Programmation et utilisation du FPGA pour la validation et la vérification de circuits électroniques

Johanna Mariani

le cnam
École d'ingénieurs eicnam

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

MEMOIRE

Présenté en vue d'obtenir

Le DIPLOME D'INGENIEUR CNAM

SPECIALITE : ELECTRONIQUE

Par

Johanna MARIANI

Programmation et Utilisation du FPGA Pour la validation et la vérification De circuits électroniques

Soutenu le 8 Février 2011

JURY

Président : M. Daniel ROVIRAS, Responsable de la Chaire Electronique au CNAM de Paris

Membres : M. Louis BALME, Responsable de la filière Electronique au CNAM de Grenoble
M. Michel CELETTE, Professeur au CNAM de Grenoble
M. Thomas URBITSCH, Responsable Ingénieur Application ST-Ericsson
M. Emmanuel ALLIER, Ingénieur concepteur à ST-Ericsson

le cnam
École d'ingénieurs eicnam

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

MEMOIRE

Présenté en vue d'obtenir

Le DIPLOME D'INGENIEUR CNAM

SPECIALITE : ELECTRONIQUE

Par

Johanna MARIANI

Programmation et Utilisation du FPGA Pour la validation et la vérification De circuits électroniques

Soutenu le 8 Février 2011

Les travaux relatifs à ce mémoire ont été effectués au sein de l'entreprise ST-Ericsson sur le site de Grenoble, sous la direction de M. Simon VALCIN.

REMERCIEMENTS

Je tiens à remercier toute l'équipe pédagogique du CNAM Grenoble et les intervenants professionnels responsables de la formation d'ingénieur CNAM en électronique, pour avoir assuré la partie théorique de celle-ci.

Je tiens à remercier pour leur implication, leur aide et leur participation durant mon stage :

Angelo NAGARI, responsable de l'équipe conception IP AMS dans laquelle j'ai évolué durant ce stage, pour m'avoir accordé toute sa confiance et pour son soutient durant toutes les phases du projet.

Simon VALCIN, mon tuteur pour le temps qu'il m'a consacré.

Emmanuel ALLIER, ingénieur concepteur, pour sa disponibilité et son expertise des circuits audio.

Tous les techniciens et ingénieurs du laboratoire d'application et de validation AMS et RF pour leurs bonnes humeurs et leurs précieux encouragements.

Je tiens à remercier en particulier Thomas URBITSCH et Jean-Louis BERNET pour leur disponibilité et leur aide en cette fin de mémoire.

Enfin, je remercie tous mes proches qui durant ces cinq années d'études, m'ont supporté et surtout encouragé.

Sommaire

AVERTISSEMENTS

Note 1 : Ce rapport contient de nombreux termes anglo-américains, pour deux raisons majeures : L'anglais est la langue officielle dans le domaine du semi-conducteur. La dimension internationale de ST-Ericsson impose une langue de référence pour communiquer. Néanmoins pour faciliter la lecture, ces termes seront traduits.

Note 2: Les références bibliographiques sont notées entre crochets de la façon suivante : [Numéro de la référence de la liste bibliographique].

Liste des acronymes

AMS : *Analog and Mixte Signal (Signal analogique et mixte)*

ADC : *Analog Digital Converter (Convertisseur Analogique-numérique)*

ASCII : *American Standard Code for Information Interchange (Code américain normalisé pour l'échange d'information)*

ASIC : *Application-Specific Integrated Circuit (Circuit intégré pour application spécifique)*

ASSP : *Application Specific Standard Product (Circuit intégré pour application standard)*

BGA : *Ball Grid Arrays (matrice de billes)*

BU : *Business Unit (Unité d'affaire)*

CIC : *Cascaded Integrator Comb (filtre d'intégrateurs en cascade)*

DAC : *Digital Analog Converter (Convertisseur numérique-analogique)*

DSP : *Digital Signal Processing (Processeur de signaux numériques)*

FFT : *Fast Fourier Transform (Transformée de Fourier Rapide)*

FIR : *Finite Impulse Response (Réponse impulsionnelle finie)*

FPGA : *Field Programmable Gates Arrays (Matrice de portes logiques programmable)*

FS : *Full Scale (Pleine échelle)*

GUI : *Graphical User Interface (Interface graphique)*

I2C : *Inter Integrated Circuit (bus série et synchrone)*

IDE : *Integrated Development Environment (Environnement de développement integer)*

IP : *Intellectual Property (cellule de propriété intellectuelle)*

LDO : *Low Drop Output (régulateur linéaire avec une tension de déchet faible entre l'entrée et la sortie)*

LSB : *Low significative Bit (plus petit pas de niveau)*

OSR : *Over Sampling Rate (Taux de surréchantillonnage)*

PDM : *Pulse Density Modulation (Modulation de densité de durée)*

R&D : *Recherche et développement*

RF : *Radio Frequency (radio fréquence)*

SMPS : *Switched Mode Power Supply (Alimentation à décycleage)*

SOPC : *System On Programmable Chip (système sur puce reprogrammable)*

SPI : *Serial Peripheral Interface (bus d'interface série et synchrone full duplex)*

SNR : *Signal to Noise Ratio (Rapport signal à bruit)*

VISA : *Virtual Instrument Software Architecture (Interface de programmation d'entrée/sortie utilisée dans l'instrumentation)*

La société ST-Ericsson

Une mission

[1][2]ST-Ericsson est un acteur de l'industrie spécialisé dans la micro-électronique. Sa mission est de concevoir et de développer des semi-conducteurs à destination de technologies sans-fil. Cette société offre un portefeuille important de produits spécifiques et standards aux grands acteurs du marché de la téléphonie mobile et autres applications portables. Ses clients principaux sont les constructeurs de téléphones mobiles comme Nokia ou Sony-Ericsson. Plus de la moitié des téléphones portables en usage aujourd'hui sont équipés de produits et de technologies ST-Ericsson.
Composée d'une majorité d'ingénieurs, 70%, ses effectifs sont surtout consacrés à la recherche et aux développements, R&D. Par ailleurs, cette nouvelle entreprise ne possède pas d'usine de fabrication. Le principal fondeur est à ce jour STMicroelectronics.

Une entreprise « jeune »

[1]C'est en février 2009 que ST NXP Wireless et les Plateformes Mobiles d'Ericsson (EMP) s'associent successivement pour créer un nouvel acteur du marché de la téléphonie mobile, ST-Ericsson. Cette initiative a permis de réunir d'importantes ressources de développements de semi-conducteurs. Ceci permettant de devenir un poids lourd européen dans ce secteur, contrant ainsi les géants américains, tel que Texas-Instrument ou Qualcomm. Cette co-entreprise d'environ 8000 salariés, est présente sur plusieurs sites en France (Grenoble, Crolles, Le Mans, Sophia, Paris), mais aussi en Europe (Suède, Finlande, Suisse, Allemagne, Belgique, Hollande, Angleterre) et enfin en dehors de l'espace Schengen (USA, Chine, Inde).

Pourtant très jeune, la société ST-Ericsson bénéficie de fondements historiques importants. Son histoire découle directement de celles de STMicroelectronics et d'Ericsson.

Concernant STMicroelectronics, c'est en 1987 que le groupe nationalisé Thomson a fusionné ses activités électroniques avec la firme publique Italienne SGS-Microelectronica. La fusion de ces deux puissants groupes de la microélectronique a permis de créer le deuxième groupe européen et le douzième mondial, SGS_THOMSON. Les deux entreprises étaient très complémentaires tant sur les produits, que sur les marchés. Suite au retrait de THOMSON en 1997, la société s'appelle désormais STMicroelectronics. En 2007, elle crée avec la partie « semi-conducteur » de la compagnie Philips, NXP, la co-entreprise ST-NXP Wireless.

[3]Concernant Ericsson, c'est une société Suédoise créée en 1876, par Lars Magnus Ericsson qui avait ouvert sa première boutique de réparation d'équipements télégraphiques. Elle s'est implantée pour la première fois en France à partir de 1911 en créant une société de téléphones Ericsson. Elle est devenue au fil des années une société de télécommunication, de services et d'infrastructures télécom à destination d'abonnés dans le monde entier. La société Ericsson se tourne dans les années

1990 vers des solutions mobiles, haut débit et multimédia. En 2001, elle fusionne avec un géant de l'électronique, Sony, pour créer une co-entreprise de fabrication de téléphones mobiles Sony-Ericsson. Enfin c'est en février 2009, qu'elle crée une seconde co-entreprise avec ST-NXP baptisée ST-Ericsson, rassemblant ses activités de plateforme mobile.

Relativement récente, la société a connu une première année difficile en termes financiers. La crise économique a fortement impacté le marché des semi-conducteurs. ST-Ericsson a été contraint d'annoncer un plan de licenciement à travers ses différents sites. Ces derniers ont connu d'importantes restructurations. Aujourd'hui, elle tend à s'ouvrir au nouveau marché chinois où se bâtissent de nombreux réseaux mobiles 3G.

Un site historique devenu stratégique

[1]Cela fait 40 ans, que le site de Grenoble accueille les différentes évolutions des sociétés mères. Il fut d'abord un laboratoire d'innovation des techniques électroniques, le LETI, puis un centre manufacturier pour SGS-Thomson avant de devenir et de rester jusqu'à aujourd'hui un centre de recherche et développement et de « divisions produits » pour STMicroelectronics et ST-Ericsson. La division produit implique que sa principale fonction est d'assurer le développement, la promotion et le support des produits et applications de chaque segment spécifique dans lequel il opère. Actuellement le site abrite les deux entités STMicroelectronics et ST Ericsson. Leur organisation interne devient de plus en plus scindée en deux. Chacun disposant de son propre système d'administration : ressources humaines, directeur de site, communication générale.

La position géographique du site est stratégique. Grenoble est un grand centre d'électronique, situé à proximité de grandes villes européennes. De plus, son emplacement, comme le montre la figure 1, est au cœur du polygone scientifique, à côté de l'école d'ingénieur PHELMA[1], du CEA[2] et proche du pôle de nanotechnologie Minatec. Ce qui lui permet un accès facile aux moyens techniques puissants à disposition de Grenoble.

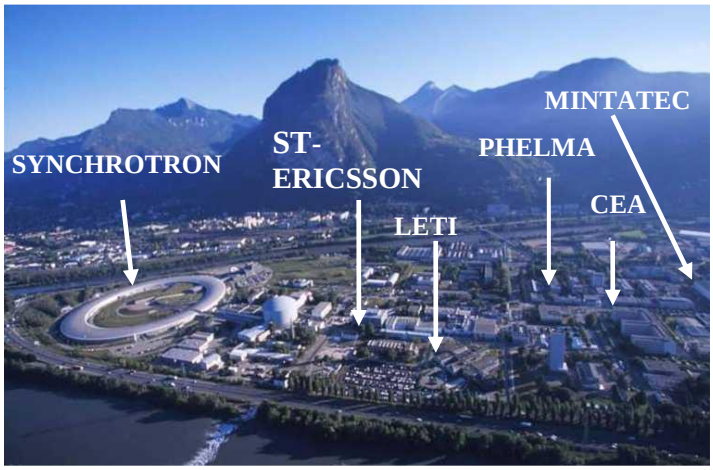


Figure 1: Situation stratégique du ST-Ericsson Grenoble

[1] PHELMA : Ecole d'ingénieur spécialisée en physique, électronique et matériaux.
[2] CEA : Commissariat à l'énergie atomique

Introduction

La conception de circuits intégrés consiste à réaliser les nombreuses étapes de développement et de validation nécessaires pour concevoir correctement et sans erreurs une puce électronique. Une forte concurrence incite les entreprises de semi-conducteur à accélérer la mise sur le marché de leur produit. Cela implique des délais de conception de plus en plus courts malgré une augmentation de la complexité des circuits. Tel est le leitmotiv des industriels de la micro et nanotechnologie depuis les années 2000. C'est pourquoi il est actuellement indispensable d'utiliser toutes les solutions possibles qui permettent de réduire ce temps de développement.

Une des solutions est d'exploiter la capacité des circuits programmables à émuler des montages numériques grâce à leur importante intégration. Ces dispositifs se programment rapidement et facilement. Ils permettent ainsi de vérifier en avance de phase le fonctionnement de montages numériques. Cela permet de corriger les défauts éventuels avant le lancement en fabrication des prototypes servant à la validation traditionnelle.

Ce stage s'est déroulé à ST-Ericsson, entreprise de développement de circuits électroniques à destination d'applications mobiles. Le travail demandé se situe à l'étape de validation des circuits. La présentation de ce contexte fera l'objet de la première partie. Cela amènera à définir l'environnement de test et l'objectif du stage. Le but est de vérifier le schéma du circuit avant son implémentation sur un circuit intégré à destination du client final. Ce qui explique l'utilisation d'une solution basée sur un FPGA.

Dans un premier temps la mission du stage concerne l'étude et l'évolution d'une solution FPGA existante. L'augmentation des besoins des concepteurs a amené à prévoir un circuit de plus grande intégration. Cette évolution a engendré plusieurs modifications au sein du projet. La présentation de ce travail fera l'objet de la seconde partie.

Le sujet de stage porte sur la conception d'une chaîne de traitement numérique pour des applications audio. Ce montage est composée de filtres servant à exploiter les signaux de sortie d'un convertisseur analogique-numérique. Afin de répondre au mieux à cette demande, une approche théorique est proposée dans une troisième partie. Elle est indispensable pour appréhender les comportements attendus lors de la vérification.

La quatrième partie concerne l'approche expérimentale de cette conception. Elle présentera les techniques utilisées pour mettre en œuvre la chaîne de traitement numérique. Ce montage sera alors implémenté dans le FPGA au cœur de l'environnement de test du convertisseur.

Ainsi, une dernière partie permettra d'analyser les résultats obtenus mais également de poser un avis critique sur la méthodologie utilisée. Une perspective d'évolution sera alors proposée.

1 LE CONTEXTE

1.1 L'environnement du stage

1.1.1 Présentation

[2]L'organisation de ST-Ericsson est divisée en trois « divisions produits ». Le stage s'inscrit dans l'une d'elle : la division 3GP pour « 3G Multimedia & Plateforms ». Le but de la division 3GP est de produire une plateforme[3] produit complète. Celle-ci doit intégrer leurs circuits intégrés et leurs solutions multimédia dans le but de créer une application embarquée (GPS, téléphone mobile, clé 3G). Cette plateforme, peut être représentée par un produit sans son boîtier, ni son écran et ni sa batterie.

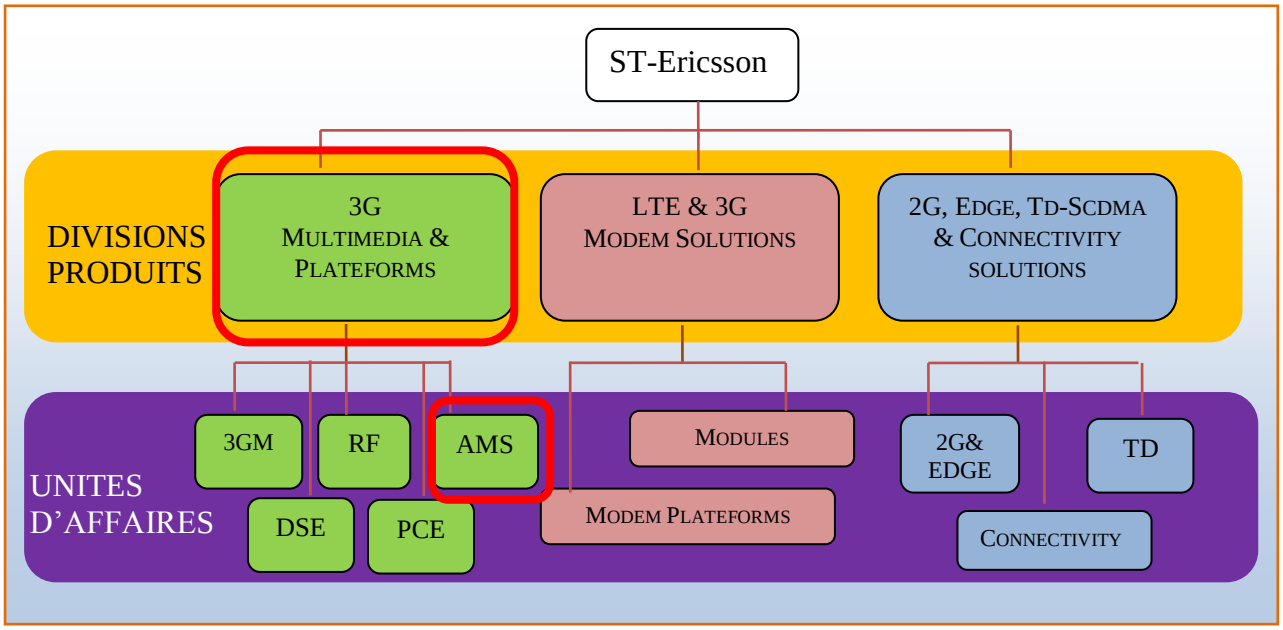


Figure 2 : Organisation de ST-Ericsson

La division 3GP est divisée en cinq unités d'affaires appelées Business Unit (BU) :

- **3G Multimedia** (3GM) se charge du développement des applications multimédia.
- **Radio Frequency** (RF) développe et fournit les solutions radio fréquence (traitement de signal émis et réceptionné au niveau de l'antenne d'un téléphone ou GPS).
- **Platform & Customer Engineering** (PCE) développe les architectures des plateformes.
- **Digital System-on-Chip Design & Engineering** (DSE) développe et fournit les circuits purement numériques pour la plateforme du produit.
- **Analog & Mixed Signal** (AMS): C'est dans cette unité que le stage s'est déroulé. Elle développe, produit et distribue les circuits analogiques et numériques pour les plateformes produits. Il y a trois sortes de circuits développés :

[3]Annexe 1 : Exemple de plateforme produit

- Les IP « Intellectual Property » sont des projets d'innovations. Les cellules IP sont les blocs de bases de tous circuits. Ils représentent une seule fonction. Ils intégreront les circuits clients une fois leur maturité arrivée à terme.
  - Les cellules de gestion d'énergie : ce sont des circuits d'alimentation linéaire comme les régulateurs LDO, ou d'alimentation à découplage comme les SMPS.
  - Les cellules Audio : il s'agit de convertisseurs analogique/numérique et numérique/analogique ainsi que des amplificateurs audio.
- Les circuits standards sont des produits qui sont proposés en l'état aux clients sous forme de catalogue. Ils sont appelés ASSP, Application Specific Standard Production.
- Les circuits spécifiques sont développés en réponse à une commande propre au client. Il n'est d'ailleurs pas possible de proposer ce circuit à d'autres clients. Ce sont les ASIC, pour Application Specific Integrated Circuit.

1.1.2 Le processus de conception d'un circuit

Chacune de ces BU développe les solutions de circuits intégrés. Plusieurs métiers sont nécessaires aux différentes étapes de conception du produit. Le but étant de mettre en production le circuit afin qu'il puisse intégrer les plateformes clients. La figure 3 présente le flot de conception d'un projet.

Figure 3 : Processus de conception d'un circuit.

Etude de marché : Tout commence soit par une étude de marché dans le cas d'anticipation des demandes soit par l'étude d'une demande en réponse à un appel d'offre du client. Ce sont les commerciaux qui cherchent les nouveaux projets en étudiant l'état du marché actuel et en suivant les stratégies de la compagnie. C'est au responsable de la division que revient la décision d'accepter ou non le projet. Pour cela il s'appuie sur une étude plus approfondie de la concordance avec les produits existants de l'entreprise.

Evaluation du concept : C'est la phase de l'étude de faisabilité du circuit. Le circuit est conceptualisé et les caractéristiques demandées sont étudiées en se basant sur ce qui existe. A partir de la, il va être possible de prédéterminer les ressources et les compétences nécessaires à la mise en œuvre du projet.

Spécifications et Architecture : Dans le cas de la faisabilité du projet, un cahier des charges est alors rédigé par le client pour des ASIC ou par ST Ericsson pour des ASSP. C'est durant cette phase d'architecture que les stratégies techniques sont validées pour mener à bien le projet. De plus, c'est à

ce moment que les ressources matérielles et humaines sont allouées. Une prévision temporelle et budgétaire est également présentée pendant cette phase.

Conception et implémentation : C'est pendant cette phase que les circuits sont conçus par les concepteurs puis validés par le laboratoire dans lequel s'est effectué le stage. Le circuit peut alors passer à l'étape de caractérisation. Cette étape dresse un rapport complet de toutes les caractéristiques du produit.

Qualification : Ce sont des tests de conformité effectués à grandes échelles pour répondre à une norme de qualité du produit. Il est par exemple demandé de vérifier la durée de vie d'un circuit.

Production : Le circuit atteint alors une maturité suffisante pour être envoyé en production. Cette étape clôture la phase de développement.

1.1.3 L'activité de validation

Nous venons de voir que la validation fait partie intégrante de l'étape de conception d'un circuit. Le graphe de la figure 4, présente l'organisation de ces activités.

La mission du laboratoire peut être scindée en deux parties : avant et après la réception des prototypes à valider (en violet sur le graphe suivant). Au moment de la réception des documents de spécification, la personne qui est en charge de ce nouveau projet étudie le fonctionnement du circuit et son domaine d'application. Cette étude spécifie les besoins en termes de solution de validation. C'est alors que débute la phase de conception de ces outils. Ces derniers doivent être fonctionnels à la réception des prototypes.

Commence donc la seconde phase de l'activité : la validation du circuit. Il s'agit de vérifier le fonctionnement des prototypes qui sont testés « physiquement » grâce aux solutions développées pendant la première phase. Tant que les résultats de ces tests ne sont pas satisfaisants, le concepteur devra corriger son schéma et relancer la production de nouvelles versions de prototypes.

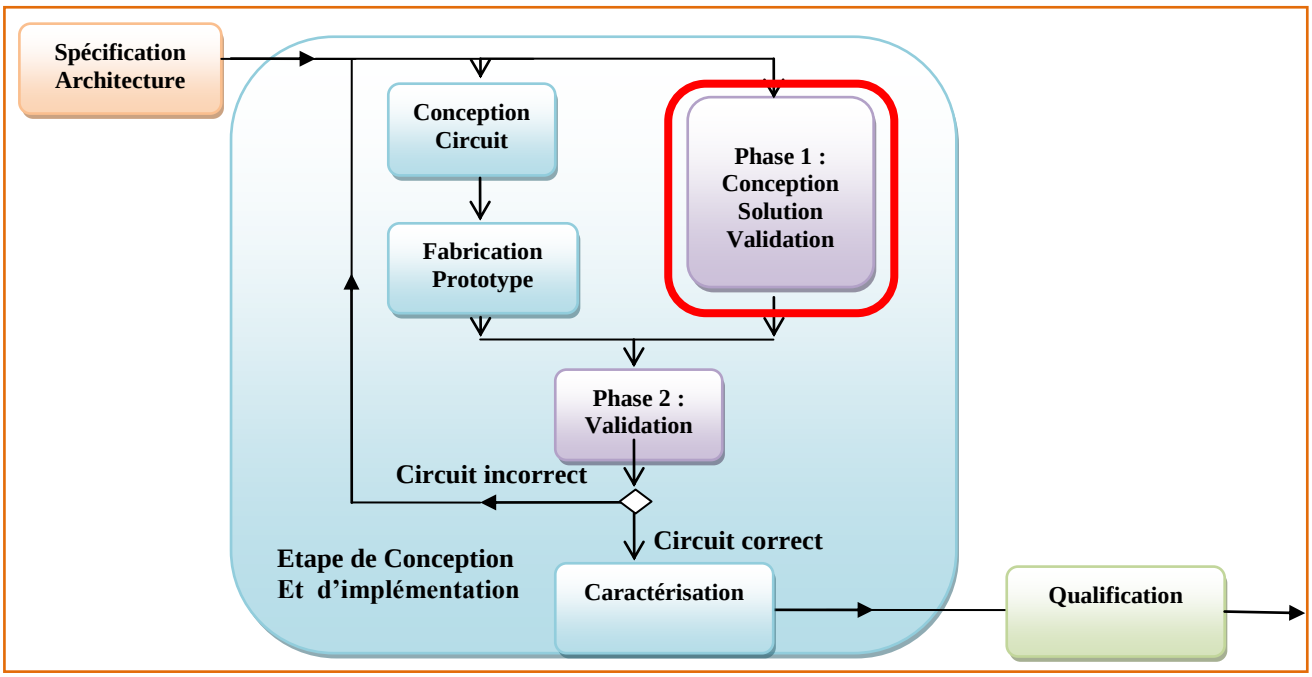


Figure 4 : Etape de conception

1.1.4 Le projet de validation existant

Le stage va se dérouler au sein d'un environnement de validation déjà existant. Cette partie se propose de présenter le travail de conception de cette solution qui a été conçues par les ingénieurs du laboratoire.

La solution globale

L'objectif de cette solution est de valider le fonctionnement des IP développées par la BU AMS. La figure 5 présente la solution globale : elle est composée d'un logiciel de pilotage, de la carte de développement d'un FPGA et d'une carte de validation. La présence du FPGA est nécessaire, car il fait le lien entre la solution logicielle et matérielle. Le but est quelles puissent se « comprendre » lors des échanges de données. Dans ce cas, le FPGA est utilisé en tant que microcontrôleur[4].Ce qui signifie qu'il interprète les commandes venant du logiciel et les renvoies au circuit situé sur la carte de test.

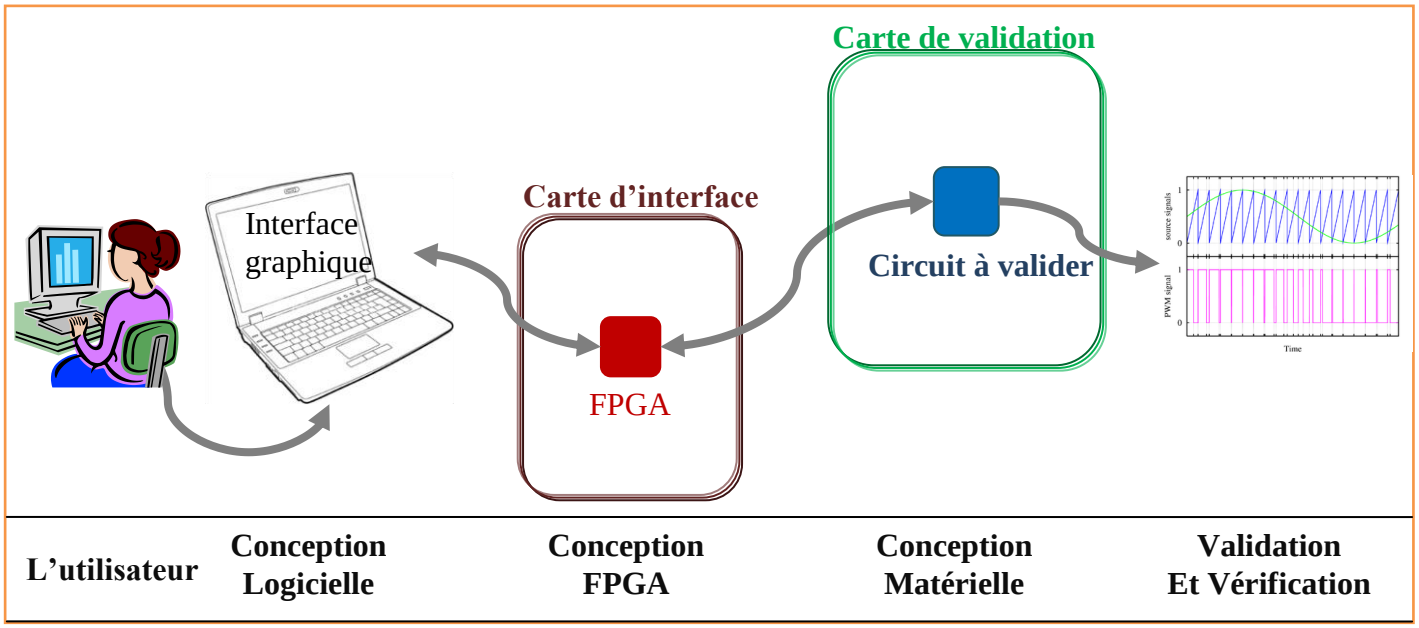


Figure 5 : Structure de la solution de validation

C'est la première fois que le laboratoire développe une solution générique. Ce qui signifie qu'elle doit s'adapter à tous les IP développées dans la BU AMS. Avant cela une solution complète était développée pour chaque IP. Cette stratégie amenait à répondre de manière très spécifique aux contraintes du produit à valider, mais cela revenait trop cher et prenait beaucoup de temps. Cependant, cette phase d'expertise a été indispensable car elle a permis une amélioration continue et très approfondie des solutions développées. Aujourd'hui, il en découle une maturité suffisante pour garantir la réalisation de développement générique et modulable suffisamment élaboré pour convenir à tous les IP.

[4]Microcontrôleur : Cf. Glossaire

La conception logicielle

Afin de tester le produit, l'utilisateur va s'aider d'un logiciel de contrôle qui a pour but de transmettre les commandes au circuit. L'intérêt est de sélectionner l'IP que l'on veut valider ou d'activer des modes de fonctionnements particuliers. La solution logicielle remplace les cavaliers manuels précédemment utilisés. La complexité du projet demande une expertise poussée de l'environnement. C'est pourquoi, il a été développé par un sous-traitant en collaboration du laboratoire de validation.

Cette solution se présente sous la forme d'une interface graphique en anglais GUI pour « Graphical User Interface », dont le but est de permettre à l'utilisateur de sélectionner ses instructions. Ces dernières sont représentées par des valeurs classées dans des registres à une adresse bien précise. Le Tableau 1 présente la table des registres proposée par l'interface graphique. L'utilisateur sélectionne un registre et lui alloue les instructions grâce aux boutons de sélections correspondant aux bits du registre.

Catégorie	Fonction	Protocole	Adresse du Registre	Valeur en hexadécimal du registre	Visualisation des bits correspondant aux instructions actives
ADC	RIGHT	I2C	0x1001	0x25	
AUDIO TX	CLOCK	SPI	0x1017	0x01	

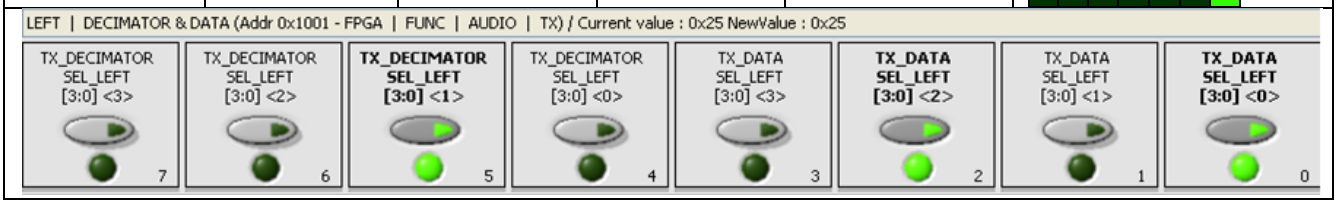


Tableau 1 : Table des registres du GUI

Pour une meilleure ergonomie, une interface customisée pour chaque circuit a été conçue. Elle ne comporte que les bits relatifs à la catégorie et à la fonction désirée. Des schémas du circuit sont retranscris pour rendre cette interface plus intuitive. La Figure 6 présente cette interface :

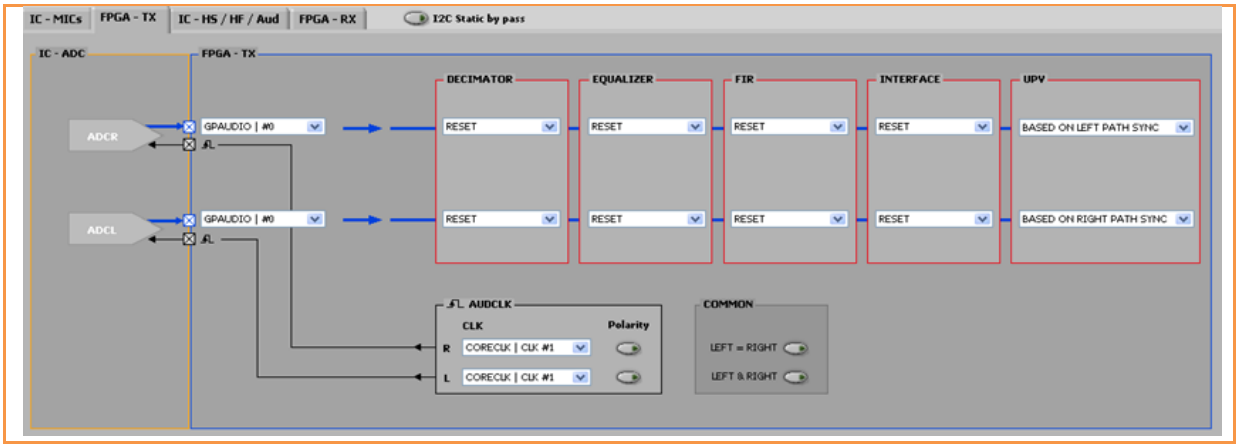
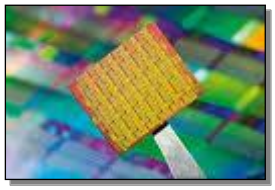


Figure 6 : Interface customisée de sélection de bits

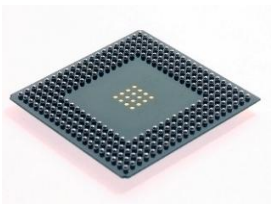
Cette solution est générique, elle est donc adaptable à plusieurs projets et à plusieurs protocoles de communication. Ce qui explique la présence d'un fichier de configuration externe. Celui-ci est chargé par le logiciel à son lancement. Il contient la table des registres et les protocoles associés. C'est dans ce fichier que sont spécifiés les noms des registres et des bits propres au projet, ces détails apparaissent ensuite sur l'interface graphique. Le développement effectué durant le stage a conduit à la modification de ces fichiers. Les filtres seront activés via cette interface. D'où l'intérêt de leur allouer un registre spécifique.

La conception matérielle

Une fois conçu et fabriqué, le circuit est placé dans un boîtier électronique servant d'interface entre la cellule et la carte de test grâce à une jonction électrique. Le boîtier permet la dissipation thermique et protège le circuit des « pollutions » extérieurs. Dans le cadre de ce projet, le choix s'est porté sur un boîtier BGA « Ball Grid Arrays » en replacement de boîtiers TQFP80, car il offre un nombre d'entrées/sorties plus important pour une surface plus réduite. Ce boîtier peut directement être soudé sur la carte mais, la validation implique un changement régulier de l'échantillon, celui-ci est alors déposé dans un réceptacle, appelé socket.


Le circuit


Le boîtier TQFP80


Le boîtier BGA


Le socket

Figure 7 : Présentation du circuit

Le développement des cartes de validation a pour but de relier les entrées/sorties du circuit aux bornes des « billes » du boîtier. Ce qui donne un accès extérieur aux montages du circuit permettant la connexion des instruments de mesures ou d'autres éléments de la solution grâce à des connecteurs. Le choix de ces derniers est fait en fonction du type de signal étudié et du câble les reliant à l'appareil. S'ajoute à cela des composants externes comme les bobines et les condensateurs servant à répondre rapidement aux fluctuations du courant et de la tension liées à la commutation des transistors du circuit. La conception de ces produits doit respecter des règles précises de placement et de routage évitant ainsi les interférences émises par les signaux entre eux comme la diaphonie[5].

[5] Diaphonie : C'est un bruit perturbateur d'un signal sur un autre. C'est souvent dû à un phénomène d'induction électromagnétique.

Le schéma : Le but de cette étape est de schématiser le montage avec les composants externes et les connecteurs utilisés. De nombreux points de test sont prévus pour visualiser un maximum de signaux.

Le placement-routage : Cette étape est effectuée par un sous-traitant. La complexité du projet demande une expertise importante quant à la prise en compte de la compatibilité électromagnétique[6] entre les éléments internes et externes de la carte. Une bonne maîtrise du placement-routage permet une intégration de tous ces éléments sur une surface limitée. Une fois routée, la carte peut alors être fabriquée et câblée.

La solution matérielle utilisée comporte plusieurs cartes, chacune ayant une fonction bien déterminée. L'intérêt est de simplifier l'implémentation de toutes les fonctionnalités nécessaires au bon fonctionnement du circuit. Cela permet également une grande modularité de la solution car elle dispose d'une carte proposant des éléments communs et une autre carte plus spécifiques. L'évolution des circuits n'impactera que la deuxième carte. Il en résulte tout naturellement des gains financiers et temporels intéressants. Ces cartes sont assemblées grâce à des connecteurs permettant de transférer les différents signaux nécessaires au fonctionnement du prototype testé.

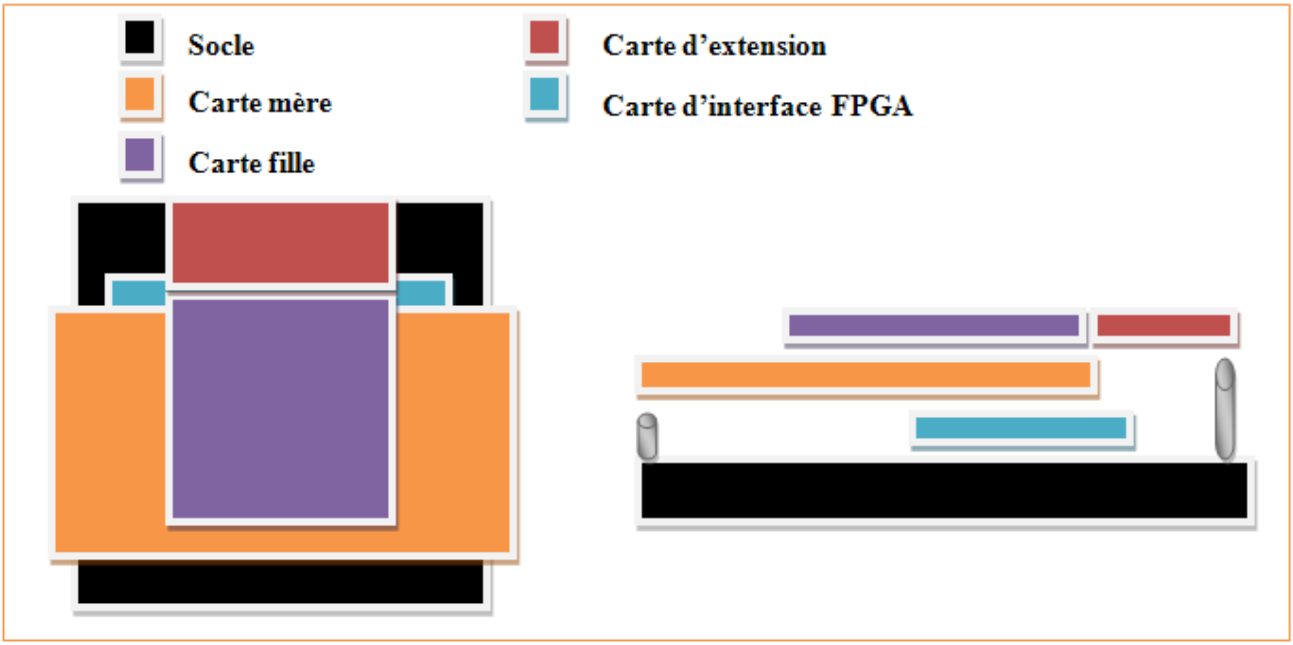


Figure 8 : Schéma de la solution existante

[6] Compatibilité électromagnétique : Aptitude d'un système à fonctionner dans un environnement électromagnétique satisfaisant sans émettre ni subir de perturbations.

1. **La carte mère** : C'est le cœur du système matériel. C'est elle qui gère et traite les signaux d'entrées et de sorties du système matériel. Elle permet de faire l'interface entre les différentes cartes et les instruments de mesure.

2. **La carte fille** : C'est elle qui est la plus spécifique. Elle contient le circuit à valider soudé ou avec socket. Elle permet de répondre aux nombreuses contraintes demandées par le concepteur. Elle ne dispose que de la gestion des signaux directe d'entrées et de sorties du circuit avec les composants externes placés au plus prés du circuit afin que leur rôle de découplage soit le plus efficace possible. Elle est reliée à la carte mère grâce à des connecteurs haut débit qui permettent de transférer très rapidement les signaux nécessaires au fonctionnement du circuit.

3. **La carte d'extension** : Elle comporte la gestion de tous les signaux communs aux différents circuits à valider. Elle se connecte directement sur la carte fille.

4. **La carte d'interface** : Il s'agit de la carte de développement FPGA. Les cartes d'interfaces sont des cartes qui interprètent les commandes entre le logiciel et le matériel. Pour cela, elles sont reliées au PC mais également à la carte mère. Grâce au FPGA utilisé, cette carte peut en plus d'interpréter des commandes, émuler des circuits pour les vérifier. C'est donc dans ce FPGA que les filtres sont implémentés.

1.2 L'objectif du stage

L'objectif du stage est de vérifier la conception et le fonctionnement d'un montage numérique. En temps normal le processus conduirai à concevoir une solution de validation pour accueillir ce circuit et à la réception du prototype le valider. Pourtant, aujourd'hui la demande a évolué. Les concepteurs souhaitent que le laboratoire apporte une vérification en avance de phase, c'est-à-dire avant la fabrication des prototypes, comme le montre la figure 9. L'idée est d'obtenir un circuit correct dès la première version.

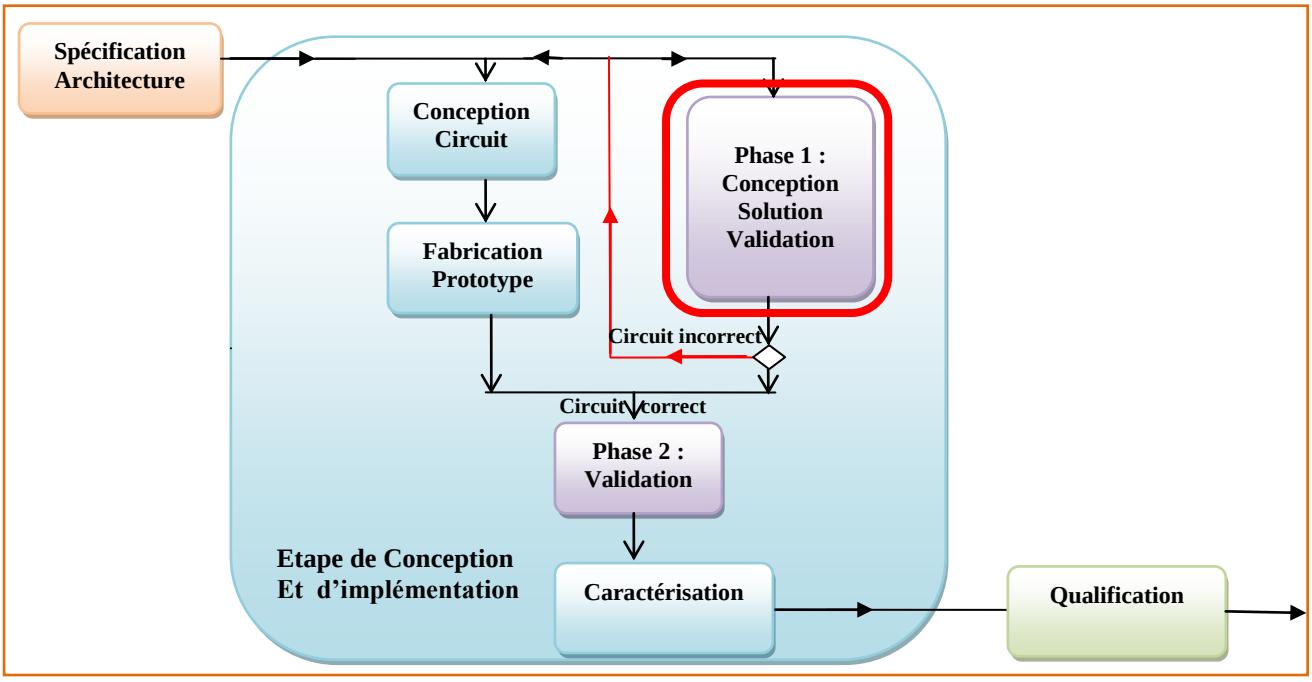


Figure 9 : Etape de conception optimisée

Le montage numérique à vérifier, permet de valider une IP audio conçue dans la BU AMS et déjà fabriquée. L'IP est un convertisseur analogique-numérique. Son signal est de sortie de fréquence de 4.8MHz. Cette fréquence trop élevée rend ce signal inexploitable par les instruments de mesures et les applications audio de téléphonies mobiles. Ils n'acceptent que des fréquences plus réduites.

Pour cela il est nécessaire d'ajouter une chaîne de traitement numérique en sortie du convertisseur afin de rendre exploitable ce signal. Cette chaîne est composée de trois filtres en cascades. Le stage consiste à concevoir le montage numérique de la chaîne pour le vérifier ce qui permettra au concepteur de valider son IP.

Cette vérification se fera sur le même environnement de test que le convertisseur. Cela implique qu'il faudra intégrer la nouvelle solution au sein même du projet de test existant.

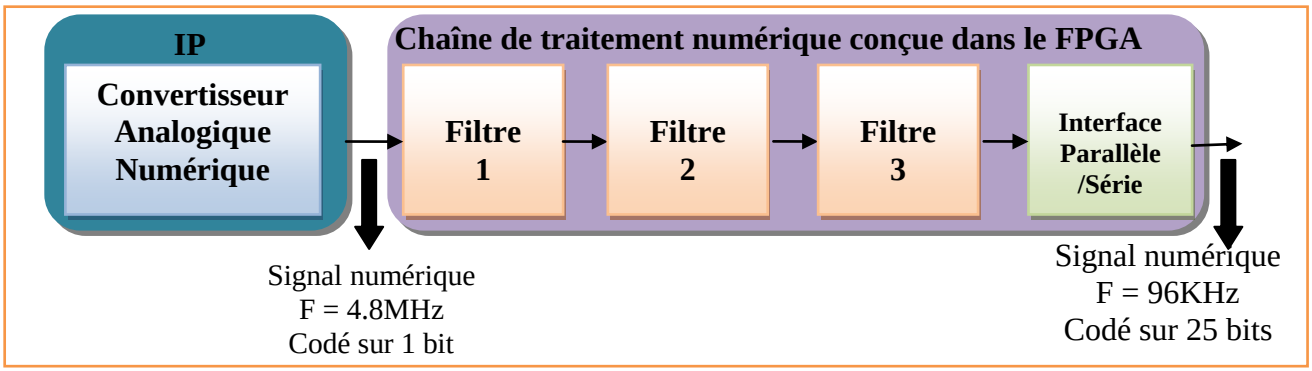


Figure 10 : Chaîne à implémenter

Ce développement repose sur l'utilisation du FPGA présent dans le projet présenté précédemment. Ce dispositif est utilisé d'une part en tant que microcontrôleur et d'autre part en tant qu'émulateur des montages numériques. Cependant, avec l'augmentation de la demande des concepteurs et de la complexité des systèmes, la solution existante va bientôt ne plus être suffisante en termes d'intégration. Par conséquent, le premier objectif du stage va être de faire évoluer cette solution, et de l'inclure dans le projet existant.

Le deuxième objectif du stage sera d'appréhender d'une part le domaine d'application audio, le convertisseur et la chaîne de traitement numérique d'autre part de connaitre l'environnement et le flot de conception d'un FPGA dans le but de concevoir les filtres numériques.

2 LA SOLUTION DE DEVELOPPEMENT

Le prototypage de circuit dans le FPGA a déjà été utilisé dans le projet existant. Avant de présenter cette solution, cette partie propose une introduction sur le FPGA.

2.1 Introduction sur le FPGA

[9][10] Les FPGA, sigle anglais qui signifie « Field Programmable Gates Arrays » traduit en français par réseau de portes programmables, sont des circuits intégrés reprogrammables. Ils offrent la possibilité de réaliser des fonctions numériques plus ou moins complexes, tout comme leurs homologues figés : les ASIC.

2.1.1 L'architecture

Structurés sous forme de matrices, les FPGA sont composés d'éléments logiques de base, constitués de portes logiques, présentes physiquement sur le circuit. Ces portes sont reliées par un ensemble d'interconnexions modifiables : d'où l'aspect programmable du circuit.

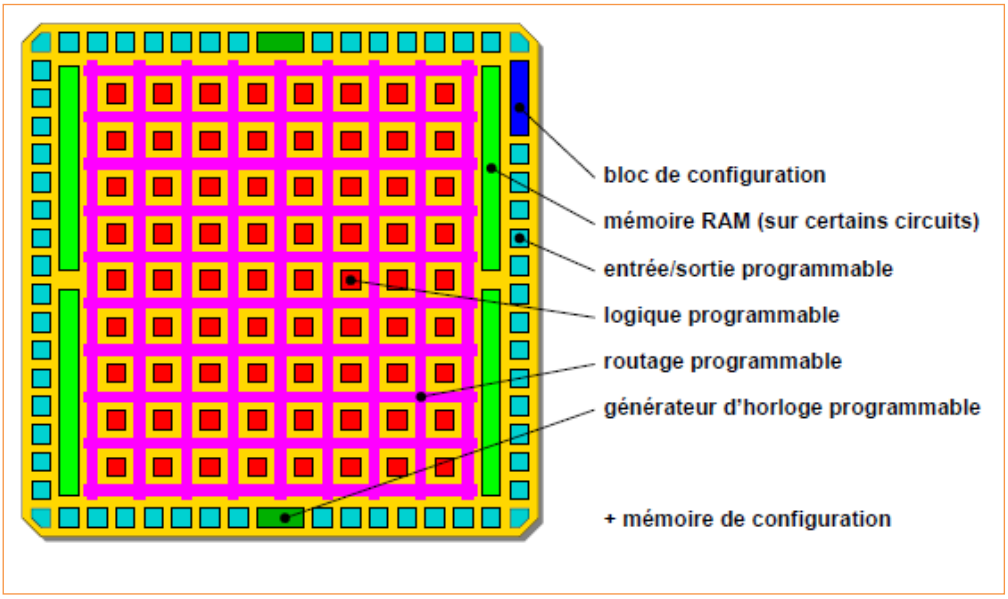


Figure 11 : Structure d'un circuit FPGA

La structure du FPGA présentée Figure 11 est composée :

- **De cellules d'entrées sorties modifiables** qui servent d'interfaces entre les broches du circuit et le cœur du FPGA pour adapter les signaux suivants :
  - Alimentation
  - Signaux d'horloge
  - Signaux de configuration du FPGA
  - Signaux de test

- **De blocs logiques ou éléments logiques** contenant les fonctions logiques combinatoires et séquentielles.
    - La partie combinatoire permet de réaliser des fonctions de complexité moyenne avec des portes classiques ET, OU et NON de deux à une dizaine d'entrées.
    - La partie séquentielle comporte une ou deux bascules généralement de type D. Compte tenu du nombre d'éléments logiques et de leur structure, leur association permet de réaliser tous les types de bascule. L'intérêt est de créer des mémoires élémentaires à un bit.

Suivant le fabricant du circuit, ces blocs contiennent un nombre différent de portes logiques et de bascules à l'intérieur d'un bloc comme le montre la Figure 12.

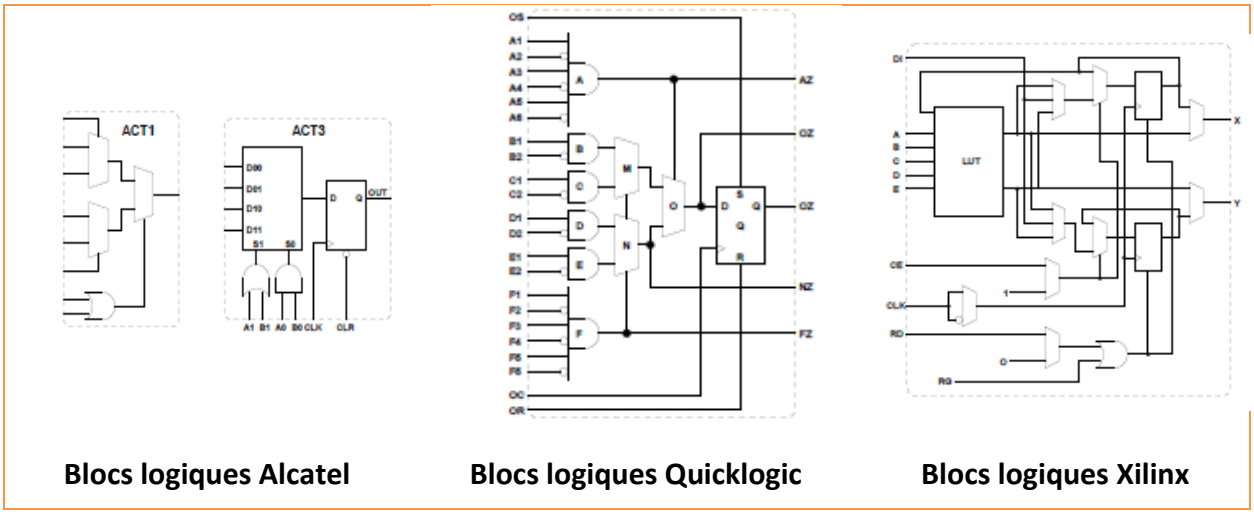


Figure 12 : Exemple de blocs logiques de différents fabricants

- **De réseaux d'interconnexions** que l'on voit en Figure 13. Ces réseaux relient entre eux les blocs logiques et les blocs d'entrées/sorties. Ces connections peuvent directement relier :
    - **Des éléments internes** dans un bloc grâce à un système de tables logiques appelées LUT. C'est une matrice de connections où les points de routage déterminent le niveau des entrées soit haut soit bas des portes logiques.
    - **Des éléments proches** : on parle de liaisons directes entre les blocs.
    - **Plusieurs blocs présents sur toute la surface** : on parle de liaisons à distance ou générales.

Certains de ses canaux sont spécifiques aux signaux d'horloges.

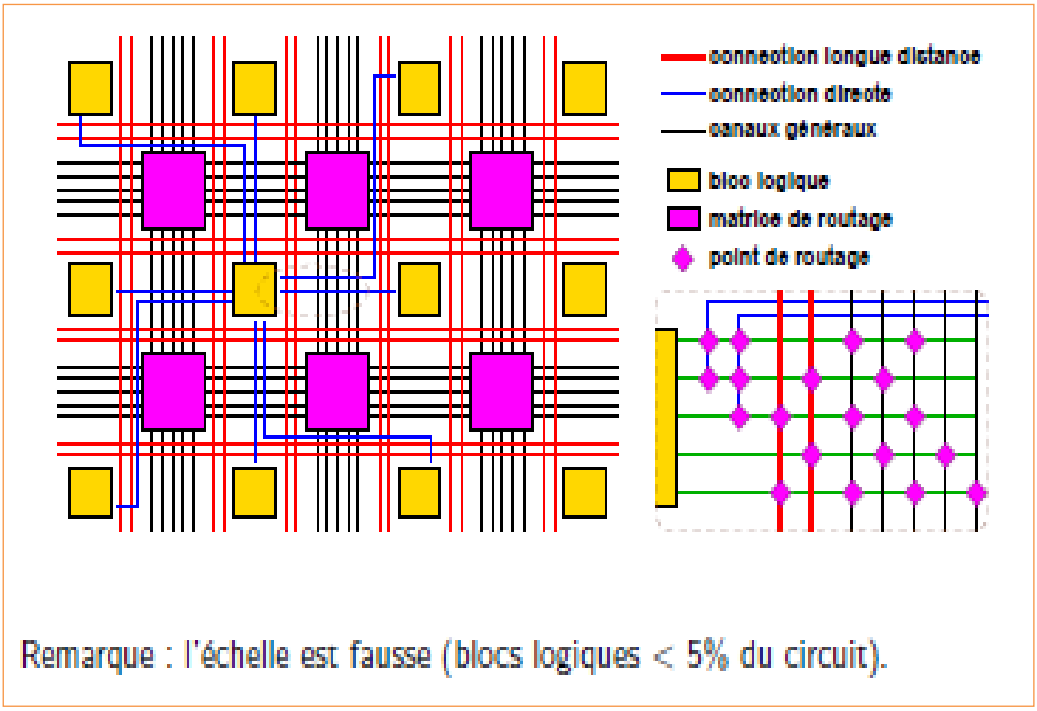


Figure 13 : Structure générale du routage

Un microprocesseur : La présence d'un processeur est indispensable pour ordonnancer les commandes reçues par le FPGA. C'est le chef d'orchestre de tout système informatique, où son rôle est de suivre des instructions qui lui ont été préalablement programmées en langage C.

Habituellement, ce processeur se trouve à l'extérieur du FPGA, mais des constructeurs ont intégré ces systèmes directement dans le FPGA. Il s'agit de processeur « soft-core » (soft pour logiciel et core pour cœur d'exécution) on parle aussi de système sur puce programmable (SOPC[7]). Il communique avec le FPGA grâce au langage de description matérielle VHDL. Ce processeur est donc reconfigurable pouvant ainsi d'adapter aux contraintes de chaque utilisation.

[7] SOPC : System On Programmable Chip

2.1.2 Les avantages du FPGA

[11] Ces notions sur le FPGA donnent des indications quant à l'intérêt de son utilisation dans le cadre du projet. Voici les avantages clés qui ont fait que le laboratoire s'est tourné vers cette solution.

- **Un circuit reprogrammable** : L'avantage du FPGA est de pouvoir être reprogrammable contrairement aux circuits intégrés de type ASIC. Ce qui rend cette solution modulable et donne la possibilité de modifier le programme générique de base afin de le rendre spécifique au circuit utilisé. Une solution de validation utilisant le FPGA peut alors convenir à beaucoup de projets et donc diffusée à plusieurs équipes.

- **Un investissement rentable** dans la durée : Cela est dû à sa reprogrammation, ce qui implique une réutilisation à destination d'autres projets, malgré un prix à l'achat supérieur à un circuit ASIC.

- **Une Reprogrammation quasi-instantanée** du circuit. Une fois le programme validé cela ne prend que quelques minutes à l'implémenter. A titre de comparaison, la fabrication d'un circuit ASIC peut prendre plusieurs semaines.

Cependant, le FPGA n'est pas le seul composant reprogrammable du marché. Le DSP « Digital Signal Processor », processeur de signal numérique, permet également d'émuler un montage numérique. Le DSP est programmable grâce au langage C, le FPGA utilise quant à lui le VHDL. Le format de description machine est généré automatiquement par les logiciels de développement des concepteurs. Il est possible alors de vérifier les circuits sans avoir à les concevoir par nous-mêmes. L'importation de leur fichier dans notre programme est suffisante. Par ailleurs, le FPGA peut disposer d'un DSP sous forme d'IP incluse dans le système du circuit.

2.2 La solution existante

[10] Cela fait déjà plusieurs années que le laboratoire a fait le choix de travailler avec des FPGA. Actuellement, la solution existante est le FPGA Cyclone II, de la série Cyclone d'Altera[8]. Cette série propose des FPGA à un prix raisonnable, offrant un bon compromis entre une bonne performance et un niveau d'intégration moyen.

Pour pouvoir être programmé, le FPGA a besoin d'une carte de développement. C'est un environnement de test et de conception. Elle est présentée figure 14. C'est une carte qui a été conçue en interne par un autre laboratoire de validation de l'entité AMS située à Prague dont voici sa composition :

- **Une connexion USB** qui lui permet de communiquer avec le PC afin de recevoir des instructions grâce à un logiciel de pilotage.
- **Des éléments de test** qui comme le logiciel, permettent d'interagir avec le circuit. Le but étant de tester si le FPGA répond correctement à des instructions simples. C'est idéal pour commencer à déboguer un montage. Ces interactions se font à l'aide de commandes envoyées par exemple par l'actionnement d'un bouton poussoir. Le FPGA peut renvoyer une information sous la forme d'une séquence d'allumage de LED[9] .
- **Un connecteur pour la programmation** présent pour brancher l'interface de programmation.
- **Une mémoire Flash** externe qui reprogramme de façon autonome le FPGA.
- **Des connecteurs** pour la connexion d'une horloge externe.
- **Des connecteurs** pour la relier à d'autres cartes de test.
- **Le circuit FPGA**.

Tout cet environnement implique la présence d'une alimentation générale de la carte de 5V.

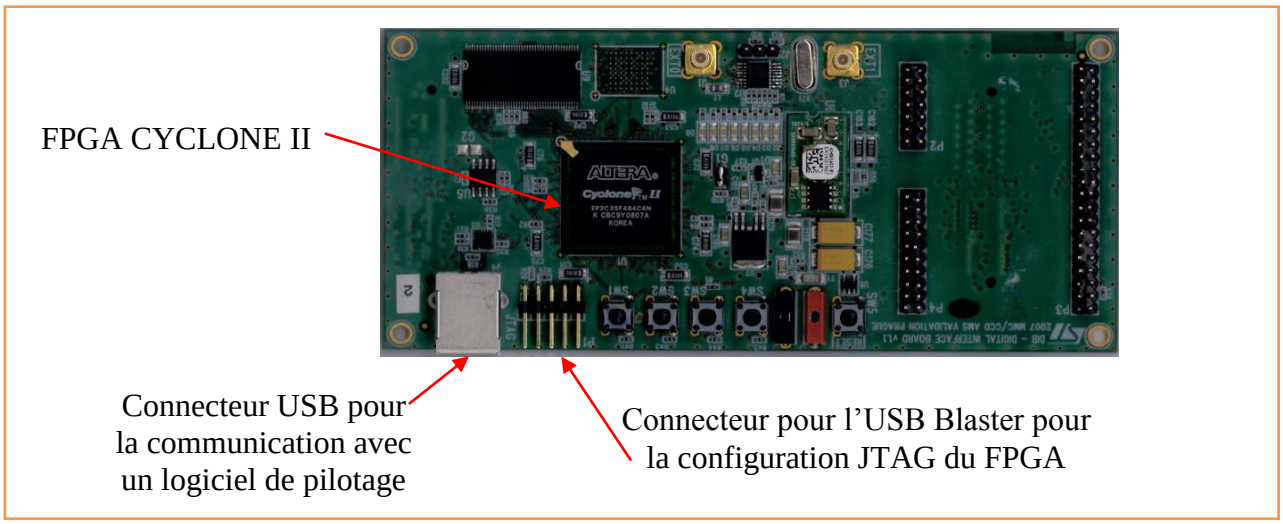


Figure 14 : Présentation de la carte du cyclone

[8] Altera : Fabricant et fournisseur de circuits programmables FPGA, d'ASIC et d'IP.
[9] DEL : Diode électroluminescente, LED en anglais. C'est un composant électronique où son allumage est conditionné par le passage ou non d'un courant.

2.3 Le besoin d'évolution

[10] Le besoin de prototypage de circuit est de plus en plus présent au laboratoire. Ce qui nécessite une capacité d'intégration toujours plus grande des circuits programmables. Cette complexité est augmentée par le double rôle que doit remplir le FPGA.

Les FPGA de la série Cyclone commencent à présenter des limites de performances liées à un nombre trop faible d'éléments logiques qu'ils contiennent. Ainsi ils ne pourront pas prétendre suivre cette évolution très longtemps. Par conséquent, afin d'anticiper les besoins futurs, la recherche d'un FPGA de plus haute performance et de plus haute intégration a été indispensable.

C'est pourquoi, le laboratoire a fait le choix de remplacer le circuit existant. Il s'est tourné vers une autre série de FPGA toujours proposée par Altera. Il s'agit de la série Stratix et du FPGA Stratix III. C'est la série d'Altera qui propose la plus grande capacité d'intégration et des performances très élevées.

L'évaluation de leur caractéristique est basée sur le nombre d'éléments logiques présents dans le circuit. Ce chiffre découle directement de la technologie utilisée. Par ailleurs, la performance est elle aussi un critère important. Elle est définie par le nombre de multiplieurs embarqués. Enfin le dernier critère notable est la taille de la mémoire interne, car c'est qui elle permet d'inclure un DSP et un processeur ainsi que les montages numériques conçus.

Le Tableau 2 montre la comparaison des caractéristiques entre les deux solutions. La Stratix III de part sa haute intégration due à la diminution de la taille des transistors, offre des performances plus élevées que la série cyclone.

	Cyclone II	Stratix III
Technologies	90nm	65nm
Nombre d'éléments logiques	68 416	200 000
Mémoire embarquée	0.2Mbits	10Mbits
Nombre de multiplieurs embarqués	150	576

Tableau 2 : Tableau de comparaison des caractéristiques des deux séries

La modification du FPGA a impliqué le changement de l'environnement de test. La carte du Cyclone n'est pas adaptable à ce nouveau circuit. Altera propose son propre kit de développement qui comprend la carte de test, le circuit FPGA et les logiciels de programmation.

La communication avec le PC ne se fait plus via le protocole réseau USB mais, via le protocole Ethernet. Ces communications sont gérées par le processeur présent dans la FPGA. Sa programmation doit par conséquent subir des modifications. La figure 15 présente ce nouvel environnement.

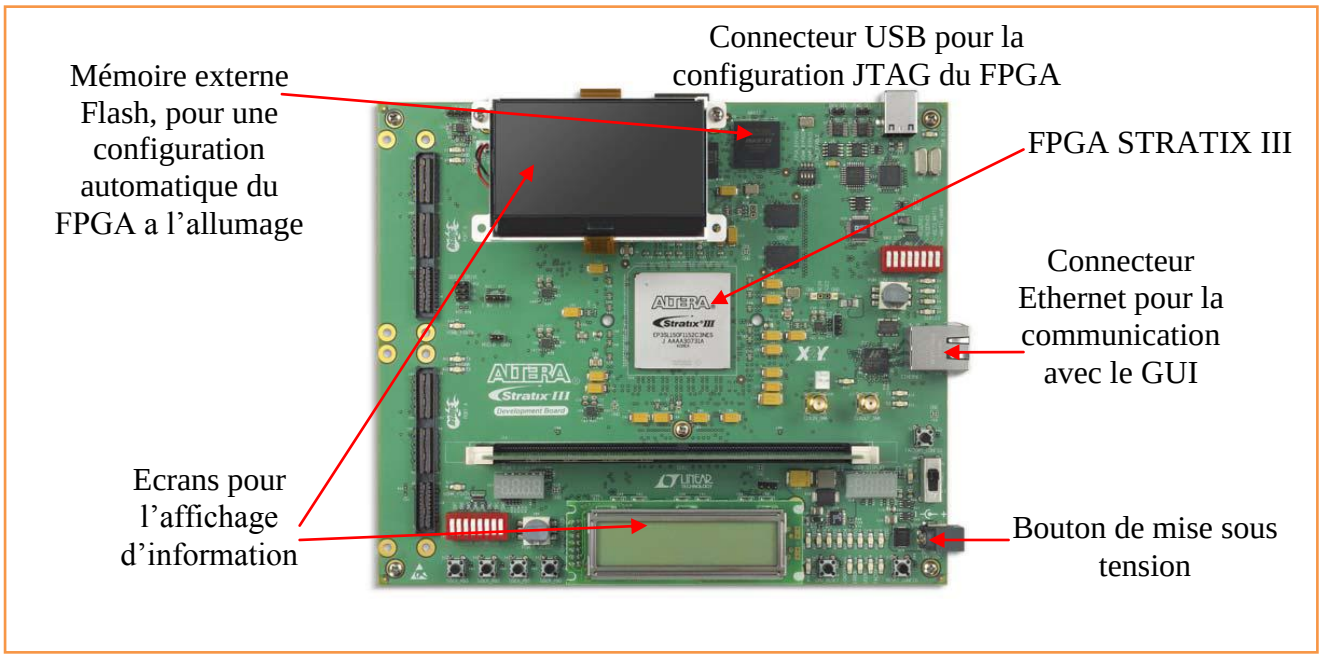


Figure 15 : Environnement de test du FPGA Stratix III

2.4 La mise en œuvre de l'évolution

[10] Le processeur embarqué dans les FPGA Cyclone et Stratix s'appelle le NIOS. C'est un produit spécifiquement créée par Altera. Le NIOS du Cyclone a déjà été programmé pour d'autres applications. Le but de cette partie est d'adapter cette solution au FPGA Stratix.

2.4.1 Le NIOS II

C'est la deuxième génération de micro-processeur d'Altera. Il est passé de 16bits à 32 bits. Ce système contient un CPU[10], mais également des mémoires (RAM, Flash), des DSP et des ports d'entrées/sorties utilisés pour contrôler les périphériques présentes sur la carte ou à l'intérieur du FPGA. Il est basé sur l'architecture matérielle RISC[11], ce qui signifie que son jeu d'instructions est réduit. Il est opposé au microprocesseur CISC[12], où son jeu d'instruction est complexe, c'est-à-dire qu'il comporte l'ensemble des instructions.

Le principal avantage de ce processeur est son cœur logiciel lui permettant d'être configurable. Ses paramètres sont ajustables. Son jeu d'instruction peut alors répondre à des besoins d'applications spécifiques. La modification de la solution profite de cette modularité.

2.4.2 L'environnement de programmation

Deux étapes sont nécessaires à la programmation du NIOS. La première partie concerne la conception du système avec le processeur embarqué NIOS II :

- Construction de la plateforme matérielle
- Choix des périphériques matériels et des interconnexions.
- Intégration de ces périphériques

Cela se fait grâce au logiciel Quartus. Cet ensemble d'outils est développé par Altera. Il fourni plusieurs environnement de conception dont l'outil de conception matérielle : SOPC Builder.

La deuxième étape concerne le jeu d'instruction programmé au niveau logiciel du NIOS en langage C. Altera a prévu un environnement de développement appelé Eclipse qui permet cette programmation, ainsi que la compilation et la simulation du programme.

[10] CPU : Central Processing Unit, c'est l'unité centrale de traitement.
[11] RISC : reduced instruction-set computer
[12] CISC : Complex Instruction-set compuer

2.4.3 Les modifications apportées

La phase de construction matérielle :

La construction matérielle consiste à récupérer des briques de fonctions qui sont utilisées pour l'application. Ces briques sont aussi appelées IP. Elles peuvent représenter un CPU, une PLL[13], des mémoires, des leds ou des boutons poussoirs disponibles sur la carte de développement.

Chaque IP est synchronisée sur les horloges de la PLL ou du CPU. Enfin, une zone d'adressage du registre leur est allouée, ainsi qu'un numéro d'interruption. Ces IP sont disponibles dans une librairie proposée ou non par Altera.

Une fois les IP récupérées et paramétrées, le logiciel Quartus synthétise cette construction et la transforme en un fichier VHDL. Ce fichier est implémentable dans tous projets FPGA.

[13] PLL : phase lock loop : boucle à vérouillage de phase, utilisé en tant qu'asservissement de fréquence de sortie sur un multiple de la fréquence d'entrée.

1) Etude de la construction du NIOS Cyclone existant:

La Figure 41 16 présente l'interface graphique de l'outil SOPC Builder montrant les IP choisies pour le FPGA cyclone.

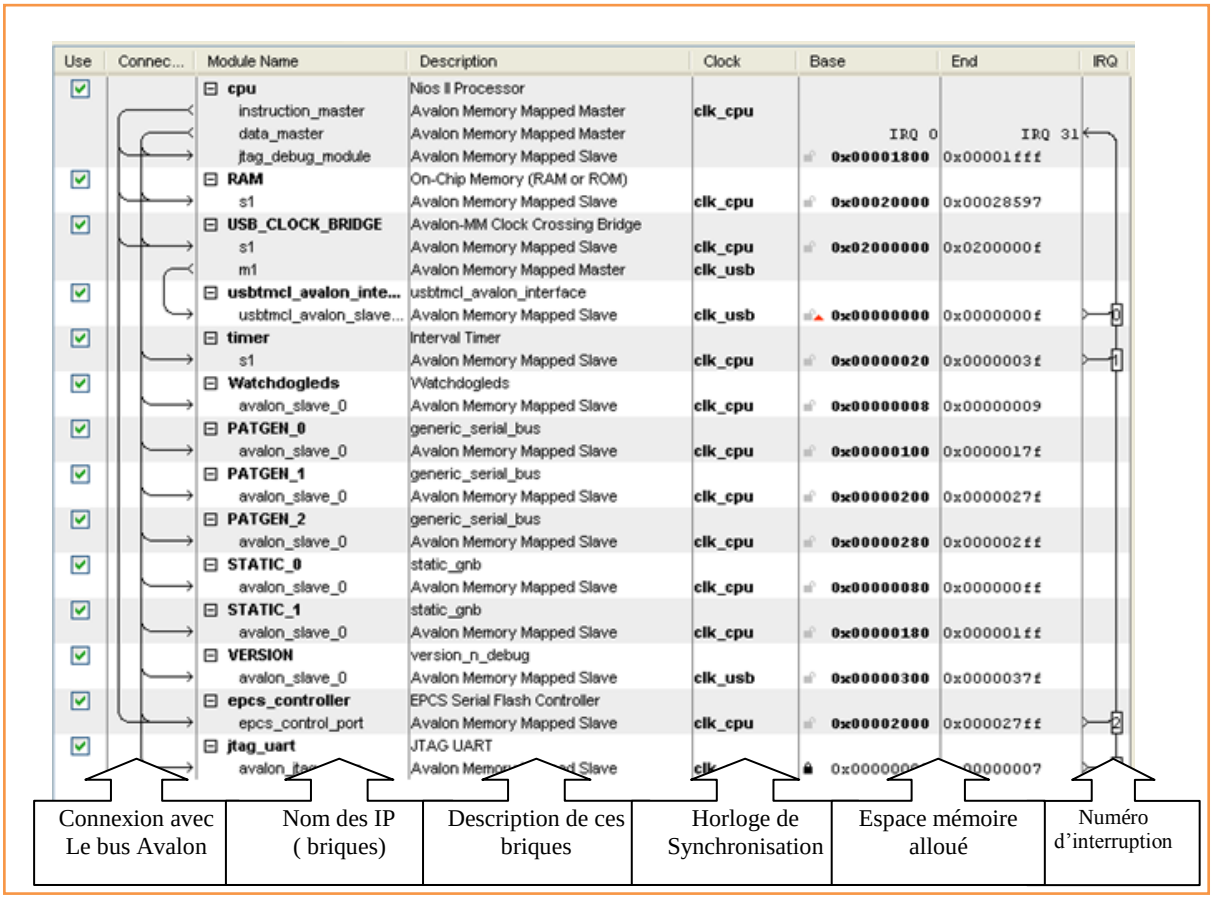


Figure 16: Composition du Nios du cyclone

Les briques de fonctionnement de base sont:

CPU : C'est l'unité centrale, le NIOS II. C'est elle qui exécute les instructions.

RAM: C'est la mémoire du système qui gère les données dans les registres. Elle contient les exécutables et les variables des futurs programmes.

TIMER : C'est l'horloge interne du micro-processeur qui gère les constantes de temps, c'est elle qui donne le séquencement des tâches. Contrairement au FPGA qui effectue des tâches en parallèle, le processeur fonctionne de manière séquentielle. Cette complémentarité donne au FPGA une grande flexibilité et une grande rapidité d'exécution.

USB_CLOCK_BRIDGE : C'est la gestion de la communication USB avec le PC qui comporte les instructions envoyées par un logiciel de pilotage.

EPCS_CONTROLEURS : Ce contrôleur permet d'accéder directement à la programmation de la mémoire Flash.

JTAG_UART : C'est ce qui permet au logiciel de programmation Eclipse de communiquer avec le processeur via le port JTAG. Il remplace les liaisons séries communément utilisées.

Les PIO : « ports inputs/outputs» ports d'entrées/sorties. C'est ce qui correspond aux briques de gestion des périphériques.

PATGEN_x : Ce sont des signaux dynamiques construits à partir des « empreintes » de protocoles de communication (I2C, SPI). Ces empreintes sont simplement la mise en forme des données synchronisées sur une horloge dans le temps. Les commandes sont envoyées au circuit à valider sous ce format prédéterminé.

STATIC_x: Contrairement aux précédents, il s'agit de signaux statiques, qui n'évoluent pas dans le temps. Ils sont de valeur « 1 » ou « 0 ». Ces signaux sont à destination des éléments de test de la carte comme les leds ou les boutons poussoirs.

WATCHDOGLEDS : C'est une séquence d'allumage des leds qui permet de contrôler le bon fonctionnement du système. Ces leds sont situées sur la carte de développement.

VERSION : C'est dans ce bloc qu'est contenue l'information donnant la version du programme en C, du NIOS logiciel. Cette information apparaît sur l'interface graphique du logiciel de pilotage. Elle apparaît également sous forme de séquence d'allumage des leds sur chacune des cartes FPGA.

Il est possible de changer les adresses ou les numéros d'interruptions manuellement et directement sur la représentation mémoire du système. Bien que sur ce système, le CPU est l'unique maître des périphériques, il est possible d'observer les connexions entre les IP connectées au bus Avalon, et d'optimiser celles-ci dans le cas où tous les maîtres ne doivent pas accéder à chaque esclave.

2) *Adaptation pour la construction du NIOS Stratix:*

1. La modification concerne la brique de la gestion de communication avec le PC. Le protocole USB va donc être remplacé par le protocole Ethernet. Pour cela, une IP à été conçue spécifiquement par la société ALSE partenaire de Altera.
   Cette IP s'appelle : **ETHERNET_GEDEK** , pour « Gigabit Data Exchange Kit ».
   C'est elle qui remplace l'IP du cyclone : **USB_CLOCK_BRIDGE.**

2. Une autre IP a été ajoutée, il s'agit d'une brique de gestion des périphériques. La carte de développement du Stratix est pourvue d'un écran LCD. J'ai profité de la modification du système pour utiliser ce périphérique. Cet écran permettra d'afficher des informations à destination de l'utilisateur. Il rend cette solution plus ergonomique.
   Cette IP s'appelle **lcd_Display.**

La phase de développement logiciel :

L'environnement de programmation (IDE[14]) Eclipse permet la saisie des instructions du processeur NIOS en langage C. Ces commandes sont ensuite traduites en langage machine au processeur via la phase de compilation.

La Figure 17 présente le schéma du flot de communication. Un utilisateur communique via une interface graphique pour envoyer des commandes au format ASCII[15] vers le FPGA. Un Bridges joue le rôle d'interface entre le logiciel de pilotage et la solution matérielle. Il permet de détecter le protocole de communication (USB ou Ethernet) afin de savoir sous quel format doit être envoyées les commandes. Le NIOS interprète ces commandes et l'envoie au destinataire. Il gère également les informations retours à destination du logiciel de pilotage toujours en passant par le Bridge. Le programme en C concerne l'exécution des commandes reçues par le NIOS.

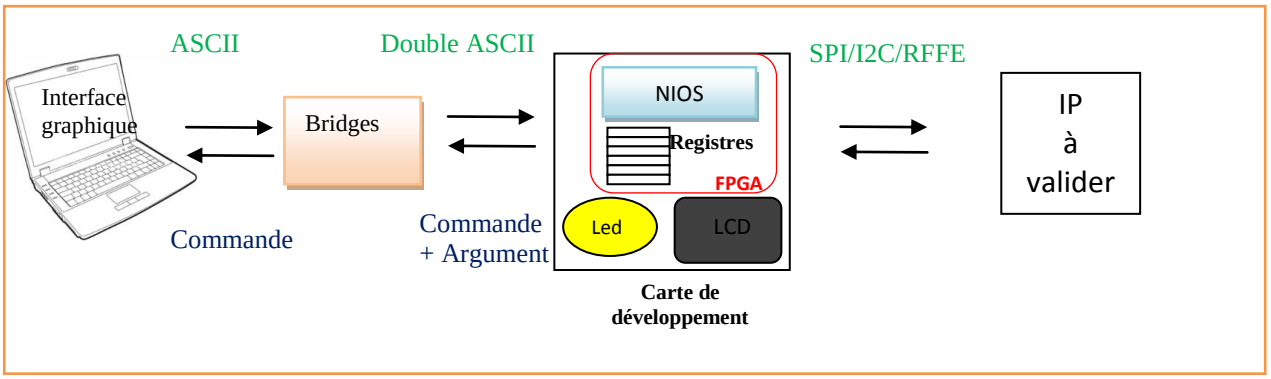


Figure 17 : Principe d'interprétation des commandes

Voici les modifications apportées :

1 Déclaration des librairies appelée :

Le programme fait appel à des fonctions qui sont fournis par ces librairies. Ces dernières peuvent être

- Soit standard au langage C.

```
#include <stdio.h>
#include <unistd.h>
#include <io.h>
#include <string.h>
```

- Soit spécifique aux projets :

```
#include "ALSE_Gedek_ST_v1.00/gedek_api.h"
```

Toutes les IP présentes dans la construction matérielle sont listées dans un fichier appelé system.h, celui-ci est généré par SOPC builder. Il fait partie des librairies appelées par le programme NIOS logiciel.

[14] IDE : Integrated Development Environment
[15] ASCII : American Standard Code for Information Interchange : C'est une norme de codage de caractère en informatique.

Modification n°1 : L'IP Gedek qui a été ajoutée lors de la construction matérielle, a été fournie avec des librairies spécifiques à la communication Ethernet. Elle remplace ainsi les librairies propres à la communication USB.

2 Déclaration des variables :

Modification n°2 : La commande reçue par le NIOS via le Bridge est une des variables du programme. Elle a nécessité une modification, car son nom faisait référence à la communication USB. Cela pouvait porter à confusion. L'idée est de donner un nom plus générique. Ce qui a impliqué une modification du nom de cette variable tout au long du programme.

usbtmcl_dev.rx_buf; a été transformé en ***message***

Modification n°3 : Plusieurs variables sont utilisées uniquement lors de la communication en USB, elles ont été remplacées par les variables propres à l'Ethernet.

3 Déclaration des fonctions :

Ces fonctions convertissent le format ASCII des commandes venant du Bridge en une instruction en hexadécimal comprise par les périphériques. Ce qui permet de faire la correspondance avec le registre cible.

Modification n°4 : Ces fonctions font appel à la variable correspondant à la commande reçue (modification n°1). Il faut donc modifier le nom de la variable.
Exemple pour la fonction "get_command_hexvalue" du cyclone :

```
int get_command_hexvalue( void ){
    char local_string[9];
    int  return_value;

    local_string[0] = usbtmcl_dev.rx_buf[2];
    local_string[1] = usbtmcl_dev.rx_buf[3];
    local_string[2] = usbtmcl_dev.rx_buf[4];
    local_string[3] = usbtmcl_dev.rx_buf[5];
    local_string[4] = usbtmcl_dev.rx_buf[6];
    local_string[5] = usbtmcl_dev.rx_buf[7];
    local_string[6] = usbtmcl_dev.rx_buf[8];
    local_string[7] = usbtmcl_dev.rx_buf[9];
    local_string[8] = 0;
    return_value = convert_8ascii_to_hex(local_string);
    return return_value;

  }
```

Modification dans la solution Stratix:

```
int get_command_hexvalue( void ){
    char local_string[9];
    int  return_value;

    local_string[0] = message[2];
    local_string[1] = message[3];
    local_string[2] = message[4];
    local_string[3] = message[5];
    local_string[4] = message[6];
    local_string[5] = message[7];
    local_string[6] = message[8];
    local_string[7] = message[9];
    local_string[8] = 0;
    return_value = convert_8ascii_to_hex(local_string);
    return return_value;}
```

Modification n°5 : La fonction « send Answer » permet de renvoyer une réponse d'acquittement « bonne réception de la commande ». Ce qui forcément fait appel au mode de communication ; d'où la modification du protocole USB en Ethernet. Pour cela, il est utilisé une fonction inclue dans librairie Gedek fourni par ALSE : ***Gedek_PutString***

Version Cyclone

```
void Send_Answer( char * ans_buff){
    // Warning !! Supposed here that argument is
    // a correct String, and OK to fit in TX buffer
    // because no test is performed on string to transmit
    int size;
    size = strlen(ans_buff);
    usbtmcl_dev.tx_buf[0] = rec_head[0];  // Header for NI Protocol
    usbtmcl_dev.tx_buf[1] = rec_head[1];  // based on last request from NI driver
    usbtmcl_dev.tx_buf[2] = rec_head[2];
    usbtmcl_dev.tx_buf[3] = rec_head[3];
    usbtmcl_dev.tx_buf[4] = (char) ((size+1) & 0x0FF); // This is the size of
payload (index 12->??)
    usbtmcl_dev.tx_buf[5] = 0; // Don't know the meaning of index 5 -> 11
    usbtmcl_dev.tx_buf[6] = 0; // ...
    usbtmcl_dev.tx_buf[7] = 0;
    usbtmcl_dev.tx_buf[8] = 1; // this one must be 1
    usbtmcl_dev.tx_buf[9] = 0;
    usbtmcl_dev.tx_buf[10] = 0;
    usbtmcl_dev.tx_buf[11] = 0; // end of unknown contents
    strcpy(usbtmcl_dev.tx_buf+12, ans_buff);
    usbtmcl_dev.tx_buf[12+size] = '\n';
    usbtmcl_dev.tx_in = 13+size;
  }

  void Send_Ack( void ){
    Send_Answer(STD_ACK);
  }
```

Version Stratix

```
void Send_Answer( char * ans_buff){
    // Warning !! Supposed here that argument is
    // a correct String, and OK to fit in TX buffer
    // because no test is performed on string to transmit
    int size;
    size = strlen(ans_buff);
    Gedek_PutString(ETHERNET_GEDEK_BASE, ans_buff, size);
  }

  void Send_Ack( void ){
    Send_Answer(STD_ACK);
  }
```

4 Le programme principal :

Ce programme commence par initialiser l'horloge présente dans le NIOS : « Timer ». Puis il commence une boucle infinie où se trouvent toutes les instructions qui sont exécutées par le NIOS.

Modification n°6 : Pour effectuer une instruction le NIOS doit s'assurer qu'elle est bien arrivée. Il s'agit la encore d'une partie spécifique à chacune des solutions, car la commande arrive sous un protocole différent. Les fonctions Gedek remplace donc les fonctions USB tout au long de ces instructions.

Modification n°7 : Il s'agit d'utiliser l'écran LCD de la carte Stratix. Dans la construction matérielle, on a définit cet écran comme une IP périphérique. Elle est définie dans le fichier system.h.

```
/*
 * lcd_display configuration
 *
 */

#define LCD_DISPLAY_NAME "/dev/lcd_display"
#define LCD_DISPLAY_TYPE "altera_avalon_lcd_16207"
#define LCD_DISPLAY_BASE 0x00001520
#define LCD_DISPLAY_SPAN 16

#define ALT_MODULE_CLASS_lcd_display altera_avalon_lcd_16207
```

Ainsi il est possible d'afficher une information sur l'écran LCD en ajoutant dans le programme principal :

```
        printf("   AMS IP BGA \n");
        printf("STRATIX SOLUTION\n");
```

Après quelques tests de fonctionnement du NIOS (allumage de leds, apparition du message sur l'écran LCD), le programme est maintenant prêt à être intégré au projet complet qui sera envoyé dans le FPGA. Mais l'arrivé de cette nouvelle solution n'a pas été sans conséquence dans la construction du projet, son intégration a entrainé quelques modifications.

2.5 Conséquence sur la structure existante

Pour comprendre la structure du programme qui est implémenté dans le FPGA, voici une description des fonctions du FPGA.

2.5.1 Les rôles du FPGA

La communication digitale

Le FPGA fait l'interface entre le circuit à valider et l'interface graphique. Ce logiciel utilise le langage VISA[16] pour envoyer les instructions. Cette programmation est dédiée normalement à la configuration et à la communication avec les instruments de mesure.

Pour communiquer, le circuit utilise quatre signaux : un signal d'horloge permettant de séquencer l'envoi des données, un signal de données d'entrées et de sorties et un signal d'adressage. La solution actuelle propose deux protocoles de communication entre le GUI et le circuit.

1. Le premier mode utilise la communication I²C, pour « Inter Integrated Circuit ». C'est le nom d'un bus qui comporte 3 signaux, un signal de données séries (SDA), un signal d'horloge (SCL) et un signal de référence, la masse.

2. Le deuxième mode utilise la communication SPI, pour « Serial Peripheral Interface ». C'est également un bus de données séries. La communication est full duplex. Le bus SPI comporte 4 signaux : un signal d'horloge, deux signaux de données, l'un généré par le maître et l'autre généré par l'esclave et un signal de sélection d'esclave.

Le prototypage de circuit

C'est le cas des filtres conçus durant ce stage. Ce n'est pas la première fois que le laboratoire propose cette vérification en avance de phase. D'autres concepteurs ont commencé à bénéficier des avantages de cette solution. Le FPGA s'impose de plus en plus en tant que solution de validation. C'est pourquoi le projet est déjà structuré pour accueillir ces montages de prototypage. La Figure 18 présente le projet d'accueil des montages émulés.

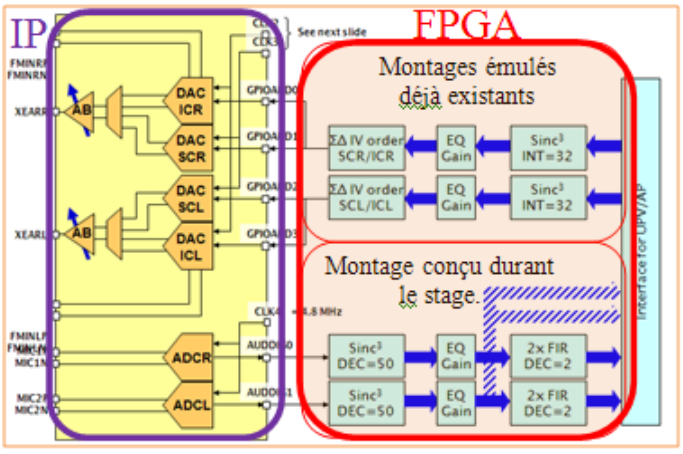


Figure 18 : Schéma du projet d'accueil du FPGA

[16] VISA : Virtual Instrument Software Architecture

2.5.2 Présentation du programme FPGA

Les commandes sont envoyées au FPGA grâce à l'interface graphique du PC. Celui-ci est connecté au port USB de la carte Cyclone. Cependant, la carte Stratix dispose d'un port Ethernet. La partie précédente à montrer que cette différence a amené à la création d'un nouveau processeur NIOS spécifique au Stratix. Cette modification de solution impacte également la construction du programme interne au FPGA. La méthode sélectionnée pour gérer cette nouvelle solution a été de créer deux projets spécifiques aux FPGA. Le schéma de la Figure 19 présente la nouvelle structure du programme.

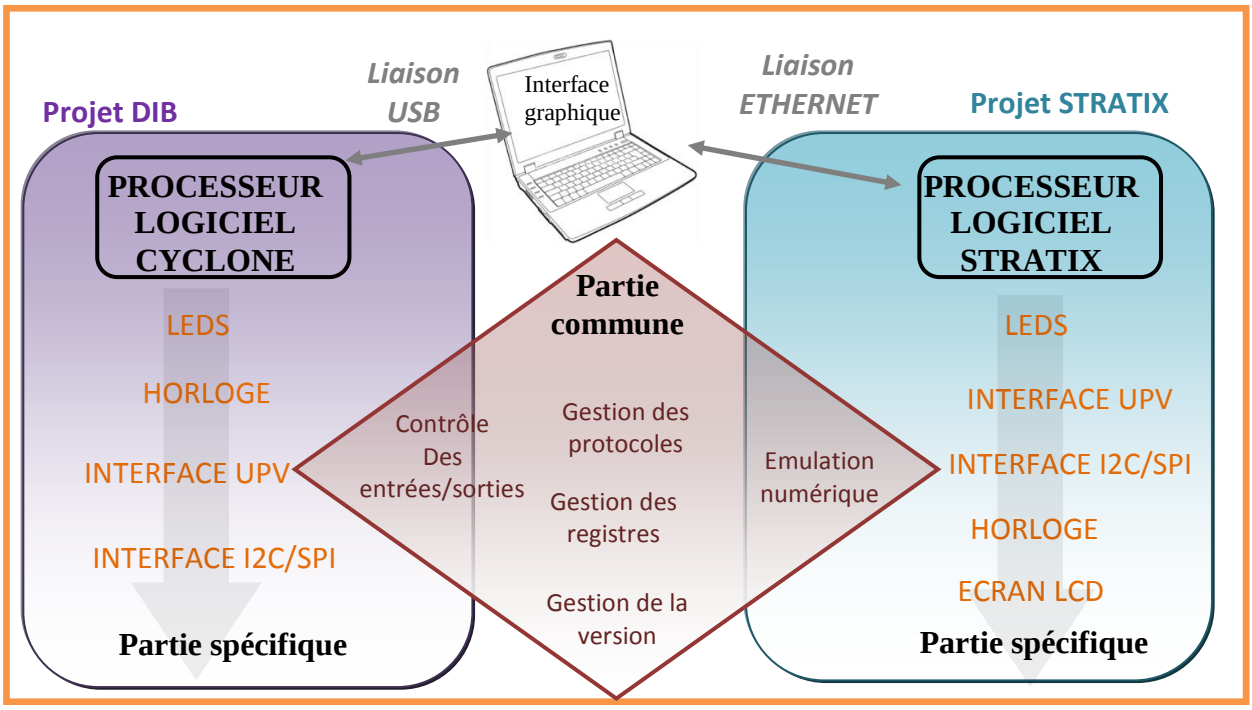


Figure 19 : Structure du projet

Cette méthode est basée sur une construction Cœur du système / Interface spécifique. Cela permet une portabilité d'une partie commune aux deux projets. Cette partie est désolidarisée de tout aspect matériel de la solution. L'avantage c'est que ça la rend générique, car elle sera capable de s'adapter à d'autres solutions de FPGA.

Partie Spécifique

Cette la partie propre au FPGA et à sa carte de développement. Elle peut-être assimilée au programme principal. C'est elle qui fait appel au « sous-programme » processeur et au « sous-programme » commun. La programmation de la partie spécifique se saisie graphiquement en créant des schémas blocs grâce au logiciel proposé par Quartus.

Son rôle est d'interfacer entre les signaux d'entrées ou de sorties qui arrivent sur chacune des pattes du FPGA en direction de ces fonctions internes. Le but est de les redistribuer vers le cœur du système ou vers le processeur. C'est aussi par cette partie du programme que les signaux « retours » sont redirigés à destination du circuit ou de l'instrument de mesure et que transitent les signaux entre le cœur du système et le processeur, comme le montre la Figure 20.

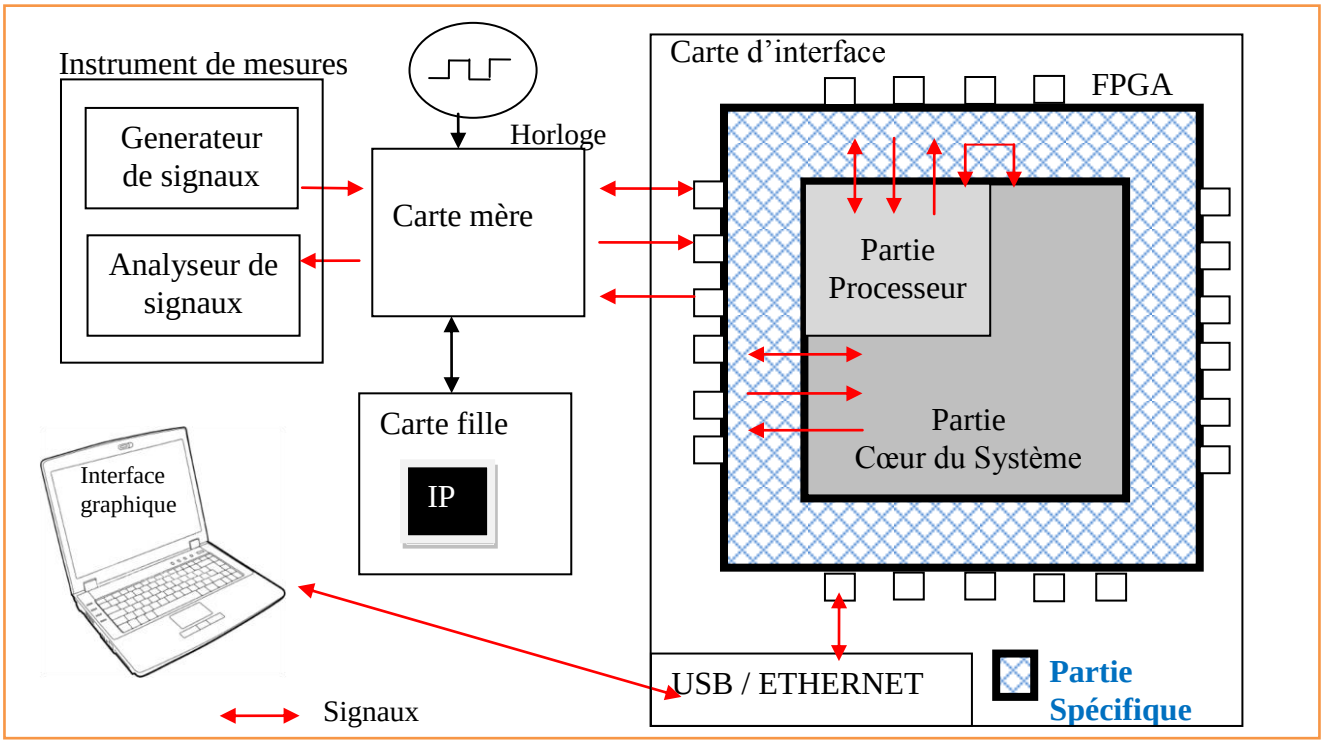


Figure 20: Transfert de tous les signaux par la partie spécifique.

Partie Commune

C'est la partie qui est assimilée au cœur du système. C'est elle qui contient les montages qui ont été développés pour valider les IP conçues dans la BU AMS. C'est bien évidemment à cet endroit que les filtres du convertisseur à valider vont s'implémenter. Cette partie est commune aux deux FPGA. Comme la précédente, elle a été structurée grâce au logiciel Quartus. Elle ne reçoit que les données envoyées par la partie projet et les lui renvoies une fois traitées. Cette partie comporte donc tous les blocs qui constituent les tâches que le FPGA doit effectuer. Chacune d'elle est représentée par un symbole qui comporte les conceptions des montages numériques, sous format VHDL ou sous format de saisie graphique. Chaque nouvelle fonction sera donc intégrée dans cette partie. Aujourd'hui elle comporte:

- **La gestion de la version du programme** : un fichier VHDL est à mettre à jour à chaque modification. Cette information est envoyée à l'interface graphique, afin d'en informer l'utilisateur. Cette gestion est indispensable pour conserver toutes les modifications et donne la possibilité de revenir en arrière.

- **La gestion des horloges** : elle est nécessaire aux différentes fonctions conçues dans cette partie. Il y a par exemple les signaux d'horloge de la PLL ou les signaux d'horloge externe. Chacune d'elle subit des divisions ou multiplications afin d'obtenir un panel de valeur.

- **La gestion des données qui sont reçues ou envoyées** : Elle inclue la construction et l'éclatement de bus de données à destination des connecteurs de la carte mère. Ce qui donne un accès extérieur aux signaux internes. La sélection des signaux accessibles se fait grâce à l'interface graphique du logiciel de pilotage.

- **La gestion des bus de données et d'horloge à destination de la partie processeur logiciel**. Le traitement de ces bus est contrôlé par le processeur.

- **La création de bus de « 0 » et de « 1 »** pour compléter les bus de données de taille standardisée sans en impacter l'information.

- **Les montages numériques de prototypage** simulant le fonctionnement des circuits. La conception de la chaîne de traitement demandée pour le stage est intégrée dans cette partie. Elle bénéficie des signaux mis à sa disposition, et renvoie les signaux une fois traitées.

Partie Processeur logiciel

La partie processeur logiciel a été présentée dans la partie précédente, lors de l'évolution de la solution. C'est bien évidemment une partie spécifique à chaque projet, Cyclone et Stratix. Son rôle est d'interpréter des commandes et de les renvoyer aux destinataires comme le circuit à valider par exemple. Il remplace ainsi la présence d'un microcontrôleur externe.

2.6 Bilan de l'évolution de la solution

Cette étude a permis de connaître les caractéristiques d'un FPGA. Ce qui a amené à constater les limites de performance de la solution existante. C'est pourquoi, le changement du FPGA et de sa carte de développement a été préférable pour les besoins futurs. Cette évolution a impliqué l'adaptation du processeur interne du FPGA et l'évolution de l'environnement de test. Elle dispose alors des deux solutions FPGA. En plus, sa construction lui donne la possibilité d'intégrer de futures solutions. Le premier objectif du stage est atteint, la solution est maintenant disposée à recevoir les prochains montages de prototypage comme la chaîne de traitement numérique.

3 APPROCHE THEORIQUE

Avant de commencer le développement des filtres, il est indispensable de se familiariser avec le domaine d'application. Cette partie propose une étude de cet environnement particulier et du circuit à valider. La connaissance des spécificités du convertisseur vont expliquer pourquoi il est indispensable de recourir au traitement numérique de ses signaux de sortie.

3.1 Le domaine Audio

[4] La fonction première d'un téléphone portable est la transmission à distance d'un signal analogique, notre voix, par réseau sans fil. Ce signal analogique est converti en un signal numérique pour être adapté au média de transmission. A la réception, ce signal est de nouveau converti en un signal analogique. La Figure 21 présente la chaîne de transmission complète. Les filtres conçus durant le stage se situe entre le convertisseur analogique-numérique et la transmission.

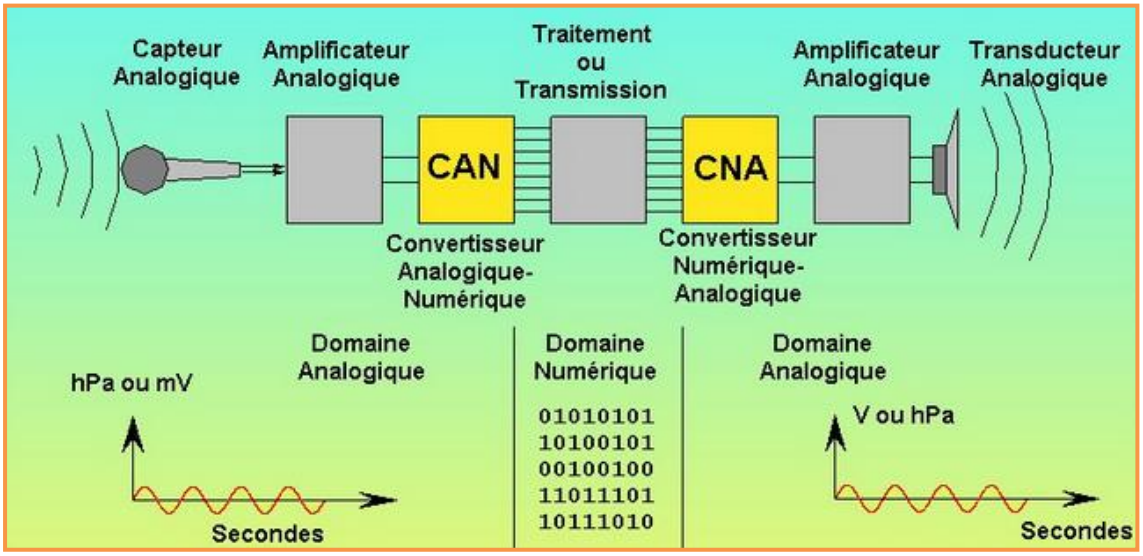


Figure 21 : Chaîne de traitement numérique

Ces fonctionnalités sont accomplies grâce à un microphone qui capte les signaux sonores analogiques. Ces signaux sont alors dirigés vers la chaîne de traitements analogiques-numériques permettant le transfert de l'information. Les signaux sont de nouveau traités mais du numérique à l'analogique pour être envoyés aux haut-parleurs du téléphone. Le passage du signal numérique au signal analogique et inversement est effectué grâce aux :

- CNA : Convertisseur Numérique-Analogique ou DAC en anglais pour « Digital to Analog Converter. »
- CAN : Convertisseur Analogique- Numérique ou ADC en anglais pour « Analog to Digital Converter. »

On parle du domaine audio lorsque l'application utilise la bande de fréquence de 20Hz à 20KHz. Cela correspond aux fréquences audibles de l'oreille humaine moyenne[17]. En dessous les sons sont qualifiés d'infrasons[18] au-delà les sons sont qualifiés d'ultrasons. Le fonctionnement de circuits audio est garanti pour cette plage de fréquences. C'est le cas du convertisseur étudié dont voici les caractéristiques.

3.2 Le convertisseur analogique-numérique

3.2.1 Le principe de conversion

[5][6] Le convertisseur analogique-numérique permet de passer du monde « réel » analogique au monde « discret » numérique. Cette opération consiste à transformer un signal analogique représenté par la variation de pression du son dans le cas du domaine audio, en un mot numérique codé sur un certain nombre de bits qui pourront être traités. C'est ce qu'on appelle l'échantillonnage[19] (Figure 22). L'une de ses valeurs caractéristiques est le LSB « Low Significative Bit » soit le plus petit pas de tension permettant le passage d'un code au suivant. Plus le pas de tension est petit, plus il y a de niveaux de tensions différents plus la résolution est importante.

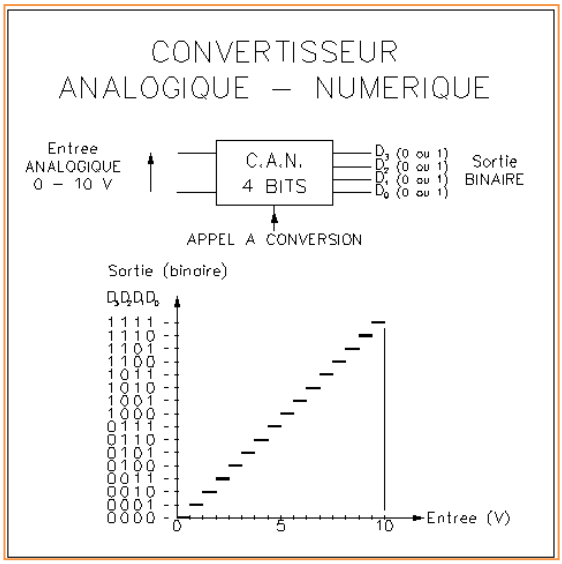


Figure 22 : Principe de conversion analogique-numérique

3.2.2 Le bruit de quantification

[5][6] La quantification ou l'échantillonnage est l'opération qui permet de passer d'un signal analogique, de nature continue dans le temps en un signal numérique, de nature discontinue dans le temps. Cet échantillonnage entraîne une perte d'information car tous les points ne sont pas représentés dans le signal numérique. Il génère également de la distorsion se traduisant par l'apparition d'harmoniques et par l'augmentation de l'amplitude du plancher de bruit dans la représentation spectrale du signal. C'est ce qu'on appelle le phénomène de bruit de quantification. La quantité d'états déterminés de possibilités définit la résolution du convertisseur exprimé en nombre de bits soit 2^N avec N le nombre de bits. Une grande résolution réduit le bruit de quantification.

[17] Annexe 2 : Perception du bruit de l'oreille humaine.
[18] Annexe 3 : Les Infrasons et les ultrasons
[19] Echantillonnage : cf. glossaire

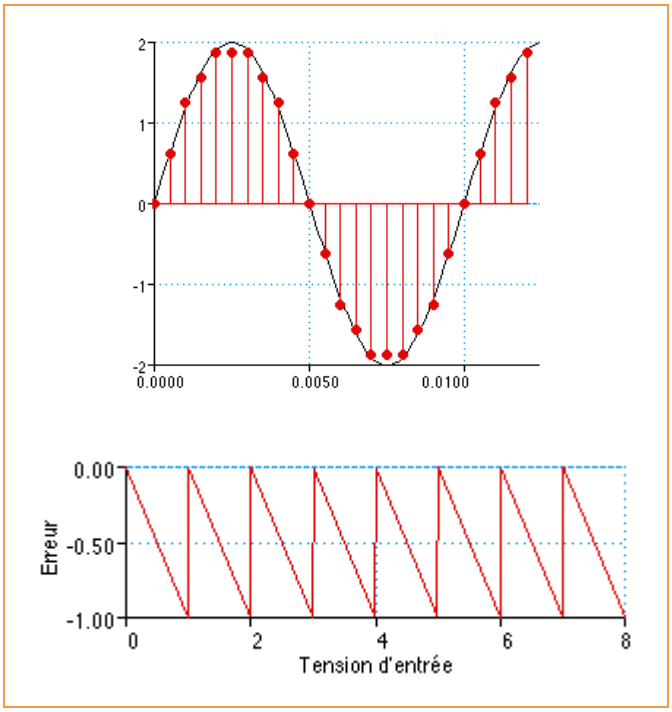


Figure 23 : L'échantillonnage et l'erreur de quantification

La Figure 23 représente l'échantillonnage qui montre bien la perte d'information due au passage du signal continu en noir à un signal discret en rouge. En dessous, est présentée la caractéristique du bruit de quantification. Elle est définit par des dents de scie correspondant à la différence entre le signal analogique d'origine et le signal quantifié (en rouge).

- Valeur analogique > Valeur numérique : l'erreur = +1/2
- Valeur analogique < Valeur numérique : l'erreur = -1/2
- Valeur analogique = Valeur numérique : l'erreur = 0

3.2.3 L'évaluation

[5][6] Le bruit de quantification se caractérise par la densité spectrale de la puissance du bruit[20], que l'on notera Pb. L'idée est d'évaluer son impact sur le signal de sortie. Pour cela la méthode est de comparer sa puissance d'origine, noté Ps, par rapport à la puissance de bruit de quantification. C'est ce qu'on appelle le rapport signal à bruit, en anglais SNR pour « Signal to Noise Ratio ».

$$(1) \quad \text{SNR en dB} = 10 \log (Ps/Pb)$$

Un ADC idéal présente un SNR théorique qui est défini par le rapport entre le signal d'entrée en pleine échelle[21] en anglais FS « full scale » et l'amplitude du bruit que l'on retrouve en sortie du résultat de la quantification. Le rapport signal à bruit idéal est le résultat de l'équation :

$$(2) \quad \text{SNR en dB} = 6.02\ N + 1.76\text{dB}$$

N étant le nombre de bits du convertisseur

[5][6] Ce rapport signal à bruit est pris comme référence lors de la validation des convertisseurs. Le but étant bien évidemment de s'en rapprocher le plus possible. L'augmentation du nombre de bits améliore la résolution du convertisseur ainsi que le rapport signal à bruit grâce à la diminution de l'erreur de quantification. On déduit de cette formule qu'un bit supplémentaire rajoute +6dB au SNR.

[20] La densité spectrale du bruit se définit comme la densité de bruit présente dans un signal, rapporté à une bande passante de 1Hz.
[21] Pleine échelle : C'est la valeur maximale du signal d'entrée qui garantit une bonne restitution du signal en sortie. Au-delà, le signal de sortie peut saturer.

3.2.4 Le convertisseur ADC Sigma-Delta

Un choix d'architecture

[5][6] Il existe plusieurs architectures de convertisseurs, chacune ayant une caractéristique précise en réponse à un domaine particulier. C'est l'application qui détermine le choix du type du convertisseur utilisé.

- **Le convertisseur sigma-delta**, basé sur le principe du suréchantillonnage et sur la mise en forme du bruit, est le plus adapté pour les signaux à bande passante modérée de l'ordre du kilohertz. Il présente la résolution la plus importante, 64 bits voire plus encore. C'est sur ce convertisseur que l'étude a été faite car c'est son architecture qui répond le mieux aux attentes des applications audio. Sa forte résolution lui permet une restitution plus fidèle du signal.

- **Le convertisseur à approximations successives** est basé sur le principe de dichotomie, cela signifie que l'intervalle de comparaison est divisé par deux à l'issue de chaque étape. Son temps de conversion est de l'ordre de la milliseconde (kilohertz) et sa résolution est moyenne de 12 à 16 bits.

- **Le convertisseur flash** est basé sur la comparaison des tensions générées par des diviseurs de tension et du signal à convertir. C'est le plus rapide mais sa résolution est très faible, moins d'une dizaine de bits. D'autre part, il se révèle être le plus cher car il est pourvu d'un grand nombre de composants permettant son architecture. Il est souvent utilisé pour des applications vidéo, où la rapidité de conversion est indispensable.

Le principe

[5][6] L'étude porte plus précisément sur un convertisseur sigma-delta. Sa structure permet d'améliorer la résolution du signal de sortie, tout en modérant le bruit de quantification. Selon le théorème de Shannon[22], afin de garantir une bonne restitution du signal, il faut un taux d'échantillonnage qui correspond au minimum au double de la fréquence maximale d'origine.

[22] Claude Shannon : (1916-2001) Ingénieur électricien et mathématicien américain.

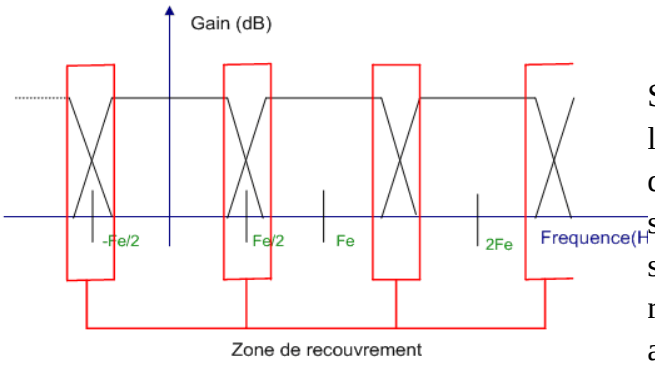


Figure 24 : Respect du théorème de Shannon

Si la condition de Shannon n'est pas respectée, les différents spectres se croisent et au niveau des zones de recouvrement, les spectres s'additionnent. On appelle cela un repliement spectral en anglais « aliasing ». Le spectre restitué après filtrage ne correspond donc plus au spectre réel du signal et les analyses en sont faussées.

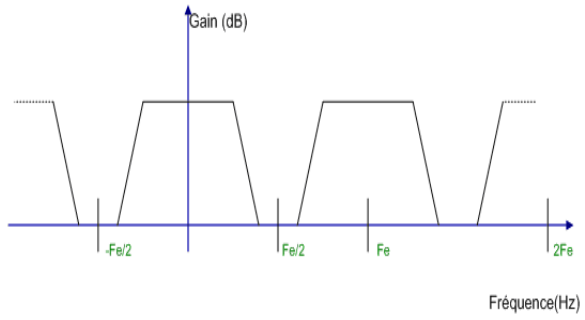


Figure 25 : Phénomène de recouvrement

Lorsque l'on observe la FFT[23] d'un signal échantillonné, il apparaît qu'une infinité de spectres sont créés et centrés autour des différents multiples de la fréquence d'échantillonnage (Fe, 2Fe, XFe)
Cela implique le respect du théorème de Shannon qui impose une fréquence d'échantillonnage au moins deux fois supérieure à la fréquence maximale du signal d'origine.
*Fe > 2 * fréquence du signal*

- La Figure 26 montre que la densité spectrale du bruit en sortie d'un convertisseur est uniformément distribuée, quelque soit la fréquence de –Fs/2 à Fs/2.

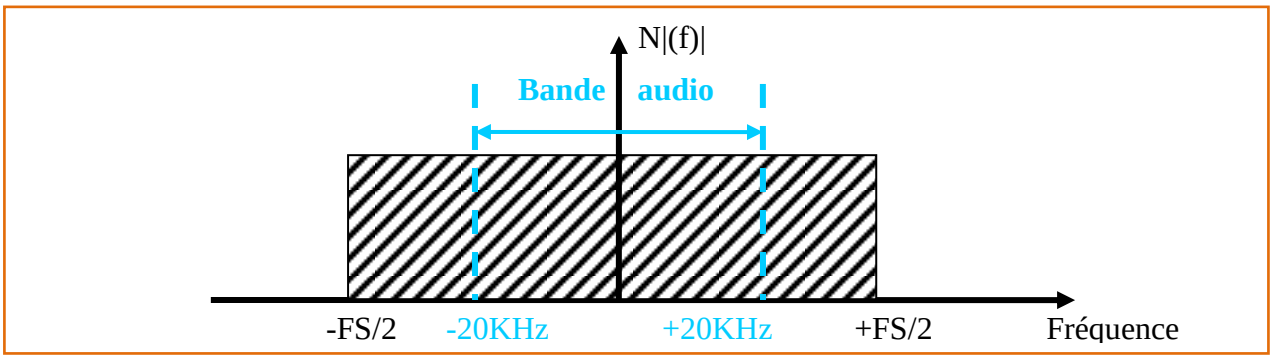


Figure 26 : Densité spectrale du bruit sur la bande audio avant le suréchantillonnage

- Le suréchantillonnage permet une grande résolution du signal numérique car il y a moins de pertes d'informations. La largeur de bande étant plus importante, la densité de bruit devient alors plus étalée (Figure 27), d'où une diminution de son amplitude et l'amélioration du rapport signal à bruit qui augmente.

[23] FFT : Fast Fourier Rapide, transformée de Fourier rapide est un algorithmes de calcul permettant la représentation fréquentielle d'un signal.

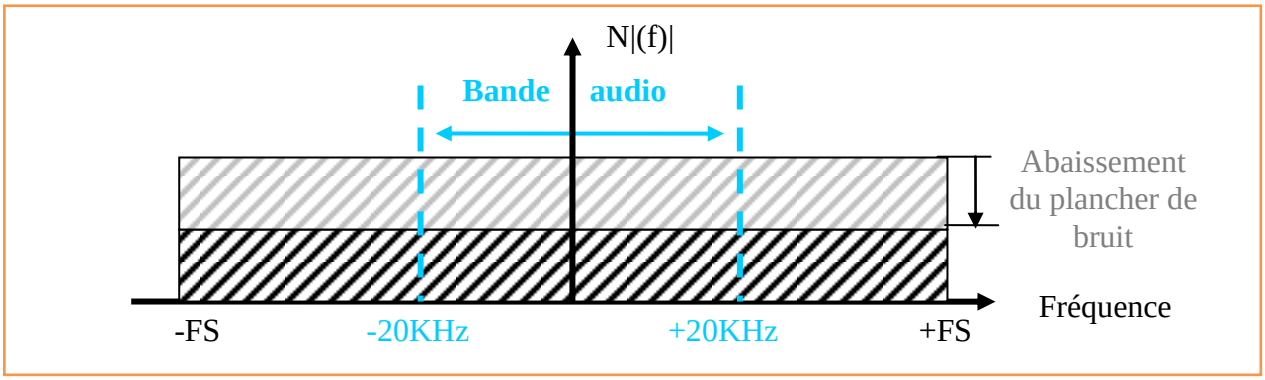


Figure 27 : Densité spectrale du bruit après le suréchantillonnage

- L'échantillonnage du convertisseur se fait sur 1 bit, à très forte fréquence il en résulte un bruit de quantification très important. Pour pallier à cela, le convertisseur dispose d'un filtre de mise en forme, « noise shaper » en anglais. Ce filtre supprime le bruit de quantification, en le repoussant au-delà de la bande de fréquence audio (Figure 28). La densité de bruit reste la même, il abaisse ainsi le plancher de bruit sur toute sa largeur de bande.

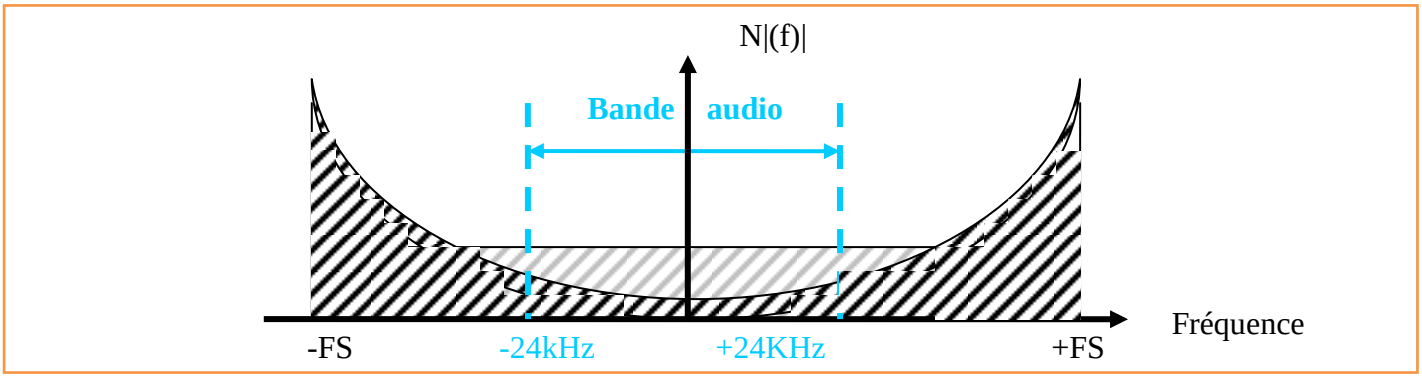


Figure 28 : Densité spectral du bruit en sortie du Noise shaper

L'architecture

[7] L'architecture générale d'un convertisseur sigma-delta, ∑Δ, est déterminée par trois paramètres :

- L'OSR « Over Sampling Rate » c'est le facteur de suréchantillonnage.
- L'ordre : Il définit la fonction de transfert du convertisseur.
- N définit le nombre de bits du quantificateur.

L'OSR du convertisseur a été fixé en fonction des contraintes de division de l'horloge générale audio, 48KHz, avec une OSR de 100, le convertisseur échantillonne à 4.8MHz.
C'est un ADC d'ordre 2 et le nombre de bit est 1 on l'appelle mono-bit.
En théorie pour un tel ∑Δ le SNR donne :

$$SNR = 6.02N + 1.76 - 12.9 + 50 \log(OSR) \quad (3)$$

Avec N = 1 et L'OSR = 100.

Le SNR théorique vaut alors 107.78dB.

La Figure 29 présente le schéma du modulateur ∑Δ étudié, il est composé de deux montages du 1er ordre. Il génère un flot de 1 bit à la fréquence de 4.8MHz.

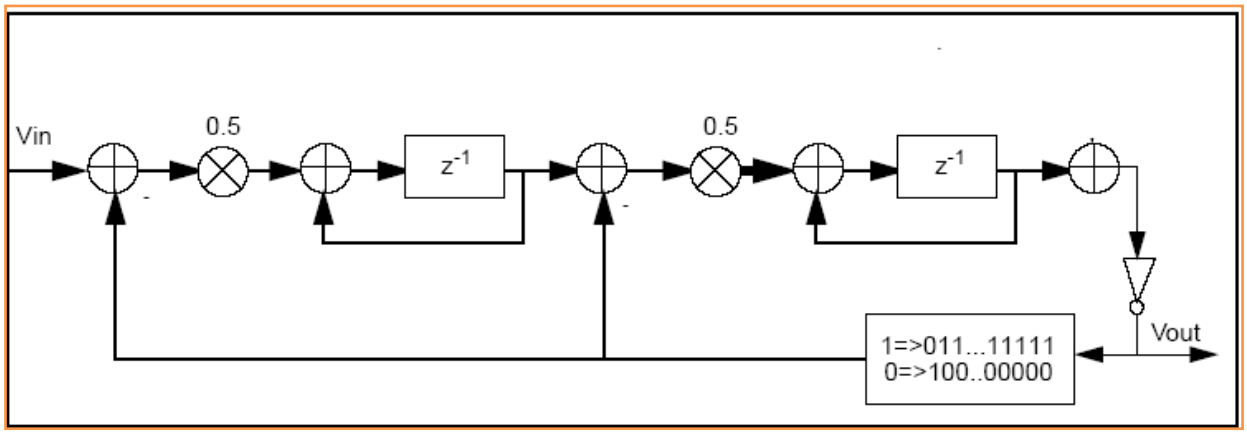


Figure 29 : Modulateur sigma delta du second ordre.

La grande résolution du convertisseur sigma-delta lui permet de s'adapter aux besoins des applications audio. Cependant la fréquence du signal modulé est très importante : c'est la conséquence du suréchantillonnage. Les applications qui reçoivent ce signal n'acceptent qu'une fréquence plus réduite et un nombre de bits contrôlés. Le SNR doit ainsi continuer à être maîtrisé. Pour cela on place des filtres numériques en sortie de l'ADC pour réduire le bruit de quantification et la fréquence d'échantillonnage.

3.3 La chaîne de traitement numérique

3.3.1 La chaîne complète

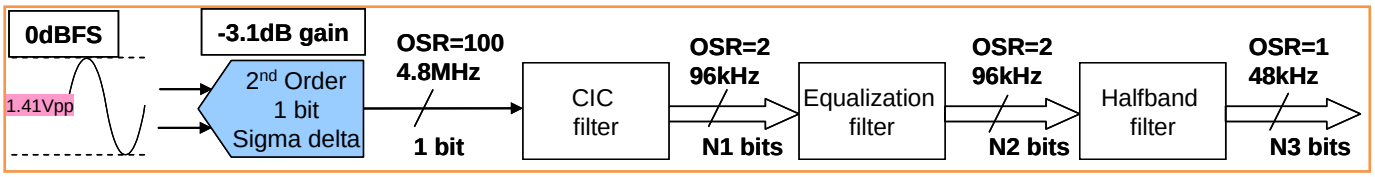


Figure 30 : Principe de la chaîne de traitement numérique

[7][8] La Figure 30 présente la chaîne complète de traitement numérique demandée pour le stage. L'ADC fournit un signal modulé, appelé PDM pour « Pulse Density Modulation », ce qui signifie modulation d'impulsion en durée. Ce signal numérique représente la sinusoïde (analogue) injectée en entrée et encodée numériquement par le convertisseur sigma-delta. Sa fréquence est de 4.8MHz codée sur 1 bit. C'est-à-dire qu'il fournit une suite de 0 et de 1 à la cadence d'un bit toutes les millisecondes. Ce flot de bits est appelé « bitstream » en anglais.

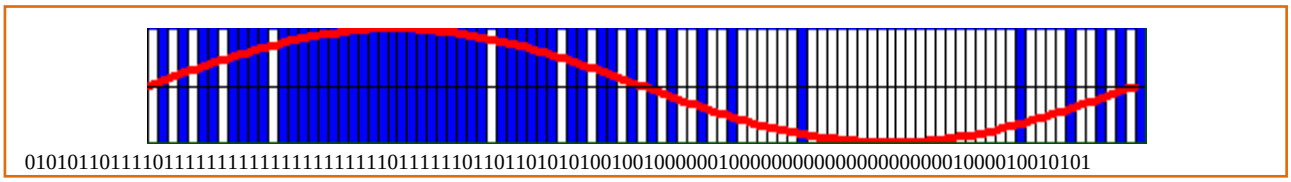


Figure 31 : Exemple d'un signal PDM

Les applications futures sollicitent un flot de données séries de fréquence de 48KHz à 25bits. Il faut donc diviser la fréquence de départ tout en conservant un SNR valable en bout de chaîne. Cette division de fréquence s'appelle la décimation. Le principe est de récupérer un échantillon sur X échantillons, avec X, le facteur de division. Par exemple une décimation de 50, revient à diviser la fréquence par 50, soit 1 échantillon sur 50.

Le choix s'est porté sur trois filtres en cascade plutôt qu'un unique filtre. La raison est qu'ils sont paramétrés par des coefficients qui vont modifier leurs impacts sur le signal. Ces coefficients admettent une multiplication importante du nombre de bits au sein du filtre. Un seul filtre serait trop complexe à implémenter.

3.3.2 Les notions de filtres

Pour permettre une meilleure compréhension de ce qui va suivre, voici quelques notions de base concernant les filtres numériques.

La fonction du filtre est de modifier la réponse en fréquence du signal[24] . Il peut soit atténuer les amplitudes du signal à certaine fréquence soit l'amplifier. Cela est dû à l'intégration de coefficients soit positifs soit négatifs, dans le filtre.

L'architecture des filtres numériques contient des éléments mathématiques :

- Eléments de sommation : Additionneurs/soustracteurs
- Eléments de multiplication : Multiplicateurs/diviseurs
- Retard unitaire (Z^{-1}) : Bascule D

Le procédé de filtrage numérique est une succession d'opérations mathématiques sur le signal discret. C'est pourquoi on appelle ce procédé : traitement du signal.

La définition d'un filtre est donnée grâce à sa fonction de transfert. Il s'agit de connaître la relation entre la sortie y(n) et l'entrée x(n) (Figure 32), grâce au module qui nous donne le gain et l'argument qui indique le déphasage.

Figure 32 : Schéma d'un filtre numérique

La fonction de transfert utilise la transformée en z.[25] C'est le rapport entre la transformée en z de la sortie sur la transformée en z de l'entrée. La fonction de transfert typique d'un filtre est de la forme suivante dans le domaine fréquentiel.

Avec $z = e^{jwt}$, $T = \frac{1}{f_s}$ fs étant la fréquence d'échantillonnage.
N et M l'ordre du filtre

$$H(z) = \frac{y(z)}{X(z)} = \frac{\sum_{K=0}^{N} bk \cdot z^{-k}}{\sum_{K=0}^{M} ak \cdot z^{-k}} \qquad (4)$$

[24]Réponse en fréquence d'un signal : C'est la variation de l'amplitude du signal sur toute la bande de fréquence.
[25] Annexe 4 : Transformée en z

$$(5) \qquad H(z) = \frac{bo + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_3 z^{-3}}$$

Les trois filtres sont à réponse impulsionnelle finie ou FIR en anglais pour « Finite Impulse Response ». L'absence de rebouclage dans leur structure fait que leur sortie ne dépend que de leur entrée : ce sont des filtres récursifs. Leur réponse est stable et finie, elle est dépendante du nombre de coefficients du filtre.

3.3.3 Première décimation grâce au filtre CIC

Fonction et définition : CIC « Cascaded Integrator Comb », signifie en français : Peigne[26] à intégrateur en cascade. Son rôle est de décimer la fréquence du signal, ce qui revient à le sous-échantillonner. Le principe est de ne garder qu'un certain nombre d'échantillons par rapport au signal de départ. Dans notre cas, le signal passe alors d'une fréquence de 4.8MHz à 96KHz soit une décimation de 50. Il ne contient pas de coefficient.

La Figure 33 montre l'architecture de ce filtre. Il est structuré en deux parties, une première composée de troid étages d'intégrateurs (I) en cascade et une seconde de trois étages de différentiateurs (C) en cascade. Le fait de diviser la fréquence au milieu du système permet de réduire la consommation pour la deuxième partie du montage. C'est pourquoi la décimation est faite au centre des deux parties.

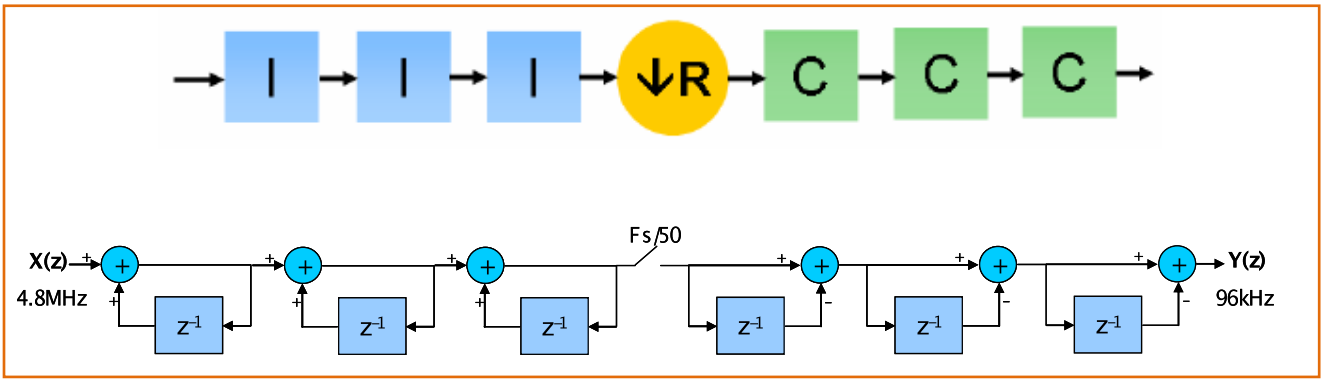


Figure 33 : Architecture du filtre CIC

Fonction de transfert [27]dans le domaine fréquentiel :

$$(6) \qquad |H(f)|_{dB} = 101.9dB + 60.\log_{10}\left[\frac{\mathrm{sinc}^3\left(\frac{MR\pi f}{96k}\right)}{\mathrm{sinc}^3\left(\frac{\pi f}{4.8M}\right)}\right]$$

[26] La réponse en fréquence du filtre "combe" se compose d'une série de pics régulièrement espacés donnant l'apparence d'un peigne.
[27] Annexe 5 : Détails des calculs de la fonction de transfert du filtre CIC.

La Figure 34 est une représentation graphique de la fonction de transfert d'amplitude du filtre CIC**:**

La fréquence est en Hz pour l'abscisse (de 0 à 5MHz) et l'amplitude est en dB pour l'ordonnée (de -20dB à +120dB).

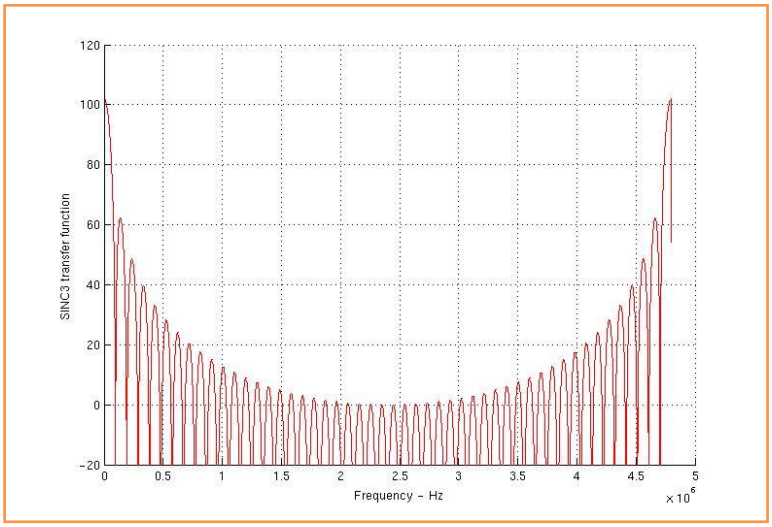


Figure 34 : Fonction de transfert du filtre CIC

La bande audio ne concerne que le premier lobe de la fonction de transfert soit jusqu'à 20KHz. C'est ce que présente la Figure 35 permettant ainsi l'analyse du gabarit du filtre : gain par rapport à la fréquence. Celui-ci est peut être observé en ordonnée.

La fréquence est en Hz pour l'abscisse (de 0 à 20kHz) et l'amplitude est en dB pour l'ordonnée (de +99.5dB à +102dB).

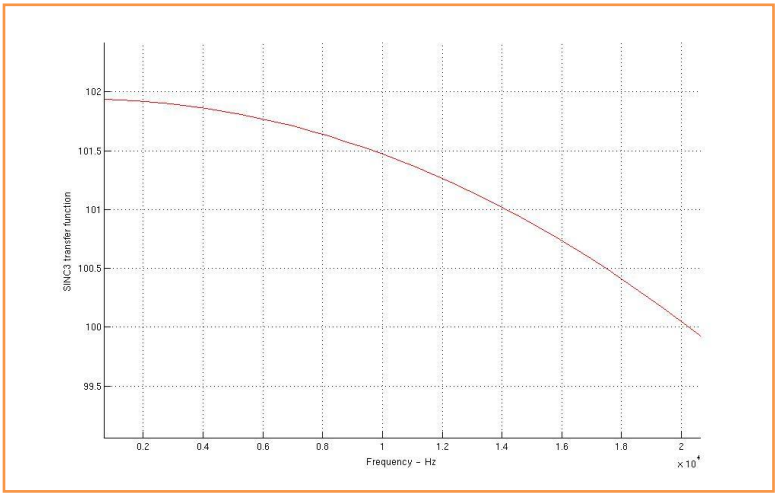


Figure 35 : Fonction de transfert du filtre CIC dans la bande audio

A 2KHz l'amplitude du signal est d'environ de 102dB et à 20KHz, le gain équivaut à 100dB, soit une atténuation de 2dB dans la bande audio.

3.3.4 Egalisation grâce au filtre FIR

Fonction et définition : Le filtre CIC a atténué de 2dB le signal en sortie de l'ADC. La fonction de l'« équalizer », égaliseur en français, est de compenser cette atténuation. Cinq coefficients de pondération permettent d'obtenir un gain constant sur toute la bande audio avec une erreur de +/- 0.1dB. Ce filtre a un facteur de décimation égale à 1, c'est-à-dire que la fréquence d'entrée est la même que la fréquence de sortie, 96kHz. La Figure 36 présente l'architecture du filtre égaliseur. Il est composé d'intégrateurs (z^{-1}) et d'additionneurs (+). Les coefficients (a0, a1, a2) dépendent de la forme de réponse souhaitée.

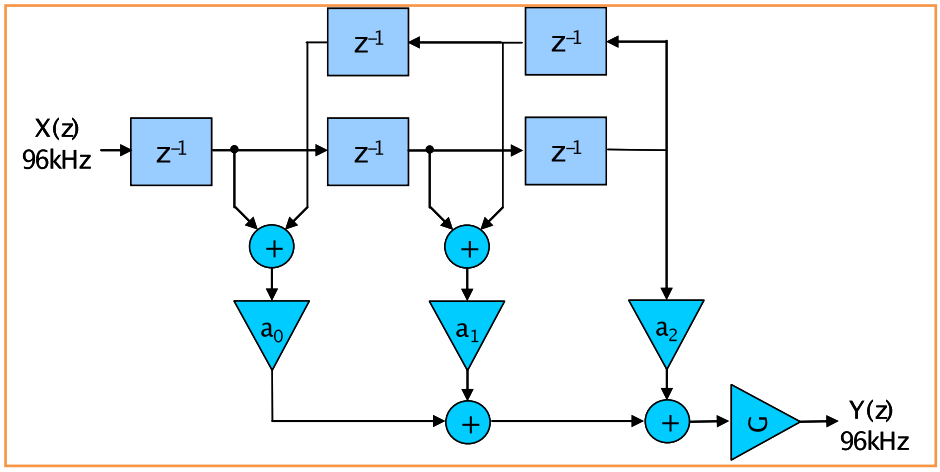


Figure 36 : Architecture du filtre égaliseur

Fonction de transfert[28] dans le domaine fréquentiel :

Soit A = *[245 -1952 11620 -1952 245]* les coefficients. La symétrie du filtre fait passer le nombre de coefficients conçus de 5 à 3. Un système d'addition permet de redistribuer les coefficients identiques

$$H(z) = 81.3dB + 20.\log_{10}\left[1 + 4.21e^{-2}\cos\left(\frac{4\pi f}{Fs}\right) - 3.36e^{-1}.\cos\left(\frac{2\pi f}{Fs}\right)\right] \quad (7)$$

La Figure 37 est une représentation graphique de la fonction de transfert du filtre égaliseur.

[28] Annexe 6 : Détails des calculs de la fonction de transfert du filtre FIR.

La fréquence est en Hz pour l'abscisse (de 0 à 100kHz) et l'amplitude est en dB pour l'ordonnée (de +78dB à +85dB).

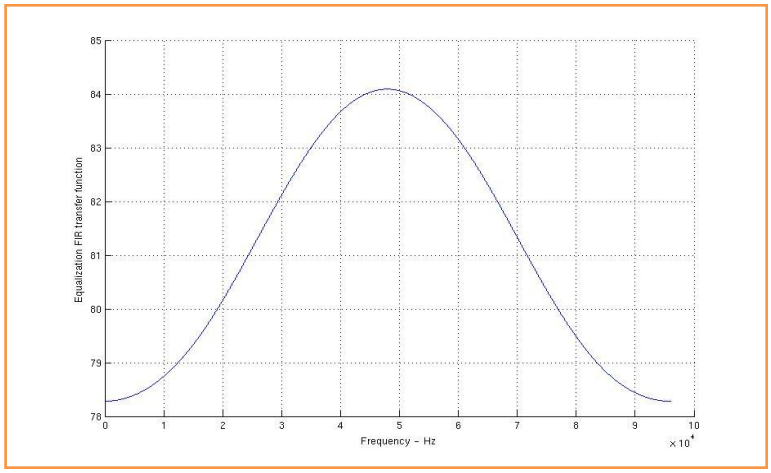


Figure 37 : Fonction de transfert de l'égaliseur

Encore une fois, nous ne nous intéressons qu'à la représentation dans la bande audio. La fonction de transfert apparaît inversée par rapport au filtre CIC. Il ne s'agit plus d'une atténuation de -2dB mais d'une amplification de +2dB entre 20Hz et 20kHz. Ce qui permet de trouver en sortie le gain constant escompté comme le montre la Figure 38.

La fréquence est en Hz pour l'abscisse (de 0 à 20kHz) et l'amplitude est en dB pour l'ordonnée (de +179.2dB à +180.6dB).

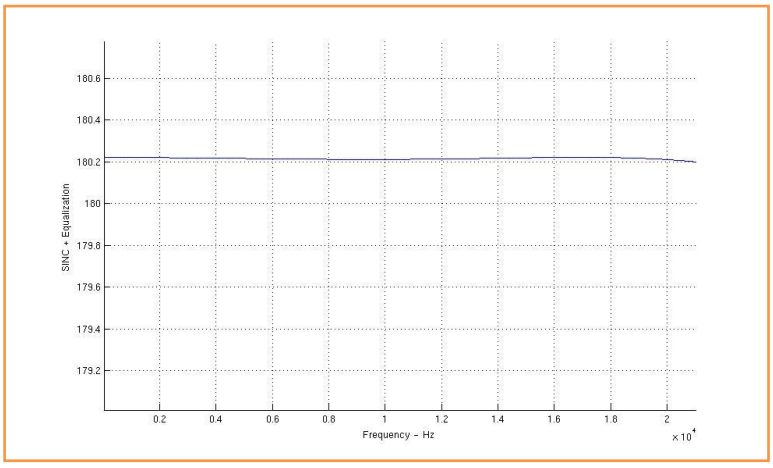


Figure 38 : Compensation obtenue grâce à l'égaliseur

3.3.5 Dernière décimation grâce au filtre FIR passe bas

Fonction et définition : La fonction de ce filtre est de décimer le signal en évitant une quelconque atténuation de l'amplitude. La fréquence en sortie est divisée par 2, soit les 48kHz attendus par l'application. Il est pourvu de 43 coefficients de pondération qui le rend complexe à concevoir comme le montre la **Error! Reference source not found.**. Il permet d'atténuer les bruits hautes fréquences se trouvant au voisinage de la bande audio.

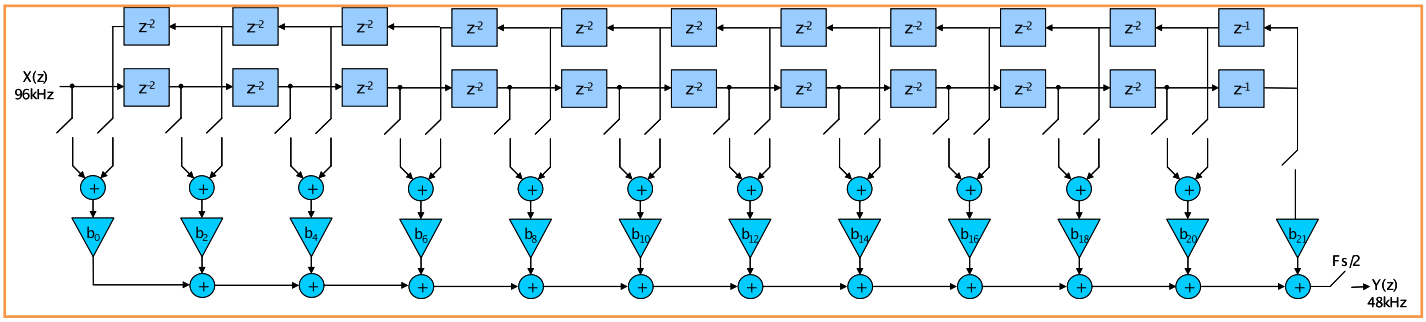


Figure 39 : Architecture du filtre FIR passe bas

Fonction de transfert[29] dans le domaine fréquentiel :

$$(8)\quad H(z) = 72.26\text{dB} + 20.\log_{10}\left[1 + 1.26\cos\left(\frac{\pi f}{48k}\right) + \ldots + 3.42e^{-3}\cos\left(21\frac{\pi f}{48k}\right)\right]$$

Les 45 coefficients =[7 0 -14 0 27 0 -48 0 78 0 -123 0 188 0 -288 0 458 0 -830 0 2594 4096 2594 0 -830 0 458 0 -288 0 188 0 -123 0 78 0 -48 0 27 0 -14 0 7] . La symétrie du filtre fait passer le nombre de coefficients de 43 à 22.

[29] Annexe 7 : Détails du calcul de la fonction de transfert du filtre FIR passe bas.

La Figure 40 est une représentation graphique de la fonction de transfert du filtre FIR passe bas. Le graphique montre effectivement l'absence d'atténuation et d'augmentation du gain dans la bande audio.

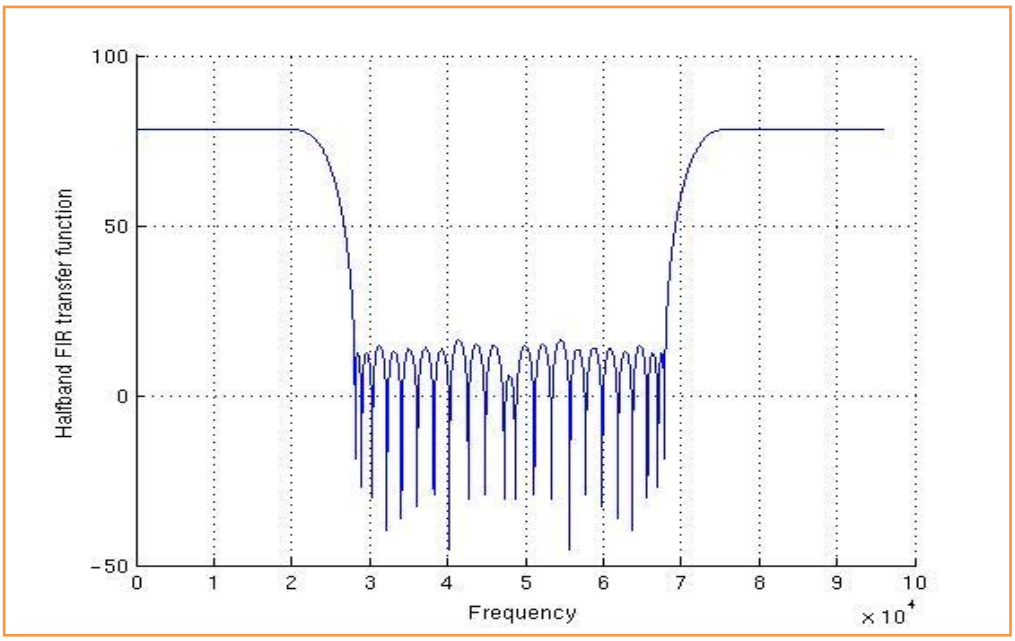


Figure 40 : Fonction de transfert du filtre FIR passe bas

3.4 Bilan de l'étude théorique

L'étude du convertisseur sigma-delta nous informe sur son intérêt dans le domaine audio. Cela est dû à sa grande résolution. Ce qui implique un signal de fréquence importante en sortie. Le principe du suréchantillonnage en est la cause ; d'où l'intérêt de la conception des filtres numériques en sortie. Cette chaîne de traitement doit réduire la fréquence du signal tout en conservant son amplitude. La validation[30] d'un tel circuit se base sur l'analyse du spectre en sortie en observant l'impact du bruit de quantification.

L'étude des paramètres de la chaîne donne toutes les indications utiles à la saisie des filtres dans le FPGA. Ce développement se fait via le logiciel de conception utilisé précédemment pour l'évolution de la solution : Quartus.

[30] Annexe 8 : La validation du convertisseur

4 APPROCHE EXPERIMENTALE

4.1 Etude de l'environnement de programmation

4.1.1 Le logiciel Quartus II

Quartus II est un logiciel de conception fourni par le constructeur de FPGA, Altera. C'est un environnement de programmation qui gère complètement le processus de conception d'un circuit programmable. La Figure 41 présente les étapes de développement qui seront décrites au fur et à mesure de l'implémentation des filtres.

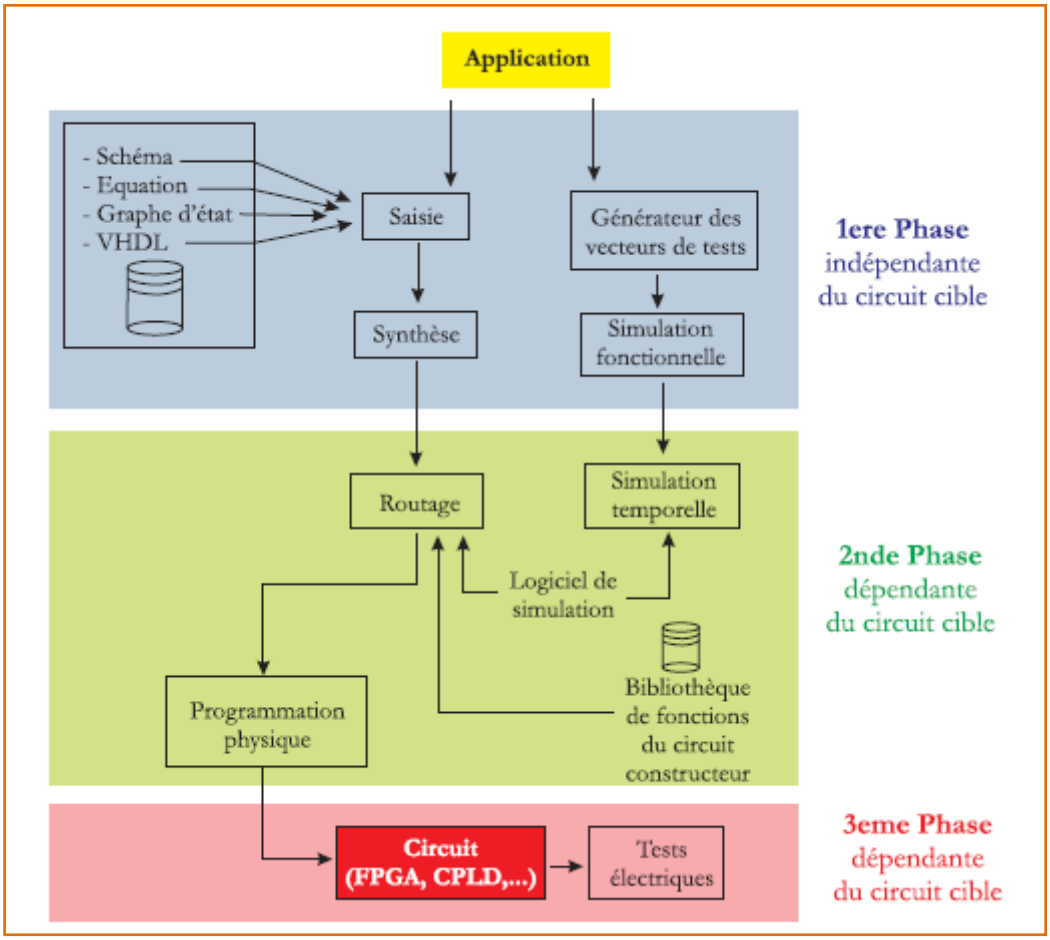


Figure 41 : Processus de conception d'un circuit FPGA

4.1.2 La méthode de saisie

La méthode de programmation qui doit être utilisée pour le stage est la description matérielle par saisie graphique. Cette méthode permet de placer directement les composants (symboles graphiques) utilisés et de les interconnecter soit à l'aide de net ou de bus lorsqu'ils contiennent plusieurs fils. La conception des filtres se fera sous forme de projet où la saisie des différentes fonctions est gérée de façon hiérarchique. Le plus haut niveau représente le schéma complet qui fait appelle aux schémas de plus bas niveaux qui sont représentés par les fonctions logiques de base. Cette structure est assimilée à un sous-projet mise sous la forme d'un symbole. La Figure 42 présente ces différents niveaux. L'utilisation de la conception hiérarchique donne une meilleure lisibilité à l'ensemble du schéma.

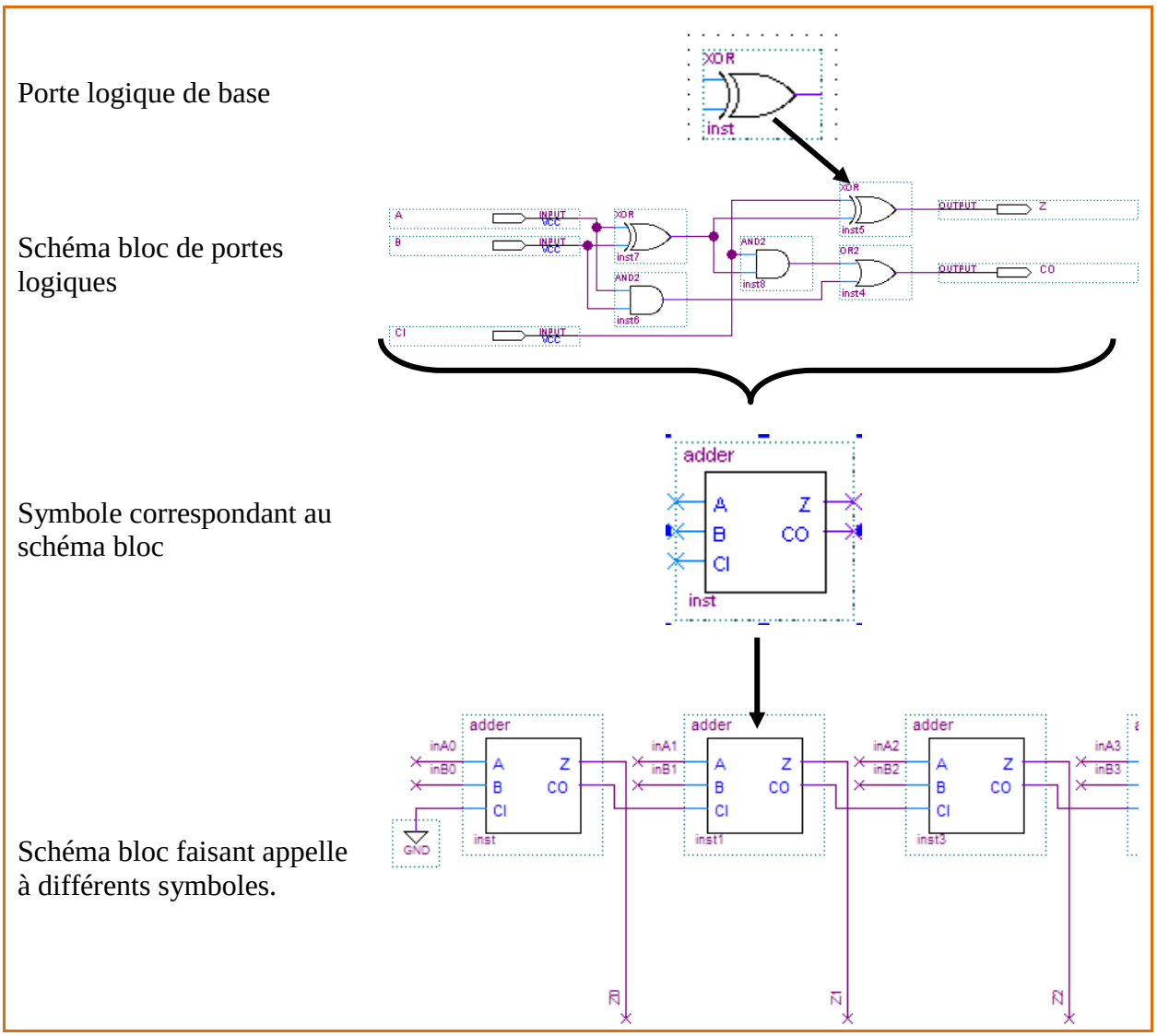


Figure 42 : Organisation hiérarchisée des éléments d'un projet

4.1.3 La gestion des librairies

La conception des filtres nécessite l'utilisation des portes logiques ou des bascules D. Ils sont disponibles dans différentes bibliothèques, et doivent être associés à la librairie propre au projet lorsqu'ils sont utilisés. Voici les deux bibliothèques de composants utilisées pour ce projet:

1. **La librairie standard d'Altera** : Quartus propose les composants les plus basiques comme les portes de la logique combinatoire (ET, NON, OU), mais également les bascules de base (D, T) de la logique séquentielle.

2. **La librairie spécifiée par le laboratoire** : Il serait fastidieux d'utiliser les composants de base en l'état. Et il serait surtout contraignant de devoir recréer toujours les mêmes blocs diagrammes. Les nombreuses structures créées lors de la conception d'un schéma sont ensuite intégrées dans cette bibliothèque. Ainsi les composants nouvellement créées sont directement disponibles pour les projets suivants.

Il est important de spécifier que l'utilisation de ces librairies est propre à Quartus, il est donc impossible l'utiliser en dehors de cet environnement. C'est le grand inconvénient de cette méthode de conceptions. Nous verrons plus tard les conséquences de ce choix de saisie.

4.2 La chaîne à implémenter

La conception des filtres se fait à partir du schéma donné par le concepteur (Cf. § 3.3). Les filtres sont présentés sous forme de blocs diagrammes comportant des éléments mathématiques qui permettent de réaliser la fonction numérique attendue. Le travail demandé consiste à saisir ces blocs à l'aide du logiciel Quartus en utilisant la méthode graphique. La Figure 43 présente cette chaîne avec les signaux à prendre en compte lors de la conception de chacun des filtres.

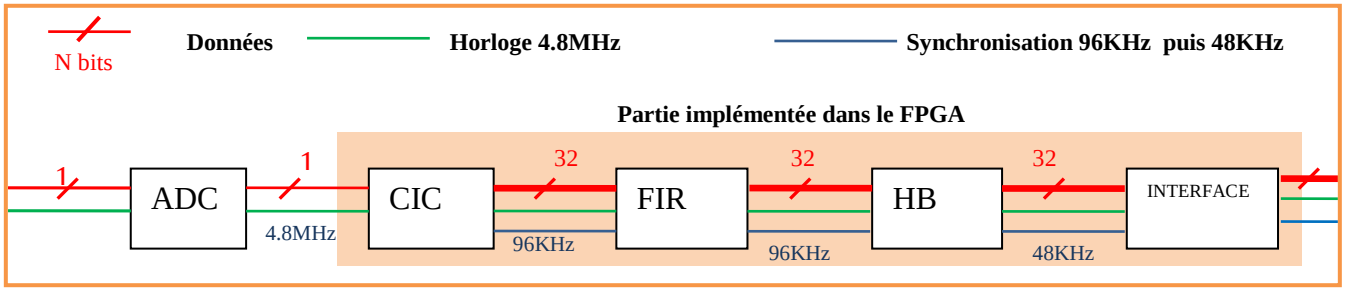


Figure 43 : La chaîne à implémenter

Ces filtres seront représentés par un symbole dans la chaîne finale. C'est à l'intérieur de celui-ci que s'effectue le traitement de l'information. La figure 44 montre le flot de conception au sein du filtre. Les signaux entrants de données et d'horloges seront mis en forme pour subir le traitement de des données (calculs mathématiques) et sont renvoyés à la sortie une fois remis en forme.

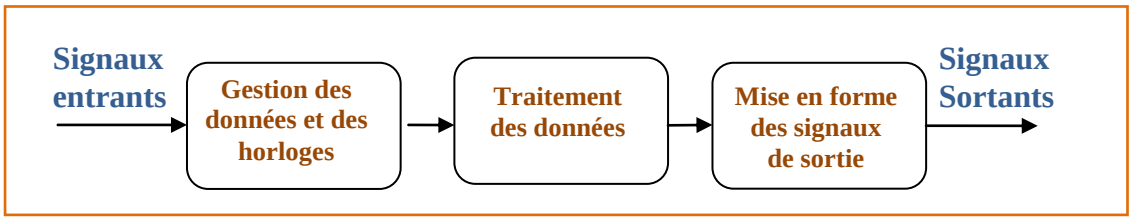
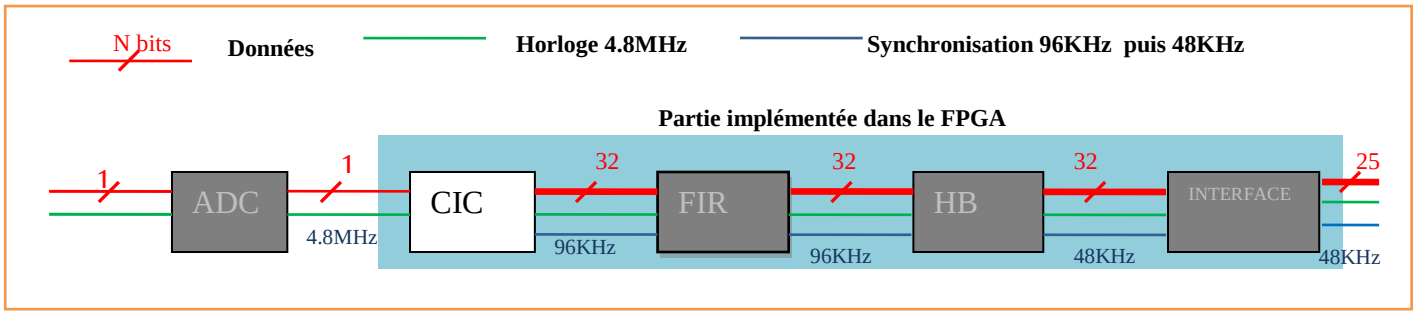
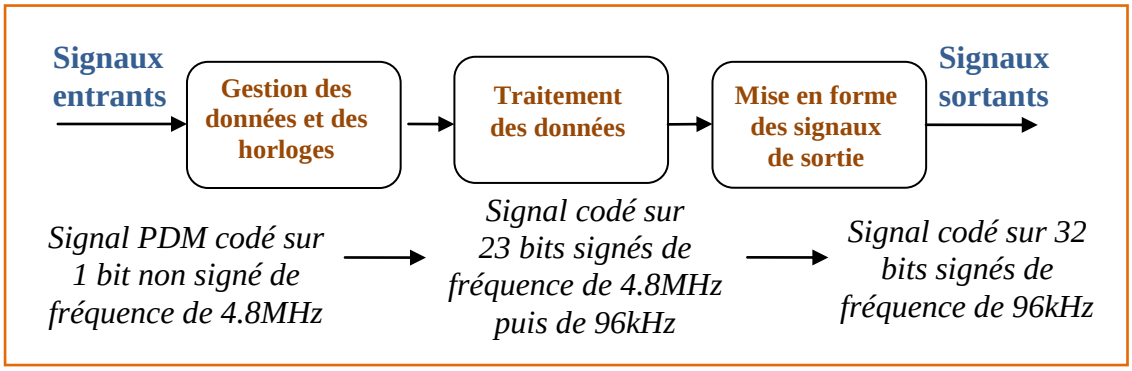


Figure 44 : Structure de la conception des filtres

4.3 Conception du filtre CIC



Comme étudié dans le chapitre 3.4.3, ce filtre se compose de trois intégrateurs en cascade suivis de trois différentiateurs en cascade également. Voici la description de la conception de ce filtre, comprenant la mise en forme des données en entrées et la gestion des fréquences et de la synchronisation des horloges.



4.3.1 Les entrées

Ce filtre est le premier situé en sortie du convertisseur analogique-numérique dont la sortie est un signal PDM codé sur 1 bit envoyé en série à la fréquence de 4.8MHz. Ce signal est représenté par les entrées suivantes:

- **pdm_input** : Signal comportant les données codées sur 1 bit non signé, de valeur « 0 » ou « 1 ».
- **pdmclk_input** : Signal d'horloge d'envoi des données, c'est la fréquence du signal PDM en sortie de l'ADC de 4.8MHz, également codé sur 1 bit.
- **resetn** : Signal codé sur 1 bit. Il est actif à l'état bas. Ce signal réinitialise le montage, lorsque cette entrée est à « 0 ». Donc pour que le système fonctionne, il faut la mettre à « 1 ».

4.3.2 Conversion des données

Une représentation signée des bits est nécessaire car elle permet la compatibilité avec tous les calculs arithmétiques. Les opérations effectuées sur les données augmentent rapidement la taille du mot reçue en sortie. D'où l'importance d'avoir un signal de bits centrés sur « 0 ». Les valeurs des bits sont donc « 1 » et « -1 », soit des bits signés. La Figure 45 montre que la moyenne du signal devient nulle, il y a donc une maîtrise de l'augmentation de la taille du mot en sortie. Des calculs théoriques effectués par le concepteur ont montré que cette augmentation ne pouvait pas excéder 17 bits.

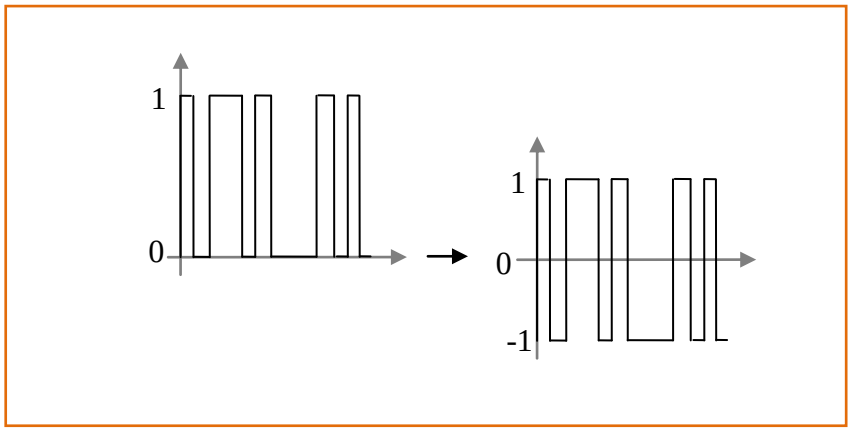
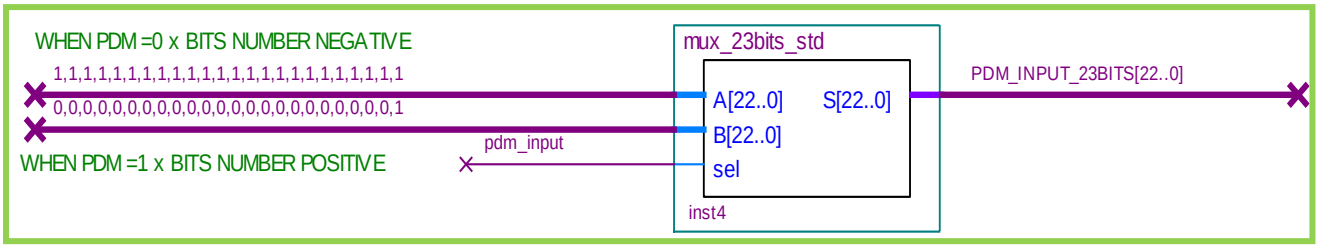


Figure 45 : Modification de la moyenne du signal de données

La première étape est donc de transformer les bits du signal de données PDM de valeur « 0 » et « 1 » d'origine en bits signés « -1 » et « 1 ». Pour permettre le traitement des signaux d'entrées, ceux-ci sont mis sous forme de bus de 23 bits signés. Ce choix a été fait car il existe déjà de nombreux composants de 23 bits dans la librairie spécifiée par le laboratoire.

Un multiplexeur logique est utilisé pour envoyer sur une même sortie une de ses entrées sélectionnées. Son fonctionnement est définit par la valeur de pdm_input qui détermine le signe du bus de 23 bits :

- Soit pdm_input = 0, c'est l'entrée A qui est choisie. Dans ce cas, on attribut à ce bus de 23 bits un nombre négatif : « -1 »
- Soit pdm_input = 1, c'est l'entrée B qui est choisie. C'est une valeur positive qui est attribuée au bus de 23 bits : « 1 »



1 / construction du bus de 23 bits *2 / Attribution du signe du mot de 23 bits*

Figure 46 : Conception d'un convertisseur de données

Les étages intégrateurs

Principe de fonctionnement :

L'intégrateur, composé d'un additionneur et d'une bascule D, accumulent les bus de 23 bits signés à la fréquence d'horloge d'origine. Dans ce cas la bascule D est utilisée comme mémoire élémentaire. Ainsi elle garde en mémoire les valeurs qui seront ensuite intégrées avec les nouvelles valeurs. Le Tableau 3 montre les différentes étapes d'accumulation des intégrateurs

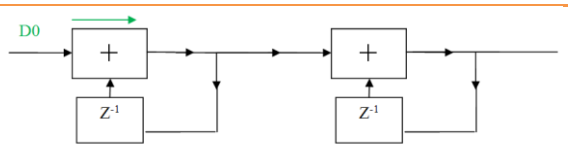
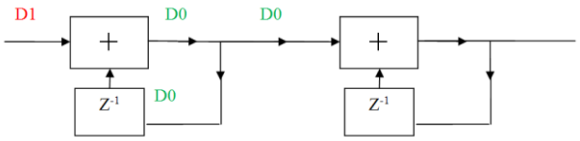
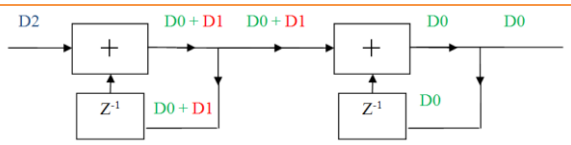
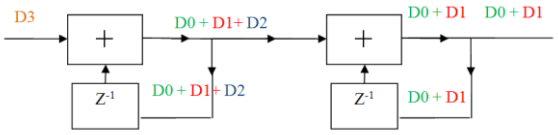
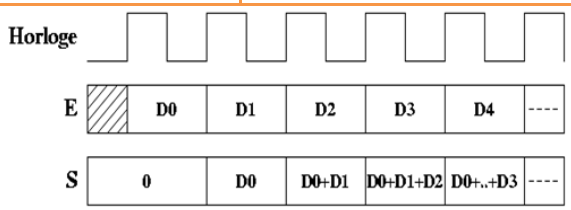
1er cycle d'horloge :	
2ème cycle d'horloge :	
3ème cycle d'horloge :	
4ème cycle d'horloge	
	

Tableau 3 : Séquence d'accumulation des données par l'intégrateur

Etude de l'additionneur :

Nous avons deux entrées A et B. Il faut traduire en porte logique une addition binaire pour les cumuler. Si les deux entrées ont plus d'un bit, il faut veiller à ce que la retenue soit bien réinjectée au bit de gauche. La Figure 47 présente le principe de l'addition binaire qui sera transcrit en un schéma logique.

Opération

0000 1110 R
0000 0111 A
+ 0100 0001 B

0100 1000 S

Table de vérité

a	b	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Formules

$S = a \oplus b \oplus r$

$R = a.b + (a \oplus b).r$

Montage numérique de l'additionneur : $S = a \oplus b \oplus r$

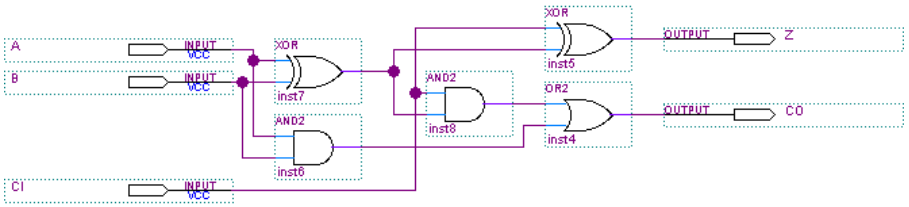


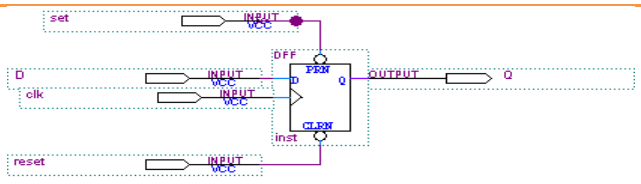
Figure 47 : Principe de l'addition binaire

Cependant, il ne s'agit pas d'un additionneur 1 bit mais 23bits. La méthode est simple, il suffit de mettre 23 additionneurs en parallèle. Pour permettre une meilleure visibilité, chaque additionneur est d'abord transformé en un symbole.

Etude de la bascule:

Il n'y a pas de conception de la bascule de base car elle existe en l'état dans la bibliothèque standard de Quartus. Simplement il faut là encore tenir compte des 23 bits du bus d'entrée, soit une mise en parallèle de 23 bascules toutes synchronisées sur le front montant de l'horloge d'origine. Pour illustrer cette notion de mise en parallèle des bascules, voici en Figure 48 l'exemple d'une bascule 2 bits.

Bascule de base



Bascule 2 bits

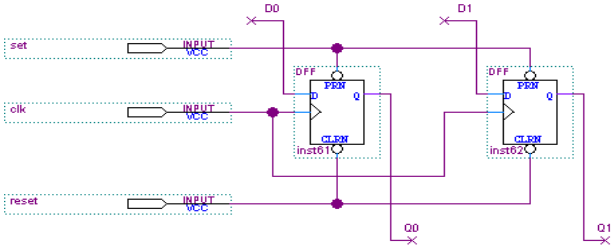


Figure 48 : Mise en parallèle de deux bascules

Mise en œuvre des montages intégrateurs.

Il reste alors à mettre en cascade les trois montages identiques contenant chacun un additionneur et une bascule correctement interconnectés. Le résultat de l'opération binaire est injecté d'une part à l'entrée D de la bascule d'autre part à l'entrée du prochain additionneur. La sortie Q de la bascule est connectée à la deuxième entrée de l'additionneur. La Figure 49 présente cette chaîne d'intégrateurs comme elle est intégrée dans la conception finale.

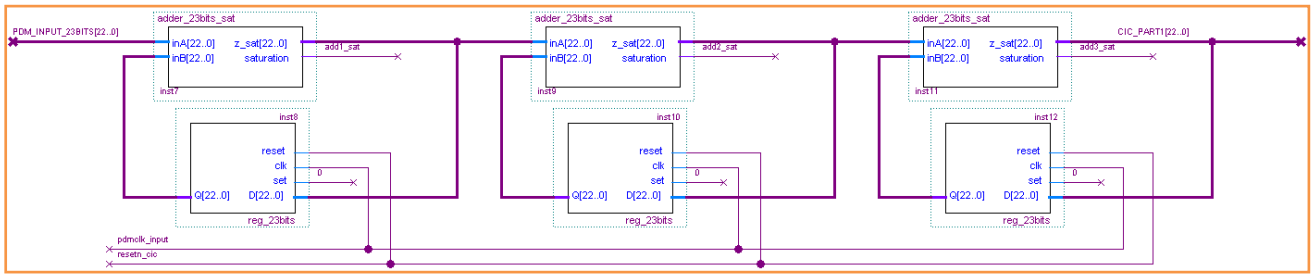


Figure 49 : Les 3 étages d'intégrateurs

4.3.3 Les étages différentiateurs.

La deuxième partie comporte les trois étages des différentiateurs. Ils fonctionnent à une fréquence plus réduite, qui correspond à la fréquence d'origine divisée par 50, soit 96KHz. Le principe est de soustraire le bus D1 avec le bus D0, le bus précédent. L'intérêt est de ne récupérer qu'un échantillon sur 50. Les mots deviennent alors plus réduits en nombres de bits. La première étape pour ce montage va être de créer une horloge de fréquence divisée par 50.

Diviseurs d'horloges

La décimation demandée est de passer de 4.8MHz à 96KHz, ce qui correspond à un échantillon tous les 50 points pris sur le signal d'horloge pdmclk_input. La Figure 50 montre cette fonction comme elle est présentée dans le schéma final de ce filtre.

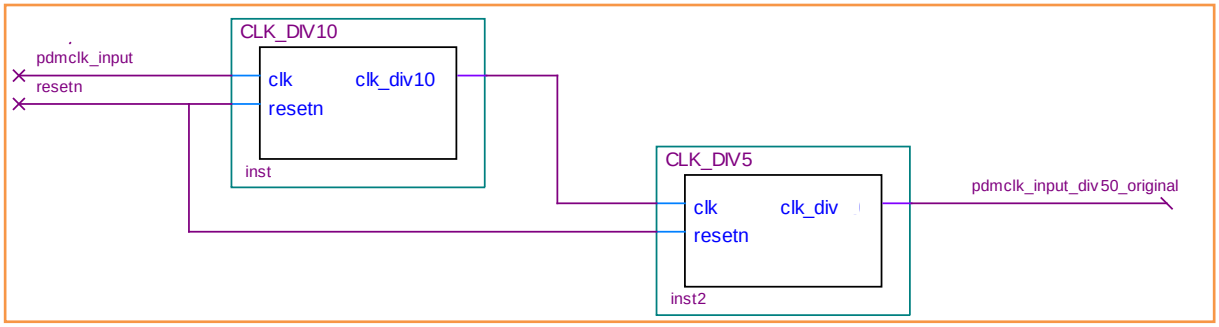


Figure 50: Schéma du diviseur par 50

Dans la librairie customisée il existait déjà un diviseur par 10. Il restait alors à concevoir un diviseur par 5. La structure du diviseur par 10 n'utilise que des bascules D en cascade car elle permet de diviser la fréquence d'entrée par 2 en sortie. Cinq bascules D en cascade ont donc été utilisées pour diviser par 10. C'est le montage présenté par la Figure 51.

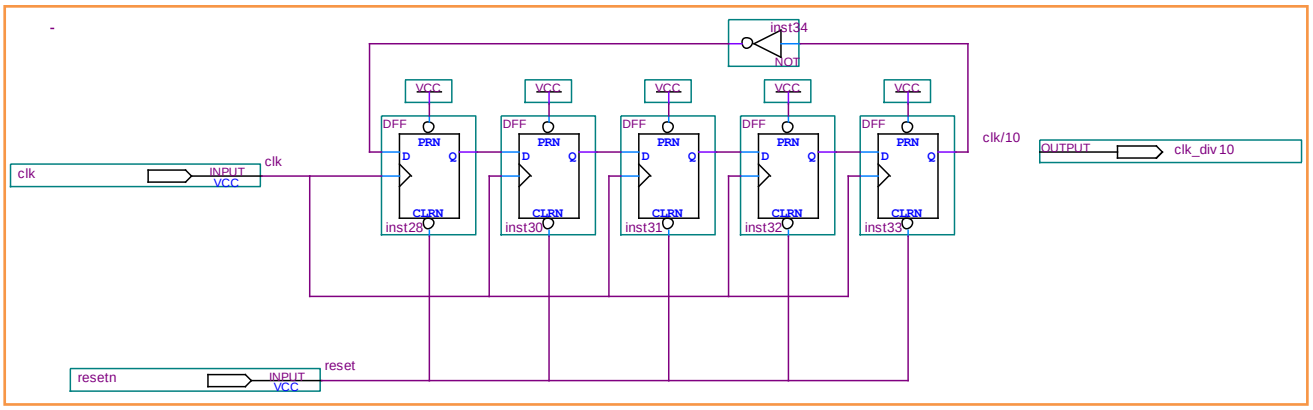


Figure 51 : Schéma du diviseur par 10

Ce schéma est vrai pour toutes les divisions faites par un chiffre pair car la logique séquentielle est suffisante. C'est grâce au fait qu'elle travaille sur des périodes entières. Cependant, la division par un chiffre impair implique l'utilisation de la logique combinatoire. C'est le cas pour diviser par 5, cela permet de récupérer des demi-périodes. La Figure 52 présente le schéma conçu.

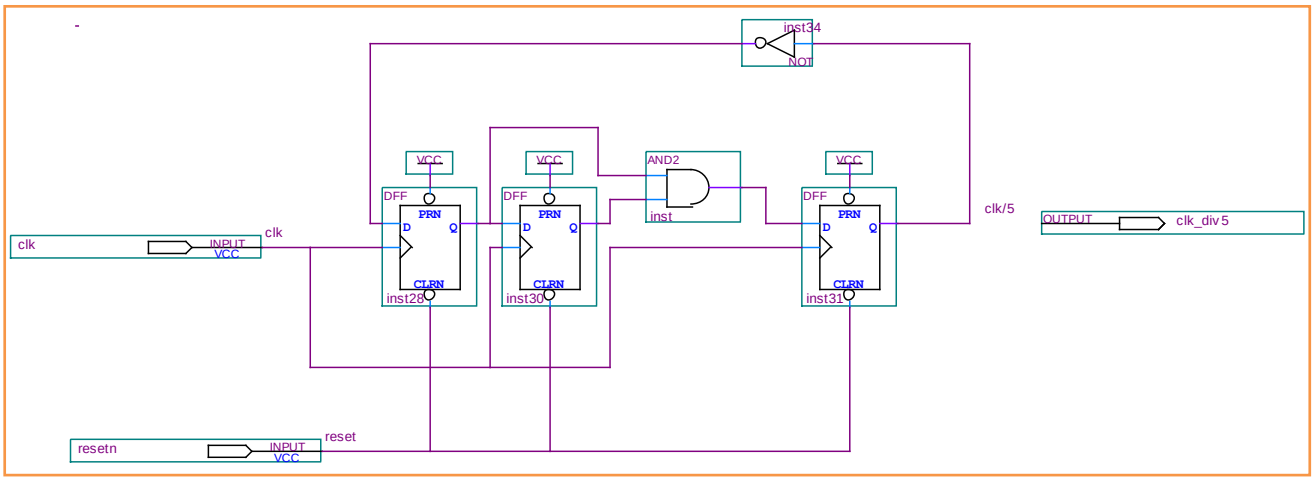


Figure 52 : Schéma du diviseur par 5

Principe de fonctionnement de la chaîne des différentiateurs :

Le montage d'un différentiateur comporte un soustracteur et une bascule D. Le Tableau 4 présente le fonctionnement de ces étages par étape.

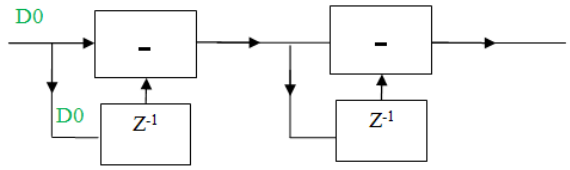
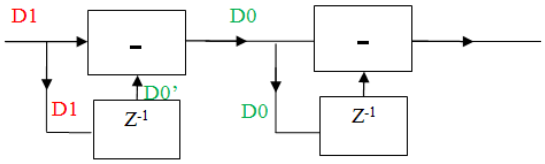
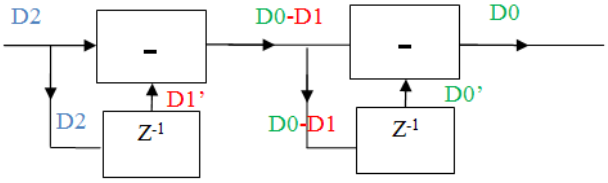
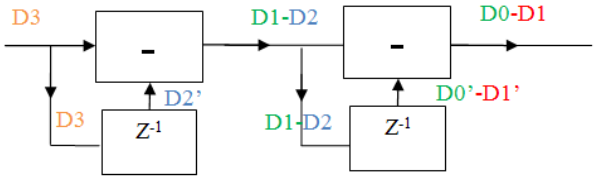
1er cycle d'horloge :	
2ème cycle d'horloge :	
3ème cycle d'horloge :	
4ème cycle d'horloge	

Tableau 4 : Séquence de soustraction des données par le différentiateur.

Etude du différentiateur

Le soustracteur n'est autre qu'un additionneur, où le second élément est inversé. Pour cela, il sera complémenté : il s'agit d'effectuer un complément à deux[31].
Ce qui revient à faire 4+ (-3) plutôt que 4-3 le résultat donne dans les deux cas 1. La méthode est d'inversé les 23 bits du bus ce qui revient à mettre tous les bits « 0 » à « 1 » et tous les bits « 1 » à « 0 ». Cette inversement est effectué grâce à une porte « non » placée sur chaque bit et d'ajouter 1 au total. Le symbole d'inversion est ajouté dans le symbole additionneur, la Figure 53 présente son implémentation. Le bloc diagramme du soustracteur est le même que l'additionneur avec la prise en compte du +1 du complément à deux.

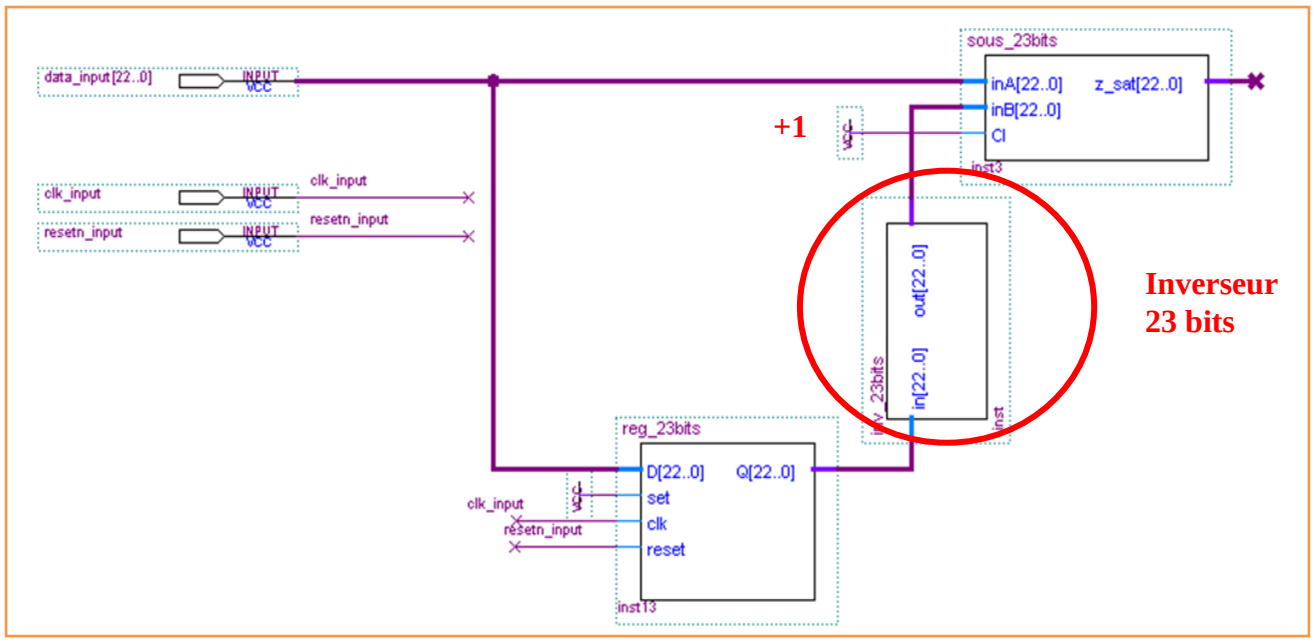


Figure 53 : Bloc diagramme du différentiateur

Mise en œuvre des montages intégrateurs.

Il s'agit d'interconnecter trois différentiateurs identiques. Les opérations seront synchronisées sur la fréquence divisée. La figure 54, présente la mise en cascade de ces 3 différentiateurs.

[31] Annexe 9 : Complément à deux.

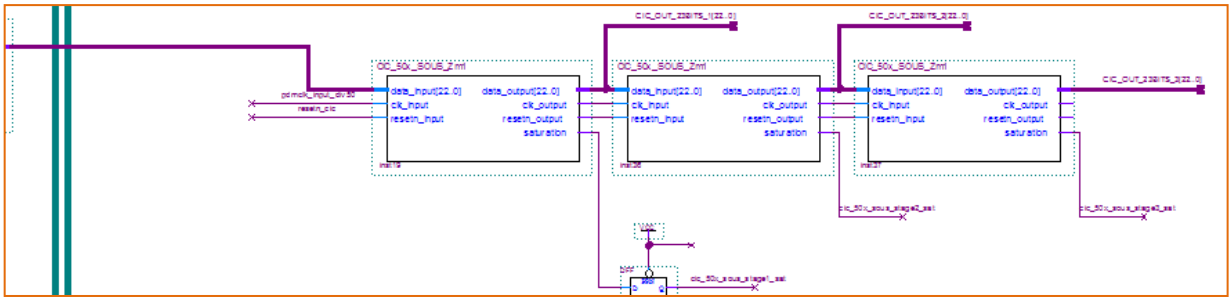


Figure 54 : Mise en cascades des différentiateurs

4.3.4 Gestion de remise à zéro et gestion de la saturation

Lors du fonctionnement du filtre, il est prévu qu'une « anomalie » vienne perturber le bon déroulement des calculs. Il est alors possible de forcer la remise à l'état initiale afin de stopper la progression. C'est le rôle de l'entrée « resetn ». Par ailleurs, l'interruption peut avoir lieu si l'un des additionneurs du filtre entre en saturation. Cela arrive lorsque la valeur du résultat de l'opération dépasse le nombre de bits prévus, soit 23 bits signés. Sachant que les valeurs possibles vont de (-2^{23}) à $(+2^{23}-1)$, au-delà de cette gamme de valeur, il y a saturation. Ce qui se traduit par la perte du signe, placé sur le bit de poids fort d'où un résultat erroné.

On obtient le tableau 5 qui représente table de vérité suivante avec en entrée :

- six additionneurs ou soustracteurs à prendre en compte pour ce filtre, soit six sorties de saturation appelé addx_sat
- le signal « resetn » qui doit être prioritaire sur les autres signaux.

	add1_sat	add2_sat	add3_sat	add4_sat	add5_sat	add6_sat	Resetn (actif à l'état bas)	Sortie reset cic (actif à l'état bas) soit saturation = 0
Evènement1	0	0	0	0	0	0	0	0
Evènement2	0	0	0	0	0	0	1	1
Evènement3	0	0	0	0	0	1	0	0
Evènement4	0	0	1	1	0	0	0	0
Note : la table de vérité a volontairement été réduite. Il apparaît juste les différents événements possibles.								

Tableau 5: Table de vérité de la gestion des interruptions

Il y a saturation pour les événements 1, 3 et 4 :

En logique cela se traduit par : si [add1_sat **ou** add2_sat **ou** add3_sat **ou** add4_sat **ou** add5_sat **ou** add6_sat **=1]** ou [resetn = 0] alors il y a saturation. Le schéma de la Figure 55, représente cette logique.

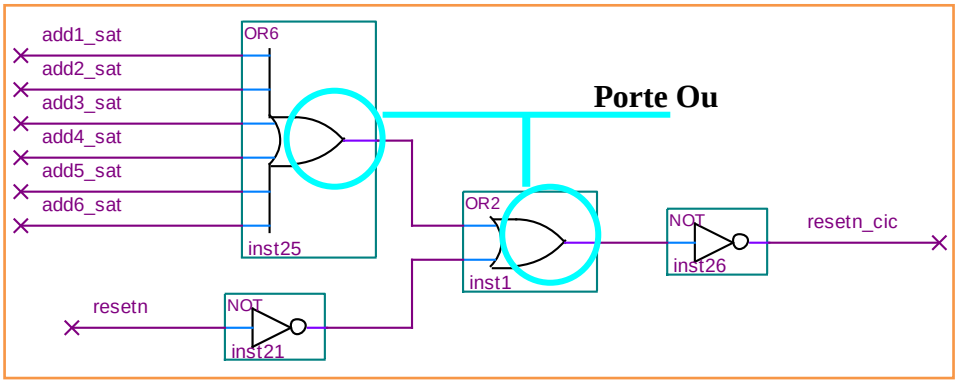


Figure 55 : Schéma de gestion de l'interruption et de la saturation.

4.3.5 Les sorties

En sortie du filtre, les données sont sous la forme de bus de 23 bits signés. Ce bus est transformé en bus de 32 bits en sortie dû à une standardisation voulue des filtres présents dans la librairie spécialisée. En sortie, il reste donc à construire un bus de 32 bits à partir du bus de 23 bits.
Le bit de poids fort du bus de 23 bits porte le signe. Pour passer à une configuration 32 bits, il suffit de court-circuiter les bits restant sur le 23ème bit, ce qui revient à repousser le bit de signe au bit de poids fort du bus 32 bits. L'intérêt est de ne pas perdre le bit de signe.

100 1101 0010 0010 1010 0001	Bus de 23 bits
1111 1111 1100 1101 0010 0010 1010 0001	Bus de 32 bits

Propagation du signe sans modifier la valeur

Par ailleurs les horloges sont également reconduites à la sortie pour pouvoir être utilisées pour le prochain filtre. Il s'agit de l'horloge de 4.8MHz et de l'horloge de synchronisation de 96kHz.

4.3.6 Le bloc diagramme du CIC

La figure 56 présente le schéma final du filtre CIC, et son symbole qui sera utilisé dans le projet final du FPGA. Il est à noter que dans le cas d'une saisie textuelle c'est-à-dire en utilisant le langage VHDL, le symbole serait le même.

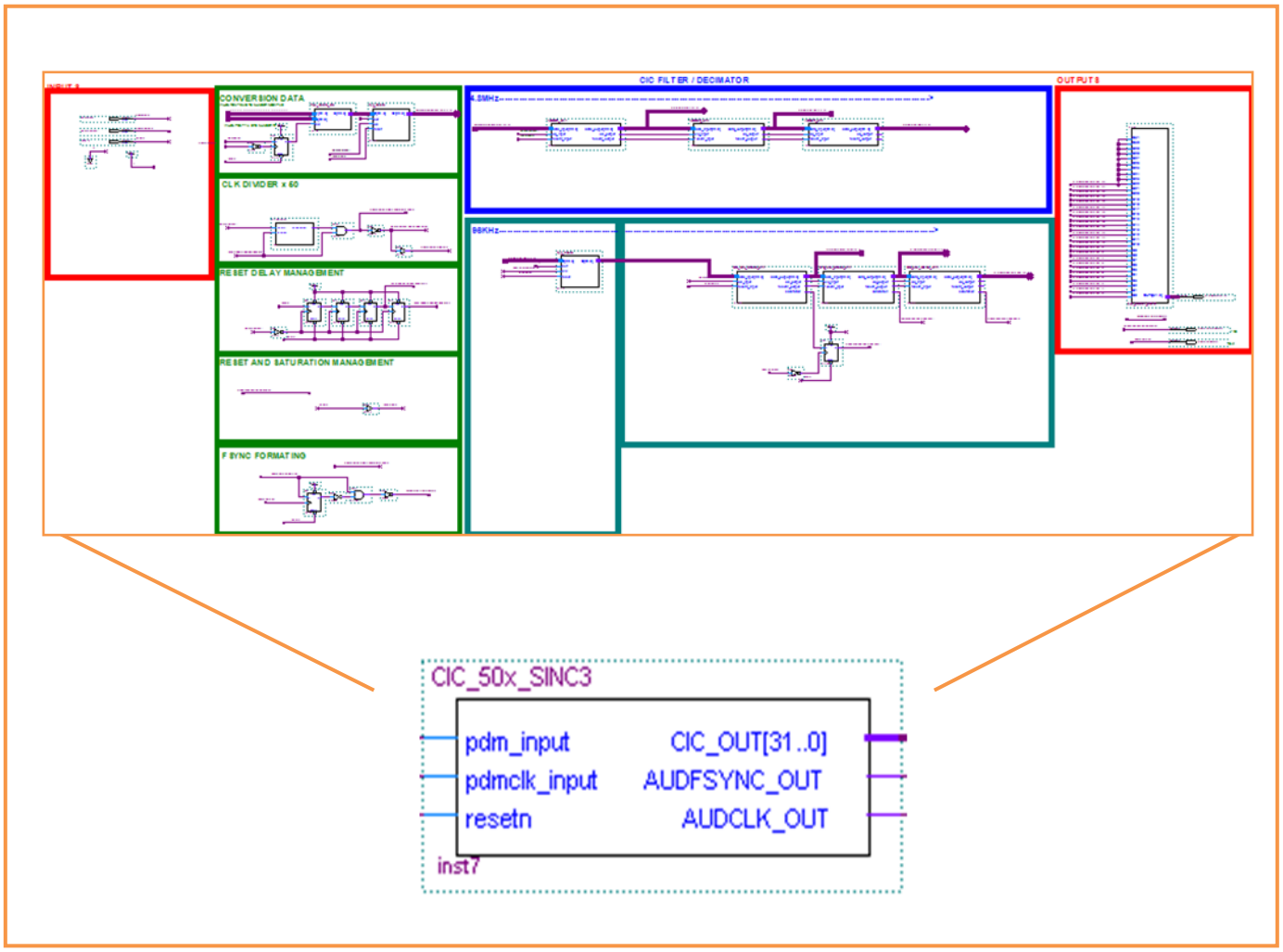
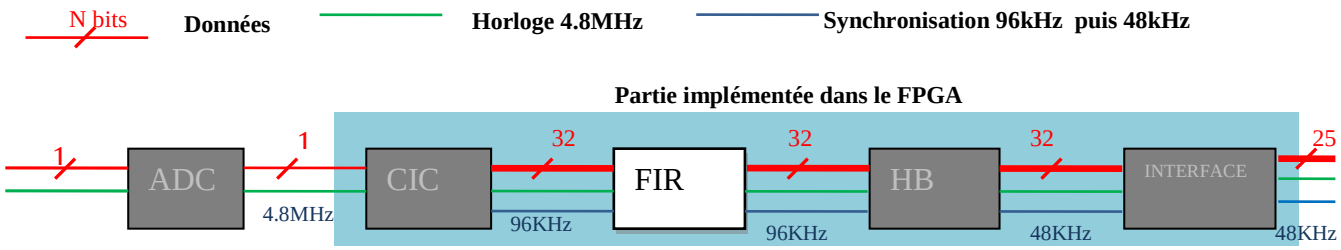
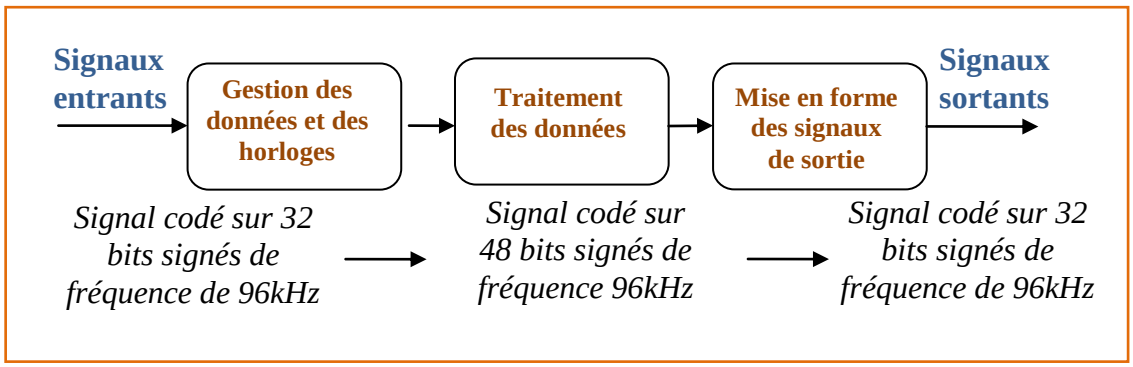


Figure 56 : Le bloc diagramme du Filtre CIC et son symbole

4.4 Conception de l'égaliseur



Contrairement au précédent filtre, celui fait appelle à des coefficients. Leur implémentation se fait grâce à des multiplieurs. Voici la description de la conception de ce filtre, avec l'explication des calculs des coefficients.



4.4.1 Les entrées

En entrée de ce filtre, nous trouvons le signal codé sur 32 bits parallèles signés de fréquence 96KHz du filtre précédent.

- **EQFIR_IN :** Signal de données codées sur 32 bits parallèles et signés.
- **AUDSYNC_IN** : Signal d'horloge de fréquence de 96KHz obtenue par division de la fréquence d'origine lors du précédent filtre.
- **AUDCLK_IN** : Signal d'horloge correspondant à la fréquence d'origine de 48MHz.
- **resetn** : Signal codé sur 1 bit, soit « 0 » soit « «1 » actif à l'état bas. Ce signal permet de réinitialiser le montage, lorsque cette entrée est à « 0 ». Pour que le système fonctionne, il faut la mettre à « 1 ».

4.4.2 Construction d'un bus de 48bits

Le but de ce filtre est de compenser l'atténuation subite par le signal par le précédent filtre. Pour cela, il a été pourvu de coefficients qui servent à pondérer davantage les bits de poids forts correspondant aux fréquences hautes : filtre passe haut. Ces coefficients ont pour conséquence d'accroître le nombre de bits du signal. Afin de prévoir cette augmentation, le bus de 32 bits est transformé en un bus de 48bits. Ce nombre est surévalué par rapport au résultat de simulation. Mais

cela ne pose pas de problème dans cette étude, car la mémoire du FPGA utilisée est suffisamment importante. Dans un souci d'optimisation, ce nombre sera calculé au plus juste, soit 24 bits, lors de la conception en circuit réel. Afin de passer d'un bus de 32 bits a 48 bits, la méthode utilisée est de propager le bit de poids fort (bit de signe) jusqu'au 32eme bit.

0000 0001 1100 1101 0010 0010 1010 0001	Bus de 32 bits
0000 0000 0000 0000 0001 1100 1100 1101 0010 0010 1010	Bus de 48 bits

Propagation du signe sans modifier la valeur

4.4.3 Gestion de remise à zéro et gestion de la saturation

Comme pour le filtre précédent, il y a une gestion de remise à zéro prévue en cas d'anomalie ou de saturation. Le même montage est réutilisé dans ce filtre.

4.4.4 Le filtre

Le fonctionnement du filtre FIR est synchronisé sur l'horloge de 96KHz. La fréquence d'origine de 4.8MHz passe directement en sortie sans être utilisée.

Délimitation des zones de fréquence

La figure 57 présente la structure de la séparation des zones de fréquence contenant des bascules D. Cette séparation permet de pondérer de manière distincte les différentes zones.

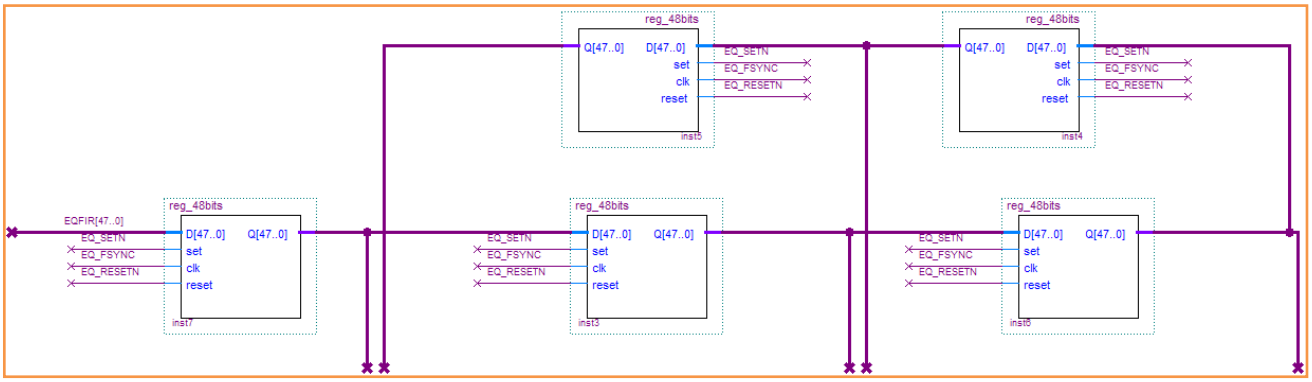


Figure 57 : Bloc diagramme des séparations des zones de fréquence

L'idée est partager la fonction de transfert en cinq zones de fréquence, d'où les cinq coefficients. Comme le montre la figure 58.

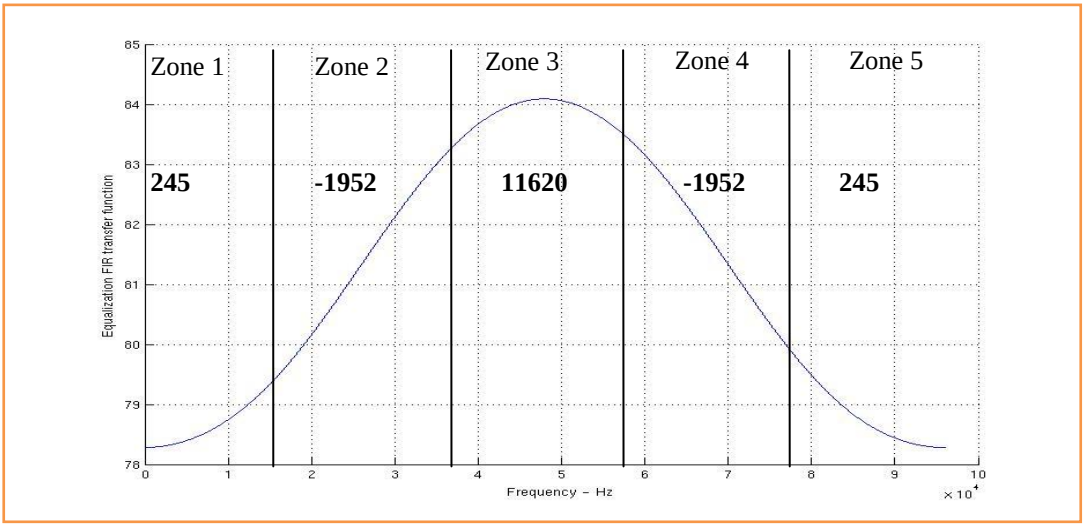


Figure 58 : Délimitation des zones de fréquence

Maintenant que les zones sont spécifiées, il faut leur attribuer un coefficient.

Méthode de conception des coefficients :

Adaptation des coefficients à un mot de 48bits.

L'objectif de ce calcul est de définir les valeurs des multiplieurs. Pour faciliter ces calculs et rendre les multiplieurs adaptables au bus de 48 bits, une simplification sur la base de l'octet à été nécessaire. L'octet, soit 8 bits est souvent utilisé en électronique numérique permettant une meilleure lisibilité du mot. Les coefficients vont alors être écrits sur plusieurs octets. Pour un octet, le plus petit nombre est « 0 » (représenté par huit « 0 » 0000 0000), et le plus grand est « 255 » (représenté par huit « 1 » 1111 1111), ce qui représente 256 possibilités de valeurs différentes. Cela permet d'adapter le coefficient au mot de 48 bits utilisé.

2^13	2^12	2^11	2^10	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

245 = 256 + (– 8)+(– 4) + 1 = 2^8 – 2^3 – 2^2 + 2^0
1952 = 2048 + (– 64)+(– 32) = 2^11 – 2^6 – 2^5
11620 = 8192 + 4096 + (– 512) + (– 128)+(– 32) + 4 = 2^13 – 2^12 – 2^9 – 2^7 – 2^5 + 2^2

Principe du décalage lié à la multiplication binaire

La pondération s'effectue grâce à des multiplicateurs qui ont pour effet de décaler les bits vers la gauche. Voici le principe:
Par exemple, deux bits suffisent à coder « 3 » en binaire **11**, il s'agit des deux bits de poids faible :

2^3	2^2	2^1	2^0
8	4	2	1
0	0	1	1

Le résultat de la multiplication décimale 3x2 donne « 6 ». En binaire, trois bits sont nécessaires pour coder le nombre « 6 », **110**, il y a un décalage vers la gauche.

2^3	2^2	2^1	2^0
8	4	2	1
0	1	1	0

En multipliant par « 2 » le chiffre « 3 », il y a un décalage en binaire qui est observé, un zéro **« 0 »** permet ce décalage vers la gauche. C'est cette propriété qui va être utilisée pour pondérer les différentes fréquences.

Ce filtre fait appelle à cinq coefficients de pondération, la symétrie de la fonction de transfert implique que certains de ces coefficients sont de valeurs égales. Soit A les différents coefficients :

A = [245 -1952 11620 -1952 245]

Conception des multiplicateurs

Les multiplicateurs sont des ressources limitées dans le FPGA contrairement aux cellules logiques majoritairement présentes. C'est pourquoi, ce sont elles qui sont utilisées pour permettre la multiplication, dont la méthode est déterminée par le principe de décalage. Le fait de multiplier par 8192, soit par 2^ **13**, revient à décaler vers la gauche la valeur d'entrée, en rajoutant **13** zéros au niveau des bits de poids faibles.

1010 1001 1100 1001 1001 1111 0001 0001 0011 1111 0001 1001 **+ 0000000000000**

Ajout de 13 « 0 » qui modifient la valeur

Gestion des multiplicateurs de signes négatifs :

Pour tenir compte du signe, un complément à deux est effectué, lorsqu'il s'agit d'un nombre négatif. Le signe est géré par le bit « negate ».

- Si le bit « negate » = 1 alors il s'agit d'une soustraction donc il y a calcul du complément à deux du résultat.
- Si le bit « negate » = 0 c'est une addition

Calcul du coefficient:

Une fois que chacun des multiplicateurs a été calculés. Il faut les additionner entre eux pour retrouver la valeur du coefficient attendue. Lorsqu'il s'agit de multiplicateur positif, c'est un additionneur qui est utilisé, mais s'il s'agit d'un multiplicateur négatif, soit un soustracteur. Ce qui revient à utiliser un additionneur avec le complément à deux du multiplicateur sans oublier la retenue, celle-ci est prise en compte seulement quand le bit « negate » vaut 1.

Voici le bloc diagramme du coefficient 245 = 256 + (– 8) + (– 4) + 1 = 2^8 – 2^3 – 2^2 + 2^0

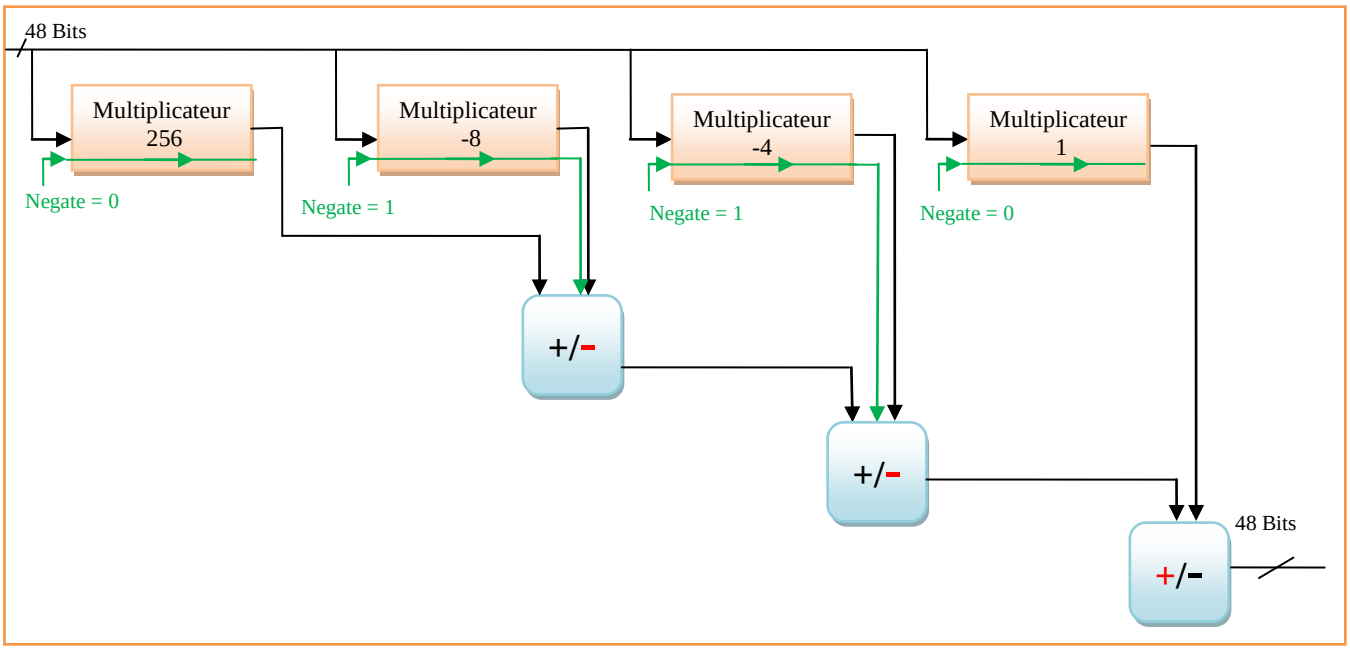


Figure 59 : Bloc diagramme du coefficient 245

Calcul des coefficients

De même que pour les multiplicateurs, les coefficients sont également additionnés ou « soustraits » selon leur signe. La méthode est identique aux calculs des multiplicateurs, avec également l'utilisation du bit « negate » égale à 1 qui lance le complément à deux.

4.4.5 Les sorties

Afin de standardiser le filtre, le nombre de bit en sortie de celui-ci est ramené à 32. Les bits de poids forts propageant le bit de signe, il est possible de les abandonner sans modifier la valeur du bus.

0000 0000 0000 0000 0001 1100 1100 1101 0010 0010 1010 Bus de 48 bits

0000 0001 1100 1101 0010 0010 1010 0001 Bus de 32 bits

Propagation du signe sans modifier la valeur

4.4.6 Le bloc diagramme du FIR

La figure 60 présente le schéma final du filtre FIR.

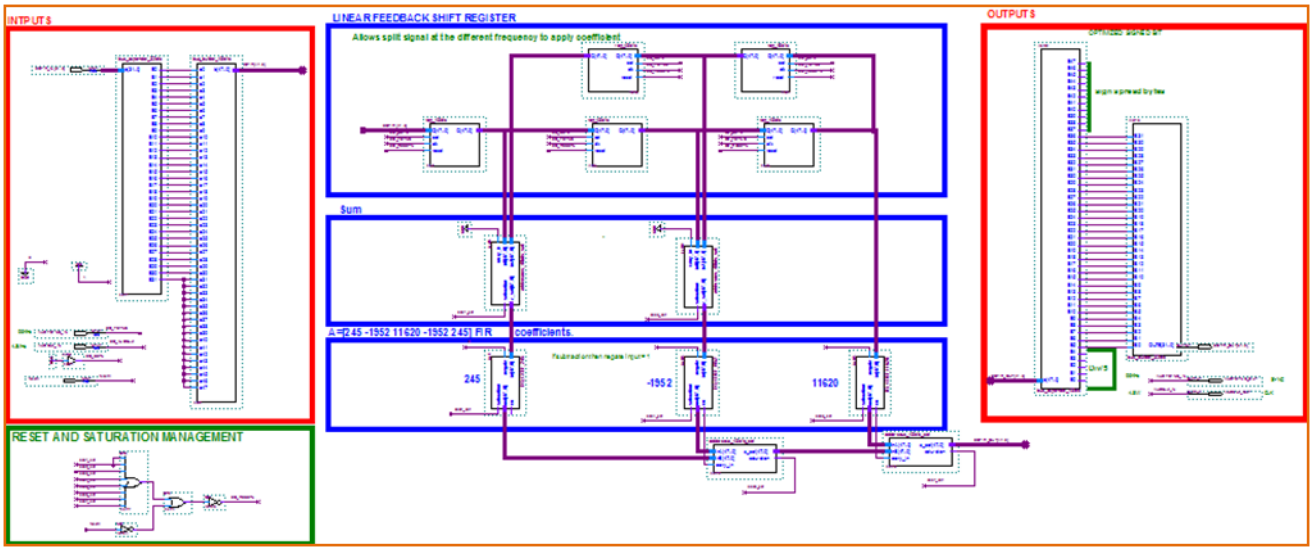


Figure 60 : Bloc diagramme du filtre FIR

4.4.7 Bilan de la saisie des deux premiers filtres

La saisie des deux premiers filtres est terminée. La saisie du dernier filtre FIR passe bas, ne se fera qu'à la suite de la vérification des deux premiers filtres. Son développement est plus complexe. La méthode de conception reprend la même stratégie du CIC pour la décimation et du FIR pour la gestion des coefficients. Une fois la méthode validée, elle pourra être réutilisée pour le filtre FIR passe bas.

Pour vérifier ces premiers développements, il faut intégrer ces filtres dans le projet existant dans un premier temps, puis lancer la compilation et la programmation du FPGA dans un deuxième temps.

4.5 Implémentation des filtres

Les filtres vont être implémentés dans la partie commune du projet c'est-à-dire dans le cœur du système. Ils pourront ainsi être appelés par le projet Stratix ou le projet Cyclone.

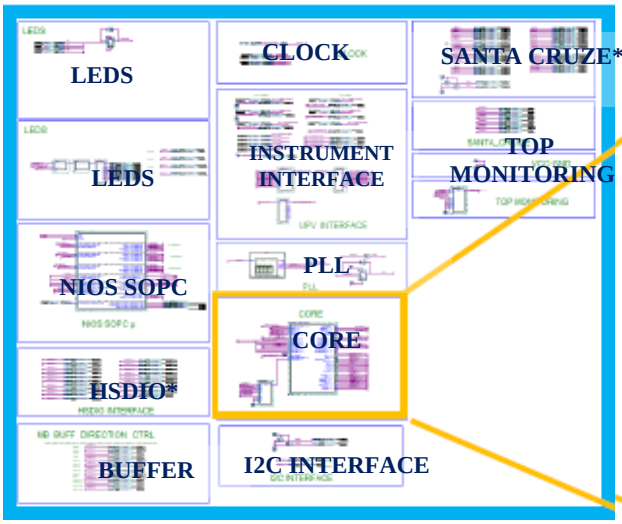
La partie commune a été structurée selon les fonctions qu'elle intègre. Il y a une zone spécifique pour les montages numériques audio(1). Le symbole principal porte le nom de TX_DIGITAL_AUDIO_TOP. C'est à l'intérieur de celui-ci que sont « rangés » les circuits de prototypage à destination du circuit ADC (2).

A l'intérieur de celui-ci, on trouve donc le symbole TX_DIGITAL_AUDIO_MONO. Il est à noter qu'il s'agit d'une chaîne de traitement numérique à destination d'une application audio fonctionnant en stéréo. Ce symbole est appelé deux fois, pour chaque coté soit droite et gauche (3).

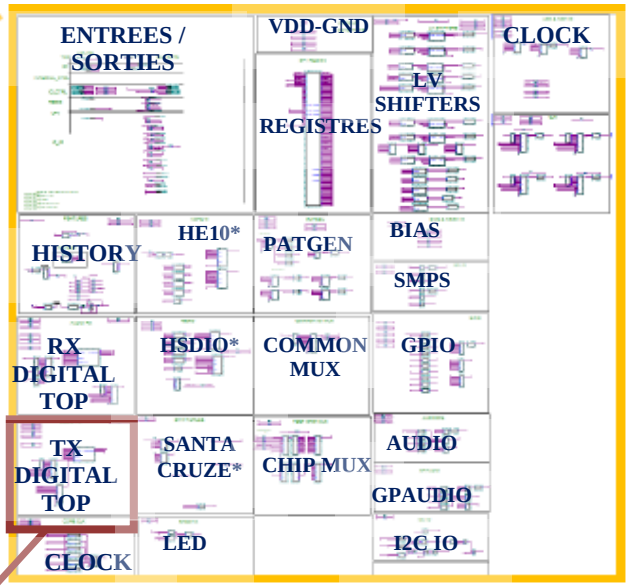
Lorsque l'on rentre dans ce symbole, on aperçoit la chaîne audio avec la présence de chacun des filtres. Cette organisation a été faite durant le stage. Par conséquent, c'est à cet endroit que les filtres CIC et FIR vont être implémentés (4).

La page suivante présente l'architecture du projet dans laquelle le filtre CIC à été implémenté.

Partie spécifique
Stratix ou Cyclone



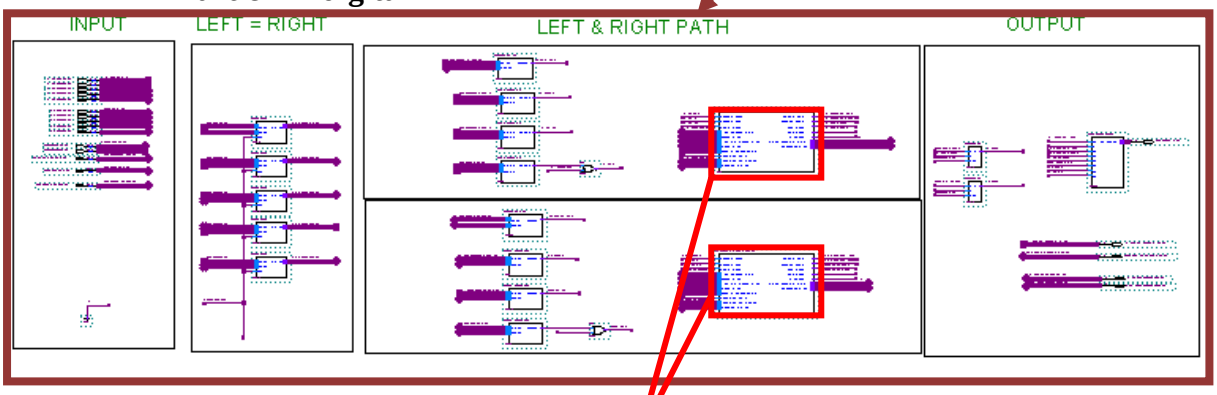
Partie Cœur du système



(1)

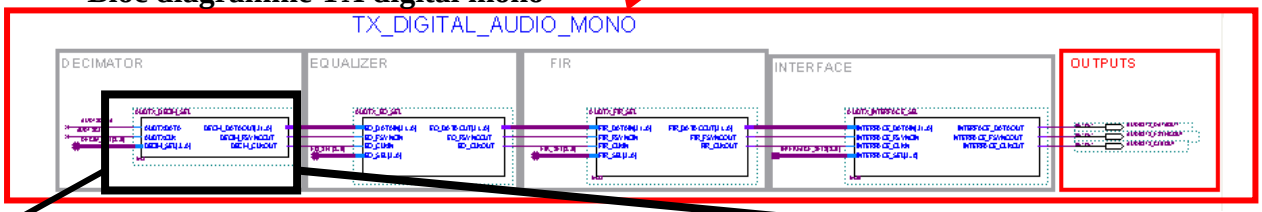
Partie TX digital

(2)



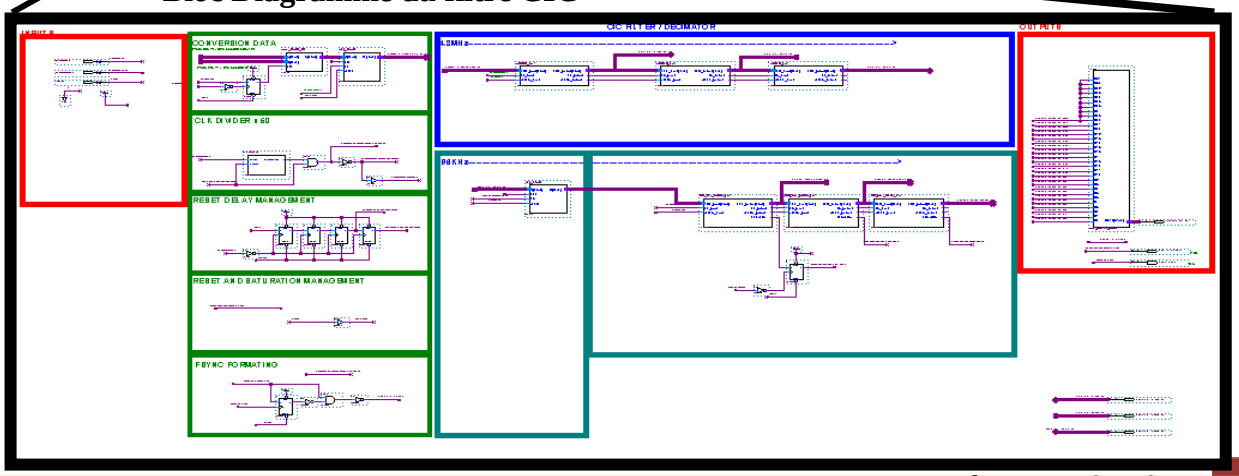
Bloc diagramme TX digital mono

(3)



Bloc Diagramme du filtre CIC

(4)



4.6 Compilation de ces conceptions

Une fois les blocs diagrammes terminés, vient l'étape de la compilation. Durant cette phase, le logiciel Quartus va réaliser plusieurs tâches de manière séquentielle.

1. **La Synthèse logique** : C'est l'étape de la transformation du programme en un schéma électronique à base de portes logiques. Même dans le cas de la saisie graphique, Quartus retrace son propre schéma contenant des portes et des bascules.

2. **La synthèse physique** : Elle permet d'utiliser les portes logiques selon les ressources matérielles disponibles du circuit et met en place les réseaux d'interconnexions.

3. **L'assemblage :** Cette étape produit les fichiers (programme transformé en valeurs binaires) permettant la programmation de la mémoire du circuit.

En cas d'erreur, ces processus s'arrêtent, et il apparaît dans une fenêtre de dialogue, les sources des erreurs. Tant quelles ne sont pas toutes corrigées, la compilation ne pourra pas effectuer complètement. Ce qui implique qu'aucun fichier de programmation ne sera généré.

4.7 Programmation du FPGA

La programmation est l'étape où le programme est envoyé dans le FPGA. Pour cela il existe plusieurs méthodes permettant de transférer les données à la carte cible où est câblée le FPGA. Ces méthodes nécessitent un logiciel « programmeur » et une solution d'interface entre le logiciel et la carte de développement du FPGA. Dans le cadre du projet, c'est l'interface JTAG[32] « Joint Test Action Group », qui est utilisée. Le FPGA est composé d'IP qui comprennent le JTAG. (Lors de la conception du NIOS). Ce sont elles qui réceptionnent le programme envoyé par cette interface.

4.7.1 L'interface

L'interface utilisée entre le PC et la carte de développement est un boîtier USB Blaster, Figure 61 : Boîtier USB Blaster d'Altera. C'est une solution que le constructeur Altera à mise en œuvre pour la programmation par port JTAG des composants FPGA qu'il commercialise. Il n'est ni plus ni moins qu'une interface JTAG pour port USB du PC. Il se connecte sur la carte soit par connexion USB soit par un connecteur JTAG à 10 broches.

[32] JTAG : Le terme désigne le groupe de travail qui a conçu cette norme, le vrai terme est Bundary Scan (scrutation des frontières) ou le sigle TAP pour Port d'Accès de Test.

Figure 61 : Boîtier USB Blaster d'Altera

4.7.2 Le logiciel programmeur

Le logiciel programmeur fait parti des applications fournis avec l'environnement de programmation Quartus d'Altera. Il suffit de spécifier l'interface et le port utilisé, soit l'interface JTAG par le port USB du PC. Puis il faut récupérer le fichier généré lors de l'étape de compilation. Enfin il suffit de lancer la programmation. Celle-ci ne prend quelques secondes. Le fichier vient se fixer dans une mémoire interne volatile du FPGA. Cela implique qu'à chaque mise sous tension, il faut le reprogrammer. Pour pallier ce problème, une mémoire non volatile mais, externe type Flash peut contenir le programme.

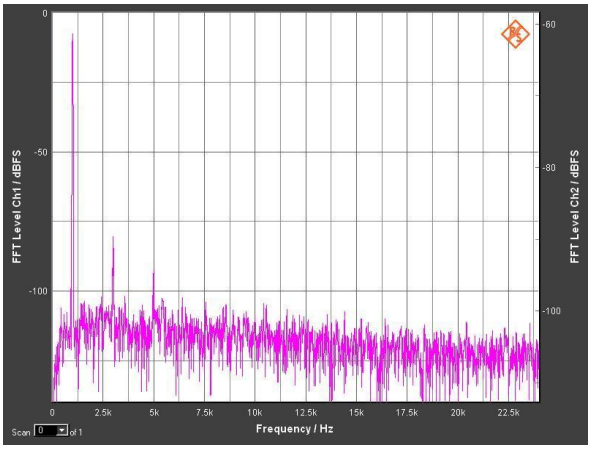
Les filtres sont maintenant placés au cœur de l'environnement de test, prêts à exploiter les signaux en sortie du convertisseur. Cependant, une dernière étape indispensable consiste à vérifier les comportements de ces filtres. C'est durant cette phase que le FPGA présente son plus grand intérêt. La correction du développement se fait « en temps réel ». C'est-à-dire qu'il est facile et rapide de rectifier les paramètres des filtres pour satisfaire à la demande, sans pour autant devoir attendre le temps de la fabrication d'un prototype. La prochaine partie présente cette étape de vérification.

5 LA VERIFICATION

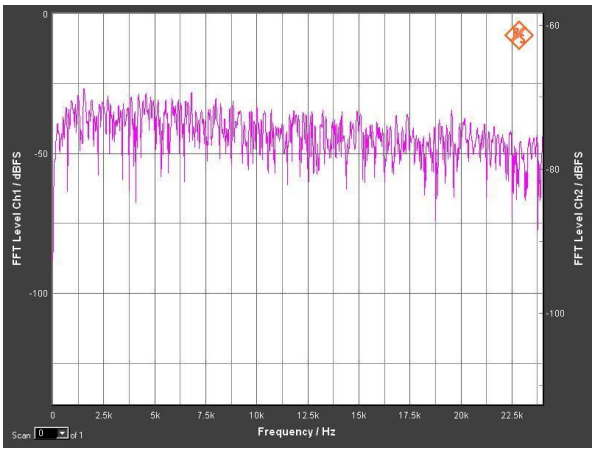
La vérification de fonctionnement de la chaîne est une étape non-négligeable de la conception. Elle permet de valider les paramètres des filtres donnés par le concepteur et de corriger les éventuels dysfonctionnements. La première phase consiste à observer le spectre obtenu en sortie de la chaîne. Le spectre doit correspondre au signal sinusoïdal généré en entrée. Une seconde phase propose de présenter les principes de correction utilisés durant le stage. Un bilan final permet de faire une analyse de ces résultats et de la méthode de conception utilisée.

5.1 Observation des résultats

Voici le spectre du signal attendu en sortie de la chaîne. Il s'agit du spectre correspondant à la sinusoïde injectée en entrée à la fréquence décimée de 48kHz.



Pourtant, lors de la mise en place des filtres, le signal de sortie ne ressemble plus au spectre attendu. Lors du traitement des données, l'information a été perdue. Les blocs diagrammes des filtres vont être analysés afin de comprendre d'où vient le problème.



5.2 Les dysfonctionnements

5.2.1 Les erreurs de saisie

Malgré la validation complète de la compilation, certaines erreurs sont encore présentes dans le schéma. La compilation ne vérifie que la syntaxe de développement. Un contrôle attentif a permis de corriger beaucoup de petites erreurs souvent liées à une mauvaise connexion des nets ou à une faute dans les noms des signaux. Lorsque le bloc diagramme devient complexe, le schéma se densifie, ce qui amène des difficultés de lisibilité et augmente le risque d'erreur de ce type.

5.2.2 L'analyse temporelle

L'analyse temporelle d'un montage numérique est indispensable lors de la vérification de fonctionnement. Elle repose sur le calcul et l'addition des délais de chaque porte logique élémentaire d'un circuit. Elle permet ainsi de vérifier que les données reçues par un élément logique sont stables au moment où celui-ci reçoit un coup d'horloge. Pour vérifier que tous les délais ont été respectés, il faut observer en détails les comportements des signaux internes et en sorties des filtres.

Il est possible que les calculs qui ont été faits au sein des filtres ont subit une perte d'information due à des données manquantes en entrée des additionneurs du fait de leur « arrivée tardive ».

5.3 Les techniques de vérification

Le but de cette vérification est de trouver à quel endroit des blocs diagrammes il y a eu perte d'information, ce qui a pour conséquence l'absence de spectre en sortie. La méthode utilisée pendant le stage consiste à corréler les résultats obtenus avec les résultats attendus fournis par le concepteur. Deux outils ont été mis à ma disposition pour mener à bien cette vérification.

5.3.1 La simulation sous Quartus

Un outil de simulation fonctionnelle et temporelle est fournit avec l'environnement de programmation Quartus. Il permet d'obtenir un chronogramme de fonctionnement. Pour pouvoir simuler le schéma, il faut isoler le bloc diagramme d'un filtre et le placer dans un banc de test virtuel. Ce banc de test est un nouveau projet. Son principe est d'injecter des stimuli aux entrées de la fonction testée et de contrôler les signaux de sortie. Les stimuli peuvent être sous forme de signaux statiques, comme une valeur DC ou de signaux dynamiques comme une liste de valeurs programmées dans une ROM, représentant un signal PDM ou un Sinus. Les résultats de cette simulation seront à comparer avec les résultats attendus par le concepteur.

5.3.2 La simulation sous LabVIEW

Pour compléter cette analyse, un autre outil a été proposé, il s'agit de LabVIEW. Ce logiciel est disponible au laboratoire. Dans le cadre du projet, ce logiciel a été utilisé pour simuler un banc de test virtuel représentant les montages numériques du stage. Il permet d'aider à la conception et la simulation de fonctions numériques.

Son grand avantage est que les paramètres des montages sont modifiables en temps réels. Par ailleurs, l'observation de signaux de sortie peut se faire sous la forme de graphique et sous la forme de tableau. LabVIEW met ainsi à disposition un grand nombre d'indicateurs et d'outils mathématiques qui aide à la compréhension du fonctionnement des filtres. Cet outil a permis de concevoir le filtre CIC comme il a été conçu dans Quartus. Pour se rapprocher des comportements réels seuls des composants de la logique numérique ont été utilisés. Ce qui permet d'effectuer les mêmes modifications dans Quartus.

L'inconvénient de cette analyse est qu'elle ne prend pas en compte l'aspect temporel du fonctionnement. C'est un système idéal. Cette simulation ne sera utilisée que pour la compréhension des comportements.

5.3.3 Résultats

Les filtres CIC et FIR font subir de nombreux calculs au signal de sortie du convertisseur. La méthode utilisée revenait à comparer les valeurs obtenues aux valeurs données par le concepteur en sortie de chaque étage. La figure 62 présente la simulation temporelle en sortie des intégrateurs. Les premières valeurs ne correspondent pas aux résultats spécifiés par le concepteur.

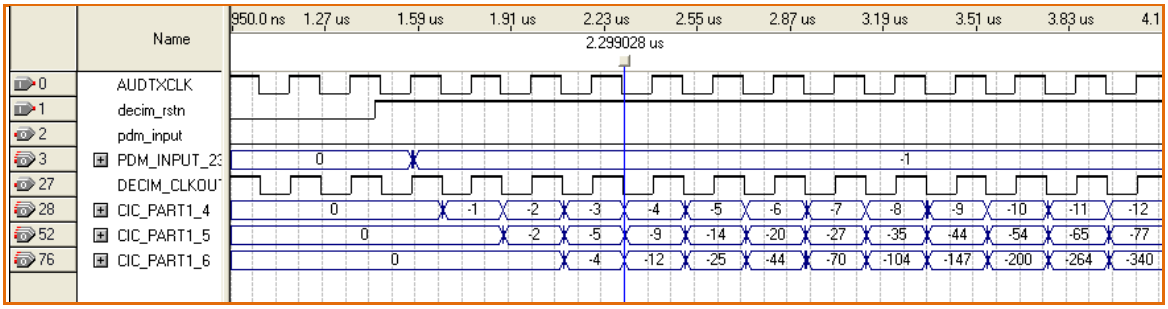


Figure 62 : résultats de simulations des étages intégrateurs du filtre CIC

Cette vérification a soulevé plusieurs problèmes liés à la conception des montages.

- La conception des filtres s'est faite sans mise en œuvre de contraintes temporelles. Pourtant cette analyse est indispensable. Elle sert à veiller au respect du temps de propagation maximal d'un point à un autre du circuit. C'est ce qui a pu engendrer les erreurs de calculs car toutes les données ne sont pas prises en compte dans le même cycle d'horloge.

- Le comportement des schémas est vérifiable grâce à l'outil de simulation de Quartus. Cette vérification aurait dû être effectuée durant la phase de conception des filtres et non à la fin de celle-ci. Ce qui aurait permis de connaître rapidement le point sensible de ce montage grâce à une correction progressive.

- La complexité des schémas actuels rends ingérables cette vérification. Une des solutions consiste à contrôler ces filtres blocs après blocs en ajoutant petit à petit des bascules qui serviraient à resynchroniser les données. La figure 63 présente une des solutions utilisées pour prendre en compte les résultats. L'idée est d'inverser le front de synchronisation des données. Ce qui laisse davantage de temps à la donnée d'atteindre son point d'arrivé.

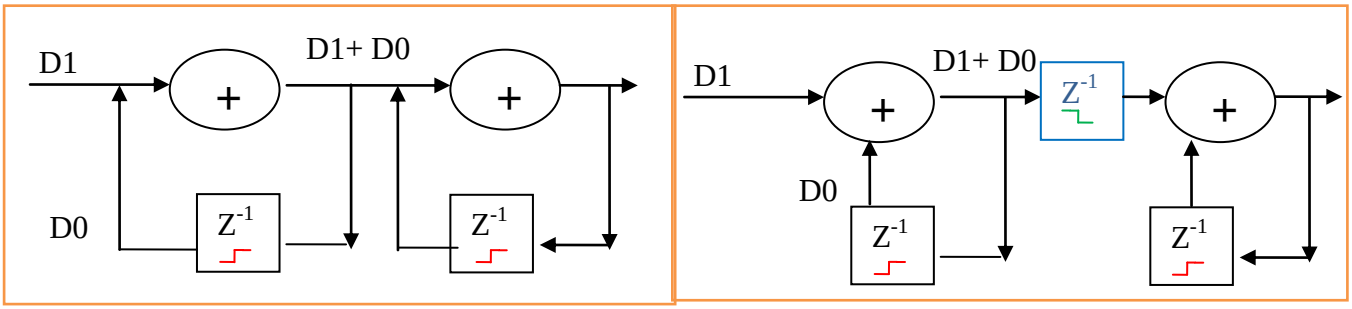


Figure 63 : Synchronisation des données grâce à l'ajout d'une bascule

Cette solution permet d'éviter la perte d'information comme le montre la figure 64, mais elle est susceptible de modifier la fonction de transfert du filtre. Le concepteur doit alors la recalculer pour voir si l'impact est important pour le fonctionnement de ce filtre.

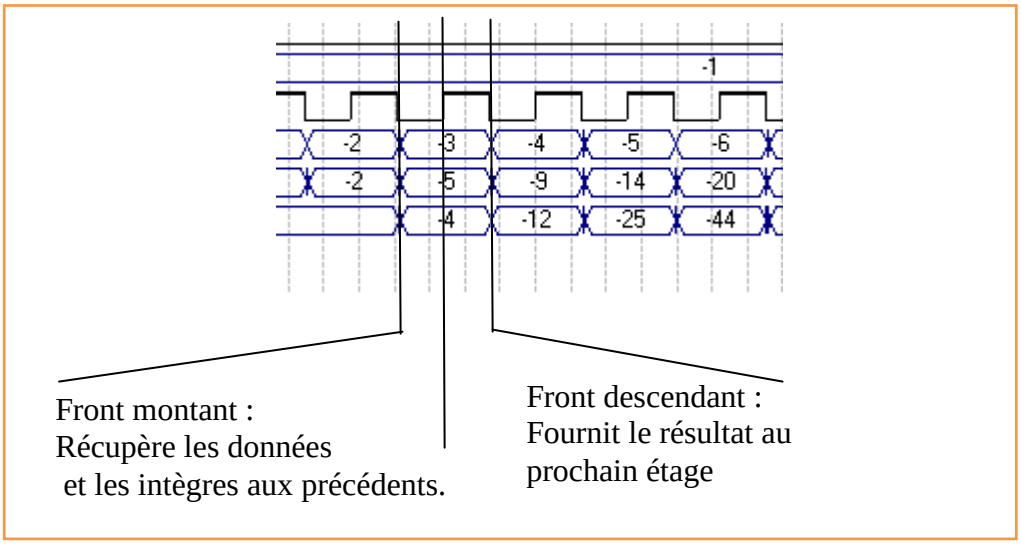


Figure 64 : Chronogramme de la bonne synchronisation

5.4 Proposition d'amélioration de la méthode

Malgré ces ajouts de bascules en divers point du schéma, le résultat de simulation ne satisfait toujours pas aux valeurs attendues. Les schémas des filtres sont trop complexes pour être corrigé ainsi. Cette vérification prouve que cette méthode de conception présente des limites de performance. La prise en compte de l'aspect temporel n'est pas évidente. C'est pourquoi il serait préférable de passer à une autre technique de conception.

Le mode de développement qui a été demandée pour mettre en œuvre cette solution est la saisie graphique proposée par l'environnement de programmation Quartus. C'est une technique très pédagogique et un très bon exercice pour appréhender la conception numérique. Cependant, elle présente de nombreux inconvénients dont voici les principaux :

- La complexité du schéma devient rapidement trop élevée. Il en résulte un développement difficile avec un risque d'erreur de saisie important et une exploitation laborieuse et délicate du bloc diagramme une fois terminé.

- La non-portabilité de la saisie schématique due à sa librairie spécifique empêche l'utilisation d'outil tel que Modelsim. Pourtant ce simulateur de description matérielle améliore le travail de vérification. Par exemple, il est possible d'observer un signal à l'intérieur même d'un symbole. Ce qui n'est pas possible avec l'outil de simulation de Quartus. La technique de contournement permet de sortir le signal à contrôler mais cela aurait comme incidence l'ajout de temps de propagation des données. Le résultat simulé disposerait alors d'une nouvelle erreur de timing.

Ces raisons font que cette méthode de conception va être remplacée à la suite du stage. Une autre méthode aurait été plus appropriée à la conception des filtres. Il s'agit de l'utilisation d'un langage standard et largement utilisé pour la programmation du FPGA, le VHDL.

La portabilité de ce langage est bien meilleure. De nombreux outils de simulations peuvent être utilisés. Ce qui permet de faire la comparaison entre les différentes analyses temporelles. De plus le VHDL est exportable aux FPGA de différents fabricants. La saisie schéma de Quartus n'était utilisable que pour la gamme de FPGA proposée par Altera.

Par ailleurs, il est possible d'ajouter des contraintes temporelles dans le VHDL sans pour autant modifier la fonction de transfert du système. Le VHDL est un langage complexe mais rigoureux. Sa programmation est rapide et structurée. En cas de besoin et pour vérifier les bonnes connexions, il est possible de se référer au fichier RTL[33] généré par l'environnement de conception. Ce fichier représente le programme sous forme de schémas blocs comme pour la saisie graphique.

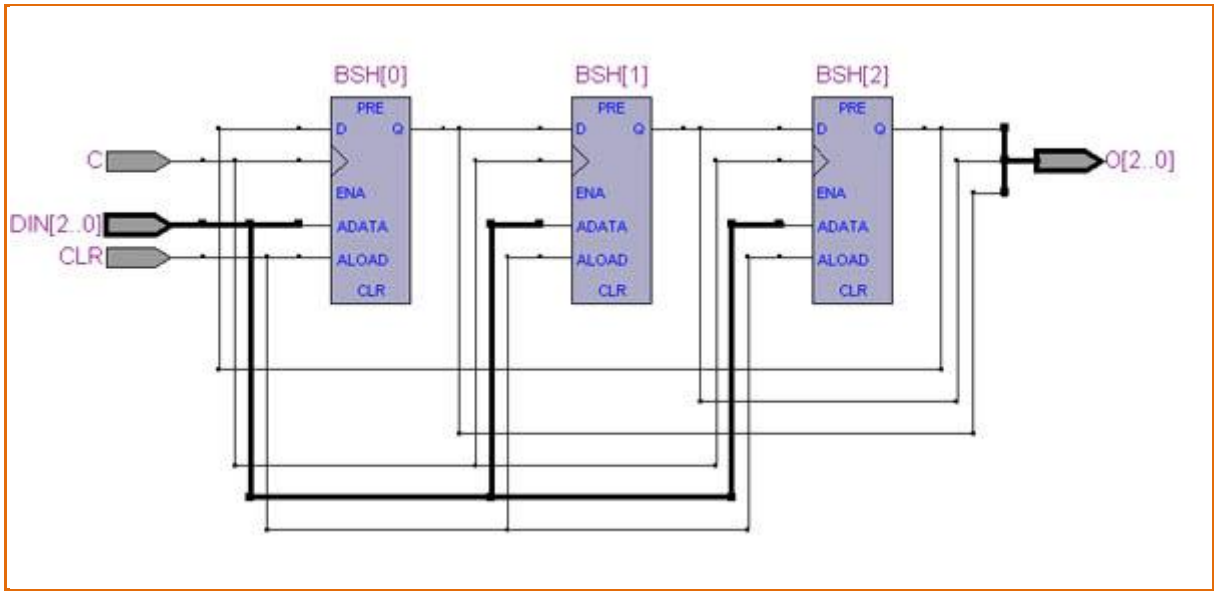


Figure 65 : Exemple de génération d'un fichier RTL

Enfin, les montages numériques fournis par le concepteur sont saisis sous le format VHDL. La connaissance de ce langage est indispensable pour apporter le support nécessaire à leur besoin de vérification en avance de phase.

Les étapes de conception reste les mêmes que pour la saisie graphique.

[33] RTL : Register Transfer Language

Conclusion

Ce stage s'inscrit dans le flot de conception d'une cellule numérique dont le but est de fournir des outils de validation au concepteur. L'approche traditionnelle consiste à proposer une solution permettant la vérification d'un prototype une fois fabriqué. Cependant cette technique présente la nécessité de relancer la fabrication d'un prototype à chaque modification du schéma du circuit. Pour cette raison, le projet du stage vise à proposer un nouveau type d'outil de validation qui permettrait de vérifier le fonctionnement d'un circuit en avance de phase. La fabrication du prototype ne se fera qu'à partir d'un schéma déjà vérifié.

Le travail répondait à un besoin précis d'un concepteur. Son circuit, un convertisseur analogique-numérique, génère des signaux inexploitables en l'état pour les applications clientes. Ils requièrent alors l'utilisation de filtres numériques en sortie. Le but de ce stage était de proposer une solution de vérification du montage des filtres. La mise en œuvre de cette réalisation consistait à concevoir leurs blocs diagrammes dans un FPGA en utilisant la méthode de saisie graphique. Le FPGA permet l'émulation de montages numériques. Sa reprogrammation quasi-instantanée offre la possibilité de régler en « temps réel » les paramètres des filtres afin d'obtenir un schéma fonctionnel avant leur implémentation en circuit final.

La réalisation de cette solution a nécessité d'atteindre différents objectifs :

- Le premier était de maîtriser les outils de conceptions du FPGA et de connaître l'architecture de ce type de circuit. L'étude de ses caractéristiques a montré que la solution actuelle ne pouvait pas suivre l'augmentation de la complexité des circuits à émuler. Le choix d'un FPGA à haute intégration a été inévitable.
- Le deuxième objectif a été d'intégrer ce nouveau FPGA dans l'environnement de test actuel. Ce qui a conduit à la création d'un nouveau programme de gestion du processeur interne. Une compréhension approfondie de l'environnement de test et l'étude de la structure du programme FPGA ont été essentielles.
- Enfin le dernier objectif était de concevoir les filtres numériques. Afin d'optimiser cette conception, il a fallu mettre en pratique de nombreuses théories étudiées en cours. En sortie du convertisseur, les filtres numériques forment une chaîne de traitement du signal, numérique dans notre cas. Ce qui explique les choix des filtres qui ont été fait. Par ailleurs, la connaissance des méthodes de conception numérique a été largement utilisée lors de la saisie des schémas.

Malgré ces acquis théoriques, la simulation des montages ont soulevés de nombreux dysfonctionnement qui à ce jour n'ont pas été résolus. En cause, la saisie graphique, bien que très

pédagogique, elle n'a pas permis de poursuivre la vérification des schémas conçus car ils sont trop complexes.

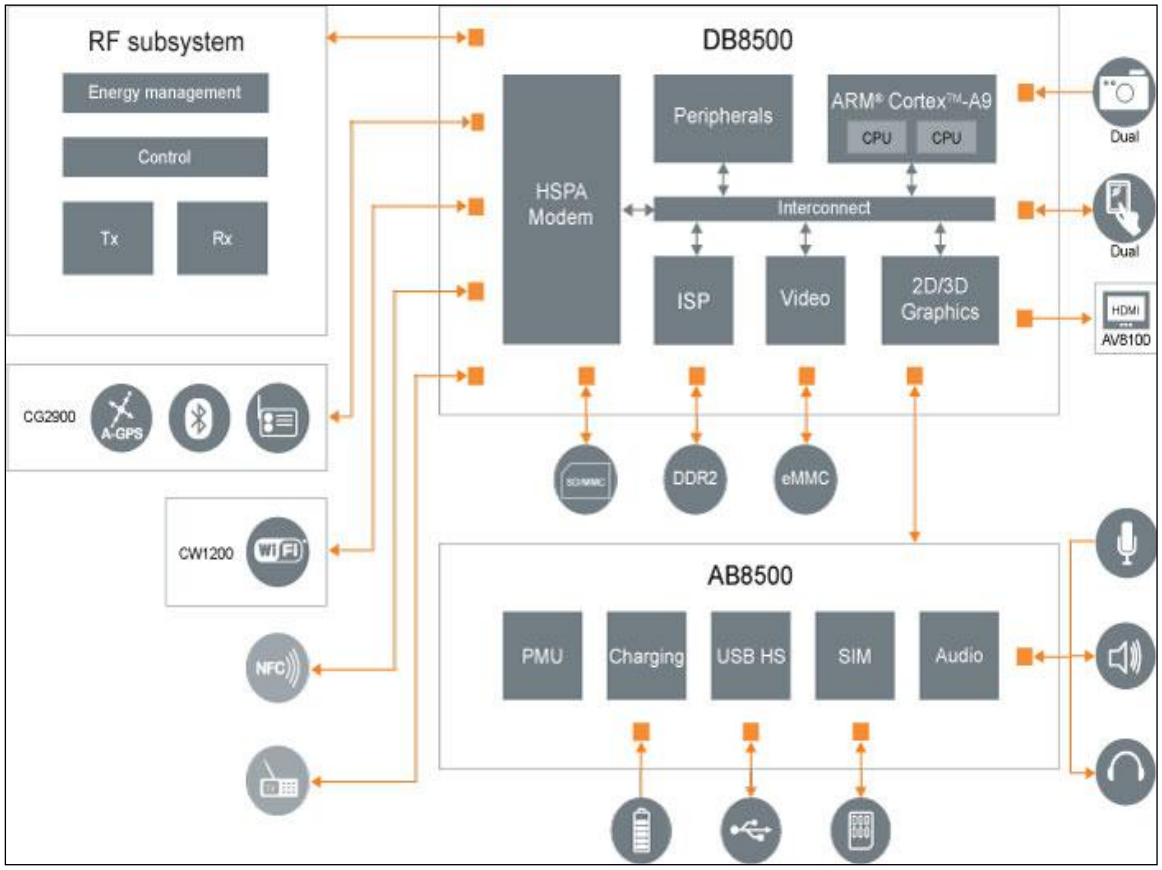
Les méthodes sélectionnées pour un projet ne sont pas toujours les bonnes. Il est important dans ce cas de remettre en question ces choix et d'évaluer l'impact de la mise en œuvre d'une nouvelle stratégie. J'ai la chance de pouvoir continuer ce projet dans le cadre de mon travail. J'ai étudié les conséquences de la modification de saisie dans la programmation des filtres. Dans un premier temps, je vais devoir me familiariser au langage VHDL et à l'analyse temporelle. Par conséquent, les délais de conception des filtres seront retardés, mais à long termes ces nouvelles compétences me permettront d'optimiser la conception de circuits futurs et de satisfaire au mieux aux demandes des concepteurs.

Annexe

Annexe 1 : Exemple de plateforme produit
Annexe 2 : Perception du bruit de l'oreille humaine
Annexe 3 : Les infrasons et les ultrasons
Annexe 4 : La transformée en z
Annexe 5 : Fonction de transfert dans le domaine fréquentiel du filtre CIC
Annexe 6 : Fonction de transfert dans le domaine fréquentiel du filtre FIR
Annexe 7 : Fonction de transfert dans le domaine fréquentiel du filtre FIR passe bas
Annexe 8 : La validation d'un convertisseur
Annexe 9 : Le complément à deux

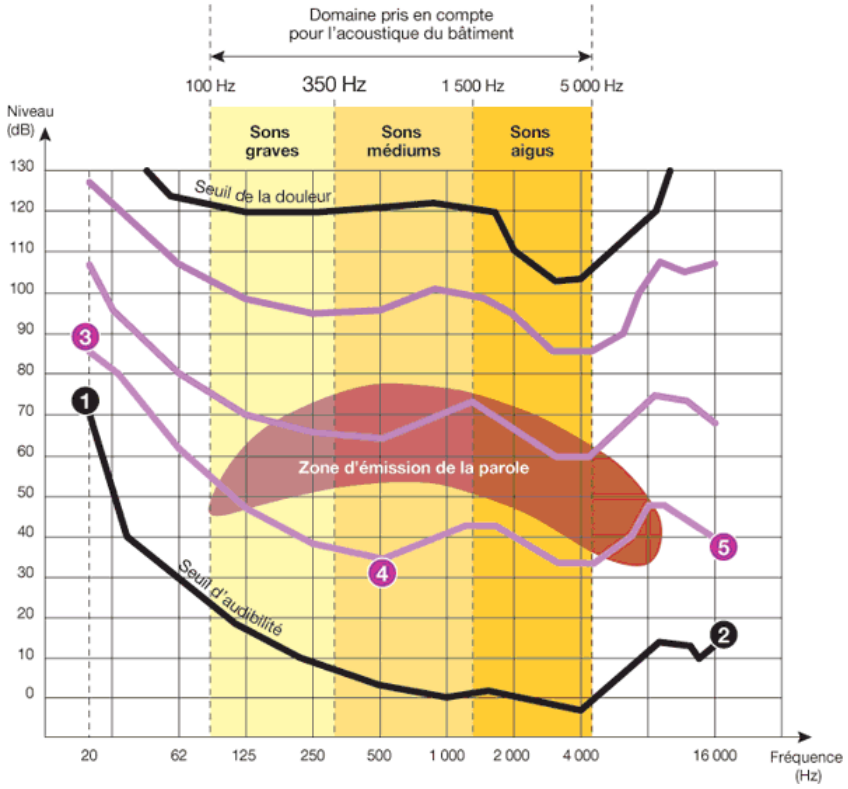
Annexe 1 : Exemple de plateforme produit

La figure suivante présente une plateforme d'un téléphone portable. Il y apparait les différents circuits conçus au sein de la division 3GP.



Plateforme d'un téléphone portable

Annexe 2 : Perception du bruit de l'oreille humaine

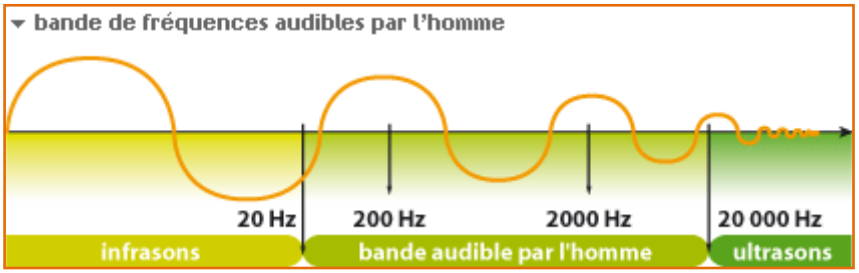


Courbe de sensibilité de l'oreille humaine

Source : http://www.spectra.fr/notions-norme-acoustique-r9.html#perception

Annexe 3 : Les infrasons et les ultrasons

Notre oreille est sensible aux vibrations entre 16 Hz et 20 000 Hz; en dessous de 16 Hz ce sont des infrasons que nous pouvons percevoir par la paroi abdominale. Au-dessus de 20 000 Hz, il s'agit d'ultrasons que seuls certains animaux perçoivent (chiens, chauve-souris, dauphins…).



Annexe 4 : La transformée en z

Soit un signal discret x(n).
La transformée en Z (bilatérale) est définie par :

$$X(z) = Z\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n}$$

Où z est une variable complexe et où X(z) est une fonction complexe de la variable z.

La transformé en z peut être considéré comme une généralisation de la transformation de Fourier à laquelle elle peut s'identifier dans un cas particulier.
La transformée en z constitue l'outil privilégie pour l'étude des systèmes discrets. Elle joue un rôle équivalent à celui de la transformée de Laplace.
Par exemple, la transformée en z permet de représenter un signal possédant une infinité d'échantillons par un ensemble fini de nombre.

	Signal $x(n)$	Transformée en Z $X(z)$	Domaine de convergence
1	$\delta[n]$	1	$\mathbb{C}$
2	$u[n]$	$\frac{1}{1-z^{-1}}$	$\lvert z\rvert > 1$
3	$a^n u[n]$	$\frac{1}{1-az^{-1}}$	$\lvert z\rvert > \lvert a\rvert$
4	$na^n u[n]$	$\frac{az^{-1}}{(1-az^{-1})^2}$	$\lvert z\rvert > \lvert a\rvert$
5	$-a^n u[-n-1]$	$\frac{1}{1-az^{-1}}$	$\lvert z\rvert < \lvert a\rvert$
6	$-na^n u[-n-1]$	$\frac{az^{-1}}{(1-az^{-1})^2}$	$\lvert z\rvert < \lvert a\rvert$
7	$\cos(\omega_0 n)u[n]$	$\frac{1-z^{-1}\cos(\omega_0)}{1-2z^{-1}\cos(\omega_0)+z^{-2}}$	$\lvert z\rvert > 1$
8	$\sin(\omega_0 n)u[n]$	$\frac{z^{-1}\sin(\omega_0)}{1-2z^{-1}\cos(\omega_0)+z^{-2}}$	$\lvert z\rvert > 1$
9	$a^n\cos(\omega_0 n)u[n]$	$\frac{1-az^{-1}\cos(\omega_0)}{1-2az^{-1}\cos(\omega_0)+a^2z^{-2}}$	$\lvert z\rvert > \lvert a\rvert$
10	$a^n\sin(\omega_0 n)u[n]$	$\frac{az^{-1}\sin(\omega_0)}{1-2az^{-1}\cos(\omega_0)+a^2z^{-2}}$	$\lvert z\rvert > \lvert a\rvert$

Annexe 5 : Fonction de transfert dans le domaine fréquentiel du filtre CIC

Pour l'étage intégrateur :

(6) $$H_I(z_I) = \frac{1}{1 - z^{-1}}$$

Pour l'étage différentiateur :

(7) $$H_C(z_C) = 1 - z^{-M}$$

Avec M = 1

$z_I \neq z_C$ Car la fréquence d'échantillonnage est différente.

Fonction de transfert de l'ensemble :

(8) $$H(z) = \left(\sum_{k=0}^{MR-1} z^{-k}\right)^N = \left(\frac{1-z^{-MR}}{1-z^{-1}}\right)^N$$

Avec N = 3 (nombre d'étage), M= 1, R = 50 (facteur de décimation)

Par définition :

(9) $$z = e^{j\omega T}$$

La formule d'Euler :

(10) $$\sin x = \frac{e^{ix} - e^{-ix}}{2i}$$

D'où la transformation de la fonction de transfert :

(11) $$H(z) = \left(\frac{1 - e^{-jMRT\omega}}{1 - e^{-j\omega T}}\right)^N = \left(\frac{e^{-j\frac{MRT\omega}{2}}}{e^{-j\frac{\omega T}{2}}}\right)^N \cdot \frac{sin^N}{sin^N} \frac{\left(\frac{MRT\omega}{2}\right)}{\left(\frac{T\omega}{2}\right)}$$

En simplifiant cette fonction de transfert avec la formule du sinus cardinal : $\operatorname{sinc} x = \frac{\sin x}{x}$

(12) $$H(z) = (AMR)^N \cdot \frac{sinc^N}{sinc^N} \frac{\left(\frac{MRT\omega}{2}\right)}{\left(\frac{T\omega}{2}\right)}$$

Par définition :

(13) $$\omega = \frac{2\pi f}{f_s}$$

Mise en forme

(14) $$|H(f)|_{dB} = 20N.log_{10}(MR) + 20N.log_{10}\left[\frac{sinc\left(\frac{MR\pi f}{fs}\right)}{sinc\left(\frac{\pi f}{fs}\right)}\right]$$

Avec l'application numérique on obtient :

(7) $$|H(f)|_{dB} = 101.9dB + 60.log_{10}\left[\frac{sinc^3\left(\frac{MR\pi f}{96k}\right)}{sinc^3\left(\frac{\pi f}{4.8M}\right)}\right]$$

Annexe 6 : Fonction de transfert dans le domaine fréquentiel du filtre FIR

Soit la fonction de transfert :

$$H(z) = \sum_{k=0}^{4} a_k z^{-k} \quad (16)$$

Alors

$$H(z) = a_2 . e^{-2Tp} . \left[\frac{a_0}{a_2}. e^{+2Tp} + \frac{a_1}{a_2}. e^{+Tp} + 1 + \frac{a_3}{a_2}. e^{-2Tp} + \frac{a_4}{a_2}. e^{-2Tp}\right] \quad (18)$$

Par définition:

$$z = e^{Tp} = e^{j\omega T} \quad (17)$$

Avec la symétrie des coefficients on a :
$a_0=a_4$ et $a_1=a_3$ alors

$$H(z) = a_2. e^{-2Tp}. \left[1 + \frac{2_{a0}}{a_2}. \cos(2T\omega) + \frac{2_{a_1}}{a_2}. \cos(T\omega)\right] \quad (19)$$

La mise en forme donne

$$H(z) = 20. log_{10}(a_2) + 20. log_{10}\left[1 + \frac{2_{a0}}{a_2}. \cos\left(\frac{4\pi f}{Fs}\right) + \frac{2_{a_1}}{a_2}. \cos\left(\frac{2\pi f}{Fs}\right)\right] \quad (20)$$

Avec application numérique, on obtient :
Soit A=[245 -1952 11620 -1952 245] les coefficients

$$H(z) = 81.3dB + 20. log_{10}\left[1 + 4.21e^{-2} \cos\left(\frac{4\pi f}{Fs}\right) - 3.36e^{-1} . \cos\left(\frac{2\pi f}{Fs}\right)\right] \quad (21)$$

Annexe 7 : Fonction de transfert dans le domaine fréquentiel du filtre FIR passe bas

Nous avons :

(22) $$H(z) = \sum_{k=0}^{42} b_k \, z^{-k}$$

Avec $b_{2i+1}=0$ et $b_{2i}=b_{42-2i}$, quelque soit i.

Alors :

(23) $$H(z) = b_{21}.e^{-21Tp}.\left[\frac{b_0}{b_{21}}.e^{+21Tp} + \frac{b_2}{b_{21}}.e^{+19Tp} + \cdots + 1 + \frac{b_{22}}{b_{21}}.e^{-Tp} + \cdots + \frac{b_{40}}{b_{21}}.e^{-19Tp} + \frac{b_{42}}{b_{21}}.e^{-21Tp}\right]$$

La fonction de transfert est tel que :

(24) $$H(z) = b_{21}.e^{-21Tp}.\left[1 + \frac{2b_{20}}{b_{21}}.\cos(\omega T) + \frac{2b_{18}}{b_{21}}.\cos(3\omega T) + \cdots + \frac{2b_{22}}{b_{21}}\cos(19\omega T) + \frac{2b_0}{b_{21}}.\cos(21\omega T)\right]$$

La mise en forme donne :

(25) $$H(z) = 20.log_{10}(b_{21}) + 20.log_{10}\left[1 + \frac{2b_{20}}{b_{21}}.\cos\left(\frac{2\pi f}{Fs}\right) + \cdots + \frac{2b_0}{b_{21}}.\cos(21\frac{2\pi f}{Fs})\right]$$

Avec l'application numérique :

(26) $$H(z) = 72.26dB + 20.log_{10}\left[1 + 1.26\cos\left(\frac{\pi f}{48k}\right) + \cdots + 3.42e^{-3}\cos(21\frac{\pi f}{48k})\right]$$

Les 45 coefficients =[7 0 -14 0 27 0 -48 0 78 0 -123 0 188 0 -288 0 458 0 -830 0 2594 4096 2594 0 -830 0 458 0 -288 0 188 0 -123 0 78 0 -48 0 27 0 -14 0 7]

Annexe 8 : La validation d'un convertisseur

La validation d'un circuit passe par une série de mesures qui permet de quantifier ses performances dans différentes conditions. L'objectif étant d'obtenir des caractéristiques du circuit qui satisfont aux attentes du client. Les mesures mettent en évidence d'éventuels disfonctionnements, ce qui après analyse de la cause et de la conséquence sur les performances générales, impliqueront ou non la correction sur les circuits suivant. Cette étape permet également de vérifier et de valider le bon fonctionnement des solutions de mesures développées dans ce projet. Cette partie commence par expliquer les notions utiles à connaître afin de mener à bien la campagne de mesure décrite juste après.

Spécificités de la validation d'un ADC

Le rôle de l'ADC est de convertir un signal analogique en un signal numérique. Les différentes mesures du convertisseur vont permettre d'observer le « comportement » du signal en sortie en réponse des variations des grandeurs du signal d'entrée. Il s'agit d'évaluer l'influence de l'énergie du bruit de quantification apparue lors de la numérisation du signal.

Techniques de mesures

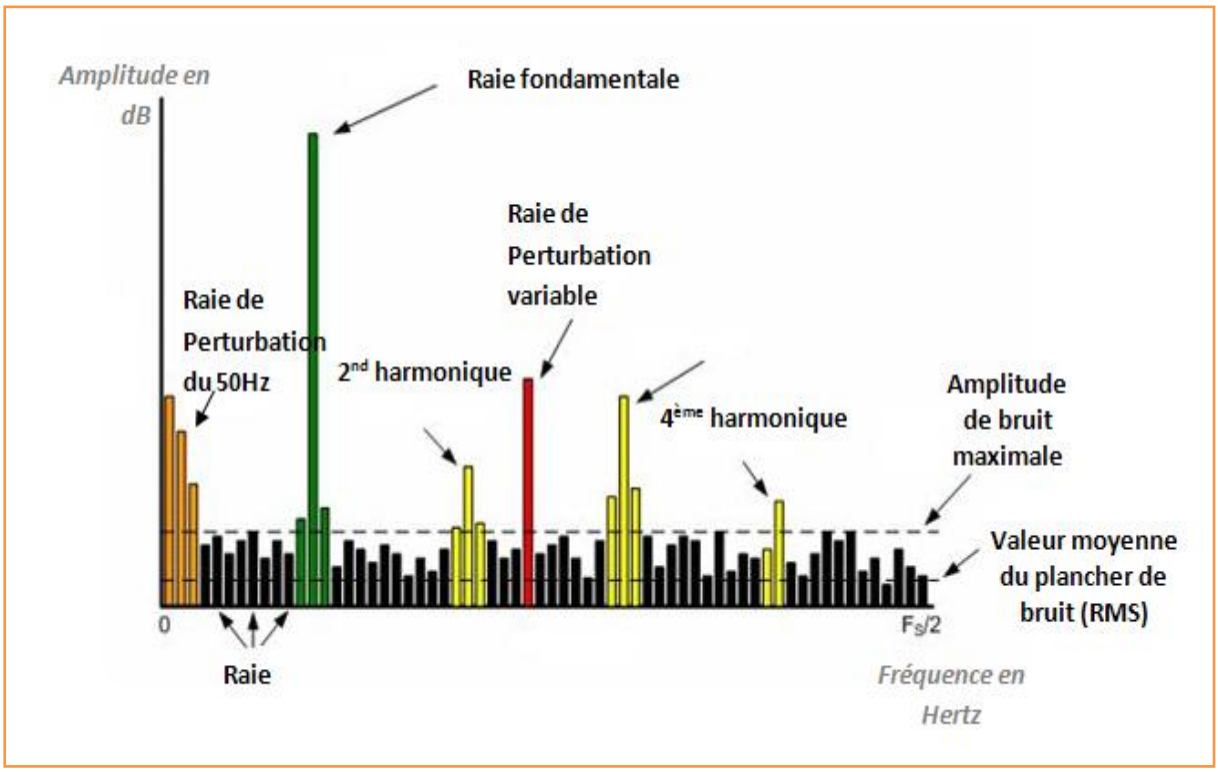
La puissance du bruit est définit pas sa densité spectrale. La mesure de l'amplitude et de la fréquence du spectre permet d'analyser le signal. Pour cela il existe deux techniques principales permettant de mesurer et quantifier les performances d'un circuit audio.

- L'histogramme
- La FFT (Fast Fourier Transform) : Transformée de Fourier Rapide

L'histogramme permet d'effectuer les mesures sur un signal statique dans le domaine temporel. Mais les applications audio et vidéo requièrent des signaux dynamiques (qui évolue dans le temps). Dans ce cas, la mesure de performance de ces signaux sollicite la technique de la FFT avec la représentation spectrale du signal dans le domaine fréquentielle. La FFT est une transformation mathématique[34] d'un signal permettant de passer du domaine temporel au domaine fréquentiel. Les instruments de mesures, tel que l'analyseur de spectre ou l'analyseur de signaux audio proposent cette représentation en effectuant automatiquement le calcul de la FFT. C'est à partir de l'analyse de ce spectre qu'en découle les différentes mesures.

La figure suivante présente les différentes composantes d'un spectre. L'échelle des abscisses indique la bande de fréquence de 0Hz à la fréquence d'échantillonnage divisée par deux, noté Fs/2. L'échelle des ordonnées indique l'amplitude en décibel du signal. La densité spectrale est un ensemble de raies qui contiennent les informations du signal :

- la fondamentale en vert qui représente la puissance du signal d'entrée.
- l'amplitude du plancher de bruit visualisé en noir.
- L'amplitude des harmoniques, en jaune.



Composantes du spectre d'un signal sinusoïdal

Mise en œuvre des mesures

C'est la différence entre ces amplitudes qui définit l'analyse de l'influence du bruit de quantification sur le signal. Ces notions vont permettre la compréhension de la mesure et permettre la mise en œuvre des mesures exposées ci-après.

Les principaux paramètres qui définissent la qualité audio du signal en sortie du circuit sont

- La distorsion du signal représenté par les d'harmoniques.
- Le rapport signal à bruit qui donne une bonne indication sur la performance du circuit.
- La réponse en fréquence qui vérifie le gabarit des filtres.

Ces mesures sont faites en fonction de l'amplitude et de la fréquence du signal d'entrée ainsi que de la largeur de bande d'étude car si la largeur de bande est plus courte, la puissance de bruit est réduite,

ce qui améliore le rapport signal à bruit. Un autre paramètre est à prendre en compte, celui de la sensibilité de l'oreille humaine. Celle-ci est capable de discerner les sons de fréquences comprises dans la bande audio, mais sa sensibilité varie selon la fréquence. Elle est beaucoup moins sensible aux basses fréquences, comprises entre 20Hz et 400Hz, qu'aux fréquences moyennes à aiguës qui correspondent à celles de la parole. Des filtres[35] ont spécialement été développés pour simuler ces différentes sensibilités en pondérant de façon plus ou moins importante chacune des fréquences de la bande audio. Ces filtres correcteurs sont intégrés directement dans les instruments de mesure. Enfin, un dernier paramètre important, la consommation du circuit est également vérifiée, d'autant qu'il s'agit d'une application fonctionnant sous batterie (téléphone portable) et que l'enjeu est de consommer le moins possible.

Le signal de sortie est mesuré grâce à un UPV, un analyseur audio.

Réponse en amplitude

Cette mesure consiste à mesurer l'amplitude (gain) du signal en sortie de l'ADC. Le signal d'entrée est de fréquence fixe égale à 1KHz, les mesures en sortie sont prises à chaque variation des valeurs de gain en entrées, soit de -20dBFS à 3dBFS. Le graphique représente l'amplitude de sortie en fonction de l'amplitude d'entrée. Le but étant d'avoir une erreur constante entre les deux grandeurs, ce qui permet de vérifier la précision du filtre. Il est alors observé qu'à partir d'une certaine amplitude d'entrée, le convertisseur ne suit plus. C'est la représentation du gabarit du filtre

La réponse en fréquence

Comme précédemment, il s'agit de la mesure de l'amplitude de la sortie, mais cette voici en fonction la variation de la fréquence en entrée dans la bande audio. L'amplitude d'entrée reste fixe.
Le graphe obtenu représente l'amplitude en fonction de la fréquence sur la bande audio. Une fois que la réponse en fréquence à été mesurée, ses valeurs peuvent être corrélées aux résultats théoriques.

Signal to noise ratio (SNR)

Le signal d'entrée est en pleine échelle, valeur maximale avant la saturation, cette amplitude reste constante, en revanche c'est sa fréquence qui varie dans la bande audio. Cette mesure ne tient pas compte de l'énergie des harmoniques. L'instrument calcul directement le rapport entre l'amplitude du signal et la densité du bruit. Le résultat est donné en décibel.

$$SNR[dB] = 20.\log\left(\frac{Energie\_du\_signal\_à\_pleine\_échelle}{Total\_des\_énergies\_hors\_harmoniques}\right)$$ Valeur en rms

Dynamic Range (DR)

La mesure de Dynamic Range est la même que la précédente, avec une très faible amplitude du signal d'entrée qui se traduit par une absence d'harmoniques en sortie, celles-ci étant plus basses que le plancher de bruit. Cette mesure est prise à des largeurs de fréquence différentes (8KHz, 16KHz et 20KHz). Le résultat donné en dB renseigne sur la résolution du circuit, grâce à la formule du SNR :

$$DR[dB] = 6{,}02n + 1{,}76$$

Avec n, le nombre de bit. Une fois le Dynamic Range mesuré il est facile de déduire le nombre de bits du circuit.

Idle Channel Noise (ICN)

Il s'agit encore d'une mesure de SNR. Sa particularité réside dans la quasi-absence de puissance du signal d'entrée. (-200dB). Cette mesure permet de mettre en évidence la présence des bruits déplaisants liés à l'environnement externe du circuit, comme la carte de test.

Cette mesure est donnée en µVRMS ou en dB

$$ICN[dB] = 20.\log\left(\frac{Energie\_du\_signal\_à\_pleine\_échelle}{Total\_des\_énergies\_hors\_signal}\right)$$ Valeurs en RMS

Ces trois mesures, SNR, DR et ICN, peuvent être vérifiées en corrélant leurs résultats.

SNR = 20 log ($S_{(\mu Vrms)}$ / $ICN_{(\mu Vrms)}$)

Total Harmonic Distortion (THD)

La distorsion est un phénomène que l'on observe visuellement sous forme d'harmoniques en sortie de l'ADC. Cette mesure consiste à calculer le rapport entre la fondamentale et les harmoniques du spectre en sortie, ce qui revient à mesurer l'énergie total du signal. Pour cela, il faut que l'amplitude d'entrée soit suffisamment importante (0dBFS) pour que les harmoniques apparaissent au-delà du plancher de bruit.

$$THD[dB] = 20.\log\left(\frac{Energie\_du\_signal}{Total\_des\_énergies\_avec\_harmoniques}\right)$$ Valeurs en RMS

Il suffit simplement de soustraire l'amplitude en dB de la fondamentale à sa première harmonique. Il faut aussi s'assurer que les harmoniques sont plus basses que la fondamentale, sinon c'est qu'il s'agit d'un « Random ton », une raie de perturbation, dans ce cas, seule une analyse approfondit du signal peut en déterminer la source. (exemple : Couplage).

Consommation

Le but étant de mesurer la consommation du circuit sur l'alimentation principale. Le signal d'entrée injecté est de -200dB, soit une très faible puissance. Pour connaître la consommation, une première mesure est faite lorsque le circuit est off, ce qui permet d'avoir une référence. Il faut ensuite refaire la mesure du circuit en marche, et soustraire ces deux valeurs, afin de ne pas tenir compte des consommations résiduelles.

Power Supply Rejection Ratio (PSRR)

Le PSRR est en fait le rapport entre le signal qu'on envoie sur l'alimentation et celui qu'on obtient en sortie. C'est à dire qu'on envoie sur l'alimentation du circuit un signal sinusoïdal et on regarde si la sortie maintient l'amplitude souhaitée et élimine les composantes sinusoïdales simulant des perturbations sur une plage de fréquences donnée.

Annexe 9 : Le complément à deux

Soit le nombre 3
0000 0011
Pour obtenir -3
Prendre le nombre 3 en binaire :
0000 0011
Inverser tous les bits :
1111 1100
Ajouter 1 : 1 + 1 = 10 on pose 0 et on retient 1(retenue à ajouter au bit de gauche)
1 + 0 = 1 on pose 1 aucune retenue
0 + 1 = 1 on pose 1 aucune retenue

1111 1100
+ 0000 0001
_____
1111 1101

On remarque bien que le MSB = 1, il s'agit donc d'un nombre négatif

Bibliographie

[1] **ST-Ericsson** : Site de présentation de l'entreprise, 2010
http://www.stericsson.com/

[2] **ST-Ericsson**: Site intranet de communication interne à l'entreprise, 2010
http://our.intranet.stericsson.com/

[3] **Ericsson** : site de présentation de l'entreprise, 2010
http://www.ericsson.com/

[4] **MARGUERY Philippe** : Introduction to the Audio définitions, note d'application interne décrivant les circuits audio, 2003

[5] **Texas Instruments** : Understanding Data Converters, article décrivant les caractéristiques générales des convertisseurs, 1995

[6] **JOHNSTON Jérôme, COFFEY Keith** : Cristal Data Acquisition Product, guide de référence sur les choix de convertisseurs, présenté lors d'un séminaire d'application de Cirrus Logic, 1999

[7] **ALLIER Emmanuel** : Spécification de l'ADC, document confidentiel décrivant les caractéristiques attendues du circuit, 2010

[8] **ALLIER Emmanuel** : Digital Filtering, Document interne décrivant les architectures et les spécificités des filtres à concevoir, 2010

[9] **TISSERAND Arnaud** : Introduction aux circuits FPGA, Présentation Séminaire MIM, 2003
http://www.irisa.fr/prive/Arnaud.Tisserand/docs/semmim-at-fpga.pdf

[10] **ALTERA Corporation** : Site d'un constructeur de FPGA, Ensemble de documents techniques concernant les FPGA, 2010
http://www.altera.com/literature/lit-index.html

[11] **THOMPSON Mike** : FPGAs accelerate time to market for industrials designs, article de l'EETimes concernant les avantages de l'utilisation du FPGA dans l'industries, 2004
http://www.design-reuse.com/articles/exit/?id=8190&url=http://www.eetimes.com/showArticle.jhtml;jsessionid=4OJLA20JQAVNSQSNDBGCKHSCJUMEKJVN?articleID=22102798

Index des figures

Index des Tableaux

Glossaire

Bus I2C : Bus de communication utilisant un protocole série entre 2 circuits mis au point par Philips au début des années 80.

Décimation : C'est le fait de réduire le taux d 'échantillonnage dû au suréchantillonnage sans perdre d'informations.

Densité spectrale de bruit : Elle se définit comme la densité de bruit présente dans un signal, rapporté à une bande passante de 1Hz.

Distorsion : C'est une altération de la forme originale d'un signal.

Echantillonnage : C'est l'opération qui consiste à passer d'un signal analogique à un signal numérique en capturant des valeurs à intervalles de temps réguliers.

Filtre de mise en forme : « Noise shaping » en anglais. Ce filtre permet de repousser le bruit de quantification en dehors de la bande d'intérêt.

Harmonique : Ce sont des raies de fréquence multiple de la fréquence du signal principal qui peuvent apparaître lors de l'analyse fréquentielle suivant le type de signal observé.

Microcontrôleur : C'est un circuit intégré qui rassemble les éléments essentiels d'un système : processeur, mémoire. Le processeur exécute les instructions d'un programme stockées dans la mémoire.

Plateforme mobile : Ensemble composée des circuits intégrés et de solutions développés pour former une application sans fil.

Quantification : C'est le procédé qui permet d'approximer un signal continu (ou à valeurs dans un ensemble discret de grande taille) par des valeurs d'un ensemble discret d'assez petite taille.

Rapport signal à bruit : Ce rapport désigne la qualité de la transmission d'une information par rapport aux parasites.

Semi-conducteur : Type de matériau dont la conductivité électrique se trouve entre celle d'un conducteur et celle d'un isolant. C'est aussi une matière qui a des propriétés de compression particulières à hautes températures.

Suréchantillonnage : C'est le fait d'échantillonner le signal à une fréquence plus forte que la fréquence de Shannon.

Wireless : sans fil.

Programmation et Utilisation du FPGA pour la validation et la vérification de circuits électroniques

Mémoire d'Ingénieur C.N.A.M., Grenoble 2011

RESUME

ST-Ericsson est un acteur majeur de l'industrie du semi-conducteur à destination d'applications mobiles. Il fournit à ses clients des solutions complètes, ainsi que des circuits intégrés individuels conçus part ses équipes de recherche et développement. Pour valider les conceptions des circuits, en plus de la phase de simulation, une étape de vérification sur prototype est indispensable aujourd'hui avant l'industrialisation du produit.

Ce mémoire est consacré à la réalisation d'une solution de validation. Elle offre la possibilité aux concepteurs de vérifier le fonctionnement de la partie numérique de leurs circuits électroniques avant même la fabrication des prototypes. La mise en œuvre de cette solution s'est faite grâce à un circuit programmable qui permet d'émuler le fonctionnement du circuit, le FPGA.

L'objectif du travail était d'étudier le circuit à valider ainsi que le projet dans lequel il doit s'implémenter. Il s'agissait de concevoir des filtres numériques permettant d'exploiter les signaux de sortie d'un convertisseur. La méthode de programmation utilisée pour le FPGA est la description matérielle par saisie graphique. Cette conception doit alors intégrer l'environnement de test du projet composé d'un logiciel de pilotage et d'une solution matérielle.

Mots clés : FPGA, Validation, Vérification, Convertisseur analogique-numérique, Filtres numériques, programmation, description matériel.

SUMMARY

ST-Ericsson is a major actor of the semiconductor industry for mobile applications. It provides to his customers complete solutions as well as individual integrated circuits designed by its research and development teams. To validate the circuit design, in addition to the simulation phase, a verification stage on prototype is essential today before the product industrialization.

This report focuses on the design of validation tool which offers the possibility to designers to verify if the digital part of the electronic circuit is working before the prototype manufacturing. The implementation of this solution was done thanks to the FPGA, a programmable circuit which allows emulating the behavior of the circuit. .

The purpose of this work was to study the circuit to validate as well as the project where it had to be implemented. It was about to design digital filters which allow to use output signals of converter. The programming method used for the FPGA, consist of hardware description by graphical input. This design must include the test environment made with software and hardware tools.

Key words: FPGA, Validation, Check, Analog to digital converter, digital filters, programming, materials description.