchronisation ultérieure  survient dans les 255 AUD-BITCLK périodes de du début de le trame, elle sera ignorées. SYNC est également utilisé comme une entrée active haute pour effectuer une  réinitialisation à chaud (asynchrone). La réinitialisation à chaud est utilisée pour effacer un état du powerdown sur l'interface de liaison codec AC. |
| AUD-RESET | T17 | Réinitialisation à froid. Ce signal actif bas provoque une réinitialisation matérielle qui remet les registres de contrôles et tous les circuits internes à leur valeur par défaut.  le LM4550 doit initialiser après l’alimentation |

Tableau ‑ : Description des entrée/sortie du lm4550

## Les registres internes du LM4550 :

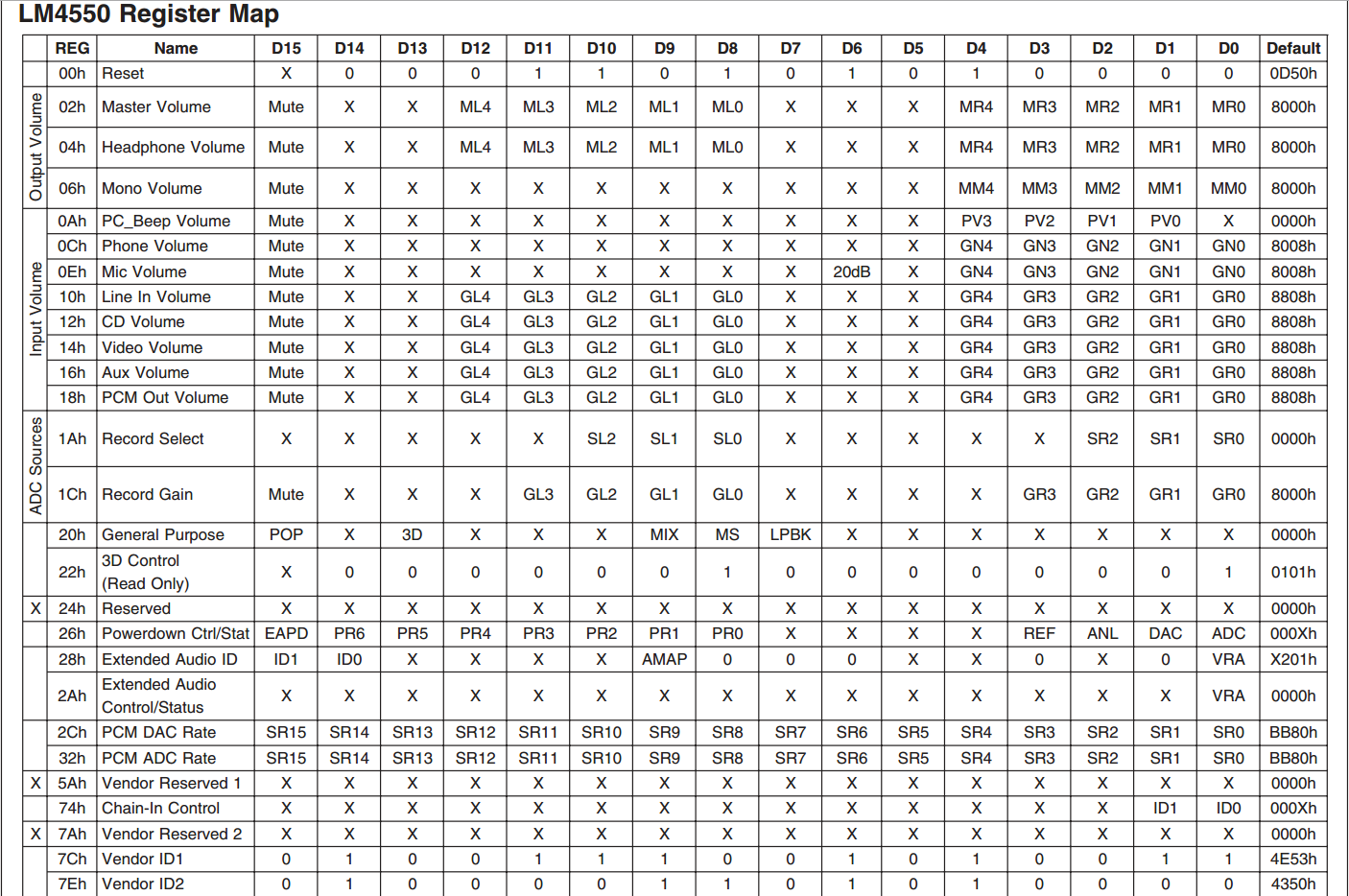


Figure ‑ : Registres internes du lm4550

## Description du fonctionnement

Le codec LM4550 peut Mixer, traiter et convertir entre analogique (stéréo et mono) et des entrées numériques (format de référence AC). Il contient quatre entrées stéréo et quatre entrées mono analogiques et deux sorties stéréo et une analogique mono. Un seul codec prend en charge des flux de données sur les deux entrées et deux canaux de sorties de l'interface numérique de référence AC simultanément.

#### Entrées et sorties ADC

Les quatre entrées stéréo analogiques et les trois entrées mono analogiques peuvent être sélectionnées pour la conversion de 18 bits avec l’ADC stéréo. La sortie numérique de la voie gauche et celle de droite est toujours situé respectivement dans les slot 3 et 4 de la trame AC. Le niveau d'entrée des deux canaux ADC peut être muté ou modifié à partir du registre de gain d'enregistrement, 1Ch. Des ajustements sont à 1,5 dB sur une plage de gain de 0 dB à +22,5 DB. L’entrée de sélection pour le DAC à travers le registre de sélection de multiplexage commandé à partir du registre 1Ah, avec la sélection du microphone commandé par le bit MS (D8) dans le registre d'usage général, 20h. Une des entrées stéréo, CD\_IN, utilise une interface à 3 broches quasi-différentielles où les deux Entrées des canaux stéréo sont référencées à la troisième broche, CD\_GND. CD\_GND AC doit être couplé à la source et fournit une rétroaction en mode commun pour

annuler le bruit de fond. Il n'est pas une prise de terre DC. Les trois autres entrées stéréo, LINE\_IN, AUX et vidéo sont des interfaces 2 broches,

### Les sorties stéréo

Le volume de sortie de LINE\_OUT et HP\_OUT peut être mute ou régler par 0 dB à 45 dB avec un pas de 3 dB sous le contrôle du volume de registres de sortie principal Volume (02h) et volume du casque (04h) respectivement. Comme avec les registres de volume d'entrée, les modifications des niveaux des deux canaux stéréo peuvent être prises indépendamment mais les canaux gauche et droite partagent un bit muet (D15).

### Les sorties mono

La sortie mono (MONO\_OUT) est entraînée par l'un des deux signaux sélectionnés par le bit de MIX (D9) dans le registre d’usage général, 20h. Le signal sélectionné par défaut (Mix = 0) est la sommation des deux canaux de mixage stéréo 3D, la sortie stéréo du MIX1. Mettre le bit MIX de contrôle = 1, sélectionne une entrée microphone, MIC1 ou MIC2. Le choix d' microphone est contrôlé par la sélection microphone (MS) bit (D8) également dans le registre à usage général, 20h.

### Le loopBack analogique et numérique

Le loopback analogique fait référence à un chemin de signal tout-analogique à partir d’une entrée analogique par l'intermédiaire des mélangeurs à une sortie analogique. Le loopaBack numérique se réfère à une conversion analogique-mode mixte et un trajet numérique du signal d'une entrée analogique via le ADC, et boucler (LPBK bits - D7, 20h) par le DAC et les mélangeurs à une sortie analogique.

### Réinitialisation(Reset)

Réinitialisation à froid. Ce signal actif bas provoque une réinitialisation matérielle qui renvoie les registres de contrôle et tous les circuits internes à leur valeur par défaut. Le lm4550 doit être réinitialisé pour initialiser le LM4550 après l’alimentation.

### Protocole d’interface série de la trame AC-LINK

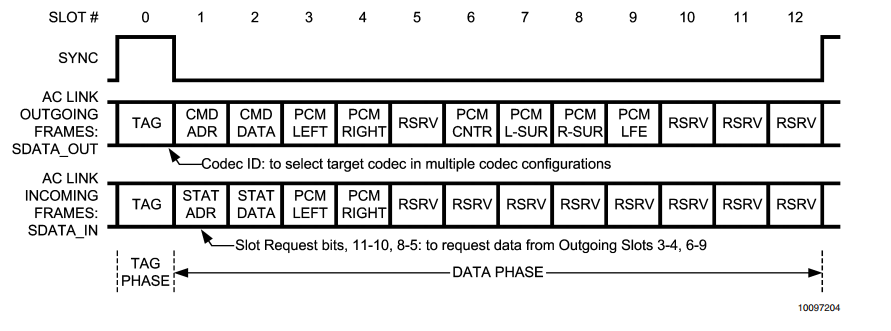


Figure ‑ : Trame Ac-Link

### Trame de sortie (SDATA\_OUT)

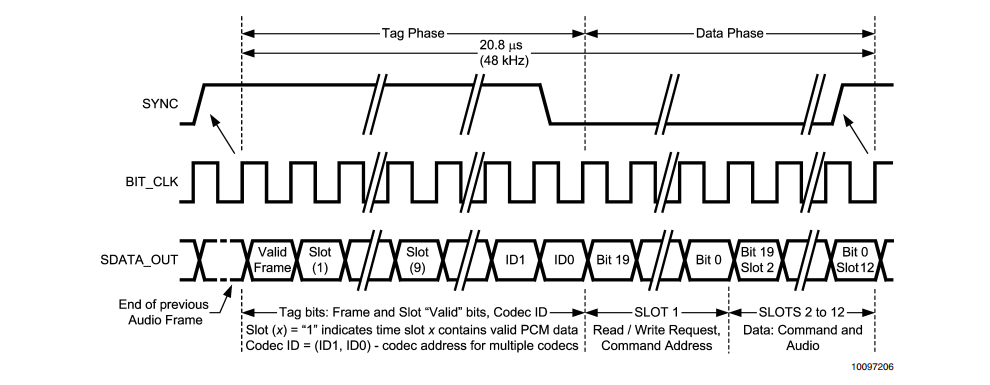


Figure ‑ : Trame de sortie

La trame AC transporte les données de contrôle ainsi que la donne PCM. Pour les registres de contrôle du LM4550 et du DAC stéréo. Les trames de sorties sont sur le signal SDATA\_OUT qui est une sortie du Contrôleur et une entrée au codec l'LM4550. Comme le montre la figure X, les trames de sortie sont constituées à partir de treize slot : un slot pour le Tag suivie de douze slots de données. Chaque trame est composée de 256 bits avec douze slot de données contenant 20 bits. Les trame d’entrée et de sorties sont alignés sur la même transition SYNC. LLM4550 accepte uniquement les données dans huit des douze slots de données, puisqu'il s'agit d'un codec de deux canaux seulement à 4 simultanément - 2 pour le contrôle, et deux pour chacune des données PCM à gauche et à droit du DAC.

Une nouvelle trame de sortie est signalée par une transition d’un niveau bas vers haut de SYNC. SYNC devrait être bloqué par le contrôleur sur front montant de BIT\_CLK et, comme le montre la figure 4, et Figure 5, le premier bit dans la trame est *"trame valide"* il devrait être synchronisé par le contrôleur sur front montant suivant du BIT\_CLK et échantillonné par le LM4550 sur front descendant. Le contrôleur AC97 doit toujours synchroniser SDATA\_OUT sur front montant de BIT\_CLK et l' LM4550 toujours échantillons SDATA\_OUT sur le prochain front descendant. SYNC est échantillonné sur le front montant de BIT\_CLK.

Le LM4550 vérifie chaque trame pour s’assurer que les 256 bits sont reçue. Si une nouvelle trame est détecté avant que les 256 bits sont reçus à partir de l'ancienne trame, la nouvelle trame est ignorée soit les données sur SDATA\_OUT sont ignorées jusqu'à ce qu'une nouvelle trame valide sera détecté.

Le LM4550 s'attend à recevoir des données MSB en premier, dans un MSB Format justifié.

#### Slot 0  - le TAG

Le premier bit du SLOT 0 est désigné pour le bit "trame valide". Si ce bit est à 1, cela indique que la trame de sortie de courante contient au moins un SLOT de données valides et de le LM4550 vérifiera les autres bits du TAG pour des données valides dans les SLOTs de données attendu. En mode primaire, le contrôleur indiquera la validé des données dans le SLOT en mettant le bit du TAG associé à 1. Puisqu'il s'agit d'un codec à deux canaux le LM4550 peut que recevoir les données seulement de quatre SLOT dans le cadre d’une trame et ainsi il vérifie le bit de validité que pour 4 SLOTs. Dans le mode primaire ces bits du TAG sont pour: SLOT 1 (adresse de la commande), emplacement 2 (donnée de la commande), la SLOT 3 (données PCM pour le DAC du canal gauche) et emplacement 4 (données PCM pour le DAC du canal droit).

Les deux derniers bits dans le TAG contiennent l'identifiant du codec utilisé pour sélectionner le codec cible pour recevoir la trame dans le cas d’une configuration a multiple codec. Lorsque la trame est envoyée à un codec dans l'un des modes secondaires, le contrôleur n'utilise pas les bits 14 et 13 pour indiquer l'adresse de la commande valide et les données dans les SLOTs 1 et 2. Au lieu de cela, ce rôle est assumé par les bits d'identification du codec le fonctionnement de la trame AC-LINK suppose que le contrôleur n’accédera pas à un codec secondaire seulement si une adresse de commande valide et /ou données a été fourni. Lorsque vous envoyez une trame de sortie à un codec en mode secondaire, un contrôleur devrait fixer bits d'étiquette 14 et 13 à zéro.

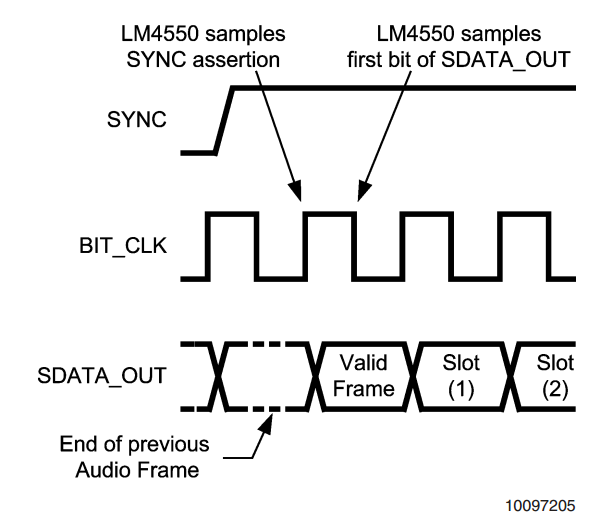


Figure ‑ : Début d’une trame AC-LINK de sortie

|  |  |  |
| --- | --- | --- |
| Bit | Description | Utilisation |
| 15 | Trame valide | 1 = Il y a au moins un SLOT de données valide |
| 14 | Registre control d’adresse | 1 = Adresse de contrôle valide dans le SLOT 1  (Mode primaire seulement) |
| 13 | Registre control de données | 1 = Données de contrôle valide dans le SLOT 2  (Mode primaire seulement) |
| 12 | Données du DAC gauche dans SLOT 3 | 1 = Données PCM valide dans le SLOT 3  (Mode primaire et secondaire 1 ; canal gauche) |
| 11 | Données du DAC droit dans SLOT 4 | 1 = Données PCM valide dans le SLOT 4  (Mode primaire et secondaire 1 ; canal droit) |
| 10 | Pas utilisé | Doit être mis à 0 |
| 9 | Données du DAC gauche dans SLOT 6 | 1 = Données PCM valide dans le SLOT 6  (secondaire 2 ; canal gauche) |
| 8 | Données du DAC droit dans SLOT 7 | 1 = Données PCM valide dans le SLOT 7  (secondaire 2 ; canal droit) |
| 7 | Données du DAC gauche dans SLOT 8 | 1 = Données PCM valide dans le SLOT 6  (secondaire 3 ; canal gauche) |
| 6 | Données du DAC droit dans SLOT 9 | 1 = Données PCM valide dans le SLOT 6  (secondaire 3 ; canal droit) |
| 5 :2 | Pas utilisé | Doit être mis à 0 |
| 1.0 | Identifiant du codec (ID0, ID1) | Utiliser dans les modes secondaires. |

Tableau ‑ : Description des bits du slot 0 de la trame de sortie

#### Slot 1 - Lecture/Ecriture, Adresse de contrôle

Le Slot 1 est utilisée par le contrôleur pour indiquer à la fois l'adresse du registre de cible dans le LM4550 et si l'opération est une lecture ou écriture du registre. Le MSB du SLOT 1 (bit 19) est mis à 1 pour indiquer que l'opération actuelle est une lecture. Les bits 18 à 12 sont utilisés pour spécifier l’adresse de registre 7-bit de l'opération de lecture ou d'écriture. Les douze bits lsb sont réservés et doivent être mis à zéro.

|  |  |  |
| --- | --- | --- |
| Bit | Description | Utilisation |
| 19 | Lecture/Ecriture | 1 = Lecture  0 = Ecriture |
| 18:12 | Adresse du registre | Identifie le registre de statuts/commande pour  Lecture/Ecriture |
| 11:0 | Réservé | mis à 0 |

Tableau ‑ : Description des bits du slot 1 de la trame de sortie

#### Slot 2 – Données de contrôle

Le SLOT 2 est utilisé pour transmettre des données de commande de 16 bits vers le LM4550 lorsque l'opération d'accès est une écriture. Les quatre bits de poids faible devraient être mis à zéros. Si l'opération d'accès est un registre de lecture, tous les bits de 19 à 0 doivent être mis à zéros.

|  |  |  |
| --- | --- | --- |
| Bit | Description | Utilisation |
| 19:4 | Données de registre de contrôle | S’il s’agit d’une opération de lecture, tous les bits doivent être mis à 0 |
| 3:0 | Réservé | mis à 0 |

Tableau ‑ : Description des bits du slot 2 de la trame de sortie

#### Slot 3 et 4 – Données PCM canal droite et gauche

Les SLOTs 3 et 4 sont des SLOT de 20 bits utilisés pour transmettre des données PCM aux deux canaux gauche et droit du DAC stéréo lorsque le codec est en mode principal ou en mode secondaire 1. Tous les bits non utilisés devraient être bourré avec des zéros. Les DAC du LM4550 ont 18 bits de résolution et va donc utiliser les 18 bits les plus significatifs des 20-bit Données PCM.

|  |  |  |
| --- | --- | --- |
| Bit | Description | Utilisation |
| 19:0 | Données PCM audio (canal droit et gauche) |  |

Tableau ‑ : Description des bits du slot 3 et 4 de la trame de sortie

### Trame d’entrée (SDATA\_IN):

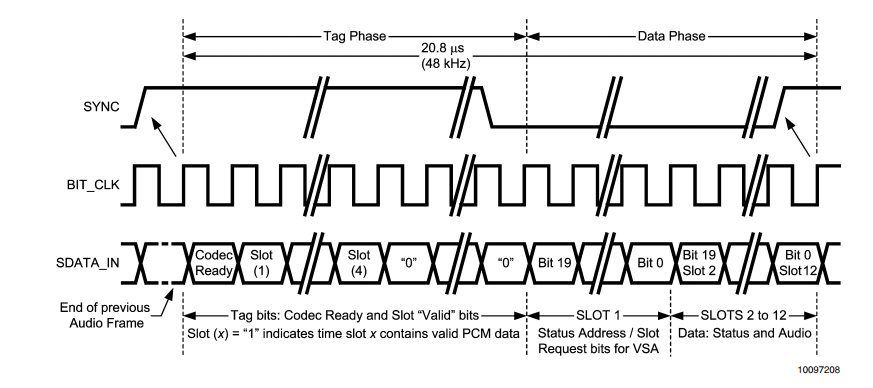


Figure ‑ : Trame d’entrée

La Trame d’entrée AC-LINK contient des données PCM et des données d'état des registres de contrôle LM4550 et de l’ADC stéréo. Les trames d'entrées sont envoyées sur le signal SDATA\_IN qui est une entrée pour le Contrôleur audio numérique et une sortie de l'LM4550 codec. Comme le montre la figure x, les trames d’entrée se constituent de treize SLOTs: un SLOT Tag suivie douze SLOTs de données. L'emplacement de Tag, emplacement 0, contient 16 bits d’une nouvelle trame d'entrée est signalée par une transition de front montant a descendant su signal SYNC. SYNC devrait être cadencé par le contrôleur sur front montant de BIT-CLK et, comme le montre la figure x, et Figure x, le premier bit dans la trame ‘’Codec Ready ‘’ est cadencé depuis LM4550 par le prochain front montant du BIT-CLK. Le LM4550 synchronise toujours les données de SDATA-IN sur un front montant de BIT-CLK et le contrôleur est prévu pour échantillonner SDATA-IN sur le front descendant suivant. Le LM4550 échantillonne SYNC sur le front montant de BIT-CLK.

L’entrée et la sortie dans les trames sont alignées sur la même transition de SYNC. Le LM4550 vérifie chaque trame pour s’assurer que les 256 bits sont reçue. Si une nouvelle trame est détectée avant que les 256 bits sont reçus à partir d’une ancienne trame, la nouvelle trame est ignorée comme si aucune donnée valide n’est envoyée sur SDATA-IN jusqu'à ce qu'une nouvelle trame valide soit détectée. Le LM4550 transmet des données MSB en premier.

#### Slot 0 - Codec/SLOT bits de statut

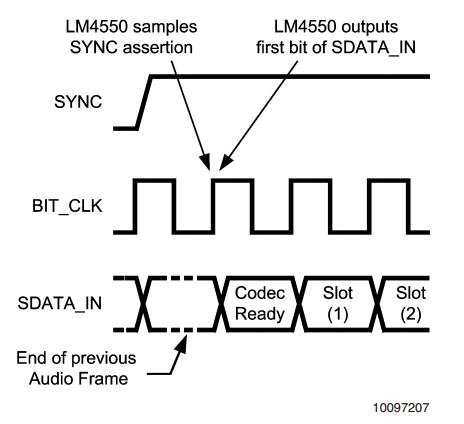


Figure ‑ : Début d’une trame AC-LINK d’entrée

Le premier bit (bit 15, "Codec Ready") du SLOT 0 dans la trame d'entrée AC-LINK indique quand le codec et ses registres d'interface sont pleinement opérationnels. Le contrôleur numérique est alors capable de lire les bits de poids faible du Powerdown Control (26h) pour déterminer le statut des quatre principales sections analogiques. Il est important de vérifier l'état de ces sections Après l'initialisation, la réinitialisation à froid ou à l'utilisation des modes du powerdown afin d' minimiser le risque de distorsion des signaux analogiques transmis avant que les sections sont prêtes. Les bits 14, 13, 12 et 11 indiquent que les données dans l’emplacement 1, 2, 3 et 4, respectivement sont valides.

|  |  |  |
| --- | --- | --- |
| Bit | Description | Utilisation |
| 15 | Codec Ready | 1 = L’interface du contrôleur est opérationnel |
| 14 | Validité des données du SLOT 1 | 1 = Adresse de statut valide ou requête du SLOT |
| 13 | Validité des données du SLOT 2 | 1 = Adresse de statut valide ou requête du SLOT |
| 12 | Validité des données du SLOT 3 | 1 = Données PCM valide (canal gauche) |
| 11 | Validité des données du SLOT 4 | 1 = Données PCM valide (canal droit) |

Tableau ‑ : Description des bits du slot 0 de la trame d’entrée

#### Slot 1 – Adresse de statut / requête de SLOT

Ce SLOT rappel (bits 18-12) l'adresse (7 bits) du registre de commande / statut su codec reçu du contrôleur comme partie d’une requête de lecture dans la trame précédente. Si aucune requête n’a été reçu, le codec met ces bits a des zéros.

Les bits 6, 11, 10, 8-5 sont des requête de SLOT qui soutiennent la capacité de l’audio c à taux variable (VRA) du LM4550. Seulement deux sont utilisés simultanément. Si le codec est en mode primaire ou en mode secondaire 1, puis les canaux gauche et droit de l'DAC vont prendre des données PCM des SLOTs 3 et 4 dans la trame de sortie respectivement.

Le codec utilise les bits 11 et 10 pour demander des données du DAC à partir de ces deux SLOTs. Si les bits 11 et 10 sont mis à 0, le contrôleur doit répondre avec des données PCM valides dans les emplacements 3 et 4 de la prochaine trame de sortie. Si les bits 11 et 10 sont mis à 1, le contrôleur ne doit pas envoyer de données. De même, si le codec est en Mode secondaire 2, les bits 7 et 6 sont utilisés pour demander des données des SLOTs 7 et 8 dans la trame de sortie. Si en mode secondaire 3, les bits 8 et 5 pour demander des données des SLOTs 6 et 9.

Le codec a le plein contrôle des bits de demande de SLOT. Par défaut, des données sont demandée à chaque trame, correspondant à un échantillon à une vitesse égale à la vitesse de défilement (fréquence SYNC) - 48 kHz lorsque XTAL\_IN = 24.576 MHz. Pour envoyer des échantillons à un taux en dessous du taux de trame, le contrôleur devrait mettre VRA = 1 (bit 0 dans le registre Control / Status, 2Ah) et programmer la vitesse désirée dans le registre de fréquence DAC PCM, 2ch. Les deux voies du DAC fonctionnent à la même fréquence d'échantillonnage. Les valeurs des fréquences d'échantillonnage sont donnés dans le Registre Description de la section (exemples registres de contrôle de la fréquence, 2ch, 32h) mais sur une fourchette comprise entre 4 kHz et 48 kHz (à une résolution de 1 Hz) est supporté.

Les requêtes de SLOT à partir du LM4550 sont déterministes. Par exemple, si un échantillon à un taux de 8000 Hz est programmé dans 2CH alors le LM4550 va toujours émettre requête de SLOT dans toutes les six trames. Les bits 9, 4, 3 ​​et 2 sont des bits de demande de SLOTs non utilisés par le LM4550 et sont rembourrés avec des zéros. Les bits 1 et 0 sont réservés et sont aussi bourré avec des zéros.

|  |  |  |
| --- | --- | --- |
| Bit | Description | Utilisation |
| 19 | Réservé | Mis à 0 pas le LM4550 |
| 18 :12 | Index du registre de statut | Reflet la requête du registre de statut |
| 11 | Bit de requête SLOT 3 | 0 = Le contrôleur doit envoyer une donnée valide  dans le SLOT 3 dans la prochaine trame de  sortie.  1 = Le contrôleur ne doit pas envoyer de données  sur le SLOT 3 |
| 10 | Bit de requête SLOT 4 | 0 = Le contrôleur doit envoyer une donnée valide  dans le SLOT 4 dans la prochaine trame de  sortie.  1 = Le contrôleur ne doit pas envoyer de données  sur le SLOT 4 |
| 9 | Bit de requête SLOT 5 | Mis à 0 par le LM4550 |
| 8 | Bit de requête SLOT 6 | 0 = Le contrôleur doit envoyer une donnée valide  dans le SLOT 6 dans la prochaine trame de  sortie.  1 = Le contrôleur ne doit pas envoyer de données  sur le SLOT 6 |
| 7 | Bit de requête SLOT 7 | 0 = Le contrôleur doit envoyer une donnée valide  dans le SLOT 7 dans la prochaine trame de  sortie.  1 = Le contrôleur ne doit pas envoyer de données  sur le SLOT 7 |
| 6 | Bit de requête SLOT 8 | 0 = Le contrôleur doit envoyer une donnée valide  dans le SLOT 8 dans la prochaine trame de  sortie.  1 = Le contrôleur ne doit pas envoyer de données  sur le SLOT 8 |
| 5 | Bit de requête SLOT 9 | 0 = Le contrôleur doit envoyer une donnée valide  dans le SLOT 9 dans la prochaine trame de  sortie.  1 = Le contrôleur ne doit pas envoyer de données  sur le SLOT 9 |
| 4 :2 | Réservé | Mis à 0 par le LM4550 |
| 1.0 | Réservé | Mis à 0 par le LM4550 |

Tableau ‑ : Description des bits du slot 1 de la trame d’entrée

#### Slot 2 – Statut

Ce SLOT renvoie les données d'état de 16 bits lus à partir d'un contrôle de codec /registre d'état. Le codec envoie les données de statut dans la trame suivant la demande de lecture par le dispositif de commande. Si aucune lecture demande a été faite dans la trame précédente le codec me ce SLOT a zéros.

|  |  |  |
| --- | --- | --- |
| Bit | Description | Utilisation |
| 19 :4 | Statut | Données lu du registre d’état. |
| 3 :0 | Réservé | Mis à 0 par le LM4550 |

Tableau ‑ : Description des bits du slot 2 de la trame d’entrée

#### Slot 3 – Données PCM canal gauche

Ce SLOT contient des données échantillonnées à partir de la voie de gauche de l'ADC stéréo. Le signal à numériser est sélectionnée en utilisant le registre de sélection d’enregistrements (1 Ah) et par la suite acheminé à travers le registre de sélection de multiplexage et l'amplificateur a gain vers l’ADC.

Le SLOT est de 20-bit et les données PCM 18 bits est transmis dans le poids fort du SLOT. Les deux bit 2 qui restent sont toujours à 0.

|  |  |  |
| --- | --- | --- |
| Bit | Description | Utilisation |
| 19 :2 | Données PCM canal gauche | 18 bit de données numérisés par l’ADC gauche |
| 3 :0 | Réservé | Mis à 0 par le LM4550 |

Tableau ‑ : Description des bits du slot 3 de la trame d’entrée

#### Slot 4 – Données PCM canal droit

Ce SLOT contient des données échantillonnées à partir de la voie de droite de l'ADC stéréo. Le signal à numériser est sélectionnée en utilisant le registre de sélection d’enregistrements (1 Ah) et par la suite acheminé à travers le registre de sélection de multiplexage et l'amplificateur a a gain vers l’ADC.

Le SLOT est de 20-bit et les données PCM 18 bits est transmis dans le poids fort du SLOT. Les deux bit 2 qui restent sont toujours à 0.

|  |  |  |
| --- | --- | --- |
| Bit | Description | Utilisation |
| 19 :2 | Données PCM canal droit | 18 bit de données numérisés par l’ADC droit |
| 3 :0 | Réservé | Mis à 0 par le LM4550 |

Tableau ‑ : Description des bits du slot 4 de la trame d’entrée

#### Slot 5 à 12

Les SLOTs de 5 à 12 ne sont pas utiliser par l’LM4550, et ils sont toujours mis à 0.

## Diagramme interne

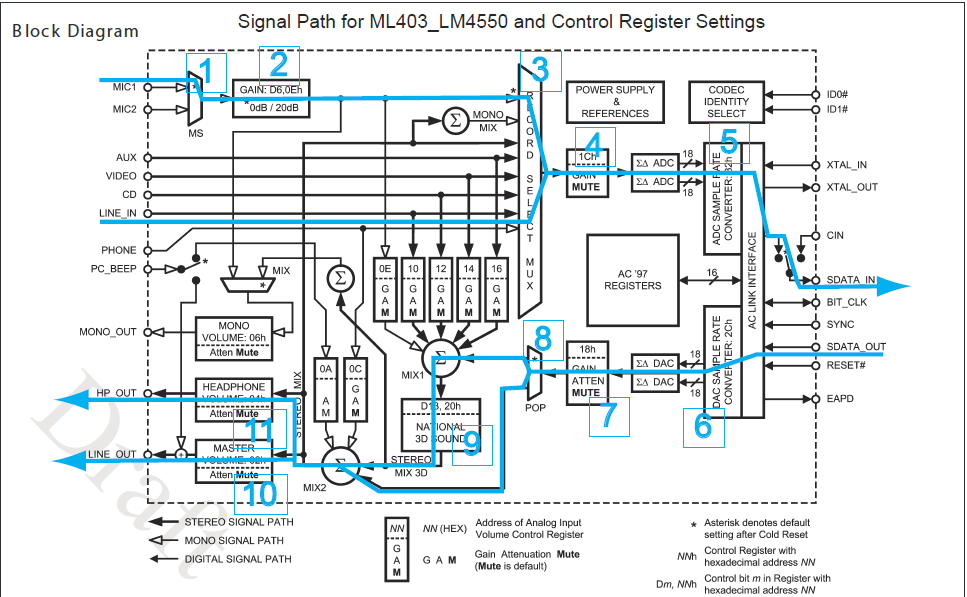


Figure ‑ : Diagramme interne du lm4550

## Description des registres interne au LM4550

La section qui suit décrit les registres a programmé pour une utilisation optimale du LM4550 dans le mode primaire.

### Sélection microphone (20h)

Les entrées microphone 1 et 2 peuvent être sélectionnées en utilisant le registre de commande à l’adresse 20h.

Figure ‑ : sélection du microphone

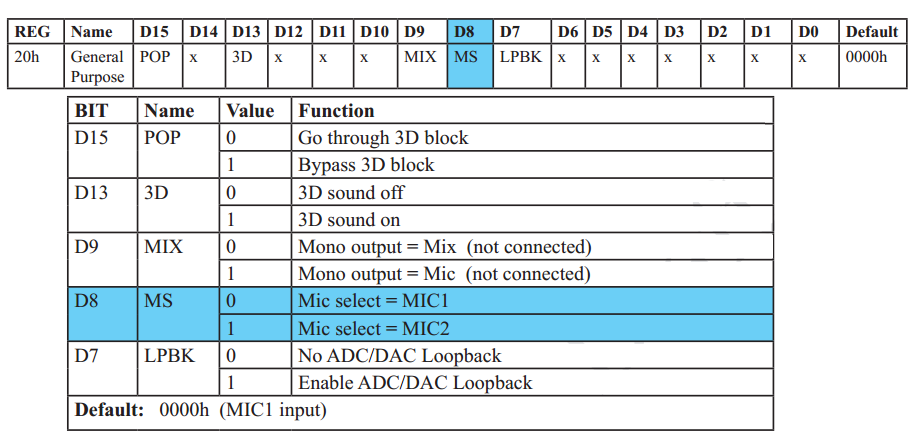


Figure ‑ : Registre 20h

### Contrôle du gain du microphone (0Eh)

L’entrée du microphone peut avoir un boost de 20 db en utilisant le bit D6 dans le registre de volume de microphone 0Eh.

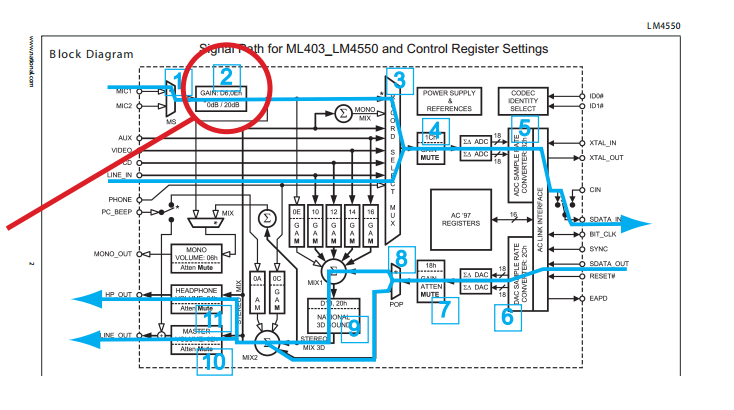


Figure ‑ : Contrôle du gain

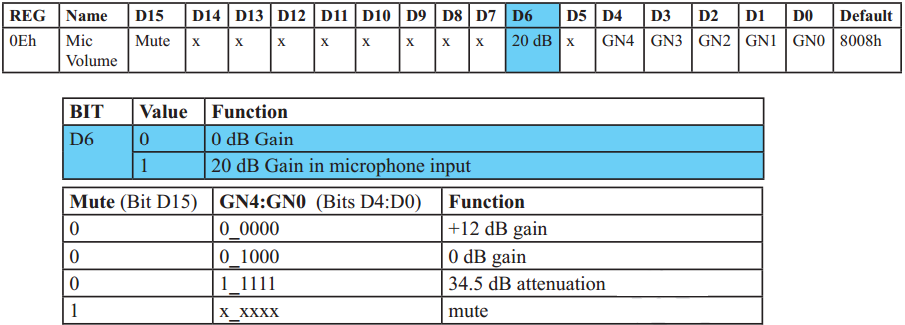


Figure ‑ : Registre 0Eh

- Valeur par défaut : 8008h (mute, 0 db de gain)

- Le Bit D6 contrôle le boost de 20dB sur l'entrée qui est acheminé à l'entrée de sélection du MUX. Les autres bits (muet et gain / atténuation) commande l’entrée du mélangeur 1.

- Les bits D4:D0 contrôle le gain de +12 dB à -34.5 dB par pas de 1,5 dB dans le mélangeur 1.

### Sélection d’entrée (1Ah)

Le multiplexeur de sélection d’entrée contrôle la sélection de l’entrée via le registre 1Ah.

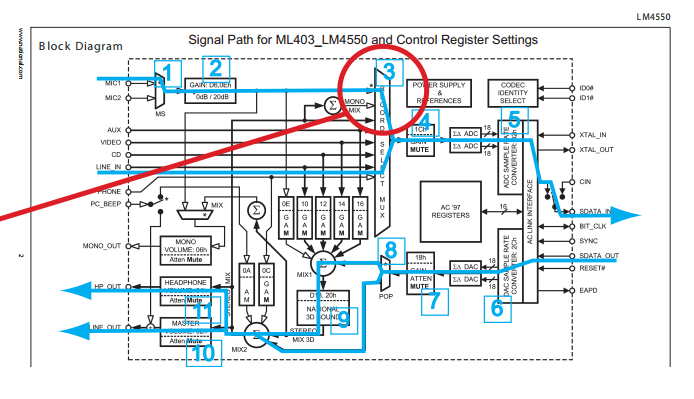


Figure ‑ : Sélection d’entrée

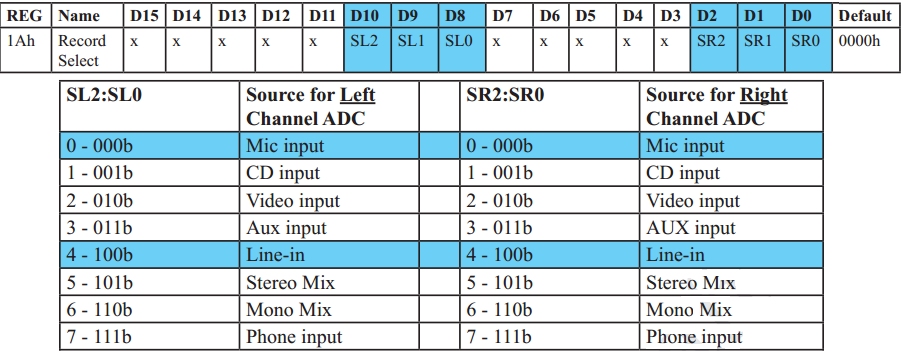


Figure ‑ : Registre 1Ah

- Valeur par défaut : 8008h (Mic sur les deux canaux droit et gauche)

- Les entrées CD, Video, AUX, Phone ne sont pas connecte.

- La sélection du canal peut être individuelle.

### Contrôle du gain d’entrée (1Ch)

L’entrée sélectionné peut avoir une amplification de gain jusqu’à 22.5 db en utilisant le registre de gain 1Ch.

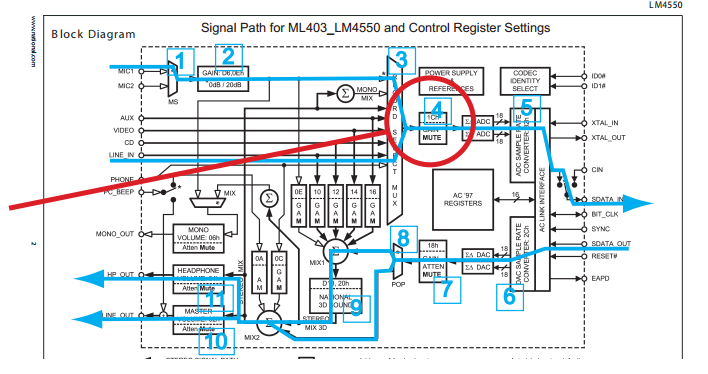


Figure ‑ : Contrôle de gain d’entrée

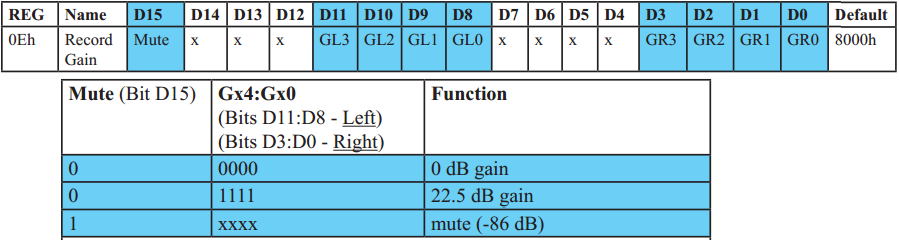


Figure ‑ : Registre 1Ch

- Valeur par défaut : 8008h (mute, 0 db de gain)

- Le bit D15 met l’entrée a muet.

- Les bits Gx4 :Gx0 contrôlent le gain de 0 à +22.5 db avec des pas de +1.5 db

### Fréquence d’échantillonnage de l’entrée (2Ah)

L'entrée peut être échantillonné à partir de 4 KHz à 48 KHz en agissant à la fois sur la fréquence d’échantillonnage de l’ADC registre 32h et sur le registre du statut/contrôle 2Ah.

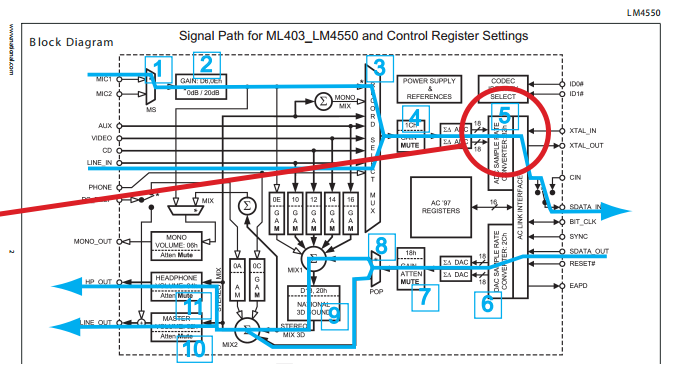


Figure ‑ : Fréquence d’échantillonnage

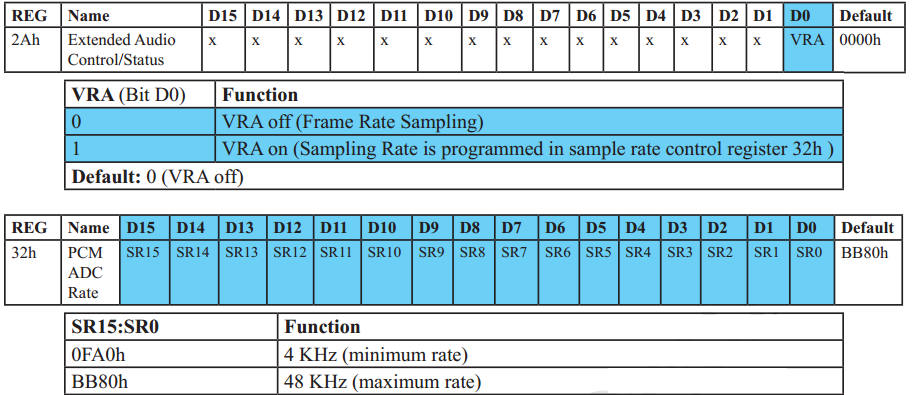


Figure ‑ : Registre 2Ah

- Valeur par défaut : BB80h (48kh)

- La fréquence d’échantillonnage peut être programmé, avec des incréments de 1 Hz pour une valeur entre 4 KHz jusqu’à 48 KHz.

### Contrôle de volume de sortie (18h)

Le volume de sortie peut contrôler via le registre de volume PCM 18h.



Figure ‑ : Contrôle de volume de sortie

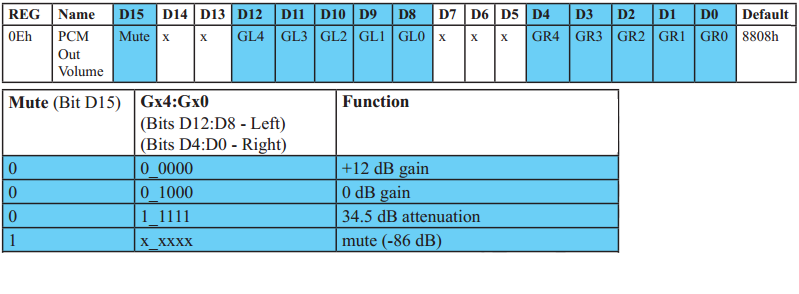


Figure ‑ : Registre 18h

- Valeur par défaut : 8808h (mute, 0 db de gain sur les deux canaux)

- Les bits Gx4 :Gx0 contrôlent le gain de +12 db jusqu’à -34.5 db avec des pas de -1.5 db.

### Sortie audio 3D (20h)

La sortie peut être router dans le bloque ‘’NATIONAL 3D SOUND ‘’ en utilisant le registre a propos générale.

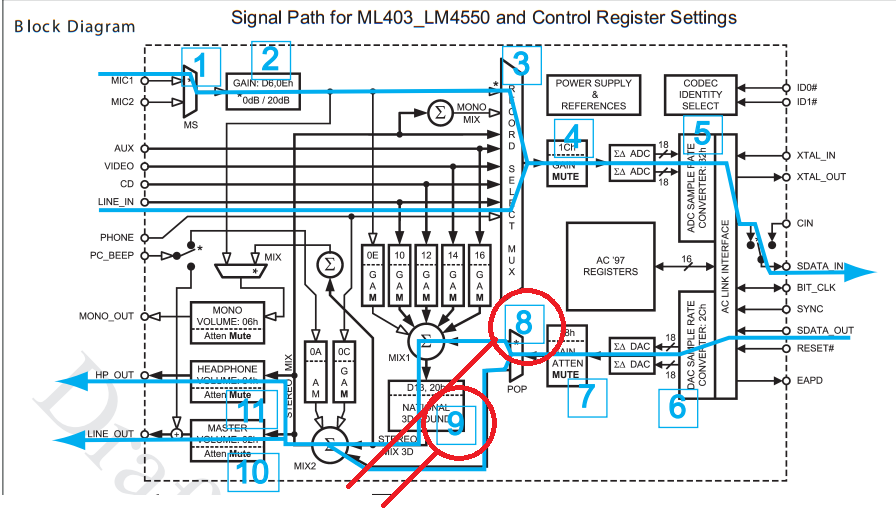


Figure ‑ : Sortie audio 3D

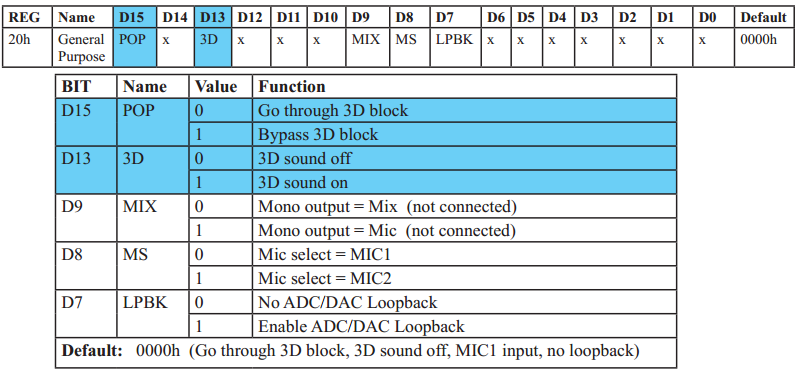


Figure ‑ : Registre 20h

### Volume du Line-Out (02h)

Le volume du line-out est régler via le registre de contrôle de volume master.

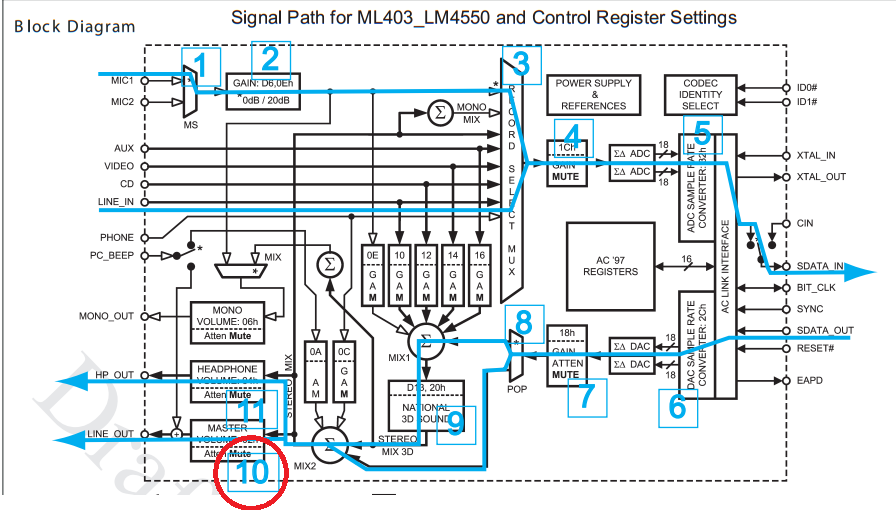


Figure ‑ : Volume du line-in

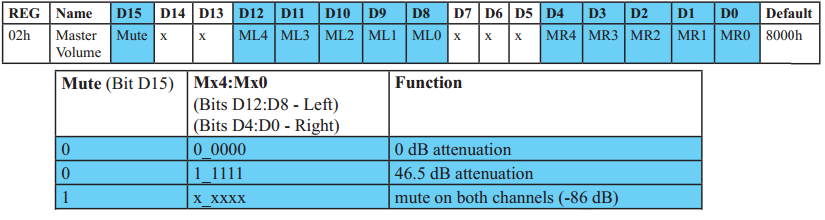


Figure ‑ : Registre 02h

- Valeur par défaut : 8808h (mute, 0 db de gain sur les deux canaux)

- Les bits Mx4:Mx0 contrôlent le gain de 0 db jusqu’à -46.5 db avec des pas de -1.5 db.

### Contrôle de volume Headphone (02h)

Le volume de la sortie headphone est régler via le registre de contrôle de volume master.

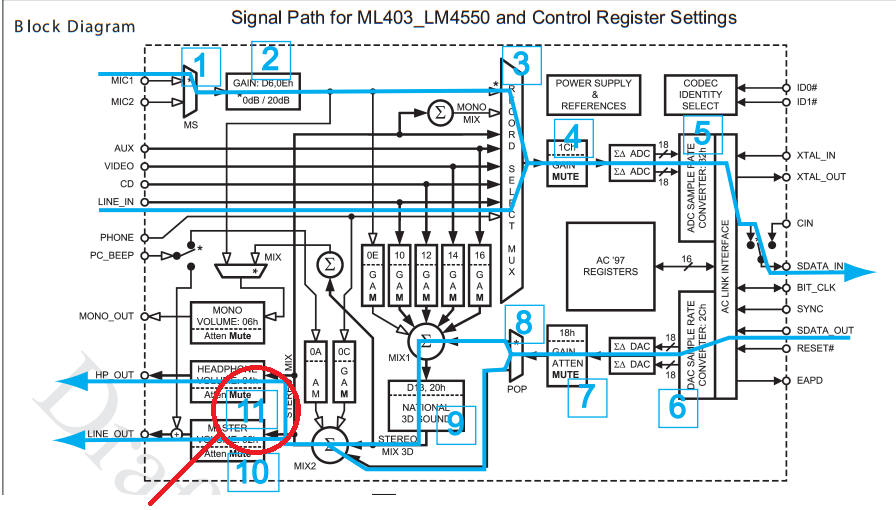


Figure ‑ : Contrôle de volume du Head phone

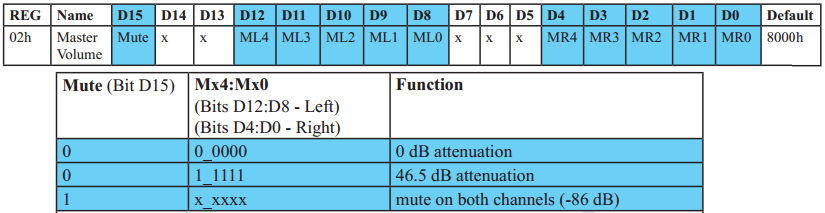


Figure ‑ : Registre 02h

- Valeur par défaut : 8808h (mute, 0 db de gain sur les deux canaux)

- Les bits Mx4:Mx0 contrôlent le gain de 0 db jusqu’à -46.5 db avec des pas de -1.5 db.

## Branchement typique

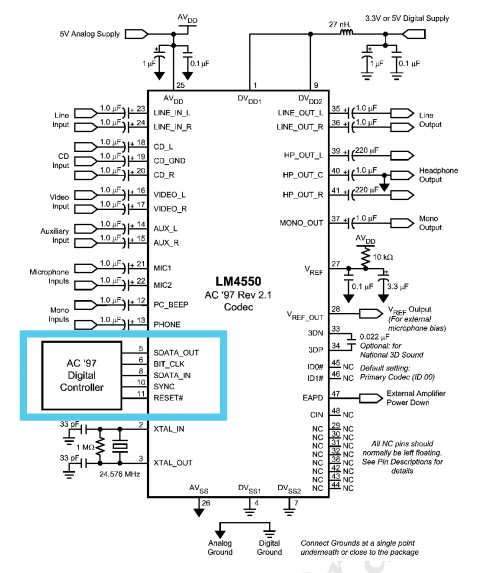


Figure ‑ : Branchement typique du lm4550

La figure 4-28 représente un branchement typique du LM4550, la partie encadré en bleu noté *‘’AC97 digital controller’’* est la partie qui doit être implémenté sur l’FPGA pour pouvoir communiquer avec le codec pour envoyer et recevoir des données audio PCM.

## Implémentation du contrôleur sur l’FPGA

Cette partie s’intéresse à l’interfacer du codec audio LM4550 avec un FPGA qui roule à une vitesse de 100 MHz. Le design peut être adapté à d'autres vitesses par la mise à l'échelle des compteurs internes, ou l'instanciation d'un PLL embarquée pour atteindre une horloge de 100 MHz. Un Spartan 6 FPGA est utilisé pour développer le contrôleur AC'97 qui se trouve sur carte de développement digilent atlys, toute fois n’importe quel FPGA peut être utilisé à condition que la fréquence du signal soit prise en compte par rapport à l'horloge de système principale.

### Description des composants

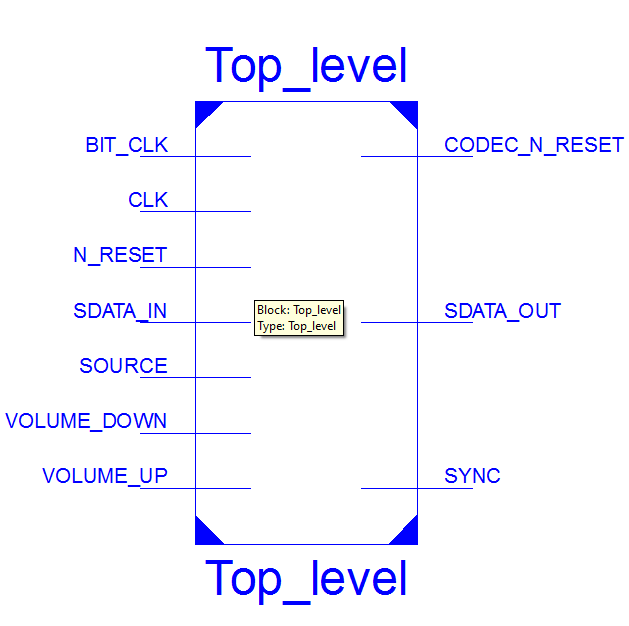
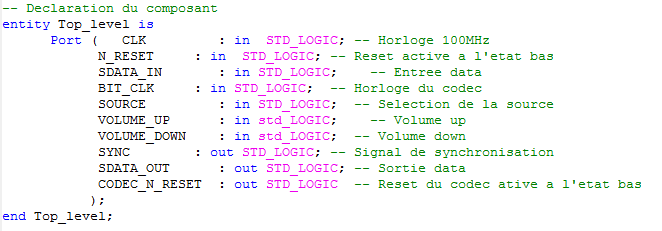


Figure ‑ : Vue rtl du Top level

Les entrées du composant comprennent l'oscillateur principal de l’FPGA (CLK), une réinitialisation (reset) actif à l’état bas, entrée de données série (SDATA\_IN), un signal d'horloge parvenant du codec AC'97 de 12,288 MHz (BIT\_CLK), un sélecteur de source (SOURCE\_SELECT) et une commande de volume de (VOLUME\_UP/VOLUME\_DOWN). Les comprennent un signal de synchronisation (SYNC), la sortie de données série (SDATA\_OUT), et un signal de réinitialisation (CODEC\_RESET) actif à l’état bas pour initialiser le codec.



Snapshot code Source ‑ : Déclaration du top level

Le composant comporte deux parties principales, le contrôleur audio pour produire les différents signaux, et la conversion de données série en 18 bits de données parallèles pour l'interfaçage et le traitement de signal, et une machine d'état qui permet de configurer les différents registres du codec dans un mode round robin. La machine d’état peut être modifiée pour inclure l’utilisation du wishbone.

Les deux parties du pilote sont synchronisés avec un seul cycle du signal impulsionnel.

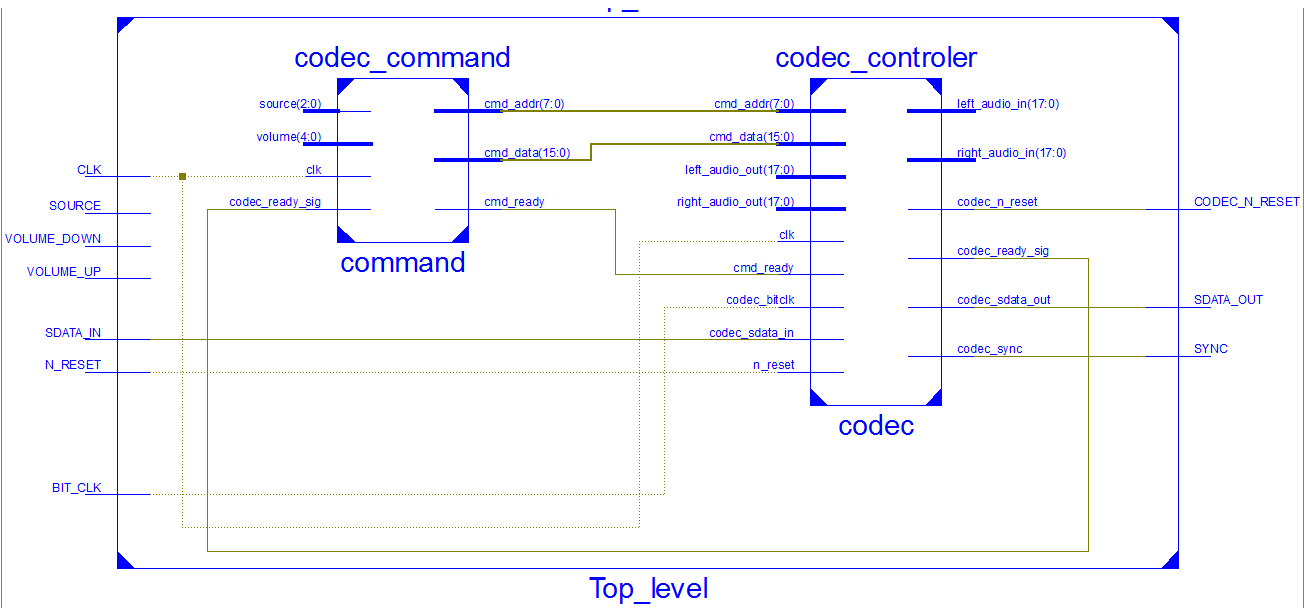


Figure ‑: Vue Rtl du contrôleur lm4550

#### Description du fonctionnement du contrôleur :

Le contrôleur est le composant responsable de la communication entre l’FPGA et le codec

audio lm4550 c’est taches principales sont :

* Constituer et envoyer les trames de données vers le codec :

1. Mettre le bit 15 (premier bit envoyé dans le TAG) dans le TAG (slot 0) à '1 'pour indiquer que la tramer présente au moins un slot de données valides. SYNC doit être maintenu

élevé ('1 ') pour une durée de 16 cycles d'horloge qui composent le TAG.

2. Mettre le bit 12 du TAG à '1 'pour indiquer que le slot 3 contient des données PCM pour le canal gauche.

3. Mettre le bit 11 du TAG à '1 'pour indiquer que le slot 4 contient des données PCM pour le canal droit.

4. Insérez les données PCM du canal gauche dans le slot 3. Les 18 bits sont envoyé poids fort en premier, le bit 17 de données PCM est envoyé en tant que le premier bit du slot 3. Ce sont les bits 56:73 dans la trame. Les deux derniers bits du slot 3 sont mis à zéro, ce sont les bits 73:75.

5. Insérez les données PCM du canal de droite dans le slot 4. Cela se fait d'une manière similaire aux données PCM gauche. Ce sont les bits 76:93 de la trame.

* Lire les trames depuis le codec

1. Lire le bit 15 du TAG (emplacement 0). Un '1 'indique que l'interface de liaison AC est prête.

2. Lire le bit 12 du TAG. Un '1 'indique que le slot 3 présente des données PCM valides à partir du canal de gauche.

3. Lire le bit 11 du TAG. Un '1 'pour indique que l'emplacement 4 a des données PCM valides à partir du canal droit.

4. Lire les données PCM du canal gauche depuis le slot 3 (bits 56:73).

5. Lire les données PCM du canal droit depuis le slot 4 (bits 76:93)

* Configurer les registres internes du codec

Le contrôleur prend les données (adresse du registre et commande) depuis le gestionnaire de commande et les envoient dans la même trame de sortie qui contient les données PCM.

1. Mettre le bit 14 du TAG à '1' pour indiquer que le slot 1 contient l'adresse de commande (adresse de registre).

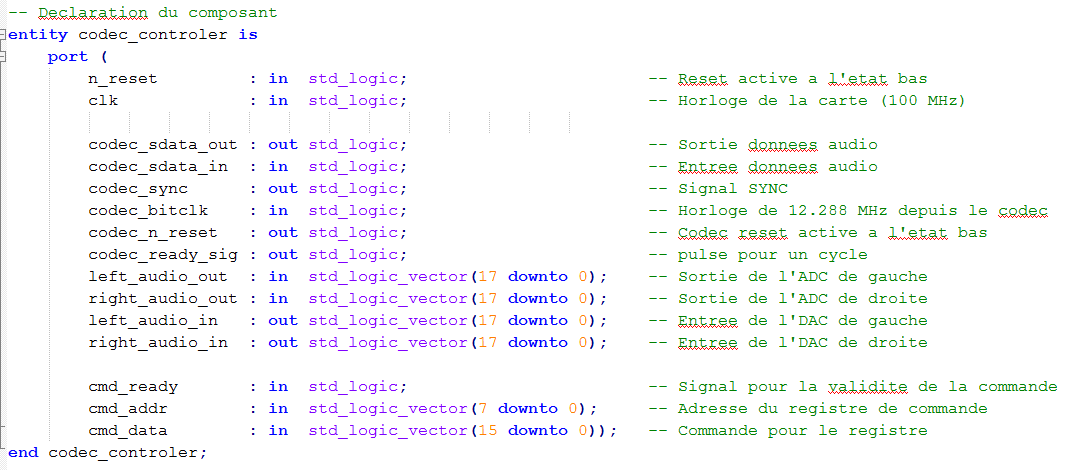
2. Réglez le bit 13 du TAG à '1 'pour indiquer que le slot 2 contient les données de contrôle.

3. Insérez l'adresse de contrôle dans le slot 1. Tout d'abord, mettre le bit 19 à '0 'pour indiquer une écriture. L'adresse de registre 7-bit est alors envoyé dans les bits 18 :12 du slot 1.

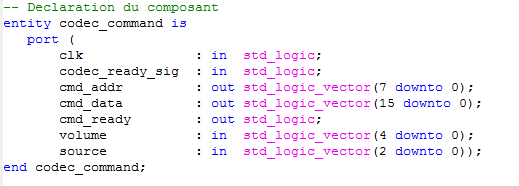
4. Insérer les données du registre dans le slot 2. Les données des registres de 16 bits dans les bits 19 :04 du slot 2.

#### Description du fonctionnement du gestionnaire de commande

Le gestionnaire de commande est une simple machine d’état qui prend les valeurs des entrées VOLUME et SOURCE ainsi que des valeurs prêt définies et l’envoi vers le contrôleur et met le signal CMD\_READY a ‘1’ pour que le contrôleur sache qu’une commande valide est prête.



Snapshot code Source ‑ : Déclaration du contrôleur



Snapshot code Source ‑ : Déclaration du gestionnaire de commande

# Traitement de signal : Filtrage

## Filtre à réponse impulsionnelle finie

En traitement numérique du signal, le filtre à réponse impulsionnelle finie ou filtre RIF (en anglais Finite Impulse Response filter ou FIR filter) est un filtre numérique qui est caractérisé par une réponse uniquement basée sur un nombre fini de valeurs du signal d'entrée. Par conséquent, quel que soit le filtre, sa réponse impulsionnelle sera stable et de durée finie dépendante du nombre de coefficients du filtre. Il peut aussi nommer filtre non récursif ou filtre à moyenne mobile, mais ces appellation sont restrictives ou abusives, parce que quand on parle de moyenne mobile pondérée on n'a en principe pas des pondérations négatives, et donc, au sens strict, les FIR de type moyenne mobile sont plutôt des passe-bas, et parce que les FIR peuvent être implémentés de façon récursive.

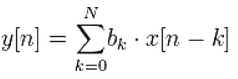
### Description d’un FIR

De façon générale le filtre à réponse impulsionnelle finie est décrit par la combinaison linéaire suivante où x[i] tel que 1 ≤ i ≤ n, représente les valeurs du signal d'entrée et y[i] tel que 1 ≤ i ≤ n les valeurs du signal de sortie.



Équation ‑ : Equation d’un FIR

En utilisant le symbole de sommation, l'équation peut être réécrite de la façon suivante :



Équation ‑ : Equation simplifié d'un FIR

Puisque la réponse est une somme d'un nombre fini de valeurs, le filtre RIF est naturellement stable d'après le critère Entrée Bornée/Sortie Bornée.

### Propriétés d’un FIR

- Les filtres RIF sont forcément stables, peu importe les coefficients utilisés.

- La complexité d'un filtre RII est moindre que celle d'un filtre RIF du même ordre. Cette propriété peut être utile sur les plateformes limitées en puissance de calcul

- Généralement, les filtres RIF sont moins sensibles aux erreurs de quantification que les filtres RII. L'absence de récursivité empêche les erreurs cumulatives.

- Un filtre RIF est moins sélectif qu'un filtre RII du même ordre. C'est-à-dire que la transition entre la bande passante et la bande rejetée est moins rapide que dans le cas du filtre RII.

- Contrairement à un RII, un filtre RIF peut avoir une réponse impulsionnelle symétrique et introduire un retard sur le signal mais aucun déphasage.

### Réalisation d’un FIR

Les filtres numériques peuvent être réalisés à l'aide de trois éléments ou opérations de base. Soit l'élément gain, l'élément de sommation et le retard unitaire. Ces éléments sont suffisants pour réaliser tous les filtres numériques linéaires possibles. La réalisation présentée dans la figure suivante est une réalisation directe du filtre RIF.

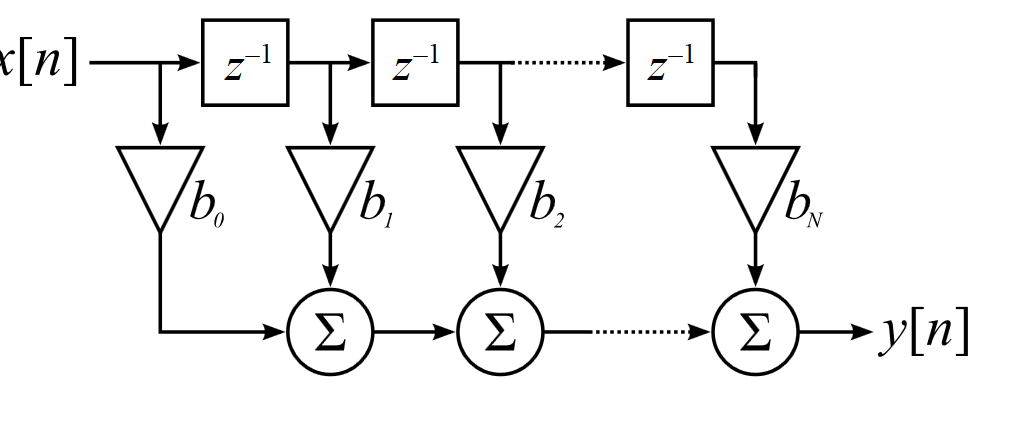


Figure ‑ : Schémas block d'un FIR

## Filtre à réponse impulsionnelle infinie

Un filtre à réponse impulsionnelle infinie ou filtre RII (en anglais infinite impulse response filter ou IIR filter) est un type de filtre électronique caractérisé par une réponse basée sur les valeurs du signal d'entrée ainsi que les valeurs antérieures.

Il est nommé ainsi parce que dans la majorité des cas la réponse impulsionnelle de ce type de filtre est de durée théoriquement infinie. Il est aussi désigné par l'appellation de filtre récursif. Contrairement au filtre RII la réponse du filtre RIF ne dépend que des valeurs du signal d'entrée. Par conséquent, la réponse impulsionnelle d'un filtre RIF est toujours de durée finie.

La plupart des filtres analogiques peuvent également être considérés comme des filtres à réponse impulsionnelle infinie.

### Description d’un IIR

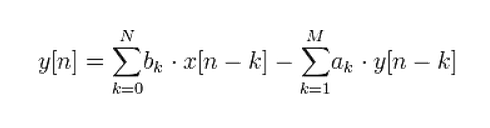
De façon générale le filtre à réponse impulsionnelle infinie est décrit par l'équation aux différences suivante où x représente les valeurs du signal d'entrée et y les valeurs du signal de sortie.



Équation ‑ : Equation d’un IIR

En utilisant le symbole de sommation, l'équation peut être réécrite de la façon suivante :

Équation ‑: Equation simplifié d'un IIR



### Propriétés d’un IIR

- Les filtres RII ne sont pas forcément stables, la stabilité dépend de la position des pôles dans le plan complexe.

- La complexité d'un filtre RIF est supérieure à celle d'un filtre RII du même ordre. Cette propriété peut être un problème sur les plateformes limitées en puissance de calcul.

- Généralement, les filtres RII sont plus sensibles aux erreurs de quantification que les filtres RIF. La récursivité peut générer des erreurs cumulatives.

- Un filtre RII est plus sélectif qu'un filtre RIF du même ordre. C'est-à-dire que la transition entre la bande passante et la bande rejetée est plus rapide que dans le cas du filtre RIF.

### Réalisation d’un IIR

Les filtres numériques peuvent être réalisés à l'aide de trois éléments ou opérations de base. Soit l'élément gain, l'élément de sommation et le retard unitaire. Ces éléments sont suffisants pour réaliser tous les filtres numériques linéaires possibles. La réalisation présentée dans la figure suivante est une réalisation directe d’un filtre RII.

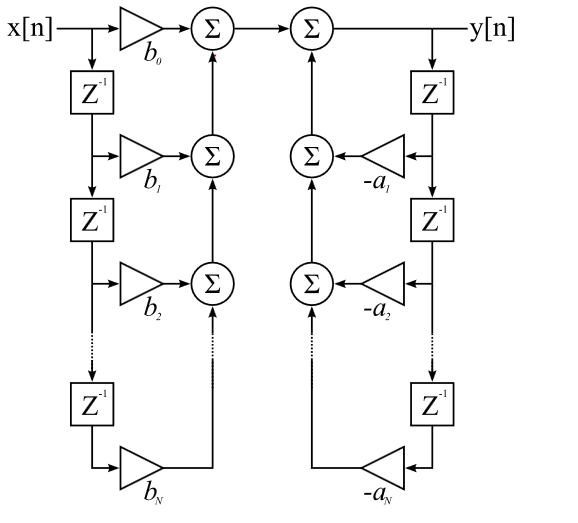


Figure ‑ schémas bloque d’un IIR

## Implémentation d’un filtre FIR en VHDL

La section suivante va aborder la conception et l’implémentation d’un filtre FIR passe bas. La conception sera faite par l’outil de conception de filtre de matlab (fda tool) pour pouvoir calculer les coefficients du filtre pour après les utilisés dans le code VHDL.

Cette présentation utilise la version 2011a de matlab, Il faut notez que les versions ultérieure (2013) le fonctionnement est un peu différent.

### Création du filtre sur matlab

Tous d’abord Il faut ouvrir le toolbox filtre design (fda tool) qui se trouve dans le menu des toolbox.

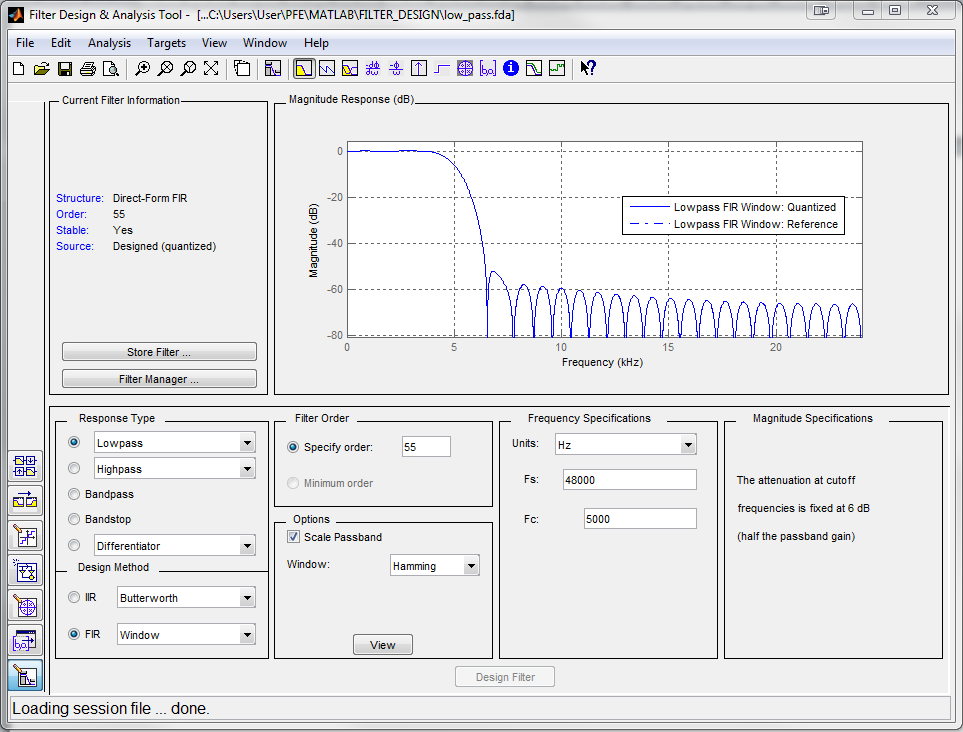


Figure ‑ : fda tool

Et puis suivre les étapes suivantes :

1. Choisir le type de filtre

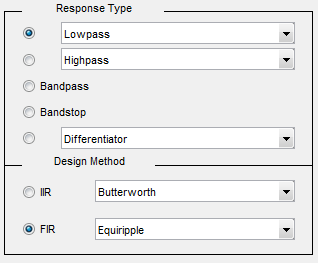


Figure ‑ : choix de filtre

Choisir ‘’Lowpass ‘’ dans ‘’Responce Type’’ et dans ‘’FIR’’ dans ‘’Design Method‘’, puisque le filtre voulu est un filtre FIR passe bas.

1. Choisir le type de fenêtre

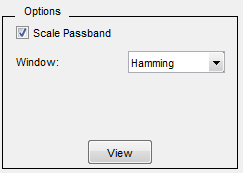


Figure ‑ : Choix de la fenêtre

1. Choisir la fréquence de coupure  et la fréquence d’échantillonnage

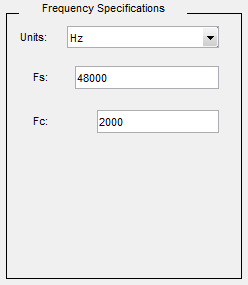


Figure ‑ : Choix de la fréquence de coupure et la fréquence d’échantillonnage

En ce qui concerne la fréquence d’échantillonnage, puisque le codec audio échantillonne a 48khz donc le filtre doit aussi utiliser la même fréquence.

1. Choisir l’ordre du filtre :

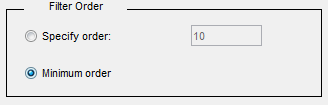


Figure ‑ : choix de l'ordre du filtre

Pour le choix du filtre deux options sont possible, laisser l’outil calculer l’ordre optimal pour le filtre en choisissant l’option ‘’minimum order’’ ou bien de de mettre l’ordre du filtre manuellement en cochant la case ‘’Specify order’’ et puis de mettre l’ordre voulu.

Pour une utilisation sur un pc l’option peut être choisie pour avoir réponse impulsionnels presque parfaite. Cependant cette option ne pourra pas marcher sur un FPGA tel que le SPATAN 6

et cela à cause du nombre limiter des multiplexeurs et qui sont un des composant essentiel pour la conception d’un filtre comme le montre les figures 4-1 et 4-2.

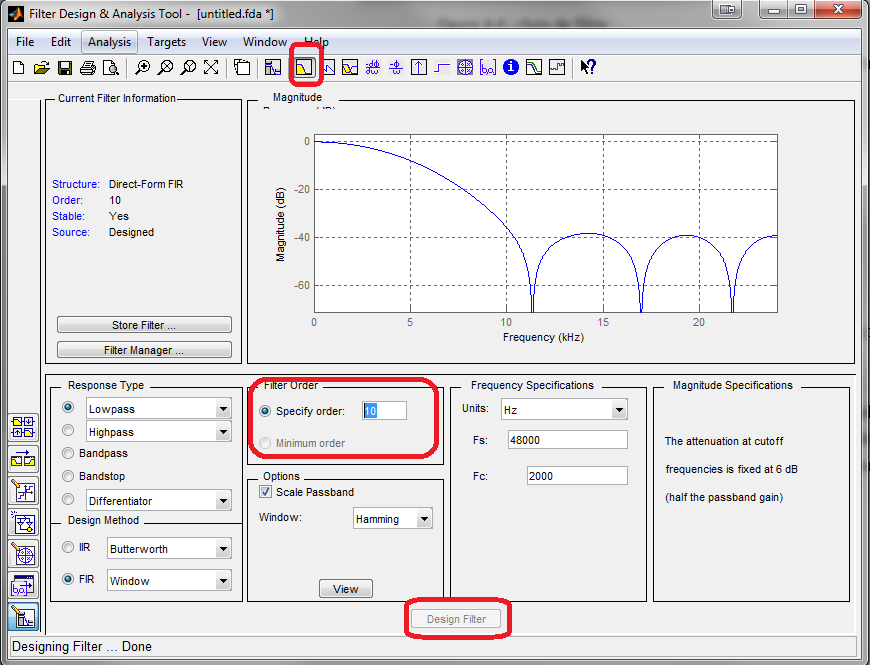


Figure ‑ : Reponce impultionel

En choisissant la valeur 10 comme ordre du filtre ce dernier peut être créé en cliquant sur le bouton ‘’Design Filter’’. On peut voir que la réponse impulsionnels su la fenêtre principale.

Pour avoir une réponse impulsionnelle meilleure on peut augmenter l’ordre du filtre.

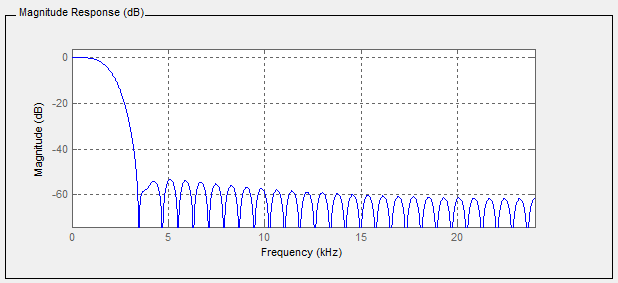


Figure ‑ : filtre ordre 60

On peut voir sur la figure 4-7 que en augmentant l’ordre du filtre on peut avoir une meilleur réponse impulsionnel.

Enfin on peut avoir tous les coefficients de notre filtre

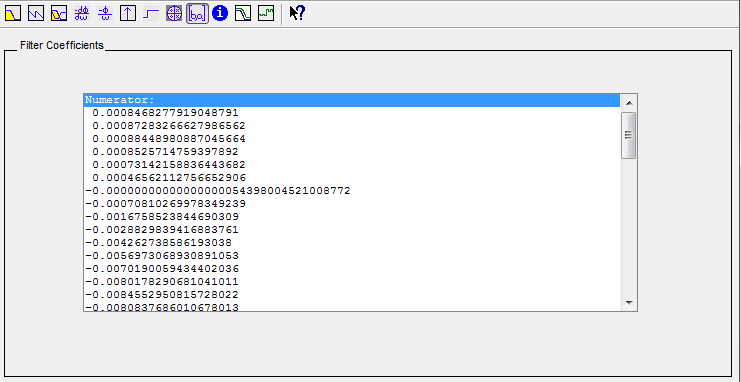
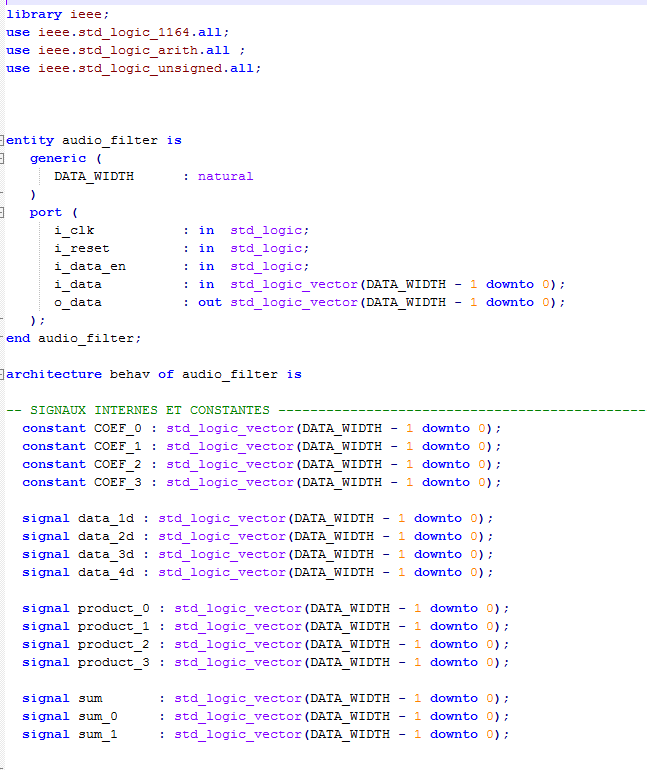
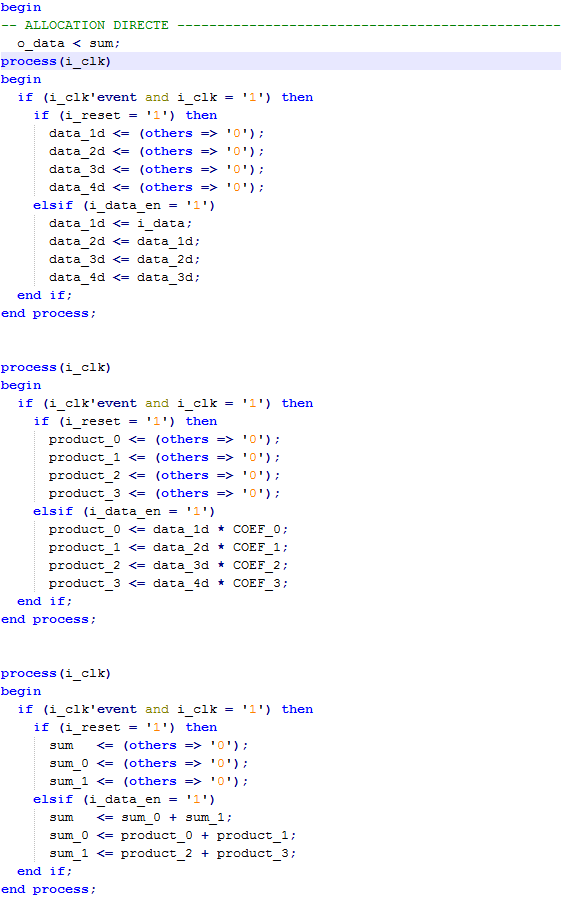


Figure ‑ : coefficients du filtre

Maintenant que les coefficients sont calculés, le filtre peut être créé en vhdl.



Snapshot code Source ‑ : Code source d'un filtre fir



Snapshot code Source ‑: Code source d'un filtre fir

Cet exemple présent une implémentation d’un filtre passe bas d’ordre 4, qui utilise

- 4 coefficients COEF\_0, COEF\_1, COEF\_2, COEF\_3

- 4 multiplicateur product\_0, product\_1, product\_2, product\_3

- 3 accumulateurs sum, sum\_0, sum\_1

- 4 Buffers temporaire data\_1d, data\_2d, data\_3d, data\_3d

Le calcule performé est exactement l’équation d’un filtre FIR comme le montre la figure 4-2.

L’implémentation d’un tel filtre est relativement facile, cependant il existe des méthodes bien plus performantes pour la création des filtres en vhdl. La première option est d’utiliser matlab pour générer directement du vhdl ou bien d’utiliser un IP CORE de Xilinx. Les deux choix sont expliqués dans les sections qui suivent.

### Génération du code vhdl depuis matlab

Pour pouvoir générer du code vhdl directement depuis matlab, il faut tout d’abord créer un

Filtre en utilisant l’outil de design de filtre comme la montré la section 4-3-1, et suivre les étapes suivante :

1. Suivre les mêmes étapes de la section 4-3-1, pour créer le filtre
2. Cliquez sur le bouton Paramètres de quantification  dans la barre d'outils de gauche. Le FDATool affiche un menu ‘’filtre arithmetic’’.

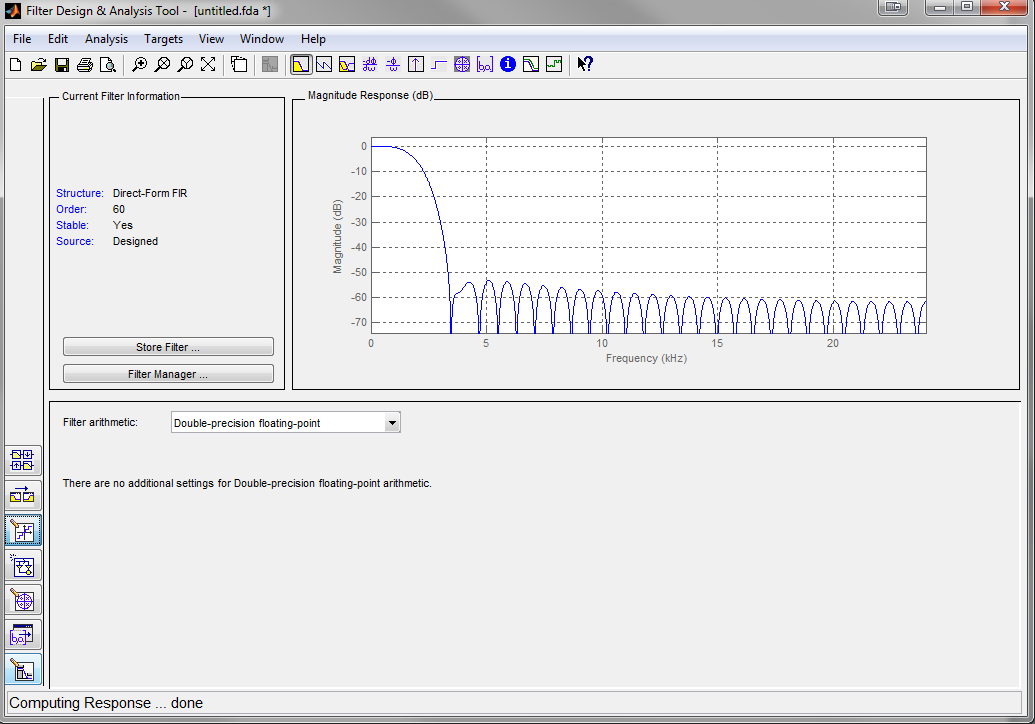


Figure ‑ : Quantification du filtre

1. Sélectionnez ‘’Fixed point’’ dans la liste. Ensuite, sélectionnez ’’Specify all’’ dans la liste de ‘’Filter precision’’. Le FDATool affiche le premier des trois onglets de paramètres de quantification dans la moitié inférieure de la boîte de dialogue.

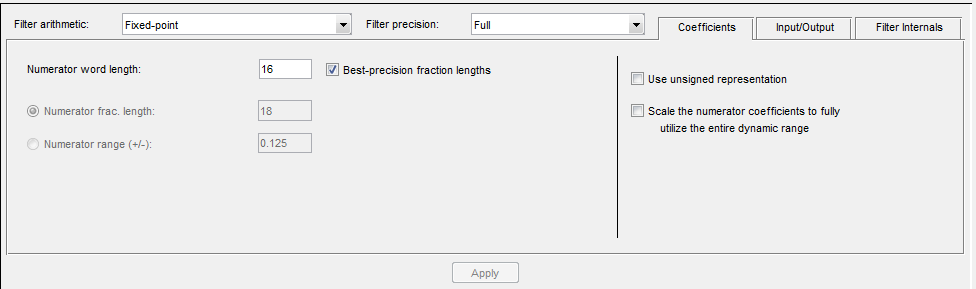


Figure ‑ : option-coefficients de quantification

1. Maintenant il faut régler les paramètres de quantification :

|  |  |  |
| --- | --- | --- |
| **Tab** | **Paramètre** | **valeur** |
| Coefficients | Numerator word length | 18 |
|  | Best-precision fraction lengths | Sélectionné |
|  | Use unsigned representation | Désélectionné |
|  | Scale the numerator coefficients to fully utilize the entire dynamic range | Désélectionné |
| Input/Output | Input word length | 18 |
|  | Input fraction length | 17 |
|  | Output word length | 18 |
| Input/Output | Rounding mode | Floor |
|  | Overflow mode | Saturate |
|  | Accum. word length | 40 |

Tableau ‑ : option de quantification

Le choix de largeur du vecteur d’entrée de 18 bit est base sur le fait que les données reçus par le codec sont de 18 bit, ainsi que la largeur du vecteur de sortie est de 18 bit car le codec doit recevoir une donnée de 18 bit. Les autres valeurs sont des valeurs suggérer par le site officiel de matlab.

1. Cliquez sur ‘’apply’’

Maintenant que le filtre est quantifié, la generation du code vhdl est possible. Pour cela

1. Pour ouvrir l’outil de génération  Targets > Generate HDL.

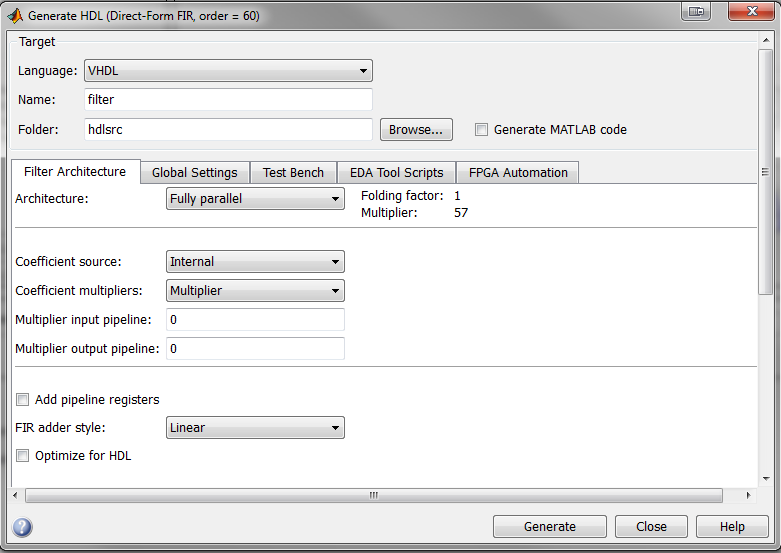


Figure ‑ : Générateur hdl

Apres il faut regler les parametres suivant :

1. Le nom du filtre

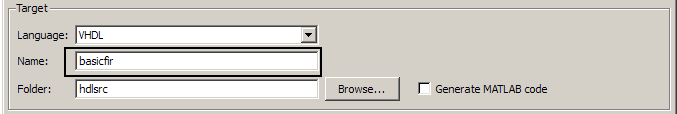


Figure ‑ : Générateur hdl - nom

1. Réglage des paramètres globaux de la génération

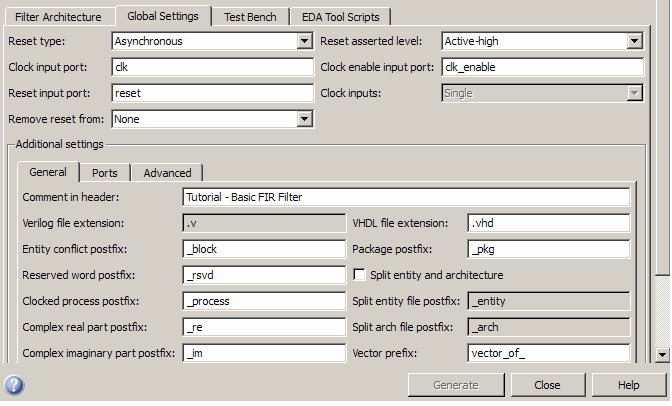


Figure ‑ : Générateur hdl - paramètres globaux

Pour les paramètres globaux, les valeurs par défaut sont correctes. On peut changer les noms des signaux d’entrée/sortie du composant qui seras générer par exemple ‘’Clock input port’’, ‘’reset input port’’… On peut aussi choisir quel type de reset on veut utiliser synchrone/asynchrone dans l’option ‘’reset type’’ et aussi sur quel front il sera active montant/descendant.

1. Enfin on clique sur ‘’generate’’.

### Génération du code depuis Xilinx IP CORE GENERATOR

L’utilisation du generateur de xilinx est similaire à celle de matlab, car les deux outils utilisent le même créateur de filtre (fdatool).

Afin de generer un filtre avec l’outil de xilinx il faut suivre les étapes suivantes :

1. Ajouter une nouvelle source a votre projet

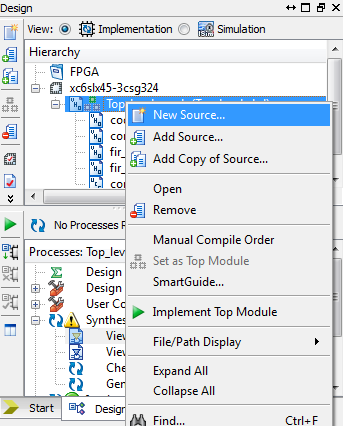


Figure ‑: Ajouter une source au projet

Puis il faut choisir IP (Core generator) dans la liste et donner au filtre un nom dans ce cas c’est ‘Filter’.

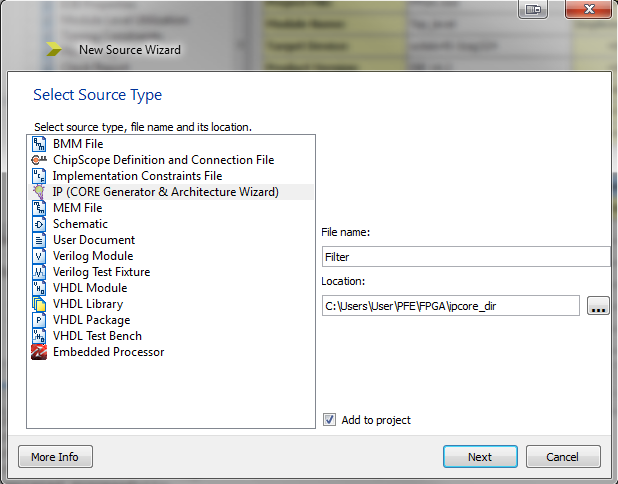


Figure ‑ : Ajouter un ip core

1. Choisir le générateur du filtre dans la section ‘’Digital Signal Processing/Filters/FIR Compiler’’ et cliquer sur ‘’next’’ puis sur ‘’Finish ‘’. Dans la section ‘Filters’ pourrait avoir un ou plusieurs ‘compiler’ avec différente versions, il faut toujours choisir la dernière version pour avoir toutes les mise a jours possible en ce qui concerne l’optimisation du filtre.

Une nouvelle fenêtre va apparaitre dans laquelle on peut régler les différents paramètres du filtre.



Figure ‑ : Fir compiler 6.3

1. Ajouter les coefficients du filtre

Tout d’abord il faut générer les coefficients, pour cela on va utiliser matlab pour générer un fichier .COE qui contient toutes les valeurs des coefficients du filtre.

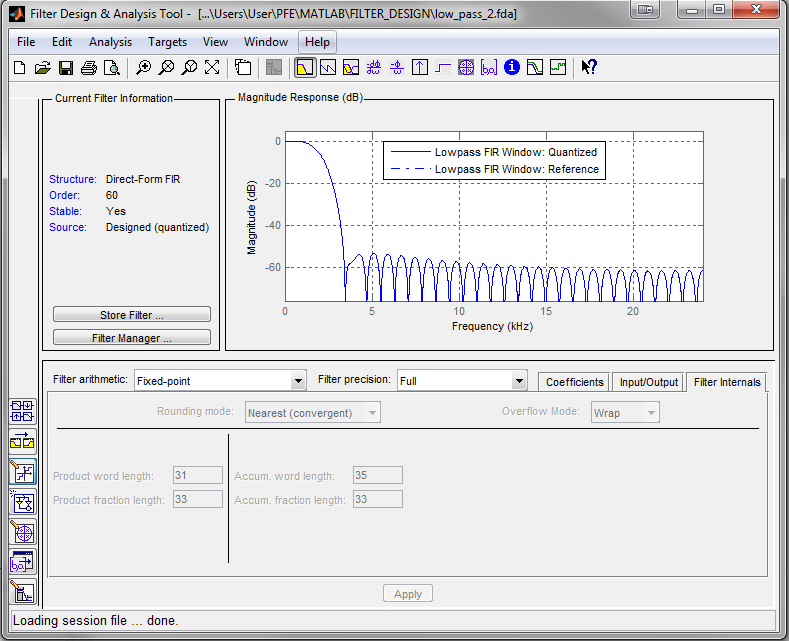


Figure ‑ : FDA tool

Pour générer le fichier .COE on utilisant les même valeurs du filtre utiliser dans la section précédente, allez dans ‘’ Targets -> XILINX Coefficient’’ puis enregistrer le fichier.

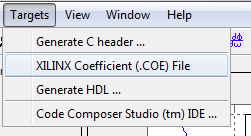


Figure ‑ : Xilinx coeficient

Dès que le fichier .COE est genere, il faut l’utiser comme source de coeficient dans ‘’Coefficient File ‘’

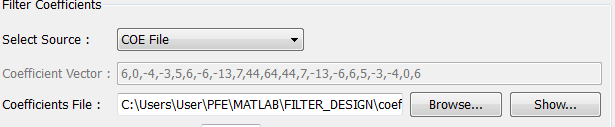


Figure ‑ : Ajouter le fichier .coe

1. Régler la largeur des vecteurs d’entrée et de sortie

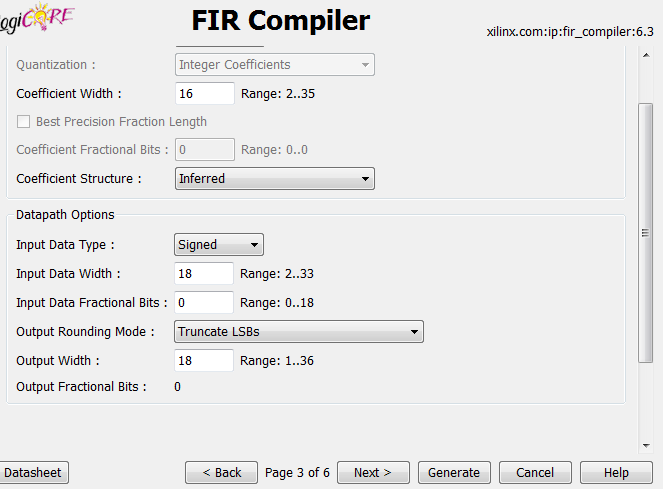


Figure ‑ : Réglage des entrée/sortie

Pour les autres paramètres il faut laisser les valeurs par défaut. Enfin cliquer sur ‘’generate’’.

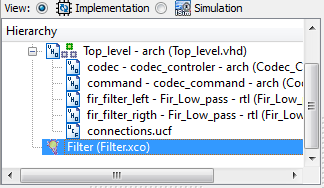


Figure ‑ : Generation final du filtre depuis ip core

Les deux sections précédentes qui résument comment générer un filtre FIR depuis matlab et aussi depuis IP Core Generator de Xilinx. Dès que le filtre est généré il pourra être intégrer au programme principal comme n’importe quel autre composant en vhdl.

# Implémentation d’un filtre RII bi quad sur la carte de développement

## Introduction

Les filtres à réponse impulsionnelle infinie peuvent être utilisés dans de nombreuses applications, dont l'audio, avec une faible surcharge d’utilisation de ressources. Une section bi-quad peut consommer aussi peu que 5 multiplicateurs de 18-bitdans un FPGA, jusqu'à 20 si la longueur des mots est de 32 bits. Les sujets de conception suivants seront abordés:

• Principes fondamentaux de conception d’un filtre IIR bi-quad

• la stabilité du filtre et la fonction de transfert en Z

• Mise à l'échelle du filtre pour une mise en œuvre plus efficace

• Caractéristiques du filtre et validation

• La conception de composants personnalisés en VHDL et instanciation avec un contrôleur audio de 18-bit.

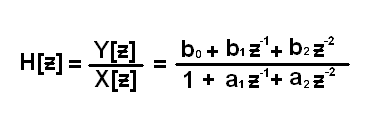
## Contexte

Cette section couvre les éléments fondamentaux de la conception du filtre IIR pour inclure la transformation en z bi-linéaire et la conception du module en VHDL. Le filtre IIR est un filtre récursif conçu à partir d'un prototype de fonction de transfert analogique. La transformation en z bilinéaire est une approximation linéaire qui mappe la fonction de transfert au domaine discrèt z.

En traitement du signal, un filtre biquad numérique est un filtre linéaire récursif de deuxième ordre, possédant au moins deux pôles et deux zéros. "Biquad’’ est une abréviation de "biquadratic", qui se réfère au fait que dans le domaine Z, sa fonction de transfert est le rapport de deux fonctions du second degré (Équation 5-2)



Équation ‑ Fonction de transfère en s d'un filtre biquad



Équation ‑ : Fonction de transfère en z d'un filtre biquad

Les filtres récursifs d'ordre élevé peuvent être très sensibles à la quantification de leurs coefficients, et peuvent facilement devenir instable. C'est beaucoup moins un problème de premier et de second ordre, par conséquent, les filtres d'ordre supérieur sont généralement mises en œuvre en des sections biquad en série en cascade. Les deux pôles du filtre biquadratique doivent être à l'intérieur du cercle unité pour qu'il soit stable.

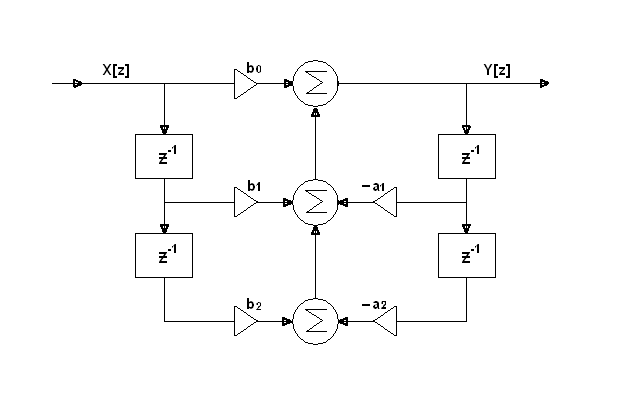


Figure ‑: Schemas bloque d'un filtre RII biquad

La récursivité peut entraîner une instabilité du filtre. Dans le domaine s (ou p laplace) est considéré comme stable si les pôles se trouvent dans le demi-plan de gauche, ces pôles se mappe à une zone à l'intérieur du cercle unité dans le domaine z. La figure 5-2 ci-dessous illustre la mise en correspondance de l'axe J-oméga sur le cercle unité.

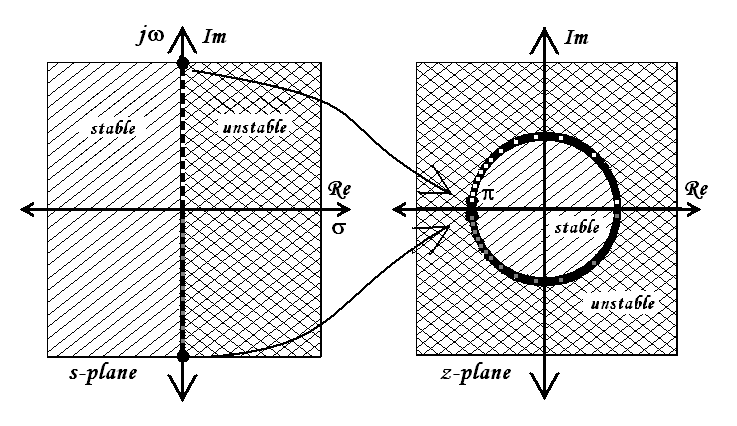
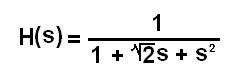


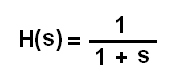
Figure ‑ : Cercle unitaire

Les fréquences sont déformés en raison de la mise en correspondance à partir du domaine s à z. Le plan z est lié au plan s essentiellement par la fonction représentée par l'équation 5-3 ci-dessus. Les extrémités de l'axe J-oméga rencontrent aux radians circulaires dans le plan z, cela se produit également à la moitié de la fréquence d'échantillonnage. On ne peut dire qui le filtre est stable en résolvant les équations du second degré du numérateur et du dénominateur (trouver les pôles et les zeros) et en les reportant dans le plan z. S’ils se trouvent à l'intérieur du cercle unité. Cela signifie que l'on peut dire que le filtre est stable.



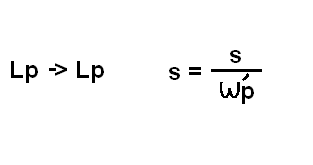
Équation ‑ : Système 2eme ordre

La première étape pour dériver les coefficients biquadratique appropriés en utilisant la transformation en z bilinéaire est de commencer avec un prototype analogique. L’équation 5-3 représente la fonction de transfère d’un filtre Butterworth d'ordre 2 utiliser pour un filtre passe-bas et passe haut, Epandant pour les filtre passe bande et réjecteur de bande deux filtre du premier ordre est en cascade représenté par la fonction de transfère dans l’équation 5-4.

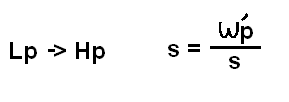


Équation ‑ : Système 1er ordre

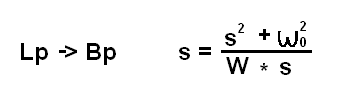
Ensuite la fonction de transfert du prototype analogique doit être dénormalisés avec l'une des transformations suivantes dans les équations 5-5, 5-6, 5-7, ou 5-8. Cela se fait par le remplacement de «s» dans le prototype analogique avec l'une des transformations ci-dessous. Le passe-bas au passe-bas et passe-bas à des transformations passe-haut ont été utilisés avec une fonction de transfère de second de l'ordre. Le passe-bas au passe-bande et passe-bas à éjecteur de bande ont avec une fonction de transfère du premier ordre.



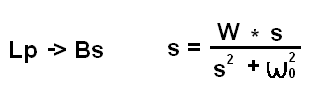
Équation ‑ : Passage passe bas vers passe bas



Équation ‑ : Passage passe bas vers passe haut



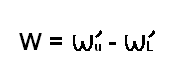
Équation ‑ : Passage passe bas vers passe bande



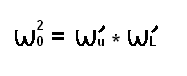
Équation ‑ : Passage passe bas vers éjecteur de bande

Le passe-bande et éjecteur de bande ont deux fréquences comme le montre les équations

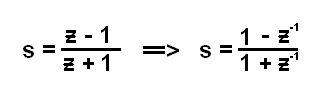
5-9 et 5-10 ci-dessous. Une fois que tous les remplacements sont effectués, la transformation en z finale peut être appliquée par la substitution dans de s par l’équation 5-11.



Équation ‑ : Fréquences de coupure du passe bande



Équation ‑ : Fréquences de coupure du éjecteur de bande



Équation ‑ : Equation de transformation en z

## L'instanciation du module de filtrage

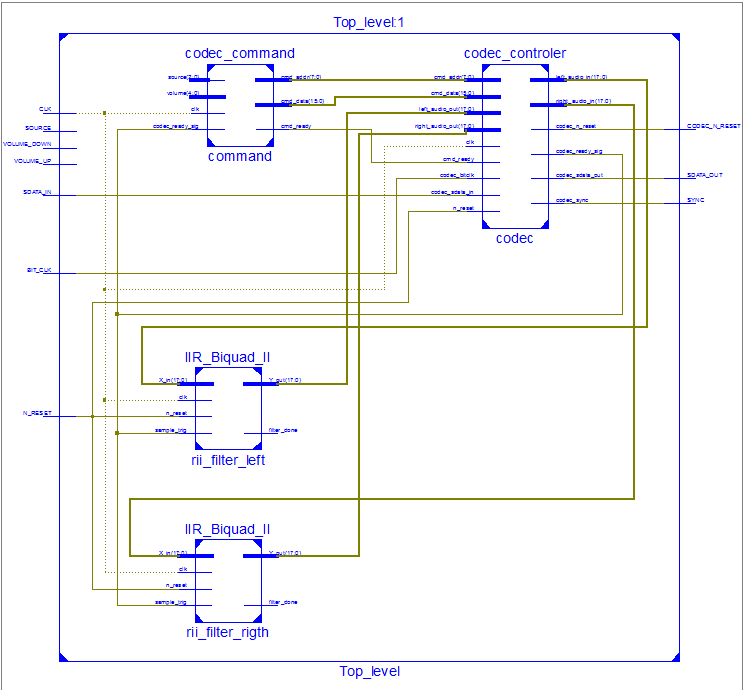


Figure ‑ : instanciation du filtre Rii

Cette application utilise une horloge externe de 100MHz capable de faire du traitement de signal à des vitesses plus rapides. Les coefficients peuvent être calculés de plusieurs façons. Par des programmes de calcul scientifique comme matlab et ou par calcule manuel à l'aide de la méthode de placement pôle-zéro (résolution d’un système de deuxième ordre). Une fois que les coefficients sont calculés, ils auront besoin d'être converti en binaire signé un format en virgule fixe "2.30".

Le format virgule fixe 2.30 peut produire une valeur entre -2 et 1. 999999999068677…

Exemple de conversion :

Coefficient = 0.061511769

1. Séparer la partie décimale de la partie décimale :

La valeur entière = 0

La valeur décimal = .061511769

1. Convertir et arrondir la partie décimale :

.061511769 x 2^30 = 66047759.043526656

Round (66047759.043526656) = 66047759

1. Convertir la valeur arrondie en binaire :

66047759 = 0011 1110 1111 1100 1111 0000 1111

1. Convertir la partie entière en binaire :

0 = 00

Donc

0.061511769 = 00. 0011 1110 1111 1100 1111 0000 1111

Coefficient = -1.99075016

1. Pour faciliter la conversion d’un nombre négative on calcule la différence entre le coefficient et -2 :

2- 1.99075016 = 0.0924984

1. Convertir et arrondir la partie décimale :

0.0924984 x 2^30 = 99319400.7330816

Round (99319400.7330816) = 99319401

1. Convertir en binaire

99319401 = 0101 1110 1011 0111 1110 0110 1001

Convertir -2 en binaire sur 2 bits

-2 = 10

Don

(-2 +0.0924984) = -1.99075016

= 10. 0101 1110 1011 0111 1110 0110 1001

## Teste et vérification

Le module a été testé à l'aide du logiciel de simulation de Xilinx, ISIM, qui est un simulateur piloté à deux Etats, les contrôles de machine d'état permettent de synchroniser les opérations mathématiques, en tronquant les opérations intermédiaires, et le verrouillage de la sortie de chaque période d'échantillonnage. La figure représente une capture d’écran du résultat de la simulation.

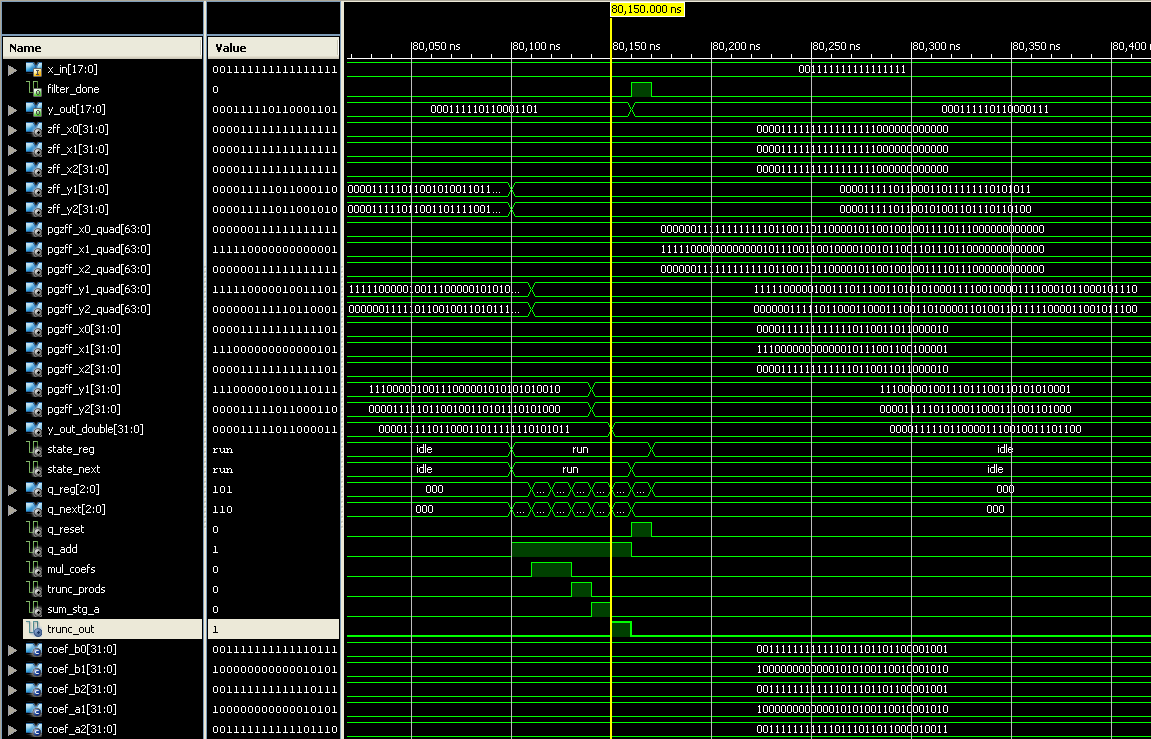


Figure ‑ : Simulation du filtre

Le module a été testé par l'utilisation du codec audio LM4550. Le filtre a testés par deux signaux de 18 bits de 0 à 24 kHz et une seule impulsion pour atteindre la réponse impulsionnelle. La réponse impulsionnelle a été soumise à une transformation de Fourier rapide pour accéder à la réponse fréquentielle. L’exemple suivant représente un filtre IIR Butterworth Notch de second ordre 60 Hz. Un filtre analogique, 60 Hz Twin T Notch, a été créé pour comparer avec la mise en œuvre numérique. Les sorties du spectre de fréquences du filtre analogique et du filtre numérique RII sont représenté dans les figure 5-5 et 5-6. Les deux filtres ont atténuation coupe-bande et dans le bruit de fond. Agrandir atténuation pourrait être réalisé si grand signal de test d'entrée devait être utilisé.

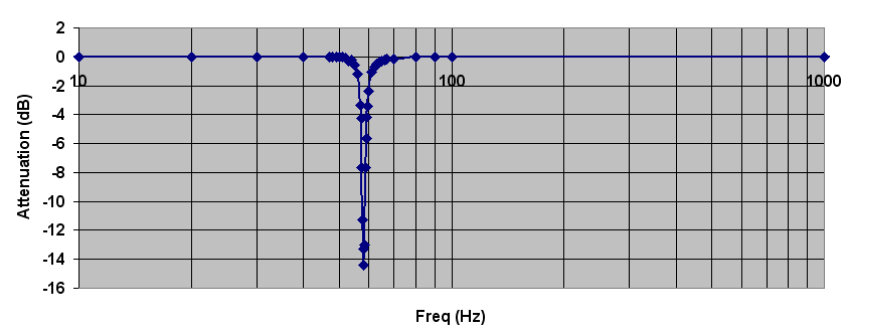


Figure ‑ : filtre analogique, 60 Hz Twin T Notch

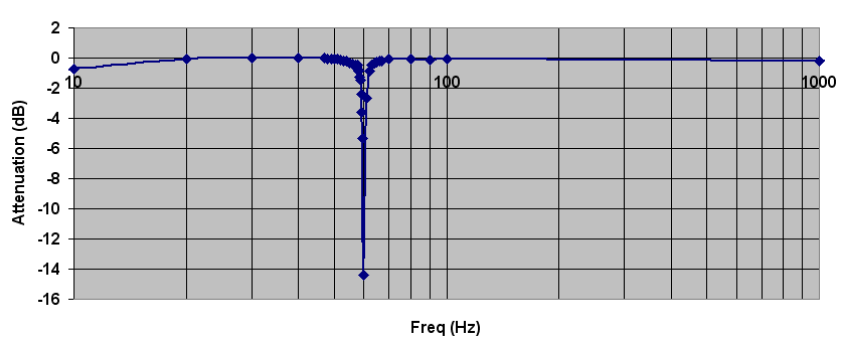
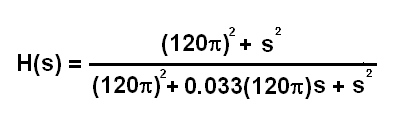
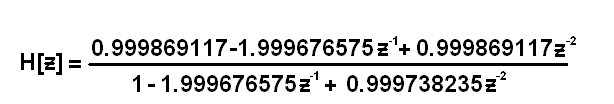


Figure ‑ : Filtre numérique IIR Butterworth Notch

Le filtre analogique souffre des imperfections de la fréquence dues à des composants de conversion nécessaires à la discrétisation, Cependant la fréquence centrale du filtre numérique IIR quasiment parfaite. Les fonctions de transfert pour chaque filtre sont représentées par les équations 5-12 et 5-13.

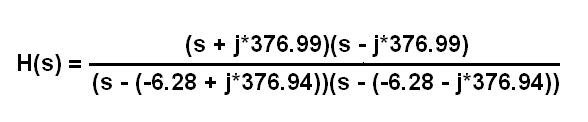


Équation ‑ : Fonction de transfère d'un filtre analogique

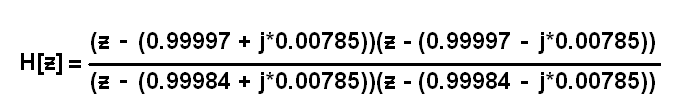


Équation ‑ : Fonction de transfère d'un filtre Rii numérique

Les zéros et les pôles peuvent être déterminés en résolvant le numérateur et le dénominateur des équations du second degré. Dans ce cas, les deux sont complexes et peuvent être vus comme le représentent les équations 5-14 et 5-15.



Équation ‑ : Fonction de transfère complexe d'un filtre analogique



Équation ‑ : Fonction de transfère complexe d'un filtre Rii analogique

Les pôles et les zéros du filtre éjecteur de bande analogique et numérique peuvent être vus dans la figure 5-7 ci-dessous. Remarquez comment le filtre est sur le bord de la stabilité dans les deux implémentations.

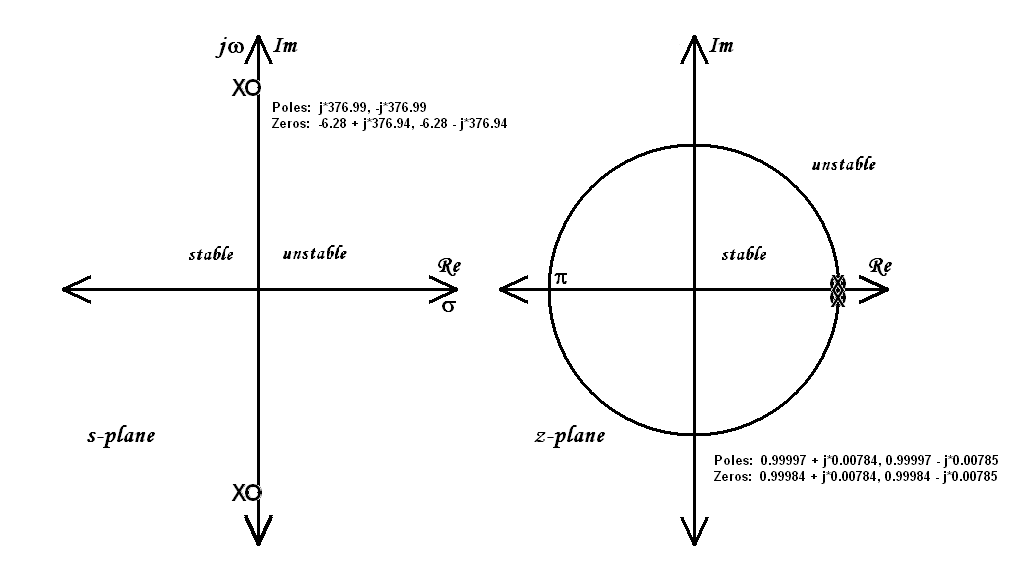
****

Figure ‑ : Représentation des pôles et des zéros

La figure 5-8 représente une capture d’écran d’un oscilloscope avec un module FFT. La réponse impulsionnelle représentée en bleu est soumis à la FFT qui donne la réponse en fréquence du filtre recrée en rouge.



Figure ‑ : Réponse fréquentiel d'un filtre éjecteur de bande

# Les outils de développement

## La gestion de versions

### À propos de la gestion de version

Qu'est-ce que la gestion de version et pourquoi devriez-vous vous en soucier ? Un gestionnaire de version est un système qui enregistre l'évolution d'un fichier ou d'un ensemble de fichiers au cours du temps de manière à ce qu'on puisse rappeler une version antérieure d'un fichier à tout moment. Bien que la gestion de version est utilisé généralement pour les codes sources, mais en réalité on puisse l'utiliser avec pratiquement tous les types de fichiers d'un ordinateur*.*

Si vous êtes un dessinateur, développeur web ou développeur logiciel, et que vous voulez conserver toutes les versions d'une image ou d'un code source, un système de gestion de version (VCS en anglais pour Version Control System) est un outil qu'il est très sage d'utiliser. Il vous permet de ramener un fichier à un état précédent, de ramener le projet complet à un état précédent, de visualiser les changements au cours du temps, de voir qui a modifié quelque chose qui pourrait causer un problème, qui a introduit un problème et quand, et plus encore. Utiliser un VCS signifie aussi généralement que si vous vous trompez ou que vous perdez des fichiers, vous pouvez facilement revenir à un état stable. De plus, vous obtenez tous ces avantages avec peu de travail additionnel.

### Les systèmes de gestion de version locaux

La méthode courante pour la gestion de version est généralement de recopier les fichiers dans un autre répertoire qui peut-être avec un nom incluant la date dans le meilleur des cas. Cette méthode est la plus courante parce que c'est la plus simple, mais c'est aussi la moins fiable. Il est facile d'oublier le répertoire dans lequel vous êtes et d'écrire accidentellement dans le mauvais fichier ou d'écraser des fichiers que vous vouliez conserver. Pour traiter ce problème, les programmeurs ont développé il y a longtemps des VCS locaux qui utilisaient une base de données simple pour conserver les modifications d'un fichier (voir figure 6-1)

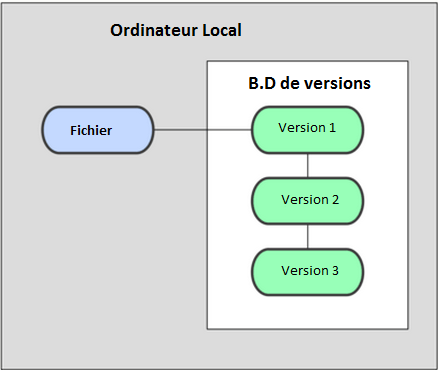


Figure ‑ : Modélisation d’un système de gestion de version local

Un des systèmes les plus populaires était RCS, qui est encore distribué avec de nombreux systèmes d'exploitation aujourd'hui. Même le système d'exploitation populaire Mac OS X inclut le programme RCS lorsqu'on installe les outils de développement logiciel. Cet outil fonctionne en conservant des ensembles de patchs (c'est-à-dire la différence entre les fichiers) d'une version à l'autre dans un format spécial sur disque ; il peut alors restituer l'état de n'importe quel fichier à n'importe quel instant en ajoutant toutes les différences.

### Les systèmes de gestion de version centralisés

Le problème majeur que les gens rencontrent est qu'ils ont besoin de collaborer avec des développeurs sur d'autres ordinateurs. Pour traiter ce problème, les systèmes de gestion de version centralisés (CVCS en anglais pour Centralized Version Control Systems) furent développés. Ces systèmes tels que CVS, Subversion, et Perforce, mettent en place un serveur central qui contient tous les fichiers sous gestion de version, et des clients qui peuvent extraire les fichiers de ce dépôt central. Pendant de nombreuses années, cela a été le standard pour la gestion de version (voir figure 6-2).

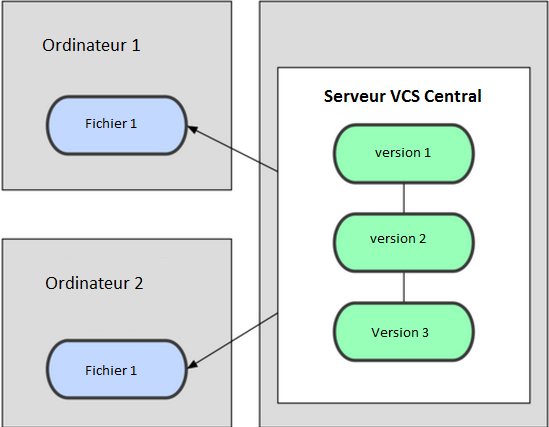


Figure ‑ : Modélisation d’un système de gestion de version centraliser

Ce schéma offre de nombreux avantages par rapport à la gestion de version locale. Par exemple, chacun sait jusqu'à un certain point ce que tous les autres sont en train de faire sur le projet. Les administrateurs ont un contrôle fin des permissions et il est beaucoup plus facile d'administrer un CVCS que de gérer des bases de données locales.

Cependant ce système a aussi de nombreux défauts. Le plus visible est le point unique de panne que le serveur centralisé représente. Si ce serveur est en panne pendant une heure, alors durant cette heure, aucun client ne peut collaborer ou enregistrer les modifications issues de son travail. Si le disque dur du serveur central se corrompt, et s'il n'y a pas eu de sauvegarde, vous perdez absolument tout de l'historique d'un projet en dehors des sauvegardes locales que les gens auraient pu réaliser sur leur machines locales. Les systèmes de gestion de version locaux souffrent du même problème — dès qu'on a tout l'historique d'un projet sauvegardé à un endroit unique, on prend le risque de tout perdre.

### Les systèmes de gestion de version distribués

C'est à ce moment que les systèmes de gestion de version distribués entrent en jeu (DVCS en anglais pour Distributed Version Control Systems). Dans un DVCS (tel que Git, Mercurial, Bazaar ou Darcs), les clients n'extraient plus seulement la dernière version d'un fichier, mais ils dupliquent complètement le dépôt. Ainsi, si le serveur disparaît et si les systèmes collaboraient via ce serveur, n'importe quel dépôt d'un des clients peut être copié sur le serveur pour le restaurer. Chaque extraction devient une sauvegarde complète de toutes les données (voir figure 6-3).

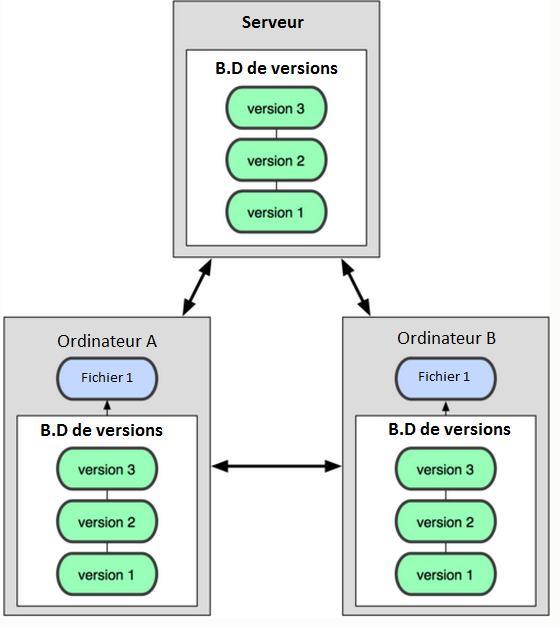


Figure ‑ : Modélisation d’un système de gestion de version distribuée

De plus, un grand nombre de ces systèmes gère particulièrement bien le fait d'avoir plusieurs dépôts avec lesquels travailler, vous permettant de collaborer avec différents groupes de personnes de manières différentes simultanément dans le même projet. Cela permet la mise en place de différentes chaînes de traitement qui ne sont pas réalisables avec les systèmes centralisés, tels que les modèles hiérarchiques.

### Quelque logiciels de gestion de version :

|  |  |  |
| --- | --- | --- |
|  | **Logiciels libres** | **Logiciels propriétaires** |
| **Gestion Locale** | GNU RCS (1892) | PVCS (1985), SCCS(1972) |
| **Client-Serveur** | CVS(1990), CVSNT(1992), SVN(2000) | CMVC(1994), sourceanywhere(2003) |
| **Décentralisé** | GNU arch(2001), Fossil(2007), Git(2005) | BitKeeper(1998), plastic SCM(2007) |

Tableau ‑ : Logiciels de gestion de version

## Git

### Historique

Git est né avec une dose de destruction créative et de controverse houleuse. Le noyau Linux est un projet libre de grande envergure. Pour la plus grande partie de sa vie (1991–2002), les modifications étaient transmises sous forme de patchs et d'archives de fichiers. En 2002, le projet du noyau Linux commença à utiliser un DVCS propriétaire appelé BitKeeper.

En 2005, les relations entre la communauté développant le noyau Linux et la société en charge du développement de BitKeeper furent rompues, et le statut de gratuité de l'outil fut révoqué. Cela poussa la communauté du développement de Linux et plus particulièrement Linus Torvalds, le créateur de Linux à développer son propre outil en se basant sur les leçons apprises lors de l'utilisation de BitKeeper.

Depuis sa naissance en 2005, Git a évolué et mûri pour être facile à utiliser tout en conservant ses qualités initiales. Il est incroyablement rapide, il est très efficace pour de grands projets et il a un incroyable système de branches pour des développements non linéaires.

### Rudiment de git

#### Des instantanés, pas des différences

La différence majeure entre Git et les autres VCS (Subversion et autres) réside dans la manière dont Git considère les données. Au niveau conceptuel, la plupart des autres VCS gèrent l'information comme une liste de modifications de fichiers. Ces systèmes (CVS, Subversion, Perforce, Bazaar et autres) considèrent l'information qu'ils gèrent comme une liste de fichiers et les modifications effectuées sur chaque fichier dans le temps, comme illustré dans la figure 6-4.

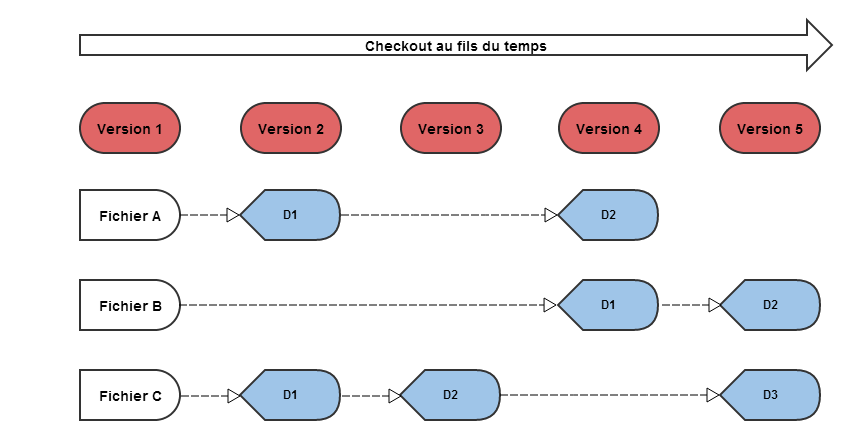


Figure ‑ : Méthode d`enregistrement pour d’autre logiciel de gestion de version.

Git ne gère pas et ne stocke pas les informations de cette manière. À la place, Git pense ses données plus comme un instantané d'un mini système de fichiers. À chaque fois que vous validez ou enregistrez l'état du projet dans Git, il prend effectivement un instantané du contenu de votre espace de travail à ce moment et enregistre une référence à cet instantané. Pour être efficace, si les fichiers n'ont pas changé, Git ne stocke pas le fichier à nouveau, juste une référence vers le fichier original qui n'a pas été modifié. Git pense ses données plus à la manière de la figure 6-5.

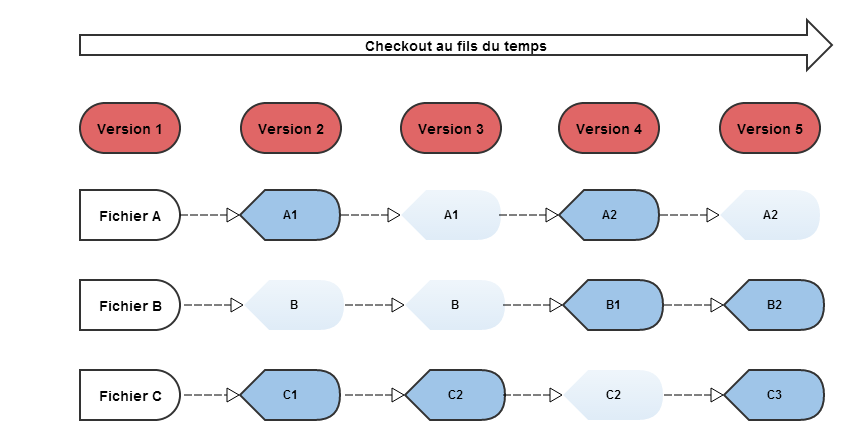


Figure ‑ : Git stocke les données comme des instantanés du projet au cours du temps.

C'est une distinction importante entre Git et quasiment tous les autres VCS. Git a reconsidéré quasiment tous les aspects de la gestion de version que la plupart des autres systèmes ont copiés des générations précédentes. Cela fait quasiment de Git un mini système de fichiers avec des outils incroyablement puissants construits dessus, plutôt qu'un simple VCS.

Presque toutes les opérations sont locales

La plupart des opérations de Git ne nécessite que des fichiers et ressources locaux — généralement aucune information venant d'un autre ordinateur du réseau n'est nécessaire. Si vous êtes habitué à un CVCS où toutes les opérations sont ralenties par la latence des échanges réseau, cet aspect de Git vous fera penser que les dieux de la vitesse ont octroyé leurs pouvoirs à Git. Comme vous disposez de l'historique complet du projet localement sur votre disque dur, la plupart des opérations semblent instantanées.

Par exemple, pour parcourir l'historique d'un projet, Git n'a pas besoin d'aller le chercher sur un serveur pour vous l'afficher ; il n'a qu'à simplement le lire directement dans votre base de données locale. Cela signifie que vous avez quasi-instantanément accès à l'historique du projet. Si vous souhaitez connaître les modifications introduites entre la version actuelle d'un fichier et son état un mois auparavant, Git peut rechercher l'état du fichier un mois auparavant et réaliser le calcul de différence, au lieu d'avoir à demander cette différence à un serveur ou à devoir récupérer l'ancienne version sur le serveur pour calculer la différence localement.

Cela signifie aussi qu'il y a très peu de choses que vous ne puissiez réaliser si vous n'êtes pas connecté ou hors VPN. Si vous voyagez en train ou en avion et voulez avancer votre travail, vous pouvez continuer à gérer vos versions sans soucis en attendant de pouvoir de nouveau vous connecter pour partager votre travail. Si vous êtes chez vous et ne pouvez avoir une liaison VPN avec votre entreprise, vous pouvez tout de même travailler. Pour de nombreux autres systèmes, faire de même est impossible ou au mieux très contraignant. Avec Perforce par exemple, vous ne pouvez pas faire grand-chose tant que vous n'êtes pas connecté au serveur. Avec Subversion ou CVS, vous pouvez éditer les fichiers, mais vous ne pourrez pas soumettre des modifications à votre base de données (car celle-ci est sur le serveur non accessible). Cela peut sembler peu important a priori, mais vous seriez étonné de découvrir quelle grande différence cela peut constituer à l'usage.

#### Git gère l'intégrité

Dans Git, tout est vérifié par une somme de contrôle avant d'être stocké et par la suite cette somme de contrôle, signature unique, sert de référence. Cela signifie qu'il est impossible de modifier le contenu d'un fichier ou d'un répertoire sans que Git ne s'en aperçoive. Cette fonctionnalité est ancrée dans les fondations de Git et fait partie intégrante de sa philosophie. Vous ne pouvez pas perdre des données en cours de transfert ou corrompre un fichier sans que Git ne puisse le détecter.

Le mécanisme que Git utilise pour réaliser les sommes de contrôle est appelé une empreinte SHA-1. C'est une chaîne de caractères composée de 40 caractères hexadécimaux (de '0' à '9' et de 'a' à 'f') calculée en fonction du contenu du fichier ou de la structure du répertoire considéré. Une empreinte SHA-1 ressemble à ceci :

24b9da6552252987aa493b52f8696cd6d3b00373

Vous trouverez ces valeurs à peu près partout dans Git car il les utilise pour tout. En fait, Git stocke tout non pas avec des noms de fichiers, mais dans la base de données Git indexée par ces valeurs.

#### Généralement, Git ne fait qu'ajouter des données

Quand vous réalisez des actions dans Git, la quasi-totalité d'entre elles ne font qu'ajouter des données dans la base de données de Git. Il est très difficile de faire réaliser au système des actions qui ne soient pas réversibles ou de lui faire effacer des données d'une quelconque manière. Par contre, comme dans la plupart des systèmes de gestion de version, vous pouvez perdre ou corrompre des modifications qui n'ont pas encore été entrées en base ; mais dès que vous avez validé un instantané dans Git, il est très difficile de le perdre, spécialement si en plus vous synchronisez votre base de données locale avec un dépôt distant.

Cela fait de l'usage de Git un vrai plaisir, car on peut expérimenter sans danger de casser définitivement son projet

#### Les trois états

Git gère trois états dans lesquels les fichiers peuvent résider : validé, modifié et indexé. Validé signifie que les données sont stockées en sécurité dans votre base de données locale. Modifié signifie que vous avez modifié le fichier mais qu'il n'a pas encore été validé en base. Indexé signifie que vous avez marqué un fichier modifié dans sa version actuelle pour qu'il fasse partie du prochain instantané du projet.

Ceci nous mène aux trois sections principales d'un projet Git : le répertoire Git, le répertoire de travail et la zone d'index.

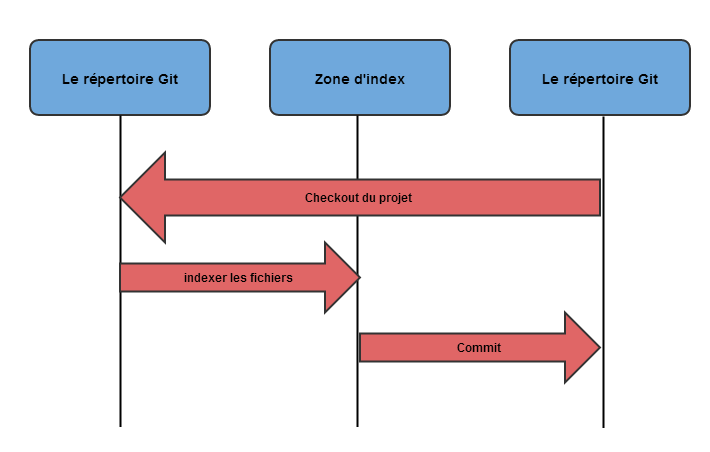


Figure ‑ : Répertoire de travail, zone d'index et répertoire Git

Le répertoire Git est l'endroit où Git stocke les méta-données et la base de données des objets de votre projet. C'est la partie la plus importante de Git, et c'est ce qui est copié lorsque vous clonez un dépôt depuis un autre ordinateur.

Le répertoire de travail est une extraction unique d'une version du projet. Ces fichiers sont extraits depuis la base de données compressée dans le répertoire Git et placés sur le disque pour pouvoir être utilisés ou modifiés.

La zone d'index est un simple fichier, généralement situé dans le répertoire Git, qui stocke les informations concernant ce qui fera partie du prochain instantané.

L'utilisation standard de Git se passe comme suit :

1. vous modifiez des fichiers dans votre répertoire de travail ;

2. vous indexez les fichiers modifiés, ce qui ajoute des instantanés de ces fichiers dans la zone d'index ;

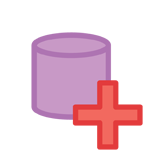
3. vous validez, ce qui a pour effet de basculer les instantanés des fichiers de l'index dans la base de données du répertoire Git.

Si une version particulière d'un fichier est dans le répertoire Git, il est considéré comme validé. S'il est modifié mais a été ajouté dans la zone d'index, il est indexé. S'il a été modifié depuis le dernier instantané mais n'a pas été indexé, il est modifié. Dans le chapitre 2, vous en apprendrez plus sur ces états et comment vous pouvez en tirer parti ou complètement les occulter.

### Base de Git :

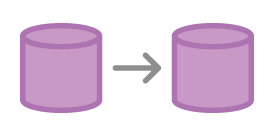
Cette partie donne un aperçu sur les commandes Git les plus importante et explique tous les outils dont vous avez besoin pour démarrer un nouveau projet, créer un dépôt Git, enregistrer des instantanés de votre projet, et consulter l'historique de votre projet sous git. Plus tard dans ce chapitre un tutorial expliquera comment utiliser toutes ces commandes.

#### Configuration d'un dépôt Git

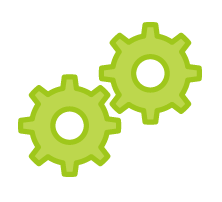
$ git init

La commande git init initialise un nouveau dépôt Git. Si vous voulez placer

un projet sous contrôle de révision, c'est la première commande que vous devez utiliser.

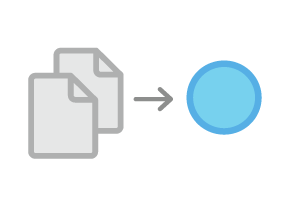
**$ git clone

La commande git clone crée une copie d'un dépôt Git existant. Le clonage est le moyen le plus courant pour les développeurs afin d'obtenir une copie de travail d'un référentiel central.

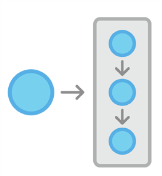
$ git config

La commande git config est un moyen pratique pour définir les options de configuration de votre installation de Git. Vous aurez généralement seulement besoin de l'utiliser immédiatement après l'installation de Git sur une nouvelle machine de développement.

#### Enregistrement d’instantané

$ git add

La commande git add déplace les modifications du répertoire de travail de la zone d'index. Cela vous donne la possibilité de préparer un instantané avant de commiter à l'histoire officielle.



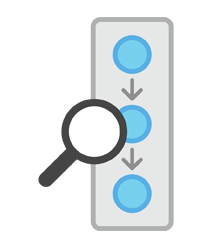
$ git commit

La commande git commit prend l’instantané indexé et le commit à l'historique du projet. Combiné avec git add, ils définissent le flux de travail de base pour tous les utilisateurs de Git.

#### L'inspection d'un dépôt Git

$ git statut

La commande git status affiche l'état du répertoire de travail et l'instantané indexé. Cette commande est très utile en collaboration avec git add et git commit pour voir exactement ce qui est inclus dans le prochain snapshot.

$ git log

La commande git log vous permet d'explorer les révisions précédentes du projet. Il offre plusieurs options de mise en forme pour l'affichage que vous pouvez voir à l’aide de la commande git log –help.

### Les branches avec Git

Quasiment tous les VCS ont une forme ou une autre de gestion de branche. Créer une branche signifie diverger de la ligne principale de développement et continuer à travailler sans se préoccuper de cette ligne principale. Dans de nombreux outils de gestion de version, cette fonctionnalité est souvent chère en ressources et nécessite de créer une nouvelle copie du répertoire de travail, ce qui peut prendre longtemps dans le cas de gros projets.

De nombreuses personnes font référence au modèle de gestion de branche de Git comme LA fonctionnalité et c'est sûrement la spécificité de Git par rapport à la communauté des gestionnaires de version. Pourquoi est-elle si spéciale ? La méthode de Git pour gérer les branches est particulièrement légère, permettant de réaliser des embranchements quasi instantanément et de basculer entre les branches généralement aussi rapidement. À la différence de nombreux autres gestionnaires de version, Git encourage à travailler avec des méthodes qui privilégient la création et la fusion de branches, jusqu'à plusieurs fois par jour. Bien comprendre et maîtriser cette fonctionnalité est un atout pour faire de Git un outil unique qui peut littéralement changer la manière de développer.

#### Branches historiques :

Au lieu d'une seule branche maîtresse, Le flux de travail utilise deux branches pour enregistrer l'historique du projet. La branche master stocke l'historique de sortie officielle, et la branche Develop sert comme une branche de l'intégration des fonctionnalités.

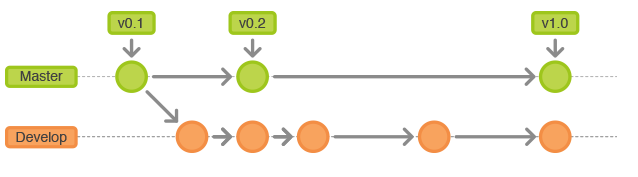


Figure ‑ : Model de branches historiques

#### Branches de fonctionnalité

Chaque nouvelle fonctionnalité devrait résider dans sa propre branche, qui peut être poussé vers le répertoire central pour la sauvegarde et la collaboration. Mais, au lieu de brancher le master, les branches de fonctionnalité utilisent Develop comme leur branche mère. Quand une fonction est terminée, elle se fait fusionnée vers Develop. Les nouvelles fonctionnalités ne doivent jamais communiquer directement avec la branche mère.

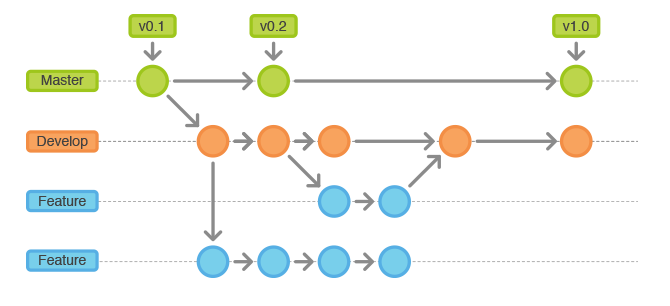


Figure ‑ : Model de branche de fonctionnalité

#### Branche de Release:

Une fois la branche Develop acquis suffisamment de fonctionnalités pour une version ou une date de sortie prédéterminé est proche, vous débourser une branche de sortie hors de Develop. La création de cette branche commence le prochain cycle de release, donc pas de nouvelles fonctionnalités qui peuvent être ajoutées après ce point uniquement des corrections de bugs, génération de documentation. Une fois que le release est prêt à être expédié, il se fusionne avec le master et étiqueté avec un numéro de version. En outre, il doit être fusionné vers Develop, qui peut avoir évolué depuis que le release a été lancé.

En utilisant une branche dédiée à la préparation des releases, il est possible pour une équipe de raffiner la version actuelle pendant qu'une autre équipe continue à travailler sur les fonctionnalités de la prochaine version.

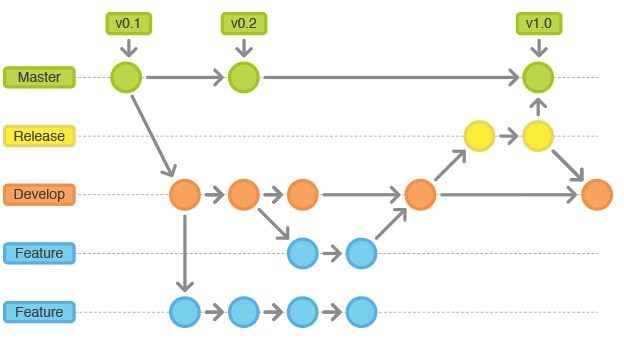


Figure ‑ : Model de branche de release.

#### Branche de maintenance

Les branches de maintenance ou "hotfix" sont utilisés pour réparer rapidement les versions de production. Il s'agit de la seule branche qui devrait débourser directement sur le master. Dès que la correction est terminée, il devrait être fusionné à la fois avec la master et la develop (ou la branche de version en cours), et le maître doit être étiqueté avec un numéro de version mise à jour.

Avoir une ligne dédiée au développement pour corriger les bugs permet à l’équipe de développement de corriger des bugs sans interrompre le flux de travail ou d'attendre le prochain cycle de release.

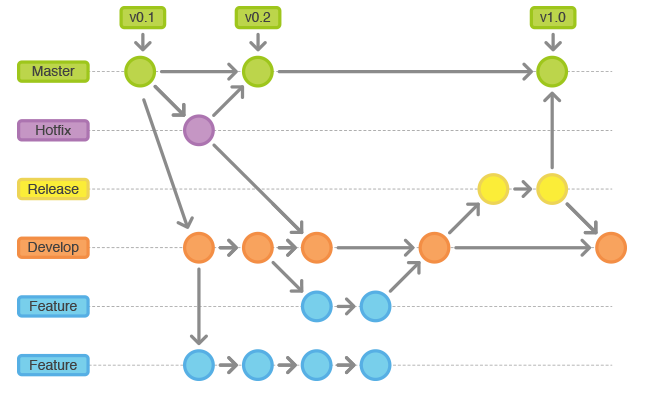


Figure ‑ : Model de branches de maintenance.

#### Conclusion

Pour conclure Git est un outil très puissant pour la gestion de version de n’importe quel fichier sur un ordinateur. Vu que le principe de git se base sur une base de données distribué, l’utilisation d’un serveur est indispensable. La prochaine partie sera dédier au plus gros hébergeur de projet sous git.

## GitHub

### GitHub c’est Quoi ?

GitHub est un service d'hébergement Web pour les projets de développement de logiciels qui utilisent le système de contrôle de révision de Git. GitHub propose deux plans payés référentiels privés, et des comptes gratuits pour les projets open source. En mai 2011, GitHub a été le site de dépôt de code le plus populaire pour les projets open source.

Le site fournit des fonctionnalités de réseau social tel que les commentaires, les fans et le graphe du réseau social pour afficher comment les développeurs travaillent sur leurs versions d'un même répertoire.

### Statistiques

GitHub a été lancé en Avril 2008.

Dans un discours au siège de Yahoo! le 24 Février 2009, les membres de l'équipe GitHub ont annoncé que durant la première année que GitHub était en ligne, il a accumulé 46.000 dépôts publics, 17.000 d'entre eux au cours du mois précédent seul. A cette époque, environ 6.200 dépôts avaient été dupliqué au moins une fois et 4,600 fusionné. Le 5 Juillet 2009, un blog Github a annoncé qu'ils ont atteint la barre des 100.000 utilisateurs.

Dans un autre discours prononcé à Yahoo! le 27 Juillet 2009, Tom Preston-Werner(1) a annoncé que les effectifs avaient augmenté de 90.000 dépôts publics uniques, 12.000 ayant été dupliqué au moins une fois, pour un total de 135.000 dépôts. En Juillet 2010, GitHub a annoncé qu'il accueille 1 million de dépôts. En Avril 2011, GitHub a annoncé qu'il accueille 2 millions de dépôts.

Le 21 Septembre 2011 GitHub a annoncé qu'il avait atteint plus d'1 million d'utilisateurs. [14]

Le 13 Septembre 2012 sur leur page d'accueil, GitHub a annoncé qu'il avait plus de 2,1 millions d'utilisateurs accueillant plus de 3,7 millions de dépôts.

Le 19 Décembre 2012 GitHub a annoncé qu'il avait plus de 2,8 millions d'utilisateurs accueillant plus de 4,6 millions de dépôts.

Le 16 Janvier 2013, GitHub a annoncé qu'il avait franchi la barre des 3 millions d'utilisateurs et accueillant plus de 5 millions de dépôts.

### Social coding : nouvelle tendance dans le développement logiciel

Le codage social ou social coding approche le développement de logiciels d'une manière qui met en évidence la collaboration entre les développeurs en leur fournissant un environnement pour discuter, partager et créer.

Appelé aussi cloud coding est vraiment une innovation au développement car elle met le travail d'équipe dans un mélange particulier, puisque ça regroupe des développeurs ayant des compétences et des expériences différentes pour résoudre un problème. Grâce à la collaboration, ils sont capables de combiner chacune de leurs spécialités afin d'améliorer des logiciels existants ou développer de nouveaux produits.

Le social coding permet aux développeurs de faire beaucoup plus que de travailler avec un logiciel. Ce qui rend le cœur de toute communauté dépasser la capacité de ses membres, est le fait qu’ils puissent communiquer entre eux. Les développeurs peuvent partager leurs idées et apprendre des autres. Ils peuvent garder un œil sur les tendances importantes ou les nouveautés de l’industrie. Ils peuvent suivre ce que les autres travaillent sur et rester à jour sur les progrès réalisés, tout en apportant des informations sur leurs propres projets.

#### Pourquoi le social coding est l'avenir du développement ?

Dans le monde numérique d'aujourd'hui en grande partie, les réseaux sociaux sont devenus une partie intégrante pour notre capacité à rester en contact avec les autres. Et, au sein des entreprises, il est de plus en plus une composante essentielle des stratégies de marketing.

Les développeurs sont déjà contestés avec un travail exigeant environnement de travail avec le code n'est pas une tâche simple. Le social coding leur donne un moyen de se connecter avec d'autres développeurs dans un environnement qui encourage la coopération et facilite la collaboration, ce qui se traduira par un meilleur produit plus rapidement.

### Comment mettre tout ça en œuvre ?

Etape 1 :

On commence par créer un compte gratuit sur github.com, L’interface est assez simple il faut juste un identifient un mot de passe et une adresse email.

Etape 2 :

Une fois le compte est crée on peut accéder à l’interface de GitHub. Maintenant il faut crée un nouvel répertoire pour contenir notre projet.

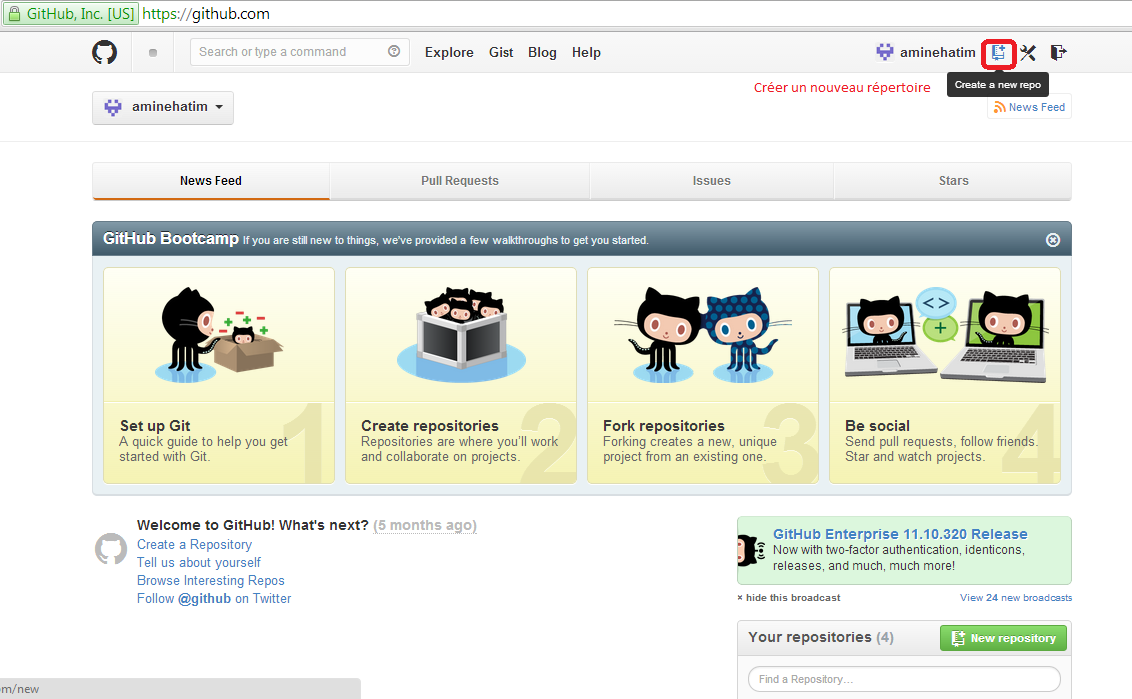


Figure ‑ : Page web github

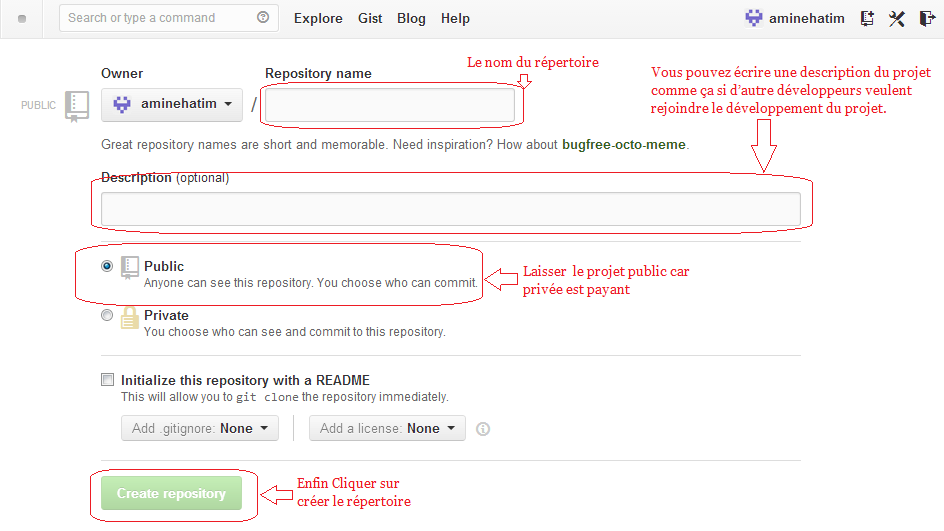


Figure ‑ : GitHub : création d'un dépôt

Une fois le répertoire sera créer vous pouvez y accéder. Dans la figure qui suit le lien ssh représente l’adresse de notre répertoire ce dernier est essentiel pour synchroniser notre répertoire local a notre répertoire distant.

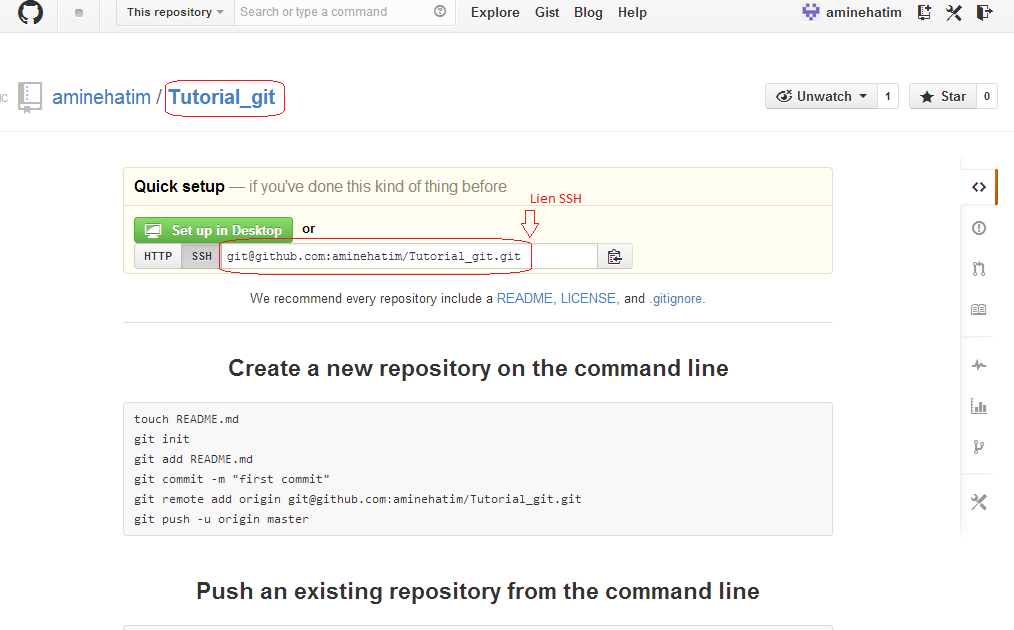


Figure ‑ : GitHub : lien ssh

Etape 3 :

Dans cette étape il faut télécharger et installer Git et TurtoiseGit. TurtoiseGit est simplement une interface graphique pour Git, qui permet d’utiliser git sans se soucier à connaitre le toutes les commandes de Git.

Installation de Git :

Téléchargez msysgit depuis sourceforge.org et puis suivez les étapes suivantes :

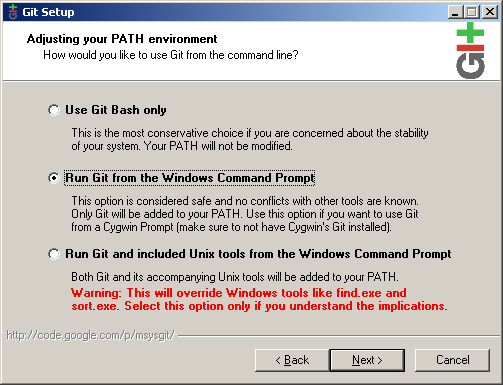


Figure ‑ : Installation de git

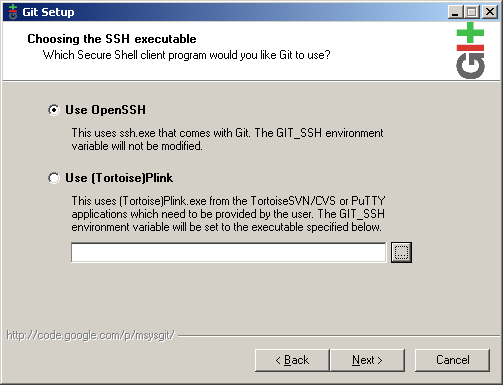


Figure ‑ : Installation de git

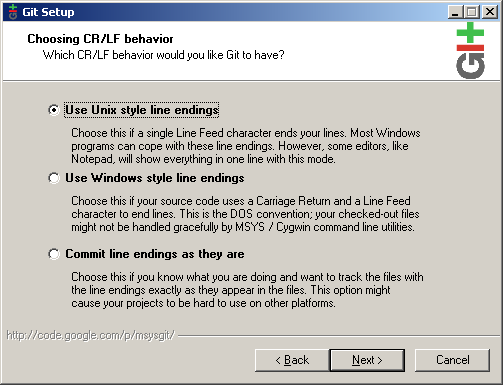


Figure ‑ : Installation de git

En ce qui concerne TurtoiseGit, il peut être téléchargé depuis sourceforge.org, et pour l’installation suivez les paramètres par défaut.

Etape 4 :

Maintenant il faut générer deux clés ssh une privé et une public. Une clé ssh est considérée comme un mot de passe mais un peu plus sécurisé. Cette clé va nous permettre une fois configurer avec Git, une authentification automatique à chaque connexion avec le répertoire distant. La clé privé reste sur votre ordinateur cependant la clé publique va être ajouté au compte GitHub.

Pour cela on va utiliser puttyGen : menu démarrer -> tous les programmes -> TurtoiseGit -> puttyGen.

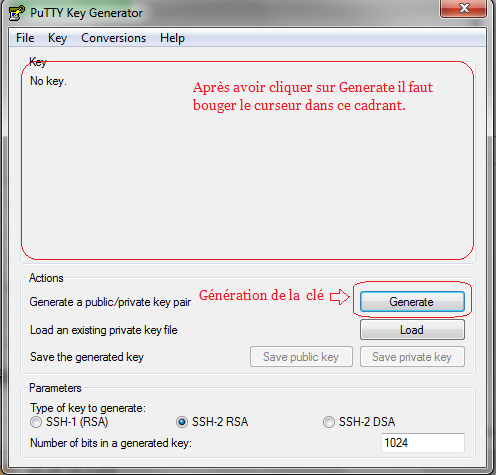


Figure ‑ : Génération de la clé ssh

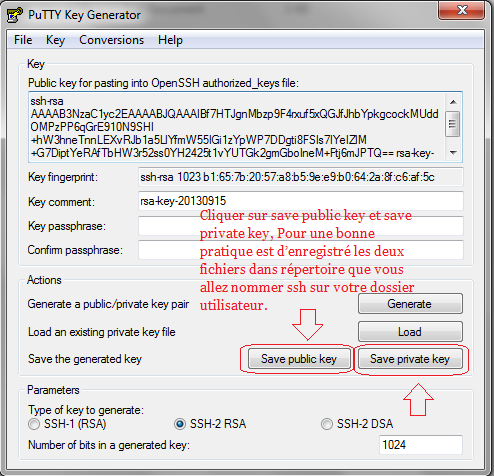


Figure ‑ : Enregistrement de la clé ssh

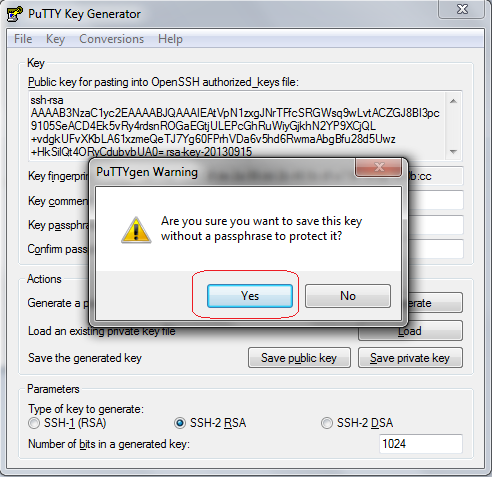


Figure ‑ : Enregistrement de la clé ssh

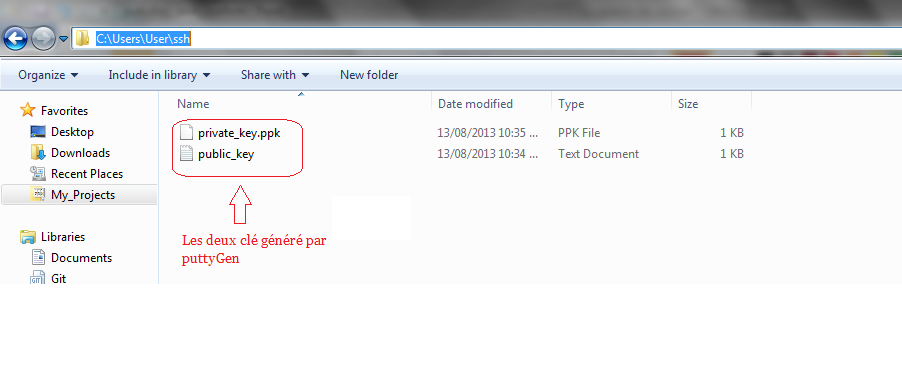


Figure ‑ : Les deux clés ssh

Etape 5 :

Ajouter la clé publique au compte GitHub. Pour cela copier le contenu de la clé publique (public\_key.txt), et suivez les étapes suivantes :

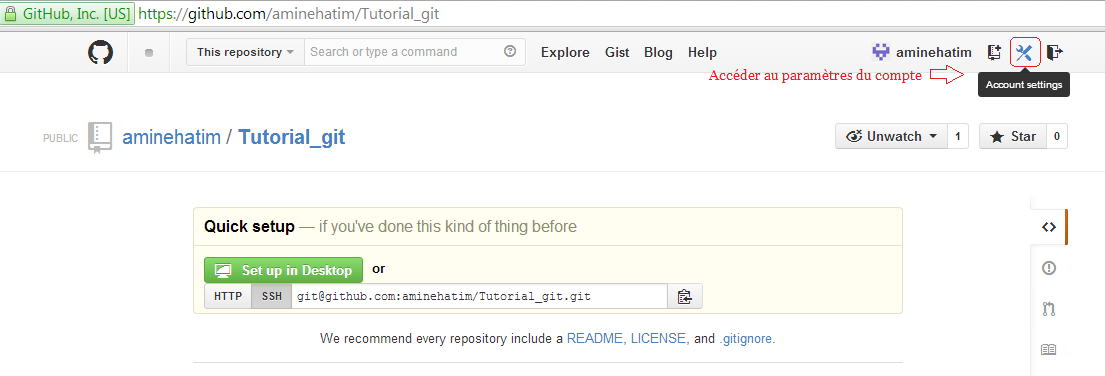


Figure ‑ : github options du compte

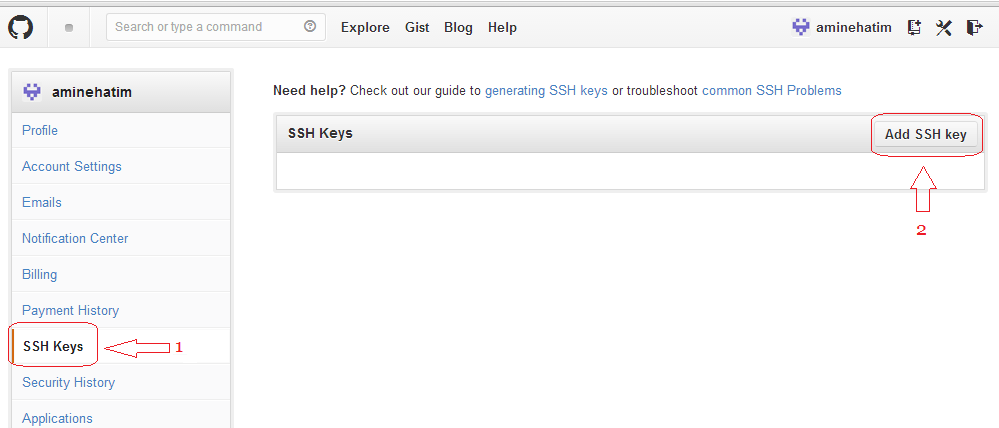


Figure ‑ : GitHub : ajout de la cle ssh



Figure ‑ : GitHub éditer la clé ssh

Etape 7 :

A cette étape il faut créer un dossier pour le projet et le lier au compte GitHub. Apres avoir créer un répertoire pour le projet dans ce cas Tutorial\_git. On doit l’initialiser comme répertoire Git.

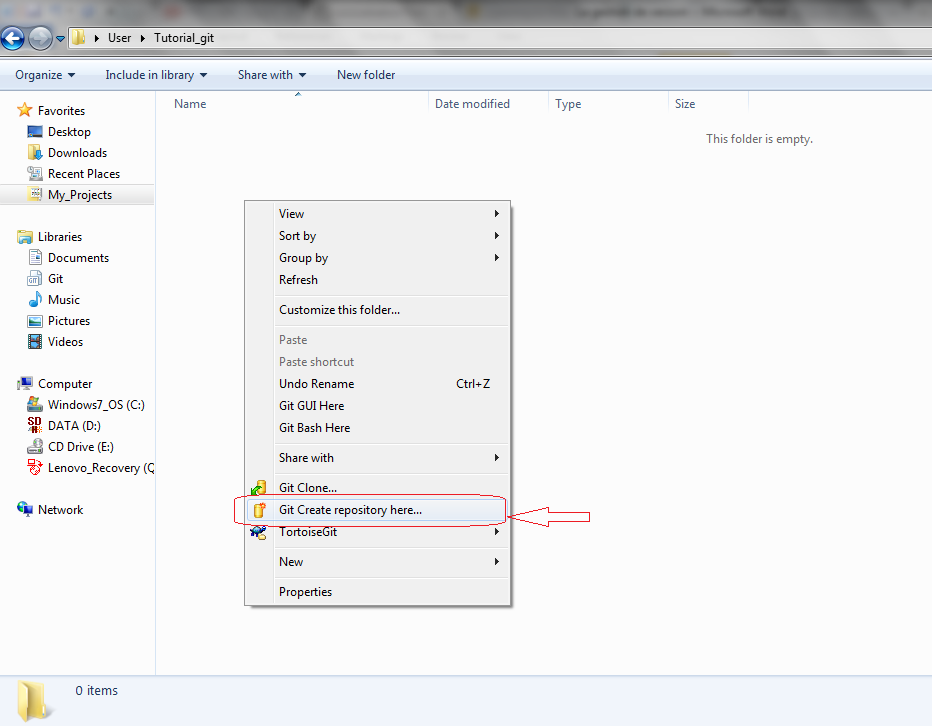


Figure ‑ : Ajout du répertoire de travail



Figure ‑ : Utiliser un répertoire ''bare''

Puisque que c’est dans ce répertoire qu’on va créer notre projet et y travailler dessus, il faut laisser la case décocher. Un ‘Bare directory’ est répertoire de sauvegarde local seulement on ne peut pas faire des modifications sur une copie ‘bare’ du projet.

Avant de commencer d’ajouter des fichiers a notre répertoire, il faut créer un utilisateur Git, bouton droit-> gitTurtoise->settings :

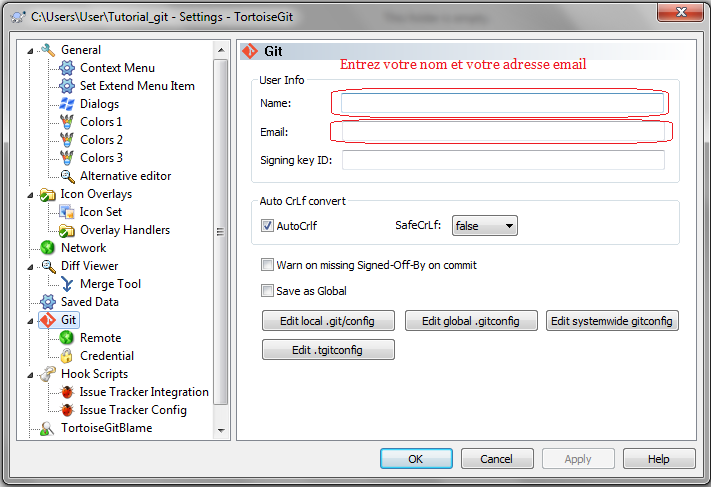


Figure ‑ : Ajout d'un nouvel utilisateur git

Ensuite il faut ajouter le lien ssh de notre répertoire pour que Git puisse synchroniser les deux répertoires distant et locale.

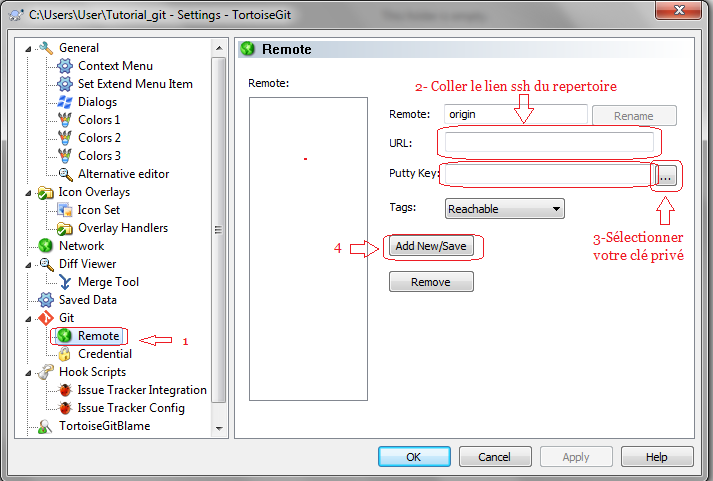


Figure ‑ : Ajout du lien ssh sur git

Dès qu’on clique sur ‘Add New/Save ‘ une fenêtre va apparaitre pour demander de faire un ‘fetch’. La commande ‘fetch’ permet de faire une copie du répertoire distant vers le répertoire local. Dans notre cas puisque le répertoire distant est vide on n’a pas besoin de faire un ‘fetch’.

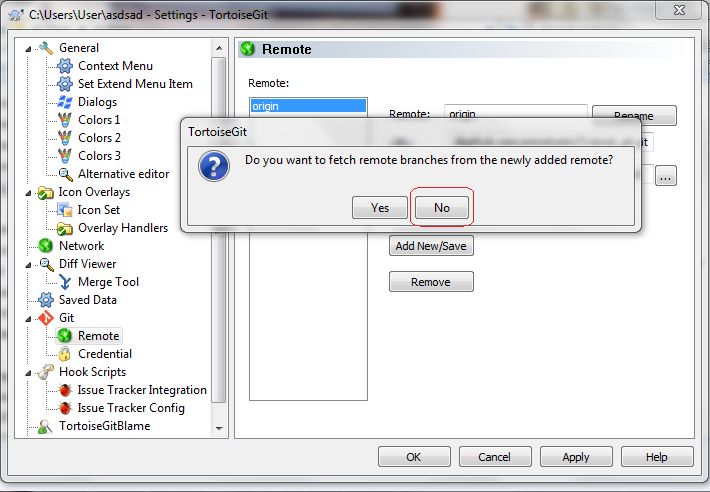


Figure ‑ : Finalisation du lien ssh

Etape 8 :

Maintenant tous est configurer, on va créer un nouveau fichier dans le répertoire de projet et puis faire le premier commit.

On va commencer par créer un simple fichier texte contenant « Hello world ! ». A cette étape le répertoire local est synchroniser avec le distant, mais aucun fichier n’est ajouter pour la gestion de version. Donc on va utiliser la commande git add pour ajouter notre nouveau fichier.



Figure ‑ : Ajout du premier fichier



Figure ‑ : Validation de l'ajout du fichier

Maintenant git commence à faire un suivi des modifications sur ce fichier. Il reste qu’a le comitter vers le répertoire distant.

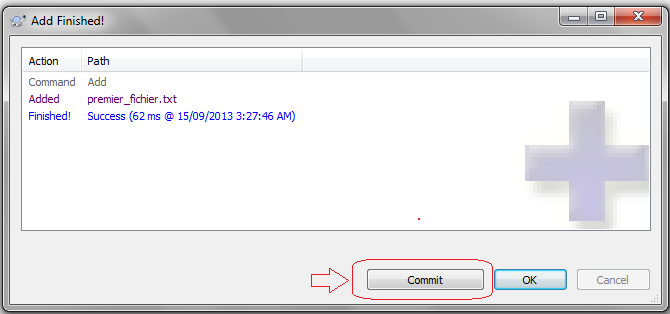


Figure ‑ : Premier commit

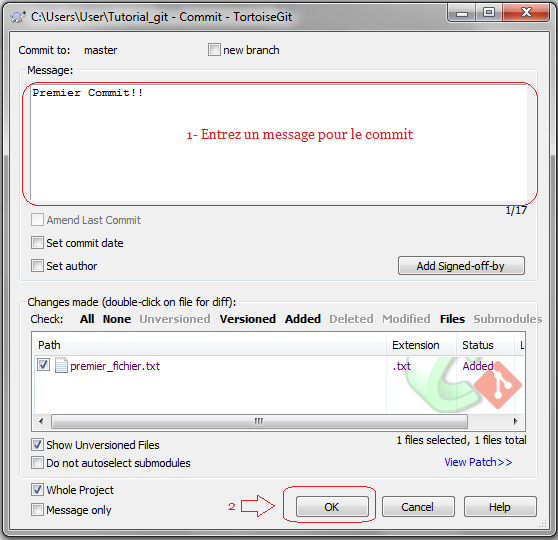


Figure ‑ : Ajout du message du commit

Le message du commit doit être claire est expliqué exactement le changement appliquer par le commit qui être très utile lorsque un problème figure dans le code source ou si un autre développeur veut exactement les changements cas à apporter ce commit.

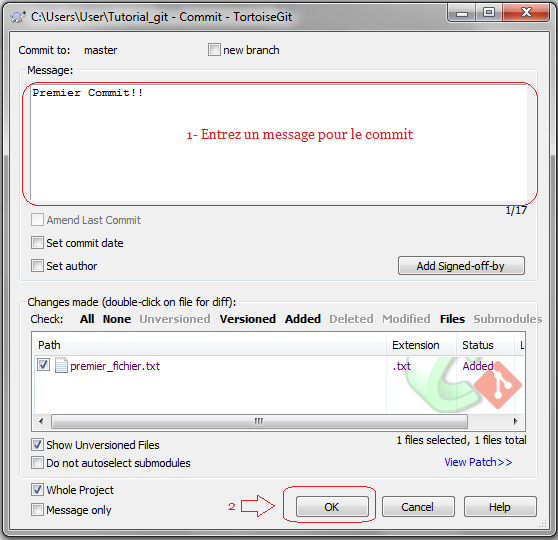


Figure ‑ : Ajout du message de commit

A ce niveau Git a enregistré les changements sur notre fichier seulement sur la version local du répertoire de travail il reste qu’à l’envoyé sur le répertoire distant pour cela on utilise la commande ‘push ‘.

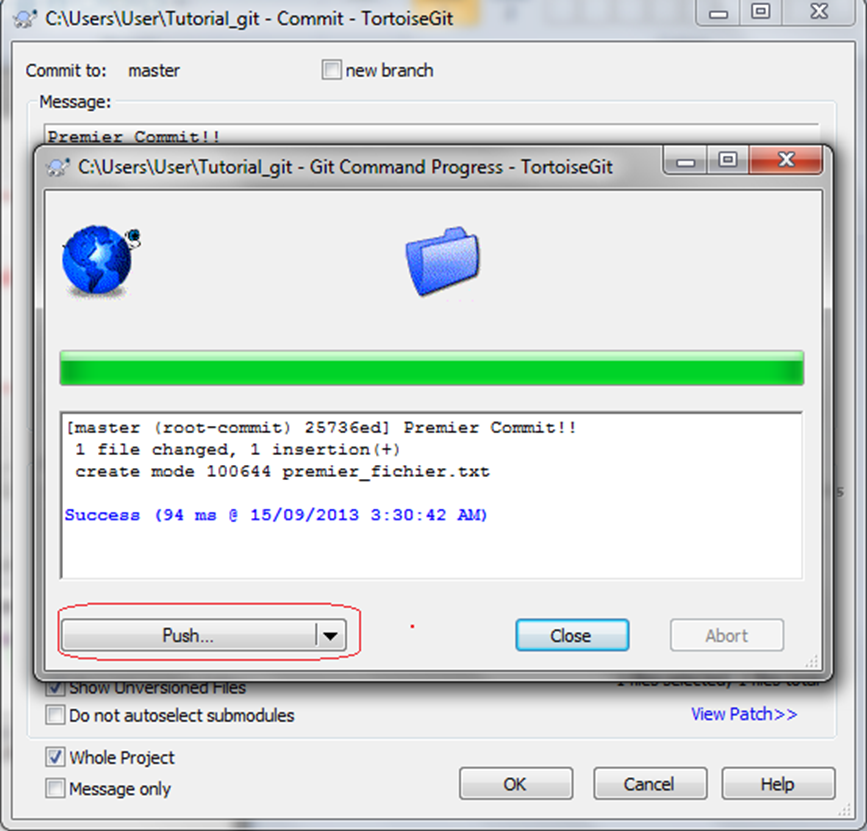


Figure ‑ : Premier push

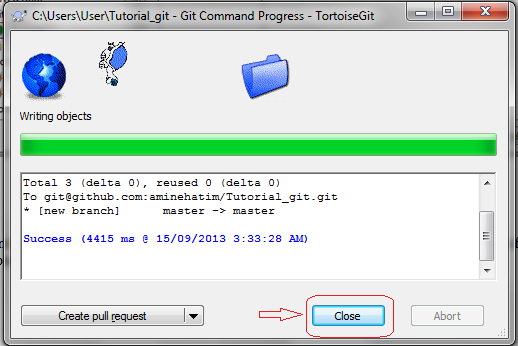


Figure ‑ : Finalisation du push

Enfin le fichier existe sur le répertoire local est distant si on retourne au répertoire distant sur GitHub on peut voir que le fichier existe sur le répertoire distant.

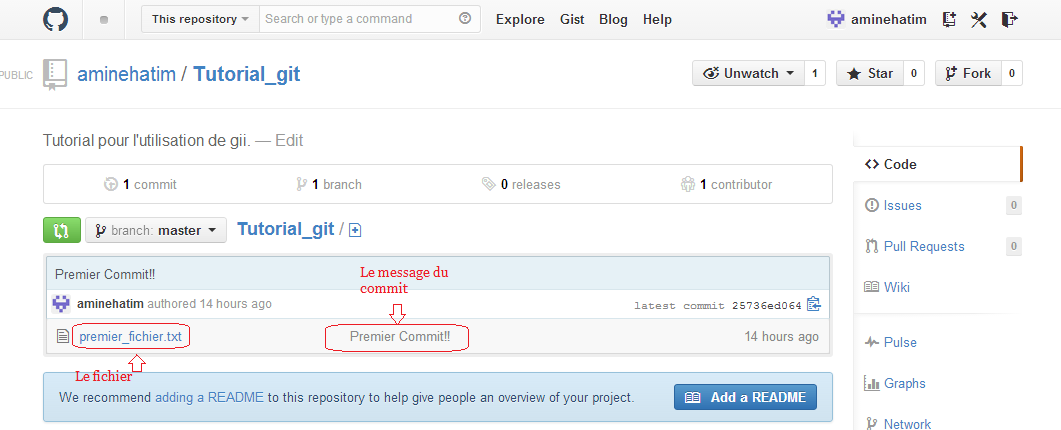


Figure ‑ : Présentation du fichier sur github

1. [↑](#footnote-ref-1)